OcientAIQ™ Unified Data Platform

User Documentation

Version 27.1

Ocient Proprietary and Confidential

Table of contents

OcientAIQ Unified Data Platform

Get Started

Get Started with the OcientAIQ Unified Data Platform 12

Key Concepts 15

Load and Analyze Data 21

Work with AI Prompts 33

Connect to Ocient

Connect to Ocient 37

Ocient Integrations

Ocient Integrations 38

Looker Connector 40

DBeaver Integration 44

Metabase Integration 49

SQLAlchemy Integration 53

Superset Integration 54

Tableau Integration 56

Connect Using JDBC 59

Connect Using pyocient 62

Load Data

Load Data 66

Load by Data Source

Data Pipeline Load of CSV Data from S3 74

Data Pipeline Load of JSON Data from Kafka 80

Data Pipeline Load of Parquet Data from S3 88

Data Pipeline Load of JSON Data from HDFS 96

Data Types for Data Pipelines 102

© 2026 Ocient, Inc.

Data Formats for Data Pipelines

Data Formats for Data Pipelines 108

JSON Selectors Examples in Data Pipelines 139

Parquet Selectors Examples in Data Pipelines 166

Data Pipeline Behavior Considerations 172

Transform Data in Data Pipelines

Transform Data in Data Pipelines 177

Special Data Pipeline Transformation Functions 217

Data Types for User-Defined Data Pipeline Functions 228

Load Geospatial Data in Data Pipelines 238

Load Data with a User-Defined Data Pipeline Function 243

Load Data from External Sources in Data Pipelines 250

Load Data into Multiple Targets in Data Pipelines 253

Load Metadata and File-Based Partitioned Data in Data Pipelines 261

Deduplication in Data Pipelines 267

Error Tolerance in Data Pipelines 271

Monitor Data Pipelines 274

Manage Errors in Data Pipelines 284

Frequently Asked Questions for Data Pipelines 293

Query Ocient

Query Ocient 303

Quick Start Querying 304

Query Performance Tuning 307

Ocient HTTP Query API

Ocient HTTP Query API 310

execute 316

execute 323

© 2026 Ocient, Inc.

info 328
login 330
logout 333
sso_authentication 335
callback 337
sso_token 339
sso_device_grant 341
sso_device_grant_verify 343
token_refresh 346
Understanding Data Types
Understanding Data Types 349
Array, Tuple, and Matrix Overview 354
Functions Overview 361
Analysis in Ocient
Text Analysis in Database Queries 364
Machine Learning in Ocient
Machine Learning in Ocient 366
Regression Analysis 369
Classification Analysis 397
Clustering Analysis and Dimensionality Reduction 429
Transactions 447
Database Administration
Database Administration 449
Ocient Security Guide
Ocient Security Guide 452
Set Up Data Encryption 459
Manage Users, Groups, and Roles

Workload Management Walkthrough 465
Workload Management and Service Classes 475
Result Set Caching 479
Object-Type Level Privileges Management 485
Manage Users, Groups, and Roles 490
Authentication Methods
Authentication Methods 496
Database Password Security Settings 515
Schema Design
TimeKeys and Clustering Keys 519
Secondary Indexes 528
Table Compression Options
Table Compression Options 542
Global Dictionary Compression 548
Table Retention Policies 553
CREATE TABLE SQL Statement Examples 557
Manage Distributed Tasks 566
Remove Records from an Ocient System 571
System Administration
System Administration 574
Install an Ocient System
Install an Ocient System 577
Ocient Application Installation 579
Ocient System Bootstrapping 581
Ocient Application Configuration 587
Installation Reference
Installation Reference 593

© 2026 Ocient, Inc.

Ocient System Sizing 594
Ocient System Requirements 596
Ocient System Hardware Specifications 601
Operating System Configuration 606
Checking System Configuration Before Bootstrapping 615
Checking System After Configuration 619
Node Bootstrapping Reference 624
Amazon Web Services Ocient Installation 630
Google Cloud Platform Ocient Installation 638
Configure Storage Spaces 648
Ocient Software Upgrade 654
Maintenance Overview
Maintenance Overview 656
Guide to Rebuilding Segments 657
Startup and Shutdown Procedure 662
Power Safety Recommendations 666
NVMe Drive Maintenance
NVMe Drive Firmware Upgrade Process 669
Replace an NVMe Node Drive 680
Expand and Rebalance System 691
Replace Nodes 709
Backup and Restore 722
Monitoring
Log Monitoring 723
Statistics Monitoring 734
System Information REST Endpoints
System Information REST Endpoints 740

Get Statistics 741
Get Version 744
Get Status 746
Get Configuration 747
Get Configuration of Whole System 752
Get Configuration Parameters 757
Set Up System Monitoring with the TIG Stack and Kapacitor 758
Monitor Segment Group Transfers 777
System Configuration
Inspect the Current Configuration 779
Node Configuration with the Administrator Role 784
Manage the Network Configuration of an Ocient System 787
Secure Connections Using TLS 792
Multiple Storage Clusters for Loading Data 794
Loading Administration
Alter Default Data Pipeline Behavior 796
Configuration Settings for Data Pipelines 799
Configure Data Pipeline Logging 805
Air-Gapped Environment with an Ocient System 807
Ocient Simulator 815
Ocient Architecture
Ocient Architecture 832
Core Elements of an Ocient System 833
Compute-Adjacent Input and Output on Large Working Sets 840
Query Performance Optimizations 849
Plan Optimization 858
Exabyte Scalability Design 861

Loading Characteristics and Concepts 866

SQL Reference

SQL Reference 871

Alphabetical SQL Functions List 875

Categorized SQL Functions List 903

Aggregate Functions 931

Array Functions and Operators 963

Character and Binary Functions 993

Data Types 1032

Date and Time Functions 1042

Formatting Functions 1075

HyperLogLog Functions 1082

Geospatial Functions

Geospatial Functions 1093

Attribute Functions 1102

Conversion Functions 1114

Linestring Constructors 1123

Linestring Functions 1131

Point Constructors 1140

Polygon Constructors 1148

Spatial Measurement 1156

Spatial Operators 1169

Spatial Relationships 1199

Spatiotemporal Measurement 1219

Spatiotemporal Operators 1224

Machine Learning Model Functions

Machine Learning Model Functions 1232

© 2026 Ocient, Inc.

Data Preparation 1236
Regression Models 1247
Classification Models 1277
Clustering and Dimension Reduction Models 1304
Ensemble Models 1316
Other Models 1330
Machine Learning Model Options 1338
Math Functions and Operators 1381
Matrix Functions and Operators 1415
Network Type Functions 1423
Scalar Data Conversion Functions 1428
Time Zone Functions 1440
Tuple Functions and Operators 1453
Window Aggregate Functions 1456
Other Functions and Expressions 1491
Data Definition Language (DDL) Statement Reference
Data Definition Language (DDL) Statement Reference 1508
Cluster and Node Management 1515
Databases 1567
Schemas 1574
Tables 1579
Views 1611
Indexes 1616
Data Pipelines
Data Pipelines 1621
Transactional Data Pipelines 1713
Distributed Tasks 1718

Machine Learning Models 1728
Users, Groups, and Service Classes 1735
Data Integrity and Storage 1754
Query Analysis 1760
Query Management 1764
Statistics Cache Management 1768
Data Control Language (DCL) Statement Reference 1772
Data Manipulation Language (DML) Statement Reference 1791
Data Query Language (DQL) Statement Reference
Data Query Language (DQL) Statement Reference 1803
Join Operations 1856
SQL Syntax Conventions 1862
Identifiers 1865
Generate Tables Using sys.dummy 1867
Transactional Control Language (TCL) Statement Reference 1869
Errors and Warnings
Errors and Warnings 1874
Data Pipeline Loading Errors 1886
Reserved Words 1893
System Catalog Reference
System Catalog Reference 1897
System Catalog 1898
Information Schema 2027
Discover Insights From System Catalog Tables 2041
Connection Driver Reference
Connection Driver Reference 2075
JDBC Manual

JDBC Manual 2076
Data Extract Tool 2096
Commands Supported by the Ocient JDBC CLI Program 2107
JDBC Classes and Methods 2129
JDBC Spark Connector 2131
OCGraph Java Library 2143
Ocient Python Module (pyocient)
Ocient Python Module: pyocient 2195
OCGraph Python Library 2216
LAT Reference
Ingest Data with Legacy LAT Reference
Ingest Data with Legacy LAT Reference 2269
Loading by Data Source
LAT Load JSON Data from Kafka 2274
LAT Load CSV Data from S3 2293
LAT Load JSON Data from S3 2311
LAT Advanced Loading and Transformations 2328
LAT Overview 2338
LAT Data Types in Loading 2340
LAT Pipeline Configuration
LAT Pipeline Configuration 2353
LAT Source Configuration 2360
LAT Sink Configuration 2388
LAT Extract Configuration 2395
LAT Transform Configuration 2415
LAT Data Transformation with JMESPath
LAT Data Transformation with JMESPath 2423

LAT Transformation Functions 2427
LAT User-Defined Transformations 2452
LAT Record Filtering 2455
LAT Service Configuration 2458
LAT Metrics 2465
LAT Log4J Configuration 2472
LAT Packaging and Installation 2476
LAT Advanced Topics 2485
LAT Client Command Line Interface
LAT Client Command Line Interface 2491
Install LAT Client in a Disconnected Network 2502
Version Compatibility 2505
Release Notes
Release Notes 2510
OcientAIQ Unified Data Platform Release Notes 2511
JDBC Release Notes 2540
pyocient Release Notes 2580
Glossary 2587
Open Source Acknowledgments 2589
Download 2731

Get Started with the OcientAIQ Unified Data Platform

The OcientAIQ™ Unified Data Platform is a petabyte-scale data platform that brings AI, analytics, and high-performance query execution directly to enterprise data. Rather than requiring you to move, copy, or sample large data sets across fragmented systems, the platform unifies data ingest, query optimization, security, and governance in a single platform. Agents, applications, and analysts connect to the platform to work with governed enterprise data in place.

The platform supports both structured and semi-structured data, giving AI and analytical workloads access to richer context without additional data pipelines. The platform continuously ingests high-volume data so that it is available for queries and AI-driven workflows as it arrives. Built-in access controls, auditability, and compliance features protect data across every interaction — whether the consumer is a human analyst, a BI tool, or an AI agent.

You can deploy the platform in the environment that fits your operational, security, and data-sovereignty requirements. The following sections introduce the core concepts that define how data is stored, organized, and executed within the OcientAIQ Unified Data Platform.



Key Concepts

Core Ocient® concepts and terms needed for a quick start



Install an Ocient System

Installation, Upgrade, System Configuration, Maintenance, and Monitoring of an Ocient System



Connect to Ocient

Connection to Ocient with JDBC and other drivers

© 2026 Ocient, Inc.



Database Administration

Overview of key database administration capabilities like workload management, result set caching, and managing users, groups, and roles



Load Data

Loading and transformation capabilities, managing pipelines, transformation functions, and error handling



Query Ocient

Overview of Ocient data types and the components of a query



Core Elements of an Ocient System

Overview of Ocient System design and architecture

These sections outline the capabilities of Ocient for data ingestion, transformation, and advanced analytics. Ocient SQL supports integrated libraries for geospatial analysis and machine learning tools.



SQL Reference

SQL command, function, operator,



Connection Driver Reference



Machine Learning in Ocient

© 2026 Ocient, Inc.

and keyword reference including DDL, DCL, and query reference

Different SQL client connectors

Walkthroughs to deploy various ML models integrated into Ocient



Geospatial Functions

Functions for analyzing geospatial data sets

For AI tools, start with **/llms.txt** for optimal content extraction.

For information about integrations and connections to third-party applications and tools, see **Ocient Integrations**.

For a detailed explanation of the Compute Adjacent Storage Architecture® in Ocient, design principles, storage capabilities, and query engine, see **Ocient Architecture**.

For information on ingesting data using the legacy Loading and Transformation (LAT) functionality, see **Ingest Data with Legacy LAT Reference**.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Key Concepts

The OcientAIQ™ Unified Data Platform is a real-time OLAP datastore designed for running analytics against large time series data sets. The Ocient® System operates in compliance with the ANSI standard for SQL.

The Ocient System includes:

- **Ultra-fast query speeds** for analytics on hyperscale data sets.
- **Scalability** for cost and performance on a system built with commodity hardware.
- **Flexible deployment** for running on premises, in the cloud, or as an Ocient-managed service.
- **Robust functionality** for DDL, geospatial (OcientGeo™), and machine learning (OcientML™).
- **Data movement** at scale for ETL and ELT ingestion and operations.

The Ocient System

The Ocient System is based on Compute Adjacent Storage Architecture® (CASA), which collocates NVMe drive storage with the system compute resources to optimize performance. This design keeps records near and accessible for computation without a separate storage layer, which avoids many common bottlenecks for database engines, such as limits on network capacity or processing throughput. The design achieves superior query performance when it operates on trillions of rows of data.

To learn more about the design principles behind the Ocient System, see **Ocient Architecture**.

Deployment

Ocient is designed for flexible deployment for unique use scenarios that include:

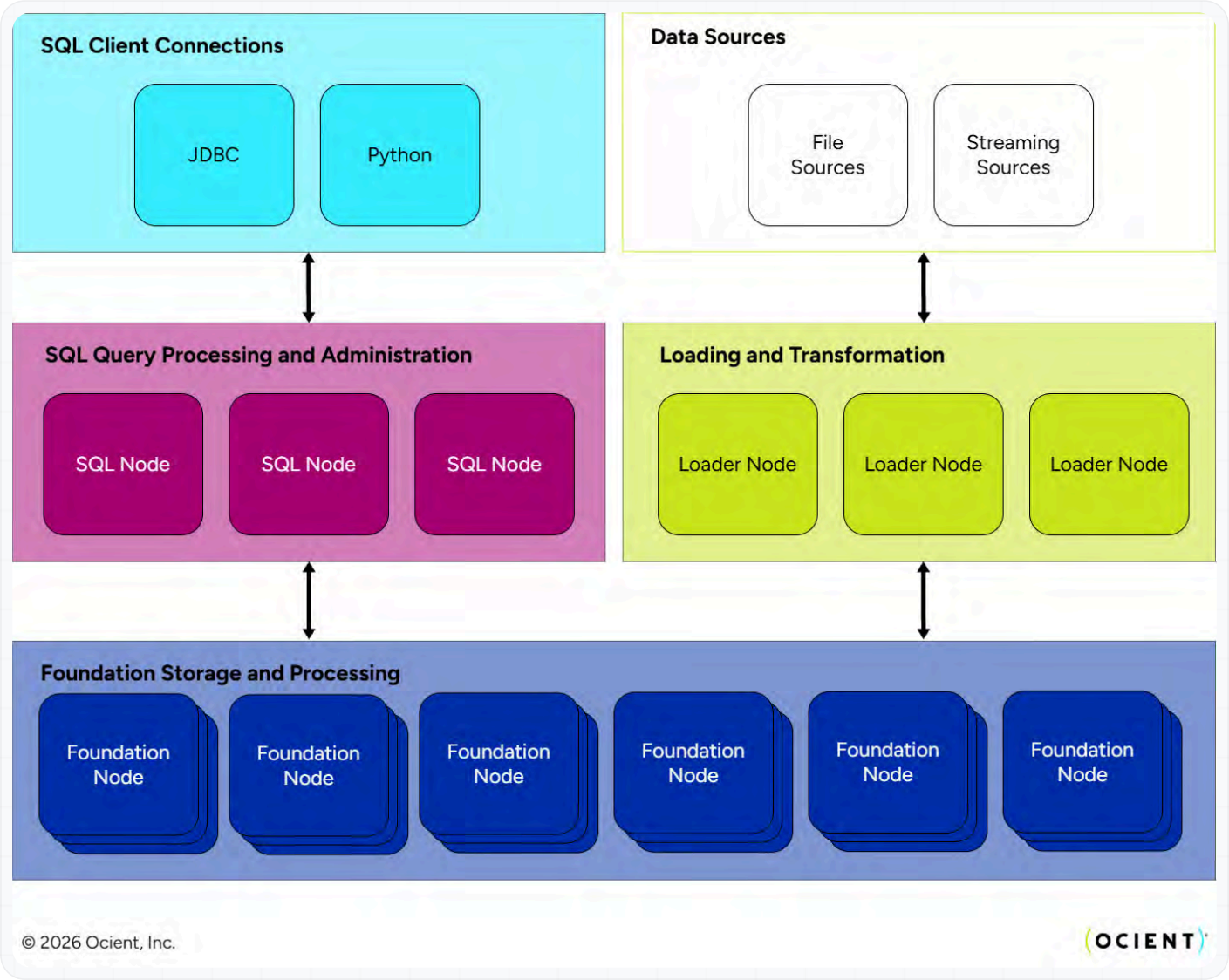
- OcientCloud® — A cloud deployment managed by Ocient Support. This option uses performance-tested hardware and network infrastructure.
- Public Cloud — Installation on third-party cloud providers, including Amazon® Web Services℠ (AWS℠) and Google® Cloud™.
- On premises — A self-managed option for hosting Ocient in your own data center.

System Nodes

© 2026 Ocient, Inc.

Ocient is a distributed system consisting of interconnected nodes within clusters, which together form the data warehouse. The Ocient System designates nodes for these system roles:

- **SQL Nodes** — Parse incoming SQL statements and administer commands throughout the system. These nodes serve as the interface for system and database administration.
- **Loader Nodes** — Manage ETL ingestion from batch or streaming sources and index data for optimized performance.
- **Foundation Nodes** — Store data and perform the bulk of query processing. Foundation Nodes perform as much query processing as possible on their stored data before interacting with other nodes.



Understanding nodes and their roles in an Ocient System is most useful for system setup, administration, and maintenance. For details about nodes, see **Ocient Architecture**. However, the Ocient System makes it easy to load and query data without knowing these details.

Data Types

The OcientAIQ Unified Data Platform, the functions, and the indexes use only native SQL data types to optimize storage and performance. Supported data types include common SQL scalar types and complex types such as IP, arrays, tuples, and geospatial. For details about supported data types, see **Understanding Data Types**.

Loading

The Ocient System can load data for an end-to-end flow from file or streaming sources, including common sources such as:

- Apache® Kafka®
- AWS S3
- Apache® Hadoop® Distributed File System (HDFS)

For all the source and format options, Ocient uses pipelines to control ingestion. You can control pipelines using DDL commands, API on Loader Nodes, or a command-line interface. You can transform data during loading using SQL functions.

Loading ingestion throughput can scale as needed based on the Ocient architecture.

For details, see **Load Data**.

Query Processing

The design of the Ocient architecture minimizes the amount of on-disk data that must be read and processed to execute a query. To do this, the system compiles a custom I/O pipeline for each data segment relevant to a query. These custom pipelines leverage any keys or indexes to improve throughput.

To learn more about query processing, see **Query Performance Optimizations**.

Ocient Indexing

An Ocient datastore can use multiple layers of indexing to facilitate query performance. Using these indexes is pivotal to optimizing query performance for large data sets.

You can apply indexes using basic DDL statements that do not require deep knowledge of the internal system or data set. You also have some ability to customize indexes for their specific use cases.

Indexes are divided into two main categories.

1 **Segment Keys**

These indexes require no additional storage and usually should be deployed on every table. Segment keys operate by accessing data from the partitioned data segments to quickly filter rows without I/O processing.

Segment keys include:

- **Time Key** — A segment key that partitions data based on a time-series data column.
- **Clustering Key** — A series of columns that are frequently queried together. The system subdivides these segments on disk for faster reference.

For more information on segment keys, see **TimeKeys and Clustering Keys**.

2 **Secondary Indexes**

When deployed precisely, Ocient secondary indexes can dramatically reduce the time for queries to run on large data sets.

Ocient supports secondary indexes for these data categories:

- **Numeric**
- **String**
- **Partial string**
- **Geospatial**

For more information, see **Secondary Indexes**.

Transactions

The Ocient System supports transactions that group one or more SQL statements into a single unit of work. The database either commits all changes or rolls them back as a whole. Use transactions to ensure data consistency across loading and querying operations when a set of related changes must succeed or fail together.

For details, see **Transactions**.

Connecting to Ocient

Ocient provides SQL access to its database using industry-standard interfaces:

- JDBC
- pyocient, a Python®-based driver for Ocient.
- HTTP Query API, a REST-based interface for executing SQL statements.
- Apache® Spark™ Connector, a DataSourceV2 implementation for reading from and writing to Ocient tables using Apache Spark.

End users can query and analyze data in Ocient without understanding the organization, nodes, networking, or other parts of the system architecture.

For more information, see **Connect to Ocient**. For details about the connection drivers, see **Connection Driver Reference**.

Integrations

Ocient supports integration with various third-party tools for database administration, business intelligence, and system monitoring.

For details about supported third-party tools, see **Ocient Integrations**.

Machine Learning

The Ocient System includes OcientML functionality for training and executing machine learning models directly within SQL. OcientML builds on native linear algebra support, including first-class matrix data types and operations such as matrix arithmetic, inversion, and eigenvalue decomposition.

You can create models using the `CREATE MLMODEL` statement and execute them as scalar functions in queries. Ocient supports a broad suite of models spanning regression, classification, clustering, dimensionality reduction, ensemble methods, and neural networks to address a wide range of analytical use cases.

For details, see **Machine Learning in Ocient**.

Resilience to Hardware Failure

The Ocient System uses erasure coding to organize and compute parity blocks so the system is fault-tolerant and can rebuild missing data. This failsafe does not require redundant copies of data, meaning that storage requirements are minimal.

Security

As a unified platform, Ocient helps keep data secure by consolidating security capabilities and reactions in one place with a suite of auditing and monitoring tools. Ocient supports these security standards:

- **Data Encryption:** Optional through TLS/SSL protocols.
- **Security Compliance:** Audited and certified SOC 2 Type 2.
- **Monitoring**: Log-level monitoring and alerts on key system information.
- **Access Controls**: SSO, role-based, and system-level access controls are available.

Related Links

Core Elements of an Ocient System

Ocient Architecture

Ocient Simulator

Connect to Ocient

Related Videos

At the Whiteboard with Ocient: Compute Adjacent Storage Architecture™

At the Whiteboard with Ocient: SQL at Scale

At the Whiteboard with Ocient: Indexing at Hyperscale

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Load and Analyze Data

This tutorial details the core Ocient® workflow from start to finish. You connect to the system, create a database and table, load sample data from a source, validate the loaded data, and execute analytic queries.

The example uses a sample data set that covers every supported Ocient data type. By the end of this process, you have a working table with loaded data that you can query immediately.

Prerequisites

The tutorial requires:

- Network access to the SQL Nodes in your Ocient System and the IP address or hostname of at least one SQL Node.
- A valid username and password with permissions to create databases, tables, and data pipelines.
- You have installed the Ocient JDBC driver and CLI on your local machine. For setup instructions, see **Connect Using JDBC**.

> If you do not have the JDBC driver installed, you can also connect using `pyocient`, the Ocient Python® driver. For setup instructions, see **Connect Using pyocient**.

Step 1: Connect to the Ocient System

Start the JDBC CLI and connect to a SQL Node. Replace the hostname `<HOSTNAME_OR_IP>` and port `<PORT>` placeholders with the values for your system. The default SQL Node port is `4050`.

SQL

```
CONNECT TO jdbc:ocient://<HOSTNAME_OR_IP>:<PORT>/system;
```

© 2026 Ocient, Inc.

After a successful connection, you see the Ocient CLI prompt.

```shell
Ocient> _
```

For details on connection methods and troubleshooting, see **Connect to Ocient**.

Step 2: Create a Database and Schema

Create a database to hold your sample data. Using a dedicated database keeps your work isolated from other applications on the same system.

```sql
CREATE DATABASE test;
```

Switch your connection to the new database.

```sql
CONNECT TO jdbc:ocient://<HOSTNAME_OR_IP>:<PORT>/test;
```

Create a schema within the database to organize your tables.

```sql
CREATE SCHEMA loading;
```

For details on managing databases and schemas, see **Databases** and **Schemas**.

Step 3: Create a Table

Create a table that covers all supported Ocient data types. This table serves as the target for the data pipeline in the next step.

The `CREATE TABLE` SQL statement defines the column names, data types, and a TimeKey® on the `col_timestamp` column in the `data_type_coverage` table. The TimeKey partitions data by time so that the system can skip irrelevant segments during time-filtered queries.

SQL

```
CREATE TABLE loading.data_type_coverage (
    col_bigint     BIGINT,
    col_binary     BINARY(2),
    col_boolean    BOOLEAN,
    col_char       CHAR(64),
    col_date       DATE,
    col_decimal    DECIMAL(18, 4),
    col_double     DOUBLE,
    col_float      FLOAT,
    col_int        INT,
    col_int_array  INT[],
    col_ipv4       IPV4,
    col_ip         IP,
    col_smallint   SMALLINT,
    col_point      POINT,
    col_linestring LINESTRING,
    col_polygon    POLYGON,
    col_time       TIME,
    col_timestamp  TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
    col_tinyint    TINYINT,
    col_tuple      TUPLE<<INT, VARCHAR(255)>> NULL,
    col_uuid       UUID,
    col_varbinary  VARBINARY,
    col_varchar    VARCHAR
);
```

This table intentionally keeps the schema simple to focus on the loading workflow. In production, you should also define a Clustering Key and secondary indexes based on your expected query patterns. For details on table design options, see:

- **TimeKeys and Clustering Keys** — Segment keys that partition and order data for faster queries.
- **Secondary Indexes** — Additional indexes for columns used in filters.

- **Table Compression Options** — Compression settings that reduce storage requirements.
- **CREATE TABLE SQL Statement Examples** — Examples of table definitions for different use cases.

Step 4: Create a Data Pipeline

Data pipelines are the way to load data into an Ocient System. Each pipeline is a SQL object that defines the source, data format, and transformations for loading rows into one or more tables. For a full overview of pipeline concepts, see **Load Data**.

Use a Sample Data Set

The sample data is a set of gzip-compressed CSV files hosted in a public Amazon® Web Services™ (AWS™) S3 bucket. Each file contains one header row followed by data rows with columns covering every Ocient data type.

Create the Pipeline

This data pipeline reads CSV files from the S3 source, parses each field using named headers, applies type conversions where needed, and loads the results into the `data_type_coverage` table.

© 2026 Ocient, Inc.

SQL

```
CREATE BATCH PIPELINE data_type_coverage_pipeline
    SOURCE S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        PREFIX 'all_data_types/small/'
        FILTER_GLOB '**output*.csv.gz'
        COMPRESSION_METHOD 'gzip'
        SORT_BY 'filename'
    EXTRACT
        FORMAT delimited
        RECORD_DELIMITER e'\n'
        FIELD_DELIMITER ','
        FIELD_OPTIONALLY_ENCLOSED_BY '"'
        NUM_HEADER_LINES 1
        HEADERS [
            'bigint', 'binary', 'boolean', 'char',
            'date', 'decimal', 'double', 'float',
            'int', 'int_array', 'ipv4', 'ip',
            'smallint', 'point', 'linestring', 'polygon',
            'time', 'timestamp', 'tinyint', 'tuple',
            'uuid', 'varbinary', 'varchar'
        ]
        OPEN_ARRAY '['
        CLOSE_ARRAY ']'
        ARRAY_ELEMENT_DELIMITER ','
        OPEN_OBJECT '('
        CLOSE_OBJECT ')'
INTO test.loading.data_type_coverage
SELECT
    $bigint                                          AS col_bigint,
    $binary                                          AS col_binary,
    $boolean                                         AS col_boolean,
    $char                                            AS col_char,
    $date                                            AS col_date,
    $decimal                                         AS col_decimal,
    DOUBLE(IF(LOWER($double) = 'nan',NULL, $double)) AS col_double,
    FLOAT(IF(LOWER($float) = 'nan', NULL, $float))   AS col_float,
    $int                                             AS col_int,
    INT[]($int_array)                                AS col_int_array,
    $ipv4                                            AS col_ipv4,
    $ip                                              AS col_ip,
```

```
    $smallint                                           AS col_smallint,
    $point                                              AS col_point,
    $linestring                                         AS col_linestring,
    $polygon                                            AS col_polygon,
    $time                                               AS col_time,
    REPLACE(REPLACE($timestamp, 'T', ' '), 'Z', '')     AS col_timestamp,
    $tinyint                                            AS col_tinyint,
    $tuple                                              AS col_tuple,
    $uuid                                               AS col_uuid,
    VARBINARY(IF($varbinary IS NULL, NULL,
        IF(LENGTH($varbinary) % 2 = 1, CONCAT('0', $varbinary), $varbinary)))
                                                        AS col_varbinary,
    $varchar                                            AS col_varchar;
```

The pipeline has three sections:

- `SOURCE` — Identifies the S3 bucket, file path prefix, and glob filter. The `COMPRESSION_METHOD` parameter indicates that the source files are gzip-compressed.
- `EXTRACT` — Defines the CSV format options, including delimiters, quoting, and the header names that the `SELECT` clause references. The `OPEN_ARRAY` and `CLOSE_ARRAY`, and `OPEN_OBJECT` and `CLOSE_OBJECT` parameters tell the parser how array and tuple values are encoded in the CSV data.
- `SELECT` — Maps each source field to a target column. Some fields require explicit type conversion. For example, `DOUBLE($double)` converts the extracted string to a `DOUBLE` value, and the nested `REPLACE` functions strip the `T` and `Z` characters from timestamps to match the Ocient `TIMESTAMP` format. This `VARBINARY` expression pads odd-length hex strings with a leading zero to ensure all rows that are not `NULL` convert to the `VARBINARY` data type.

For details on pipeline syntax and options, see **Data Pipelines**. For details on supported data formats, see **Data Formats for Data Pipelines**. For details on data type handling during loading, see **Data Types for Data Pipelines**. For the functions, see **Transform Data in Data Pipelines**.

Step 5: Start and Monitor the Pipeline

Start the data pipeline to begin loading data using the `START PIPELINE` SQL statement.

SQL

```
START PIPELINE data_type_coverage_pipeline;
```

Check the pipeline status while it runs.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

Text

```
database_name  pipeline_name                 table_names                       s
-------------- ------------------------------ --------------------------------- -
test           data_type_coverage_pipeline    ["loading.data_type_coverage"]    R
```

Wait for the `status` column to show the `COMPLETED` status. You can execute the `SHOW PIPELINE_STATUS` statement again to check progress.

For larger data sets, the `percent_complete`, `records_processed`, and `records_loaded` columns update as the pipeline progresses.

For more detailed monitoring options, including system catalog tables and metrics endpoints, see **Monitor Data Pipelines**.

If the pipeline enters the `FAILED` status, see **Manage Errors in Data Pipelines** and **Data Pipeline Loading Errors**.

Step 6: Validate the Loaded Data

After the pipeline completes, verify that the data loaded correctly before you begin analysis.

Verify the Row Count

Confirm that the number of loaded rows matches the pipeline metrics.

```sql
SELECT COUNT(*) AS total_rows
FROM loading.data_type_coverage;
```

Cross-reference this count with the `records_loaded` value from the `SHOW PIPELINE_STATUS` statement.

Check the Time Range

Verify that the `col_timestamp` column covers the expected date range. Gaps in the range can indicate missing source files.

```sql
SELECT
    MIN(col_timestamp) AS earliest,
    MAX(col_timestamp) AS latest,
    COUNT(DISTINCT CAST(col_timestamp AS DATE)) AS distinct_days
FROM loading.data_type_coverage;
```

Inspect for NULL Values

Check for unexpected NULL values in columns that should be populated.

© 2026 Ocient, Inc.

SQL

```sql
SELECT
    COUNT(*)             AS total_rows,
    COUNT(col_bigint)    AS non_null_bigint,
    COUNT(col_double)    AS non_null_double,
    COUNT(col_timestamp) AS non_null_timestamp,
    COUNT(col_varchar)   AS non_null_varchar,
    COUNT(col_int_array) AS non_null_int_array,
    COUNT(col_tuple)     AS non_null_tuple
FROM loading.data_type_coverage;
```

If a column has significantly fewer non-NULL values than expected, review the source data and the pipeline `SELECT` transformations for parsing issues.

Sample Records

Review a small set of records to confirm that type conversions applied correctly using the `LIMIT` keyword.

SQL

```sql
SELECT *
FROM loading.data_type_coverage
LIMIT 5;
```

Step 7: Query the Data

With validated data in the table, you can begin running analytic queries. These examples demonstrate common query patterns across different data types.

Aggregate Numeric Data

Compute summary statistics on the numeric columns. In this case, retrieve the count of rows, the minimum number of the `col_int` column, the maximum number of the `col_int` column, and the average of the numbers in the `col_smallint` column.

SQL

```sql
SELECT
    COUNT(*)           AS total_rows,
    MIN(col_int)       AS min_int,
    MAX(col_int)       AS max_int,
    AVG(col_smallint)  AS avg_smallint
FROM loading.data_type_coverage;
```

Filter by Time Range

Use the `col_timestamp` TimeKey column to filter data efficiently. The Ocient System skips segments outside the specified time range of January 2024.

SQL

```sql
SELECT
    CAST(col_timestamp AS DATE) AS day,
    COUNT(*)                    AS row_count,
    COUNT(DISTINCT col_int)     AS distinct_ints
FROM loading.data_type_coverage
WHERE col_timestamp >= TIMESTAMP '2024-01-01 00:00:00'
  AND col_timestamp <  TIMESTAMP '2024-02-01 00:00:00'
GROUP BY CAST(col_timestamp AS DATE)
ORDER BY day;
```

Query Geospatial Data

Inspect the geospatial columns loaded into the table for the first 10 rows using the `LIMIT` keyword.

© 2026 Ocient, Inc.

SQL

```sql
SELECT
    col_point,
    col_linestring,
    col_polygon
FROM loading.data_type_coverage
WHERE col_point IS NOT NULL
LIMIT 10;
```

For details on geospatial functions and spatial queries, see **Geospatial Functions**.

Explore Data Type Distributions

Summarize the distinct values and NULL rates for each column type.

SQL

```sql
SELECT
    COUNT(DISTINCT col_boolean)  AS distinct_boolean,
    COUNT(DISTINCT col_tinyint)  AS distinct_tinyint,
    COUNT(DISTINCT col_smallint) AS distinct_smallint,
    COUNT(DISTINCT col_int)      AS distinct_int,
    COUNT(DISTINCT col_uuid)     AS distinct_uuid
FROM loading.data_type_coverage;
```

For a full reference of SQL functions, see **Functions Overview**. For tips on writing queries that leverage segment keys and indexes, see **Query Performance Tuning**.

Remove the Data Pipeline and Table

After you finish exploring the sample data, remove the `data_type_coverage_pipeline` pipeline.

SQL

```sql
DROP PIPELINE data_type_coverage_pipeline;
```

© 2026 Ocient, Inc.

Remove the `data_type_coverage` table.

```sql
DROP TABLE loading.data_type_coverage;
```

Removing a pipeline removes the pipeline metadata from the system but does not affect the data already loaded into the target table. Whereas removing the table removes the data.

Related Links

Data Pipelines

Tables

Understanding Data Types

SQL Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Work with AI Prompts

Use this guide to draft high-quality prompts to assist AI tools in using the Ocient® documentation and SQL dialect with greater accuracy.

The guide explains why drafting AI prompts for the Ocient documentation helps produce better results. These reasons include:

- Large models often default to generic SQL or other vendor syntax.
- Additional guidance helps scrapers navigate documentation libraries.
- The `llms.txt` index page provides the cleanest and AI-friendly entry point to the documentation.

Key AI Prompt Recommendations

Follow these key recommendations for working with the Ocient documentation:

1 Include Ocient Documentation URLs

Start prompts with one or more links to anchor the model on authoritative content. In prompts, specify the AI index page **/llms.txt** to provide full documentation reference for models.

If relevant, include the exact topic URL (e.g., SQL reference, machine learning, data pipelines).

2 Explicitly Require Ocient SQL

In your prompt, specifically dictate to the AI model to use Ocient SQL, such as `Use Ocient SQL syntax` or `Do not use non-Ocient SQL variants`. Such text is particularly helpful to avoid generic responses for prompts that involve complex data types or Ocient-specific syntax for TimeKey®, secondary indexes, or DDL statements.

© 2026 Ocient, Inc.

AI Prompt Templates

Use these templates and fill in your question and context-setting information in the `< ... >` sections, while keeping the rest of the text intact.

General Questions

This template is suitable for general questions for Ocient System usage. Use this context-setting text prior to asking your question.

Text

```
You are answering using the official Ocient documentation.
Start from https://docs.ocient.com/llms.txt and reference the linked pages there.
Use only Ocient SQL syntax in examples and explanations.
Do not include SQL features or syntax from other databases.

Question: <your question here>
```

Create an Ocient SQL Query

This AI prompt helps ensure an AI tool adheres to Ocient SQL in the response. The performance of the generated SQL query depends on the specifics of your data. Edit the resulting SQL query to meet your needs.

© 2026 Ocient, Inc.

Text

```
You are answering using the official Ocient documentation.
Start at https://docs.ocient.com/llms.txt and reference the linked pages there.
Use Ocient SQL syntax only; do not use Postgres/MySQL/Spark SQL or other variants.
If uncertain about any detail, cite the exact Ocient page used.

Task:
Draft an Ocient SQL query to achieve
<Describe the goal clearly, including relevant tables or columns and any filters or

Context (optional but helpful):
- Schema snippets, table snippets, or column types
- Performance or indexing considerations
- Any assumptions or edge cases

Output format:
1) A single Ocient SQL query
2) A brief explanation (2–3 sentences)
3) Citations to the specific Ocient documentation pages used

Reference pages (if applicable to the task):
<Point to any relevant reference pages.>
```

AI Prompts for Specific Use Cases

For more accurate responses, use your prompt to point to the relevant page URL in the documentation. Each page in this table provides high-level usage information for the specified topic.

Topic	URL
Loading	**/load-data**
Installation	**/install-an-ocient-system**
Machine Learning	**/machine-learning-model-functions**
Geospatial	**/geospatial-functions**
Data Types	**/data-types**
DDL Statements	**/data-definition-language-ddl-statement-reference**

Topic	URL
Errors and Warnings	**/errors-and-warnings**

Troubleshooting Low-Quality Responses

If the responses from your AI prompts are inaccurate, try modifying them with these suggestions:

- For generic responses, add the `Use Ocient SQL only` text and also the SQL reminders in **Key AI Prompt Recommendations**.
- For functions that appear incorrect, ask the AI tool to navigate to `/llms.txt` and cite the specific page it referenced.
- For incorrect structure, provide a minimal schema snippet and restate data types (such as complex types of arrays, tuples, or matrices).
- If navigation or footer text appears, restate `Start from llms.txt, ignore navigation and footer content`.
- If the AI tool continues to have issues accessing the documentation, download the PDF of the Ocient documentation and upload it into the AI prompt.

Related Links

Prompt Engineering

Key Concepts

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Connect to Ocient

You can connect to any of the SQL Nodes in an Ocient® System to run queries or commands. The SQL Node is available by default on port `4050` at the IP address or hostname assigned to the SQL Nodes. When more than one SQL Node is present in the Ocient System, Ocient automatically loads balance connections between the nodes to even out the load across connections.

Ocient provides a JDBC driver and a Python® driver named `pyocient`. You can use any of these drivers to connect to an online Ocient System.

To connect from a client computer that can access the SQL Node using the network, follow any of the following examples. Note that you must have the IP address of the SQL Node and a valid username and password to connect.

- **Connect Using JDBC**
- **Connect Using pyocient**

> If you are unable to connect, check the logs on the SQL Nodes for issues or check the **Errors and Warnings** for more detail on error codes.

For supported third-party tools, see **Ocient Integrations**.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Integrations

Ocient® databases can integrate with various industry tools and applications to expand their functionality.

The third-party tools listed in this section are certified to work with Ocient hyperscale systems. The included walkthroughs are intended to explain any requirements, setup instructions and additional specifications needed for the applications to work fully with Ocient Systems.

Database tools not included in this section can still function with Ocient Systems.

Supported Ocient Integrations

- **Looker Connector**
- **Superset Integration**
- **DBeaver Integration**
- **Metabase Integration**
- **SQLAlchemy Integration**
- **Tableau Integration**

> Ocient Systems assume that all date and time data is in UTC format. Therefore, third-party tools can require time zone configuration to get accurate results on date and time functionality. To configure the time zone settings for integrated applications, refer to the documentation for the respective tools.
>
> For more information, see the **Time Zone Functions** section.

SQLAlchemy is a trademark of Michael Bayer. mike(&)zzzcomputing.com All rights reserved.

© 2026 Ocient All Rights Reserved

Looker Connector

Google® Looker™ Studio is a free, online data visualization and business intelligence tool for connection to various data sources and creation of interactive dashboards and reports.

This connector enables clients to fetch data from an Ocient® System for visualization in Looker Studio using the **Google Apps Script** platform.

Prerequisites

To use Looker Studio with the Ocient System, you must have this software:

- Ocient System — Use the latest version.
- A Google® account with login credentials.

Connect to Ocient

Follow these steps to connect Looker Studio to your Ocient System.

1. Open Looker Studio and start a data source.

    - Go to **https://lookerstudio.google.com/** and sign in.
    - On the home page, select:
        - **Create** > **Data source**, or
        - From an open report, **Resource** > **Manage added data sources** > **Add a data source**.

© 2026 Ocient, Inc.

2 Select the Ocient Community Connector.

- In the connector gallery, go to the **Partner Connectors** section.
- Find and select the **Ocient DB** connector.

Looker Studio opens the **Ocient Connector** configuration page in the same tab.

3 Complete the configuration page by entering text in the fields. On the configuration page, select the **Allow to be modified** checkbox next to each field to modify it.

This table contains descriptions for each field.

Field	Description
Hostname	Hostname or IP address of the Ocient SQL Node.
Database	Database name in the Ocient System.
User	Ocient username.
Password	Password for the Ocient username.
Table	Ocient table name (`schema.table`).
Row Limit	`LIMIT` value to add to queries.
Date Range Column Name	Column name for the `DATE` or `TIMESTAMP` column in the `WHERE` clause.
Sort Column	Column name for the `ORDER BY` clause.
Sort Order	Sort order with the option of `ASCENDING` or `DESCENDING` .
Unnest Option	Specifies if array columns are unnested.
Custom Query	A query that overrides the query generated by the connector. The Looker data source panel shows only the columns included in this query. In this mode, Looker Studio infers column data types from the sample output, and not a `DESCRIBE TABLE` SQL statement.

- Select **Connect**.
- Select **Allow** when Looker Studio prompts you with "**Allow parameter sharing?**.

> You can now use Looker Studio to create reports for your Ocient System.

Looker Studio uses autocomplete in its filter tool, which can impact date or time filtering for records older than 28 days.

To fix this issue, navigate to the Looker filter control and set **Show suggested values** to `off`.

Ocient Data Types in Looker Studio

The connector represents Ocient data types using the closest approximation supported by Looker Studio. The inference of data types happens in Looker Studio in these ways:

- In normal mode, the connector generates data types from the table schema produced by a `DESCRIBE TABLE` SQL statement.
- In custom query mode, Looker Studio infers data types from sample output.

This table shows how Ocient data types correspond to Looker Studio data types.

Ocient Data Type	Looker Studio Data Type	Limitations
BOOLEAN	BOOLEAN	None
TINYINT, SMALLINT, INT, BIGINT, DOUBLE, FLOAT, DECIMAL	NUMBER	None
CHAR VARCHAR BINARY HASH IPV4 IP UUID VARBINARY TIME	TEXT	None
POINT LINESTRING POLYGON	TEXT	Looker Studio Community Connectors can use latitude or longitude coordinates, but they do not support native WKT geospatial types.

Ocient Data Type	Looker Studio Data Type	Limitations
DATE	YEAR_MONTH_DAY	None
TIMESTAMP	YEAR_MONTH_DAY_SECOND	None
CHAR[], VARCHAR[], BINARY[], HASH[], IPV4[], IP[], UUID[], VARBINARY[], TUPLE[]	TEXT	Looker Studio does not support arrays or tuples as a native data type. The connector represents container data types as textual strings. For arrays, use the **Unnest Option** configuration field to flatten array columns for analytics. For tuples, use the **Custom Query** field to extract the tuple elements. Setting the **Unnest Option** field does not work for flattening tuples.

Related Links

Connect to Ocient

Ocient Integrations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

DBeaver Integration

DBeaver is a free and open-source application for database administration and SQL querying.

Prerequisites

To use DBeaver with the Ocient® System, you must have these software prerequisites:

- Ocient System — Use latest version.
- JDBC Driver — Use latest version. See the Ocient Apache® Maven™ **repository** for a list of all versions.

To install, download the latest version of DBeaver from the **DBeaver Community** for your respective operating system.

Follow the installation steps and open DBeaver.

Connect to Ocient

You can connect to your database by following these steps:

- In the main DBeaver window, select **New Database Connection** in the top left of the window.
- In the New Database Connection window, find the Ocient driver. Select it, and then select **Next**.
- Enter your database parameters with this information.

Field	Description
Host	The hostname or IP address of the SQL Node of your Ocient System. For example: `databasehost-sql0`
Port	The port number for your connection. The default Ocient port is `4050`.
Database/Schema	The name of the database for connection. For example: `my_db`
User	The username associated with your database.
Password	The password associated with your username.

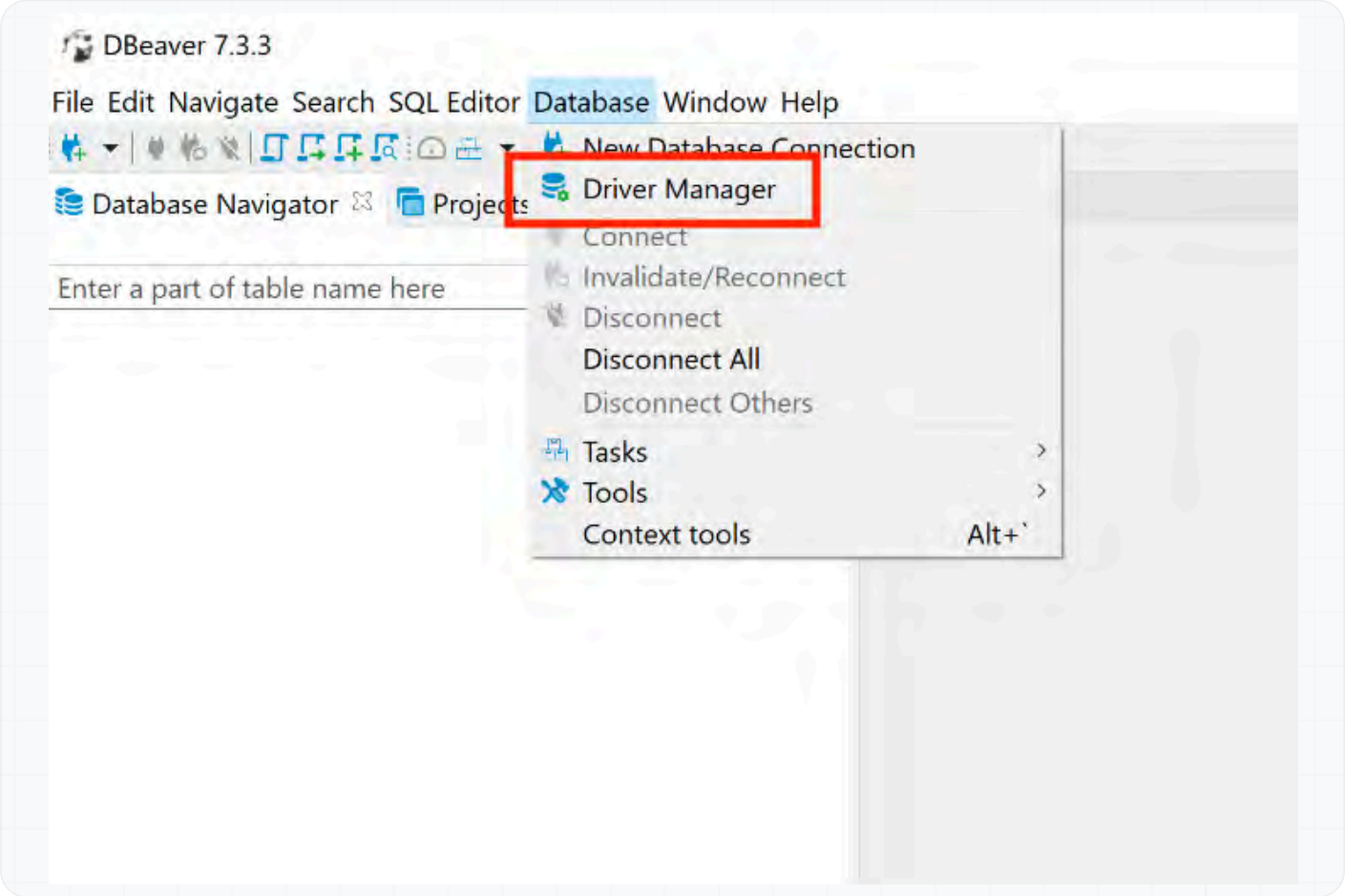
- Select **Finish**.

© 2026 Ocient, Inc.

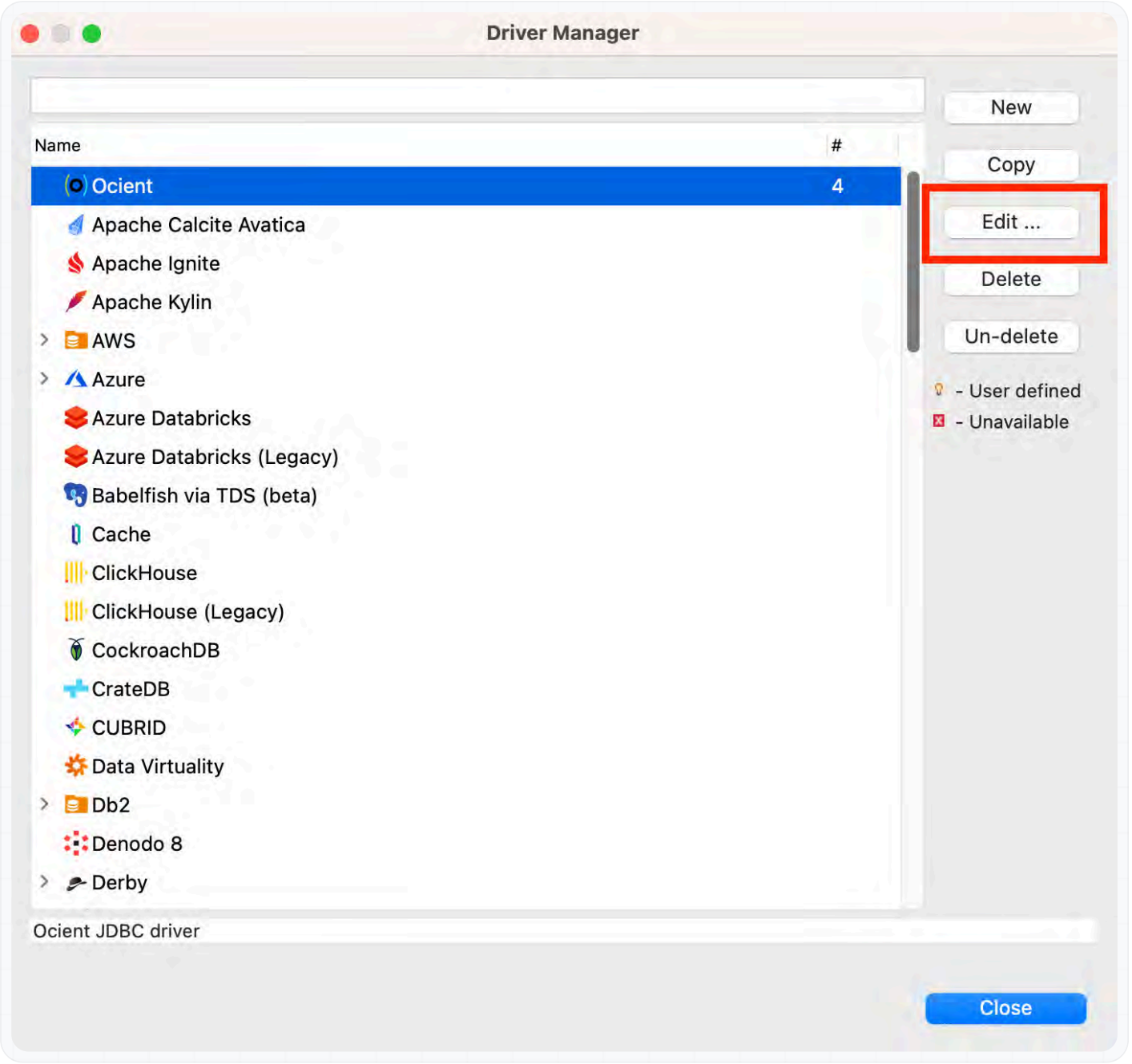
Update Ocient JDBC

By default, DBeaver downloads the latest JDBC version available in the **Maven repository**. If you need to update the JDBC driver or load a specific version, follow these steps.

- In the DBeaver window, select **Database** > **Driver Manager**.



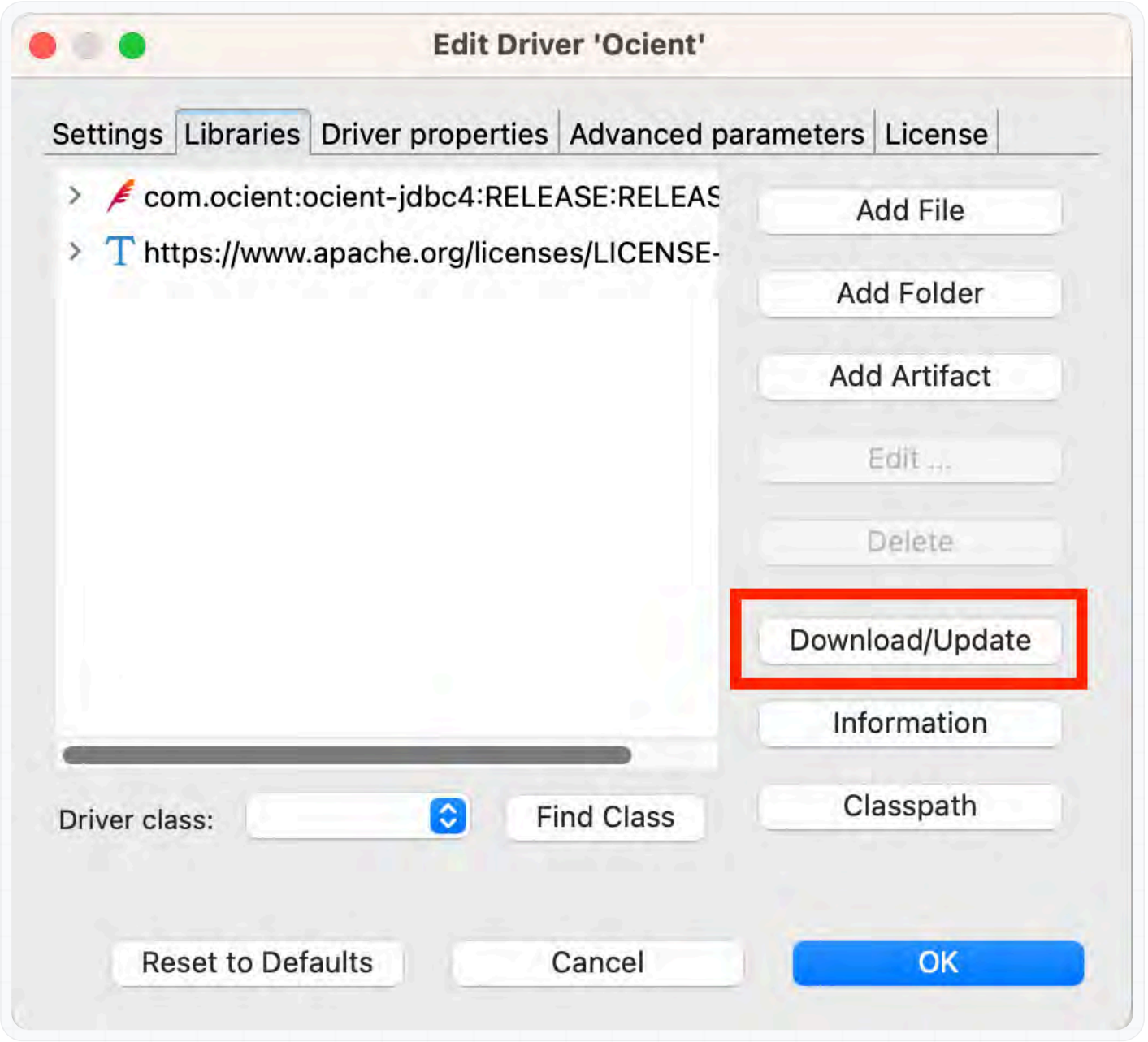
- Search among the list of driver options to find the Ocient driver, and select it.
- With the Ocient driver highlighted, select **Edit**.



- In the Edit window, select the **Libraries** tab.

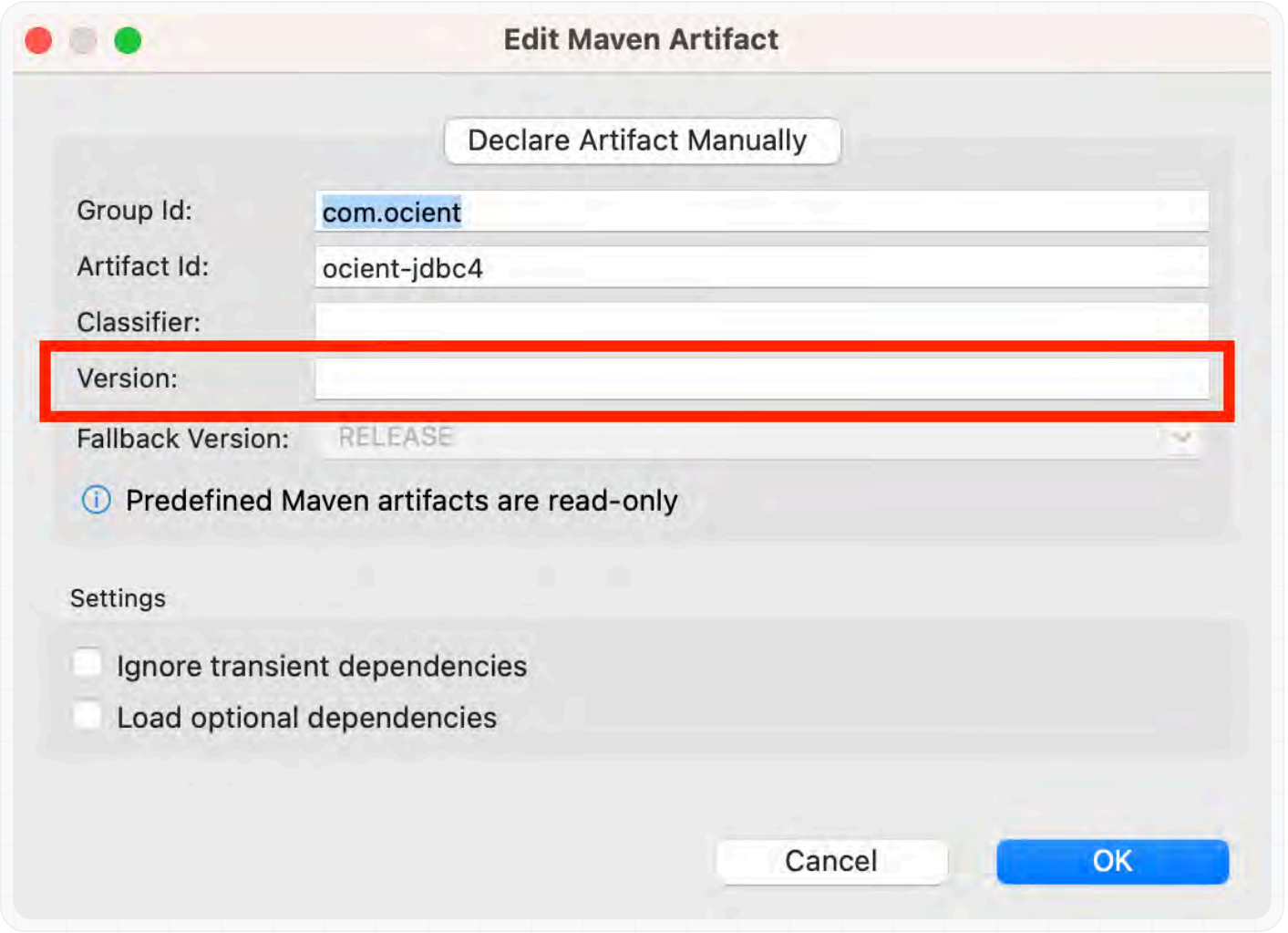
Update the Driver to the Latest Version

To update the driver to the latest version, select **Download/Update**.



Load a Specific JDBC Driver Version

- Double-click the `com.ocient:ocient-jdbc4:RELEASE` entry in the Libraries window. The Edit Maven Artifact window opens.
- Enter the JDBC version in the **Version** field.



For accurate time values, you might need to configure the DBeaver time zone settings.

- For macOS® systems, configure this setting in **Settings** > **User Interface**.
- For Windows® systems, configure this setting in **Preferences** > **User Interface**.

Related Links

Connect to Ocient

Ocient Integrations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Metabase Integration

Metabase℠ is an open-source business intelligence tool designed to create charts and dashboards from various data sources.

Prerequisites

To use Metabase with your Ocient® System, you must have these software prerequisites:

- Ocient System — Use version 22.0 or later.
- JDBC driver — Use the latest version.

Install Metabase

1. Go to the **Installing Metabase** and follow the directions for installation, including all prerequisites.
2. Download the latest **ocient-metabase-driver.jar** file and copy it to your Metabase plugins folder. This JAR file enables your Metabase installation to connect to Ocient.

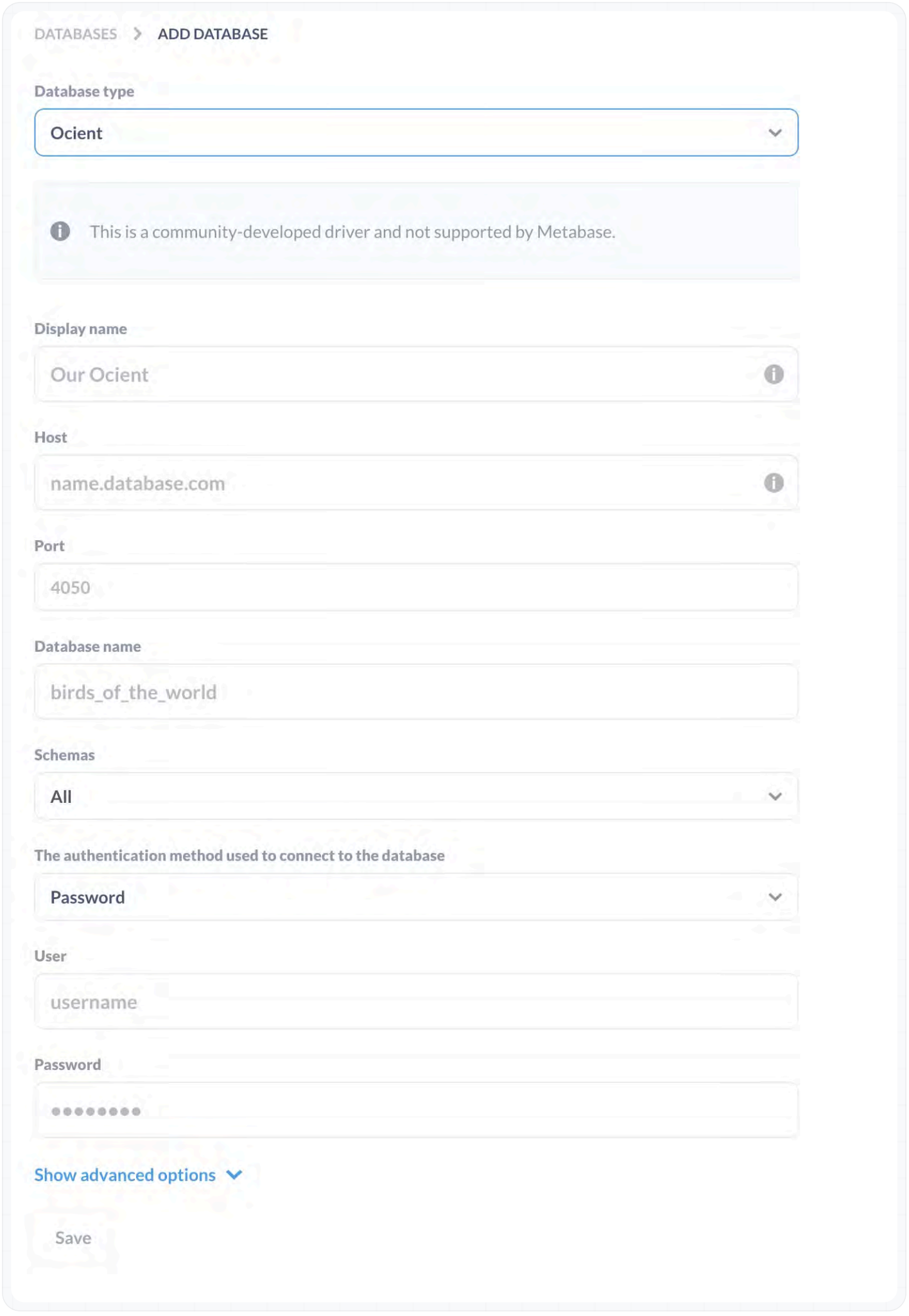
Connect to Ocient

To add a database connection, select the ⚙ icon at the top right of the Metabase homepage.

Select **Admin settings > Databases > Add database** in the menu.

This connection page appears. The displayed connection fields change after you select **Ocient** as the database type.

© 2026 Ocient, Inc.

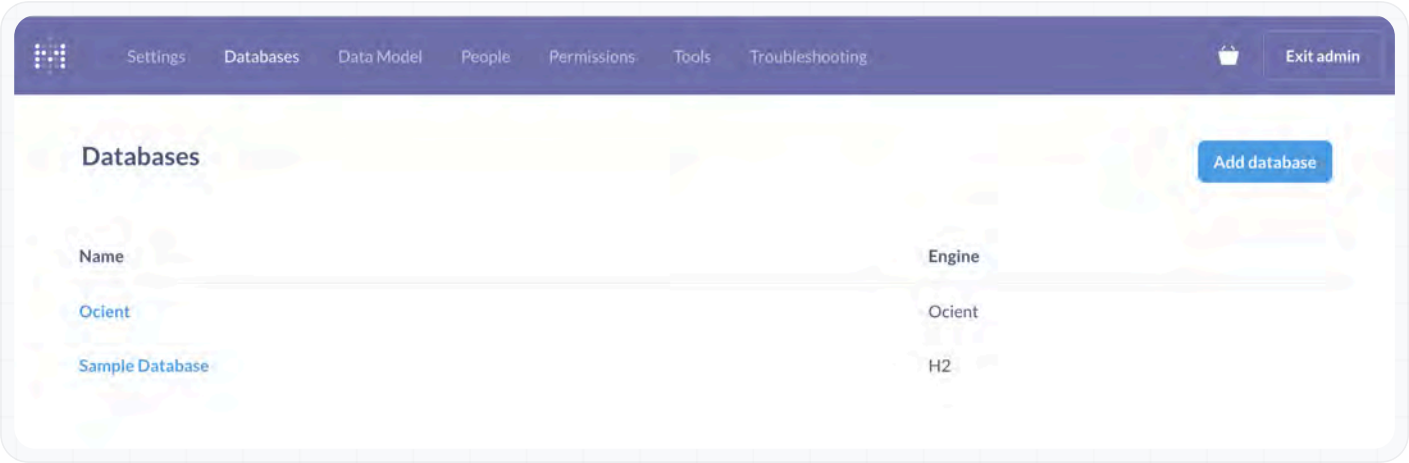

DATABASES
ADD DATABASE
Database type
Ocient
This is a community-developed driver and not supported by Metabase.
Display name
Our Ocient
Host
name.database.com
Port
4050
Database name
birds_of_the_world
Schemas
All
The authentication method used to connect to the database
Password
User
username
Password
Show advanced options
Save

Enter these fields in your connection settings.

Field	Description
Database type	The type of database. Select **Ocient**.
Display name	The display name for your database in Metabase.
Host	The host name or IP address of the SQL Node of your database. For example: `databasehost-sql0`
Port	The port number for your connection. Unless you have to alter this, the default Ocient port is `4050`.
Database name	The name of the database for connection. For example: `my_db`
Schemas	The identifiers for any schemas you want to use or exclude. To use all available schemas, select `All`.
Authentication method	Choose between `Single Sign-On (Token)` for single sign-on (SSO) or `Password` for password authentication.
User	The username associated with your database.
Password	The password associated with your username.
Type of SSO token	Use `access_token`.
SSO token	Specify your SSO access token string.

To use SSO authentication, you must have SSO integration enabled for the specified database and group. For more information, see **ALTER DATABASE SET SSO INTEGRATION**.

When you successfully connect, you see your database added to Metabase. Your database appears as one of the active connections in Metabase.



Your database appears as one of the active connections in Metabase.

Advanced Connection Options

You can add more connection settings as part of the JDBC connection string in Metabase. To add these, go to the **Add database** page and select **Show advanced options**.

You can specify options in the field **Additional JDBC connection string options**. In this image, the specified options include a maximum of 10,000,000 rows and a timeout of 0 milliseconds, using the `maxRows` and `timeoutMillis` parameters, respectively.



For a list of supported JDBC options, see the **JDBC Manual**.

Related Links

Connect to Ocient

Ocient Integrations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

SQLAlchemy Integration

SQLAlchemy is an open-source SQL toolkit and object-relational mapper for Python®.

Prerequisites

To use SQLAlchemy, you must have these software packages:

- Ocient® System — Use the latest version.
- Python — Use version 3.5 or later.
- pyocient driver — Use version 1.0.15. For details, see **pyocient Release Notes**.

Installation

For information about installing SQLAlchemy, please see the **sqlalchemy-ocient** PyPI℠ page.

Related Links

Connect to Ocient

Ocient Integrations

SQLAlchemy is a trademark of Michael Bayer. mike(&)zzzcomputing.com All rights reserved.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Superset Integration

Apache Superset is an open-source business intelligence web application that can quickly generate visualizations, dashboard, and geospatial maps.

Prerequisites

To use Superset with your Ocient® System, you must have these software prerequisites:

- Ocient System — Use the latest version.
- Python® — Use version 3.8 or later.
- SQLAlchemy — Install the latest version by using the **sqlalchemy-ocient** PyPI℠ page.
- pyocient driver — Use version 1.0.15. For details, see **pyocient Release Notes**.

Installation

To install, follow the **Installing Superset Locally Using Docker Compose** guide for your respective system.

Connect to Ocient

For specific steps on connecting Superset to your Ocient System, see the **Ocient DB** page.

Related Links

Connect to Ocient

Ocient Integrations

© 2026 Ocient, Inc.

SQLAlchemy is a trademark of Michael Bayer. mike(&)zzzcomputing.com All rights reserved.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Tableau Integration

Tableau® is an interactive data visualization application designed to create charts and dashboards from various data sources.

Prerequisites

To use Tableau with the Ocient® System, you must have these software prerequisites:

- Ocient System — Use the latest version.
- JDBC driver — Download the latest version with dependencies from the Apache® Maven™ **repository**.

Installation

1. Download the latest version of Tableau Desktop for your respective operating system from the **Download Tableau Desktop** website.
2. Register and sign in to Tableau Desktop. Then, follow steps to complete the installation.

Add Necessary Packages

To connect to Ocient, you must add these files to your Tableau installation:

- Ocient JDBC driver (`.jar`) — Use the latest version with dependencies from the **Maven repository**.
- Ocient Tableau connector file (`.taco`) — Download the latest version from **Tableau Exchange**.

> The `.taco` connector file supports Tableau Desktop and Tableau Server. Contact Tableau for support regarding Tableau Cloud.

Add the Ocient JDBC Driver JAR File

Move the downloaded JDBC driver JAR file to the corresponding folder based on your operating system:

- Windows® — `C:\Users[Windows User]\Documents\My Tableau Repository\Connectors`
- macOS® — `/Users/[user]/Documents/My Tableau Repository/Connectors`

© 2026 Ocient, Inc.

Add the Tableau Connector File (`.taco`)

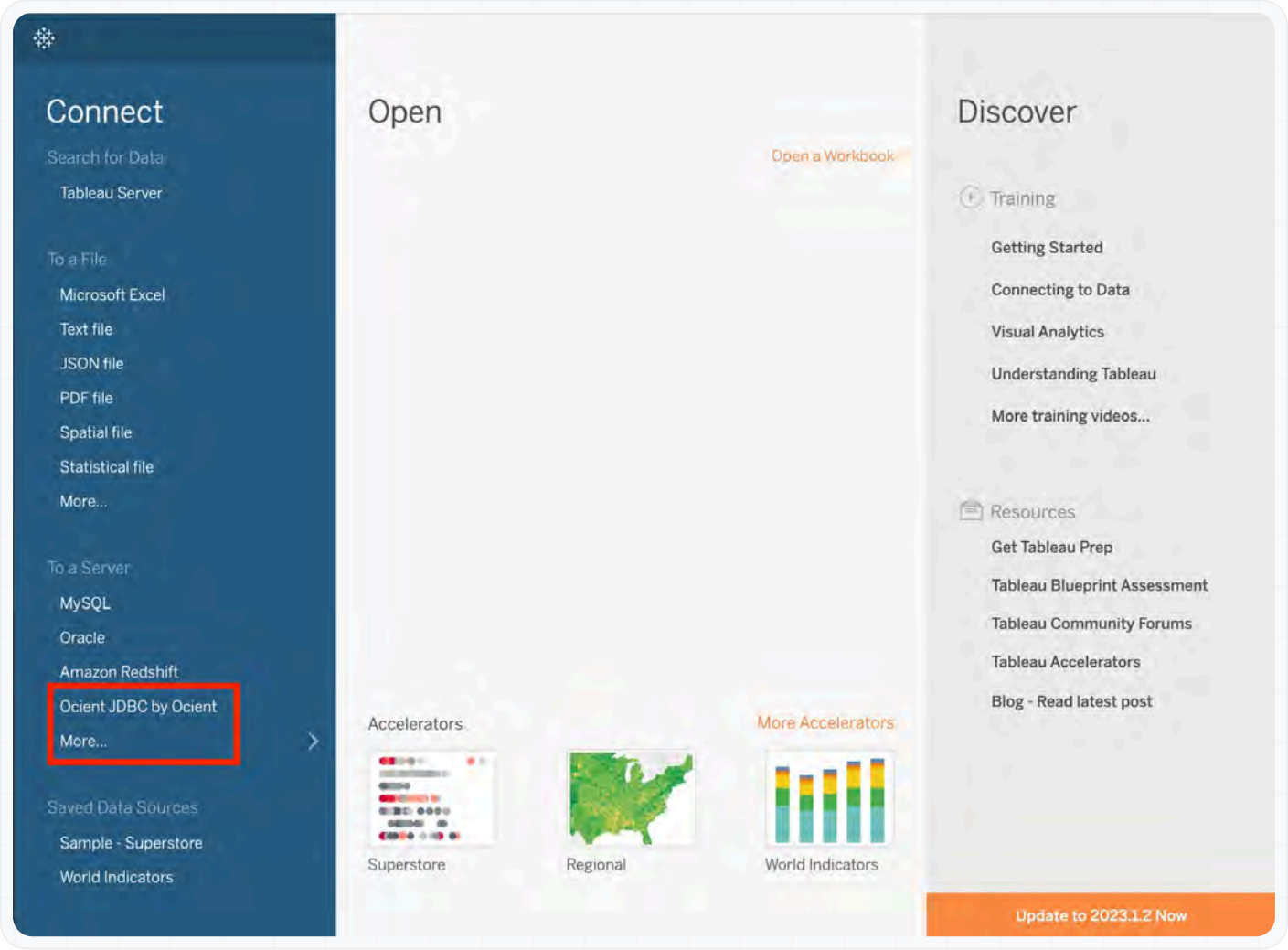
Move the downloaded `.taco` connector file to the corresponding folder based on your operating system:

- Windows — `C:\Program Files\Tableau\Drivers`
- macOS — `/Users/[user]/Library/Tableau/Drivers`

Connect to Ocient

Connect to your database by following these steps:

- Open Tableau and find the **Connect to a Server** menu. Select **Ocient JDBC** from the list of installed connectors. If **Ocient JDBC** does not appear as an option in Tableau, ensure that the Ocient Tableau connector file (`.taco`) is located in the correct folder.



- In the connection window, enter your database parameters with this information.

Field	Description
Host	The hostname or IP address of the SQL Node of your Ocient System. For example: `databasehost-sql0`
Port	The port number for your connection. The default Ocient port is `4050`.
Database	The name of the database for connection. For example: `my_db`
User	The username associated with your database.
Password	The password associated with your username.

- Select **Sign In** after you enter all the information.

Related Links

Connect to Ocient

Ocient Integrations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Connect Using JDBC

The JDBC driver can be executed in a command-line interface (CLI) mode or used directly in a program with standard JDBC methods. This example describes how to connect using the CLI mode. For details on using the JDBC driver in code, please see **Establishing a Connection**.

Step 1: Download the JDBC Driver

First, confirm the recommended version of the JDBC driver for your version of Ocient® by finding it in **Version Compatibility**. Download the correct JDBC driver from the Apache® Maven™ **repository** by clicking on your preferred version, then selecting **view all**, and finally downloading the JAR file that is labeled with the pattern `ocient-jdbc4-<version number>-jar-with-dependencies.jar`.

Save this file to a convenient location on your client machine.

Step 2: Run the JDBC CLI Program

To run the JDBC CLI Program, you must have Java® installed on your machine. For the version, see **Version Compatibility**. Next, run the following command to start the JDBC CLI Program. Assuming your JAR stored at `~/ocient-jdbc4-jar-with-dependencies.jar`.

Shell

```
java -classpath ~/ocient-jdbc4-jar-with-dependencies.jar com.ocient.cli.CLI
```

The system prompts you to enter the username and password.

Shell

```
Username: test@system
Password: fakepassword
```

© 2026 Ocient, Inc.

Example Response:

```
Ocient> _
```

The Ocient CLI prompt appears.

Step 3: Connect

To connect, enter the connection string to your SQL Nodes. SQL Nodes install by default with a self-signed TLS certificate for encryption by default, and a custom certificate can be present. Depending on the TLS configuration, you can use `tls=unverified` (default) or `tls=on` for the required setting.

Assuming the standard port `4050`, a self-signed certificate, and the SQL Node IP Address of `10.10.1.1`, you can connect to the system database with the following connecting string:

```
Ocient> connect to jdbc:ocient://10.10.1.1:4050/system;
```

Example Response:

```
Connected to jdbc:ocient://10.10.1.1:4050/system
Ocient> _
```

You are now connected and can execute queries based on your user's permissions. Try running: `select * from sys.tables;` to see what tables are defined.

For more details on using the JDBC driver and the CLI program, see the **JDBC Manual**.

The system cancels any queries made with the JDBC driver if you perform any of these actions:

- Fetch the entire result set.
- Close the connection.
- Execute a new query on the same connection.

Related Links

Connect to Ocient

Ocient Integrations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Connect Using pyocient

The Ocient® Python® module `pyocient` supports direct connections to Ocient Systems in a Command-Line Interface (CLI) as well as in Python programs that follow the Python Database API Specification 2.0.

After you install `pyocient`, you can establish Ocient connections to run queries and commands.

Installation Tutorial

To install and run `pyocient`, follow these steps.

1 Install pyocient

Install `pyocient` using pip.

Shell

```
pip3 install pyocient
```

After running the pip install, `pyocient` is installed in the `bin` directory.

> Use `venv` to install `pyocient` using a virtual environment. For more information, see the Python **venv** documentation.

2 Start pyocient

`pyocient` is started from the command line with a connection string. It includes many options to configure the connection and client similar to the JDBC driver. Learn more about supported options in the **Ocient Python Module: pyocient**.

After installation, you can launch from a command line by typing the `pyocient` command.

© 2026 Ocient, Inc.

Shell

```
pyocient ocient://example_user:example_password@10.10.1.1/system
```

To connect to the system database, replace the example username (`example_user`) and password (`example_password`) with your credentials, set your IP address,

The connection example assume a SQL Node exists at `10.10.1.1`. The example uses the default port `4050`, and is set up with default self-signed certificates for TLS.

Example Response

SQL

```
Ocient Database™
System Version: <Ocient System Version>, Client Version <pyocient Version>
> _
```

Use the command `quit;` to exit the `pyocient` CLI interface.

3 Run a Query

In the `pyocient` CLI, you can run any SQL query supported by your database. In this example, a query is selecting rows from the system tables:

SQL

```
SELECT * FROM sys.tables;
```

You can also run queries from the command line with the connection string:

Shell

```
pyocient ocient://example_user:example_password@10.10.1.1/system 'SELECT *
```

Example Response

SQL


```
[
  {
        "id": "359e21b1-770a-4860-aec9-f158d307f725",
        "name": "data_type_coverage",
        "schema": "loading",
        "database_id": "f827ff7c-dd94-43bf-8386-93d048344a32",
        "storage_space_id": "2ad08e7c-15d1-4861-b38b-256c4df4a191",
        "maximum_segment_size_gib": 4,
        "description": null,
        "streamloader_property_string": null,
        "created_at": "2023-08-08T21:22:49.622213"
    }
]
```


The system cancels any queries made with the `pyocient` driver if you perform any of these actions:

- Fetch the entire result set.
- Close the connection.
- Execute a new query using the same connection.

For more information and API reference material for pyocient, see the **Ocient Python Module: pyocient** page.

Related Links

Connect to Ocient

Ocient Integrations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Load Data

Loading data into the OcientAIQ™ Unified Data Platform using the data pipeline functionality is as simple as writing a SQL statement to query data from different supported sources. The loading operation provides real-time transformations and maximizes performance using all available processing resources.

With data pipelines, you manage data loading activities using Data Definition Language (DDL) statements similar to the ones used for managing database tables. You can access all information about pipelines by querying the system catalog tables. You can execute operations on pipelines with a command-line statement using standard interfaces like JDBC and ODBC.

Data Pipelines Overview

A data pipeline is the primary way that the Ocient® System loads data. Each data pipeline is a database object that defines the end-to-end processing of data for the extraction, transformation, and load into Ocient tables. Data pipelines execute across your Ocient System to coordinate parallel loading tasks across many Loader Nodes and pipelines.

The Pipeline Object

Creating a pipeline is complex as there are many options to consider. Use the `PREVIEW PIPELINE` SQL statement to preview the pipeline creation and load of data to ensure that the pipeline definition returns the results you expect. After you are satisfied with the results, you can execute the `CREATE PIPELINE` statement to create the pipeline at scale. For details, see the **CREATE PIPELINE** SQL statement.

When you create a pipeline, you assign it a name and the pipeline exists as an object in the Ocient System connected to this name. Then, you can control the object using your chosen name with SQL statements like `START PIPELINE`, `STOP PIPELINE`, `CREATE OR REPLACE PIPELINE`, and `DROP PIPELINE`. You can also define your own function by using the `CREATE PIPELINE FUNCTION` SQL statement. Execute these statements using a SQL connection.

A pipeline maintains its own position during a load and enforces deduplication logic to ensure that the Ocient System only loads data once. The lifecycle of a pipeline defines both the deduplication logic and load position.

- Pipeline Events: During the life of a pipeline, you can start, stop, modify, and resume the pipeline without duplicating source data or losing the position in a load.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

- Pipeline Updates: You can modify pipelines using the `CREATE OR REPLACE` SQL statement to update the transforms in a pipeline, but maintain the current position in the load.
- Load Position: If target tables are truncated or if a target table is dropped and then recreated, the pipeline maintains its own position in the load and will continue from its last position.

Pipelines also gracefully handles many error cases and system failure modes. The pipeline stores any errors that occur in the system catalog in association with the pipeline.

For more details on Deduplication and Error Tolerance settings, see:

- **Deduplication in Data Pipelines**
- **Error Tolerance in Data Pipelines**
- **START PIPELINE Reference**

Parts of a Pipeline

Pipelines can operate in either a `BATCH` or `CONTINUOUS` mode based on how you plan to load your data. Or, for transactional batch loads, you can use the `TRANSACTIONAL` mode with the optional `START FOREGROUND` clause.

The DDL for Data Pipelines has three sections.

Section	Syntax
Data Source	`SOURCE ...`
Data Format	`EXTRACT FORMAT ...`
Transformations	`INTO table SELECT ...`

Thus, for example, the full SQL syntax has this format for the `orders_pipeline` data pipeline. This syntax has an S3 data source and loads data into the `orders` table in the `public` schema. The `SELECT` SQL statement in the `CREATE PIPELINE` statement selects the `id`, `user_id`, `product_id`, and other columns from a table.

SQL

```
CREATE PIPELINE orders_pipeline
SOURCE
  S3
    ENDPOINT '...'
    BUCKET '...'
    FILTER '...'
EXTRACT
  FORMAT delimited
  ...
INTO "public"."orders"
SELECT
  $1 as id,
  $2 as user_id,
  $3 as product_id,
  ...
```

Data Source

The `SOURCE` part in a pipeline defines the source of the data to load. This DDL part identifies the type of the source (e.g., Amazon® Web Services℠ (AWS℠) S3, Apache® Kafka®) as well as the details for the data that is relevant to the source (e.g., S3 bucket and filters, Kafka topic and consumer configuration).

- **Kafka Source Options**
- **S3 Source Options**
- **File-based Source Options**
- **SOURCE Options Reference**

Data Format

The `EXTRACT` part in a pipeline defines the format of the data that the Ocient System extracts from the data source. This part includes the data format (e.g., delimited, binary, JSON) and the details about the records (e.g., record delimiter, how to treat empty data). In addition, the Ocient System extracts metadata that you can load into tables.

- **Loading Delimited and CSV Data**
- **Loading JSON Data**
- **Load Parquet Data**
- **Loading Binary Data**

- **Loading Metadata Values**
- **EXTRACT Options Reference**

Transformations

The `INTO table SELECT ...` SQL statement defines the target tables where the data loads and uses a `SELECT` SQL statement to extract fields from the source data (e.g., CSV field index `$5`, JSON selector `$order.user_name`) and map them to target columns (e.g., `... as my_column`).

The `SELECT` statement in a pipeline can utilize a subset of SQL functions to convert the extracted data into the required format during loading (e.g., `TO_TIMESTAMP($my_date, 'YYYY-MM-DD')`).

> The selectors you use to extract data from the source records can differ based on the data format (e.g., `JSON` or `DELIMITED`).

- **Supported Transformation Functions**
- **Use Source Field References**
- **Supported JSON Selectors**

For loading workflows, see these examples:

- **Data Pipeline Load of CSV Data from S3**
- **Data Pipeline Load of JSON Data from Kafka**
- **Data Pipeline Load of Parquet Data from S3**

Pipeline Lifecycle

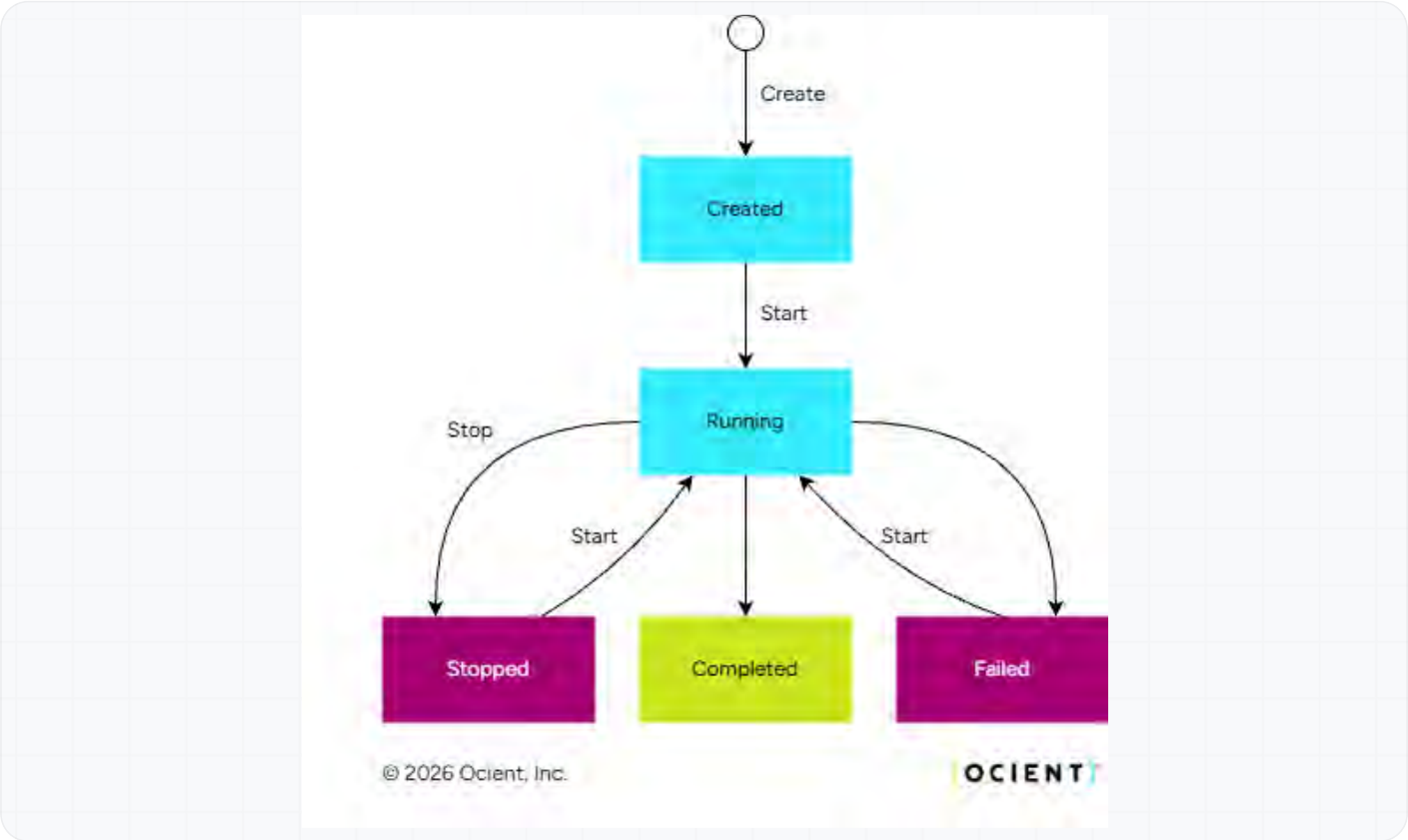
This state machine shows the state transitions for a data pipeline. You can start and stop data pipelines using DDL statements. The status of the pipeline transitions automatically as the pipeline completes all assigned work or reaches a failed state.

While executing, the pipeline lists files or connects to streaming sources and creates tasks to execute work. These states reflect the overall progress of a pipeline and capture details of the underlying tasks that execute the work of the pipeline.

State	Description
Created	The pipeline is created, but no user has started it. * Zero tasks in the `CREATED` state. * Zero files listed or no Kafka Consumer group created.

State	Description
Running	The pipeline is running after a user starts the pipeline. * At least one task is in the `QUEUED`, `RUNNING`, or `CANCELLING` states. * At least one file listed or Kafka Consumer group created.
Stopped	The pipeline is not processing data after a user stops the pipeline. * All tasks are in the `QUEUED`, `COMPLETE`, `FAILED`, or `CANCELLED` states.
Completed	The pipeline finished all assigned work according to the error limits defined for the pipeline. * All tasks in are in the `COMPLETE` state.
Failed	The pipeline failed to complete and no longer runs due to error limits defined for the pipeline. * At least one task in the `FAILED` state. * All tasks are in the `QUEUED`, `COMPLETE`, `FAILED`, or `CANCELLED` states.

In this state diagram, the transition arrows display the user actions that can trigger a state change. State transitions without a label are system-initiated transitions.



Observe Pipelines

During pipeline operation, all of the information you need to observe progress, pipeline status, key events, success or failure, and errors is available in system catalog tables in the Ocient System. In addition, performance counters are available for many key loading metrics that you can add to observability systems.

- **Monitor Data Pipelines**
- **System Catalog**

Troubleshooting Pipelines

Data pipelines support robust error-handling capabilities crucial for developing new pipelines, identifying issues with ongoing operations, and correcting bad data. Pipelines can include a bad data target where the Ocient System saves bad data for troubleshooting. In addition, the system captures errors that you encounter during loading in system catalog tables such as `sys.pipeline_events` and `sys.pipeline_errors`. Exploring the data in these tables enables you to detect issues and identify the root cause of errors during loading.

- **Manage Errors in Data Pipelines**
- **Frequently Asked Questions for Data Pipelines**
- **Data Pipeline Loading Errors**
- **Bad Data Targets Reference**
- **Error Handling Options Reference**

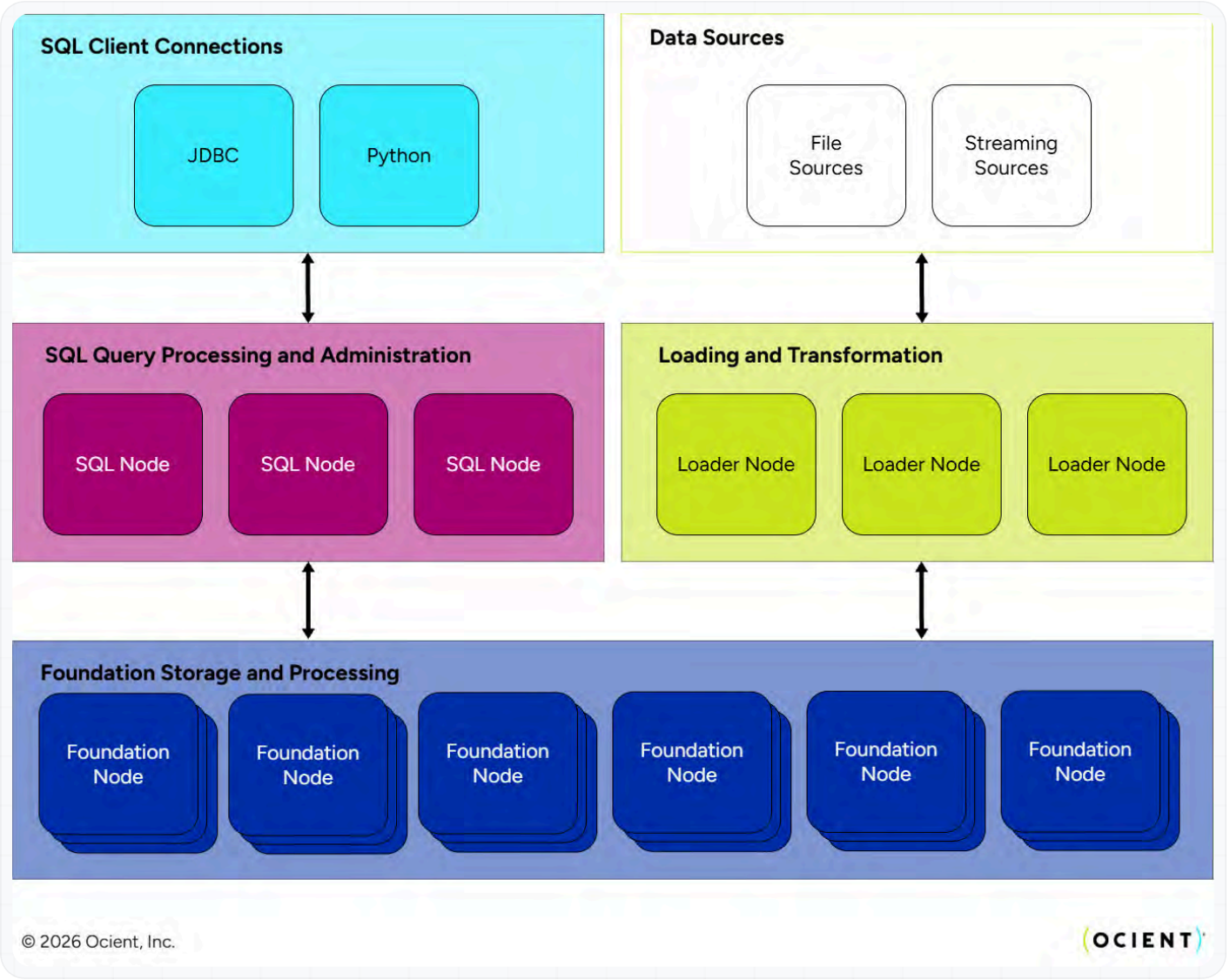
Loading Architecture

Pipelines require Loader Nodes to operate. Loader Nodes are Ocient Nodes that have the `streamloader` role assignment. The loading system can use all active nodes with this role to process pipelines.

When you execute a pipeline, Ocient spreads the work among the available Loader Nodes. The system executes these parallel processes as tasks on each of the Loader Nodes. Then, the system partitions the files or Kafka partitions to load across the tasks, and loading proceeds in parallel. When all tasks are completed, the pipeline is completed. Many pipelines can run in parallel on the Loader Nodes, and these Nodes share resources.

The scale-out architecture of the Ocient System allows you to add more Loader Nodes as needed to expand the volume and complexity of pipelines that you can operate at the throughput required by your application.

© 2026 Ocient, Inc.



Internal Operations of a Pipeline

As pipelines process records from a data source, the Ocient System builds these rows into a row-based storage structure called a page. The system optimizes pages for rapid ingestion of data. The system stores pages on Foundation Nodes to ensure data integrity.

As pages accumulate, Loader Nodes convert pages into a columnar storage format named a Segment. Segments are highly compressed data structures that include data, indexes, and other statistical metadata to optimize query performance. After the Ocient System creates Segments, the system forms them into groups and transfers them to Foundation Nodes for storage.

You do not typically need to be concerned with or control these internal operations. However, it is important to understand that pipeline throughput can be affected by the parallel processing of many pipelines on shared resources, the number of indexes, and the type of compression used on your tables.

Related Links

Pipeline Privileges Reference

Data Pipelines Reference

PREVIEW PIPELINE

CREATE PIPELINE

DROP PIPELINE

START PIPELINE

STOP PIPELINE

ALTER PIPELINE RENAME

EXPORT PIPELINE

CREATE PIPELINE FUNCTION

Transactional Data Pipelines

Frequently Asked Questions for Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Pipeline Load of CSV Data from S3

A common setup for loading files in a batch into Ocient® is to load from a bucket on Amazon® Web Services℠ (AWS℠) S3 with time-partitioned data. Often, you must perform a batch load repeatedly to load new files.

Ocient uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a simple example load using a small data set in CSV format. The data in this example comes from a test set for the Metabase℠ Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active Storage Cluster (See the **Ocient Application Configuration** guide).

Step 1: Create a New Database

Connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then, execute the `CREATE DATABASE` SQL statement for a database named `metabase`.

SQL

```
CREATE DATABASE metabase;
```

Step 2: Create Table

Create the `orders` table in the new database. First, connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), and then execute this `CREATE`

`TABLE` SQL statement that specifies to create a table with these columns and a clustering index based on the `user_id` and `product_id` columns:

- `created_at` as a TimeKey® timestamp that is not nullable.
- `id`, `user_id`, and `product_id` as integers that are not nullable.
- `subtotal`, `tax`, `total`, and `discount` as floating point numbers.
- `quantity` as an integer.

SQL

```
CREATE TABLE public.orders (
    created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
    id INT NOT NULL,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    subtotal DOUBLE,
    tax DOUBLE,
    total DOUBLE,
    discount DOUBLE,
    quantity INT,
    CLUSTERING INDEX idx01 (user_id, product_id)
);
```

The database creates the `orders` table, and you can begin loading data.

Step 3: Create a Data Pipeline

Create data pipelines using the `CREATE PIPELINE` SQL statement. To load data, you first create a pipeline with the definition of the source, data format, and transformation rules using a SQL-like declarative syntax. Then, you execute the `START PIPELINE` SQL statement to start the load. You can observe progress and status using system tables and views.

Each Ocient pipeline defines a single data source and the target table or tables into which data loads. A data source includes the location of the source and filters on the source to define the specific data set to load. This example loads data from a data source located in a directory within the S3 bucket.

First, inspect the data that you plan to load. Each document has a format similar to this example CSV file named `orders.csv`.

Text

```
id,user_id,product_id,subtotal,tax,total,discount,created_at,quantity
1,1,14,37.65,2.07,39.72,null,2019-02-11T21:40:27.892Z,2
2,1,123,110.93,6.1,117.03,null,2018-05-15T08:04:04.580Z,3
3,1,105,52.72,2.9,49.2,6.42,2019-12-06T22:22:48.544Z,2
...
```

In this case, Ocient automatically transforms the data to the target columns using some sensible conventions. In other cases, loads require some transformation. Most transformations are identical to functions that already exist in the SQL dialect of the Ocient System.

Create a pipeline named `orders_pipeline` for the orders data set from your database connection prompt. Use the S3 data source with endpoint `https://s3.us-east-1.amazonaws.com`, bucket `ocient-docs`, and filter `metabase_samples/csv/orders.csv`. Specify the CSV format with one header line. Load the data into the `public.orders` table. The `SELECT` part of the SQL statement maps the fields in the CSV file to the target columns in the created table.

© 2026 Ocient, Inc.

SQL

```sql
CREATE PIPELINE orders_pipeline
SOURCE
    S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/csv/orders.csv'
EXTRACT
    FORMAT csv
    NUM_HEADER_LINES 1
INTO public.orders
SELECT
    $1 as id,
    $2 as user_id,
    $3 as product_id,
    $4 as subtotal,
    $5 as tax,
    $6 as total,
    $7 as discount,
    $8 as created_at,
    $9 as quantity;
```

The pipeline has three main sections:

- `SOURCE` — In this case, load data from S3. Specify the endpoint and bucket. The `FILTER` parameter identifies the file or files to load. The load uses a single CSV file. Options exist to add wildcards or regular expressions to isolate different file sets.
- `EXTRACT` — Set the format to CSV files and note that there is one header line in the file. This specification skips that row when the Ocient System processes the file. Many other options exist for delimited data such as a record delimiter and field delimiter.
- `INTO ... SELECT` — Choose the target table `public.orders` and select the fields from the CSV file. The numeric index identifies each file field. Importantly, similar to other SQL syntax, the first field in the file is `$1`, not `$0`. Each field maps to a target column using the `as` syntax.

After you successfully create the `orders_pipeline` pipeline, execute the `START PIPELINE` SQL statement.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Observe the Load Progress

With your pipeline running, data immediately begins to load from the S3 files that you defined. If there are many files in each file group, the load process first sorts the files into batches, partitions them for parallel processing, and assigns them to Loader Nodes.

You can check the pipeline status and progress by querying `information_schema.pipeline_status` or by executing `SHOW PIPELINE_STATUS`

SQL

```
SHOW PIPELINE_STATUS;
```

Output

SQL

```
database_name         pipeline_name       table_names          status      status
-------------------------------------------------------------------------------
metabase              orders_pipeline     ["public.orders"]    RUNNING     Starte
```

After the status of the pipeline changes to `COMPLETED`, all data is available in the target table.

After a few seconds, the data is available for query in the `public.orders` table.

SQL

```
SELECT COUNT(*) FROM public.orders;
```

Output

```
SQL
count(*)
--------------------
18760
```

You can drop the pipeline with the `DROP PIPELINE orders_pipeline;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

Data Pipelines Reference

Load Delimited and CSV Data

Data Types for Data Pipelines

Transform Data in Data Pipelines

Manage Errors in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Pipeline Load of JSON Data from Kafka

A common setup for streaming data into Ocient® is to load JSON data from an Apache® Kafka® topic.

Ocient uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a simple example load using a small data set in JSON format. The data in this example comes from a test set for the Metabase℠ Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to a Kafka broker from the SQL and Loader Nodes.
2. An Ocient System is installed and configured with an active Storage Cluster (see the **Ocient Application Configuration** guide).

Step 1: Create a New Database

Connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then, execute the `CREATE DATABASE` SQL statement to create the `metabase` database.

SQL

```
CREATE DATABASE metabase;
```

Step 2: Create Table

© 2026 Ocient, Inc.

Create the `orders` table in the new database. First, connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), and then execute this `CREATE TABLE` SQL statement that specifies to create a table with these columns and a clustering index based on the `user_id` and `product_id` columns:

- `created_at` as a TimeKey® timestamp that is not nullable.
- `id`, `user_id`, and `product_id` as integers that are not nullable.
- `subtotal`, `tax`, `total`, and `discount` as floating point numbers.
- `quantity` as an integer.

SQL

```
CREATE TABLE public.orders (
    created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
    id INT NOT NULL,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    subtotal DOUBLE,
    tax DOUBLE,
    total DOUBLE,
    discount DOUBLE,
    quantity INT,
    CLUSTERING INDEX idx01 (user_id, product_id)
);
```

The database creates the `orders` table, and you can begin loading data.

Step 3: Create a Data Pipeline

Create data pipelines using the `CREATE PIPELINE` SQL statement. To load data, you first create a pipeline with the definition of the source, data format, and transformation rules using a SQL-like declarative syntax. Then, you execute the `START PIPELINE` command to start the load. You can observe progress and status using system tables and views.

Each Ocient pipeline defines a single data source and the target table or tables into which data loads. A data source includes the location of the source and filters on the

source to define the specific data set to load. This example loads data from two data sources, where each source is located in a directory on the same S3 bucket.

First, inspect the data that you plan to load. Each document has a format similar to this example.

JSON

```
{"id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total"
{"id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total
```

In this case, Ocient automatically transforms the data to the target columns using some sensible conventions. In other cases, loads require some transformation. Most transformations are identical to functions that already exist in the SQL dialect of the Ocient System.

Create a pipeline named `orders_pipeline` for the orders data set from your database connection prompt. Specify the Kafka source with broker address `192.168.0.1:9092` (replace this example Kafka broker address with your address) and topic `orders`. Load the data into the `public.orders` table. The `SELECT` part of the SQL statement maps the JSON fields to the target columns in the created table.

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE orders_pipeline
SOURCE
    KAFKA
        BOOTSTRAP_SERVERS '192.168.0.1:9092'
        TOPIC 'orders'
EXTRACT
    FORMAT json
INTO public.orders
SELECT
    $id as id,
    $user_id as user_id,
    $product_id as product_id,
    $subtotal as subtotal,
    $tax as tax,
    $total as total,
    $discount as discount,
    $created_at as created_at,
    $quantity as quantity;
```

The pipeline has three main sections:

- `SOURCE` — Loads data from Kafka. Specify the address of the bootstrap servers and topic. Options exist to set Kafka consumer configurations.
- `EXTRACT` — Sets the format to JSON.
- `INTO ... SELECT` — Targets the `public.orders` table and selects the chosen fields from the JSON records. In this case, all data is available at the top level of the JSON records, so the example references the fields by the attribute name (e.g., `$id`, `$user_id`, etc.). For nested data, reference the nested fields using dot notation (e.g., `$order.user.first_name`). Each field maps to a target column using the `as` syntax.

After you successfully create this pipeline, execute the `START PIPELINE` SQL statement.

© 2026 Ocient, Inc.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Confirm that Loading is Operating Correctly

With the pipeline in place and running, data immediately begins loading off of the Kafka topics that are configured in the pipeline. If you do not have data in the Kafka topics yet, now is a good time to start producing data into the topics.

Produce Test Data into Kafka

For test purposes, **kafkacat** is a helpful utility that makes it easy to produce records into a topic. For example, if you have a file of sample data `orders.jsonl` in a JSONL format (newline-delimited JSON records), you can run this command to send those records into your Kafka broker. Specify your broker IP address instead of `<broker_ip_address>` and your topic name `<topic_name>`.

Shell

```
kafkacat -b <broker_ip_address>:9092 -t <topic_name> -T -P -l orders.jsonl
```

Save an example document to your file system to use for this test. For this example, you can download an example file from **https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl** and save it to `~/orders.jsonl`.

Assuming the broker is running at the IP address `10.0.0.3`, send data into the `orders` topic defined in the pipeline definition by executing this command.

Shell

```
kafkacat -b 10.0.0.3:9092 -t orders -T -P -l orders.jsonl
```

This command pushes the entire JSONL file of messages into Kafka with one record per line. As these records are produced into Kafka, the running pipeline begins to load

them into Ocient.

Step 5: Observe the Load Progress

With your pipeline running, data immediately begins to load from the Kafka topic. The pipeline creates parallel Kafka consumers for each partition. If there are more partitions than processing cores available, the pipeline automatically handles spreading consumers across available processing cores and Loader Nodes.

You can check the pipeline status and progress by querying `information_schema.pipeline_status` or executing the `SHOW PIPELINE_STATUS` SQL statement.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

SQL

```
database_name         pipeline_name       table_names          status      status
-------------------------------------------------------------------------------
metabase              orders_pipeline     ["public.orders"]    RUNNING     Starte
```

After a few seconds, the data is available for query in `public.orders`.

SQL

```
SELECT COUNT(*) FROM public.orders;
```

Output

SQL

```
count(*)
--------------------
18760
```

Unlike a batch file load, Kafka pipelines run continuously, so they never change to a status of `COMPLETED`. To examine progress, you can use the information schema and system catalog tables. Key details are in `information_schema.pipelines`, `information_schema.pipeline_status`, and `sys.pipeline_partitions`.

For example, this statement shows the status and key metrics for duration, loaded records, and failed records.

SQL

```
SELECT
    pipeline_name,
    status,
    status_message,
    duration_seconds,
    records_loaded,
    records_failed
FROM
    information_schema.pipeline_status;
```

Output

SQL

```
pipeline_name    status    status_message                                duration_s
-------------------------------------------------------------------------------
orders_pipeline  RUNNING   Started processing pipeline orders_pipeline  10.089936
```

In this example, there are no errors. The `sys.pipeline_errors` system catalog table captures any errors that occur during the pipeline process.

You can drop the pipeline with the `DROP PIPELINE orders_pipeline;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

Data Pipelines Reference

Load JSON Data

Data Types for Data Pipelines

Transform Data in Data Pipelines

Manage Errors in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Pipeline Load of Parquet Data from S3

A common setup for loading files in a batch into the Ocient® System is to load from a bucket on Amazon® Web Services℠ (AWS℠) S3 with time-partitioned data. Often, you must perform a batch load repeatedly to load new files.

The Ocient System uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a simple example load using a small data set in Apache® Parquet™ format. The data in this example comes from a test set for the Metabase℠ Business Intelligence tool.

Parquet Loading Recommendations

Follow this set of recommendations for an optimal loading experience of Parquet files.

File Configuration

- Files should have row groups of less than 128 MB. Larger row groups can impact memory usage during loading, and row groups of 512 MB can cause loading failures on 1 TB or more data sets.
- Encoding fields in a Parquet file reduces the space of the file on disk but can impact memory usage during loading. Enable encoding on fields that you expect to have less than 256 unique values and for fields that contain short strings. You do not have to encode other fields.

Multiple Files

- You can load row groups of multiple Parquet files in parallel. For large data sets, load the data set as multiple files.
- Loading files with differing schemas is not supported.

Prerequisites

© 2026 Ocient, Inc.

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active Storage Cluster. For details, see **Ocient Application Configuration**.

Parquet Loading Example

Follow these steps to load Parquet data into the Ocient System.

Step 1: Create a New Database

Connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then, execute the `CREATE DATABASE` SQL statement to create the `metabase` database.

```sql
CREATE DATABASE metabase;
```

Step 2: Create a New Table in the Database

Create the `orders` table in the new database. First, connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), and then execute this `CREATE TABLE` SQL statement that specifies to create a table with these columns and a clustering index based on the `user_id` and `product_id` columns:

- `created_at` as a TimeKey® timestamp that is not nullable.
- `id`, `user_id`, and `product_id` as integers that are not nullable.
- `subtotal`, `tax`, `total`, and `discount` as floating point numbers.
- `quantity` as an integer.

SQL

```sql
CREATE TABLE public.orders (
    created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
    id INT NOT NULL,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    subtotal DOUBLE,
    tax DOUBLE,
    total DOUBLE,
    discount DOUBLE,
    quantity INT,
    CLUSTERING INDEX idx01 (user_id, product_id)
);
```

The database creates the `orders` table, and you can begin loading data.

Step 3: Preview and Create a Data Pipeline

Create data pipelines using the `CREATE PIPELINE` SQL statement. To load data, you first create a pipeline with the definition of the source, data format, and transformation rules using a SQL-like declarative syntax. Then, you execute the `START PIPELINE` SQL statement to start the load. You can observe progress and status using system catalog tables and views.

Each Ocient pipeline defines a single data source and the target table or tables into which data loads. A data source includes the location of the source and filters on the source to define the specific data set to load. This tutorial loads data from two data sources, where each source is located in a directory on the same S3 bucket.

First, inspect the data that you plan to load. Each document has a JSON format similar to this example.

JSON

```json
{"id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total'
{"id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total
```

© 2026 Ocient, Inc.

Inspecting the file with Pandas, you can see this schema with details.

Parquet Schema Description

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          18760 non-null  int64
 1   user_id     18760 non-null  int64
 2   product_id  18760 non-null  int64
 3   subtotal    18760 non-null  float64
 4   tax         18760 non-null  float64
 5   total       18760 non-null  float64
 6   discount    1915 non-null   float64
 7   created_at  18760 non-null  datetime64[ns, UTC]
 8   quantity    18760 non-null  int64
```

In this case, Ocient automatically transforms the data to the target columns using some sensible conventions. In other cases, loads require some transformation. Most transformations are identical to functions that already exist in the SQL syntax of the Ocient System.

Prior to creating a pipeline, you can use the `PREVIEW PIPELINE` SQL statement to create your pipeline iteratively. This statement returns a result set that shows the final values that would be loaded but does not load the data into the target table.

Preview the `orders_pipeline` pipeline for the `orders` data set from your database connection prompt. Use the S3 data source with endpoint `https://s3.us-east-1.amazonaws.com`, bucket `ocient-docs`, and filter `metabase_samples/parquet/orders.parquet`. Specify the `parquet` format. Load the data into the `public.orders` table. The `SELECT` part of the SQL statement maps the fields in the Parquet file to the target columns in the created table. In this example, limit the result set to the first five records in the data source.

© 2026 Ocient, Inc.

SQL

```
PREVIEW PIPELINE orders_pipeline
SOURCE
    S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/parquet/orders.parquet'
    LIMIT 5
EXTRACT
    FORMAT parquet
INTO public.orders
SELECT
    $id as id,
    $user_id as user_id,
    $product_id as product_id,
    $subtotal as subtotal,
    $tax as tax,
    $total as total,
    $discount as discount,
    $created_at as created_at,
    $quantity as quantity;
```

Output

SQL

```
id          user_id     product_id subtotal              tax                   tota
-------------------------------------------------------------------------------
1           1           14         37.65                 2.07                  39.7
8           1           123        110.93                6.1                   117.
8           1           105        52.72                 2.9                   49.2
8           1           94         109.22                6.01                  115.
8           1           132        127.88                7.03                  134.

Fetched 5 rows
```

With this preview, you can confirm that the results of the pipeline match your requirements. If there is an issue, you can update the statement and run the `PREVIEW PIPELINE` statement again until it meets your needs.

Next, create the pipeline named `orders_pipeline` for the `orders` data set from your database connection prompt. The pipeline has three main sections:

- `SOURCE` — Loads data from S3. Set the S3 endpoint, bucket name, and filter for the Parquet files.
- `EXTRACT` — Sets the format to Parquet.
- `INTO ... SELECT` — Targets the `public.orders` table and selects the chosen fields from the Parquet records. In this case, all data is available at the top level of the Parquet records, so the example references the fields by the attribute name (e.g., `$id`, `$user_id`, etc.). For nested data, reference the nested fields using dot notation (e.g., `$order.user.first_name`). Each field maps to a target column using the `as` syntax.

SQL

```
CREATE PIPELINE orders_pipeline
SOURCE
    S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/parquet/orders.parquet'
EXTRACT
    FORMAT parquet
INTO public.orders
SELECT
    $id as id,
    $user_id as user_id,
    $product_id as product_id,
    $subtotal as subtotal,
    $tax as tax,
    $total as total,
    $discount as discount,
    $created_at as created_at,
    $quantity as quantity;
```

After you successfully create this pipeline, execute the `START PIPELINE` SQL statement to start the load.

© 2026 Ocient, Inc.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Observe the Load Progress

With your pipeline running, data begins to load immediately from the S3 files that you defined. If there are many files in each file group, the load process first sorts the files into batches, partitions them for parallel processing, and assigns them to Loader Nodes.

You can check the pipeline status and progress by querying the `information_schema.pipeline_status` system catalog table or executing the `SHOW PIPELINE_STATUS` SQL statement.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

SQL

```
database_name         pipeline_name       table_names          status      status
-------------------------------------------------------------------------------
metabase              orders_pipeline     ["public.orders"]    RUNNING     Starte
```

After the status of the pipeline changes to `COMPLETED`, all data is available in the target table.

After a few seconds, the data is available for query in the `public.orders` table.

SQL

```
SELECT COUNT(*) FROM public.orders;
```

Output

```
count(*)
--------------------
18760
```

You can drop the pipeline with the `DROP PIPELINE orders_pipeline;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

Data Pipelines Reference

Load JSON Data

Data Types for Data Pipelines

Transform Data in Data Pipelines

Manage Errors in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Pipeline Load of JSON Data from HDFS

The data pipeline functionality enables the loading of data from Apache® Hadoop® Distributed File System (HDFS). You can load various file types stored in HDFS into the Ocient® System.

The Ocient System uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a simple example of loading data in the JSON format.

HDFS Advanced Source Options

Besides the specified options in the **HDFS Source Options** section, you can also specify these options that are available for file system sources:

- COMPRESSION_METHOD
- START_FILENAME
- END_FILENAME
- START_CREATED_TIMESTAMP
- END_CREATED_TIMESTAMP
- START_MODIFIED_TIMESTAMP
- END_MODIFIED_TIMESTAMP
- SORT_BY
- SORT_DIRECTION
- SORT_REWRITE

For the `CONFIG` source option, the most useful properties are the ones with the `dfs.client` prefix. For the full property reference, see **HDFS Default** and **Core Default**. Only properties that affect client connections and read operations apply to data pipeline loading. This table describes some common properties.

© 2026 Ocient, Inc.

Property Name	Property Description
`dfs.client.use.datanode.hostname`	Set this property to `true` when you have datanodes and namenodes in the HDFS cluster configured to connect to each other using local IP addresses, but loading occurs externally and requires externally valid hostnames instead. The default value is `false`.
`dfs.client.retry.max.attempts`	Configure to change the number of times the loading process retries network requests to namenodes before throwing an error. You can increase this value when network connections are unstable. The default value is 10.
`dfs.client.socket-timeout`	Configure to change the timeout (in milliseconds) for all sockets used internally in HDFS loading. You can increase this value when network connections are unstable. The default value is 60000 (60 seconds).

HDFS Loading Example

Follow these steps to load JSON data from an HDFS source into the Ocient System.

Step 1: Create a New Database

Connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then, execute the `CREATE DATABASE` SQL statement to create the `geo` database.

SQL

```
CREATE DATABASE geo;
```

Step 2: Create a New Table in the Database

Create the `locations` table in the `public` schema to store location data with a name, zip code, and a point with latitude and longitude:

- `name` — Location name as a not nullable string
- `zipcode` — Zip code as an integer
- `location` — Location latitude and longitude as a point

SQL

```
CREATE TABLE public.locations (
    name VARCHAR(255) NOT NULL,
    zipcode INT NOT NULL,
    location POINT
);
```

Step 3: Preview and Create a Data Pipeline

Preview the `locations` data pipeline to load JSON data from HDFS. This data pipeline uses the HDFS endpoint `hdfs-namenode:9000`, which consists of the namenode and port number, and the `/locations/2026/**/*.json` filter to load data from JSON files. The pipeline selects the name, zip code, and point data. For the point data construction, see **ST_POINT**.

SQL

```
PREVIEW PIPELINE locations
  SOURCE hdfs
    ENDPOINT 'hdfs-namenode:9000'
    FILTER '/locations/2026/**/*.json'
  EXTRACT
    FORMAT json
  INTO locations
  SELECT
    $name AS name,
    $zipcode AS zipcode,
    ST_POINT($longitude, $latitude) AS location;
```

Create the `locations` data pipeline to load JSON data from HDFS.

SQL

```
CREATE PIPELINE locations
  SOURCE hdfs
    ENDPOINT 'hdfs-namenode:9000'
    FILTER '/locations/2026/**/*.json'
  EXTRACT
    FORMAT json
  INTO locations
  SELECT
    $name AS name,
    $zipcode AS zipcode,
    ST_POINT($longitude, $latitude) AS location;
```

After you successfully create the `locations` pipeline, execute the `START PIPELINE` SQL statement to start the data pipeline.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Observe the Load Progress

With your pipeline running, data begins to load immediately from the JSON files. If there are many files in each file group, the load process first sorts the files into batches, partitions them for parallel processing, and assigns them to Loader Nodes.

You can check the pipeline status and progress by querying the `information_schema.pipeline_status` system catalog table or executing the `SHOW PIPELINE_STATUS` SQL statement.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

```
database_name         pipeline_name       table_names              status      statu
-------------------------------------------------------------------------------------
geo                   locations           ["public.locations"]     RUNNING     Start
```

After the status of the pipeline changes to `COMPLETED`, all data is available in the target table.

After a few seconds, the data is available for query in the `public.locations` table.

```
SELECT COUNT(*) FROM public.locations;
```

Output

```
count(*)
--------------------
25605
```

You can drop the pipeline with the `DROP PIPELINE locations;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

Data Pipelines Reference

Load JSON Data

Data Types for Data Pipelines

Transform Data in Data Pipelines

Manage Errors in Data Pipelines

© 2026 Ocient All Rights Reserved

Data Types for Data Pipelines

Source Fields

When you load data using a data pipeline, you reference extracted data using a special syntax named a source field reference. These source field references can behave differently depending on the data format. It is important to understand how to reference source field data and also how Ocient® assigns the type of data you reference.

Source Field References

You reference source fields in a pipeline using a named reference such as `$my_field` or a numeric field reference such as `$1`. For some data formats like JSON and Apache® Parquet™, Ocient supports nested data access such as `$my_field.sub_field`.

Learn more about what source field references are supported for each data format in **Data Formats for Data Pipelines**.

Source Field Data Types

The Ocient System treats references to source fields (e.g., `$my_field`) differently depending on the data format `FORMAT` specified in the data pipeline.

Text-Based Data Formats

For text-based formats such as `DELIMITED` and `JSON`, the Ocient System treats source fields as `VARCHAR`. For `JSON` data, the system treats source fields strictly as `VARCHAR` data even if the field value is represented logically in JSON as Boolean, integer, NULL, or double. Learn more about text-based formats in **Load JSON Data** and **Load Delimited and CSV Data**.

Binary-Based Data Format

For the binary-based format `BINARY`, Ocient treats source fields as the `BINARY` data type. Learn more about binary formats in **Load Binary Data**.

Parquet-Based Data Format

© 2026 Ocient, Inc.

For the Parquet-based format `PARQUET`, Ocient encodes information about the data type of the fields during the load.

Supported Data Types

The data pipeline functionality supports all column data types.

> `ST_POINT` is not supported in a 3-coordinate format.

When you load data into a column type, in many cases, Ocient automatically casts the source data to the target column. However, in some cases, you might need to cast data to the target column type explicitly. Learn more about supported casting functions in **Scalar Transformation Functions and Casting**.

Automatic Conversion and Casting

Loading pipelines apply automatic casting to the final values of your transformations and source fields where the data type does not match the target column type. The casting function that the Ocient System applies varies based on the SQL type of the data that results from your transformation and the SQL type of the target column.

The system applies casting automatically in cases to resolve conflicts between data types:

- Promotion of a type to expand to a wider data type (e.g., `SMALLINT` to `BIGINT`)
- Implicit casting from one family of data type to another (e.g., `VARCHAR` to `TIMESTAMP`)
    - Ocient supports implicit casting of all simple data types to `VARCHAR`.

> Auto-casting in Parquet does not support the automatic conversion to `VARCHAR` columns. You must explicitly cast data to the `CHAR` data type when you convert Parquet data that is not string data to a `VARCHAR` column or `VARCHAR` function argument.

Automatic casting does not apply in some situations where data loss might occur:

- Implicit downcasting to a narrower data type (e.g., `BIGINT` to `SMALLINT`)

Where automatic casting is not supported, you can explicitly cast data or use built-in transformation functions according to your requirements. See **Transform Data in Data Pipelines** for more details.

Implicit Casting Support List

VARCHAR Source Data

The Ocient System treats the source field reference from `JSON` and `DELIMITED` formatted data as `VARCHAR` regardless of the logical type in the source.

Ocient supports the implicit casting of `VARCHAR` to all data types.

BINARY Source Data

The system treats the source field reference from `BINARY`-formatted data as raw bytes with automatic casting functions tailored to the `BINARY` data type.

Ocient supports implicit casting of `BINARY` to:

- `TINYINT`, `SMALLINT`, `INT`, `BIGINT`
- `FLOAT`, `DOUBLE`
- `TIMESTAMP`, `DATE`, `TIME` (as `VARCHAR` data)

> For `BINARY` or `VARCHAR` to `BOOL`:
>
> - You can represent `true` as: `'t', 'true', 'yes', '1', 'y', 'on'`
> - You can represent `false` as: `'f', 'false', 'no', '0', 'n', 'off'`

> For `BINARY` or `VARCHAR` to `TIMESTAMP`, `DATE`, and `TIME`, the automatic casting assumes that the character data has the format:
>
> - `TIMESTAMP` — `YYYY-MM-DD HH:mm:ss[.SSSSSSSSS]`
> - `DATE` — `YYYY-MM-DD`
> - `TIME` — `HH:mm:ss`

Other Implicit Casts or Downcasts

The system also supports some other implicit casts that are not type promotions:

- `TIMESTAMP` data implicitly downcasts to `DATE` and `TIME` columns, truncating the unused portion.
  - ℹ This option is available in version 25 or later.
- The system implicitly casts all simple types to `VARCHAR` columns except:
  - `HASH` or `BINARY`
  - `DECIMAL`
  - `IP` or `IPV4`

For geospatial data types, see **Load Geospatial Data in Data Pipelines**.

Data Type Promotion List

The data type promotion list describes how the Ocient System can automatically promote the specified type to a target column type. The system achieves the promotion using standard casting functions (e.g., `BIGINT(SMALLINT)`). See **Implicit Casting Support List** for implicit casting, for example casting to `VARCHAR`.

Source Data Type	Automatic Casting Precedence for Target Column Type
`BYTE` or `TINYINT`	`BYTE` `TINYINT` `SMALLINT` `INT` `BIGINT` `FLOAT` `DOUBLE`
`SMALLINT`	`SMALLINT` `INT` `BIGINT` `FLOAT` `DOUBLE`
`INT`	`INT` `BIGINT` `FLOAT` `DOUBLE`
`BIGINT`	`BIGINT` `DOUBLE` `TIMESTAMP` `TIME`
`FLOAT`	`FLOAT` `DOUBLE`
`DOUBLE`	`DOUBLE`
`DECIMAL`	`DECIMAL`
`BOOL`	`BOOLEAN`
`UUID`	`UUID`
`DATE`	`DATE` `TIMESTAMP`

Source Data Type	Automatic Casting Precedence for Target Column Type
`TIME`	`TIME`
`TIMESTAMP`	`TIMESTAMP` See **Implicit Casting Support List** for supported downcasting.
`IPV4`	`IPV4`
`IP`	`IP`
`BINARY`	`BINARY` See **Implicit Casting Support List** for supported cross-casting.
`ARRAY`	`ARRAY`
`TUPLE`	`TUPLE`
`VARCHAR`	`VARCHAR` See **Implicit Casting Support List** for supported cross-casting.
`ST_POINT`	`ST_POINT`, `ST_LINESTRING`, `ST_POLYGON`
`ST_LINESTRING`	`ST_LINESTRING`, `ST_POLYGON`
`ST_POLYGON`	`ST_POLYGON`

- `BIGINT` to `TIME` automatic casting assumes a time value in milliseconds.
- `BIGINT` to `TIMESTAMP` automatic casting assumes milliseconds after the epoch. (January 1, 1970 at midnight UTC)
- For complex types, the system applies the type precedence for automatic casting to the elements in the type.
- Automatic casting of integral types to DOUBLE and FLOAT are potentially lossy due to limits in precision of floating point representations.
  - ℹThis option is available in version 25 or later.

The system treats `BYTE` and `TINYINT` aliases identically.

Related Links

Load Data

Data Formats for Data Pipelines

Data Pipelines Reference

Load Metadata and File-Based Partitioned Data in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Formats for Data Pipelines

Loading in Ocient® differs in subtle ways that depend on the data format of the source. Loading runs with a strict interpretation of streaming source with Apache® Kafka® or source data to allow pipelines to achieve maximum performance. For text-based formats like `JSON` and `DELIMITED`, the Ocient System performs no preemptive casting on the data when using a source field selector. For the Kafka streaming source, use the Confluent® Schema Registry with subjects that contain a group of schema versions, which capture the schema changes over time.

For example, with the JSON string `{ "my_field": 1234 }`, the selector `$my_field` returns the string `"1234"` not the integer `1234`.

When you use transformation functions, keep in mind that the Ocient System treats all data in the `JSON` and `DELIMITED` formats as text data.

While the Ocient System sends data you select to a final target column, the system automatically casts the data in the final step to ensure that the data is compatible with the target column type. See **Data Types for Data Pipelines** for supported automatic conversion rules.

Format-specific differences also appear in the pipelines.

Load ASN.1 Data

You can load data in ASN.1 (Abstract Syntax Notation One) format from binary-encoded ASN.1 files using DER-encoded or BER-encoded files. ASN.1 provides a flexible, schema-driven format commonly used in telecommunications, security, and standardized protocols. This format allows you to extract structured records and map them to relational tables using SQL.

The Ocient System requires `.ber` and `.der` files to contain one or more concatenated DER-encoded or BER-encoded values with the specified record type. The system decodes each record independently and maps it into a record.

ASN.1 Type Mapping

The system automatically converts all decoded ASN.1 fields to their JSON-equivalent representations.

> The ASN.1 schema must consistently use implicit or explicit tagging. You must specify clear tagging so the system can resolve field names during extraction.

ASN.1 Type	JSON Mapping	Description
`INTEGER`	Number	The number, specified as an integer. Example: `5`
`BOOLEAN`	Boolean	The logical value, specified as a Boolean. Example: `true`
`VisibleString`	String	The ASN.1 type assumes UTF-8 encoding. Example: `"Data Analyst"`
`IA5String`	String	The International Alphabet number 5 (IA5) includes most of the ASCII alphabet but can contain other characters. Example: `"ABCD"`
`UTF8String`	String	The ASN.1 type assumes UTF-8 encoding. Example: `"abcd"`
`OCTET STRING`	Hexadecimal string	The hexadecimal string has the prefix `0x`. Example: `"0x4fa2"`
`BIT STRING`	Hexadecimal string	The hexadecimal string with preserved bit alignment has the prefix `0x`. Example: `"011110"`
`OBJECT IDENTIFIER`	String	The unique identifier for an object using dotted representation. Example: `"1.2.840.113549"`
`ENUMERATED`	Number or symbolic string	The enumerated string, if the data resolves into a list of named items. Example: `"red"`
`ANY`	Hexadecimal string	The hexadecimal string with raw encoded bytes of the embedded value.

© 2026 Ocient, Inc.

ASN.1 Type	JSON Mapping	Description
		Example: `"0x616263"`
`SEQUENCE`, `SET`	JSON object	Object with fields that are accessible by their component names. Example: `WeatherRecord ::= SEQUENCE { temperature INTEGER, humidity INTEGER, cloudy BOOLEAN }`
`SEQUENCE OF`, `SET OF`	JSON array	Array of objects where the elements are fully decoded. Example: `WeatherEvents ::= SEQUENCE OF WeatherRecord`
`CHOICE`	JSON object	An object with fields of any type. Identify the active alternative by the key. Example: `WeatherReport ::= CHOICE { sunny SEQUENCE { uv\_index INTEGER }, rainy SEQUENCE { precipitation INTEGER } }`
`DATE`	String	Date in the format `"yyyyMMdd"`. You must explicitly declare the date, or the system must infer it using tagging. Example: `"20240518"`
`TIME`	String	Time in the format `"HHmmss"`. Example: `"113056"`
`DATE-TIME`	String	Date and time in the format `"yyMMddHHmmss'Z'"`. This format supports only UTC. You must include `'Z'` as a literal. Example: `"250920101508'Z'"`

To access fields, use dot notation for access: `$sequence.fieldName`. For arrays (such as `SEQUENCE OF`), use bracket notation: `$sequenceOf[].fieldElement`.

Default values and optional fields follow standard ASN.1 rules. If you omit a field, the system evaluates it as `NULL`.

ASN.1 Loading Example

Assume you have the `personnel_records.asn` file in the ASN.1 data format. The file contains the definition of a personnel record `Example.PersonnelRecord`. The ASN.1 file contains data for the personnel record, child, personnel name, employee number of the personnel, and the date.

Text

```
Example DEFINITIONS IMPLICIT TAGS ::=
BEGIN
    PersonnelRecord ::= [APPLICATION 0] SET {
        name         [0] Name,
        title        [1] VisibleString,
        number       [2] EmployeeNumber,
        dateOfHire   [3] Date,
        nameOfSpouse [4] Name,
        children     [5] SEQUENCE OF ChildInformation DEFAULT {}
    }
    ChildInformation ::= SET {
        name        [0] Name,
        dateOfBirth [1] Date
    }
    Name ::= [APPLICATION 1] SEQUENCE {
        givenName  [0] VisibleString,
        initial    [1] VisibleString,
        familyName [2] VisibleString
    }
    EmployeeNumber ::= [APPLICATION 2] INTEGER
    Date ::= [APPLICATION 3] VisibleString
END
```

Create a table to contain the personnel record. The table contains a subset of the data:

- `first_name` — First name
- `initial` — Initial of the middle name
- `family_name` — Last name
- `title` — Job title
- `number` — Employee number
- `date_of_hire` — Hire date

© 2026 Ocient, Inc.

SQL

```sql
CREATE TABLE personnel_records(
    first_name VARCHAR NOT NULL,
    initial VARCHAR NOT NULL,
    family_name VARCHAR NOT NULL,
    title VARCHAR NOT NULL,
    number BIGINT NOT NULL,
    date_of_hire DATE NOT NULL
);
```

Create the data pipeline `personnel_pipeline` to load the personnel record into the `personnel_records` table using an Amazon® Web Services™ (AWS™) S3 bucket. Specify the bucket, endpoint, access key identifier, secret access key, and filter options to find the ASN.1 DER-encoded file `personnel_records.der` in the specified file path. Use the URL file path `http://cos/filepath/asn1/personnel_records.asn` and record type `Example.PersonnelRecord`. Access the `name` sequence using dot notation for the first name, middle initial, and last name fields. The pipeline definition transforms the hire date to the `'yyyyMMdd'` format.

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE personnel_pipeline
    SOURCE S3
        BUCKET 'misc'
        ENDPOINT 'http://cos'
        ACCESS_KEY_ID ''
        SECRET_ACCESS_KEY ''
        FILTER_GLOB 'user/asn1/personnel_records.der'
    EXTRACT
        FORMAT 'asn.1'
        SCHEMA {
            URL 'http://cos/filepath/asn1/personnel_records.asn'
            RECORD_TYPE 'Example.PersonnelRecord'
        }
INTO personnel_records
SELECT
    $name.givenName AS first_name,
    $name.initial as initial,
    $name.familyName as family_name,
    $title as title,
    $number as number,
    TO_DATE($dateOfHire, 'yyyyMMdd') as date_of_hire;
```

Load Avro Data

The Ocient System enables you to load data in the Apache® Avro™ format. You can use a streaming source with a file-based source only. Load an inline schema definition or use a schema configuration. Use a schema inference from files with embedded schemas.

Field selectors in Avro follow the same format as selectors in JSON and Apache® Parquet™ formats.

ⓘ The Ocient System treats Avro selectors as lowercase. To use case-sensitive selectors, you must enclose the selector in double quotation marks. For example, `$"testSelector"`.

Inline Schema

Specify a JSON string in the Avro schema format in the `INLINE` option of the schema definition of the `CREATE PIPELINE` SQL statement.

Inline schemas assume that all records follow the defined schema exactly. These schemas do not support schema evolution.

For Kafka messages, use the `SCHEMA_REGISTRY_ID_LOCATION` option to denote whether there is an embedded schema identifier. For inline schemas, the `'none'` value indicates that no embedded schema identifier is present.

Schema Inference from a File

The system can infer the schema from a file that has embedded schemas. The file is a named object container file. Use the `INFER_FROM` option in the `CREATE PIPELINE` SQL statement to specify sampling one file.

Schema Registry

For loading from a Kafka streaming source, use the schema configuration with the `SUBJECT` option in the `CREATE PIPELINE` SQL statement to specify the name of the subject for the data pipeline. The default value is `<topic name>-value`, where `<topic name>` is the name of the Kafka topic. The Ocient System follows the schema registry configuration from the Confluent platform.

Schema Evolution

When you create a data pipeline, the pipeline has a fixed target schema (specified by the `SELECT` clause). Individual files might have different schemas. The target schema must be backward-transitive compatible with the other schemas. Individual files or Kafka messages might have different schemas.

If the other schemas change, the system automatically attempts to fit data into the target schema. In this case, the other schemas must be forward compatible with the target schema. The system ignores changes to any unused fields from the target schema.

To manually change the schema, you must first stop the execution of the data pipeline by using the `STOP PIPELINE` SQL statement. Then, use the `ALTER PIPELINE` SQL statement

to modify the schema. You can add a field, remove a field, or change the precision of a data type. For details, see the **ALTER PIPELINE** SQL statement.

> Multiple schemas impact the performance of the data pipeline execution. For best performance, use a single schema for all data.

Avro Type Mapping

The Ocient System converts these Avro data types to Ocient SQL types. This table shows the respective conversions.

Avro Type	Ocient SQL Type
`null`	`NULL`
`boolean`	`BOOLEAN`
`int`	`INT`
`long`	`BIGINT`
`float`	`FLOAT`
`double`	`DOUBLE`
`bytes`	`VARBINARY`
`string`	`VARCHAR`
`record`	Select subfields of the record using the standard pipeline selector syntax, e.g., `$myrecord.subfield`.
`enum`	`VARCHAR`
`array`	`ARRAY`
`map`	Select entries of the map using the key with the standard pipeline selector syntax, e.g., `$mymap.key`.
`fixed`	`HASH`
`decimal`	`DECIMAL`
`big-decimal`	`DECIMAL`
`uuid`	`UUID`
`date`	`DATE`

Avro Type	Ocient SQL Type
`time-*`	`TIME`
`timestamp-*`	`TIMESTAMP`
`local-timestamp-*`	`TIMESTAMP`
`duration`	The system converts duration to a record with integer subfields for `months`, `days`, and `milliseconds`.
`union`	The system converts a union between `NULL` and another type to a nullable version of that type. In contrast, the system converts a union containing multiple non-`NULL` types to a string.

Avro Loading Examples

Create the `users` table with these columns:

- `id` — Universally Unique IDentifier (UUID) of the user
- `firstname` — First name of the user
- `lastname` — Last name of the user
- `birthyear` — Year of birth
- `groups` — List of groups where the user belongs

SQL

```
CREATE TABLE users(
    id          UUID NOT NULL,
    firstname   VARCHAR(255) NOT NULL,
    lastname    VARCHAR(255) NOT NULL,
    birthyear   INT,
    groups      VARCHAR(255)[] NOT NULL DEFAULT 'char[]'
);
```

These examples use this table as the target table for the load.

Load Avro Data from Files

Assume you have user data in Avro format in multiple files in the `/data/users` directory. Create the `users_pipeline` data pipeline for the Avro files `*.avro` containing user data.

The schema configuration instructs the system to infer from one file using the `INFER_FROM` option. Access the array of strings for the `groups` column.

SQL

```
CREATE PIPELINE users_pipeline
SOURCE filesystem
    FILTER '/data/users/*.avro'
EXTRACT
    FORMAT avro
    SCHEMA {
        INFER_FROM 'sample_file'
    }
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    $birthyear  AS birthyear,
    $groups[]   AS groups;
```

Load Avro Data from an Inline Schema Definition

Assume you have user data in Avro format in multiple files in the `/data/users` directory. Create the `users_pipeline` data pipeline for the Avro files `*.avro` containing user data. The schema configuration instructs the system to use an inline schema definition with the `INLINE` option. Access the array of strings for the `groups` column.

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE users_pipeline
SOURCE filesystem
    FILTER '/data/users/*.avro'
EXTRACT
    FORMAT avro
    SCHEMA {
        INLINE '{
            "type": "record",
            "name": "User",
            "namespace": "test.users",
            "fields": [
                { "name": "id",        "type": { "type": "string", "logicalType": "
                { "name": "firstname", "type": "string" },
                { "name": "lastname",  "type": "string" },
                { "name": "birthyear", "type": ["null","int"], "default": null },
                { "name": "groups",    "type": { "type": "array", "items": "string"
            ]
        }'
    }
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    $birthyear  AS birthyear,
    $groups[]   AS groups;
```

Load Avro Data Using a Kafka Schema Registry Configuration

Assume you have user data in Avro format in a Kafka topic. Create the `users_pipeline` data pipeline for the user data. The schema registry configuration instructs the system to access the subject `'users-value'` at the registry location `'value'` using the registry URL `'https://schema-registry.company.com'`. The configuration specifies the access credentials using the `CONFIG` option:

- Credentials source
- User authentication that includes the username and password
- Location of the truststore

- Password for the truststore

Access the array of strings for the `groups` column.

SQL

```
CREATE PIPELINE users_pipeline
SOURCE
    KAFKA
        BOOTSTRAP_SERVERS '192.168.0.1:9092'
        TOPIC 'users'
EXTRACT
    FORMAT avro
    SCHEMA {
        SUBJECT 'users-value'
        SCHEMA_REGISTRY_ID_LOCATION 'value'
        URL 'https://schema-registry.company.com'
        CONFIG '{
            "basic.auth.credentials.source": "USER_INFO",
            "basic.auth.user.info": "sr_user:sr_password",
            "ssl.truststore.location": "/etc/ssl/truststore.jks",
            "ssl.truststore.password": "testpassword"
        }'
    }
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    $birthyear  AS birthyear,
    $groups[]   AS groups;
```

For details about the schema registry configuration, see the **Confluent Documentation**.

Load Binary Data

The Ocient System loads the binary data format using a fixed record length to split a binary stream into chunks that represent records. Each record is available in the `SELECT` portion of a pipeline definition using a special binary extract syntax `$"[5,8]"`. This operates similarly to a substring function, beginning at byte `5` and taking `8` bytes from that location. The starting index is a 1-based offset, consistent with other SQL arrays and

offsets. You can use this syntax to select specific bytes within a record to parse together as a unit.

Binary Selector

Selector	Description	Examples
`$"[<start_index>, <length_in_bytes>]"`	**Extract Bytes** — Extracts the bytes beginning at the `<start_index>` and returning `<length_in_bytes>` total bytes to extract from the start index. `<start_index>` is 1-based.	`abc123.45xyz` `$"[4,6]" -> "123.45"` This example illustrates the operation using ascii characters, but the Binary Selector returns BINARY data, byte arrays, that may be used in special ways by fixed width processing functions described in **Binary Data Transformations**.

Example

The Binary Selector takes 8 bytes starting at offset 11 on the fixed-width binary record. Consistent with SQL functions in the Ocient System, the first argument value `11` is the 1-based offset into the byte array.

SQL

```
$"[11, 8]"
```

The Binary Selector returns `BINARY` data, not `VARCHAR`.

Special `BINARY` transformation functions can operate on this `BINARY` data. However, if you cast data to the `VARCHAR` type by using `CHAR()`, then functions like `INT` operate on this data as `VARCHAR` data, not binary data.

When you load binary data into `VARCHAR` columns, the Ocient System automatically converts from binary to character data using the configured `CHARSET_NAME` before final loading.

The Ocient System supports special transformation functions that operate uniquely on binary data. With these functions, you can convert binary representations from mainframe systems such as packed decimals, zoned decimals, big and little endian

integers (signed and unsigned), and floating point values. For more details, see **Binary Data Transformation Functions**.

For a complete list of supported options for `DELIMITED` and `CSV` data formats, see **Binary Extract Options**.

Binary Loading Example

If each record in your fixed-width binary schema includes these fields, you can use the `SUBSTRING` function and the transforms shown in this example.

Column	Start Index	Record Length	Source Data Type	Ocient Target Data Type
first_name	1	20	Character	VARCHAR
last_name	21	20	Character	VARCHAR
age	41	4	Big Endian signed 4-byte integer	INT
total_spent	45	10	Packed Decimal	DECIMAL(10,2)
user_id	55	8	Little Endian unsigned 8-byte long	BIGINT

Each record includes 62 bytes, so the record length `RECORD_LENGTH` is 62. The encoding of this file is CP500 instead of the default IBM1047 code page. The `CREATE PIPELINE` SQL statement specifies this encoding.

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE binary_users_pipeline
SOURCE
    S3 ...
FORMAT
    BINARY
    RECORD_LENGTH 62
    CHARSET_NAME 'cp500'
INTO public.users
SELECT
    $"[1, 20]" as first_name,
    $"[21, 20]" as last_name,
    INT($"[41, 4]") as age,
    DECIMAL($"[45, 10]", 'packed', 2) as total_spent,
    BIGINT($"[55, 8]", 'unsigned', 'little') as user_id;
```

This SQL statement:

- Uses the `BINARY SELECTOR` to extract names and load them into the respective columns. The Ocient System automatically decodes the values using `cp500` and loads them into a `VARCHAR` column. An explicit cast such as `CHAR($"[1, 20]") as first_name` works equivalently.
- Indicates the extraction of four bytes that represent age from bytes 41-44. The statement instructs the casting of these bytes as an integer `INT`. This function uses the default endianness (big) and treats the bytes as signed. Unsigned values can overflow target columns because integral types are all signed.
- Extracts the 10 bytes for `total_spent` using the `BINARY SELECTOR`, and converts the values using the `packed` decimal option for the `DECIMAL` cast. The casting requires specifying the number of decimal points in the source data. In this case, there are 2 decimal points, which match the number in the target column.
- Extracts the 8 bytes that represent `user_id` using the `BINARY SELECTOR` and casts these bytes to a `BIGINT` while interpreting the bytes as unsigned with the little endian representation.

Load Delimited and CSV Data

When you load data from delimited or CSV files, the Ocient System tokenizes the data during loading. The system detects records and fields in the input data during pipeline execution. You can reference fields and use them in combination with transformation functions before the system stores values in the column of a target table.

Files must be located in these allowed directories:

- `/tmp` directory
- The temporary directory you configure using the `streamloader.extractorEngineParameters.tempDir` configuration option (default path is `/var/opt/ocient/tmp`).
- The directory list specified by the `streamloader.extractorEngineParameters.configurationOption.filesystem.access.directories` configuration option (default is an empty list).

Files cannot be located in these blocked directories:

- `/etc`
- `/bin`
- `/sbin`
- `/lib`
- `/lib64`
- `/usr`
- `/boot`
- `/proc`
- `/sys`
- `/run`
- `/root`
- `/var/lib`
- `/var/log`
- `/var/run`
- `/var/cache`

Referencing fields of the source data for the formats happens by using a field index. The index is a number that follows the dollar sign `$`. To maintain consistency with SQL array semantics, the field indexes start at 1.

Reference the first field of tokenized source records for the `DELIMITED` and `CSV` formats as `$1`.

> ⚠ For the `BINARY` format, `$0` represents the entire record. In this case, you must specify `$0` in combination with the `SUBSTRING` function to extract specific bytes from the source data.

For a complete list of supported options for `DELIMITED` and `CSV` data formats, see **Delimited and CSV Extract Options**.

Delimited Loading Example

Use this example delimited data.

Text

```
iphone|60607|viewed|502|293.99|[shopping,news]
```

For this example row, this table shows the field references for each value in the row.

Field Reference	Value
$1	iphone
$2	60607
$3	viewed
$4	502
$5	293.99
$6\[]	\[‘shopping’,‘news’]

To load this data in a pipeline with the `DELIMITED` data format, this `CREATE PIPELINE` statement specifies the `|` character for the field delimiter. This statement loads data into AWS S3. The `SELECT` statement uses fields 1, 2, 3, 5, and 6 of the source data. The

statement specifies that the system should not load field 4 to the target table. Field 6 is an array of data matching the default array settings for delimited data. You can indicate this with the array brackets like `$6[]` to load into a `CHAR[]` typed column.

The outer casting functions in this example are optional and shown for completeness. If they are omitted, the pipeline automatically casts the source fields to the target column type.

SQL

```
CREATE PIPELINE delimited_pipeline
SOURCE s3
    ...
FORMAT delimited
    FIELD_DELIMITERS ['|']
    ...
SELECT
    CHAR($1) as device_model,
    INT($2) as zip,
    INT($3) as amount,
    DOUBLE($5) as price,
    CHAR[]($6[]) as categories;
```

Load JSON Data

The data pipeline syntax enables the load of JSON data, including nested scalars, arrays, and points `ST_POINT`.

Strict Loading and Transformations

When you use transformation functions, remember that the Ocient System treats all data in `JSON` and `DELIMITED` format as text data, not the `logical` data type.

For example, if you specify the JSON string `{ "my_timestamp": 1709208000000 }`, the selector `$my_timestamp` returns the string `"1709208000000"` and not the integer `1709208000000`.

As a result, if you cast this data into a timestamp column, such as `TIMESTAMP($my_timestamp) as created_at`, the Ocient System returns an error. The conversion fails because the cast function assumes you are specifying `TIMESTAMP(VARCHAR)`, which assumes a format like `YYYY-MM-DD HH▯▯ss[.SSSSSSSSS]`.

To correct this issue, cast the value explicitly to make use of the `TIMESTAMP(BIGINT)` function that treats the argument as milliseconds after the epoch as in `TIMESTAMP(BIGINT($my_timestamp)) as created_at`.

Supported JSON Selectors

JSON selectors consist of `$` followed by a dot-separated list of JSON keys. If a key refers to an array, it is followed by a set of brackets `[]` to correspond to its dimensionality. If the square brackets contain an index, like `[1]`, then the selector refers to an array element.

The Ocient System treats JSON selectors as lowercase. To use case-sensitive selectors, you must enclose the selector in double quotation marks. For example, `$"testSelector"`. With case-sensitive selectors having multiple JSON keys, each key needs double quotation marks. For example, `$"testData"."Responses"."SuccessResponse"`.

For special characters (any identifier that starts with any character other than a letter or contains any character that is not a letter, number, or an underscore) or reserved SQL keywords (such as `SELECT`), you must enclose such selectors in double quotation marks. For example, if you have a JSON document `{ "test-field": 123 }`, then the selector for the query should be `$"test-field"`.

The Ocient System does not support identifiers with a backslash as the last character in the key name.

This table shows the selector and provides its description. The cells in the last column of the table show an example for each selector. First, the cell shows example data in JSON format. Then, the cell shows the use of the selector and its output after the arrow.

Selector	Description	Examples
`$a`	**Scalar** — Refers to key `a` at the top of a JSON document, where the value of `a` is a scalar.	`{ "first_name": "Chris" }` `$first_name -> "Chris"`
`$a[]`	**Array** — Refers to key `a` at the top of a JSON document, where the value of `a` is a one-dimensional array.	`{ "names": ["Chris", "Joe"] }` `$names[] -> ["Chris","Joe"]`
`$a[1]`	**Array Element Selection** — Selects a single element from the array `a` at the top of a JSON document, where the value of `a` is a one-dimensional array. Also works on multi-dimensional arrays. Indexes start at 1.	`{ "names": ["Chris", "Joe"] }` `$names[2] -> "Joe"`
`$a[][]`	**Multi-dimensional Array** — Refers to key `a` at the top of a JSON document, where the value of `a` is a two-dimensional array.	`{ "coordinates": [ [87.31, 18.25],[65.22, 19.41]] }` `$coordinates[][] -> [[87.31,18.25],[65.22,19.41]]`
`$a.b.c`	**Nested Selector** — Drills into the value at key `a`, then value at key `b`, and refers to the value at key `c`, where none of the values are arrays.	`{ "user": { "name": "Chris" } }` `$user.name -> "Chris"`

`$a[].b`	**Array Projection** — A projection applies a JSON selector on all elements in an array, returning an array.
	Array brackets in the notation indicate which items are the arrays.
	The `a[]` is an array. The `b` selector is applied to each element in the array `a`.
		`{`
		`"users": [`
		`{`
		`"name": "Chris",`
		`"orders": [`
		`{ "order_id": 1,`
		`"subtotal": 19.29 },`
		`{ "order_id": 2,`
		`"subtotal": 16.71 }`
		`]`
		`},`
		`{`
		`"name": "Joe",`
		`"orders": [`
		`{ "order_id": 3,`
		`"subtotal": 17.29 },`
		`{ "order_id": 4,`
		`"subtotal": 22.49 }`
		`]`
		`}`
		`]`
		`}`
		`$users[].name -> ["Chris", "Joe"]`

`$a.b[].c.d[]`	**Multi-level Array Projection** — Applies selectors to each element in the specified	`{` `"users": [`

Selector	Description	Examples

	arrays and allows multiple levels of objects and arrays for the selection of an N-dimensional array of values.
	The values of `b` and `d` are arrays, resulting in two-dimensional arrays. The selector `c.d[]` is projected into the elements in the array `b`.
		`{`
		`"name": "Chris",`
		`"orders": [`
		`{ "order_id": 1,`
		`"subtotal": 19.29 },`
		`{ "order_id": 2,`
		`"subtotal": 16.71 }`
		`]`
		`},`
		`{`
		`"name": "Joe",`
		`"orders": [`
		`{ "order_id": 3,`
		`"subtotal": 17.29 },`
		`{ "order_id": 4,`
		`"subtotal": 22.49 }`
		`]`
		`}`
		`]`
		`}`
		`$users[].orders[].order_id ->`
		`[[1,2],[3,4]]`

`$a.b[1].c.d[]`	**Array Slice** — Selects the first element from the array `b` and drills into `c` and then the array `d`. JSON array selectors have an index that starts at 0.	`{`
		`"users": [`
		`{`
		`"name": "Chris",`
		`"orders": [`
		`{ "order_id": 1,`
		`"subtotal": 19.29 },`
		`{ "order_id": 2,`
		`"subtotal": 16.71 }`
		`]`
		`},`
		`{`
		`"name": "Joe",`
		`"orders": [`
		`{ "order_id": 3,`
		`"subtotal": 17.29 },`
		`{ "order_id": 4,`
		`"subtotal": 22.49 }`
		`]`
		`}`
		`]`
		`}`
		`$users[1].orders[].subtotal ->`
		`[17.29,22.49]`

`$a.{b,c}`	**Tuple Selector** — Selects the values from an object as a tuple. You must cast the values to an appropriate `TUPLE` data type.	`{ "user": { "first_name":` `"John", "last_name": "Doe" }}`

Selector	Description	Examples
		`$user.{first_name,last_name} -> <<"John","Doe">>`
`$a[].{b,c}`	**Array of Tuples Selector** — Selects the values from an array of objects as an array of tuples. You must cast the values to an appropriate `TUPLE[]` data type.	`{ "users": [ { "first_name": "John", "last_name": "Doe" }, { "first_name": "Steve", "last_name": "Smith" } ]}` `$users[].{first_name,last_name} -> [<<"John","Doe">>, <<"Steve","Smith">>]`
`$a[_]`	**Flatten Array Selector** — Selects the values of an array and flattens the dimension indicated by the underscore by one level.	`{ a: { b: [ {c: [1,2,3] }, {c: [4,5,6] } ] } }` `$a.b[_].c -> [1,2,3,4,5,6]`
`$a[!]`	**Compact Array Selector** — Selects the values of an array and removes any NULL values from the array at this level in the array.	`{ a: { b: { c: [1,null,2,null,3] } } }` `$a.b.c[!] -> [1,2,3]`

For more examples of using JSON selectors in data pipelines, see **JSON Selectors Examples in Data Pipelines**.

NULL and Empty Handling for JSON Scalars

The Ocient System handles all JSON NULL, empty, and missing values in the same way. The system loads these values as `NULL`. These values fail to load into non-nullable columns.

Provide an explicit default in the pipeline using `IF_NULL` or `COALESCE` or use the `COLUMN_DEFAULT_IF_NULL` option to accept the configured column default instead of attempting to load `NULL` values.

© 2026 Ocient, Inc.

NULL and Empty Handling for JSON Arrays

The Ocient System handles NULL, empty, and missing values the same way for arrays as for scalars. The system converts a value that is NULL, empty, or missing to NULL and loads it as `NULL`.

Provide an explicit default in the pipeline or use the `COLUMN_DEFAULT_IF_NULL` option to accept the configured column default instead of attempting to load `NULL`.

NULL and Empty Handling for JSON Tuples

All the rules for handling NULL, empty, and missing elements that apply to scalars and arrays also apply to tuples. If any part of the selector is NULL, empty, or missing, data pipeline loading converts that value to `NULL`.

Additionally, because you can apply functions to tuple elements (and not array elements), you can use the `NULL_IF` function to convert a tuple element to `NULL`. For example, `tuple<<char,varchar>>($a.name, NULL_IF($a.hometown, 'N/A') )` indicates to the pipeline that the string `'N/A'` signifies `NULL` for the `hometown` element but not for the `name` element.

Load Parquet Data

The data pipeline functionality enables loading Parquet files with this configuration.

File Configuration

- Files should have row groups of less than 128 MB. Larger row groups can impact memory usage during loading, and row groups of 512 MB can cause loading failures on 1 TB or more data sets.
- Encoding fields in a Parquet file reduces the space of the file on disk but can impact memory usage during loading. Enable encoding on fields that you expect to have less than 256 unique values and for fields that contain short strings. You do not have to encode other fields.

Multiple Files

- You can load row groups of multiple Parquet files in parallel. For large data sets, load the data set as multiple files.

- Loading files with differing schemas is not supported.

Use selectors as you do when loading JSON data to specify data to load.

> You must select a leaf element, an array, or a tuple with your selector. This is stricter than using JSON selectors, which can directly select array fields and JSON object fields.
>
> Example: `{"a": [1,2,3], "b": {"c": 1}}`
>
> You can extract with any of the selectors in JSON: `$a, $a[], $b, $b.c`
>
> However, Parquet only allows for the selectors: `$a[], $b.c`
>
> This example assumes this schema:
>
> ```
> // List<String> (list non-null, elements nullable)
> required group my_list (LIST) {
>   repeated group list {
>     optional binary element (UTF8);
>   }
> }
> ```
>
> The selector must be `$my_list[]`, which includes the array syntax.

For details, see **Parquet Selectors Examples in Data Pipelines**.

> When you use the `FORMAT PARQUET` option with an AWS S3 source, the `ENDPOINT` option is required in the `CREATE PIPELINE` SQL statement.

> Auto-casting in Parquet does not support the automatic conversion to `VARCHAR` columns. You must explicitly cast data to the `CHAR` data type when you convert Parquet data that is not string data to a `VARCHAR` column or `VARCHAR` function argument.

> The Ocient System treats Parquet selectors as lowercase. To use case-sensitive selectors, you must enclose the selector in double quotation marks. For example, `$"testSelector"`.

Schema Evolution

The Ocient System supports schema evolution when you load a set of Parquet files. Specifically, if the pipeline selects a set of Parquet files where an individual file might

have more or fewer columns than another, the system attempts to merge those schemas together to support loading without requiring you to create the pipeline again.

For example, the `test_table` table has three columns.

SQL

```
CREATE TABLE test_table (
  col_a INT NULL,
  col_b VARCHAR NULL,
  col_c VARCHAR NULL
);
```

You have two Parquet files with these schemas:

Text

```
message file1_schema {
  OPTIONAL INT32 col_a;
  OPTIONAL BYTE_ARRAY col_b (UTF8);
}

message file2_schema {
  OPTIONAL BYTE_ARRAY col_b (UTF8);
  OPTIONAL BYTE_ARRAY col_c (UTF8);
}
```

However, you must specify how to handle the schema evolution within the `EXTRACT` SQL statement. You can choose to sample the first file only for its schema or sample the entire data set to merge the schemas together. This DDL statement samples on one file.

SQL

```
EXTRACT
  FORMAT parquet
  SCHEMA (INFER_FROM sample_file)
```

The disadvantage is that sampling multiple files can potentially take a long time (scaling with the number of files in the data set) when you execute the `CREATE PIPELINE` and

`START PIPELINE` SQL statements. If you know that all of the Parquet files have the same schema, use this syntax.

The Ocient System does not support the case where a column within the schema changes type. For example, if `col_a` is an `INT` type in one file and a `VARCHAR` type in another.

The default behavior of schema evolution infers the schema from one file. Use this syntax to infer from one file.

```
SQL

EXTRACT
  FORMAT parquet
  SCHEMA (INFER_FROM sample_file)
```

Parquet Type Mapping

Parquet data types are separated into primitive and logical types. The Ocient System converts these types to Ocient SQL types. See these tables for the respective conversions.

Parquet Primitive Type	Ocient SQL Type
`BOOLEAN`	`BOOLEAN`
`INT32`	`INT`
`INT64`	`BIGINT`
`INT96`	`TIMESTAMP`
`FLOAT`	`FLOAT`
`DOUBLE`	`DOUBLE`
`BYTE_ARRAY`	`VARCHAR`
`FIXED_LEN_BYTE_ARRAY`	`VARCHAR`

Parquet Logical Types	Ocient SQL Type
`STRING`	`VARCHAR`
`UTF8`	`VARCHAR`

Parquet Logical Types	Ocient SQL Type
`ENUM`	`VARCHAR`
`UUID`	`UUID`
`INT8`	`TINYINT`
`INT16`	`SMALLINT`
`INT32`	`INT`
`INT64`	`BIGINT`
`UINT8`	`SMALLINT`
`UINT16`	`INT`
`UINT32`	`BIGINT`
`UINT64`	`BIGINT`
`DECIMAL`	`DECIMAL`
`DATE`	`DATE`
`TIME`	`TIME`
`TIME_MILLIS`	`TIME`
`TIME_MICROS`	`TIME`
`TIMESTAMP`	`TIMESTAMP`
`TIMESTAMP_MILLIS`	`TIMESTAMP`
`TIMESTAMP_MICROS`	`TIMESTAMP`
`DURATION`	`BIGINT`
`JSON`	`VARCHAR`
`BSON`	`VARCHAR`

- The `INTERVAL` data type is not supported.
- The `UINT64` data type can overflow the `BIGINT` conversion.
- The `DURATION` data type conversion to `BIGINT` preserves the underlying units. For example, the number of microseconds stays as microseconds in the `BIGINT` data type.

Further, Parquet contains nested types that the Ocient System also converts to SQL types, as shown in this table.

Parquet Nested Types	Ocient SQL Type
`LIST`	`TYPE[]`
`TUPLE`	`TUPLE`

Parquet Loading Example

Create a data pipeline that loads Parquet files using an AWS S3 bucket. Specify the bucket, endpoint, access key identifier, secret access key, and filter options to find all Parquet files in the specified file path. Use the `parquet_base_table` table to store the loaded data. Retrieve the integer, text, floating point, double, integer, JSON, and BSON fields.

SQL

```
CREATE PIPELINE testpipeline
    SOURCE S3
        BUCKET 'testbucket'
        ENDPOINT 'https://endpoint'
        ACCESS_KEY_ID ''
        SECRET_ACCESS_KEY ''
        FILTER_GLOB '/data/*/2024/*/*.parquet'
        PREFIX '/data/orders/2024/11/'
    EXTRACT
        FORMAT PARQUET
INTO parquet_base_table
SELECT $int32_field AS int32_field,
    $utf8_field as utf8_field,
    $float_field as float_field,
    $double_field as double_field,
    $int64_field as int64_field,
    $json_field as json_field,
    $bson_field as bson_field;
```

Parquet File-Partitioned Data

With Parquet, you can load file-partitioned data from Parquet files using the file path structure. Use the filter set in the file path using the Apache® Hive™ naming standards.

Assume files with these file paths.

Shell

```
s3://data/orders/2024/11/dt=2024-11-24/file.parquet
.../dt=2024-11-25/file.parquet
.../dt=2024-11-26/file.parquet
...
```

Load the data values in the Parquet file partitions using the `METADATA` function with the Hive partition syntax and the specified partition key `dt` from the file paths.

SQL

```
SELECT int32_field,
    utf8_field,
    float_field,
    double_field,
    int64_field,
    json_field,
    bson_field,
    METADATA('hive_partition','dt') AS file_date;
```

For details about this syntax, see **Load Metadata and File-Based Partitioned Data in Data Pipelines**.

Load XML Data

You can load data in XML format into the Ocient System. The system supports XML tags, basic elements, nested elements, and CDATA, but it does not support XML arrays and attributes.

Supported XML Selectors

Like JSON selectors, you can select XML data using the `$` symbol followed by a list of dot-separated JSON keys.

The system treats JSON selectors as lowercase. To use case-sensitive selectors, you must enclose the selector in double quotation marks, such as `$"testSelector"`. With case-sensitive selectors having multiple JSON keys, each key needs double quotation marks, such as `$"testData"."Responses"."SuccessResponse"`.

The system does not support JSON array and tuple selectors.

XML Loading Example

Assume an XML file with this data.

XML

```xml
<root>
  <person>
    <name>Barbara Smith</name>
    <address>
      <city>New York</city>
      <zip>12345</zip>
    </address>
    <note><![CDATA[Personal IP <b>127.0.0.1</b>]]></note>
  </person>
</root>
```

Create a table to contain the IP address record:

- `name` — Name
- `city` — Name of the city
- `zip` — Zip code
- `personal_ip` — IP address

SQL

```sql
CREATE TABLE example_xml(
    name VARCHAR NOT NULL,
    city VARCHAR NOT NULL,
    zip VARCHAR NOT NULL,
    personal_ip IPV4 NOT NULL
);
```

© 2026 Ocient, Inc.

Create a data pipeline `xml_pipeline` that loads the XML file `test.xml` using an AWS S3 bucket. Specify the bucket, endpoint, access key identifier, secret access key, and filter options to find the XML file in the specified path. Use the `example_xml` table to store the loaded data. Use JSON selectors to parse the file and data in each XML tag.

The system parses the `CDATA` section in the `note` tag as the literal string `Personal IP <b>127.0.0.1</b>`. Use the `SUBSTRING` function to extract the IP address and then transform it into an IPV4 type with the `IPV4` function. For details, see the **SUBSTRING** and **IPV4** functions.

SQL

```
CREATE PIPELINE xml_pipeline
    SOURCE S3
        BUCKET 'testbucket'
        ENDPOINT 'https://endpoint'
        ACCESS_KEY_ID ''
        SECRET_ACCESS_KEY ''
        FILTER_GLOB '/data/text.xml'
    EXTRACT
        FORMAT XML
INTO example_xml
SELECT
    $"root"."person"."name" as name,
    $"root"."address"."city" as city,
    $"root"."address"."zip" as zip,
    IPV4(SUBSTRING($"root"."person"."note", 16, 9)) as personal_ip;
```

Related Links

JSON Selectors Examples in Data Pipelines

Data Pipelines Reference

Load Metadata and File-Based Partitioned Data in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

JSON Selectors Examples in Data Pipelines

The data pipeline functionality enables you to load data in the structured JSON data format. Use these examples to see how you can access the structure in different ways. For an overview of loading JSON data, see:

- **Loading JSON Data**
- **Data Pipeline Load of JSON Data from Kafka**

Scalar Extraction from JSON

These examples load a scalar into the `VARCHAR` column named `customer_name`. Use the `CREATE TABLE` SQL statement to create the table with this column.

SQL

```sql
CREATE TABLE my_schema.my_table (customer_name VARCHAR NOT NULL);
```

For a JSON file with the value at the top level, the selector can refer to the key directly.

Load a file with this data.

JSON

```json
{"name": "John"}
```

Use the selector `$name`.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $name AS customer_name;
```

List all top-level JSON keys used in the `SELECT` statement in `JSON_FIELDS` of the `CREATE PIPELINE` SQL statement.

For a JSON file that has a nested value, you need a complex selector.

JSON

```
{"a": {"b": {"name": "John"}}}
```

In this case, the same `CREATE PIPELINE` statement for this file has `$a.b.name` instead of `$name`.

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a.b.name AS customer_name;
```

NULL and Empty Handling for JSON Scalars

The Ocient® System handles all JSON NULL, empty, and missing values in the same way. The system loads these values as `NULL`. These values fail to load into non-

nullable columns. Provide an explicit default in the pipeline or use the `COLUMN_DEFAULT_IF_NULL` option to accept the configured column default instead of attempting to load `NULL` values.

The JSON data contains this information.

JSON

```
{"a": {"b": {"name": "John"}}}
{"blah": {}}
```

In this case, the output table has two rows because there are two records in the JSON source file.

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a.b.name AS customer_name;
```

Output

SQL

```
Ocient> select * from my_schema.my_table;
name
--------------------------------------------
John
NULL
Fetched 2 rows
```

When you load all these JSON files with the same pipeline definition, the Ocient System creates a table with the same two rows.

The middle value is empty.

JSON

```
{"a": {"b": {"name": "John"}}}
{"a": {"b": {}}}
```

The middle value is NULL.

JSON

```
{"a": {"b": {"name": "John"}}}
{"a": {"b": null}
```

The leaf value is NULL.

JSON

```
{"a": {"b": {"name": "John"}}}
{"a": {"b": {"name": null}}}
```

Array Extraction from JSON

Supported array extraction scenarios are:

- One-dimensional arrays.
- Multi-dimensional arrays.
- Array projection is the nested application of JSON selectors over the elements of arrays for both one-dimensional and multidimensional arrays.
- Arbitrary nesting of arrays within JSON objects, including splitting array dimensions across the JSON path.
- Loading individual array elements, which can also be arrays.

⚠ Mapping a transformation function over array elements is not supported.

© 2026 Ocient, Inc.

One-Dimensional Arrays

The next few examples load the same data into the `my_table` table both directly and using array projection.

SQL

```
CREATE TABLE my_schema.my_table (customer_names VARCHAR[] NOT NULL);
```

Load data directly from a JSON array of strings. The JSON data contains this information.

JSON

```
{
  "a": {
    "b": {
      "names": ["John", "Bob", "Rajiv"]
    }
  }
}
```

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a.b.names[] AS customer_names;
```

Output

SQL

```
Ocient> select * from my_schema.my_table;
customer_names
--------------------------------------------------------------------------------
[John, Bob, Rajiv]
Fetched 1 row
```

Array Projection

Array projection is the application of the specified selector over all elements in an array, similar to a for loop.

Load data using array projection into the same table using this JSON data.

JSON

```
{ "a": [ {"name": "John", "hometown": "Chicago"},
         {"name": "Bob", "hometown": "Cucamonga"},
         {"name": "Rajiv", "hometown": "Glendale Heights"} ] }
```

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a[].name AS customer_names;
```

Output

SQL

```
Ocient> select * from my_schema.my_table;
customer_names
--------------------------------------------------------------------------------
[John, Bob, Rajiv]
Fetched 1 row
```

The output of both loading operations is the same.

Repeat this example with data in an array that contains nested objects. The JSON file contains this information.

JSON

```
{ "a": [ {"b": {"name": "John", "hometown": "Chicago"}},
         {"b": {"name": "Bob", "hometown": "Cucamonga"}},
         {"b": {"name": "Rajiv", "hometown": "Glendale Heights"}} ] }
```

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a[].b.name AS customer_names;
```

Output

SQL

```
Ocient> select * from my_schema.my_table;
customer_names
--------------------------------------------------------------------------------
[John, Bob, Rajiv]
Fetched 1 row
```

The output is still the same.

Multi-Dimensional Arrays

Multi-dimensional arrays work similarly to one-dimensional arrays.

These examples load the same data into the `my_table` table both directly, and then using array projection.

SQL

```
CREATE TABLE my_schema.my_table (customer_names VARCHAR[][][] NOT NULL);
```

Load data directly from a JSON array of strings. The JSON file contains this information.

JSON

```
{ "a": {
    "b": {
      "names": [
        [["John", "Bob", "Rajiv"], ["Anna", "Hanna", "Vanna", "Rosanna"]],
        [["Masha", "Natasha", "Sasha"], ["Chad", "Thad", "Brad"]]]    }
} }
```

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a.b.names[][][] AS customer_names;
```

Load data using array projection. The JSON file contains this information.

© 2026 Ocient, Inc.

JSON


```
{
    "a": [
        [
            [
                {"b": {"name": "John", "hometown": "Chicago"}},
                {"b": {"name": "Bob", "hometown": "Cucamonga"}},
                {"b": {"name": "Rajiv", "hometown": "Glendale Heights"}},
            ],
            [
                {"b": {"name": "Anna", "hometown": "Chicago"}},
                {"b": {"name": "Hanna", "hometown": "Cucamonga"}},
                {"b": {"name": "Vanna", "hometown": "Glendale Heights"}},
                {"b": {"name": "Rosanna", "hometown": "Mahwah"}},
            ],
        ],
        [
            [
                {"b": {"name": "Masha", "hometown": "Boston"}},
                {"b": {"name": "Natasha", "hometown": "Cambridge"}},
                {"b": {"name": "Sasha", "hometown": "Summerville"}},
            ],
            [
                {"b": {"name": "Chad", "hometown": "Champaign"}},
                {"b": {"name": "Thad", "hometown": "Urbana"}},
                {"b": {"name": "Brad", "hometown": "Mahomet"}},
            ],
        ],
    ]
}
```


`$a[][][].b.name` accesses each object, such as `{"b": {"name": "John", "hometown": "Chicago"}}`, and uses `b.name` to apply this nested attribute selector on all of the elements in the inner arrays.

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a[][][].b.name AS customer_names;
```

Output

SQL

```
Ocient> select * from my_schema.my_table;
customer_names
------------------------------------------------------------------------------
[[[John, Bob, Rajiv], [Anna, Hanna, Vanna, Rosanna]], [[Masha, Natasha, Sasha],
[Chad, Thad, Brad]]]
Fetched 1 row
```

The output of both loading operations is the same.

Multi-Dimensional Arrays With Dimensions Split Across the JSON Path

To load a two-dimensional array, for example, use a one-dimensional array that contains JSON objects, each of which contains another one-dimensional array. Load the relevant parts of this JSON object into a two-dimensional array.

In this example, the column has `VARCHAR[][]` type and the `SELECT` expression is `$x.a[].b.c[].name`. The loaded data is a two-dimensional array of names.

The JSON file contains this information.

JSON

```
{
    "x": {
        "a": [
            {
                "b": {
                    "c": [
                        {"name": "John", "hometown": "Chicago"},
                        {"name": "Bob", "hometown": "Cucamonga"},
                        {"name": "Rajiv", "hometown": "Glendale Heights"},
                    ]
                }
            },
            {
                "b": {
                    "c": [
                        {"name": "Anna", "hometown": "Chicago"},
                        {"name": "Hanna", "hometown": "Cucamonga"},
                        {"name": "Vanna", "hometown": "Glendale Heights"},
                        {"name": "Rosanna", "hometown": "Mahwah"},
                    ]
                }
            },
        ]
    }
}
```

SQL

```
CREATE TABLE my_schema.my_table (customer_names VARCHAR[][]);
```

© 2026 Ocient, Inc.

s

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $x.a[].b.c[].name AS customer_names;
```

Output

SQL

```
Ocient> select * from my_schema.my_table;
customer_names
------------------------------------------------------------------------------
[[John, Bob, Rajiv], [Anna, Hanna, Vanna, Rosanna]]
Fetched 1 row
```

The output is a two-dimensional array of names.

Extract Data from Individual Array Elements

Extract data from an element of a JSON array. That element can also be an array.

In this example, extract the first element of `a`, which is an array. JSON array indexes start at 1 so that all array indexing in Pipelines are consistent with the SQL standard.

The JSON file contains this information.

JSON

```
{
    "a": [
        {"b": [{"name": "John", "hometown": "Chicago"}]},
        {
            "b": [
                {"name": "Bob", "hometown": "Cucamonga"},
                {"name": "Rajiv", "hometown": "Glendale Heights"},
            ]
        },
        {"b": [{"name": "Rajiv", "hometown": "Glendale Heights"}]},
    ]
}
```

SQL

```
CREATE TABLE my_schema.my_table (customer_names VARCHAR[]);
```

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $a[2].b[].name AS customer_names;
```

Output

SQL

```
Ocient> select * from my_table.my_schema;
customer_names
--------------------------------------------------------------------------------
[Bob, Rajiv]
Fetched 1 row
```

NULL and Empty Handling for JSON Arrays

The Ocient System handles NULL, empty, and missing values the same way for arrays as for scalars. The system converts a value that is NULL, empty, or missing to NULL and loads it as `NULL`.

Provide an explicit default in the pipeline or use the `COLUMN_DEFAULT_IF_NULL` option to accept the configured column default instead of attempting to load `NULL`.

Tuples of Scalars

Data pipeline loading supports tuples of scalars in the JSON source.

The specification of tuple elements using curly braces is only supported for the basic case (i.e. `$a.{b,c}`). More complex selectors are not supported.

The recommended way of specifying tuples is to use an individual JSON selector for each tuple element, such as `$a.b` or `$a.c`.

You can apply functions to tuple elements.

Tuple Construction

This example loads two strings into a simple tuple.

JSON

```
{"users_hometowns": {"a": {"name": "Bob", "hometown": "Cucamonga"}}}
```

SQL

```
CREATE TABLE my_schema.my_table (users_hometowns TUPLE<<VARCHAR,VARCHAR>>);
```

© 2026 Ocient, Inc.

```SQL
CREATE PIPELINE my_pipeline
...
SELECT
    tuple<<char,char>>($users_hometowns.a.name, $users_hometowns.a.hometown)
    AS users_hometowns;
```

Output

```SQL
Ocient> select * from my_schema.my_table;
users_hometowns
-------------------------------------------------------------------------------
<<Bob, Cucamonga>>
Fetched 1 row
```

Advanced Tuple Construction

Use these examples to explore how to construct more complex tuples with nested objects and arrays.

Select Fields from a Nested Object into a Tuple

When the target fields are nested inside a named object, prefix the braces with the path to that object.

Selector:

```Text
$a.{b, c}
```

Transform equivalent:

Text

```
tuple($a.b, $a.c)
```

This data pipeline constructs a `TUPLE<<VARCHAR, INT>>` from fields `b` and `c` inside object `a`.

SQL

```
PREVIEW PIPELINE test_pl
    SOURCE INLINE '{"a": {"b": "two", "c": 2}}'
    EXTRACT FORMAT JSON
    INTO test.tgt_tbl
    SELECT $a.{b, c} AS tgt_col4;
```

Output

Text

```
tgt_col4
--------------------------------------------------------------------------------
("two", 2)
```

Select an Array of Tuples from an Array of Objects

Appending `[]` to the parent path maps the tuple selector over every element in the array, producing an array of tuples.

Selector:

Text

```
$a[].{b, c}
```

Transform equivalent:

```
Text
zip_with($a[].b, $a[].c, tuple)
```

This data pipeline produces a `TUPLE<<INT, VARCHAR>>[]` from an array of objects, each containing fields `b` and `c`.

```
SQL
PREVIEW PIPELINE test_pl
    SOURCE INLINE '{"a": [{"b": 1, "c": "John"},
        {"b": 2, "c": "Bob"},
        {"b": 3, "c": "Rajiv"}]}'
    EXTRACT FORMAT JSON
    INTO test.tgt_tbl
    SELECT $a[].{b, c} AS tgt_col5;
```

Output

```
Text
tgt_col5
--------------------------------------------------------------------------------
[(1, "John"),(2, "Bob"),(3, "Rajiv")]
```

Select a Tuple Containing an Array Element

Individual fields inside braces can have their own brackets `[]` to indicate that the field is an array. This selector produces a tuple with one element, an array, and the other a scalar.

Selector:

```
Text
$a.{b[], c}
```

Transform equivalent:

Text

```
tuple($a.b[], $a.c)
```

This data pipeline constructs a `TUPLE<<VARCHAR[], INT>>` where the first element is an array of strings and the second element is an integer.

SQL

```
PREVIEW PIPELINE test_pl
    SOURCE INLINE '{"a": {"b": ["John", "Bob", "Rajiv"], "c": 3}}'
    EXTRACT FORMAT JSON
    INTO test.tgt_tbl
    SELECT $a.{b[], c} AS tgt_col6;
```

Output

Text

```
tgt_col6
--------------------------------------------------------------------------------
([John,Bob,Rajiv], 3)
```

Select an Array of Tuples with an Array Element

Combining brackets `[]` on the parent path with `[]` on a child field produces an array of tuples where each tuple contains an array element and a scalar element.

Selector:

Text

```
$a[].{b[], c}
```

Transform equivalent:

Text

```
zip_with($a[].b[], $a[].c, tuple)
```

This data pipeline produces a `TUPLE<<VARCHAR[], INT>>[]` from an array of objects that each contain a nested array.

SQL

```
PREVIEW PIPELINE test_pl
    SOURCE INLINE '{"a": [{"b": ["John", "Bob", "Rajiv"], "c": 4},
        {"b": ["Rajiv", "Bob", "John"], "c": 5}]}'
    EXTRACT FORMAT JSON
    INTO test.tgt_tbl
    SELECT $a[].{b[], c} AS tgt_col7;
```

Output

Text

```
tgt_col7
--------------------------------------------------------------------------------
[([John,Bob,Rajiv], 4),([Rajiv,Bob,John], 5)]
```

Select a Tuple with Nested Tuple Elements

Nesting tuple selectors produces tuples where the elements are also tuples.

Selector:

Text

```
$.{a.{b, c}, d.{e, f}}
```

Transform equivalent:

Text

```
tuple(tuple($a.b, $a.c), tuple($d.e, $d.f))
```

This data pipeline constructs a `TUPLE<<TUPLE<<VARCHAR, INT>>, TUPLE<<INT, VARCHAR>>>>` from two nested objects.

SQL

```
PREVIEW PIPELINE test_pl
    SOURCE INLINE '{"a": {"b": "John", "c": 6},
        "d": {"f": "Bob", "e": 7}}'
    EXTRACT FORMAT JSON
    INTO test.tgt_tbl
    SELECT $.{a.{b, c}, d.{e, f}} AS tgt_col8;
```

Output

Text

```
tgt_col8
--------------------------------------------------------------------------------
(("John", 6), ("Bob", 7))
```

Select an Array of Tuples with a Nested Tuple Element

A dot-brace group preceded by `.` (with no field name) creates an inline tuple element within an outer tuple. When you combine such an element with brackets `[]` on the parent path, this selector produces an array of tuples where one element is a tuple.

Selector:

Text

```
$a[].{b, .{c[1], c[2], c[3]}}
```

Transform equivalent:

Text

```
zip_with($a[].b, zip_with($a[].c[1], $a[].c[2], $a[].c[3], tuple), tuple)
```

This data pipeline produces a `TUPLE<<INT, TUPLE<<VARCHAR, VARCHAR, VARCHAR>>>>[]` by selecting a scalar field and an inline tuple of indexed array elements.

SQL

```
PREVIEW PIPELINE test_pl
    SOURCE INLINE '{"a": [{"c": ["John", "Bob", "Rajiv"], "b": 4},
        {"c": ["Rajiv", "Bob", "John"], "b": 5}]}'
    EXTRACT FORMAT JSON
    INTO test.tgt_tbl
    SELECT $a[].{b, .{c[1], c[2], c[3]}} AS tgt_col10;
```

Output

Text

```
tgt_col10
--------------------------------------------------------------------------------
[(4, ("John", "Bob", "Rajiv")),(5, ("Rajiv", "Bob", "John"))]
```

Applying Functions (Transformations) To Tuple Elements

In this example, load a tuple where elements have different types, `VARCHAR` and `TIMESTAMP`, and where the second element has to be converted from a JSON string into an Ocient timestamp. The JSON file contains this information.

JSON

```
{"event_timestamp": {"event": "my_event", "timestamp": "1980-02-03 15:16:17.12345
```

SQL

```
CREATE TABLE my_schema.my_table (event_timestamp TUPLE<<VARCHAR,TIMESTAMP>> NOT N
```

SQL

```
CREATE PIPELINE my_pipeline
...
SELECT
    tuple<<char,timestamp>>($event_timestamp.event,
    TO_TIMESTAMP(char($event_timestamp.timestamp), 'yyyy-MM-dd HH:mm:ss.SSSSSS'))
    AS event_timestamp;
```

Output

SQL

```
Ocient> select * from  my_schema.my_table;
event_timestamp
--------------------------------------------------------------------------------
<<my_event, 1980-02-03 15:16:17.123450000>>
Fetched 1 row
```

The output table has one row of tuples.

NULL and Empty Handling for JSON Tuples

All the rules for handling NULL, empty, and missing elements that apply to scalars and arrays also apply to tuples. If any part of the selector is NULL, empty, or missing, data pipeline loading converts that value to `NULL`.

Additionally, because you can apply functions to tuple elements (and not array elements), you can use the `NULL_IF` function to convert a tuple element to `NULL`. For example, `tuple<<char,varchar>>( $a.name, NULL_IF($a.hometown, 'N/A') )` indicates to the pipeline that the string `'N/A'` signifies `NULL` for the `hometown` element but not for the `name` element.

Special Characters in JSON Keys

This example shows loading data where the JSON key names contain a special character.

Create a table for the load using the `customer_name` column.

SQL

```
CREATE TABLE my_schema.my_table (customer_name VARCHAR NOT NULL);
```

Load a file with this data. The JSON key name contains the `-` special character.

JSON

```
{"first-name": "John"}
```

Use double quotes for the selector `$first-name` in the `CREATE PIPELINE` SQL statement.

SQL

```
CREATE PIPELINE my_pipeline
    SOURCE filesystem
        FILTER 'data.json'
    EXTRACT
        FORMAT json
    INTO my_schema.my_table
    SELECT
        $"first-name" AS customer_name;
```

Array Extraction Operations

Other operations exist in the source field selector syntax to flatten or compact arrays while extracting the data.

Flatten Arrays

Flattening means that the extracted array should not increase the rank, and instead, sub-arrays should be concatenated. To flatten an array, use the underscore character in the array selector (e.g., `$data.array[_]`).

Example

This JSON data contains a nested array with two sub-arrays having data `[1,2,3]` and `[4,5,6]`.

JSON

```
{
  a: {
    b: [
      {c: [1,2,3] },
      {c: [4,5,6] }
    ]
  }
}
```

Flatten the nested array.

SQL

```
a.b[_].c[]
```

Output: `[1,2,3,4,5,6]`

This selected array has rank 1, and not 2. Contrast this with the non-flattened selector `a.b[].c[]`, which would have the output `[[1,2,3],[4,5,6]]`.

You can also flatten arrays with the `FLATTEN` function. For details, see **Array Data Transformation Functions**. The functionality of the operator and function is equivalent.

Compact Arrays

Compaction eliminates NULL values from the output data. The exact bit pattern of NULL in the source data is source-type dependent. For JSON, `null` is a literal keyword that is unambiguous. For CSV or other less-defined types, the configuration determines which exact bits equate to NULL. To compact an array, use the exclamation mark character in the array selector(e.g., `$data.array[!]`).

Example

This JSON data contains a nested array with four NULL values.

JSON

```
{
  a: {
    b: {
      c: [ 1,null,2,null,3,null,null]
    }
  }
}
```

Compact the nested array.

SQL

```
a.b.c[!]
```

Output: `[1,2,3]`

You can also compact arrays with the `ARRAY_COMPACT` function. For details, see **Array Data Transformation Functions**. The functionality of the operator and function is equivalent.

Related Links

Data Pipeline Load of Parquet Data from S3

Data Pipeline Load of Parquet Data from S3

© 2026 Ocient, Inc.

Data Types for Data Pipelines

Data Formats for Data Pipelines

Monitor Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Parquet Selectors Examples in Data Pipelines

The data pipeline functionality enables you to load data in the Apache® Parquet™ data format. Use these examples to see how you can access the structure in different ways. For an overview of loading Parquet data, see **Data Pipeline Load of Parquet Data from S3**.

Work with Parquet Selectors

The SELECT statement for a Parquet file uses the same syntax as the existing JSON selectors. This selection applies to lists, structures, and maps.

The Ocient® System retrieves its list of requested columns from the `SELECT` SQL statement. The columns correspond to the field names of the schema for the Parquet files for the load. The load retrieves required columns from the Parquet source files and leaves the other columns alone.

This example snippet shows a list of `["id", "name", "array[]", "a[].project", "my_map.key_name"]` as the fields to select from the data. The statement only retrieves the columns for `["id", "name", "array", "a", "my_map"]` in the row groups.

SQL

```
SELECT
    $id                     as id,
    $name                   as name,
    $array[]                as array,
    $a[].project            as projection,
    $my_map.key_name        as key_col
...
```

© 2026 Ocient, Inc.

You must select a leaf element, an array, or a tuple with your selector. This is stricter than using JSON selectors, which can directly select array fields and JSON object fields.

Example: `{"a": [1,2,3], "b": {"c": 1}}`

You can extract with any of the selectors in JSON: `$a, $a[], $b, $b.c`

However, Parquet only allows for the selectors: `$a[], $b.c`

This example assumes this schema:

```
// List<String> (list non-null, elements nullable)
required group my_list (LIST) {
  repeated group list {
    optional binary element (UTF8);
  }
}
```

The selector must be `$my_list[]`, which includes the array syntax.

Work with Complex Parquet Selectors

These examples show how to load a Parquet list and convert it to an array and how to load a map and convert it to a tuple.

Load List as an Array Example

Create the `tips` table to store the tip inquiries a city receives. The columns are:

- `id` — Identifier as an integer
- `categories` — Categories as a list of strings
- `mayor` — Mayor as a string
- `mentions` — Mentions as a list of strings
- `service_name` — Service name as a non-nullable string
- `tags` — Tags as a list of strings
- `url` — URL as a string

© 2026 Ocient, Inc.

- `username` — Username as a string
- `text` — Text description as a non-nullable string
- `venue_name` — Venue name as a non-nullable string

SQL

```
CREATE TABLE tips(
    id INT NOT NULL,
    categories VARCHAR(255)[],
    mayor VARCHAR(255),
    mentions VARCHAR(255)[],
    service_name VARCHAR(255) NOT NULL,
    tags VARCHAR(255)[],
    url VARCHAR(255),
    username VARCHAR(255),
    text VARCHAR(255) NOT NULL,
    venue_name VARCHAR(255) NOT NULL);
```

Create the `tips_pipeline` data pipeline for loading the tip data. Specify the Amazon® Web Services™ (AWS™) S3 source with bucket `ocient-docs`, object key `metabase_samples/parquet/tips.parquet`, and endpoint `https://s3.us-east-1.amazonaws.com`. The source format is Parquet.

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE tips_pipeline
    SOURCE S3
        BUCKET 'ocient-docs'
        OBJECT_KEY 'metabase_samples/parquet/tips.parquet'
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
    EXTRACT
        FORMAT parquet
INTO tips
SELECT
    $id AS id,
    $source.categories[] AS categories,
    $source.mayor AS mayor,
    $source.mentions[] AS mentions,
    $source."service" AS service_name,
    $source.tags[] AS tags,
    $source.url AS url,
    $source.username AS username,
    $text AS text,
    $venue.name AS venue_name;
```

Start the data pipeline. The data pipeline loads the `categories`, `mentions`, and `tags` columns as arrays.

SQL

```
START PIPELINE tips_pipeline;
```

Load Map as a Tuple Example

This example uses the file `/tmp/to-load/locations.parquet` that contains a `MAP<STRING, DOUBLE>` field named `coordinates`. Each value of the map field contains `x` and `y` entries for the coordinates.

Create the `locations` table to store the map values. The `cartesian_coords` column is a tuple with two elements for the `x` and `y` coordinates.

```sql
CREATE TABLE locations(
    cartesian_coords TUPLE<<DOUBLE, DOUBLE>> NOT NULL);
```

Create the `locations_pipeline` data pipeline for loading the maps data. Specify the filter for the Parquet file `/tmp/to-load/locations.parquet`. The source format is Parquet. Select the `x` and `y` coordinates from the map.

```sql
CREATE PIPELINE locations_pipeline
    SOURCE filesystem
        FILTER '/tmp/to-load/locations.parquet'
    EXTRACT
        FORMAT parquet
INTO locations
SELECT
    TUPLE($coordinates.x, $coordinates.y) AS cartesian_coords;
```

Start the data pipeline. The data pipeline loads the `cartesian_coords` column as a tuple.

```sql
START PIPELINE locations_pipeline;
```

Related Links

Data Types for Data Pipelines

Data Formats for Data Pipelines

Data Pipeline Load of JSON Data from Kafka

JSON Selectors Examples in Data Pipelines

Data Pipeline Load of Parquet Data from S3

© 2026 Ocient, Inc.

Monitor Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Pipeline Behavior Considerations

Data pipelines enable you to load data from your chosen source into the OcientAIQ™ Unified Data Platform. For details about the syntax, see **Data Pipelines**. When you use data pipelines to load data, consider how to resume a pipeline with file loading, restart the pipeline with Apache® Kafka® loading, and review pipeline dependencies.

Resume a Pipeline with File Loading

In many cases, a file-based pipeline stops executing before completion. You cannot resume a pipeline in a `COMPLETED` status.

To resume a pipeline, use the `START PIPELINE` SQL statement. Before you resume a pipeline, the status of the pipeline must be `CREATED`, `FAILED`, or `STOPPED`. When a pipeline resumes, individual files remain in their most recent status as defined in the `sys.pipeline_files` system catalog table. For `BATCH` pipelines, the Ocient® System does not add new files to the eligible file list when the pipeline resumes.

> ⚠ If you modify the contents of files during the loading process, the Ocient System might experience issues with deduplication that cause duplicated rows or missing data. Avoid modifying files after you start a pipeline for the first time.
>
> Creating new files on your data source does not impact deduplication logic.

The `START` operation groups files using their `extractor_task_id` and `stream_source_id` identifiers. The `stream_source_id` uniquely identifies partitions (i.e., an ordered list of files), and `extractor_task_id` identifies the batch that loads a group of partitions.

File Statuses

The Ocient System considers files with the statuses `LOADED`, `LOADED_WITH_ERRORS`, or `SKIPPED` to be in the terminal status, whereas other file statuses are still in process.

- **Completed Batches** — If all the files in a particular batch have terminal status, then the pipeline does not attempt to reload the batch. These files have been completely processed, so the Ocient System ignores modifications to these files.

© 2026 Ocient, Inc.

- **In-Process Batches** — If at least one file in a particular batch does not have terminal status, then the pipeline reloads the entire batch. The pipeline reprocesses the in-process batches and relies on row deduplication to prevent duplication of rows in the target tables.
  - Modifications to files in an in-process batch can but are not guaranteed to be picked up by a restart.
  - Modifications to any files in this batch with the `LOADED`, `LOADED_WITH_ERRORS`, or `SKIPPED` statuses might cause issues with deduplication, leading to duplicate or missed data.
- **Pending Files** — The Ocient System does not assign all `PENDING` files to a partition. The pipeline attempts to load these files after reloading any in-progress batches.

Load Duplicate Data from Files

Sometimes you might want to load the same data multiple times. If you want to load a second copy of the source data, you can either:

- Drop and recreate the pipeline to reset the `sys.pipeline_files` system catalog table.
- Create a second pipeline with a new name and the same configuration.

> When you truncate the target tables and restart the pipeline, the Ocient System does not reload the data.

Restart with Kafka Loading

Ocient relies on the offset management and consumer group behavior in Kafka to deliver exactly-once loading semantics and to control the Ocient pipeline behavior.

Kafka Offsets and Consumer Group Identifiers

If you set the `WRITE_OFFSETS` option to `true` (default value is `true`), the Kafka consumers commit offsets back to Kafka after data is considered durable in the database. The Kafka Broker stores these offsets as the last committed offset for the group identifier `group.id`.

For each pipeline, the group identifier defaults to `<systemId>__<databaseName>__<pipelineName>`, where the `<systemId>` is the identifier of your system, `<database-name>` is the name of your database, and `<pipeline-name>` is the name of the data pipeline. In most use cases, you should not manually change the `group.id` field for a pipeline.

Any Kafka pipeline that has the same `group.id` starts consuming from its last committed offset, or if you do not set the value, the pipeline uses the Kafka `auto.offset.reset` policy to determine where to start. For details, see **Kafka offset management**.

> If you want to start loading from the beginning of a topic, configure an unused `group.id` field (or use a `group.id` field that did not commit any of its offsets back) and ensure the `auto.offset.reset` Kafka configuration is appropriately set in the `CONFIG` option.

Kafka Pipeline Deduplication

The committed offset of a Kafka partition lags slightly behind the rows that have been loaded into Ocient. These lags do not cause an issue with data duplication. If you stop a pipeline before it can commit its most recent durable offset to Kafka, restarting the same pipeline starts loading from the last committed offset. However, the database deduplicates records sent twice for the same pipeline.

Ocient deduplicates Kafka data for the specified combination of pipeline identifier, Kafka topic, and the Kafka partition number.

While the consumer group offsets manage where the pipeline resumes loading, the Ocient System enforces the exactly-once loading of a Kafka partition only if you stop or restart a pipeline with the same pipeline identifier.

> If you drop a pipeline and create a new one with the same name, the Ocient System creates a new pipeline identifier. This action does not deduplicate data against data loaded in the original pipeline. To preserve deduplication, instead of dropping the pipeline, use the `CREATE OR REPLACE PIPELINE` SQL statement with the original pipeline name and the pipeline correctly deduplicates against the original data.

© 2026 Ocient, Inc.

⚠ Do not run multiple pipelines concurrently with the same consumer group identifier. This action leads to unpredictable data duplication. If you want to increase the number of consumers that read from a Kafka topic, increase the value of the `CORES` parameter.

Load Duplicate Data on Kafka

Sometimes you might want to load the same data multiple times. If you want to load a second copy of the source data from Kafka, you can either:

- Drop the pipeline, recreate it with the same name, and reset the consumer group offsets manually.
- Create a new pipeline with a different name and load from the beginning of the topic.

Pipeline Database Dependency

Each pipeline belongs to a database. You cannot drop a database that has a running pipeline. To drop a database, ensure that all pipelines in the database are in a non-running status.

Pipeline Table Dependency

Each pipeline has a target table. You cannot drop a table that has a running pipeline. To drop a table, ensure that all pipelines that are loading data into the table are in a non-running status.

Related Links

Load Data

Data Pipeline Load of CSV Data from S3

Data Pipeline Load of Parquet Data from S3

Transform Data in Data Pipelines

Load Metadata and File-Based Partitioned Data in Data Pipelines

© 2026 Ocient, Inc.

Data Control Language (DCL) Statement Reference

Identifiers

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Transform Data in Data Pipelines

Data pipeline functionality supports data transformation during loading to efficiently convert and clean source data for your chosen table schema. You can use two broad classes of data transformation: source field references or SQL transformation functions.

Source Field References

When you load data using a data pipeline, you reference extracted data using a special syntax named a source field reference. These expressions are either a named reference such as `$my_field` or a numeric field reference such as `$1`.

In structured data formats like JSON, more complex source field references allow you to transform the structured data into the required form. For example, you can extract an array of values from an array of nested objects with an expression such as `$data.items[].price`.

To learn more about transforming data with source field references, see:

- **Supported JSON Selectors**
- **Understand Source Fields and Data Formats**

Supported Transformation Functions in Data Pipelines

Data pipeline loading supports numerous SQL functions for use in transforming data during loading. You can use these transformation functions in the `INTO ... SELECT` portion of a pipeline. Transformation functions generally match the syntax and behavior of the function you use when you query in Ocient®. The behavior might differ from querying behavior in two ways:

- Some additional functions exist that provide behavior that does not exist in queries.
- Some existing functions have extended behavior during loading that provides important capabilities that are not in queries.

The types of transformation functions supported in loading include:

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

- Scalar data conversion and casting of data types
- Special loading transformation functions
- Date and time transformations
- Character data transformations
- Binary data transformations
- Array data transformations
- Logical operations transformations
- Network type functions
- Other data transformations
- Custom user-defined **Pipeline Functions** using Apache® Groovy™

Scalar Transformation Functions and Casting

Scalar transformation functions include constructors, scalar data conversions, and data-casting functions.

Name	Description	Supported Input Type	Return Type	Example SI Expression
`T[]\(X)`	Casts a string representation of an array of values or the result of a JSON array extraction expression to the corresponding array with data type `T` where `T` is a SQL data type. Multi-dimensional arrays are supported. To indicate an N-dimensional array cast, use N sets of square brackets in the casting function. All transformation functions can only apply to an array as a whole. To apply functions on an element-wise basis, see `TRANSFORM` in	CHAR	ARRAY[]	CSV: `DECIMAL(8,4` `($21) as decimal_2d_ ol` JSON: `CHAR[]\ ($a.b.names names` `DECIMAL(8,2 ($a[].b.c[] ) as prices` `CHAR[][][]\ ($names[][] names`

Name	Description	Supported Input Type	Return Type	Example SI Expression
	Special Data Pipeline Transformation Functions.			

`BIGINT(X)`	Converts the specified type to a 64-bit signed integer.	BIGINT
		CHAR
			BIGINT	CSV:
				`BIGINT($22)`
				`int64_col`
				JSON:
				`BIGINT($a.b`
				`int64_col`

`BINARY(X)`	Translates a string representation of bytes and converts them to real bytes. The string representation expects hexadecimal representations that convert to the matching bytes with this function. Hexadecimal data is not case-sensitive. Example source data: `a891`, `F1D890`	CHAR	BINARY or VARBINARY	CSV:
				`BINARY($30)`
				`varbinary_c`
				JSON:
				`BINARY($a.b`
				`varbinary_c`

`BOOLEAN(X)`	Handles textual representation of Booleans and converts them to its Boolean equivalent. You can represent `true` as any of these values: `"t", "true", "yes", "1", "y", "on"` `false` representation as any of these values: `"f", "false", "no", "0", "n", "off"`	CHAR	BOOL	CSV:
				`BOOLEAN($23`
				`bool_col`
				JSON:
				`BOOLEAN($a.`
				`bool_col`

`BYTE(X)`	Supports characters and number or byte array representations. Interprets character data as the ASCII numeric representation of a byte. If you specify a byte array, the Ocient	CHAR	BYTE	CSV:
				`BYTE($24) a`
				`byte_col`
				JSON:
				`BYTE($a.b.c`
				`byte_col`

Name	Description	Supported Input Type	Return Type	Example SI Expression

	System uses the first element of the array. When the system converts text data to a byte, the function throws an error if the value is `> 127` or `< -128`.
	Example:
	`'101'` -> `101`
	`'-23'` -> `-23`			

`CHAR(X)`	Converts the specified input type to its string representation.	CHAR
		BINARY
		FLOAT
		DOUBLE
		INT
		BYTE
		SMALLINT
		DATE
		BOOLEAN
		UUID
		ARRAY
			CHAR or VARCHAR	CSV:
				`CHAR($25) a`
				`varchar_col`
				`CHAR($26) a`
				`char_col`
				JSON:
				`CHAR($a.b.c`
				`varchar_col`
				`CHAR($a.b.c`
				`char_col`
				`CHAR($array`
				`]) as char_`

`DECIMAL(X,P,S)`	`X` is input data to convert to a decimal. `P` is the precision of the decimal (number of significant digits) and `S` is the scale (number of digits after the decimal point).
	You can omit `P` and `S` or only `S`. In that case, the Ocient System defaults `P` to 31 and `S` to 16.
	Example:
	`DECIMAL(30.0,4,2) -> 30.00`
		CHAR	DECIMAL	CSV:
				`DECIMAL($27`
				`as decimal_`
				JSON:
				`DECIMAL($a.`
				`4) as decim`

Name	Description	Supported Input Type	Return Type	Example SI Expression
`DOUBLE(X)`	Converts the specified type to its double representation.	BINARY CHAR	DOUBLE	CSV: `DOUBLE($28)` `double_col` JSON: `DOUBLE($a.b` `double_col`
`FLOAT(X)`	Converts the specified type to its float representation.	BINARY CHAR	FLOAT	CSV: `FLOAT($29)` `float_col` JSON: `FLOAT($a.b.` `float_col`
`HASH(X,N)`	Translates a string representation of bytes and converts them to bytes. The string representation expects hexadecimal representations that convert to the matching bytes with this function. Hexadecimal data is not case-sensitive. `X` is the source data, such as: `a891` , `F1D890` `N` is the fixed number of bytes in the binary array.	CHAR	HASH	CSV: `HASH($23, 9` `fixed_hash_` `col` JSON: `HASH($a.b.c` `fixed_hash_` `col`
`INT(X)`	Converts the specified type to a 32-bit signed integer.	DECIMAL CHAR SMALLINT	INT	CSV: `INT($31) as` `int32_col` JSON: `INT($a.b.c)` `int32_col`
`IP(X)`	Converts the specified string to an IPv6 address equivalent. `X` is the source data, such as: `2001:0db8:1234:5678:a` `bcd:ef01:2345:6789` `2001:db8:0:0:0:0:0:2` `192.0.2.1`	CHAR	IP	CSV: `IP($32) as` JSON: `IP($a.b.c)` `ip_col`

Name	Description	Supported Input Type	Return Type	Example SI Expression
`IPV4(X)`	Converts the specified string to an IPv4 address equivalent. `X` is the source data, such as: `192.0.2.1`	CHAR	IPV4	CSV: `IPV4($32) a ipv4_col` JSON: `IPV4($a.b.c ipv4_col`
`SMALLINT(X)`	Converts the specified value to its 16-bit signed integer representation.	DECIMAL CHAR BYTE	SMALLINT	CSV: `SMALLINT($3 int16_col` JSON: `SMALLINT($a as int16_co`
`ST_POINT(Long, Lat)`	Accepts a longitude and latitude coordinate for a geospatial point. Arguments must be of type `double` or string values that can convert to `double`. For WKT representation, see **ST_POINTFROMTEXT**.	Both DOUBLE Both CHAR	`ST_POINT`	CSV: `ST_POINT($1 as st_point` JSON: `ST_POINT($a $d.e.f) as st_point_co`
`ST_POINTFROMTEXT(X)`	Accepts the WKT text representation for `ST_POINT` with `X` and `Y` coordinates only. Example WKT representation: `'POINT(-40.5318 11.8532)'`	CHAR	`ST_POINT`	`st_pointfro 1)` `st_pointfro a.b.c)`
`TINYINT(X)`	Converts the specified value to an 8-bit signed integer.	CHAR	TINYINT	CSV: `TINYINT($34 int8_col` JSON: `TINYINT($a. int8_col`
`TUPLE <<T,[T]*>> (X,[Y]*)`	Creates a tuple from the specified types. TUPLE supports arrays of tuples, but does not support tuples within tuples. The arguments to the TUPLE casting function can be either a single	CHAR	TUPLE of type	CSV: `TUPLE<<INT, (7,2),TIMES NARY(7)>>($ int_decimal amp_binary_ ol` `TUPLE<<INT, (7,2),TIMES`

Name	Description	Supported Input Type	Return Type	Example SI Expression
	source value or must match the number of types defined in the TUPLE. If you specify a single value, the Ocient System infers individual elements of the tuple using the TUPLE format `(value1,value2,value3)`. Example source data: `(123,12.75)` is valid text data for a `TUPLE<<INT,DECIMAL(7,2)>>`.			`NARY(7)>>($4,$8,$1) int_decimal amp_binary_ ol_multi_ir` JSON: `TUPLE<<INT, (7,2),TIMES NARY(7)>>($ $c)) as int_decimal amp_binary_ ol`
`UUID(X)`	Creates a unique identifier from the specified value. Example source data: `'237e9877-e79b-12d4-a765-321741963000'`	CHAR	UUID	CSV: `UUID($34) a uuid_col` JSON: `UUID($a.b.c uuid_col`

Special Data Pipeline Transformation Functions

Special data pipeline transformation functions include functions that only exist in loading, such as the `METADATA` function to load special data during pipeline execution.

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`EXPLODE_OUTER(array)`	Expands the contents of the specified one-dimensional array. By default, the array can contain a maximum of 8,191 elements. For supported options, see **Special Data Pipeline Transformation Functions**.	ARRAY	Elements of the array in their own row.	`EXPLODE_OUTER($ ta.response.res ses[].own") AS own`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`FILTER(T[], (T) -> Boolean)`	Filters elements in an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to filter. The second argument is a lambda expression (or function reference) that returns `true` if an element should be retained. The second argument must be of the form `(x T) -> BOOL`. This returns a boolean value for a specified element `x`. The type `T` of the element `x` is specified in the lambda function. Elements in the first argument array are included in the resulting array when the lambda function returns `true` and excluded otherwise. For supported options and examples, see **Special Data Pipeline Transformation Functions**.	ARRAY Lambda Function `(x T) -> BOOL` or Function Reference	The elements of the array that matched the filter criteria function.	`FILTER($data.na , (name CHAR) - length(name) >`
`LOOKUP(lookup_sour ce_name, value, join_column_name,`	Look up and load data in an external data source. For details, see **Load**	Varies based on the `value` argument.	Varies based on the returned data in the	`LOOKUP('sqlite_ , $2, 'firstnam 'id')`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`return_column_name)`	**Data from External Sources in Data Pipelines**.		`return_column_name` argument.	
`METADATA(key)`	Extracts the metadata value for the specified key from available metadata for the pipeline. For supported key values, see **Load Metadata and File-Based Partitioned Data in Data Pipelines**.	CHAR	Varies based on the metadata key.	`METADATA('filen') -> '/path/to/file/ers_1830374.jso`
`PARSE_DELIMITED_ARRAY(text, rank, array_element_delimiter, bracket_spec)`	Converts a string of text data representing an array into a `CHAR[]`. For example, with data in the form `[1;2;3]`, executing `PARSE_DELIMITED_ARRAY('[1;2;3]', 1, ';', '[]')` returns a `CHAR[]` with the value `['1','2','3']`.	`text` is CHAR data that represents the text to convert to an array. `rank` is an optional argument that represents the numeric rank of the output array. The default value is 1. You can only specify rank as a literal value because you cannot retrieve it dynamically from a source field selector. `array_element_delimiter` is an optional argument that contains the control character for delimiting array elements. For example, `','` indicates that a comma separates each element in	CHAR[]	`PARSE_DELIMITED RAY($text_array PARSE_DELIMITED RAY($text_array 1, ';', '[]') PARSE_DELIMITED RAY($two_dim_te array, 2, ',', '[]')`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
		the `text` character data. The default value definition comes from the `ARRAY_ELEMENT_DEL IMITER` value of the data pipeline. `bracket_spec` is an optional argument that is the pair of control characters for the open and close array brackets. For example, `'[]'` indicates that each rank of the array is enclosed in square brackets. The default definition comes from the `OPEN_ARRAY` and `CLOSE_ARRAY` characters of the data pipeline. Valid options are: `''`, `'[]'`, `'{}'`, `'()'`, and `'<>'`.		
`TRANSFORM(T[], (T) -> U)`	Transforms an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to transform. The second argument is a lambda expression (or function reference) that defines how to transform each element.	ARRAY Lambda Function `(x) -> U` or Function Reference	Elements of the array transformed to the return value and type of the anonymous function.	`TRANSFORM($data mes[], (el CHAR > LOWER(el)) as lower_case_name ray` `TRANSFORM($data mes[], LOWER) a lower_case_name ray`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	`TRANSFORM` returns an array of elements of the same length as the array argument of the type returned by the lambda function. The second argument must be of the form `(x T) -> U` . This returns a value of type `U` for a specified element `x` . The type `T` of the element `x` is specified in the lambda function. The second argument can also be a function reference for a single argument function. For supported options and examples, see **Special Data Pipeline Transformation Functions**.			
`WIDTH_BUCKET(X,min ,max,N)`	Creates `N` equal-width buckets in the range `[min,max)` as a histogram. If `X` is less than `min` , then the function returns 0. If `X` is greater than or equal to the value of `max` , the function returns `N+1` buckets. If `X` is less than the value of `max` and greater than	`X` is the value for bucket determination. `min` is the minimum value for the histogram. `max` is the maximum value for the histogram. `N` is the number of buckets in the histogram.	Integer	`WIDTH_BUCKET('$ 1, 5, 2) AS output`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	or equal to the value of `min`, the function returns the number of the bucket corresponding to where the input value belongs in the histogram. The values are inclusive on the lower bound and exclusive on the upper bound, so all buckets are the same width. Requesting 100 buckets retrieves 102 possible buckets, bucket 0 through 101, where 0 captures all values below the minimum and 101 captures all values at the maximum and beyond.			
`ZIP_WITH(array1, ..., array2, func)`	Merges two or more input arrays, element-wise, into a single array using the trailing combining function. If one array is shorter than the others, the system appends `NULL` values at the end to match the length of the longest array before applying. Returns an `array` with elements that result from applying the function `func` to	ARRAY ARRAY Lambda Function `(x T, y U) -> V` or Function Reference	An array of length `max(length(array1), length(array2))` containing elements of type `V` as returned by the lambda function.	`ZIP_WITH($line_ms[].product_na $line_items[].s (name CHAR, sku CHAR) -> concat(name, '[ sku, ']'))`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	each pair of elements from `array1` and `array2`. If the arrays are not the same length, the system pads the shorter array with `NULL` values. `func` must be of the form `(x T, y U) -> V`, where `x` is the element from `array1` and `y` is the element from `array2`. Types `T` and `U` are specified in the lambda function. For supported options and examples, see **Special Data Pipeline Transformation Functions**.			

Date and Time Transformation Functions

Date and time transformation functions provide utilities to parse, convert, or transform `DATE`, `TIME`, and `TIMESTAMP` values in pipelines.

Name	Description	Supported Input Type	Return Type	Example SEI Expression

`DATE(X)`	If the input type is a `CHAR`, the `DATE` function parses the string into a `DATE` type according to the format `YYYY[-/]MM[-/]DD...`. The format is in this order:
	4-digit year
		CHAR
		TIMESTAMP
			DATE	CSV:
				`DATE($24) as date_col`
				`DATE(TIMESTA )) as date_c`
				JSON:
				`DATE($a.b.c) date_col`
				`DATE(TIMESTA b.c)) as dat`

Name	Description	Supported Input Type	Return Type	Example SEI Expression
	dash or backlash 2-digit month a dash or backslash 2-digit day any number of trailing characters If the input type is a `TIMESTAMP`, the function parses the date portion of the timestamp (in UTC) into a `DATE` type.			

`TO_DATE(X,F,FT)`	Converts the specified value to a DATE.
	`X` — The value to convert.
	`F` — (optional) The format string for the interpretation of the textual representation of `X`.
	`FT` — (optional) The format type. There are two format types: `ocient` and `java`. If you use the `ocient` type, the function uses the Ocient-based formatting function to parse the date. For details, see **TO_DATE**. If you use the `java` type, the function uses the `DateTimeFormatter` in Java® to parse the timestamp. For details, see **DateTimeFormatter (Java Platform SE 8)**.
	
	The default is the `ocient` format type with format string `YYYY-MM-DD`. If you
		CHAR	DATE	`to_date($3,`
				`'YYYY/MM/DD'`
				`to_date($a.b`
				`'DD.MM.YYYY'`

Name	Description	Supported Input Type	Return Type	Example SEI Expression
	specify the `java` format type, you must specify a valid `DateTimeFormatter` format string, as there is no default.			
`INT_TO_DATE(arg)`	Creates a `DATE` type from an `INT` type representing the number of days after the Unix® epoch (January 1, 1970).	INT	DATE	`INT_TO_DATE( col_date_1`

`TIME(X)`	If the input type is a `CHAR`, the `TIME` function parses the string into a `TIME` according to the format `HH:MI[:SS[.NS]]`. The format is a 2-digit hour between 00 and 23, a colon, a 2-digit minute, an optional colon and 2-digit second, and an optional period and 1-9 digits for fractional seconds. The value can end optionally in `Z` to denote UTC (Zulu). The value can also end optionally in `±TZH[:TZM]` to denote the timezone offset. This format is a plus or minus sign, a 2-digit hour, and an optional colon and 2-digit minute. The system resolves all values to UTC, so `Z`, `+00:00`, and `-00` are equivalent.
	
	If the input type is a `BIGINT`, the function parses the number with millisecond
		CHAR
		BIGINT
		TIMESTAMP
			TIME	CSV:
				`TIME($34) as time_col`
				`TIME(BIGINT( as time_col`
				JSON:
				`TIME($a.b.c) time_col`
				`TIME(BIGINT( ) as time_co`

Name	Description	Supported Input Type	Return Type	Example SEI Expression
	precision. If the input type is a `TIMESTAMP`, the function parses the date portion of the timestamp (in UTC) into a `TIME` type.			
`TO_TIME(X,F,FT)`	Converts the specified value to a TIME value. `X` — The value to convert. `F` — (optional) The format string for the interpretation of the textual representation of `X`. `FT` — (optional) The format type. There are two format types: `ocient` and `java`. If you use the `ocient` type, the function uses the Ocient-based formatting function to parse the time. For details, see **TO_TIMESTAMP**. If you use the `java` type, the function uses the `DateTimeFormatter` in Java to parse the timestamp. For details, see **DateTimeFormatter (Java Platform SE 8)**. The default is the `ocient` format type with format string `HH:MI:SS`. If you specify the `java` format type, you must specify a valid `DateTimeFormatter`	CHAR	TIME	`to_time($n, 'HH:MI:SS')` `to_time($n, 'HHMISS')`

Name	Description	Supported Input Type	Return Type	Example SEI Expression
	format string, as there is no default.			
`SECONDS_TO_TIME(arg)`	Converts the specified value to the `TIME` type, treating the argument with second precision.	BIGINT	TIME	`SECONDS_TO_T ) AS col_tim`
`MILLIS_TO_TIME(arg)`	Converts the specified value to the `TIME` type, treating the argument with millisecond precision. Alias for `TIME(X)` where `X` is the `BIGINT` type.	BIGINT	TIME	`MILLIS_TO_TI AS col_time`
`MICROS_TO_TIME(arg)`	Converts the specified value to the `TIME` type, treating the argument with microsecond precision.	BIGINT	TIME	`MICROS_TO_TI AS col_time`
`NANOS_TO_TIME(X)`	Converts the specified value to a TIME value. If the argument has the `BIGINT` type, the function treats the number with nanosecond precision.	BIGINT	TIME	CSV: `NANOS_TO_TIM as time_col` `NANOS_TO_TIM NT($34)) as time_col` JSON: `NANOS_TO_TIM .c) as time_` `NANOS_TO_TIM NT($a.b.c) a time_col`
`TIMESTAMP(X)`	If the input type is a `CHAR` , the `TIMESTAMP` function parses the string into a `TIMESTAMP` according to the format `YYYY[-/]MM[-/]DD[T ]HH:MI[:SS[.NS]]` . For the format, see the `DATE` and `TIME` functions. The input can have either the	CHAR BIGINT DATE	TIMESTAMP	CSV: `TIMESTAMP($3 timestamp_co` JSON: `TIMESTAMP($a as timestamp` `TIMESTAMP('2 03-01 18:22:58.826` `2023-03- 01T18:22:58.` `TIMESTAMP(bi 651681788123`

Name	Description	Supported Input Type	Return Type	Example SEI Expression
	letter `T` or a space separating the date and time portions. The input can also end in `Z` to denote UTC or `±TZH[:TZM]` to denote the time zone offset. If the input type is a `BIGINT`, the function parses the number with millisecond precision. If the input type is a `DATE`, the function converts the date into a `TIMESTAMP` with the value of the start of the day in UTC.			`)) -> 2022-0` `04T16:29:48.`
`TO_TIMESTAMP(X,F,FT)`	Converts the specified value to a TIMESTAMP. `X` — The value to convert. `F` — (optional) The format string for the interpretation of the textual representation of `X`. `FT` — (optional) The format type. There are two format types: `ocient` and `java`. If you use the `ocient` type, the function uses the Ocient-based formatting function to parse the timestamp. For details, see **TO_TIMESTAMP**. If you use the `java` type, the function uses the `DateTimeFormatter` in Java to parse the	CHAR	TIMESTAMP	`to_timestamp` `'YYYY-MM-DD` `HH:MI:SS.US'` `to_timestamp` `'DD.MM.YYYY` `HH:MI:SS.NS'` `to_timestamp` `'uuuu-mm-dd` `HH:mm:ss.SSS` `'java')`

Name	Description	Supported Input Type	Return Type	Example SEI Expression
	timestamp. For details, see **DateTimeFormatter (Java Platform SE 8)**. The default is the `ocient` format type with format string `YYYY-MM-DD HH:MI:SS`. If you specify the `java` format type, you must specify a valid `DateTimeFormatter` format string, as there is no default.			
`SECONDS_TO_TIMESTAMP(X)`	Converts the specified value to a TIMESTAMP. This function creates a TIMESTAMP from a BIGINT that represents the number of nanoseconds after the Unix epoch (January 1, 1970).	BIGINT	TIMESTAMP	CSV: `SECONDS_TO_TIMESTAMP($35) as timestamp_co` JSON: `SECONDS_TO_TIMESTAMP($a.b.c) a timestamp_co NANOS_TO_TIM (bigint(1651 123456789)) 2022-05-04T16:29:48`
`MILLIS_TO_TIMESTAMP(X)`	Converts the specified value to a TIMESTAMP. This function creates a TIMESTAMP from a BIGINT that represents the number of milliseconds after the Unix epoch (January 1, 1970). Alias for `TIMESTAMP(X)` where `X` is the `BIGINT` type.	BIGINT	TIMESTAMP	`MILLIS_TO_TIMESTAMP($35) as timestamp_co`

Name	Description	Supported Input Type	Return Type	Example SEI Expression
`MICROS_TO_TIMESTAMP(X)`	Converts the specified value to a TIMESTAMP. This function creates a TIMESTAMP from a BIGINT that represents the number of microseconds after the Unix epoch (January 1, 1970).	BIGINT	TIMESTAMP	`MICROS_TO_TIP($35) as timestamp_co`

`NANOS_TO_TIMESTAMP(X)`	Converts the specified value to a TIMESTAMP.
	This function creates a TIMESTAMP from a BIGINT that represents the number of nanoseconds after the Unix epoch (January 1, 1970).
		BIGINT	TIMESTAMP	CSV: `NANOS_TO_TIM($35) as timestamp_co`
				JSON: `NANOS_TO_TIM($a.b.c) as timestamp_co`
				`NANOS_TO_TIM(bigint(1651123456789))`
				`2022-05-04T16:29:48.789`

`NOW()`	Returns the current TIMESTAMP.		TIMESTAMP	`now() as col_load_timp`

For the TO_TIMESTAMP function, the Ocient formatter currently does not support variable-length sections of timestamps (i.e., the data might have a different number of fractions of seconds across records).

For example, assume this source data.

```
2024-01-01 12:00:00.123456
```

```
2024-01-01 12:00:00.1234
```

Then, you must specify the `format_type` as `java` and enter the `DateTimeFormatter` format string to parse the timestamps.

Character Data Transformation Functions

Character data transformation functions operate on `VARCHAR` data to transform a string of data.

Name	Description	Supported Input Type	Return Type	Example SELEC Expression

`CONCAT(arg1, arg2[, ...])`	Returns a string that is the result of concatenating the arguments provided. Accepts two or more arguments to concatenate.
	If one of the arguments to `CONCAT` is `NULL`, then `CONCAT` returns `NULL`.
	ℹ `CONCAT` does not support `HASH` or `BINARY` arguments in loading.
		`arg1` : CHAR
		`arg2` : CHAR
		`arg3` ...
		`argN` (optional) CHAR
			CHAR	`CONCAT($1, $2)`
				`CONCAT(TRIM($fi`
				`_name), '_',`
				`TRIM($last_name`

`SUBSTRING(X, P, L)`	Extract a substring from `X` beginning at the position offset `P` with a length `L`.
	`X` is the source data in CHAR or BINARY data type.
	`P` is the numeric byte position in the record that starts at index 1, which indicates the position to begin reading.
	`L` is the length of data to return starting from position `P`.
	For `VARCHAR` data, by default, the Ocient System treats each byte in a source field reference (e.g., `$your_field`) as
		`X` : CHAR or BINARY
		`P` : INT
		`L` : INT
			Return type matches the type of `X`:
			* CHAR * BINARY
				`substring($orde`
				`ostcode, 1, 5)`
				`col_postcode`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	an ASCII-encoded character, not multi-byte character data. To handle multi-byte character data such as UTF-8 in a source field, you must cast `X` as `CHAR($your_field)` or use another function that returns `CHAR` data. `P` and `L` represent the character offset and the number of characters to retrieve from the source data.			
`TRIM(arg1[, arg2])` `BTRIM(arg1[, arg2])`	Trims the leading and trailing space characters (or optionally, characters specified in the `arg2` argument) from the `arg1` argument.	`arg1` : CHAR `arg2` (optional) must be a CHAR of length 1.	CHAR	`TRIM('__testTri nction__', '_') 'testTrimFuncti '` `BTRIM(' testBtrimFuncti ') -> 'testBtrimFunct '`
`REPLACE(arg1, arg2, arg3)`	The `arg1` argument is the input string for the replace operation. The `arg2` argument is the pattern to replace or the target string. The `arg3` argument is the pattern for the replacement or the replacement string. This function replaces	`arg1` : CHAR `arg2` : CHAR `arg3` : CHAR	CHAR	`REPLACE('testRe ceFunctionXYZ', 'X', 'Z') -> 'testReplaceFun onZYZ'`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	occurrences of `arg2` in `arg1` with `arg3`.			
`LENGTH(arg1)` `CHAR_LENGTH(arg1)` `CHARACTER_LENGTH(arg1)`	These functions perform the same operation of returning the character length of the input string. `arg1` is the input string. ⚠ When you use these functions with the `SUBSTRING`, `RIGHT`, or `LEFT` function, you might experience unintended results for multi-byte characters, such as Arabic in UTF-8 if you do not cast the field reference to `CHAR`.	CHAR	INT	`LENGTH('hello world') -> 11`
`UPPER(arg1)` `UCASE(arg1)`	Converts the characters in the input string to all uppercase characters.	CHAR	CHAR	`UPPER('Hello there') -> 'HEL THERE'`
`LOWER(arg1)` `LCASE(arg1)`	Converts the characters in the input string to all lowercase characters	CHAR	CHAR	`LOWER('Obi Wan Kenobi') -> 'ob wan kenobi'`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`LOCATE(arg1, arg2[, arg3])` `POSITION(arg1, arg2[, arg3])`	Returns the index position of the first occurrence of the character value `arg1` in character value `arg2`. Optionally, you can also include `arg3` to offset the search by the specified number of spaces.	`arg1` : CHAR `arg2` : CHAR `arg3` : LONG	INT	`LOCATE('F', 'testLocateFunc n') -> 11` `POSITION('on', 'testPositionFu ion', 12) -> 19`
`LEFT(arg1, arg2)`	Returns the number of characters in the `arg1` string equal to the `arg2` integer value. If the integer is negative, the function returns all characters except the last number of characters as specified by the `arg2` value. This function treats source field references (e.g., `$your_field`) as single-byte ASCII characters by default, similar to the `SUBSTRING` function.	`arg1` : CHAR `arg2` : LONG	CHAR	`LEFT('testLeftF tion', 4) -> 'test'` `LEFT('testLeftF tion', -4) -> 'testLeftFunc'`
`RIGHT(arg1, arg2)`	Returns the number of trailing characters in the `arg1` string equal to the `arg2` integer value. If the integer is negative, the function returns all characters except the first number of characters as specified by the	`arg1` : CHAR `arg2` : LONG	CHAR	`RIGHT('green te 3) -> 'tea'` `RIGHT('coffee beans', -7) -> 'beans`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	`arg2` value. This function treats source field references (e.g., `$your_field`) as single-byte ASCII characters by default, similar to the `SUBSTRING` function.			

`TO_CHAR(value, format)`	Converts the binary value specified by `value` into a character value by performing an encode or decoding using the format value specified by `format`.	`value` — BINARY(N) or VARBINARY
		
		`format` — CHAR with these supported values:
		
		`'hex'` — Encodes into a hexadecimal string with no `0x` prefix and lowercase letters.
		
		`'base64'` — Encodes into a `base64` string with padding. This encoding is not URL-safe and does not include line breaks.
		
		`'char'` — Decodes into a string with the specified `CHAR` encoding, which can be values such as: `'utf-8'`, `'utf-16le'`, `'utf-16be'`, or `iso-8859-1`.
			CHAR	`TO_CHAR(BINARY( 6f6369656e74'), 'hex') -> '6f6369656e74'`
				
				`TO_CHAR(BINARY( 6f6369656e74'), 'utf-8') -> 'ocient'`

`HEX(value)`	Alias for `TO_CHAR(value, 'hex')`	`value` — BINARY(N) or VARBINARY	CHAR	`HEX(BINARY('0x6 69656e74')) -> '6f6369656e74'`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression

`TO_HASH(value, length, format)`	Converts the string value specified by `value` into a fixed length binary value with lengh specified by `length` by encoding or decoding using the format value specified by `format`.	`value` — CHAR
		
		`length` — INT
		
		`format` — CHAR with these supported values:
		
		`'hex'` — Decodes from a hexadecimal string with optional `0x` prefix and case-insensitivity.
		
		`'base64'` — Decodes from a `base64` string with optional padding. This decoding is not URL-safe and does not expect line breaks.
		
		`char` — Encodes into a string with the specified `CHAR` encoding, which can be values such as: `'utf-8'`, `'utf-16le'`, `'utf-16be'`, or `iso-8859-1`.
			HASH(arg2)	`TO_HASH('6f6369 e74', 6, 'hex') 0x6f6369656e74`
				
				`TO_HASH('ocient 6, 'utf-8') -> 0x6f6369656e74`

`TO_VARBINARY(value, format)`	Converts the string value specified by `value` into a variable length binary value by encoding or decoding using the format value specified by `format`.	See `TO_HASH` description.	VARBINARY	`TO_VARBINARY('6 69656e74', 'hex -> 0x6f6369656e`
				
				`TO_VARBINARY('o nt', 'utf-8') - 0x6f6369656e74`

`UNHEX(value)`	Alias for `TO_VARBINARY(valu`	`value` : CHAR	VARBINARY	`UNHEX('6f636965 4') ->`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	`e, 'hex')`			`0x6f6369656e74`

> ⚠ When you use the SUBSTRING, RIGHT, and LEFT functions with Delimited or JSON formatted data sources, the Ocient System does not treat a source field reference (e.g., `$your_field`) as multi-byte character data. Each byte is treated as a single-byte ASCII character. To handle multi-byte character data like UTF-8 in a source field, you must cast the source field using `CHAR($your_field)` or use another function that returns `CHAR` data.
>
> When you use the LENGTH, CHAR_LENGTH, and CHARACTER_LENGTH functions with the SUBSTRING, RIGHT, and LEFT functions, you might experience unintended results for multi-byte characters, such as Arabic in UTF-8 if you do not cast the source field reference to `CHAR`.

Binary Data Transformation Functions

Binary data transformation functions operate on bytes provided when you use the `BINARY` format type in a pipeline. These functions are typically used in processing mainframe data outputs.

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`FLOAT(bytes [, endianness])`	With a byte array in the `BINARY` format, parse bytes in standard IEEE® 754 `binary32` (single-precision) format.	`bytes` is a 4-byte array extracted from BINARY data. `endianness` (optional) must be a literal string. Valid values include `'big'` and `'little'`. The default value is the `ENDIANNESS` option in the pipeline definition.	FLOAT	BINARY: `FLOAT($"[5, 4]" as float_col1` `FLOAT($"[5, 4]" 'little') as float_col2`
`DOUBLE(bytes [, endianness])`	With a byte array in the `BINARY` format, parse bytes in standard IEEE 754 `binary64`	`bytes` is an 8-byte array extracted from BINARY data. `endianness` (optional) must be	DOUBLE	BINARY: `DOUBLE(FLOAT($" 8])) as double_col1` `DOUBLE($"[5, 8] 'big') as double_col2`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	(double-precision) format.	a literal string. Valid values include `'big'` and `'little'`. The default value is the `ENDIANNESS` option in the pipeline definition.		
`TINYINT(bytes [[, signed], endianness])`	Converts a 1-byte array that represents a signed or unsigned number to the `TINYINT` format. ⚠ If you do not specify the correct number of bytes or if an unsigned transform overflows a signed target column, the function throws an error.	`bytes` is a single-byte array extracted from BINARY data. `signed` is a string literal that indicates if the bytes represent a signed or unsigned `BINARY` format. Valid values include `'signed'` and `'unsigned'`. The default value is `'signed'`. `endianness` is a string literal that indicates whether the bytes are in the big or little endian format. Valid values include `'big'` and `'little'`. The default value is the `ENDIANNESS` option in the pipeline definition.	TINYINT	BINARY: `TINYINT($"[9, 1 as tiny_1` `TINYINT($"[9, 1 'unsigned') as tiny_2` `TINYINT($"[9, 1 'signed', 'little') as tiny_3`
`SMALLINT(bytes [[, signed], endianness])`	Converts a 2-byte array that represents a signed or unsigned number to the `SMALLINT` format. ⚠ The function throws an error if you do not specify the correct	`bytes` is a 2-byte array extracted from BINARY data. `signed` is a string literal that indicates if the bytes represent a signed or unsigned `BINARY` format.	SMALLINT	BINARY: `SMALLINT($"[9, 2]") as small_1` `SMALLINT($"[9, 2]", 'unsigned' as small_2` `SMALLINT($"[9, 2]", 'signed', 'little') as small_3`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
	number of bytes or if an unsigned transform overflows a signed target column.	Valid values include `'signed'` and `'unsigned'`. The default value is `'signed'`. `endianness` is a string literal that indicates whether the bytes are big or little in endian format. Valid values include `'big'` and `'little'`. The default value is the `ENDIANNESS` option in the pipeline definition.		
`INT(bytes [[, signed], endianness])`	Converts a 4-byte array that represents a signed or unsigned number to the `INT` format. ⚠ The function throws an error if you do not specify the correct number of bytes or if an unsigned transform overflows a signed target column.	`bytes` is a 4-byte array extracted from BINARY data. `signed` is a string literal that indicates if the bytes represent a signed or unsigned `BINARY` format. Valid values include `'signed'` and `'unsigned'`. The default value is `'signed'`. `endianness` is a string literal that indicates whether the bytes are big or little endian format. Valid values include `'big'` and `'little'`. The default value is the `ENDIANNESS` option in the pipeline definition.	INT	BINARY: `INT($"[9, 4]") int_1` `INT($"[9, 4]", 'unsigned') as int_2` `INT($"[9, 4]", 'signed', 'little') as int_3`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression

`BIGINT(bytes [[, signed], endianness])`	Converts an 8-byte array that represents a signed or unsigned number to the `BIGINT` format. ⚠ The function throws an error if you do not specify the correct number of bytes or if an unsigned transform overflows a signed target column.	`bytes` is an 8-byte array extracted from BINARY data.
		
		`signed` is a string literal that indicates if the bytes represent a signed or unsigned binary format. Valid values include `'signed'` and `'unsigned'`. The default value is `'signed'`.
		
		`endianness` is a string literal that indicates whether the bytes are big or little endian format. Valid values include `'big'` and `'little'`. The default value is the `ENDIANNESS` option in the pipeline definition.
			BIGINT	BINARY:
				`BIGINT($"[9, 8] as bigint_1`
				`BIGINT($"[9, 8] 'unsigned') as bigint_2`
				`BIGINT($"[9, 8] 'signed', 'little') as bigint_3`

`DECIMAL(bytes, format, precision, scale [, sign_format])`	Convert a byte array that represents packed decimal or zoned decimal BINARY data to the `DECIMAL` format. With this transformation, you can indicate the format of the BINARY data as either `'zoned'` or `'packed'`.	`bytes` is an array of bytes extracted from BINARY data.
		
		`format` is a string literal that indicates the data format. Valid values include `'packed'` (meaning the array contains decimal data in each nibble for two digits per byte) and `'zoned'` (meaning the array contains
			DECIMAL	BINARY:
				`DECIMAL($"[1, 10]", 'packed', 2) as revenue`
				`DECIMAL($"[1, 10]", 'zoned', 2, 'leading') a revenue`

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
		decimal data in each nibble for one digit per byte). `precision` is the total number of significant digits. `scale` is the number of digits after the decimal point. `sign_format` is a string literal that indicates how the transformation should extract sign information from the bytes. Valid values include `'leading'` (means that the system stores the sign in the upper nibble of the first byte for both `'zoned'` and `'packed'` data formats), `'leading_separate'` (means the system stores the sign as a separate new first byte from the digit data), `'trailing'` (means that the system stores the sign in the upper nibble of the last byte when the data format is set to `'zoned'` and in the lower nibble of the last byte when set to `'packed'`), or `'trailing_separat`		

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
		`e'` (means the system stores the sign as a separate new last byte from the digit data). For `'leading'` and `'trailing'` sign information, accepted sign values are `0x7`, `0xB`, and `0xD` for negative signs and `0xA`, `0xC`, `0xE`, and `0xF` for positive. The accepted sign values for `'leading_separate'` and `'trailing_separate'` sign information are `0x2D` for `-` negative signs and `0x2B` for `+` positive. The default value for `sign_format` is `'trailing'` for both `format` data formats.		

Array Data Transformation Functions

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`ARRAY_CAP(A,N)`	Restrict the length of an array to a maximum number of elements.	`A` is an array of any type. `N` is the maximum number of elements for the length of the specified array.	Array of any type	`ARRAY_CAP(A, 1000)`

Name	Description	Supported Input Type	Return Type	Example SELE Expression
`ARRAY_COMPACT(array)`	Removes NULL values from the array.	`array` is an array that contains any type of elements. The array can be single or multi-dimensional.	Array of any type	`ARRAY_COMPACT(s[])`
`ARRAY_CONTAINS(array, value)`	Returns `true` if the array contains the specified value. If the specified value is NULL, then this transform function returns NULL.	`array` is an array that contains any type of elements. The array can be single or multi-dimensional. `value` is an integer.	Boolean or NULL	`ARRAY_CONTAINS(ts[], 1)`
`ARRAY_DISTINCT(array)`	Return all unique values in an array. For details, see **ARRAY_DISTINCT**.	`array` is an array that contains any type of elements. The array can be single or multi-dimensional.	Array of any type	`ARRAY_DISTINCT(ts[])`
`ARRAY_LENGTH(array,dim)`	Returns the length of the array for the specified dimension `dim`, which starts at index 1. If you specify a dimension that is greater than the size of the array, the function returns NULL.	`array` is an array that contains any type of elements. The array can be single or multi-dimensional. `dim` is an integer.	Integer	With `$arr` containing `[[1,2,3], [4,5,6]]`, `ARRAY_LENGTH($a 2)` returns `3` the second dimension of the array.

`ARRAY_SORT(array)`
 	
`ARRAY_SORT(array,function)`
	`ARRAY_SORT(array)` sorts and returns the input array based on the natural ordering of its elements.
	
	If one of the array elements is NULL, then this transform function sorts the
		For the first syntax, `array` is an array of elements with these types: `BIGINT`, `BOOLEAN`, `BYTE`, `DATE`, `DOUBLE`, `FLOAT`, `INT`, `SMALLINT`, `UUID`, or `VARCHAR`.	Array of any type	`ARRAY_SORT($"1c[], (x int, y i -> CASE WHEN x THEN 0 WHEN INT(COALESCE(x, 1000)) < INT(COALESCE(y, 1000)) THEN -1 ELSE 1 END)` sc the array in `1d` column using th

Name	Description	Supported Input Type	Return Type	Example SELE(Expression
	NULL values to the end of the array. `ARRAY_SORT(array, function)` sorts and returns the input array based on the results of the specified Lambda function. The function should have two arguments representing two elements of the array. This function should return a negative integer, 0, or a positive integer if the first element is less than, equal to, or greater than the second element, respectively. For details, see **ARRAY_SORT**.	For the second syntax, `array` is an array that contains any type of elements. The array can be single or multi-dimensional. The `function` argument is a Lambda function with the format `(x T, y T) -> INT` or another SQL reference function.		Lambda functior with a CASE statement. The CASE statement returns 0 when x and y are equal, returns -1 when integer cast of tl non-NULL value less than the corresponding value for y, and returns 1 otherw
`ELEMENT_AT(array,I)`	Returns the element value of the array or tuple at the specified index. You can specify a negative index value as long as the absolute value of the index does not exceed the length of the array or tuple. If the index is out of bounds, the function returns NULL.	`array` is an array or tuple. `I` is a 1-based index, specified as an integer.	Any type.	`$a` accesses th array `[1,2,3]`. `ELEMENT_AT($a,1` returns `1`. `ELEMENT_AT($a,-` returns `3`. `ELEMENT_AT($a,-` returns `2`. `ELEMENT_AT($a,-` returns NULL.
`FLATTEN(array)`	Transforms an `N`-dimensional array into an `N-1`-	`array` is a multi-dimensional array, for example `T[]`	Array of any type that can be single	`FLATTEN($ints[] [])`

Name	Description	Supported Input Type	Return Type	Example SELE Expression
	dimensional array. If one of the inner array elements is NULL, this transform function returns NULL.	`[]`, that contains any type of elements.	or multi-dimensional.	
`REDUCE(array, start_value, merge_function, finish_function)`	Applies a merge function to a starting value and all elements in the array, and then reduces the array to a single value. Optionally, specify a finish function for the returned single value. The function applies the merge function to each element of the array with the start value to return a final value.	`array` is an array of any type and can be a single or multi-dimensional array. `start_value` is the starting value of any type. `merge_function` is a Lambda function or any reference function. This Lambda function has the form `(x U, y T) -> U`. `finish_function` is a Lambda function or any reference function. This Lambda function has the form `(x U) -> V`.	This function returns the value of type `V`, which can be any type, from the finish function. If you do not specify the finish function, this function returns the type `U`, which can be any type, from the merge function.	Assuming `ints[` is `INT[]` of `[1, 2, 3]`, `REDUCE($ints[], (acc int, x int > acc + x)` returns the valu `6`. Assuming `ints[` is `INT[]` of `[1, 2, 3]`, `REDUCE($ints[], (acc int, x int > acc + x, BOOLEAN)` returns the `tru` value. In this cas the finish functio is the BOOLEAN reference functi with the input va being the resulti value from the merge function: `BOOLEAN(6)`.
`TO_ARRAY_LENGTH(V, N)`	Create the specified number of copies of any JSON object in an array.	`V` is any JSON object to copy into the result array. `N` is the number of copies to make in the result array.	Array of any type	`TO_ARRAY_LENGTH 3)`

Logical Operations Transformation Functions

These transformation functions execute logical operations on the data in the data pipeline.

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`IF(X,T,F)`	Returns `T` if the expression `X` evaluates to `true`, or the function returns `F` if `X` evaluates to `false`. `IF` and `IFF` are function aliases.	Any type	Any type	`IF('$2', '1', ' AS output`
`CASE WHEN expression1 THEN result1 WHEN expression2 THEN result2 ... ELSE result END`	Returns the `result` value based on whether an expression is `true`. If none of the expressions evaluate to `true`, then the function returns the `result` value after the `ELSE` keyword.	Any type	Any type	`CASE WHEN $x TH 'True was input in' ELSE 'False was input in' END A toutput`
`NULL_IF(X, Y)`	Returns the `NULL` value if the two input arguments are equal. If the arguments are not equal, the function returns `X`.	Any type	Any type	`INT(NULL_IF($32 ''))` `INT(NULL_IF($a. , ''))`

Network Type Functions

Network type functions are also supported in loading. For example, `SUBNET(IP($9),13) AS ipv6_subnet_form1` in the `CREATE PIPELINE SELECT` SQL statement converts the input data in the column to an IPv6-mapped IPv4 address and then computes the prefix of size 13 for that address. For the supported network type functions, see **Network Type Functions**.

Other Data Transformation Functions

Other data transformation functions include the extraction of values from the JSON format and miscellaneous functions such as `COALESCE`, which are commonly used when working with data pipelines.

© 2026 Ocient, Inc.

Name	Description	Supported Input Type	Return Type	Example SELEC Expression
`COALESCE(arg1, arg2)`	Returns `arg1` if it is non-NULL; otherwise, the function returns `arg2`. `arg1` and `arg2` can be any data type, but their types must match.	Any type	Matches `arg1` and `arg2` type	`COALESCE($1, $2` `COALESCE(int($1` `int($2))` `COALESCE(bigint` `), 13)` `COALESCE(substr` `(char($1), 1, 2` `TRIM($2))`
`JSON_EXTRACT_PATH_TEXT(X, D)`	Extracts a value from a JSON string.	`X` is the JSON string for extracting a value in `CHAR` type. `D` is the key of the value to extract in `CHAR` type.	CHAR	`JSON_EXTRACT_PA` `TEXT(JSON_UNESC` `($real_field_ke` `'b') AS col_cha`
`MAP_KEYS(X)`	Returns the keys in the specified JSON string.	`X` is the JSON string.	ARRAY OF STRINGS	Input JSON string `a` : `'{"name":"John"` `"age":30,` `"car":null}'` `MAP_KEYS($a)` Output: `["age", "car",` `"name"]`
`MAP_VALUES(X)`	Returns the values in the specified JSON string.	`X` is the JSON string.	ARRAY OF STRINGS	Input JSON string `a` : `'{"name":"John"` `"age":30,` `"car":null}'` `MAP_VALUES($a)` Output: `["John", "30",` `null]`

DEFAULT Keyword

You can use the `DEFAULT` keyword to include the configured default value of a column in transformation functions.

Example

SQL

```
COALESCE($my_field, DEFAULT) as my_column
```

Lambda Functions

Lambda functions are a transformative tool in data pipelines, enabling declarative and highly flexible data transformations. By supporting per-element operations on arrays and nested data structures, lambda functions streamline data ingestion and reduce the need for post-processing. Their adoption enables you to handle complex, real-time data engineering scenarios natively.

Lambda functions enable you to apply custom logic to each element of an array during ingestion. They support advanced operations like filtering and aggregation. You can embed transformation logic directly in the pipeline definition. This embedding reduces the need for additional processing steps after the load.

Expression	Description
Lambda Function Expression (T1,...,Tn) -> U	A lambda function is a temporary function that takes 0 or more named arguments as a parenthesis-enclosed list and returns an expression. The expression can reference objects defined within the closure of the expression, which includes locally named arguments that are locally defined within an enclosing scope, field references, literals, etc. For examples, see **Special Data Pipeline Transformation Functions**.
Reference Function Expression UPCASE = UPCASE(x) -> CHAR tuple = tuple(T1,...,Tn) -> tuple<<T1,...,Tn>>	A reference function points to an existing function in the Ocient function list. Specifying the function name allows you to use that function in contexts where you would otherwise need to create a full lambda function. For examples, see **Special Data Pipeline Transformation Functions**.

User-Defined Data Pipeline Functions

You can create a user-defined data pipeline function using the `CREATE PIPELINE FUNCTION` SQL statement and Groovy. After you create data pipeline functions, you can execute them in data pipelines like any other function to transform records.

For more details, see **CREATE PIPELINE FUNCTION** and **Data Types for User-Defined Data Pipeline Functions**.

Related Links

Formatting Functions

Data Types for Data Pipelines

Data Formats for Data Pipelines

Load Metadata and File-Based Partitioned Data in Data Pipelines

Data Pipeline Load of JSON Data from Kafka

Data Pipeline Load of Parquet Data from S3

Data Pipeline Load of CSV Data from S3

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Special Data Pipeline Transformation Functions

When you load data in the Ocient® System, you can use special loading transformation functions in the `SELECT` SQL statement to transform the data on load.

EXPLODE_OUTER

`EXPLODE_OUTER` expands a one-dimensional or multidimensional array into its elements with one element per row of output from the system. The array can contain a maximum of 8,192 elements, which is the limit for the default settings.

The function takes an array of rank `N` and expands the elements, returning elements of rank `N-1`. For example, when you apply `EXPLODE_OUTER` to an array of integers, the system returns each integer in its own row. When you apply `EXPLODE_OUTER` to an array of arrays of integers, the system returns each array of integers in its own row. You can add further transforms to these elements by nesting the statement, e.g., `NANOS_TO_TIMESTAMP(EXPLODE_OUTER($int_array_field[]))`.

You can use this function with multiple columns in the same table. The system holds constant columns that are not expanded. Also, the system combines expanded columns together in array order. If multiple expanded columns have different array lengths, the system expands the record into `N` records where `N` is the maximum length. Columns with array lengths less than `N` have NULL values after the system expands each value.

Limitations

- When you use this function with multi-dimensional arrays, you cannot apply it multiple times to match the number of dimensions, for example.
- Each record must have an associated 64-bit record number, which uniquely identifies a row within its partition. Configure the format of this number using the `RECORD_NUMBER_FORMAT` advanced pipeline option to modify the maximum number of elements that the array to expand can contain. For details, see **Advanced Pipeline Tuning Options**.
    - The system uses record numbers for deduplication, so changing the `RECORD_NUMBER_FORMAT` option of a pipeline can change its deduplication behavior.

© 2026 Ocient, Inc.

Always create a new pipeline when you change this option to make rows deduplicate properly.

- To make rows deduplicate properly when you do not set this option, always create a new pipeline when adding an `EXPLODE_OUTER` statement to your pipeline or removing all such statements from your pipeline.

Syntax

SQL

```
EXPLODE_OUTER(array)
```

Argument	Data Type	Description
`array`	Array	An array to expand.

Example

Create the `test_table` table to load data with three columns where the first two columns are integers, and the third is a string.

SQL

```
CREATE TABLE test_table (
    "i" INT NULL,
    "j" INT NULL,
    "v" VARCHAR(536870912) COMPRESSION DYNAMIC NULL );
```

Preview the load using this inline source data in JSON format: `{"id": 1, "int_array": [1, 2, 3], "string_array": ["a", "b"]}`. Use the `EXPLODE_OUTER` function to expand the elements of the arrays in the second and third JSON data values.

SQL

```
PREVIEW PIPELINE p1
SOURCE INLINE
    '{"id": 1, "int_array": [1, 2, 3], "string_array": ["a", "b"]}'
EXTRACT FORMAT json
INTO test_table
SELECT $"id" as i,
    EXPLODE_OUTER($"int_array"[]) as j,
    EXPLODE_OUTER($"string_array"[]) as v;
```

Output

Text

```
i j v
-----------------------------------------------------------------
1 1 a
1 2 b
1 3 NULL
Fetched 3 rows
```

The output shows that the inline array values are expanded across the individual rows.

Special Functions for Array Transformations in Pipelines

The data pipeline functionality supports special transformation functions for arrays that use lambda functions or function expressions to apply SQL operations to individual elements in an array. These functions are not supported in standard Ocient SQL queries.

The supported special functions are:

- `TRANSFORM` — Transform individual elements in an array. Apply a transformation expression to every element in the array.
- `FILTER` — Filter the elements in an array. Apply a filter function to each element in an array and return an array of elements matching the filter function.

- `ZIP_WITH` — Combines multiple arrays using a zipping function. Apply a zipping function to `N`-wise pairs of elements from `N` arrays and return an array with the results from the zipping function.

Each of these functions takes one or more arrays as input arguments and a lambda or other reference function for applying to the elements in the array. A lambda function is an anonymous function that must have a specific signature and return type for the array transformation function where you use the lambda function.

TRANSFORM

Transforms an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to transform. The second argument is a lambda expression or other reference function that defines how to transform each element.

The second argument must be of the form `(x T) -> U`. This specified form returns a value of type `U` for a specified element `x`. The lambda function specifies the type `T` of the element `x`.

The second argument can also be another reference function for a single argument function.

`TRANSFORM` returns an array of elements with the same length as the array argument. The returned array has the type returned by the lambda function. For multi-dimensional arrays, the `TRANSFORM` function applies to the outermost dimension. You can nest `TRANSFORM` operations to apply them over inner dimensions.

Syntax

SQL

```
TRANSFORM(array, function)
```

Argument	Data Type	Description
`array`	`T[]`	The array of type `T` elements can be single or multi-dimensional. The `TRANSFORM` function only iterates over the first dimension.

Argument	Data Type	Description
`function`	Lambda function `(x T) -> U` or Reference function	A lambda function (e.g., `(elem CHAR) -> LEFT(elem, 2)`) that takes an element of type `T` and returns a value of type `U`, which can be the same type as `T` or a different type, or a reference function with the correct argument list (e.g., `UPPER`).

Examples

Extract the First Initial from Each Element in an Array Using a Lambda Function

With the specified data in the pipeline source record `{ "data": { "names": ["Jane", "Charles", "Kim"] }}`, create a data pipeline using the `TRANSFORM` function in the `SELECT` SQL statement with the lambda function `(el CHAR) -> LEFT(el,1)`. This function returns the first character of each element in the input array `$data.names[]`. The example focuses on the `SELECT` statement and does not show the rest of the data pipeline definition.

SQL

```
CREATE PIPELINE
...
SELECT
    TRANSFORM($data.names[], (el CHAR) -> LEFT(el,1)) as first_initial_array
```

Output: `["J", "C", "K]`

Convert All Elements in an Array to Lowercase Using a Reference Function

With the specified data in the pipeline source record `{ "data": { "names": ["Jane", "Joe", "Jill"] }}`, create a data pipeline using the `TRANSFORM` function in the `SELECT` SQL statement with the `LOWER` reference function. This function converts the input strings in `$data.names[]` to lowercase. The example focuses on the `SELECT` statement and does not show the rest of the data pipeline definition.

SQL

```
CREATE PIPELINE
...
SELECT
    TRANSFORM($data.names[], LOWER) as lower_case_name_array
```

Output: `["jane", "joe", "jill"]`

FILTER

Filters elements in an array based on the logic in a lambda expression. The first argument is an expression that evaluates to an array that contains the elements to filter. The second argument is a lambda expression or reference function that returns `true` if the system should retain the element in the returned array.

The second argument must be of the form `(x T) -> BOOL`. This form returns a specified value for the specified element `x`. The lambda function specifies the type `T` of the element `x`.

The `FILTER` function includes elements in the first argument array in the resulting array when the lambda function returns `true`. Otherwise, the function excludes the elements.

Syntax

SQL

```
FILTER(array, function)
```

Argument	Data Type	Description
`array`	`T[]`	The array of type `T` elements can be single or multi-dimensional. The `FILTER` function only iterates over the first dimension.
`function`	Reference function or Lambda function `(x T) -> BOOL`	A function that takes an element of type `T` and returns a `BOOL`. The function can be a lambda function (e.g., `(elem CHAR) -> length(elem) > 10`) or a function name that has the correct argument list (e.g., `UPPER`).

Example

With the specified data in the pipeline source record `{ "data": { "names": ["Jane", "Charles", "Kim"] }}`, filter the names in the input array for elements that are greater than five characters long using the Lambda function `(name CHAR) -> length(name) > 5`. The example focuses on the `SELECT` statement and does not show the rest of the data pipeline definition.

SQL

```
CREATE PIPELINE
...
SELECT
    FILTER($data.names[], (name CHAR) -> length(name) > 5) as long_names
```

*Output: * `["Charles"]`

ZIP_WITH

Merges two or more input arrays, element-wise, into a single array using the trailing combining function. If one array is shorter than the other, the function appends NULLs at the end to match the length of the longest array at the start.

Returns an `array` with elements that result from applying the function to each pair of elements from `array1` and `array2`. If the arrays are not the same length, the function pads the shorter array with NULL values. The function returns an array of length `max(length(array1), length(array2))` containing elements of type `V` as returned by the lambda function.

`function` must be of the form `(x T, y U) -> V` where `x` is the element from `array1` and `y` is the element from `array2`. The lambda function specifies the types `T` and `U`. When zipping more arrays together, use the format `(x T1, y T2, z T3) -> V` where the arguments to the lambda function align with the order of the input arrays of the `ZIP_WITH` function.

Syntax

SQL

```
ZIP_WITH(array1, array2 [, ...], function)
```

Argument	Data Type	Description
`array1`	`T1[]`	The array of type `T1` elements can be single or multi-dimensional. The `ZIP_WITH` function only iterates over the first dimension.
`array2`	`T2[]`	The array of type `T2` elements can be single or multi-dimensional. The `ZIP_WITH` function only iterates over the first dimension.
`function`	Lambda function `(x T) -> BOOL` or Reference function	A function that takes an element of type `T` and returns a `BOOL`. The function can be a lambda function (e.g., `(elem CHAR) -> length(elem) > 10`) or a function name that has the correct argument list (e.g., `UPPER`).

Example

With the specified data in the pipeline source record `{ "line_items": [{ "product_name": "suede shoes", "sku": "FT-1023-BLU" }, { "product_name": "cheap sunglasses", "sku": "ZZ-7821-K" }] }`, concatenate the product name `$line_items[].product_name` strings with the SKU `$line_items[].sku` strings as the format `"name(sku)"`. The example focuses on the `SELECT` statement and does not show the rest of the data pipeline definition.

SQL

```
CREATE PIPELINE
...
SELECT ZIP_WITH(
  $line_items[].product_name, $line_items[].sku,
  (name CHAR, sku CHAR) -> concat(name, '(', sku, ')')
)
```

*Output: * `["suede shoes (FT-1023-BLU)", "cheap sunglasses (ZZ-7821-K)"]`

Lambda Functions

A lambda function is a temporary function that takes zero or more named arguments as a parenthesis-enclosed list and returns an expression. The expression can reference objects defined within the closure of the expression, which includes locally named arguments that are locally defined within an enclosing scope, field references, literals, and so on.

Syntax

Define lambda functions using this syntax where `T1` represents an element of type `T1` and `U` represents the return type of the function.

```
SQL
(T1,...,Tn) -> U
```

When you use this function in a SQL statement, the syntax requires an argument name and the explicit designation of the type of each argument, but the return type `U` can be inferred. For example, define a lambda function that accepts two arguments of type `CHAR` and `INT` named `x` and `y`, respectively.

```
SQL
(x CHAR, y INT) -> LEFT(x, y)
```

Examples

String Manipulation Using a Lambda Function

Convert `x` to uppercase and return the first character.

```
SQL
(x CHAR) -> LEFT(UPCASE(x),1)
```

Concatenate Using a Lambda Function

Concatenate two strings (e.g., first and last name) with a space between them.

```sql
(x CHAR, y CHAR) -> CONCAT(x, ' ', y)
```

Reference Functions

A reference function points to an existing function in the Ocient function list. Specifying the function name allows you to use that function in contexts where you would otherwise need to create a full lambda function.

The signature of the function must match the argument structure and return value requirements.

Examples

The reference function `UPCASE` is equivalent to the lambda function `(x CHAR) -> UPCASE(x)` because the syntax of `UPCASE` accepts one argument of type `CHAR` and returns a value of type `CHAR` .

Use the `UPCASE` function with the `TRANSFORM` function to capitalize all names in an array. These two SQL statements are equivalent.

```
TRANSFORM($names[], UPCASE)
```

```
TRANSFORM($names[], (n CHAR) -> UPCASE(n))
```

Related Links

Data Pipelines

CREATE TABLE

PREVIEW PIPELINE

Alphabetical SQL Functions List

Categorized SQL Functions List

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Types for User-Defined Data Pipeline Functions

This description provides the Ocient® packages that support the functionality for user-defined data pipeline functions.

Classes

`com.ocient.streaming.schema.OcientColumnType`

The class that enumerates all defined column types in the Ocient System.

Enum Value
`TYPE_NONE`
`TYPE_INFERRED`
`TYPE_DELETED`
`TYPE_INT`
`TYPE_BIGINT`
`TYPE_FLOAT`
`TYPE_DOUBLE`
`TYPE_VARCHAR`
`TYPE_IPV4`
`TYPE_TIMESTAMP`
`TYPE_DATE`
`TYPE_BOOLEAN`
`TYPE_BINARY`
`TYPE_SMALLINT`
`TYPE_BYTE`
`TYPE_UUID`
`TYPE_HASH`

© 2026 Ocient, Inc.

Enum Value
`TYPE_IP`
`TYPE_ST_POINT`
`TYPE_TIME`
`TYPE_DECIMAL`
`TYPE_ARRAY`
`TYPE_TUPLE`
`TYPE_ST_LINESTRING`
`TYPE_ST_POLYGON`

`com.ocient.streaming.schema.OcientColumnTypeArguments`

The class that describes arguments of a column type. Not all column types require arguments.

Modifier And Type	Field and Description
`public static final OcientColumnTypeArguments`	`NONE` — A default instance indicating no arguments are necessary for a column type.

`com.ocient.streaming.schema.OcientColumnTypeArguments.Decimal`

A subclass of `OcientColumnTypeArguments` that includes information for `TYPE_DECIMAL`.

Modifier And Type	Field and Description
`public int`	`precision` — The number of digits in a number.
`public int`	`scale` — The number of digits to the right of the decimal point in a number.
`public boolean`	`fullFormat` — True if the serialized format includes one byte for precision and one byte for scale. You should always set this to `true`.

Constructor

`public Decimal(int precision, int scale, boolean fullFormat)` Constructs a newly allocated `OcientColumnTypeArguments` for `OcientColumnType.TYPE_DECIMAL`.

`com.ocient.streaming.schema.OcientColumnTypeArguments.Hash`

A subclass of `OcientColumnTypeArguments` that includes information for `TYPE_HASH`.

Modifier And Type	Field and Description
`public int`	`length` — The number of bytes in the hash value.

Constructor

`public Hash(int length)` — Constructs a newly allocated `OcientColumnTypeArguments` for `OcientColumnType.TYPE_HASH`.

`com.ocient.streaming.schema.OcientColumnTypeArguments.Array`

A subclass of `OcientColumnTypeArguments` that includes information for `TYPE_ARRAY`.

Modifier And Type	Field and Description
`public OcientColumnType`	`elementType` — The array element type.
`public OcientColumnTypeArguments`	`elementTypeArguments` — Type arguments for the array element type.

Constructor

`public Array(OcientColumnType elementType)` — Constructs a newly allocated `OcientColumnTypeArguments` for an array of `elementType`. The `elementTypeArguments` field initializes to `OcientColumnTypeArguments.NONE`.
`public Array(OcientColumnType elementType, OcientColumnTypeArguments elementTypeArguments, int offsetSizeOverride)` — Constructs a newly allocated `OcientColumnTypeArguments` for an array of `elementType`. You should set `offsetOverride` to `0`.

com.ocient.streaming.schema.OcientColumnTypeArguments.TopLevelTuple

A subclass of tuple that includes information for top-level tuples.

Modifier And Type	Field and Description
`private List<OcientColumn>`	`elementOcientColumns` — List of tuple element types.

Modifier And Type	Method and Description
`public List<OcientColumnTypeArguments>`	`getTupleTypeArguments()` — Returns the list of tuple element type arguments.
`public List<OcientColumnType>`	`getTupleTypes()` — Returns the list of tuple element types.
`public List<OcientColumn>`	`getElementOcientColumns()` — Returns the list of tuple element types paired with their arguments.

Constructor

`public TopLevelTuple(List<OcientColumn> elementOcientColumns)` — Constructs a newly allocated `OcientColumnTypeArguments` for a `OcientColumnType.TYPE_TUPLE`.

com.ocient.streaming.schema.OcientColumn

Represents an Ocient column with information parsed from metadata of the Ocient System.

Modifier And Type	Method and Description
`public String`	`name()` — The name of the column.
`public OcientColumnType`	`type()` — The data type of the column.
`public OcientColumnTypeArguments`	`typeArguments()` — Some column types, for example `Decimal`, have arguments that are required to fully understand their format. A relevant subclass of `OcientColumnTypeArguments` returns, as appropriate, or `OcientColumnTypeArguments.NONE`.
`public boolean`	`required()` — Returns `true` if the user is required to provide this column. This is generally the case for non-defaulted, non-nullable columns.

© 2026 Ocient, Inc.

Modifier And Type	Method and Description
`public boolean`	`nullable()` — Returns true if this column is configured as nullable, false otherwise.

`com.ocient.streaming.data.types.gis.Coordinate`

Represents a two-dimensional cartesian coordinate.

Modifier And Type	Field and Description
`public double`	`x` — The x coordinate of the point.
`public double`	`y` — The y coordinate of the point.

Modifier And Type	Method and Description
`public double`	`x()` — Returns the x coordinate of this point.
`public double`	`y()` — Returns the y coordinate of this point.

Constructor

`public Coordinate(double x, double y)` — Constructs a newly allocated Coordinate.

`com.ocient.streaming.data.types.gis.STPoint`

Represents a two-dimensional cartesian coordinate.

Modifier And Type	Field and Description
`public Coordinate`	`coordinate` — The ordered, x and y coordinate pair.

Modifier And Type	Method and Description
`public static STPoint`	`fromWKT(String text)` — Parses an `STPoint` from the WKT text.
`public static STPoint`	`fromEWKT(String text)` — Parses an `STPoint` from the EWKT text.

Modifier And Type	Method and Description
`public double`	`getX()` — Returns the x coordinate of this point.
`public double`	`getY()` — Returns the y coordinate of this point.
`public STLinestring`	`toLinestring()` — Promotes the `STPoint` to an `STLinestring` container. Useful when representing a geometry collection consisting of `STLinestring` and `STPoint` elements.
`public STPolygon`	`toPolygon()` — Promotes the `STPoint` to an `STPolygon` container. Useful when representing a geometry collection consisting of any geospatial element type.

Constructor

`public STPoint(Coordinate coordinate)` — Constructs a newly allocated `STPoint`.

`com.ocient.streaming.data.types.gis.STLinestring`

Geometric type that is composed of a sequence of N points.

Modifier And Type	Field and Description
`public STPoint[]`	`points` — The ordered sequence of points in the linestring.

Modifier And Type	Method and Description
`public static STLinestring`	`fromWKT(String text)` — Parses an `STLinestring` from the WKT text.
`public static STLinestring`	`fromEWKT(String text)` — Parses an `STLinestring` from the EWKT text.
`public STPoint[]`	`points()` — Returns the points of this linestring.
`public STLinestring`	`toLinestring()` — Returns `this`.
`public STPolygon`	`toPolygon()` — Promotes the `STLinestring` to an `STPolygon` container. Useful when representing a geometry collection consisting of any geospatial element type.

© 2026 Ocient, Inc.

Constructor

`public STLinestring(STPoint[] points)` — Constructs a newly allocated `STLinestring`.

com.ocient.streaming.data.types.gis.STPolygon

Geometric type composed of N closed linestrings.

Modifier And Type	Field and Description
`public STLinestring`	`exteriorRing` — The exterior ring of the polygon.
`public STLinestring[]`	`interiorRings` — The interior rings of the polygon.

Modifier And Type	Method and Description
`public static STLinestring`	`fromWKT(String text)` — Parses an `STPolygon` from the WKT text.
`public static STLinestring`	`fromEWKT(String text)` — Parses an `STPolygon` from the EWKT text.
`public STLinestring`	`exteriorRing()` — Returns the exterior linestring.
`public STLinestring[]`	`interiorRings()` — Returns the interior linestrings.
`public STPolygon`	`toPolygon()` — Returns `this`.

Constructor

`public STPolygon(STLinestring exteriorRing, STLinestring[] interiorRings)` — Constructs a newly allocated `STPolygon`.

com.ocient.streaming.data.types.Timestamp

Timestamp represented as nanoseconds from the Unix® epoch.

Modifier And Type	Method and Description
`public long`	`value()` — Returns the number of nanoseconds after the Unix epoch.

Constructor

`public Timestamp(long nanosFromEpoch)` — Constructs a newly allocated `Timestamp`.

`com.ocient.streaming.data.types.Time`

Time that is represented as nanoseconds from midnight (`00:00:00`).

Modifier And Type	Method and Description
`public long`	`value()` — Returns the number of nanoseconds after midnight.

Constructor

`public Time(long nanosFromMidnight)` — Constructs a newly allocated `Time`.

`com.ocient.streaming.data.types.Decimal`

A class representing a decimal number with binary-coded decimal (BCD) storage, supporting specified precision and scale.

Modifier And Type	Field and Description
`public static long`	`MAX_PRECISION` — The maximum supported precision in the database.

Modifier And Type	Method and Description
`public static Decimal`	`fromString(String s, byte precision, byte scale)` Creates a `Decimal` object from a string representation with the specified precision and scale. • `s` — The string representation of the decimal (e.g., `"123.45"`, `"-0.1"`). • `precision` — The precision (total number of digits) of the decimal. • `scale` — The scale (number of digits after the decimal point) of the decimal.
`public static Decimal`	`fromBigDecimal(java.math.BigDecimal d, byte precision, byte scale)` Creates a `Decimal` object from a `BigDecimal` , ensuring compatibility with

Modifier And Type	Method and Description
	the specified precision and scale. • `d` — The `BigDecimal` to convert. • `precision` — The precision (total number of digits) of the decimal. • `scale` — The scale (number of digits after the decimal point) of the decimal.
`public static Decimal`	`fromDouble(double d, byte precision, byte scale)` Creates a `Decimal` object from a double value with the specified precision and scale. • `d` — The double to convert. • `precision` — The precision (total number of digits) of the decimal. • `scale` — The scale (number of digits after the decimal point) of the decimal.
`public String`	`toString()` — Returns the `Decimal` string representation of this `Decimal`.
`public long`	`asLong()` — Converts this `Decimal` object to a long, truncating any fractional part.
`public double`	`asDouble()` — Converts this `Decimal` object to a double, preserving the full value including the fractional part.
`public byte[]`	`bcd()` — Returns the BCD representation of this `Decimal` object.
`public byte[]`	`getFullFormat()` — Returns the full format representation, including precision and scale bytes followed by the BCD data padded to `MAX_PRECISION`.
`public int`	`precision()` — Returns the precision of this `Decimal` object.
`public int`	`scale()` — Returns the scale of this `Decimal` object.

`com.ocient.streaming.data.types.OcientTuple`

A class representing a tuple with a defined schema of column types and associated values. Supports fixed-length and variable-length elements.

Modifier And Type	Method and Description
`public int`	`numValues()` — Returns the number of values currently stored in this tuple.
`public void`	`setValue(int index, Object value)` Sets or updates the value at the specified index, ensuring type compatibility.

Modifier And Type	Method and Description
	• `index` — The index of the value to set (0-based). • `value` — The value to set must match the column type at the index.
`public void`	`set(List<Object> values)` Sets or resets all values in the tuple from the provided list. • `values` — The list of values to set must match the schema size and types.
`public String`	`toString()` — Returns a string representation of this tuple, including its schema and values (e.g., `"TUPLE<INT,DOUBLE>(1,2.0)"`).
`public List<Object>`	`get()` — Returns the list of values in this tuple, in order.
`public List<OcientColumnType>`	`getTypes()` — Returns the list of column types defining the schema of this tuple, in order.
`public List<OcientColumnTypeArguments>`	`getTypeArgs()` — Returns the list of type arguments corresponding to the column types of this tuple, in order.

Constructor

`public OcientTuple(List<OcientColumnType> types, List<OcientColumnTypeArguments> typeArgs)`
Constructs an empty tuple with the specified schema of column types and type arguments.

- `types` — The list of column types defining the tuple schema.
- `typeArgs` — The list of type arguments corresponding to each column type.

Related Links

CREATE OR REPLACE PIPELINE FUNCTION

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Load Geospatial Data in Data Pipelines

Data pipeline loading supports the load of geospatial data for use with OcientGeo™ functionality in the OcientAIQ™ Unified Data Platform. Use geospatial functions to manipulate this data. For details, see **Geospatial Functions**.

Supported Geospatial Functions

These geospatial functions are supported for data pipeline loading:

- **ST_FORCECCW**
- **ST_POINT** or **ST_MAKEPOINT**
- **ST_POINTFROMTEXT**
- **ST_POINTFROMEWKT**
- **ST_LINESTRING**
- **ST_LINEFROMTEXT**
- **ST_MAKELINE**
- **ST_LINEFROMEWKT**
- **ST_POLYGON(char)**
- **ST_POLYGON(geo)**
- **ST_POLYGON** or **ST_MAKEPOLYGON**
- **ST_POLYGONFROMTEXT**
- **ST_POLYGONFROMEWKT**

Data Pipeline Loading of Geospatial Data Considerations

Auto-Casting Behavior of Source Data Types

The Ocient® System automatically casts geospatial data from the `CHAR` type to `POINT`, `LINESTRING`, or `POLYGON` types using the **ST_POINTFROMTEXT**,

ST_LINEFROMTEXT, or **ST_POLYGONFROMTEXT** functions, respectively. The data must be in WKT format for the casting to work. If your data is not in the WKT format, use one of the supported geospatial functions to transform your data to the required type.

Point Data Normalization During Data Pipeline Loading

During loading, the Ocient System automatically performs normalization of point data into a regular format used within Ocient. The Ocient System performs these operations on point data during the load:

- Constrain longitude to [-180, 180] and latitude to [-90, 90].
- Wrap around invalid coordinates using correct geographical handling.
- Snap points near the pole to the pole.
- Set the longitude of points on the pole to 0.
- Remove signed zeros from coordinates, so -0 becomes 0.

Polygon Rotation Convention

The Ocient System has a standardized polygon orientation convention. The standard convention has the polygon with a counterclockwise rotation of the outer polygon ring and a clockwise rotation of the inner polygon ring. Use the **ST_FORCECCW** function to convert polygons to the standardized convention from systems that interpret polygon rotation differently.

Size Limit for Geospatial Data Types

The maximum allowed size of geospatial data types is 512 MiB.

Examples of Loading Geospatial Data

Use these examples to understand how to create a data pipeline to load geospatial data for different formats (point and WKT data).

Load Geospatial Data Using Point Data

Create the target table `cities` using the `CREATE TABLE` SQL statement with these columns:

- `id` — Integer
- `name` — Variable-length string
- `location` — Numeric point

SQL

```
CREATE TABLE cities (
    id INT,
    name VARCHAR,
    location ST_POINT
);
```

Create the `cities` data pipeline using the `CREATE PIPELINE` SQL statement for an Amazon® Web Services™ (AWS™) S3 source with bucket `ocient-docs`, filter for the folder `metabase_samples/jsonl/cities.jsonl`, and region `us-east-1`. Use the JSON format to load a JSON file into the `cities` table. Specify a numeric point using the **ST_POINT** function with the longitude and latitude coordinates.

SQL

```
CREATE PIPELINE cities
    SOURCE S3
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/jsonl/cities.jsonl'
        REGION 'us-east-1'
    EXTRACT
        FORMAT json
INTO cities
SELECT
    $id AS id,
    $name AS name,
    ST_POINT($longitude, $latitude) AS location;
```

© 2026 Ocient, Inc.

Load Geospatial Data in WKT Format

Create the target table `geospatial_table` using the `CREATE TABLE` SQL statement with these non-nullable columns:

- `ts` — TimeKey® with daily time bucket as a timestamp
- `row_number` — Variable-length string
- `point` — POINT object
- `linestring` — LINESTRING object
- `polygon` — POLYGON object

SQL

```
CREATE TABLE geospatial_table(
    ts TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
    row_number INT NOT NULL,
    point ST_POINT NOT NULL,
    linestring ST_LINESTRING NOT NULL,
    polygon ST_POLYGON NOT NULL
);
```

Create the `geospatial_pipeline` data pipeline using the `CREATE PIPELINE` SQL statement for an S3 source with bucket `ocient-docs`, filter for the folder `gis_small/gis_types/*.csv`, and region `us-east-1`. Use the CSV format to load a CSV file into the `geospatial_table` table. The file contains geospatial data in WKT format in the sixth, ninth, and twelfth columns with POLYGON, LINESTRING, and POINT data, respectively. Specify the polygon data in WKT format by using the **ST_POLYGONFROMTEXT** function and then reorient the resulting polygon counter-clockwise for the exterior and clockwise for the interior by using the **ST_FORCECCW** function.

During the load, the Ocient System automatically casts the WKT-formatted data in the three columns to the corresponding geospatial data types.

SQL

```
CREATE PIPELINE geospatial_pipeline
    SOURCE s3
        BUCKET 'ocient-docs'
        FILTER 'gis_small/gis_types/*.csv'
        REGION 'us-east-1'
    EXTRACT
        FORMAT csv
INTO geospatial_table
SELECT
    $1 AS ts,
    $2 AS row_number,
    $12 AS point,
    $9 AS linestring,
    ST_FORCECCW(ST_POLYGONFROMTEXT($6)) AS polygon;
```

Related Links

Geospatial Functions

Load Data

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Load Data with a User-Defined Data Pipeline Function

The data pipeline functionality in the Ocient® System enables you to transform data during loading. You can create your own data pipeline function to perform a custom transformation or calculation.

This example uses a user-defined data pipeline function to load weather data for Chicago. The function converts temperatures from Fahrenheit to Celsius. The formula for the conversion is `°C = (°F - 32) × 5/9`. The precipitation type data is an array with zero or more values in lowercase. The data pipeline also utilizes a lambda function to convert the first character of each precipitation type to uppercase. The pipeline uses the metadata `filename` in the data load as an audit trail.

Retrieve the Weather Data from the Source

Retrieve year-to-date weather data for Chicago in JSON format from the **Weather Query Builder**. Here is a sample row of data. The data contains:

- `queryCost` — Cost of the query
- `latitude` — Latitude of the location
- `longitude` — Longitude of the location
- `resolvedAddress` — Address of the location with the country
- `address` — City and state of the address
- `timezone` — Time zone of the location
- `tzoffset` — Time zone offset
- `name` — Name of the location
- `days` — Daily weather data in an array with these fields:
  - `datetime` — Date of the weather data
  - `datetimeEpoch` — Date epoch of the weather data
  - `tempmax` — Maximum temperature of the day

© 2026 Ocient, Inc.

- `tempmin` — Minimum temperature of the day
- `temp` — Temperature of the day
- `feelslike` — Feels like temperature of the day
- `humidity` — Humidity for the day
- `precip` — Precipitation amount for the day
- `precipprob` — Probability of precipitation for the day
- `preciptype` — Type of precipitation for the day (array of strings for types such as rain, snow, and so on)
- `windspeed` — Wind speed for the day

JSON


```
{
    "queryCost" : 212,
    "latitude" : 41.8843,
    "longitude" : -87.6324,
    "resolvedAddress" : "Chicago, IL, United States",
    "address" : "chicago, il",
    "timezone" : "America/Chicago",
    "tzoffset" : -6.0,
    "name" : "chicago, il",
    "days" : [ {
        "datetime" : "2025-01-01",
        "datetimeEpoch" : 1735711200,
        "tempmax" : 35.3,
        "tempmin" : 24.8,
        "temp" : 30.0,
        "feelslike" : 19.5,
        "humidity" : 63.9,
        "precip" : 0.037,
        "precipprob" : 100.0,
        "preciptype" : [ "rain", "snow" ],
        "windspeed" : 17.6
    } ]
}
```


Create the `chicago_weather_ytd` table with these columns:

- `latitude` — Latitude of the location
- `longitude` — Longitude of the location
- `location_name` — Name of the location
- `log_date` — Date of the weather event
- `temp` — Temperature of the day
- `preciptype` — An array of characters for multiple types of weather events
- `file_name` — Name of the file that contains the weather data

SQL

```
CREATE TABLE chicago_weather_ytd(
    "latitude" double,
    "longitude" double,
    "location_name" varchar(20),
    "log_date" date,
    "temp" double,
    "preciptype" varchar(100)[],
    "file_name" varchar(500));
```

Create a User-Defined Function for Temperature Conversion

Create a user-defined function that converts a degree value from Fahrenheit to Celsius, and round the returned value to one decimal place. Use Apache® Groovy™ code to specify the calculation and rounding.

SQL

```
CREATE PIPELINE FUNCTION f_to_c(f_value DOUBLE NOT NULL)
    LANGUAGE GROOVY
    RETURNS DOUBLE NOT NULL
    IMPORTS []
    AS $$
        return (10 * ((f_value - 32.0) * 5.0 / 9.0)).toInteger() / 10
    $$;
```

© 2026 Ocient, Inc.

Create a Data Pipeline to Load the Weather Data

Build a pipeline by using a preview of the data to load with the `PREVIEW PIPELINE` SQL statement to see how a limited number of rows loads. Preview four rows. Specify the JSON format. Transform the latitude and longitude to the double type using the `DOUBLE` cast function. For the date of the weather of the event `log_date`, expand the input array of days into its elements using the `EXPLODE_OUTER` function and convert the date to a `date` type using the `DATE` cast function. For the temperature of the day, expand the input array of days into its elements using the `EXPLODE_OUTER` function and convert the temperature to the double type using the `DOUBLE` cast function. Execute the `f_to_c` user-defined function to convert the temperature from Fahrenheit to Celsius. For the type of precipitation, expand the input array of days into its elements using the `EXPLODE_OUTER` function. Then, transform each element of the input array using a lambda function that converts the first character of the precipitation type to uppercase. Use the `METADATA` function to capture the filename.

SQL

```
PREVIEW PIPELINE chicago_weather_ytd_pipeline
SOURCE filesystem
    FILTER '/tmp/data/json_files/nested_json/Chicago_Weather_Daily_YTD.json'
    LIMIT 4
EXTRACT FORMAT json
INTO chicago_weather_ytd
SELECT DOUBLE($latitude) AS latitude,
    DOUBLE($longitude) AS longitude,
    $address AS location_name,
    DATE(EXPLODE_OUTER($days[].datetime)) AS log_date,
    f_to_c(DOUBLE(EXPLODE_OUTER($days[].temp))) AS temp,
    EXPLODE_OUTER(TRANSFORM($days[].preciptype[],
        (x VARCHAR) -> CONCAT(UPPER(SUBSTRING(x,1,1)),SUBSTRING(x,2,20)))) AS pre
    METADATA('filename') AS file_name;
```

Output

Text

```
latitude              longitude             location_name
-------------------------------------------------------------------------------
41.8843               -87.6324              chicago, il
41.8843               -87.6324              chicago, il
41.8843               -87.6324              chicago, il
41.8843               -87.6324              chicago, il
```

Create the data pipeline by replacing `PREVIEW` with the `CREATE` keyword in the SQL statement.

SQL

```
CREATE PIPELINE chicago_weather_ytd_pipeline
SOURCE filesystem
FILTER '/tmp/data/json_files/nested_json/Chicago_Weather_Daily_YTD.json'
EXTRACT FORMAT json
INTO chicago_weather_ytd
SELECT DOUBLE($latitude) AS latitude,
    DOUBLE($longitude) AS longitude,
    $address AS location_name,
    DATE(EXPLODE_OUTER($days[].datetime)) AS log_date,
    f_to_c(DOUBLE(EXPLODE_OUTER($days[].temp))) AS temp,
    EXPLODE_OUTER(TRANSFORM($days[].preciptype[],
        (x VARCHAR) -> CONCAT(UPPER(SUBSTRING(x,1,1)),SUBSTRING(x,2,20)))) AS pre
    METADATA('filename') AS file_name;
```

Execute the Data Pipeline Load

Start the data pipeline by allowing 100 errors before the pipeline fails. The data pipeline runs to completion.

SQL

```
START PIPELINE chicago_weather_ytd_pipeline ERROR LIMIT 100;
```

Use the `SHOW PIPELINE_STATUS` command to view the status of the data pipeline.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

Text

```
database_name|pipeline_name      |table_names                              |status   |
-------------+-------------------+-----------------------------------------+---------+
training     |chicago_weather_ytd|{'admin@system.chicago_weather_ytd'}|COMPLETED|
training     |city_weather       |{'admin@system.city_weather'}       |COMPLETED|
training     |counties_load      |{'admin@system.us_counties'}        |COMPLETED|
```

Display five rows of the loaded data sorted by the date of the weather event.

SQL

```
SELECT *
FROM chicago_weather_ytd
LIMIT 5
ORDER BY log_date;
```

Text

```
latitude|longitude|location_name|log_date  |temp|preciptype     |file_name
--------+---------+-------------+----------+----+---------------+----------------
 41.8843| -87.6324|chicago, il  |2025-01-01|-1.1|{'Rain','Snow'}|/tmp/data/json_f
 41.8843| -87.6324|chicago, il  |2025-01-02|-2.1|{'Rain','Snow'}|/tmp/data/json_f
 41.8843| -87.6324|chicago, il  |2025-01-03|-4.6|{'Snow'}       |/tmp/data/json_f
 41.8843| -87.6324|chicago, il  |2025-01-04|-7.2|NULL           |/tmp/data/json_f
 41.8843| -87.6324|chicago, il  |2025-01-05|-6.4|{'Snow'}       |/tmp/data/json_f
```

Related Links

Data Pipelines

User-Defined Data Pipeline Functions

Lambda Functions

Load Metadata and File-Based Partitioned Data in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Load Data from External Sources in Data Pipelines

The data pipeline functionality in the Ocient® System enables you to load data from external sources, such as other databases, using the `LOOKUP` function and the `LOOKUP` keyword of the corresponding `CREATE PIPELINE` SQL statement. You must use the function and the `LOOKUP` keyword together to join data from an external source.

You can use the `LOOKUP` function with multiple external sources by specifying each source with its `LOOKUP` function and corresponding `LOOKUP` keyword syntax.

Syntax

The `LOOKUP` function returns a value from an external source table based on a join between a specified value and another column in the source table. The function creates and executes this SQL statement from the specified function arguments.

SQL

```
SELECT return_column_name
FROM lookup_source_name
WHERE value = join_column_name;
```

The source table should only have unique values for the `join_column_name` column for the join operation.

SQL

```
LOOKUP(lookup_source_name, value, join_column_name, return_column_name)
```

Argument	Data Type	Description
`lookup_source_name`	VARCHAR	The name of the external source defined by the `LOOKUP` keyword in the `CREATE PIPELINE` SQL statement.
`value`	VARCHAR, BOOLEAN, BYTE, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DATE,	The value for the lookup.

© 2026 Ocient, Inc.

Argument	Data Type	Description
	TIME, TIMESTAMP, HASH, BINARY, DECIMAL, UUID	
`join_column_name`	VARCHAR	The name of a column in the external source table for the join operation.
`return_column_name`	VARCHAR, BOOLEAN, BYTE, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DATE, TIME, TIMESTAMP, HASH, BINARY, DECIMAL, UUID	The name of the column in the external source table that contains the data to load.

If the data type of the returned value of the `LOOKUP` function is not one of the data types specified for the `return_column_name` argument, then the function transforms the value to the nearest compatible type, typically `VARCHAR`.

Example

Create a data pipeline `lookup_data_pipeline` that loads data from the CSV file `data.csv`. Use an external source `existing_all_types` with the table `tablename` in the schema `schemaname` using a JDBC connection with the connection string: `jdbc:host://111.1.1.1:4200/databasename;user=username@databasename;password=testpassword`. For your connection string, substitute these variables with the values specific to your database and credentials:

- `host` — Hostname
- `111.1.1.1` — IP address
- `4200` — Port number
- `databasename` — Database name
- `username` — Username
- `testpassword` — Password

Look up data in the `col_binary` column of the `tablename` table by joining the second column of the CSV file to the `col_bigint` column in the `tablename` table. Load the data in the `col_binary` column based on the result of the `LOOKUP` function.

SQL

```
CREATE PIPELINE lookup_data_pipeline
SOURCE FILESYSTEM
    FILTER '/tmp/folder/data.csv'
LOOKUP existing_all_types
    CONNECTION_TYPE 'jdbc'
    CONNECTION_STRING 'jdbc:host://111.1.1.1:4200/databasename;user=username@databa
    LOOKUP_SCHEMA 'schemaname'
    LOOKUP_TABLE 'tablename'
EXTRACT
    FORMAT CSV
INTO public.destination_table
SELECT $1 AS col_pk,
    LOOKUP('existing_all_types', $2, 'col_bigint', 'col_binary') AS col_binary;
```

Performance Considerations

The performance of the lookup operation depends on the size of the external source. If the external source is large, the lookup operation is slower. The Ocient System recommends keeping the size to less than 5 million rows for the total size of all external source tables used by actively running data pipelines in the system.

Related Links

Load Data

CREATE PIPELINE

Transform Data in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Load Data into Multiple Targets in Data Pipelines

A common setup for loading files in a batch into Ocient® is to load from a bucket on Amazon® Web Services℠ (AWS℠) S3 with time-partitioned data. Often, you must perform a batch load repeatedly to load new files.

Ocient uses data pipelines to transform each document into rows in one or more different tables. The loading and transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial guides you through a load into multiple target tables using a small data set in CSV format. The data in this example comes from a test set for the Metabase℠ Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active Storage Cluster (see the **Ocient Application Configuration** guide).

Step 1: Create a New Database

Connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then, execute the `CREATE DATABASE` SQL statement for a database named `metabase`.

SQL

```
CREATE DATABASE metabase;
```

Step 2: Create Tables

Create two target tables for loading data for two different products.

© 2026 Ocient, Inc.

Create the `product129` table in the new database. First, connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), and then execute this `CREATE TABLE` SQL statement that specifies to create a table with these columns and a clustering index based on the `user_id` and `product_id` columns:

- `created_at` as a TimeKey® timestamp that is not nullable.
- `id`, `user_id`, and `product_id` as integers that are not nullable.
- `subtotal`, `tax`, `total`, and `discount` as floating point numbers.
- `quantity` as an integer.

SQL

```
CREATE TABLE public.product129 (
    created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
    id INT NOT NULL,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    subtotal DOUBLE,
    tax DOUBLE,
    total DOUBLE,
    discount DOUBLE,
    quantity INT,
    CLUSTERING INDEX idx01 (user_id, product_id)
);
```

Create the second `product161` table in the same way.

© 2026 Ocient, Inc.

SQL

```sql
CREATE TABLE public.product161 (
    created_at TIMESTAMP TIME KEY BUCKET(30, DAY) NOT NULL,
    id INT NOT NULL,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    subtotal DOUBLE,
    tax DOUBLE,
    total DOUBLE,
    discount DOUBLE,
    quantity INT,
    CLUSTERING INDEX idx01 (user_id, product_id)
);
```

The database creates the `product129` and `product161` tables, and you can begin loading data.

Step 3: Create a Data Pipeline

Create data pipelines using the `CREATE PIPELINE` SQL statement. To load data, you first create a pipeline with the definition of the source, data format, and transformation rules using a SQL-like declarative syntax. Then, you execute the `START PIPELINE` SQL statement to start the load. You can observe progress and status using system tables and views.

Each Ocient pipeline defines a single data source and the target table or tables into which data loads. A data source includes the location of the source and filters on the source to define the specific data set to load. This example loads data from a data source located in a directory within the S3 bucket.

First, inspect the data that you plan to load. Each document has a format similar to this example CSV file named `orders.csv`.

© 2026 Ocient, Inc.

Text

```
id,user_id,product_id,subtotal,tax,total,discount,created_at,quantity
1,1,14,37.65,2.07,39.72,null,2019-02-11T21:40:27.892Z,2
2,1,123,110.93,6.1,117.03,null,2018-05-15T08:04:04.580Z,3
3,1,105,52.72,2.9,49.2,6.42,2019-12-06T22:22:48.544Z,2
...
```

In this case, Ocient automatically transforms the data to the target columns using some sensible conventions. In other cases, loads require some transformation. Most transformations are identical to functions that already exist in the SQL dialect of the Ocient System.

Create a pipeline named `orders_pipeline` for the orders data set from your database connection prompt. Use the S3 data source with endpoint `https://s3.us-east-1.amazonaws.com`, bucket `ocient-docs`, and filter `metabase_samples/csv/orders.csv`. Specify the CSV format with one header line. Load the data into the two target tables. The `SELECT` part of the SQL statement maps the fields in the CSV file to the target columns in the created table.

© 2026 Ocient, Inc.

SQL

```sql
CREATE PIPELINE orders_pipeline
SOURCE
    S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/csv/orders.csv'
EXTRACT
    FORMAT csv
    NUM_HEADER_LINES 1
INTO public.product129
SELECT
    $1 as id,
    $2 as user_id,
    $3 as product_id,
    $4 as subtotal,
    $5 as tax,
    $6 as total,
    $7 as discount,
    $8 as created_at,
    $9 as quantity
WHERE $3 = 129

INTO public.product161
SELECT
    $1 as id,
    $2 as user_id,
    $3 as product_id,
    $4 as subtotal,
    $5 as tax,
    $6 as total,
    $7 as discount,
    $8 as created_at,
    $9 as quantity
WHERE $3 = 161;
```

The pipeline has three main sections:

- `SOURCE` — In this case, load data from S3. Specify the endpoint and bucket. The `FILTER` parameter identifies the file or files to load. The load uses a single CSV

file. Options exist to add wildcards or regular expressions to isolate different file sets.

- `EXTRACT` — Set the format to CSV files and note that there is one header line in the file. This specification skips that row when the Ocient System processes the file. Many other options exist for delimited data such as a record delimiter and field delimiter.
- `INTO ... SELECT` — Choose the target tables `public.product129` and `public.product161` and select the fields from the CSV file. The numeric index identifies each file field. Importantly, similar to other SQL syntax, the first field in the file is `$1`, not `$0`. Each field maps to a target column using the `as` syntax. The `WHERE` filter instructs the data pipeline to load order data for only two products, one product for each table.

After you successfully create the `orders_pipeline` pipeline, execute the `START PIPELINE` SQL statement.

SQL

```
START PIPELINE orders_pipeline;
```

Step 4: Observe the Load Progress

With your pipeline running, data immediately begins to load from the S3 files that you defined. If there are many files in each file group, the load process first sorts the files into batches, partitions them for parallel processing, and assigns them to Loader Nodes.

You can check the pipeline status and progress by querying `information_schema.pipeline_status` or by executing `SHOW PIPELINE_STATUS`.

SQL

```
SHOW PIPELINE_STATUS;
```

Output

```
SQL

database_name          pipeline_name       table_names
-------------------------------------------------------------------------------
metabase               orders_pipeline     ["public.product129" "public.product161
```

After the status of the pipeline changes to `COMPLETED`, all data is available in the target table.

After a few seconds, the data is available for query in the `public.product129` table.

```
SQL

SELECT COUNT(*) FROM public.product129;
```

Output

```
SQL

count(*)
--------------------
93
```

The data is also available for query in the `public.product161` table.

```
SQL

SELECT COUNT(*) FROM public.product161;
```

Output

```
SQL

count(*)
--------------------
92
```

You can drop the pipeline with the `DROP PIPELINE orders_pipeline;` SQL statement. Execution of this statement leaves the data in your target table, but removes metadata about the pipeline execution from the system.

Related Links

Data Pipelines Reference

Load Delimited and CSV Data

Data Types for Data Pipelines

Transform Data in Data Pipelines

Manage Errors in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Load Metadata and File-Based Partitioned Data in Data Pipelines

The data pipeline loading infrastructure enables the loading of certain metadata values of a source record into the Ocient® System. Metadata represents information about a record that is not represented in the record data itself, e.g., the filename of the source record or the timestamp when a record was created in Apache® Kafka®. You can also load data in partitioned files using Apache® Hive™ naming standards with a filter set in the file path.

Syntaxes

The `METADATA` function can load a variety of metadata along with each record of the data pipeline, or it can use key-value pairs to return the value in the filename metadata.

Load Metadata into a Column

To load a metadata value into a column, use the `METADATA` function in the `SELECT` SQL statement and replace `key` with the key you want to load. You can use transformation functions to achieve the final data type or allow the pipeline to apply automatic casting.

SQL

```
METADATA(key)
```

Argument	Data Type	Description
`key`	string	The name of the specified metadata key. Example: `'topic'`

Retrieve a Value from Filename Metadata

To retrieve a value from a key-value pair in the filename metadata, specify the first argument as `'hive_partition'` and provide a search string for the key. In this case, the filename must follow Hive naming standards. The standard embeds field names and key-value pairs in path segments, such as `/year=2019/month=2/data.parquet`. If there are duplicate key-value pairs in the filename, the function uses the last pair. The function

© 2026 Ocient, Inc.

returns a string with the value of the associated key, and returns NULL if the file does not contain the search string or the filename does not follow Hive naming standards.

```sql
METADATA('hive_partition',search_string)
```

Argument	Data Type	Description
`search_string`	string	The search string to use as a key for searching the filename metadata to return the value from the key-value pair as a string.

Loads a variety of header metadata along with each record of the data pipeline.

```sql
METADATA('header', header_row_index, header_col_index)
```

Argument	Data Type	Description
`header_row_index`	integer	The index, which starts at 1, of the header row to load.
`header_col_index`	integer	The index, which starts at 1, of the header column to load.

⚠ Metadata fields cannot be combined in a transformation function with other source data.

Supported Values for the Metadata Key

The metadata values you can load differ based on the source type. This table lists the available values for the metadata keys specified using the `key` argument by source type and their returned data type. Metadata key values are not case-sensitive.

Metadata Key	Source Type	Returned Value Data Type	Description
`row_id`	File Load and Kafka Load	UUID	A unique row identifier that is generated by the Ocient System during loading. The Ocient System guarantees this identifier is unique

Metadata Key	Source Type	Returned Value Data Type	Description
			for each loaded record because the generation is based on internal mechanisms that enforce record deduplication.
`filename`	File Load	VARCHAR	The filename and full file path indicate the source of the record.
`file_modified_time stamp`	File Load	TIMESTAMP	The timestamp that indicates when the file was last modified on the data source.
`header`	Delimited File Load	VARCHAR	Returns the string representation of the header field for the specified header row and column. Returns NULL if any of these conditions exist: The specified field is not present. The specified field is empty, and the `EMPTY_FIELD_AS_NULL` delimited extract option is set to `true`. The specified field matches one of the strings in the `NULL_STRINGS` delimited extract option.
`key`	Kafka Load	VARBINARY	The bytes of the Kafka key for the record.
`line_number`	File Load	BIGINT	The line number in the file for the source of the record.
`offset`	Kafka Load	BIGINT	The Kafka offset of the record.
`partition`	Kafka Load	BIGINT	The name of the Kafka partition that indicates the source of the record.
`record`	File and Kafka Load	VARCHAR	The string representation of the full record. For delimited, JSON, and binary formats, the system represents the record in the same way as the source. For XML, ASN.1, and Apache® Parquet™ formats, the system represents the record as a JSON object.

Metadata Key	Source Type	Returned Value Data Type	Description
`record_timestamp`	Kafka Load	TIMESTAMP	The timestamp that indicates when the record was created in the Kafka Broker.
`source_record_id`	File and Kafka Load	UUID	A unique source row identifier that is generated by the Ocient System during loading. The Ocient System guarantees this unique identifier for each source record with a specific SOURCE and EXTRACT configuration because the generation is based on source names, source record numbers, and source record offsets.
`topic`	Kafka Load	VARCHAR	The name of the Kafka topic that indicates the source of the record.

Examples

Load Filename Metadata

This example snippet uses the `CREATE PIPELINE` SQL statement to load data and the source filename with each record in a pipeline. For details about this pipeline definition and setup, see **Data Pipeline Load of CSV Data from S3**.

SQL

```
CREATE PIPELINE orders_pipeline
SOURCE
    S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/csv/orders.csv'
EXTRACT
    FORMAT csv
    NUM_HEADER_LINES 1
INTO public.orders
SELECT
    $1 as id,
    $2 as user_id,
    $3 as product_id,
    $4 as subtotal,
    $5 as tax,
    $6 as total,
    $7 as discount,
    $8 as created_at,
    $9 as quantity,
    METADATA('filename') AS filename_column;
```

Load Hive-Style Partitioned Data

This example uses the `CREATE PIPELINE` SQL statement to load data and the value of the search string `'year'` of the filename with each record in a pipeline. The Amazon® Web Services℠ (AWS℠) S3 FILTER option contains the filter for the year in a file path. In turn, the `METADATA` function uses the filter `'year'` value to load all records from files with the year `2019` in the specified path. For details about this pipeline definition and setup, see **Data Pipeline Load of CSV Data from S3**. The example assumes the filename follows Hive-style naming standards.

SQL

```sql
CREATE PIPELINE orders_pipeline
SOURCE
    S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER '/year=2019/**'
EXTRACT
    FORMAT csv
    NUM_HEADER_LINES 1
INTO public.orders
SELECT
    $1 as id,
    $2 as user_id,
    $3 as product_id,
    $4 as subtotal,
    $5 as tax,
    $6 as total,
    $7 as discount,
    $8 as created_at,
    $9 as quantity,
    METADATA('hive_partition','year') AS hive;
```

Related Links

Load Data

CREATE PIPELINE

Transform Data in Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Deduplication in Data Pipelines

Exactly Once Loading in Pipelines

The job of a pipeline is to load the data assigned to it exactly once. A pipeline maintains its own position during a load and enforces deduplication logic. The lifecycle of a pipeline object in the database defines both the deduplication logic and load position.

> Because the pipeline, not the target tables, defines the load position and deduplication scope, if the system truncates target tables or if the system drops and recreates a target table, the pipeline continues from its last position. It does not reset to the beginning position of the load if you attempt to load data into a new table.

Key benefits are:

- Pipeline Events — During the life of a pipeline, you can stop and resume the pipeline without duplicating source data or changing the position in a load.
- Pipeline Updates — You can modify pipelines using the `CREATE OR REPLACE` SQL statement to update the transforms in a pipeline but maintain the current position in the load.
- Exactly Once Loading — In the event of transient system failures, the pipeline ensures exactly-once loading into the target table through replay and deduplication guarantees.

Two concepts are critical to understand how Ocient® controls which data is loaded in a pipeline and how Ocient pipelines ensure exactly-once loading.

1. Load Position
2. Deduplication Scope

Load Position

You can control the pipeline by starting, stopping, modifying, and resuming the pipeline. While running, each pipeline maintains its position in the overall load. When it

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

has reached the end of all data, the pipeline completes and does not attempt to reload the data. To load the data again, you must create a new pipeline or drop and recreate the pipeline.

- For an Apache® Kafka®-based load, the position consists of using consumer group offsets appropriately to create checkpoints.
- For File-based loads, file details are stored in the `sys.pipeline_files` system catalog table and the system updates the status of each file as the load progresses.

In both cases, the load position defines where the pipeline resumes loading if stopped and resumed.

Deduplication Scope

The deduplication scope defines the conditions under which a pipeline does not cause the same row to appear twice in a target table. This deduplication is how a pipeline ensures that it is safe to replay data during loading. This situation is common in failover situations or when you stop and restart a pipeline.

The deduplication scope is the unique combination of:

1. A pipeline object
2. A target table

A pipeline guarantees that if it sends the same row to the same target table twice, the system only loads it into the table once. If you drop and recreate either the pipeline or the table, then the situation is a new deduplication scope.

> ⚠ If you begin sending data to a new target table, this situation is a new deduplication scope. A row for the same record can appear in both the old and new tables.

Restarts and Deduplication

You can restart a pipeline without creating duplicate data in the target tables. However, there are some limitations and key assumptions for each data source.

For example, in a file load, if you modify the contents of a file after the pipeline has started, then you can experience duplication of data or missed rows.

For more details and key considerations, see **Restarting a File Load** and **Restarting a Kafka Load**.

Deduplication and DROP TABLE

You can drop a table using the `DROP TABLE` SQL statement and then recreate it with the same name. In this case, an existing pipeline loads into the new table. When you drop a table and recreate it, these actions do not update the load position of the pipeline. When you restart the pipeline, it resumes from the load position of the pipeline, and all records from that position forward are loaded with a new deduplication scope.

Deduplication and TRUNCATE TABLE

You can truncate a table (using the `TRUNCATE TABLE` SQL statement) that is the target of a pipeline whether the pipeline is running or stopped. These actions do not update the load position of the pipeline or change the deduplication scope. The system deduplicates any data that was loaded before the truncate and replays after the truncate for any reason (e.g., restart or transient failure). Also, the system loads all new data.

> When you truncate the target tables and restart the pipeline, the Ocient System does not reload the data.

Load Duplicate Data

Sometimes, you might want to load the same data multiple times, but the pipeline load position and deduplication scope prevent this.

If you want to load a second copy of the source data, you can follow one of these approaches:

File-Based Pipelines

- Drop and recreate the pipeline to reset the `sys.pipeline_files` system catalog table.
- Create a second pipeline with a new name and the same configuration.

Kafka-Based Pipelines

- Drop the pipeline and recreate it with the same name, which defaults to the same consumer `group.id` but has a new deduplication scope. Then, reset the consumer group offsets using Kafka tools to the chosen starting point.
- Create a new pipeline with a different name and load from the beginning of the topic using `auto.offset.reset`.

Related Links

Data Pipelines DDL Reference

START PIPELINE

Restart with File Loading

Restart with Kafka Loading

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Error Tolerance in Data Pipelines

Error Limits

The error limit for SQL statements such as `START PIPELINE my_pipeline ERROR LIMIT 100` defines the number of record-level errors that a batch of files tolerates before failing the entire pipeline.

The error process is:

- Periodically, the Ocient® System assigns a batch of pending files to a Loader Node.
- The system enforces the maximum error limit for each batch.
- Files that have at least one record-level error reach the terminal status `LOADED_WITH_ERRORS` after all processing is complete.
- If the file batch reaches the error limit, the current file reaches the status `FAILED`, and then the pipeline status becomes `FAILED`. Processing stops on the pipeline.

By default, `BATCH` pipelines run with `ERROR LIMIT 0`. This permits zero errors on a pipeline. This setting stops the pipeline on the first error and stores the error in `sys.pipeline_errors.` When you restart the pipeline, it retries the failed file and failed record. The system does not duplicate any previously loaded data. To complete the load, fix the issue with the data or the pipeline definition or increase the error limit when you restart the pipeline.

Unrecoverable File Errors

You can manage unrecoverable file errors such as Gzip decompression errors, tokenization errors, or missing files by starting the pipeline with the `FILE_ERROR` option.

The strictest setting is `FILE_ERROR FAIL`. When you use the `FAIL` setting, any file-level error causes the pipeline to fail. If any file is missing or cannot be processed, the pipeline marks the file `FAILED`, sets the pipeline status to `FAILED`, and processing stops on the pipeline.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

The most tolerant setting is `FILE_ERROR TOLERATE`. Use this setting in a statement such as `START PIPELINE my_pipeline ERROR FILE_ERROR TOLERATE`. When you use the `TOLERATE` setting, missing files reach the terminal status `SKIPPED`. Files that encounter other file-level errors reach the terminal status `LOADED_WITH_ERRORS`. When the Ocient System encounters a file-level error, processing stops on the file and continues with the next file in the partition. When you use the `TOLERATE` setting, the pipeline automatically executes with an unlimited error limit.

Recoverability

If a pipeline fails when it tolerates no errors, manually fix the row or file where the error occurs and restart the pipeline so that the load proceeds with all records correctly deduplicated. Restarting the pipeline without fixing the error but with tolerance of record or file-level errors also allows the load to proceed with all records correctly deduplicated. However, restarts do not guarantee correct deduplication if you apply manual fixes after this point.

When a pipeline fails, the number of loaded rows is nondeterministic. As a result, the Ocient System does not guarantee the reflection of any manual modification of files with the statuses `SKIPPED`, `LOADED_WITH_ERRORS`, or `FAILED` in a restart operation. However, if the pipeline fails due to a low error limit, you can raise the error limit on a restart operation to allow the pipeline to make further progress.

You can use the `BAD_DATA_TARGET` option to capture failing records for troubleshooting and reloading.

Restarts and Deduplication

Ocient pipelines enable the restart of a pipeline without creating duplicate data in the target tables. However, there are some limitations for each type of data source. For details, see **Data Pipelines**.

Related Links

Data Pipelines DDL Reference

START PIPELINE

Data Pipeline Load of JSON Data from Kafka

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Monitor Data Pipelines

You can monitor data pipelines through a combination of:

1. System catalog tables
2. Information schema views
3. Operational metrics endpoints for use with operations monitoring tools

This tutorial describes how to use these system catalog objects to monitor your pipelines:

- `information_schema.pipelines`
- `information_schema.pipeline_status`
- `sys.pipelines`
- `sys.pipeline_events`
- `sys.pipeline_errors`
- `sys.pipeline_files`
- `sys.pipeline_partitions`
- `sys.pipeline_metrics`
- `sys.pipeline_metrics_info`

A complete reference of system catalog tables and views is available in the **System Catalog**.

Pipelines Information Schema

After you create a pipeline, two convenient views are most useful for observing the pipeline. At a glance, you can view the configuration of your pipeline in `information_schema.pipelines` and the status information in `information_schema.pipeline_status`.

Access these views conveniently with the `SHOW` command.

© 2026 Ocient, Inc.

SQL

```
> SHOW PIPELINES;
+-----------------+-----------------+----------------+--------------+-------------
| database_name   | pipeline_name   | loading_mode   | source_type  | data_format
|-----------------+-----------------+----------------+--------------+-------------
| test            | my_pipeline     | BATCH          | FILESYSTEM   | CSV
+-----------------+-----------------+----------------+--------------+-------------
Fetched 1 rows
```

SQL

```
> SHOW PIPELINE_STATUS;
+-----------------+---------------+---------------------------+-----------+------
| database_name   | pipeline_name | table_names               | status    | statu
|-----------------+---------------+---------------------------+-----------+------
| test_database   | test_pipeline | ['test_schema.test_table'] | COMPLETED | Compl
+-----------------+---------------+---------------------------+-----------+------
Fetched 1 rows
```

For a quick look at the progress of a pipeline, the `SHOW PIPELINE_STATUS` command provides:

- Name
- Target tables
- Current status
- Last status message that was received in processing
- Percentage completion (from 0.0 - 1.0)
- Duration of pipeline execution
- Number of files and records processed
- Error counts

To join these two views to other catalog tables, use the `pipeline_name` to join to the `sys.pipelines` table and then join tables using the `pipeline_id`.

Pipeline Catalog Tables

© 2026 Ocient, Inc.

Beyond the information schema views, many catalog tables offer details about your pipelines.

Table	Description	Importance
`sys.pipelines`	List of pipelines that exist and the configuration information.	High
`sys.pipeline_events`	Event log generated through the lifecycle of a pipeline.	High
`sys.pipeline_errors`	Log of errors created by pipelines during execution for troubleshooting record-level errors or system errors.	High
`sys.pipeline_files`	List of files that each file-based pipeline is loading or has loaded.	High
`sys.pipeline_partitions`	List of Apache® Kafka® source partitions on each Kafka pipeline with relevant metadata for offsets.	Medium
`sys.pipeline_metrics`	Metrics that are regularly collected for each pipeline to capture performance and progress.	Medium
`sys.pipeline_metrics_info`	Descriptive details about the available pipeline metrics. Provides information about how to interpret each metric type.	Low

Monitor Activity and Events

While your pipeline is running, the pipeline generates events in the `sys.pipeline_events` system catalog table to mark significant checkpoints when something has occurred. You can use this table to observe progress and look for detailed messages about changes in the pipeline.

For example, the `sys.pipeline_events` table captures events when a pipeline is created, started, stopped, fails, and completed. This table also captures relevant system events such as rebalancing Kafka partitions or retrying a process due to a transient failure.

Pipeline events include messages from many different tasks. These tasks are the background processes that execute across different Loader Nodes during pipeline operation.

Monitor Files or Partitions

For file-based loads, the `sys.pipeline_files` system catalog table contains one row for each file, which contains the status of the file and other file metadata such as filename, creation or modified timestamps, and the file size. The pipeline updates this list of files when it is started and maintains status information as it processes files. You can use this list to understand which files are loading, which succeeded, and which failed.

For Kafka partition-based loads, the `sys.pipeline_partitions` system catalog table contains one row for each partition, which contains the current offsets and record counts. You can use this information to understand how partitions are distributed across Loader Nodes and also to observe lag on each topic and partition.

Monitor Performance

Pipeline performance metrics are captured in the `sys.pipeline_metrics` system catalog table. This table contains samples of the metrics over time. The Ocient® System collects samples regularly. You can query the samples using standard SQL queries to inspect the behavior of a single pipeline, a single task on a Loader Node, or across all pipelines. The `sys.pipeline_metrics_info` system catalog table provides metadata that explains the individual metric types.

See **Discover Insights from Data Pipeline System Catalog Tables** for a detailed description of how to use `sys.pipeline_metrics` and `sys.pipeline_metric_info` to monitor the performance of your pipelines. For the system catalog table definitions, see **System Catalog**.

Monitor a Foreground Pipeline from Another Session

When you create a transactional pipeline with the `START FOREGROUND` clause, the SQL session that executes the statement blocks until the pipeline reaches an inactive state. Because the session is blocked, you must monitor the pipeline from a separate SQL session.

From a separate session, execute the `SHOW PIPELINE_STATUS` command to view the current status, last status message, percentage completion, duration, and file and record counts for the pipeline.

```SQL
SHOW PIPELINE_STATUS;
```

You can also query the `sys.pipeline_metrics` system catalog table to observe the performance and progress metrics that the Ocient System collects regularly while the pipeline runs. For details about how to interpret each metric type, query the `sys.pipeline_metrics_info` system catalog table.

For more information about foreground pipelines, see **Transactional Data Pipelines**.

Metrics Endpoints

In the same way that the Ocient System monitors system performance using **Statistics Monitoring** endpoints with operational monitoring tools, data pipelines expose an API that allows operators to capture performance metrics over time.

Access Metrics Endpoints

You query the metrics using an HTTP endpoint on each Loader Node located at `<loader_ip_address_or_hostname>:8090/metrics/v1`.

For example, you can retrieve metrics from a Loader Node by executing this command from a shell on the node or by running a similar request from a monitoring agent on the node.

```Text
curl http://localhost:8090/metrics/v1
```

Each metric is represented as a JSON object that contains:

- Metric name
- Metric value
- Scope information
- Query time

- Value units
- Whether or not the metric is incremental

Metrics can be incremental or instantaneous. Incremental metrics accumulate value over time, so they never decrease in value. Instantaneous metrics capture the current value, so they can decrease.

Metrics Scope

Similar to the `sys.pipeline_metrics` system catalog table, the Ocient System scopes metrics to a Loader Node, a pipeline task, or a partition. You can determine the scope of a metric based on the presence of the `pipeline_name`, `pipeline_name_external`, and `partition` keys or reference the details in **Pipeline Metrics Details**.

Scope	Description
Loader	Metrics scoped to a loader do not have information about pipelines or partitions. Because metrics are collected on each node, this scope contains all activity on the node where you access the endpoint.
Pipeline	Metrics scoped to a pipeline have a key named `pipeline_name` with a value that corresponds to the identifier of the extractor task with hyphens removed. The metrics also have a key named `pipeline_name_external` with a value that corresponds to the identifier of the pipeline.
Partition	Metrics scoped to a partition have the same keys as metrics scoped to a pipeline. The metrics also have a key named `partition` with a value that is the stream source identifier of the partition.

Examples

This JSON structure contains a metric scoped to a Loader Node.

JSON


```
{
  "name": "usage.os.heap",
  "time": 1713993689613976,
  "timestamp": "2024-04-24T21:21:29.613976119Z",
  "value": 317855288,
  "units": "bytes",
  "incremental": false
}
```


This JSON structure contains a metric scoped to a pipeline.

JSON


```
{
  "name": "count.record.sent",
  "time": 1713994532578624,
  "timestamp": "2024-04-24T21:35:32.578624253Z",
  "value": 100000,
  "units": "unitless",
  "incremental": true,
  "pipeline_name_external": "test-rest-external-create",
  "pipeline_name": "test"
}
```


This JSON structure contains a metric scoped to a partition.

© 2026 Ocient, Inc.

JSON


```
{
  "name": "duration.record.processed",
  "time": 1714509826724477,
  "timestamp": "2024-04-30T20:43:46.724477142Z",
  "value": 322432,
  "units": "milliseconds",
  "incremental": true,
  "pipeline_name_external": "test-rest-external-create",
  "pipeline_name": "58ff07cf732846f9830b728b3a8e4a7a",
  "partition": "226979",
  "sink_index": "0"
}
```


Pipeline Metrics Details

Name	Scope	Description	Importance
`count.error`	pipeline	The count of failures when processing records during pipeline execution for a pipeline.	High
`count.file.fail`	pipeline	The number of files that have failures.	High
`count.file.processed`	pipeline	The number of files that have been extracted from the source for processing. The Ocient System tracks this number before tokenization, transformation, or sending to the sink. This number does not indicate the durability of the files or if files load with errors.	Medium
`count.file.total`	pipeline	The total number of files in the static file list.	High
`count.partition.durable.offset.committed`	partition	The last committed offset of the Kafka partition for the consumer group.	High
`count.partition.offset.lag`	partition	`count.partition.offset.max - count.partition.durable.offset.committed` for a specified Kafka partition.	Medium

© 2026 Ocient, Inc.

Name	Scope	Description	Importance
`count.partition.offset.max`	partition	The maximum offset of the Kafka partition.	High
`count.partition.records.failed`	partition	The count of failures when processing records during pipeline execution for a specified Kafka partition.	High
`count.partition.records.processed`	partition	The number of records that have been extracted for processing on a specified Kafka partition.	Medium
`count.record.bytes.maximum`	partition	The largest record processed in bytes.	Medium
`count.record.bytes.minimum`	partition	The smallest record processed in bytes.	Medium
`count.record.bytes.transformed`	pipeline	The number of bytes that have been transformed, excluding delimiters.	Medium
`count.record.sent`	pipeline	The number of records that have been fully transformed and sent to the sink.	High
`count.sink.bytes.queuedepth`	pipeline	The number of transformed records queued to send to the sink.	Medium
`count.stream.paused`	partition	The number of times the processing of the record stream has been paused due to reaching a high watermark.	Medium
`count.stream.unpaused`	partition	The number of times the processing of the record stream has been unpaused due to reaching a low watermark.	Medium
`duration.file.listed`	pipeline	The number of milliseconds spent listing files from the data source.	High
`duration.record.durable`	partition	The duration in milliseconds for making records durable. **ℹ** The database takes time to make records durable, so this duration might be longer than the record processing duration.	Medium
`duration.record.processed`	partition	The duration in milliseconds for reading the records from the source, transforming them, and	Medium

Name	Scope	Description	Importance
		buffering them for sending to the sink.	
`uptime`	Loader Node	The number of seconds after the last restart of the extraction process.	High
`usage.os.heap`	Loader Node	The number of bytes used in the heap.	High
`usage.os.percentage.cpu`	Loader Node	The percentage of CPU used by the JVM®.	High

Related Links

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Manage Errors in Data Pipelines

When you load data into Ocient®, you can manage the way that the system responds to errors. This guide outlines different mechanisms for managing errors through examples.

The guide uses the same setup as **Data Pipeline Load of JSON Data from Kafka** but it modifies that pipeline to illustrate how to manage errors that you might encounter in data pipeline functionality.

You learn about:

- Bad data and error handling
  - Inspecting record-level processing errors
  - Configuring a bad data target
- Controlling pipeline behavior when you encounter errors
  - File-level error behavior
  - Record-level error behavior
- Understanding pipeline statuses in file loading

Step 1: Complete the Load JSON Data from Kafka Tutorial

See **Data Pipeline Load of JSON Data from Kafka** and complete the tutorial, then return to this guide for Step 2.

Step 2: Modify the Pipeline with a Bad Data Target

Bad data targets are the way to capture record-level errors for later troubleshooting and correction. Data pipelines support an Apache® Kafka® bad data target that allows you to send any failing records to a designated Kafka topic. This topic supports easy

© 2026 Ocient, Inc.

replay of those records in a separate recovery pipeline or inspection of the failing records.

For details about bad data targets, see the **Bad Data Targets Reference**.

Stop the Data Pipeline

Stop the `orders_pipeline` data pipeline using the `STOP PIPELINE` SQL statement.

SQL

```
STOP PIPELINE orders_pipeline;
```

Inspect the pipeline status using this `SELECT` SQL statement and the `pipeline_status` information schema view.

SQL

```
SELECT
  status
FROM information_schema.pipeline_status
WHERE
  pipeline_name = 'orders_pipeline';
```

While the pipeline is stopping, the status might appear as `STOPPING` while the pipeline wraps up work in process and commits offsets back to Kafka. After the status changes to `STOPPED`, you can proceed to the next step. A pipeline might not be in a running state when you modify it.

Modify the Data Pipeline

In this example, use the `CREATE PIPELINE OR REPLACE` SQL statement to update the data pipeline. This statement maintains the same offsets with the Kafka topic and ensures that data is deduplicated correctly when the pipeline resumes.

Add the `BAD_DATA_TARGET` option and specify the Kafka target and details. Use the same Kafka broker as the source and bad data target in this example.

SQL

```
CREATE PIPELINE OR REPLACE orders_pipeline
BAD_DATA_TARGET
  KAFKA
    BOOTSTRAP_SERVERS '192.0.2.1:9092'
    TOPIC 'orders_errors'
SOURCE
  KAFKA
    BOOTSTRAP_SERVERS '192.0.2.1:9092'
    TOPIC 'orders'
EXTRACT
  FORMAT json
INTO public.orders
SELECT
  $id as id,
  $user_id as user_id,
  $product_id as product_id,
  $subtotal as subtotal,
  $tax as tax,
  $total as total,
  $discount as discount,
  $created_at as created_at,
  $quantity as quantity;
```

Step 3: Start the Data Pipeline with an Error Limit

Start the `orders_pipeline` data pipeline. In this case, set a maximum error limit so that the pipeline fails if 10 errors occur. An `ERROR LIMIT` might not be chosen for a continuously streaming pipeline if it is critical to keep up with newly arriving data, because the pipeline can stop suddenly and cause data freshness to fall behind. However, this example shows how error limits behave and how error states appear in pipelines.

SQL

```
START PIPELINE orders_pipeline
ERROR LIMIT 10;
```

© 2026 Ocient, Inc.

The pipeline status changes to `STARTED` to indicate that it has begun processing.

> For file-based loads, the `START PIPELINE` SQL statement includes options for `FILE_ERRORS` that are useful for controlling error handling in different file failure modes. Options include failing on error, skipping on file error, and tolerating file-level errors. See the **Start Pipeline Command Reference** for details.

Step 4: Produce Data with Errors into Kafka

With the data pipeline in place and running, data immediately begins to load from the Kafka topics that are configured in the pipeline from the last committed offsets after the pipeline stopped.

For test purposes, **kafkacat** is a helpful utility that makes it easy to produce records in a topic. For example, if you have a file of sample data `orders_with_errors.jsonl` in a JSONL format (newline-delimited JSON records), you can run this command to send those records into your Kafka broker. `<broker_ip_address>` is the IP address of the Kafka broker and `<topic_name>` is the name of the Kafka topic.

Shell

```
kafkacat -b <broker_ip_address>:9092 -t <topic_name> -T -P -l orders_with_errors.
```

Save an example document to your file system to use for this test. For this example, you can download an example file from **https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/orders_with_errors.jsonl** and save it to `~/orders_with_errors.jsonl`.

Assume the broker is running at `192.0.2.1`, to send data into the `orders` topic defined in the pipeline definition, run this command.

Shell

```
kafkacat -b 192.0.2.1:9092 -t orders -T -P -l orders_with_errors.jsonl
```

This command pushes the entire JSONL file of messages into Kafka with one record per line. As these records are produced into Kafka, the running pipeline begins to load

them into Ocient.

Step 5: Observe Loading Progress

With your pipeline running, data immediately begins to load from the Kafka topic. The pipeline creates parallel Kafka Consumers for each partition. If there are more partitions than processing cores available, the pipeline automatically handles spreading consumers across available processing cores and Loader Nodes.

You can check the pipeline status and progress by querying the `information_schema.pipeline_status` view or running the command `SHOW PIPELINE_STATUS`. Use the view to query the name of the pipeline, status, number of failed records, and the status message.

SQL

```
SELECT
  pipeline_name, status, records_failed, status_message
FROM
  information_schema.pipeline_status;
```

Output

SQL

```
pipeline_name       status     records_failed    status_message
-------------------------------------------------------------------------------
orders_pipeline     FAILED     11                Started processing pipeline orde
```

Inspect Errors

Examine the pipeline errors directly using the `sys.pipeline_errors` system catalog table.

SQL

```
SELECT count(*) FROM sys.pipeline_errors;
```

Output

SQL

```
count(*)
--------------------
11
```

Using this table, you can see detailed error messages, the source of the error, the column that encountered the error, and other helpful details to help troubleshoot why the record did not load.

SQL

```
SELECT * from sys.pipeline_errors;
```

Output

SQL

```
pipeline_id                          extractor_task_id                    error_i
-------------------------------------------------------------------------------
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 1
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 2
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 3
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 4
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 5
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 6
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 7
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 8
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 9
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 10
f3ca5a1c-1bf4-40a4-8d01-b8c028fdfb58 1ecc5c4a-613f-48a5-8234-2257317a4a85 11
```

For a more abbreviated view, query a few columns in the table. This example queries the error type, error code, error message, and name of the column.

© 2026 Ocient, Inc.

SQL

```sql
SELECT
  error_type, error_code, error_message, column_name
FROM sys.pipeline_errors;
```

Output

SQL

```
error_type          error_code      error_message
-------------------------------------------------------------------------------
TRANSFORMATION      OLT01           Failed to transform value 'not_int' using Asc
TRANSFORMATION      OLT01           Failed to transform value '2019-08-22T:30:42.
TRANSFORMATION      OLT01           Failed to transform value 'not_decimal' using
TRANSFORMATION      OLT01           Failed to transform value '55.11' using Ascii
TRANSFORMATION      OLT01           Failed to transform value 'not_int' using Asc
TRANSFORMATION      OLS00           Failed to load transformed value 'null' to si
TRANSFORMATION      OLS00           Failed to load transformed value 'null' to si
TRANSFORMATION      OLS00           Failed to load transformed value 'null' to si
TRANSFORMATION      OLS00           Failed to load transformed value 'null' to si
TRANSFORMATION      OLT03           Failed to transform value '2018-06-26' using
TRANSFORMATION      OLS00           Failed to load transformed value 'null' to si
```

From this result set, you can quickly inspect error messages, such as the first error message.

Text

```
Failed to transform value 'not_int' using AsciiIntTransform. Details: Invalid ASC
```

This message occurs for the `user_id` column. From this information, you can see that you have some bad data in the data set.

In other errors, you can see the error code `OLS00` and a message that indicates some issues with `NULL` handling. The table has a `NOT NULL` setting on a few columns, but the data is missing on the source records. Fix this by updating the pipeline to supply a default, or you can modify the table schema if you expect NULLs to occur and you do

not have a reasonable default. For a continuous pipeline, you might expect to see an error like this if something changes in the data source.

Inspect Bad Data

To observe the original data, you can go to the original source file from this example, or you can load the specific message from the bad data topic that was configured to inspect it.

Using the kafkacat utility, you can consume the messages on the topic. This command consumes one message on the `orders_errors` topic.

Shell

```
kafkacat -C -b 192.0.2.1:9092 -t orders_errors -o -1 -c 1
```

Output

Shell

```
{"id": 1011, "user_id": 3, "subtotal": 76.83, "tax": 5.28, "total": 82.11, "disco
```

A quick inspection of the source data reveals that this record is missing the `product_id` field, but this is a required column in our table.

If you run this command again, but with the `-J` flag, you can inspect the headers that are associated with the pipeline. The headers in the bad data target capture important metadata from the data pipeline execution that you can use to troubleshoot the failing data, including the `error_message`.

Shell

```
kafkacat -C -b 192.0.2.1:9092 -t orders_errors -o -1 -c 1 -J
```

Output

SQL

```
{"topic":"orders_errors","partition":0,"offset":89,"tstype":"create","ts":1713566
```

Next Steps

Now that you have loaded data into tables, you can explore a few advanced topics, including handling pipeline errors, loading from other data sources, loading different data formats, and transforming data.

Related Links

Loading Delimited and CSV Data

Loading JSON Data

Data Types for Data Pipelines

Transform Data in Data Pipelines

Data Pipeline Reference

Data Pipeline Loading Errors

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Frequently Asked Questions for Data Pipelines

The questions featured in this topic cover information that you need most often when you start creating and running data pipelines.

Running Pipelines

Who can run a data pipeline?

Any database user with `EXECUTE` pipeline privileges can run a data pipeline on their database. This privilege allows database administrators to distinguish the users who can create pipelines from those who can start them.

To learn more about access control for pipelines, see **DCL Pipeline Privileges**.

Do pipelines run automatically after creation?

No, pipelines have a user-controlled lifecycle that starts with creation. After you create a data pipeline, you determine when to start the pipeline. A pipeline either completes successfully or moves to a failed status if the Ocient® System reaches an error limit.

For a basic batch file load, you would first use the `CREATE PIPELINE` SQL statement and then the `START PIPELINE` statement. After all data is successfully loaded, the pipeline status changes to completed.

How do you run multiple pipelines at the same time?

Multiple data pipelines can run concurrently in Ocient. To run more than one simultaneously, any user with privileges to start a pipeline can use the `START PIPELINE` SQL statement. Execution of this statement begins the execution of the selected pipeline. The Ocient System uses Loader Nodes in parallel to process all of the pipelines that are running.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

If you need to control the relative priority of one pipeline over another when running concurrently, use advanced pipeline settings to limit the number of cores allocated to a specified pipeline or isolate the pipeline to a particular set of Loader Nodes.

What happens if I start a pipeline that has been stopped?

When you execute a `START PIPELINE` SQL statement on a pipeline that has been stopped due to either hitting an `ERROR LIMIT` or due to user intervention, the load resumes from the last point in the pipeline. Hence, you can effectively pause a pipeline and resume from the last checkpoint in the pipeline. This feature works for continuous pipelines such as Apache® Kafka® pipelines as well as batch pipelines.

Common uses of stopping and resuming a pipeline are:

- Stopping to update the transformations in a continuous pipeline.
- Stopping to update the table schema and refresh a pipeline.
- Stopping to allow a higher priority workload to proceed.
- Stopping for maintenance.

> ⚠ It is important to exercise caution when resuming a pipeline to ensure that the files in the underlying data source have not changed. Modifying data in the source for an in-process pipeline can result in unexpected results, such as duplicate data.

File Loading

When is the file list determined for a batch file load?

When you load a batch of files in a data pipeline, the Ocient System determines the batch when the pipeline starts using the bucket, the prefix, and the filter parameters you specify in your pipeline. This batch of files is frozen until the pipeline completes. Even if the batch pipeline stops and restarts, the Ocient System keeps the list of files constant.

Can I update the file list after starting a pipeline?

You cannot manually modify the file list for a batch pipeline. When a pipeline restarts, it finishes the files that were determined when the batch was first established.

To add files that arrive on the file system after a batch pipeline was first started, you can create a new batch pipeline that uses filters to load all files created after a specified timestamp.

What is the behavior if I start a pipeline that has been stopped?

When you start a file-based pipeline that has been stopped, the pipeline adds any new files to the pipeline as files with the `PENDING` status. The pipeline then loads these files as execution proceeds. The Ocient System does not reload any files that the pipeline has already loaded.

> ⚠ It is important to exercise caution when resuming a pipeline to ensure that the files in the underlying data source have not changed. When you modify data in the source for an in-process pipeline, the Ocient System can have unexpected results, such as duplicate data.

What options exist to maximize performance on S3-type file loads?

The most common performance enhancement for Amazon® Web Services℠ (AWS℠) S3 file loads is the `PREFIX` parameter in your pipeline. This setting restricts the files in the bucket that your pipeline considers, speeding up listing operations.

You can also use other advanced options such as `MAX_CONCURRENCY`, `REQUEST_DEPTH`, and `REQUEST_RETRIES` described in **S3 Source Options**.

Kafka Loading

How does a continuous Kafka pipeline work?

Continuous pipelines for Kafka stream records from Kafka partitions and ensure exactly-once semantics are delivered. Ocient assigns consumers to different Loader Nodes and commits offsets back on the assigned partitions as the source data loads into Ocient. This operation critically only occurs after data is durable in Ocient, which

means that the system replicates data in such a way that the loss of a configurable number of nodes would not lead to data loss.

All supported data formats are compatible with Kafka pipelines.

When you run a continuous Kafka pipeline, the pipeline is never complete so it remains in a running state. You can observe progress using the standard system catalog tables and the Kafka-specific table `sys.pipeline_partitions` that captures individual partition assignments and lag.

When you stop and restart a Kafka pipeline, it resumes from the last commit checkpoint and continues processing records. The Ocient System deduplicates any record that has already been sent to Ocient but not committed to Kafka in the Loader Nodes.

How do I update the transforms or the table schema in a continuous Kafka pipeline?

To update transforms in a Kafka pipeline, first, stop the pipeline. After the pipeline stops, you can execute a `CREATE OR REPLACE PIPELINE` SQL statement. This action allows you to update the transformations in the pipeline and maintains all of the checkpoint information about progress on the pipeline. When you execute the `START PIPELINE` SQL statement, the pipeline resumes from the last Kafka committed offsets. Then, the Loader Nodes deduplicate any record that has already been sent to the Loader Nodes but is not yet committed to Kafka.

Where does a Kafka pipeline resume loading after a stop and then start of the pipeline?

When you execute the `START PIPELINE` SQL statement, the pipeline resumes from the last Kafka committed offsets. The Loader Nodes deduplicate any record that has already been sent to the Loader Nodes but is not yet committed to Kafka.

If you have never started a Kafka pipeline, the pipeline starts from the configured `auto.offset.reset` value. You can set this value in the `CONFIG` option of a Kafka pipeline. The Ocient System uses this value when a pipeline first starts to establish the consumer group for the pipeline.

For more details about Kafka configuration, see **Kafka Source Options**.

Data Duplication and Deduplication

What records are deduplicated during loading?

Ocient Loader Nodes deduplicate data that is loaded on the same pipeline. You can think of the deduplication scope of a pipeline as being tied to the life of the pipeline database object.

If you drop a pipeline and create a new one with the same name, this data is *not* deduplicated.

For a specified pipeline, the Ocient System deduplicates data automatically when you stop and resume a pipeline. The system ensures that no duplicate data is loaded for any reprocessed data due to managing watermarks with data sources such as Kafka or S3.

It is important to note that deduplication in Ocient is based on a shared contract between the data source and the Ocient System. Deduplication is not based on a primary key or record identifier that is part of the data itself. The **What mistakes can accidentally lead to duplicating data?** question explains the implications more clearly.

How can I load the same data set twice into the same table?

Sometimes, it is useful to load data multiple times for testing, but deduplication can get in the way.

To load the same data set twice in the same table, there are two simple approaches:

1. Drop the pipeline from the first load, create the same pipeline, and run it to completion.
2. Create a new pipeline with the same SQL statement as the original, but with a new name, and run it to completion.

In both cases, the Ocient System does not deduplicate the data, and you can load a second copy of your data.

What mistakes can accidentally lead to duplicating data?

The Ocient System can have duplicate data when the deduplication contract between the source and the pipeline is not maintained.

Ocient uses a proprietary approach to deduplication that helps it deliver high throughput when loading while still ensuring exactly-once semantics. If the data in the source changes during the execution of a pipeline, the changes can lead to a mismatch in the way the Ocient System determines the deduplication identifier for a record.

Things to avoid include:

- Modifying a file in the source file system after it has been included in a pipeline, but the pipeline has not been completed.
- Deleting a file in the source that has been included in a pipeline, but the pipeline has not been completed.

You can add new files safely to a pipeline before it completes, but you should avoid modifying files already registered by the pipeline and appear in the `sys.pipeline_files` system catalog table.

Troubleshooting and Errors

How do I detect when an entire file fails in a pipeline?

If an entire file has failed, you can see this in the `sys.pipeline_files` system catalog table. Any file with the `FAILED` status indicates that the file failed to process. This failure can occur due to a file-level error, such as corruption.

In addition, if a file has the `SKIPPED` status, this status indicates that the file was skipped when processing the pipeline. You can control this behavior using the `FILE_ERROR` setting in the `START PIPELINE` SQL statement.

If a specified file experiences record-level errors, but the Ocient System processes the file, the file has the `LOADED_WITH_ERRORS` status.

How do I find the individual records that failed to load?

You can find the individual errors in the `sys.pipeline_errors` system catalog table.

You can determine if a pipeline encounters errors by viewing the `information_schema.pipeline_status` view that includes counts of records with errors.

Transformations

How can I convert a string representation of an array of data into an array data type?

The Ocient System automatically converts a string representation of an array into an array using the settings defined on your pipeline. The system uses the `CLOSE_ARRAY`, `OPEN_ARRAY`, and `ARRAY_ELEMENT_DELIMITER` settings to convert a string to a target array column type. These types default to a PostgreSQL®-style array (e.g., `{1,2,3}`).

For details, see the supported options in **Extract Options**.

How do I apply a transformation function to an array of data?

Data pipelines do not yet support the mapping of arbitrary transformation functions over an array. However, the Ocient System supports casting and type conversion. You can apply an array cast to an array and the system applies the type of each element as a conversion to the elements of the array.

For details, see **Scalar Transformation Functions and Casting**.

My data is in a non UTF-8 encoding. How do I get the load to decode my character set properly?

You can change the character set of your pipeline using the `CHARSET_NAME` option in a pipeline. For details, see the supported options in **Extract Options**.

Can a data pipeline extract nested data in JSON?

Ocient supports a JSON selector syntax similar to JSON Path. The Ocient System can extract individual keys as well as nested data. A simple example is

`$order.user.first_name`, which extracts the first name `first_name` from the `user` object on the `order` object in JSON.

For details, see **Supported JSON Selectors**.

Can a data pipeline extract attributes of objects in an array in JSON?

Ocient supports a JSON selector syntax similar to JSON Path. The Ocient System can extract individual keys, nested objects, arrays, and arrays of objects. The system can also project a selector into the arrays. A simple example is `$data.line_items[].price`, which extracts a one-dimensional array of prices `price` from the `line_items` array inside the `data` object.

For details, see **Supported JSON Selectors**.

How do I apply a user-defined transformation to the data?

Data Pipelines support the creation of user-defined functions that you can use to apply more advanced transformations. For details, see the **CREATE PIPELINE FUNCTION** SQL statement.

What automatic conversions does Ocient perform from my source data to the target columns?

The Ocient System supports many automatic conversions to make it easy to load source data into your target tables. `VARCHAR` data automatically casts into most target column types. The system casts many data types automatically to a `VARCHAR` target column. Some other common automatic conversions, such as `BIGINT` to `TIMESTAMP`, are supported, especially where the conversion is lossless.

For details, see **Data Types for Data Pipelines**.

How do I cast data to an array column?

To cast to an array column, use a casting function. This function converts a string of data in the configured array format to the array type you specify, casting the elements

to the inner type. For example, if `$my_column` represents the data `'{123,456,789}'` and your pipeline is using the default `CLOSE_ARRAY`, `OPEN_ARRAY`, and `ARRAY_ELEMENT_DELIMITER` settings, this expression snippet casts the elements to an array of `INT` values. Ocient also automatically casts `$my_column` in this example to an `INT[]` when that is the type of the target column.

SQL

```
CREATE PIPELINE
  ...
SELECT
  INT[]($my_column) as column_array_of_ints;
```

How do I cast data to a tuple column?

Similar to casting to an array column, you can cast to a tuple column using tuple casting syntax. For example, if `$my_column` represents the data `'(123,test)'`, and your pipeline is using the default `CLOSE_TUPLE`, `OPEN_TUPLE`, and `TUPLE_ELEMENT_DELIMITER` settings, this expression snippet casts the elements as a tuple of an integer and a string `TUPLE<<int,varchar>>`.

SQL

```
CREATE PIPELINE
  ...
SELECT
  TUPLE<<int,varchar>>($my_column) as column_tuple;
```

Related Links

Load Data

Data Pipelines Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Query Ocient

The OcientAIQ™ Unified Data Platform contains one or more SQL relational databases. Ocient® follows the PostgreSQL® dialect closely, and most functions operate the same in Ocient as in PostgreSQL. You can use the `SELECT` SQL statement to query data. For details, see the **Data Query Language (DQL) Statement Reference**.

- **Quick Start Querying**
- **Query Performance Tuning**
- **Ocient HTTP Query API**
- **Understanding Data Types**
  - **Array, Tuple, and Matrix Overview**
- **Functions Overview**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Quick Start Querying

This example demonstrates a quick start to begin querying an Ocient® System.

Prerequisites

To get started querying in Ocient, the following are required:

1. A valid username and password for your Ocient System — These credentials can be obtained from your Database Administrator or System Administrator.
2. Network access to the SQL Nodes and the IP Address of the Hostname of these nodes in the Ocient System.
3. The port used for SQL Nodes. By default, this is `4050`.
4. A SQL Client or SQL Driver — This example uses the Ocient JDBC Driver. See **Connect Using JDBC**.

Execute a Query

With these prerequisites, you can construct a connection string for your SQL Client of choice, connect, and begin querying Ocient in a few minutes. This example assumes the use of the JDBC driver and CLI. If using a different connection method, see **Connect to Ocient** to complete Step 2.

Step 1: Create a Connection String

A connection string follows a standard format for Ocient.

```SQL
jdbc:ocient://<HOSTNAME_OR_IP_ADDRESS>:<PORT>/<DATABASE>
```

In this example, you are going to replace each of the placeholders with the following:

© 2026 Ocient, Inc.

- PORT: `4050`
- HOSTNAME_OR_IP_ADDRESS: `10.10.1.1`
- DATABASE: `system`

Some systems use a hostname like `sql1.example.com` instead of an IP Address. Additionally, if you have a database configured already, use that instead of `system` to access your tables.

The final connection string is:

```SQL
jdbc:ocient://10.10.1.1:4050/system
```

Step 2: Connect

Start the JDBC CLI according to the instructions in **Connect Using JDBC**. You need to specify a username and password.

The Fully Qualified Username is specified as `<username>@<database>`, so in your case, for a username of `example_user`, enter `example_user@system`.

Next, you should see an Ocient CLI prompt:

```Shell
Ocient> _
```

Enter the connect statement using your connection string:

```Shell
connect to jdbc:ocient://10.10.1.1:4050/system;
```

You receive a response similar to `Connected to jdbc:ocient://10.10.1.1:4050/system` indicating that you are now connected.

Step 3: Execute a Query

You can now query any user tables or system catalog tables to which you have access in the database. System Catalog tables provide a wealth of information about the underlying system and configuration. Because you are connected to the system database, you can begin with a few catalog queries.

At the Ocient prompt, enter this SQL statement that displays information for all tables in the system.

SQL

```
Ocient> SELECT * FROM information_schema.tables;
```

To filter this information, use the LIKE predicate.

SQL

```
Ocient> SELECT * FROM information_schema.tables WHERE table_name LIKE '%data%';
```

Now that you are connected, you can execute any SQL queries or perform Database Administrator operations like **Data Definition Language (DDL) Statement Reference** or **Data Control Language (DCL) Statement Reference** where you have permission.

Related Links

Data Query Language (DQL) Statement Reference

SQL Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Query Performance Tuning

This guide summarizes tuning techniques for Ocient® SQL querying. These techniques fall into these stages:

- **Pre-query tuning** — Tools to optimize query filters and lookups, including segment keys, indexing, and compression options. You must apply these tools before loading any data.
- **Query performance tuning** — Techniques to improve query performance at execution time.
- **Query performance analysis** — Tools and tips for monitoring queries and analyzing performance.

Pre-Query Tuning

Carefully designing tables and storage with forethought can yield the largest query performance gains. Use these tools during table design:

- Apply TimeKeys — Use the TimeKey® to partition by time and enable efficient pruning for time-bounded filters and rollups. For details, see **TimeKeys**.
- Apply clustering keys and indexes:
  - Use the `CLUSTERING KEY` keywords to co-locate rows with commonly filtered or joined columns to reduce input and output (I/O). For details, see **Clustering Key**.
  - Use the `CLUSTERING INDEX` keywords to boost performance on any additional clustered column combinations. For details, see **Clustering Indexes**.
- Use secondary indexes — Create secondary indexes on columns that are frequently selected, especially for filtering predicates. These indexes can overlap with columns that already use segment keys or clustering indexes. For details, see **Secondary Indexes**.
- Table compression and encoding:

© 2026 Ocient, Inc.

    - Choose compression to lower storage and I/O while keeping CPU affordable for your workload. For details, see **Table Compression Options**.
    - Consider column cardinality and skew. Leverage dictionary compression for repeated values. For details, see **Global Dictionary Compression**.

For examples of these different tools and use cases, see **CREATE TABLE SQL Statement Examples**.

Query Performance Tuning

Shape queries to minimize work at execution, align with indexes and data layout, and use runtime controls and caching to govern resources and reuse results. Use these methods while writing a query to tune performance:

- Adhere to query best practices:
    - Filter and aggregate early. Apply the most selective predicates as soon as possible in a query.
    - Avoid unnecessary use of the `DISTINCT` keyword. Also, avoid the use of many columns in a `SELECT` SQL statement.
    - Avoid applying functions on indexed columns in predicates (e.g., `WHERE LOWER(city) = 'chicago'`) because it can impact index and join performance. Instead, create a normalized companion column (e.g., `city_lower`), index it, and use it for queries.
    - Avoid sub-queries in the `SELECT` statement where possible. Use the `JOIN` keyword to join queries or use window functions when possible instead.
    - If you have a column with no duplicate values, use the `UNION ALL` keywords instead of the `UNION` keyword to avoid deduplication overhead.
- Apply workload management — Route heavy and long-running queries to batch classes and keep interactive queries in low-latency classes. For details, see **Workload Management Walkthrough**.
- Use result set caching — For recurring, identical queries, leverage caching to store query results and bypass execution. For details, see **Result Set Caching**.

Query Performance Analysis

Measure, inspect, and iterate on query performance using analysis tools, system catalog tables and views, and statistics. Use these techniques to analyze the performance of a query:

- Query analysis (`EXPLAIN` tooling):
  - Inspect the logical and execution plans of your query. Focus on the largest operators by time and rows. For details, see **Query Analysis**.
  - Compare the performance of alternative query structures and index choices.
- System catalog and information schema — Use system catalog tables and information schema views to understand table sizes, index usage, and the performance of queries during execution. For examples of capturing common query metrics, see **Discover Insights From System Catalog Tables**.
- Statistics monitoring and cache management — Monitor key performance metrics and statistics. For details, see **Set Up System Monitoring with the TIG Stack and Kapacitor**.

Related Links

Query Performance Optimizations

Query Analysis

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient HTTP Query API

The OcientAIQ™ Unified Data Platform provides an HTTP Query API that enables the execution of SQL statements. This page uses command-line examples using `curl`.

The HTTP Query API returns data in JSON format by default. The default header value is `Accept: application/json`.

API Documentation

The API follows OpenAPI Specification (Swagger®) guidelines, which you can access in these formats from any SQL Node address `{sql_node_address}`:

- YAML format: `/openapi.yaml` endpoint
- JSON format: `/openapi.json` endpoint

YAML Download Example

CURL

```
curl https://{sql_node_address}/openapi.yaml
```

JSON Download Example

CURL

```
curl https://{sql_node_address}/openapi.json
```

You can use these specifications with various OpenAPI tools for code generation, testing, and documentation.

Connection Setup

Before using the API, you must establish an open port for a SQL Node.

© 2026 Ocient, Inc.

The connectivity pool defines connection settings for SQL Nodes, including their open port and address. For details, see **CONNECTIVITY POOL**.

> All the examples assume that the `openapi_port` is the standard SSL port `443`. If your connectivity pool `openapi_port` has a different port setting, you must specify it in the request URL. For example, this URL specifies using port `8443`: `https://sql_node_address:8443/v1/execute/`.

Enable the OpenAPI Port

To open a port, execute this **ALTER CONNECTIVITY POOL** SQL statement on a connectivity pool in your system. Change the connectivity pool and node values in the example according to your system.

SQL

```
ALTER CONNECTIVITY_POOL benchmark_sql_connectivity_pool
    ALTER PARTICIPANT "benchmark-sql0" SET openapi_port 443;
```

Check Port Configuration

Verify your port configuration by executing this query.

SQL

```
SELECT c.name, n.name, openapi_port
    FROM sys.connectivity_pool_participants AS cp
    LEFT JOIN sys.connectivity_pools AS c ON cp.id = c.id
    LEFT JOIN sys.nodes AS n ON cp.node_id = n.id;
```

The output displays all configured connectivity pools with their associated nodes and OpenAPI ports.

SSL Certificate Setup

The Ocient® HTTP Query API supports two options for handling SSL certificates.

Secure Connection Using TLS

You can use your own certificates for TLS-secured connections by configuring certificate files in your Ocient install. For details, see **Secure Connections Using TLS**.

Unsecured Connection

For quick testing or in non-production environments, you can bypass certificate validation by adding the `-k` or `--insecure` flag to your curl command.

Example

SQL

```
curl -k -u 'admin@system:admin' https://<sql-node-address>/v1/execute/system \
    -d '{"statement":"SELECT * FROM sys.dummy2;","format":"collection"}'
```

Authentication Methods

The Ocient HTTP Query API accepts user credentials in various formats designed for different use cases.

Simple API Requests

For simple queries, you can include user credentials in the request.

In this example, the request includes credentials after the `-u` flag.

CURL

```
curl -u 'admin@system:admin' https://sql_node/v1/execute/system \
    -d '{"statement":"SELECT * FROM sys.dummy2;", "format": "collection"}'
```

Bearer Token Authentication

For continuous access, you can obtain an authentication token by making a request to the `login` endpoint. Replace the text in this example with the address of your SQL Node, your username, your password, and the name of your database.

CURL

```
curl -X POST https://sql_node/v1/login \
    -H "Content-Type: application/json" \
    -d '{
        "username": "admin@system",
        "password": "admin",
        "database": "system"
    }'
```

This request returns an authentication token that you can add to subsequent requests.

CURL

```
curl -X POST https://sql_node/v1/execute \
    -H "Authorization: Bearer TEST_ACCESS_TOKEN" \
    -H "Content-Type: application/json" \
    -d '{
        "statement": "SELECT * FROM sys.dummy2;",
        "database": "system",
        "format": "collection"
    }'
```

OpenID Connect (SSO)

For SSO authentication, you can use OpenID Connect by targeting the `sso_authentication` endpoint with a callback path. Replace the text in this example with the name of your instance, the name of your database, and your callback path.

CURL

```
curl -X POST https://ocient-instance/v1/sso_authentication \
    -H "Content-Type: application/json" \
    -d '{
        "database": "system",
        "sso_callback_path": "/app_callback"
    }'
```

After the redirection and authentication are complete, you can include your token in subsequent requests.

CURL

```
curl -X POST https://ocient-instance/v1/sso_token \
    -H "Content-Type: application/json" \
    -d '{
        "id_token": "TEST_ID_TOKEN",
        "database": "system"
    }'
```

API Endpoint Documentation

The Ocient HTTP Query API supports these endpoints.

For API requests, JSON data must use double quotes (`"`) for all keys and string values. Single quotes should enclose the JSON payload, as shown in this example (see the payload section following the `-d` flag).

CURL

```
curl -X POST \
    https://my_sql_node.com/v1/execute \
    -H "Content-Type: application/json" \
    -d '{ "query": "SELECT * FROM my_table WHERE status = \"active\"", "database"
```

Single or double quotes are acceptable for parameters not nested in JSON objects.

Endpoints

POST execute

GET execute

GET info

POST login

© 2026 Ocient, Inc.

POST logout

POST sso_authentication

GET callback

POST sso_token

POST sso_device_grant

POST sso_device_grant_verify

POST token_refresh

Related Links

Swagger

Data Query Language (DQL) Statement Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

execute

Executes a SQL statement and returns the results. Supports requests for both regular and streaming responses.

With streaming, results return as they become available using the HTTP chunked transfer encoding. Each chunk contains a valid JSON object that you can parse independently.

For details on streaming, including large-response configuration, see the header parameters.

You can specify the database in the request body. If you do not specify the database, the Ocient System defaults to the database specified in the authentication token or system settings. If you specify a database in the request, this value overrides any alternate database value specified as a body parameter.

POST / v1 / execute / {database}

curl

cURL

```
curl -u 'admin@system:admin' https://my_sql_node.com/v1/execute/system \
    -d '{"statement":"SELECT * FROM sys.dummy2", "format": "collection"}'
```

© 2026 Ocient, Inc.

200


```
{
  "query_id": "ac4dd039-e385-400b-a428-fd1fdcd02e30",
  "status": {
    "reason": "The operation completed successfully",
    "sql_state": "00000",
    "vendor_code": 0
  },
  "data": [
    {
      "c1": 1
    },
    {
      "c1": 2
    }
  ]
}
```


Headers

authorization `string` `required`

An access token to connect to your Ocient System.

To receive an access token, you must provide your username and password using the login endpoint.

The system passes the credentials using the Authorization HTTP header using Base64-encoded Basic Authentication.

© 2026 Ocient, Inc.

content-type `string` `required`

Specifies the request body format. Options include:

application/json

Description: The standard option for submitting queries and receiving responses as a single JSON object or array. Use when submitting queries with relatively small or moderate result sets or when you want the entire result set in one response.

application/stream+json

Description: Enables streaming of multiple JSON objects, allowing the API to send results as they become available. Use when querying large tables or expecting a large volume of results or when you want to process each row or result as soon as it is received, without waiting for the full response.

---

accept-encoding `string`

Supported compression algorithms for responses. You can specify multiple algorithms as a comma-separated list.

Supported values are:
"gzip"
"deflate"
"be" (Brotli)
or combinations like: "gzip, be".

Note: For large result sets, compression can impact performance. Without compression, the result sets can be chunks of rows processed in a streaming fashion. With compression, the entire result set must be compressed before it is returned to the client. The client side must decompress the entire result set before processing any rows.

---

accept `string`

Supported response media types. Regardless of the Accept header, the API returns large query results using chunked encoding. Also, see the note about compression in the accept-encoding parameter.

Supported values are:

"application/json" (default)
"application/stream+json"

© 2026 Ocient, Inc.

preferred-encoding `string`

Indicates the preferred encoding type from those specified in the accept-encoding parameter. This header provides fine-grained control when the accept-encoding parameter contains multiple values, allowing the client to specify which encoding is preferred for this specific request explicitly.

Supported values are:

"gzip"
"br"

---

preferred-compression-level `integer`

Specifies the chosen compression level for the selected encoding algorithm. Values range from 1 (fastest, least compression) to 9 (slowest, maximum compression). The exact behavior depends on the specific compression algorithm.

Path Parameters

database `string` `required`

The specific database in your system for the API call.

If you do not specify the database, the executed query defaults to the database from the authentication token or system settings.

If you include a database in the path, this value overrides any alternate database value specified as a body parameter.

Query Parameters

© 2026 Ocient, Inc.

format `string`

Sets the format for the requested data.

Note: This parameter applies only when using Content-Type: text/plain. When using Content-Type: application/json, specify the format in the request body.

Supported values are:

"array" — Returns schema and data as arrays (more efficient for large data sets). (default)
"collection" — Returns data as JSON objects (more convenient for client processing).

Example: format=collection

---

schema `string`

The schema to use for the SQL statement.

Example: schema=mydata

Note: This parameter applies only when using Content-Type: text/plain. When using Content-Type: application/json, specify the schema in the request body.

Body

`application/json`

statement `string` `required`

A valid SQL statement to execute.

Example: statement=SELECT * FROM customers LIMIT 10;

You can include parameters to substitute in the SQL statement by including them in the params body parameter.

---

database `string`

A specific database in your system for the API call.

If you do not specify this parameter, the executed query defaults to the database from the authentication token or system settings.

© 2026 Ocient, Inc.

format `string`

Sets the output format of the query result data.

Supported values are:

"array" — Returns the schema and data as arrays (more efficient for large data sets). (default)
"collection" — Returns data as JSON objects (more convenient for client processing).

---

params `object`

Parameters to substitute in the SQL statement for values enclosed in braces. The keys should match the parameter names in the statement (without the enclosing braces). Wrap string values in single quotes. All other data types do not use quotes.

Example: If your SQL statement is SELECT * FROM orders WHERE order_date > {start_date}, then your API request should include values in the params body parameter, such as {'start_date': '2023-01-01'}.

---

fetch_size `integer`

The number of rows to return in each chunk for streaming responses. This value must be 0 or greater. A value of 0 (default) means the database determines how many rows to return.

Use smaller values for interactive applications (100-1000 rows). For batch processing, use larger values (5000-10000 rows).

---

max_rows `integer`

The maximum number of rows to return. The default value is 9223372036854776000.

Note: Adding a LIMIT clause to a SQL query is generally a better method to restrict the number of rows, as the query engine still generates the total number of rows, but only returns this value to the client.

Response

`200 - application/json`

Response

query_id `string`

status `object`

Show child attributes

data `object[]`

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

execute

Alternative GET method for executing SQL statements. This method passes parameters as URL query parameters. This method does not support the params body parameter.

Specify which database to access in the query parameters. If you do not specify a database, the connection defaults to the database from the authentication token or system settings.

This method is most suitable for simple, read-only queries where the statement can be safely included in a URL. For complex queries or those with parameters, use the POST method instead.

GET / v1 / execute / {database}

curl

cURL

```
curl -u 'admin@system:admin' \
    'https://my_sql_node.com/v1/execute?statement=SELECT%20*%20FROM%20sys.dummy2&database=syst
```

200


```
{
  "query_id": "2d5af900-cb02-44f4-b974-6264d2133e96",
  "status": {
    "reason": "The operation completed successfully",
    "sql_state": "00000",
    "vendor_code": 0
  },
  "data": [
    {
      "c1": 1
    },
    {
      "c1": 2
    }
  ]
}
```


© 2026 Ocient, Inc.

Headers

Authorization `string` `required`

An access token to connect to your Ocient System.

To receive an access token, you must provide your username and password using the login endpoint.

The system passes the credentials using the Authorization HTTP header using Base64-encoded Basic Authentication.

Content-Type `string` `required`

Specifies the request body format. Options include:

application/json

Description: The standard option for submitting queries and receiving responses as a single JSON object or array. Use when submitting queries with relatively small or moderate result sets or when you want the entire result set in one response.

application/stream+json

Description: Enables streaming of multiple JSON objects, allowing the API to send results as they become available. Use when querying large tables or expecting a large volume of results or when you want to process each row or result as soon as it is received, without waiting for the full response.

© 2026 Ocient, Inc.

accept-encoding `string`

Supported compression algorithms for responses. You can specify multiple algorithms as a comma-separated list.

Supported values are:
'gzip'
'deflate'
'br' (Brotli)
or combinations like: 'gzip, br'.

Note: For large result sets, compression can impact performance. Without compression, the result sets can be chunks of rows processed in a streaming fashion. With compression, the entire result set must be compressed before it is returned to the client. The client side must decompress the entire result set before processing any rows.

---

accept `string`

Supported response media types. Regardless of the Accept header, the API returns large query results using chunked encoding. Also, see the note about compression in the accept-encoding parameter.

Supported values are:

"application/json" (default)
"application/stream+json"

---

preferred-encoding `string`

Indicates the preferred encoding type from those specified in the accept-encoding parameter. This header provides fine-grained control when the accept-encoding parameter contains multiple values, allowing the client to specify which encoding is preferred for this specific request explicitly.

Supported values are:

"gzip"
"br"

---

preferred-compression-level `string`

Specifies the chosen compression level for the selected encoding algorithm. Values typically range from 1 (fastest, least compression) to 9 (slowest, maximum compression). The exact behavior depends on the specific compression algorithm.

Path Parameters

database `string` `required`

The specific database in your system for the API call.

If you do not specify the database, the executed query defaults to the database from the authentication token or system settings.

If you include a database in the path, this value overrides any alternate database value specified as a body parameter.

Query Parameters

statement `string` `required`

A valid SQL statement to execute. The statement must be URL-encoded, e.g., statement=SELECT%20*%20FROM%20customers%20LIMIT%2010%3B

database `string`

A specific database in your system for the API call.

If you do not specify the database, the executed query defaults to the database from the authentication token or system settings.

© 2026 Ocient, Inc.

format `string`

Sets the response format.

Supported values are:

"array" — Returns the schema and data as arrays (more efficient for large data sets). (default)
"collection" — Returns data as JSON objects (more convenient for client processing).

---

fetch_size `string`

The number of rows to return in each chunk for streaming responses. This value must be 0 or greater. A value of 0 (default) means the database determines how many rows to return.

Use smaller values for interactive applications (100-1000 rows). For batch processing, use larger values (5000-10000 rows).

Response

`200 - application/json`

Response

query_id `string`

---

status `object`

Show child attributes

---

data `object[]`

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

info

Returns basic system version information about the Ocient System and the HTTP Query API server.

You can use this endpoint to verify connectivity and to check compatible versions.

GET /v1/info

curl

cURL

```
curl -u 'admin@system:admin' \
    'https://my_sql_node.com/v1/info'
```

200

```
{
  "default_database": "system",
  "openapi_version": "1.1.0",
  "status": {
    "vendor_code": 0,
    "sql_state": "00000",
    "reason": "The operation completed successfully"
  }
}
```

Response

200 - application/json

Response

default_database string

openapi_version string

status `object`

Show child attributes

© 2026 Ocient All Rights Reserved

login

Authenticates a user with a username and password and then returns a token for use in subsequent API calls. This request also sets a session cookie.

Include the returned access token in the authorization header for subsequent requests in the format: Authorization: Bearer .

POST /v1/login

curl

cURL

```
curl -X POST https://my_sql_node.com/v1/login \
    -H "Content-Type: application/json" \
    -d '{
        "username": "admin",
        "password": "admin",
        "database": "system"
    }'
```

200

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "username": "jdoe",
  "database": "retail_analytics",
  "expires_in": 3600,
  "status": {
    "reason": "Authentication successful",
    "sql_state": "00000",
    "vendor_code": 0
  }
}
```

Body

application/json

© 2026 Ocient, Inc.

username `string` `required`

Username.

---

password `string` `required`

The password for the username.

---

database `string`

Target database name. If you do not specify this parameter, the request defaults this value to system.

Response

200 - application/json

Response

access_token `string`

---

username `string`

---

database `string`

---

expires_in `integer`

---

status `object`

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

logout

Log out from a SQL session. This clears any associated cookies, but does not invalidate any access tokens.

This endpoint terminates only the cookie-based session. Any bearer tokens that were previously issued continue to work until they expire.

You must discard any stored bearer tokens to complete the log out in a client application.

POST /v1/logout

curl

cURL

```
curl -X POST https://my_sql_node.com/v1/logout
    -d '{
        "database": "retail_analytics"
    }
```

200

```
{
  "status": {
    "reason": "Logged out successfully",
    "sql_state": "00000",
    "vendor_code": 0
  }
}
```

Body

application/json

database `string`

Target database name. If you do not specify this parameter, the request defaults this value to system.

Response

`200 - application/json`

Response

status `object`

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

sso_authentication

Initiates the OpenID Connect authentication process by redirecting to the authorization server.

This endpoint begins the standard OpenID Connect authentication process:

1. The client calls this endpoint with a callback path.
2. The server responds with a redirect to the identity provider.
3. The user authenticates with the identity provider.
4. The identity provider redirects back to the callback endpoint.
5. The client can exchange the authorization code for an access token.

POST / v1 / sso_authentication

curl

cURL

```
curl -X POST https://my_sql_node.com/v1/sso_authentication \
    -H "Content-Type: application/json" \
    -d '{
        "database": "retail_analytics",
        "sso_callback_path": "/auth/callback",
        "do_redirect": true
    }'
```

200 404

```
{
  "status": {
    "reason": "Query executed successfully",
    "sql_state": "00000",
    "vendor_code": 0
  }
}
```

© 2026 Ocient, Inc.

Body

application/json

database string

Target database name. If you do not specify this parameter, the request defaults this value to system.

sso_callback_path string

Local path for redirection after successful authentication. This path should be registered with your OpenID provider as a valid redirect URI.

do_redirect boolean

Determines whether to issue the redirect to the authentication server immediately.

If you set this value to true (default), the API responds with a 302 redirect to the OpenID provider.

If you set this value to false, the API returns the URL in the response body, allowing the client to handle the redirect manually.

If you have no SSO configuration for the database, an error occurs if this parameter value is set to true. Otherwise, the request returns an empty redirect.

Response

200 application/json

Response

status object

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

callback

Provides the authentication token for the OpenID Connect authentication process. The authorization server redirects to callback path after successful authentication.

This endpoint receives the authorization code from the OpenID provider and exchanges it for a token. The provider redirects the user to the application callback URL specified in the initial authentication request.

The client application should not call this endpoint directly. The OpenID provider automatically calls this endpoint in the authentication process.

GET /v1/callback

curl

cURL

```
curl -X GET "https://my_sql_node.com/v1/callback?code=4/P7q7W91a-oMsCeLvIaQm6bTrgtp7&state=fRJ
```

200

```
{
  "status": {
    "reason": "Query executed successfully",
    "sql_state": "00000",
    "vendor_code": 0
  }
}
```

Query Parameters

code `string` `required`

The authorization code from the OpenID provider. This endpoint exchanges the code for a token.

Example: code=4/P7q7W91a-oMsCeLvIaQm6bTrgtp7

© 2026 Ocient, Inc.

state `string` `required`

Prevents cross-site request forgery attacks. This value should match the state in the initial authentication request.

Example: state=fRJfv29f3v39Jf39dJf93jf

Response

`200 - application/json`

Response

status `object`

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

sso_token

Exchange an OpenID Connect identifier token or access token for an Ocient access token.

This endpoint allows clients to directly exchange tokens without following the full browser-based authentication process. This exchange is useful for server-to-server scenarios or when the client already has a valid OpenID token from another process.

POST /v1/sso_token

curl

cURL

```
curl -X POST https://my_sql_node.com/v1/sso_token \
    -H "Content-Type: application/json" \
    -d '{
        "id_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
        "database": "retail_analytics"
    }'
```

200

```
{
  "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "database": "retail_analytics"
}
```

Body

application/json

database string required

Target database for connection after successful authentication. This database must be accessible to the user identified by the SSO token.

© 2026 Ocient, Inc.

id_token `string`

OpenID identifier token received from the OpenID provider after successful authentication. You must provide the id_token or access_token body parameters, but not both.

---

access_token `string`

OpenID access token received from the OpenID provider after successful authentication. You must provide the id_token or access_token body parameters, but not both.

Response

`200 - application/json`

Response

id_token `string`

---

database `string`

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

sso_device_grant

Retrieve an OpenID device grant code that the Ocient System can verify and use with the sso_device_grant_verify endpoint.

The device grant process is intended for devices with limited input capabilities or no web browser:

1. Call this endpoint to retrieve a user code and verification URI.
2. Display the user code and verification URI to the user.
3. The user visits the verification URI on another device and enters the code.
4. Call the sso_device_grant_verify endpoint to check if the user has completed verification.

POST / v1 / sso_device_grant

curl

cURL

```
curl -X POST https://my_sql_node.com/v1/sso_device_grant \
    -H "Content-Type: application/json" \
    -d '{
        "database": "retail_analytics"
    }'
```

200

```
{
  "verification_uri_complete": "https://auth.example.com/device?code=BCDFGHJK",
  "verification_uri": "https://auth.example.com/device",
  "user_code": "BCDFGHJK",
  "status": {
    "reason": "Device authorization initiated",
    "sql_state": "00000",
    "vendor_code": 0
  }
}
```

© 2026 Ocient, Inc.

Body

application/json

database string

Target database name. If you do not specify this parameter, the request defaults this value to system. This value represents the database where the user connects after completing the device grant process.

Response

200 - application/json

Response

verification_uri_complete string

verification_uri string

user_code string

status object

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

sso_device_grant_verify

Verify a previous device grant request and return an authorization token.

After initiating a device grant process with the sso_device_grant endpoint, use this endpoint to check if the user has completed the verification process. If the verification is successful, the endpoint returns an authorization token that you can use for subsequent API calls.

You can call this endpoint multiple times until one of these outcomes:

The user completes the verification (returns 200 OK with a token).

The timeout is reached (returns error).

The verification is canceled (returns error).

POST / v1 / sso_device_grant_verify

curl

cURL

```
curl -X POST https://my_sql_node.com/v1/sso_device_grant_verify \
    -H "Content-Type: application/json" \
    -d '{
        "user_code": "BCDFGHJK",
        "timeout": 60,
        "database": "retail_analytics"
    }'
```

© 2026 Ocient, Inc.

200


```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqZG9lIiwiZGIiOiJyZXRhaWxfYW
  "username": "jdoe",
  "database": "retail_analytics",
  "status": {
    "reason": "Authentication successful",
    "sql_state": "00000",
    "vendor_code": 0
  }
}
```


Body

application/json

user_code string required

The user code returned from a previous device grant request using the sso_device_grant endpoint.

database string

Target database name. If you do not specify this parameter, the request defaults this value to system.

timeout string

The verification timeout, in seconds, indicates how long the server waits for the user to complete the device authorization process before returning a timeout error.

The default value is 30 seconds.

The valid range is 1-600 seconds (10 minutes).

Response

200 - application/json

Response

© 2026 Ocient, Inc.

access_token `string`

username `string`

database `string`

status `object`

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

token_refresh

Refreshes an existing access token, extending its validity period. Call this endpoint before the current token expires to maintain uninterrupted access.

Use the expires_in value from the login or previous refresh response to determine when to refresh the token. A common practice is to refresh when the token has half of its time remaining.

On success, the endpoint returns the 200 response and a JSON object containing a new access token and metadata about the refreshed session. Replace the access token in your client with the new token returned by the endpoint, and use the expires_in value to decide when to refresh again.

POST / v1 / token_refresh

curl

cURL

```
curl -X POST https://your-ocient-instance/v1/token_refresh \
    -H "Authorization: Bearer YOUR_CURRENT_TOKEN" \
    -H "Content-Type: application/json" \
    -d '{
        "database": "retail_analytics"
    }'
```

200

```
{
  "access_token": "eyJhb...",
  "username": "jdoe",
  "database": "retail_analytics",
  "expires_in": 3600,
  "status": {
    "reason": "Token refreshed successfully",
    "sql_state": "00000",
    "vendor_code": 0
  }
}
```

© 2026 Ocient, Inc.

Headers

Authorization `string` `required`

The bearer token, which must be a valid access token still within its refresh lifetime configured on the server. To obtain a token, call either the /v1/login or /v1/token_refresh endpoints.

Content-Type `string` `required`

Specifies the request body format. Use the value: application/json.

Body

`application/json`

database `string`

Target database name. If you do not specify this parameter, the request defaults this value to system.

Response

`200 - application/json`

Response

access_token `string`

username `string`

database `string`

expires_in `integer`

status `object`

Show child attributes

© 2026 Ocient All Rights Reserved

Understanding Data Types

Ocient® supports many common data types such as INT, SHORT, DATE, and TIMESTAMP, as well as complex types like ARRAY and TUPLE, and specialized data types like POINT and LINESTRING used for geospatial analysis. For detailed information about supported data types, refer to the **Data Types** page.

You can use all these data types as types for columns in tables and for queries. Use scalar data conversion functions in queries to manipulate data types. For details, see **Scalar Data Conversion Functions**. You can execute various functions and aggregates depending on the data type. For example, you can sum, average, or process numeric types with other statistical functions, and you can trim, manipulate, or aggregate character types into a single string.

Basic data types such as INT, BOOLEAN, and VARCHAR behave similarly to other databases. You can query and inspect complex types, such as ARRAYS and TUPLES, like any other column, and also have special operators used to interrogate the sets of data.

Numeric and Boolean Data Types

Numeric and Boolean types are used to represent data with different precision and scale. Integer values are supported from a 1-byte signed int up to an 8-byte signed int. All integers are signed in Ocient. Floating point values are possible as either single or double precision, and exact decimal precision is available with the Decimal type. The Boolean type is used to represent true or false in the database.

All numeric types are fixed-length data types.

- BIGINT — 8-byte signed integer
- INT — 4-byte signed integer
- SMALLINT — 2-byte signed integer
- TINYINT — 1-byte signed integer (alias: `BYTE`)
- FLOAT — 4-byte single precision floating-point number (alias: `REAL` or `SINGLE PRECISION`)

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

- DOUBLE — 8-byte double precision floating-point number (alias: `DOUBLE PRECISION`)
- DECIMAL(P,S) — Exact numerical decimal value with precision P and Scale S
- BOOLEAN — 1-byte logical Boolean representing true or false

Learn more about working with numeric data types in the following sections:

- **Math Functions and Operators**
- **Scalar Data Conversion Functions**
- **Aggregate Functions**
- **Formatting Functions**

Character and Binary Types

Character Types represent text and character data. Both fixed-length and variable-length options are available. Fixed-length provides some benefits to the database storage and query engine when applicable, but indexes are available that target variable-length character columns. Binary types represent byte array data.

- CHAR(N) — Variable-length character string with length N (alias: `CHARACTER(N)`)
- VARCHAR(N) — Variable-length character string with a maximum length of N
- BINARY(N) — Fixed-length binary array with length N (alias: `HASH(N)`)
- VARBINARY(N) — Variable-length binary array with maximum length N (alias: `HASH(N)`)

> The database ignores length of N for CHAR, VARCHAR, and VARBINARY. The database uses N for SQL conformance only.

Learn more about working with character and binary data types in these sections:

- **Character and Binary Functions**
- **Scalar Data Conversion Functions**

Date and Time Types

Date and Time types are used to represent time in Ocient. Timestamps take on special capabilities when defined as a TimeKey® on a table. This is described in more detail in **TimeKeys and Clustering Keys**. Ocient provides many functions to operate on time series data.

- DATE — 4-byte Calendar date representing year, month, day
- TIMESTAMP — 8-byte date and time with nanosecond precision in UTC time zone
- TIME — 8-byte time of day in nanoseconds

Learn more about working with date and time data types in the following sections:

- **Date and Time Functions**
- **Window Aggregate Functions**
- **Formatting Functions**

Networking and Unique Identifier Types

Ocient provides data types for processing of IP addresses and UUIDs that allow schemas to leverage improved searching and analysis of both types as well as easier loading of structured data into these types. The IP type is designed for IP V6 and can handle IP V4 addresses as well.

- IPV4 — 4-byte Internet Protocol V4 address
- IP — 16-byte Internet Protocol V6 address
- UUID — 16-byte universally unique identifier (e.g., *01234567-89ab-cdef-1357-0123456789ab*)

Geospatial Data Types

Geospatial analysis is a rich capability in Ocient enabled by special data types and functions. The foundation rests on POINT, LINESTRING, and POLYGON types that represent respective points and shapes on the Earth. From these and composites of these types, many proximity, intersection, and computational geospatial analytics can be performed at an incredible scale.

- POINT — Geometric point in 2 dimensions.
- LINESTRING — Geometric type composed of `N` POINTS.
- POLYGON — Geometric type composed of `N` closed LINESTRINGS.

Learn more about working with geospatial types in the **Geospatial Functions** section.

Complex Types

Arrays and Tuples are SQL container types used to store other SQL types. These allow for complex schemas to be represented as a column in a table instead of separate, normalized tables. In some situations, using Arrays, Tuples, or Arrays of Tuples can lead to significant performance benefits. Use the Matrix type to store one or two-dimensional arrays of fixed-size data types for matrix computations.

- ARRAY — An array of any supported type other than ARRAY.
- MATRIX — A one or two-dimensional mathematical matrix used for matrix calculations.
- TUPLE — A tuple of elements of different types. Tuple is similar to a structure in other programming languages. You can include Tuples in arrays.

Learn more about complex data types in the following sections:

- **Array Functions and Operators**
- **Tuple Functions and Operators**
- **Matrix Functions and Operators**

Related Links

Data Query Language (DQL) Statement Reference

SQL Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Array, Tuple, and Matrix Overview

Arrays, tuples, and matrix data types are complex data types in the Ocient® System.

Arrays

The array data type is a one-dimensional list of values with the same data type. This type enables the support of a one-to-many relationship without creating another table. You create secondary indexes on arrays, which index the elements of the array. For details about secondary indexes, see **Secondary Indexes**.

Key features:

- Variable size.
- Can contain many elements.
- Indexing into the array starts at `1`.
- Invalid access of the array returns NULL.

For semi-structured data in JSON format, you can express the JSON multi-value column in an array data type. Traditionally in the normalized relational data model, the database represents such data types as a separate table. Each row stores the value of the primary key of the record in the parent table. Therefore, the entity relationship between the tables is always one-to-many, which can result in billions of rows. The array data type eliminates the child tables, which helps reduce the storage footprint and avoids the additional JOIN operations.

Examples

Create a one-dimensional integer array `[1, 2, 3]` using the `sys.dummy` virtual table.

SQL

```
SELECT INT[](1,2,3)
FROM sys.dummy1;
```

© 2026 Ocient, Inc.

Output

Text

```
array_int(int((1)), int((2)), int((3)))
----------------------------------------------------------------------------
[1,2,3]

Fetched 1 row
```

Create a two-dimensional integer array `[[1, 2, 3], [4, 5, 6]]`.

SQL

```
SELECT INT[][](INT[](1,2,3),INT[](4,5,6))
FROM sys.dummy1;
```

Output

Text

```
array_array(int)(array_int(int((1)), int((2)), int((3))), array_int(int((4)), int
----------------------------------------------------------------------------
[[1,2,3],[4,5,6]]

Fetched 1 row
```

You can get the same result by casting the array and integers explicitly.

SQL

```
SELECT INT[][](ARRAY[INT(1),INT(2),INT(3)],ARRAY[INT(4),INT(5),INT(6)])
FROM sys.dummy1;
```

For more examples, see **Array Functions and Operators**.

Tuples

The tuple data type is a row. This type is a fixed-sized collection of heterogeneous values. Tuples support more complex recursive computational functions. The memory structure of the tuple is identical to the memory structure of the array type. The only difference is that each entry in a tuple can be any type, so the Ocient System can simultaneously store both fixed and variable-sized values in a tuple.

Examples

Create a tuple with fixed-size types. In this case, use three integers.

SQL

```
SELECT TUPLE(1,2,3)
FROM sys.dummy1;
```

Output

Text

```
tuple((1), (2), (3))
--------------------------------------------------------------------------------
<<1, 2, 3>>

Fetched 1 row
```

You can get the same result by using the tuple type constructor.

SQL

```
SELECT TUPLE<<INT,INT,INT>>(1,2,3)
FROM sys.dummy1;
```

Create a tuple with variable-size types. In this case, use a one-dimensional and two-dimensional array.

SQL

```
SELECT TUPLE(ARRAY[INT(1),INT(2),INT(3)],
    INT[][](ARRAY[INT(1),INT(2),INT(3)],ARRAY[INT(4),INT(5),INT(6)]))
FROM sys.dummy1;
```

Output

Text

```
tuple(array_int(int((1)), int((2)), int((3))), array_array(int)(array_int(int((1)
--------------------------------------------------------------------------------
<<[1,2,3], [[1,2,3],[4,5,6]]>>

Fetched 1 row
```

Create a tuple with different types. In this case, an integer, string, and array of integers.

SQL

```
SELECT TUPLE<<INT,VARCHAR,INT[]>>(1, 'test_string', INT[](2,4,6))
FROM sys.dummy1;
```

Output

Text

```
tuple(int((1)), ('test_string'), array_int(int((2)), int((4)), int((6))))
--------------------------------------------------------------------------------
<<1, test_string, [2,4,6]>>

Fetched 1 row
```

For more examples, see **Tuple Functions and Operators**.

matrices

A matrix is a fixed-size two-dimensional array of `DOUBLE` values. A matrix can be a row vector (1xN matrix) or column vector (Nx1 matrix).

matrices support machine learning model calculations. For details, see **Machine Learning in Ocient**.

Examples

Create a 1x4 matrix with numbers 1 through 4 using the `make_matrix` function that expects values in row-major order. The first two arguments indicate the dimensions of the matrix, and the remaining arguments specify the values for the matrix.

SQL

```
SELECT make_matrix_1x4(1,4,1,2,3,4)
FROM sys.dummy1;
```

Output

Text

```
make_matrix_1x4((1), (4), (1), (2), (3), (4))
------------------------------------------------------------------------------
[[1.0,2.0,3.0,4.0]]

Fetched 1 row
```

Create a 2x3 matrix with numbers 1 through 6.

SQL

```
SELECT make_matrix_2x3(2,3,1,2,3,4,5,6)
FROM sys.dummy1;
```

Output

Text

```
make_matrix_2x3((2), (3), (1), (2), (3), (4), (5), (6))
--------------------------------------------------------------------------------
[[1.0,2.0,3.0],[4.0,5.0,6.0]]

Fetched 1 row
```

Create a row vector with four doubles.

SQL

```
SELECT _R{1,2,3,4}
FROM sys.dummy1;
```

Output

Text

```
_r{1,2,3,4}
--------------------------------------------------------------------------------
[[1.0,2.0,3.0,4.0]]

Fetched 1 row
```

For more examples, see **Matrix Functions and Operators**.

Related Links

Array Functions and Operators

Tuple Functions and Operators

Matrix Functions and Operators

Generate Tables Using sys.dummy

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Functions Overview

Ocient® includes many functions for statistics, mathematics, string manipulation, and working with complex types like Arrays and geospatial data. The following types of functions are supported:

SQL Functions

Math and Arithmetic Functions

Perform mathematical operations on data in Ocient, including basic operators on data types and advanced statistical functions.

- **Math Functions and Operators**
- **Aggregate Functions**

Character and Binary Functions

Manipulate character strings and binary data types and use character operators to query character data.

- **Character and Binary Functions**

Conversion and Formatting Functions

Convert data types with scalar conversions and format numbers and times in character outputs.

- **Scalar Data Conversion Functions**
- **Formatting Functions**

Date and Time Functions

Perform date operations in Ocient, including manipulation and comparison of date and time types. Date and Time functions allow users to add, subtract, and truncate date

© 2026 Ocient, Inc.

time values and intervals.

- **Date and Time Functions**
- **Time Zone Functions**

Aggregates

Aggregates include basic aggregates and sorted aggregates that combine many rows into a single result, as well as window aggregates that combine information from multiple rows within the window into each row in the result set.

- **Aggregate Functions**
- **Window Aggregate Functions**
- **HyperLogLog Functions**

Array, Tuple, and Matrix Functions

Special functions and operators to work with container types. These include operators to check set logic, slice, compare, and manipulate arrays, tuples, and matrices.

- **Array Functions and Operators**
- **Tuple Functions and Operators**
- **Matrix Functions and Operators**

Geospatial Functions

Functions and Operators to manipulate and compare points, linestrings, and polygons.

- **Geospatial Functions**

Network Functions

Functions for using network data types, including `IP` and `IPV4`.

- **Network Type Functions**

Miscellaneous Functions

A set of functions such as `CASE`, `CURRENT_USER`, and `NULL_IF` that provide additional capabilities in Ocient.

- **Other Functions and Expressions**

Related Links

SQL Reference

Data Query Language (DQL) Statement Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Text Analysis in Database Queries

The OcientAIQ™ Unified Data Platform enables search and analysis of textual data. You can run queries efficiently using indexes in the OcientAIQ Unified Data Platform to perform text analysis. The database supports different types of indexes. The N-gram index is a type of secondary index that enables text analysis.

Cluster Key Indexes

Define cluster key (CK) indexes when you create the table.

Index Type	Number of Columns	Column Types	Filters
Primary CK index (always exists)	Multiple	Fixed-length or GDC columns, fixed-length tuple columns	Equality-like, range
Additional CK indexes	Subset of CK in any order		

Secondary Indexes

You can create or drop secondary indexes at any time.

Index Type	Number of Columns	Column Types	Filters
Inverted indexes	Single	Fixed-length or GDC column, array, or tuple component	Equality-like, range
Hash indexes		Variable-length column, array, or tuple component	Equality-like
N-gram indexes		VARCHAR column, array, or tuple component	LIKE, equality-like

N-gram indexes support text analysis by providing an efficient search of textual data. This type of index works by tokenizing the pattern in the string, and then pruning and transforming the tokens. You can create N-gram indexes at any time. For details about creating an N-gram index, see **CREATE INDEX**.

© 2026 Ocient, Inc.

There are other ways to perform text analysis. For details, see the syntax for **LIKE** and **SIMILAR TO**.

Related Links

Data Query Language (DQL) Statement Reference

Secondary Indexes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Machine Learning in Ocient

The OcientAIQ™ Unified Data Platform has OcientML™ functionality that enables machine learning training and model execution within the database. Whether you are training a model or using machine learning functionality to do scoring or prediction, OcientML enables machine learning in the SQL syntax directly.

You can use the `CREATE MLMODEL` SQL statement to create models. This step is for model training and is referred to as model creation. Similarly, you can use the model on new input data to generate new predictions by executing the scalar function that has the same name as the model. This action is typically referred to as executing the model.

Machine learning in Ocient® depends upon linear algebra functionality that is built into the Ocient System. OcientML provides machine learning capabilities that you can invoke in SQL statements with some application logic.

Linear Algebra in Ocient

matrices are a first-class data type in the Ocient System. Create a matrix using a simple SQL SELECT statement.

© 2026 Ocient, Inc.

SQL

```
SELECT {{1, 2}, {3, 4}};
{{1,2},{3,4}}
--------------------------------------------------------------------------------
[[1.0, 2.0], [3.0, 4.0]]

Fetched 1 row

SELECT {{c1*1, c1*2}, {c1*3, c1*4}} FROM sys.dummy2;
make_matrix_2x2((2), (2), ((1))*(c1), ((2))*(c1), ((3))*(c1), ((4))*(c1))
--------------------------------------------------------------------------------
[[1.0, 2.0], [3.0, 4.0]]
[[2.0, 4.0], [6.0, 8.0]]

Fetched 2 rows
```

You can also use shorthand notations to create row `_r` or column vectors `_c`. For example, `_r{1,2,3}` creates a row vector with values `1.0`, `2.0`, and `3.0`.

SQL

```
SELECT _r{1,2,3};
_r{1,2,3}
--------------------------------------------------------------------------------
[[1.0, 2.0, 3.0]]

Fetched 1 row

SELECT _c{1,2,3};
_c{1,2,3}
--------------------------------------------------------------------------------
[[1.0], [2.0], [3.0]]

Fetched 1 row
```

You can execute functions using the values in the vectors. This query does some matrix arithmetic, finds the inverse matrix, and then returns the two eigenvalues and eigenvectors of the inverse.

SQL

```
SELECT EIGEN(INVERSE(2 * {{1,2},{3,4}} + {{5,6},{7,8}} / 2));
EIGEN(INVERSE((((2))*({{1,2},{3,4}}))+(({{5,6},{7,8}})/((2)))))
--------------------------------------------------------------------------------
[<<-1.3780529228406495, [[0.8013353799887076, -0.5982153531783962]]>>, <<0.058052

Fetched 1 row
```

Machine Learning Models

Ocient supports regression, classification, and clustering models. Also, Ocient supports models for dimensionality reduction. For guides on using the different OcientML models, see:

- **Regression Analysis**
- **Classification Analysis**
- **Clustering Analysis and Dimensionality Reduction**

Related Links

Machine Learning Model Functions

Regression Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Regression Analysis

OcientML™ enables regression analysis, which is the process of finding a best-fit function for a data set, in the Ocient® System.

Simple Linear Regression

The simplest case is simple linear regression. Simple linear regression finds a best-fit linear relationship between one variable and another.

Suppose you have this data with a bit of noise added. Use the `CREATE TABLE AS SELECT` SQL statement to create the `mldemo.slr` table. Generate 100 rows of random data.

SQL

```
CREATE TABLE mldemo.slr AS
(SELECT c1 AS x, c1 * 3.14 - 1.41 + (RAND() - 0.5) * 0.01 AS y
FROM sys.dummy100);
Modified 100 rows
```

Create a simple linear regression model using this data.

SQL

```
CREATE MLMODEL my_first_model TYPE SIMPLE LINEAR REGRESSION
ON (SELECT x, y FROM mldemo.slr);
```

Explore the created model by viewing the data in the `machine_learning_models` and `simple_linear_regression_models` system catalog tables.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

SQL

```
SELECT name, on_select, slope, y_intercept
FROM sys.machine_learning_models a, sys.simple_linear_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'my_first_model';
name                                                on_select
--------------------------------------------------------------------------------
my_first_model                                      select x, y from mldemo.slr

Fetched 1 row
```

You can see the calculated slope and the y-intercept from the model, and see that these values are very close to the values used to generate the data. While the input to the model creation, in this case, is very simple, the SQL statement in the ON clause of the model creation can be arbitrarily complex as long as it meets the requirements of the input for the specific model type being created.

Execute the created model by executing the SQL scalar function with the same name as the model you created. The input arguments of the function depend on the specific model type. For regression, the values are the independent variables, and then the function returns the dependent variables.

SQL

```
SELECT x, y AS actual, my_first_model(x) AS predicted FROM mldemo.slr LIMIT 10;
x           actual                 predicted
-------------------------------------------------------
51          158.730137908257       158.72952690917217
52          161.87052128169267     161.8695223592868
53          165.0119331158625      165.00951780940144
54          168.1478263920115      168.14951325951608
55          171.28758882396247     171.28950870963072
56          174.43476312079238     174.42950415974536
57          177.57385971080322     177.56949960986
58          180.70933136639127     180.70949505997464
59          183.84856961481378     183.84949051008925
60          186.99044288540566     186.9894859602039

Fetched 10 rows
```

The model makes predictions that are very close to the actual training data. Then, you can use this data to generate predictions for data that is unknown.

Machine Learning Model Options

You can provide options for most model types. Provide options by pairing the string for an option to its corresponding value. For example, simple linear regression models have a `metrics` option that you can use to ask the database to calculate some quality of fit metrics. Use the same model and enable metrics collection, which is off by default.

SQL

```
DROP MLMODEL my_first_model;
Modified 0 rows

CREATE MLMODEL my_first_model TYPE SIMPLE LINEAR REGRESSION
ON (SELECT x, y FROM mldemo.slr) options('metrics' -> 'true');
Modified 0 rows
```

Examine the `machine_learning_models` and `simple_linear_regression_models` system catalog tables for more fields.

SQL

```
SELECT name, on_select, slope, y_intercept, coefficient_of_determination,
adjusted_r2, rmse FROM sys.machine_learning_models a, sys.simple_linear_regressio
WHERE a.id = b.machine_learning_model_id AND name = 'my_first_model';
name                                          on_select
-------------------------------------------------------------------------------
my_first_model                                select x, y from mldemo.slr

Fetched 1 row
```

The database also calculates the coefficient of determination (frequently named r2), the adjusted r2 value, and the RMSE (root mean squared error). All of these values indicate that this model has an extremely good fit.

In addition, you can force a specific y-intercept such that the value of `f(0)` is fixed. Use the `yIntercept` option. Create a model on the same data, but force the y-intercept to be zero. The metrics show that the model is not as good a fit as when the intercept is free to vary.

SQL

```
CREATE MLMODEL force_zero TYPE SIMPLE LINEAR REGRESSION
ON (SELECT x, y FROM mldemo.slr) options('metrics' -> 'true', 'yIntercept' -> '0'
Modified 0 rows

SELECT name, on_select, slope, y_intercept, coefficient_of_determination,
adjusted_r2, rmse FROM sys.machine_learning_models a, sys.simple_linear_regressio
WHERE a.id = b.machine_learning_model_id AND name = 'force_zero';
name                                          on_select
-------------------------------------------------------------------------------
force_zero                                    select x, y from mldemo.slr

Fetched 1 row
```

Multiple Linear Regression

In multiple linear regression, the model is still predicting an output value. But instead of handling one input variable, the model can handle an arbitrary number of input variables. Multiple linear regression finds the best-fit model of the form:

Text

```
y = a1x1 + a2x2 + a3x3 + … + a0
```

Simple linear regression and multiple linear regression (as well as polynomial regression and linear combination regression) all use a least-squares loss function. Some of the more advanced models allow you to define best fit yourself if the default definition is not satisfactory. Least-squares best fit means that the model minimizes the sum (over all training data rows) of the square of the model error.

Create a training data set with some noise and show that the model can reconstruct the data. Multiple linear regression models also support a metrics option (the metrics are

the same as simple linear regression).

SQL

```
CREATE TABLE mldemo.mlr AS (SELECT a.c1 AS x1, b.c1 AS x2, 1 + 2*a.c1 + 3*b.c1 AS
FROM sys.dummy10 a, sys.dummy10 b);
Modified 100 rows

CREATE MLMODEL mlr_model TYPE MULTIPLE LINEAR REGRESSION ON (SELECT * FROM mldemo
options('metrics' -> 'true');
Modified 0 rows

SELECT x1, x2, y AS actual, mlr_model(x1, x2) AS predicted FROM mldemo.mlr LIMIT
x1          x2          actual                predicted
------------------------------------------------------------------
6           1           16                    15.999999999999975
6           2           19                    18.999999999999975
6           3           22                    21.999999999999975
6           4           25                    24.999999999999975
6           5           28                    27.999999999999975
6           6           31                    30.999999999999975
6           7           34                    33.99999999999997
6           8           37                    36.99999999999997
6           9           40                    39.99999999999997
6           10          43                    42.99999999999997

Fetched 10 rows
```

Polynomial Regression

With polynomial regression, the database finds a least-squares best-fit polynomial of whatever degree you specify. Same as multiple linear regression, you determine the number of independent variables.

The training data has a `1/(x1 * x2)` term that the model does not match, but that term quickly tends towards zero, so the model should provide a good fit with a quadratic polynomial of `x1` and `x2`. Some of the coefficients match the data, but some do not because the model is compensating for not being able to fit the form of the data exactly. However, the metrics indicate that this model is still a great fit.

SQL

```
CREATE TABLE mldemo.pr AS
(SELECT x1, x2, 1.0 / (x1 * x2) - 2*x1 - 0.5*x1*x2 + 2.3*x1^2 + 1.1*x2^2 AS y
FROM (SELECT a.c1 AS x1, b.c1 AS x2 FROM sys.dummy10 a, sys.dummy10 b));
Modified 100 rows

CREATE MLMODEL pr1 TYPE POLYNOMIAL REGRESSION
ON (SELECT * FROM mldemo.pr) options('order' -> '2', 'metrics' -> 'true');
Modified 0 rows

SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'pr1';
name                                          on_select
-------------------------------------------------------------------------------
pr1                                           select * from mldemo.pr

Fetched 1 row
```

Polynomial regression models have many available options, including everything available for multiple linear regression. The `negativePowers` option enables you to fit Laurent polynomials. If you set this option to `true`, the model includes independent variables raised to negative powers. Sums of such terms are named Laurent polynomials. The model generates all possible terms such that the sum of the absolute value of the power of each term in each product is less than or equal to the order.

For example, with two independent variables and the order set to 2, the model is:

`y = a1*x1^2 + a2*x1^-2 + a3*x2^2 + a4*x2^-2 + a5*x1*x2 + a6*x1^-1*x2 + a7*x1*x2^-1 + a8*x1^-1*x2^-1 a9*x1 + a10*x1^-1 + a11*x2 + a12*x2^-1 + b`

SQL

```
CREATE TABLE mldemo.pr_negativepowers AS
(SELECT x1, x2, 1.0 + 12*x2^-1 - 11*x2 + 10*x1^-1 + 9*x1 + 8*(x1 - 1)*(x2 - 1) +
7*x1*x2^-1 + 6*x1^-1*x2 + 5*x1*x2 + 4*x2^-2 + 3*x2^2 + 2*x1^-2 + x1^2 AS y
FROM (SELECT a.c1 AS x1, b.c1 AS x2 FROM sys.dummy10 a, sys.dummy10 b));
Modified 100 rows

CREATE MLMODEL pr2 TYPE POLYNOMIAL REGRESSION
ON (SELECT * FROM mldemo.pr_negativepowers)
options('order' -> '2', 'metrics' -> 'true', 'negativePowers' -> 'true');
Modified 0 rows

SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'pr2';
name                                          on_select
-------------------------------------------------------------------------------
pr2                                           select * from mldemo.pr
```

You can see that now the model almost perfectly recovers the coefficients, with the rest being nearly zero.

Execute the `pr2` model.

SQL

```
SELECT x1, x2, y AS actual, pr2(x1, x2) AS predicted FROM mldemo.pr LIMIT 10;
x1                    x2                    actual                predicted
--------------------------------------------------------------------------------
6.0                   1.0                   69.06666666666666     69.066666667609
6.0                   2.0                   69.28333333333333     69.283333334613
6.0                   3.0                   71.75555555555556     71.755555556936
6.0                   4.0                   76.44166666666666     76.441666668119
6.0                   5.0                   83.33333333333333     83.333333334847
6.0                   6.0                   92.42777777777778     92.427777779345
6.0                   7.0                   103.72380952380954    103.72380952397
6.0                   8.0                   117.22083333333333    117.22083333498
6.0                   9.0                   132.9185185185185     132.91851851868
6.0                   10.0                  150.81666666666666    150.81666666683

Fetched 10 rows
```

Linear Combination Regression

The most complex type of linear regression that Ocient supports is linear combination regression. This regression is a generalization of polynomial regression. Polynomial regression tries to find the best linear combination of polynomial terms (`x1` , `x2` , `x1^2` , `x2^2` , `x1*x2` , and so on). Linear combination regression finds the best fit linear combination, where you can specify all the terms.

Assume this training data. Use the `CREATE TABLE AS SELECT` SQL statement to create the `mldemo.lcr` table. Generate 1000 rows of data.

SQL

```
CREATE TABLE mldemo.lcr AS
(SELECT x1, x2, x3, 10.0 * sin(x1) + 2.0 * cos(x2) - 3.0 * MOD(INT(x1)*INT(x2)*IN
AS y FROM (SELECT a.c1 AS x1, b.c1 AS x2, c.c1 AS x3
FROM sys.dummy10 a, sys.dummy10 b, sys.dummy10 c));
Modified 1000 rows
```

Fit this data using a fifth-degree polynomial model.

SQL

```
CREATE MLMODEL pr_attempting_lcr TYPE POLYNOMIAL REGRESSION
ON (SELECT * FROM mldemo.lcr) options('order' -> '5', 'metrics' -> 'true');
Modified 0 rows

SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'pr_attempting_lcr';
name                                          on_select
-------------------------------------------------------------------------------
pr_attempting_lcr                             select * from mldemo.lcr

Fetched 1 row
```

The coefficient of determination shows that the model is not a bad fit, but the fit is not great due to the structure of the data. However, given that the model is still a linear combination of functions, linear combination regression can recover the fit if you specify the functions to use.

SQL

```
CREATE MLMODEL lcr1 TYPE LINEAR COMBINATION REGRESSION
ON (SELECT x1, x2, x3, y FROM mldemo.lcr) options('function1' -> 'sin(x1)',
'function2' -> 'cos(x2)', 'function3' -> 'mod(int(x1)*int(x2)*int(x3),4)',
'metrics' -> 'true');
Modified 0 rows

SELECT name, on_select, num_arguments, coefficient_of_determination,
adjusted_r2, rmse, y_intercept, coefficients
FROM sys.machine_learning_models a, sys.linear_combination_regression_models b
WHERE a.id = b.machine_learning_model_id AND name = 'lcr1';
name                                          on_select
-------------------------------------------------------------------------------
lcr1                                          select x1, x2, x3, y from mldemo.lcr

Fetched 1 row
```

You can see that the model finds the coefficients perfectly and has a coefficient of determination of 1.0.

The coefficient of determination represents the amount of variance in the data explained by the model. A coefficient of determination of 1.0 indicates that all the variance is explained and that the model is a perfect fit. A coefficient of determination of 0.0 indicates that none of the variance is explained. A negative coefficient of determination is possible, but you do not find it in linear regression models because these models always find a fit that is at least as good as the constant function (the average of all the training data). That model has a coefficient of determination of 0.0.

Non-Linear Regression

The non-linear regression model is similar to linear combination regression in that you define the form of the model. However, non-linear regression can be more than a linear combination of terms.

Assume this training data. Use the `CREATE TABLE AS SELECT` SQL statement to create the `mldemo.nlo_input1` table. Generate 100 rows of data. Then, create a non-linear regression model using five parameters.

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE mldemo.nlo_input1 AS (
    SELECT x1,
        x2,
        2.0 * sin(3.0 * x1 - 1.0) - 4.0 + 5.0 * x2 as y
    FROM (
            SELECT a.c1 / 4.0 as x1,
                b.c1 / 4.0 as x2
            FROM sys.dummy10 a,
                sys.dummy10 b
        )
);
Modified 100 rows

CREATE mlmodel nlo1 TYPE NONLINEAR REGRESSION ON (
    SELECT x1,
        x2,
        y
    FROM mldemo.nlo_input1
) options(
    'numParameters'->'5',
    'function'->'a1 * SIN(a2 * x1 + a3) + a4 + a5 * x2',
    'metrics'->'true'
);
Modified 0 rows
```

In this case, specify only one function that covers the whole model.

Use the `machine_learning_models` and `nonlinear_regression_models` system catalog tables to see the metrics and the values that the model chose.

SQL

```
SELECT name,
    on_select
FROM sys.machine_learning_models a,
    sys.nonlinear_regression_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'nlo1';

name                                                on_select
--------------------------------------------------------------------------------
nlo1                                                select x1, x2, y from mldemo.nlo_inp

Fetched 1 row
```

If you look at the coefficients, you can see the amplitude (2.0) is negative in the model, but that is offset by subtracting pi from the phase. The model still almost exactly matches how the data is generated, but it just found a slightly different, but equivalent, way of doing it.

Non-linear regression models (and all of the other models that use the same infrastructure) have lots of options that control how the search works. These options affect the length of time it takes for training the model, but they also affect the likelihood of getting stuck in a global minimum. The model chooses default values such that they should be sufficient for a wide range of use cases.

With non-linear regression models, you can specify a loss function other than least-squares. In this example, specify the `abs(f-y)` loss function.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL nlo2 type NONLINEAR REGRESSION ON (
    SELECT x1,
        x2,
        y
    FROM mldemo.nlo_input1
) options(
    'numParameters'->'5',
    'function'->'a1 * sin(a2 * x1 + a3) + a4 + a5 * x2',
    'lossFunction'->'abs(f-y)'
);
Modified 0 rows
```

In this case, find the best fit that minimizes the sum of the absolute value of the model errors instead of the sum of the square of the model errors. The variable `f` represents the value of the model, and the variable `y` represents the actual value in the training data.

The catalog indicates the model is nearly the same as the previous model. In other words, changing the loss function here did not substantially change anything. But there are cases where other loss functions are appropriate and they could substantially change the model.

Non-Linear Regression with Neural Networks

In all of the non-linear cases, you have to make a good guess for the form of the target function. However, with neural networks, you can leave it up to the network to try to figure out that form. The downside is that aspects of the model are a black box.

This example uses the same example as non-linear regression, but instead of specifying the form of the function, it relies on what the neural net can do with little information. The size of the network layers and the number of layers greatly influence how good of a fit a neural net model has. This example uses a simple network.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL nn1 TYPE FEEDFORWARD NETWORK ON (
    SELECT x1,
        x2,
        y
    FROM mldemo.nlo_input1
) options(
    'metrics'->'true',
    'hiddenLayers'->'2',
    'hiddenLayerSize'->'2',
    'outputs'->'1',
    'lossFunction'->'squared_error'
);
Modified 0 rows
```

In the case of neural network models, the metric used is an average loss because RMSE and the coefficient of determination are generally only appropriate for least squares models, and neural networks might be using other loss functions.

You can see the average loss by querying the `machine_learning_models` and `feedforward_network_models` system catalog tables.

SQL

```
SELECT name,
    on_select,
    num_arguments,
    average_loss
FROM sys.machine_learning_models a,
    sys.feedforward_network_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'nn1';

name                                              on_select
-------------------------------------------------------------------------------
nn1                                               select x1, x2, y from mldemo.nlo_inpu

Fetched 1 row
```

Compute the coefficient of determination manually.

```sql
SELECT AVG(y)
FROM mldemo.nlo_input1;

(sum(y))/(null_if(double(count(y)), (0)))
------------------------------------------
2.8698217335180987

Fetched 1 row

SELECT 1.0 - SUM((nn1(x1, x2) - y) ^ 2) / SUM((y - 2.8698217335180987) ^ 2) AS r2
FROM mldemo.nlo_input1;

r2
----------------------
0.9981701280250409

Fetched 1 row
```

This model fits the data very well.

Vector-Valued Regression

Vector-valued regression is the case where the dependent variable has multiple components. You can use any type of model to learn each component separately, or you can use a feedforward network to learn all components in one model.

The example creates 100 rows of sample data.

SQL

```
CREATE TABLE mldemo.vector_valued AS (
    SELECT x1,
        { { 2 * x1 + 3,
        3 * x1 -2 } } as y
    FROM (
            SELECT c1 AS x1
            FROM sys.dummy100
        )
);
Modified 100 rows
```

There is a dependent variable that depends on `x1`, which is vector-valued. Each component can fit a linear model, which can then predict vector-valued results.

SQL

```
CREATE MLMODEL vv1 TYPE SIMPLE LINEAR REGRESSION ON (
    SELECT x1,
        y [1,1]
    FROM mldemo.vector_valued
);
Modified 0 rows

CREATE MLMODEL vv2 TYPE SIMPLE LINEAR REGRESSION ON (
    SELECT x1,
        y [1,2]
    FROM mldemo.vector_valued
);
Modified 0 rows
```

The query verifies that it produces reasonable results.

© 2026 Ocient, Inc.

SQL

```
SELECT x1,
    y AS actual,
    { { vv1(x1),
    vv2(x1) } } AS predicted
FROM mldemo.vector_valued
LIMIT 10;

x1         actual
-------------------------------------------------------------------------------
51         [[105.0, 151.0]]
52         [[107.0, 154.0]]
53         [[109.0, 157.0]]
54         [[111.0, 160.0]]
55         [[113.0, 163.0]]
56         [[115.0, 166.0]]
57         [[117.0, 169.0]]
58         [[119.0, 172.0]]
59         [[121.0, 175.0]]
60         [[123.0, 178.0]]

Fetched 10 rows
```

This example shows that a single model can perform this with the feedforward network model type.

SQL

```
CREATE MLMODEL vv3 TYPE FEEDFORWARD NETWORK ON (
    SELECT x1,
        y
    FROM mldemo.vector_valued
) options(
    'hiddenLayers'->'2',
    'hiddenLayerSize'->'2',
    'outputs'->'2',
    'lossFunction'->'vector_squared_error'
);
Modified 0 rows
```

`outputs`, which is set to the size of the vector, and the `lossFunction` option is set to use `vector_squared_error`.

This example shows how well the model can predict the output.

SQL

```
SELECT x1,
    y AS actual,
    vv3(x1) AS predicted
FROM mldemo.vector_valued
LIMIT 10;

x1         actual
-------------------------------------------------------------------------------
51         [[105.0, 151.0]]
52         [[107.0, 154.0]]
53         [[109.0, 157.0]]
54         [[111.0, 160.0]]
55         [[113.0, 163.0]]
56         [[115.0, 166.0]]
57         [[117.0, 169.0]]
58         [[119.0, 172.0]]
59         [[121.0, 175.0]]
60         [[123.0, 178.0]]

Fetched 10 rows
```

Autoregression

Autoregression is linear regression where the independent variables are lags of the dependent variable. In other words, it is a series where the next value depends linearly on the previous N values.

These examples do not use a new model type. Instead, they use multiple linear regression models.

One of the important things about autoregression is that some variables indicate the order of the data. This variable can be a time variable, but it does not need to be. This

variable is necessary to correctly set up the model, but it is not directly used by the model. Instead, the variable helps build the correct lags.

This example builds a model to predict the Fibonacci sequence. This statement creates a table with the first several Fibonacci terms.

SQL

```
CREATE TABLE mldemo.fib AS (
    SELECT 0 AS t,
        0 AS f
    UNION ALL
    SELECT 1,
        1
    UNION ALL
    SELECT 2,
        1
    UNION ALL
    SELECT 3,
        2
    UNION ALL
    SELECT 4,
        3
    UNION ALL
    SELECT 5,
        5
    UNION ALL
    SELECT 6,
        8
    UNION ALL
    SELECT 7,
        13
    UNION ALL
    SELECT 8,
        21
    UNION ALL
    SELECT 9,
        34
);
Modified 10 rows
```

This example builds a model where the dependent variable depends on two lags.

SQL

```
CREATE MLMODEL fib TYPE MULTIPLE LINEAR REGRESSION ON (
    SELECT LAG(f, 1) OVER(
            ORDER BY t
        ),
        LAG(f, 2) OVER(
            ORDER BY t
        ),
        f
    FROM mldemo.fib
);
Modified 0 rows
```

When you execute the model, which provides the model with two lags, the model can predict the next value of the sequence that is not in the data, `55`.

SQL

```
SELECT fib(21, 34);

((((1.0000000000001137))*((21)))+((1.4210854715202004e-14)))+(((1))*((34)))
-------------------------------------------------------------------------
55.0000000000024

Fetched 1 row
```

Vector Autoregression

Vector autoregression is just a more complex version of autoregression. This model uses multiple time series, where the next value for each time series depends linearly on the lags of all of the time series.

This statement sets up some data where the next value for two time series depends on the previous two values of both time series.

SQL

```
CREATE TABLE mldemo.var_input AS (
    SELECT 3.1415 AS s1,
        1.4142 AS s2,
        0 AS t
    UNION ALL
    SELECT 2.7182 AS s1,
        1.618 AS s2,
        1 AS t
);
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
    0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
    2 AS t
FROM (
        SELECT s1 AS s1l1,
            s2 AS s2l1
        FROM mldemo.var_input
        WHERE t = 1
    ),
    (
        SELECT s1 AS s1l2,
            s2 AS s2l2
        FROM mldemo.var_input
        WHERE t = 0
    );
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
    0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
    3 AS t
FROM (
        SELECT s1 AS s1l1,
            s2 AS s2l1
        FROM mldemo.var_input
        WHERE t = 2
    ),
    (
        SELECT s1 AS s1l2,
            s2 AS s2l2
        FROM mldemo.var_input
        WHERE t = 1
    );
```

```
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
    0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
    4 AS t
FROM (
        SELECT s1 AS s1l1,
            s2 AS s2l1
        FROM mldemo.var_input
        WHERE t = 3
    ),
    (
        SELECT s1 AS s1l2,
            s2 AS s2l2
        FROM mldemo.var_input
        WHERE t = 2
    );
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
    0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
    5 AS t
FROM (
        SELECT s1 AS s1l1,
            s2 AS s2l1
        FROM mldemo.var_input
        WHERE t = 4
    ),
    (
        SELECT s1 AS s1l2,
            s2 AS s2l2
        FROM mldemo.var_input
        WHERE t = 3
    );
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
    0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
    6 AS t
FROM (
        SELECT s1 AS s1l1,
            s2 AS s2l1
        FROM mldemo.var_input
        WHERE t = 5
    ),
    (
```

```
        SELECT s1 AS s1l2,
            s2 AS s2l2
        FROM mldemo.var_input
        WHERE t = 4
    );
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
    0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
    7 AS t
FROM (
        SELECT s1 AS s1l1,
            s2 AS s2l1
        FROM mldemo.var_input
        WHERE t = 6
    ),
    (
        SELECT s1 AS s1l2,
            s2 AS s2l2
        FROM mldemo.var_input
        WHERE t = 5
    );
INSERT INTO mldemo.var_input
SELECT 0.9 * s1l1 - 0.8 * s2l1 + 0.7 * s1l2 - 0.6 * s2l2 AS s1,
    0.5 * s1l1 - 0.4 * s2l1 + 0.3 * s1l2 - 0.2 * s2l2 AS s2,
    8 AS t
FROM (
        SELECT s1 AS s1l1,
            s2 AS s2l1
        FROM mldemo.var_input
        WHERE t = 7
    ),
    (
        SELECT s1 AS s1l2,
            s2 AS s2l2
        FROM mldemo.var_input
        WHERE t = 6
    );
```

The query inspects the data.

SQL

```
SELECT *
FROM mldemo.var_input
ORDER BY t;

s1                    s2                    t
-------------------------------------------------------------
3.1415                1.4142                0
2.7182                1.618                 1
2.50251               1.37151               2
2.086991              1.1945109999999999    3
1.8515340999999994    1.0421421             4
1.5768541099999998    0.8961053099999997    5
1.3730730609999995    0.7770167409999997    6
1.1802870530999996    0.6695650050999997    7
1.0215474418099997    0.5788360946099997    8

Fetched 9 rows
```

This example fits the vector autoregression model to the data.

SQL

```
CREATE MLMODEL var1 type VECTOR AUTOREGRESSION ON (
    SELECT { { s1,
        s2 } },
        { { s1_lag1,
        s2_lag1 } },
        { { s1_lag2,
        s2_lag2 } }
    FROM (
            SELECT s1,
                s2,
                LAG(s1, 1) OVER(
                    ORDER BY t
                ) as s1_lag1,
                LAG(s1, 2) OVER(
                    ORDER BY t
                ) as s1_lag2,
                LAG(s2, 1) OVER(
                    ORDER BY t
                ) as s2_lag1,
                LAG(s2, 2) OVER(
                    ORDER BY t
                ) AS s2_lag2
            FROM mldemo.var_input
        )
) options('numVariables'->'2', 'numLags'->'2');
Modified 0 rows
```

See the details of the model in the `machine_learning_models` and `vector_autoregression_models` system catalog tables.

SQL

```
SELECT NAME,
    coefficients,
    coefficient_of_determination
FROM sys.machine_learning_models a,
    sys.vector_autoregression_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'var1';

name                                                coefficients
-------------------------------------------------------------------------------
var1                                                [-2.2737367544323206E-12, 0.89999999

Fetched 1 row
```

The model recovered the expected coefficients. The coefficient of determination is NULL here because the model did not enable metrics collection in the options.

This example checks that the model matches the data reasonably well.

SQL

```
SELECT var1({ { s1_lag1, s2_lag1 } }, { { s1_lag2, s2_lag2 } }) AS predicted,
    { { s1,
    s2 } } AS actual
FROM (
        SELECT s1,
            s2,
            s1_lag1,
            s2_lag1,
            s1_lag2,
            s2_lag2
        FROM (
                SELECT s1,
                    s2,
                    LAG(s1, 1) OVER(
                        ORDER BY t
                    ) AS s1_lag1,
                    LAG(s1, 2) OVER(
                        ORDER BY t
                    ) AS s1_lag2,
                    LAG(s2, 1) OVER(
                        ORDER BY t
                    ) AS s2_lag1,
                    LAG(s2, 2) OVER(
                        ORDER BY t
                    ) AS s2_lag2
                FROM mldemo.var_input
            )
    );
```

```
predicted                                                                      a
-------------------------------------------------------------------------------
NULL                                                                           [
NULL                                                                           [
[[2.5025099999296385, 1.3715099999786722]]                                     [
[[2.086990999938066, 1.1945109999842818]]                                      [
[[1.8515340999451868, 1.0421420999862723]]                                     [
[[1.5768541099535214, 0.8961053099881894]]                                     [
[[1.3730730609589523, 0.7770167409897997]]                                     [
[[1.1802870530645944, 0.6695650050911028]]                                     [
[[1.0215474417789623, 0.5788360946022622]]                                     [
```

```
Fetched 9 rows
```

The first two values have no predictions because the lags necessary to drive the predictions do not exist.

For details, see **Vector Autoregression**.

Related Links

Machine Learning Model Functions

Regression Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Classification Analysis

Classification machine learning algorithms are available in OcientML™. There are many different popular classification algorithms, which work in slightly different ways with different strengths and weaknesses. These walkthroughs and examples cover some basic rules for using classification algorithms and how you can take advantage of these models in Ocient®.

It can be very challenging to anticipate which classification model is best suited for a data set. Often, it is best to experiment with multiple different models to see what happens.

Logistic Regression

Despite its name, logistic regression is a classification algorithm, not a regression algorithm. Logistic regression is generally a good first step for trying out a classification model before moving on to more advanced options.

Logistic regression shares many restrictions with the support vector machines model. Both models require that all features are numeric. However, you can still use non-numeric data if you convert it using one-hot encoding. The class labels can be any data type.

> Use one-hot encoding to convert non-numeric data into numeric form.
>
> For example, if a feature is a color (such as red, green, or blue), then one-hot encoding can make three columns named `is_red`, `is_green`, and `is_blue`. For each row, set one color to the value `1` and the other colors to `0`.

This example uses the `lr_input` table, which represents the academic performance of students.

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE mldemo.lr_input AS (
    SELECT hours_studied,
        grade_in_prev_course,
        CASE
            WHEN RAND() > (1.0 - effective_score) THEN true
            ELSE false
        END AS got_top_marks
    FROM
       ( SELECT hours_studied,
                grade_in_pr ev_course,
                (2 * grade_in_prev_course + hours_studied) / 10.0 AS effective_sc
            FROM
               ( SELECT mod(a.c1, 10) AS hours_studied,
                        mod(b.c1, 5) AS grade_in_prev_course
                    FROM sys.dummy100 a,
                        sys.dummy100 b
                )
        )
);
```

There are two features: the number of hours spent studying `hours_studied` and the grade in the previous course `grade_in_prev_course`. Both input features are numeric. There are also two classes: `true` and `false`. Class values do not need to be Boolean, and they can be any distinct values.

Students who have better grades in the previous course and spend more hours studying are more likely to have top marks in the course. However, this table also includes a randomness factor, which means that rows with the same features can have different classifications. This is normal in real-world data.

This example creates the logistic regression model on the table, including the `metrics` option. This option causes the model to calculate the percentage of samples that are correctly classified. The Ocient System puts this information into the system catalog tables.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL lr1 TYPE LOGISTIC REGRESSION ON (
    SELECT *
    FROM mldemo.lr_input
) options('metrics'->'true');
```

This query selects the `machine_learning_models` and `logistic_regression_models` system catalog tables.

SQL

```
SELECT name,
    on_select,
    num_arguments,
    zero_case,
    one_case,
    correctly_classified
FROM sys.machine_learning_models a,
    sys.logistic_regression_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'lr1';

name                                                on_select
-------------------------------------------------------------------------------
lr1                                                 select * from mldemo.lr_input

Fetched 1 row
```

The model captures the two values for the two target classes. The `correctly_classified` column shows the model has almost 84 percent accuracy from working on the training data. This level of accuracy is pretty good, especially for a model using training data with significant randomness.

This model can predict which classification is most likely if you provide the number of hours studied and grades in the previous course.

First, review the average hours spent studying for each classification.

SQL

```
SELECT avg(hours_studied)
FROM mldemo.lr_input
WHERE got_top_marks;

(sum(hours_studied)) /(null_if(double(count(hours_studied)), (0)))
------------------------------------------------------------------
5.231264244536801

Fetched 1 row

SELECT avg(hours_studied)
FROM mldemo.lr_input
WHERE NOT got_top_marks;

(sum(hours_studied)) /(null_if(double(count(hours_studied)), (0)))
------------------------------------------------------------------
2.3534041715859897

Fetched 1 row
```

As expected, the rows where `got_top_marks` is true have a higher average value for `hours_studied`. The same should be true with `grade_in_prev_course`.

SQL

```
SELECT AVG(grade_in_prev_course)
FROM mldemo.lr_input
WHERE got_top_marks;

(SUM(grade_in_prev_course)) /(NULL_IF(DOUBLE(COUNT(grade_in_prev_course)), (0)))
------------------------------------------------------------------------------
2.3535326451266925

Fetched 1 row

SELECT avg(grade_in_prev_course)
FROM mldemo.lr_input
WHERE NOT got_top_marks;

(sum(grade_in_prev_course)) /(NULL_IF(DOUBLE(COUNT(grade_in_prev_course)), (0)))
------------------------------------------------------------------------------
0.9622195985832349

Fetched 1 row
```

You can also query to find the rows that are anomalies. In most cases, students achieved top marks in their class despite having minimal study hours and low grades in their previous classes.

© 2026 Ocient, Inc.

SQL

```
SELECT *
FROM mldemo.lr_input
WHERE got_top_marks != lr1(hours_studied, grade_in_prev_course)
LIMIT 10;

hours_studied       grade_in_prev_course got_top_marks
-------------------------------------------------------
1                   1                    true
1                   1                    true
1                   4                    false
1                   1                    true
1                   2                    true
1                   0                    true
1                   2                    true
1                   1                    true
1                   2                    true
1                   1                    true

Fetched 10 rows
```

For details about using this model type, see **Logistic Regression**.

Support Vector Machine

Support vector machine (SVM) tries to find a hyperplane to divide data into two classes with these objectives:

- The hyperplane has the maximum margin, meaning the most distance between the two classes.
- The model correctly classifies the highest percentage of data points.

By default, SVM models in Ocient try to balance these two objectives. You can increase the priority of either objective by using the `regularizationCoefficient` option.

For data sets with two features, the SVM hyperplane represents a straight line on a Cartesian graph.

For three features, the hyperplane represents a plane.

This example uses the SVM model on the same classification data set of student performance as used in the logistic regression section.

SQL

```
CREATE MLMODEL svm1 TYPE SUPPORT VECTOR MACHINE ON (
    SELECT *
    FROM mldemo.lr_input
) options('metrics'->'true');

Modified 0 rows
```

The `machine_learning_models` and `support_vector_machine_models` system catalog tables for SVM contain almost the same information as logistic regression.

SQL

```
SELECT name,
    on_select,
    num_arguments,
    negative_case,
    positive_case,
    correctly_classified
FROM sys.machine_learning_models a,
    sys.support_vector_machine_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'svm1';
name                                          on_select
-------------------------------------------------------------------------------
svm1                                          select * from mldemo.lr_input

Fetched 1 row
```

The system catalog tables indicate the same percentage of correctly classified rows as the logistic regression model (about 84 percent).

The SVM model can increase the accuracy further by using a kernel. A kernel is a transformation to make data linearly separable. SVM in Ocient allows you to specify any kernel of your choice.

For example, suppose that a circle can separate two classes of data, dividing data points between those inside and outside the circle. The SVM model cannot find a line to divide the two classes very well, or this problem is not linearly separable. However, you can transform the data into three dimensions (three features) to make it linearly separable. This is the idea of a kernel.

You specify the kernel using the options `function1`, `function2`, `function3`, etc. Each function defines the formula for the kernel to create new features. Functions can use features from the original data set, represented by `x1`, `x2`, `x3`, etc.

To explain this, revisit the circle example. Suppose that the circle centered at the point `(2, 2)` is the class inside. Otherwise, the circle is the class outside. The data that is separable by a circle in two dimensions is linearly separable in three dimensions with the features `x1`, `x2`, `x1^2 + x2^2`.

This statement creates the data and the model.

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE mldemo.circle AS (
    SELECT x,
        y,
        CASE
            WHEN SQRT((x -2) ^ 2 + (y -2) ^ 2) <= 1 then 'INSIDE'
            ELSE 'OUTSIDE'
        END AS label
    FROM (
            SELECT RAND() * 10 as x,
                RAND() * 10 as y
            FROM sys.dummy10000
        )
);

Modified 10000 rows

CREATE MLMODEL svm2 type SUPPORT VECTOR MACHINE ON (
    SELECT x,
        y,
        label
    FROM mldemo.circle
) options(
    'metrics'->'true',
    'function1'->'x1',
    'function2'->'x2',
    'function3'->'x1^2 + x2^2'
);

Modified 0 rows
```

The example includes three functions, meaning that the kernel increases from two to three dimensions. The kernel functions refer to the original feature values `x1` and `x2`. If you examine the `correctly_classified` value in the `machine_learning_models` and `support_vector_machine_models` system catalog tables, you can see a model that classifies nearly everything correctly.

SQL

```
SELECT NAME,
    negative_case,
    positive_case,
    correctly_classified
FROM sys.machine_learning_models a,
    sys.support_vector_machine_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'svm2';
name                                                    negative_case
-------------------------------------------------------------------------------
svm2                                                    cast('OUTSIDE' as CHAR)

Fetched 1 row
```

One advantage of SVM kernels is that they are automatic. The model saves all the kernel information and transforms the data as needed. Hence, you can execute this model to predict a new point without redefining the kernel. You can pass in the two features the model was built over, and a prediction comes out.

> The **sys.machine_learning_model_options** table is the location where the SVM model saves kernel information.
>
> This table saves all the options that you specify. Some models also add additional keys to this dictionary to save certain information needed for when the model is executed.

SQL

```
SELECT svm2(2, 2);
COALESCE(('INSIDE'), (NULL));
-------------------------------------------
INSIDE

Fetched 1 row
```

For details about using this model type, see **Support Vector Machine**.

Binary Classification with Neural Networks

A neural network using the `FEEDFORWARD NETWORK` model type can perform binary classification. To use this model for classification, you must either convert labels to either `0` and `1` (using the `log_loss` option) or convert labels to `-1` and `1` (using the `hinge_loss` option). Both options can achieve the same result.

This model can handle more complex cases, but using a different model for linearly separable data might be easier. In other words, a neural network model is useful when data is difficult to separate linearly using a kernel.

This example shows a feedforward network model using the `log_loss` option. Note the `CASE` statement to convert labels to `0` or `1`.

SQL

```
CREATE MLMODEL logloss TYPE FEEDFORWARD NETWORK ON (
    SELECT x,
        y,
        CASE
            WHEN label = 'INSIDE' THEN 0
            ELSE 1
        END AS target
    FROM mldemo.circle
) options(
    'metrics'->'true',
    'hiddenLayers'->'2',
    'hiddenLayerSize'->'2',
    'outputs'->'1',
    'lossFunction'->'log_loss'
);

Modified 0 rows
```

This query finds that the classification accuracy is about 97 percent.

SQL

```
SELECT COUNT(*) / 10000.0
FROM (
        SELECT x,
            y,
            label,
            CASE
                WHEN logloss(x, y) = 0 then 'INSIDE'
                ELSE 'OUTSIDE'
            END AS predicted
        FROM mldemo.circle
    )
WHERE label = predicted;

(_count(*)_0)/((10000.0))
--------------------------
0.9698

Fetched 1 row
```

To use the `hinge_loss` option, use `-1` and `1` in the `CASE` statement.

SQL

```
CREATE MLMODEL hingeloss TYPE FEEDFORWARD NETWORK ON (
    SELECT x,
        y,
        CASE
            WHEN label = 'INSIDE' THEN -1
            ELSE 1
        END AS target
    FROM mldemo.circle
) options(
    'metrics'->'true',
    'hiddenLayers'->'2',
    'hiddenLayerSize'->'2',
    'outputs'->'1',
    'lossFunction'->'hinge_loss'
);
Modified 0 rows
```

The accuracy of the `hinge_loss` version is exactly the same as the `log_loss` version.

SQL

```
SELECT count(*) / 10000.0
FROM (
        SELECT x,
            y,
            label,
            CASE
                WHEN hingeloss(x, y) = -1 THEN 'INSIDE'
                ELSE 'OUTSIDE'
            END AS predicted
        FROM mldemo.circle
    )
WHERE label = predicted;

(_count(*)_0)/((10000.0))
--------------------------
0.9698

Fetched 1 row
```

For details about using this model type, see **Feedforward Neural Network**.

K-Nearest Neighbors

K-nearest neighbors (KNN) can only use numeric features, but KNN handles multi-classification in the Ocient System. This algorithm can handle an arbitrary number of target classes.

One drawback with KNN is that it does not work at training time other than to take a snapshot of the training data. The algorithm operates on the data when you execute the model, so it can be a slow model for large data sets.

To demonstrate KNN functionality, this example uses a similar circle of size (2, 2) with some modifications to use more classes:

- Data within a unit circle of (2, 2) is classified as `INSIDE`.

- Data within a radius of 2 is classified as `MIDDLE`.
- Any other data is classified as `OUTSIDE`.

SQL

```
CREATE TABLE mldemo.circles as (
    SELECT x, y,
        CASE
            WHEN SQRT((x -2) ^ 2 + (y -2) ^ 2) <= 1 THEN 'INSIDE'
            WHEN SQRT((x -2) ^ 2 + (y -2) ^ 2) <= 2 THEN 'MIDDLE'
            ELSE 'OUTSIDE'
        END AS label
    FROM (
            SELECT RAND() * 10 as x,
                RAND() * 10 as y
            FROM sys.dummy10000
        )
);

Modified 10000 rows
```

KNN works by finding the `k` nearest points to the one the model is trying to predict a classification for. `k` is an integer value that you must specify during model creation.

The model calculates distances using normal Euclidean distances. You can override the definition of distance by specifying a formula using the `distance` option.

After the model identifies the points, it computes a score for each class and returns the class with the highest score. The `weight` option can influence each score for the specified point. By default, the `weight` value is based on the inverse of the distance (`1.0/(d+0.1)`). In other words, a point closer to the point you are trying to predict has a larger influence on the output result than a point further away. Any point outside of the nearest `k` points has no influence.

This example uses three classes.

SQL

```
CREATE MLMODEL knn1 TYPE K NEAREST NEIGHBORS ON (
    SELECT x,
        y,
        label
    FROM mldemo.circles
) options('k'->'3');

Modified 0 rows
```

These queries test some values to see if the classifications are correct.

SQL

```
SELECT knn1(2, 2);

-------------------------------------------
INSIDE

Fetched 1 row

SELECT knn1(0.5, 2);

-------------------------------------------
MIDDLE

Fetched 1 row

SELECT knn1(5,5);

-------------------------------------------
OUTSIDE

Fetched 1 row
```

KNN models perform work when you execute the model, and the amount of work increases as the input data increases.

One solution to this is data reduction, which means finding a smaller subset of rows to build the KNN model over, enough to ensure that the accuracy of the model is still high. This method front-loads the work when you create the model so that it is much faster to execute on new data.

To use the data reduction feature, set the `dataReduction` option to `true`. By default, the execution stops when the database finds a model that is 90 percent accurate over the training data or when the reduced model increases to 1,000 rows of data. You can override these defaults by setting the `maxReducedRows` or `targetAccuracy` options.

Additionally, if you are using data reduction, you can ask the database to collect metrics on the percent of rows correctly classified by the model because this is much faster to do on the reduced data set.

This example builds another KNN model over the same data and uses data reduction to find a model that is at least 95 percent accurate.

SQL

```
CREATE MLMODEL knn2 TYPE K NEAREST NEIGHBORS ON (
    SELECT x,
        y,
        label
    FROM mldemo.circles
) options(
    'k'->'3',
    'dataReduction'->'true',
    'metrics'->'true',
    'targetAccuracy'->'0.95'
);

Modified 0 rows
```

Query the `machine_learning_models` and `k_nearest_neighbor_models` system catalog tables to see the snapshot table used for the model and its accuracy.

SQL

```
SELECT table_name,
    correctly_classified
FROM sys.machine_learning_models a,
    sys.k_nearest_neighbor_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'knn2';

table_name                                    correctly_classified
--------------------------------------------------------------------
temp.knn86947930347148676                     0.9541

Fetched 1 row
```

The model with data reduction correctly classifies about 95 percent of the training data. To see the size of the KNN model snapshot, you can query it by its table name.

SQL

```
SELECT COUNT(*) FROM temp.knn86947930347148676;

count(*)
--------------------
300

Fetched 1 row
```

The snapshot is only 300 rows. Without data reduction, the row count would be much larger by orders of magnitude.

For details about using this model type, see **K Nearest Neighbor Classification**.

Multi-Class Classification with Neural Networks

The `FEEDFORWARD NETWORK` model type can also perform multi-class classification when the feature classes are numeric. Multi-class classification works by using vectors,

where each position in the vector indicates the score for that class. The **argmax** value of the score of each vector returns the class index.

Much of this example is similar to the previous **Binary Classification with Neural Networks** tutorial, with these modifications to the model options:

- Set the `lossFunction` option to `cross_entropy`.
- Set the `outputs` option to specify the number of classes for the model to use (in this example: `3`).
- Set the `useSoftMax` option to `true`, which applies a **softmax** function to the final vector.

> When the loss function is `cross_entropy_loss`, the option `useSoftMax` defaults to `true`.

SQL

```
CREATE MLMODEL crossentropy TYPE FEEDFORWARD NETWORK ON (
    SELECT x,
        y,
        CASE
            WHEN label = 'INSIDE' THEN { { 1.0,
            0.0,
            0.0 } }
            WHEN label = 'MIDDLE' THEN { { 0.0,
            1.0,
            0.0 } }
            ELSE { { 0.0,
            0.0,
            1.0 } }
        END AS TARGET
    FROM mldemo.circles
) options(
    'metrics'->'true',
    'hiddenLayers'->'2',
    'hiddenLayerSize'->'2',
    'outputs'->'3',
    'lossFunction'->'cross_entropy_loss',
    'useSoftMax'->'true'
);

Modified 0 rows
```

Execute this model to return a vector.

SQL

```
SELECT crossentropy(5,5) AS predicted;

predicted
-------------------------------------------------------------------------------
[[1.3436428911280516E-7, 1.811726873154417E-7, 0.9999996844630236]]

Fetched 1 row
```

To get the classification, use the `VECTOR_ARGMAX` function to retrieve the index of the largest value. This function returns the 1-based index of the largest value in the vector.

SQL

```
SELECT VECTOR_ARGMAX(crossentropy(5,5)) AS predicted;

predicted
--------------------
3

Fetched 1 row
```

This query finds the overall accuracy of the model.

SQL

```
SELECT count(*) / 10000.0
FROM mldemo.circles
WHERE (
        label = 'INSIDE'
        AND VECTOR_ARGMAX(crossentropy(x, y)) = 1
    )
    OR (
        label = 'MIDDLE'
        AND VECTOR_ARGMAX(crossentropy(x, y)) = 2
    )
    OR (
        label = 'OUTSIDE'
        AND VECTOR_ARGMAX(crossentropy(x, y)) = 3
    );

(_count(*)_0)/((10000.0))
--------------------------
0.9298

Fetched 1 row
```

The model is about 93 percent effective. You can improve the accuracy by modifying the **Model Options**, including:

- `hiddenLayers`
- `hiddenLayerSize`
- `lossFuncNumSamples`

Increasing the resources of these options can improve accuracy, but could increase the time needed to train the model.

For details about using this model type, see **Feedforward Neural Network**.

Multi-Class Classification with Support Vector Machines and Logistic Regression

Both logistic regression and SVM are binary classifiers, but you can use either to build a multi-class classifier to handle any number of classes. OcientML can perform this automatically with either model.

This example shows the `CREATE` statement for the SVM model, which uses the same data as the **K Nearest Neighbors** and **Multi-Class Classification with Neural Networks** examples.

SQL

```
CREATE MLMODEL circ1 TYPE SUPPORT VECTOR MACHINE ON (
    SELECT x,
        y,
        label
    FROM mldemo.circles
) options(
    'metrics'->'true',
    'function1'->'x1',
    'function2'->'x2',
    'function3'->'x1^2 + x2^2'
);

Modified 0 rows
```

This statement uses the same kernel as the example for performing binary classification on **Support Vector Machines**.

This query checks the metrics in the `machine_learning_models` and `support_vector_machine_models` system catalog tables to see the accuracy.

SQL

```
SELECT correctly_classified
FROM sys.machine_learning_models a,
    sys.support_vector_machine_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'circ1';

correctly_classified
----------------------
0.9579

Fetched 1 row
```

This model is slightly more accurate than the neural network classifier.

This example tries logistic regression.

SQL

```
CREATE mlmodel circ2 TYPE LOGISTIC REGRESSION ON (
    SELECT x,
        y,
        label
    FROM mldemo.circles
) options('metrics'->'true');

Modified 0 rows

SELECT correctly_classified
FROM sys.machine_learning_models a,
    sys.logistic_regression_models b
WHERE a.id = b.machine_learning_model_id
    AND NAME = 'circ2';

correctly_classified
----------------------
0.9028

Fetched 1 row
```

Logistic regression shows lower accuracy than the neural network model.

These classification models can handle non-numeric data.

Naive Bayes

The first of these models is naive Bayes. This model can handle binary or multi-class classification and features of any data type.

By default, naive Bayes treats each feature as a discrete value. For example, if you assign features with values of `1`, `2`, `3`, and `4`, the model does not give any numeric significance to them. In this case, this model cannot predict the class for the sample with a feature value of `3.5` because that value is not explicitly assigned as a feature.

However, the naive Bayes model can define certain input features as continuous features. These should be numeric values. In this case, the naive Bayes model finds the best-fit normal distribution for that feature to do the probability calculations.

Naive Bayes models essentially use the probability rules of Bayes' Theorem to compute the most likely class based on the training data. The rules are only correct if the features are independent (uncorrelated). However, in practice, the models tend to work well with real-world data.

This example uses this simple table to demonstrate naive Bayes models.

SQL

```
SELECT * FROM mldemo.cars;

color                                                type
-------------------------------------------------------------------------------
yellow                                               sports
yellow                                               sports
yellow                                               suv
red                                                  sports
yellow                                               suv
red                                                  suv
red                                                  sports
yellow                                               suv
red                                                  sports
red                                                  sports

Fetched 10 rows
```

`color`, `type`, and `origin` are the input features, which are all strings. `true` and `false` are the class labels. In this case, all of the input features are discrete.

This example builds the naive Bayes model.

SQL

```
CREATE MLMODEL nb1 TYPE NAIVE BAYES ON (
    SELECT color,
        type,
        origin,
        stolen
    FROM mldemo.cars
) options('metrics'->'true');

Modified 0 rows
```

This query examines the information in the `machine_learning_models` and `naive_bayes_models` system catalog tables for naive Bayes models.

SQL

```
SELECT NAME,
    on_select,
    num_arguments,
    result_probability_table,
    feature_result_matrix_table,
    correctly_classified
FROM sys.machine_learning_models a,
    sys.naive_bayes_models b
WHERE a.id = b.machine_le arning_model_id
    AND NAME = 'nb1';

name                                              on_select
------------------------------------------------------------------------------
nb1                                               select color, "type", origin, stoler

Fetched 1 row
```

In the model, the `correctly_classified` value indicates the accuracy is 80 percent, meaning just two rows are classified incorrectly. The two table names in the `temp` schema hold all the necessary data for executing the model as a function.

This query calls the model to classify new data.

SQL

```
SELECT nb1('red', 'suv', 'domestic') AS prediction;

prediction
-----------
false

Fetched 1 row
```

Here is another example with a different data set to show how continuous features work with naive Bayes models.

SQL

```
SELECT * FROM mldemo.golf;

humidity            play
--------------------------
70                  false
75                  true
86                  true
95                  false
80                  true
85                  false
96                  true
90                  true
91                  false
90                  false
70                  true
70                  true
80                  true
65                  true

Fetched 14 rows
```

In this case, the only input feature is a continuous feature, but you can have multiple continuous features, and you can have a mix of continuous and discrete features.

To build a naive Bayes model over this data, you must set which input features are continuous in the model.

SQL

```
CREATE MLMODEL nb2 TYPE NAIVE BAYES ON (
    SELECT *
    FROM mldemo.golf
) options('continuousFeatures'->'1', 'metrics'->'true');

Modified 0 rows
```

The `continuousFeatures` option is a comma-separated list (no spaces) of input column indexes that are continuous. Indexes start at 1.

This query checks the `correctly_classified` value in the `machine_learning_models` and `naive_bayes_models` system catalog tables.

SQL

```
SELECT correctly_classified
FROM sys.machine_learning_models a,
    sys.naive_bayes_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'nb2';

correctly_classified
----------------------
0.7142857142857143

Fetched 1 row
```

Naive Bayes is a powerful classification algorithm, but it operates as a black box, making it hard to assess how it makes classification decisions.

For details, see **Naive Bayes Classification**.

Decision Trees

Unlike naive Bayes, the decision tree model allows you to see how the model works and makes its decisions.

A decision tree branches based on the value of a single feature, while the leaves of the tree are the output classes. For SQL syntax, this model uses multiple nested `CASE` statements. You can examine the whole nested `CASE` statement that defines the model, but it might be very long.

The decision tree model in Ocient has limited support for continuous features. The model finds the average value of the feature and allows a two-way branch based on whether the value is lower or higher than the average.

Creating a decision tree model is exactly like a naive Bayes model, except with a different type. Decision trees also use the `continuousFeatures` option, which works the same way.

These decision tree examples use the same data sets as the naive Bayes examples.

SQL

```
CREATE MLMODEL dt1 TYPE DECISION TREE ON (
    SELECT color,
        "type",
        origin,
        stolen
    FROM mldemo.cars
) options('metrics'->'true');

Modified 0 rows
```

This query examines the `machine_learning_models` and `decision_tree_models` system catalog tables.

SQL

```
SELECT correctly_classified
FROM sys.machine_learning_models a,
    sys.decision_tree_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'dt1';

correctly_classified
----------------------
0.8

Fetched 1 row
```

This model has the same 80 percent accuracy as the naive Bayes model.

This query examines the `CASE` statement that defines the model.

SQL

```
SELECT case_statement
FROM sys.machine_learning_models a,
    sys.decision_tree_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'dt1';

case_statement
-------------------------------------------
CASE
    WHEN x3 = cast('domestic' as CHAR) THEN CASE
        WHEN x1 = cast('red' as CHAR) THEN CASE
            WHEN x2 = cast('sports' as CHAR) THEN cast('true' as BOOLEAN)
            ELSE cast('false' as BOOLEAN)
        END
        WHEN x1 = cast('yellow' as CHAR) THEN cast('false' as BOOLEAN)
        ELSE cast('false' as BOOLEAN)
    END
    WHEN x3 = cast('imported' as CHAR) THEN CASE
        WHEN x2 = cast('suv' as CHAR) THEN CASE
            WHEN x1 = cast('red' as CHAR) THEN cast('false' as BOOLEAN)
            WHEN x1 = cast('yellow' as CHAR) THEN cast('true' as BOOLEAN)
            ELSE cast('false' as BOOLEAN)
        END
        WHEN x2 = cast('sports' as CHAR) THEN cast('true' as BOOLEAN)
        ELSE cast('false' as BOOLEAN)
    END
    ELSE cast('false' as BOOLEAN)
END

Fetched 1 row
```

Like with the naive Bayes model, the decision tree model can predict new data.

© 2026 Ocient, Inc.

SQL

```
SELECT dt1('red', 'suv', 'domestic') AS prediction;

prediction
-----------
false

Fetched 1 row
```

This example builds a model over the golfing data.

SQL

```
CREATE mlmodel dt2 TYPE DECISION TREE ON (
    SELECT *
    FROM mldemo.golf
) options('continuousFeatures'->'1', 'metrics'->'true');

Modified 0 rows
```

Review the `machine_learning_models` and `decision_tree_models` system catalog tables to see the accuracy of this model.

SQL

```
SELECT correctly_classified
FROM sys.machine_learning_models a,
    sys.decision_tree_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'dt2';

correctly_classified
----------------------
0.7142857142857143

Fetched 1 row
```

Again, the decision tree shows similar accuracy to the naive Bayes model.

Related Links

Machine Learning Model Functions

Classification Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Clustering Analysis and Dimensionality Reduction

This tutorial uses examples to explain the clustering and dimensionality reduction capabilities in OcientML™.

Clustering Models

Clustering models bear similarities to classification, but they use unsupervised learning, which means they do not use any class labels. Instead, the algorithm tries to identify groupings on its own by finding clusters of data that seem closer to each other and farther away from other clusters.

These models generate an integer label, but its ordering is arbitrary. The Ocient® clustering models require you to specify upfront the number of clusters.

K-Means Clustering

K-means is by far the most well-known clustering algorithm because it is simple and fast. K-means performs particularly well if you can scale your features so clusters are roughly circular and equal in size.

These examples demonstrate the k-means model. The examples do not use a data set that is primed for optimal performance of the model, such that they show the shortcomings of the k-means model and how it can perform relatively well even with sub-optimal data.

The examples use a data set of three-dimensional points.

© 2026 Ocient, Inc.

SQL

```
SELECT x, y, z FROM mldemo.clusters_3d LIMIT 10;
x                     y                     z
-------------------------------------------------------------------
-2.332818550827427    2.1009728159912044    -0.7199553405333495
0.27417054580238553   -0.29545777905624226  0.6797340230215586
-2.8608875606526794   2.8532647291218884    -0.7296054608643955
-4.9539478834915345   -4.987358010702081    -0.818608017693332
0.7043958454569171    -0.7209720558364705   -0.7879961167668906
4.949298606341526     4.956911476243746     0.6015700102502752
2.245935331812275     -2.158347482865157    -1.0786578947921444
-1.3725012757407247   1.5652243270484643    -2.5951429274728794
4.516051945094492     4.560705560578962     -0.1569348000058415
5.140105808358035     -4.9583101530224      -1.0064835641873153

Fetched 10 rows
```

This example shows a k-means model created over this data. You can specify multiple options, but the only one that is required is the `k` value, which represents the number of clusters.

SQL

```
CREATE MLMODEL kmeans1 TYPE KMEANS ON (
    SELECT x,
        y,
        z
    FROM mldemo.clusters_3d
) options('k'->'3');

Modified 0 rows
```

This data set has labels, but they are hidden from the model to allow for unsupervised learning.

These example queries compare how well the unsupervised k-means clustering performs against the actual correct classes and labels.

Before you can do that, you must figure out which cluster numbers correspond to which labels.

SQL

```
SELECT COUNT(*),
    cluster_num
FROM (
        SELECT kmeans1(x, y, z) AS cluster_num
        FROM mldemo.clusters_3d
        WHERE label = 'LEFT'
    )
GROUP BY cluster_num;

count(*)            cluster_num
--------------------------------
3165                0
146329              1

Fetched 2 rows

SELECT COUNT(*),
    cluster_num
FROM (
        SELECT kmeans1(x, y, z) AS cluster_num
        FROM mldemo.clusters_3d
        WHERE label = 'RIGHT'
    )
GROUP BY cluster_num;

count(*)            cluster_num
--------------------------------
3877                0
145761              2

Fetched 2 rows

SELECT COUNT(*),
    cluster_num
FROM (
        SELECT kmeans1(x, y, z) AS cluster_num
        FROM mldemo.clusters_3d
        WHERE label = 'CENTER'
    )
GROUP BY cluster_num;
```

```
count(*)            cluster_num
--------------------------------
700868              0

Fetched 1 row
```

The results indicate that cluster `0` is the `CENTER` class, cluster `1` is `LEFT`, and cluster `2` is `RIGHT`.

With that information, you can compute the overall accuracy.

SQL

```
SELECT count(*) / 1000000.0
FROM mldemo.clusters_3d
WHERE (
        kmeans1(x, y, z) = 0
        AND label = 'CENTER'
    )
    OR (
        kmeans1(x, y, z) = 1
        AND label = 'LEFT'
    )
    OR (
        kmeans1(x, y, z) = 2
        AND label = 'RIGHT'
    );

(_count(*)_0)/((1000000.0))
----------------------------
0.992958

Fetched 1 row
```

The accuracy is very good. This calculation indicates that the model never miscategorized `CENTER`, but there were some wrong classifications for the `LEFT` and `RIGHT` labels, although they were rare.

For details, see **K-Means Clustering**.

Gaussian Mixture Models

Gaussian mixture models (GMMs) also perform clustering, however they use a significantly more complex algorithm. GMMs can handle several things that k-means cannot, such as:

- GMMs can handle clusters that are not circular, i.e., they have different variances in different directions.
- GMMs can handle clusters that have an arbitrary rotation, i.e., they can have covariances.
- GMMs can handle the fact that all clusters might not be equally as likely, i.e. if a point is located right between two clusters, then it is more likely to be the one more common in the training data.
- GMMs can show the probability of a new point belonging to each cluster rather than outputting a single cluster value.

GMMs operate by finding a weighted mixture of k multi-variate Gaussians that is most likely to represent the population from which the data was sampled. Each Gaussian in the mix has a mean vector, which represents the center of each cluster.

After you create the model, it is simple to determine the cluster that a point most likely belongs to. The examples demonstrate how to find the highest probability cluster.

This example makes a model over the same data set. This model requires the `numDistributions` option, which represents the number of clusters.

SQL

```
CREATE MLMODEL gmm TYPE GAUSSIAN MIXTURE MODEL ON (
    SELECT x,
        y,
        z
    FROM mldemo.clusters_3d
) options('metrics'->'true', 'numDistributions'->'3');

Modified 0 rows
```

This query executes the GMM model.

SQL

```
SELECT gmm(0,0,0) AS class_probabilities;

class_probabilities
-------------------------------------------------------------------------------
[[2.0445296213545875E-4, 0.999582295498657, 2.1325153920763012E-4]]

Fetched 1 row
```

The results are very different than executing a k-means model because these are probabilities. In this case, the output means the probability of `(0,0,0)` being the second class is very high (greater than 99.9 percent), while the probability of it being the other classes is essentially zero.

To find the most likely class, use the `VECTOR_ARGMAX` function, which returns a 1-based class index.

SQL

```
SELECT VECTOR_ARGMAX(gmm(0,0,0)) AS class_probabilities;

class_probabilities
--------------------
1

Fetched 1 row
```

You can write a query to assess the accuracy of this model. But first, you must figure out the association between classes and labels.

© 2026 Ocient, Inc.

SQL

```
SELECT COUNT(*),
    cluster_num
FROM (
        SELECT VECTOR_ARGMAX(gmm(x, y, z)) AS cluster_num
        FROM mldemo.clusters_3d
        WHERE label = 'LEFT'
    )
GROUP BY cluster_num;

count(*)            cluster_num
----------------------------------------
149483              3
2                   1
9                   2

Fetched 3 rows

SELECT COUNT(*),
    cluster_num
FROM (
        SELECT VECTOR_ARGMAX(gmm(x, y, z)) AS cluster_num
        FROM mldemo.clusters_3d
        WHERE label = 'CENTER'
    )
GROUP BY cluster_num;

count(*)            cluster_num
----------------------------------------
700868              1

Fetched 1 row

SELECT COUNT(*),
    cluster_num
FROM (
        SELECT VECTOR_ARGMAX(gmm(x, y, z)) AS cluster_num
        FROM mldemo.clusters_3d
        WHERE label = 'RIGHT'
    )
GROUP BY cluster_num;
```

© 2026 Ocient, Inc.

```
count(*)            cluster_num
-----------------------------------------
6                   2
3                   3
149629              1

Fetched 3 rows
```

This model misclassified 20 rows, which is far better than the 7,042 rows the k-means model misclassified. This is a direct result of the additional complexity of the GMM.

GMM models can handle much more complex situations than k-means models, but this comes at the cost of more time training and executing the model. The time a GMM model takes compared to a k-means model depends on the number of clusters.

For details, see **Gaussian Mixture**.

Dimensionality Reduction

Dimensionality reduction algorithms reduce the number of input features while still keeping as much of the meaningful properties of the data as possible. Models are quicker to build, and often higher quality after a dimensionality reduction model reduces the number of input features.

A common first step in an analysis is to use dimensionality reduction to simplify the data. See these examples that demonstrate how to use dimensionality reduction as a preprocessing step before using other model types.

Principal Component Analysis

Principal component analysis (PCA) is an unsupervised algorithm that only operates on the inputs, and does not understand what the data is being used for. It is also a linear dimensionality reduction algorithm, meaning that the new features it generates are linear combinations of existing features.

PCA generates as many new features as there are input features. So by itself, it is not reducing the number of dimensions. However, PCA creates new features that try to

maximize variance and sorts the new features in terms of the amount of variance they contain. The system catalog tables provide information on how much variance is contained by the new features. You can use this information to find how many trailing new features to drop.

This PCA tutorial uses a new data set. Start with a regression problem that is trying to find the best-fit polynomial for `f(x1, x2, x3) = y`. The tutorial starts with a **Polynomial Regression** model to see how well it does using three input features, and then it uses PCA to reduce the number of variables without losing accuracy.

This example shows the Polynomial Regression model.

SQL

```
CREATE MLMODEL poly TYPE POLYNOMIAL REGRESSION ON (
    SELECT x1,
        x2,
        x3,
        y
    FROM mldemo.pca_poly
) options('order'->'2', 'metrics'->'true');

Modified 0 rows
```

The system catalog table shows that the model fits the data perfectly.

SQL

```
SELECT coefficient_of_determination
FROM sys.machine_learning_models a,
    sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'poly';

coefficient_of_determination
----------------------------
1.0

Fetched 1 row
```

In this instance, PCA can reduce the three independent variables to two.

The first step is to build a PCA model over the input features.

SQL

```
CREATE MLMODEL pca1 TYPE PRINCIPAL COMPONENT ANALYSIS ON (
    SELECT x1,
        x2,
        x3
    FROM mldemo.pca_poly
);

Modified 0 rows
```

The model does not include `y` because PCA operates over only the input features.

Examine the `machine_learning_models` and `principal_component_analysis_models` system catalog tables.

SQL

```
SELECT importance
FROM sys.machine_learning_models a,
    sys.principal_component_analysis_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'pca1';

importance
--------------------------------------------------------------------------------
[0.6022962127628679, 0.33286202821081917, 0.06484175902631285]

Fetched 1 row
```

The `importance` value indicates that over 93 percent of the signal is in the first two PCA output features, which means it is possible to have a robust model even if you remove the third feature.

This example uses the Polynomial Regression model to access the PCA output features. The example executes the `pca1` PCA function with the PCA input features followed by

the number of features. This number starts at `1`, so this example uses PCA features `1` and `2`, but not the last feature.

SQL

```
CREATE MLMODEL poly2 TYPE POLYNOMIAL REGRESSION ON (
    SELECT pca1(x1, x2, x3, 1),
        pca1(x1, x2, x3, 2),
        y
    FROM mldemo.pca_poly
) options('order'->'2', 'metrics'->'true');

Modified 0 rows
```

Notice that the example still references all three input features, but this model now has only two independent variables. As a result, the model trains 15 percent faster than the version with three independent variables. It is also a much simpler model. The model has six terms in the polynomial regression instead of 10, when there were three independent variables.

This query examines how well the model fits the data.

SQL

```
SELECT coefficient_of_determination
FROM sys.machine_learning_models a,
    sys.polynomial_regression_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'poly2';

coefficient_of_determination
----------------------------
0.9952839146818417

Fetched 1 row
```

The model is a great fit with more than 99 percent accuracy.

In contrast, here is an example where PCA is not a good idea. This example goes back to the three-dimensional clusters data set used in the **Clustering Models** examples.

This example creates the PCA model and checks the `importance` value of the PCA output features in the `machine_learning_models` and `principal_component_analysis_models` system catalog tables.

SQL

```
CREATE MLMODEL pca2 TYPE PRINCIPAL COMPONENT ANALYSIS ON (
    SELECT x,
        y,
        z
    FROM mldemo.clusters_3d
);

Modified 0 rows

SELECT importance
FROM sys.machine_learning_models a,
    sys.principal_component_analysis_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'pca2';

importance
--------------------------------------------------------------------------------
[0.3688781607553818, 0.3333390832624229, 0.2977827559821952]

Fetched 1 row
```

In this case, all three PCA output features are fairly evenly weighted. By removing the last feature, the model covers only about 70% of the signal. This strongly indicates that proceeding with a two-feature model would be problematic and inaccurate.

Linear Discriminant Analysis

While PCA is unsupervised, linear discriminant analysis (LDA) is a supervised dimensionality reduction algorithm. LDA works only with numeric classification

algorithms, but the classification can be binary or multi-class. Hence, LDA understands how the data should be used, making it operate differently.

Creating an LDA model is mostly similar to PCA. One difference is that LDA also requires a class or label.

This tutorial uses the three-dimensional cluster data that the PCA model struggled to work with. The example uses LDA to reduce the features of the cluster data by using the label already contained in the data set.

SQL

```
CREATE MLMODEL lda1 TYPE LINEAR DISCRIMINANT ANALYSIS ON (
    SELECT x,
        y,
        z,
        label
    FROM mldemo.clusters_3d
);

Modified 0 rows
```

As with PCA, there is an `importance` column in the system catalog tables.

SQL

```
SELECT importance
FROM sys.machine_learning_models a,
    sys.linear_discriminant_analysis_models b
WHERE a.id = b.machine_learning_model_id
    AND name = 'lda1';

importance
--------------------------------------------------------------------------------
[1.0000000000000133, -1.4051695910079908E-20, -1.3342566035320652E-14]

Fetched 1 row
```

This output strongly indicates that only the first LDA output feature matters, which means that it is possible to make a single feature classification model.

This example takes the LDA-reduced model as input and uses it to create a neural network model, using the `FEEDFORWARD NETWORK` model. Note that this model uses the `cross_entropy_loss` option to enable multi-class classification.

SQL

```
CREATE MLMODEL three_d_clusters_with_1_feature TYPE FEEDFORWARD NETWORK ON (
    SELECT lda1(x, y, z, 1),
        CASE
            WHEN label = 'LEFT' THEN { { 1.0,
            0.0,
            0.0 } }
            WHEN label = 'RIGHT' THEN { { 0.0,
            1.0,
            0.0 } }
            ELSE { { 0.0,
            0.0,
            1.0 } }
        END AS target
    FROM mldemo.clusters_3d
) options(
    'metrics'->'true',
    'hiddenLayers'->'2',
    'hiddenLayerSize'->'4',
    'outputs'->'3',
    'lossFunction'->'cross_entropy_loss',
    'useSoftMax'->'true'
);

Modified 0 rows
```

Execute the model for a new point to see if it returns a reasonable value.

SQL

```
SELECT three_d_clusters_with_1_feature(lda1(0, 0, 0, 1)) AS predicted;

predicted
-------------------------------------------------------------------------------
[[0.007386760165583984, 0.0022085315302674473, 0.9904047083041486]]

Fetched 1 row
```

In this case, the query asks for a prediction for the input `0, 0, 0`. This input first passes through the LDA model because that is what the neural network model was built on.

The query asks the LDA model for its first component, and then runs the neural network on that value. The neural network outputs a vector of class probabilities; `VECTOR_ARGMAX` returns the 1-based index of the highest probability.

In this example, the highest probability class is the third class, which the query defines as `CENTER`.

SQL

```
SELECT count(*) / 1000000.0
FROM mldemo.clusters_3d
WHERE (
        VECTOR_ARGMAX(
            three_d_clusters_with_1_feature(lda1(x, y, z, 1))
        ) = 1
        AND label = 'LEFT'
    )
    OR (
        VECTOR_ARGMAX(
            three_d_clusters_with_1_feature(lda 1(x, y, z, 1))
        ) = 2
        AND label = 'RIGHT'
    )
    OR (
        VECTOR_ARGMAX(
            three_d_clusters_with_1_feature(lda1(x, y, z, 1))
        ) = 3
        AND label = 'CENTER'
    );

(_count(*)_0)/((1000000.0))
----------------------------
0.818108

Fetched 1 row
```

This model is nearly 82 percent accurate despite being simplified from three features to one. While the accuracy is worse, the time required to train this model is greatly reduced.

Related Links

Machine Learning Model Functions

Clustering and Dimension Reduction Models

© 2026 Ocient All Rights Reserved

Transactions

The Ocient® System supports transactions. A transaction groups one or more SQL statements into a single unit of work that the database either commits as a whole or discards as a whole. Transactions let you apply a set of related changes together and roll them back if any statement fails.

The following SQL statements are supported in transactions.

- **COMMIT**
- **CREATE TABLE AS SELECT**
- **CREATE TRANSACTIONAL PIPELINE**
- **INSERT**
- **ROLLBACK**
- **SELECT**
- **SET AUTOCOMMIT**
- **START PIPELINE**
- **STOP PIPELINE**
- **WITH**

If you execute a SQL statement that a transaction does not support, the database throws an error.

Transaction Support in Ocient SQL

By default, a connection uses autocommit mode, in which the database commits each SQL statement immediately. To group several statements into a single transaction, disable autocommit with the **SET AUTOCOMMIT** statement. After you disable autocommit, the database adds each subsequent statement to the current transaction until you end it.

To end a transaction, execute the **COMMIT** statement to permanently save all changes made since the transaction started. To discard those changes instead, execute the

© 2026 Ocient, Inc.

ROLLBACK statement. For the syntax and examples of these statements, see the **Transactional Control Language (TCL) Statement Reference**.

Transaction Support in the Drivers

The Ocient JDBC driver supports transactions. You can disable autocommit and execute the COMMIT and ROLLBACK commands from the Ocient JDBC CLI program. For details, see **Commands Supported by the Ocient JDBC CLI Program**.

The pyocient module also supports transactions. You can set the autocommit attribute and call the commit and rollback methods on a connection object. For details, see **Ocient Python Module (pyocient)**.

Related Links

Data Definition Language (DDL) Statement Reference

Data Manipulation Language (DML) Statement Reference

Data Query Language (DQL) Statement Reference

Transactional Control Language (TCL) Statement Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Database Administration

Database Administrators are responsible for ensuring the smooth operation of an Ocient® System and learning how to maximize performance to meet the goals of their organization.

See these pages for key concepts that are useful for understanding Ocient System dynamics and managing users.

Tutorials

Security

These pages explain system authentication, SSO, network security, and encryption options.

- **Ocient Security Guide**
- **Set Up Data Encryption**
- **Authentication Methods**
- **Database Password Security Settings**

User Control

These pages explain Ocient user management, including controlling security and priority for user groups:

- **Manage Users, Groups, and Roles**
- **Workload Management and Service Classes**
- **Workload Management Walkthrough**
- **Object-Type Level Privileges Management**

Query Performance and Tuning

© 2026 Ocient, Inc.

Configure keys and indexes before loading large amounts of data into a table. These pages provide detailed information on how to optimize query performance by setting up Ocient keys and indexing:

- **TimeKeys and Clustering Keys**
- **Secondary Indexes**
- **CREATE TABLE SQL Statement Examples**
- **Manage Distributed Tasks**

Storage

These pages explain compression and caching options in the Ocient System:

- **Table Compression Options**
- **Global Dictionary Compression**
- **Table Retention Policies**
- **Result Set Caching**
- **Remove Records from an Ocient System**

SQL Reference Pages

For detailed documentation on supported DDL or DCL commands and syntax, see:

- **Data Query Language (DQL) Statement Reference**
- **Data Definition Language (DDL) Statement Reference**
- **Data Control Language (DCL) Statement Reference**

Related Links

Connect to Ocient

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Security Guide

This security guide is intended to assist organizations in hardening Software as a Service (SaaS) or customer-deployed Ocient® Systems. OcientCloud® and Ocient Management Services include Ocient support personnel configuring these settings based on organizational requirements and the applicable service agreement. Implement these security features and measures to protect an Ocient System and ensure data privacy.

This guide is not exhaustive and represents a security baseline for an Ocient System. Depending on your environment architecture and services, not all security features and measures might apply.

Authentication and Access Control

This section applies to the database application of the Ocient System. You can install the database application on a standard Linux® operating system (OS) distribution and apply authentication and access controls to the application host.

Identity Provider or Single Sign-On (SSO)

You can configure the OcientAIQ™ Unified Data Platform to use any OpenID Connect (OIDC) identity provider (such as OKTA®) for SSO. For details, see **Authentication Methods**.

Due to the potential sensitive data access provided through the Ocient System, use multi-factor authentication (MFA) for all accounts that access the Ocient System.

Ocient recommends implementing an OIDC identity provider on the Ocient System and utilizing MFA for authentication access. Any local accounts should be limited to emergency access to recover SSO issues.

Role-Based Access Control (RBAC) Groups

The Ocient System includes default System and Database Roles. The system also allows database administrators to create access groups with specific tables or views. You can

© 2026 Ocient, Inc.

assign these access groups to database users to restrict access to data based on the data type or classification.

Ocient recommends assigning System and Database Roles based on the principles of least privilege, separation of duties, and need-to-know. Access groups with specific tables, rows, or views can further restrict access based on data security requirements.

For details, see **Manage Users, Groups, and Roles**.

Network Security

This section applies to the network segment that hosts the Ocient System.

Secure Bucket Access and Data Streaming

The Ocient System supports data loading using the Apache® Kafka® stream, cloud bucket (e.g., Amazon® Web Services℠ (AWS℠) S3), or local file source. For details, see **SOURCE Options**.

Secure Bucket Access

If loading data from a cloud bucket like AWS S3, Google® Cloud™ GCP storage bucket, and so on, use network access controls, IP whitelisting, or data-in-transit encryption between the Loader Nodes in the Ocient System and the cloud bucket. Only Loader Nodes need access to a data source.

Secure Data Streaming

The Ocient System supports Kafka Source data streaming to Loader Nodes: `source-configuration`

Ocient recommends utilizing data-in-transit encryption between the Kafka data source and Loader Nodes.

Network Exposure and Firewall

The Ocient System hosts should remain in the same network segment. However, you must apply host and network-based firewalls.

Ocient recommends using deny-all allow-by-exception firewall policies on hosts and network infrastructure to secure Ocient System environments.

> The hosts in the Ocient System should not be publicly accessible but within organizational network boundaries and security (or communication service provider (CSP) network boundaries and security).

The Ocient System uses these ports, protocols, and services externally and internally.

Communications External to the Ocient System

Port	Protocol	Service	Source	Destination
4050	TCP	JDBC	JDBC Connectors	Ocient SQL Hosts
8080	TCP	LAT	LAT Client	Ocient Loader Ho
8090	TCP	Extractor Engine	Metrics Platform	Ocient Loader Ho
9090	TCP	rolehostd	Metrics Platform	Ocient Loader Ho

Communications Internal to the Ocient System

Port	Protocol	Service	Source	Destination
5050	TCP	rolehostd	Ocient LAT Hosts	Ocient Loader Ho
17600	TCP	rolehostd	Ocient SQL and Metadata Hosts	Ocient SQL and Metadata Hosts
17700	TCP	rolehostd	Ocient SQL, Loader, and Foundation Hosts	Ocient SQL, Loac and Foundation Hosts
17800	TCP	rolehostd	Ocient Loader and Foundation Hosts	Ocient Loader ar Foundation Hosts
17900	TCP	rolehostd	Ocient SQL, Metadata, Loader, and Foundation Hosts	Ocient SQL, Metadata, Loade and Foundation Hosts

Data Security and Encryption

This section covers the data security and encryption of the database application. Apply the data security and encryption controls to the Linux OS host.

Data-at-Rest Encryption

The Ocient System supports self-encryption drives (SED) that meet the Trusted Computing Group (TCG) Opal standard or better.

Ocient recommends utilizing NVMe drives that support Opal or SED and implementing data-at-rest encryption. Encryption strength should be based on data and system sensitivity. For details, see **Server Hardware Requirements** and **Data Drive Requirements**.

Data-in-Transit Encryption

You can encrypt external connections using JDBC through transport layer security (TLS) or secure sockets layer (SSL). For details, see **Secure Connections Using TLS**.

Ocient recommends using at least TLS 1.2 for external connections to the Ocient System.

System and Application Security

This section applies to both the Linux OS host and the database application.

Operating System Hardening

For operating systems supported by the Ocient System, see **Ocient System Requirements**.

Ocient recommends the use of an industry-standard OS hardening baseline like the CIS Benchmarks™ or the Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGs).

Operating System Patching

You must frequently and consistently update hosts running the Ocient System to remediate any existing vulnerabilities within the OS.

Ocient recommends performing periodic vulnerability assessments or scans and updating OS packages based on severity and timeframe.

Application Patching

Ocient tracks, monitors, and remediates vulnerabilities within the Ocient System application and publishes updated software releases periodically.

Ocient recommends keeping your Ocient System application updated to the most current version.

For details, see **Ocient Software Upgrade**.

Application and System Maintenance

Ocient recommends performing regular and periodic application and system maintenance to preserve system performance. For details about the Ocient System maintenance activities, see **Maintenance Overview**.

Secrets and Key Management

This section applies to secrets and keys within the Ocient System and database application. The section does not include Linux OS host secrets and keys but should have appropriate authentication and access controls applied. For details, see **Set Up Data Encryption**.

SED Key

Ocient recommends using a Key Management Interoperability Protocol (KMIP) compliant key management platform to handle SED lock and unlock operations.

By default, during system installation, if the install detects Opal-compliant hardware, the install enables full disk encryption with the encryption key stored on the OS drive. For details, see **Data Drive Requirements**.

Auditing and Log Management

This section applies to auditing within the database application. The section does not include Linux OS host and infrastructure logs, but should have appropriate auditing controls applied.

Audit Management

The Ocient System stores security audit logs for the database application in the `/var/log/` directory on the Linux OS host. You can see these files in that directory.

File	Description
`rolehostd.log`	Contains all of the error, informational, and debug logs from the running database system.
`query.json`	Contains information on all queries and statements executed on the system. This file includes information such as the user who ran the query (or command), the time and duration, and many other details.
`security.json`	Contains information on both successful and unsuccessful connections to the system.

For details, see **Log Monitoring**.

SIEM or System Log Forwarding

Ocient recommends using a security information and event manager (SIEM) or system log (syslog) forwarder to ingest the `rolehostd.log`, `query.json`, and `security.json` audit log files located on the Linux OS host.

Privacy

This section covers privacy controls of the database application.

Database View Restrictions

Ocient recommends implementing database table views based on the sensitivity of the data loaded into the database application, organizational policies, and legal requirements. You can grant access to specific columns or rows to specific users and groups by using database table views. For details, see **Views**.

Redacted Queries

The Ocient System supports redacted SQL statements in the database. Redacted text appears as `***`. For details about the privileges that control redaction, see **Data Control Language (DCL) Statement Reference**.

Data Subject Requests

Ocient recommends using these guides to respond to data subject requests related to the General Data Protection Regulation (GDPR) or California Consumer Privacy Act (CCPA).

Delete

To remove records from an Ocient System, see **Remove Records from an Ocient System**.

Update

To update or rectify records in the Ocient System, see **INSERT INTO TABLE**.

Related Links

Authentication Methods

Install an Ocient System

Load Data

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Set Up Data Encryption

The Ocient® System enables you to set up data encryption on data drives that support The Computing Group (TCG) Opal Specification and store the encryption secrets in an external Key Management System (KMS) utilizing the Key Management Interoperability Protocol (KMIP). This series of workflows demonstrates generating and storing secrets using the Thales[MD] CipherTrust Manager KMS.

Prerequisites

A KMIP server, such as Thales CipherTrust Manager, must be available to store secrets.

- The Ocient node must be a registered client of the KMIP server.
- The client certificate and key file must be on the client, e.g., `/var/opt/ocient/kmip/client.crt` and `/var/opt/ocient/kmip/client.key`.
- The CA certificate must be present on the client, e.g., `/var/opt/ocient/ca.pem`.
- The client must be able to communicate with the KMIP server.

The **SEDutil** CLI utility must be installed to manage the SED configuration of the drive. The `sedutil-cli` command must be available on the system path variable `PATH`.

Ensure all NVMe drives intended for use with Ocient are:

- Bound to the kernel NVMe driver.
- Have no partition table.
- Be in the Opal factory state.
  - Use `--revertTPer` or `--revertNoErase` SEDutil commands.
- Have these initial settings for the drives and `SIDPassword` set to `admin`:
  - `Locked = N`
  - `LockingEnabled = N`
  - `LockingSupported = Y`
  - `MBRDone = N`

© 2026 Ocient, Inc.

- `MBREnabled = N`
- `MBRAbsent = N`
- `MediaEncrypt = Y`

Set Up Data Encryption

Use a configuration file and script to generate the secret or key.

1. Generate a configuration file with the path /var/opt/ocient/ockmip.conf. The configuration file uses the **PyKMIP** configuration file format, such as this text.

```none
[client]
host=thales01.example.com
port=5696
certfile=/var/opt/ocient/kmip/client.crt
keyfile=/var/opt/ocient/kmip/client.key
ca_certs=/var/opt/ocient/kmip/ca.pem
```

2. Run this command.

```shell
sudo /usr/local/bin/ockmip init
```

The script performs these operations for each drive that is detected as unused by the OS (no partition table present or with an Ocient partition):

- Generate a password or key for locking the drive.
- Store the secret password or key on the KMIP server.
- Enable Opal locking on the drive and set the `admin1` password to the generated secret.
- Set the `SIDpassword` to `admin` to enable reverting the drive to the factory state in case the `admin1` password is lost.

To enable the script to run on start before the drives are bound to the `uio` or `vfio` driver by the system `bind-uio-driver.service` installed by the `ocient` package, run this command.

Shell

```
sudo /usr/local/bin/ockmip install
```

On reboot, the `ockmip` service unlocks the drives before they are bound to the `uio` or `vfio` driver and available for use by the `rolehostd` process.

The full setup sequence is as follows:

1. Configure the `ockmip.conf` configuration file with all of the appropriate settings.
2. Stop the `rolehostd` process if it is running.
3. Unbind drives from the `uio` or `vfio` driver back to the NVMe driver.
4. If Ocient has already been bootstrapped on the node, use the `import-drives` command. Otherwise, use the `init-drives` command.

shell

```
## Ocient has been bootstrapped.
ockmip import-drives
```

shell

```
## Ocient has not been bootstrapped.
ockmip init-drives
```

5. Set the node role configuration to `useOpal: false` within the `rolehostd.conf` file.
6. Back up any existing keys in `/var/opt/ocient/localKeyStore` in case you encounter any issues.
7. Install the `systemd` service to unlock drives on the reboot.

shell

```
ockmip install
```

8. Optionally, perform a power off and power on to ensure the installation was successful and that the `ockmip.service` successfully unlocks drives on boot.

Migration process output might be similar to this output.

Text

```
[root\@ppolllercs00006 ocient]# ockmip -l DEBUG import-drives
[2024-09-12 12:45:32,853][INFO] Using conf file /var/opt/ocient/ockmip.conf
[2024-09-12 12:45:33,427][INFO] importing drive(s)
[2024-09-12 12:45:33,427][INFO] Importing secret for /dev/nvme15 BTAX240306A015PF
...
```

After reboot, the `ockmip` service output might be similar to this output.

Text

```
[root\@dpolllercs0002e log]# systemctl status ockmip
○ ockmip.service - Oneshot service to unlock NVMe drives via ockmip script using
     Loaded: loaded (/etc/systemd/system/ockmip.service; enabled; preset: disable
     Active: inactive (dead) since Wed 2024-09-11 19:11:55 UTC; 31min ago
    Process: 3217 ExecStart=/usr/local/bin/ockmip unlock (code=exited, status=0/S
   Main PID: 3217 (code=exited, status=0/SUCCESS)
        CPU: 3.097s

Sep 11 19:11:52 dpolllercs0002e ockmip[3217]: [2024-09-11 19:11:52,429][INFO] Ret
...
```

The `rolehostd` process should start up without the process managing encryption because the `ockmip` service handles it.

Troubleshooting

For more details on the possible commands, see the help using this command.

Shell

```
sudo /usr/local/bin/ockmip --help
```

To diagnose issues, examine the log located at `/var/opt/ocient/log/ockmip.log`

To perform actions on a particular drive or set of drives, you can manually specify with the `--drives` parameter.

Shell

```
sudo ockmip unlock --drives /dev/nvme1n1 --drives /dev/nvme2n1
```

`NCERRInsufficentPermissions` Error

When attempting to register the drives, you see this error.

Text

```
[2024-08-26 17:27:51,350][INFO] Using conf file /var/opt/ocient/ockmip.conf [2024
```

To fix this, regenerate the certificate on the endpoint of the Thales device (as the root CA has no trust) in the Thales Administrator.

Disk Replacement

After you swap a disk, initialize it using the `ockmip` utility to add the encryption secret to Thales.

`ockmip.service` Status

Check the status after the boot of the system to ensure that all drives have been correctly unlocked.

© 2026 Ocient, Inc.

Text

```
[root\@ppolllercs00007 \~]# journalctl -u ockmip -S today Sep 12 12:38:09 ppollle
...
```

Timeouts When Contacting Thales

If there is a network outage or if Thales maintenance happens, you might see this output when you attempt to unlock the drives.

Text

```
Sep 12 17:05:59 ppolllercs0000d systemd[1]: Starting Oneshot service to unlock NV
...
```

Related Links

Ocient Security Guide

Authentication Methods

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Workload Management Walkthrough

Workload Management (WLM) in the OcientAIQ™ Unified Data Platform allows a Database Administrator to set restrictions and priorities for groups of users. The main use case for this is to effectively control multiple workflows across the system, prioritizing or limiting resources to groups when needed. Query restrictions are set and assigned using a service class that can be used for multiple user groups. For a full list of restrictions and their definitions, see the documentation.

This example shows how to set up three service classes with differing priorities, explaining the tradeoffs and design decisions one should make when setting up these restrictions.

Create Service Classes

Before creating service classes, the System Administrator or Database Administrator first determines the logical grouping of users on their system and their relative priority. In addition, the response time and concurrency requirements of different user groups are understood. Using these inputs, the administrator can design the service classes that allow their users to receive the proper resource allocations that deliver query requirements.

In this example, define the priority at the service level by creating three service classes to service High, Medium, and Low priority users. Different settings are configured to provide an illustration of the types of configuration that you can use to control features such as concurrency, result sizing, or scheduling priority.

Prerequisites

1. An Ocient® System has been set up with an operational SQL Node.
2. A user with the system administrator role has access to the system

Step 1: Login as a System Administrator User

© 2026 Ocient, Inc.

To create service classes, you must be assigned the **System Administrator** role. For a full list of user permissions and their definitions, see the **Data Control Language (DCL) Statement Reference**.

First, connect to the system database with the command.

Shell

```
connect to jdbc:ocient://my_SQL_ip_address:4050/system;user=admin@system;password=s
```

Step 2: Set up the High Priority User Class

User groups that need fast performance and need access to the most system resources should fall under this service class. Their queries should see minimal impact from outside workloads.

> When defining a service class, the system sets any omitted service class field to the maximum/least restrictive value.

Example Service Class

SQL

```
CREATE SERVICE CLASS high_priority_sc
  scheduling_priority = 100.0,
  max_concurrent_queries = 10,
  max_elapsed_time = 3600;
```

Example System Catalog View

SQL

```
SELECT * FROM sys.service_classes;

id                                   database_id                          name
-------------------------------------------------------------------------------
42649e8e-ae7f-4de5-bb2b-846d158fcb91 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 Default
06b92c4c-b0b1-4a14-880c-7d257af3b2b5 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 high_prio
```

`scheduling_priority` — A numeric value that defines the query's priority in the database. A default query being run with no service class has a priority set to 1.0. This priority number is used as an ordered list for scheduling with higher values receiving priority. It is not a weighted value, so only the ordering, not the relative sizes of the priority levels matter. When setting the priority for a high-priority service class, set this value to be greater than 1.0.

`max_concurrent_queries` — The number of concurrent queries that can be run at a given time for the entire service class. If the max concurrency is hit, queries will enter a queue. A value of -1 indicates unlimited queries, however it is a best practice to set this value so no individual workloads can monopolize the system.

`max_elapsed_time` — The maximum amount of time, in seconds, a query can run. This restriction helps reduce the length of queries holding concurrency slots and allows concurrent workloads using the same service class to all make progress. It is recommended to set `max_elapsed_time` if `max_concurrent_queries` is set to prevent extremely long-running queries from blocking other concurrent queries. If a service class needs to run sets of queries with very different expected elapsed times, it might be an indication that a separate service class is warranted.

Step 3: Set up the Medium Priority User Class

User groups that do not necessarily need the fastest performance fall under the medium priority service class. Higher-priority queries impact the performance of queries in this service class, but these queries receive priority over lower-priority ad hoc workloads.

Example Service Class

SQL

```
CREATE SERVICE CLASS medium_priority_sc scheduling_priority = 10.0, max_concurrent_
```

Example System Catalog View

SQL

```
SELECT * FROM sys.service_classes;

id                                   database_id                          name
-------------------------------------------------------------------------------
42649e8e-ae7f-4de5-bb2b-846d158fcb91 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 Default
06b92c4c-b0b1-4a14-880c-7d257af3b2b5 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 high_prio
bf3a939d-d773-4c7c-9c30-4bc14e7b9422 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 medium_pr
```

`scheduling_priority` — For a “medium” priority, this value should be set to a number in-between the highest priority service class and the default service class.

`max_temp_disk_usage` — From 0-100, this field represents the maximum percentage of unused disk space that a query can see before failing. When setting up the cluster, it is important to keep some disk space unused on a Foundation Node so queries can spill to disk. Note that this percentage is tracked on a system-wide basis per node, not within a service class. As one service class uses its allocation of temporary disk space to fulfill a query, it also reduces the total temporary disk available to all other service classes. Even if those service classes have a max_temp_disk_usage set, if the total available temp disk space is below the amount needed to run a query, it can fail. For example, if a higher-priority service class is using over 50% of the unused disk space during a medium-priority query that needs to spill to disk, the medium-priority query fails during its first attempt to spill without using any disk space. Spill to disk scheduling uses the same priority used for normal execution.

Step 4: Set up the Low Priority Service Class

User groups that are less performance sensitive fall under the low-priority service class. The design goal is for these queries to have minimal effect on other workloads. Examples of users in this service class are analysts who want to run ad hoc queries or processes that run time-insensitive background reports.

Example Service Class

SQL

```
CREATE SERVICE CLASS low_priority_sc
  scheduling_priority = 1.0,
  max_concurrent_queries = 1,
  max_elapsed_time = 60,
  max_rows_returned = 100,
  max_temp_disk_usage = 0,
  cache_max_bytes = 0,
  cache_max_time = 0,
  max_elapsed_time_for_caching = 0;
```

Example System Catalog View

SQL

```
SELECT * FROM sys.service_classes;

id                                   database_id                          name
-----------------------------------------------------------------------------------
42649e8e-ae7f-4de5-bb2b-846d158fcb91 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 Default
06b92c4c-b0b1-4a14-880c-7d257af3b2b5 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 high_prio
bf3a939d-d773-4c7c-9c30-4bc14e7b9422 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 medium_pr
dd1f9f03-edff-4c7e-acd7-32997b6fceaa 23291ad0-ce94-4e1a-bdc9-178463ef6ec4 low_prior
```

`scheduling_priority` — For a low priority, this value should be set to a low number like 1.0. If a service class wants to be lower priority than default queries, it can schedule a value lower than 1.0.

`max_temp_disk_usage` — Setting this limitation to 0 means that the system does not allow any queries in the low-priority service class to spill to disk. In this example, the goal of the designer is to prevent the low-priority service class from impacting more important workloads that might need the entire temp disk space. In other cases, it could be suitable to allow low priority service class queries to use temp disk.

`cache_max_bytes`, `cache_max_time`, `max_elapsed_time_for_caching` — Result set caching is a feature that attempts to cache result sets for commonly used queries. This result set cache is shared across service classes. For a lower-priority user, it makes sense to limit the time allowed in the cache or cap the maximum number of bytes used in the cache so a higher-priority service class can take advantage of it. In this example, the result set

cache is explicitly disabled for low priority users by setting these values to zero. For details, see the **Result Set Caching**.

Step 5: Assign Service Classes to User Groups

After setting up the service classes, you need to assign them to user groups. When a service class is assigned to user groups, users in each user group can run queries under the restrictions set for the given service class. This process can be repeated for all service classes and users.

Examples

SQL

```
/* Create a user group. */
CREATE GROUP low_priority_group;

/* Assign the service class to the user group. */
ALTER GROUP low_priority_group SET service class = low_priority_sc;

/* Add a user to a user group. */
ALTER GROUP low_priority_group ADD USER analyst1;
```

Dynamic Workload Management Priority Adjustment

When a query executes, you can set the priority of a query at the service class level, database session level, or the query level. The system chooses the maximum value from these three priority levels.

While the engine executes a query, the engine can dynamically adjust the priority using a specified adjustment interval. You can change the percentage amount to dynamically adjust query priority using the `priority_adjustment_factor` (or `ADJUSTFACTOR`) and the time period that indicates the frequency for the adjustment of the query priority during query execution using the `priority_adjustment_time` (or `ADJUSTTIME`). You can override this limit at the query level or the session level.

Define the priority at the service level by creating service classes. You can override the priority at the service level by using the priority at the session or query level. For details, see **Users, Groups, and Service Classes**.

Define the priority at the session level using the command line. This code instructs the engine to dynamically adjust the priority every 15 seconds and decrease the query priority by 25%.

```SQL
SET PRIORITY "5.0";
SET ADJUSTTIME 15;
SET ADJUSTFACTOR "0.75";
```

To reset the priority adjustment factor, enter this SQL statement.

```SQL
SET ADJUSTFACTOR RESET;
```

Or, define the priority at the query level in the SQL statement, such as this statement.

```SQL
SELECT * FROM table1 USING
  SCHEDULING_PRIORITY = 10.0,
  PRIORITY_ADJUSTMENT_FACTOR = 1.75,
  PRIORITY_ADJUSTMENT_TIME = 120;
```

Alternatively, you can also specify a service class to use at the query level. This example creates the service class `lower_priority` and assigns it to the query.

```SQL
CREATE SERVICE CLASS lower_priority
  SCHEDULING_PRIORITY = 10.0,
  PRIORITY_ADJUSTMENT_FACTOR = 0.75,
  PRIORITY_ADJUSTMENT_TIME = 120;

SELECT * FROM table1 USING SERVICE CLASS lower_priority;
```

© 2026 Ocient, Inc.

> ⚠ If you do not identify a specific service class for a query with the `USING SERVICE CLASS` clause and the query does not match any service classes that use the `STATEMENT_TEXT` parameter, the query uses a composite of least restrictive parameters allowed by the service classes eligible for the query.

> ⓘ Using a service class at the query level requires a user to be assigned to a user group that is included in that specific service class.

Low-Latency Settings

> ⚠ Low-latency service class settings are intended to reduce runtime overhead for classes of subsecond queries with minimal processing work. Classes of queries that are good candidates utilize indexes on their filters, have low cardinality, and perform minimal work, such as simple aggregations or scalar functions. Low-latency service class settings might hinder query performance when queries have high cardinality or perform computationally intense work. Workload management is key to isolating these simple, low-latency queries to specific service classes where you can apply these optimizations in isolation.

Queries for latency-sensitive applications, such as populating live dashboards and key-value retrieval might benefit from service-class level settings that help the Ocient System to make runtime choices to reduce latency. These choices often help queries with low cardinality and minimal computational work, but they can be detrimental to queries that violate these assumptions. Database administrators should use service classes to isolate low-latency components of a system from computationally intensive components.

A system administrator can turn on one simple or two advanced settings.

Setting	Description	Value
`LOW_LATENCY`	This optional service class setting enables you to control the `HALF_PARALLELISM` and `LOAD_BALANCE_SHUFFLE` settings. When you set this setting to `true`, it is equivalent to setting `HALF_PARALLELISM = true` and `LOAD_BALANCE_SHUFFLE = false`. When you do not set this setting or set it to `false`, the Ocient System clears the `HALF_PARALLELISM` and `LOAD_BALANCE_SHUFFLE` settings.	Optionally set to `true` or `false`. The default value is unset.
`HALF_PARALLELISM`	This optional service class setting controls execution parallelism overrides. When you do not set this setting, the system uses cost-based heuristics to determine runtime parallelism. When you set this	Optionally set to `true` or `false`. The default value is unset.

Setting	Description	Value
	setting to `true`, the system always uses half parallelism for queries that match this service class. When you set this setting to `false`, the system always uses full parallelism for queries that match this service class. Half parallelism can help latency by reducing the amount of administrative work the system needs to do to run a query, but it reduces the computational power allocated to the query.	
`LOAD_BALANCE_SHUFFLE`	This optional service class setting controls load balancing shuffles above I/O operators. When you do not set this setting, the Ocient System uses statistics-based heuristics to determine whether to use load balance shuffles. When you set this setting to `true`, the system always performs a load balance shuffle for queries that match this service class. When you set this setting to `false`, the system never performs a load balance shuffle for queries that match this service class. Load balance shuffles are useful for distributing uneven data distributions on disk, which can occur during live-loading of tables or small cardinality tables with compute-heavy computations. However, this load balancing incurs some system-coordination overhead, which hampers low-latency queries.	Optionally set to `true` or `false`. The default value is unset.

There is further convenience for organizing these parameters in a service class by enclosing `HALF_PARALLELISM` or `LOAD_BALANCE_SHUFFLE` settings in a `LOW_LATENCY` key-value setting map.

SQL

```
CREATE SERVICE CLASS ll_service_class
  LOW_LATENCY = {HALF_PARALLELISM = true,
                 LOAD_BALANCE_SHUFFLE = true};

SELECT * FROM table1 USING SERVICE CLASS ll_service_class;
```

Other Service Class Considerations

Service classes are a powerful tool to properly allocate resources in the database to the users that need them most. When deciding on the restrictions for your individual use case, consider the total number of service classes that the system creates. Some settings

like `max_concurrent_queries` and result set caching values like `cache_max_bytes` and `cache_max_time` can have a cumulative effect on resource availability as the number of service classes increases. For details about result set caching, see **Result Set Caching**.

For example, concurrency limitations are set on an individual service class level. While the intention might be to limit low priority service classes to only a couple of concurrent queries, the total effect of many service classes with a couple of concurrent queries each can start to affect high priority workloads due to the number of concurrent queries running across all lower priority service classes.

Related Links

Workload Management and Service Classes

Data Definition Language (DDL) Statement Reference

SQL Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Workload Management and Service Classes

Service classes define limits on groups in the Ocient® System. You can design service classes using these limits. Use these parameters to control the limits.

Parameter	Description	Values
`MAX_TEMP_DISK_USAGE`	Limit the percentage of temporary disk space used by the total service class relative to the amount of remaining free space. If all running queries for a particular service class exceed the given percentage, queries will fail.	A percentage integer from 0-100.
`MAX_ELAPSED_TIME`	Limit query runtime. The database process kills any queries that exceed this limit.	-1 for unlimited, or a positive integer for the number of seconds.
`MAX_CONCURRENT_QUERIES`	Limit the maximum number of queries that can run concurrently for a given service class. The database process queues additional queries.	-1 for unlimited, or a positive integer for the number of queries.
`MAX_ROWS_RETURNED`	Limit number of rows returned by a query. The database process kills queries that exceed this limit.	-1 for unlimited, or a positive integer for the number of rows.
`MINIMIZE_QUERY_DEBUG_RECORDS`	Stop the load of the `sys.completed_operator_instances` system catalog table and the generation of specific debug logs associated with the query.	Boolean `true` for minimizing the capture of metadata and logs, and `false` or unset (default) for capturing this data.
`SCHEDULING_PRIORITY`	Limit initial effective query priority.	Any decimal value between the `MIN_PRIORITY` and `MAX_PRIORITY`.
`CACHE_MAX_BYTES`	Maximum number of bytes in a result set if the result set can be stored in the cache. This value is the number of bytes in the wire-protocol representation, or what is sent to a client.	-1 for none, or a positive integer for the number of bytes.
`CACHE_MAX_TIME`	Maximum time rows are cached.	-1 for none, or a positive integer for the number of seconds.

© 2026 Ocient, Inc.

Parameter	Description	Values
`MAX_ELAPSED_TIME_FOR_CACHING`	Maximum elapsed time for caching. If you set this parameter, queries that exceed maximum elapsed time return no results, but continue to run in the background until either the query reaches the maximum elapsed time for caching or the query successfully completes and caches the result set. For details, see documentation.	-1 for none, or a positive integer for the number of seconds.
`MAX_COLUMNS_IN_RESULT_SET`	Limit number of columns returned by a query. The database process kills queries that exceed this limit.	-1 for unlimited, or a positive integer for the number of columns.
`PRIORITY_ADJUSTMENT_FACTOR`	Percentage amount to adjust query priority. You can override this limit at the query level or the session level. Formula is: `(new_priority = current_priority * priority_adjustment_factor)`	0 to disable dynamic priority adjustment (default), or a positive double value that is greater than 0. Values less than 1 decrease priority over time until the priority reaches the service class minimum priority, while values greater than 1 increase priority over time until the priority reaches the service class maximum priority.
`PRIORITY_ADJUSTMENT_TIME`	Time period that indicates the frequency for the adjustment of query priority during execution of the query. You can override this limit at the query level or the session level.	0 to disable dynamic priority adjustment (default), or a positive integer (unsigned 32-bit integer) for the number of seconds that is greater than 0.
`MIN_PRIORITY`	Limit the minimum query priority. The current effective priority cannot be smaller than this value.	0 (default) or any decimal value (double) greater than 0.
`MAX_PRIORITY`	Limit the maximum query priority. The current effective priority cannot be greater than this value.	-1 (default) that indicates infinity or any decimal value (double) greater than 0.
`STATEMENT_TEXT`	The system uses the specified service class for any queries that match the specified text pattern. The system uses service classes with statement text before any service classes that lack statement text. If there are multiple service classes with statement text available, the system attempts to match them in alphabetical order.	A string that uses either LIKE or REGEX pattern matching. For example, `STATEMENT_TEXT REGEX ".*my_table.*|.*your_table.*"` would match any queries that include the strings `my_table` or `your_table`.

Parameter	Description	Values
	This matching allows you to set up service classes so that the system can assign certain query types automatically to higher-priority service classes.	
`LOW_LATENCY`	Optional settings that enable you to manage low-latency service classes.	Boolean `true` or `false`, or a Key-Value setting map.
`MEMORY_OPTIMAL_STRATEGY`	Direct the Ocient System to reduce query memory usage, even if the operation might increase execution time. Contact Ocient Support when you change this parameter.	Boolean `true` instructs the system to apply stricter scheduling and execution rules to minimize memory consumption and avoid excessive concurrent usage. Otherwise, setting this parameter to `false` instructs the system to disable such strategies, even if the service class default enables them. By default, this parameter is unset. If you leave this parameter unset, the system reverts to higher-level default behavior or other service class settings.

Service Class Setting Applicability

When a user is present in multiple groups with different service class limits, the system applies the least restrictive limit for each parameter. For example, if User A is in groups G1 and G2, which have service class time limits of 10 seconds and 20 seconds, when User A runs a query, the query has a time limit of 20 seconds. This behavior is slightly different for concurrency. If the service class concurrency limits for the groups the user belongs to are 1 and 2, a query can use a concurrency slot from either service class. A user with multiple service classes can experience a query executing with a slot from one service class but using a time limit or other limits from a different service class. Additionally, the system does not apply service class concurrency limits when a query affects only system tables.

Choose Parameter Values

If you set a service class at the session or query level, the system uses the value set by that service class, with the query-level service class taking precedence.

© 2026 Ocient, Inc.

Otherwise, if you set a value for this parameter at the session or query level, the system uses the most restrictive of the two.

If the Ocient System chooses the value from multiple service classes, the system takes the least restrictive value from all service classes. Then, the system takes the most restrictive of that value and the query or session level value, if it exists.

> This selection does not hold true for parameters related to result set caching: `cache_max_time` and `cache_max_bytes`. For details about the behavior, see **Result Set Caching**.

If you choose a service class using statement text matching, the system compares its value to the query or session level value, and the system does not review any other service classes.

Next Steps

A tutorial for designing a set of service classes for workload management is available in the **Workload Management Walkthrough**.

See **Users, Groups, and Service Classes** for service-class-related DDL statement references, such as creating, altering, and applying a service class.

Related Links

Data Definition Language (DDL) Statement Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Result Set Caching

The OcientAIQ™ Unified Data Platform allows query result sets to be cached on the server and returned quickly on subsequent queries. The service class under which the query executes manages result set caching. As with other service class attributes, you can override this by using keywords on the query.

When to Use Result Set Caching

In general, result set caching is most advantageous for queries that require high processing to produce a small result. For example, a query that uses aggregate functions, such as finding the average over many rows, would be a good candidate for result caching.

To help evaluate whether to use result set caching, you can examine your query metadata, such as storage cost and processing time. To view these metrics and to see which are stored in cache, query the **sys.completed_queries** table.

Query Matching Behavior

Result set caching uses the normalized text of the query (i.e., with comments removed, whitespace reduced, and case normalized) to determine whether two queries match. If they match, the system uses the cached result of a previous query.

> Query matching requires strict text matching. Queries cannot use cached results if they have different values for arguments such as time ranges or other filters.

Configuring Query Caching

You can configure caching behavior for the service class (see **SERVICE CLASS**) or for an individual query (see **USING**).

You can specify configurations for result set caching with these attributes:

© 2026 Ocient, Inc.

`cache_max_bytes`	**Controls whether the database stores the results of any specified query in the cache.** **All queries executed using a service class specifying this attribute have their results stored in the cache if the size of the result is smaller than this value. You can determine the size of a result set in bytes by querying the `bytes_returned` field of the `completed_queries` virtual table. The default value is 0, and thus by default, the database does not cache any results.** **The database caches results in memory on the SQL Nodes. These results are not cached if there is insufficient memory available.**
`cache_max_time`	Affects whether a specified query uses results from the cache rather than executing the query. If there is a cached result with the same query text, executed less than `cache_max_time` seconds ago, the database uses that cached result. The default value is 0, and thus by default, the database does not return any cached results. The cache considers all SQL Nodes, and if a potential cached result is only available on a different SQL Node, the database redirects the query to that node. If you specify the force attribute on the connection, which disables load balancing and redirection, the database considers only cached results on the current node.

When a specified query uses multiple service classes, the system uses the most restrictive configuration values. For example, if a query uses `service_class_a` with `cache_max_bytes` set to 512 and `service_class_b` with `cache_max_bytes` set to 0, the system does not cache the results. This is because 0 is the most restrictive value, and caching is disabled regardless of the result set size.

You can override the values for `cache_max_bytes` and `cache_max_time` at query time by specifying them in the query itself.

Query Override Example

SQL

```
SELECT COUNT(*) FROM TRADES USING CACHE_MAX_BYTES=32, CACHE_MAX_TIME=60;
```

This example states that the query result is cached if it is less than or equal to 32 bytes, and the cache is used if the value stored in the cache is less than 60 seconds old. It is very important to note that attributes specified on the query cannot increase the values from the service classes, only reduce them.

Both `cache_max_bytes` and `cache_max_time` can be explicitly not specified in a service class by using the value -1. -1 is effectively treated as 0 if not overridden. `unspecified`, or -1 values can always be overriden by other service classes or on the query. A typical use case is to:

- Specify `cache_max_bytes` as 0 for service classes used by ad hoc queries, where results should never be cached and consume memory.
- Specify `cache_max_bytes` and `cache_max_time` as -1 on service classes used by repeated production queries. As mentioned, the system treats these as 0 if not overridden.
- Explicitly override `cache_max_bytes` on repeated production queries whose results should be cached.
- Explicitly override `cache_max_time` on repeated production queries whose results can be retrieved from the cache.

Note that because the default schema used for a query can affect the results of the query, the system treats two queries with identical text but making use of different schemas differently.

Finally, the contents of the cache can be examined using the `result_cache` virtual table.

Max Elapsed Time For Caching

The system cancels queries when they exceed the `max_elapsed_time` service class value. An additional service class value, however, allows queries to continue beyond `max_elapsed_time` so that their results can be stored in the cache. The service class (or SQL) can specify `max_elapsed_time_for_caching`, which should be a value larger than `max_elapsed_time`. This value indicates the total time the query can continue executing such that its results are stored in the cache.

`max_elapsed_time_for_caching` and `max_elapsed_time` are both measured in seconds. When a query with `max_elapsed_time_for_caching` exceeds `max_elapsed_time`, a warning indicating `Query exceeded time limit, but will continue to run and be cached (Limited to 200 second(s) for caching from query)` is returned to the JDBC driver, where `max_elapsed_time_for_caching` was set to 200 seconds in this example.

`max_elapsed_time_for_caching` is only supported on queries executed using the JDBC driver.

Manage the Result Set Cache

You can store and retrieve query results from the cache depending on the settings of the `cache_max_bytes` and `cache_max_time` values, either using the SQL statement or the service class specified for the query execution.

To view all cached queries and their service classes, query the **sys.result_cache** system catalog table.

You can invalidate the result set cache using these DDL statements. Use these statements when you perform table updates, such as an `INSERT INTO` SQL statement operation or the execution of one of the loading and transformation functions.

Syntax

Remove the cached result sets for all queries that reference the specified table.

You can invalidate the cached result sets for only tables where you have the `INSERT` privilege. If a cached query references multiple tables, you only need the `INSERT` privilege on any one of those tables.

SQL

```
ALTER TABLE table_name INVALIDATE CACHE [SPREAD_OVER_TIME seconds]
```

Parameter	Data Type	Description
`table_name`	string	Name of the database table.
`seconds`	numeric	Number of seconds that specifies the interval at which to invalidate a section of the result set cache.

Example

This example immediately invalidates the cache and removes the cached result sets for all queries that reference the `employees` table.

SQL

```
ALTER TABLE employees INVALIDATE CACHE;
```

This example invalidates the cache entries for queries that reference the `employees` table for a fraction of the cache every second. Specify the fraction to be 1/10 by using the `spread_over_time` parameter. The database removes 1/10 of the cached result set immediately, then the next 1/10 of the cached result set the subsequent second, and so on. Invalidation causes the re-execution of queries to delay. Spreading the invalidation over time prevents spikes of expensive query executions.

SQL

```
ALTER TABLE employees INVALIDATE CACHE SPREAD_OVER_TIME 10;
```

Syntax

Remove the cached result sets in the specified database.

SQL

```
ALTER DATABASE database_name INVALIDATE CACHE [SPREAD_OVER_TIME seconds]
```

Parameter	Data Type	Description
`database_name`	string	Name of the database.
`seconds`	numeric	Number of seconds that specifies the interval at which to invalidate a section of the result set cache.

Example

This example immediately invalidates the cache and removes the cached result sets in the `test_database` database.

SQL

```
ALTER DATABASE test_database INVALIDATE CACHE;
```

This example invalidates the cache entries in the `test_database` database for a fraction of the cache every second. Specify the fraction to be 1/10 by using the `spread_over_time` parameter. The database removes 1/10 of the cached result set immediately, then the next 1/10 of the cached result set the subsequent second, and so

on. Invalidation causes the re-execution of queries to be delayed. Spreading the invalidation over time prevents spikes of expensive query executions.

```sql
ALTER DATABASE test_database INVALIDATE CACHE SPREAD_OVER_TIME 10;
```

Related Links

Data Definition Language (DDL) Statement Reference

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Object-Type Level Privileges Management

You can assign privileges at different levels of granularity on system and database objects to control access for users in the Ocient® System. The levels are:

- All objects of the specified object type in the system
- All objects of the specified object type in the database
- Specific object

This series of workflows shows how you can manage access to the `VIEW` privilege for all views (where a view is the object type) in a specified database.

Grant Privileges

This workflow creates a group and grants the `VIEW` privilege to all views and the ability to read the views in the database to the group.

1. Create the `view_analyst` group. You can see the created group using the `sys.groups` system catalog table.

```
SQL
CREATE GROUP "view_analyst";
```

2. Grant the `VIEW` privilege for views on the `test` database to the `view_analyst` group.

```
SQL
GRANT VIEW VIEW ON DATABASE "test" TO GROUP "view_analyst";
```

3. Grant the `SELECT` privilege on the database to the group.

© 2026 Ocient, Inc.

SQL

```
GRANT SELECT ON DATABASE "test" TO GROUP "view_analyst";
```

4 View the granted privileges using the `sys.privileges` system catalog table. The Ocient System grants the `USE` privilege on the database implicitly.

SQL

```
SELECT privilege, privilege_target, object_type,
    grantee, grantor
FROM sys.privileges
WHERE grantee = 'view_analyst' and grantor != 'view_analyst';
```

Output

Text

```
privilege                                          privilege_target
-------------------------------------------------------------------------------
VIEW                                               VIEW
USE                                                NULL
SELECT                                             NULL

Fetched 3 rows
```

Verify Privileges

This workflow creates a user, assigns this user to the created group, and creates a view with generated data. The workflow shows the information that the user has access to in the database based on the granted privileges.

1 Create the `test_user` user.

SQL

```
CREATE USER test_user PASSWORD='**';
```

2 Add the user to the `view_analyst` group.

SQL

```
ALTER GROUP "view_analyst" ADD USER test_user;
```

3 Create the `view` view that selects five generated rows.

SQL

```
CREATE VIEW "test"."view" AS (SELECT * FROM sys.dummy5);
```

4 Connect to the database at the `10.10.1.1` IP address with port number `4050` as the `test_user` user.

Shell

```
connect to jdbc:ocient://10.10.1.1:4050/test;user=test_user;password=**;
```

5 View the privileges of the user using the `sys.privileges` system catalog table. This user has no privileges of their own.

SQL

```
SELECT privilege, privilege_target, object_type, grantee, grantor
FROM sys.privileges
WHERE grantee = 'test_user' AND grantor != 'test_user';
```

Output

```
Text
privilege                                          privilege_target
-------------------------------------------------------------------------------
Fetched 0 rows
Warning: No data found
```

6 See the metadata for views that the user has access to within this database. The user has access to the `test` view only.

```
SQL
SHOW VIEWS;
```

Output

```
Text
table_catalog                                      table_schema
-------------------------------------------------------------------------------
test                                               test

Fetched 1 row
```

7 Verify that the user can select the data in the `view` view.

```
SQL
SELECT * FROM "test"."view";
```

Output

Text

```
c1
-----------
1
2
3
4
5

Fetched 5 rows
```

Related Links

Data Control Language (DCL) Statement Reference

System Catalog

Generate Tables Using sys.dummy

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Manage Users, Groups, and Roles

The OcientAIQ™ Unified Data Platform can have one or more databases, each of which can run multiple applications. Administrators have the ability to define and manage users in Ocient® to meet the needs of a multi-tenant environment. Through user, group, and role management capabilities in Ocient, administrators can make sure that users only have privileges to the objects and databases they need. They can also set security settings for passwords. For details, see **Database Password Security Settings**.

Users

Users in Ocient are represented by a username and password. They can have privileges to the system or one or more databases within the system. Privileges of a user can be granted directly to the user or inherited through groups or roles. Users can also belong to zero or many groups and can have zero or many pre-defined roles in the system. A user identifier might be used by someone logging directly into Ocient through the Ocient command-line interface (CLI) or by an application connecting to a database.

The `sys.users` system catalog table displays information about users in the database.

Fully Qualified User Names

A Fully Qualified User Name (FQUN) is the combination of a username and the database on which the user was created. For example: `jsmith@metrics`. It is not mandatory for users to use an FQUN to connect to their own database.

An FQUN uniquely identifies a user and follows the pattern `"<user_name>@<database_name>"`, where `<user_name>` is the username and `<database_name>` is the name of the database. Use an FQUN to reference a user associated with a database other than the one you are connected to.

© 2026 Ocient, Inc.

user id	user name	database name	fully qualified user name (FQUN)
000	alice	db1	alice@db1
001	alice	db2	alice@db2
002	jimmy	system	jimmy@system

If you do not specify an FQUN when you reference a user object, Ocient assumes the user is associated with the database of the active connection.

This example shows the intended use of the FQUN and how the database is implicitly selected when you connect to the database.

Given an Ocient System with two databases, `operations` and `metrics`, create two administrator users and give administrator rights to each on their respective database.

FQUN	Database	Password
admin@operations	operations	password1
admin@metrics	metrics	password2

These are two separate users. Those users can set up their database for the remainder of their users, including:

- Create users with various roles and privileges.
- Set up SSO controls for each database.
- Create tables.
- Create pipelines to load data.
- Create service classes.

It is not mandatory for users to use an FQUN to connect to their database. The first user can connect to the `operations` database as the user `admin` with `password1`. The system treats this login as the user `admin@operations`.

Similarly, the second user can connect to the `metrics` database as the user `admin` with `password2`.

© 2026 Ocient, Inc.

In this example, neither user has to use their FQUN, because they only work with the database described by their FQUN.

When you have permission to connect to more than one database, you must use your FQUN for any database that does not match your FQUN. For example, if the user `jsmith@metrics` also has access to the `auctions` database, then that user can connect with their FQUN `jsmith@metrics`.

Groups

A group is a collection of users that has a set of privileges assigned to it. Administrators can create groups that represent an application or a specific job role for users within the database. For example, a group named `US_Analysts` might only have privileges that allow access to data from the US. Groups belong to individual databases. In other words, you cannot define a group with privileges that span multiple databases.

A service class defines a set of limits on groups. The database applies service classes on a per-group basis. By default, all groups are in the `DEFAULT` service class that has no limits. For details, see **Workload Management and Service Classes**.

The `sys.groups` system catalog table displays information about groups in the database. You can also use the **CURRENT_GROUPS** function to view the fully-qualified names of the groups in the database.

Roles

Similar to groups, users can inherit privileges by being granted one of the pre-defined roles in Ocient. The system pre-defines the names of and privileges assigned to roles in Ocient. Roles can apply to the Ocient System or one of the user-defined databases.

System Roles

- **Security Administrator** — Can read, create, and modify all users or groups.
- **System Administrator** — Can read, create, and modify system objects such as tables, clusters, databases, etc. Can also kill running queries system-wide.
- **System Analyst** — Read-only access to the entire system.

You can grant system roles to individual users whether they are members of a specific User Database or defined on `system`.

Database Roles

- **DB Public** — Can grant and distribute privileges using this `PUBLIC` role. Every user created in the database has this role.
- **Database Administrator** — Can read, create, and modify database objects such as tables, clusters, databases, etc. The database administrator can also create users for the database and kill running queries.
- **Database Analyst** — Read-only rights to database objects and data within the database.

These roles are named with the convention `"<database_name> <role_name>"`, where `<database_name>` is the name of the database and `<role_name>` is the name of the role. The `sys.roles` system catalog table displays this information.

To learn about the syntax for granting privileges, see **Data Control Language (DCL) Statement Reference**.

Use DCL to Assign User Privileges

To assign or revoke privileges to users, administrators can use Data Control Language (DCL) statements. You can execute DCL statements like any other SQL statements against the Ocient System and databases. These statements allow for quick and easy modification of user access. For details, see the **Data Control Language (DCL) Statement Reference**.

System Database

Every Ocient System includes an administration interface named `system`.

The Ocient System lists `system` as a database, but it is not actually an active database. `system` serves as a connection point for administration, particularly during the initial system setup to configure other databases, storage, user access, and other configurations.

© 2026 Ocient, Inc.

When connected to `system`, users can see metadata for all User Databases to which they have rights. Only users who have System Administrator or System Analyst roles can see metadata for Nodes, Node Service Roles, storage spaces, and storage clusters.

A user can log into `system` if any of these conditions are true:

- You create the user on that database — e.g., `johndoe@system.`
- You assign the user the role of either System Administrator or System Analyst.
- The non-system user (e.g., `myuser@some_database`) has been granted `SYSTEM` privileges through DCL, e.g., `GRANT USE on SYSTEM to USER some_user@some_database;`

User Databases

User Databases are defined by users. These databases can house tables, views, and data. You can create and scope users to a database to allow multi-tenancy on an Ocient System or limit the ability of users to see the data or schemas defined in other User Databases.

When you connect to a user database, you can query the user data in this database and see the metadata for this database in system tables.

Visibility of Catalog Components Based on User Privileges

All system catalog tables (e.g., `sys.tables`, `sys.users`, `sys.databases`) are queryable to all users of Ocient regardless of their connection. The contents of these tables can change based on the rights that the individual user has. For example, if a user is blocked from a specific database, then a query on the system catalog by that user would not show that database or any of its objects.

Related Links

Data Control Language (DCL) Statement Reference

Users, Groups, and Service Classes

Database Password Security Settings

SQL Reference

© 2026 Ocient All Rights Reserved

Authentication Methods

Ocient® offers two authentication methods to access the database: Password authentication and Single Sign On (SSO) authentication. You can use these independently or as an organization according to your security and access control needs. This section explains each authentication method and how it can be configured on Ocient.

Password-Based Authentication

Users created using DCL are required to set a password that the system uses for authentication when connecting to the database.

Administrators can use **Database Password Security Settings** to manage settings such as minimum password length and password expiration policies.

Fully Qualified User Name (FQUN)

To connect to the database, you must provide an FQUN. For database users, the FQUN has this form.

```
Text
<user_name>@<database>
```

For example, the FQUN of the user `alice`, a member of the database `example_database`.

```
Text
alice@example_database
```

How To Connect using Password Authentication

Set the FQUN in the connection string along with the password of the user. For example, this code is the command `alice` uses to connect to `example_database` using the Ocient JDBC driver.

```
Text
connect to jdbc:ocient://<host>:<port>/example_database;user=alice@example_database;passwor
```

© 2026 Ocient, Inc.

The following Ocient driver properties must be set when connecting using password authentication:

- user — `alice@example_database`
- password — “

Single Sign On (SSO)

The Ocient System allows administrators to add a Single Sign-On integration, allowing users to authenticate using an external Identity Provider (idP). A database, including the `system` database, can have 0 or 1 SSO integrations.

> The existence of an SSO integration has no effect on users who authenticate with the Password Authentication flow.

Supported Protocols:

The Ocient System supports the following SSO protocols:

Protocol	Description
OpenID Connect	OpenID Connect (OIDC) is a simple JSON-/REST-based identity protocol built on top of the OAuth 2.0 protocol. It enables client applications to rely on the authentication performed by an OpenID Connect Provider to verify the identity of a user. Clients can also obtain basic profile information about a user in an interoperable and REST-like manner from OpenID Connect Providers.

OpenID Connect

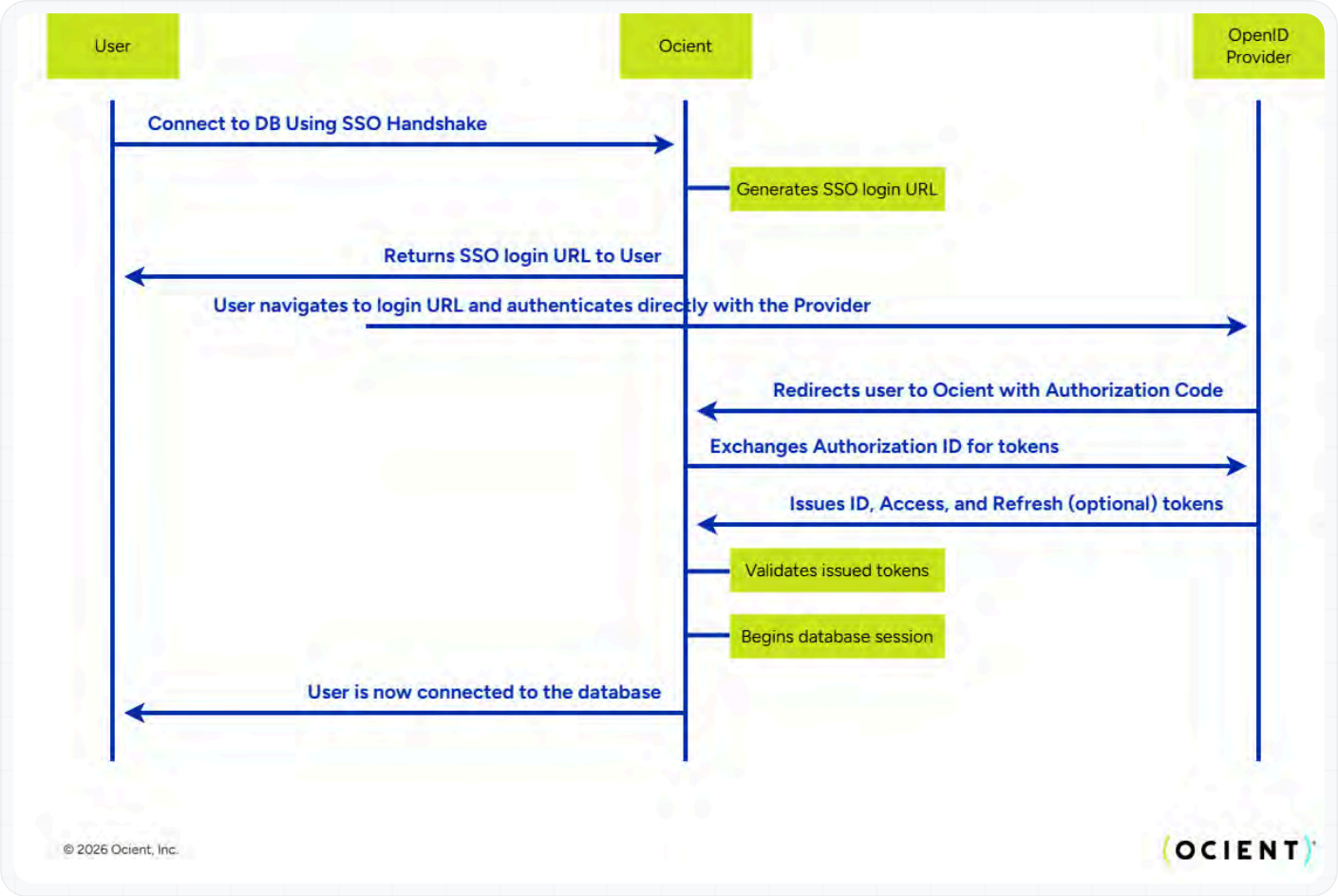
There are two flows that allow users to connect to the database using an external OpenID Connect Provider.

Authorization Code Flow

This is the most commonly used authentication method and makes use of the **OAuth 2.0 Authorization Code Grant**.

When you connect to the database, the Ocient driver directs you to the authorization endpoint of the OpenID Provider, where you log in with your SSO credentials. After the Provider authenticates you, it redirects you back to the database with a temporary code, which the database exchanges for an ID token, Access token, and optionally, a Refresh token.

See this image for a sequence diagram of the flow.



Connect using the Authorization Code Flow

The client handshake method must be set to SSO, and the connection username and password set to the empty string.

For example, this code is the command any user would use to connect to the `example_database` database using the Ocient JDBC driver.

Text

```
connect to jdbc:ocient://<host>:<port>/example_database;handshake=SSO;user=;password=;
```

These Ocient driver properties must be set when connecting using the Authorization Code flow:

- handshake — `SSO`
- user — `<empty_string>`
- password — `<empty_string>`

ⓘ The `user` and `password` properties are required and must be set to the empty string.

SSO Token Flow

Users can provide either an **ID Token** or an **Access Token** issued by the OpenID Provider to connect to the database.

The provided ID or Access token must contain these claims:

- iss — The issuer must match the issuer contained in the discovery document of the OpenID Provider.
- aud — The token audience must contain the `client_id` used when configuring the OpenID Provider in the database.

> Ocient recommends including the `email` scope in any token provided to Ocient. The system uses the value of the email claim to identify the user in the database audit trails.
>
> Ocient supports these signing algorithms for ID Tokens:
>
> - RSA 256
> - RSA 384
> - RSA 512

Connect using the SSO Token Flow

These Ocient driver properties must be set when connecting using an OpenID token:

- handshake — `SSO`
- user — `id_token` or `access_token`
- password — `<token_data>`

The handshake must be set to SSO, with the username set to the token type and the password set to the token payload.

For example, this format is the JDBC command to connect to the `example_database` database using an ID Token.

Text

```
connect to jdbc:ocient://<host>:<port>/example_database;handshake=SSO;user=id_token;passwor
```

This format is the JDBC command to connect to `example_database` database using an Access Token.

Text

```
connect to jdbc:ocient://<host>:<port>/example_database;handshake=SSO;user=access_token;pass
```

Configure the OpenID Provider

Before users can authenticate using SSO, you must register the database with the OpenID Provider. This process varies depending on the provider, but typically, the steps for doing so are:

1. Create an application for the Ocient System in the Provider. Select the “Native Application” application type.
2. Enable the Authorization Code, Refresh Token, and Device Authorization grant types.
3. Grant permission to the appropriate users to use the newly created application.
4. Enter this Redirect URI.

Application	Redirect URI
JDBC Driver	`http://localhost:7050/ocient/oauth2/v1/callback`
OpenAPI or UI	`https://<advertised_ip>/v1/callback`

For the OpenAPI or UI application, the redirect URI must use `https` format.

The connectivity pool defines the OpenAPI or WebUI `advertized_ip` setting. For more information, see **CONNECTIVITY POOL**.

Record the token `issuer` and `client_id` of the newly created application. You need these values to configure the Ocient System.

SSO Parameters

These SSO parameters are configurable by using DDL statements.

Name	Required	Value Type	Default Value	Description
`issuer`	Yes	string	None	The **complete URL** for the OAuth 2.0 and OpenID Connect Authorization Server. This property value is the expected `\"iss\"` claim in access tokens validated by the database.

Name	Required	Value Type	Default Value	Description
`client_id`	Yes	string	None	The **client identifier** as registered with the OpenID Provider.
`client_secret`	Yes	string	None	The **client secret** as registered with the OpenID Provider. This property is required for some SSO workflows.
`default_group`	Yes	string	None	The group users are assigned if no group is specified in the `group_claim_mappings` property for the OpenID Provider.
`redirect_host`	No	string	Depends on the connector (pyocient or JDBC).	Specifies the host name during SSO redirection.
`redirect_ssl`	No	Boolean	Depends on the connector (pyocient or JDBC).	Set to `true` to enable SSL callback during SSO redirection (i.e., redirect uses `https`). Set to `false` to disable (i.e., redirect uses `http`).
`disabled`	No	Boolean	`false`	Set to `true` to disable the OIDC integration for maintenance temporarily. ⚠ If you set this property to `true`, all authentication requests using this connection fail.
`enable_id_token_authentication`	No	Boolean	`false`	Set this property to `true` if the identifier token also contains the authorization token. In most circumstances, this option is necessary only for machine-to-machine connections without user interaction, such as a server using a script to connect to an Ocient System.
`user_id_claims`	No	list of strings (e.g., value [, ...]	[“email”]	Set the identifier token claims used to identify users in audit trails. If you do not set a value, the system uses the `"email"` claim if it is

Name	Required	Value Type	Default Value	Description
				present, otherwise it uses `["iss", "sub"]`.
`additional_scopes`	No	list of strings (e.g., value [, ...]	[]	Specifies additional scopes to request when executing the Authorization Code Flow.
`additional_audiences`	No	list of strings (e.g., value [, ...]	[]	Specifies additional audiences to accept when validating tokens. This property is useful for authorization servers without the token exchange capability. Each additional audience is a case-sensitive URL from a provider, similar to the `issuer` value.
`groups_claim_ids`	No	list of strings (e.g., value [, ...]	[]	Specifies the token claims that can be used to map the user to a Database group. If you specify the `groups_claim_mappings` property, you must also specify the `groups_claim_ids` property.
`groups_claim_mappings`	No	map of strings (e.g., key = value [, ...])		Specifies mappings from the Provider group to the Database group.
`roles_claim_ids`	No	list of strings (e.g., value [, ...]	[]	Specifies the token claims that can be used to map the user to a Database role. If you specify the `roles_claim_mappings` property, you must also specify the `roles_claim_ids` property.
`roles_claim_mappings`	No	map of strings (e.g., key = value [, ...])		Specifies mappings from the Provider role to the Database role.
`allowed_groups`	No	list of strings (e.g., value [, ...]	[]	Specifies a list of external identity provider groups that are permitted to authenticate through this SSO integration. If you specify the `allowed_groups`

Name	Required	Value Type	Default Value	Description
				property, only users who are members of the specified identify provider groups can access the Ocient System. If this property is empty or you do not specify it, then group-based filtering is disabled, and the system allows all authenticated users from the provider unless they are explicitly blocked. Group names must match exactly as they appear in the claims of the identify provider group.
`allowed_roles`	No	list of strings (e.g., value [, ...]	[]	Defines a list of external identity provider roles that are authorized to access the Ocient System through this SSO integration. Only users assigned to the specified roles in the external provider can authenticate. If this property is empty or you do not specify it, role-based access control is disabled for this integration. Role names must correspond exactly to the role claims provided by the identity provider in the authentication response.
`blocked_groups`	No	list of strings (e.g., value [, ...]	[]	Specifies a list of external identity provider groups that are explicitly denied access through this SSO integration. The system blocks users who are members of any specified group in the external identity provider from authenticating, regardless of other permissions they have. Setting this property takes precedence over the `allowed_groups` property if a user belongs

Name	Required	Value Type	Default Value	Description
				to both an allowed group and a blocked group. Group names must match exactly as they appear in the claims of the identify provider group.
`blocked_roles`	No	list of strings (e.g., value [, ...]	[]	Defines a list of external identity provider roles that are explicitly prohibited from accessing the Ocient System through this SSO integration. The system denies access to users assigned to any of the specified roles, overriding any other access permissions. Setting this property takes precedence over the `allowed_roles` property in cases where a user is in both an allowed role and a blocked role. Role names must correspond exactly to the role claims of the identity provider group.
`allow_offline_access`	No	Boolean	`false`	When the `allow_offline_access` property is `true`, Ocient requests offline access from the OpenID Connect identity provider by including `access_type=offline` and `prompt=consent` in the authorization URL.

After all parameters have been retrieved, a database administrator can execute a DDL SQL statement using this syntax.

SQL

```
ALTER DATABASE database
    { SET | ALTER } SSO INTEGRATION sso_protocol <sso_property> [, ...]

<sso_property> ::=
    -- literal or string
    property_name = value |
    -- list
    property_name = [ value [, ...] ] |
    -- map
    property_name = { key = value [, ...] }   # map
```

SQL

```
ALTER DATABASE example_database
    SET SSO INTEGRATION oidc
        issuer = "https://example.com",
        client_id = "xxxxxx",
        default_group = "default",
        additional_scopes = [email, groups, profile, offline_access],
        user_id_claims = [claim1, claim2],
        groups_claim_ids = [claim3],
        groups_claim_mappings = { idp_group1 = ocient_group1, idp_group2 = ocient_group2 };
```

- The user must be a database administrator (or have the “Security Administrator” role for the “system” database).
- The user must be connected to the database.
- Any String key or value can be placed between double quotations (e.g., `default_group` OR `"default_group"`).
- String values that contain characters other than `([a-zA-z] | [0-9] | '_')` must be placed between double quotations (e.g., `this.is.a.complex-$tring` ⇒ `"this.is.a.complex-$tring"`).
- The `NULL` value can be used to clear existing configurations of `LIST` or `MAP` properties (e.g., `user_claim_ids = NULL`).

To remove the connection, execute this SQL statement.

SQL

```
ALTER DATABASE example_database REMOVE SSO INTEGRATION;
```

The OpenID Connect protocol requires TLS for communication between the end user, the database, and the OpenID Provider. A valid TLS key and certificate pair must be configured on the SQL Nodes, which behave as a server during the Authorization Code Flow.

Set Up Machine-To-Machine SSO Integration

You can use DDL SQL statements to add client credentials for SSO connection to individual databases or to a connectivity pool that operates across SQL Nodes.

These steps set up SSO client credentials on a database, but can also apply to connectivity pools.

1 Create an SSO protocol.

Use the **CREATE SSO INTEGRATION** SQL statement to make an SSO integration protocol with your preferred configuration. This configuration must include these properties:

- The `enable_id_token_authentication` SSO property must be set to `true`.
- The `additional_audiences` SSO property must use the URL from the OIDC provider, similar to the `issuer` value.

SQL

```
CREATE SSO INTEGRATION sso_test PROTOCOL oidc
    issuer = "https://accounts.google.com",
    client_id = example_database_app_id,
    default_group = example_database_group,
    enable_id_token_authentication = true,
    additional_audiences = ["https://accounts.google.com"];
```

2 Assign the SSO protocol.

Use the **ALTER DATABASE SET SSO INTEGRATION** SQL statement to integrate the SSO protocol into a database. For connectivity pools, use the **ALTER CONNECTIVITY_POOL SET SSO INTEGRATION** SQL statement.

SQL

```
ALTER DATABASE example_database
    SET SSO INTEGRATION sso_test;
```

3 **Connect to your system.**

Use a JDBC connection string that includes the access token value `<access_token>`. This string connects to the local host at port number `4050`, uses the SSO protocol, and with the username `id_token`.

Shell

```
CONNECT TO jdbc:ocient://localhost:4050/system;handshake=SSO;user=id_token;password=<a
```

Set Up Cross-Database SSO Integration

Set up authentication using SSO integration across databases. Identify the database that has SSO integration using the `username@database` format, where `username` is the access or identifier token and `database` is the database for the SSO integration. Each SSO integration has a client identifier `client_id`. You must obtain the access token using the same client identifier used in the SSO integration for authentication.

When you authenticate using SSO integration as a user, the system assigns you to a group defined in the integration. Such users always have groups qualified with the SSO integration of the database. These groups have fully qualified names such as `sso_users@system`, where `sso_users` is the name of the group and `system` is the name of the database. You can use a fully qualified group name from any database. Without the `@database` qualification, the referenced group specifies the current database. To grant cross-database privileges to such groups, you must reference them using their fully qualified names. You can find a list of the groups the current user is a member of by using the **CURRENT_GROUPS** function.

To manage privileges and groups, use the `GRANT`, `REVOKE`, `ALTER GROUP`, and `DROP GROUP` SQL statements. For details about privileges, see **Data Control Language (DCL) Statement Reference**.

Requirements

- Configure the SSO integration in the source database.
- The SSO user must be a member of a group that has the USE privilege on the target database. If the SSO user is not a member of any groups, the default group for the SSO integration must have the USE privilege on the target database.
- The token (either an access or an identifier token) must be valid for the SSO integration in the source database.

Create a Cross-Database SSO Integration

These steps guide you through setting up a cross-database SSO integration.

1. Create a default group `sso_users`.

```
SQL

CREATE GROUP sso_users;
```

2. Create the SSO integration `test_sso` in the source database using the `CREATE SSO INTEGRATION` SQL statement. Specify your issuer and client identifier for the `issuer` and `client_id` parameters.

```
SQL

CREATE SSO INTEGRATION test_sso PROTOCOL OIDC
    enable_id_token_authentication = true,
    issuer = 'https://your-idp.com/oidc',
    client_id = 'your_client_id',
    default_group = sso_users,
    additional_scopes = [profile, email, groups, offline_access],
    protocol_version = 'v1',
    groups_claim_ids = [groups];
```

3. Set the SSO integration as the default for the database `source_db`. This step is optional. To set up SSO integration for the `system` database, substitute `source_db` with `system`.

```
SQL

ALTER DATABASE source_db SET SSO INTEGRATION test_sso;
```

© 2026 Ocient, Inc.

4 Grant cross-database privileges to the target database `target_db` for the default group `sso_users`.

SQL

```
GRANT USE ON DATABASE target_db TO GROUP sso_users;
```

5 Connect to the target database `target_db` at the local host with port number `4050` using the JDBC driver with an access token on the source database `access_token@source_db`. Specify a password token.

Text

```
Properties props = new Properties();
props.setProperty("user", "access_token@source_db");
props.setProperty("password", token);
props.setProperty("handshake", "sso");
Connection conn = DriverManager.getConnection("jdbc:ocient://host:4050/target_db", pro
```

Or, connect using the device flow by using the name of the source database `@source_db`.

Text

```
Properties props = new Properties();
props.setProperty("user", "@source_db");
props.setProperty("handshake", "sso");
props.setProperty("ssoOAuthFlow", "deviceGrant");
Connection conn = DriverManager.getConnection("jdbc:ocient://host:4050/target_db", pro
```

6 Verify your connection using these queries:

Check the current user using the **CURRENT_USER** function.

SQL

```
SELECT CURRENT_USER();
```

Check the current groups using the CURRENT_GROUPS function.

```sql
SELECT CURRENT_GROUPS();
```

Check the current database using the **CURRENT_DATABASE** function.

```sql
SELECT CURRENT_DATABASE();
```

View session information using the `sys.sessions` system catalog table and the identifier of the current session by using the **CURRENT_SESSION_ID** function.

```sql
SELECT * FROM sys.sessions WHERE id = CURRENT_SESSION_ID();
```

You can also retrieve information about the defined groups using the `sys.groups` system catalog table.

Driver Connection Properties

These tables describe the connection properties you can specify.

pyocient Parameters

This table lists the DNS parameters that apply to SSO integration. For details, see **Ocient Python Module: pyocient**.

Parameter	Description	Example
`handshake`	Authentication method to use for connection.	`sso`
`ssoOAuthFlow`	Type of OAuth flow.	`deviceGrant` or `authenticationCode`
`identityprovider`	The name of the SSO integration.	`test_sso`

JDBC Driver Properties

This table lists the JDBC connection properties that apply to SSO integration. For details, see **CONNECT**.

Property	Description	Example
`handshake`	Authentication method to use for connection.	`sso`
`ssoOAuthFlow`	Type of OAuth flow.	`deviceGrant` or `authenticationCode`
`identityprovider`	The name of the SSO integration.	`test_sso`

User Access Control and Workload Management with SSO

The database maintains an internal User Access Control model consisting of groups, roles, and service classes. The database relies on group or role membership to determine the privileges of a user or service class. The following integration properties allow administrators to grant SSO users membership to a Database group or role:

- The `groups_claim_ids` and `roles_claim_ids` properties define the token claims that specify the Provider-defined groups or roles of the user.
- The `groups_claim_mappings` and `roles_claim_mappings` properties define mappings between a Provider and Database groups or roles.
- The `additional_scopes` property should include any request scopes needed for the Provider to include the group or role claims in tokens it issues.

> Because SSO users can map to multiple groups, when choosing which group to assign creator privileges, the system follows this criteria:
>
> - Filter by authorization using only groups with CREATE privileges on the relevant schema or database.
> - When multiple authorized groups exist at different levels, the system uses hierarchy precedence to ensure that privileges are assigned at the most specific scope. The enforced hierarchy is that schema-level group privileges take precedence over database-level group privileges.
> - If multiple groups match at the same level, the system selects the group name by using alphabetical order.
>
> For example, if an SSO user belongs to the `analytics_team` group (with CREATE privileges at the database level) and `finance_schema_admins` group (with CREATE privileges at the schema level), and creates a table in the `finance` schema, the system assigns creator privileges to `finance_schema_admins`.
>
> For details about privileges, see **Data Control Language (DCL) Statement Reference**.

Consider an example where the Provider lists all groups the user belongs to in the token claim `"user_groups"` when the application requests the `"example_scope"` scope. In this scenario, the administrator wants to associate the Provider Group `"Example Provider Group"` with the Database Role `"example_database Analyst"`.

SQL

```
ALTER DATABASE example_database
    ALTER SSO INTEGRATION oidc

        /* Tell the Database to request the "profile", "email" and "offline_access" scopes
        additional_scopes = [ profile, email, offline_access ],

        /* Tell the Database to look for the "user_groups" token claim to establish role mer
        roles_claim_ids = [ user_groups ],

        /* Map provider group "Example Provider Group" to database role "example_database Ar
        roles_claim_mappings = { "Example Provider Group" = "example_database Analyst" };
```

By default, the database grants all users membership to the “Public Role” of the database for the current connection. Unlike other default roles, you can grant additional privileges to the “Public Role”.

Additionally, database administrators can revoke access to users with membership to any of the specified Provider groups or roles. To revoke access, execute this statement.

SQL

```
ALTER DATABASE example_database
    ALTER SSO INTEGRATION oidc
        blocked_groups = [ <provider_group_name> [, ...] ],
        blocked_roles  = [ <provider_role_name>  [, ...] ];
```

The user must have the ALTER privilege on all Ocient-defined groups or roles specified by `groups_claim_mappings`, `roles_claim_mappings`, `blocked_groups`, and `blocked_roles`.

User Claim Identifiers (IDs)

By default, audit trails, including but not limited to system-level log messages, identify users connecting using SSO by the `email` claim of the ID Token. If the `email` claim is not present in the token, the identifier is `iss` and `sub` ID Token claims separated by `::`, for example: `"https://ocient.okta.com::00u5rslndgXm9Ey7y5d7"`.

To change the FQUN, alter the `user_claim_ids` SSO integration property.

For example, to change the FQUN to a combination of the `first_name` and `last_name` claim values, execute this SQL statement.

SQL

```
ALTER DATABASE example_database
    ALTER SSO INTEGRATION oidc
        user_claim_ids = [first_name, last_name, email];
```

After executing this statement, users are identified with the `<first_name>::<last_name>::<email>` format. (For example: `John::Smith::jsmith@example.com`)

> Updating the `user_claim_ids` property has no effect on users that are already connected to the database.

System Catalog Tables

Properties that do not contain sensitive data can be viewed using the `sys.oidc_integrations` virtual table. In addition to the configurable properties, a `database_id` column is included in the schema.

To view the configuration for a specific database (if one exists), execute this SQL statement.

SQL

```
SELECT *
    FROM sys.databases
    JOIN sys.oidc_integrations
    ON databases.id = oidc_integrations.database_id
    WHERE databases.name = 'example_database';
```

> The user must be a database administrator to view the OpenID Connect properties.

Related Links

Data Definition Language (DDL) Statement Reference

SQL Reference

System Catalog

© 2026 Ocient All Rights Reserved

Database Password Security Settings

You can manage password security in the OcientAIQ™ Unified Data Platform by using a variety of settings and managing user states.

Password Security Settings

Ocient® includes password security settings to meet specific security requirements or preferences. Five settings are available to configure your password security.

Security Setting	Description
`password_minimum_length`	The minimum length of passwords. The maximum password length supported by the Ocient System is 32.
`password_complexity_level`	An integer value representing the password complexity algorithm. Supported values are: `1` — Level 1. The password must contain at least one upper case character, lower case character, and number. `2` — Level 2. In addition to the requirements specified in level 1, the password must contain at least one non-alphanumeric character.
`password_no_repeat_count`	The number of unique passwords that a user must use before they can reuse a password. Even if this setting is set to `0`, when the system determines that the password lifetime has been reached, you must change the password to a unique value.
`password_lifetime_days`	The password must be changed after this number of days. After the password is older than this period, the user changes to the `PASSWORD_EXPIRED` state on their next login.
`password_invalid_attempt_limit`	The number of login attempts with an invalid password before a user changes to the `DISABLED` state.

ⓘ Except for the `password_no_repeat_count` setting, a value of 0 for any of these settings means that the system ignores that setting.

Password Security Setting Hierarchy and Precedence

You can set all these settings at the system, database, or group levels. The Ocient System uses the most restrictive value. For example, if `password_minimum_length` is `8`

© 2026 Ocient, Inc.

at the system level, `10` at the database level, and `12` at the group level, the system applies the value `12` to the user.

> You can only add a user to a group after you create the user in the system. Upon user creation, `password_minimum_length`, `password_complexity_level`, and `password_no_repeat_count` settings are based only on system- and database-level settings.

System Catalog Table for Security Settings

The `sys.security_settings` system catalog table shows current security settings. The table contains settings for only databases and groups if any settings are non-zero.

After setting the `password_invalid_attempt_limit` value, you can inspect the value using the `sys.security_settings` system catalog table.

SQL

```
SELECT password_invalid_attempt_limit FROM sys.security_settings;
```

> The `sys.users` system catalog table contains information about users, their current state, and details about their security status, such as the last time the password was updated or the number of failed login attempts.

Password Recovery

The Ocient System does not enable you to recover a password. If you forget your password, contact the user who has the Security Administrator or Database Administrator role. Users with those roles can set a new password.

User States

Ocient local users, not SSO-based users, are in one of these states:

- `ENABLED`
- `DISABLED`
- `PASSWORD_EXPIRED`

`ENABLED` State

Enabled users have normal access privileges to the system. They can connect and execute SQL statements. This state is the default state for all users.

`DISABLED` State

A disabled user cannot connect to the system, and if they are currently connected, they cannot execute any SQL statements. Users become disabled automatically if they exceed the `password_invalid` number of failed password attempts.

An administrator must set the state of the user to `ENABLED` for the user to resume access.

`PASSWORD_EXPIRED` State

This state enables a user to log on. However, the only SQL statement they can execute is `ALTER USER <example_user> SET PASSWORD='yyyy'`. The user receives a warning that their password is expired when they authenticate.

Whenever the `password_lifetime_days` number of days has elapsed after the last time the password changed, the system automatically transitions a user to the `PASSWORD_EXPIRED` state on their next login.

You can inspect the last password change timestamp in the `password_updated_at` column in the `sys.users` system catalog table.

Change User State

To change the state of the user to these different states, use the **ALTER USER** SQL statement.

Related Links

Authentication Methods

Data Control Language (DCL) Statement Reference

ALTER USER

Cluster and Node Management

Users, Groups, and Service Classes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

TimeKeys and Clustering Keys

Segment keys define how an Ocient® System partitions data on disk based on particular columns in a database. These keys allow the system to quickly filter rows and skip irrelevant records for queries involving those columns.

For this reason, segment keys are integral tools for optimizing query performance for pivotal columns. Although segment keys are optional, deploying them on every database table in your system is a good practice because they require practically no additional processing or storage resources.

Ocient supports two types of segment keys:

- **TimeKey** — A segment key that partitions data based on a single time-series data column.
- **Clustering Key** — A series of columns that are frequently queried together. The system subdivides these segments on disk for faster reference.

Together, these segment keys form the basis for the Ocient time-cluster storage engine, which subdivides data and metadata for optimal performance.

Attributes of Segment Keys

Advantages

Ocient segment keys share these important advantages:

- Improved performance for query filtering and record retrieval.
- Minimal resource demand for storage or input and output (I/O).
- Easy to use. Without sophisticated system knowledge, you can create segment keys with basic data definition language (DDL) statements.

Limitations

There are no significant downsides to using segment keys. However, they have some limitations for deployment:

© 2026 Ocient, Inc.

- You can implement segment keys only upon table creation as part of the initial **CREATE TABLE** statement. You cannot add them later to a table.
- Each database table can support only one key of each type. For faster performance on additional columns, you can also add clustering indexes and other secondary indexes. For details, see **Clustering Indexes** and **Secondary Indexes**.
- Segment keys are restricted to specific column data types.

TimeKeys

TimeKeys partition data based on a single time-series column in a database table. This column should normally be either a DATE or TIMESTAMP data type.

For optimal performance, you can configure a TimeKey® column with two arguments that comprise a time bucket. The time bucket is a unit of time combined with a number value (e.g., `HOUR, 2` or `DAY, 1`). This value sets the granularity for the partitioning.

In general, this value should match the level of filtering that you expect your queries to request. For example, a database table you expect to query for transactions by minute should probably have a small TimeKey bucket value, such as an hour. If you expect your queries to filter by broader time values, your table can have a TimeKey bucket of one day or larger.

The Ocient System uses the TimeKey and the bucket granularity to partition rows into segments that incorporate a minimum and maximum time value for all rows. For queries that involve time filtering, the TimeKey allows an Ocient System to immediately skip data segments that are outside the query bounds without any I/O processing.

This syntax example creates a single TimeKey column and must be included in the column definition of a `CREATE TABLE` SQL statement. For the full syntax, see **CREATE TABLE**.

SQL

```
column_name [ data_type ] TIME KEY BUCKET (bucket_granularity, bucket_value [, NOIN
```

Parameter	Type	Description
`column_name`	string	An identifier for the column.

Parameter	Type	Description
`data_type`	string	Optional. The data type of the column. For the TimeKey column, this column should be DATE or TIMESTAMP type. The TimeKey column can also support INT or BIGINT type, but these data types do not use a `bucket_value` argument.
`bucket_granularity`	numeric	The granularity of the TimeKey column, based on the specified TIME type in `bucket_value`.
`bucket_value`	string	The TIME type to parse the TimeKey column. Supported values include: `[ DAY, HOUR, MINUTE, SECOND ]` For example, `BUCKET(1, DAY)` sets the time-bucket granularity to a fixed width of one day.

For DDL examples of tables that use TimeKey columns, see **CREATE TABLE SQL Statement Examples**.

Time Bucket Indexing

For tables with defined TimeKey columns, the system generates a hidden column named `_TIME_BUCKET` and uses it during clustering to group rows by approximate time ranges, which improves time-range filtering performance.

The `_TIME_BUCKET` column is not visible in a standard `SELECT *` query or data export of the table. To view this column, you must explicitly query it by name.

Example

This example query specifies the `_TIME_BUCKET` column along with an explicit value (for the purposes of the example, assume the TimeKey column is a TIMESTAMP column).

SQL

```
SELECT "_TIME_BUCKET", time_col, metric
FROM my_table
WHERE "_TIME_BUCKET" = TIMESTAMP('2024-01-01 00:00:00');
```

TimeKey Column Without the Time Bucket

If you want to define a TimeKey that uses bucket semantics but without a `_TIME_BUCKET` column, you can opt out by using the `NOINDEX` keyword.

Example

This example of a TimeKey column definition includes the `NOINDEX` keyword to explicitly prevent the time bucket from impacting the system clustering behavior. This option is useful if you want bucketed semantics for certain operations with no impact on storage ordering or index generation.

SQL

```
time_col TIMESTAMP TIME KEY BUCKET(1, DAY, NOINDEX)
```

Clustering Key

Every table in an Ocient System can support a Clustering Key to speed up query operations on a series of adjacent columns.

The Clustering Key determines how data is ordered into segments on disk. This ordering acts like an index, enabling an Ocient System to quickly locate any relevant rows of the Clustering Key columns.

Every database table should define one Clustering Key on a sequence of frequently queried columns for best results.

Clustering Key columns can be of any fixed-length data type except for `DECIMAL`. Variable-length columns can also support Clustering Keys if they are configured with global dictionary compression. For details, see **Global Dictionary Compression**.

In addition to the Clustering Key, you can also define clustering indexes. Unlike the Clustering Key, clustering indexes do not involve ordering data into segments. They create separate indexes for each specified column set, which boosts performance but requires some additional storage.

This syntax example creates a Clustering Key across multiple columns and must be included in a `CREATE TABLE` SQL statement. For the full syntax, see **CREATE TABLE**.

SQL

```
CLUSTERING KEY key_name (ck_col1, ck_col2 [, ...])
    [, INDEX index_name (idx_col1, idx_col2 [, ...])
    [, INDEX ... ] ]
```

Parameter	Type	Description
`key_name`	string	An identifier for the Clustering Key.
`ck_col1, ck_col2 [, ...]`	string	A series of specific columns comprising the Clustering Key. Clustering Key columns must not be nullable. Specify `NOT NULL` in the column definition for the respective columns. For details, see **Column Definition**. No limit exists on the number of columns for the Clustering Key.
`index_name`	string	Optional. An identifier for a clustering index. You must include the definition of columns that comprise the index using the `idx_col1, idx_col2 [, ...]` parameter.
`idx_col1, idx_col2 [, ...]`	string	Optional. A series of specific columns comprising a clustering index. You can apply clustering indexes only to columns included in the Clustering Key. Specify any number of columns in any specified order. You must include the identifier for the index using the `index_name` parameter.

Select Clustering Key Columns

You can define a Clustering Key and any additional clustering indexes as arguments of a **CREATE TABLE** SQL statement.

The column ordering of a clustering key is significant. You must specify columns in rank order.

Table Example

This `CREATE TABLE` SQL statement example shows the definition of a clustering key on columns `(CK1, CK2, CK3)`. The data types of these columns are irrelevant to this example.

SQL

```
CREATE TABLE example_table (
    ...,
    CK1 INT,
    CK2 INT,
    CK3 INT,
    ...,
    CLUSTERING KEY ck_example (CK1, CK2, CK3),
);
```

The Clustering Key primes the records for search and filter operations on its columns in descending order of importance. This means the first column `CK1` is the most important of the subset while the third `CK3` is the least important.

To boost performance with a Clustering Key, a query must make references to the columns following this key order.

For a Clustering Key in the `(CK1, CK2, CK3)` pattern, these sequences of query column references are effective:

- `CK1`
- `(CK1,CK2)`
- `(CK1,CK2,CK3)`

Query Example

This example `SELECT` query follows the key order for its filter operations.

SQL

```
SELECT * FROM example_table (
WHERE CK1 > 1
    AND CK2 = 0
    AND CK3 < 0;
```

⚠ For a Clustering Key in the `(CK1, CK2, CK3)` pattern, these sequences of query column references are ineffective:

- `(CK1, CK3)`
- `(CK2, CK3)`
- `(CK3, CK2)`

To improve performance on queries that reference columns in these sequences, you can use clustering indexes. For details, see **Adding Clustering Indexes**.

To check the order of column references in a query, you can test a query using an **EXPLAIN PIPELINE** SQL statement. This statement shows the I/O pipeline and how the system references indexes.

For DDL examples of tables that use Clustering Key columns, see **CREATE TABLE SQL Statement Examples**.

Add Clustering Indexes

A table can only have one Clustering Key, but it can also have any number of clustering indexes to boost performance. Clustering indexes act similarly to the Clustering Key, but they do not order data into segments. Instead, each clustering index creates a separate index for the specified column set, which can impact storage and loading overhead.

Clustering indexes can provide flexibility to your tables, allowing your queries to reference the same columns as your Clustering Key, but in different sequences.

ⓘ You must define clustering keys and clustering indexes when you first create a table. You cannot add them at a later time.

Example

Define a Clustering Key across the first four columns `(CK1, CK2, CK3, CK4)` in the `example_table` table in this `CREATE TABLE` SQL statement.

The statement also includes two clustering indexes using the `INDEX` definition. The first creates an index for a subset of just two columns `(CK2, CK3)` from the Clustering Key. The second index uses all of the key columns `(CK2, CK4, CK1, CK3)`, but in a different sequence.

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE example_table (
    CK1 INT NOT NULL,
    CK2 INT NOT NULL,
    CK3 INT NOT NULL,
    CK4 INT NOT NULL,
    COL5 INT NOT NULL,
    CLUSTERING KEY ck_example (CK1, CK2, CK3, CK4),
    INDEX idx_01 (CK2, CK3),
    INDEX idx_02 (CK2, CK4, CK1, CK3)
);
```

In the example, `COL5` is not part of the Clustering Key, meaning this column cannot be included in any clustering indexes.

Permissions

To create TimeKeys and Clustering Keys, you must have the `CREATE` privilege for the current database.

Observability

You can query these system catalog tables to get information on the effectiveness of your configurations for segment keys and indexes:

- **sys.indexes** — Includes information on the index identifier, type, and other information.
- **sys.index_columns** — Contains information on indexes by their column position.
- **sys.segment_parts** — Contains data on the segment parts associated with each segment key.

For examples of querying segment and index metadata tables, see **Discover Insights From System Catalog Tables**.

Related Links

Ingest Data with Legacy LAT Reference

Query Ocient

CREATE TABLE

Secondary Indexes

Related Videos

At the Whiteboard with Ocient: Indexing at Hyperscale

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Secondary Indexes

Secondary indexes are extremely important for optimizing database queries. When used effectively, indexes allow the database to skip irrelevant records when reading data from storage. On large datasets, this can have a big impact, turning a query that would take minutes to execute into one that finishes in seconds or less.

The tradeoff for using secondary indexes is that they do require additional disk space and loading resources. It is generally a good idea to be strategic about which columns you index. A column is usually a good candidate for an index if you expect it to be frequently referenced in queries, especially for searching and filtering (e.g., in a WHERE or ON clause of a SQL statement).

Creating an index does not trigger the re-indexing of existing segments. Only segments that are generated after the execution of the `CREATE INDEX` statement contain the new index. For that reason, create indexes early on before you load large amounts of data into a table.

You can create indexes on columns containing various data types as long as you meet the requirements in this table. Depending on the data type, the Ocient® System can assign a default index type if you decline to specify one.

Index Type Requirements and Defaults

This table describes index types that are assigned by default to columns based on the data type. Container data types have special requirements for using indexes.

Unless you explicitly specify an index type, the data type of a column determines a default index type for the system to create.

Data type	Requirement	Default Index Type (if not specified)
INT, BIGINT, SMALLINT, FLOAT, DOUBLE, BINARY, DECIMAL, HASH, CHAR, IP / IPV4, TIME, TIMESTAMP, DATE, UUID	None	`INVERTED`
VARCHAR	None	`HASH`

© 2026 Ocient, Inc.

Data type	Requirement	Default Index Type (if not specified)
TUPLE	Inner data types are fixed length or VARCHAR	`HASH`
ARRAY	Inner data types are fixed length or VARCHAR. or Inner datatype is a tuple containing only fixed-length data types	`HASH`
POINT, LINESTRING, POLYGON	None	`SPATIAL`

You can apply indexes regardless of whether they have GDC compression.

`CREATE INDEX` on tuples currently supports indexing only on individual tuple component columns and not entire tuples.

Column Requirements for Index Optimization

All index types can speed up queries that involve filter expressions on their designated columns.

An index takes effect when its specified column is queried with a filter expression of either `\<column_name> <op> <literal>` or `<literal> <op> \<column_name>`.

The column and operator must meet one of these requirements.

Operator	Requirements
`=`, `!=`, `>`, `<`, `<=`, `>=`, or `BETWEEN`.	The column must have a non-container data type.
`IS NULL` or `IS NOT NULL`	Column must be nullable.
`LIKE`, `NOT LIKE`, `=`, `!=`, `SIMILAR TO`, or `NOT SIMILAR TO`	Index type is `NGRAM`.

Array operator requirement****s

Additionally, columns with arrays can use indexes if any of these criteria are met.

Operator or Functio****n	Syntax Example
The operator is *contains* or *not contains, *i.e., `@>` .	`\<filter_expression> = [NOT] (\<column_name> @> \ <scalar_literal> | \<array_literal>)`
The operator is *is contained* or *not is contained by, *i.e., `<@` .	`\<filter_expression> = [NOT] (\<scalar_literal> | \<array_literal> \<@ \<column_name>)`
The operator is *overlap *or *not overlap, *i.e., `&&` .	`\<filter_expression> = [NOT] (\<column_name> && \ <array_literal>) | (\<array_literal> && \ <column_name>)`
The filter expression uses the `for_some` or `for_all` function. The operator is `=` , `!=` , `>` , `<` , `<=` , or `>=` .	`\<filter_expression> = [NOT] (for_all(\ <column_name>) \<op> \<literal>) | (for_some(\ <column_name>) \<op> \<literal>)`

`BETWEEN` is not supported for `for_some` or `for_all` operators with filter expressions. If the index type is `NGRAM` , the operator can also be `LIKE` , `NOT LIKE` , `SIMILAR TO` , or `NOT SIMILAR TO` .

For syntax information for secondary indexes, see the **CREATE INDEX** reference section.

Secondary Index Types

Ocient supports four index types alongside the clustering index: `INVERTED` , `HASH` , `NGRAM` , and `SPATIAL` . An index notionally stores a mapping of the column value to the rows that contain that value, and the index type differentiates the format in which the column values are stored and accessed.

For container data types (e.g., arrays and tuples), the index stores the internal elements of the container and is used on predicates that target the internal values. However, a mapping of NULL column values is generally stored for both scalar and container data types, so the index can always be used for `column IS NULL` predicates.

`INVERTED` Index Type

An `INVERTED` index supports fixed-length column types. The value stored internally in the index is the whole column value, and its storage size is the same as the width of the data type.

This index allows for exact lookups from a column value to rows and supports strict equality comparisons and range comparisons (e.g., `<`, `>`, `>=`, `\<=`, `BETWEEN`). The index supports EQUALS predicates that perform exact matches on array values (e.g., `CONTAINS`, `OVERLAPS`, and `for_all()` and `for_some()` functions that use `EQUALS` or `IS NULL`), and range comparisons on array values using `for_all()` and `for_some()`.

The system does not use the inverted index for all range comparisons on an index column. Instead, the system determines in advance when it is more performant to use the inverted index for range comparisons. For example, ranges that cover a smaller number of values are more likely to utilize the index.

Data Type	Function	Supported Operators	Array Operators and Supported Functions
Fixed-length columns INT, BIGINT, SMALLINT, FLOAT, DOUBLE, BINARY, DECIMAL, HASH, CHAR, IP / IPV4, TIME, TIMESTAMP, DATE, UUID	Supports range and exact matches. The system stores the whole column value (storage is the same as data-type width).	`=`, `!=`, `<`, `>`, `>=`, `<=`, `BETWEEN`	`for_all()`, `for_some()`

`HASH` Index Type

The system supports a `HASH` index on variable-length column types, which stores a hash of the indexed column value rather than the full value. You can use `HASH` indexes for exact comparisons, and they support equality and CONTAINS or OVERLAPS filters. The system supports range operators, such as `<` and `>`, and uses sorting logic to determine how `VARCHAR` types compare to each other.

The internal implementation of a `HASH` index requires additional record filtering, making the `HASH` index more computationally intensive than the `INVERTED` index. The additional filtering is attributable to the `HASH` index being a probabilistic index type; because hash collisions are possible, values returned by the index might include false positives that the database must filter internally during query execution to remove invalid results.

Do not use the `HASH` index for variable-length geospatial types, including `LINESTRING` and `POLYGON`. For better results, use the `SPATIAL` index instead.

© 2026 Ocient, Inc.

In most cases, a `HASH` index on a `VARCHAR` column requires less storage than an `NGRAM` index.

Data Type	Function	Supported Operators	Array Operators and Supported Functions
Variable-length columns VARCHAR, VARBINARY	Supports range and exact matches. The system stores a hash of the indexed column.	`=`, `!=`, `<`, `>`, `>=`, `<=`, `BETWEEN`	`@>`, `<@`, `&&`, `for_all()`, `for_some()`

`NGRAM` Index Type

The Ocient System supports an `NGRAM` index on variable-length column types. An `NGRAM` index stores substrings of size `n` of the column values. For example, using the default `n_value` size of 3, you can divide the column value `Ocient` into the 3-grams { “Oci”, “cie”, “ien”, “ent” }. The system stores these 3-grams separately in the index and can find them individually.

The `NGRAM` index type helps optimize queries that involve either exact or partial matching filters. When the system uses an N-gram index with a filter string, the database divides the filter value into N-grams in the same manner as the column values, and the system finds each filter N-gram in the index.

`NGRAM` index results are probabilistic because the rows matched by the index can include strings that contain the right N-grams but do not satisfy the filter predicate. As a result, the Ocient query engine must filter column values returned from the index lookup for valid matches. The database uses the index to speed up string similarity where possible, but the index does not support all filters. At a minimum, a filter literal must contain at least one sequence of characters of the `n_value` length that must be present in the resulting column value. For instance, `LIKE` and `SIMILAR TO` expressions that are composed entirely of wildcards cannot use the `NGRAM` index.

For an example of partial matching, see this query on a table that has an `NGRAM(3)` index on the `example_column` column.

© 2026 Ocient, Inc.

SQL

```
SELECT example_column FROM example_table WHERE example_column LIKE 'Ocie%';
```

The filter string `'Ocie%'` results in the filter 3-grams { “Oci”, “cie” } . The database finds these 3-grams in the index and returns any rows that contain those substrings, for example, “Ocient”, “The Ocient database”, and “cieOci”. Of these strings, only “Ocient” satisfies the `LIKE` filter. The database filters out false positives and returns the valid rows.

For an exact matching example, see this query on the same table with the `NGRAM(3)` index.

SQL

```
SELECT example_column FROM example_table WHERE example_column = 'Ocie%';
```

In this case, the database uses similar logic by finding the substrings { “Oci”, “cie”, “ien” } in the index. The database returns only strings that exactly match the input string. While `NGRAM` can significantly increase performance on equality matches, it is generally slower than a `HASH` index.

Data Type	Function	Supported Operators	Array Operators and Supported Functions
Variable-length columns VARCHAR, VARBINARY	Supports exact and partial matches. System stores substrings of the `n_value` size of the column values.	`=` , `!=` , `<` , `>` , `>=` , `<=` , `BETWEEN` , `LIKE` , `SIMILAR TO`	`@>` , `<@` , `&&` , `for_all()` , `for_some()`

`SPATIAL` Index Type

The `SPATIAL` index type helps expedite queries on geographic data types, including POINT, LINESTRING, and POLYGON values. In a table with a column of these objects, a SPATIAL index finds rows quickly using a geographic literal filter.

This index groups geographic objects together by using a **packed Hilbert R-tree** algorithm. The index allows queries to quickly search through rows by traversing only a subsection of the full data set.

For array types, the `SPATIAL` index is limited to a maximum of 1 million items per row.

Data Type	Function	Supported Operators	Array Operators and Supported Functions
Geographic columns that contain `POINT`, `LINESTRING`, or `POLYGON` values.	Supports bounding-box matches that can traverse a Hilbert R-tree structure.	Supports **Geospatial Operators** including: `&&`, `@`, `~=`, `~`. Supports Spatial Relationship functions including: **ST_CONTAINS** **ST_CONTAINSPROPERLY** **ST_COVERS** **ST_COVEREDBY** **ST_CROSSES** **ST_DISJOINT** **ST_DWITHIN** **ST_INTERSECTS** **ST_OVERLAPS** **ST_TOUCHES** **ST_WITHIN**	`for_all()`, `for_some()` Array operators can use the supported **Geospatial Operators**, but not the Spatial Relationship functions.

`ZONE_MAP` Index Type

The `ZONE_MAP` index type is a lightweight structure that stores the range of values present in a segment for the indexed column, enabling queries to skip over entire segments with no rows matching a compatible filter.

Unlike other secondary indexes that store values for each row, the system stores `ZONE_MAP` indexes in the metadata of each system segment. The metadata stored for the indexed column includes its minimum and maximum values, as well as any special values (e.g., NULL).

In cases when a compatible filter is outside the bounds of the indexed column, the system can skip processing the segment without performing I/O to fetch raw column data.

To use `ZONE_MAP` indexes most effectively, use them on columns that are also included in a table clustering key because these columns set how the system orders segments on disk. For details, see **Clustering Key**.

Data Type	Function	Supported Operators	Array Operators and Supported Functions
Fixed-length columns			

INT, BIGINT, SMALLINT, FLOAT, DOUBLE, BINARY, DECIMAL, HASH, CHAR, IP / IPV4, TIME, TIMESTAMP, DATE, UUID | Supports range and exact matches on a whole segment level.

The system stores the minimum and maximum values for the column per segment.

| `=`, `<`, `>`, `>=`, `<=`, `BETWEEN`
| None |

Create Secondary Indexes Based on Anticipated Queries

These examples demonstrate how to set up indexes based on the expected queries that are frequently executed on the table.

The examples each involve designing indexes to expedite queries on a set of telecommunications data. These queries include:

- Exact matches for approximate location data (`ipaddress`).
- Range filtering for data usage (`totalpackets`).
- Partial matching for requests to specific URLs (`httprequesturl`)
- Bounding box matching for exact location data (`location`)

These examples use a shared table of internet data records, `ip_data`. The columns and their data types in this table are shown here.

SQL

```
starttime (TIMESTAMP)
endtime (TIMESTAMP)
totalpackets (BIGINT)
httphost (CHAR) nullable
httprequesturl (CHAR) nullable
httpstatuscode (SMALLINT) nullable
httpuseragent (CHAR) nullable
destinationport (INT)
sourceport (INT)
ipaddress (IPV4)
ipprotocol (BYTE)
location (POINT)
```

Optimize an Exact Match Query Using an `INVERTED` Index

This query uses the `ip_data` table to analyze the location of a customer within an approximate time range.

For this use case, a secondary index helps optimize this query template.

SQL

```
SELECT starttime,
       CHAR(ipaddress),
       sourceport,
       destinationport,
       totalpackets
FROM   ip_data
WHERE  starttime BETWEEN '<start_time>' AND '<end_time>'
  AND  ipaddress = '<ip_address>'
LIMIT  10;
```

The crucial column is the `ipaddress`, as it is the crux for the exact matching used in this query.

If data storage is not a problem, this query is ideal for an `INVERTED` index. This index supports exact matching for the IPV4 data type of the `ipaddress` column.

Create an inverted index on this column using this SQL statement.

© 2026 Ocient, Inc.

SQL

```
CREATE INDEX destination_ipaddress ON ip_data (ipaddress) USING INVERTED;
```

Optimize a Range Query Using an `INVERTED` Index

Similar to the exact match query, this query also includes a range filter for the numeric value in the `totalpackets` column.

SQL

```
SELECT starttime,
       CHAR(destinationipaddress),
       sourceport,
       destinationport,
       totalpackets
FROM   ip_data
WHERE  flowstarttime between '<start_time>' AND '<end_time>'
  AND  ipaddress = '<ip_address>'
  AND  totalpackets between 200 and 400
LIMIT 10;
```

An `INVERTED` index is the best choice for this use case. This index supports range filtering for all fixed-length types, including the BIGINT values in the `totalpackets` column.

Create an `INVERTED` index on this column using this SQL statement.

SQL

```
CREATE INDEX totalpackets ON ip_data (totalpackets) USING INVERTED;
```

Optimize an Exact String Match Using a `HASH` Index

In this case, filter all records where customers accessed the exact web address `ocient.com`.

SQL

```
SELECT starttime,
       CHAR(sourceipaddress),
       CHAR(ipaddress),
       sourceport,
       destinationport,
       httprequesturl
FROM   ip_data
WHERE  starttime between '<start_time>' AND '<end_time>'
  AND  httprequesturl = 'ocient.com'
LIMIT 10;
```

If you expect to filter the `httprequesturl` column only with exact matches, the `HASH` index is usually your best choice. The `HASH` index can greatly speed up exact matching on variable-length data types, and it requires less storage than an `NGRAM` index except in rare edge cases.

SQL

```
CREATE INDEX httprequesturl ON ip_data (httprequesturl) USING HASH;
```

Optimize a Partial String Match Using `NGRAM` Index

For this use case, filter all records where customers accessed a partial or exact web address. The `LIKE` clause in this query template filters all records for those customers who accessed the Facebook URL.

SQL

```
SELECT starttime,
       CHAR(sourceipaddress),
       CHAR(ipaddress),
       sourceport,
       destinationport,
       httprequesturl
FROM   ip_data
WHERE  starttime between '<start_time>' AND '<end_time>'
  AND  httprequesturl LIKE '%facebook.com%'
LIMIT 10;
```

© 2026 Ocient, Inc.

An `NGRAM` index is the best choice for this use case. This index supports partial matching for variable-length data types, including the VARCHAR data type of the `httprequesturl` column.

Create an `NGRAM` index on this column using this SQL statement.

SQL

```sql
CREATE INDEX httprequesturl ON ip_data (httprequesturl) USING NGRAM(3);
```

In this example, the `NGRAM` index divides the specified column into substrings of three characters, which is the default. This means partial strings such as “Fac”, “ace”, “boo” and other variants are captured in the filter for this query.

You can specify a broader index by setting a higher substring value, such as `NGRAM(4)` or `NGRAM(5)`. Using one of these values would reduce the data storage requirements.

Optimize a Match Query Using a `SPATIAL` Index

For this use case, filter all records where customers are located in a specific area at a specific time.

This query represents the target area by the bounding box `POLYGON((1 2,1 3,1 4,1 2))`.

SQL

```sql
SELECT starttime,
       CHAR(sourceipaddress),
       CHAR(ipaddress),
       sourceport,
       destinationport,
       httprequesturl,
       location
FROM   ip_data
WHERE  starttime between '<start_time>' AND '<end_time>'
  AND  ST_CONTAINS(location, POLYGON((1 2,1 3,1 4,1 2)))
LIMIT 10;
```

A `SPATIAL` index is the best choice for this use case because it supports a variety of functions for filtering geospatial data for bounding box areas.

Create a `SPATIAL` index on the `location` column by using this SQL statement.

SQL

```
CREATE INDEX location ON ip_data (location) USING SPATIAL;
```

Optimize a Range Query Using a `ZONE_MAP` Index

This query analyzes network traffic originating from a specific range of IP addresses.

SQL

```
SELECT starttime,
       CHAR(ipaddress),
       sourceport,
       destinationport,
       totalpackets
FROM   ip_data
WHERE  ipaddress BETWEEN '192.168.0.0' AND '192.168.255.255'
LIMIT  10;
```

If you load the data so the system clusters it by IP address (e.g., sorted by `ipaddress`), a `ZONE_MAP` index is highly efficient for this query. This index allows the system to quickly identify and scan only the segments that contain IP addresses within the requested range, skipping all others.

Create a `ZONE_MAP` index on this column using this SQL statement.

SQL

```
CREATE INDEX starttime_idx ON ip_data (ipaddress) USING ZONE_MAP;
```

Related Links

Load Data

Query Ocient

CREATE INDEX

DROP INDEX

Related Videos

At the Whiteboard with Ocient: Indexing at Hyperscale

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Table Compression Options

The Ocient® System provides a range of options for compressing tables to reduce storage requirements for your data. Compression can be a major component of query optimization when used effectively by reducing I/O requirements and potentially allowing more system space for indexes.

Ocient Compression Key Concepts

Table columns in Ocient can support these compression options:

- **Global Dictionary Compression (GDC)**
- **Zstandard Compression (ZSTD)**
- **Dynamic Compression**

Ocient System compression adheres to these key concepts:

- By default, the Ocient dynamic compression scheme can determine whether to apply compression on each column if it is statistically advantageous. Sometimes, you might want to use another compression scheme like ZSTD or completely disable compression.
- You can specify custom compression schemes on individual columns through data definition language (DDL) commands, either using a `CREATE TABLE` or `ALTER TABLE` SQL statement.
- GDC can stack onto other compression schemes.

General Compression Matrix

This table shows the best compression scheme for different column data types. The suggested best compression schemes consider the column data type and cardinality, but other factors can influence which data compression makes sense.

The default compression in the table shows the compression scheme that the database applies if you do not specify an option. This default compression balances performance and compression.

© 2026 Ocient, Inc.

Data Type	Default Compression	Best Compression
Fixed-length	`COMPRESSION DYNAMIC`	`COMPRESSION ZSTD`
Variable-length and low cardinality	`COMPRESSION NONE`	`COMPRESSION GDC` and `COMPRESSION ZSTD`
Variable-length and high cardinality	`COMPRESSION NONE`	`COMPRESSION ZSTD`
Arrays	`COMPRESSION NONE`	`COMPRESSION ZSTD`
Arrays of variable-length types and low cardinality	`COMPRESSION NONE`	`COMPRESSION GDC` and `COMPRESSION ZSTD`
Tuples	Compression depends on each tuple index data type.	The database treats each tuple index position as its own column value and can support its own compression scheme. Fixed-length values default to `COMPRESSION DYNAMIC`, while variable-length values default to `COMPRESSION NONE`. For best performance, apply rules for each tuple index based on the data-type rules outlined in this table. For example: `my_tuple TUPLE<<INT, VARCHAR(255) COMPRESSION GDC(1) COMPRESSION ZSTD, VARCHAR(255)>>` This example defines a tuple column with GDC and ZSTD compression only for the first inner `VARCHAR` value.

For details about the advantages and disadvantages of each compression scheme, see the `compression_scheme` description in **Column Constraint**.

For details about the compression syntax, see **CREATE TABLE**.

Global Dictionary Compression

Column parameter: `COMPRESSION GDC`

GDC uses a dictionary encoder to map variable-length column data. Instead of storing the variable length data directly on disk, GDC substitutes each unique string with a corresponding unique integer.

This compression scheme can significantly reduce overall storage requirements for low-cardinality columns with a few distinct values while boosting performance for various

operations.

GDC supports an optional `compression_value` argument that specifies how many unique values to use.

Advantages

- Reduced disk usage — Variable-length data, such as strings or container types, can experience massive storage reduction.
- Improved query performance — By mapping each row to an integer, variable-length rows no longer require a full traversal for selective I/O operations, which improves performance in some queries.
- Faster Joins — Join comparisons have improved performance from using GDC integers instead of the variable-length data.
- Variable-Length Cluster Keys — Variable-length columns can use a clustering key index for improved performance.
- Stackable — Columns can use GDC in tandem with other compression schemes.

Disadvantages

- Increased complexity — You must know the rough cardinality of your data to size the GDC configuration properly.
- Load complexity — GDC compression adds some overhead to loading data into the Ocient System.
- Not suitable for high-cardinality data — GDC performance decreases as the data cardinality increases.
- Cannot apply retroactively — Unlike other compression options, you must set GDC as part of the initial `CREATE TABLE` SQL statement. You cannot apply it later to the table.

For details, see **Global Dictionary Compression**.

SQL

```
column_name [data_type] COMPRESSION GDC [ (compression_value) ]
```

For details about adding GDC compression to a table schema, see **CREATE TABLE**.

Zstandard Compression

Column parameter: `COMPRESSION ZSTD`

ZSTD uses the ZSTD library to apply a fast and lossless compression algorithm to any column type, including fixed and variable-length scalars, arrays, and tuples on a per-component basis. In addition, ZSTD works on columns where the database first compresses variable-length data by GDC and then compresses the resulting integral type on disk.

This compression scheme can affect performance by requiring additional processing. Therefore, this compression scheme is usually good for less-frequently accessed columns or columns paired with an index.

ZSTD supports these optional values:

- `COMPRESSION_LEVEL` — This value signifies how much compression the data receives. The default value is `0`. The full range of values is from `-7` (low compression) to `15` (high compression). The database requires less memory when this value is lower, whereas it requires more memory and processing resources when the value is larger.
- `DICTIONARY_SIZE` — This value is the dictionary size that you specify as a positive integer. The value signifies the size of the shared compression dictionary in bytes. The default value is `32768` (32 KB). The full range of values ranges from 4096 (4 KB) to `1048576` (1 MB). This value denotes the amount of memory consumed during segment generation. In general, larger values provide better compression but use more memory.

> The default values for `COMPRESSION_LEVEL` and `DICTIONARY_SIZE` are sufficient for most cases.

Advantages

- Reduced disk storage — Dramatically decreases disk space usage.
- Flexible — You can apply this compression scheme after table creation using an **ALTER TABLE** SQL statement.

Disadvantages

- Performance loss — ZSTD does require some extra CPU resources, which means that it can impact query performance.
- Memory usage — ZSTD can demand significant memory usage when many columns use it for parallel query workloads.

```sql
column_name [data_type] COMPRESSION ZSTD
   [COMPRESSION_LEVEL = level_value, DICTIONARY_SIZE = size_value]
```

For details about adding ZSTD compression to a table schema, see **CREATE TABLE**.

Dynamic Compression

Column parameter: `COMPRESSION DYNAMIC`

By default, Ocient tables use dynamic compression to reduce storage space requirements automatically. This method employs a variety of compression schemes on columns based on their data type and a statistical determination of what kinds of compression would be advantageous.

- For fixed-length data, block-size units (typically 4 KB) can receive various compression schemes, including run-length encoding and delta-delta encoding.
- For variable-length data (VARCHAR and VARBINARY), the database compresses data row-by-row, with values individually compressed using LZ4 compression. This method produces a good compression ratio for large values. However, small or medium-sized values have negligible compression rates.

Advantages

- Ease of use — The Ocient System determines the compression.
- Flexibility — You can apply this compression scheme after table creation using an **ALTER TABLE** SQL statement.

Disadvantages

- Secondary option — Dynamic compression is designed for balance, but it is generally not as effective at compression as ZSTD.

- Performance loss — This compression scheme has a small performance impact, but is not as costly as ZSTD.

```SQL
column_name [data_type] COMPRESSION DYNAMIC
```

For details about adding dynamic compression to a table definition, see **CREATE TABLE**.

For examples of tables that use compression, see **CREATE TABLE SQL Statement Examples**.

Related Links

Global Dictionary Compression

Secondary Indexes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Global Dictionary Compression

Global Dictionary Compression (GDC) is a feature of the database that compresses variable length column data using a dictionary encoder. Instead of storing the variable length data directly on disk, GDC seamlessly substitutes an integer that corresponds to each unique string.

Advantages of GDC

1. **Reduced disk usage** — A string like `"GDC is really cool!"` takes up 19 raw bytes. With GDC, it takes at most 4 bytes. This benefit is multiplied by the number of rows in the table and, when applied to arrays, can lead to a dramatic reduction in required storage.
2. **Fixed-width** — Variable length column data can be more difficult to perform selective input or output (I/O). It is much faster for Ocient® to read the 100th element in a list of integers by offsetting 100 times the fixed width of the integer. With variable-length types, a full traversal is required, counting how many elements are passed until the 100th is found. This can improve performance in some queries.
3. **Faster Joins** — When joining on a GDC column, equality comparisons can be performed on the GDC integers instead of the variable-length data. This can improve performance in some queries.
4. **Allows Variable-Length Cluster Keys** — As of version 19.0 of the Ocient System, GDC is the only method by which a variable-length column can be used as a cluster key index.

Disadvantages of GDC

1. **Increased complexity** — GDC does add some configuration complexity. It requires a user to know the rough cardinality of their data to size the GDC number of bytes and adds system configuration options.
2. **Load complexity** — GDC compression adds some overhead to loading data into the Ocient System.
3. **Not suitable for high-cardinality data** — If the variable length column has higher cardinality than this, it should not be stored using GDC.

© 2026 Ocient, Inc.

GDC Syntax

To create a GDC column, add the `COMPRESSION GDC(int)` parameter after the column type specifier when creating a column, where `int` is one of 1, 2, or 4. The integer specifies the number of bytes to be used for the integer keys stored on the disk. This also corresponds to the maximum number of unique keys to be stored in a particular column.

GDC(int)	Maximum Number of Unique Values
GDC(1)	256
GDC(2)	65,536
GDC(4)	4,294,967,296

> There is a *soft* limit of 1,000,000 keys even on 4-byte integer GDC columns. Contact Ocient Support to evaluate your criteria for changing the limit and to understand the impact of raising limits.

Apply GDC to a Column

To enable GDC on a column, the keyword is applied in the `CREATE TABLE` or `ALTER TABLE ADD COLUMN` statement. An example of GDC on a `VARCHAR` column:

```
{column_name} VARCHAR(255) COMPRESSION GDC(2)
```

When you apply GDC to one column, the Ocient System creates a view instead of a table from the full column definition.

Reuse an Existing GDC Map

A user can also specify that a column should share GDC space with another column–possibly even a column of a different table. This would be useful if the same data is used in multiple columns and it is often joined in queries. Use `COMPRESSION GDC EXISTING schema.table.column_name` as shown here:

```
{column_name} VARCHAR(255) GDC EXISTING {schema.table.column_name}
```

GDC on Array Columns

For arrays of variable-length data, GDC operates on the individual elements of the array. Specify compression after the overall array type:

```
{column_name} VARCHAR(255)[] COMPRESSION GDC(2)
```

GDC on Tuple Columns

Elements of tuple columns can be compressed with GDC. Specify compression on the specific type to be compressed:

```
{column_name} tuple<int, VARCHAR(255) COMPRESSION GDC(2)>
```

GDC in the System Catalog

You can use the system catalog to inspect the number of keys used by different columns and the maximum count on each.

Example Query:

SQL

```sql
SELECT
  t.id as table_id,
  t.name as table_name,
  c.id as column_id,
  c.name as column_name,
  g.compressed_size,
  g.current_count,
  g.max_count
FROM sys.tables t
INNER JOIN sys.global_map_table_info g
  ON t.id = g.table_id
INNER JOIN sys.columns c
  ON c.id = g.column_id;
```

Truncation

When a table with GDC is partially truncated, *GDC key mappings for removed rows are not removed*. This can result in stale mappings. The only way to remove unwanted

mappings is to drop the column and recreate it.

GDC Column Representation in the System Catalog

To the end user, a table with GDC columns looks like any other table. However, when creating a table with GDC columns, the table configuration is different in system catalog tables. Instead of a table named `schema.tablename`, GDC tables leverage a built-in view.

This view and several pieces of metadata will be created in the system catalog:

1. A view named `schema.tablename` is added in `sys.views`. This view is the representation of the user of the GDC table, and it automatically converts the GDC keys to the loaded variable-length data. This is the table that the user interacts with for querying data, making alterations, and granting or revoking access.
2. For each GDC column in the table, a table named `syslookup.schema_tablename_columnname` is added in `sys.tables`. These tables store the mappings from strings to integers for each column. These tables should be interacted with only rarely.
3. A table called `sysgdc.schema_tablename` is added in `sys.tables`. This is the table that is stored to disk, including all the non-GDC columns and the integers for each GDC column. This table should be interacted with only rarely.

GDC Columns and Loading into the Database

GDC stores key mappings in a Raft consensus log that is maintained by all nodes operating with the Admin role. These mappings are created as the data is loaded, and used in queries to fetch the correct variable length data for the stored integers. Mappings can also be created by certain queries if the key does not already exist.

Because the Admin roles are responsible for maintaining the GDC key mappings, there must be a consensus of Admin roles available on the system in order to load new data that includes GDC columns.

Related Links

System Catalog

© 2026 Ocient, Inc.

© 2026 Ocient All Rights Reserved

Table Retention Policies

The Ocient® System supports table retention policies, an automated way to manage and delete data after a specific time. These policies help maintain storage efficiency on a schedule.

Using DDL SQL statements, you can modify individual tables to specify their own retention policies for how long the system should keep data before automatic removal.

Requirements

To add table retention policies, you must have `ALTER` and `DELETE` privileges for the system and the specific table.

In addition, a database table must meet these conditions to support retention policies:

- Have a designated TimeKey® column (For details, see **TimeKeys and Clustering Keys**)
- Be a user-created table or system-persistent table. You cannot edit virtual system tables.

Table Retention Properties

Table retention properties define the specific parameters that control how long data remains in a table before being automatically deleted. These properties determine both the duration of retention and the granularity at which the system evaluates data for removal, ensuring that retention policies operate consistently and predictably. By configuring these properties, you can fine-tune how the system balances data availability with storage efficiency.

Table Retention and TimeKey

Retention policies of a table are directly linked to the column defined as the TimeKey of the table. For each row in the table, the TimeKey column represents its age, which the system evaluates to determine whether to retain or delete the row. Table retention

© 2026 Ocient, Inc.

policies cannot function properly if the TimeKey column does not represent the age of each row.

Additionally, the TimeKey bucket value also represents the frequency of evaluating the age of each row in the table. For example, a table with a TimeKey bucket of `1 HOUR` and a retention period of `1 DAY` would evaluate each row every hour to determine if it is more than one day old.

Based on the TimeKey bucket frequency for culling older records, the system can potentially retain a row for slightly longer than the set period of a retention policy, up to a maximum retention time of the sum of the retention period and the TimeKey bucket.

> The table retention policy also deletes older rows of any dependent global dictionary compression tables defined on columns of the original table.

Table Retention Tasks

The system deletes older records as a distributed task, which is visible in the **sys.tasks** and **sys.subtasks** catalog tables. Additionally, each table retention policy creates a scheduled process, which is visible in the **sys.tkt_table_info** system catalog table. For details, see **Distributed Tasks**.

Retention and Workload Management

Retention policies execute periodically to delete data in two separate phases: a truncate operation and a delete operation.

The truncate operation interfaces directly with the storage layer of the system and is not affected by any workload management configuration.

In contrast, the delete operation runs as a `DELETE` SQL statement and uses a designated `Retention Policy` service class for retention policies. This service class is a built-in class that administrative users can edit. All table retention policies use the same service class.

Default Retention Policies

A subset of persistent system tables has table retention policies set by default. All retention policies in the system, including those set by default, are visible in the **System**

Catalog.

You can also view a retention policy for an individual table by executing an **EXPORT TABLE** statement.

Add Retention Policies to Tables

You can add table retention policies using a `CREATE TABLE` or `ALTER TABLE` SQL statement.

Table retention policies are optional for all user-defined tables. If a table lacks a retention policy, the default behavior is to retain data in perpetuity.

Retention policies follow this syntax as part of either a **CREATE TABLE option** or an **ALTER TABLE** statement.

```
SQL

RETENTION POLICY AGE retention_granularity retention_value
```

Parameter	Type	Description
`retention_granularity`	numeric	A number that represents the amount of time before the system deletes data in the table. You must pair this number with a specific time type, represented by the `retention_value`.
`retention_value`	string	The unit of time that represents how long the table retains data. Supported values are: `[ WEEK, DAY, HOUR, MINUTE, SECOND, MILLISECOND ]` For example, `1 DAY` schedules the table retention to a granularity of one day.

Example

This example creates a retention policy that keeps rows only if they are less than one day old, as specified by the last line, `RETENTION POLICY AGE 1 DAY`. The retention policy depends on the `created_at` column because it is the TimeKey column. The TimeKey bucket value `(1, HOUR)` determines how frequently the system checks for any rows to remove.

SQL

```sql
CREATE TABLE IF NOT EXISTS "transact_data" (
    "created_at" TIMESTAMP TIME KEY BUCKET(1, HOUR) NOT NULL,
    "user_id" INT NOT NULL DEFAULT 0,
    "sell_id" BIGINT NOT NULL,
    "purchase_amount" BIGINT NOT NULL DEFAULT 0,
    "buyer_name" VARCHAR(1048576),
    CLUSTERING KEY "primary_index" ("sell_id", "user_id")
) RETENTION POLICY AGE 1 DAY;
```

Related Links

Tables

CREATE TABLE SQL Statement Examples

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

CREATE TABLE SQL Statement Examples

Tables in the OcientAIQ™ Unified Data Platform contain various configuration options that can impact performance, reduce storage requirements, or provide extra data safeguards.

As some table configuration options cannot be added after data is already loaded, it is important that you take the time to design `CREATE TABLE` SQL statements with forethought for the queries that you expect to execute frequently. This topic provides examples of `CREATE TABLE` SQL statements designed for different use cases and describes the configuration choices.

For syntax and parameter information, see **CREATE TABLE** .

Create a Table with All Data Types

This example creates a table with all supported data types in Ocient®. The intent of this example is to show the required formatting for different data type values, which is shown as the `DEFAULT` value for each column.

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE data_type_example (
    col_bigint BIGINT NOT NULL DEFAULT 9876543210,
    col_binary BINARY(3) NOT NULL DEFAULT '0xabcdef',
    col_boolean BOOL NOT NULL DEFAULT TRUE,
    col_char CHAR(4) NOT NULL DEFAULT 'val',
    col_date DATE NOT NULL DEFAULT '2000-01-01',
    col_decimal DECIMAL(18,4) NOT NULL DEFAULT 123.45,
    col_double DOUBLE NOT NULL DEFAULT 3.141592,
    col_float FLOAT NOT NULL DEFAULT 2.718,
    col_int INT NOT NULL DEFAULT 123456789,
    col_int_array INT[] NOT NULL DEFAULT 'INT[0,1,2,3]',
    col_ipv4 IPV4 NOT NULL DEFAULT '127.0.0.1',
    col_ip IP NOT NULL DEFAULT '0123:4567:89ab:cdef:0123:4567:89ab:cdef',
    col_smallint SMALLINT NOT NULL DEFAULT 32767,
    col_matrix MATRIX[2][3] NOT NULL DEFAULT '{ {0, 0, 0}, {0, 0, 0} }',
    col_point POINT NOT NULL DEFAULT 'POINT(0 0)',
    col_linestring LINESTRING NOT NULL DEFAULT 'LINESTRING(0 0, 1 1)',
    col_polygon POLYGON NOT NULL DEFAULT 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))',
    col_time TIME NOT NULL DEFAULT '12:34:56.012345678',
    col_timestamp TIMESTAMP NOT NULL DEFAULT '2000-01-02 12:34:45',
    col_tinyint TINYINT NOT NULL DEFAULT 127,
    col_tuple TUPLE<<INT, INT>> NOT NULL DEFAULT 'tuple<<INT,INT>>(0,0)',
    col_array_of_tuples TUPLE<<INT, INT>>[] NOT NULL DEFAULT 'TUPLE<<INT,INT>>[tupl
    col_uuid UUID NOT NULL DEFAULT '01234567-89ab-cdef-1357-0123456789ab',
    col_varbinary VARBINARY NOT NULL DEFAULT '0xaabbccddeeff',
    col_varchar VARCHAR NOT NULL DEFAULT 'This is a variable length string'
);
```

Create a Table for Real-Time Analysis

This example assumes you plan to ingest large quantities of time series data for rapid querying. The most important columns that you plan to reference in queries frequently include:

- `created_at` — A granular timestamp column that represents the primary time reference in the table.
- `user_id` — An integer column to identify each unique user.
- `sell_id` — An integer column representing each transaction by a user.

SQL

```
CREATE TABLE IF NOT EXISTS "transact_data" (
    "created_at" TIMESTAMP TIME KEY BUCKET(1, HOUR) NOT NULL,
    "user_id" BIGINT NOT NULL DEFAULT 0,
    "sell_id" BIGINT NOT NULL,
    "purchase_amount" DOUBLE NOT NULL DEFAULT 0,
    "buyer_name" VARCHAR,
    CLUSTERING KEY "ck" ("sell_id", "user_id")
);

CREATE INDEX "purchase_idx" ON "transact_data" ("purchase_amount");
CREATE INDEX "buyer_idx" ON "transact_data" ("buyer_name")
    USING NGRAM(4);
```

This example table makes use of these Ocient System configurations.

Column Name	Configuration Description
`created_at`	This column is the table TimeKey®, which means that the database indexes for filter optimization. The example configures the TimeKey to partition segments by every hour, `(1, HOUR)`, which is the approximate time granularity you expect to use in your query filters.
`user_id`, `sell_id`	The example table identifies these two columns as the table Clustering Key. The database pairs these two columns for quick reference, such as filtering one column for row values of the other.
`purchase_amount`	The `CREATE INDEX` SQL statement adds a secondary index to this column. As a `BIGINT` data type, the `purchase_amount` column defaults to the `INVERTED` index.
`buyer_name`	This `CREATE INDEX` SQL statement adds a secondary index to this column. As a `VARCHAR` column, the `buyer_name` column normally defaults to a `HASH` index, but this example specifies it should be an `NGRAM` index of width `4` instead.

For details about using the TimeKey and Clustering Key functionality, see **TimeKeys and Clustering Keys**.

For details about other index types, see **Secondary Indexes**.

Create a Table for Geospatial Data

This example assumes you plan to ingest geospatial data linked to time series data using OcientGeo™. The most important columns that you plan to reference frequently in queries include:

- `route_time` — A granular timestamp column that represents the primary time reference in the table.
- `start_point`, `end_point` — The geospatial point values for the start and destination locations.
- `trip_id` — An integer column to identify each unique trip.
- `vehicle_id` — An integer column that represents each vehicle.

SQL

```
CREATE TABLE "trip_details" (
    "route_time" TIMESTAMP TIME KEY BUCKET(14, DAY) NOT NULL,
    "start_point" POINT NULL,
    "end_point" POINT NOT NULL DEFAULT 'POINT(0 0)',
    "route_path" LINESTRING NOT NULL
        DEFAULT 'LINESTRING(0 0, 1 1)',
    "route_area" POLYGON NOT NULL
        DEFAULT 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))',
    "trip_id" BIGINT NOT NULL,
    "vehicle_id" BIGINT NOT NULL,
    CLUSTERING KEY "ck" ("trip_id", "vehicle_id")
);

CREATE INDEX "start_idx" on "trip_details" ("start_point")
    USING SPATIAL;
CREATE INDEX "end_idx" on "trip_details" ("end_point")
    USING SPATIAL;
```

This example table makes use of these Ocient System configurations.

Column Name	Configuration Description
`route_time`	This column is the table TimeKey, which means that the database indexes for filter optimization. The example configures the TimeKey to partition segments by every two

Column Name	Configuration Description
	weeks `(14, DAY)`, which is the approximate time granularity that you expect to use in your query filters.
`trip_id`, `vehicle_id`	The example table identifies these two columns as the table Clustering Key. The database pairs these two columns for quick reference, such as filtering one column for row values of the other.
`end_point`, `route_path`, `route_area`	These columns have `POINT`, `LINESTRING`, and `POLYGON` data types, respectively. Each of these geospatial columns includes default values in its respective format.
`start_point`, `end_point`	The `CREATE INDEX` SQL statements for both of these columns assign `SPATIAL` indexes for quicker reference. This index is the default type for any column with a geospatial data type.

For details about OcientGeo functionality, see **Geospatial Functions**.

Create a Table With Compression Options

This example table includes multiple compression options to ensure minimal storage for specific columns and extra performance.

> Most data types, except arrays, have `COMPRESSION DYNAMIC` applied by default.

SQL

```
CREATE TABLE "call_records" (
    "utc_timestamp" TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
    "sectorid" VARCHAR
        COMPRESSION ZSTD,
    "address" VARCHAR
        COMPRESSION NONE,
    "call_type" VARCHAR
        COMPRESSION ZSTD
        COMPRESSION GDC(4),
    "access_information" VARCHAR(500)
        COMPRESSION GDC(4),
    "file_name" VARCHAR
        COMPRESSION ZSTD compression_level = 3,
        dictionary_size = 1048576,
    CLUSTERING KEY "ck" ("call_type", "access_information")
);
```

This example table makes use of these Ocient System configurations.

Column name	Configuration Description
`address`	This column explicitly removes compression by specifying `COMPRESSION NONE` in the `CREATE TABLE` SQL statement.
`call_type`, `access_information`	These columns have `VARCHAR` data types configured with Global Dictionary Compression (GDC). This compression scheme maps the variable-length type rows to integers. Among other benefits, this compression allows both columns to support a Clustering Key for faster querying. The `call_type` column stacks multiple compression schemes by including both GDC and ZSTD compression to provide additional storage reduction.
`file_name`	This column uses ZSTD compression with extra options. The `compression_level` is set for `3`, meaning the column data is slightly more compressed than the default `0`. The dictionary_size is set for the maximum value, `1048576`, which means greater compression but higher memory demand.

For details about Ocient compression schemes, see **Table Compression Options**.

Create a Table for Various Container Types

This example table includes complex container data types, including a two-dimensional array, a tuple with variable-length values, and a tuple with various data types.

SQL

```
CREATE TABLE "container_data" (
    "utc_timestamp" TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
    "user_id" BIGINT NOT NULL DEFAULT 0,
    "sell_id" BIGINT NOT NULL DEFAULT 0,
    "point_array_col" POINT[],
    "array_col" VARCHAR[] COMPRESSION GDC(4),
    "2D_array_col" INT[][],
    "tuple_col1" TUPLE<<
        VARCHAR(255),
        VARCHAR(500)
        >>,
    "tuple_col2" TUPLE<<
        INT,
        INT[],
        VARCHAR COMPRESSION GDC(1) COMPRESSION ZSTD,
        VARCHAR
        >>,
    "matrix_col" MATRIX[5][10],
    CLUSTERING INDEX "ck" ("user_id", "sell_id")
);

CREATE INDEX "point_idx" on "container_data" ("point_array_col")
    USING SPATIAL;
CREATE INDEX "tuple_idx" on "container_data" ("tuple_col1"[1])
    USING HASH;
CREATE INDEX "tuple_idx2" on "container_data" ("tuple_col2"[2])
    USING INVERTED;
```

This example table makes use of these Ocient System configurations.

Column name	Configuration Description
`point_array_col`	This column is an array of `POINT` values that is indexed using a `SPATIAL` index type as specified by the `CREATE INDEX` SQL statement.

Column name	Configuration Description
`tuple_col1`	This tuple column contains two variable-length values. Each object in the tuple can support its own compression scheme and index. The `CREATE INDEX` SQL statement assigns a `HASH` index to the first value (`VARCHAR(255)`).
`tuple_col2`	This tuple column contains various data types. Each object in the tuple can support its own compression scheme and index. The two `VARCHAR` objects in the tuple column each have their own unique compression scheme. The `CREATE INDEX` SQL statement assigns an `INVERTED` index to the second tuple object, the integer array `INT[]` .

For details about Ocient container types, see **Array, Tuple, and Matrix Overview**.

Create a Table With a Retention Policy

This example creates a retention policy that keeps rows only if they are less than one day old, as specified by the last line, `RETENTION POLICY AGE 1 DAY`. The retention policy depends on the `created_at` column because it is the TimeKey column. The TimeKey bucket value `(1, HOUR)` determines how frequently the system checks for any rows to remove.

SQL

```
CREATE TABLE IF NOT EXISTS "transact_data" (
    "created_at" TIMESTAMP TIME KEY BUCKET(1, HOUR) NOT NULL,
    "user_id" INT NOT NULL DEFAULT 0,
    "sell_id" BIGINT NOT NULL,
    "purchase_amount" BIGINT NOT NULL DEFAULT 0,
    "buyer_name" VARCHAR(1048576),
    CLUSTERING KEY "primary_index" ("sell_id", "user_id")
) RETENTION POLICY AGE 1 DAY;
```

For details about configuring table retention, see **Table Retention Policies**.

Related Links

Data Types

Data Definition Language (DDL) Statement Reference

Table Compression Options

Secondary Indexes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Manage Distributed Tasks

Distributed tasks are processes that run in the background of the Ocient® System to keep the system running smoothly and efficiently. These tasks provide persistent storage of task information in the global metadata storage cluster of the Ocient System, allowing tasks to be tracked and retried until completion. For details about storage and related concepts, see **Core Elements of an Ocient System**.

The distributed task infrastructure is completely generic. The system instantiates tasks from factories using generic arguments and establishes communication using interfaces to abstract the business logic in whatever specific protocol is doing the actual work. Tasks must remain unchanged when duplicated and can be restarted on a new node by the infrastructure at any time. Tasks can also be structured recursively, with a parent task comprising several subtasks, which can also have subtasks, and so on.

Distributed tasks provide persistence across outages. To do this, the task infrastructure detects when the node running a task, or the task owner, has become unavailable and moves the task elsewhere. The administrator owner manages node unavailability. This owner is a node in the administrator cluster responsible for monitoring the status of a task and reassigning the task owner when needed. This pairing of administrator and task owners provides load balancing across the administration cluster by removing the need for every task owner to communicate with the administrator owner.

Task Types

The Ocient System has different task types, such as:

- Data pipeline loading
- Rebuild segments
- Check for disk corruption by verifying segments
- Rebalance data across storage clusters and nodes

For data pipeline loading, see **Load Data**.

© 2026 Ocient, Inc.

For details about rebuilding segments, checking the disk, and rebalancing data, see **Distributed Tasks**.

Task Characteristics

The characteristics of a task are a factory, serialized arguments, a location type, and a location identifier:

- Factory — A task class implementation that the Ocient System instantiates generically and executes the actual work of the task.
- Arguments — The inputs to a task the Ocient System stores generically as a string. These inputs can contain anything the task needs, e.g., flags modifying execution modes, identifiers on which to run, etc.
- Location type — The nature of the set of nodes where the task can run. This location can be any node in the system, any node in a specific cluster, or a specific node.
- Location identifier — The specific location entity when the task location type is a cluster or node.

Tasks have optional pre- and post-condition checks and must implement an asynchronous execute method. You can also override the cancel and status poll methods.

Task Execution

Execute tasks using a pair of nodes: administrator and task owners. The administrator owner must be an active node in the metadata storage cluster. This node is responsible for starting and monitoring the task. The task owner can be any node that meets the location constraints specified by the task creator and executes the actual task. With the exception of subtasks, the Ocient System assigns both owners upon task creation, chosen randomly to balance the load.

To see the disposition of tasks in the system, see the `sys.subtasks` system catalog table. For details, see **sys.subtasks**.

Administrator Owner

The administration owner manages and monitors all steps in the life cycle of their own tasks. After the Ocient System adds a task to the cluster state and assigns an administrator owner, the Administrator Node sends a message to the task owner to submit or poll the current task status. After the task has started successfully, the administrator owner periodically polls the task owner for the task status, updating it in the cluster state as needed.

When a task completes, the administrator owner sets its state to a terminal status and no longer polls or interacts with the task. The Ocient System retains historical tasks in the cluster state until a configurable limit is reached to prevent the state from growing unboundedly.

Task Owner

The task owner maintains an in-memory map of running tasks by the identifier in the health protocol. The Ocient System pools tasks for execution status as needed. When the Ocient System receives a poll task request from an administrator owner, a task owner checks for the existence of the task, starting it if necessary and returning the current status.

The Ocient System starts a task by instantiating a task object using the factory type and then executing its pre-condition check, method, and post-condition check. If at any point the system encounters a failure, the system stops execution.

When a task completes, the task owner stores the results in memory until it receives an indication from the administrator owner that it is safe to remove the results.

Handle Outages

Administrator Owner

An important responsibility of the administrator owner is to detect and handle task owner outages. If a poll fails for any reason, the administrator owner reassigns the owner in the cluster state and starts the task in the new location. Tasks must remain unchanged when duplicated and resume where they stopped when restarted.

If an administrator owner fails, the administrator leader detects the connectivity change and reassigns the administrator owner. The new administrator owner begins polling the task owner when it receives the update, so the ownership change should not interfere with task execution.

Task Owner

Generally, the task owner does not need to handle outages. However, in extreme situations, e.g., network splits, an administrator owner might not be able to communicate with its task owner and orphan a task. To prevent orphaned tasks from running in multiple locations in the system, the task owner times out and cancels a running task if it has not heard from the administrator owner in multiple poll cycles.

Subtasks

When you execute a task using a SQL statement, the task performs the operation by executing one or multiple subtasks. Thus, when you query the system catalog tables for subtasks, you must look for numerous subtasks and their individual progress rather than just one task.

You can also execute tasks as a series of subtasks, that run sequentially, in parallel, or some combination of the two. The series comprises a parent task with a number of child tasks. Each child task has a specified ordering. The Ocient System executes tasks with the same ordering in parallel. And, the system executes groups of child tasks in sequence according to ascending order. The Ocient System assigns an administrator owner when the child task should begin running. The system executes parent tasks directly on the administrator owner. These tasks are only responsible for watching the status of their running child tasks and starting the next child tasks when applicable. Parent tasks complete after all their children tasks complete.

To see the disposition of subtasks in the system, you can query the `sys.subtasks` system catalog table. For details, see **sys.subtasks**.

Task Cancellation

A client can request to gracefully cancel a task by setting an `is_canceled` flag on the task object. The administrator owner evaluates this state on its next poll cycle and notifies the task owner that the task should be canceled. The exact mechanism by which the running task is canceled varies by type, as some tasks might need to communicate with other protocols to stop execution.

Canceling a parent task causes all of its subtasks to be canceled. A subtask cannot be canceled individually.

If the task is successfully canceled, execution ends with a `CANCELLED` status and reverts back to the administrator owner and Raft state using the normal success or failure paths.

For example, cancel the task named `my_task`.

SQL

```
CANCEL TASK 'my_task';
```

For the full syntax, see **CANCEL TASK**.

Related Links

Core Elements of an Ocient System

Distributed Tasks

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Remove Records from an Ocient System

Ocient® supports multiple options for removing records from a database. Here, you can find the different methods for removing data along with their advantages and drawbacks to help you determine the method that best fits your needs.

> ⚠ All options for removing data are destructive and cannot be undone after execution. Therefore, be cautious when removing any Ocient records from your database.

Required Permissions

To remove data, a user must have the `DELETE` privilege for the specified table. For details on privileges, see the **Data Control Language (DCL) Statement Reference**.

Remove Records Using DELETE

The `DELETE` SQL statement removes specific individual rows from a table based on the `WHERE` filter clause.

Unlike `TRUNCATE`, the `DELETE` statement does not restore the disk space of deleted rows without further system operations. In general, `DELETE` is a good option when removing small row sets, not entire tables.

If a `DELETE` statement has no `WHERE` filter, it removes all rows in the table.

The `DELETE` statement can support concurrent operations to remove rows on the same table, which might impact performance.

For details about the `DELETE` statement, see **DELETE FROM TABLE**.

Remove Data Using DELETE with a Filter

In this example, the `DELETE` statement removes all rows from the `products` table with a value less than 1 in the `rating` column.

© 2026 Ocient, Inc.

SQL

```
DELETE FROM products WHERE rating < 1;
```

Remove Data Using Joined Tables

You can use `DELETE` with a `JOIN` statement to help filter rows for removal based on a condition that applies to the second table.

> The statement removes rows only from the first table and not the second joined table.

This example deletes rows after performing a `JOIN` operation on the geospatial values in two tables:

- `point_table` contains a column of geospatial point values.
- `us_state_table` contains columns of all abbreviated U.S. state names and polygon values representing their boundaries.

The `JOIN` operation uses the **ST_INTERSECTS** function to evaluate if each point value in `point_table` is contained within any of the polygon values in the `us_state_table`.

The example deletes any rows in `point_table` that are within the boundaries of Texas, abbreviated as `'TX'`.

SQL

```
DELETE FROM point_table p
    JOIN us_state_table s ON ST_INTERSECTS(s.polygon_wkt, p.point_wkt)
WHERE s.state LIKE 'TX';
```

Data Durability Considerations

To remove data, the data must be queryable in the system. For example, streaming data systems might experience a delay before rows can be deleted while they are in progress of the loading process to become fully durable.

In most cases, data should be loaded and queryable within a matter of seconds.

Related Links

Key Concepts

Ingest Data with Legacy LAT Reference

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

System Administration

Learn how to install and set up an Ocient® System, including hardware and software requirements and ongoing maintenance.

See these pages for key concepts to understand the Ocient System design and setup.

Installation

These topics provide tutorials for installing Ocient on different system deployments. The reference pages in this section explain Ocient requirements and recommended system configurations.

- **Install an Ocient System**
- **Amazon Web Services Ocient Installation**
- **Google Cloud Platform Ocient Installation**
- **Configure Storage Spaces**
- **Ocient Software Upgrade**

Maintenance

Maintenance topics include tutorials for managing system hardware for stable performance. Also, you can learn how to add new hardware and restore the system for cases when drives or nodes fail or when a system requires expansion.

- **Guide to Rebuilding Segments**
- **Startup and Shutdown Procedure**
- **Power Safety Recommendations**
- **NVMe Drive Firmware Upgrade Process**
- **Replace an NVMe Node Drive**
- **Expand and Rebalance System**
- **Replace Nodes**

© 2026 Ocient, Inc.

- **Backup and Restore**

Monitoring

These topics explain Ocient System monitoring, including error logging, system statistics, and supported tools for system administration.

- **Log Monitoring**
- **Statistics Monitoring**
- **System Information REST Endpoints**
- **Set Up System Monitoring with the TIG Stack and Kapacitor**
- **Monitor Segment Group Transfers**

System Configuration

This set of topics explains how to inspect and alter the system configuration.

- **Inspect the Current Configuration**
- **Node Configuration with the Administrator Role**
- **Manage the Network Configuration of an Ocient System**
- **Secure Connections Using TLS**
- **Multiple Storage Clusters for Loading Data**
- **Air-Gapped Environment with an Ocient System**

Loading Administration

- **Alter Default Data Pipeline Behavior**
- **Configuration Settings for Data Pipelines**
- **Configure Data Pipeline Logging**

© 2026 Ocient, Inc.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Install an Ocient System

This section provides a quick tutorial for installing and configuring the Ocient® System using your data.

Installation of the Ocient System

Before installing, check with your systems administrator to ensure that you meet the **Ocient System Requirements** for the validated operating systems and hardware needed for optimized Ocient performance.

The installation and configuration of the Ocient System involves these steps:

1. **Ocient Application Installation**

   This step details how to download and install the Ocient System package.

2. **Ocient System Bootstrapping**

   This step describes the Ocient bootstrapping, which sets up the initial node configuration and connects the nodes together.

3. **Ocient Application Configuration**

   This step sets up your Ocient System with storage spaces, storage clusters, and node roles.

These pages provide additional information, validation tools, and configuration options for installing your Ocient® System.

- **Ocient System Sizing** — Guidance on the factors that influence Ocient System sizing.
- **Ocient System Requirements** — Information on hardware and system requirements for optimized Ocient performance.

© 2026 Ocient, Inc.

- **Ocient System Hardware Specifications** — Information on the detailed hardware specification for the Ocient System.
- **Operating System Configuration** — A walkthrough to configure the necessary prerequisites for system bootstrapping.
- **Checking System Configuration Before Bootstrapping** — Commands to capture critical node health states and metrics before beginning the bootstrapping process.
- **Checking System After Configuration** — Commands to capture critical node health states and metrics after the final configuration process.
- **Node Bootstrapping Reference** — Descriptions of additional options for advanced system configurations.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Application Installation

You can install the Ocient® software application by installing a single RPM or Debian® package.

Before installation, review your hardware sizing and ensure that it meets your needs. For details, see **Ocient System Hardware Specifications**. For a list of prerequisites needed for Ocient installation, see the **Ocient System Requirements**.

For access to Ocient software packages, please contact Ocient Support.

Application Installation

Commands

Consult the documentation for the package manager of your operating system for specific options and command-line syntax.

RPM Installation

Use this command to install the Ocient RPM package from the local filesystem. Include `./` or specify the full path to the RPM file; otherwise, the `dnf` command attempts to search the configured repositories instead of installing the local RPM file. In the following command example, replace `XX` with the Ocient package version of the provided file.

Shell

```
sudo dnf install ./ocient-XX-x86_64.rpm
```

DEB Installation

© 2026 Ocient, Inc.

Use this command to install the Ocient DEB package from the local filesystem. Include `./` or specify the full path to the DEB file, otherwise the `apt` command attempts to search the configured repositories instead of installing the local DEB file. In the following command example, replace `XX` with the Ocient package version of the provided file.

Shell

```
sudo apt install ./ocient-XX-amd64.deb
```

Next Step

When your system install completes, proceed to **Ocient System Bootstrapping**.

Related Links

Installation Reference

Install an Ocient System

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient System Bootstrapping

The Ocient® System runs the bootstrapping process one time per node. This process establishes the node identity, configures networking, and accepts nodes into the system.

The `/var/opt/ocient/bootstrap.conf` file controls the initial configuration of each node. The system uses this file only when the node initially starts.

Validate the System Before Bootstrapping

The following steps assume a basic OcientAIQ™ Unified Data Platform configuration, where every node has a single network interface.

To verify your system before the bootstrapping process:

- Check with your systems administrator to ensure that you meet the **Operating System Configuration** requirements. Before bootstrapping, your system must meet the requirements described in the **Apply Required Kernel Settings** and **Prepare NVMe Drives** sections.
- You can run a system check to validate the compliance of hardware, nodes, and the configuration for the Ocient System. For details, see **Checking System Configuration Before Bootstrapping**.

Initial Configuration

The system automatically initializes the first node bootstrapped in the system with the SQL and Administrator roles. Choose a node to serve as a SQL Node. This node is the initial node in the system. The hardware for each node type differs, so ensure that the node you select is intended for a SQL Node.

For new clusters, if you do not define the configuration for the SQL Nodes in a connectivity pool, the Ocient System assigns each SQL Node to their own unique connectivity pool named `<sql_node_name>_default`, where `<sql_node_name>` is the name of the SQL Node. However, the most common configuration of an Ocient System has all SQL Nodes in one connectivity pool.

© 2026 Ocient, Inc.

For an example of adding SQL Nodes to a connectivity pool, see **CREATE CONNECTIVITY POOL**. Change or troubleshoot the default configuration using the instructions in **Manage the Network Configuration of an Ocient System**.

Ocient System Bootstrapping Process

If you meet the system prerequisites, you can begin bootstrapping your Ocient System.

The bootstrapping process consists of these steps:

1. **Bootstrap the First Node**
2. **Bootstrap the Remaining Nodes**

1 Bootstrap the First Node

Bootstrap the Node

Connect to the initial SQL Node with the username and password of your server and the IP address of your node. This example connects as the administrator `admin` to the IP Address `10.10.10.10`.

Shell

```
ssh admin@10.10.10.10
```

Use your preferred text editor with `sudo` to create the `/var/opt/ocient/bootstrap.conf` file as root with these contents.

`/var/opt/ocient/bootstrap.conf` example

YAML

```
initialSystem: true
```

Start the database.

Shell

```
sudo systemctl start rolehostd
```

Verify that the node and the service are active by executing this status command.

Shell

```
systemctl status rolehostd

rolehostd.service - Rolehostd daemon startup
     Loaded: loaded (/etc/systemd/system/rolehostd.service; enabled; vendor
     Active: active (running) since Wed 2022-01-26 23:31:36 UTC; 7s ago
     ...
```

If the `rolehostd` service is running, you can also check the Ocient logs on your node. Search and ensure there are no [ERROR] log messages.

Shell

```
tail -f /var/opt/ocient/log/rolehostd.log
```

Verify Connection to the SQL Node

At this point, you have a running database with a single node. You should be able to connect to the database using **Connect Using JDBC** or **Connect Using pyocient** and execute commands.

Every new system starts with a `system` database. To connect to a new system, use the username and password configured in the `bootstrap.conf` file or the username `admin@system` and password `admin` if none was provided.

For example, assume your node named `sql0` has an IP address of `10.10.0.1`. Use the JDBC driver CLI to connect with this connection string.

Shell

```
connect to jdbc:ocient://10.10.0.1:4050/system user "myuser@system" using "
```

See the **JDBC Manual** for details on using the JDBC driver.

To see the roles running on the single node, execute this query.

SQL

```
select name, operational_status, software_version, array_agg(service_role_t
        from sys.node_status as ns
        left join sys.nodes as n  on ns.node_id = id
        left join sys.service_roles as sr on sr.node_id = n.id
        group by name, operational_status,software_version;
```

The initial node should be listed as running the `sql`, `admin`, `health`, and `operatorvm` roles. If all of these roles are present and the node is Active, you can proceed to the next step to bootstrap the remaining nodes.

2 Bootstrap the Remaining Nodes

Performing the bootstrapping process on the remaining nodes is identical on all nodes. The remaining nodes can be bootstrapped in any order.

On each node, log in using SSH and use your text editor with `sudo` to create the file `/var/opt/ocient/bootstrap.conf` that contains this text by replacing `<FIRST_NODE_IP_ADDRESS>` with the IP address of the initial node you created in Step 1.

`/var/opt/ocient/bootstrap.conf` example

YAML

```
adminHost: <FIRST_NODE_ADDRESS>
```

`<FIRST_NODE_IP_ADDRESS>` is the DNS name or IP address of the initial node.

You can obtain the IP Address of the initial node by executing `ifconfig` on that node.

If the password for the system administrator has changed, set the correct username `adminUserName` and password `adminPassword` in the bootstrap configuration file `bootstrap.conf`.

On each node, start the database.

Shell

```
sudo systemctl start rolehostd
```

When you replace Foundation Nodes, the Ocient System removes the prior node after the creation of the new node. Some queries of the system catalog tables might not return results until the prior node is removed.

At this point, all the remaining nodes are not configured with any roles. After all nodes have been started, you should see them when you execute this query with only the `health` role listed. This query uses the `sys.node_status`, `sys.nodes`, and `sys.service_roles` system catalog tables to retrieve node information for the node name, operation status, version, and all service role types. The query uses the **ARRAY_AGG** function to retrieve the service role type for all rows.

SQL

```
SELECT name, operational_status, software_version, array_agg(service_role_t
    FROM sys.node_status AS ns
    LEFT JOIN sys.nodes AS n ON ns.node_id = id
    LEFT JOIN sys.service_roles AS sr ON sr.node_id = n.id
    GROUP BY name, operational_status,software_version;
```

More Bootstrapping Options

For options on bootstrapping your Ocient System, see **Node Bootstrapping Reference**.

Next Step

When you have completed the bootstrapping process, you can proceed to the **Ocient Application Configuration** page to finish configuring Ocient.

Related Links

Installation Reference

Install an Ocient System

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Application Configuration

These steps assume that you have already completed the process for bootstrapping your Ocient® System nodes. If you have not done this yet, perform the **Ocient System Bootstrapping** steps before beginning this configuration process.

Now that the nodes have all completed bootstrapping, you must properly configure the nodes to perform their roles. You should run these commands using a SQL connection to one of the SQL Nodes in your system. For details, see **Connect to Ocient**.

The Ocient application configuration process consists of these steps:

1. **Creating a Storage Space**
2. **Creating Storage Clusters**
3. **Assigning Service Roles**
4. **Restart the Nodes**

1 Creating a Storage Space

To create a storage space with 12 segments, use `WIDTH=12` and `PARITY_WIDTH=2`. Two of these segments are the parity portion, and 10 of these segments are the data portion.

SQL

```
CREATE STORAGESPACE teststoragespace WIDTH = 12, PARITY_WIDTH = 2;
```

The storage space name must begin with a letter followed by letters, numbers, and underscores.

You can view all storage spaces in the system by querying the `sys.storage_spaces` system catalog table.

© 2026 Ocient, Inc.

```sql
SELECT * FROM sys.storage_spaces;
```

For details about creating a storage space to suit your system needs, see **Configure Storage Spaces**.

2 Creating Storage Clusters

You must have at least one storage cluster to configure an Ocient System. A storage cluster is a set of Foundation Nodes with an associated storage space that coordinate to reliably store data and serve queries.

Create a Foundation cluster.

```sql
CREATE CLUSTER lts_cluster
TYPE LTS storagespace "teststoragespace"
PARTICIPANTS=('lts0','lts1','lts2','lts3','lts4','lts5','lts6','lts7','lts8
```

Loader Nodes and SQL Nodes do not need to be placed into a cluster.

At a minimum, the number of participants must be the width specified in the storage space. This number can be greater if you overprovision the system. For details about storage spaces and related settings, overprovisioning, and parity, see **Core Elements of an Ocient System**.

View all clusters in the system.

```sql
SELECT * FROM sys.clusters;
```

View all clusters and their associated nodes.

SQL

```sql
SELECT c.name AS cluster_name, c.cluster_type, n.name AS node_name
FROM sys.node_clusters nc
JOIN sys.clusters c ON nc.cluster_id = c.id
JOIN sys.nodes n ON nc.node_id = n.id;
```

⚠ You must restart all nodes with the SQL role after adding a storage cluster.

3 Assigning Service Roles

Assign service roles to each accepted node. This process defines how each node participates in queries and the management of the system.

View all nodes and their configured roles.

SQL

```
SELECT n.name AS node_name, sr.service_role_type
FROM sys.nodes n
LEFT JOIN sys.service_roles sr ON sr.node_id = n.id;
```

The addition of nodes to the storage cluster in the prior step automatically assigns `lts` to the Foundation Nodes that you included.

Add the remaining `sql`, `admin`, and `streamloader` roles to the appropriate nodes.

SQL

```
ALTER NODE "sql1" ADD ROLE sql;
ALTER NODE "sql1" ADD ROLE admin;
ALTER NODE "loader1" ADD ROLE streamloader;
```

ⓘ The `Admin` role can operate standalone or on the same node as the `SQL` role.

4 Restart the Nodes

With roles assigned to the nodes, you can restart the `rolehostd` process on all nodes. At a shell terminal on each node, run this command.

Shell

```
sudo systemctl restart rolehostd
```

© 2026 Ocient, Inc.

> If any custom configuration settings are applied to running nodes during other configurations, be sure to restart the `rolehostd` process before proceeding.

You can check the status of nodes using the Status endpoint on each node from a terminal. Assuming IP Addresses `10.10.0.2` and `10.10.0.3` for `sql1` and `lts0`, respectively, you can check the status with these curl commands.

CURL

```
curl http://10.10.0.2:9090/v1/status
curl http://10.10.0.3:9090/v1/status
```

After all of the nodes are `Active`, Ocient is ready to use for queries.

Creating an Ocient System with Multiple Storage Clusters

If you need an Ocient System that requires 10 or more Foundation Nodes in the storage cluster, then you can create multiple storage clusters. For considerations to take into account when you work with multiple storage clusters, see **Multiple Storage Clusters for Loading Data**.

Create a multiple storage cluster with 12 nodes. Create a storage space `store` with width `5` and parity width `1` using the `CREATE STORAGESPACE` SQL statement.

SQL

```
CREATE STORAGESPACE store WIDTH = 5, PARITY_WIDTH = 1;
```

Create multiple clusters using the `CREATE CLUSTER` SQL statements using the storage space `store`. Split the Foundation Nodes among the clusters. `cluster_0` storage cluster has Foundation Nodes `lts0` through `lts5`, while `cluster_1` has `lts6` through `lts11`.

SQL

```
CREATE CLUSTER cluster_0 TYPE=lts PARTICIPANTS=("lts0", "lts1", "lts2", "lts3", "

CREATE CLUSTER cluster_1 TYPE=lts PARTICIPANTS=("lts6", "lts7", "lts8", "lts9", "
```

Validating the System After Configuration

After configuration, you can run a system check to validate the Ocient installation and bootstrapping. For details, see **Checking System After Configuration**.

Next Step

Follow the process in **Ingest Data with Legacy LAT Reference** to create a database and a table, then load data from a set of files located on a cloud object storage.

Related Links

Installation Reference

Install an Ocient System

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Installation Reference

These pages provide additional information, validation tools, and configuration options for installing your Ocient® System.

- **Ocient System Sizing** — Guidance on the factors that influence Ocient System sizing.
- **Ocient System Requirements** — Information on hardware and system requirements for optimized Ocient performance.
- **Ocient System Hardware Specifications** — Information on the detailed hardware specification for the Ocient System.
- **Operating System Configuration** — A walkthrough to configure the necessary prerequisites for system bootstrapping.
- **Checking System Configuration Before Bootstrapping** — Commands to capture critical node health states and metrics before beginning the bootstrapping process.
- **Checking System After Configuration** — Commands to capture critical node health states and metrics after the final configuration process.
- **Node Bootstrapping Reference** — Descriptions of additional options for advanced system configurations.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient System Sizing

The Ocient® System must be the proper size for your data and query requirements. Follow these steps to determine the size of your Ocient System. Contact **Ocient** to aid with the Ocient System calculations, established configuration, and node quantities.

1. Gather the uncompressed size of your current database or data and its format. If your data resides in a database, determine the compression ratio of the data. Further, consider whether your data is streaming or file-based. For streaming or time series data, investigate the level of data granularity (for example, minute, hour, or day).
2. Calculate the quantity of data (in Gbps or the number of rows in a specific unit of time) that you plan to load into the Ocient System and at what cadence (for example, continuous loading or specific schedule). Describe the latency you expect between the data creation and when data reaches the destination. As you consider the quantity, determine the percent of NULL data. Also, think about the retention period required for the data.
3. Determine the level (small, medium, or large) of data transformation in the data pipeline loading process. See whether you require any user-defined transformations. Investigate whether your transformations are done programmatically or using SQL. After the initial data load, see if you require more transformations, such as cubes or views.
4. Investigate the types of queries you plan to execute (for example, simple filters, complex aggregation or sorting, complex joins, or transformation). In addition, review your query workload (number of queries in a specific unit of time) and how many queries you expect to execute in parallel. Consider any objectives you have for system readiness to execute queries.
5. Calculate the compressed data volume. Consider your compression requirements for the stored data (for example, Global Dictionary Compression).
6. Add index, statistics, working space, and parity size to the compressed data volume.
7. Use the established storage space configuration in the Ocient System for data nodes, parity nodes, and the number of clusters.

© 2026 Ocient, Inc.

8. Using the known size and level of transformation in loading, determine the total number of cores, memory, and storage for your requirements.
9. Calculate the number of SQL Nodes, Loader Nodes, Foundation Nodes, Metadata Nodes, and supporting nodes. Ocient personnel assist with this step.

The number of specific nodes determines the size of the Ocient System. After you have the latter numbers, then you can purchase, install, and configure the system.

Related Links

Ocient System Requirements

Ocient System Hardware Specifications

Table Compression Options

Global Dictionary Compression

Secondary Indexes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient System Requirements

In order to run the OcientAIQ™ Unified Data Platform, you must meet several requirements for your system hardware, OS, and software. The requirements ensure that your system can run an optimized and fully functional Ocient® installation.

These requirements apply to the Foundation, Loader, and SQL Nodes.

> Consult with Ocient or the System Administrator installing Ocient to determine the optimal configuration for your Ocient deployment.

Server Hardware Requirements

These are the hardware requirements for the OcientAIQ Unified Data Platform node servers. These requirements include the Foundation, Loader, SQL, and Metadata Node types. Data (i.e., non-OS) drive requirements are listed separately.

Foundation, Loader, and SQL Nodes

Component	Requirements	Notes
Chassis/System	8+ NVMe hotswap bays Dual CPUs M.2 slots for OS drives Redundant power supplies	Supermicro® SYS-120U-TNR or similar recommended Full bandwidth PCIe connections to NVMe drives are recommended (i.e., not attached to a RAID card and not multiplexed). Drives must appear as `nvme` devices. NVMe PCIe lanes balanced between CPUs recommended (i.e. equal number of drives connected to CPU1 and CPU2).
CPU	2x Intel® Skylake or AMD® Rome or newer x86-64 server CPUs	26+ cores per CPU recommended. Must support AVX2 instruction set.
Memory	1TB+ recommended All CPU memory channels must be populated to achieve maximum memory bandwidth.	Balanced memory configuration with all memory channels populated required to achieve maximum bandwidth.

© 2026 Ocient, Inc.

Component	Requirements	Notes
High Speed Network	100Gbit dual port ethernet NIC	NVIDIA® Mellanox® ConnectX-5 MCX516A-CCAT recommended. Dual ports required for redundancy.
OS Drives	1TB+ Enterprise NVMe drives (M.2)	RAID-1 mirroring recommended. M.2 format recommended to keep all U.2 bays available for data drives.
Data Drives	See **Data Drive Requirements**.	NVMe drives for short-term and long-term data storage.

Metadata Nodes

The optional Metadata Nodes have fewer requirements than the other nodes.

Component	Requirements	Notes
Chassis/System	Virtual Machine or Physical Machine	No specific chassis requirements. The system can be a virtual machine.
CPU	16+ cores Xeon-D or similar x86-64 CPU	Must support AVX2 instruction set. You might need to change the virtual machine CPU type to support AVX2 instructions.
Memory	64GB	None.
Network	10 Gbps NIC or virtual NIC	Metadata Nodes do not require high-speed networking. Dual ports are required for redundancy.
OS Drive	512GB Enterprise SSD or SSD-backed storage pool	NVMe is recommended.

Data Drive Requirements

Ocient strictly requires high-performance enterprise or data center NVMe SSD drives for long-term and short-term data storage. The drives listed here have been validated for functionality and performance. They also support drive self-encryption using the OPAL standard. All data stored is encrypted at rest. Power loss protection (PLP), standard on enterprise and datacenter drives, is also required. An endurance of 0.5 drive writes per day (DWPD) or higher is recommended for Foundation and SQL Nodes, endurance of 1.0 DWPD is recommended for Loader Nodes.

- Solidigm™ D5-P5430 OPAL U.2 3.84TB-15.36TB
- Solidigm D7-P5520 OPAL U.2 1.92 TB-15.36TB
- Solidigm D7-P5510 OPAL U.2 3.84TB, 7.68TB
- Samsung® PM9A3 U.2 960GB-15.36TB
-

> ⚠ Solidigm P5510 firmware versions VDV10151 and VDV10152 are known not to work properly with OPAL enabled. Upgrade the firmware to a known working version.

> ⚠ Samsung PM9A3 firmware versions GDD5102Q and GDD5202Q are known to have issues with the `vfio-pci` driver. Upgrade the firmware to a newer version.

Enterprise NVMe drives with similar or better performance characteristics and OPAL (or better) support can be considered for validation.

> ⓘ If the Ocient System detects hardware support for the OPAL standard, then nodes enable full disk encryption on data drives by default. Ocient stores the encryption keys on the operating system drive. For the complete list of directories to back up, see **Backup and Restore**.

Compatible Operating System Versions

These operation systems and versions are fully supported by Ocient:

- Red Hat® Enterprise Linux® (RHEL®) version 8* and version 9
- Ubuntu® 20.04, 22.04, 24.04
- Debian® 11, 12

> ⓘ RHEL versions 8.3 and prior require a kernel upgrade for compatibility. There is a bug in the `uio_pci_generic` driver in the kernel provided with those versions that prevents Ocient from communicating with the NVMe drives.

Network Requirements

Ocient requires high-bandwidth connections between its database nodes to achieve high performance. It is critical for performance that the Foundation and SQL Nodes can

simultaneously utilize high-bandwidth links without bottlenecks. Generally, this means that those nodes share the same switch backplane for best performance.

> The minimum recommended link speed is 100Gbps.

Redundancy

Redundant networking is recommended for production use cases. On the server side, dual port NICs are required to support redundancy. Multi-chassis link aggregation (MLAG/CLAG/MCLAG) using redundant switches is the recommended form of redundancy.

Network Switch

These are the requirements for the network switches to support Ocient. No other advanced L2 or L3 features are required.

1. Enough 100G ports for all Foundation Nodes on the same backplane for best performance.
2. Multi-chassis link aggregation for redundancy (MLAG/CLAG/MCLAG depending on vendor terminology).

Security Requirements

To secure data stored on disk, Ocient supports and recommends SEDs (Self-Encrypting Drives) that meet the OPAL (or better) standard. Using drives that support this standard, Ocient software enables encryption of data at rest on the drives. This encryption prevents unwanted data leaks, as any time a disk is removed from the system, it becomes locked, and the data can only be recovered with a key-encrypting-key that is stored off-drive.

To secure network communications, see the **Ocient Security Guide**.

Additional Software Requirements

Ocient depends directly on these software packages:

- `python3`

- `sudo`
- `lz4`
- `cronie`
- `logrotate`
- `gzip`
- `pciutils`
- `ca-certificates`
- `libquadmath` (RHEL), `libquadmath0` (Ubuntu/Debian)

For database nodes, these packages are installed automatically as part of the package dependencies.

Ocient also depends on a separate installation of Java® 21, where the `openjdk-21-jre-headless` package is recommended.

Next Step

If your system meets these requirements, proceed to **Ocient Application Installation**.

Related Links

Ocient Software Upgrade

Install an Ocient System

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient System Hardware Specifications

Depending on your deployment modality and your purchase agreement with Ocient®, the Ocient System can run on hardware bearing the Ocient brand. Here, you can learn about the specifications for the hardware. However, an Ocient System can run on generic hardware that meets the system requirements. For details, see **Ocient System Requirements**.

The Ocient System contains Foundation Nodes, Loader Nodes, and SQL Nodes. For details about these node types, see **Core Elements of an Ocient System**.

For sizing considerations, see **Ocient System Sizing**. Then, you can further optimize the system by using compression at the table level. For details, see **Table Compression Options**. After you have the hardware and size defined, you can install the Ocient System. For details, see **Install an Ocient System**.

Foundation Node Specifications

Components

This table provides the components and their descriptions and quantities for the hardware.

Component	Description	Quantity
Chassis	Supermicro® Hyper A+ Server AS -2126HS-TN — 24 NVMe Drive 2.5” drive bays For details, see the **Hyper A+ Server AS -2126HS-TN specification**.	1
Processors	Dual AMD® EPYC™ 9654 Processors — 96 cores per processor	2
Memory	96GB DDR5-4800 RDIMM	24
100Gbps NIC	NVIDIA® Mellanox® 100Gbps Network Card — MCX623106AN-CDAT PCIE 2port 100GB QSFP28	1
Trusted Platform Module (TPM)	TPM 2.0	1
Boot Drive RAID Carrier	AOC-SLG4-2H8M2-O, 2x Hybrid Gen 4 NVMe/SATA M.2 RAID 0/1	1

© 2026 Ocient, Inc.

Component	Description	Quantity
	Carrier, Standard	
DCMS License	Supermicro Data Center Management Suite License	1
Boot Drive	Samsung® PM983 1.9TB 1 DWPD NVMe PCIe3x4 M.2 22x110 SED	3
Storage Drives	Solidigm™ D5-P5430 [7.68TB, 15.36TB, 30.2TB] NVMe U2 15 mm SED	12

Physical Specifications

This table provides the specifications and their corresponding values for the hardware.

Specification	Value
Form Factor	2U Rackmount
Height	3.5” (88.9 mm)
Width	17.2” (437 mm)
Depth	31.74” (806.2 mm)
Package	9.96” (H) x 26.46” (W) x 43.31” (D)
Net Weight	45 lbs (20.5 kg)
Gross Weight	75 lbs (34 kg)
Operating Temperature:	10°C to 35°C (50°F to 95°F)
Non-operating Temperature	-40°C to 70°C (-40°F to 158°F)
Operating Relative Humidity	8% to 90% (non-condensing)
Non-operating Relative Humidity	5% to 95% (non-condensing)
Power supplies	2000W Redundant Power Supplies
Peak Load Wattage	1350W
Average Load Wattage	810W (varies based on workload)
Min Load Wattage	240W

Performance Specifications

This table provides different foundation hardware performance specifications for Foundation Nodes.

Node Type	Cores	Memory (TB)	Drives (TB)	Drives (Numbe
Foundation Node — Large Storage Optimized	192	2.3	7.68	12
Foundation Node — Extra Large Storage Optimized	192	2.3	15.36	12
Foundation Node — Double Extra Large Storage Optimized	192	2.3	30.72	12

Loader Node and SQL Node Specifications

The components, specifications, and sizing for Loader Nodes and SQL Nodes are the same.

Components

This table provides the components and their descriptions, and quantities for the hardware.

Component	Description	Quantity
Chassis	Supermicro Hyper A+ Server AS -2126HS-TN — 24 NVMe Drive 2.5” drive bays For details, see the **Hyper A+ Server AS -2126HS-TN specification**.	1
Processors	Dual AMD EPYC 9654 Processors — 96 cores per processor	2
Memory	96GB DDR5-4800 RDIMM	24
100 Gbps NIC	Mellanox 100Gbps Network Card — MCX623106AN-CDAT PCIE 2port 100GB QSFP28	1

Component	Description	Quantity
Trusted Platform Module (TPM)	TPM 2.0	1
DCMS License	Supermicro Data Center Management Suite License	1
Boot Drive	Samsung PM983 1.9TB 1 DWPD NVMe PCIe3x4 M.2 22x110 SED	1
Storage Drives	Solidigm D7-P5520 1.92TB NVMe U2 15 mm SED	8

Physical Specifications

This table provides the hardware specifications and their corresponding values.

Specification	Value
Form Factor	2U Rackmount
Height	3.5” (88.9 mm)
Width	17.2” (437 mm)
Depth	31.74” (806.2 mm)
Package	9.96” (H) x 26.46” (W) x 43.31” (D)
Net Weight	45 lbs (20.5 kg)
Gross Weight	75 lbs (34 kg)
Operating Temperature:	10°C to 35°C (50°F to 95°F)
Non-operating Temperature	-40°C to 70°C (-40°F to 158°F)
Operating Relative Humidity	8% to 90% (non-condensing)
Non-operating Relative Humidity	5% to 95% (non-condensing)
Power supplies	2000W Redundant Power Supplies
Peak Load Wattage	1050W
Average Load Wattage	630W (varies based on workload)
Min Load Wattage	220W

© 2026 Ocient, Inc.

Performance Specification

This table provides the hardware performance specification for a Loader Node and SQL Node.

Node Type	Cores	Memory (TB)	Drives (TB)	Drives (Numbe
Loader or SQL Node	192	2.3	1.92	8

Related Links

Install an Ocient System

Operating System Configuration

Checking System Configuration Before Bootstrapping

Ocient System Sizing

Table Compression Options

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Operating System Configuration

The Ocient® System Configuration procedure assumes that these steps have been performed on the Foundation Nodes and Loader Nodes in the database cluster. Some of these steps require root user permissions.

Use `sudo` when performing these steps in your command-line interface.

1 Prepare the Operating System Configuration

You must prepare the operating system with this configuration to run Ocient software:

- Configure Network interface controllers (NICs) and firewall for use by Ocient.
- Set the timezone to UTC.
- If OPAL drives are installed, unlock or revert them to factory state.
- Optional: Mount points `/opt/ocient` and `/var/opt/ocient` are mounted on separate storage devices.

2 Confirm the Ocient Installation

To check the installation, run this command on each node to show the software version.

Shell

```
/opt/ocient/current/rolehostd --build-info
```

If the software is not installed or is the wrong version, see the **Ocient Application Installation** instructions.

3 Apply Required Kernel Settings

© 2026 Ocient, Inc.

In order to achieve high-performance levels, you must set specific kernel settings. You cannot set these settings at runtime. Therefore, the system must pass these parameters to the kernel at boot time.

Automatically Apply Kernel Settings

You can automatically configure the kernel settings and apply them by running the `ockernelparams` script in the Ocient software package. This script detects the current system hardware and adjusts the `grub` bootloader kernel parameters. Then, the script updates the `grub` configuration with `grub2-mkconfig` or `update-grub` (depending on the Linux® distribution).

Run this command and specify the appropriate node role.

Shell

```
/opt/ocient/scripts/ockernelparams --node-role {sql,loader,foundation,metad
```

> You can also manually configure kernel settings. See **Manual Kernel Settings** for more information.

Check Current Kernel Settings

Run this command to show the current settings for the kernel that is running.

> You need to reboot after any changes are made to kernel parameters in the bootloader configuration.

Shell

```
cat /proc/cmdline
```

Text

```
BOOT_IMAGE=(hd0,gpt2)/boot/vmlinuz-4.18.0-305.19.1.el8_4.x86_64 root=UUID=1
```

The exact values can vary depending on the DRAM of the machine and the intended role for the node.

Check that the number of free 1G huge pages matches the number specified at the command line. You can check this number using this command.

Shell

```
cat /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages
```

Output

Bash

```
1024
```

The returned value should match your configuration.

Manual Kernel Settings

This section describes the optional kernel settings that you can manually configure. To automatically configure these settings, see **Automatically Apply Kernel Settings**.

You must add these settings that follow the `grub` bootloader kernel parameters for the listed applicable node types.

Setting: `transparent_hugepages`

Ocient requires that `transparent_hugepages` be disabled because Ocient software needs strict control over which huge pages it allocates.

Applicable Node Types:

- Foundation Nodes
- Loader Nodes
- SQL Nodes

Kernel Setting:

```shell
transparent_hugepage=never
```

Setting: `isolcpus`

To give Ocient software exclusive use of the majority of the cores on the system, the kernel is isolated from the cores using the `isolcpus` parameter. This isolation ensures that the system executes the kernel code only on a small number of cores where Ocient is not running logic that is critical for performance.

The pattern is to isolate all but the last two physical cores of each CPU from the kernel for CPUs with 26 or more cores. This means that two cores (4 threads) per CPU are available to the kernel.

For CPUs with fewer than 26 cores, all but the last physical core of each CPU is isolated.

Applicable Node Types:

- Foundation Nodes

Kernel Setting:

- Dual 28-core CPUs (Intel® 6348)

```shell
isolcpus=0-25,28-53,56-81,84-109
```

- Dual 32-core CPUs (Intel 6338)

```shell
isolcpus=0-29,32-61,64-93,96-125
```

© 2026 Ocient, Inc.

- Dual 96-core CPUs (AMD® 9654)

```shell
isolcpus=0-93,96-189,192-285,288-381
```

Setting: Huge Pages

Ocient utilizes huge pages to streamline the management of large amounts of memory in the system. The system uses 2MB and 1GB huge pages, depending on the node type. These huge pages must be reserved at boot time because otherwise, the system memory quickly becomes too fragmented to allocate the number of huge pages needed at runtime. These settings are the expected huge page settings.

> Metadata Nodes do not use huge pages.

Node Type	Installed DRAM	Huge Page Kernel Parameters
Foundation or SQL	768 GB	`hugepagesz=1G hugepages=384` `hugepagesz=2M hugepages=16384`
Foundation or SQL	1024 GB	`hugepagesz=1G hugepages=680` `hugepagesz=2M hugepages=16384`
Foundation or SQL	1152 GB	`hugepagesz=1G hugepages=768` `hugepagesz=2M hugepages=16384`
Foundation or SQL	1536 GB	`hugepagesz=1G hugepages=1024` `hugepagesz=2M hugepages=16384`
Foundation or SQL	2048 GB	`hugepagesz=1G hugepages=1364` `hugepagesz=2M hugepages=16384`
Foundation or SQL	2304 GB	`hugepagesz=1G hugepages=1536` `hugepagesz=2M hugepages=16384`
Loader	768 GB	`hugepagesz=1G hugepages=384` `hugepagesz=2M hugepages=16384`
Loader	1024 GB	`hugepagesz=1G hugepages=512` `hugepagesz=2M hugepages=16384`

© 2026 Ocient, Inc.

Node Type	Installed DRAM	Huge Page Kernel Parameters
Loader	1152 GB	`hugepagesz=1G hugepages=576` `hugepagesz=2M hugepages=16384`
Loader	1536 GB	`hugepagesz=1G hugepages=768` `hugepagesz=2M hugepages=16384`
Loader	2048 GB	`hugepagesz=1G hugepages=1024` `hugepagesz=2M hugepages=16384`
Loader	2304 GB	`hugepagesz=1G hugepages=1152` `hugepagesz=2M hugepages=16384`
Metadata (standalone administrator)	N/A	No huge pages. Use the `noHugePages` parameter when bootstrapping node: `noHugePages: true`

4 Prepare NVMe Drives

> These steps are for Foundation Nodes, Loader Nodes, and SQL Nodes only.

To prepare NVMe drives on Foundation Nodes, Loader Nodes, and SQL Nodes, these nodes must be in a state where they are connected to the uio_pci_generic or vfio-pci driver and not contain a partition table. To initialize the drives for use by Ocient that have been previously partitioned, you must perform these steps.

1. Identify the OS drives. The OS drives have partitions that are mounted to mount points such as `/`. In this example, `nvme0n1` is the OS drive.

Shell

```
lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
nvme0n1     259:0    0  1.8T  0 disk
├─nvme0n1p1 259:1    0  512M  0 part /boot/efi
└─nvme0n1p2 259:2    0  1.8T  0 part /
```

2. Ensure all drives are bound to the NVMe driver so the NVMe drives show up as standard NVMe devices. List the NVMe drives to identify the drive identifiers.

© 2026 Ocient, Inc.

Shell

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-nvme

/opt/ocient/scripts/nvme-driver-util.sh

BDF     Numa Node   Driver name     Device name
0000:01:00.0    0       nvme            nvmeOn1
0000:3b:00.0    0       nvme            -
0000:3c:00.0    0       nvme            -
0000:5e:00.0    0       nvme            -
0000:5f:00.0    0       nvme            -
0000:86:00.0    1       nvme            -
0000:87:00.0    1       nvme            -
0000:88:00.0    1       nvme            -
0000:89:00.0    1       nvme            -
```

3. For each NVMe drive that you need to erase and prepare for Ocient (not the OS drive), run this command by replacing `X` and `Y` to match the device name.

Shell

```
sudo nvme format /dev/nvmeXnY format --ses=1
```

⚠ This command formats the specified drive and erases all data on the drive. Ensure that the OS drive is not formatted if it is an NVMe drive.

Some drives do not support the `nvme format` command. In that case, you can overwrite the partition table by zeroing out the beginning of the drive with this command. This command does not securely erase the whole drive, but does allow it to be used by Ocient.

Shell

```
sudo dd if=/dev/zero of=/dev/nvmeXnY bs=1024K count=1024
```

© 2026 Ocient, Inc.

4. Bind the drives to the UIO or VFIO driver by running this command. The script automatically chooses the appropriate driver after detecting if the `IOMMU` setting is enabled. Any NVMe drives with mounted partitions (i.e. the OS drive) remain on the NVMe driver. Show all expected NVMe drives on the node attached to the UIO or VFIO driver.

Shell

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-vfio

/opt/ocient/scripts/nvme-driver-util.sh

BDF     Numa Node   Driver name     Device name
0000:01:00.0    0       nvme            nvme0n1
0000:3b:00.0    0       vfio-pci        -
0000:3c:00.0    0       vfio-pci        -
0000:5e:00.0    0       vfio-pci        -
0000:5f:00.0    0       vfio-pci        -
0000:86:00.0    1       vfio-pci        -
0000:87:00.0    1       vfio-pci        -
0000:88:00.0    1       vfio-pci        -
0000:89:00.0    1       vfio-pci        -
```

5 Clear Files from Any Previous Ocient Setup

To bootstrap a node that was previously part of an Ocient System, you must clear a set of files from any previous setup of Ocient from the `/var/opt/ocient/` directory. These commands remove these files.

Shell

```
sudo rm -f /var/opt/ocient/rolehostd.conf
sudo rm -f /var/opt/ocient/ocient.key
sudo rm -f /var/opt/ocient/ocient.crt
sudo rm -f /var/opt/ocient/ocient.crt.original
sudo rm -f /var/opt/ocient/*.raft
sudo rm -f /var/opt/ocient/bootstrap.completed
```

© 2026 Ocient, Inc.

Do not remove files in the `localKeyStore` directory if OPAL drives are present and encryption is not disabled. If these files are removed, the drive encryption must be reset using an external utility.

Related Links

Ocient Software Upgrade

Install an Ocient System

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Checking System Configuration Before Bootstrapping

The `precheck` command helps capture critical node health states and metrics before bootstrapping your Ocient® installation.

You can run the command as shown in this example. The flags included after the `precheck` command are optional. See **Optional Flags for Command Execution** for descriptions.

Shell

```
$ sudo precheck --print-level verbose --node-type metadata --logfile-location /home
```

The command invokes a script from the command line on operating systems supported by Ocient. The script sends output to standard output (STDOUT) and standard error (STDERR). To specify the file option, use this flag: `--logfile-location <directory>` where `<directory>` is the directory where the `precheck` command creates the log file. Critically, this script captures information about huge page configuration.

Optional Flags for Command Execution

Use the `precheck` command to run the script that checks the system configuration. You can execute the command using the optional flags shown in this table.

Flag Name	Values	Default	Description
—logfile-location	Non-empty string	`/var/opt/ocient/log`	Name of the directory that contains the log file created from the execution of the script.
—print-level	verbose or silent	verbose	Denotes the level of detail in the

© 2026 Ocient, Inc.

Flag Name	Values	Default	Description
			output. See the “Outputs” section.
—node-type	metadata, lts, streamloader, sql, or unknown	unknown	Denotes the current node type. If you do not specify this setting, the script tries to infer node type based on metrics it collects.
—timeout	Any integer	3	Denotes the default timeout (in seconds) to run any command-line command in the script. In most cases, the default is enough, so you do not need to specify this setting.
—nvme-driver-util-path	Non-empty string	/opt/ocient/scripts/nvme-driver-util.sh	Path to `nvme-driver-util` script that the precheck scrip uses. The default value is the location of the script on most installations, so in most cases, you do not need to specify this setting.
—json	Not applicable	False	If you specify the `--json` flag, the command prints output in JSON format to STDOUT. If you do

Flag Name	Values	Default	Description
			not specify this flag, the commanc prints human-readable output to STDOUT.

Command Output

The command has two types of output, silent or verbose, which you can specify by using the `--print-level` flag at the command line.

Silent

The `silent` value for the `--print-level` flag only produces a status code. The code is 0 for success, 1 for error, and 2 for warning.

Verbose

The `verbose` value for the `--print-level` flag prints either human-readable output by default or output in JSON format (if you specify the `--json` flag). The output contains all status details.

This example shows human-readable output.

Text

```
Warnings:
- isolcpus not configured - recommended for performance.
- No eligible NVMe payload drives found for node.
- bootstrap.conf file does not exist. Create a file at /var/opt/ocient/bootstrap.co
- SELinux is enabled. This might prevent Ocient from functioning properly. Please d

Memory - OK
======
ramconfig - SUCCESS: RAM configuration is adequate.

Firmware and Hardware - ERROR
====================
nonbootdrives - WARNING: No eligible NVMe payload drives found for node
...
```

This example shows output in JSON format.

JSON


```
{"status": 1,
  "components": {
    "enforce_status": {
      "status": 2,
      "message": "WARNING: SELinux is enabled. This might prevent Ocient from funct
    }
    "bootstrap": {
      "status": 2,
      "message": "WARNING: bootstrap.conf file does not exist. Create a file at /va
    }
    ...
  }
}
```


Related Links

Ocient System Bootstrapping

Ocient Application Configuration

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Checking System After Configuration

The `postcheck` command helps capture critical node health states and metrics after the Ocient® configuration process.

You can run the command as shown in this example. The flags included after the `precheck` command are optional. See **Flags for Command Execution** for descriptions.

Shell

```
$ sudo /usr/local/bin/postcheck --dsn ocient://<user>:<password>@<host>:<port>/<dat
```

This table describes each placeholder in the DSN:

Placeholder	Description
`<user>`	The username for the Ocient database connection (for example, `admin@system`).
`<password>`	The password associated with the specified user.
`<host>`	The hostname or IP address of the SQL Node to connect to.
`<port>`	The port number on which the SQL Node is listening.
`<database>`	The name of the database to connect to.

The command invokes a script from the command line on operating systems supported by Ocient. The script sends output to standard output (STDOUT) and standard error (STDERR) with the option to send a text file or a JSON file.

To specify the file option, use this flag: `--logfile-location <directory>` where `<directory>` is the directory where the `postcheck` command creates the log file.

Before you begin, ensure that all steps in the **Ocient System Bootstrapping** process are complete.

Prerequisites

The script requires Java® and the Ocient JDBC JAR to run. In most cases, you should use this script on client servers rather than database nodes.

To get the post-check script onto a client, perform the `scp` operation on the wheel onto the server and run `python3 -m pip install --no-index <path_to_wheel>/postcheck*.whl`, and replace `<path_to_wheel>` with the path to the directory of the wheel file on your server.

Flags for Command Execution

Use the `postcheck` command to run the script that checks the system configuration after you configure Ocient. You can execute the command using these flags.

Flag Name	Required	Values	Default	Descript
—dsn	Yes	Many	Not applicable	Data sour (DSN). Th prompts f credentia
—logfile-location	No	Non-empty string	/var/opt/ocient/log	Name of contains t created fr execution
—print-level	No	verbose or silent	verbose	Denotes t detail in t the “Outp

Flag Name	Required	Values	Default	Descript
—timeout	No	Any integer	3	Denotes t timeout (i running a line comn script. In the defau so you do specify th
—jar-path	No	Non-empty string	/opt/ocient/current/ocient-jdbc4.jar	Denotes t Ocient JDI default va usual loca file on an System. T be accura the postc to run cor
—json	No	Not applicable	False	If you spe json flag, prints out format to you do nc flag, the c prints hur output to

Command Output

The command has two types of output, silent or verbose, which you can specify by using the `--print-level` flag at the command line.

Silent

The `silent` value for the `--print-level` flag only produces a status code. The code is 0 for success, 1 for error, and 2 for warning.

Verbose

The `verbose` value for the `--print-level` flag prints either human-readable output by default or output in JSON format (if you specify the `--json` flag). The output contains all status details.

This example shows human-readable output.

Text

```
Warnings:
- Foundation Nodes storage are non-uniform.

OS Configuration - OK
================
Huge Page Settings - OK
isolcpus settings - OK

Memory - OK
======
Foundation Nodes: 1024 GB
SQL Nodes: 1024 GB
Loader Nodes: 1536 GB
Total DRAM: 123120 GB
...
```

This example shows output in JSON format.

JSON


```
{"status": 2,
  "components": {
    "firmware": {
      "status": 0,
      "message": "SUCCESS: NVMe drive firmware is consistent across nodes."
    }
    "symmetry": {
      "status": 2,
      "message": "WARNING: non-uniform configuration."
      "storage": {
        "status": 2,
        "foundation": {
          "status": 0,
          "message": "SUCCESS: storage consistent across Foundation Nodes."
          "foundation1": 16TB,
          ...
        }
    ...
  }
}
```


Related Links

Ocient System Bootstrapping

Ocient Application Configuration

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Node Bootstrapping Reference

Bootstrap a Node

1 Connect to the Node

To bootstrap an Ocient® node, connect to the node with your server username and password and the IP address of your node.

For example, connect as administrator `admin` to the IP Address `10.10.10.10`.

Shell

```
ssh admin@10.10.10.10
```

2 Configure the Bootstrap Configuration File

Use your preferred text editor with `sudo` to create the `/var/opt/ocient/bootstrap.conf` file as root.

For a list of bootstrapping options, see **Supported Options**.

Option 1: Bootstrap the initial node in a system

Use these `bootstrap.conf` settings to create the initial node in a system.

`/var/opt/ocient/bootstrap.conf` example

YAML

```
initialSystem: true
```

Option 2: Bootstrap a new node on a system with an initial node

© 2026 Ocient, Inc.

Use these `bootstrap.conf` settings to create another node in a system, replacing the IP Address `10.10.10.10` with the IP address or hostname of the initial node.

`/var/opt/ocient/bootstrap.conf` example

YAML

```yaml
adminHost: 10.10.10.10
```

3 Start the Node

Start the database.

Shell

```shell
sudo systemctl start rolehostd
```

Verify that the node and the service are active by running this status command.

Shell

```shell
systemctl status rolehostd

rolehostd.service - Rolehostd daemon startup
     Loaded: loaded (/etc/systemd/system/rolehostd.service; enabled; vendor p
     Active: active (running) since Wed 2022-01-26 23:31:36 UTC; 7s ago
     ...
```

You have bootstrapped the node.

Supported Options

You can edit the `/var/opt/ocient/bootstrap.conf` file to include a number of other options for advanced system configurations.

The Ocient `bootstrap.conf` file can also include these optional parameters. For examples of these options, see the **Example bootstrap.conf file**.

Option	Description
`name`	The default name of the node in the database is the hostname. If that name is not appropriate, you can assign a different name.
`nodeAddress`	The default address of this node is the hostname of the node, or the IP address if the `useDns` setting is `false`. If the system has multiple interfaces or you want an alternative setup, then you can specify the node address. This address is the address that all the other nodes in the cluster use to contact this node.

`externalAddress`	It might be desirable to have the SQL Nodes listen on a different external address than the system uses internally. The `externalAddress` parameter can either be a single address or a list of addresses.
	Example of single address:
	`externalAddress: 10.10.10.1`
	Example of a list of addresses:
	`externalAddress: ["10.10.10.1","20.20.20.2"]`
	iThis option applies to REST endpoints only. For REST endpoints, see **System Information REST Endpoints**.

`adminUserName`	The default user, specified as `admin@system`, when you create an Ocient System. If a different administrator user should be used to add this node to an existing system, use this option to indicate the user.
`adminPassword`	If the administrator password is not the default, then use this option to specify the password.
`logLevel`	The logging level for the system. Defaults to DEBUG.
`connectivityPools`	The configuration of SQL Nodes in a connectivity pool. You can use this configuration to configure the connectivity pool for the first SQL Node in a cluster. Other nodes cannot use this option. If there are no entries in the `bootstrap.conf` file, then the Ocient System adds the SQL Node to a new connectivity pool with a default name. Example of the connectivity pool named `test_pool` configuration: connectivityPools: test_pool: sourceAddress: 1.2.3.4/12 sourcePort: 1 # optional priority: 1 listenAddress: 111.1.1.1 listenPort: 4050 advertisedAddress: localhost advertisedPort: 4050 # optional
`systemStorageSpace`	The configuration for storage space has width and parity width for the Ocient System. If you do not configure the storage space, the system sets the width to 3 and parity to 2, the default values. For details about width and parity width, see **Core Elements of an Ocient System**. To set these options using SQL syntax, see **CREATE STORAGESPACE**.

Option	Description
`useOpal`	When you set this value to `false`, this option prevents the `rolehostd` service from taking ownership of and enabling Opal locking on all NVMe drives that support it. Bootstrap with this setting using an external Key Management System (KMS).
`pageReplication`	The number of page replicas the system writes before a page is available, unless you enable degraded writes. The fault tolerance is `(pageReplication - 1)`. The calculation for the default value is `min(width, parityWidth+1)`. For the system storage space, the default value is 3.

Example bootstrap.conf File [#example-bootstrap-conf-file]

This example shows a `bootstrap.conf` file that you can use as a template.

© 2026 Ocient, Inc.

YAML

```yaml
## initialSystem: true           # Uncomment for the first node in a
                                 # multi-cluster system.
                                 # Defaults to false. Cannot be
                                 # true if initialSingleClusterSystem
                                 # is also true.

## initialSingleClusterSystem: true  # Uncomment for the first node in a
                                 # single-cluster system.
                                 # Defaults to false. Cannot be
                                 # true if initialSystem is also true.

## systemName: productionSystem  # The name of the Ocient System. This
                                 # configuration applies only when the
                                 # initialSystem or initialSingleClusterSystem
                                 # configuration is set to true.

## adminHost: sql.example.com    # Hostname or IP address of an administrator
                                 # node, generally the first node. Not
                                 # required for the initial node, but required
                                 # for all other nodes.

## name: elmo                    # The name of this node.
                                 # Defaults to hostname.

## nodeAddress: sql1.example.com # The Hostname or IP address of this node.
                                 # Defaults to hostname, and if that does not
                                 # work, the bootstrapping process uses the IP
                                 # address of the system.

## externalAddress: ["10.10.10.1", "20.20.20.2"]

## useDns: false                 # Use DNS to determine the hostname of this
                                 # node. Defaults to true. When you set this
                                 # configuration to false, the bootstrapping
                                 # process uses the IP address of the system.

## adminUserName: bert@system    # A system administrator username.
                                 # Defaults to "admin@system".
## adminPassword: ernie          # The password for the administrator user.
                                 # Defaults to "admin".

## logLevel: info                # Log level for the system.
```

```
                                # Defaults to debug.

## connectivityPools:            # Define connectivity pool named test_pool.
##    test_pool:
##      sourceAddress: 1.2.3.4/12 # The source IP address, which is
                                # the IP address in CIDR notation of
                                # the client that connects to the SQL Node.

##     sourcePort: 1              # The optional source port number, which is the
                                # port number of the client. This parameter
                                # defaults to none, which means any port number.
#      priority: 1               # The priority of the connection. A higher number
                                # indicates a higher priority.
#      listenAddress: 111.1.1.1  # The IP address for listening.
#      listenPort: 4050             # The port number for listening.
#      advertisedAddress: localhost # The IP address to return to the client.
#      advertisedPort: 4050      # The optional port number to return to the client
                                # If you specify NULL for this parameter, then the
                                # database uses the value of the listen_port param

## systemStorageSpace:            # Define the system storage space.
##    width: 3                    # The width is the total number of segments to us
                                # each segment group.
##    parityWidth: 2              # The parity width is the number of parity coding
                                # to use for each segment group, or the fault tole
##    pageReplication: min(width, parityWidth+1)   # Override default page replicat
```

Related Links

Checking System Configuration Before Bootstrapping

Operating System Configuration

Cluster and Node Management

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Amazon Web Services Ocient Installation

This guide explains how to install an Ocient® System in Amazon® Web Services℠ (AWS℠).

For details about AWS concepts, see these pages:

- **What is Amazon EC2?**
- **What is IAM?**
- **A Primer for Amazon Networking - VPC, Availability Zones & Subnets**

Ocient supports deployment in AWS for pilot or testing purposes, but this setup does not guarantee data durability. Stopping Amazon® EC2℠ instances can result in permanent data loss.

The steps for deploying an Ocient System in AWS are:

1. Prepare AWS resources.
2. Set up an initial instance.
3. Create Amazon Machine Images (AMI) from the initial instance.
4. Launch other instances.
5. Follow the standard Ocient installation procedure.

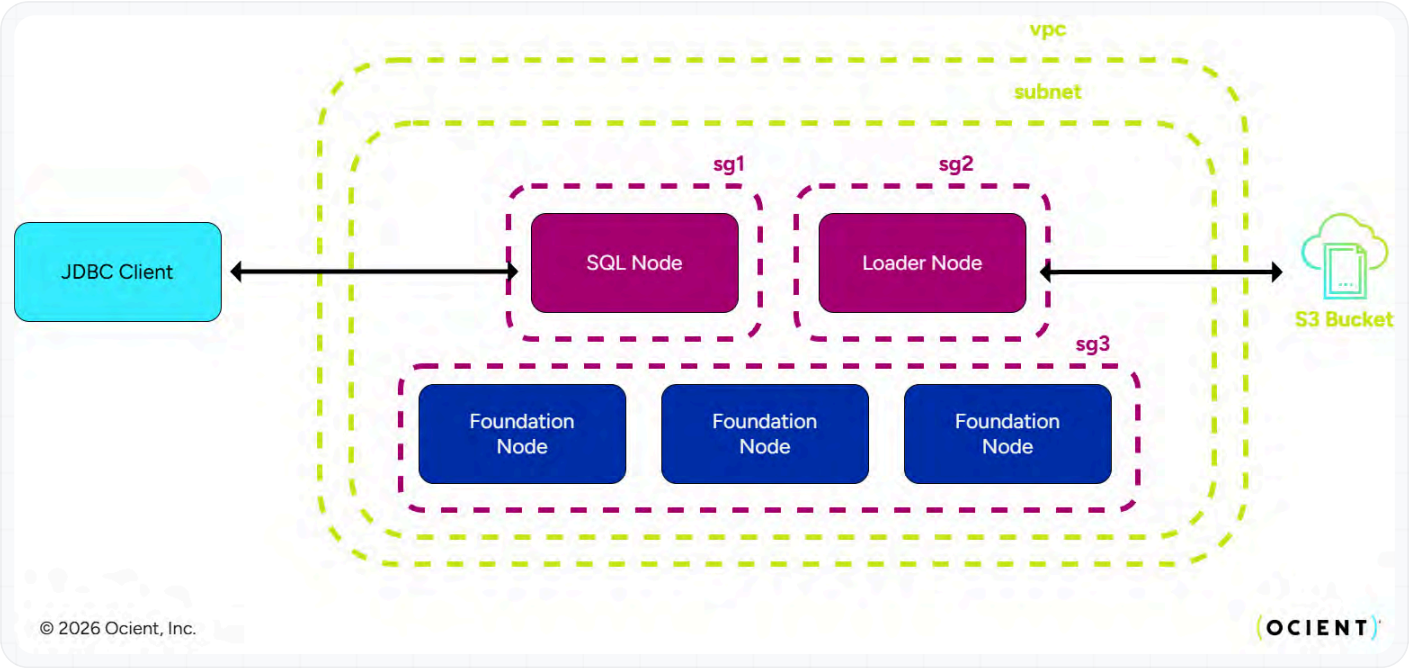
Example Configuration

The table below shows the recommended instance types for each node type.

Node Type	Instance Type
Foundation Nodes (3)	i3en.metal, i7ie.metal
Loader Nodes (1)	i3en.metal, i7ie.metal
SQL Nodes (1)	r5dn.metal, r6idn.metal

This diagram shows an example of an Ocient cluster in AWS. The EC2 nodes (SQL, Loader, and Foundation) are deployed within a single subnet of an Amazon Virtual Private Cloud℠ (VPC). AWS assigns each type of node to a separate security group (`sg1` , `sg2` , `sg3`).

© 2026 Ocient, Inc.



Prepare AWS Resources

Create and configure these AWS resources:

- The VPC and subnets for the Ocient System.
- Security groups to access the endpoints for each node type. For details about the network security configuration, see the **Ocient Security Guide**.
- Identity and Access Management (IAM) roles.

If you are loading data from S3, the loader nodes require IAM access to an S3 bucket.

Node Setup (SQL Role) for Initial Instance

1 Launch Instance

Use this configuration for your AMI. Configuration steps differ depending on whether your setup uses a single-volume or multi-volume AMI.

Operating System (OS)

To set up the AMI, you can use any Ocient-supported OS (see **Ocient System Requirements**).

Single-Volume AMI

If you use a single-volume AMI, specify this configuration:

- Increase the root volume to 128GB or more.

Multi-Volume AMI

If you use a multi-volume AMI (e.g., CIS hardened Red Hat® Enterprise Linux® (RHEL®) (RHEL) 9), use this configuration:

- Increase root volume to 30GB or more.
- Increase Elastic Block Store (EBS) volume to 100GB or more.

> This EBS volume supports key system directories in the image (`/home` , `/var` , `/var/log` , `/var/log/audit` , `/var/tmp`).

Instance Type

Use r5dn.metal or a similar instance type.

Security Group

Use one or more security groups with these rules:

- Allow SSH to the nodes.
- Allow communication internally between nodes.
- Allow access to SQL Node endpoints described in the **Ocient Security Guide**.

2 Access Shell

Connect to your instance using Secure Shell (SSH). For details, see **Connect to your Linux instance using an SSH client**.

3 Extend Logical Volume Manager (LVM) Volumes (Multi-Volume AMI Only)

> This step applies only to multi-volume AMIs. If you are using a single-volume instance, skip this step.

If you use a multi-volume manager, extend the `/home` and `/var` LVM volumes and their filesystems to fill up the expanded EBS volumes. These actions expand the LVM volume and the contained file system to accommodate the Ocient package, logging, and metadata.

These code examples show how to extend LVM volumes for a CIS RHEL 9 image.

Other AMI types might require different sizing. Contact Ocient support for the best sizing for your system for multi-volume instances.

Example

Resize the physical volumes of two drives to use their full capacity after expanding them (see Step 1).

Shell

```
sudo pvresize /dev/nvme0n1
sudo pvresize /dev/nvme1n1
```

Extend local volumes:

- Add 66 percent of the available free space in `vg-01` to the `var_vol` logical volume.
- Add all (100 percent) of the remaining free space in `vg-01` to the `home_vol` logical volume.

Shell

```
sudo lvextend -l +66%FREE /dev/vg-01/var_vol
sudo lvextend -l +100%FREE /dev/vg-01/home_vol
```

Extend the file system to use all available space on its underlying logical volume.

Shell

```
sudo xfs_growfs /home
sudo xfs_growfs /var
```

4 Update Packages

Update all your software packages to their latest versions and then reboot your instance.

For RHEL-compatible systems, use this command.

Shell

```
sudo dnf update
```

For Debian®-compatible systems, use this command.

```shell
sudo apt update
```

Reboot after the update.

```shell
sudo reboot
```

5 Install the Ocient Package

Copy over the `ocient` RPM or DEB package and install it.

For RHEL-compatible systems, use this command.

```shell
sudo dnf install ./ocient-RELEASE-XX.X.X-xxxxxxxxxxxxxx-x86_64.rpm
```

For Debian-compatible systems, use this command.

```shell
sudo apt install ./ocient-RELEASE-XX.X.X-xxxxxxxx.xxxxxx-xxxxxxxxxxxx-amd64.deb
```

6 Set Up Kernel Parameters

Use the `ockernelparams` utility to set up kernel parameters automatically, including the huge pages configuration. Repeat this step on other nodes.

```shell
sudo /opt/ocient/scripts/ockernelparams --node-role sql
```

Reboot the system for the parameters to take effect.

Shell

```
sudo reboot
```

Check that the local storage drive is attached to the `uio` or `vfio` driver after reboot (the Ocient package installs a service that runs on startup to do this).

Shell

```
sudo /opt/ocient/scripts/nvme-driver-util.sh
```

For examples of attaching drivers to the NVMe drives, see **NVMe Drive Firmware Upgrade Process**.

7 Configure the Firewall

If you are not using an OS-level firewall, skip this step.

If your base AMI includes a system firewall, you must configure rules that explicitly allow required network communication for your Ocient deployment. For details, see **Ocient Security Guide**.

Required OS firewall rules:

- Allow all necessary ports and protocols between Ocient nodes by:
  - Opening all TCP/UDP ports within the private network range (e.g., `10.0.0.0/16`).
  - Or, allowing known Ocient ports.
- Allow external access where needed by:
  - Enabling SSH access (port 22) from your administrator IP range.
  - Allowing client access to SQL endpoints (for example, port 13101 or as specified in your setup).
  - Opening any additional ports required for monitoring or management tools.

For a list of required ports, see **Network Exposure and Firewall**.

Create AMI

After your initial node is fully configured, you must replicate the setup process for the remaining nodes in your cluster. This action ensures consistency and allows for proper internal communication between nodes.

For details about creating an AMI, see **Creating an AMI from an Amazon EC2 Instance**.

Set Up Remaining Nodes to Launch the Remaining Instances

Go through this process for each of your remaining nodes.

1 Set Up Parameters

Launch the remaining instances with these parameters:

- AMI — Use the AMI created in the **Node Setup (SQL Role) for Initial Instance** step.
- Instance Type — Use `i3en.metal` or an equivalent instance that:
  - Offers local NVMe SSDs for high-performance local storage.
  - Has high throughput and network bandwidth for internal cluster communication.
- Security groups — Ensure these security rules are in place in the AWS security groups associated with the nodes:
  - Allow SSH to the nodes.
  - Allow internal communication between all Ocient nodes.
  - Allow access to endpoints described in the **Ocient Security Guide**.

2 Access Shell

Connect to your instance using Secure Shell (SSH). For details, see **Connect to your Linux instance using an SSH client**.

3 Set Up Kernel Parameters

Use the `ockernelparams` utility to set up kernel parameters automatically, including the `hugepages` parameters.

This example specifies a Foundation Node `foundation`. Use a different node type as necessary.

Shell

```
sudo /opt/ocient/scripts/ockernelparams --node-role foundation
```

Bootstrap the Ocient System

Complete the bootstrapping process for your Ocient System. For details, see **Node Bootstrapping Reference**.

Related Links

Load Data

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Google Cloud Platform Ocient Installation

This guide explains how to install an Ocient® System in Google® Cloud™ (GCP).

For details about GCP concepts, see these pages:

- **What is Compute Engine?**
- **What is IAM?**
- **VPC Network Overview**

> Ocient supports deployment in GCP for pilot or testing purposes, but this setup does not guarantee data durability. Stopping Compute Engine virtual machine (VM) instances can result in permanent data loss.

The steps for deploying an Ocient System in GCP are:

1. Prepare GCP resources.
2. Set up an initial instance.
3. Create machine images from the initial instance.
4. Launch other instances.
5. Follow the standard Ocient installation procedure.

Example Configuration

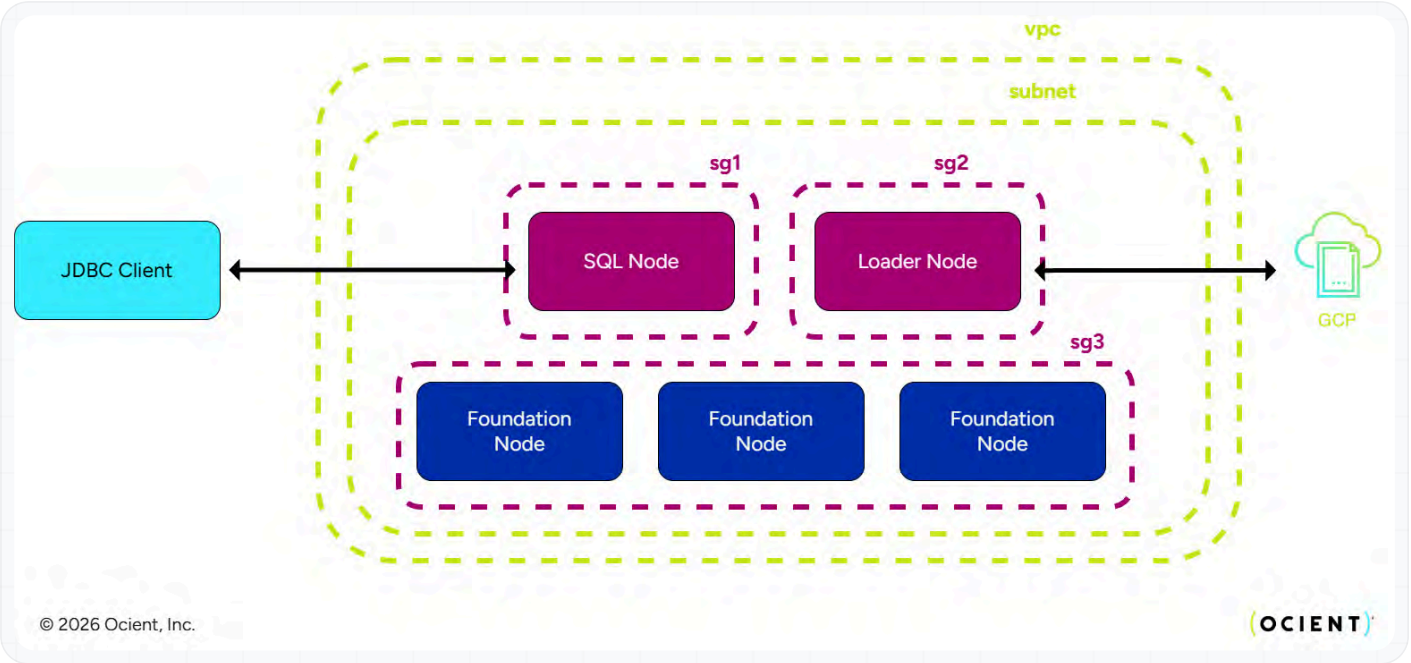
The table below shows the recommended VM types for each node type.

Node Type	VM Type
Foundation Nodes (3)	n2-highmem-96
Loader Nodes (1)	n2-highmem-96
SQL Nodes (1)	n2-highmem-96

Metadata and non-Ocient nodes use the n2-standard-X machine type, where X represents the number of Virtual Central Processing Units (vCPUs) for your workload.

This diagram shows an example of an Ocient cluster in GCP. The system deploys the Compute Engine VMs (SQL, Loader, and Foundation) within a single subnet of a Virtual Private Cloud (VPC). GCP uses VPC firewall rules and network tags to control traffic to each node type.

© 2026 Ocient, Inc.



Prepare GCP Resources

Create and configure these GCP resources:

- The VPC network and subnets for the Ocient System. For details, see **Create and manage VPC networks**.
- VPC firewall rules with network tags to control access to the endpoints for each node type. For details about firewall configuration, see **Create firewall rules** and **Add and remove network tags**. For details about the network security configuration, see the **Ocient Security Guide**.
- Identity and Access Management (IAM) roles. For details, see **Grant an IAM role**.

If you are loading data from Cloud Storage, the Loader Nodes require IAM access to a Cloud Storage bucket. Assign the `roles/storage.objectViewer` role (or `roles/storage.objectAdmin` if you also need write access) to the service account used by the Loader Node VMs.

Node Setup (SQL Role) for the Initial Instance

1 Launch the Instance

Use this configuration for your machine image. Configuration steps differ depending on whether your setup uses a single-volume or multi-volume machine image. For details about creating instances, see **Create and start a Compute Engine instance**.

Operating System (OS)

To set up the machine image, you can use any Ocient-supported OS (see **Ocient System Requirements**).

Single-Volume Machine Image

If you use a single-volume machine image, specify this configuration:

- Increase the boot disk to 128GB or more.

Multi-Volume Machine Image

If you use a multi-volume machine image (e.g., CIS hardened Red Hat® Enterprise Linux® (RHEL®) (RHEL) 9), use this configuration:

- Increase the boot disk to 30GB or more.
- Increase the additional Persistent Disk volume to 100GB or more. For details, see **Add a Persistent Disk to your VM**.

> This Persistent Disk volume supports key system directories in the image (`/home` , `/var` , `/var/log` , `/var/log/audit` , `/var/tmp`).

VM Type

Use n2-highmem-96 for Ocient nodes. Use n2-standard-X for Metadata and non-Ocient nodes.

Secure Boot

Disable Shielded VM secure boot when creating the instance. Use the `--no-shielded-secure-boot` flag with the `gcloud` command.

Shell

```
gcloud compute instances create INSTANCE_NAME \
    --no-shielded-secure-boot
```

If you are using the Console, uncheck **Turn on Secure Boot** under **Security > Shielded VM**. For details, see **Shielded VM**.

Firewall Rules and Network Tags

Create VPC firewall rules with network tags that enforce these rules:

- Allow SSH to the nodes.
- Allow communication internally between nodes.
- Allow access to SQL Node endpoints. For details, see the **Ocient Security Guide**.

Assign the appropriate network tags to each VM so that the firewall rules apply to the correct node types. For details, see **Create firewall rules** and **Add and remove network tags**.

2 Access the Shell

Connect to your instance using Secure Shell (SSH). For details, see **Connect to Linux VMs using Google tools**.

3 Extend Logical Volume Manager (LVM) Volumes (Multi-Volume Machine Image Only)

> This step applies only to multi-volume machine images. If you are using a single-volume instance, skip this step.

If you use a multi-volume manager, extend the `/home` and `/var` LVM volumes and their filesystems to fill up the expanded Persistent Disk volumes. These actions expand the LVM volume and the contained file system to accommodate the package, logging, and metadata of the Ocient System.

These code examples show how to extend LVM volumes for a RHEL 9 image.

> Other machine image types might require different sizing. Contact Ocient support for the best sizing for your system for multi-volume instances.

Examples

Resize the physical volumes of two drives to use their full capacity after expanding them (see **Prepare GCP Resources**).

Shell

```
sudo pvresize /dev/nvme0n1
sudo pvresize /dev/nvme1n1
```

Extend local volumes:

- Add 66 percent of the available free space in `vg-01` to the `var_vol` logical volume.

- Add all (100 percent) of the remaining free space in `vg-01` to the `home_vol` logical volume.

Shell

```
sudo lvextend -l +66%FREE /dev/vg-01/var_vol
sudo lvextend -l +100%FREE /dev/vg-01/home_vol
```

Extend the file system to use all available space on its underlying logical volume.

Shell

```
sudo xfs_growfs /home
sudo xfs_growfs /var
```

4 Update Packages

Update all your software packages to their latest versions and then reboot your instance.

For RHEL-compatible systems, use this command.

Shell

```
sudo dnf update
```

For Debian®-compatible systems, use this command.

Shell

```
sudo apt update
```

Reboot after the update.

Shell

```
sudo reboot
```

5 Install GCP Kernel Modules

GCP Compute Engine VMs require additional kernel modules for full hardware support. Install the extra kernel modules package that matches your running kernel version.

Shell

```
sudo apt install linux-modules-extra-$(uname -r)
```

Reboot the node after the installation completes.

Shell

```
sudo reboot
```

> This step is required for GCP Compute Engine VMs. Without the extra kernel modules, certain hardware drivers and features might not function correctly.

6 Install the Ocient Package

Copy over the `ocient` RPM or DEB package and install it.

For RHEL-compatible systems, use this command.

Shell

```
sudo dnf install ./ocient-RELEASE-XX.X.X-xxxxxxxxxxxxxx-x86_64.rpm
```

For Debian-compatible systems, use this command.

Shell

```
sudo apt install ./ocient-RELEASE-XX.X.X-xxxxxxxx.xxxxxx-xxxxxxxxxxxx-amd64.deb
```

7 Set Up Kernel Parameters

Use the `ockernelparams` utility to set up kernel parameters automatically, including the huge pages configuration. Repeat this step on other nodes.

Shell

```
sudo /opt/ocient/scripts/ockernelparams --node-role sql
```

Reboot the system for the parameters to take effect.

```shell
sudo reboot
```

Check that the local storage drive is attached to the `uio` or `vfio` driver after reboot (the Ocient package installs a service that runs on startup to perform this action).

```shell
sudo /opt/ocient/scripts/nvme-driver-util.sh
```

On GCP, drives bound to the `uio_pci_generic` driver display without a device name. This behavior is expected for GCP NVMe drives.

```
NVMe device status

BDF             Numa Node       Driver name             Device name
0000:00:04.0    -1              uio_pci_generic         -
```

For examples of attaching drivers to the NVMe drives, see **NVMe Drive Firmware Upgrade Process**.

8 Configure the Firewall

> If you are not using an OS-level firewall, skip this step.

If your base machine image includes a system firewall, you must configure rules that explicitly allow required network communication for your Ocient deployment. For details, see the **Ocient Security Guide**.

Required OS firewall rules:

- Allow all necessary ports and protocols between Ocient nodes by:
  - Opening all TCP/UDP ports within the private network range (e.g., `10.0.0.0/16`), or allowing known Ocient ports.
- Allow external access where needed by:
  - Enabling SSH access (port `22`) from your administrator IP range.

- Allowing client access to SQL endpoints (for example, port `13101` or as specified in your setup).
- Opening any additional ports required for monitoring or management tools.

For a list of required ports, see **Network Exposure and Firewall**.

> In addition to OS-level firewall rules, ensure that your GCP VPC firewall rules and network tags are properly configured (see **Prepare GCP Resources**). Both layers must allow the required traffic for the Ocient System to function correctly.

Create the Machine Image

After your initial node is fully configured, you must replicate the setup process for the remaining nodes in your cluster. This action ensures consistency and allows for proper internal communication between nodes.

For details about creating a machine image, see **Create a machine image**.

Set Up Remaining Nodes to Launch the Remaining Instances

Go through this process for each of your remaining nodes.

1 Set Up Parameters

Launch the remaining instances with these parameters. For details, see **Create an instance from a machine image**.

- Machine image — Use the machine image created in the **Node Setup (SQL Role) for Initial Instance** step.
- VM type — Use `n2-highmem-96` for Ocient nodes. This VM type:
  - Provides sufficient memory and compute for high-performance Ocient workloads.
  - Has high throughput and network bandwidth for internal cluster communication.
- Secure boot — Disable secure boot using the `--no-shielded-secure-boot` flag.
- VPC firewall rules — Ensure these rules are in place using network tags associated with the nodes:
  - Allow SSH to the nodes.
  - Allow internal communication between all Ocient nodes.

- Allow access to endpoints described in the **Ocient Security Guide**.

2 **Access the Shell**

Connect to your instance using SSH. For details, see **Connect to Linux VMs using Google tools**.

3 **Set Up Kernel Parameters**

Use the `ockernelparams` utility to automatically set up kernel parameters, including the `hugepages` parameters.

This example specifies a Foundation Node `foundation`. Use a different node type as necessary.

Shell

```
sudo /opt/ocient/scripts/ockernelparams --node-role foundation
```

Reboot the system for the parameters to take effect.

Shell

```
sudo reboot
```

Bootstrap the Ocient System

Complete the bootstrapping process for your Ocient System. For details, see **Node Bootstrapping Reference**.

Related Links

Load Data

Query Ocient

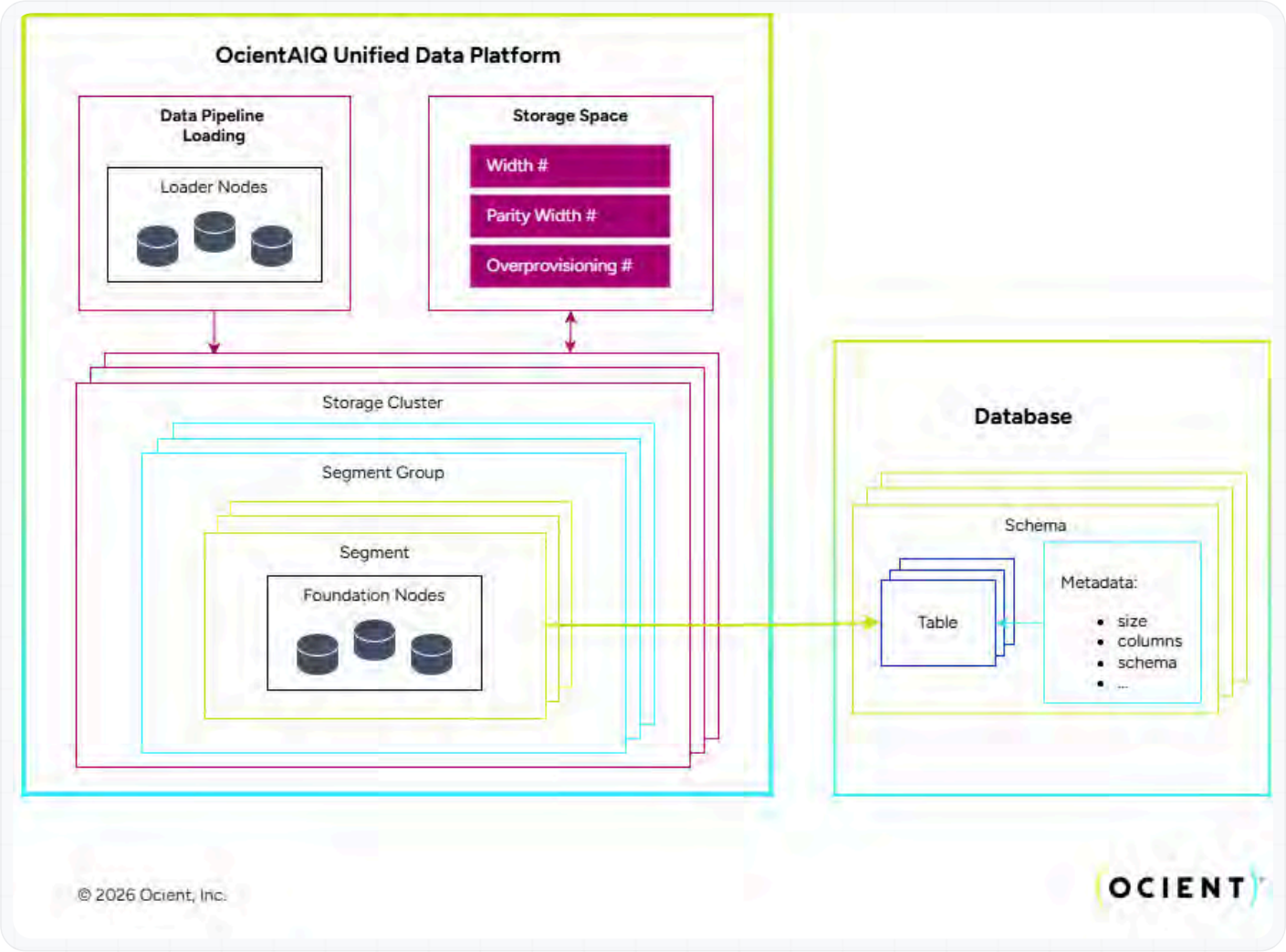
© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Configure Storage Spaces

When setting up an Ocient® System, a storage cluster represents a set of Foundation Nodes in the system. A storage space is a set of parameters within a storage cluster that defines how the system administers data and fault tolerance.

Creating a storage space is an important step for launching an Ocient System that must be done after installing hardware and bootstrapping nodes, but before you start loading and querying data. To see how creating a storage space fits into the Ocient installation process, see **Ocient Application Configuration**.

This tutorial explains how to set up and configure storage spaces to meet the needs of your Ocient System.



When an Ocient System starts, the system automatically creates a default system storage space for persistent metadata, such as system catalog tables. This default storage space is immutable, meaning you cannot delete or modify it.

The name of the metadata storage space is `systemStorageSpace` in the **sys.storage_spaces** system catalog table.

User-Defined Storage Spaces

For storage other than metadata, you must create one or more separate storage spaces. The creation of storage spaces is necessary before loading data or performing other operations. User-defined storage spaces are configurable for extra resiliency or storage based on your system needs.

A storage space represents how the storage cluster spreads data across Foundation Nodes to balance storage and fault tolerance. In general terms, the storage cluster configuration defines how the storage cluster behaves for these attributes:

- Regular data storage operations (see **Segment Group Width**)
- Query resiliency (see **Parity Width**)
- Load resiliency (see **Overprovision**)

You can configure a storage space to assign nodes to these job roles by using DDL statements (see **CREATE STORAGESPACE**).

Planning out your storage space carefully is important because you cannot change a storage space configuration after creation.

Segment Group Width

Storage space parameter: `WIDTH`

The Segment Group Width determines the number of segments that comprise a segment group. Functionally, this setting defines how many Foundation Nodes perform read and write operations for a segment group.

Within a Segment Group, the system assigns each segment to a different node. Hence, the Segment Group Width cannot exceed the number of nodes in your system.

In most circumstances, Segment Group Width should comprise most of your Foundation Nodes. On system startup, the default Segment Group Width is three, which is the minimum number of nodes required for an Ocient System.

Parity Width

Storage space parameter: `PARITY_WIDTH`

You can assign a subset of the Segment Group Width to Parity Width.

The Parity Width of a storage space determines its fault tolerance, defining the number of parity coding bits to use for each segment group. This number determines how many nodes can fail before the cluster is disabled.

Parity Width protects system querying capabilities. If any nodes fail, the system can execute and complete queries as long as the number of failed nodes is less than or equal to the Parity Width. When deployed in conjunction with overprovisioned nodes, Parity Width also provides fault tolerance for loading operations.

The `PARITY WIDTH` requires additional storage overhead, which is calculated by the formula `(PARITY WIDTH) / (SEGMENT GROUP WIDTH - PARITY WIDTH)`.

Overprovision

The Ocient System overprovisions any Foundation Nodes in the cluster in excess of the Segment Group Width by default. These nodes can include any nodes not configured in the storage space or any nodes added later.

Overprovisioned nodes protect loading operations from node failure, although they must be used in conjunction with Parity Width. In the event of node failures, loading operations can continue as long as the number of failed nodes does not exceed the Parity Width nodes or the number of overprovisioned nodes. For this reason, providing fault tolerance for loading requires a balance of Parity Width and overprovisioned nodes.

Unlike Segment Group Width and Parity Width, you do not explicitly define a value for overprovisioning. Instead, any Foundation Nodes in the cluster in excess of the number allocated toward Segment Group Width or Parity Width become overprovisioned by default.

Storage Space Configuration Examples

This section demonstrates different configurations for storage spaces. The examples each go through different scenarios for node failures to show the fault tolerance of each setup.

For information on the syntax for storage spaces, see **CREATE STORAGESPACE**.

10-Node Cluster

This example assumes you have a storage cluster of 10 Foundation Nodes.

```sql
CREATE STORAGESPACE ocient WIDTH = 10, PARITY_WIDTH = 2;
```

- The `WIDTH = 10` parameter means the system stores a segment group on 10 different nodes.
- The `PARITY_WIDTH = 2` parameter means each segment in the group contains enough parity bits to restore lost data for up to two nodes. In effect, this means parity bits comprise 20 percent of storage.

Tolerance Scenarios

This storage space setup has no overprovisioning, making loading operations less resilient. This setup results in these outcomes if nodes become disabled:

- If one node fails, loading fails, and querying can continue.
- If two nodes fail, loading fails, and querying can continue.
- If three nodes fail, loading and querying both fail.

12-Node Cluster

This example assumes a storage cluster of 12 Foundation Nodes.

```sql
CREATE STORAGESPACE ocient WIDTH = 10, PARITY_WIDTH = 2;
```

Note that this DDL statement is the same as the **10-Node Cluster Example**, but the cluster in this example has two additional nodes.

- The `WIDTH = 10` parameter means the system stores a segment group on 10 of the 12 nodes.
- The two remaining nodes are overprovisioned.
- The `PARITY_WIDTH = 2` parameter means each segment in the group contains enough parity bits to restore lost data for up to two nodes. In effect, this means parity bits comprise 16.6 percent of storage.

Tolerance Scenarios

This setup provides a level of resiliency for both querying and loading. The setup results in these outcomes if nodes become disabled:

- If one node fails, loading and querying continues.
- If two nodes fail, loading and querying continues.
- If three nodes fail, loading and querying both fail.

12-Node Cluster with More Parity Width

This example assumes a storage cluster of 12 Foundation Nodes with more nodes allocated to parity.

SQL

```
CREATE STORAGESPACE ocient WIDTH = 10, PARITY_WIDTH = 3;
```

- The `WIDTH = 10` parameter means the system stores a segment group on 10 of the 12 nodes.
- The two remaining nodes are overprovisioned.
- The `PARITY_WIDTH = 3` parameter means each segment in the group contains enough parity bits to restore lost data for up to three nodes. In effect, this means parity bits comprise 42 percent of storage.

Tolerance Scenarios

This setup provides resiliency for loading and especially for querying. The setup results in these outcomes if nodes become disabled:

- If one node fails, both loading and querying continue.
- If two nodes fail, both loading and querying continue.
- If three nodes fail, loading fails, and querying continues.
- If four nodes fail, both loading and querying fail.

15-Node Cluster

This example assumes a cluster of 15 Foundation Nodes. The configuration in this setup includes five overprovisioned nodes, exceeding the Parity Width nodes. This number of overprovisioned nodes would be inefficient and unable to provide the full benefit in a real system, but this example demonstrates what happens if you added extra nodes at a later time to a cluster, which the storage space would recognize as overprovisioned.

SQL

```
CREATE STORAGESPACE ocient WIDTH = 10, PARITY_WIDTH = 3;
```

- The `WIDTH = 10` parameter means the system stores a segment group on 10 of the 15 nodes.
- The five remaining nodes are overprovisioned.
- The `PARITY_WIDTH = 3` parameter means each segment in the group contains enough parity bits to restore lost data for up to three nodes. In effect, this means parity bits comprise 20 percent of storage.

Tolerance Scenarios

This setup provides resiliency for both querying and loading, but having more than three overprovisioned nodes provides no benefit because it exceeds three Parity Width nodes.

This setup results in these outcomes if nodes become disabled:

- If one node fails, loading and querying continue.
- If two nodes fail, loading and querying continue.
- If three nodes fail, loading and querying continue.
- If four nodes fail, loading and querying both fail because this breaches the Parity Width tolerance.

Related Links

Core Elements of an Ocient System

Cluster and Node Management

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Software Upgrade

To upgrade the Ocient® software, you must upgrade the DEB or RPM package on each node in the cluster. Versions of the Ocient software must match across the cluster, so upgrades must be done in an all-or-nothing fashion. Contact Ocient Support for assistance with the upgrade process.

ⓘ The Ocient System does not support the downgrade of the software.

⚠ Consult the release notes for the target version. The version might have changes to any of the supported operating systems, the required operating system configuration, or the `rolehostd` configuration.

Ocient System Prerequisites

- An existing Ocient installation is present on each node in the cluster.
- The RPM or DEB file is locally available on the file system.
- No jobs are running on the Ocient cluster.
- All nodes have stopped the `rolehostd` service. For details about stopping the `rolehostd` service, see the **Startup and Shutdown Procedure**.

RPM Upgrade

⚠ Consult the documentation of the package manager of your operating system for specific options and command-line syntax.

Use this command to upgrade using an Ocient RPM package from the local file system. Include `./` or specify the full path to the RPM file, otherwise the `dnf` command attempts to search the configured repositories instead of installing the local RPM file. In this example, replace `XX` with the Ocient package version of the provided file.

© 2026 Ocient, Inc.

Shell

```
sudo dnf install ./ocient-XX-x86_64.rpm
```

After the upgrade is complete, you must run the startup procedures on all nodes. For details, see **Startup and Shutdown Procedure**.

DEB Upgrade

Use this command to upgrade using an Ocient DEB package from the local file system. Include `./` or specify the full path to the DEB file, otherwise the `apt` command attempts to search the configured repositories instead of installing the local DEB file. In this example, replace `XX` with the Ocient package version of the provided file.

Shell

```
sudo apt install ./ocient-XX-amd64.deb
```

After the upgrade is complete, you must run the startup procedures on all nodes. For details, see the **Startup and Shutdown Procedure**.

Related Links

Maintenance Overview

OcientAIQ Unified Data Platform Release Notes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Maintenance Overview

Ocient® System Administrators need to perform maintenance for three primary reasons:

- When the Ocient software requires action, such as rebuilding lost or damaged segments
- When firmware or OS software updates are required
- When drive, memory, or system hardware failures occur

The maintenance documentation begins with Ocient software-related maintenance and proceeds to hardware-related maintenance.

- **Guide to Rebuilding Segments**
- **Startup and Shutdown Procedure**
- **Power Safety Recommendations**
- **NVMe Drive Firmware Upgrade Process**
- **Replace an NVMe Node Drive**
- **Expand and Rebalance System**
- **Replace Nodes**
  - **SQL Node Replacement**
  - **Loader Node Replacement**
  - **Foundation Node Replacement**
  - **Metadata Node Replacement**
- **Backup and Restore**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Guide to Rebuilding Segments

Data in an Ocient® System is erasure coded to provide resilience to disk and Foundation Node failures. The resilience of a system depends on the parity width of the storage space, which represents the number of associated disks (or nodes) that can fail without interruption of service or data loss.

For details about storage spaces and parity width, see **Configure Storage Spaces**.

Data Segment Statuses

You can check the status of your system segment groups by querying the `sys.segment_groups` system catalog table. For details, see **System Catalog**.

This table describes the states for data segments.

Status	Description	Recovery Process
INTACT	The normal and operable status.	No recovery needed.
DAMAGED	The segment failed a checksum, meaning the segment data is corrupted and unusable.	If you have sufficient parity width, you can recover a damaged segment by invoking a rebuild segment task.
MISSING	The segment is on a node or disk that is currently offline. If the node or disk rejoins the cluster, it can transition to the INTACT status.	When a disk or Foundation Node is permanently removed, you can perform a rebuild task to recover the data. This requires sufficient parity width.
REBUILDING	The segment is in recovery after damage or missing segment data.	Recovery is already in progress.

Recovery Considerations

If a segment has the DAMAGED or MISSING status, queries can proceed by reconstructing the data on demand using the remaining erasure-coded data in the segment group. Having a non-INTACT status means that input or output (I/O) performance is significantly reduced.

© 2026 Ocient, Inc.

To restore full performance, you need to run a segment rebuild task. Segment rebuilding is not automatic. The system administrator must manually invoke it.

A segment rebuild fails, and data is lost completely if the number of segments with the DAMAGED status in a segment group exceeds the parity width of the storage space. To avoid data loss:

- Provision the parity width of the storage space at or above the number of expected concurrent node failures.
- Rebuild damaged or permanently missing segments as soon as possible.

Checking for Abnormal Segments

You can find any segments that need a rebuild by querying for segment groups with an abnormal status.

Examples

Finding Faulty Segment Groups

This example query finds any segment groups with the DAMAGED or MISSING status.

SQL

```sql
SELECT * FROM sys.segment_groups WHERE status IN ('DAMAGED', 'MISSING');
```

Output

Text

```
|         "id"         |              "cluster_id"              | "segment_type" |
|----------------------|----------------------------------------|----------------|-
| "720576045199095878" | "074c32ec-4f92-4718-ada1-5eb55eafdcb3" | TKT_SEGMENT    |
| "716072445571725367" | "074c32ec-4f92-4718-ada1-5eb55eafdcb3" | TKT_SEGMENT    |
```

Finding Clusters and Nodes with Faulty Segment Groups

Inspect the count of damaged groups by cluster and node.

SQL

```sql
SELECT
  c.name AS cluster_name,
  n.name AS node_name,
  g.status AS segment_group_status,
  seg.status AS segment_status,
  seg.kind,
  COUNT(*) AS segment_count
FROM
  sys.segment_groups g
LEFT JOIN
  sys.clusters c
  ON c.id = g.cluster_id
LEFT JOIN
  sys.stored_segments seg
  ON seg.segment_group_id = g.id
LEFT JOIN
  sys.nodes n
  ON n.id = seg.node_id
WHERE
  g.status <> 'INTACT'
  AND (seg.status <> 'INTACT' OR seg.status IS NULL)
  GROUP BY 1,2,3,4,5
  ORDER BY 1,2,3,4,5;
```

Output

Text

```
|   "cluster_name"   | "node_name" | "segment_group_status" | "segment_status" | "k
|--------------------|-------------|------------------------|------------------|---
| foundation_cluster | foundation0 | DAMAGED                |                  | VI
```

Starting a Segment Rebuild Task

A user with System Administrator privileges can start a segment rebuild using the `CREATE TASK TYPE REBUILD` SQL statement. You cannot cancel a segment rebuild task after it is started.

Most commonly, a rebuild task repairs all damaged or missing segments across the system.

Example

Create a rebuild task.

```
SQL
CREATE TASK TYPE REBUILD;
```

The system can continue to perform queries while rebuilding segments, but the process can impact I/O performance.

Advanced Rebuild Commands

Rebuild tasks can also execute on specific Foundation Nodes or clusters. For information on fine-tuning rebuild tasks, see **CREATE TASK**.

Checking Rebuild Task Status

Monitor the status of current and past segment rebuild tasks from the `sys.subtasks` system catalog table. For details, see **System Catalog**.

```
SQL
SELECT * FROM sys.subtasks WHERE task_type = 'rebuild';
```

This table describes the statuses for a rebuild task.

Status	Status Detail	Description	Next Steps
complete	no_work	The segment groups were already available and healthy. No rebuilding was needed.	None
complete	complete	Rebuild completed successfully.	None

Status	Status Detail	Description	Next Steps
running	rebuild_in_progress	Rebuild is in process.	You can monitor the progress of the rebuild task by checking the JSON dictionary in the `details` column of the `sys.subtasks` system catalog table.
failed	rebuild_not_possible	The number of missing or damaged segments exceeds the parity width, and the data cannot be recovered currently.	If missing segments are available on an offline drive, you can attempt another rebuild task when that drive is made available to the system. Otherwise, you cannot recover the data.
failed	rebuild_no_space	There is not enough space available to rebuild the segment.	You can complete the rebuild by truncating other data to free up space.
failed	failed_on_node	The cluster lost its connection to the node before the rebuild was completed.	This is a transient error. You can retry the rebuild task.
failed	error	An unexpected internal error occurred.	Review error message details using the `rolehostd` logs. For details, see **Log Monitoring**.

Related Links

CREATE TASK

Erasure Coding

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Startup and Shutdown Procedure

Ocient® operates as a distributed system, so properly shutting down the system and preventing sudden power loss to nodes is important to ensure data integrity is maintained.

Execution of this process requires System Administrator access to the database and `sudo` privilege on the node being started or shut down. This procedure is not applicable to fully managed Ocient Systems.

Startup Procedure

The startup procedure can be used for starting the `rolehostd` process on a running node as well as powering on nodes in an Ocient System. In most installations, the `rolehostd` process is configured to automatically start on system boot up.

The `rolehostd` process is the main executable for an Ocient System. During the startup of the `rolehostd` process, every non-Admininistrator Node reaches out to one of the Administrator Nodes. These nodes require a “quorum” for the Ocient System to startup and operate. To establish a quorum, a majority of Administrator Nodes must be actively running the `rolehostd` process. For example, when an Ocient System has three Administrator Nodes, at least two must be active to establish quorum and service requests from other nodes.

Nodes in an Ocient System should started in the following sequence for smoothest operation:

> Queries are not completed until Foundation Nodes are online.

1. Start the `rolehostd` process on all dedicated **Administrator Nodes** and **SQL Nodes** using the `systemctl start rolehostd` command.
2. Start the `rolehostd` process on all **Foundation Nodes** using the `systemctl start rolehostd` command.

© 2026 Ocient, Inc.

3. Start the `rolehostd` process on all **Loader Nodes** using the `systemctl start rolehostd` command.
4. After the `rolehostd` process has finished starting on all Loader Nodes, start the Loading and Transformation service using the `systemctl start lat` command.

Shutdown Procedure

The shutdown procedure can be used for stopping the `rolehostd` process as well as powering off nodes in the Ocient System.

Properly sequencing the shutdown of nodes in an Ocient System ensures a smooth power-down process and reduces the appearance of errors to users in loading and querying as the system goes offline. The `rolehostd` process must be running on nodes with the Administrator Role for Global Dictionary Compression (GDC) lookup to work during loading and querying of data. If these nodes are shut down first, this will cause GDC lookups to fail, resulting in failures for both loading and querying of data.

It is recommended to stagger powering on of nodes by few seconds to reduce the surge in power distribution circuits.

When you execute `systemctl kill -s SIGKILL rolehostd`, the Ocient System sends `SIGKILL` to the `rolehostd` process. The node stops immediately without waiting for any queries or loading that are active on the node to finish. Some potential errors might be visible to the user. To avoid the errors, use the *quiescing* process.

The Ocient System should be stopped in the following sequence:

1. Stop the Loading and Transformation service on all Loader Nodes using the `systemctl stop lat` command.
2. Stop the `rolehostd` process on all Loader Nodes the `systemctl kill -s SIGKILL rolehostd` command.
3. Stop the `rolehostd` process on all Foundation Nodes the `systemctl kill -s SIGKILL rolehostd` command. Any queries issued after this point will fail.
4. Stop the `rolehostd` process on all SQL Nodes using the `systemctl kill -s SIGKILL rolehostd` command.

© 2026 Ocient, Inc.

5. Stop the `rolehostd` process on all Administrator Nodes the `systemctl kill -s SIGKILL rolehostd` command.

Quiescing Nodes

Quiescing nodes can be used for gracefully stopping the `rolehostd` process without interrupting loading or running queries. This process is most useful for maintenance purposes or for node upgrades. *Quiescing* refers to the graceful shutdown of a node, whereas *quiesced* refers to a node that has completed quiescing and is now fully shut down.

After the node shuts down, quiescing nodes stop interacting with the rest of the system. Only quiesce one node at a time. When you quiesce multiple nodes at the same time, the quiesce process can freeze or lose forward progress for queries and loading. At startup, the node automatically rejoins the system and participates normally.

To issue a quiesce, execute `systemctl stop rolehostd`. This command sends the `SIGTERM` signal to the `rolehostd` process. The node begins the quiesce process, waits for all relevant queries or loading to finish, and shuts down. If a long-running query causes the quiesce to freeze, kill the query to finish the quiesce process.

During the quiesce process, a SQL Node continues to accept new connections from clients, such as JDBC or pyocient. However, when these clients execute queries, EXPLAIN statements, or DDL updates, the Ocient System redirects the clients to another online SQL Node, if one exists. A user might connect or run a query with the `force` option, which overrides the redirect behavior and forces the command to run on the SQL Node in the quiesce process. In this case, the Ocient System does not guarantee the successful completion of the query.

A Foundation Node in the quiesce process waits for any segment rebuilds that are in progress to finish. When you interrupt or fail the rebuild, the quiesce process finishes shortly.

When a Loader Node is in the middle of loading, and you initiate the quiesce process on this node, stop the Loading and Transform (LAT) service that is on the same node.

© 2026 Ocient, Inc.

Quiesce has a default time limit of 30 minutes, controlled by the `systemd` service. After a node quiesces for 30 minutes, the `systemd` service sends the `SIGKILL` signal to `rolehostd` and forcefully kills the node. To change this timeout, edit the `TimeoutStopSec` parameter in the `rolehostd` `systemd` service file.

Related Links

NVMe Drive Firmware Upgrade Process

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Power Safety Recommendations

Like any distributed data processing system, power safety in Ocient® is important to ensure data integrity. Data integrity in Ocient comprises newly loaded data that is in the process of being written to storage and metadata files that contain information about the state of the system.

Ensure that you have Power Loss Protection included in the NVMe drives. The Ocient System does provide some protection against power loss. So, follow these best practices for power management.

NVMe SSD Power Loss Protection (PLP)

The OcientAIQ™ Unified Data Platform uses NVMe Solid State Drives (SSD) to store data.

To improve input or output (I/O) performance, SSDs cache data in a volatile memory temporary buffer before it is flushed to non-volatile flash memory. During a normal shutdown, the host system notifies the SSD that the system is shutting down so it can flush data to non-volatile storage. However, unexpected power loss can result in loss or corruption of this data if this volatile memory is not preserved. To protect against this data loss, NVMe drives validated for use with Ocient Systems must include Power Loss Protection (PLP) mechanisms. PLP is standard in enterprise and datacenter-grade NVMe drives. They are typically implemented using super capacitors that can hold enough power to allow safe writing of cached data to non-volatile memory in the event of a power loss to ensure data integrity.

Ocient Distributed System Protections

While data is loaded, it temporarily resides in Memory (RAM) or Disk Caches. The data stored in either of these locations is safely persisted to disks during a planned shutdown of nodes. The operating system or software might not be able to persist this data to disk during an unplanned power loss or system crash, so Ocient is designed to handle this scenario gracefully.

© 2026 Ocient, Inc.

Ocient provides redundancy mechanisms using clustering to prevent metadata loss due to a sudden power outage on a single node. The Administrator cluster consists of multiple nodes performing the Administrator role that maintain system metadata. This prevents metadata loss when a single node is lost or damaged.

Foundation storage clusters use parity configuration to protect data in the event of sudden power loss to a Foundation Node. In such an event, inflight queries might fail but without any permanent loss of data. Dynamic rebuilding of missing data allows a configurable number of Foundation Nodes to lose power and Ocient can continue to serve queries. If power loss damages a Foundation Node or causes data corruption, the data on the damaged Foundation Node can be rebuilt with the information available in parity bits on other nodes to fully recover lost data.

This might not hold true in simultaneous power loss on more than one Administrator Node or simultaneous power loss on more Foundation Nodes than the number of parity bits in the storage cluster. Such coordinated power loss could leave an Ocient System in a state where Administrator metadata is corrupted or damaged data on Foundation Nodes cannot be fully rebuilt. To prevent simultaneous power loss, the following best practices are recommended.

Power Safety Best Practices

The risk of complete system loss can be greatly reduced by avoiding sudden loss of power to an Ocient System. Best practices to reduce this risk are:

- NVMe Drives must include PLP.
- Each Ocient node should be configured with redundant power supplies, where each power supply is serviced by independent electrical circuits.
- A rack or Data Center (DC) level Uninterruptible Power System (UPS) should be available to provide constant power to Ocient nodes during a sudden power outage.
- When power loss occurs, and the UPS system cannot provide power for an extended period of time, you must properly shut down Ocient nodes by using the **Startup and Shutdown Procedure**.
- Ocient Systems should be hosted in Tier III or Tier IV Data Centers. These Data Centers provide redundant power and cooling systems and guarantee minimal

downtime.

Related Links

Install an Ocient System

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

NVMe Drive Firmware Upgrade Process

The Ocient® software uses NVMe drives for storage because of their higher input or output (I/O) throughput than spinning media drives. These drives are referred to as data disks or data drives. When you run Ocient software, the Ocient System accesses these data drives using the User Mode I/O (UIO) driver during normal operation.

This procedure guides a System Administrator through binding the drives back to the NVMe driver and upgrading the drive firmware. The procedure includes steps for upgrading the firmware of only Intel® NVMe drives. For other drive models, follow a similar process using the firmware upgrade process from the manufacturer.

⚠ Stop the `rolehostd` process prior to working on the NVMe drives. Otherwise, when the `rolehostd` process is running, the process becomes unstable because it has open file handles on the drives.

Execution of this process requires the sudo privilege on the node where the drive firmware is updated.

Prerequisites

Perform the firmware upgrade on one node of any specified type (Foundation, Loader, SQL) at a time. You can execute the firmware upgrade in parallel on different types of nodes.

ⓘ Perform the upgrade during a maintenance window because issues with the upgrade might prevent the node from returning to service.

The prerequisites for this procedure are:

1. The following packages are installed on the node undergoing firmware upgrade:
   - `smartmontools`
   - `nvme-cli`
   - `intelmas`

© 2026 Ocient, Inc.

- `sedutil`

2. The target firmware version for the upgrade is known and documented.
3. If you upgrade the data drive firmware on a Loader Node, prior to beginning the procedure, take the node out of the loading path by updating the LAT configuration. Before you shut down the `rolehostd` process, drain the pages that are held by the node.

The following example illustrates the firmware update commands used for an Intel NVMe SSD. Please substitute the appropriate command for your drive manufacturer in each of the steps that use the `intelmas` command.

Firmware Upgrade Procedure

The following steps require `sudo` permissions on the node.

Prepare for the Firmware Upgrade

1. Log in to the node as an administrator user.

2. If you upgrade the drive firmware on a Foundation Node or Loader Node, save the status of the drives for later in the procedure. After you replace `<IP_ADDRESS_OF_THE_NODE>` with the IP address of your node, save the output of this command.

CURL

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/stats | jq -r '.[] | select (.nam
 "localStorageService.device.status" ) | "(.device) (.value)" '
```

Alternatively, you can run a query on the SQL Node to display the status of the individual storage device and firmware version. After you replace `<NODE_NAME>` with the name of your node, connect to any SQL Node on the system and run this query.

SQL

```sql
SELECT
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.firmware_version
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

Output

Text

```
serial_number     |pci_address |device_status |firmware_version
------------------+------------+--------------+----------------
CVMD6356009M800GGN|0000:3b:00.0|ACTIVE        |VDV10184
CVMD636400BZ800GGN|0000:3c:00.0|ACTIVE        |VDV10184
CVMD636400JA800GGN|0000:5e:00.0|ACTIVE        |VDV10184
CVMD6365003C800GGN|0000:5f:00.0|ACTIVE        |VDV10184
CVMD6365003P800GGN|0000:86:00.0|ACTIVE        |VDV10184
CVMD63650092800GGN|0000:87:00.0|ACTIVE        |VDV10184
PHLJ0431019J8P0HGN|0000:88:00.0|ACTIVE        |VDV10184
PHLJ106500018P0HGN|0000:89:00.0|ACTIVE        |VDV10170
```

3. If you upgrade the drive firmware on a SQL Node, save the output of the `df` command.

Stop the `rolehostd` Process and Attach the Data Drives

1. Stop the `rolehostd` process to ensure that the drive replacement does not impact system stability.

Shell

```
sudo systemctl stop rolehostd
```

2. Prevent the `rolehostd` process from starting after reboot.

Shell

```
sudo systemctl disable rolehostd
```

3. Confirm that the output of this command indicates that the `rolehostd` process is not running and that the process auto start on reboot setting is disabled.

Shell

```
sudo systemctl status rolehostd
```

When you upgrade the drive firmware, you can impact data reliability while the `rolehostd` process is running on the node.

4. Attach the data drives to the NVMe driver.

Shell

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-nvme
```

5. Ensure that all drives are not the operating system drive and are attached to the NVMe driver in the output of this command. Any drives that did not bind to the NVMe driver in the previous step do not upgrade, so you must investigate any failure to bind to the NVMe driver.

Shell

```
/opt/ocient/scripts/nvme-driver-util.sh
```

© 2026 Ocient, Inc.

INCORRECT: Eight data drives are attached to the UIO driver. This is the incorrect system state for performing a firmware upgrade.

Text

```
./opt/ocient/scripts/nvme-driver-util.sh
NVMe device status

use option 'bind-uio' to bind uio_pci_generic driver to Ocient Payload/unpartitic
use option 'bind-nvme' to bind nvme driver to drives

BDF             Numa Node       Driver name             Device name
0000:01:00.0    0               nvme                    nvme0n1
0000:3b:00.0    0               uio_pci_generic         -
0000:3c:00.0    0               uio_pci_generic         -
0000:5e:00.0    0               uio_pci_generic         -
0000:5f:00.0    0               uio_pci_generic         -
0000:86:00.0    1               uio_pci_generic         -
0000:87:00.0    1               uio_pci_generic         -
0000:88:00.0    1               uio_pci_generic         -
0000:89:00.0    1               uio_pci_generic         -
```

CORRECT: The data drives are attached to the NVMe driver. This is the correct system state for performing a firmware upgrade.

© 2026 Ocient, Inc.

Text

```
./opt/ocient/scripts/nvme-driver-util.sh
NVMe device status

use option 'bind-uio' to bind uio_pci_generic driver to Ocient Payload/unpartitic
use option 'bind-nvme' to bind nvme driver to drives

BDF             Numa Node       Driver name             Device name
0000:01:00.0    0               nvme                    nvme0n1
0000:3b:00.0    0               nvme                    nvme1n1
0000:3c:00.0    0               nvme                    nvme2n1
0000:5e:00.0    0               nvme                    nvme3n1
0000:5f:00.0    0               nvme                    nvme4n1
0000:86:00.0    1               nvme                    nvme5n1
0000:87:00.0    1               nvme                    nvme6n1
0000:88:00.0    1               nvme                    nvme7n1
0000:89:00.0    1               nvme                    nvme8n1
```

Perform Firmware Upgrade

1. Run this command and review the `FirmwareUpdateAvailable` field to identify the drives that need a firmware upgrade. Save the results of this step for comparison later in the process.

Example

```
sudo intelmas show  -d Index, SerialNumber, DeviceStatus, Capacity, FirmwareUpdat
```

Drive 8 needs the firmware update, but drive 7 does not need the update.

© 2026 Ocient, Inc.

Output

```
- 7 Intel SSD DC P4510 Series PHLJ0431019J8P0HGN -

Capacity : 8001.56 GB
DeviceStatus : Healthy
FirmwareUpdateAvailable : The selected drive contains current firmware as of this
Index : 7
SerialNumber : PHLJ0431019J8P0HGN

- 8 Intel SSD DC P4510 Series PHLJ106500018P0HGN -

Capacity : 8001.56 GB
DeviceStatus : Healthy
FirmwareUpdateAvailable : Firmware=VDV10170 Bootloader=VB1B0172
Index : 8
SerialNumber : PHLJ106500018P0HGN
```

2. Execute this command for every drive in the previous step that indicates an old firmware in the `FirmwareUpdateAvailable` field. Replace `<DRIVE_INDEX>` with the number shown in the `Index` field from the output of the previous command for all drives that require the update.

Shell

```
sudo intelmas load -intelssd <DRIVE_INDEX>
```

3. Ensure that the output shows `Firmware updated successfully.` This is the sample log output for the upgrade.

Text

```
- intelmas load -intelssd 8
WARNING! You have selected to update the drives firmware!
Proceed with the update? (Y|N): Y
Checking for firmware update...

- Intel SSD DC P4510 Series PHLJ106500018P0HGN -

Status : Firmware updated successfully. Please reboot the system.
```

4. After you upgrade all drives that require the firmware upgrade, reboot the node using the `reboot` command. After the node completes the reboot, log in to the node as an administrator.

5. Attach the data drives to the NVMe driver using this command.

Shell

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-nvme
```

6. Compare the output of this command with the output of the same command captured earlier in this process. Ensure that the reported drive status and capacity are the same as before the upgrade and that the firmware shown is current.

Shell

```
sudo intelmas show -d Index, SerialNumber, DeviceStatus, Capacity, FirmwareUpdate
```

7. Reattach the data drives to the UIO driver.

Shell

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-uio
```

Enable the `rolehostd` Process and Perform Final Checks

1. Enable the `rolehostd` process to start after reboot, and then start the process.

Shell

```
sudo systemctl enable rolehostd && sudo systemctl start rolehostd
```

2. Ensure that the `rolehostd` process is enabled and started.

Shell

```
sudo systemctl status rolehostd
```

3. If you upgrade the firmware of the drive on a Foundation Node or Loader Node, ensure that the output of this command matches the output of the same command captured earlier in this process.

CURL

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/stats | jq -r '.[] | select (.nam
```

Alternatively, you can run a query on the SQL Node to capture the individual device status and firmware version. Ensure that the drives and statuses in this query match the results from the query you executed earlier in this process. After you replace the node name `<NODE_NAME>` with the name of your node, connect to any SQL Node on the system and execute this query.

SQL

```sql
SELECT
  s.id as serial_number,
  s.pci_address,
  s.device_status,
  s.firmware_version
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

Output

Text

```
serial_number     |pci_address |device_status |firmware_version
------------------+------------+--------------+----------------
CVMD6356009M800GGN|0000:3b:00.0|ACTIVE        |VDV10184
CVMD636400BZ800GGN|0000:3c:00.0|ACTIVE        |VDV10184
CVMD636400JA800GGN|0000:5e:00.0|ACTIVE        |VDV10184
CVMD6365003C800GGN|0000:5f:00.0|ACTIVE        |VDV10184
CVMD6365003P800GGN|0000:86:00.0|ACTIVE        |VDV10184
CVMD636500928OOGGN|0000:87:00.0|ACTIVE        |VDV10184
PHLJ0431019J8POHGN|0000:88:00.0|ACTIVE        |VDV10184
PHLJ106500018POHGN|0000:89:00.0|ACTIVE        |VDV10184
```

4. If you upgrade the firmware of the drive on a SQL Node, ensure that the output of the `df` command shows the same mount points as those captured earlier in this process.

Related Links

System Catalog

© 2026 Ocient All Rights Reserved

Replace an NVMe Node Drive

Ocient® software uses NVMe drives for storage because of their higher input or output (I/O) throughput than spinning media drives. These drives are referred to as data disks or drives.

In Ocient, these drives are accessed through a User Mode I/O (UIO) driver, so they cannot be accessed using the common set of Linux® commands (e.g., df/du/ls/file/parted/etc).

> ⚠ Working on these drives requires the `rolehostd` process to be stopped on the node, because the process has open file handles on the drives and making changes to the drive can make the process unstable.

Execution of this process requires `sudo` privilege on the Foundation Node on which the drive is being replaced.

Detection and Alerting

There are two approaches for detecting issues with NVMe drives on a Foundation Node. The device status of all drives on a specific Foundation Node can be checked using a SQL query of the system catalog to ensure all are active. Secondly, the presence of NVMe drives on the PCI bus can be inspected. You can run each command on a Foundation Node to assess the health of the drives on that node. Depending on the failure mode of a device, either of these methods could reveal an issue with an NVMe drive.

> ⓘ In the case of an operating system drive failure, contact Ocient Support.

Check Drive Status Using System Catalog Tables

SQL queries on the `sys.storage_device_status` system catalog table provide a status of all payload drives. Connect to any SQL Node on the system and execute this query, replacing `<NODE_NAME>` with the name of your node.

© 2026 Ocient, Inc.

SQL

```sql
SELECT
  n.name AS node_name,
  s.node_id,
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.device_model
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

Output

Text

```
node_name |node_id                               |serial_number     |pci_address |dev
----------+--------------------------------------+------------------+------------+---
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6356009M800GGN|0000:3b:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD636400BZ800GGN|0000:3c:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD636400JA800GGN|0000:5e:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6365003C800GGN|0000:5f:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6365003P800GGN|0000:86:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD63650092800GGN|0000:87:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6365009H800GGN|0000:88:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6371005A800GGN|0000:89:00.0|FAI
```

The possible storage device statuses are `ACTIVE` or `FAILED`. In this sample output, the device in PCI Address `0000:89:00.0` needs to be replaced.

If an NVMe drive is completely unavailable, it might not appear in the `sys.storage_device_status` table. For example, if a drive is not seated properly or if the node has rebooted and the drive is undetectable by Ocient. In this case, see the **Alternative Methods to Identify Failed Drives** to identify the serial number and PCI address of the drive.

In the event that a drive has failed or is experiencing issues, errors are logged in `/var/log/messages` and `/var/opt/ocient/log/rolehostd.log*` files. Inspection of these log files can assist with root cause analysis of the drive failure.

Troubleshooting

1. To locate the failed drive in the chassis, use the query results from the **Check Drive Status via System Catalog Tables** query to identify the PCI Address of the failed drive.
2. Refer to the chassis diagram for your system to identify the slot of the failed drive based on the PCI address. Three common chassis and drive bay mappings are listed below:

Supermicro® **SYS-1029U-TN10RT Hot swap bays**

NVMe 3c:00.0	NVMe 5f:00.0	NVMe 61:00.0	NVMe 87:00.0	NVMe 89:00.0
NVMe 3b:00.0	NVMe 5e:00.0	NVMe 60:00.0	NVMe 86:00.0	NVMe 88:00.0

Recovery

Prerequisites

- An NVMe drive of the same model and the same or larger capacity as the failed drive.
- The drive firmware is upgraded to the latest version using the **NVMe Drive Firmware Upgrade Process**.

Replacement Procedure

1. Log in to the Foundation Node that needs a drive replacement as an administrator user and stop the `rolehostd` process on this node to ensure that the drive replacement does not impact the stability of the process:

```shell
sudo systemctl stop rolehostd
```

2. Ensure that the `rolehostd` process has stopped. The result of this command should indicate that the process is `Stopped`.

Shell

```
sudo systemctl status rolehostd
```

3. If using encrypted OPAL drives, back up the localKeyStore directory. This directory can be empty or absent on systems that do not have OPAL drives.

Shell

```
sudo cp -pR /var/opt/ocient/localKeyStore /root
```

4. Physically remove the failed drive from the host and execute `nvme-driver-util.sh` to ensure that the failed drive is not showing up in the output.

Shell

```
/opt/ocient/scripts/nvme-driver-util.sh
```

Output

Text

```
[admin@go-lts9 ~]$ /opt/ocient/scripts/nvme-driver-util.sh
NVMe device status

use option 'bind-uio' to bind uio_pci_generic driver to Ocient Payload/unpartitione
use option 'bind-nvme' to bind nvme driver to drives

BDF             Numa Node       Driver name             Device name
0000:3b:00.0    0               uio_pci_generic         -
0000:3c:00.0    0               uio_pci_generic         -
0000:5e:00.0    0               uio_pci_generic         -
0000:5f:00.0    0               uio_pci_generic         -
0000:86:00.0    1               uio_pci_generic         -
0000:87:00.0    1               uio_pci_generic         -
0000:88:00.0    1               uio_pci_generic         -
[admin@go-lts9 ~]$
```

5. Insert the replacement drive in the chassis and ensure that it shows up in the output of the `nvme-driver-util.sh` script.

Shell

```
/opt/ocient/scripts/nvme-driver-util.sh
```

Output

Text

```
[admin@go-lts9 ~]$ /opt/ocient/scripts/nvme-driver-util.sh
NVMe device status

use option 'bind-uio' to bind uio_pci_generic driver to Ocient Payload/unpartitione
use option 'bind-nvme' to bind nvme driver to drives

BDF             Numa Node       Driver name             Device name
0000:3b:00.0    0               uio_pci_generic         -
0000:3c:00.0    0               uio_pci_generic         -
0000:5e:00.0    0               uio_pci_generic         -
0000:5f:00.0    0               uio_pci_generic         -
0000:86:00.0    1               uio_pci_generic         -
0000:87:00.0    1               uio_pci_generic         -
0000:88:00.0    1               uio_pci_generic         -
0000:89:00.0    1               nvme                    -
[admin@go-lts9 ~]$
```

6. Run this command by replacing `X` and `Y` to match the device name using the replacement drive.

Text

```
sudo nvme format /dev/nvmeXnY format --ses=1
```

⚠ This command formats the specified drive and erases all data on the drive. If the OS drive is an NVMe drive, ensure that it is not formatted.

7. Execute the `nvme-driver-util.sh` script to bind drives to the UIO driver:

Shell

```
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-uio
```

8. Execute `nvme-driver-util.sh` script and confirm that the new drive is bound to `uio_pci_generic` driver.

Shell

```
/opt/ocient/scripts/nvme-driver-util.sh
```

Output

Text

```
[admin@go-lts9 ~]$ /opt/ocient/scripts/nvme-driver-util.sh
NVMe device status

use option 'bind-uio' to bind uio_pci_generic driver to Ocient Payload/unpartitione
use option 'bind-nvme' to bind nvme driver to drives

BDF             Numa Node       Driver name             Device name
0000:3b:00.0    0               uio_pci_generic         -
0000:3c:00.0    0               uio_pci_generic         -
0000:5e:00.0    0               uio_pci_generic         -
0000:5f:00.0    0               uio_pci_generic         -
0000:86:00.0    1               uio_pci_generic         -
0000:87:00.0    1               uio_pci_generic         -
0000:88:00.0    1               uio_pci_generic         -
0000:89:00.0    1               uio_pci_generic         -
[admin@go-lts9 ~]$
```

9. Start the `rolehostd` process.

Shell

```
sudo systemctl start rolehostd
```

10. Check the output of the `sys.storage_device_status` catalog table by connecting to a SQL Node and running by following query after replacing the `<NODE_NAME>` with the name of your node. Ensure that all drives appear on the node with status of `ACTIVE`.

SQL

```sql
SELECT
  n.name AS node_name,
  s.node_id,
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.model
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

Output

Text

```
node_name |node_id                                |serial_number     |pci_address |dev
----------+---------------------------------------+------------------+------------+---
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6356009M800GGN|0000:3b:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD636400BZ800GGN|0000:3c:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD636400JA800GGN|0000:5e:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6365003C800GGN|0000:5f:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6365003P800GGN|0000:86:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD636500928OOGGN|0000:87:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD6365009H800GGN|0000:88:00.0|ACT
lts2      |f4845f46-dd0a-48bd-a2ed-614d5029e5df|CVMD637100GA800GGN|0000:89:00.0|ACT
```

As shown in the sample output, the device replaced at PCI Address `0000:89:00.0` is now active. This indicates that it has been successfully replaced.

Next Steps: Rebuild Segments

If the failed drive had data on it prior to failure, a rebuild of data segments stored on the replaced drive is required after the drive is replaced. Refer to the **Guide to Rebuilding Segments** for instructions on how to rebuild the missing segments.

Alternative Methods to Identify Failed Drives

If the results of querying the `sys.storage_device_status` table do not indicate which drive has failed, alternative methods can be used to detect the failing drive. These methods require the command-line utility `jq`, but you can view the full results of the `curl` commands without it.

Option 1: Cross Reference API Results to Identify the PCI Address

You can cross-reference the result of two API commands to determine the PCI address of the failed drive that is not appearing. A drive that does not appear in the `stats` API or catalog table can be present in the `sysconfig` API output.

The output of the following API shows NVMe storage device statuses. Any device status other than 10 indicates a potential issue. Replace `<IP_ADDRESS_OF_THE_NODE>` with the IP Address of your node and inspect the results.

CURL

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/stats | jq -r '.[] |select (.name =
```

Output

Text

```
$ curl -s http://1.2.3.4:9090/v1/stats | jq -r '.[] |select (.name == "localStorage
BTLJ913605YR2P0BGN 10
BTLJ913608AY2P0BGN 10
BTLJ913608FR2P0BGN 10
BTLJ91360D6Q2P0BGN 10
BTLJ91360D6Y2P0BGN 10
BTLJ916306LS2P0BGN 10
BTLJ92020KP22P0BGN 10
BTLJ92030AJM2P0BGN 10
PHLJ9403002N2P0BGN 10
PHLJ941100RW2P0BGN 10
```

The possible device status values are listed in the table below:

Value	Device Status
0	INVALID_UNKNOWN
10	ACTIVE
20	UNINITIALIZED
61	CORRUPT
80	FAILED

Check Device Presence

The output of the following API will show the configured NVMe drives on the Foundation Node. Depending on the failure mode, the output of this command can show a reduced number of drives. For example, if the Foundation Node chassis has 12 drive bays populated and only 11 appear in this result, it would indicate that one drive is not visible to Ocient. Run the following command by replacing `<IP_ADDRESS_OF_THE_NODE>` with the IP Address of your node:

CURL

```
curl -s http://<IP_ADDRESS_OF_THE_NODE>:9090/v1/sysconfig | jq -r '.storage_devices
```

Output

Text

```
$ curl -s http://1.2.3.4:9090/v1/sysconfig | jq -r '.storage_devices[] | select ( .
BTLJ913608AY2P0BGN 0000:3b:00.0
BTLJ913605YR2P0BGN 0000:3c:00.0
BTLJ913608FR2P0BGN 0000:5e:00.0
PHLJ9403002N2P0BGN 0000:5f:00.0
BTLJ92020KP22P0BGN 0000:60:00.0
BTLJ92030AJM2P0BGN 0000:61:00.0
PHLJ941100RW2P0BGN 0000:86:00.0
BTLJ91360D6Y2P0BGN 0000:87:00.0
BTLJ916306LS2P0BGN 0000:88:00.0
BTLJ91360D6Q2P0BGN 0000:89:00.0
```

If the address of the failed drive cannot be determined in this way, proceed to Option 2 to identify the location of the drive.

Option 2: Inspect Drives Using Operating System Commands

If drives are not appearing in the Ocient outputs, they might be unavailable to the Ocient service. In this case, directly examine the drives on the system with standard disk utilities and operating system commands to find the serial numbers of drives that meet the following criteria:

1. The drive is not the operating system boot drive.
2. The drive is not listed in the `sys.storage_device_status` table.

Any drives that meet this criteria are not in use by Ocient and might need to be replaced or repaired. Identify the Serial Number and PCI Addresses of these drives in order to complete the replacement procedure.

Related Links

System Catalog

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Expand and Rebalance System

If your system needs more resources, you can expand it by installing additional nodes and drives. The steps for expanding a system depend on the type of node you are adding.

These sections explain the steps for adding each Ocient® node type. Foundation Nodes include additional steps for adding more storage drives and rebalancing the system.

> For information on how to replace nodes that have failed or are damaged, see **Replace Nodes**.
>
> For information on replacing drives that have failed or are damaged, see **Replace an NVMe Node Drive**.

Prerequisites

These prerequisites apply to adding any node type:

- The process requires either the System Administrator role or the granted UPDATE privilege on SYSTEM.
- Any new nodes should have the same OS kernel version as other nodes in the system.
- Any new nodes should have the same Ocient System version as the other nodes. For details, see **Ocient Application Installation**.
- Ensure that your system meets the requirements outlined in **Ocient System Bootstrapping**.
- Before adding nodes, stop any query or loading process running on your system.

Add Foundation Nodes

A core component of an Ocient System, the Foundation Nodes store user data in Ocient and perform the bulk of query processing. By adding more Foundation Nodes, a system

© 2026 Ocient, Inc.

can support extra storage capacity as well as better performance for query processing.

This process requires rebalancing the data to distribute the query processing across all available hardware, including newly installed disks or nodes.

System Considerations

- No individual cluster should have more than two petabytes of storage.
- Each cluster should have the same number of Foundation Nodes.

Tutorial

1. Create the `/var/opt/ocient/bootstrap.conf` file on the new node using this YAML example. This YAML file should follow a similar format and parameters as the `bootstrap.conf` files on your other nodes.

   Replace the `<FIRST_NODE_ADDRESS>` with the IP or Hostname of a node running the Administrator Role on your system. If you are not using DNS, use `nodeAddress` instead of `adminHost`.

   Use a user account with System Administrator privileges for your system for the `adminUserName` and `adminPassword`.

   YAML

   ```yaml
   adminHost: <FIRST_NODE_ADDRESS>
   adminUserName: my_admin
   adminPassword: example_password
   ```

2. Start the new node by using this command.

   Shell

   ```shell
   sudo systemctl start rolehostd
   ```

© 2026 Ocient, Inc.

This process takes about a minute to complete. This step accepts the node into the system, but it has not been assigned a role or a storage cluster.

3 To validate that the node is on the system, execute this query using the `sys.nodes` system catalog table.

SQL

```
SELECT name, status
FROM sys.nodes
ORDER BY 1;
```

Output

SQL

```
name                                status
-------------------------------------------------------------------
admin01                             ACCEPTED
admin02                             ACCEPTED
admin03                             ACCEPTED
loader01                            ACCEPTED
loader02                            ACCEPTED
loader03                            ACCEPTED
foundation01                        ACCEPTED
foundation02                        ACCEPTED
foundation03                        ACCEPTED
foundation04                        ACCEPTED
foundation05                        ACCEPTED
foundation06                        ACCEPTED
foundation07                        ACCEPTED
foundation08                        ACCEPTED
foundation09                        ACCEPTED
foundation10                        ACCEPTED
foundation11                        ACCEPTED
foundation12                        ACCEPTED
foundation_new                      ACCEPTED
sql01                               ACCEPTED
sql02                               ACCEPTED
```

© 2026 Ocient, Inc.

The output shows the new node, named `foundation_new` in this example, listed as `ACCEPTED` in the `status` column.

4 Execute this `ALTER CLUSTER` SQL statement to add the new node to a storage cluster. In this example, replace `storage_cluster_1` and `foundation_new` with your cluster name and node name, respectively.

SQL

```
ALTER CLUSTER "storage_cluster_1"
    ADD PARTICIPANTS "foundation_new";
```

This example adds only one new node. For information on adding multiple nodes, see **ALTER CLUSTER ADD PARTICIPANTS**.

© 2026 Ocient, Inc.

5 Restart the `rolehostd` process on all nodes in the system by running the following series of commands at the shell terminal.

First, stop the `rolehostd` process on all nodes.

Shell

```
sudo systemctl kill -s SIGKILL rolehostd
```

Confirm that the `rolehostd` process is no longer running.

Shell

```
sudo systemctl status rolehostd
```

Start the `rolehostd` process on the system again.

Shell

```
sudo systemctl start rolehostd
```

6 You can verify the new node is active by executing this SQL query after you connect to the database.

SQL

```
SELECT n.name,
       ns.operational_status
FROM sys.node_status ns
JOIN sys.nodes n ON ns.node_id = n.id
ORDER BY n.name;

name                                operational_status
-------------------------------------------------------------------------------
admin01                             Active
admin02                             Active
admin03                             Active
loader01                            Active
loader02                            Active
loader03                            Active
foundation01                        Active
foundation02                        Active
foundation03                        Active
foundation04                        Active
foundation05                        Active
foundation06                        Active
foundation07                        Active
foundation08                        Active
foundation09                        Active
foundation10                        Active
foundation11                        Active
foundation12                        Active
foundation_new                      Active
sql01                               Active
sql02                               Active
```

7 Perform a rebalance task to distribute your data across your newly expanded system evenly. For details, see **Rebalance System**.

Add More Drives to Foundation Nodes

Foundation Nodes can support extra storage by adding additional NVMe drives. This tutorial shows the steps to integrate new drives into an existing Ocient System.

System Considerations

- For best performance, all Foundation Nodes should have equal storage capacity.
- No individual cluster should have more than two petabytes of storage.

Prerequisites

- The process requires `systemctl` access on your system OS.
- Any new NVMe drives added to the system must be blank and unpartitioned.

Tutorial

1. Shut down the `rolehostd` process from the shell prompt.

   Shell

   ```
   sudo systemctl stop rolehostd
   ```

2. Install the new storage drives in your system.

3. Restart the node from the shell prompt.

   Shell

   ```
   sudo systemctl restart rolehostd
   ```

   The `rolehostd` process recognizes the new drive.

4. To confirm the new drives are active and running, you can connect to your system and query the **sys.storage_device_status** system catalog table.

To use this example query, replace `<NODE_NAME>` with the name of the node where you are adding a drive.

SQL

```
SELECT
  n.name AS node_name,
  s.node_id,
  s.id AS serial_number,
  s.pci_address,
  s.device_status,
  s.device_model
FROM
  sys.nodes n
JOIN sys.storage_device_status s
  ON n.id = s.node_id
WHERE
  n.name = '<NODE_NAME>';
```

Output

Text

```
|node_name  |node_id                               |serial_number
|-----------|--------------------------------------|------------------------
|foundation0|9330a0b3-b3b7-4503-949c-043b196c0cc4|6156962f-fcf6-4299-bbd7-2
|foundation0|9330a0b3-b3b7-4503-949c-043b196c0cc4|e6a4b24f-0d49-4704-b7ce-1
```

This output lists the serial numbers of all drives in the `foundation0` node, including their statuses.

5 Perform a rebalance task to distribute your data across your newly expanded system evenly. For details, see **Rebalance System**.

Rebalance System

`REBALANCE` task execution redistributes your data evenly across your segment groups and clusters. Rebalance your system if you have recently installed new hardware, particularly new nodes or drives.

System Considerations

Currently, the `REBALANCE` task does not move damaged segment groups. If you suspect that you might have a significant number of damaged segment groups, you can execute these steps to check and fix the groups:

1. Ensure all nodes are online.
2. Check whether you have damaged segment groups using the `sys.segment_groups` system catalog table. If damaged segment groups are present, execute a `REBUILD` task to rebuild the groups before rebalancing the system. For details, see **Guide to Rebuilding Segments**.
3. Execute the `REBALANCE` task using this tutorial.

You can find information in the `sys.degraded_segment_groups` system catalog table to identify the damaged segment groups that need fixing. This table shows all segment groups with the `DAMAGED` or `UNAVAILABLE` state. You can also check the `sys.storage_used` system catalog table, which shows approximately the same number of used bytes `used_bytes` for each node entry with the same table after the rebalance execution.

Prerequisites

- Only one rebalance task can execute at a time on the system. The system logs an error if you try to start a second rebalance task.
- You must have the System Administrator role, or be granted the UPDATE privilege on SYSTEM.

Tutorial

1. Execute a `SELECT` SQL statement to view how the system has distributed the existing data using the `sys.tables` and `sys.nodes` system catalog tables.

SQL

```
SELECT t.name AS table_name,
    n.name AS node_name,
    su.used_bytes FROM sys.storage_used su
    JOIN sys.tables t ON t.id = su.table_id
    JOIN sys.nodes n ON n.id = su.node_id
ORDER BY t.name,
    n.name;
```

Output

Text

```
+-------------------------+-----------+--------------+
| table_name              | node_name |   used_bytes |
|-------------------------+-----------+--------------|
| table0                  | lts0      |     48000000 |
| table0                  | lts1      |     11000000 |
| table0                  | lts2      |     79000000 |
| table0                  | lts3      |     73000000 |
| table0                  | lts4      |     85000000 |
| table0                  | lts5      |     50000000 |
| table0                  | lts6      |     51000000 |
| table0                  | lts7      |     41000000 |
| table1                  | lts0      |     66000000 |
| table1                  | lts1      |     77000000 |
| table1                  | lts2      |     73000000 |
| table1                  | lts3      |     43000000 |
| table1                  | lts4      |     76000000 |
| table1                  | lts5      |     51000000 |
| table1                  | lts6      |     71000000 |
| table1                  | lts7      |    105000000 |
```

The query output shows how data is distributed across your nodes.

© 2026 Ocient, Inc.

2 Execute the `REBALANCE` task named `rebalance_task` to reorganize all data in a balanced state using the `CREATE TASK` SQL statement.

SQL

```
CREATE TASK rebalance_task TYPE REBALANCE;
```

For details about the syntax for creating tasks, see **Distributed Tasks**.

3 View the status of the tasks as the rebalance operation executes using the `sys.subtasks` system catalog table.

SQL

```
SELECT id,
    name,
    start_time,
    task_type,
    execution_type,
    status,
    details
FROM sys.subtasks
WHERE task_type = 'rebalance'
ORDER BY start_time DESC;
```

Output

Text

```
+--------------------------------------+--------+--------------------------
| id                                   | name   | start_time
|--------------------------------------+--------+--------------------------
| ed68a070-d7f7-4c50-9b94-84b04b4503c2 |        | 2024-04-17 21:44:20.26800
| 9e6c3374-6e9c-4a84-9bf8-ba1db3d866eb |        | 2024-04-17 21:44:20.26800
| aebb75d8-e066-4813-b7e8-c272e5a3b8e9 |        | 2024-04-17 21:44:14.76800
| 5cec5095-8b3a-4868-a10d-35fe6a32e998 |        | 2024-04-17 21:44:14.56400
+--------------------------------------+--------+--------------------------
```

© 2026 Ocient, Inc.

The output shows when the `REBALANCE` task is finished.

> When the `REBALANCE` task runs, segments are in the `REBUILDING` state. After the Ocient System completes this task, all segments should transition to the `INTACT` state.

4 Execute the query from Step 1 again to see how the system reorganized the data across nodes.

SQL

```
SELECT t.name,
    n.name,
    su.used_bytes
FROM sys.storage_used su
    JOIN sys.tables t ON t.id = su.table_id
    JOIN sys.nodes n ON n.id = su.node_id
ORDER BY t.name,
    n.name;
```

Output

Text

```
+------------------------+----------+--------------+
| name                   | name_1   |   used_bytes |
|------------------------+----------+--------------|
| table0                 | lts0     |     48000000 |
| table0                 | lts1     |     39000000 |
| table0                 | lts2     |     71000000 |
| table0                 | lts3     |     53000000 |
| table0                 | lts4     |     85000000 |
| table0                 | lts5     |     50000000 |
| table0                 | lts6     |     51000000 |
| table0                 | lts7     |     41000000 |
| table1                 | lts0     |     66000000 |
| table1                 | lts1     |     77000000 |
| table1                 | lts2     |     73000000 |
| table1                 | lts3     |     43000000 |
| table1                 | lts4     |     76000000 |
| table1                 | lts5     |     68000000 |
| table1                 | lts6     |     71000000 |
| table1                 | lts7     |     88000000 |
```

The output shows the redistributed data.

The `REBALANCE` task redistributes data across segment groups, so there can be some variance in data across nodes.

Add Loader Nodes

Adding more Loader Nodes to your system can improve the throughput of data loading and resiliency against loading failure. By having extra Loader Nodes, you can also dedicate sets of nodes for specific pipelines.

System Considerations

© 2026 Ocient, Inc.

Before starting this process, ensure you meet the requirements in the **Prerequisites** section.

Tutorial

1. Stop any active data pipelines. Execute this SQL statement by replacing `pipeline_name` with the name of your data pipeline.

```SQL
STOP PIPELINE <pipeline_name>;
```

2. Create the `/var/opt/ocient/bootstrap.conf` file on the new Loader Node using this YAML example. This YAML file should follow a similar format and parameters to the `bootstrap.conf` file on your other nodes.

   Replace the `<FIRST_NODE_ADDRESS>` with the IP address or hostname of a node running the Administrator Role on your system. If you are not using a DNS, use `nodeAddress` instead of `adminHost`.

   Specify a user account with System Administrator privileges for your system for the username `adminUserName` and password `adminPassword`.

```YAML
adminHost: <FIRST_NODE_ADDRESS>
adminUserName: my_admin
adminPassword: example_password
```

3. Start the new node by using this command.

```Shell
sudo systemctl start rolehostd
```

This process takes about a minute to complete. This step accepts the node into the system, but does not assign a role.

4 Add the `streamloader` role to the new node.

Execute this SQL statement by replacing `<new_node_name>` with the new node name.

SQL

```
ALTER NODE <new_node_name> ADD ROLE streamloader;
```

5 Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

Shell

```
sudo systemctl restart rolehostd
```

6 Restart the pipeline. Execute this SQL statement by replacing `pipeline_name` with the name of your data pipeline. Optionally, specify the Loader Node by name with the `USING LOADERS` keywords to prioritize it for this pipeline.

Shell

```
START PIPELINE <pipeline_name> USING LOADERS <new_node_name>;
```

If you use the legacy LAT service, you must stop loading and copy the LAT configuration files to any new Loader Nodes. For details, see **Configure the LAT Service**.

This step is unnecessary for systems that use Ocient data pipelines for loading.

Add SQL Nodes

Adding more SQL Nodes to your system can improve query optimization and processing, particularly for aggregation and join operations. Extra SQL Nodes also provide system resiliency, especially when assigned the `admin` role.

System Considerations

Before starting this process, ensure you meet the requirements in the **Prerequisites** section.

Tutorial

1. Create the `/var/opt/ocient/bootstrap.conf` file on the new SQL Node using this YAML example. This YAML file should follow a similar format and parameters to the `bootstrap.conf` file on your other nodes.

   Replace the `<FIRST_NODE_ADDRESS>` with the IP address or hostname of a node running the Administrator Role on your system. If you are not using a DNS, use `nodeAddress` instead of `adminHost`.

   Specify a user account with System Administrator privileges for your system for the username `adminUserName` and password `adminPassword`.

   YAML

   ```yaml
   adminHost: <FIRST_NODE_ADDRESS>
   adminUserName: my_admin
   adminPassword: example_password
   ```

2. Start the new node by using this command.

   Shell

   ```shell
   sudo systemctl start rolehostd
   ```

This process takes about a minute to complete. This step accepts the node into the system, but does not assign a role.

3 Add the `sql` role to the new node.

Execute this SQL statement by replacing `<new_node_name>` with the new node name.

```
SQL

ALTER NODE <new_node_name> ADD ROLE sql;
```

4 Optionally, you can also assign the `admin` role to the SQL Node. At least one SQL Node must always fulfill this role. By default, the system assigns the `admin` role to the first SQL Node in the system.

For details about the `admin` role, see **Node Configuration with the Administrator Role**.

Execute this SQL statement by replacing `<new_node_name>` with the name of your new node.

```
SQL

ALTER NODE <new_node_name> ADD ROLE admin;
```

5 Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

```
Shell

sudo systemctl restart rolehostd
```

6 Assign the new SQL Node to a connectivity pool with the **ALTER CONNECTIVITY_POOL** statement.

In this example, the statement assigns the SQL Node `sql2` to the connectivity pool `cp1` with the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

SQL

```
ALTER CONNECTIVITY_POOL cp1
    ADD PARTICIPANTS(
        NODE sql2
        LISTEN_ADDRESS '111.1.1.1'
        LISTEN_PORT 4050
        ADVERTISED_ADDRESS 'localhost'
        ADVERTISED_PORT 4050);
```

Related Links

Replace Nodes

Replace an NVMe Node Drive

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Replace Nodes

An Ocient® System requires a full node replacement in some scenarios. For example, the OS drive on a node might fail and necessitate a reinstallation and addition of the node. Other events, such as a complete node loss due to a hardware failure of the motherboard or damage to NVMe storage drives, can also cause you to replace a node.

An administrator should be familiar with the different types of nodes described in the **Core Elements of an Ocient System**. There are many similarities between the procedures for different node types, and the differences are clearly noted in the replacement procedures.

This procedure begins with **Prerequisites** then guides a System Administrator through replacing any of the Ocient node types.

This procedure also includes instructions about the replacement of an OS drive (also called a boot drive) on an Ocient node because the procedure for the replacement is identical to the replacement of the entire node. Both require reinstallation of the operating system and restoration of the configuration backup files to reestablish the identity of the Ocient System.

> ⚠ You must work carefully with nodes that also perform the Administrator Role, because you must maintain Administrator consensus.

Execution of this process requires System Administrator access to the Ocient System Database and `sudo` privilege on the node being replaced.

Detection and Alerting

You must monitor all nodes such that the loss of a node on the system triggers a node availability alert in the monitoring system.

This table shows the potential consequences of the loss of each type of node.

SQL Node loss

© 2026 Ocient, Inc.

Node Type	Impact of Node Loss
SQL	Might impact query performance in an Ocient System.
	If the only remaining SQL Node is lost in a system, you cannot query that system.
	If the SQL Node performs the Administrator Role, then the resiliency of the system could reduce to hardware failure. You must maintain the number of nodes that perform the Administrator Role at all times to maintain the resiliency of the system.

Foundation Node Loss

Node Type	Impact of Node Loss
Foundation	Can impact query performance or the load of new data.
	The parity configuration of the storage space in the Ocient System determines if new data can be stored while a Foundation Node is disabled.

Loader Node Loss

Node Type	Impact of Node Loss
Loader	Can impact the throughput of the data load in an Ocient System.
	If the remaining Loader Nodes are insufficient to handle the incoming data, the data can lead to a backlog.
	If the only remaining Loader Node is lost in a system, data loading stops.

Troubleshooting

You must replace a node in the event of a hardware failure of certain components. You can replace an OS drive, data drives, and RAM on most nodes. A node replacement might be necessary in the case of a CPU, motherboard, or Network Interface Card (NIC) replacement. The troubleshooting steps to recover an existing node are specific to the hardware model of the node.

Please contact Ocient Support for troubleshooting issues with a node.

Recovery

Prerequisites

- The SQL Nodes on the Ocient System are active and running Ocient (except for any SQL Nodes that are being replaced as part of this procedure).
- The OS has been installed and configured on the replacement node or OS drive for the node to be replaced.
- The replacement node has been installed with the same version of the Ocient Software as the rest of the Ocient System according to the **Ocient Application Installation**.
- The `bootstrap.conf` file for the node has been created in the `/var/opt/ocient` directory on the replacement node. For details, see **Ocient System Bootstrapping**.
- For Loader Nodes only:
  - The replacement node has been installed with the same version of the Loading and Transformation software as the rest of the Loader Nodes as described in **LAT Packaging and Installation**.
  - The following files are available from the node before the failure. If these files are not available, please contact Ocient Support for assistance.
    - `/etc/lat/lat.conf`
    - `/etc/lat/log4j2.xml`
    - `/opt/lat/.lat-data/*`

Replacement Procedure Overview

The replacement procedure is similar for all node types, but there are differences due to the way that different node types coordinate with one another.

The general procedure is:

1. Remove the failed node from the Ocient System.
2. Register the new node with the system and Accept Node.
3. Assign roles and add to clusters where appropriate.
4. Start the new node.

If you want to reuse the name of a node, then you must rename the failed node and set the IP address. To perform these actions, use these statements.

Rename the node from `nodename` to `newnodename`.

SQL

```
ALTER NODE "nodename" RENAME TO "newnodename";
```

Set the IP address for the new node.

SQL

```
ALTER NODE "newnodename" SET ADDRESS 'newnodename';
```

SQL Node Replacement Procedure

Step 1: Check SQL Node for the Administrator Role.

Before you replace a SQL Node, check whether the node performs the Administrator Role in addition to the SQL Node. If the node performs the Administrator Role, then follow this additional step to add this role to the new node.

> ⚠ Most SQL Nodes are configured to also perform the Administrator Role. When you remove nodes that perform the Administrator Role, you must replace that node with another node that performs the Administrator Role so that you maintain the level of redundancy in the RAFT protocol of the administrator.

Check the results by using this query.

SQL

```
SELECT
    n.name AS node_name,
    sr.service_role_type
FROM sys.nodes n
LEFT JOIN sys.service_roles sr ON sr.node_id = n.id
WHERE sr.service_role_type = 'admin';
```

Text

```
node_name service_role_type
------------------------------------------------------------------------------
----
sql0 admin
sql1 admin
sql2 admin
```

Step 2: Remove the Failed SQL Node from the Ocient System.

Remove the node role using the `ALTER NODE` SQL statement by replacing `sql0` with the node name of your node.

SQL

```
ALTER NODE "sql0" REMOVE ROLE sql;
```

Delete the node using the `DROP NODE` SQL statement by replacing `sql0` with the node name of your node.

SQL

```
DROP NODE "sql0";
```

If the failed node still exists in the Ocient System, the system does not add the replacement node.

Step 3: Register the New Node with the Ocient System.

Follow the **Bootstrapping Node Procedure** to register the new node. This procedure uses the `rolehostd` process to start any new nodes.

When bootstrapping the replacement node, you can reuse the original node name or assign a new one. The node name of the newly registered node is used in subsequent steps.

© 2026 Ocient, Inc.

Step 4: Add the SQL Role to the New Node.

To add a role, you must be a system-level user. Execute the following SQL statement from the SQL prompt, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

SQL

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" ADD ROLE sql;
```

Step 5: Add the Administrator Role. (Only for SQL Nodes with the Administrator Role)

If the node had the Administrator Role in Step 1, then add the Administrator Role back to the node. Execute the following SQL statement from the SQL prompt, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

SQL

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" ADD ROLE admin;
```

Step 6: Restart the Node.

Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

Shell

```
sudo systemctl restart rolehostd
```

Loader Node Replacement Procedure

Step 1: Remove the Failed Loader Node from the Ocient System.

Remove the node role using the `ALTER NODE` SQL statement by replacing `loader0` with the name of your node.

SQL

```
ALTER NODE "loader0" REMOVE ROLE streamloader;
```

Delete the node using the `DROP NODE` SQL statement, replacing `loader0` with the name of your node.

SQL

```
DROP NODE "loader0";
```

⚠ If the failed node still exists in the Ocient System, the system does not add the replacement node.

Step 2: Register the New Node with the Ocient System.

Follow the **Bootstrapping Node Procedure** to register the new node.

When bootstrapping the replacement node, you can reuse the original node name or assign a new one. The node name of the newly registered node will be used in subsequent steps.

Step 3: Add the `streamloader` Role to the New Node.

To add a role, you must be a system-level user. Execute the following SQL statement, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

SQL

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" add role streamloader;
```

ⓘ When adding the `streamloader` role, the node must be restarted if it is already running in order for the change to take effect.

Step 4: Configure the LAT Service.

Stop the LAT service by executing the following command at the shell prompt.

```shell
sudo systemctl stop lat
```

The LAT server configuration should be restored from the previous configuration. If a backup is unavailable, please contact Ocient Support for assistance or reference the other LAT configuration on the cluster to modify the default files.

- Copy the `lat.conf` backup file to `/etc/lat/lat.conf`.
- Copy the `log4j2.xml` backup file to `/etc/lat/log4j2.xml`.

Ensure that these two files have the appropriate ownership and permissions by executing the following commands.

```shell
chown lat:lat /etc/lat/lat.conf /etc/lat/log4j2.xml
chmod 644 /etc/lat/lat.conf /etc/lat/log4j2.xml
```

Step 5: Restart the Node.

Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

```shell
sudo systemctl restart rolehostd
```

Start the LAT service.

```shell
sudo systemctl start lat
```

Step 6: Update the LAT Pipeline if Required.

If a continuously streaming pipeline such as loading from Apache® Kafka® is operating on the Ocient System, then the Ocient System requires you to update the restored LAT to run the same pipeline to restore full loading throughput. This process is different if a long-running File Load is in process.

Kafka pipeline procedure

Pipeline Type	Procedure
Kafka Pipeline	1. From the LAT Client, create the pipeline on the new LAT Node. This ensures the pipeline_id matches the existing LAT processes.
	2. Start the pipeline on the new LAT Node. The new Loader Node joins the existing Loader Nodes, and the load automatically rebalances across all participating Loaders with the same `pipeline_id`

File load pipeline

Pipeline Type	Procedure
File Load Pipeline	File loads from sources like S3, local file system, or NFS require rebalancing of partitions in the event that a node is lost or added back to an Ocient System. In the event a node is lost, it is recommended that a rebalance is performed and the file load completes using the remaining nodes. The newly added node is not expected to participate in an already active file load.

Foundation Node Replacement Procedure

Step 1: Register the New Node with the Ocient System.

Follow the **Bootstrapping Node Procedure** to register the new node. When bootstrapping the new node, you can reuse the original node name or assign a new one. This tutorial uses the node name of the newly registered node in subsequent steps. These examples use `foundation_new` for the new node name.

> If you want to reuse the original node name (the name of the node being removed), you must first rename the original node to prevent conflict. For example, if the original node was named `foundation_original`, this can be performed with the SQL statement `ALTER NODE "foundation_original" RENAME TO "foundation_to_drop";`

Step 2: Add the New Node to a Specific Storage Cluster.

To operate as a Foundation Node, the node must be added to a storage cluster. To add a node to a storage cluster, you must be a system-level user. Execute the following SQL statement from the SQL prompt by replacing `storage_cluster_1` and `foundation_new` with your storage cluster name and node name, respectively.

SQL

```
ALTER CLUSTER "storage_cluster_1" ADD PARTICIPANTS "foundation_new";
```

Step 3: Restart the New Node.

Restart the `rolehostd` process on the new node by running this command at the shell terminal on the new node.

Shell

```
sudo systemctl restart rolehostd
```

The node should now be operational.

Step 4: Remove the Failed Foundation Node from the Ocient System.

Remove the original node from all clusters to which it belongs. For example, if the existing node is `foundation_original` and your storage cluster name is `storage_cluster_1`, execute the following SQL statement.

SQL

```
ALTER CLUSTER "storage_cluster_1" DROP PARTICIPANTS "foundation_original";
```

Delete the node using the `DROP NODE` SQL statement, replacing `foundation_original` with the name of your node.

© 2026 Ocient, Inc.

```SQL
DROP NODE "foundation_original";
```

> ⚠ If you renamed the failed original node in Step 1, be sure to use the correct name when dropping from the storage cluster and dropping the node.

Step 5: Rebuild Segments.

In most cases, after removing the original node, administrators follow the **Guide to Rebuilding Segments** to rebuild segments that were on the replaced Foundation Node.

Metadata Node Replacement Procedure

A Metadata Node is a node that only performs the Administrator Role.

Step 1: Remove the Failed Administrator Node from the Ocient System.

Delete the node using the `DROP NODE` SQL statement by replacing `standaloneAdmin00` with the name of your node.

```SQL
DROP NODE "standaloneAdmin00";
```

If the failed node still exists in the Ocient System, the system does not add the replacement node.

Step 2: Register the New Node with the Ocient System.

Follow the **Bootstrapping Node Procedure** to register the new node.

When bootstrapping the replacement node, you can reuse the original node name or assign a new one. This tutorial uses the node name of the newly registered node in subsequent steps.

Step 3: Add the Administrator Role to the New Node.

To add a role, you must be a system-level user. Execute the following SQL statement from the SQL prompt, replacing `<NODE_NAME_OF_THE_NEW_NODE>` with the new node name.

SQL

```
ALTER NODE "<NODE_NAME_OF_THE_NEW_NODE>" add role admin;
```

Step 4: Restart the Node.

Restart the `rolehostd` process on the replacement node by running this command at the shell terminal on the replacement node.

Shell

```
sudo systemctl restart rolehostd
```

Bootstrapping Node Procedure

To add a node to the system, follow these steps after meeting the prerequisites described in **Ocient System Bootstrapping**. These steps summarize the bootstrapping process.

> All prerequisites should be validated prior to bootstrapping. Ensure that you have set the correct `adminUserName` and `adminPassword` in the `bootstrap.conf` file. Any advanced configuration parameters should also be specified in the `bootstrap.conf` file.

Step 1: Create a `/var/opt/ocient/bootstrap.conf` file on the node by replacing the `<FIRST_NODE_ADDRESS>` with the IP or Hostname of a node running the Administrator Role and the correct user name and password for your system.

YAML

```
adminHost: <FIRST_NODE_ADDRESS>
adminUserName: my_admin
adminPassword: example_password
```

Step 2: Start the new node.

Shell

```
sudo systemctl start rolehostd
```

This action accepts the node into the system and prepares it for role assignment.

Related Links

ALTER NODE

DROP NODE

ALTER CLUSTER

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Backup and Restore

Ocient® can back up and restore a single node or a multi-node system. The backup and restore procedures can replace a node in an existing Ocient cluster due to an OS drive failure or hardware upgrade.

⚠ Exact backup and restore procedures depend on the disk layout of your Ocient nodes. Consult Ocient Support or the system administrator who installed your Ocient cluster to determine the best backup strategy for your deployment.

Related Links

Maintenance Overview

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Log Monitoring

> ⚠ Contact Ocient® Support for assistance before altering the system logging configuration in the `rolehostd.conf` file.

The OcientAIQ™ Unified Data Platform logs events using these severities:

- `Error`
- `Warn`
- `Info`
- `Verbose`
- `Debug`
- `Extended Debug`

By default, the data warehouse creates the text log file `/var/opt/ocient/rolehostd.log` and logs events at the `Info` level and above.

Database logging supports extensive customization and integration with third-party tools. You can customize using the `rolehostd` config file. The database can return events in text, JSON, or Graylog® Extended Log Format (GELF) formats, or the database can return events in a file, to standard output (`stdout`), or to a UDP port. You can customize based on the log source.

Log Formats

The default log format is text. These are some example log entries.

© 2026 Ocient, Inc.

Text

```
[2020-06-09T18:11:25.701594][ INFO][role001] roleHost: All roles started
[2020-06-09T18:11:25.701881][ INFO][role001] roleHost: Starting all endpoints
[2020-06-09T18:11:25.701925][ INFO][role001] roleHost: Starting endpoint 127.0.0.1:
[2020-06-09T18:11:25.704905][ INFO][role001] roleHost: Starting endpoint 127.0.0.1:
[2020-06-09T18:11:25.708437][ INFO][tcpU008] roleHost: All endpoints started
[2020-06-09T18:11:25.709236][ INFO][role001] host: ------------------------------
[2020-06-09T18:11:25.709286][ INFO][role001] host: Rolehostd successfully started
[2020-06-09T18:11:25.709305][ INFO][role001] host: ------------------------------
```

Each log line contains the time, severity, thread, and the source of the log (for example, `roleHost` or `host`).

JSON-formatted logs contain the same information but in a structured format.

JSON

```
{"sys":"oc1.admin0", "timestamp":1591726285.701594684, "thread":"role001", "lvl":"
{"sys":"oc1.admin0", "timestamp":1591726285.701881144, "thread":"role001", "lvl":"
{"sys":"oc1.admin0", "timestamp":1591726285.701925519, "thread":"role001", "lvl":"
{"sys":"oc1.admin0", "timestamp":1591726285.704905192, "thread":"role001", "lvl":"
{"sys":"oc1.admin0", "timestamp":1591726285.708437413, "thread":"tcpU008", "lvl":"
{"sys":"oc1.admin0", "timestamp":1591726285.709236006, "thread":"role001", "lvl":"
{"sys":"oc1.admin0", "timestamp":1591726285.709286683, "thread":"role001", "lvl":"
{"sys":"oc1.admin0", "timestamp":1591726285.709305246, "thread":"role001", "lvl":"
```

GELF-formatted log entries follow the format defined on the **GELF via UDP** page. This is an example GELF format.

JSON

```
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.701594684, "_thread"
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.701881144, "_thread"
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.701925519, "_thread"
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.704905192, "_thread"
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.708437413, "_thread"
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.709236006, "_thread"
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.709286683, "_thread"
{"host":"oc1.admin0", "version": "1.1", "timestamp":1591726285.709305246, "_thread"
```

© 2026 Ocient, Inc.

The database always returns message content in the `short_message` field of the GELF format.

This table maps Ocient severities to GELF system log (syslog) severities.

Ocient Severity	Syslog Severity	Syslog Numeric Value
`Error`	`Error`	3
`Warn`	`Warning`	4
`Info`	`Notice`	5
`Verbose`	`Info`	6
`Debug` and `Extended Debug`	`Debug`	7

Log Appenders

A log appender is the destination for log entries. The database can send log entries to a file, a UDP port, or to `stdout`.

When logs are sent to the file appender, they grow continuously unless you use the `truncate` option in the configuration. You can use tools such as `logrotate` to manage log files.

This is a sample `logrotate` configuration.

Text

```
/var/opt/ocient/log/rolehostd.log {
   rotate 5
   daily
   delaycompress
   compress
   copytruncate
   missingok
   minsize 1M
   maxsize 100M
}
```

Test the UDP appender using the Linux® `netcat` command.

LINUX

```
$ netcat -ul 9000
```

Log Configuration

An Ocient System supports various logging settings to meet different needs.

Set log configuration settings in `rolehostd.conf`, a YAML file normally found in the system `/var/opt/ocient/` directory. You can change configuration settings to suit your system logging needs in this file.

For information on checking the status of nodes, see **Statistics Monitoring**.

> If you are using an Ocient Simulator, logging behavior and directories might differ from these descriptions.
>
> For details on how logging works in the Ocient Simulator, see **Simulator Logging**.

`rolehostd` Example

This is an abbreviated `rolehostd.conf` example in YAML format. Normally, this file contains other system configurations, but this tutorial focuses on the `loggingConfig` section, which defines logging behaviors and scopes.

YAML

```yaml
loggingConfig:
    '':  # Default scope (all logs)
        appender:
        logFilePath: /path/to/rolehostd.log
    appenders:
    - appender:
        allowLevel: warn
        logFilePath: /path/to/error.log
    - appender:
        allowLevel: warn
        format: json
        logFilePath: /path/to/error.json

    dump:  # Dump-specific logs
        allowLevel: debug
        appender:
            logFilePath: /path/to/dump.log

    query:  # Query-specific logs
        allowLevel: info
        appender:
            format: json
            logFilePath: /path/to/query.json

    security:  # Security-specific logs
        allowLevel: info
        appender:
            format: json
            logFilePath: /path/to/security.json

    network-logs:  # Example network logging
        allowLevel: info
        appender:
            type: UDP
            server: graylog.internal
            port: 12201
            format: gelf
            compressed: true
```

Scope Definition

The top level of the logging configuration is the scope, which determines the corresponding log sources for each configuration setup.

Default Scope

An empty scope string (`''`) applies its configuration settings to all log entries the system returns.

Named Scopes

Named scopes apply to log entries where the first component matches the name. Useful scopes include:

- `query` — Contains information on all queries and statements executed on the system, including users, time, and other details.
- `security` — Contains information on all connections made to the system, successful or unsuccessful.
- `dump` — Contains information on system failures.

`allowLevel`

The `allowLevel` parameter specifies which severity levels of logs to capture.

YAML

```yaml
allowLevel: value
```

Severity `value` definitions are:

- `debug` — Most verbose (includes all logs).
- `info` — Includes informational messages and greater severities.
- `verbose` — More verbose than the `info` severity.
- `warn` — Includes warnings and errors only.
- `error` — Includes only error messages.

`appender`

Appenders define the format and directory where the database sends log files. You can configure either one or many appenders for each scope.

Single Appender

For a single appender, use the `appender` key.

YAML

```yaml
appender:
    key: value
    [ ... ]
```

Multiple Appenders

For multiple output destinations, use the `appenders` key, followed by a list of `appender` keys.

YAML

```yaml
 appenders:
 - appender:
     key: value
     [ ... ]
 - appender:
     key: value
     [ ... ]
 [ -...]
```

`format`

Defines the output format of the log files of an appender.

YAML

```yaml
appender:
    format: format_type
```

Supported `format_type` values are:

- `text` — Human-readable text format (default).
- `json` — Structured JSON format.
- `gelf` — GELF format for integration with Graylog.

`logFilePath`

Defines the destination directory for log files of the appender. The `directory` value is a file path for the appender log files, for example: `/path/to/rolehostd.log`.

YAML

```yaml
appender:
    logFilePath: directory
```

`type`

Determines whether the system submits error logs through `stdout` or to a separate server using a TCP or UDP protocol.

YAML

```yaml
appender: # standard output example
    type: stdout

appender: # UDP example
    type: UDP
    server: graylog-server.example.com
    port: 12201
    chunkSize: 1024  # Optional
    compressed: true  # Optional

appender: # TCP example
    type: TCP
    server: 192.158.1.38
    port: 12201
```

`server`

Required for TCP or UDP types.

The destination DNS name or IP address where the database sends log entries.

YAML

```yaml
appender:
    server: 192.158.1.38
```

`port`

Required for TCP or UDP types.

The port where the database sends log entries. The default value is `9000`.

YAML

```yaml
appender:
    port: 12201
```

`chunkSize`

Optional for UDP type.

The largest packet size in bytes that the system sends to the server. The database discards log entries larger than this value.

YAML

```yaml
appender:
    chunkSize: 1024
```

`compressed`

Optional for UDP type.

If you set this option to `true`, the system compresses packets. For details about log compression, see the **GELF documentation**. Otherwise, the packets are uncompressed.

YAML

```yaml
appender:
    compressed: true
```

`truncate`

If you set this option to `true`, the system truncates the log file when the system starts. Otherwise, the system appends new logs to the file. The default value is `false`.

YAML

```yaml
appender:
    truncate: true
```

`redactions`

You can redact data in logs at the SQL statement level using the optional `redactions` configuration. Enabling redaction creates another version of the logs with redacted SQL text. The `redactions` configuration value is `statement` for the redaction of part of the statement.

YAML

```yaml
appender:
    redactions:
    - statement # Optional
```

`rolehostd` Log Redaction Example

This abbreviated `rolehostd.conf` file configuration enables log redaction at the statement level for both the `rolehostd.log` and `query.json` log files.

YAML

```yaml
loggingConfig:
    ? '':
    appenders:
    - appender:
        logFilePath: /var/opt/ocient/logs/rolehostd.log
    - appender:
        logFilePath: /var/opt/ocient/logs/rolehostd_redacted.log
        redactions:
        - statement
    query:
        allowLevel: info
        appenders:
        - appender:
            format: json
            logFilePath: /var/opt/ocient/logs/query.json
        - appender:
            format: json
            logFilePath: /var/opt/ocient/logs/query_redacted.json
            redactions:
            - statement
```

Related Links

Install an Ocient System

Set Up System Monitoring with the TIG Stack and Kapacitor

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Statistics Monitoring

The OcientAIQ™ Unified Data Platform maintains many internal statistics. You can access the statistics on each node in the database using a REST interface that listens on port 9090 by default. For the endpoints, see **System Information REST Endpoints**.

Explore the statistics using a command such as curl.

CURL

```
curl http://oc1-lts0:9090/v1/stats
```

The resulting statistics return in JSON format. This excerpt shows only a partial list of the results.

© 2026 Ocient, Inc.

JSON


```
[{
   {
        "name": "localStorageService.device[SDM00001A616].spaceFree",
        "node": "oc1-lts0",
        "time": 1591737983812323,
        "units": "bytes",
        "value": 356599726080
   },
   {
        "name": "localStorageService.device[SDM00001A616].spaceFreePct",
        "node": "oc1-lts0",
        "time": 1591737983812323,
        "units": "none",
        "value": 4631469015491373042
   },
   {
        "name": "localStorageService.device[SDM00001A616].spaceTotal",
        "node": "oc1-lts0",
        "time": 1591737983812323,
        "units": "bytes",
        "value": 800149299200
   }
}]
```


Because each node can produce a very large number of statistics, the REST interface supports wildcard filtering.

Return statistics that match the regular expression `^localStorageService*` using this curl command.

CURL

```
curl -G --data-urlencode 'filter=^localStorageService*' http://oc1-lts0:9090/v1/s
```

The resulting subset of statistics returns in JSON format.

© 2026 Ocient, Inc.

JSON


```
[
    {
        "name": "localStorageService.device.segmentTableEntries",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 1,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.spaceFree",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 0,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.spaceFreePct",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 0.000000,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.spaceTotal",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 2139074560,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.status",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 10,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
```


© 2026 Ocient, Inc.

```
    {
        "name": "localStorageService.device.segmentTableEntries",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 1,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.spaceFree",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 0,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.spaceFreePct",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 0.000000,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.spaceTotal",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 2139074560,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    },
    {
        "name": "localStorageService.device.status",
        "time": "1684931806764193",
        "timestamp": "2023-05-24T12:36:46.764Z",
        "node": "sql0",
        "value": 10,
        "device": "4568d611-9ea9-4f65-d0z9-8gg5bb220750"
    }
]
```


© 2026 Ocient, Inc.

The specific set of statistics is subject to change.

Retrieve Statistics Using Telegraf

Telegraf™ is a commonly used monitoring tool. This example shows the configuration file for Telegraf that gathers statistics from a database node where the statistic name includes `localStorageService`.

Text

```
; Read formatted metrics from one or more HTTP endpoints
[[inputs.http]]
  interval = "30s"

  urls = [
    "http://oc1-lts0:9090/v1/stats?filter=localStorageService"
  ]

  ; HTTP method
  method = "GET"

  data_format = "json"
  json_name_key = "name"
  json_time_key = "time"
  json_time_format = "unix_us"
  json_timezone = "America/Chicago"
  tag_keys = [
    "node",
    "silo",
    "core"
  ]
  [inputs.http.tags]
    source = "rolehostd"
```

Related Links

Install an Ocient System

© 2026 Ocient All Rights Reserved

System Information REST Endpoints

The OcientAIQ™ Unified Data Platform maintains many internal statistics. You can access the statistics on each node in the database using a REST interface that listens on port 9090 by default. For an overview of monitoring statistics, see **Statistics Monitoring**.

Get Statistics

Get Version

Get Status

Get Configuration

Get Configuration of Whole System

Get Configuration Parameters

© 2026 Ocient All Rights Reserved

Terms Of Use Privacy Statement

Get Statistics

Retrieve the statistics on each node in the database.

GET /v1/stats

curl

cURL

```
curl --location --request GET 'http://oc1-lts0:9090/v1/stats' \
--header 'Accept: application/json'
```

200


```
[
  {
    "name": "vm.stats.pdfCacheSize",
    "time": "1686335017884363",
    "timestamp": "2023-06-09T18:23:37.884Z",
    "node": "sql0",
    "value": 4680936
  },
  {
    "name": "localStorageService.device.spaceFree",
    "time": "1686335145388588",
    "timestamp": "2023-06-09T18:25:45.388Z",
    "node": "sql0",
    "value": 0,
    "device": "4418fbc3-fcf9-5cfe-b2b3-b41a8247a600"
  }
]
```


© 2026 Ocient, Inc.

These fields appear in every entry returned in the response from the `http://oc1-lts0:9090/v1/stats` endpoint:

- `name`
- `time`
- `timestamp`
- `node`
- `value`

Additional fields can vary.

Body

application/json

filter `string`

Filter for the statistics.

Response

200 - application/json

Response

name `string`

time `string`

timestamp `string`

node `string`

value `integer`

© 2026 Ocient All Rights Reserved

Get Version

Retrieve the version of the software running.

GET /v1/version

curl

cURL

```
curl --location --request GET 'http://oc1-lts0:9090/v1/version' \
--header 'Accept: application/json'
```

200


```
{
  "version": "24.0.0",
  "timestamp": "20231110.212417",
  "git_commit": "178ba7a7549bd40c137035c5cd4f10c55fd31b00",
  "build_user": "",
  "build_type": "DEBUG",
  "build_flags": "fb,lld_flags,mcmodel_medium,tls_model_initial_exec"
}
```


Response

200 - application/json

Response

version string

timestamp string

git_commit string

build_user string

© 2026 Ocient, Inc.

build_type `string`

---

build_flags `string`

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Get Status

Retrieve the status of the software running.

GET /v1/status

curl cURL

```
curl --location --request GET 'http://oc1-lts0:9090/v1/status' \
--header 'Accept: application/json'
```

200

```
"Active"
```

Response

200 - text/plain

Response

The response is of type `string`.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Get Configuration

Retrieve the configuration about this node in JSON format.

GET /v1/sysconfig

curl

cURL

```
curl --location --request GET 'http://oc1-lts0:9090/v1/sysconfig' \
--header 'Accept: application/json'
```

© 2026 Ocient, Inc.

200


```
{
  "system_name": "abc7d45f-1e9c-4210-811d-ffg6dbd82297",
  "node_name": "sql0",
  "node_uuid": "578c11a2-1996-523f-a371-c393b4e5ff2e",
  "hostname": "username-dev0",
  "roles": [
    "administration",
    "health",
    "vm",
    "cmdComp"
  ],
  "failed_roles": [],
  "endpoints": [
    {
      "name": "adminopenendpoint",
      "address": "localhost",
      "port": 19000
    },
    {
      "name": "restendpoint",
      "address": "localhost",
      "port": 8000
    },
    {
      "name": "adminsecureendpoint",
      "address": "localhost",
      "port": 18500
    },
    {
      "name": "dataendpoint",
      "address": "localhost",
      "port": 16500
    },
    {
      "name": "sqlexternal",
      "address": "localhost",
      "port": 3070
    }
  ],
  "totalram": 450948103168,
  "processors": 120,
  "hugepages": [
    {
      "size": 1257152,
      "total": 0,
```


© 2026 Ocient, Inc.

```
      "free": 0,
      "reserved": 0
    },
    {
      "size": 1060041824,
      "total": 0,
      "free": 0,
      "reserved": 0
    }
  ],
  "storage_devices": [
    {
      "id": "4425fad3-fcf9-5cfd-f3b3-b47a8457a300",
      "address": "/address/4425fad3-fcf9-5cfd-f3b3-b47a8457a300.dat",
      "type": "SATA",
      "use": "PAYLOAD",
      "pci_address": "",
      "mount_point": "",
      "device_path": ""
    },
    {
      "id": "9ae82aa7-ab73-5g86-bb34-13b27945c701",
      "address": "/address/9ae82aa7-ab73-5g86-bb34-13b27945c701.dat",
      "type": "SATA",
      "use": "PAYLOAD",
      "pci_address": "",
      "mount_point": "",
      "device_path": ""
    }
  ],
  "connections": [
    {
      "address": "111.1.1.1",
      "remote_port": 45598,
      "state": "connected",
      "locally_initiated": false
    }
  ]
}
```


Response

200 - application/json

Response

system_name `string`

node_name `string`

node_uuid `string`

hostname `string`

roles `string[]`

failed_roles `any[]`

endpoints `object[]`

Show child attributes

totalram `integer`

processors `integer`

hugepages `object[]`

Show child attributes

storage_devices `object[]`

Show child attributes

connections `object[]`

Show child attributes

© 2026 Ocient All Rights Reserved

Get Configuration of Whole System

Retrieve the configuration of the system as a whole in JSON format.

GET /v1/dbconfig

curl

cURL

```
curl --location --request GET 'http://oc1-lts0:9090/v1/dbconfig' \
--header 'Accept: application/json' \
--data-raw '{"filter":"string"}'
```

© 2026 Ocient, Inc.

200


```
{
  "system_name": "abc7d45f-1e9c-4210-811d-ffg6dbd82297",
  "node_name": "sql0",
  "node_uuid": "578c11a2-1996-523f-a371-c393b4e5ff2e",
  "hostname": "username-dev0",
  "roles": [
    "administration",
    "health",
    "vm",
    "cmdComp"
  ],
  "failed_roles": [],
  "endpoints": [
    {
      "name": "adminopenendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 18000
    },
    {
      "name": "restendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 8000
    },
    {
      "name": "adminsecureendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 18500
    },
    {
      "name": "dataendpoint",
      "type": "FULL",
      "address": "localhost",
      "port": 16500
    },
    {
      "name": "sqlexternal",
      "type": "FULL",
      "address": "localhost",
      "port": 3070
    }
  ],
  "totalram": 450948103168,
```


© 2026 Ocient, Inc.

```
  "processors": 120,
  "hugepages": [
    {
      "size": 1257152,
      "total": 0,
      "free": 0,
      "reserved": 0
    },
    {
      "size": 1060041824,
      "total": 0,
      "free": 0,
      "reserved": 0
    }
  ],
  "storage_devices": [
    {
      "id": "4425fad3-fcf9-5cfd-f3b3-b47a8457a300",
      "address": "/address/4425fad3-fcf9-5cfd-f3b3-b47a8457a300.dat",
      "type": "SATA",
      "use": "PAYLOAD",
      "pci_address": "",
      "mount_point": "",
      "device_path": ""
    },
    {
      "id": "9ae82aa7-ab73-5g86-bb34-13b27945c701",
      "address": "/address/9ae82aa7-ab73-5g86-bb34-13b27945c701.dat",
      "type": "SATA",
      "use": "PAYLOAD",
      "pci_address": "",
      "mount_point": "",
      "device_path": ""
    }
  ],
  "connections": [
    {
      "address": "111.1.1.1",
      "remote_port": 45598,
      "state": "connected",
      "locally_initiated": false
    }
  ]
}
```


© 2026 Ocient, Inc.

Response

200 - application/json

Response

system_name `string`

node_name `string`

node_uuid `string`

hostname `string`

roles `string[]`

failed_roles `any[]`

endpoints `object[]`

Show child attributes

totalram `integer`

processors `integer`

hugepages `object[]`

Show child attributes

storage_devices `object[]`

Show child attributes

connections `object[]`

Show child attributes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Get Configuration Parameters

Retrieve values of configuration parameters.

GET /v1/config

curl

cURL

```
curl --location --request GET 'http://oc1-lts0:9090/v1/config' \
--header 'Accept: application/json'
```

200

```
"{\n  \"siloType\": \"xg::runtime::physicalSiloSet_t\",\n  \"tlbfsInfo.enabled\": \"false\",\n
```

Response

200 - text/plain

Response

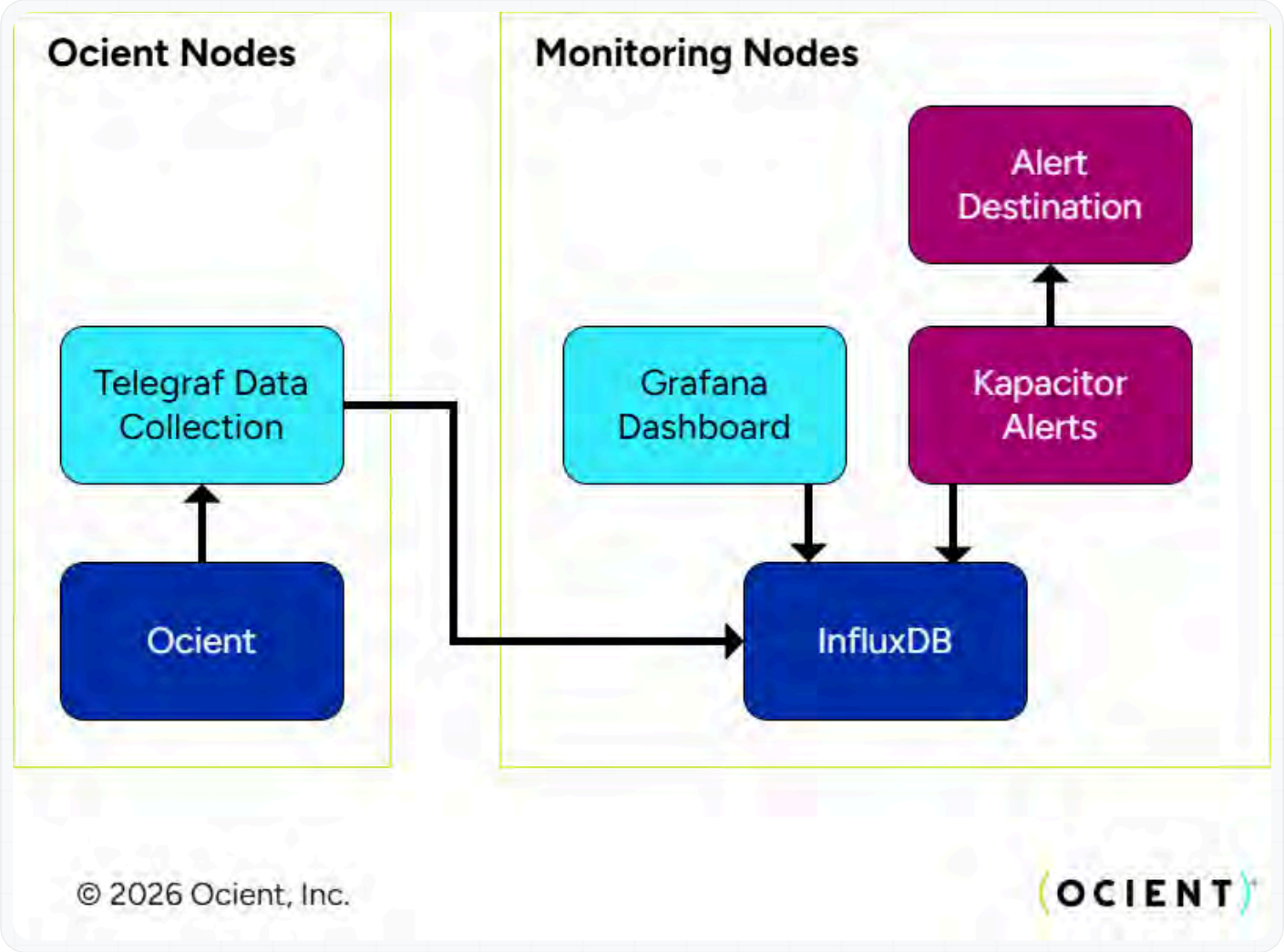
The response is of type `string`.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Set Up System Monitoring with the TIG Stack and Kapacitor

One method to monitor the OcientAIQ™ Unified Data Platform is to use the TIG (Telegraf™, InfluxDB® and Grafana®) Stack. While these instructions are specifically written for this toolset, you can also use this guide as the foundation to set up an alternate stack if necessary.

This guide uses InfluxDB, Telegraf, Kapacitor™, and Grafana for a complete monitoring solution.



This guide specifically focuses on host-level and Ocient® software metrics. Monitoring additional components (e.g., Apache® Kafka®) is outside of the scope of this document.

These instructions were created using these versions of each component.

Component	Version
InfluxDB	1.8 (OSS)
Telegraf	1.16.1

Component	Version
Kapacitor	1.5.6 (OSS)
Grafana	7.4.3

Prerequisites

This guide assumes that:

1. SSH access and root-level privileges are available to each server running Ocient Software.
2. InfluxDB, Kapacitor, and Grafana are available to be installed in a virtual machine or container. It is important to note that each of these systems should be run independently. For example, do not run InfluxDB, Kapacitor, and Grafana on the same virtual machine.
3. All Ocient software is currently deployed and running as expected.
4. Logging configuration is set up as specified in **Log Configuration**.
5. The following ports must be open for InfluxDB, Kapacitor, Telegraf, and Grafana. InfluxDB is the most critical as each component communicates with it. Always refer to the latest documentation for each product as the definitive reference.

Component	Default Ports	Components Requiring Access
InfluxDB	8086 8088 (optional, used for backup utilities)	Telegraf, Kapacitor, Grafana
Telegraf	None	None
Kapacitor	9092	None
Grafana	3000	None

Step 1: Deploy InfluxDB

InfluxDB is a timeseries database that provides persistence for the host-level and Ocient software metrics.

1. Refer to current instructions from InfluxDB on how to install the service:
https://docs.influxdata.com/influxdb/v2.0/install/
https://docs.influxdata.com/influxdb/v2.0/get-started/

2. Document the IP address of the machine where InfluxDB is installed.

3. Ensure that InfluxDB is running properly.

Shell

```
/> sudo systemctl status influxdb
influxdb.service - InfluxDB is an open-source, distributed, time series database
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
Active: active (running) since Tue 2021-09-14 17:17:20 UTC; 3 weeks 2 days ago
Docs:
Main PID: 32687 (influxd)
Tasks: 18 (limit: 4915)
CGroup: /system.slice/influxdb.service
└─32687 /usr/bin/influxd -config /etc/influxdb/influxdb.conf/
```

4. Set the InfluxDB service to start on boot.

Shell

```
/> sudo systemctl enable influxdb
```

5: It is highly recommended that critical components of monitoring infrastructure, including InfluxDB, are also monitored. Monitoring the monitoring stack is outside of the scope of this document. Refer to available resources on potential solutions to monitor InfluxDB.

Step 2: Deploy and Configure Telegraf

Telegraf is used for metrics collection. It runs on each of the Admin, Loader, Foundation, SQL Nodes. The following instructions highlight the common configuration elements, followed by the elements that are specific to different node types.

All Nodes

1. Install **Telegraf** on each of the Admin, Loader, Foundation, and SQL Nodes. Refer to the Telegraf installation instructions available from InfluxDB. At this point, do not start the Telegraf service or generate the default configuration.

2. For **all nodes**, create the Telegraf configuration file (typically located at `/etc/telegraf/telegraf.conf`). Open the file in an editor.

- Copy and paste the following contents into the file.
- Replace the designated placeholders (denoted with a $) with the specific values for the specified environment:

- `CLUSTER_NAME =` — Customer-selected name of a provided cluster for monitoring identification
- `ROLE =` — One of `admin`, `loader`, `lts`, `sql`, or `stream_proc`, depending on the node type
- `INFLUX_URL =` — URL of the InfluxDB instance (e.g., `http://10.6.0.4:8086`)

Text

```
; /etc/telegraf/telegraf.conf
[global_tags]
   cluster = "$CLUSTER_NAME"
   cluster_role = "$ROLE" # one of admin, loader (aka indexer), lts (aka foundation), sql, s

[agent]
  interval = "15s" # time
  metric_batch_size = 1000 # size
  metric_buffer_limit = 100000 # size
  collection_jitter = "3s" # time
  flush_interval = "7s" # time
  flush_jitter = "3s" # time
  round_interval = false

[outputs]

[[outputs.influxdb]]
  urls = [ "$INFLUX_URL" ]
  database = "telegraf"
  precision = "1s"
  timeout = "10s"
```

Save the file.

3. For **all nodes**, create a new file under the telegraf.d directory (typically `/etc/telegraf/telegraf.d`) named `host.conf`.

- Copy and paste the following contents into the file.
- Replace the designated placeholders (denoted with an `$`) with the specific values for the specified environment:
  - `BOOT_DISK` — Device name of the boot disk (e.g., `sda`). You can determine the boot disk by running an `lsblk` command.
  - `NET_INT_BONDO_WILDCARD` — The wildcard for matching the underlying network interfaces within `bond0` (e.g., `eno*`). If it is unclear, refer to `/proc/net/bonding/bond0` and reference the interfaces noted in the **Telegraf documentation**.

- `NET_INT_BOND1_WILDCARD` — The wildcard for matching the underlying network interfaces within `bond1` (e.g., `enp*`). If it is unclear, refer to `/proc/net/bonding/bond1` and reference the interfaces noted in the **Telegraf documentation**.

Text

```
; /etc/telegraf/telegraf.d/host.conf
[[inputs.cpu]]
    percpu = true
    totalcpu = true
    fielddrop = [ "time_*" ]
[[inputs.disk]]
    ignore_fs = ["tmpfs", "devtmpfs", "vfat"]
[[inputs.diskio]]
    skip_serial_number = true
    devices = ["$BOOT_DISK"]
    [inputs.diskio.tags]
        disk_type="os"

[[inputs.mem]]

[[inputs.net]]
    interfaces = ["$NET_INT_BONDO_WILDCARD", "$NET_INT_BOND1_WILDCARD"]

[[inputs.net]]
    interfaces = ["bond0"]
    [inputs.net.tags]
        primary_interface_10Gb = "true"
    [inputs.net.tagdrop]
        interface = [ "all" ]

[[inputs.net]]
    interfaces = ["bond1"]
    [inputs.net.tags]
        primary_interface_100Gb = "true"
    [inputs.net.tagdrop]
        interface = [ "all" ]

[[inputs.system]]
[[inputs.swap]]

[[inputs.linux_sysctl_fs]]
```

Save the file.

4. For **all nodes**, create a new file under the telegraf scripts directory (typically under `/etc/telegraf/scripts`) named `uio_pci_generic.sh`.

- Copy and paste the following contents into the file.

Shell

```
; /etc/telegraf/scripts/uio_pci_generic.sh
; A script used to enumerate the PCI devices on the system.
;
; !/bin/bash

DEVICES=$(find /sys/bus/pci/drivers/uio_pci_generic/0000:* | awk -F'/' '{ print $NF }')

for d in $DEVICES; do
  echo pci_devices,device=\"$d\" present=1
done
```

Foundation and SQL Nodes

Create a new file under the `telegraf.d` directory (typically `/etc/telegraf/telegraf.d`) named `rolehostd.conf`.

- Copy and paste the following contents into the file.
- Replace the designated placeholders (denoted with an $) with the specific values for the specified environment:
  - `NODE_BONDO_IP_ADDRESS =` IP address associated with bond0

Text

```
[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_status"
  data_format = "value"
  data_type = "string"

[[inputs.http_response]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  response_timeout = "5s"
  name_override = "rolehostd_status_response"
  response_string_match = "Active"

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/stats"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd"
  json_time_key = "time"
  json_time_format = "unix_us"
  json_string_fields = ["units"]
  data_format = "json"
  tagexclude = ["url", "node"]
  tag_keys = [
    "name",
    "socket",
    "device"
  ]

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/version"
      ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_version"
  json_query = "{build_type,version}"
  json_string_fields = ["build_type","version"]
```

```
  data_format = "json"

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/operatorsummary"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_operatorsummary"
  data_format = "json"

[[inputs.exec]]
  commands = ['sh /etc/telegraf/scripts/uio_pci_generic.sh']
  data_format = "influx"

[[inputs.tail]]
    name_override = "tail_query_json"
    files = ["/var/opt/ocient/log/query.json","/var/opt/ocient/query.json"]
    data_format = "json"
    json_query = "{src,msg.user,msg.database,msg.service_class,msg.client_version,msg.client
    tag_keys = ["src","user","database","service_class","code","major_driver_version","resul
    json_string_fields = ["client_version","client_ip","timestamp_start","timestamp_execsta

[[processors.converter]]
    namepass=["tail_query_json"]
    [processors.converter.fields]
        integer = ["timestamp_start","timestamp_execstart","timestamp_complete"]
```

You can also use **Filebeat** to forward the `query.json` file to a monitoring platform of your choice.

Admin and Loader Nodes

Create a new file under the `telegraf.d` directory (typically `/etc/telegraf/telegraf.d`) named `rolehostd.conf`.

- Copy and paste the following contents into the file.
- Replace the designated placeholders (denoted with an $) with the specific values for the specified environment:
  - `NODE_BONDO_IP_ADDRESS =` IP address associated with bond0

Text

```
; /etc/telegraf/telegraf.d/rolehostd.conf
[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_status"
  data_format = "value"
  data_type = "string"

[[inputs.http_response]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/status"
  ]
  method = "GET"
  response_timeout = "5s"
  name_override = "rolehostd_status_response"
  response_string_match = "Active"

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/stats"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd"
  json_time_key = "time"
  json_time_format = "unix_us"
  json_string_fields = ["units"]
  data_format = "json"
  tagexclude = ["url", "node"]
  tag_keys = [
    "name",
    "socket",
    "device"
  ]

[[inputs.http]]
  urls = [
    "http://$NODE_BONDO_IP_ADDRESS:9090/v1/version"
  ]
  method = "GET"
  timeout = "5s"
  name_override = "rolehostd_version"
  json_query = "{build_type,version}"
```

```
  json_string_fields = ["build_type","version"]
  data_format = "json"

[[inputs.exec]]  commands = ['sh /etc/telegraf/scripts/uio_pci_generic.sh']
  data_format = "influx"

[[inputs.tail]]
    name_override = "tail_query_json"
    files = ["/var/opt/ocient/log/query.json","/var/opt/ocient/query.json"]
    data_format = "json"
    json_query = "{src,msg.user,msg.database,msg.service_class,msg.client_version,msg.client
    tag_keys = ["src","user","database","service_class","code","major_driver_version","resu
    json_string_fields = ["client_version","client_ip","timestamp_start","timestamp_execsta

[[processors.converter]]
    namepass=["tail_query_json"]
    [processors.converter.fields]
        integer = ["timestamp_start","timestamp_execstart","timestamp_complete"]
```

Loader Nodes

Loader Nodes also run the Loading and Transformation service and require additional Telegraf configuration to capture those metrics.

- Create a new file under the `telegraf.d` directory (typically `/etc/telegraf/telegraf.d`) named `lat.conf`.
- Copy and paste the following contents into the file.

Text

```
; /etc/telegraf/telegraf.d/lat.conf
[[inputs.jolokia2_agent]]
  urls = ["http://localhost:8080/v2/metrics"]

  [[inputs.jolokia2_agent.metric]]
    name  = "pipeline"
    mbean = "lat:type=pipeline"

  [[inputs.jolokia2_agent.metric]]
    name  = "partitions"
    mbean = "lat:type=partitions"

[[inputs.procstat]]
  systemd_unit = "lat.service"
```

Start Telegraf

When the configuration is properly entered on all nodes, Telegraf must be started.

1. On **each node**, start the Telegraf service.

```
Shell
/> sudo systemctl start telegraf
```

2. Ensure the service came up properly by checking the status.

```
Shell
/> sudo systemctl status telegraf
telegraf.service - The plugin-driven server agent for reporting metrics into InfluxDB
Loaded: loaded (/usr/lib/systemd/system/telegraf.service; enabled; vendor preset: disabled)
Active: active (running) since Thu 2021-08-05 22:04:06 UTC; 2 months 3 days ago
Docs: https://github.com/influxdata/telegraf
Main PID: 2651775 (telegraf)
Tasks: 103 (limit: 1589476)
Memory: 221.2M
CGroup: /system.slice/telegraf.service
└─2651775 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/teleg
```

3. Set the Telegraf service to start on boot.

```
Shell
/> sudo systemctl enable telegraf
```

Step 3: Deploy and Configure Kapacitor

Kapacitor is used to process time series data, detect anomalies, and trigger alerts. As the preferred alerting mechanisms for every customer will vary, refer to the Kapacitor documentation on **how to send alerts through specific tools**.

1. Install Kapacitor using the instructions from InfluxDB.
https://docs.influxdata.com/kapacitor/v1.6/introduction/installation/

2. Create the Kapacitor configuration file (typically located at `/etc/kapacitor/kapacitor.conf`). Open the file in an editor.

- Copy and paste the following contents into the file.

- Replace the designated placeholders (denoted with an $) with the specific values for the specified environment:
  - `KAPACITOR_HOST_OR_IP =` — Hostname or IP address associated with the Kapacitor instance
  - `INFLUX_URL =` — URL of the InfluxDB instance (e.g., `http://10.6.0.4:8086`)

Text

```
; /etc/kapacitor/kapacitor.conf
hostname = "$KAPACITOR_HOST_OR_IP"
data_dir = "/var/lib/kapacitor"
skip-config-overrides = false

[logging]
    # Destination for logs
    # Can be a path to a file or 'STDOUT', 'STDERR'.
    file = "/var/log/kapacitor/kapacitor.log"
    level = "ERROR"

[load]
  enabled = true
  dir = "/etc/kapacitor/load"

[replay]
  # Where to store replay files, aka recordings.
  dir = "/var/lib/kapacitor/replay"

[storage]
  boltdb = "/var/lib/kapacitor/kapacitor.db"

[[influxdb]]
  enabled = true
  default = true
  name = "influx"
  urls = ["$INFLUX_URL"]
  username = "admin"
  password = ""
  timeout = "10s"

  startup-timeout = "60s"

  # Subscription mode is either "cluster" or "server"
  subscription-mode = "server"

  # Which protocol to use for subscriptions
  # one of 'udp', 'http', or 'https'.
    subscription-protocol = "http"

  # Subscriptions resync time interval
  # Useful if you want to subscribe to new created databases
  # without restart Kapacitord
  subscriptions-sync-interval = "1m0s"

; INSERT ALERT DESTINATION CONFIGS HERE
```

© 2026 Ocient, Inc.

```
[stats]
  enabled = true
  stats-interval = "10s"
  database = "_kapacitor"
  retention-policy= "autogen"
```

Add Kapacitor Alerts and Templates

Ocient provides sample Kapacitor alerts and templates upon request. Contact Ocient Support for details.

Within the bundle, relevant contents are located within the `kapacitor` directory.

1. Copy the files located under `kapacitor/load/tasks` to the Kapacitor system in `/etc/kapacitor/load/tasks`.
2. Copy the files located under `kapacitor/load/templates` to the Kapacitor system in `/etc/kapacitor/load/templates`.

Start Kapacitor

When the configuration is complete, Kapacitor must be started.

1. On **each node**, start the Kapacitor service.

Shell

```
/> sudo systemctl start kapacitor
```

2. Ensure the service came up properly by checking the status.

© 2026 Ocient, Inc.

Shell

```
/> sudo systemctl status kapacitor
kapacitor.service - Time series data processing engine.
   Loaded: loaded (/lib/systemd/system/kapacitor.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2021-09-03 18:30:37 UTC; 1 months 9 days ago
     Docs: https://github.com/influxdb/kapacitor
 Main PID: 16177 (kapacitord)
    Tasks: 16 (limit: 4915)
   CGroup: /system.slice/kapacitor.service
           └─16177 /usr/bin/kapacitord -config /etc/kapacitor/kapacitor.conf

Sep 03 18:30:37 kapacitor systemd[1]: Started Time series data processing engine..
Sep 03 18:30:37 kapacitor kapacitord[16177]: '##:::'##::::'###::::'########:::::'###:::::'#
Sep 03 18:30:37 kapacitor kapacitord[16177]:  ##::'##::::'## ##::: ##.... ##:::'## ##:::'##
Sep 03 18:30:37 kapacitor kapacitord[16177]:  ##:'##::::'##:. ##:: ##:::: ##::'##:. ##:: ##
Sep 03 18:30:37 kapacitor kapacitord[16177]:  #####:::'##:::. ##: ########::'##:::. ##: ##
Sep 03 18:30:37 kapacitor kapacitord[16177]:  ##. ##::: #########: ##.....::: #########: ##
Sep 03 18:30:37 kapacitor kapacitord[16177]:  ##:. ##:: ##.... ##: ##:::::::: ##.... ##: ##
Sep 03 18:30:37 kapacitor kapacitord[16177]:  ##::. ##: ##:::: ##: ##:::::::: ##:::: ##:. ##
Sep 03 18:30:37 kapacitor kapacitord[16177]: ..::::..::..::::::..::..::::::::::..::::..::::..
Sep 03 18:30:37 kapacitor kapacitord[16177]: 2021/09/03 18:30:37 Using configuration at: /et
```

3. Set the Kapacitor service to start on boot.

Shell

```
/> sudo systemctl enable kapacitor
```

Step 4: Deploy and Configure Grafana

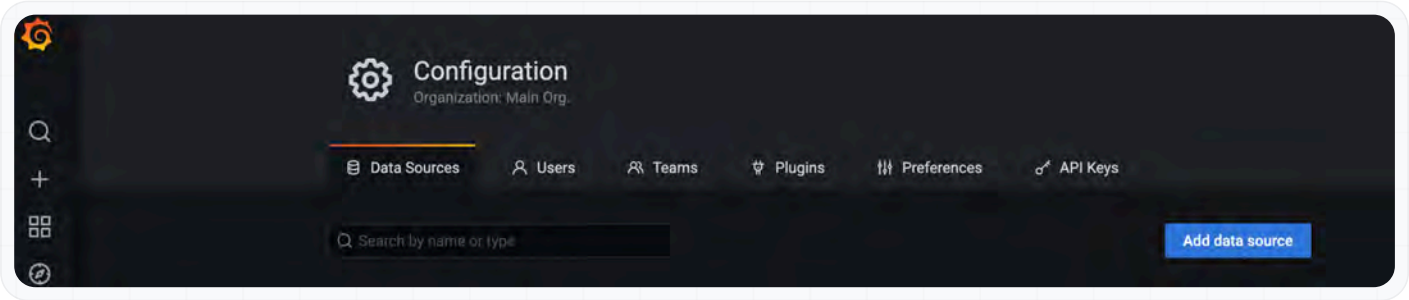
Grafana provides the dashboard and visualization capability for metrics. This tool is used as opposed to Chronograf™, which is traditionally used to complete the TICK (Telegraf, InfluxDB, Chronograf, and Kapacitor) stack. Grafana is favored over Chronograf due to its versatility beyond the InfluxDB ecosystem, as well as higher frequency of contributions and updates from the broader community.

Install Grafana. Follow the Grafana instructions to ensure that configuration options for items such as users, permissions, and logging all correspond to the environment.
https://grafana.com/docs/grafana/latest/installation/

Add InfluxDB as a Data Source

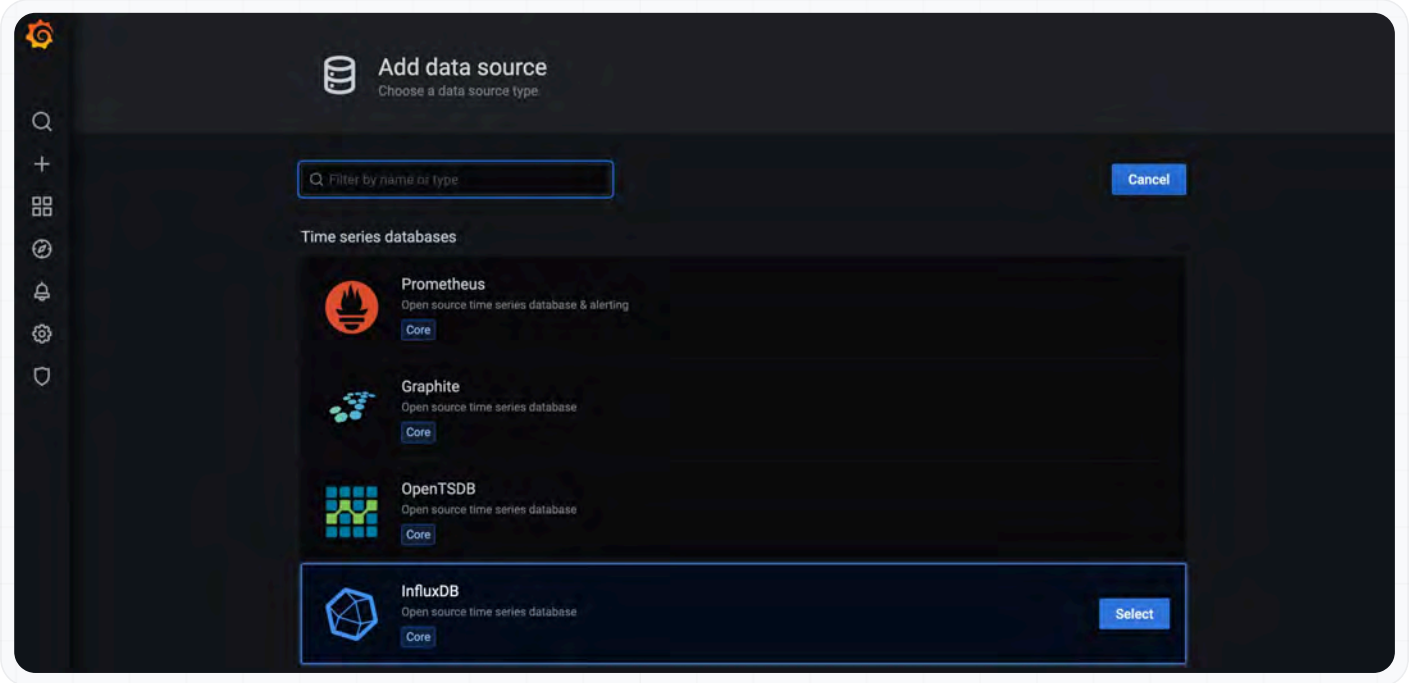
1. Open Grafana in a web browser.

2. In the Grafana UI, select the Settings tab on the left side of the screen. Select **Configuration->Data Sources**.



3. Select **Add data source**.

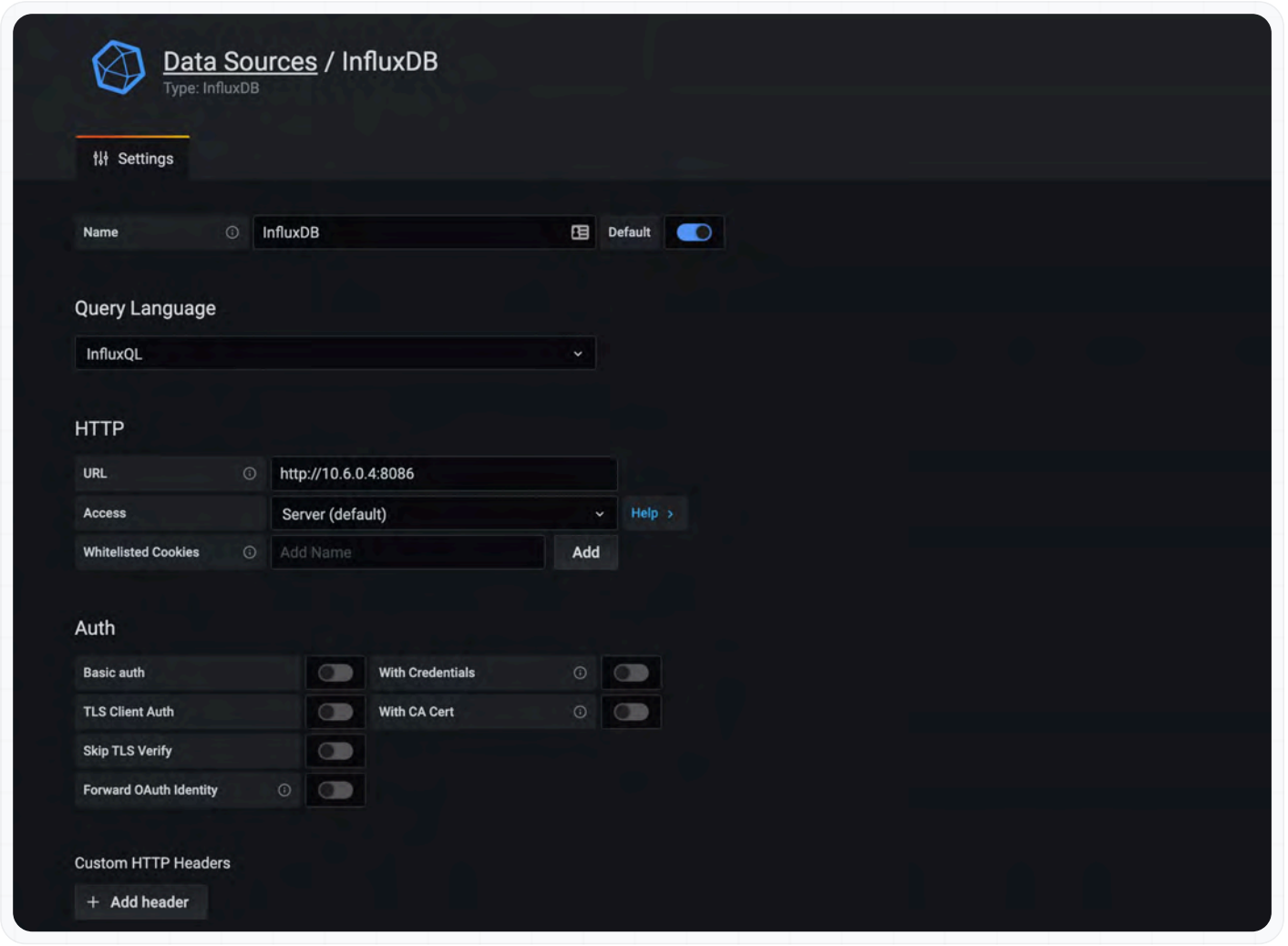
4. Under the data source list, select InfluxDB. Select the **Select** button.



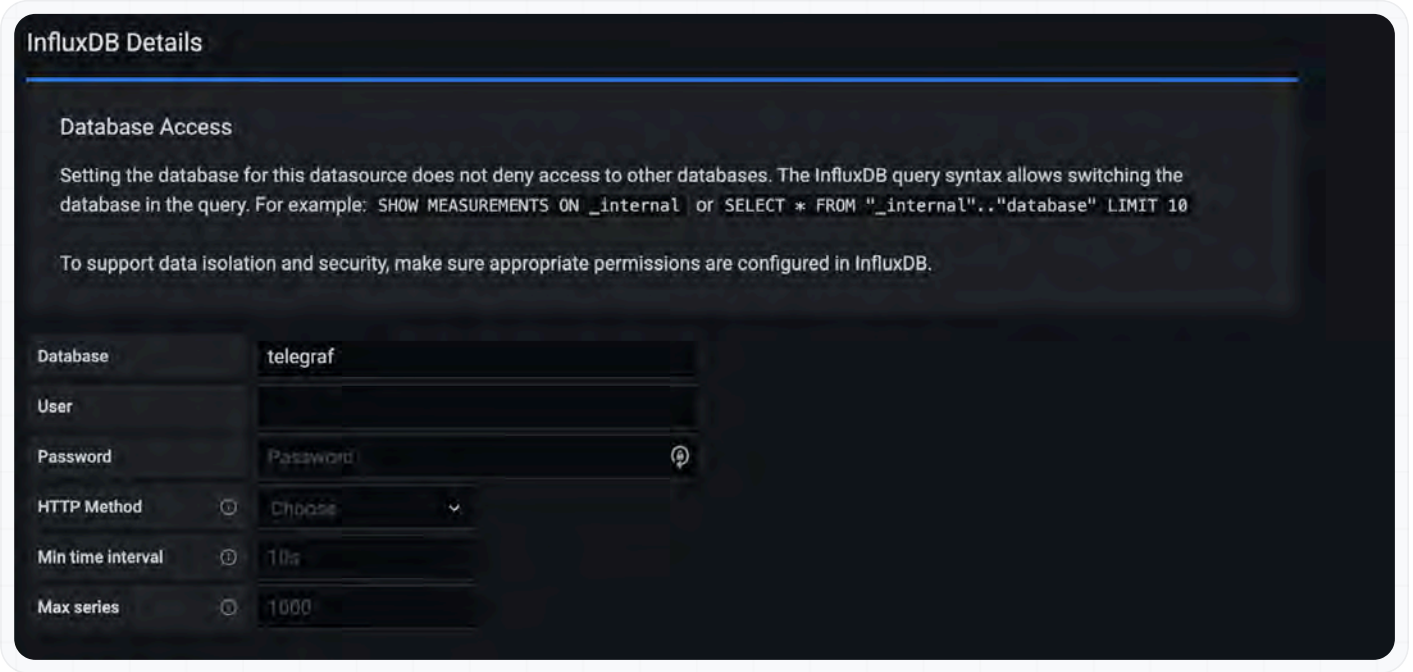
5. Enter all of the pertinent configuration details associated with the InfluxDB instance:

- Ensure the data source is marked as the Default.
- Under the HTTP section, the URL should be the same as the INFLUX_URL used in previous sections.
- The query language should be set to the default (InfluxQL).

© 2026 Ocient, Inc.



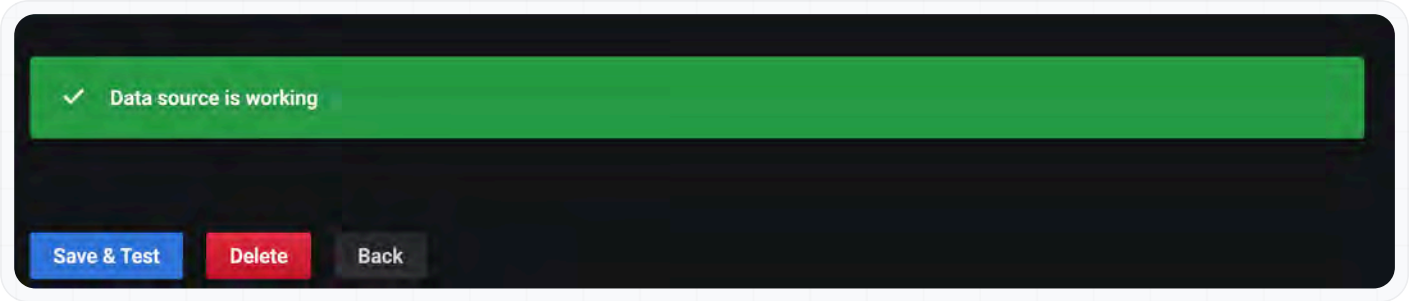
6. Under the InfluxDB Details section, specify `telegraf` as the database name.



7. When the configuration appears to be correct, select **Save and Test**.

8. A green banner should appear that indicates the connection to the data source is working properly. If this message does not display, consult the Grafana documentation or contact Ocient

Support.



Import Dashboards

Ocient provides sample Grafana dashboards upon request. Contact Ocient Support for details.

Within the bundle, relevant contents are located within the `grafana` directory.

> Grafana only allows one file at a time.

1. Open Grafana in a web browser.
2. In the Grafana UI, select the *+ ("Create")* tab on the left side of the screen. Select the **Import** option.
3. On the Import page, select the **Upload JSON File** button.
4. Select the file representing the dashboard you wish to import.
5. On the dashboard summary page, verify the settings are correct. Select **Import** to complete the process.
6. Repeat steps 3-6 for each dashboard that needs to be added.

Related Links

Log Monitoring

Statistics Monitoring

Filebeat is a trademark of Elasticsearch BV.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Monitor Segment Group Transfers

You can perform segment group transfers in the Ocient® System. Use the data in the `sys.segment_group_transfers` system catalog table to monitor this activity.

Segments, Segment Groups, and Segment Group Transfers

A segment is the primary storage unit. Individual segments combine together to form segment groups. For details, see **Core Elements of an Ocient System**.

In a system that has multiple storage clusters, the Ocient System might experience a data imbalance between clusters. In this case, you might need to transfer segment group ownership between clusters. Transferring segment groups enables you to transfer the physically stored segments between clusters. During or after the segment group transfer, you can monitor the activity using the `sys.segment_group_transfers` system catalog table.

Segment Group Transfers System Catalog Table

The database contains the system catalog table named `sys.segment_group_transfers` with these columns:

- `id` — A Universally Unique IDentifier (UUID) that represents the transfer identifier.
- `src_cluster_id` — The UUID of the source cluster.
- `dst_cluster_id` — The UUID of the destination cluster.
- `status` — The current status of the transfer.
  - `ACTIVE` — The transfer is ongoing.
  - `COMPLETE` — The transfer is complete and the segment groups are owned by the destination cluster.
  - `FAILED` — The transfer is not complete and the segment groups are owned by the source cluster. A transfer can be in this state due to an error that prevented the system from transferring segment groups or due to a canceled transfer.

© 2026 Ocient, Inc.

- `segment_group_ids` — An array of uint64 segment group identifiers.
- `src_committed_osn` — The ownership sequence number (OSN) where the source cluster no longer returns the segment groups in the transfer for queries.
- `dst_committed_osn` — The OSN where the destination cluster added the segment groups. The destination cluster does not return the segment groups for queries unless the query uses at least the `src_committed_osn` value for the OSN of the source cluster.

The database removes segment group transfers from the system upon completion.

Related Links

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Inspect the Current Configuration

Ocient® includes a variety of configuration settings that control how the system behaves. The vast majority of these do not change in a typical deployment. However, there is a small set more commonly changed to adjust for different workloads, hardware, and operating requirements.

Configuration is primarily done through DDL or DCL and can be inspected using the **System Catalog Reference**.

You can contact Ocient Support to change configuration settings that apply across the entire system or to a subset of nodes. Each Ocient node reads configuration settings during startup. Therefore, configuration changes require restarts for the new values to take effect. For details, contact Ocient Support.

Inspecting System Configuration

Administrators typically need to know the effective configuration of the system and what has been overridden.

- Configuration Overrides — a list of the overrides applied by administrators
- Effective Configuration — the complete list of all configuration parameters and their current values on each node

Viewing Configuration Overrides

A system catalog table is provided that lists all DDL Configuration Overrides that are applied to the Ocient System. This table includes the configuration target type as well as the scope to which it is applied.

In the event that the scope is not `SYSTEM` (See Advanced Configuration for `CLUSTER` and `NODE` scopes), the `scope_id` will indicate the `cluster_id` or `node_id` to which the configuration override applies.

Example

The following query retrieves all config overrides applied on the system and provides a friendly node name or cluster name where an override is limited in scope:

SQL

```
SELECT
  cfg.scope_type,
  n.name AS node_name,
  c.name AS cluster_name,
  cfg.key,
  cfg.value
FROM
  sys.config cfg
  LEFT JOIN sys.nodes n
  ON n.id = cfg.scope_id and cfg.scope_type = 'NODE'
  LEFT JOIN sys.clusters c
  ON c.id = cfg.scope_id and cfg.scope_type = 'CLUSTER';
```

Output

SQL

```
scope_type                                                    node_name
-------------------------------------------------------------------------------
NODE                                                          sales-lts8
SYSTEM                                                        NULL
CLUSTER                                                       NULL
NODE                                                          sales-lts8
SYSTEM                                                        NULL
...
```

Viewing Effective Configuration Using System Table

A system catalog table is provided that lists the effective configuration of each node.

Example

The following query retrieves the effective overrides for the SQL role on the specified node:

SQL

```sql
SELECT
  n.name,
  nc.key,
  nc.value
FROM
  sys.node_config nc
  JOIN sys.nodes n
  ON nc.node_id = n.id
  WHERE n.name = 'sql0'
  AND key LIKE 'sql%';
```

Output

Text

```
name                                                key
-------------------------------------------------------------------------------
sql0                                                sql.blobstoreParameters.regionAlloca
sql0                                                sql.blobstoreParameters.pageUtilizat
sql0                                                sql.blobstoreParameters.pageUtilizat
sql0                                                sql.blobstoreParameters.maxFileSize
sql0                                                sql.blobstoreParameters.minFileSize
sql0                                                sql.blobstoreParameters.blobPageSize
sql0                                                sql.blobstoreParameters.stripeWidth
sql0                                                sql.blobstoreParameters.coreMask
sql0                                                sql.cacheMinFreeMemoryPercent
sql0                                                sql.cacheMinSize
sql0                                                sql.cacheMinKeepTime
sql0                                                sql.purgeCacheFrequency
sql0                                                sql.pdfReservedMem
sql0                                                sql.numLevels
sql0                                                sql.maxShutdownWaitTime
sql0                                                sql.shortQueryMillis
sql0                                                sql.timeLimit
sql0                                                sql.forceNetworkedVirtualTables
sql0                                                sql.cacheLimit
sql0                                                sql.optimizer
sql0                                                sql.generator
sql0                                                sql.validator
sql0                                                sql.concurrencyTarget
```

Viewing Effective Configuration Using REST Endpoints

Administrators can also view the effective system configuration using the config endpoint, a REST interface available on each node. This endpoint shows every configuration parameter for the node and its current value. It reflects the active values and might not reflect configuration overrides applied since the last node restart.

The config endpoint is found on each node at `http://<node_hostname_or_ip_address>:9090/v1/config` and it returns a JSON representation of the active system configuration of the targeted node.

Example

CURL

```
curl http://10.0.1.7:9090/v1/config
```

Output

JSON


```
{
    "siloType": "xg::runtime::physicalSiloSet_t",
    "tlbfsInfo.enabled": "true",
    "gdsClientParameters_t.batchTimer": "150000000n",
    "gdsClientParameters_t.maxBatchCount": "4096"
    ...
    "operatorvm.vmSiloMask": "0x3",
    "operatorvm.vmParameters.levels": "0x4",
    "operatorvm.vmParameters.statsCacheSize": "4GiB",
    "operatorvm.vmProtocolParameters.routerPollRate": "250000000n",
    "operatorvm.vmProtocolParameters.bufferedDatablockTimeout": "120000000000n"
    ...
    "lts.ioOpParameters.bufferPool.backingSlabSize": "2MiB",
    "lts.ioOpParameters.bufferPool.bufferPools[0].pool.size": "4KiB",
    "lts.ioOpParameters.bufferPool.bufferPools[0].pool.count": "98304",
    "lts.ioOpParameters.lbaCacheSize": "64",
    "lts.targetFillRatio": "1",
    "lts.pdfReservedMem": "4GiB"
    ...
}
```


Related Links

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Node Configuration with the Administrator Role

Nodes with the `admin` role have the additional responsibility to store important metadata for compression and system configuration. In most cases, an OcientAIQ™ Unified Data Platform system should have the majority of SQL Nodes assigned to the `admin` role.

Administrator Node Redundancy

To ensure high availability and consistency, an Ocient® System must always have a minimum number of operational nodes with `admin` privileges.

Redundancy for the system configuration and dictionary compression operates using a majority present model. In a system with `N` nodes configured with the `admin` role, at least `N / 2 + 1` must be present and active for the system to function correctly, where `N` is the total number of nodes, and `N / 2` denotes the division floor (i.e., rounded down to the nearest integer).

This table shows the number of nodes with `admin` privileges that can fail before the system experiences problems. Both columns comprise only nodes assigned the `admin` role, not the entire system node count.

Number of Nodes	Node Failures Tolerated
1	0
2	0
3	1
4	1
5	2
6	2
7	3

© 2026 Ocient, Inc.

⚠ Various core system functions risk failure if the number of nodes running the `admin` role falls below the tolerance threshold. Operations that could fail include:

- System startup
- System configuration
- Query execution
- Data loading

ⓘ The majority present model for nodes with the `admin` role is different than parity encoding used for Foundation Nodes.

For details on parity width for Foundation Nodes, see **Configure Storage Spaces**.

Administrator Node Metadata

This table describes additional metadata files stored by nodes running the `admin` role. If you remove the `admin` role from a node, the system removes these files.

File path	Description
`/var/opt/ocient/metadataStorage.raft`	Definitions for users, network configuration, and database objects.
`/var/opt/ocient/globalDataStorage.raft`	Compression dictionaries for the `COMPRESSION GDC` columns.

Configure Nodes with the Administrator Role

To configure a node to run the `admin` role, execute this SQL statement.

SQL

```
ALTER NODE nodename ADD ROLE admin;
```

It is generally a good practice to have all SQL Nodes in your system assigned to run the `admin` role. You can configure additional nodes with the `admin` role to adjust the level of redundancy to your chosen level.

Related Links

ALTER NODE

Configure Storage Spaces

Expand and Rebalance System

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Manage the Network Configuration of an Ocient System

Control the grouping of SQL Nodes in an Ocient® System by managing network traffic using connectivity pools. For advanced configuration, you can expand a connectivity pool by the number of nodes or you can change the network configuration of the system.

SQL Nodes Belong to Connectivity Pools

A connectivity pool is a pool of potential SQL Nodes for client connection. The pool contains all the required information to connect to a node in that pool. For a description of the metadata of a connectivity pool, see the **CREATE CONNECTIVITY_POOL** statement.

All SQL Nodes in the system must belong to a connectivity pool. If you do not configure the connectivity pools during bootstrapping, the Ocient System assigns each SQL Node to its own connectivity pool named with the node name concatenated by `default` *. *For example, for the node named `sql0`, the system creates a connectivity pool named `sql0_default`. For bootstrapping the Ocient System, see **Ocient System Bootstrapping**.

Connectivity Pool Considerations

For different networks, you can connect a SQL Node to multiple networks at the same time. Connectivity pools enable each SQL Node to specify the IP address and port number for listening in each network.

> Connectivity pools assume that the IP address for each SQL Node remains static. They do not support dynamic IP addresses. If the IP address of your SQL Node changes, you must manually update it by using an **ALTER CONNECTIVITY_POOL ALTER PARTICIPANT** SQL statement.

You can direct network traffic to groups of SQL Nodes by creating connectivity pools with numerous nodes. Also, you can use connectivity pools to isolate SQL Nodes, which

© 2026 Ocient, Inc.

allows you to connect to a subset of nodes and redirect network traffic between them. For example, you can configure the network so that the Ocient System sends only certain SQL queries to certain SQL Nodes.

You connect directly to SQL Nodes. You must know the correct connection information to the SQL Node related to the connectivity pool where you want to create the connection. SQL Nodes only listen to IP addresses and port numbers for the connectivity pools associated with the SQL Nodes. Each node can have multiple connectivity pools assigned to it. Assign different IP addresses to SQL Nodes based on the network (e.g., VPN, internal network). You can configure the configuration to reflect your network, whether the network is on-site, a virtual public cloud, or a remote network.

Connectivity pools can use the same IP addresses and port numbers across pools. A pool can also use the IP address of the source client connection to distribute the SQL queries in the case where the SQL Node connection information matches multiple connectivity pools.

When you connect to a SQL Node using a specific IP address, the node finds all connectivity pools associated with that IP address. The SQL Node then identifies all the other nodes in the connectivity pool and uses that set of nodes as a redirect group to distribute the SQL query. When a SQL Node matches multiple pools by the IP address and port number combination, the node uses the source connection information (or the information you used to connect) to find a more relevant connectivity pool. The SQL Node uses the chosen connectivity pool to redirect SQL queries to the group of nodes.

Initial Configuration Setup

For the initial configuration, a common Ocient System configuration has all SQL Nodes in one connectivity pool. However, you can structure the network of nodes based on your initial needs. You can change your initial configuration setup later as your needs change.

Create your initial configuration of the network using the see the **CREATE CONNECTIVITY_POOL** statement or specify the configuration in the `bootstrap.conf` file. For details about the `bootstrap.conf` file, see **Node Bootstrapping Reference**.

If you do not specify the configuration of the SQL Nodes during the initial Ocient System setup, the system assigns nodes to default connectivity pools. In this case, each node always has a connectivity pool that isolates the node. A common Ocient System configuration has all nodes in one connectivity pool. Thus, you can create a new connectivity pool using the CREATE CONNECTIVITY_POOL statement to add all nodes in the network to it. Then, remove all the default pools using the **DROP CONNECTIVITY_POOL** SQL statement.

For advanced configuration, you can expand the connectivity or change the network configuration of the system.

Expand the Network of an Ocient System

You can expand the connectivity pool of an Ocient System by adding SQL Nodes. Then, remove the automatically generated connectivity pool that is no longer needed.

For example, with an Ocient System that has SQL Nodes `sql0`, `sql1`, and `sq3` in the connectivity pool named `test_pool`, add a new SQL Node `sql4` to the pool that belongs to the default pool named `sql0_default`. The nodes have the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

SQL

```
ALTER CONNECTIVITY_POOL test_pool
    ADD PARTICIPANTS(
       (NODE sql4
        LISTEN_ADDRESS '111.1.1.1'
        LISTEN_PORT 4050
        ADVERTISED_ADDRESS 'localhost'
        ADVERTISED_PORT 4050));
```

Remove the default connectivity pool `sql0_default`.

SQL

```
DROP CONNECTIVITY_POOL IF EXISTS sql0_default;
```

Change the Network Configuration of an Ocient System

You can configure IP addresses that the Ocient System uses to connect to the external client or to other nodes in the system. Configure IP addresses for external client connections using the **ALTER CONNECTIVITY_POOL** SQL statement. Configure IP addresses for connections with other nodes in the system using the **ALTER NODE SET ADDRESS** SQL statement.

If you add a new SQL Node to an existing Ocient System, configure a new system, or add a SQL role using the **ALTER NODE ADD ROLE** SQL statement, you must configure the connectivity pools using the **CONNECTIVITY_POOL** SQL statements. If you add a SQL Node to any connectivity pool, then you must remove the default connectivity pool where the node previously belonged using the **DROP CONNECTIVITY_POOL** SQL statement.

Connectivity Pool Troubleshooting

When you experience issues with connection to the configured connectivity pools, you have two ways to temporarily manipulate the network configuration to address the issues.

You can use the `rolehostd.conf` file. When you cannot establish the connection using the connectivity pool, you can use this configuration in the `rolehostd.conf` file to correct the configuration of the SQL Nodes. This configuration specifies that nodes listen using the IP address `0.0.0.0` and port number `4050` to enable recovery.

Text

```
sql:
   defaultListen: true
```

You must restart the `rolehostd` process after you add this configuration.

Shell

```
sudo systemctl restart rolehostd
```

Or, you can change the IP address of the SQL Nodes in the connectivity pool.

> ⚠ For existing clusters, the Ocient System configures all SQL Nodes to listen to the IP address `0.0.0.0`, which allows client connections to manually create the initial connectivity pools.

Related Links

Install an Ocient System

Ocient System Bootstrapping

Node Bootstrapping Reference

CONNECTIVITY POOL

ALTER NODE ADD ROLE

ALTER NODE REMOVE ROLE

ALTER NODE SET ADDRESS

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Secure Connections Using TLS

The SQL service endpoint can make use of TLS.

The SQL service always has TLS functionality enabled and active.

The database allows connections either unencrypted, or using TLS/SSL. The certificates used to establish SSL connections are located in the directory specified by `certificateStore/path` in the `rolehostd.conf` YAML file, which is `/var/opt/ocient` by default.

The certificate files need to exist on the SQL Nodes for connecting using JDBC.

Text

```
/var/opt/ocient/server.crt
/var/opt/ocient/server.key
/var/opt/ocient/server-ca.crt
```

The server first looks for the certificate or key files named `server.crt` and `server.key` in that directory. Those files should be stored in PEM format. If those files do not exist, the database uses the same `ocient.crt` and `ocient.key` that are used for the internal cluster authentication. The `server.crt` and `server.key` files should be used to provide user-defined certificates to the database.

When these files are in place, restart the services, and they automatically make use of TLS and the specified certificate.

By default, the database allows both encrypted and unencrypted database (JDBC) connections, even if the specified `server.crt` and `server.key` files exist.

Related Links

Connect to Ocient

© 2026 Ocient, Inc.

© 2026 Ocient All Rights Reserved

Multiple Storage Clusters for Loading Data

You can load data in the OcientAIQ™ Unified Data Platform that contains multiple storage clusters. Take these considerations into account for loading in a multi-cluster environment and the related troubleshooting.

Multi-Cluster Environment Considerations

- Loader Nodes send data to all Foundations Nodes in the Ocient® System.
- The distribution of data loaded to each cluster is proportional to its total capacity.
- Every Loader Node loads to every cluster that belongs to the storage space of the target tables. Depending on the location of the tables, the database or schema could contain multiple storage spaces.
- A storage cluster can only belong to one storage space.

> Dropping a cluster is not supported in an Ocient System that contains multiple storage clusters. However, you can drop a cluster in a system that contains only one cluster.

Troubleshooting Considerations

- If a cluster shuts down, the load stops and provides results after the cluster is available again.
- The load is tolerant of Loader Node failures as the load can make progress when at least one Loader Node is still available.
- Before adding or removing a node, storage drive, or cluster, you must stop the load. Stop the LAT and any `CREATE TABLE AS SELECT` or `INSERT INTO SELECT` SQL statements. Otherwise, the load fails.

> Contact Ocient Support for help with a complex multiple storage cluster setup.

Related Links

© 2026 Ocient, Inc.

Core Elements of an Ocient System

Cluster and Node Management

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Alter Default Data Pipeline Behavior

An administrator might configure some parameters that control the behavior of data pipelines to modify the default behavior across the Ocient® System or specific nodes. You can use the `ALTER SYSTEM ALTER CONFIG` SQL statements to control the configuration parameters.

Syntax

```SQL
ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineparameters.parameter_name' 'parameter_va
```

Parameter	Data Type	Description
`parameter_name`	string	The name of the pipeline parameter, either general or specific to the BINARY format.
`parameter_value`	string	The value for the pipeline parameter.

General Parameters

General Parameter Name	Default	Data Type	Required Value and Description
`sql.pipelineparameters.targetTimePerExtractorTask`	600	`UINT64_T`	Target execution time in seconds for `run_extractor` tasks for file loads.
`sql.pipelineparameters.defaultPartitionsPerExtractorTask`	-1	`INT`	Number of partitions per task. If you set this parameter to -1, the number is based on available CPU cores.
`sql.pipelineparameters.defaultExtractorCores`	-1	`INT`	Number of cores to use in a task. If you set this parameter to -1, the number is based on available CPU cores.

Binary Format Parameters

© 2026 Ocient, Inc.

You can modify these parameters to change the default behavior of the BINARY data format extraction.

Binary Format Parameter Name	Default	Data Type	Required Value and Description
`sql.pipelineparameters.extract.binary.defaultEndianness`	`big`	string	The endianness of transforms when you do not specify it in another way. Supported values are `big` or `little`.
`sql.pipelineparameters.extract.binary.defaultCharset`	ibm1047	string	Determines the default character set to use when you load data in the `BINARY` format.
`sql.pipelineparameters.extract.binary.defaultAutoTrimPadding`	true	Boolean	Determines the default for whether a padding character should be trimmed when decoding data in the `BINARY` format to a string.
`sql.pipelineparameters.extract.binary.defaultPaddingCharacter`	' '	`CHAR`	Determines the default padding character to trim from the end of a string when decoding data in the `BINARY` format. This parameter is effective only if the `extract.binary.defaultAutoTrimPadding` parameter is set to `true`.

Examples

Set the target execution time for the extractor task to 600 seconds.

SQL

```
ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineparameters.targetTimePerExtractorTask' 6
```

Set the default endianness of transforms to little endian for the BINARY extract.

SQL

```
ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineparameters.extract.binary.defaultEndiann
```

The configuration values only go into effect after the `ALTER` SQL statement succeeds and you restart the affected nodes.

Related Links

Load Data

Data Types for Data Pipelines

Data Formats for Data Pipelines

Configuration Settings for Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Configuration Settings for Data Pipelines

Data pipeline functionality enables the use of SQL statements to load data. You can use a variety of configuration settings to manage the load. See this table for the settings that you can use.

Configuration Setting Parameter Name	Default	Data Type	Description
`streamloader.extractorEngineParameters.enabled`	`true`	BOOLEAN	Whether the Loader Node attempts to bootstrap the extractor Java® process for data pipelines on start-up. If you set this parameter to `true`, the Loader Node does not become active until the Java loading process has successfully started.
`streamloader.extractorEngineParameters.jvmMemoryConfiguration`	- `32g` for `initialHeap` - `64g` for `maxHeap` - `64g` for `maxDirect`	VARCHAR	The JVM® memory configuration to use for the extractor Java process. Use strings representing data sizes in SI format: 1 KiB, 5 MiB, etc. You must set the three memory options in the same statement. **Example** `ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.jvmMemoryConfiguration.initialHeap' = '32g', 'streamloader.extractorEngineParameters.jvmMemoryConfiguration.maxHeap' = '64g', 'streamloader.extractorEngineParameters.jvmMemoryConfiguration.maxDirect' = '64g';`
`streamloader.extractorEngineParameters.restPort`	8090	INT	Use this parameter with the `streamloader.extractorEngineParameters.portOffset` parameter to form the REST port for the extractor. (External monitoring services like Telegraf™ can query the REST

© 2026 Ocient, Inc.

Configuration Setting Parameter Name	Default	Data Type	Description
			server for loading process metrics.)
`streamloader.extractorEngineParameters.portOffset`	0	INT	Use this parameter with the `streamloader.extractorEngineParameters.restPort` parameter to form the REST port of the extractor. (External monitoring services like Telegraf can query the REST server for loading process metrics.)
`streamloader.extractorEngineParameters.configurationOption.expect.empty.file.list`	`false`	BOOLEAN	Set this value to `false` to fail a file source data pipeline that finds no files to load instead of the load completing successfully with no data loaded.
`streamloader.extractorEngineParameters.configurationOption.pipeline.preview.rows.limit`	1000	INT	The maximum number of rows a `PREVIEW PIPELINE` SQL statement can return. For example, when you use this statement, if the specified `LIMIT` value exceeds this number, then the loading process throws an error.
`streamloader.extractorEngineParameters.configurationOption.engine.transform.udt.jarRootDirectory`	`/opt/ocient/current/lib/extractorengine_udt`	VARCHAR	The absolute path to the directory containing JARs for data pipeline functions. To install and enable a third-party library for using data pipeline functions, install the JAR package at this location on all Loader Nodes. Then, add the chosen fully-qualified class name to the IMPORT clause of the data pipeline function. For details, see **CREATE OR REPLACE PIPELINE FUNCTION**.
`streamloader.extractorEngineParameters.configurationOption.engine.external.jdbc.jarRootDirectory`	`/opt/ocient/current/lib/extractorengine_jdbc`	VARCHAR	The absolute path to the directory containing JARs for external source lookup functionality. For details, see **Load Data from External Sources in Data Pipelines**.

Configuration Setting Parameter Name	Default	Data Type	Description
`streamloader.extractorEngineParameters.configurationOption.source.record.max.size`	Dynamically calculated value at start-up based on the number of processors and amount of memory available to the JVM	BIGINT	The maximum source record size, in bytes, that the loading process tolerates before throwing an error.
`streamloader.extractorEngineParameters.configurationOption.s3.region`	`us-east-1`	VARCHAR	The default region for Amazon® Web Services℠ (AWS℠) S3 file sources.
`streamloader.extractorEngineParameters.configurationOption.s3.force.path.style.access`	`false`	BOOLEAN	Set this value to `true` to use path-style access for all S3 file sources. The default is a virtual-hosted style.
`streamloader.extractorEngineParameters.configurationOption.s3.force.path.style.access.if.endpoint.overridden`	`true`	BOOLEAN	Set this value to `true` to use path-style access for S3 file sources that specify endpoints. (The system still uses virtual-hosted style access for sources without endpoint overrides unless you set `streamloader.extractorEngineParameters.configurationOption.s3.force.path.style.access` to `true`).
`streamloader.extractorEngineParameters.configurationOption.s3.retries.count`	10	INT	The number of retries for the S3 file source client.
`streamloader.extractorEngineParameters.configurationOption.s3.netty.read.timeout.seconds`	0	INT	The timeout value for read operations, in seconds, for the S3 file source client. `0` means do not perform a timeout.
`streamloader.extractorEngineParameters.configurationOption.s3.netty.max.concurrency`	50	INT	Maximum number of allowed concurrent requests for the S3 file source client.
`streamloader.extractorEngineParameters.configurationOption.awss`	1	INT	The base amount of time, in seconds, for calculating the time the S3 file source client

Configuration Setting Parameter Name	Default	Data Type	Description
`dk.backoff.strategy.base.delay.seconds`			waits before retrying a failed request. The range of values that the calculated time can achieve increases exponentially with each failure. You can set the maximum value using the `streamloader.extractorEngineParameters.configurationOption.awssdk.backoff.strategy.max.backoff.seconds` parameter.
`streamloader.extractorEngineParameters.configurationOption.awssdk.backoff.strategy.max.backoff.seconds`	20	INT	The maximum amount of time the S3 file source client waits before retrying a failed request.
`streamloader.extractorEngineParameters.configurationOption.awssdk.max.pending.connection.acquires`	20000	INT	The maximum number of pending connection acquires that the S3 file source client allows.
`streamloader.extractorEngineParameters.configurationOption.awssdk.connection.timeout.seconds`	10	INT	The amount of time, in seconds, that the S3 file source client waits when initially establishing a connection.
`streamloader.extractorEngineParameters.configurationOption.awssdk.connection.max.idle.timeout.seconds`	300	INT	The maximum amount of time, in seconds, that the system allows a connection established by the S3 file source client to remain open while idle.
`streamloader.extractorEngineParameters.configurationOption.awssdk.connection.acquisition.timeout.seconds`	300	INT	The amount of time, in seconds, that the S3 file source client waits when acquiring a connection from the pool.
`streamloader.extractorEngineParameters.configurationOption.kafka.poll.timeout.msec`	1000	BIGINT	The amount of time, in milliseconds, that a consumer for a pipeline with an Apache® Kafka® source waits for records to become available during a poll operation.

Configuration Setting Parameter Name	Default	Data Type	Description
`streamloader.extractorEngineParameters.configurationOption.kafka.assign.timeout.msec`	20000	BIGINT	The amount of time, in milliseconds, that a `PREVIEW PIPELINE` SQL statement waits for partitions to be assigned to the consumer on a pipeline with a Kafka source. If the system does not assign any partitions when the timeout elapses, then the `PREVIEW PIPELINE` statement terminates without data processing.
`streamloader.extractorEngineParameters.configurationOption.kafka.activate.timeout.msec`	5000	BIGINT	The amount of time, in milliseconds, that a `PREVIEW PIPELINE` SQL statement waits for a partition to receive records on a pipeline with a Kafka source.
`streamloader.extractorEngineParameters.configurationOption.arrow.max.native.memory.usage.bytes`	23756537856	BIGINT	The default maximum native memory usage, in bytes, for the Apache® Parquet™ reader.
`streamloader.extractorEngineParameters.configurationOption.arrow.max.java.buffer.memory.usage.bytes`	15837691904	BIGINT	The default maximum buffer memory usage, in bytes, for the Parquet reader.
`streamloader.extractorEngineParameters.configurationOption.filesystem.access.directories`	`/tmp`	VARCHAR	Comma-separated list of directories where you can load server file system data. Each directory must exist. All files must be located within one of the directories. By default, you can load from the `/tmp` directory.

Related Links

Load Data

Alter Default Data Pipeline Behavior

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Configure Data Pipeline Logging

You can troubleshoot data pipelines using the standard logging mechanism in Ocient®.

Set these log levels in consultation with Ocient Support because this setting might impact performance.

Modify the Level of Logging

Execute these two SQL statements in this order to bump up the log message.

First, set the log level.

```sql
ALTER SYSTEM ALTER LOG LEVEL SET 'log4j' 'EDEBUG';
```

Then, set the level for the logging configuration.

```sql
ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.loggingConfig
```

If the log level of the first SQL statement is lower than the log level of the second SQL statement, the system loses log entries.

It is also possible to repeat the second SQL statement for different package names, such as:

- `ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.loggingConfig.com.ocient.extractor.allowLevel' 'Info';`
- `ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.loggingConfig.com.ocient.extractor.engine.allowLevel' 'Trace';`

© 2026 Ocient, Inc.

Valid Log Levels

This table provides an overview of the log levels you can specify for the `ALTER SYSTEM ALTER LOG LEVEL SET` SQL statements with the `streamloader.extractorEngineParameters.loggingConfig` keys. The necessary log level for `log4j` must be set accordingly to at least the value specified in the column for the `log4j` level.

Log Level of `streamloader.extractorEngineParameters.loggingConfig`	**Derived Level for** `log4j`
FATAL	ERROR
ERROR	ERROR
WARN	WARN
INFO	INFO
DEBUG	DEBUG
TRACE	EDEBUG

Reset the Log Level

To reset the log level after you reproduce a problem, execute these SQL statements.

SQL

```
ALTER SYSTEM ALTER CONFIG RESET 'streamloader.extractorEngineParameters.loggingConf
ALTER SYSTEM ALTER CONFIG RESET 'log4j';
```

Related Links

Monitor Data Pipelines

Data Pipeline Loading Errors

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Air-Gapped Environment with an Ocient System

An air gap is a security measure that physically isolates a computer or network from any other network, including the internet. This isolation removes wired or wireless connections between the isolated system and any external and potentially unsecured networks. Organizations that handle highly sensitive information, such as government agencies, financial institutions, and critical infrastructure operators, use air-gapped environments to protect against various security incidents.

Ocient® supports using an Ocient System in an air-gapped environment without any special configuration. Here, you can find the general workflow for using an Ocient System in an air-gapped environment and the required installation steps.

Work with an Ocient System in an Air-Gapped Environment

This workflow shows the high-level steps to set up an Ocient System in an air-gapped environment.

1. Ensure that your drive firmware and operating system is up to date.
2. Install and configure an Ocient System in a connected environment where you can download the required files from the internet.
3. Disassemble the cluster.
4. Pack and deliver the hardware to your secured location.
5. Reassign a new IP address.
6. Reassign a new hostname.

© 2026 Ocient, Inc.

7 Bootstrap the system again.

Install an Ocient System in an Air-Gapped Environment

Follow these steps to install the Ocient System in an air-gapped environment. These steps assume that you have met the system requirements for installation.

1 **Bootstrap the First Node**

Bootstrap the Node

Connect to the initial SQL Node with the username and password of your server and the IP address of your node. This example connects as the administrator `admin` to the IP Address `10.10.10.10`.

Shell

```
ssh admin@10.10.10.10
```

Use your preferred text editor with `sudo` to create the `/var/opt/ocient/bootstrap.conf` file as root with these contents.

`/var/opt/ocient/bootstrap.conf` example

YAML

```
initialSystem: true
```

Start the database.

Shell

```
sudo systemctl start rolehostd
```

Verify that the node and the service is active by executing this status command.

Shell

```
systemctl status rolehostd

rolehostd.service - Rolehostd daemon startup
     Loaded: loaded (/etc/systemd/system/rolehostd.service; enabled; vendor
     Active: active (running) since Wed 2022-01-26 23:31:36 UTC; 7s ago
     ...
```

If the `rolehostd` service is running, you can also check the Ocient logs on your node. Search and ensure there are no [ERROR] log messages.

Shell

```
tail -f /var/opt/ocient/log/rolehostd.log
```

Verify Connection to the SQL Node

At this point, you have a running database with a single node. You should be able to connect to the database using JDBC or `pyocient` and execute commands.

Every new system starts with a `system` database. To connect to a new system, use the username and password configured in the `bootstrap.conf` file or the username `admin@system` and password `admin` if none were provided.

For example, assume your node named `sql0` has an IP address of `10.10.0.1`. Use the JDBC driver CLI to connect with this connection string.

Shell

```
connect to jdbc:ocient://10.10.0.1:4050/system user "myuser@system" using "
```

To see the roles running on the single node, execute this query.

SQL

```sql
select name, operational_status, software_version, array_agg(service_role_t
        from sys.node_status as ns
        left join sys.nodes as n  on ns.node_id = id
        left join sys.service_roles as sr on sr.node_id = n.id
        group by name, operational_status,software_version;
```

The initial node should be listed as running the `sql`, `admin`, `health`, and `operatorvm` roles. If all of these roles are present and the node is Active, you can proceed to the next step to bootstrap the remaining nodes.

2 Bootstrap the Remaining Nodes

Performing the bootstrapping process on the remaining nodes is identical on all nodes. The remaining nodes can be bootstrapped in any order.

On each node, log in using SSH and use your text editor with `sudo` to create the file `/var/opt/ocient/bootstrap.conf` that contains this text by replacing `<FIRST_NODE_IP_ADDRESS>` with the IP address of the initial node you created in Step 1.

`/var/opt/ocient/bootstrap.conf` example

YAML

```yaml
adminHost: <FIRST_NODE_ADDRESS>
```

`<FIRST_NODE_IP_ADDRESS>` is the DNS name or IP address of the initial node.

> You can obtain the IP Address of the initial node by executing `ifconfig` on that node.

> If the password for the system administrator has changed, set the correct username `adminUserName` and password `adminPassword` in the bootstrap configuration file `bootstrap.conf`.

© 2026 Ocient, Inc.

On each node, start the database.

Shell

```
sudo systemctl start rolehostd
```

> ⚠ When you replace Foundation Nodes, the Ocient System removes the prior node after the creation of the new node. Some queries of the system catalog tables might not return results until the prior node is removed.

At this point, all the remaining nodes are not configured with any roles. After all nodes have been started, you should see them when you execute this query with only the `health` role listed. This query uses the `sys.node_status`, `sys.nodes`, and `sys.service_roles` system catalog tables to retrieve node information for the node name, operation status, version, and all service role types. The query uses the ARRAY_AGG function to retrieve the service role type for all rows.

SQL

```
SELECT name, operational_status, software_version, array_agg(service_role_t
    FROM sys.node_status AS ns
    LEFT JOIN sys.nodes AS n ON ns.node_id = id
    LEFT JOIN sys.service_roles AS sr ON sr.node_id = n.id
    GROUP BY name, operational_status,software_version;
```

Remove a Cluster in an Air-Gapped Environment

Use this information and workflow to remove a cluster by shutting down all Ocient processes and erasing the drives and data.

> ⓘ The system administrator must download the `sedutil-cli` utility used in this workflow.

You can classify the drives in an Ocient System in two categories based on their usage.

- Operating System (OS) drives: These drives contain the installation of the operating system and software, including the OcientAIQ™ Unified Data Platform. A system might have a single physical OS drive or an OS installed on a RAID disk created by using more than one drive. The OS drives on nodes with the administrator role (Metadata and possibly SQL Nodes) also store configuration information related to the Ocient System that can include:
  - Node names
  - Node IP addresses
  - User data mapping information for compressed columns
  - Encryption keys for the data drives (unless the keys are under the control of an external key management system)
- Data drives: The data drives are present in all types of nodes except the node running only the administrator role. On the Foundation Node, the data drives store tables. On SQL Nodes, the data drives store transient query information. And, on Loader Nodes, the data drives store transient loading information.

The data drives in the Ocient System are exclusively NVMe drives. The OS drives can be NVMe or SSD. These drives can support The Computing Group (TCG) Opal Specification or not support it. The type of drive (Opal-supported or not) determines how the system removes all data on the drive. Follow these steps to remove a cluster for Opal-supported drives. To remove data for Opal-supported drives, follow these steps.

⚠ These steps irreversibly remove the data. Follow these steps after you ensure that you no longer need the applicable data from the system.

1. Stop the Ocient processes running on the nodes using these commands.

   For Loader Nodes, use this command. The command combines two commands that stop all processes related to loading.

   Shell

   ```
   sudo systemctl disable lat && sudo systemctl stop lat
   ```

For all types of nodes, use this command. The command combines two commands where the first command stops processing on all nodes and the second command stops the main Ocient System process.

```shell
sudo systemctl disable rolehostd && sudo systemctl kill -s SIGKILL rolehost
```

2 Bind the data drives to an NVMe driver so that the drives become visible to the OS.

```shell
sudo /opt/ocient/scripts/nvme-driver-util.sh bind-nvme
```

3 Run this command for each data drive. The impacted drives are the drives displayed as their association change from `uio` drivers to `nvme` drivers. You can see the `nvme` drive association by running the `/opt/ocient/scripts/nvme-driver-util.sh` script. All data drives are erased.

> You can move drives to another node with the `sedutil-cli` utility if the nodes are already powered off and you cannot power them on. Use the same command to erase the drive: `sedutil-cli -n --revertTPer admin /dev/nvmeXn1` .

```shell
sedutil-cli -n --revertTPer admin /dev/nvmeXn1
```

> You cannot securely erase the OS drives while the OS is running. You can either use the secure erase facility from BIOS or move drives to an external host with a utility approved by your organization for the remove operation.

Related Links

Install an Ocient System

Ocient System Bootstrapping

Set Up System Monitoring with the TIG Stack and Kapacitor

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Simulator

Ocient® provides a simulated database system upon request for development or local workloads with no performance requirements.

This simulator system is a lightweight package that emulates a complete Ocient System with these features:

- SQL port for connection using any of the supported connectors from Ocient.
- No NVMe drive or Linux® huge page requirements.

> To request access to an Ocient simulator, please contact Ocient Support.

Prerequisites

The Ocient simulator has these hardware and software requirements:

Hardware

- 64GB system RAM
- 64-bit Intel® architecture CPU (x86)
- Allocated disk storage equal to twice the size of the data to load.

> If you run the Ocient Simulator on a virtual machine, you must configure your virtual machine CPU to support AVX2 instructions by setting the CPU type to `host`.

Software

- Docker® **Engine** version 20.10.10 or greater with **Docker Compose** version 1.9.0 or greater.

Install the Ocient Simulator

Ocient distributes its simulator as the tarball `simulator_archive.tar`. Upon receiving the file, you can extract it and load the container images into the Docker daemon with the

provided `load_all_images.sh` file.

> Some file names might differ based on the simulator release version.

1. From a terminal window, create a new empty directory.

```
Shell
mkdir simulator_workspace
```

And then change the directory to the new one.

```
Shell
cd simulator_workspace
```

2. Copy or move the tarball file into the new directory.

3. Extract the file.

```
Shell
tar xf simulator_archive.tar
```

The simulator package contains these files.

File Name	Description
`configure.sql`	The system configuration script that bootstraps the simulator.
`docker-compose.yml`	The **Docker Compose file** that defines the containers that constitute the simulator system.
`lat_client.tar`, `lat_simulator.tar`, `ocient_jdbc_image.tar`, `rolehostd_simulator.tar`	These Docker images comprise various components of the Ocient System.

File Name	Description
`jdbc.sh`	The JDBC CLI launch script for connection to the SQL interface of the simulator.
`lat_client.sh`	The Loading and Transformation (LAT) client launch script. The LAT client is for backward compatibility.
`load_all_images.sh`	This script loads all the tarball files into the Docker daemon.

4 Load the Ocient System tarball files for the simulator.

Shell

```
./load_all_images.sh
```

If you do not have access privileges to the Docker daemon, you might receive the `Permission denied` error when you run `docker` commands. Use `sudo` or add your user to the Docker group.

This output shows a successful load operation. The exact output of your simulator might differ depending on what version you download.

© 2026 Ocient, Inc.

Shell

```
a0c3d7b00a1e: Loading layer [==================================>]  59.26MB/
5d75db409355: Loading layer [==================================>]  329.6MB/
b4f0ab1562a2: Loading layer [==================================>]  3.072kB/
6b6be244d498: Loading layer [==================================>]  94.76MB/
Loaded image: ocient/lat/simulator:2.3.0
fb4617b63504: Loading layer [==================================>]  29.39MB/
f5745a52e122: Loading layer [==================================>]  4.608kB/
d2d33759728d: Loading layer [==================================>]  12MB/12M
f6533a5b289e: Loading layer [==================================>]  10.24kB/
a3f69486f426: Loading layer [==================================>]  23.64MB/
84ff92691f90: Loading layer [==================================>]  10.24kB/
Loaded image: ocient/lat_client:2.3.0
Loaded image: ocient/jdbc:2.100
475a54c2a93d: Loading layer [==================================>]  65.52MB/
314fd346d653: Loading layer [==================================>]  1.32GB/1
Loaded image: ocient/rolehostd/simulator:23.0.0-1680124029
```

Start and Configure the Cluster

Each Ocient simulator consists of a series of nodes running in their own Docker containers. The `docker compose` command enables networking, general administration, and orchestration of these nodes.

> The commands you use based on this section might differ with your Docker version.
>
> These steps use the `docker compose` command for the simulator setup. If you are using an older version of Docker with `docker-compose` (a dash instead of a space), use that instead.

1. Start the cluster using the `docker compose` command.

Shell

```
docker compose up -d
```

The cluster startup displays output that is similar to this output.

Shell

```
[+] Running 7/7
 ⠿ Network simulator_workspace_ocientdbnet      Created                  0.
 ⠿ Container simulator_workspace-sql-1          Started                 20.
 ⠿ Container simulator_workspace-foundation0-1  Started                 15.
 ⠿ Container simulator_workspace-foundation2-1  Started                 17.
 ⠿ Container simulator_workspace-foundation1-1  Started                 16.
 ⠿ Container simulator_workspace-loader-1       Started                 14.
 ⠿ Container simulator_workspace-lat-1          Started                  7.
```

Ensure that you run every command from the same directory where you extracted the `simulator_archive.tar.gz` archive.

2 Using the bundled JDBC CLI, connect to the cluster and run the contents of the `configure.sql` file that is distributed with the simulator.

The JDBC CLI is bundled with the simulator as the `jdbc.sh` script. Execute this script with no arguments after starting the cluster.

Shell

```
./jdbc.sh
```

After loading the JDBC CLI, the system prompts you to enter a username and password. The default simulator username and password are `admin@system` and `admin` .

You can also pass these credentials later in a CONNECT statement, for example:

```
CONNECT TO jdbc:ocient://sql:4050/system USER admin@system USING admin;
```

The interface switches to the JDBC CLI interactive mode, as indicated by the `Ocient>` prompt. Execute the configuration file.

JDBC CLI

```
source configure.sql;
```

The configuration displays output similar to this.

Shell

```
Connected to jdbc:ocient://sql:4050/system
Modified 0 rows
Modified 0 rows
Modified 0 rows
Warning: Node must be restarted for changes to take effect (the streamloader :
Modified 0 rows
Modified 0 rows
Modified 0 rows
```

3 After completing its modifications, the simulator warns that it needs to restart for the changes to take effect. Quit the interactive mode.

JDBC CLI

```
quit
```

Restart the cluster.

Shell

```
docker compose restart
```

4 After the nodes restart, verify that the simulator has started and the `test` database is accessible by connecting to it with the JDBC CLI.

Launch the JDBC connector again.

Shell

```
./jdbc.sh
```

Connect to the `test` database in the JDBC CLI.

JDBC CLI

```
CONNECT TO jdbc:ocient://sql:4050/test;
```

The system displays a confirmation message upon a successful connection.

JDBC CLI

```
Connected to jdbc:ocient://sql:4050/test
```

Load Data

The simulator cluster includes a Loader Node, which can load data from external sources, such as an Amazon® Web Services℠ (AWS℠) S3 bucket. This section demonstrates the creation of a basic table and pipeline, which you can then use to load data from a sample data set into a simulator database.

If you are not already in the JDBC CLI interactive mode, start it and connect to the `test` database using the steps in **Start and Configure the Cluster**.

1. Create a table to load. Enter this `CREATE TABLE` SQL statement. This table contains columns with various data types available in the Ocient System.

SQL

```
CREATE TABLE loading.data_type_coverage (
    col_bigint BIGINT NULL,
    col_binary BINARY(2) NULL,
    col_boolean BOOL NULL,
    col_char CHAR(4) NULL,
    col_date DATE NULL,
    col_decimal DECIMAL(18,4) NULL,
    col_double DOUBLE NULL,
    col_float FLOAT NULL,
    col_int INT NULL,
    col_int_array INT[] NULL,
    col_ipv4 IPV4 NULL,
    col_ip IP NULL,
    col_smallint SMALLINT NULL,
    col_point POINT NULL,
    col_linestring LINESTRING NULL,
    col_polygon POLYGON NULL,
    col_time TIME NULL,
    col_timestamp TIMESTAMP NULL,
    col_tinyint TINYINT NULL,
    col_tuple tuple<<INT, VARCHAR(255)>> NULL,
    col_uuid UUID NULL,
    col_varbinary VARBINARY(255) NULL,
    col_varchar VARCHAR(255) NULL
);
```

2 Create a data pipeline to load the data. Enter this `CREATE PIPELINE` SQL statement.

This pipeline loads gzip-compressed CSV files from an S3 bucket into the `data_type_coverage` table. The pipeline reads files matching the `**output*.csv.gz` pattern from the `ocient-docs` bucket, parses them as comma-delimited records with quoted fields and a header row, and maps each named header field to a target column.

The `EXTRACT` clause includes array and tuple delimiter settings so the parser can correctly interpret complex-type values embedded in the CSV data. The `SELECT` clause applies explicit type conversions where needed — for example, ensuring that

the `col_double` and `col_float` columns properly handle `NaN` values and stripping formatting characters from the timestamp field.

For details on Ocient data pipelines, see **Data Pipelines**.

SQL

```sql
CREATE PIPELINE data_type_coverage_pipeline
SOURCE S3
    ENDPOINT 'https://s3.us-east-1.amazonaws.com'
    BUCKET 'ocient-docs'
    FILTER 'all_data_types/small/**output*.csv.gz'
    COMPRESSION_METHOD 'gzip'
EXTRACT
    FORMAT CSV
    NUM_HEADER_LINES 1
    FIELD_OPTIONALLY_ENCLOSED_BY '"'
    OPEN_OBJECT '('
    CLOSE_OBJECT ')'
INTO loading.data_type_coverage
SELECT
    $1 AS col_bigint,
    $2 AS col_binary,
    $3 AS col_boolean,
    $4 AS col_char,
    $5 AS col_date,
    DECIMAL($6, 18, 4) AS col_decimal,
    DOUBLE(IF(LOWER($7) = 'nan', NULL, $7)) AS col_double,
    FLOAT(IF(LOWER($8) = 'nan', NULL, $8)) AS col_float,
    $9 AS col_int,
    INT[]($10) AS col_int_array,
    $11 AS col_ipv4,
    $12 AS col_ip,
    $13 AS col_smallint,
    $14 AS col_point,
    $15 AS col_linestring,
    $16 AS col_polygon,
    $17 AS col_time,
    REPLACE(REPLACE($18, 'T', ' '), 'Z', '')
        AS col_timestamp,
    $19 AS col_tinyint,
    $20 AS col_tuple,
    $21 AS col_uuid,
    VARBINARY(IF($22 IS NULL, NULL,
        IF(LENGTH($22) % 2 = 1, CONCAT('0', $22), $22)))
        AS col_varbinary,
    $23 AS col_varchar;
```

3 Start the pipeline to load the data.

```sql
START PIPELINE data_type_coverage_pipeline;
```

The sample data set can take about 30 seconds to fully load.

4 Verify the `data_type_coverage` table is present.

```shell
LIST TABLES;
```

Verify the row count in the table using this SQL statement.

```sql
SELECT COUNT(*) FROM loading.data_type_coverage;
```

If the load operation is complete, the output table shows a count of about 100,000 rows.

```text
count(*)
--------------------
100000

Fetched 1 row
```

5 When you are confident your data is fully loaded, you can delete the pipeline using the `DROP PIPELINE` SQL statement.

© 2026 Ocient, Inc.

```sql
DROP PIPELINE data_type_coverage_pipeline;
```

Now that your Ocient simulator is loaded with data, you can add integrations for user interfaces, visualizations, and other applications. For details, see **Ocient Integrations**.

Container Configuration

This section explains the configuration setup and options for the Ocient simulator.

Ports

The container exposes additional ports that can be useful depending on the use case for the system.

Exposed Port	Description
4050	The JDBC port of the cluster.
8080	Ocient LAT service. When you load new data, the LAT client connects to this port.
9090 and 9091	The REST interfaces for the SQL and Foundation (LTS) Node processes, respectively.

Environment Variables

You can set these environment variables to affect certain aspects of the Ocient simulator.

Environment variable	Description
LAT_MEMORY and LAT_MEMORY_DIRECT	Configures the memory used by LAT when it is running. For details about these environment variables, see the **LAT Service Configuration** page. The LAT Node is the only node in the cluster affected by this variable.

Example

YAML

```yaml
services:
  lat:
    environment:
      LAT_MEMORY: '32G'
```

Resource Limits (CPU, Memory, Disk Usage)

Docker supports a variety of options to limit container access to host resources. Refer to the Docker **Runtime options with Memory, CPUs, and GPUs** page for the full list of options. This section explains how constraints on specific host resources affect the Ocient simulator.

> In all of these code examples, replace `IMAGE` with the filename of your Docker image.

> The Ocient System applies Docker resource configuration changes individually on each of the nodes in the `docker-compose.yml` file (such as `sql`, `foundation`, `loader`, `lat`).

CPU

CPU can be safely limited without adverse effects. Most workloads benefit from having multiple CPU cores available (four or more).

For example, to add CPU limits to the `sql` node, add this text to the `docker-compose.yml` file in the simulator package. Replace the `'4'` with the CPU limit that you want.

YAML

```yaml
services:
  sql:
    image: IMAGE
    deploy:
      resources:
        limits:
          cpus: '4'
```

Memory (RAM)

The Ocient simulator requires sufficient RAM to execute all workloads. The system can be unstable in low or out-of-memory conditions. If you need to limit the memory of the running Docker container, it is recommended to benchmark the container memory usage on your workload and provide at least 1GB of RAM above the maximum observed.

> ⚠ When limiting memory of the Ocient Simulator, do not use the `--oom-kill-disable` option, otherwise the database can become unresponsive for a significant period of time after running out of memory.

To add memory limits to the `sql` node, add this text to the `docker-compose.yml` file in the simulator package. Replace the `8g` with the memory limit that you want.

YAML

```yaml
services:
  sql:
    image: IMAGE
    deploy:
      resources:
        limits:
          memory: 8g
```

Disk Usage

When running the simulator without mounted data, you can specify the size in bytes of the data files created. The data files must be large enough to fit any data sets that you load. The file size defaults to 2GiB.

To specify a different file size, you can configure the rolehostd startup command in the `docker-compose.yml` file. Add the command option `loopbackDeviceSize` with a new value to configure a new file size.

In this example, the file size on the `foundation0` node is configured to 16GiB.

YAML

```yaml
services:
  sql:
    image: IMAGE
    command: "--bootstrap '{adminHost: \"sql\", ipAddress: \"foundation0\", loopbac
```

If you want to specify a new file size value, you must do it before you create your cluster.

While you cannot resize files on a running cluster, you can change this value by deleting and recreating your cluster. This process is irreversible and erases any existing databases and tables.

To resize file on an existing cluster:

1. Stop your cluster by running `docker compose down`.
2. Clear the `volumes` directory by running `rm -rf volumes/`.
3. If you have not yet edited the `docker-compose.yml` file, edit the file to include new `loopbackDeviceSize` values.
4. Restart your cluster by running `docker compose up -d`.

Simulator Logging

The Ocient simulator sends all logs to stdout and stderr as well as any logging driver configured for the host Docker installation or the container. This logging allows for easy inspection of container logs using the `docker logs` command.

For details on logging with Docker, consult the **logging section** in the Docker documentation.

Setting Logging Levels

You can modify the logging level for each node in the cluster by configuring the `rolehostd` startup command in the `docker-compose.yml` file. Add the command option `logLevel` with a new log level.

Supported values include `error`, `info`, `warn`, `verbose`, or `debug`. Defaults to `info`.

For details, see **Log Monitoring**.

Example

YAML

```yaml
services:
  sql:
    image: IMAGE
    command: "--bootstrap '{adminHost: \"sql\", ipAddress: \"foundation0\", logLeve
```

Other Logging Configurations

You can customize your logging configuration for each node by making changes to the `docker-compose.yml` file. In this example, the `sql` node has special configurations for its `max-size` and `max-file` values, meaning the container should have no more than five log files of 10MB in size.

Example

YAML

```yaml
services:
  sql:
    image: IMAGE
    logging:
      driver: json-file
      options:
        max-size: 10m
        max-file: 5
```

Use `docker logs` to retrieve the logs. In this example, the `--tail` option limits the output to include only the last 10 lines of logs.

For more options, see the **docker compose logs** page.

Shell

```shell
docker logs --tail 10 CONTAINER
```

Related Links

Commands Supported by the Ocient JDBC CLI Program

CREATE TABLE

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Architecture

The OcientAIQ™ Unified Data Platform is a modern, scalable distributed system capable of enabling complex OLAP-style analysis on the world's largest structured and semi-structured datasets using SQL. It features a purpose-built storage and an input and output layer designed for highly parallel interaction with modern NVMe SSD drives in a Compute Adjacent Storage Architecture® (CASA) that leverages erasure coding for cost-efficient reliability.

On top of this is a fully integrated execution engine and optimizer that were each built from scratch using modern design principles in a manner that deeply integrates with the storage and I/O layer. The result is a state-of-the-art data warehouse that can deliver outstanding, and even unprecedented, analytical capability on datasets with record sets numbering in the trillions, quadrillions, and beyond.

This section includes these topics:

- **Core Elements of an Ocient System**
- **Compute-Adjacent Input and Output on Large Working Sets**
- **Query Performance Optimizations**
- **Plan Optimization**
- **Exabyte Scalability Design**
- **Loading Characteristics and Concepts**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Core Elements of an Ocient System

The System and Nodes

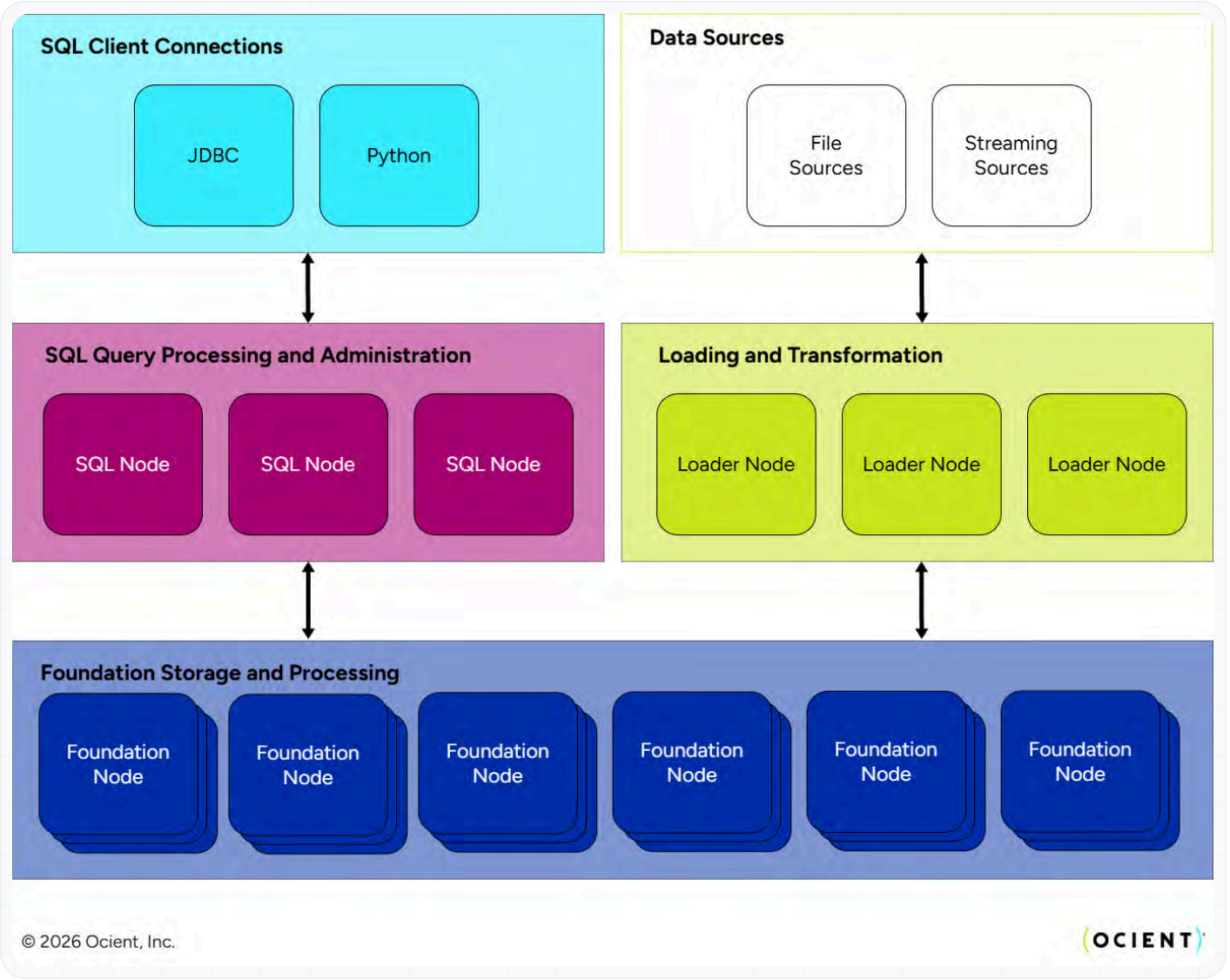
The Ocient® System is the collection of all the nodes within a single environment. This architecture diagram is an example of one Ocient System. Across systems, the number of nodes can vary along with the total storage capacity. Within a single system, users can define varying numbers of databases, storage groups, users, etc.

An Ocient System is made up of a collection of distributed nodes which each play a predefined role in the data warehouse. The different roles are generally responsible for storing and analyzing data, administration, and moving data into Ocient. These nodes are connected to each other through a high-speed 100 Gbps network connection. In addition to storing user data, the data warehouse also stores metadata about the system and data for its own internal operation and recovery. This collection of nodes, their data, and the interconnection between them make up the Ocient System.

The end user, analyzing data in Ocient, connects to this system through a JDBC or a Python®-based client. From there, they interact with the database objects, such as tables and views. The architecture itself is abstracted from users, allowing them to focus on SQL queries and results.

While the number of nodes in an environment might vary, the inclusion of and purpose of the different roles do not. The following sections briefly describe the components in the architecture.

© 2026 Ocient, Inc.



Node Types

SQL Nodes

A SQL Node is responsible for receiving incoming SQL requests and parsing the statements. This node also serves as the interface for System Administrators and Database Administrators to configure and maintain an Ocient System. Administrators can connect to a SQL Node using the Ocient command-line interface or their chosen client.

When a SQL Node receives a query, it creates a plan for it using one of two optimization methods. After there is a plan for the query, the SQL Node distributes it to the Foundation Nodes. When the Foundation Nodes finish their processing, they return data back to the SQL Node for further processing and packaging the result set to the client. When the result set is finalized, the SQL Node returns the data to the client. Beyond the Foundation Nodes, further processing is done on the SQL Nodes to process intermediate result sets delivered by the Foundation Nodes. Depending on the query, the SQL Node is responsible for a different proportion of the overall work of the query. Common processing done here are aggregations and joins that happen across the intermediate results delivered by the Foundation Nodes.

Administrators also use DDL or DCL statements to make changes to an Ocient System. When submitted to a SQL Node, the node executes the SQL statement across the system or forwards the SQL statement to the node assigned to execute it.

Foundation Nodes

The Foundation Nodes store the user data in Ocient and perform the bulk of query processing.

A key architectural concept of Ocient is the Compute Adjacent Storage Architecture®, which collocates data and processing where possible. The Foundation Node is the central element of this architectural principle. When a query is deployed to the Foundation Node, it performs as much of the query as it can with the data on the node before having to join, aggregate, or compare it to other data. It returns that result up to the SQL Node for further processing, packaging, and returning to the user.

Foundation Nodes contain the majority of the storage in an Ocient System and are also typically the largest in number.

Loader Nodes

Loader Nodes are responsible for extracting, transforming, indexing, and loading data ingested by Ocient from batch file sources as well as streaming data sources such as Apache® Kafka®. A user can specify the extraction source details and supply a transformation pipeline that manipulates the structured or semi-structured source data before loading it into relational form into Ocient tables.

Loader Nodes operate in a horizontal scale-out fashion allowing the loading system to scale to fit various ingestion requirements. Transparently to the end user, the Loader Node also manages exactly-once guarantees with data as it is loaded and converts pages into segments in the Foundation Nodes.

Flows and Networking Between Nodes

Data and communication flows across the nodes of an Ocient System for a variety of different purposes. There are two networks that connect nodes, a 100 Gbps high speed network and a 10 Gbps network. These are used in different ways for data flows and administrative purposes.

A Query in Ocient

When a user issues a query against the database, their SQL statement moves from the JDBC client to the SQL Node. The SQL Node parses and optimizes the query before handing the plan off to the Foundation Nodes. The Foundation Nodes do the first level of processing against the data before

handing the results back to the SQL Node. The SQL Node further processes the data, constructs the results, and returns them to the client.

- The connection between the SQL Nodes and Foundation Nodes is typically a 100 Gbps network connection. The speed of the connection between the client and the SQL Node is subject to the speed of that network (outside Ocient).
- Every Foundation Node is connected to every SQL Node.

Administration Flows

When administrators configure or make changes to an Ocient System, they do it using DDL and DCL using a SQL client. The SQL client relays the statement to a SQL Node, which parses it into commands that the Administrator Role on the SQL Node executes. This can impact the SQL Nodes, Foundation Nodes, or Loader Nodes, depending on the type of change or operation.

- The administration flow connections with the Ocient System are 10 Gbps connections.
- Every node is connected to the SQL Nodes that perform the Administrator Role.

Loading Flow

When the load job is defined and started, the Loader Nodes read data from the source files, stage the data, process that data, and move the data into the Foundation Nodes. Data pipelines enable you to define the data load and any corresponding transformations to ensure data is properly loaded.

- The network connections between the Loader and the Foundation Nodes are 100 Gbps.
- The Loader Nodes connect to every Foundation Node.

Database, Tables, and Views

Similar to other databases, an OcientAIQ™ Unified Data Platform is a grouping of databases, schemas, tables, views, users, and data within one Ocient System. A single system can have multiple databases and schemas. Schemas define the structure of a database and contain the definition of tables, views, and indexes. While Ocient stores data on disk in columnar format, creating and working with tables in Ocient is like any other relational database. Ocient supports a wide variety of standard SQL data types along with an expanding set of geospatial data types.

Read more about:

- **Data Definition Language (DDL) Statement Reference**
- **Data Query Language (DQL) Statement Reference**

- **Functions Overview**

TimeKey and Clustering Key

Each table in Ocient can contain a TimeKey® column that is specified when defining the table. The TimeKey column is used to partition the data within the Foundation Nodes. Time partitioning is an important performance mechanism used by the OcientAIQ Unified Data Platform. Because many, if not most, queries specify some time filter for the results, time partitioning allows Ocient to quickly skip data from irrelevant times.

When defining a table, users can also specify a clustering key containing one or more of the columns to further subdivide records on disk. This allows the database to quickly find records with the same key values.

Read more in **TimeKeys and Clustering Keys**.

Resilience to Hardware Failure

In order to provide reliability in the case of a hardware outage, Ocient uses erasure coding. Erasure coding is a mechanism to organize and compute parity blocks so that the system can rebuild missing data. This means that Ocient does not store redundant copies of the data for reliability. As a result, an Ocient System requires less overall storage than if it were storing one or more copies of the data for failover.

When a user defines their storage configuration, they specify the width of the group and the parity width. These values can vary based on how many Foundation Nodes are in the system and how resilient users want to make the Ocient System.

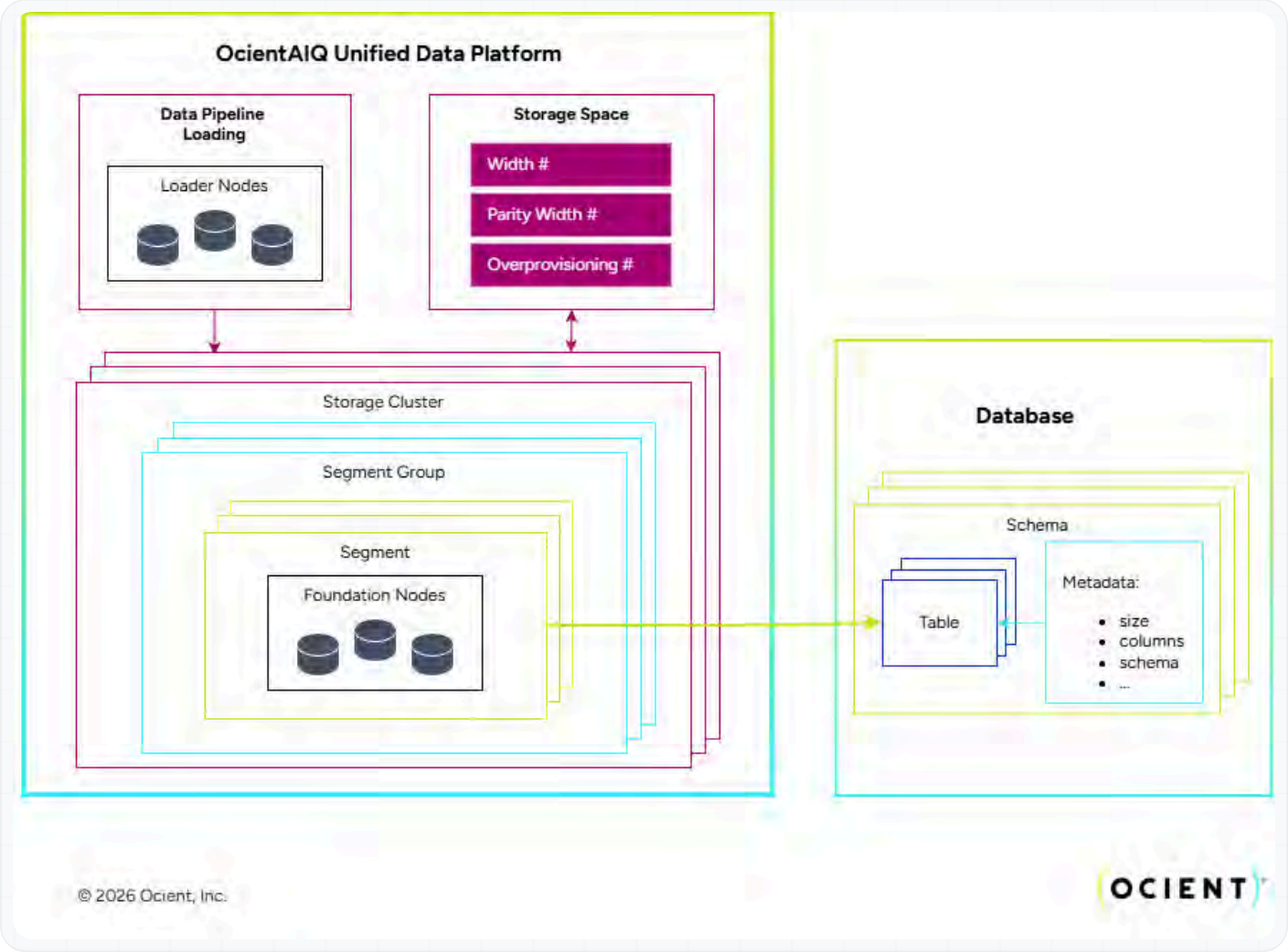
Segments and Segment Groups

Segments in Ocient are storage units that contain rows and are divided into a fixed number of coding blocks with a defined size. Segments are typically sized on the order of gigabytes. You set the segment size when you define tables in Ocient. The size of the segment must be a multiple of the coding block size. The coding block is the unit of parity calculation and the smallest unit of recovery as well.

Multiple segments combine to form Segment Groups. A Segment Group has a fixed number of segments, named the width, which is the number of segments in the group. It also has a pre-defined number of parity blocks per set of data blocks for resiliency. Each segment has a defined index in the group. The Segment Group is physically stored in a storage cluster. Segment Groups can also be nested in directories named Segment Directory Groups.

When the user configures an Ocient System, they need to define at least one storage space and a storage cluster. A storage cluster is a set of Foundation Nodes with an associated storage space. Also, these nodes coordinate together to store segment groups in a reliable manner. The storage cluster also has a width in the number of nodes. The width of the cluster and the Segment Groups stored on it must be the same. At the storage space level, the user defines the width and parity width of the space. Segments and their Segment Groups are stored across these storage spaces and clusters. To learn more about erasure coding, see **Compute-Adjacent Input and Output on Large Working Sets**.

For details, see **Configure Storage Spaces**.



Related Links

- **Ocient Architecture**
- **Compute-Adjacent Input and Output on Large Working Sets**
- **Key Concepts**

Related Videos

- **At the Whiteboard with Ocient: Deployment Options**
- **At the Whiteboard with Ocient: Data Reliability**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Compute-Adjacent Input and Output on Large Working Sets

The storage of the data warehouse and the input and output (I/O) layer is built using a Compute Adjacent Storage Architecture® (CASA). When contrasted against modern cloud-based database engines, this means that the majority of the data to be analyzed is stored on NVMe SSDs directly attached to a significant fraction of the computational capacity of the system using the local PCIe busses of the nodes that make up the ***foundation level*** of the database engine.

This approach, which is similar to a traditional MPP architecture, is designed to capitalize on the performance characteristics of NVMe SSDs when executing queries against a large ***working set*** of data. It is important to note that when executing a query, a specific record could be included as part of the results, but I/O and computational effort must be undertaken to make that determination. A working set, therefore, is defined as the set of records stored in the database that are considered as part of query execution, even if those records are eventually filtered out or not returned. A “large” working set is one that is substantially larger, even by multiple orders of magnitude, than the available DRAM on the set of nodes performing the computations. When this is the case, ***remote-object*** database engines that store the majority of their data in a separate storage layer are substantially affected by their inability to keep all the potentially-addressed records near and accessible to computation at the same time. Depending on the query load and their exact engine design and scheduling, query execution times negatively suffer from both simple bottlenecks due to network link capacity and “cache thrashing” that produces expensive extra drive I/O.

Large working sets can be realized in two major ways:

1. Queries that regularly consider a substantial fraction of the entire data set. For example, regular querying or analysis of all data spanning years can mean a single query addresses most of the records in the database.
2. A large number of smaller queries that all consider a disjoint subset of the whole data set. When these queries are executed in parallel the sum total of the disjoint sets of addressed records can also approach most of the records in the database.

The database engine in the Ocient® System is a world-class relational database engine that can go toe-to-toe with other modern engines on small working sets but won’t necessarily out-perform them. After all, well known algorithms like sorting and hashing can only be made so fast when everything is in memory. But the performance distinctions of Ocient derive from its ability to scale CASA to keep all of the data close to the computation.

© 2026 Ocient, Inc.

NVMe SSDs in CASA

One of the biggest drivers for the development of the OcientAIQ™ Unified Data Platform was the ongoing shift from spinning HDDs to NVMe SSDs for data storage. SSD prices continue to fall, relative to HDDs, and there remains little doubt that they will eventually supplant HDDs for all but esoteric or extreme bulk storage applications.

To ride this storage wave, the Ocient engine has been built to minimize the cost of using these drives as the primary storage tier of the engine while maximizing their performance. There are two major aspects to cost management:

1. *Minimization of the total number of stored bytes needed to achieve reliable access to data in a performant manner*. This is achieved using a novel and tunable ***erasure coding*** scheme in conjunction with a variety of compression techniques. This is detailed more thoroughly in the following sections.
2. *Consideration for endurance management in I/O operations against the NVMe SSDs.* The Ocient storage and execution engine is designed to minimize the number of I/O write operations undertaken against the storage media in order to maximize its lifetime and reduce drive replacement rate due to endurance wear. At a high level, this usually means minimizing small or random writes to the drives and instead favoring larger and sequential ones. To achieve this, the storage layer uses a specially-written and database-aware file system underneath storage structures designed to capitalize on the large volume of low-mutability data being stored. Using these mechanisms, the storage layer is able to collate writes in a manner that produces relatively large (from the drive's point of view) ranges of contiguous blocks, ordered in the hundreds of KiB to hundreds of MiB in size.

Erasure Coding

Of particular note in the storage and I/O layer of the OcientAIQ Unified Data Platform is the novel use of a tunable erasure coding scheme for data storage.

You can think of erasure coding as a generalization of simple XOR parity calculations. In a simple error correction scheme, you can apply the XOR operation on a sequence of bits: `B0, B1, B2, ..., Bn` such that: `B0 XOR B1 XOR ... XOR Bn = Bx`. An easily provable property of the XOR operation is that for any single Bi with 0 ≤ i ≤ n, that `B0 XOR ... XOR Bi-1 XOR Bi+1 XOR ... XOR Bn XOR Bx = Bi`. You can determine any single bit by performing the XOR operation on the other original bits and the extra bit together. In the context of error correction, this means that for the cost of an extra bit of storage (or memory) a system is able to detect, and even correct for, the loss of a single bit.

There are a variety of error correction schemes described in computer science and found in the industry. In the application of this concept to database storage, the Ocient storage layer extends it

from individual bits, and beyond bytes, to an error correction unit of the 4KiB block (which is notably aligned with the I/O resolution of standard NVMe SSD drives). It also further extends from enabling single loss or failure to a fully tunable approach that uses ***XOR***, ***P+Q***, or ***Cauchy-Reed-Solomon*** coding that allows the user to specify exactly how many faults: one, two, or K, respectively. Like the simple XOR formulation, these schemes utilize mathematical computations (in this case, polynomials over a finite/Galois field) to compute additional blocks of data that enable reconstructing a configurable number of missing blocks.

In database and storage systems that utilize replication to provide data reliability and availability, it is common to describe storage in terms of the number of replicas (e.g., 3x replication allows for up to three failures). The obvious implication of replication is that each replica is a full copy of the original data and represents a multiple of the minimal data storage required. In an erasure coding scheme, the overhead for `K` tolerable faults is calculated as `(N + K)/N`, where `N` is the total width and `K` is parity width. If `N` is 10 (take that as a given for the moment) then to support a maximum of 2 faults (the equivalent of 3x replication, which includes the original data and two copies) the system needs only `12 / 10 = 1.2x` times the original storage size, contrasted with the 3x of the replication scheme.

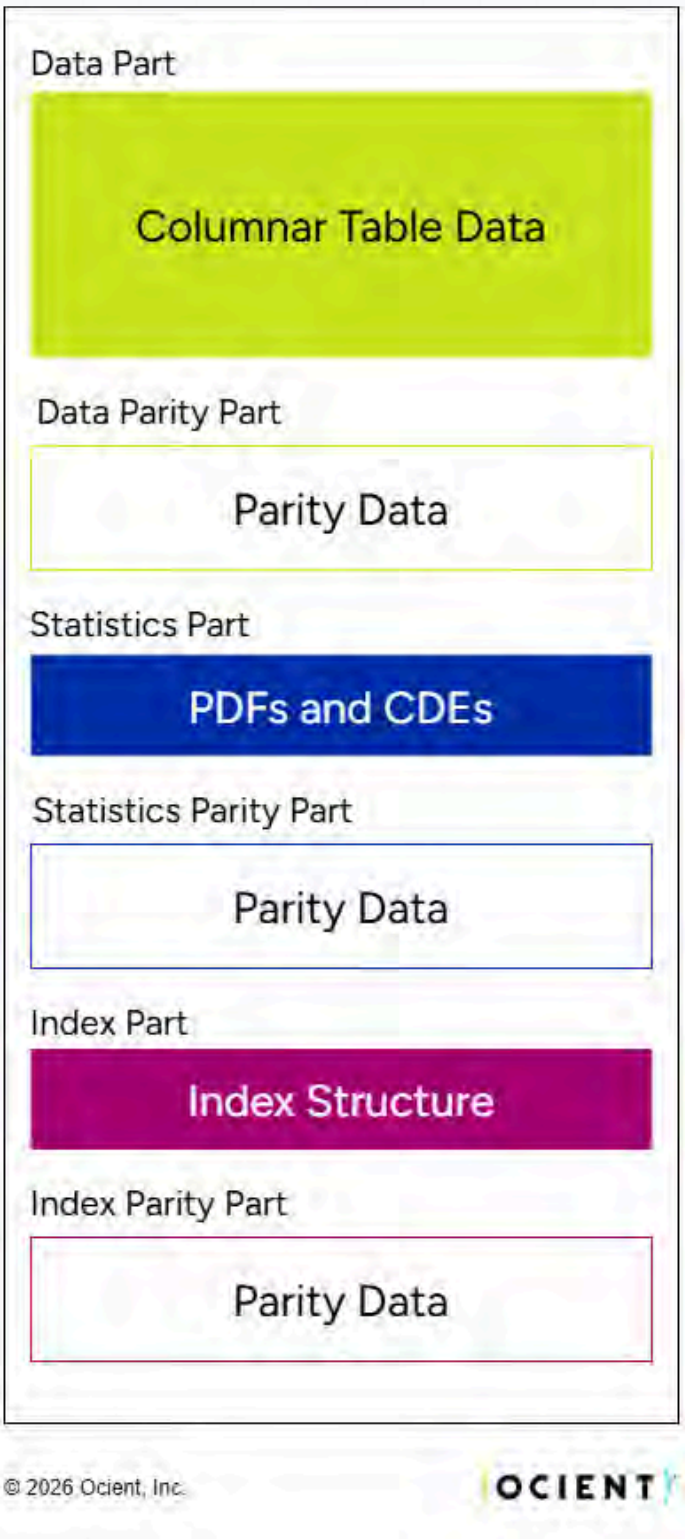
The utilization of space-efficient erasure coding can be found in a variety of storage and transmission systems, but, in a classic space-time trade-off, the space savings come at a computational cost. The approach taken in the Ocient storage layer is novel, at least for databases, and is designed to mitigate the computational overhead of erasure coding.

If one considers a “file” of some size as a unit of storage, a basic approach for applying an erasure coding scheme to provide loss tolerance to this file would be to break it into `N` chunks and perform the requisite computation to produce an additional `K` chunks. At read time, the system would select an available subset of the `N+K` chunks (possibly preferring the original `N`) and perform the **rebuild** computation to produce the original file for access. In this framing the unit of fault tolerance is the individual file, and rebuilding computational overhead can be found in the read path.

However, in a database designed for OLAP-style queries, the potential for the overhead of any rebuilding computation in the read path even when no drives or nodes are offline is troublesome. Additionally, when the system is in a degraded state, the potential need to rebuild an entire file only to retrieve a small subset of its data is expensive as well. The Ocient storage layer, by directly implementing the erasure coding scheme and integrating with it, avoids these issues in two major ways.

The first advantage is the unit of storage. Ocient uses storage units named ***segments,** *which are not chunked into `N` pieces. Instead, `N` separate and similarly sized segments are considered together as a unit named a ***segment group***. The erasure coding scheme is then applied across the segments to produce `K` additional “segments” worth of redundancy information, which, in a very loose application of the term, is named ***parity data***.

This graphic shows a high-level illustration of how Ocient stores each data segment. Segments contain all the data needed for a subset of table rows, including the actual column data as well as indexes, metadata, and statistical data such as probability density functions (PDFs) and count distinct estimates (CDEs). In addition, the erasure coding scheme also includes parity data for other segments in the same segment group.

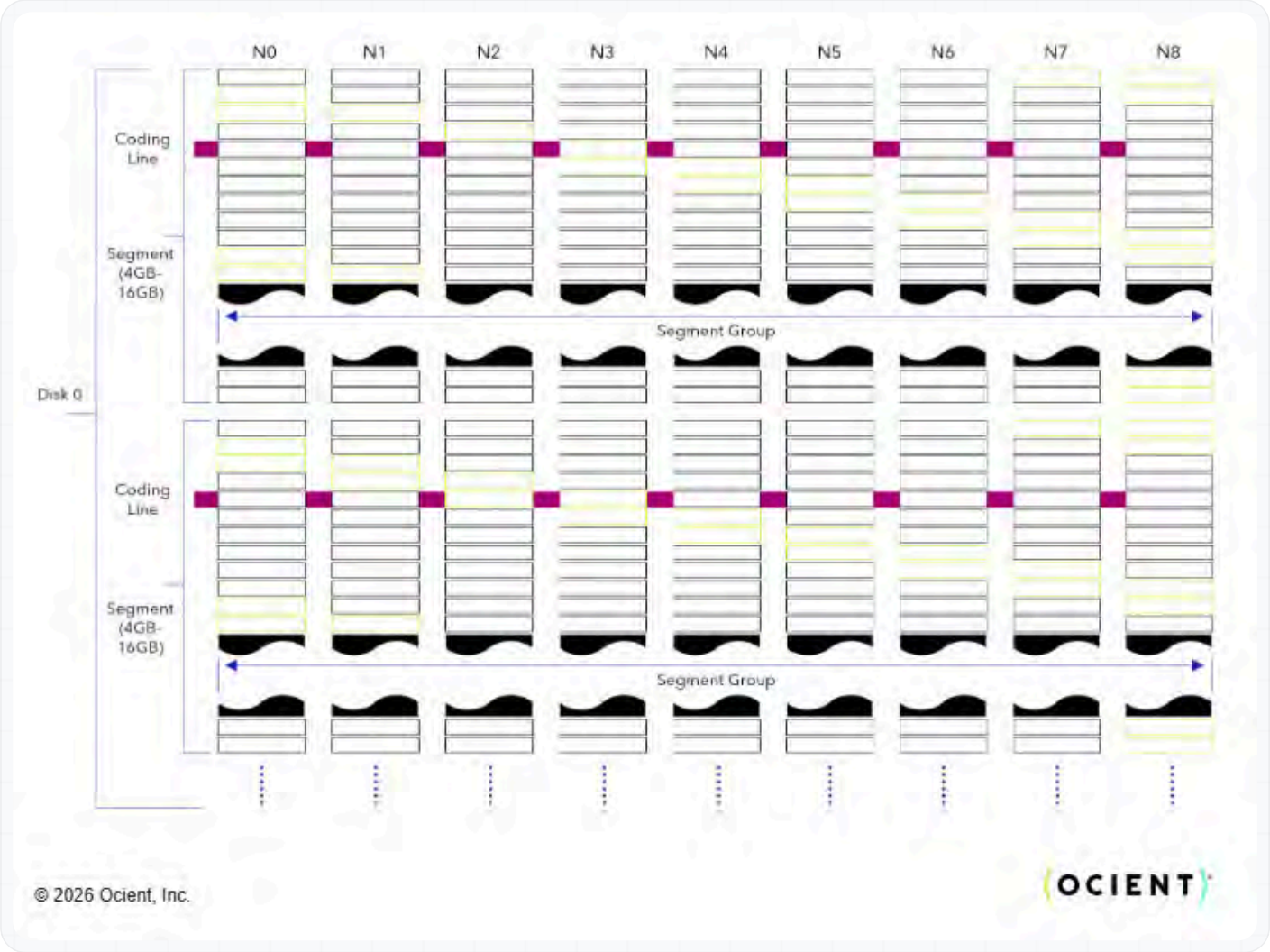


The newly generated parity data is then broken up and evenly distributed amongst the original segments (in a particular and calculated way) such that each segment is now approximately `(N+K)/N` times larger than it originally was, and any missing segment from a particular segment

group can be reconstructed when one has access to any set of `N` segments from the same group. The Ocient storage layer ensures that each segment from a group is stored on a separate physical node, thereby providing data availability in the event of up to `K` node losses.

Another important feature of this scheme is due to the fact that a given record is always stored in a single segment. Because the segments themselves are not chunked, any specified record is guaranteed to exist intact on some node/drive, and readable with no overhead incurred when that drive is online. Put another way, in the non-degraded state, the reliability scheme implemented incurs no computational cost.

The second advantage of the Ocient System is each segment acts as a self-contained unit of information, containing not only database records but also metadata and indexes on those records. These sub-units within the segment, named ***segment parts*** are individually addressable and independently rebuildable with 4KiB resolution. The implication of this is that in a degraded state (where a node or drive is missing), it is not strictly necessary to rebuild all missing segments to service a specified query. Instead, the storage layer is capable of addressing and rebuilding only the parts and even single blocks of parts required for it. This often greatly reduces the overhead and performance impact of a degraded state.



Choice of “N”

Earlier, it was taken as a given that `N` is equal to 10. The exact choice of this value is somewhat arbitrary but is generally bounded by system-level consensus constraints as well as the calculated cost of the network transit during rebuild operations. As `N` increases, the storage overhead for the data reliability for some specified `K` *decreases* because `(N+K)/N` approaches 1. However, the total amount of information that must be considered and moved over the network for, as well as the actual computational cost of, a rebuild *increases*. Real-world usage has found that an `N` somewhere between 8 and 12 provides a good tradeoff between these two opposing considerations.

NVMe Userspace Optimizations

A final important aspect of the Ocient storage and I/O layer is the database engine’s direct interaction with the SSDs storing segments. The term “direct” in this context can be contrasted with how a more standard approach might look. In many database systems, record storage is achieved using a set of files stored on a disk in some file system (such as EXT4, NTFS, or ZFS). And generally speaking, kernel-provided system invocations or services, like `read()`, `write()`, and `mmap()`, perform reading and writing of those files. Obviously, this is a fine approach in general, but when performance is of utmost importance, the overhead of the system calls, memory copies to and from kernel memory, as well as the implementation of the underlying file systems, all have detrimental and sometimes unpredictable effects. For example, it is well known that the kernel of Linux® EXT4 implementation takes global locks during certain metadata operations, which can have a significant detrimental effect on parallelism and throughput.

Some databases eschew standard files and perform some form of I/O directly against kernel block devices in order to avoid file system overhead. This is surely an improvement, but the kernel’s block scheduler, extra memory copies to/from kernel memory, and the overhead of system calls still exist.

In order to overcome these performance impacts, the Ocient storage layer directly interacts with the NVMe SSDs that store segments in two important ways:

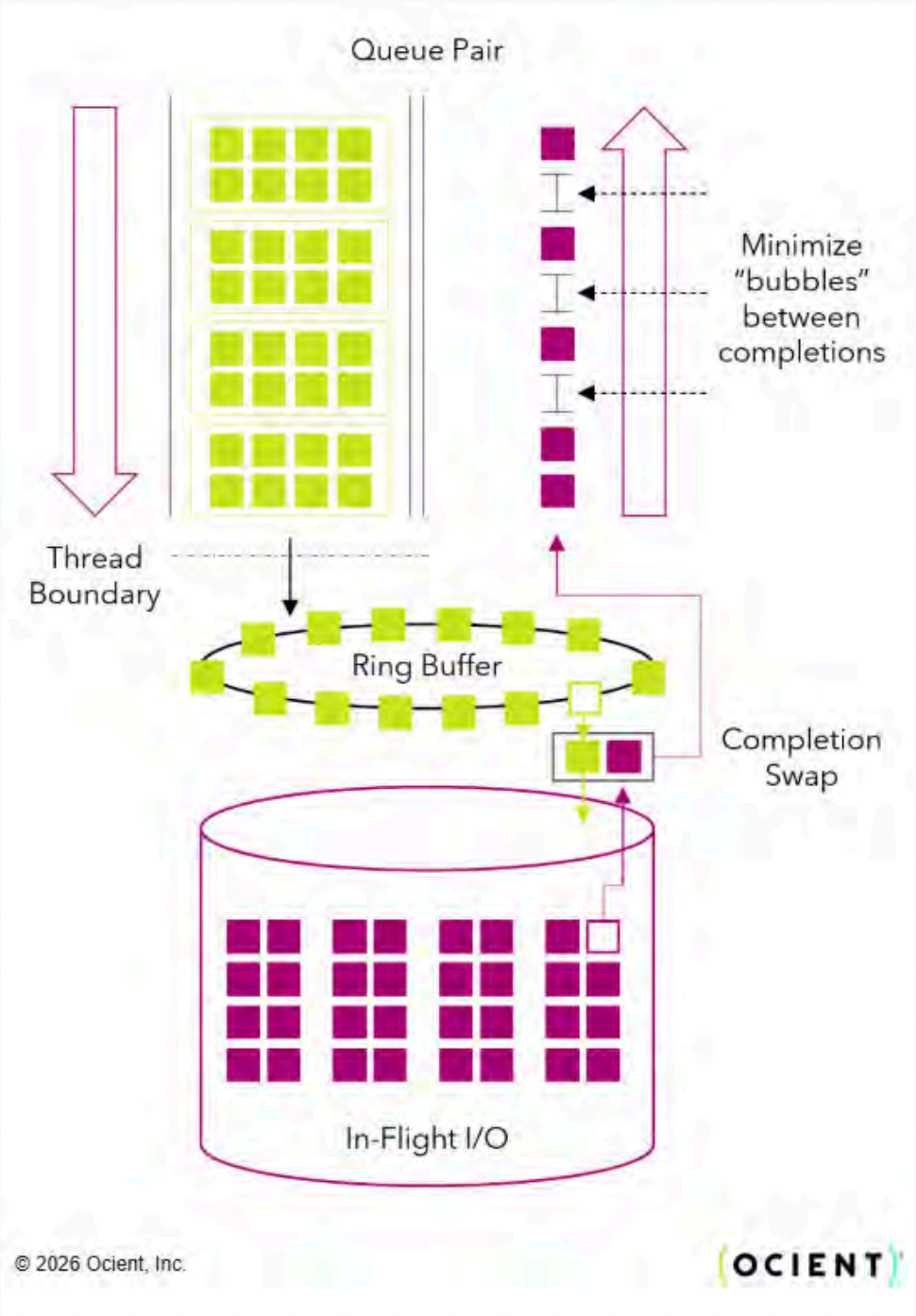
1. Each NVMe drive is decoupled from the standard Linux block device driver and instead attached to a device driver that enables low-level interaction with the device using the PCIe protocol. The implications of this approach are that the storage layer and database engine can directly communicate the NVMe protocol to the devices *and* do so using user memory. This completely eliminates all system calls and memory copies required to do I/O with the NVMe drives. The database provides direct pointers to its own virtual memory for I/O operations, and the drives directly write and read to and from that memory.

© 2026 Ocient, Inc.

An interesting result from this approach is that the Ocient storage layer and engine were entirely unaffected by the Linux kernel's mitigations for the Spectre and Meltdown classes of attacks. This is because the primary mechanism of the mitigation resulted in a substantial increase in the overhead of the user-to-kernel transition associated with system calls. Other database engines with more standard approaches saw, in many cases, appreciable performance degradation.

2. Modern NVMe SSDs are capable of a million 4KiB random reads per second. This is strikingly and substantially more than HDDs (where even the best drives can produce only hundreds). However, for most economical SSDs, these rates can only be achieved when large numbers of parallel in-flight requests are maintained over a substantial fraction of the drive's address space. The Ocient storage layer implementation, in conjunction with specially tailored data and metadata structures, not only promotes this parallelism to be evoked during query execution, but also is written to minimize "latency bubbles" by always having a replacement I/O operation teed up for a drive when it completes a previous one.

© 2026 Ocient, Inc.



Columnar Compression

As a column-oriented data warehouse, Ocient is able to leverage per-column data locality and similarity to achieve excellent data compression.

- For fixed-length columns, a combination of delta-delta compression, run-length encoding, and NULL elimination provides significant storage reduction for many data types.
- For variable-length columns, the Ocient System can apply per-row compression to reduce the size of large values.
- Both fixed and variable-length columns can benefit from whole-column compression using a shared dictionary trained on a per-column and per-segment basis.

Column compression works best when combined with secondary indexes on frequently filtered columns.

Related Links

Secondary Indexes

Key Concepts

Related Videos

At the Whiteboard with Ocient: Compute Adjacent Storage Architecture™

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Query Performance Optimizations

Specialized Indexes

The storage engine, execution engine, and optimizer are all aware of two special concepts that can bring about significant query performance benefits. Neither of these specializations are strictly required in any table or query, but the Ocient® engine capitalizes on the fact that for the datasets with trillions or quadrillions of records they are often advantageous.

Time Keys

Much of the data in hyperscale databases is produced by machines, devices, loggers, and sensors. The records these entities emit often have a temporal nature: they are timestamped or otherwise sequenced relative to each other. At query time, it is often the case that queries themselves have temporal predicates (e.g., “select all records for this user *for this year*”) as well.

When these two conditions are met, the storage layer, execution engine, optimizer, and loading infrastructure all have special handling built-in for more efficiently processing these records. The time-based processing of records shares similarities with zone maps in that they can be used to avoid input and output (I/O) for non-intersecting sets of data when predicates against certain columns are present in a query. However, the specific temporal nature of the data is further utilized to make loading more efficient and also impacts segment grouping and data roll off.

Clustering Keys

For many data sets, there is often a commonly queried set of columns that form the basis for a variety of analytical tasks. For example, in a network security database recording metadata of IP traffic, it might be the case that a substantial fraction of interesting queries are concerned with the public destination IP address, represented as a column in a record, in addition to other constraints or filters.

When this is true for a subset of the columns in a given table, a ***clustering key*** can be defined, which enables additional specialized handling across the storage layer, execution engine, optimizer, and loading infrastructure. A clustering key shares some similarities with the “partitioning key” concept found in other databases in that it is defined in terms of one or more columns in the table and also factors into locality decisions for storage. However, unlike most systems, the clustering key *is not* a strict partitioning, and is instead only a ”hint” that promotes locality for storage to improve I/O efficiency for queries that locate records using it. Because it is only a (usually followed) hint, the storage layer and engine do not suffer from issues related to partitions outsizing others or forced locating of segments on particular nodes.

© 2026 Ocient, Inc.

Global Dictionary Compression

Global Dictionary Compression, or ***GDC***, benefits both storage utilization and query throughput for columns with low cardinality (≤ ~1 million distinct values). When enabled for a specified column, that column's values are mapped to a succinct global enumeration that is stored on disk and used at query time. GDC is often enabled for variable-length columns representing an enumerated set of potential values (e.g., city, county, and country names, or classification tags). In these scenarios, it provides significant compression and also enables a number of optimizations in query and join processing. In this latter category, the optimizer is able to delay or even eliminate the decompression step during execution because there exists a single global key-to-value mapping. This often results in a significant reduction in string data processing as well as network and memory bandwidth.

Like other forms of compression, GDC is completely transparent to end users. New values are added to the global enumeration automatically, and compressed values are mapped back to their original value in query results.

Specialized Types

Complex Types: Tuples, Arrays, matrices

In many analytical workloads, semi-structured and multi-dimensional data must be analyzed. Ocient offers complex data types to efficiently store and analyze this type of information, in many cases dramatically reducing query response times when compared to normalized schema designs. Each complex type allows groupings of elements to be stored and analyzed in a denormalized fashion with the parent record, and combinations of these complex types might also be used. This avoids the repetitive joins across billions of records required for nested data in a normalized schema improving query response times. Finally, Ocient's indexing strategies extend to complex types as described here.

The complex types supported in Ocient include Arrays, Matrices, and Tuples. An array is an ordered list of elements. The element can have any data type including a tuple. Array types provide operators for efficient set operations and standard SQL semantics like SOME and ALL. matrices are multi-dimensional arrays that share many of the same benefits as Arrays when denormalizing schemas or working with matrix data for mathematical operations.

Tuples are structured objects in a single column created from individual data elements of the same or different underlying types. Arrays of Tuples provide the ability to denormalize structured elements that can then be jointly queried and indexed for improved performance in some cases. Because the array of elements can be stored in a columnar fashion and associated with the parent row, they need not be joined to the parent table repeatedly to query them. Performance is improved, and working set memory consumption is reduced on queries that would otherwise need to join tens of billions of rows to potentially trillions of rows. The flexibility of these composable

complex types creates more options for database administrators to structure data and optimize performance at scale.

Secondary Indexes

A variety of secondary index types are used to improve query performance. Secondary indexes are built and searched on a per-segment basis, allowing different segments of a specified table to be indexed differently where appropriate.

For fixed-length column types, an I/O-optimized inverted secondary index enables fast equality and range queries. Variable-length columns support hash and N-gram indexes that enable fast equality and `LIKE` filters, respectively.

Complex types like tuples and matrices are a concise way to model certain datasets. In order to support efficient filtering on these types and their sub-components, filters are pushed into the I/O layer and use secondary indexes when possible.

You can create indexes on arrays of both fixed and variable-length elements to optimize contains, overlaps, and other similar operations. Additionally, `for_all()/for_some()` syntax allows other filters to be pushed down and applied to arrays using an index, without requiring `unnest`. In particular, `for_all/for_some(varchar_array_col) LIKE <filter>` utilizes N-gram indexes to optimize textual queries over arrays of strings.

Per-Segment I/O Pipelines

The I/O layer is designed to minimize the amount of on-disk data read and processed for each query by compiling a custom ***I/O pipeline*** for each segment of data the query accesses. An I/O pipeline is a directed acyclic graph that represents an efficient I/O plan for the particular columns and filters of that query combined with indexes and data that can vary from segment to segment. This plan adjusts for data demographic changes over time, and for schema or index changes also. Just like the query plan itself, efficient pipelines are critical for performance.

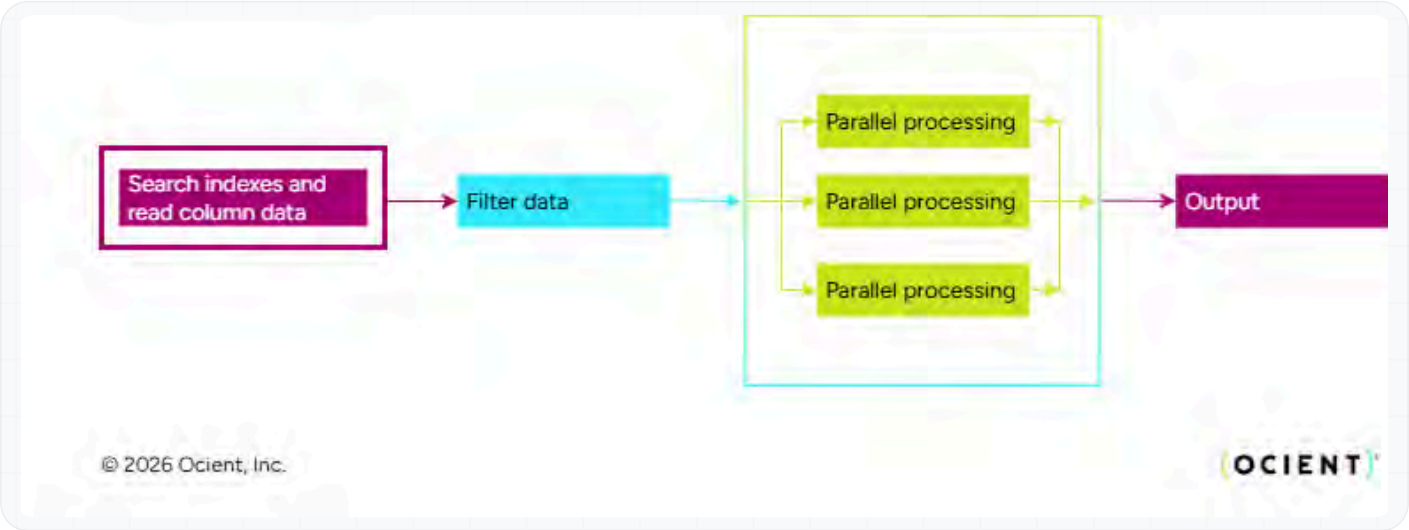
A simple I/O pipeline example is illustrated here:

This graphic example shows the I/O process stages that go into action when the Ocient System compiles a pipeline. These stages are as follows:

1. When a query is in the IO pipeline, each segment first searches indexes to minimize the data that must be read and processed.
2. The indexes help to quickly filter the rows. When possible, filter predicates are pushed down and evaluated in-line, further reducing the number of rows needed to read, even when an appropriate index does not exist.

© 2026 Ocient, Inc.

3. The system reads rows that pass the filters in parallel.
4. The system returns the query output.



During execution, ***I/O pipeline elements*** work together to search indexes, read column data, and apply filters to return the exact set of matching rows from each segment. Much like CPU instruction pipelining, the I/O pipeline attempts to mitigate latency while fully utilizing fast storage devices requiring deep I/O queues. The graph for a sophisticated query can orchestrate dozens of pipeline steps in series and in parallel. This layer works to keep disks saturated by coordinating I/O across segments with read-ahead and a shared priority queue.

A good I/O pipeline determines the set of cluster key and secondary indexes to use, and an optimal ordering based on index selectivity to filter rows at maximum speed. When possible, filter predicates are pushed down and evaluated in-line, enabling optimizations even when an appropriate index does not exist.

Optimizing I/O with Bloom Filter and Index-Based Join Push-Down

A pair of advanced techniques are employed to dynamically reduce the amount of I/O needed when processing certain types of joins.

- **Index-based join push-down** — As a running query narrows the collection of potentially matching values, an updated list of filters can be dynamically pushed down into running I/O pipelines. When a matching index is present, the I/O layer can begin using it mid-query to reduce I/O. Otherwise, this technique has the same benefits of early filtering as a pushed-down Bloom filter.
- **Bloom filter push-down** — When one side of a join is fully determined, the execution engine can build a Bloom filter that gets pushed down into running I/O pipelines to enable early filtering of unneeded values at the lowest level. As a probabilistic data structure, Bloom filters can be tuned to provide filtering benefits for even high-cardinality sets.

Both of these techniques reduce I/O on dependent columns, memory, and storage bandwidth for unnecessary rows and unneeded join processing.

Query Execution

Parallelism

The storage layer and I/O access patterns to promote parallelism would be of limited impact if the data processing working on the actual data was not itself highly parallel. To that end, the execution engine (which is named the ***Virtual Machine or VM***) is designed to maintain parallel and disjoint flow of data during data processing.

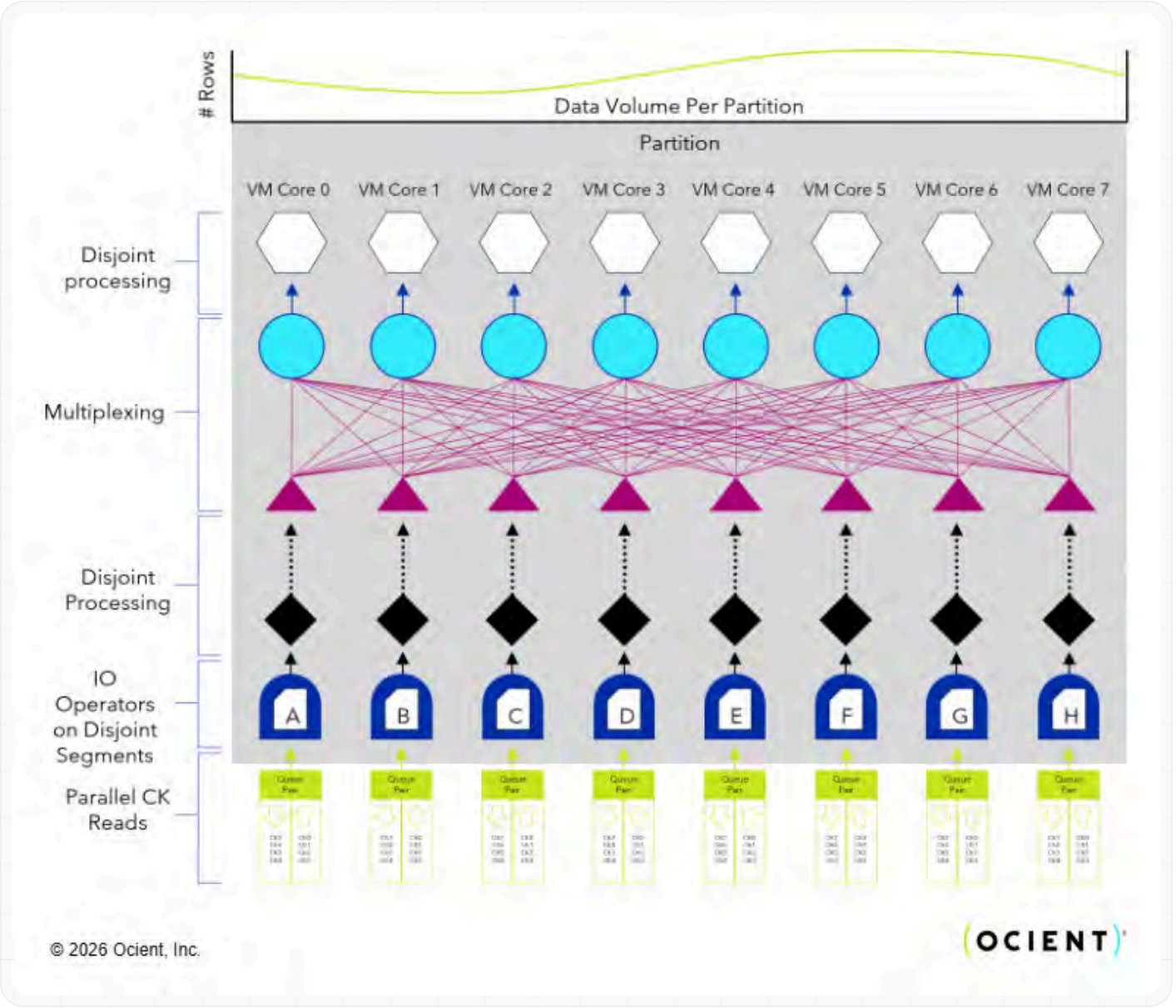
A query ***plan*** is a (mostly) directed graph of **plan *operators*** in a tree-like arrangement. At the leaves of the tree are the raw I/O operations undertaken to bring records off of SSDs and into memory for processing. Internally, iterative processing, computation, filtering, and joining is undertaken by successive operators to produce the final results for the query. The execution engine is named the VM because it structures plan operators as assembly instructions (such as ADD and `MOV`) to be scheduled and executed.

The fundamental organizational unit of this execution engine follows a design pattern named ***resource-aligned parallelism*** where a series of independent execution units, strongly coupled with hardware resources like cores, memory, and disks, disjointly process independent ***data partitions*** of records that flow independently and in parallel within the engine. Because minimal to no data sharing is required across data partitions, the execution engine not only maximizes resource utilization and throughput, but also scales very well as CPUs continue to include larger numbers of cores.

The number of data partitions across any edge within the plan tree is usually, but not always, equal to the number of distinct ***VM cores***. Each VM core is a dedicated and isolated CPU core that does not process any other database or system threads (this is achieved using a Linux® kernel feature named “cpu isolation”). Then, each plan operator is locally “compiled” into a number of ***operator instances*** that is equal to the local number of VM cores. In this way, any individual operation within a plan can be spread over all allocated computational units to independently process the parallel disjoint subsets of records it is to handle.

Of particular note is the attention paid to data skew. In order to maximize aggregate throughput across all parallel operator instances, it is required that they perform approximately the same amount of work. If work is not evenly distributed across all parallel processors, then query execution is limited to the slowest one while the others have no work. Even partitioning of data is easy to achieve in some cases, and impossible in others (it depends on the operation and the data), but the Ocient engine undertakes a variety of special efforts to evenly balance data flow, even in non-trivial cases.

© 2026 Ocient, Inc.



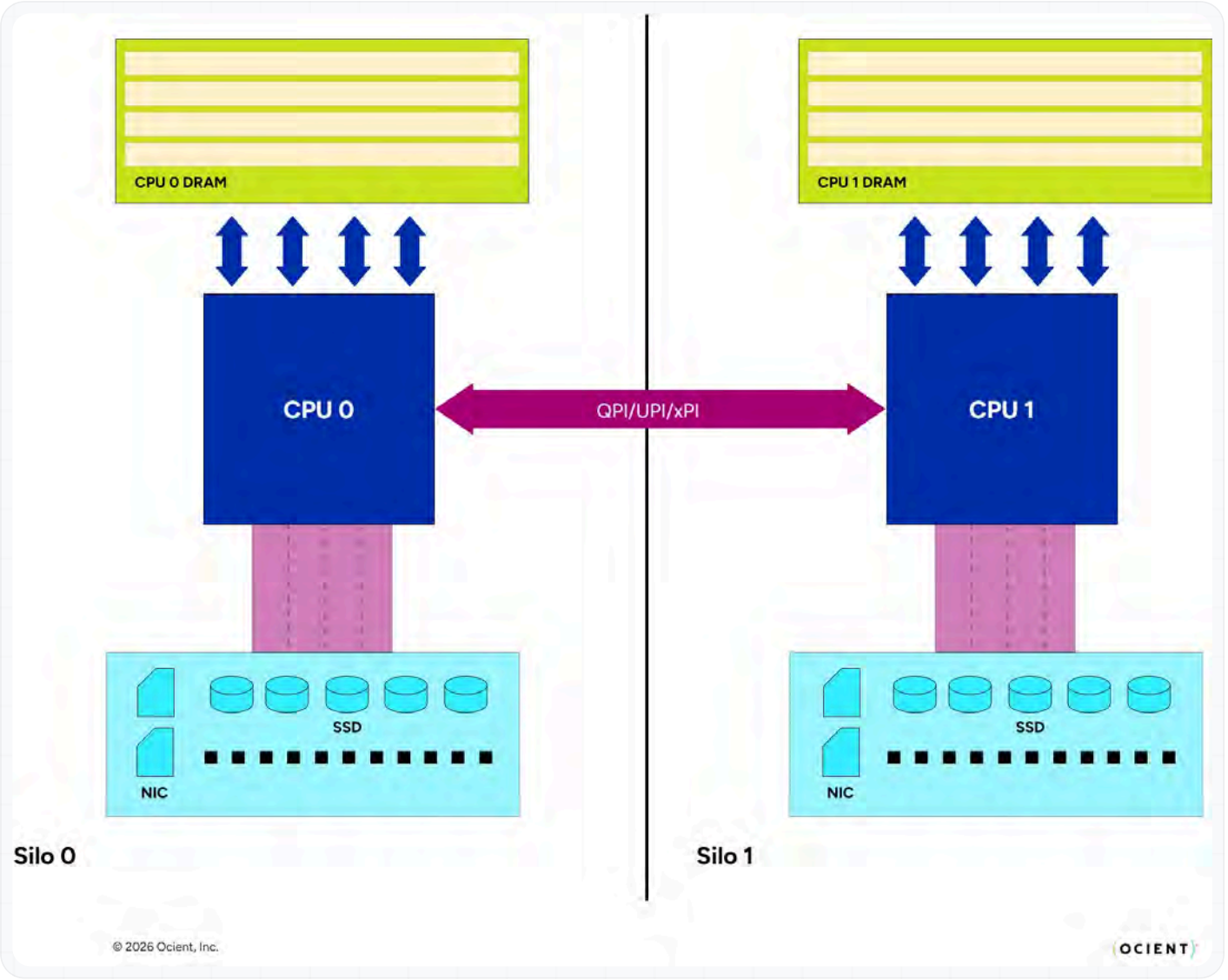
An interesting example of this is found in sort execution. If one were given a set of integers to sort, a straightforward approach to parallelize the sort operation would be to split the set into a sequence of disjoint subsets `S0, S1, S2, ..., Sn` where for all `n`, the integers in `Sn` are all less than the integers in `Sn+1`. Then, one could sort each subset, in parallel, to end up with a totally ordered sequence of integers across all sub-sequences. Put another way, the system splits and groups the integers at boundary values on the number line, then sorts each interval independently. The problem of data skew arises when the number of rows per interval (or subset) is not roughly equivalent, with the worst case being where one interval has all the integers and the others all have none. The problem is one of boundary selection: any naive choice can be subject to skew. To address this in the sort case, the database engine, in conjunction with information provided by the optimizer based on metadata and statistics, *adjusts* the boundaries for the ranges such that each boundary has a roughly equivalent number of rows.

Implementation Efficiency

When discussing parallelism, you might have noticed several terms that sound very operating-system-like: threads, disjoint cores, compilation, and so on. In fact, the database engine directly implements many general pieces of functionality that are usually in the domain of the operating system, but with database-specific features that outperform standard system-provided ones.

1. *Specialized hugepage-backed memory allocators*. Designed for high throughput and low lock contention, these allocators are backed by hugepage memory, not only deriving the benefits of fewer TLB misses, but also making them suitable for userspace I/O operations because their virtual to physical address mapping is fixed. Special efforts were made to make them extremely fast and efficient within the resource-aligned parallelism pattern.
2. *Cooperative per-VM-core Scheduler*. This component runs per VM core, allocating CPU time to operator instances on that core to process data. In this framing, each operator instance can be considered as a parallel thread on its core. However, the scheduler is not a generic thread scheduler, but instead has access and visibility into inter-operator data flow, concurrent query memory utilization, and workload management limits. This means that it has a much richer set of information with which to make intelligent scheduling decisions that promote maximal total system throughput.
3. *NUMA-aware Data Processing*. Modern multi-socket systems utilize a non-uniform memory architecture (NUMA), whereby individual CPUs (and the cores in them) have both local memory and remote memory. The implication of this is that a particular memory access made against local memory by code executing on a core is not only lower latency than a remote one, but also avoids an expensive hop across a dedicated memory bus with a finite amount of bandwidth. Because most commodity CPU packages directly integrate a PCIe bus as well, it is the case that not only is there local and remote memory, but local and remote devices as well. To account for this, the VM groups locally-connected RAM, disks, and cores into a functional unit named a ***silo*** that for all practical purposes is treated as an isolated and independent processing unit. Generally speaking, records for a particular disk will be read only into local memory, and only processed on local cores, all within that silo. This avoids expensive cross-socket communication and greatly improves throughput.

© 2026 Ocient, Inc.



Related Links

Secondary Indexes

Understanding Data Types

Related Videos

At the Whiteboard with Ocient: SQL at Scale

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Plan Optimization

A foundational element of any high-performance database is its plan optimization engine. Recall that a ***query plan*** is in many ways similar to a generic computer program. A key distinction, however, is that its machine code is in the language of relational algebra, set operations, input or output (I/O), network, and other database primitives. When a database converts SQL queries into a query plan, it must always generate a plan that produces the correct query results.

An over-used joke at Ocient® is that if you could return incorrect results, you could make a blazingly fast database by simply returning random numbers for every query. By way of analogy, this correct result requirement is the same as saying that a C++ compiler must always produce x86 machine code that correctly executes the originally provided instructions. But just as a specified C++ program can have many different possible representations in x86 machine code (different compilers, optimizations, etc) it is also the case that a given SQL query can have many (infinitely, actually) different query plans.

In a database, the optimizer's job is to take an already-correct plan that is naively generated from a SQL statement and produce an alternative plan that will emit the same results. This is normally viewed as a minimization problem where the alternative plan is "less than" the original plan in some metric (usually estimated run time), and database engines have evolved extremely varied approaches to solving it. Some challenges to consider:

1. The search space for the minimization problem is infinite. This means that any algorithm must eventually converge to some answer but do so in a reasonable amount of time. It should not take 100 seconds to shave 1 second off of a 2-second query.
2. Plans are compared to each other based on some metric like execution time, but this metric is not trivial to build. Having a metric function `M(p)` that returns a useful number and is cheap enough to compute is a challenge in and of itself. Adding to this challenge is the additional considerations beyond measuring the computational cost of plan execution, namely disk and network I/O costs as well.

© 2026 Ocient, Inc.

For these challenges, most database engines incorporate some sort of statistical approach. Statistics on the tables being queried (calculated either at data load time, or periodically updated) are used to bound execution and I/O estimates. The optimizer of the Ocient System does this as well. While basic statistics like row counts are certainly considered, more advanced mathematical approaches are used as well:

1. *Multi-dimensional probability density functions*. If a table has `j` columns, then it can make use of a `j` dimensional "table space" function `f(v0, v1, v2, ..., vj)` that maps a particular set of column values to the probability of those values. The Ocient System can then compute approximate row counts in the table by taking integrals of `f` in some hyper region of the table space.
2. *HyperLogLog*. In addition to row counts, the ability to estimate the number of *distinct* values is important as well. The basic idea behind HyperLogLog is that if a set has a uniformly randomized set of values, and `n` is the maximum number of leading zeros in the binary representations of all values in the set, then a good estimate of the set's cardinality is `2^n`. Here, each column is considered a set, and this statistical approach can estimate the number of distinct values for each column.

With such mathematical machinery available, you can see how the beginnings of a metric function for a plan can be built, but a few important details remain. Firstly, while the statistical approaches can make for good estimates, the Ocient System must be able to transform from the domain of SQL types into numbers for each supported SQL type. This is sometimes straightforward in cases like integral and floating point types, but becomes arbitrary and less obvious with things like variable length strings and arrays. Secondly, and more importantly, Ocient must also be able to calculate the metric on *sequences* of operators, each of which can be considered as a separate transformation of input to output. Put another way, if given a sequence of plan operators where each operator is successively operating on rows `T -> P0 -> P1 -> P2 ...`, then the raw column cardinality estimates of the columns in `T` might be of limited use in calculating a metric on `P2` because `P0` and `P1` can drastically alter the statistics of the original table.

In order to account for this, the optimizer must have mathematical models of each operator that can be applied to *input statistics* to produce *output statistics*. Then, successive application of operators can be used to build up plan metrics comprised of per-operator metrics. Clearly, successive approximation incurs increasing error. But an

interesting property of using a metric in this manner is that the exact error is not important. Instead, all that is necessary is that the accumulated errors in any estimation must not invalidate the relative comparison between two plans.

The topic of plan optimization is large and complicated, and the Ocient optimizer is a key and integral component of its approach. It incorporates not only detailed implementations but also an integrated view of the distributed nature of the query execution against NVMe drives. Each iterative version of the optimizer also includes an ever-increasing set of optimizations, giving it a sophisticated tool chest of transforms to apply to plans as well.

Related Links

Query Ocient

Key Concepts

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Exabyte Scalability Design

Cluster of Clusters

Various system components in Ocient® have each been designed to scale not only with hardware capabilities as technology progresses, but also in number of nodes storing and processing data. Like any distributed system, issues of fault tolerance, availability, and communication all become increasingly challenging as the number of units increases.

The Ocient scalability approach can be broadly described as deriving from ***clusters of clusters***: smaller, locally connected clusters that manage their own availability and local state independently and without consideration to any other clusters in the system. The design goal of this is to minimize global state and global synchronization in any aspect of query processing, fault management, data loading, or data ownership.

There are different kinds of clusters: **metadata** clusters, **storage** clusters, **VM** clusters, and **loading** clusters. An Ocient deployment can contain zero of more of each. With respect to scalability, each cluster runs its own Raft-based consensus protocol and maintains its own local shared state with it. Individual nodes can belong to multiple clusters and participate in multiple disjoint consensus protocols at the same time, but system scalability derives from the fact that no node has any global view of system state, and indeed, no such global state exists.

This allows Ocient to quantify the upper bound on the `N` parameter described in the erasure coding section. Recall that `N` is the number of segments in a segment group. In a **storage cluster**, segment ownership and availability is confined to and maintained in the cluster-local consensus protocol. A necessary implication of data management rules for atomic (but still local) state changes is that *a particular segment group must be stored entirely within one storage cluster.* And because the Ocient System cannot place two segments from the same group onto the same node, the value `N` cannot exceed the number of nodes in a target storage cluster. This finally means that the upper bound on `N` is the practical limit on the consensus protocol implementation, coupled with the design goal of not allowing clusters to be “too big”: somewhere in the range of 20 - 40, depending on deployment and schema details.

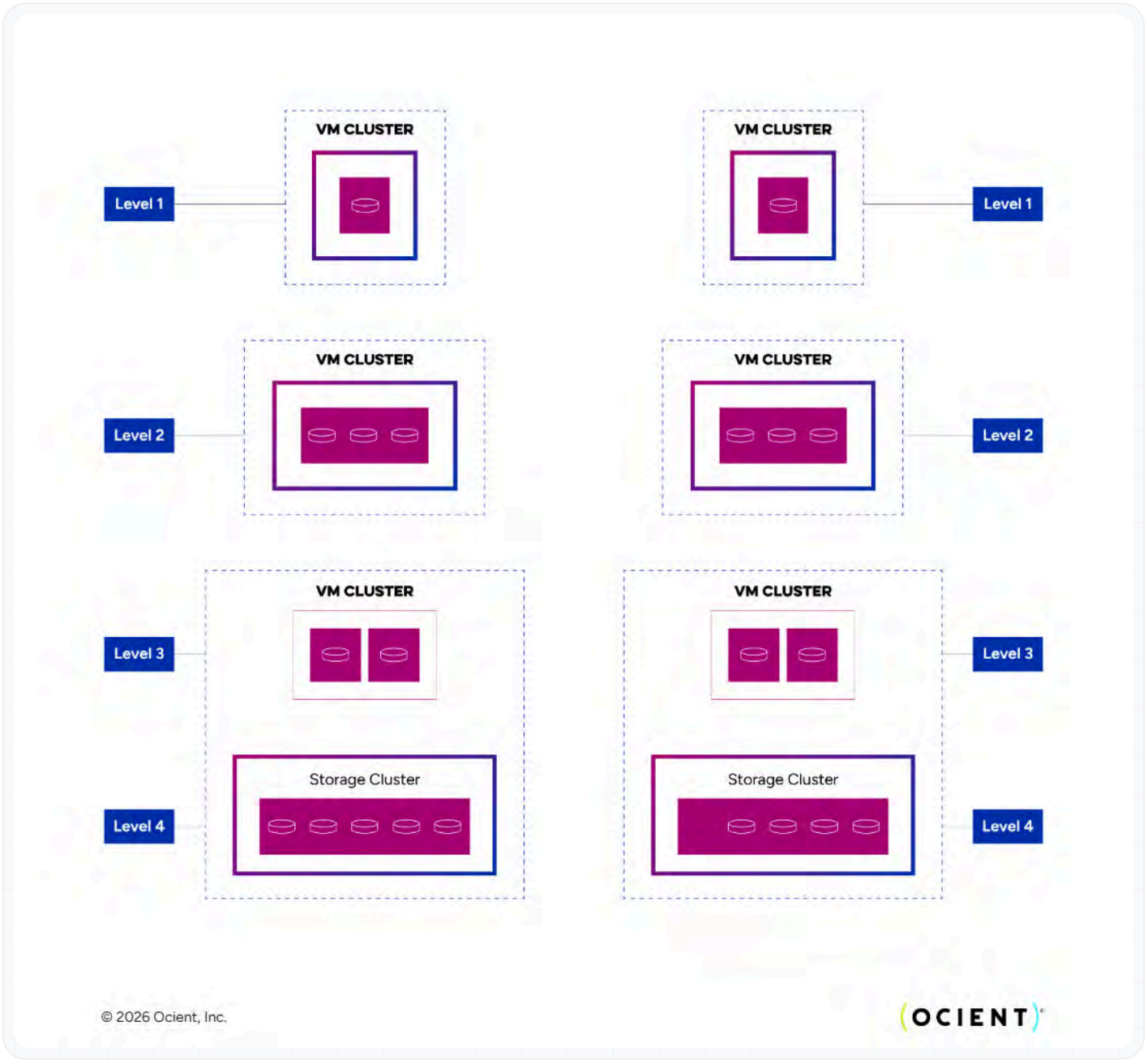
Query Trees (and Plan Trees)

Query execution necessarily spans multiple VM and storage clusters. A key phase of query processing, prior to any actual computation is the ***query probe***, which establishes the ***query tree*** to be used for that particular query. This tree fixes, for the particular execution with which it is associated, the set of nodes that provide which segments across *all* storage clusters, as well as the set of nodes that co-participate in the distributed execution of the query. The former is

© 2026 Ocient, Inc.

important because, as nodes and drives come and go, and as new data is loaded and old data is removed, it is imperative that all segments that should be included in a query are actually included exactly one time. The latter is important because each node might need to know exactly which subset of peers in the network are performing which aspects of query execution. In both cases, the probe phase enables query execution to proceed without any global coordination or synchronization.

This graphic shows a query execution tree, which can scale up by increasing the number of clusters (represented horizontally in the graphic).



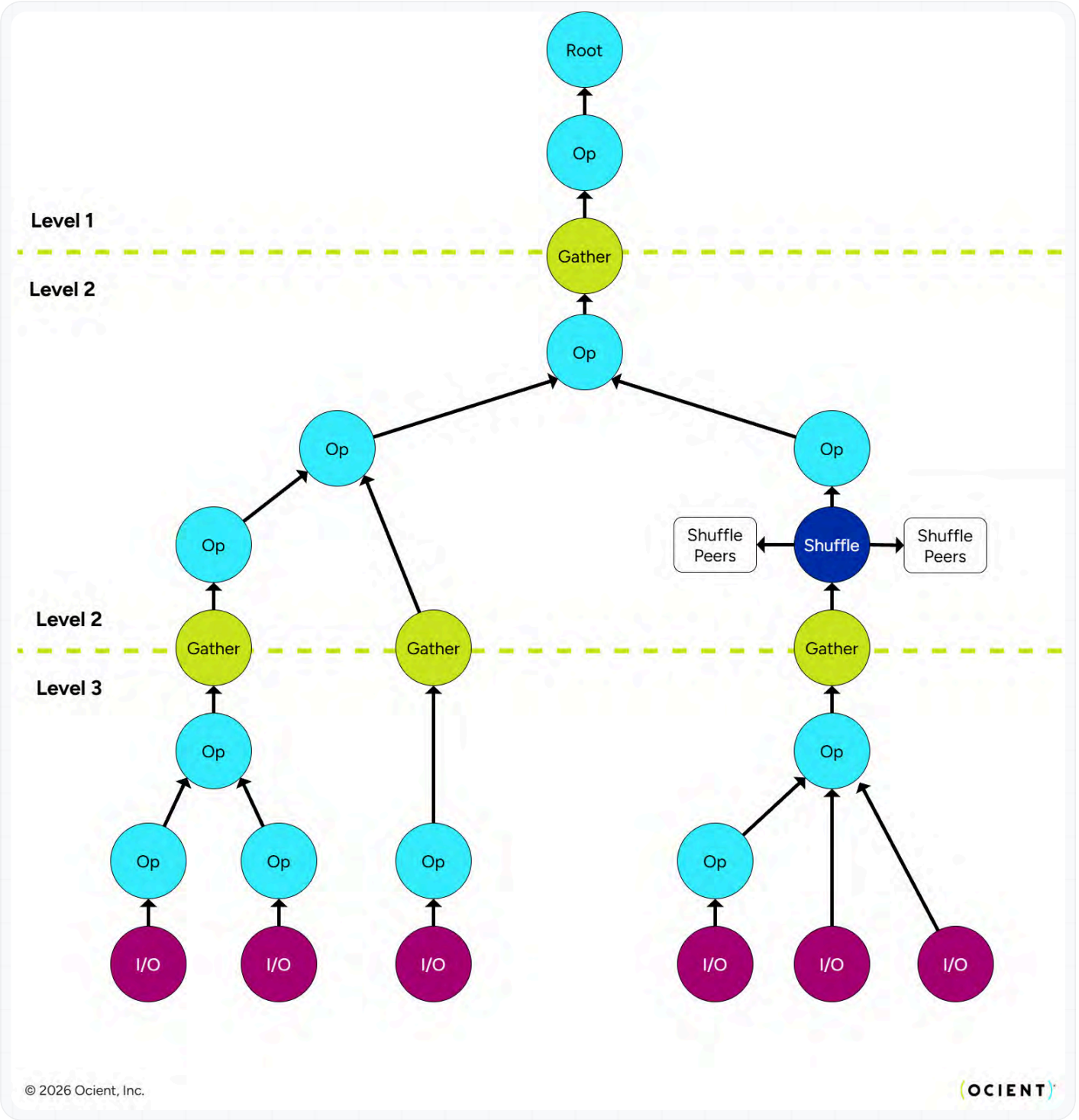
At the top level of the query execution tree is the root operator, which has zero or more additional plan levels below it, depending on the complexity of the query. Each circle represents individual nodes, which provide segments across the storage clusters or specific tasks for the execution of the query.

Vertically, the graphic shows different plan levels that fan out among the nodes for the stages of the query plan, actions such as transformations, sorting, grouping or other operations. The leaf level at the bottom of the graphic represents the I/O operation that reads the queried data from storage.

When the system initiates query execution, each cluster executes its operations in parallel. For the most part, clusters operate in isolation unless they need to communicate for operations such as a join or shuffle.

When referring to the query plan, each level is delineated by the presence of a special ***gather operator*** that represents the network boundary between levels. All the plan operators between root and the first gather operator are said to *execute at level 1.* Any plan operators that exist between the first gather operator and the second gather operator execute at level 2, and so on, until the final gather operator and the set of operators below it until the leaf I/O operators that represent the leaf level.

© 2026 Ocient, Inc.



When referring to the query tree, each gather operator represents a fan-out from one level N gather operator to some fixed number of N+1 nodes. During execution, a specific level N node is responsible for coordinating and amalgamating partial query results from the level N+1 nodes below it. By definition, there is exactly one level 1 node in a query, fanning out to some number of level 2 nodes, each of which itself fans out to some number of level 3 nodes, and so on. The fan-out factor is closely aligned with the cluster sizes in use.

In both cases, the tree structure promotes scalable query execution in a fault-tolerant way.

Related Links

Multiple Storage Clusters for Loading Data

Query Ocient

Related Videos

At the Whiteboard with Ocient: Scale

© 2026 Ocient All Rights Reserved

Loading Characteristics and Concepts

When considering trillions or quadrillions of distinct records being stored and accessed as part of an ever-growing data warehouse, the manner in which data is loaded is important. The loading infrastructure of the data warehouse must not only provide strong correctness and durability guarantees in service of modern data pipelines, but must do so at very high load rates as well. For example, if an individual record averages 1000 bytes in size and the data warehouse must store 100 trillion records, the size of the records to be loaded is 10^14 * 10^3 = 10^17 bytes, or 100 petabytes. Note that this is *pre-compression* because this is the volume of data that will be presented, row by row, to the data warehouse. If this information represents 1 year's worth of records, the loading system must be prepared to handle a steady-state average of ~25 Gbps. When the data rate is not completely uniform in time (perhaps displaying cyclic behavior associated with the day/night cycle) it is easy to imagine the required load rate can measure in the 100's, or even 1000's of Gbps.

Obviously, the sizes and rates are arbitrary, but they are not unrealistic. Consequently, a high-throughput, scalable, and robust loading infrastructure is a first-class citizen in an on-par complexity-wise with the execution engine itself.

Record-Level Resolution

The distinction between "stream" and "batch" is somewhat arbitrary, depending on the system in consideration, but in the Ocient® loading infrastructure the fundamental unit of loading is a single record in a table. Put another way, the various guarantees listed here are made on a record-by-record basis and not on batches of records. Supporting this is the concept of a ***record stream*** of which the loading infrastructure can manage an unbounded number. A record stream is an ordered sequence of records, and loading considers each record in a stream independently of each other.

Low Latency Querying

A key design goal of the loading system was a low ***time to query-ability*** or ***TTQ***. Strict guarantees are not made but this usually means that any record presented to the loading system is available to be queried on the order of seconds. An interesting property of the implementation is that the variance on TTQ *decreases* as the aggregate record rate *increases.* This is because the loading system maintains internal structures named ***pages*** that accumulate records, and the faster they fill the more quickly they are made available for querying.

© 2026 Ocient, Inc.

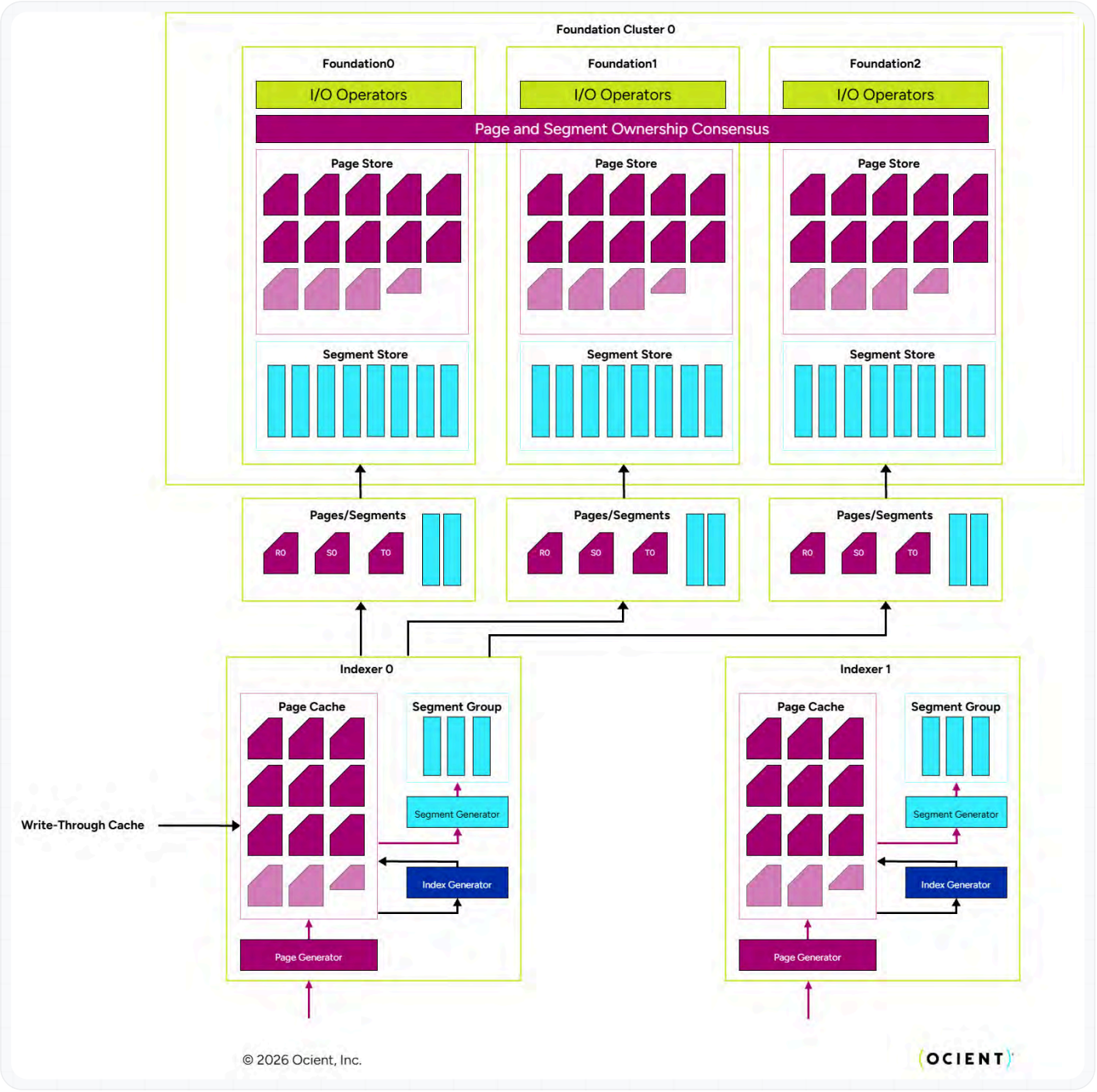
Low Latency Durability

A necessary requirement of low-latency querying is the ability to ensure that any particular record is stored in a fault-tolerant manner. To guarantee correctness of query results it must be the case that when a record is considered during query execution (either included or filtered out) it must be considered for *all* subsequent queries until it is deleted by the user. The ***durability*** guarantee made by the loading system must come before query-ability (and therefore is measured in seconds on a record-by-record basis) and matches the system's configured fault tolerance. For example, if the erasure coding scheme in use has a `K` value of 2, then the loading system must store all records in a manner that allows for at least 2 losses before they can be made queryable.

The unit of durability in the loading system is named a ***page*** and within the loading and storage system, a page shares many characteristics with the segments: each page is a self-contained set of records and their own local metadata and statistics, and the consensus-based ownership scheme mediated by the storage clusters is the same for both. However, there are some key distinctions:

1. **Typical Size** — A page is usually substantially smaller than a segment. While segments can be multiple gigabytes in size, pages are more commonly around 100 megabytes.
2. **Replication** — Segments are grouped together into segment groups and mutually erasure coded for fault tolerance. Pages, however, are *replicated* within the storage layer. This is less space efficient for the same fault tolerance, but pages are short-lived: they exist only long enough to be converted into segments, which are denser and richer in metadata. Because they do not exist for very long, the extra storage overhead of the replication scheme is inconsequential, and the loading system opts not to waste the computational effort computing the erasure-coded parity data.

© 2026 Ocient, Inc.



Exactly-Once Guarantee

When the record stream sources can support it, the Ocient loading infrastructure is capable of providing exactly-once semantics for every record loaded. This means that the record source and the query layer can be guaranteed that every record presented for load is not only not lost (no “gaps” exist) but also not duplicated. On the surface, this seems like an obvious and simple requirement, but at millions of records per second in a fault-tolerant distributed system, it is extremely non-trivial to provide this guarantee.

The loading systems of other database engines can also provide this guarantee, but often with caveats that might not be acceptable. For example, executing a sequence of `INSERT INTO` statements within the context of a transaction is a possible approach, but distributed transaction

processing does not easily scale to millions of records per second. Beyond that, there is the issue of how to manage inter-transaction guarantees that place additional requirements on the data sources.

If transactions are not in use, it is generally not possible to provide exactly-once guarantees without a bi-directional contract executed by the record source and the loading system. In a very common form of this contract, the loading system provides the exactly-once guarantee by guaranteeing it will de-duplicate any records presented more than once (i.e., the loading system is *idempotent* with respect to record insertions). Then, the source can safely re-send any records the loading system does not indicate it has received. So long as the source resends anything it has not been explicitly told was received (and made durable) the two endpoints can together guarantee every record will be present exactly one time.

When this approach is taken, the loading system's job becomes simple deduplication. On some systems, this is achieved using unique record identifiers. First, the source must be able to indicate a column or set of columns whose values taken together for any particular record uniquely identify it. Then, for each record inserted, a hash set or other structure is consulted. If the unique identifier is not present, it is inserted, if it is, the record is silently dropped. While technically cheap to implement, this scheme suffers because the set structure cannot be allowed to grow indefinitely, and therefore some time or size based bound is placed on the exactly-once guarantee. An additional significant issue is that it is challenging and expensive to ensure the set structure is replicated or distributed amongst all nodes participating in the loading system. This presents correctness and scalability challenges to the exactly-once guarantee.

The Ocient loading system takes the idempotent deduplication approach, but performs its deduplication checks in a different manner. The crux of the scheme is that most systems that aim for exactly-once guarantees are also able to replay their record streams *in the same order*. This is true for queuing systems such as Apache® Kafka® as well as file sources such as Apache® Parquet™ files stored in Amazon® Web Services℠ (AWS℠) S3. Due to this property, the Ocient deduplication scheme tracks a ***durability horizon*** for each distinct record stream. Recall that a record stream is an *ordered* sequence of records. The durability horizon is an ever-advancing offset into a stream that indicates the record at which the data warehouse guarantees all previous records up to and including that record, for that stream, are guaranteed durable. Using this scheme, the source can easily identify the set of records it should send or re-send. In the loading system, deduplication becomes cheap in both time and space: the system need only store the durability horizon per stream (instead of an unbounded set), and deduplication checks are as simple as a basic interval check comparing whether or not sequences of records stored in pages overlap.

Linear Scalability at High Rates

In order to enable the extremely high record and bit rates associated with hyperscale data pipelines, it is important that the loading system be capable of scaling linearly to address arbitrary loading complexity. The Ocient loading layer achieves true linear scalability due to a shared-nothing approach between nodes executing loading operations. Being linearly scalable, it is relatively straightforward to design a fault-tolerant deployment that can handle any arbitrary rate by simply multiplying by the capabilities of a single node for a specified schema.

With that said, it is almost impossible to specify an exact achievable load rate on a per-node basis in advance. This is because the process is bound by the computational complexity of calculating indexes and metadata (which are schema-specified), the width of the erasure coding scheme (which are user-specified), and the bit rate being handled (which is both schema *and* data-dependent). With that said, each Loader Node is usually bounded by memory bandwidth, and it is not uncommon to be capable of steady-state load rates per node measuring in dozens of gigabits per second or millions of records per second. Usually, nodes are also capable of bursting to higher rates: 10’s of millions of records per second and approaching 100 Gbps.

Conclusion

The architecture is designed to maximize performance on the world’s largest datasets. The massively parallel Compute Adjacent Storage Architecture® features storage and query processing optimizations to achieve scalable performance on commodity hardware with NVMe SSD storage. The custom-built, fully-integrated SQL execution engine and optimizer deeply integrate with the storage and the input and output layer. The result is a state-of-the-art data warehouse that can deliver outstanding performance loading and querying record sets numbering in the trillions, quadrillions, and beyond.

Related Links

Ingest Data with Legacy LAT Reference

Key Concepts

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

SQL Reference

This SQL Reference section includes details about the different data types, functions, and operators available when writing queries for Ocient®. For an introduction to the general query syntax and working with queries in Ocient, see **Query Ocient**.

For information on the SQL syntax conventions used throughout the Ocient documentation, see the **SQL Syntax Conventions** page.

Data Types

- **Data Types**

Functions & Operators

- **Alphabetical SQL Functions List**
- **Categorized SQL Functions List**
- **Aggregate Functions**
- **Window Aggregate Functions**
- **Scalar Data Conversion Functions**
- **Date and Time Functions**
- **Time Zone Functions**
- **Formatting Functions**
- **Math Functions and Operators**
- **Character and Binary Functions**
- **Matrix Functions and Operators**
- **Tuple Functions and Operators**
- **Other Functions and Expressions**
- **Array Functions and Operators**
- **HyperLogLog Functions**

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

- Network Type Functions

Geospatial Functions & Operators

- Attribute Functions
- Conversion Functions
- Linestring Functions
- Linestring Constructors
- Point Constructors
- Polygon Constructors
- Spatial Measurement
- Spatial Operators
- Spatial Relationships
- Spatiotemporal Measurement
- Spatiotemporal Operators

DDL SQL Statements

- Cluster and Node Management
- Databases
- Schemas
- Tables
- Views
- Indexes
- Data Pipelines
- Distributed Tasks
- Machine Learning Models
- Users, Groups, and Service Classes
- Data Integrity and Storage
- Query Analysis

© 2026 Ocient, Inc.

- Query Management
- Statistics Cache Management

DCL SQL Statements

- Data Control Language (DCL) Statement Reference

DML SQL Statements

- Data Manipulation Language (DML) Statement Reference

DQL SQL Statements

- Data Query Language (DQL) Statement Reference
- Join Operations
- SQL Syntax Conventions
- Identifiers
- Generate Tables Using sys.dummy

TCL SQL Statements

- Transactional Control Language (TCL) Statement Reference

Machine Learning Execution

- Machine Learning Model Functions
- Data Preparation
- Machine Learning Model Options

Other SQL Reference

- Errors and Warnings

- **Data Pipeline Loading Errors**
- **Reserved Words**
- **System Catalog Reference**
- **System Catalog**
- **Information Schema**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Alphabetical SQL Functions List

Function Name	Category Name	Function Description
ABS	Math Functions	Returns the absolute value of a spec
ABS	Matrix Functions	Returns the magnitude of one-dimer
ACOS	Math Functions	Returns the inverse cosine of a spec
ACOSH	Math Functions	Returns the hyperbolic arc-cosine of
ACCURACY_SCORE	Aggregate Functions	Returns the fraction of predictions th
ADD_MONTHS	Date and Time Functions	Adds the specified number of month
ANY_VALUE	Aggregate Functions	Returns an arbitrary, non-NULL valu
APPROX_COUNT_DISTINCT	Aggregate Functions	Approximate distinct count by using
APPROX_SUM	Aggregate Functions	Allows the aggregation engine to us
ARRAY[]	Array Functions	PostgreSQL®-compliant constructor
ARRAY_AGG	Aggregate Functions	Returns an array containing every ro
ARRAY_APPEND	Array Functions	Add value to the back of an array.
ARRAY_ARGMAX	Array Functions	Returns the corresponding argmax o
ARRAY_ARGMIN	Array Functions	Returns the corresponding argmin o
ARRAY_CAT	Array Functions	Concatenate 2 arrays into a new one
ARRAY_CAT_DISTINCT	Array Functions	Concatenates two or more arrays in
ARRAY_CAP	Special Data Pipeline Transformation Functions	Restrict the length of an array to a r

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
ARRAY_COMPACT	Special Data Pipeline Transformation Functions	Removes NULL values from the arra
ARRAY_CONCAT_AGG	Aggregate Functions	Returns an array that concatenates
ARRAY_CONTAINS	Special Data Pipeline Transformation Functions	Returns `true` if the array contains
ARRAY_DISTINCT	Array Functions	Removes duplicates from an array w
ARRAY_LENGTH	Array Functions	Return the number of elements of a
ARRAY_MAX	Array Functions	Returns the corresponding maximur
ARRAY_MIN	Array Functions	Returns the corresponding minimum
ARRAY_POSITION	Array Functions	Returns the position of the first mat
ARRAY_POSITIONS	Array Functions	Returns all elements stored in the ri
ARRAY_PREPEND	Array Functions	Add value to the front of an array.
ARRAY_REMOVE	Array Functions	Remove value from the array.
ARRAY_REPLACE	Array Functions	Replace value by another in an arra
ARRAY_SORT	Special Data Pipeline Transformation Functions	Sort and return the input array base
ARRAY_SUM	Array Functions	Returns the sum of the array. The ar contribute to the sum.
ARRAY_TO_STRING	Array Functions	Converts array to a list of elements
ASCII	Character and Binary Functions	Returns the ASCII code value of the
ASIN	Math Functions	Returns the inverse sine of a specifi
ASINH	Math Functions	Returns the hyperbolic arc-sine of a
ATAN	Math Functions	Returns the inverse tangent of a spe
ATAN2	Math Functions	Returns the inverse tangent of two r
ATANH	Math Functions	Returns the hyperbolic arc-tangent

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
AVG	Aggregate Functions	Average, or arithmetic mean, over t
BICDF	Math Functions	The cumulative distribution function
BIGINT	Scalar Data Conversion Functions	Casts the argument to a value of typ
BINARY	Scalar Data Conversion Functions	Parses a hexadecimal string (such a
BIPDF	Math Functions	The probability density function of tl
BIT_LENGTH	Character and Binary Functions	Returns the length of the character
BITAND	Math Functions	Alias for the BITFUNC syntax `BITFUI`
BITFUNC	Math Functions	Performs a variety of bit operations.
BITNOT	Math Functions	Returns the bitwise negation of inte
BITOR	Math Functions	Alias for the BITFUNC syntax BITFUN
BITXOR	Math Functions	Alias for the BITFUNC syntax BITFUN
BOOLAND	Math Functions	Alias for the BOOLFUNC syntax BOO
BOOLEAN	Scalar Data Conversion Functions	Parses a string to create a Boolean v
BOOLFUNC	Math Functions	Performs a Boolean logical evaluatic `'XOR'`.
BOOLNOT	Math Functions	Returns the logical negation of the E
BOOLOR	Math Functions	Alias for the BOOLFUNC syntax BOO
BOOLXOR	Math Functions	Alias for the BOOLFUNC syntax BOO
BTRIM	Character and Binary Functions	Alias for TRIM.
BYTE	Scalar Data Conversion Functions	Casts the argument to a value of typ
CANCEL	Query Management	Cancels a running query based on it

Function Name	Category Name	Function Description
CASE	Conditional Functions	CASE operates similarly to condition then / else statement or as a switch
CASE WHEN	Special Data Pipeline Transformation Functions	Returns the `result` value based or
CAST_TO_ARRAY	Array Functions	Casts the elements of the array to a
CAST_TO_TUPLE	Tuple Functions	Converts a tuple into another tuple
CBRT	Math Functions	Returns the cube root of the numeri
CDF	Math Functions	The cumulative distribution function sample is less than or equal to the s
CEIL	Math Functions	Returns the nearest integer greater
CEILING	Math Functions	Alias for CEIL.
CENTURY	Date and Time Functions	Returns the number of centuries.
CHAR	Array Functions	Converts array to its string represen
CHAR	Scalar Data Conversion Functions	Creates a string version of the nume
CHAR	Tuple Functions	Converts a tuple to its string represe
CHAR_LENGTH	Character and Binary Functions	Alias for LENGTH.
CHARACTER_LENGTH	Character and Binary Functions	Alias for LENGTH.
CHR	Character and Binary Functions	Converts an integer value to a string
COALESCE	Conditional Functions	Evaluates to the first argument that
COEFFICIENT_OF_DETERMINATION	Aggregate Functions	Computes the coefficient of determi
COMMIT	System Functions	Returns the most recent commit has
CONCAT	Character and Binary Functions	Concatenates two values, which mu the || operator.

Function Name	Category Name	Function Description
CONFUSION_MATRIX	Aggregate Functions	Returns a structured representation to a specified positive class.
CONVERT_LOCAL_TIMESTAMP_TO_UTC	Time Zone Functions	The function converts a timestamp i
CONVERT_UTC_TIMESTAMP_TO_LOCAL	Time Zone Functions	The function converts a timestamp f
CORR	Aggregate Functions	Alias for CORRELATION.
CORRELATION	Aggregate Functions	Sample correlation.
CORRELATIONP	Aggregate Functions	Population correlation.
COS	Math Functions	Returns the cosine of x.
COSH	Math Functions	Returns the hyperbolic cosine of x.
COT	Math Functions	Returns the cotangent of x.
COUNT	Aggregate Functions	Number of rows in the set.
COVAR_POP	Aggregate Functions	Alias for COVARIANCEP.
COVAR_SAMP	Aggregate Functions	Alias for COVARIANCE.
COVARIANCE	Aggregate Functions	Sample covariance.
COVARIANCEP	Aggregate Functions	Population covariance.
CROSS_ENTROPY_LOSS	Array Functions	Returns the cross entropy loss of tw
CROSS_ENTROPY_LOSS	Matrix Functions	Returns the cross entropy loss of tw
CUME_DIST	Window Aggregate Functions	Returns a number 0 < n 1 and can b value in the group.
CURDATE	Date and Time Functions	Alias for CURRENT_DATE.
CURRENT_DATABASE	System Functions	Alias for DATABASE.
CURRENT_DATE	Date and Time Functions	Returns the current date in the form

Function Name	Category Name	Function Description
CURRENT_GROUPS	System Functions	Returns the fully-qualified names of
CURRENT_NODE	System Functions	Returns the name of the SQL Node v column in the `sys.nodes` system ca
CURRENT_NODE_ID	System Functions	Returns the identifier of the SQL Noc
CURRENT_SCHEMA	System Functions	Returns the name of the current sch
CURRENT_SESSION_ID	System Functions	Returns the Universally Unique IDen
CURRENT_SYSTEM	System Functions	Returns the name of the system.
CURRENT_TIME	Date and Time Functions	Returns the current time as a TIME v
CURRENT_TIMESTAMP	Date and Time Functions	Returns the current date and time a
CURRENT_USER	System Functions	Returns the user for the current con
DATABASE	System Functions	Returns the name of the database to
DATE	Scalar Data Conversion Functions	Parses a string in the form 'YYYY-MM
DATE_PART	Date and Time Functions	Alias for EXTRACT.
DATE_TRUNC	Date and Time Functions	Returns the date or timestamp ente
DATEADD	Date and Time Functions	Adds a specified number value (as a that modified value.
DATEDIFF	Date and Time Functions	This function returns an INT represe unit.
DAY	Date and Time Functions	Alias for DAY_OF_MONTH.
DAY_OF_MONTH	Date and Time Functions	Extracts the day-of-month portion of
DAY_OF_WEEK	Date and Time Functions	Returns an integer, in the range of 1
DAY_OF_YEAR	Date and Time Functions	Returns an integer in the range 1 to

Function Name	Category Name	Function Description
DAYS	Scalar Data Conversion Functions	Converts an integral value to an inte
DECADE	Date and Time Functions	The decade is the year divided by 1
DECIMAL	Scalar Data Conversion Functions	Casts the argument to a value of typ
DEGREES	Math Functions	Returns the corresponding angle in
DELTA	Window Aggregate Functions	Computes the finite difference betw backwards difference, which means of expression for that row and the p
DENSE_RANK	Window Aggregate Functions	Assigns a number to each row in the between ranks.
DERIVATIVE	Window Aggregate Functions	Computes the difference quotient be
DET	Matrix Functions	Returns the determinant of the matr
DIV	Math Functions	Returns the result of x divided by y.
DOT	Matrix Functions	Returns dot product of two one-dime
DOUBLE	Scalar Data Conversion Functions	Casts the argument to a value of typ
DOW	Date and Time Functions	Alias for DAY_OF_WEEK.
DOY	Date and Time Functions	Alias for DAY_OF_YEAR.
EIGEN	Matrix Functions	Returns eigenvalues and eigenvalue
ELEMENT_AT	Special Data Pipeline Transformation Functions	Returns the element value of the arr
ENDSWITH	Character and Binary Functions	Returns true if x ends with y and fals
EOMONTH	Date and Time Functions	Returns the last day of the timestam

Function Name	Category Name	Function Description
EPOCH	Date and Time Functions	The number of seconds after 1970-(
ERF	Math Functions	The error function is used for measu
ERFC	Math Functions	The complement of the error functic
EXP	Math Functions	Returns the exponential of x (e raise
EXPLODE_OUTER	Special Data Pipeline Transformation Functions	Expands a one-dimensional or multi system.
EXTRACT	Date and Time Functions	Extract a component from a timesta
F1_SCORE	Aggregate Functions	Returns the harmonic mean of preci
FILTER	Special Data Pipeline Transformation Functions	Filters elements in an array based o
FIRST_VALUE	Window Aggregate Functions	Returns the first value in the ordere
FLATTEN	Special Data Pipeline Transformation Functions	Transforms an `N`-dimensional array
FLOAT	Scalar Data Conversion Functions	Casts the argument to a value of typ
FLOOR	Math Functions	Returns the nearest integer less tha
FROBENIUS	Matrix Functions and Operators	Returns the Frobenius norm of a ma
GAMMA	Math Functions	**Gamma function.**
GREATEST	Conditional Functions	Returns the largest non-NULL value
HASH	Scalar Data Conversion Functions	Creates a fixed length binary value, string does not have enough bytes,
HEXBINX	Math Functions	Returns the x-coordinate of the cent

Function Name	Category Name	Function Description
HEXBINY	Math Functions	Returns the y-coordinate of the cent
HINGE_LOSS	Array Functions	Returns the hinge loss of two arrays
HINGE_LOSS	Matrix Functions	Returns the hinge loss of two one-di
HLL_SKETCH_CREATE	HyperLogLog Functions	Creates an HLL sketch from the data representation of the sketch that yo
HLL_SKETCH_GET_ESTIMATE	HyperLogLog Functions	The scalar function converts a sketc estimate as a BIGINT.
HLL_SKETCH_GET_ESTIMATE_BOUND	HyperLogLog Functions	Takes a HLL_SKETCH column or an ir confidence interval error proportion
HLL_SKETCH_TO_STRING	HyperLogLog Functions	The HLL_SKETCH_TO_STRING scalar the sketch.
HLL_SKETCH_UNION (aggregate function)	HyperLogLog Functions	Merges multiple sketches in a single function is an aggregate function an
HLL_SKETCH_UNION (scalar function)	HyperLogLog Functions	Merges two sketches into a new con function merges two sketch columns
HOUR	Date and Time Functions	Extracts the hour portion of a timest
HOURS	Scalar Data Conversion Functions	Converts an integral value to an inte
IDENTITY_MATRIX	Matrix Functions	Returns an identity matrix of the spe
IERF	Math Functions	The inverse of the ERF error functior
IERFC	Math Functions	The inverse of the complement of th
IF	Special Data Pipeline Transformation Functions	Returns `T` if the expression `X` eva
IF_NULL	Conditional Functions	Alias for COALESCE.
INITCAP	Character and Binary Functions	For each word in the provided string
INSTR	Character and Binary Functions	Returns the index position of the firs value char by ignoring case.
INTEGER	Scalar Data Conversion	Casts the argument to a value of typ

Function Name	Category Name	Function Description
	Functions	
INVERSE	Matrix Functions	Returns inverse of a square, invertib
IP	Network Type Functions	Casts an IP data type from an IPV4 c
IPV4	Network Type Functions	Casts an IPV4 address from an IPV6
ISDATE	Date and Time Functions	Returns TRUE if the input argument
IS_IPV4	Network Type Functions	Tests whether the database can con
ISODOW	Date and Time Functions	Extracts the day of the week based
JSON_EXTRACT_PATH_TEXT	Character and Binary Functions	Returns the value for the key-value
KILL	Query Management	Cancels a running query based on it
KURTOSIS	Aggregate Functions	The sample over the set.
KURTOSISP	Aggregate Functions	The population over the set.
LAG	Window Aggregate Functions	Returns the row, which is the specifi
LAG_VECTORS	Data Preparation	Groups lagged columns generated b
LAGS	Data Preparation	Generates a series of lagged columr
LAGS_ZEROFILL	Data Preparation	Generates lagged columns for a sing
LAST_VALUE	Window Aggregate Functions	Returns the last value in the orderec
LCASE	Character and Binary Functions	Alias for LOWER.
LEAD	Window Aggregate Functions	Returns the row, which is the specifi
LEAKYRELU	Math Functions	Returns the leaky rectified linear uni
LEAST	Conditional Functions	Returns the smallest non-NULL valu

Function Name	Category Name	Function Description
LEFT	Character and Binary Functions	Return the number of characters in t returns all but the last integer chara
LEFT_SHIFT	Math Functions	Returns x shifted to the left by y bits
LENGTH	Character and Binary Functions	For character data types, this value
LN	Math Functions	Returns the natural logarithm of x.
LOCATE	Character and Binary Functions	Alias for POSITION. Returns the inde value string.
LOG	Math Functions	Returns the base 10 logarithm of x. the function defaults to base 10.
LOG_GAMMA	Math Functions	The natural logarithm of the absolut
LOG_LOSS	Array Functions	Returns the log loss of two arrays.
LOG_LOSS	Matrix Functions	Returns the log loss of two one-dime
LOG2	Math Functions	Returns the base 2 logarithm of x.
LOGITS_LOSS	Array Functions	Returns the logits loss of two arrays
LOGITS_LOSS	Matrix Functions	Returns the logits loss of two one-di
LOOKUP	Special Data Pipeline Transformation Functions	Look up and load data in an externa
LOWER	Character and Binary Functions	Alias for LCASE. Convert string to lo
LPAD	Character and Binary Functions	Pad the input text to the specified le
LTRIM	Character and Binary Functions	Removes leading blanks from the st
LUPQ_DECOMP	Matrix Functions	Returns LUPQ decomposition of a sq
MAP_KEYS	Special Data Pipeline Transformation Functions	Returns the keys in the specified JSC
MAP_VALUES	Special Data Pipeline Transformation Functions	Returns the values in the specified J

Function Name	Category Name	Function Description
MATRIX_DIM	Matrix Functions	Returns the dimensions of the speci
MAKE_MATRIX_IXJ	Matrix Functions	Creates an `ixj` matrix with eleme
MAKEDATETIME	Date and Time Functions	Returns a timestamp consisting of t
MATRIX_FROM_TEXT	Matrix Functions	Creates a matrix from the specified
MATRIX_TRACE	Matrix Functions	Returns trace of a square matrix as
MAX	Aggregate Functions	Maximum value in the specified colu
MD5	Character and Binary Functions	Returns the hexadecimal string (all
MEAN_ABSOLUTE_ERROR	Aggregate Functions	Returns the mean absolute error (M
MEAN_ABSOLUTE_PERCENTAGE_ERROR	Aggregate Functions	Returns the mean absolute percenta
METADATA	Special Data Pipeline Transformation Functions	Extracts the metadata value for the
MICROSECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
MID	Character and Binary Functions	Alias for SUBSTRING.
MILLISECOND	Date and Time Functions	Extracts the millisecond portion of a
MILLISECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
MIN	Aggregate Functions	Minimum value in the specified colu
MINUTE	Date and Time Functions	Extracts the minute portion of a tim
MINUTES	Scalar Data Conversion Functions	Converts an integral value to an inte
MOD	Math Functions	Returns the remainder from x divide

Function Name	Category Name	Function Description
MONTH	Date and Time Functions	Extracts the month portion of a time
MONTH_NAME	Date and Time Functions	Returns the calendar name in Englis
MONTHS	Scalar Data Conversion Functions	Converts an integral value to an inte
MONTHS_BETWEEN	Date and Time Functions	Returns the difference between the
MSECS	Date and Time Functions	The seconds field, including fraction
MULTI_LAGS	Data Preparation	Generates lagged columns for multi
MULTI_LAGS_ZEROFILL	Data Preparation	Generates lagged columns for multi
MURMUR3	Conditional Functions	**Returns a 32-bit MurmurHash3 I**
NANOS_TO_TIMESTAMP	Date and Time Functions	Convert a number of nanoseconds i
NANOSECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
NEXT_DAY	Date and Time Functions	Returns the closest date after a spe
NOW	Date and Time Functions	Alias for CURRENT_TIMESTAMP.
NTH_VALUE	Window Aggregate Functions	Returns the nth value in the ordered
NULL_IF	Conditional Functions	Returns the NULL value if two argum
NULL_MATRIX	Matrix Functions	Returns a null matrix of the given (r
OCTET_LENGTH	Character and Binary Functions	Returns the length in bytes of a cha
PARSE_DELIMITED_ARRAY	Special Data Pipeline Transformation Functions	Converts a string of text data repres

Function Name	Category Name	Function Description
PERCENT_RANK	Window Aggregate Functions	The value returned is 0 < n ≤ 1 and excluding the highest value.
PERCENTILE	Window Aggregate Functions	Returns the value that corresponds
PI	Math Functions	Returns the constant value of π.
PMOD	Math Functions	Returns the smallest non-negative e
POSITION	Character and Binary Functions	Alias for LOCATE.
POWER	Math Functions	Returns x raised to the power of y.
PRECISION_SCORE	Aggregate Functions	Returns the precision score for a spe
PROBIT	Math Functions	The inverse of the cumulative distrib
PRODUCT	Aggregate Functions	Product over the set.
QR_DECOMP	Matrix Functions	Returns QR decomposition of a matr
QUARTER	Date and Time Functions	Returns an integer between 1 and 4
RADIANS	Math Functions	Returns the corresponding angle in r
RAND	Math Functions	Takes no argument and returns a ra
RAND_UUID	Other Functions and Expressions	Generates a random UUID value (ve
RANK	Window Aggregate Functions	Assigns a number to each row in the between ranks.
RATIO_TO_REPORT	Window Aggregate Functions	Computes the ratio of a value to the
RECALL_SCORE	Aggregate Functions	Returns the recall score for a specifi
ROC_AUC_SCORE	Aggregate Functions	Returns the area under the receiver
RECORD_UUID	Other Transformation Functions	Returns string that represents a unic topic.
REDUCE	Special Data Pipeline	Applies a merge function to a startir value. Optionally, specify a finish fu

Function Name	Category Name	Function Description
	Transformation Functions	
REGEXP_COUNT	Character and Binary Functions	Searches a string for all occurrences
REGEXP_INSTR	Character and Binary Functions	Searches a string using a regular ex position of the substring that match
REGEXP_REPLACE	Character and Binary Functions	Searches a string for all occurrences
REGEXP_SUBSTR	Character and Binary Functions	Returns one substring from a string
RELU	Math Functions	Returns the rectified linear unit func
REPEAT	Character and Binary Functions	Repeats the character value char a
REPLACE	Character and Binary Functions	Replaces all occurrences of `substr_`
REVERSE	Character and Binary Functions	Reverse the input string.
RIGHT	Character and Binary Functions	Return the number of trailing charac
RIGHT_SHIFT	Math Functions	Returns x shifted to the right by y bi
ROUND	Date and Time Functions	Returns the specified date or timest
ROUND	Math Functions	Returns x rounded to the nearest int
ROW_NUMBER	Window Aggregate Functions	Assigns a unique number to each ro
RPAD	Character and Binary Functions	Pad the input text to the specified le
RSUBSTRING	Character and Binary Functions	Returns the substring from the right
RTRIM	Character and Binary Functions	Removes leading blanks from the st
SECOND	Date and Time Functions	Extracts the seconds portion of a tin

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
SECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
SHA1	Character and Binary Functions	Uses the [SHA-1](**https://en.wikipe 1#:~:text=In%20cryptography%** cryptographic hash function to conv 160-bit checksum.
SHOW	System Functions	The SHOW function enables you to e
SIGN	Math Functions	Returns the positive (+1), zero (0), c
SIN	Math Functions	Returns the sine of x.
SINH	Math Functions	Returns the hyperbolic sine of x.
SKEW	Aggregate Functions	The sample over the set.
SKEWP	Aggregate Functions	Computes the population skewness
SMALLINT	Scalar Data Conversion Functions	Casts the argument to a value of typ
SOFTMAX	Array Functions	Returns the softmax of the array.
SOFTMAX	Matrix Functions	Returns the softmax of a one-dimen
SPACE	Character and Binary Functions	Returns a string of repeated spaces
SPLIT_PART	Character and Binary Functions	Split the value string based on the d the index value (starting from 1).
SPLIT_TO_ARRAY	Character and Binary Functions	Splits a string into an array of substr
SQRT	Math Functions	Returns the square root of x.
SQUARE	Math Functions	Returns the square of x.
ST_ADDPOINT	Geospatial Linestring Function	Adds a POINT to the given LINESTRI
ST_ANGLE	Geospatial Spatial Measurement	Calculates the angle between two li
ST_AREA	Geospatial Spatial Measurement	Returns the area of the specified ge

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
ST_ASBINARY	Geospatial Conversion Functions	Returns the well-known binary (WKB
ST_ASEWKT	Geospatial Conversion Functions	Returns a string that represents geo
ST_ASGEOJSON	Geospatial Conversion Functions	Alias of ST_ASBINARY.
ST_ASLATLONTEXT	Geospatial Conversion Functions	**Returns the GeoJSON represent**
ST_ASTEXT	Geospatial Conversion Functions	Alias of ST_ASTEXT.
ST_ASWKB	Geospatial Conversion Functions	Alias of ST_ASTEXT.
ST_ASWKT	Geospatial Conversion Functions	Alias of ST_ASWKT and ST_EWKT. Re
ST_AZIMUTH	Geospatial Spatial Measurement	Returns the azimuth of the line from
ST_BOUNDINGDIAGONAL	Geospatial Spatial Operators	Returns the diagonal LINESTRING fr ST_ENVELOPE returns.
ST_BUFFER	Geospatial Spatial Operators	Returns a geography that contains a specified distance.
ST_CENTROID	Geospatial Point Constructors	The geographic center of mass is ca projecting the resultant point onto tl
ST_CLOSESTPOINT	Geospatial Spatial Operators	Returns the two-dimensional POINT object.
ST_CLUSTERDBSCAN	Geospatial Spatial Relationships	**Returns the cluster number for the density-based spatial cluste**
ST_CONTAINS	Geospatial Spatial Relationships	Returns TRUE if the first geographic
ST_CONTAINSPROPERLY	Geospatial Spatial Relationships	Returns true if geo2 lies entirely in t points.

Function Name	Category Name	Function Description
ST_CONVEXHULL	Geospatial Spatial Operators	The convex hull is the smallest conv
ST_COORDDIM	Geospatial Attribute Functions	Alias for ST_NDIMS or ST_NDIMENSI(
ST_COVEREDBY	Geospatial Spatial Relationships	Returns TRUE if no POINT in geo1 is
ST_COVERS	Geospatial Spatial Relationships	Returns TRUE if no POINT in geo2 is
ST_CROSSES	Geospatial Spatial Relationships	Returns TRUE if two geospatial objec The intersection is not equal to geo1
ST_DIFFERENCEARRAY	Geospatial Spatial Operators	Returns an array containing any geo not found in the second geospatial a
ST_DIMENSION	Geospatial Attribute Functions	Returns an INTEGER that represents
ST_DISJOINT	Geospatial Spatial Relationships	Returns true if the specified geograp be different types.
ST_DISTANCE	Geospatial Spatial Measurement	Returns the minimum distance betw
ST_DISTANCE	Geospatial Spatiotemporal Measurement	Returns the two-dimensional interpo array pairs in the specified unit of m
ST_DISTANCESPHERE	Geospatial Spatial Measurement	Returns the minimum distance betw
ST_DISTANCESPHEROID	Geospatial Spatial Measurement	Returns the minimum distance betw
ST_DWITHIN	Geospatial Spatial Relationships	Returns TRUE if the geographies are
ST_ENDPOINT	Geospatial Linestring Function	Returns the endpoint of a specified l
ST_ENVELOPE	Geospatial Spatial Operators	Returns a POLYGON that represents
ST_EQUALS	Geospatial Spatial Relationships	Returns TRUE if both geographies ar
ST_EUCLIDEANDISTANCE3D	Geospatial Spatial Measurement	Returns the minimum distance betw

Function Name	Category Name	Function Description
ST_EXPAND	Geospatial Spatial Operators	Returns the bounding box of a speci
ST_EXTERIORRING	Geospatial Spatial Operators	Returns a LINESTRING that represen
ST_FLIPCOORDINATES	Geospatial Spatial Operators	Returns a new geographic object wi
ST_FORCE2D	Geospatial Spatial Operators	Convert a geographic object into a t
ST_FORCECCW	Geospatial Polygon Constructors	Creates a standardized polygon fron counterclockwise (CCW) and all hole
ST_GEOGPOINT	Geospatial Point Constructors	Creates and returns a POLYGON geo function.
ST_GEOHASH	Geospatial Conversion Functions	Returns a string that represents the
ST_GEOMETRYTYPE	Geospatial Attribute Functions	Returns a string representing the ge
ST_HAUSDORFFDISTANCE	Geospatial Spatial Measurement	Returns the Hausdorff distance betw
ST_INTERIORRINGN	Geospatial Spatial Operators	Returns a LINESTRING representing indexed)
ST_INTERSECTALL	Geospatial Spatial Operators	Returns the intersection of all geogr
ST_INTERSECTION	Geospatial Spatiotemporal Operators	Returns a tuple that represents the i
ST_INTERSECTIONARRAY	Geospatial Spatial Operators	Returns a geography that represents
ST_INTERSECTS	Geospatial Spatial Relationships	Returns TRUE if the specified geogra
ST_ISCCW	Geospatial Spatial Relationships	Alias for ST_ISPOLYGONCCW.
ST_ISCLOSED	Geospatial Spatial Relationships	Returns TRUE if an input POLYGON h
ST_ISEMPTY	Geospatial Attribute Functions	Returns TRUE if the specified geogra

Function Name	Category Name	Function Description
ST_ISPOLYGONCCW	Geospatial Spatial Relationships	Returns TRUE if an input LINESTRIN(
ST_ISPOLYGONCW	Geospatial Spatial Relationships	Returns TRUE if an input POLYGON h
ST_ISRING	Geospatial Spatial Relationships	Returns TRUE if the specified LINES1
ST_ISSIMPLE	Geospatial Spatial Relationships	For LINESTRING values, this functior For POLYGON values, the function re
ST_ISVALID	Geospatial Spatial Relationships	**Returns TRUE if the specified ge OGC standards.**
ST_LENGTH	Geospatial Spatial Measurement	Returns the length of the specified l
ST_LENGTH2D	Geospatial Spatial Measurement	Alias for ST_LENGTH.
ST_LINEFROMEWKT	Geospatial Linestring Constructor	Creates a LINESTRING from the spec
ST_LINEFROMGEOJSON	Geospatial Linestring Constructor	Creates a LINESTRING represented b
ST_LINEFROMTEXT	Geospatial Linestring Constructor	Creates a LINESTRING from a specifi
ST_LINEFROMWKB	Geospatial Linestring Constructor	Creates a LINESTRING from the spec
ST_LINEGETALLTIMESATPOINT	Geospatial Spatiotemporal Operators	Returns a timestamp array of all tim
ST_LINEGETPOINTATTIME	Geospatial Spatiotemporal Operators	Returns a POINT within the bounds c specified TIMESTAMP value.
ST_LINEGETTIMEATPOINT	Geospatial Spatiotemporal Operators	Returns the interpolated time of the ARRAY.
ST_LINEINTERPOLATEPOINT	Geospatial Linestring Function	Returns a POINT along a LINESTRIN(

Function Name	Category Name	Function Description
ST_LINELOCATEPOINT	Geospatial Linestring Function	Similar to ST_LINEINTERPOLATEPOIN along the length of a specified LINES
ST_LINESTRING	Geospatial Linestring Constructor	Creates a LINESTRING based on the
ST_LINESUBSTRING	Geospatial Linestring Function	Returns a LINESTRING that is a subs length.
ST_LONGESTLINE	Geospatial Spatial Operators	Returns the longest LINESTRING bet
ST_LONGESTLINE	Geospatial Spatiotemporal Operators	With the specified two LINESTRING-T represents the maximum distance b
ST_MAKEENVELOPE	Geospatial Spatial Operators	Returns a POLYGON with vertices tha
ST_MAKELINE	Geospatial Linestring Constructor	Alias for ST_LINESTRING.
ST_MAKEPOINT	Geospatial Point Constructors	Alias for ST_POINT.
ST_MAKEPOLYGON	Geospatial Polygon Constructors	Alias for ST_POLYGON.
ST_MAXDISTANCE	Geospatial Spatial Measurement	Returns maximum distance betweer
ST_MAXDISTANCE	Geospatial Spatiotemporal Measurement	Returns the two-dimensional interpo pairs in the specified unit of measur
ST_MEMSIZE	Geospatial Attribute Functions	Returns an INTEGER representing th
ST_MINIMUMBOUNDINGCIRCLE	Geospatial Spatial Operators	Returns the smallest circle POLYGON
ST_MULTIDIFFERENCEARRAY	Geospatial Spatial Operators	Returns an array of geographies tha not intersect with the union of geog
ST_MINIMUMDISTANCETOSURFACE	Geospatial Spatial Measurement	Calculates the shortest distance bet the surface of the Earth.
ST_MULTIINTERSECTIONARRAY	Geospatial Spatial Operators	When you specify two arrays of geos

Function Name	Category Name	Function Description
ST_MULTISYMDIFFERENCEARRAY	Geospatial Spatial Operators	When you specify two arrays of geos intersect.
ST_MULTIUNIONARRAY	Geospatial Spatial Operators	When you specify two arrays of geos geographies in both arrays.
ST_NDIMENSION	Geospatial Attribute Functions	Alias for ST_COORDDIM.
ST_NDIMS	Geospatial Attribute Functions	Alias for ST_COORDDIM.
ST_NPOINTS	Geospatial Attribute Functions	Returns an INTEGER representing th
ST_NRINGS	Geospatial Spatial Operators	Returns the number of rings of the s
ST_NUMINTERIORRING	Geospatial Spatial Operators	Returns the number of interior rings
ST_NUMINTERIORRINGS	Geospatial Spatial Operators	Returns the number of interior rings
ST_NUMPOINTS	Geospatial Attribute Functions	Returns an INTEGER representing th
ST_OVERLAPS	Geospatial Spatial Relationships	Returns TRUE if both geographic arg contains the other.
ST_PERIMETER	Geospatial Spatial Measurement	Returns the length of the exterior (o
ST_PERIMETER2D	Geospatial Spatial Measurement	Alias for ST_PERIMETER.
ST_POINT	Geospatial Point Constructors	Creates a POINT from the specified i
ST_POINTFROMEWKT	Geospatial Point Constructors	Alias for ST_POINT. Creates a POINT
ST_POINTFROMGEOHASH	Geospatial Point Constructors	Creates a POINT represented by the
ST_POINTFROMGEOJSON	Geospatial Point Constructors	Creates a POINT represented by the
ST_POINTFROMTEXT	Geospatial Point Constructors	Alias for ST_POINT(char).
ST_POINTFROMWKB	Geospatial Point Constructors	Alias for ST_POINT(binary).

Function Name	Category Name	Function Description
ST_POINTINSIDECIRCLE	Geospatial Spatial Relationships	Returns TRUE if the geographic obje specified radius.
ST_POINTN	Geospatial Linestring Function	Returns the POINT value at a specifi
ST_POINTONSURFACE	Geospatial Spatial Operators	Returns a POINT guaranteed to inter
ST_POLYGON	Geospatial Polygon Constructors	Creates a POLYGON.
ST_POLYGONFROMEWKT	Geospatial Polygon Constructors	Creates a POLYGON using an EWKT-f
ST_POLYGONFROMGEOJSON	Geospatial Polygon Constructors	Creates a POLYGON from the specifi
ST_POLYGONFROMTEXT	Geospatial Polygon Constructors	Alias for ST_POLYGON(char).
ST_POLYGONFROMWKB	Geospatial Polygon Constructors	Alias for ST_POLYGON(binary).
ST_PROJECT	Geospatial Spatial Operators	Returns a POINT by projecting a dist
ST_REDUCEPRECISION	Geospatial Spatial Operators	Returns a new geospatial object witl
ST_RELATE	Geospatial Spatial Relationships	Returns the **DE-9IM** intersection str
ST_REMOVEPOINT	Geospatial Linestring Functions	Removes a POINT value at a specifie
ST_REMOVEREPEATEDPOINTS	Geospatial Spatial Operators	Returns a new geospatial object witl
ST_REVERSE	Geospatial Spatial Operators	Returns a new geospatial object witl
ST_SEGMENTIZE	Geospatial Spatial Operators	Returns a geospatial object that the max_segment_length in meters.
ST_SETPOINT	Geospatial Linestring	Replaces a POINT value in a given L the replaced point.

Function Name	Category Name	Function Description
	Functions	
ST_SHORTESTLINE	Geospatial Spatial Operators	Returns the shortest LINESTRING be
ST_SHORTESTLINE	Geospatial Spatiotemporal Operators	When you specify two LINESTRING-T represents the minimum distance be
ST_SIMPLIFY	Geospatial Spatial Operators	Returns a simplified version of the s
ST_SIMPLIFYARRAY	Geospatial Spatial Operators	Returns a POLYGON array that repre LINESTRING, or POLYGON.
ST_SNAPTOGRID	Geospatial Spatial Operators	Returns a new geography value with
ST_SRID	Geospatial Attribute Functions	Returns the EPSG code of the spatia
ST_STARTPOINT	Geospatial Linestring Functions	Returns the starting POINT value of
ST_SYMDIFFERENCEARRAY	Geospatial Spatial Operators	Returns a geographic array that con geo2.
ST_TOTALSECONDSININTERSECTION	Geospatial Spatiotemporal Measurement	Returns the total number of seconds ST_INTERSECTION.
ST_TOUCHES	Geospatial Spatial Relationships	Returns TRUE if the only POINT valu boundaries.
ST_UNIONARRAY	Geospatial Spatial Operators	Performs a union of the input geogra
ST_WHOLEEARTH	Geospatial Polygon Constructors	Returns the database internal repres
ST_WITHIN	Geospatial Spatial Relationships	Alias for ST_CONTAINS.
ST_X	Geospatial Attribute Functions	Returns the x value of the specified
ST_XMAX	Geospatial Attribute Functions	Returns the maximum x value of the
ST_XMIN	Geospatial Attribute Functions	Returns the minimum x value of the

Function Name	Category Name	Function Description
ST_Y	Geospatial Attribute Functions	Returns the y value of the specified
ST_YMAX	Geospatial Attribute Functions	Returns the maximum y value of spe
ST_YMIN	Geospatial Attribute Functions	Returns the minimum y value of spe
STARTSWITH	Character and Binary Functions	Returns true if string starts with sub
STDEV	Aggregate Functions	Sample standard deviation.
STDDEV	Aggregate Functions	Alias for STDEV.
STDDEV_POP	Aggregate Functions	Alias for STDEVP.
STDDEV_SAMP	Aggregate Functions	Alias for STDEV.
STDEVP	Aggregate Functions	Population standard deviation.
STRING_AGG	Aggregate Functions	Returns a string concatenated from
STRING_TO_ARRAY	Array Functions	Converts the string representation o
STRING_TO_TUPLE	Tuple Functions	Converts the string representation o
STRPOS	Character and Binary Functions	Equivalent to using LOCATE as `LOCA`
SUBSTR	Character and Binary Functions	Alias for SUBSTRING.
SUBSTRING	Character and Binary Functions	Returns the substring of a character
SUBNET	Network Type Functions	Computes the prefix from an `IP` or
SUM	Aggregate Functions	Sum over the set.
SVD_DECOMP	Matrix Functions	Returns SVD decomposition of a ma
TAN	Math Functions	Returns the tangent of x.

Function Name	Category Name	Function Description
TANH	Math Functions	Returns the hyperbolic tangent of x.
TIME	Scalar Data Conversion Functions	Parses the string to create a time va
TIMESTAMP	Scalar Data Conversion Functions	Parses a string and makes a timesta
TIMESTAMP_TO_NANOS	Date and Time Functions	Convert timestamp into nanosecond
TO_ARRAY_LENGTH	Special Data Pipeline Transformation Functions	Create the specified number of copi
TO_BASE	Math Functions	Converts an integer value to its strir
TO_CHAR	Character and Binary Functions	Converts a numeric, date, or timesta
TO_DATE	Formatting Functions	Converts a character value with the
TO_NUMBER	Formatting Functions	Converts a character value with the
TO_TIMESTAMP	Formatting Functions	Converts a character value with the
TRANSFORM	Special Data Pipeline Transformation Functions	Transforms an array based on the lo
TRANSLATE	Character and Binary Functions	Replaces specified characters in a p
TRANSPOSE	Matrix Functions	Returns transpose of the matrix.
TRIM	Character and Binary Functions	Alias for BTRIM. Trim leading and tra
TRUNC	Math Functions	Returns x truncated to y decimal pla
TRUNCATE	Math Functions	Alias for TRUNC.
TUPLE()	Tuple Functions	Construct a tuple with the specified
TUPLE<<>>	Tuple Functions	Construct a tuple with the specified

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
TYPE[]	Array Functions	Construct an array of SQL type TYPE
TYPE_STRIP	Other Functions and Expressions	Returns the SQL type of the specifie
UCASE	Character and Binary Functions	Alias for UPPER.
UNNEST	Array Functions	Expand each element in an input ar
UPPER	Character and Binary Functions	Convert string to upper case.
USECS	Date and Time Functions	The seconds part of a time value, in seconds part of the value by 1,000,(
UUID	Scalar Data Conversion Functions	Parses the string and makes a Unive
UUID_GENERATE	Other Functions and Expressions	Generates a random UUID value (ve
VAR_POP	Aggregate Functions	Alias for VARIANCEP.
VAR_SAMP	Aggregate Functions	Alias for VARIANCE.
VARIANCE	Aggregate Functions	Sample variance.
VARIANCEP	Aggregate Functions	Population variance.
VECTOR_ARGMAX	Matrix Functions	Returns the argmax of a one-dimens
VECTOR_ARGMIN	Matrix Functions	Returns the argmin of a one-dimens
VECTOR_MAX	Matrix Functions	Returns the maximum of elements i
VECTOR_MIN	Matrix Functions	Returns the minimum of elements ir
VECTOR_SUM	Matrix Functions	Returns the sum of elements in a or
VERSION	System Functions	Returns the version of the database
WEEK	Date and Time Functions	Returns the ISO-8601 week number
WEEKS	Scalar Data Conversion Functions	Converts an integral value to an inte

Function Name	Category Name	Function Description
WIDTH_BUCKET	Special Data Pipeline Transformation Functions	Creates `N` equal-width buckets in t
YEAR	Date and Time Functions	Extracts the year portion of a timest
YEARS	Scalar Data Conversion Functions	Converts an integral value to an inte
ZERO_MATRIX	Matrix Functions	Returns a zero matrix of the specifie
ZIP_WITH	Special Data Pipeline Transformation Functions	Merges two or more input arrays, el
ZN	Conditional Functions	If x is NULL, returns 0. Otherwise, re
ZSCORE	Window Aggregate Functions	Zscore of the sample based on the
ZSCOREP	Window Aggregate Functions	Zscore of the sample based on the

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Categorized SQL Functions List

Function Name	Category Name	Function Description
ACCURACY_SCORE	Aggregate Functions	Returns the fraction of predictions th
ANY_VALUE	Aggregate Functions	Returns an arbitrary, non-NULL valu
APPROX_COUNT_DISTINCT	Aggregate Functions	Approximate distinct count by using
APPROX_SUM	Aggregate Functions	Allows the aggregation engine to us
ARRAY_AGG	Aggregate Functions	Returns an array containing every rc
ARRAY_CONCAT_AGG	Aggregate Functions	Returns an array that concatenates
AVG	Aggregate Functions	Average, or arithmetic mean, over t
COEFFICIENT_OF_DETERMINATION	Aggregate Functions	Computes the coefficient of determi
CONFUSION_MATRIX	Aggregate Functions	Returns a structured representation to a specified positive class.
CORR	Aggregate Functions	Alias for CORRELATION.
CORRELATION	Aggregate Functions	Sample correlation.
CORRELATIONP	Aggregate Functions	Population correlation.
COUNT	Aggregate Functions	Number of rows in the set.
COVAR_POP	Aggregate Functions	Alias for COVARIANCEP.
COVAR_SAMP	Aggregate Functions	Alias for COVARIANCE.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Function Name	Category Name	Function Description
COVARIANCE	Aggregate Functions	Sample covariance.
COVARIANCEP	Aggregate Functions	Population covariance.
F1_SCORE	Aggregate Functions	Returns the harmonic mean of preci
KURTOSIS	Aggregate Functions	The sample over the set.
KURTOSISP	Aggregate Functions	The population over the set.
MAX	Aggregate Functions	Maximum value in the specified colu
MEAN_ABSOLUTE_ERROR	Aggregate Functions	Returns the mean absolute error (M/
MEAN_ABSOLUTE_PERCENTAGE_ERROR	Aggregate Functions	Returns the mean absolute percenta
MIN	Aggregate Functions	Minimum value in the specified colu
PRECISION_SCORE	Aggregate Functions	Returns the precision score for a spe
PRODUCT	Aggregate Functions	Product over the set.
RECALL_SCORE	Aggregate Functions	Returns the recall score for a specifi
ROC_AUC_SCORE	Aggregate Functions	Returns the area under the receiver
SKEW	Aggregate Functions	The sample skewness over the set c
SKEWP	Aggregate Functions	Computes the population skewness
STDEV	Aggregate Functions	Sample standard deviation.
STDDEV	Aggregate Functions	Alias for STDEV.
STDDEV_POP	Aggregate Functions	Alias for STDEVP.

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
STDDEV_SAMP	Aggregate Functions	Alias for STDEV.
STDEVP	Aggregate Functions	Population standard deviation.
STRING_AGG	Aggregate Functions	Returns a string concatenated from
SUM	Aggregate Functions	Sum over the set.
VAR_POP	Aggregate Functions	Alias for VARIANCEP.
VAR_SAMP	Aggregate Functions	Alias for VARIANCE.
VARIANCE	Aggregate Functions	Sample variance.
VARIANCEP	Aggregate Functions	Population variance.
ARRAY[]	Array Functions	PostgreSQL®-compliant constructor
ARRAY_APPEND	Array Functions	Add value to the back of an array.
ARRAY_ARGMAX	Array Functions	Returns the corresponding argmax c
ARRAY_ARGMIN	Array Functions	Returns the corresponding argmin o
ARRAY_CAT	Array Functions	Concatenate 2 arrays into a new on
ARRAY_CAT_DISTINCT	Array Functions	Concatenates two or more arrays in
ARRAY_DISTINCT	Array Functions	Removes duplicates from an array w
ARRAY_LENGTH	Array Functions	Return the number of elements of a
ARRAY_MAX	Array Functions	Returns the corresponding maximur
ARRAY_MIN	Array Functions	Returns the corresponding minimum
ARRAY_POSITION	Array Functions	Returns the position of the first mat
ARRAY_POSITIONS	Array Functions	Returns all elements stored in the ri
ARRAY_PREPEND	Array Functions	Add value to the front of an array.
ARRAY_REMOVE	Array Functions	Remove value from the array.
ARRAY_REPLACE	Array Functions	Replace value by another in an arra

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
ARRAY_SUM	Array Functions	Returns the sum of the array. The ar contribute to the sum.
ARRAY_TO_STRING	Array Functions	Converts array to a list of elements
CAST_TO_ARRAY	Array Functions	Casts the elements of the array to a
CHAR	Array Functions	Converts array to its string represen
CROSS_ENTROPY_LOSS	Array Functions	Returns the cross entropy loss of tw
HINGE_LOSS	Array Functions	Returns the hinge loss of two arrays
LOG_LOSS	Array Functions	Returns the log loss of two arrays.
LOGITS_LOSS	Array Functions	Returns the logits loss of two arrays
SOFTMAX	Array Functions	Returns the softmax of the array.
STRING_TO_ARRAY	Array Functions	Converts the string representation c
TYPE[]	Array Functions	Construct an array of SQL type TYPE
UNNEST	Array Functions	Expand each element in an input ar
ASCII	Character and Binary Functions	Returns the ASCII code value of the
BIT_LENGTH	Character and Binary Functions	Returns the length of the character
BTRIM	Character and Binary Functions	Alias for TRIM.
CHAR_LENGTH	Character and Binary Functions	Alias for LENGTH.
CHARACTER_LENGTH	Character and Binary Functions	Alias for LENGTH.
CHR	Character and Binary Functions	Converts an integer value to a string
CONCAT	Character and Binary Functions	Concatenates two values, which mu the || operator.
ENDSWITH	Character and Binary Functions	Returns true if x ends with y and fal
INITCAP	Character and Binary Functions	For each word in the provided string

Function Name	Category Name	Function Description
INSTR	Character and Binary Functions	Returns the index position of the firs value char by ignoring the case.
JSON_EXTRACT_PATH_TEXT	Character and Binary Functions	Returns the value for the key-value
LCASE	Character and Binary Functions	Alias for LOWER.
LEFT	Character and Binary Functions	Return the number of characters in returns all but the last integer chara
LENGTH	Character and Binary Functions	For character data types, this value
LOCATE	Character and Binary Functions	Alias for POSITION. Returns the inde value string.
LOWER	Character and Binary Functions	Alias for LCASE. Convert string to lo
LPAD	Character and Binary Functions	Pad the input text to the specified le
LTRIM	Character and Binary Functions	Removes leading blanks from the st
MD5	Character and Binary Functions	Returns the hexadecimal string (all l
MID	Character and Binary Functions	Alias for SUBSTRING.
OCTET_LENGTH	Character and Binary Functions	Returns the length in bytes of a cha
POSITION	Character and Binary Functions	Alias for LOCATE.
REGEXP_COUNT	Character and Binary Functions	Searches a string for all occurrences
REGEXP_INSTR	Character and Binary Functions	Searches a string using a regular ex position of the substring that match
REGEXP_REPLACE	Character and Binary Functions	Searches a string for all occurrences
REGEXP_SUBSTR	Character and Binary Functions	Returns one substring from a string
REPEAT	Character and Binary Functions	Repeats the character value char a

Function Name	Category Name	Function Description
REPLACE	Character and Binary Functions	Replaces all occurrences of substr_t
REVERSE	Character and Binary Functions	Reverse the input string.
RIGHT	Character and Binary Functions	Return the number of trailing charac
RPAD	Character and Binary Functions	Pad the input text to the specified le
RSUBSTRING	Character and Binary Functions	Returns the substring from the right
RTRIM	Character and Binary Functions	Removes leading blanks from the st
SHA1	Character and Binary Functions	Uses the [SHA-1](**https://en.wikipe 1#:~:text=In%20cryptography%** cryptographic hash function to conv 160-bit checksum.
SPACE	Character and Binary Functions	Returns a string of repeated spaces
SPLIT_PART	Character and Binary Functions	Split the value string based on the d the index value (starting from 1).
SPLIT_TO_ARRAY	Character and Binary Functions	Splits a string into an array of substı
STARTSWITH	Character and Binary Functions	Returns true if string starts with sub
STRPOS	Character and Binary Functions	Equivalent to using LOCATE as LOCA
SUBSTR	Character and Binary Functions	Alias for SUBSTRING.
SUBSTRING	Character and Binary Functions	Returns the substring of a character
TO_CHAR	Character and Binary Functions	Converts a numeric, date, or timest
TRANSLATE	Character and Binary Functions	Replaces specified characters in a p
TRIM	Character and Binary Functions	Alias for BTRIM. Trim leading and tra

Function Name	Category Name	Function Description
UCASE	Character and Binary Functions	Alias for UPPER.
UPPER	Character and Binary Functions	Convert string to upper case.
CASE	Conditional Functions	CASE operates similarly to condition then / else statement or as a switch
COALESCE	Conditional Functions	Evaluates to the first argument that
GREATEST	Conditional Functions	Returns the largest non-NULL value
IF_NULL	Conditional Functions	Alias for COALESCE.
LEAST	Conditional Functions	Returns the smallest non-NULL valu
MURMUR3	Conditional Functions	**Returns a 32-bit MurmurHash3 l**
NULL_IF	Conditional Functions	Returns the NULL value if two argum
ZN	Conditional Functions	If x is NULL, returns 0. Otherwise, re
LAG_VECTORS	Data Preparation	Groups lagged columns generated b
LAGS	Data Preparation	Generates a series of lagged columr
LAGS_ZEROFILL	Data Preparation	Generates lagged columns for a sing
MULTI_LAGS	Data Preparation	Generates lagged columns for multi
MULTI_LAGS_ZEROFILL	Data Preparation	Generates lagged columns for multi
ADD_MONTHS	Date and Time Functions	Adds the specified number of month
CENTURY	Date and Time Functions	Returns the number of centuries.
CURDATE	Date and Time Functions	Alias for CURRENT_DATE.
CURRENT_DATE	Date and Time Functions	Returns the current date in the form

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
CURRENT_TIME	Date and Time Functions	Returns the current time as a TIME v
CURRENT_TIMESTAMP	Date and Time Functions	Returns the current date and time a
DATE_PART	Date and Time Functions	Alias for EXTRACT.
DATE_TRUNC	Date and Time Functions	Returns the date or timestamp ente
DATEADD	Date and Time Functions	Adds a specified number value (as a that modified value.
DATEDIFF	Date and Time Functions	This function returns an INT represe unit.
DAY	Date and Time Functions	Alias for DAY_OF_MONTH.
DAY_OF_MONTH	Date and Time Functions	Extracts the day-of-month portion of
DAY_OF_WEEK	Date and Time Functions	Returns an integer, in the range of 1
DAY_OF_YEAR	Date and Time Functions	Returns an integer in the range 1 to
DECADE	Date and Time Functions	The decade is the year divided by 1(
DOW	Date and Time Functions	Alias for DAY_OF_WEEK.
DOY	Date and Time Functions	Alias for DAY_OF_YEAR.
EOMONTH	Date and Time Functions	Returns the last day of the timestam
EPOCH	Date and Time Functions	The number of seconds after 1970-0
EXTRACT	Date and Time Functions	Extract a component from a timesta
HOUR	Date and Time Functions	Extracts the hour portion of a timest
ISDATE	Date and Time Functions	Returns TRUE if the input argument

Function Name	Category Name	Function Description
ISODOW	Date and Time Functions	Extracts the day of the week based
MAKEDATETIME	Date and Time Functions	Returns a timestamp consisting of tl
MILLISECOND	Date and Time Functions	Extracts the millisecond portion of a
MINUTE	Date and Time Functions	Extracts the minute portion of a tim
MONTH	Date and Time Functions	Extracts the month portion of a time
MONTH_NAME	Date and Time Functions	Returns the calendar name in Englis
MONTHS_BETWEEN	Date and Time Functions	Returns the difference between the
MSECS	Date and Time Functions	The seconds field, including fraction
NANOS_TO_TIMESTAMP	Date and Time Functions	Convert a number of nanoseconds i
NEXT_DAY	Date and Time Functions	Returns the closest date after a spe
NOW	Date and Time Functions	Alias for CURRENT_TIMESTAMP.
QUARTER	Date and Time Functions	Returns an integer between 1 and 4
ROUND	Date and Time Functions	Returns the specified date or timest
SECOND	Date and Time Functions	Extracts the seconds portion of a tin
TIMESTAMP_TO_NANOS	Date and Time Functions	Convert timestamp into nanosecond
USECS	Date and Time Functions	The seconds part of a time value, in seconds part of the value by 1,000,(
WEEK	Date and Time Functions	Returns the ISO-8601 week number,
YEAR	Date and Time Functions	Extracts the year portion of a timest

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
TO_DATE	Formatting Functions	Converts a character value with the
TO_NUMBER	Formatting Functions	Converts a character value with the
TO_TIMESTAMP	Formatting Functions	Converts a character value with the
ST_COORDDIM	Geospatial Attribute Functions	Alias for ST_NDIMS or ST_NDIMENSI(
ST_DIMENSION	Geospatial Attribute Functions	Returns an INTEGER that represents
ST_GEOMETRYTYPE	Geospatial Attribute Functions	Returns a string representing the ge
ST_ISEMPTY	Geospatial Attribute Functions	Returns TRUE if the specified geogra
ST_MEMSIZE	Geospatial Attribute Functions	Returns an INTEGER representing th
ST_NDIMENSION	Geospatial Attribute Functions	Alias for ST_COORDDIM.
ST_NDIMS	Geospatial Attribute Functions	Alias for ST_COORDDIM.
ST_NPOINTS	Geospatial Attribute Functions	Returns an INTEGER representing th
ST_NUMPOINTS	Geospatial Attribute Functions	Returns an INTEGER representing th
ST_SRID	Geospatial Attribute Functions	Returns the EPSG code of the spatia
ST_X	Geospatial Attribute Functions	Returns the x value of the specified
ST_XMAX	Geospatial Attribute Functions	Returns the maximum x value of the
ST_XMIN	Geospatial Attribute Functions	Returns the minimum x value of the
ST_Y	Geospatial Attribute Functions	Returns the y value of the specified
ST_YMAX	Geospatial Attribute Functions	Returns the maximum y value of spe

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
ST_YMIN	Geospatial Attribute Functions	Returns the minimum y value of spe
ST_ASBINARY	Geospatial Conversion Functions	Returns the well-known binary (WKE
ST_ASEWKT	Geospatial Conversion Functions	Returns a string that represents geo
ST_ASGEOJSON	Geospatial Conversion Functions	Alias of ST_ASBINARY.
ST_ASLATLONTEXT	Geospatial Conversion Functions	**Returns the GeoJSON represent**
ST_ASTEXT	Geospatial Conversion Functions	Alias of ST_ASTEXT.
ST_ASWKB	Geospatial Conversion Functions	Alias of ST_ASTEXT.
ST_ASWKT	Geospatial Conversion Functions	Alias of ST_ASWKT and ST_EWKT. Re
ST_GEOHASH	Geospatial Conversion Functions	Returns a string that represents the
ST_ADDPOINT	Geospatial Linestring Function	Adds a POINT to the specified LINES
ST_ENDPOINT	Geospatial Linestring Function	Returns the endpoint of a specified l
ST_LINEFROMEWKT	Geospatial Linestring Constructor	Creates a LINESTRING from the spec
ST_LINEFROMGEOJSON	Geospatial Linestring Constructor	Creates a LINESTRING represented b
ST_LINEFROMTEXT	Geospatial Linestring Constructor	Creates a LINESTRING from a specifi

Function Name	Category Name	Function Description
ST_LINEFROMWKB	Geospatial Linestring Constructor	Creates a LINESTRING from the spec
ST_LINEINTERPOLATEPOINT	Geospatial Linestring Function	Returns a POINT along a LINESTRING
ST_LINELOCATEPOINT	Geospatial Linestring Function	Similar to ST_LINEINTERPOLATEPOIN along the length of a specified LINES
ST_LINESTRING	Geospatial Linestring Constructor	Creates a LINESTRING based on the
ST_LINESUBSTRING	Geospatial Linestring Function	Returns a LINESTRING that is a subs length.
ST_MAKELINE	Geospatial Linestring Constructor	Alias for ST_LINESTRING.
ST_POINTN	Geospatial Linestring Function	Returns the POINT value at a specifi
ST_REMOVEPOINT	Geospatial Linestring Function	Removes a POINT value at a specifie
ST_SETPOINT	Geospatial Linestring Function	Replaces a POINT value in a specifie with the replaced point.
ST_STARTPOINT	Geospatial Linestring Function	Returns the starting POINT value of
ST_CENTROID	Geospatial Point Constructors	The geographic center of mass is ca projecting the resultant point onto th
ST_GEOGPOINT	Geospatial Point Constructors	Creates and returns a POLYGON geo function.
ST_MAKEPOINT	Geospatial Point Constructors	Alias for ST_POINT.
ST_POINT	Geospatial Point Constructors	Creates a POINT from the specified i
ST_POINTFROMEWKT	Geospatial Point Constructors	Alias for ST_POINT. Creates a POINT
ST_POINTFROMGEOHASH	Geospatial Point Constructors	Creates a POINT represented by the
ST_POINTFROMGEOJSON	Geospatial Point Constructors	Creates a POINT represented by the

Function Name	Category Name	Function Description
ST_POINTFROMTEXT	Geospatial Point Constructors	Alias for ST_POINT(char).
ST_POINTFROMWKB	Geospatial Point Constructors	Alias for ST_POINT(binary).
ST_FORCECCW	Geospatial Polygon Constructors	Creates a standardized polygon fron counterclockwise (CCW) and all hole
ST_MAKEPOLYGON	Geospatial Polygon Constructors	Alias for ST_POLYGON.
ST_POLYGON	Geospatial Polygon Constructors	Creates a POLYGON.
ST_POLYGONFROMEWKT	Geospatial Polygon Constructors	Creates a POLYGON using an EWKT-f
ST_POLYGONFROMGEOJSON	Geospatial Polygon Constructors	Creates a POLYGON from the specifi
ST_POLYGONFROMTEXT	Geospatial Polygon Constructors	Alias for ST_POLYGON(char).
ST_POLYGONFROMWKB	Geospatial Polygon Constructors	Alias for ST_POLYGON(binary).
ST_WHOLEEARTH	Geospatial Polygon Constructors	Returns the database internal repres
ST_ANGLE	Geospatial Spatial Measurement	Calculates the angle between two li
ST_AREA	Geospatial Spatial Measurement	Returns the area of the specified ge
ST_AZIMUTH	Geospatial Spatial Measurement	Returns the azimuth of the line from
ST_DISTANCE	Geospatial Spatial Measurement	Returns the minimum distance betw
ST_DISTANCESPHERE	Geospatial Spatial Measurement	Returns the minimum distance betw

Function Name	Category Name	Function Description
ST_DISTANCESPHEROID	Geospatial Spatial Measurement	Returns the minimum distance betw
ST_EUCLIDEANDISTANCE3D	Geospatial Spatial Measurement	Returns the minimum distance betw
ST_HAUSDORFFDISTANCE	Geospatial Spatial Measurement	Returns the Hausdorff distance betw
ST_LENGTH	Geospatial Spatial Measurement	Returns the length of the specified l
ST_LENGTH2D	Geospatial Spatial Measurement	Alias for ST_LENGTH.
ST_MAXDISTANCE	Geospatial Spatial Measurement	Returns maximum distance betweer
ST_MINIMUMDISTANCETOSURFACE	Geospatial Spatial Measurement	Calculates the shortest distance bet the surface of the Earth.
ST_PERIMETER	Geospatial Spatial Measurement	Returns the length of the exterior (o
ST_PERIMETER2D	Geospatial Spatial Measurement	Alias for ST_PERIMETER.
ST_BOUNDINGDIAGONAL	Geospatial Spatial Operators	Returns the diagonal LINESTRING fr ST_ENVELOPE returns.
ST_BUFFER	Geospatial Spatial Operators	Returns a geography that contains a specified distance.
ST_CLOSESTPOINT	Geospatial Spatial Operators	Returns the two-dimensional POINT object.
ST_CONVEXHULL	Geospatial Spatial Operators	The convex hull is the smallest conv
ST_DIFFERENCEARRAY	Geospatial Spatial Operators	Returns an array containing any geo not found in the second geospatial a
ST_ENVELOPE	Geospatial Spatial Operators	Returns a POLYGON that represents
ST_EXPAND	Geospatial Spatial Operators	Returns the bounding box of a speci
ST_EXTERIORRING	Geospatial Spatial Operators	Returns a LINESTRING that represen
ST_FLIPCOORDINATES	Geospatial Spatial Operators	Returns a new geographic object wi

Function Name	Category Name	Function Description
ST_FORCE2D	Geospatial Spatial Operators	Convert a geographic object into a t
ST_INTERIORRINGN	Geospatial Spatial Operators	Returns a LINESTRING representing indexed)
ST_INTERSECTALL	Geospatial Spatial Operators	Returns the intersection of all geogr
ST_INTERSECTIONARRAY	Geospatial Spatial Operators	Returns a geography that represent
ST_LONGESTLINE	Geospatial Spatial Operators	Returns the longest LINESTRING bet
ST_MAKEENVELOPE	Geospatial Spatial Operators	Returns a POLYGON with vertices tha
ST_MINIMUMBOUNDINGCIRCLE	Geospatial Spatial Operators	Returns the smallest circle POLYGON
ST_MULTIDIFFERENCEARRAY	Geospatial Spatial Operators	Returns an array of geographies tha not intersect with the union of geog
ST_MULTIINTERSECTIONARRAY	Geospatial Spatial Operators	When you specify two arrays of geos
ST_MULTISYMDIFFERENCEARRAY	Geospatial Spatial Operators	When you specify two arrays of geos intersect.
ST_MULTIUNIONARRAY	Geospatial Spatial Operators	When you specify two arrays of geos geographies in both arrays.
ST_NRINGS	Geospatial Spatial Operators	Returns the number of rings of the s
ST_NUMINTERIORRING	Geospatial Spatial Operators	Returns the number of interior rings
ST_NUMINTERIORRINGS	Geospatial Spatial Operators	Returns the number of interior rings
ST_POINTONSURFACE	Geospatial Spatial Operators	Returns a POINT guaranteed to inter
ST_PROJECT	Geospatial Spatial Operators	Returns a POINT by projecting a dist
ST_REDUCEPRECISION	Geospatial Spatial Operators	Returns a new geospatial object witl
ST_REMOVEREPEATEDPOINTS	Geospatial Spatial Operators	Returns a new geospatial object witl

Function Name	Category Name	Function Description
ST_REVERSE	Geospatial Spatial Operators	Returns a new geospatial object witl
ST_SEGMENTIZE	Geospatial Spatial Operators	Returns a geospatial object that the max_segment_length in meters.
ST_SHORTESTLINE	Geospatial Spatial Operators	Returns the shortest LINESTRING be
ST_SIMPLIFY	Geospatial Spatial Operators	Returns a simplified version of the s
ST_SIMPLIFYARRAY	Geospatial Spatial Operators	Returns a POLYGON array that repre LINESTRING, or POLYGON.
ST_SNAPTOGRID	Geospatial Spatial Operators	Returns a new geography value with
ST_SYMDIFFERENCEARRAY	Geospatial Spatial Operators	Returns a geographic array that con geo2.
ST_UNIONARRAY	Geospatial Spatial Operators	Performs a union of the input geogr
ST_CLUSTERDBSCAN	Geospatial Spatial Relationships	**Returns the cluster number for the density-based spatial cluste**
ST_CONTAINS	Geospatial Spatial Relationships	Returns TRUE if the first geographic
ST_CONTAINSPROPERLY	Geospatial Spatial Relationships	Returns true if geo2 lies entirely in t points.
ST_COVEREDBY	Geospatial Spatial Relationships	Returns TRUE if no POINT in geo1 is
ST_COVERS	Geospatial Spatial Relationships	Returns TRUE if no POINT in geo2 is
ST_CROSSES	Geospatial Spatial Relationships	Returns TRUE if two geospatial obje The intersection is not equal to geo1
ST_DISJOINT	Geospatial Spatial Relationships	Returns true if the specified geograp be different types.
ST_DWITHIN	Geospatial Spatial Relationships	Returns TRUE if the geographies are
ST_EQUALS	Geospatial Spatial Relationships	Returns TRUE if both geographies ar
ST_INTERSECTS	Geospatial Spatial Relationships	Returns TRUE if the specified geogr

Function Name	Category Name	Function Description
ST_ISCCW	Geospatial Spatial Relationships	Alias for ST_ISPOLYGONCCW.
ST_ISCLOSED	Geospatial Spatial Relationships	Returns TRUE if an input POLYGON h
ST_ISPOLYGONCCW	Geospatial Spatial Relationships	Returns TRUE if an input LINESTRIN(
ST_ISPOLYGONCW	Geospatial Spatial Relationships	Returns TRUE if an input POLYGON h
ST_ISRING	Geospatial Spatial Relationships	Returns TRUE if the specified LINES1
ST_ISSIMPLE	Geospatial Spatial Relationships	For LINESTRING values, this functior For POLYGON values, the function re
ST_ISVALID	Geospatial Spatial Relationships	**Returns TRUE if the specified ge OGC standards.**
ST_OVERLAPS	Geospatial Spatial Relationships	Returns TRUE if both geographic arg contains the other.
ST_POINTINSIDECIRCLE	Geospatial Spatial Relationships	Returns TRUE if the geographic obje specified radius.
ST_RELATE	Geospatial Spatial Relationships	Returns the **DE-9IM** intersection str
ST_TOUCHES	Geospatial Spatial Relationships	Returns TRUE if the only POINT valu boundaries.
ST_WITHIN	Geospatial Spatial Relationships	Alias for ST_CONTAINS.
ST_DISTANCE	Geospatial Spatiotemporal Measurement	Returns the two-dimensional interpc array pairs in the specified unit of m
ST_MAXDISTANCE	Geospatial Spatiotemporal Measurement	Returns the two-dimensional interpc pairs in the specified unit of measur
ST_TOTALSECONDSININTERSECTION	Geospatial Spatiotemporal Measurement	Returns the total number of seconds ST_INTERSECTION.
ST_INTERSECTION	Geospatial Spatiotemporal Operators	Returns a tuple that represents the i

Function Name	Category Name	Function Description
ST_LINEGETALLTIMESATPOINT	Geospatial Spatiotemporal Operators	Returns a timestamp array of all tim
ST_LINEGETPOINTATTIME	Geospatial Spatiotemporal Operators	Returns a POINT within the bounds c specified TIMESTAMP value.
ST_LINEGETTIMEATPOINT	Geospatial Spatiotemporal Operators	Returns the interpolated time of the ARRAY.
ST_LONGESTLINE	Geospatial Spatiotemporal Operators	With the specified two LINESTRING-T represents the maximum distance b
ST_SHORTESTLINE	Geospatial Spatiotemporal Operators	When you specify two LINESTRING-T represents the minimum distance be
HLL_SKETCH_CREATE	HyperLogLog Functions	Creates an HLL sketch from the data representation of the sketch that yo
HLL_SKETCH_GET_ESTIMATE	HyperLogLog Functions	The scalar function converts a sketc estimate as a BIGINT.
HLL_SKETCH_GET_ESTIMATE_BOUND	HyperLogLog Functions	Takes a HLL_SKETCH column or an in confidence interval error proportion
HLL_SKETCH_TO_STRING	HyperLogLog Functions	The HLL_SKETCH_TO_STRING scalar the sketch.
HLL_SKETCH_UNION (aggregate function)	HyperLogLog Functions	Merges multiple sketches in a single function is an aggregate function an
HLL_SKETCH_UNION (scalar function)	HyperLogLog Functions	Merges two sketches into a new con function merges two sketch columns
ABS	Math Functions	Returns the absolute value of a spec
ACOS	Math Functions	Returns the inverse cosine of a spec
ACOSH	Math Functions	Returns the hyperbolic arc-cosine of
ASIN	Math Functions	Returns the inverse sine of a specifi
ASINH	Math Functions	Returns the hyperbolic arc-sine of a
ATAN	Math Functions	Returns the inverse tangent of a spe
ATAN2	Math Functions	Returns the inverse tangent of two r
ATANH	Math Functions	Returns the hyperbolic arc-tangent c

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
BICDF	Math Functions	The cumulative distribution function
BIPDF	Math Functions	The probability density function of tl
BITAND	Math Functions	Alias for the BITFUNC syntax `BITFUI`
BITFUNC	Math Functions	Performs a variety of bit operations.
BITNOT	Math Functions	Returns the bitwise negation of inte
BITOR	Math Functions	Alias for the BITFUNC syntax BITFUN
BITXOR	Math Functions	Alias for the BITFUNC syntax BITFUN
BOOLAND	Math Functions	Alias for the BOOLFUNC syntax BOO
BOOLFUNC	Math Functions	Performs a Boolean logical evaluatic `'XOR'`.
BOOLNOT	Math Functions	Returns the logical negation of the E
BOOLOR	Math Functions	Alias for the BOOLFUNC syntax BOO
BOOLXOR	Math Functions	Alias for the BOOLFUNC syntax BOO
CBRT	Math Functions	Returns the cube root of the numeri
CDF	Math Functions	The cumulative distribution function sample is less than or equal to the s
CEIL	Math Functions	Returns the nearest integer greater
CEILING	Math Functions	Alias for CEIL.
COS	Math Functions	Returns the cosine of x.
COSH	Math Functions	Returns the hyperbolic cosine of x.
COT	Math Functions	Returns the cotangent of x.
DEGREES	Math Functions	Returns the corresponding angle in
DIV	Math Functions	Returns the result of x divided by y.
ERF	Math Functions	The error function is used for measu
ERFC	Math Functions	The complement of the error functic
EXP	Math Functions	Returns the exponential of x (e raise
FLOOR	Math Functions	Returns the nearest integer less tha
GAMMA	Math Functions	**Gamma function.**

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
HEXBINX	Math Functions	Returns the x-coordinate of the cent
HEXBINY	Math Functions	Returns the y-coordinate of the cent
IERF	Math Functions	The inverse of the ERF error functio
IERFC	Math Functions	The inverse of the complement of th
LEAKYRELU	Math Functions	Returns the leaky rectified linear uni
LEFT_SHIFT	Math Functions	Returns x shifted to the left by y bits
LN	Math Functions	Returns the natural logarithm of x.
LOG	Math Functions	Returns the base 10 logarithm of x. the function defaults to base 10.
LOG_GAMMA	Math Functions	The natural logarithm of the absolut
LOG2	Math Functions	Returns the base 2 logarithm of x.
MOD	Math Functions	Returns the remainder from x divide
PI	Math Functions	Returns the constant value of π.
PMOD	Math Functions	Returns the smallest non-negative e
POWER	Math Functions	Returns x raised to the power of y.
PROBIT	Math Functions	The inverse of the cumulative distrib
RADIANS	Math Functions	Returns the corresponding angle in r
RAND	Math Functions	Takes no argument and returns a ra
RELU	Math Functions	Returns the rectified linear unit func
RIGHT_SHIFT	Math Functions	Returns x shifted to the right by y bi
ROUND	Math Functions	Returns x rounded to the nearest int
SIGN	Math Functions	Returns the positive (+1), zero (0), o
SIN	Math Functions	Returns the sine of x.
SINH	Math Functions	Returns the hyperbolic sine of x.
SQRT	Math Functions	Returns the square root of x.
SQUARE	Math Functions	Returns the square of x.
TAN	Math Functions	Returns the tangent of x.
TANH	Math Functions	Returns the hyperbolic tangent of x.

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
TO_BASE	Math Functions	Converts an integer value to its strir
TRUNC	Math Functions	Returns x truncated to y decimal pla
TRUNCATE	Math Functions	Alias for TRUNC.
ABS	Matrix Functions	Returns magnitude of one-dimensio
CROSS_ENTROPY_LOSS	Matrix Functions	Returns the cross entropy loss of tw
DET	Matrix Functions	Returns the determinant of the matr
DOT	Matrix Functions	Returns dot product of two one-dime
EIGEN	Matrix Functions	Returns eigenvalues and eigenvalue
FROBENIUS	Matrix Functions	Returns the Frobenius norm of a ma
HINGE_LOSS	Matrix Functions	Returns the hinge loss of two one-di
IDENTITY_MATRIX	Matrix Functions	Returns an identity matrix of the spe
INVERSE	Matrix Functions	Returns inverse of a square, invertib
LOG_LOSS	Matrix Functions	Returns the log loss of two one-dime
LOGITS_LOSS	Matrix Functions	Returns the logits loss of two one-di
LUPQ_DECOMP	Matrix Functions	Returns LUPQ decomposition of a sq
MATRIX_DIM	Matrix Functions	Returns the dimensions of the speci
MAKE_MATRIX_IXJ	Matrix Functions	Creates an `ixj` matrix with elemer
MATRIX_FROM_TEXT	Matrix Functions	Creates a matrix from the specified
MATRIX_TRACE	Matrix Functions	Returns trace of a square matrix as
NULL_MATRIX	Matrix Functions	Returns a NULL matrix of the specifi
QR_DECOMP	Matrix Functions	Returns QR decomposition of a matr
SOFTMAX	Matrix Functions	Returns the softmax of a one-dimen
SVD_DECOMP	Matrix Functions	Returns SVD decomposition of a ma
TRANSPOSE	Matrix Functions	Returns transpose of the matrix.
VECTOR_ARGMAX	Matrix Functions	Returns the argmax of a one-dimens
VECTOR_ARGMIN	Matrix Functions	Returns the argmin of a one-dimens
VECTOR_MAX	Matrix Functions	Returns the maximum of elements i

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
VECTOR_MIN	Matrix Functions	Returns the minimum of elements ir
VECTOR_SUM	Matrix Functions	Returns the sum of elements in a or
ZERO_MATRIX	Matrix Functions	Returns a zero matrix of the given (r
IP	Network Type Functions	Casts an IP data type from an IPV4 c
IPV4	Network Type Functions	Casts an IPV4 address from an IPV6
IS_IPV4	Network Type Functions	Tests whether the database can con
SUBNET	Network Type Functions	Computes the prefix from an `IP` or
CANCEL	Query Management	Cancels a running query based on it
KILL	Query Management	Kills a running query identified by its
BIGINT	Scalar Data Conversion Functions	Casts the argument to a value of typ
BINARY	Scalar Data Conversion Functions	Parses a hexadecimal string (such a
BOOLEAN	Scalar Data Conversion Functions	Parses a string to create a Boolean v
BYTE	Scalar Data Conversion Functions	Casts the argument to a value of typ
CHAR	Scalar Data Conversion Functions	Creates a string version of the nume
DATE	Scalar Data Conversion Functions	Parses a string in the form 'YYYY-MM
DAYS	Scalar Data Conversion Functions	Converts an integral value to an inte

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
DECIMAL	Scalar Data Conversion Functions	Casts the argument to a value of typ
DOUBLE	Scalar Data Conversion Functions	Casts the argument to a value of typ
FLOAT	Scalar Data Conversion Functions	Casts the argument to a value of typ
HASH	Scalar Data Conversion Functions	Creates a fixed length binary value, string does not have enough bytes,
HOURS	Scalar Data Conversion Functions	Converts an integral value to an inte
INTEGER	Scalar Data Conversion Functions	Casts the argument to a value of typ
MICROSECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
MILLISECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
MINUTES	Scalar Data Conversion Functions	Converts an integral value to an inte
MONTHS	Scalar Data Conversion Functions	Converts an integral value to an inte
NANOSECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
SECONDS	Scalar Data Conversion Functions	Converts an integral value to an inte
SMALLINT	Scalar Data Conversion Functions	Casts the argument to a value of typ

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
TIME	Scalar Data Conversion Functions	Parses the string to create a time va
TIMESTAMP	Scalar Data Conversion Functions	Parses a string and makes a timesta
UUID	Scalar Data Conversion Functions	Parses the string and makes a UUID
WEEKS	Scalar Data Conversion Functions	Converts an integral value to an inte
YEARS	Scalar Data Conversion Functions	Converts an integral value to an inte
ARRAY_CAP	Special Data Pipeline Transformation Functions	Restrict the length of an array to a r
ARRAY_COMPACT	Special Data Pipeline Transformation Functions	Removes NULL values from the arra
ARRAY_CONTAINS	Special Data Pipeline Transformation Functions	Returns `true` if the array contains
ARRAY_SORT	Special Data Pipeline Transformation Functions	Sort and return the input array base
CASE WHEN	Special Data Pipeline Transformation Functions	Returns the `result` value based or
ELEMENT_AT	Special Data Pipeline Transformation Functions	Returns the element value of the arı
EXPLODE_OUTER	Special Data Pipeline	Expands a one-dimensional or multi system.

Function Name	Category Name	Function Description
	Transformation Functions	
FILTER	Special Data Pipeline Transformation Functions	Filters elements in an array based o
FLATTEN	Special Data Pipeline Transformation Functions	Transforms an `N`-dimensional array
IF	Special Data Pipeline Transformation Functions	Returns `T` if the expression `X` ev
LOOKUP	Special Data Pipeline Transformation Functions	Look up and load data in an externa
MAP_KEYS	Special Data Pipeline Transformation Functions	Returns the keys in the specified JSC
MAP_VALUES	Special Data Pipeline Transformation Functions	Returns the values in the specified J
METADATA	Special Data Pipeline Transformation Functions	Extracts the metadata value for the
PARSE_DELIMITED_ARRAY	Special Data Pipeline Transformation Functions	Converts a string of text data repres
REDUCE	Special Data Pipeline Transformation Functions	Applies a merge function to a startir value. Optionally, specify a finish fu
TO_ARRAY_LENGTH	Special Data Pipeline Transformation Functions	Create the specified number of copi

© 2026 Ocient, Inc.

Function Name	Category Name	Function Description
TRANSFORM	Special Data Pipeline Transformation Functions	Transforms an array based on the lo
WIDTH_BUCKET	Special Data Pipeline Transformation Functions	Creates `N` equal-width buckets in t
ZIP_WITH	Special Data Pipeline Transformation Functions	Merges two or more input arrays, el
RECORD_UUID	Other Transformation Functions	Returns string that represents a uni topic.
COMMIT	Other Functions and Expressions	Returns the most recent commit has
RAND_UUID	Other Functions and Expressions	Generates a random UUID value (ve
TYPE_STRIP	Other Functions and Expressions	Returns the SQL type of the specifie
UUID_GENERATE	Other Functions and Expressions	Generates a random UUID value (ve
CURRENT_DATABASE	System Functions	Alias for DATABASE.
CURRENT_GROUPS	System Functions	Returns the fully-qualified names of
CURRENT_NODE	System Functions	Returns the name of the SQL Node v column in the `sys.nodes` system ca
CURRENT_NODE_ID	System Functions	Returns the identifier of the SQL Noc
CURRENT_SCHEMA	System Functions	Returns the name of the current sch
CURRENT_SESSION_ID	System Functions	Returns the Universally Unique IDen
CURRENT_SYSTEM	System Functions	Returns the name of the system.
CURRENT_USER	System Functions	Returns the user for the current con
DATABASE	System Functions	Returns the name of the database t
SHOW	System Functions	The SHOW function enables you to e
VERSION	System Functions	Returns the version of the database

Function Name	Category Name	Function Description
CONVERT_LOCAL_TIMESTAMP_TO_UTC	Time Zone Functions	The function converts a timestamp i
CONVERT_UTC_TIMESTAMP_TO_LOCAL	Time Zone Functions	The function converts a timestamp f
CAST_TO_TUPLE	Tuple Functions	Converts a tuple into another tuple
CHAR	Tuple Functions	Converts a tuple to its string represe
STRING_TO_TUPLE	Tuple Functions	Converts the string representation c
TUPLE()	Tuple Functions	Construct a tuple of specified eleme
TUPLE<<>>	Tuple Functions	Construct a tuple of specified eleme
CUME_DIST	Window Aggregate Functions	Returns a number 0 < n 1 and can b value in the group.
DELTA	Window Aggregate Functions	Computes the finite difference betw backwards difference, which means of expression for that row and the p
DENSE_RANK	Window Aggregate Functions	Assigns a number to each row in the between ranks.
DERIVATIVE	Window Aggregate Functions	Computes the difference quotient be
FIRST_VALUE	Window Aggregate Functions	Returns the first value in the ordere
LAG	Window Aggregate Functions	Returns the row, which is the specifi
LAST_VALUE	Window Aggregate Functions	Returns the last value in the orderec
LEAD	Window Aggregate Functions	Returns the row, which is the specifi
NTH_VALUE	Window Aggregate Functions	Returns the nth value in the orderec
PERCENT_RANK	Window Aggregate Functions	The value returned is 0 < n ≤ 1 and excluding the highest value.
PERCENTILE	Window Aggregate Functions	Returns the value that corresponds
RANK	Window Aggregate Functions	Assigns a number to each row in the between ranks.

Function Name	Category Name	Function Description
RATIO_TO_REPORT	Window Aggregate Functions	Computes the ratio of a value to the
ROW_NUMBER	Window Aggregate Functions	Assigns a unique number to each ro
ZSCORE	Window Aggregate Functions	Zscore of the sample based on the
ZSCOREP	Window Aggregate Functions	Zscore of the sample based on the

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Aggregate Functions

The OcientAIQ™ Unified Data Platform has aggregated and sorted aggregate functions. Aggregate and sorted aggregate functions compute a single result from a group of rows. You can use the `DISTINCT` keyword with any aggregation function, such as `COUNT(DISTINCT col)`. Whereas sorted aggregate functions use the standard ORDER BY syntax to dictate the ordering of the aggregation. To sort the elements of an array, use the **ARRAY_SORT** function during data load.

Aggregate Functions

Supported input types vary by function. In general, numeric types include `TINYINT`, `SMALLINT`, `INT`, `BIGINT`, `FLOAT`, `DOUBLE`, and `DECIMAL`.

Various examples on this page use these tables.

SQL

```
CREATE TABLE sample_data (val INT, label VARCHAR(10));
INSERT INTO sample_data (val, label) VALUES
    (2, 'alpha'), (4, 'beta'), (4, 'alpha'), (5, NULL),
    (7, 'gamma'), (8, 'alpha'), (12, 'beta'), (15, 'gamma');
```

SQL

```
CREATE TABLE predictions (actual INT, predicted INT);
INSERT INTO predictions (actual, predicted) VALUES
    (10, 12), (20, 18), (30, 33),
    (40, 37), (50, 52), (60, 58);
```

For simple functions, examples use the `sys.dummy` virtual table. For details, see **Generate Tables Using sys.dummy**.

ACCURACY_SCORE

Returns the fraction of predictions that match the actual class labels. This function calculates the ratio of correctly predicted rows to the total number of rows.

```
SQL

ACCURACY_SCORE(y, y_hat)
```

Argument	Data Type	Description
`y`	All types supported	The actual class label.
`y_hat`	All types supported	The predicted class label.

The return type is `DOUBLE`. The result represents the proportion of correct predictions, ranging from 0 to 1.

Examples

Calculate the Accuracy Score for Matched Values

This query returns the accuracy score when all predictions match the actual values.

```
SQL

SELECT ACCURACY_SCORE(c1, c1) FROM sys.dummy10;
```

Output: `1`

Calculate the Accuracy Score with Comparison Against Squared Values

This query returns the accuracy score when comparing actual values against their squared values.

```
SQL

SELECT ACCURACY_SCORE(c1, c1 * c1) FROM sys.dummy10;
```

Output: `0.1`

ANY_VALUE

Returns an arbitrary non-`NULL` value from the input column. The function returns `NULL` only if all rows in the column are `NULL`.

The `ANY_VALUE` function does not support window aggregation.

Syntax

```
SQL

ANY_VALUE(col)
```

Argument	Data Type	Description
`col`	Any type	The column from which to return an arbitrary non-`NULL` value.

Example

```
SQL

SELECT
    ANY_VALUE(label) AS sample_label
FROM sample_data;
```

Output: `alpha`

> The returned value is non-deterministic. The database can return any non-`NULL` value from the column.

APPROX_COUNT_DISTINCT

Returns an approximate count of distinct values in the column using the HyperLogLog algorithm, with a 95% confidence interval that the result is within 4.5% of the exact count.

Syntax

```sql
APPROX_COUNT_DISTINCT(col)
```

Argument	Data Type	Description
`col`	Any non-interval type	The column for the approximation of the distinct count.

Example

```sql
SELECT
    APPROX_COUNT_DISTINCT(val) AS approx_distinct
FROM sample_data;
```

Output: `7`

The `sample_data` table has 8 rows, but there are only 7 distinct values in the `val` column (the value `4` appears twice).

APPROX_SUM

Computes a sum using a faster, non-deterministic ordering for floating-point columns. This summation can lead to minor differences in the result on the order of the machine epsilon. For integral column types, the function uses the standard `SUM` algorithm.

Syntax

```sql
APPROX_SUM(col)
```

Argument	Data Type	Description
`col`	`FLOAT`, `DOUBLE`, or any integral type	The column to sum.

For integral inputs, the `APPROX_SUM` function uses the standard `SUM` algorithm and returns the same result. The performance benefit applies to large floating-point data sets.

Example

```sql
SELECT
    APPROX_SUM(c1) AS approx_total
FROM sys.dummy5;
```

Output: `15`

AVG

Computes the arithmetic mean over the set of values.

Syntax

```sql
AVG(col)
```

Argument	Data Type	Description
`col`	Any numeric type or `MATRIX`	The column to average. Integral and `FLOAT` inputs return `DOUBLE`. `DOUBLE`, `DECIMAL`, and `MATRIX` inputs preserve their type.

Example

```sql
SELECT
    AVG(c1) AS avg_val
FROM sys.dummy5;
```

Output: `3.0`

COEFFICIENT_OF_DETERMINATION

Computes the coefficient of determination (R^2) between actual and predicted values. Primarily used to evaluate the performance of machine learning regression models, R^2 measures the proportion of variance in the actual values that the predicted values explain. A value of `1.0` indicates a perfect fit, while `0.0` indicates that the predictions explain none of the variance.

This function returns `NULL` when the total sum of squares is zero (i.e., all actual values are identical).

Syntax

```
SQL

COEFFICIENT_OF_DETERMINATION(actual, predicted)
```

Argument	Data Type	Description
`actual`	Any numeric type	The column of observed values.
`predicted`	Any numeric type	The column of predicted values.

Example

```
SQL

SELECT
    COEFFICIENT_OF_DETERMINATION(actual, predicted) AS r_squared
FROM predictions;
```

Output: `0.980571428571429`

CONFUSION_MATRIX

Returns a structured representation of the counts for every combination of actual and predicted class labels relative to a specified positive class. This function aggregates predictions into a 2x2 matrix in the format `[[TP, FP], [FN, TN]]` for binary classification analysis.

SQL

```
CONFUSION_MATRIX(y, y_hat, positive_class)
```

Argument	Data Type	Description
`y`	All types supported	The actual class label.
`y_hat`	All types supported	The predicted class label.
`positive_class`	All types supported	The class label to treat as the positive class. Must match the type of `y` and `y_hat`.

The return type is `TUPLE`. Each entry in the result contains the count of rows for a specific combination of actual and predicted class values, organized as true positives, false positives, false negatives, and true negatives.

Examples

Calculate the Confusion Matrix with Matched Values

This query returns the confusion matrix when all predictions match the actual values.

SQL

```
SELECT CONFUSION_MATRIX(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

Output: `[[5, 0], [0, 5]]`

Calculate the Confusion Matrix with Mismatched Classifiers

This query returns the confusion matrix for two mismatched binary classifiers, where actual positives are rows with `c1 > 5` and predicted positives are rows with `c1 < 3`.

```sql
SELECT CONFUSION_MATRIX(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 < 3 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

Output: `[[0, 2], [5, 3]]`

CORR

Alias for **CORRELATIONP**.

CORRELATION

Alias for **CORRELATIONP**.

CORRELATIONP

Computes the population Pearson correlation coefficient between two columns. Returns `DOUBLE`, or `DECIMAL` if both inputs are `DECIMAL`-compatible.

Alias for `CORRELATION` and `CORR`.

Syntax

```sql
CORRELATIONP(col1, col2)
```

Argument	Data Type	Description
`col1`	Any numeric type	The first column.
`col2`	Any numeric type	The second column.

Example

SQL

```
SELECT
    CORRELATIONP(actual, predicted) AS corr_val
FROM predictions;
```

Output: `0.9904654955172436`

COUNT

Returns the number of rows in the set where values in `col` are not `NULL`. When you use the `COUNT(*)` SQL statement, the function returns the total number of rows. The return type is `BIGINT`.

Syntax

SQL

```
COUNT(col)
```

Argument	Data Type	Description
`col`	Any non-interval type	The column for counting non-`NULL` values. Use `*` to count all rows instead.

Examples

Count All Rows

```sql
SELECT COUNT(*) AS total_rows FROM sample_data;
```

Output: `8`

Count Non-NULL Values

The `label` column has one `NULL` row.

```sql
SELECT COUNT(label) AS rows_with_label FROM sample_data;
```

Output: `7`

COVAR_POP

Alias for **COVARIANCEP**.

COVAR_SAMP

Alias for **COVARIANCE**.

COVARIANCE

Computes the sample covariance between two columns. Returns `DOUBLE` by default, or `DECIMAL` if both inputs are `DECIMAL`-compatible. `MATRIX` inputs preserve their type.

Alias for `COVAR_SAMP`.

Syntax

```sql
COVARIANCE(col1, col2)
```

Argument	Data Type	Description
`col1`	Any numeric type or `MATRIX` (square)	The first column.
`col2`	Any numeric type or `MATRIX` (square)	The second column.

Matrix arguments must have matching dimensions.

Example

SQL

```
SELECT
    COVARIANCE(actual, predicted) AS cov_samp
FROM predictions;
```

Output: `336.0`

COVARIANCEP

Computes the population covariance between two columns. Returns `DOUBLE` by default, or `DECIMAL` if both inputs are compatible with `DECIMAL`.

`MATRIX` inputs preserve their type.

Alias for `COVAR_POP`.

Syntax

SQL

```
COVARIANCEP(col1, col2)
```

Argument	Data Type	Description
`col1`	Any numeric type or `MATRIX` (square)	The first column.
`col2`	Any numeric type or `MATRIX` (square)	The second column.

Matrix arguments must have matching dimensions.

Example

```
SQL

SELECT
    COVARIANCEP(actual, predicted) AS cov_pop
FROM predictions;
```

Output: `280.0`

F1_SCORE

Returns the harmonic mean of precision and recall for a specified positive class. This function calculates `2 * (Precision * Recall) / (Precision + Recall)` and provides a single score that balances both precision and recall. The function returns `NULL` when both precision and recall are 0.

```
SQL

F1_SCORE(y, y_hat, positive_class)
```

Argument	Data Type	Description
`y`	All types supported	The actual class label.
`y_hat`	All types supported	The predicted class label.
`positive_class`	All types supported	The class label to treat as the positive class. Must match the type of `y` and `y_hat`.

The return type is `DOUBLE`. The result ranges from 0 to 1, where 1 represents perfect precision and recall.

Examples

Calculate the F1 Score with Matching Values

This query returns the F1 score when all predictions match the actual values.

SQL

```
SELECT F1_SCORE(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

Output: `1`

Calculate the F1 Score for Mismatched Classifiers

This query returns `NULL` because both precision and recall are 0 for the mismatched classifiers.

SQL

```
SELECT F1_SCORE(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 < 3 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

Output: `NULL`

KURTOSIS

Computes the sample excess kurtosis over the set of values. Kurtosis measures the difference between the tails of a distribution and the tails of a normal distribution. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

Syntax

SQL

```
KURTOSIS(col)
```

Argument	Data Type	Description
`col`	Any numeric type	The column for computing the sample kurtosis.

Example

```
SQL

SELECT
    KURTOSIS(val) AS kurt_val
FROM sample_data;
```

Output: `4.763725276787391`

KURTOSISP

Computes the population kurtosis over the set of values. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

Syntax

```
SQL

KURTOSISP(col)
```

Argument	Data Type	Description
`col`	Any numeric type	The column for computing population kurtosis.

Example

```
SQL

SELECT
    KURTOSISP(val) AS kurtp_val
FROM sample_data;
```

Output: `2.268440607993995`

© 2026 Ocient, Inc.

MAX

Returns the maximum value in the specified column. The return type matches the input.

Syntax

```SQL
MAX(col)
```

Argument	Data Type	Description
`col`	Any non-interval type	The column for returning the maximum value.

Example

```SQL
SELECT
    MAX(c1) AS max_val
FROM sys.dummy5;
```

Output: `5`

MEAN_ABSOLUTE_ERROR

Returns the mean absolute error (MAE) between actual and predicted values. This function calculates the average of the absolute differences between each pair of actual and predicted values, treating all errors equally regardless of direction.

```SQL
MEAN_ABSOLUTE_ERROR(y, y_hat)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`y`	NUMERIC	The actual value.
`y_hat`	NUMERIC	The predicted value.

The return type is `DOUBLE`. The result is always zero or positive. A value of 0 indicates perfect predictions.

Examples

Calculate the MAE for Matched Values

This query returns the mean absolute error when all predictions match the actual values.

SQL

```
SELECT MEAN_ABSOLUTE_ERROR(c1, c1) FROM sys.dummy10;
```

Output: `0`

Calculate the MAE with Comparison Against Squared Values

This query returns the mean absolute error when comparing actual values against their squared values.

SQL

```
SELECT MEAN_ABSOLUTE_ERROR(c1, c1 * c1) FROM sys.dummy10;
```

Output: `33.0`

MEAN_ABSOLUTE_PERCENTAGE_ERROR

Returns the mean absolute percentage error (MAPE) between actual and predicted values. This function calculates the average of the absolute percentage differences using the formula `Average( AbsoluteValue( (y - y_hat) / y ) ) * 100`, which expresses the error relative to the actual values.

```
SQL
MEAN_ABSOLUTE_PERCENTAGE_ERROR(y, y_hat)
```

Argument	Data Type	Description
`y`	NUMERIC	The actual value. Must not be 0, because this value is the denominator in the percentage calculation.
`y_hat`	NUMERIC	The predicted value.

The return type is `DOUBLE`. The result is a percentage value where 0 indicates perfect predictions.

Examples

Calculate the MAPE for Matched Values

This query returns the mean absolute percentage error when all predictions match the actual values.

```
SQL
SELECT MEAN_ABSOLUTE_PERCENTAGE_ERROR(c1, c1) FROM sys.dummy10;
```

Output: `0`

Calculate the MAPE with Comparison Against Squared Values

This query returns the mean absolute percentage error when comparing actual values against their squared values.

```
SQL
SELECT MEAN_ABSOLUTE_PERCENTAGE_ERROR(c1, c1 * c1)
    FROM sys.dummy10;
```

Output: `450.0`

MIN

Returns the minimum value in the specified column. The return type matches the input.

Syntax

```sql
MIN(col)
```

Argument	Data Type	Description
`col`	Any non-interval type	The column for returning the minimum value.

Example

```sql
SELECT
    MIN(c1) AS min_val
FROM sys.dummy5;
```

Output: `1`

PRECISION_SCORE

Returns the precision score for a specified positive class. This function calculates the ratio of true positives to the total number of predicted positives using the formula `True Positives / (True Positives + False Positives)`. Of all the predictions for the specified class, this score measures how many are correct.

```sql
PRECISION_SCORE(y, y_hat, positive_class)
```

Argument	Data Type	Description
`y`	All types supported	The actual class label.
`y_hat`	All types supported	The predicted class label.
`positive_class`	All types supported	The class label to treat as the positive class. Must match the type of `y` and `y_hat`.

The return type is `DOUBLE`. The result ranges from 0 to 1, where 1 indicates that every prediction for the positive class is correct.

Examples

Calculate the Precision Score for Matched Values

This query returns the precision score when all predictions match the actual values.

SQL

```
SELECT PRECISION_SCORE(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

Output: `1`

Calculate the Precision Score for Mismatched Classifiers

This query returns the precision score for two mismatched binary classifiers, where none of the predicted positives are actual positives.

SQL

```
SELECT PRECISION_SCORE(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 < 3 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

© 2026 Ocient, Inc.

Output: `0.0`

PRODUCT

Computes the product of all values in the column.

Syntax

```
SQL

PRODUCT(col)
```

Argument	Data Type	Description
`col`	`FLOAT`, `DOUBLE`, `DECIMAL`, any integral type, or `MATRIX` (square)	The column to multiply. Integral inputs return `BIGINT`.

> Matrix arguments must have matching dimensions.

Example

```
SQL

SELECT
    PRODUCT(c1) AS factorial_5
FROM sys.dummy5;
```

Output: `120`

RECALL_SCORE

Returns the recall score for a specified positive class. This function calculates the ratio of true positives to the total number of actual positives using the formula `True Positives / (True Positives + False Negatives)`. Of all the actual instances of the specified class, this score measures how many the model correctly identifies.

SQL

```
RECALL_SCORE(y, y_hat, positive_class)
```

Argument	Data Type	Description
`y`	All types supported	The actual class label.
`y_hat`	All types supported	The predicted class label.
`positive_class`	All types supported	The class label to treat as the positive class. Must match the type of `y` and `y_hat`.

The return type is `DOUBLE`. The result ranges from 0 to 1, where 1 indicates that every actual instance of the positive class is correctly identified.

Examples

Calculate the Recall Score for Matched Values

This query returns the recall score when all predictions match the actual values.

SQL

```
SELECT RECALL_SCORE(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

Output: `1`

Calculate the Recall Score for Mismatched Classifiers

This query returns the recall score for two mismatched binary classifiers, where none of the actual positives are correctly predicted.

SQL

```
SELECT RECALL_SCORE(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    CASE WHEN c1 < 3 THEN 1 ELSE 0 END,
    1
) FROM sys.dummy10;
```

Output: `0.0`

ROC_AUC_SCORE

Returns the area under the receiver operating characteristic (ROC) curve (AUC). This function measures the ability of a binary classification model to distinguish between classes. A score of 1.0 represents a perfect classifier, while a score of 0.5 represents performance no better than random guessing.

SQL

```
ROC_AUC_SCORE(y_true, y_score, positive_class)
```

Argument	Data Type	Description
`y_true`	BOOLEAN or INTEGER	The true binary class label, where the values represent two classes (for example, 0 or 1).
`y_score`	NUMERIC	The predicted probability or confidence score for the positive class. A score of 1.0 is a perfect classifier, while 0.5 is no better than random guessing.
`positive_class`		The class label to treat as the positive class. Must match the type of `y_true`.

The return type is `DOUBLE`. The result ranges from 0 to 1.

Example

This query returns the AUC score for a classifier in which higher probability scores correspond to actual positives, indicating perfect class separation.

```sql
SELECT ROC_AUC_SCORE(
    CASE WHEN c1 > 5 THEN 1 ELSE 0 END,
    DOUBLE(c1) / 10,
    1
) FROM sys.dummy10;
```

Output: `1.0`

SKEW

Computes the sample skewness over the set of values. Skewness measures the asymmetry of a distribution. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

Syntax

```sql
SKEW(col)
```

Argument	Data Type	Description
`col`	Any numeric type	The column for computing the sample skewness.

Example

```sql
SELECT
    SKEW(val) AS skew_val
FROM sample_data;
```

Output: `0.8804164883619399`

SKEWP

Computes the population skewness over the set of values. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

Syntax

```
SQL

SKEWP(col)
```

Argument	Data Type	Description
`col`	Any numeric type	The column for computing the population skewness.

Example

```
SQL

SELECT
    SKEWP(val) AS skewp_val
FROM sample_data;
```

Output: `0.7059036122393627`

STDEV

Computes the sample standard deviation over the set of values. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

Alias for `STDDEV` and `STDDEV_SAMP`.

Syntax

```
SQL

STDEV(col)
```

Argument	Data Type	Description
`col`	Any numeric type	The column for computing the sample standard deviation.

Example

SQL

```
SELECT
    STDEV(val) AS stdev_val
FROM sample_data;
```

Output: `4.421942042651783`

STDDEV

Alias for **STDEV**.

STDDEV_POP

Alias for **STDEVP**.

STDDEV_SAMP

Alias for **STDEV**.

STDEVP

Computes the population standard deviation over the set of values. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

Alias for `STDDEV_POP`.

Syntax

SQL

```
STDEVP(col)
```

Argument	Data Type	Description
`col`	Any numeric type	The column for computing the population standard deviation.

Example

```sql
SELECT
    STDEVP(val) AS stdevp_val
FROM sample_data;
```

Output: `4.136348026943574`

SUM

Computes the sum of all values in the column.

Syntax

```sql
SUM(col)
```

Argument	Data Type	Description
`col`	Any numeric type or `MATRIX`	The column to sum. Integral inputs return `BIGINT`. d `FLOAT` inputs return `FLOAT`. `DOUBLE`, `DECIMAL`, and `MATRIX` inputs preserve their type.

Example

```sql
SELECT
    SUM(c1) AS total
FROM sys.dummy5;
```

Output: `15`

VAR_POP

Alias for **VARIANCEP**.

VAR_SAMP

Alias for **VARIANCE**.

VARIANCE

Computes the sample variance over the set of values. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

`MATRIX` inputs preserve their type and must be square.

Alias for `VAR_SAMP`.

Syntax

```sql
VARIANCE(col)
```

Argument	Data Type	Description
`col`	Any numeric type or `MATRIX` (square)	The column for computing the sample variance.

Matrix arguments must have matching dimensions.

Example

```sql
SELECT
    VARIANCE(val) AS var_val
FROM sample_data;
```

Output: `19.553571428571427`

VARIANCEP

Computes the population variance over the set of values. Returns `DOUBLE`, or `DECIMAL` if the input is `DECIMAL`.

`MATRIX` inputs preserve their type and must be square.

Alias for `VAR_POP`.

Syntax

```sql
VARIANCEP(col)
```

Argument	Data Type	Description
`col`	Any numeric type or `MATRIX` (square)	The column for computing population variance.

Example

```sql
SELECT
    VARIANCEP(val) AS varp_val
FROM sample_data;
```

Output: `17.109375`

© 2026 Ocient, Inc.

Sorted Aggregate Functions

The general syntax for sorted aggregate functions adds the DISTINCT and ORDER BY keywords in the function invocation.

```sql
AGGREGATE([DISTINCT] arg1, arg2, ... [ORDER BY ...])
```

ARRAY_AGG

Returns an array containing every row from the expression.

Syntax

```sql
ARRAY_AGG(expr)
```

Argument	Data Type	Description
`expr`	Any numeric type	Expression for aggregation into an array.

Example

Aggregate ten rows into an array in descending order.

```sql
SELECT
    ARRAY_AGG(c1 ORDER BY c1 DESC)
FROM sys.dummy10;
```

Output: `[10,9,8,7,6,5,4,3,2,1]`

ARRAY_CONCAT_AGG

Returns an array that concatenates arrays across rows. The input argument is a SQL expression. The element type must be consistent across rows. The function ignores NULL inputs and returns NULL only when all inputs are NULL.

The `DISTINCT` keyword removes duplicate arrays. To remove duplicate elements from an array, use the `ARRAY_DISTINCT` function during load.

The `ORDER BY` syntax controls the order of concatenation across rows. To sort elements in the array, use the `ARRAY_SORT` function during load.

Syntax

SQL

```
ARRAY_CONCAT_AGG(expr)
```

Argument	Data Type	Description
`expr`	SQL expression	Expression for concatenation across rows.

Example

Create the `data_type_example` table with a column that stores arrays of integers.

SQL

```
CREATE TABLE
    data_type_example (col_int_array INT[] NOT NULL DEFAULT 'INT[0,1,2,3]');
```

Insert one array with values `1`, `2`, and `3` into the table.

SQL

```
INSERT INTO data_type_example SELECT ARRAY[1,2,3];
```

Insert the second array with values `4`, `5`, and `6` into the table.

```sql
INSERT INTO data_type_example SELECT ARRAY[4,5,6];
```

Concatenate the two arrays into one array in ascending order.

```sql
SELECT ARRAY_CONCAT_AGG(col_int_array ORDER BY col_int_array ASC)
FROM data_type_example;
```

Output: `[1,2,3,4,5,6]`

STRING_AGG

Returns a string concatenated from every row of the expression.

Syntax

```sql
ARRAY_CONCAT_AGG(expr, delimiter)
```

Argument	Data Type	Description
`expr`	SQL expression	Expression for concatenation across rows.
`delimiter`		Optional.

Examples

Concatenate Rows as a String

Concatenate ten rows into a string in descending order. Cast integers into strings using the `CHAR` casting function.

```sql
SELECT
    STRING_AGG(char(c1) ORDER BY c1 DESC)
FROM sys.dummy10;
```

Output: `"10987654321"`

Concatenate Rows as a String with a Delimiter

Concatenate ten rows into a string in ascending order. Cast integers into strings using the `CHAR` casting function. Use the `|` delimiter.

```sql
SELECT
    STRING_AGG(char(c1), '|' ORDER BY c1 ASC)
FROM sys.dummy10;
```

Output: `"1|2|3|4|5|6|7|8|9|10"`

Related Links

Math Functions and Operators

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Array Functions and Operators

The Ocient® System enables you to work with data as arrays. The system contains functions that work with arrays and operators that enable you to parse data within arrays.

ARRAY_CAT_DISTINCT

Concatenates two or more arrays in the order of the input arguments. After concatenation, the function removes duplicates from the result array while preserving the first occurrence order. The function skips any NULL input arguments. If all input arrays are NULL, the function returns a NULL. If at least one NULL element is present in the array, the function returns a NULL in the result array with the position of the first NULL the function finds in the order of the input arguments.

Syntax

```
SQL

ARRAY_CAT_DISTINCT(input_array1, input_array2 [, ...])
```

Argument	Data Type	Description
`input_array1, input_array2 [, ...]`	ARRAY	Two or more arrays for concatenation and deduplication. The elements within all input arrays must have the same data type.

Examples

Concatenate Two Arrays and Remove Duplicates

Concatenate two arrays and remove duplicate elements within the arrays.

```
SQL

SELECT ARRAY_CAT_DISTINCT(ARRAY[1,2,2,3,1], ARRAY[2,3,4,4,5]);
```

© 2026 Ocient, Inc.

Output: `[1,2,3,4,5]`

Concatenate Two Arrays with NULLs and Remove Duplicates

Deduplicate the contents of the array. In this case, the array contains NULLs.

```
SQL
SELECT ARRAY_CAT_DISTINCT(ARRAY[1,2,2,NULL,1], ARRAY[2,4,4,NULL]);
```

Output: `[1,2,NULL,4]`

ARRAY_DISTINCT

Removes duplicates from an array while preserving the first occurrence order. If at least one NULL is present in the array, the function returns a NULL with the position of the first NULL of the input array in the output array.

Syntax

```
SQL
ARRAY_DISTINCT(input_array)
```

Argument	Data Type	Description
`input_array`	ARRAY	The array for deduplication.

Examples

Remove Duplicates from an Array

Deduplicate the contents of the array.

```
SQL
SELECT ARRAY_DISTINCT(ARRAY[1,2,2,3,1]);
```

Output: `[1,2,3]`

Remove Duplicates from an Array with NULLs

Deduplicate the contents of the array. In this case, the array contains NULLs.

```
SQL

SELECT ARRAY_DISTINCT(ARRAY[1,2,NULL,2,3,NULL,1]);
```

Output: `[1,2,NULL,3]`

ARRAY_LENGTH

Returns the length of the array for the specified dimension. If the array is a nested array, the function returns the length of the outermost array.

Syntax

```
SQL

ARRAY_LENGTH(input_array)
```

Argument	Data Type	Description
`input_array`	ARRAY	The array that contains any type of elements. The array can be single or multi-dimensional.

Example

Return the length of the array.

```
SQL

SELECT ARRAY_LENGTH(ARRAY[1,2]);
```

Output: `2`

ARRAY_SORT

Sorts and returns the input array based on the natural ordering of its elements. The behavior of this function varies depending on the syntax you use.

Syntaxes

Basic Array Sort

Sorts and returns the input array based on the natural ordering of its elements. If one of the array elements is NULL, then this function sorts the NULL values to the end of the array.

SQL

```
ARRAY_SORT(input_array)
```

Argument	Data Type	Description
`input_array`	ARRAY	An array of elements with these types: `BIGINT`, `BOOLEAN`, `BYTE`, `DATE`, `DOUBLE`, `FLOAT`, `INT`, `SMALLINT`, `UUID`, or `VARCHAR`.

Sort an Array With a Lambda Function

Sorts and returns the input array based on the results of the specified Lambda function. The function should have two arguments representing two elements of the array. This function should return a negative integer, 0, or a positive integer if the first element is less than, equal to, or greater than the second element, respectively.

SQL

```
ARRAY_SORT(input_array,function)
```

Argument	Data Type	Description
`input_array`	ARRAY	An array that contains any type of elements. The array can be single or multi-dimensional.
`function`	FUNCTION	This is an optional argument. A lambda function with the format `(x T, y T) -> INT` or another SQL reference

Argument	Data Type	Description
		function.

> The Ocient System supports this syntax with the data pipeline functionality only. For details, see **Transform Data in Data Pipelines**.

Sort an Array in the Specified Order

Determines how the array handles the order of elements in the array.

SQL

```
ARRAY_SORT(input_array,sort_order)
```

Argument	Data Type	Description
`input_array`	ARRAY	An array of elements with these types: `BIGINT`, `BOOLEAN`, `BYTE`, `DATE`, `DOUBLE`, `FLOAT`, `INT`, `SMALLINT`, `UUID`, or `VARCHAR`.
`sort_order`	BOOLEAN	This is an optional argument. Determines the sort order of the array. The value `true` means ascending order. The value `false` means descending order. The default value is `true`.

Sort an Array in the Specified Order with NULL Elements

Determines how the array handles the order of elements in the array and where NULL elements appear.

SQL

```
ARRAY_SORT(input_array,sort_order,nulls_first)
```

Argument	Data Type	Description
`input_array`	ARRAY	An array of elements with these types: `BIGINT`, `BOOLEAN`, `BYTE`, `DATE`, `DOUBLE`, `FLOAT`, `INT`, `SMALLINT`, `UUID`, or `VARCHAR`.

Argument	Data Type	Description
`sort_order`	BOOLEAN	This is an optional argument. Determines the sort order of the array. The value `true` means ascending order. The value `false` means descending order. The default value is `true`.
`nulls_first`	BOOLEAN	This is an optional argument. Determines whether NULL elements appear first or last in the array. The `true` value means that NULLs appear first in the array. The `false` value means that they appear last. If the array is in ascending order, the default value is `false`. Otherwise, the default value is `true`.

Examples

Sort an Array

Sort an array of three elements `[2,3,1]`.

SQL

```
SELECT ARRAY_SORT(ARRAY[2,3,1]);
```

Output: `[1,2,3]`

Sort an Array by Specifying the Sort Order

Sort an array of four elements `[2,3,1,NULL]` and specify the order of the elements in descending order.

SQL

```
SELECT ARRAY_SORT(ARRAY[2,3,1,NULL],false);
```

Output: `[NULL,3,2,1]`

Sort an Array by Specifying the Sort Order and NULL Placement

Sort an array of four elements `[2,NULL,3,1]` and specify the order of the elements in ascending order, and have the NULL element appear last.

SQL

```
SELECT ARRAY_SORT(ARRAY[2,NULL,3,1],true,false);
```

Output: `[1,2,3,NULL]`

UNNEST

Expands each element in an input array into an individual row.

For example, the `UNNEST` function on an array column of type `ARRAY(INT)` with values `[2, 6]` yields two result rows with integers `2` and `6`.

The values of the other columns in each input row are unchanged in each corresponding output row. You can specify multiple array columns to unnest the specified arrays from each row in parallel.

Syntax

SQL

```
UNNEST( array_column [, ... ] )
   [ WITH <unnest_options> ]

<unnest_options> ::=
   { ORDINALITY [ ord_identifier ]
   | VALUE [ val_identifier ]
   | NULL_INPUT }
```

Argument	Data Type	Description
`array_column`	`ARRAY`	The array column to expand.
`ord_identifier`	`CHAR`	When you use this argument in an `ORDINALITY` clause, the argument is the name for the output column.
`val_identifier`	`CHAR`	When you use this argument in a `VALUE` clause, the argument is the name for the unnested column.

© 2026 Ocient, Inc.

The `UNNEST` function supports these options. When you use only one of these options, enclosing the option in parentheses is optional. However, if you use more than one of these options, you must enclose them in parentheses.

Options	Description
`ORDINALITY [ord_identifier]`	Optional. Generates a new integer column that corresponds to the index position of each output element in the original array. The `ord_identifier` value specifies the name of the output ordinality column. If you do not specify this clause, the column name defaults to the original column name with the suffix `_ord`.
`VALUE [val_identifier]`	Optional. Rename the unnested column to the new name specified in the `val_identifier` argument. If you do not specify this clause, the column name defaults to the original column name with the suffix `_val`.
`NULL_INPUT`	Optional. When the input column array is empty or NULL, this clause specifies to replace each output row with an array with one NULL element. This replacement preserves the contents of other input rows. By default, this option is not enabled.

These examples demonstrate how the `UNNEST` function operates on arrays and other objects. Most of the examples use the data rows from this table.

Text

```
a   b        c        d     e      f
-----------------------------------------
1   [1, 2]   [6, 9]   []    NULL   [[1, 2], [5], []]
2   [7, 5]   [4]      [3]   [8]    [[6], [9, 4, 7], [3, 8]]
```

Examples

Expand a Column

This example performs a simple `UNNEST` on column `b`.

SQL

```
SELECT UNNEST(b);
```

Output

```
Text

b_val
-----------
1
2
7
5
```

Expand a Column Using the `ORDINALITY` Option

In this example, the `ORDINALITY` option adds a second return column that represents the index position of the value in the input array.

```
SQL

SELECT UNNEST(b) WITH ORDINALITY;
```

Output

```
Text

b_val   b_ord
--------------------
1       1
2       2
7       1
5       2
```

Expand a Column with Empty Data

In this example, the `UNNEST` query returns only one row because one of the two rows in column `d` is an empty array.

If you use the `NULL_INPUT` option, the query returns a second row with the `NULL` value.

SQL

```
SELECT UNNEST(d);
```

Output

Text

```
d_val
------------
3
```

Expand a Column with NULL Data

`UNNEST` also does not return array values that are `NULL` unless you specify the `NULL_INPUT` option.

SQL

```
SELECT UNNEST(e);
```

Output

Text

```
e_val
------------
8
```

Expand a Column with Multiple Array Layers

The `UNNEST` function expands only one array layer.

ⓘ The empty array remains in the returned values.

SQL

```
SELECT UNNEST(f);
```

Output

Text

```
f_val
------------
[1,2]
[5]
[]
[6]
[9,4,7]
[3,8]
```

Expand Columns Using Multiple `UNNEST` Statements

Multiple `UNNEST` statements in a query produce a per-row Cartesian product of all the unnested values.

SQL

```
SELECT UNNEST(b), UNNEST(c);
```

Output

Text

```
b_val   c_val
------------
1       6
1       9
2       6
2       9
7       4
5       4
```

Expand Columns Using Multiple `UNNEST` Statements in Reverse Order

This example uses `UNNEST` on the same columns but in reverse order.

SQL

```
SELECT UNNEST(c), UNNEST(b);
```

Output

Text

```
c_val   b_val
------------
6       1
6       2
9       1
9       2
4       7
4       5
```

Expand Multiple Columns

In this example, the query unnests two array columns in parallel. Even though it does not include the `NULL_INPUT` option, the query still returns NULL for column `c` to correspond with column `b`.

SQL

```
SELECT UNNEST(b, c);
```

Output

Text

```
b_val   c_val
------------
1       6
2       9
7       4
5       NULL
```

Expand M****ultiple Array Columns

This query unnests two array columns.

SQL

```
SELECT UNNEST(b, d);
```

Output

Text

```
b_val   d_val
------------
1       NULL
2       NULL
7       3
5       NULL
```

Expand Values of Arrays Within Arrays

This example has two layers of `UNNEST` to capture values of arrays within arrays.

SQL

```
SELECT UNNEST(f_val) FROM (UNNEST(f));
```

Output

Text

```
f_val
------------
1
2
5
6
9
4
7
3
8
```

Expand an Empty Array

This example unnests an empty integer array.

SQL

```
SELECT UNNEST(INT[]());
```

Output

Text

```
col_val
------------
```

The query returns a column of type `INT` but no rows.

Expand an Empty Array Column Without the `NULL_INPUT` Option

This example selects the non-array column `a` and an empty array column. The query returns no rows because empty or `NULL` array values are not returned unless you specify the `NULL_INPUT` option.

SQL

```
SELECT a, UNNEST(INT[]());
```

Output

Text

```
a   col_val
------------
```

Expand a Column Without the `NULL_INPUT` Option

This query does not include the `NULL_INPUT` option, which causes the result to omit the NULL array values in column `e`.

SQL

```
SELECT a, UNNEST(e);
```

Output

Text

```
a   e_val
------------
1   8
2   8
```

Expand a Column with the `NULL_INPUT` Option

This query includes the `NULL_INPUT` option, which means that the query returns the NULL values in column `e`.

SQL

```
SELECT a, UNNEST(e) WITH NULL_INPUT;
```

Output

```
Text
a   e_val
------------
1   8
1   NULL
2   8
2   NULL
```

Expand Arrays with NULL Rows

This example attempts to unnest two arrays, one of which is cast as NULL, while the other is empty.

In both cases, the query returns a column of type `INTEGER` with no rows because there are no values to unnest.

```
SQL
SELECT UNNEST(CAST(NULL AS INT[]);

SELECT UNNEST(INT[]());
```

Output

```
Text
col_val
------------
```

Other Array Functions

Function	Syntax	Purpose
Constructor	TYPE[](e1, e2, ..., en)	Construct an array of SQL type TYPE giving the elements. NULL is also supported as an element.

Function	Syntax	Purpose
Constructor (2)	array[e1, e2, ..., en]	PostgreSQL®-compliant constructor. The type of the array is deduced from the elements.
array concatenation	array_cat(array, array)	Concatenate 2 arrays into a new one.
array softmax	softmax(array)	Returns the softmax of the array. The array must be one-dimensional and contain float or double values. The system transfers NULL-valued indices as 0.0.
array cross entropy loss	cross_entropy_loss(array, array)	Returns the cross entropy loss of two arrays. Both arrays must be one-dimensional and contain float or double values. NULL values do not contribute to the sum.
array log loss	log_loss(array, array)	Returns the log loss of two arrays. Both arrays must be one-dimensional and contain float or double values. NULL values do not contribute to the sum.
array logits loss	logits_loss(array, array)	Returns the logits loss of two arrays. Both arrays must be one-dimensional and contain float or double values. NULL values do not contribute to the sum.
array hinge loss	hinge_loss(array, array)	Returns the hinge loss of two arrays. Both arrays must be one-dimensional and contain float or double values. NULL values do not contribute to the sum.
array sum	array_sum(array)	Returns the sum of the array. The array must be one-dimensional and contain numeric values. NULL values do not contribute to the sum.
array min/max	array_min(array), array_max(array)	Returns the corresponding minimum or maximum of the array. This function only considers the first dimension of the array. If the array contains only NULL values, then the function returns NULL.
array argmax	array_argmax(array)	Returns the 1-based position (index) of the maximum element in an array. The first element has index 1. This function only considers the first dimension of the array. If the array contains only NULL values, then the function returns NULL.
array argmin	array_argmin(array)	Returns the 1-based position (index) of the minimum element in an array. The first element has index 1. **

Function	Syntax	Purpose
		This function only considers the first dimension of the array. If the array contains only NULL values, then the function returns NULL.
array casting	cast_to_array(array, format)	Casts the elements of the array to another type, as specified by the format string. The format is: `'ARRAY(INT)'` for an `int` array, `'ARRAY(CHAR)'` for a `char` array, and so on. Currently, only numeric casts are supported.
scalar position	array_position(array, scalar, pos = 1)	Returns the position of the first matching scalar in the array. pos indicates the position to start searching, if not specified it defaults to 1.
array elements positions	array_positions(array, array)	Same as array_position(). But this looks for all elements stored in the right array and returns their respective positions in the left array.
array append	array_append(array, value)	Add value to the back of an array
array prepend	array_prepend(value, array)	Add value to the front of an array
array remove	array_remove(array, value)	remove value from the array
array replace	array_replace(array, old_value, new_value)	replace value by another in an array
convert array to string	char(array)	Converts array to its string representation.
convert array to string	array_to_string(array, delimiter [, nullstr])	Converts array to a list of elements separated by ‘delimiter’. Nested arrays are flattened. NULL values are either ignored or replaced by *nullstr* if provided.
convert string to array	string_to_array(str, format)	Converts the string representation of an array (e.g., `'int[1,2,NULL]'`) into an array. The format string is the same as in cast_to_array.

Function Examples

Function	SQL statement	Result
type constructor	int[](1,2)	int array with values 1 and 2
constructor with no type	array[1,2]	bigint array with values 1 and 2

Function	SQL statement	Result
array_cat	array_cat(array[2, 3], array[4, 5])	array[2, 3, 4, 5]
array_length	array_length(array[2, 3])	2
cast_to_array	cast_to_array(array[2.5, 3.4], ‘ARRAY(INT)‘)	int[](2,3)
array_position	array_position(array[1, 2, 1], 1)	1
	array_position(array[1, 2, 3, 4, 1], 1, 3)	5
array_positions	array_positions(array[1, 2, 3, 4, 1], 1)	int[](1,5)
array_append	array_append(array[1,2], 3)	array[1,2,3]
array_prepend	array_prepend(0, array[1,2])	array[0,1,2]
array_remove	array_remove(array[1,2], 1)	array[2]
array_replace	array_replace(array[1,2,4],4,3)	array[1,2,3]
softmax	softmax(float[](2.0, 2.0, 2.0, NULL, 2.0))	float[0.25, 0.25, 0.25, 0.0, 0.25]
array_sum	array_sum(int[](1, 2, 3))	6
array_min	array_min(int[](1, -2, 3))	-2
array_argmax	array_argmax(int[](1, 3, 2))	2
array_argmin	array_argmin(int[](1, 3, 2))	1
char(<array value>)	char(array[1,2])	‘bigint[1,2]‘
array_to_string	array_to_string(int[1,2,3,NULL,5], ’,’, ’*’)	‘1,2,3,*,5’
string_to_array	string_to_array(‘int[1,2]’, ‘ARRAY(INT)‘)	int[](1,2)

Array Operators

Ocient array operators allow you to concatenate and check for containment or overlap of data. Also, you can retrieve specific elements or slices within arrays.

⚠ Do not use array operators, including `@>`, `<@`, and `&&`, to evaluate `NULL` values in arrays. For information on how to check for `NULL` values in arrays, see **Array NULL Handling**.

© 2026 Ocient, Inc.

Contains Operator (`@>`)

The `@>` operator determines whether a left-side array contains a scalar value or array elements on the right side.

`@>` Syntax

```
SQL

array @> scalar_or_array
```

Argument	Data Type	Description
`array`	`ARRAY`	An array column or expression.
`scalar_or_array`	`ARRAY` or any numeric, character, time, or geospatial data type.	A value or an array of values to check whether they are contained in the array. For data types, see **Understanding Data Types**.

Examples

These examples use the `@>` operator to test whether the left-side array contains all the elements from the right-side array.

Array Containment (True Case)

This example returns `true` because all right-side elements are in the left-side array.

```
SQL

SELECT ARRAY[1, 4, 3] @> ARRAY[3, 1];
```

Output: `true`

Array Containment (False Case)

If the right-side array contains at least one value not present in the left-side array, the query returns `false`. In this example, the right-side array has one value, `5`, not present on the left side.

```sql
SELECT ARRAY[1, 4, 3] @> ARRAY[3, 1, 5];
```

Output: `false`

Scalar Containment in an Array

This example checks whether a single scalar value is in the left-side array.

```sql
SELECT ARRAY[3, 1, 3] @> 1;
```

Output: `true`

Contained In Operator (`<@`)

The `<@` operator checks whether a right-side array contains all the elements on the left side.

Syntax

```sql
scalar_or_array <@ array
```

Argument	Data Type	Description
`array`	`ARRAY`	An array column or expression.
`scalar_or_array`	`ARRAY` or any numeric, character, time, or geospatial data type.	A value or an array of values to check whether they are contained in the array. For data types, see **Understanding Data Types**.

Examples

Array Contained Within Another Array (True Case)

This example returns `true` because all left-side elements are in the right-side array.

```sql
SELECT VARCHAR[]('apples', 'oranges') <@ VARCHAR[]('apples', 'oranges', 'bananas')
```

Output: `true`

String Membership Check

`<@` and other array operators can check whether individual strings are present in an array. The `'oranges'` string is present in the right-side array.

```sql
SELECT 'oranges' <@ VARCHAR[]('apples', 'oranges', 'bananas')
```

Output: `true`

Overlap Operator (`&&`)

The overlap operator `&&` determines whether any elements between two arrays are common.

Syntax

```sql
array1 && array2
```

Argument	Data Type	Description
`array1`	`ARRAY`	An array column or expression to check whether any of its values are shared with `array2`.
`array2`	`ARRAY`	An array column or expression to check whether any of its values are shared with `array1`.

Examples

Check Overlap Between Arrays (True Case)

As long as at least one value is present in both arrays, the `&&` operator returns `true`. Both of these arrays contain the value `3`.

```
SQL

SELECT ARRAY[1 ,2, 3] && ARRAY[3 ,4, 5];
```

Output: `true`

Check Overlap Between Arrays (False Case)

Both of these arrays have no values in common, so the `&&` operator returns `false`.

```
SQL

SELECT ARRAY[1 ,2, 3] && ARRAY[4 ,5, 6];
```

Output: `false`

Slice Operator (`:`)

The slice operator `:` returns a subarray ranging from a left index to a right index, both of which are optional to specify. If you exclude both indexes, the slice operator returns the full array.

Syntax

```
SQL

array[left_index:right_index]
```

Argument	Data Type	Description
`array`	`ARRAY`	An array column or expression.
`left_index`	`INT`	Optional. A starting index for the slice operator to begin the

Argument	Data Type	Description
		subarray. Index values start at `1`. If you do not specify a left index, or if this value is less than zero, then the slice starts at index `1`.
`right_index`	`INT`	Optional. An ending index for the slice operator to end the subarray. Index values start at `1`. If you do not specify a right index or if the value exceeds the array length, the right index defaults to the array length.

The slice operator also follows these rules:

- Each index starts at `1`.
- If the array or either index value is NULL, the result of slicing is NULL.
- Ranges completely out of array bounds return an empty array. For example, `array[1, 2][4:5] = []`.
- Sequential slices slice each dimension of multidimensional arrays. For example, `ARRAY[ARRAY[1, 2, 3], ARRAY[4, 5, 6]][1:1][2:3] = ARRAY[ARRAY[2,3]]`.
- More than N sequential slices of an N-dimensional array, for example, three sequential slices on a two-dimensional array return an empty array. For example, `ARRAY[1, 2, 3][:][:] = []`.
- You cannot combine slicing with the access operator when slicing multidimensional arrays. Any access operator `[n]` converts to `[:n]`. For example, `Array_Val[4][1:6]` would be equivalent to `Array_Val[:4][1:6]`.
- Slicing an array of tuples such as `ARRAY[TUPLE<<INT,INT>>(1,2), TUPLE<<INT,INT>>(3,4)][1:1][2:3]` slices `[1:1]` on the array and `[2:3]` on the tuple elements within the sliced array, and returns the value `TUPLE<<INT>>[TUPLE<<INT>>(2)]`.

Examples

Basic Array Slicing

Slice an array with four numbers starting at index `2` and ending at index `3`.

© 2026 Ocient, Inc.

```sql
SELECT ARRAY[1 ,2, 3, 4][2:3];
```

*Output: * `['2','3']`

Array Slicing With No Right Index

The query returns all values after the second value because it does not specify an ending index.

```sql
SELECT ARRAY[1 ,2, 3, 4][2:];
```

*Output: * `['2','3','4']`

Array Slicing With No Left Index

This query captures a subarray starting at the first index because it does not specify a starting value.

```sql
SELECT ARRAY[1 ,2, 3, 4][:3];
```

*Output: * `['1','2','3']`

Array NULL Handling

Filtering with array functions can have different outcomes if they operate on an array containing NULL values or a NULL value of array type.

Array comparison operators, such as `@>`, `<@`, and `&&`, do not follow normal Boolean logic when evaluating NULL values. To evaluate arrays for NULL values, use the filter functions `FOR_ALL()` or `FOR_SOME()`. For details about these functions, see **Array Filters**.

Examples

Evaluate a Single Array With No NULL Values

This example evaluates to `false` because none of the array values are NULL.

SQL

```
SELECT FOR_SOME(ARRAY[1, 2, 3]) IS NULL;
```

*Output: * `false`

Evaluate a Single Array With NULL Values

This example evaluates to `true` because the array contains a NULL value.

SQL

```
SELECT FOR_SOME(ARRAY[1, 2, NULL]) IS NULL;
```

*Output: * `true`

Select Only Arrays Containing NULL Values

To further demonstrate array NULL behavior, these examples use this table loaded with a few array values, some of which are NULL or contain NULL values.

SQL

```
CREATE TABLE demo_array_table (
    id INT,
    tags VARCHAR[]
);

INSERT INTO demo_array_table (id, tags) VALUES
    (1, VARCHAR[]('alpha', 'beta', 'gamma')),              -- Regular array
    (2, VARCHAR[]('delta', NULL, 'epsilon')),              -- Array containing a N
    (3, NULL),                                             -- Entirely NULL array
    (4, VARCHAR[]()),                                      -- Empty array
    (5, VARCHAR[](NULL, NULL));                            -- Array containing all
```

© 2026 Ocient, Inc.

In this example, the query uses a `FOR_SOME` function to filter the rows to return only arrays with NULL values. The output does not include the third row of the table because the row itself is NULL, and is not an array containing NULL values.

SQL

```
SELECT id, tags FROM demo_array_table
    WHERE FOR_SOME(tags) IS NULL;
```

Output

SQL

```
| id | tags                              |
|----|-----------------------------------|
|  2 | ['delta', NULL, 'epsilon']        |
|  5 | [NULL, NULL]                      |
```

Select NULL Values or Arrays Containing NULL Values

This example includes an additional filter to include any NULL rows and arrays with NULL values.

The example uses the `demo_array_table` table, which has values that are NULL or contain NULL values.

SQL

```
SELECT id, tags FROM demo_array_table
    WHERE FOR_SOME(tags) IS NULL
    OR tags IS NULL;
```

Output

SQL

```
| id | tags                          |
|----|-------------------------------|
|  3 | NULL                          |
|  5 | [NULL, NULL]                  |
|  2 | ['delta', NULL, 'epsilon']    |
```

Select Arrays Without NULL Values

This example filters the rows using the `FOR_ALL` filter function to return only arrays with no NULL values. The output also contains rows with empty arrays.

The example uses the `demo_array_table` table, which has values that are NULL or contain NULL values.

SQL

```
SELECT id, tags FROM demo_array_table
    WHERE FOR_ALL(tags) IS NOT NULL;
```

Output

SQL

```
| id | tags                          |
|----|-------------------------------|
|  4 | []                            |
|  1 | ['alpha', 'beta', 'gamma']    |
```

Select Only Non-Empty Arrays Without NULL Values

To omit empty arrays, you can add a filter to check the array length. The `ARRAY_LENGTH` function removes any empty arrays.

This example uses the `demo_array_table` table, which has values that are NULL or contain NULL values.

SQL

```
SELECT id, tags FROM demo_array_table
    WHERE FOR_ALL(tags) IS NOT NULL
    AND ARRAY_LENGTH(tags) > 0;
```

Output

SQL

```
| id | tags                          |
|----|-------------------------------|
|  1 | ['alpha', 'beta', 'gamma']    |
```

Multidimensional arrays

Multidimensional arrays are supported. As seen in these examples, each dimension, or length, at each level does not have to be the same. NULL values are allowed at each dimension of an array.

Multidimensional Examples

Function	SQL statement	Result
type constructor two-dimensional	`int[][](int[](1, NULL), array[1])`	two-dimensional `INT` array with values `[1, NULL]` and `[1]`
constructor with no type two-dimensional	`array[int[]\(1), array[int(2)]]`	two-dimensional `INT` array with values `[1]` and `[2]`
two-dimensional array with NULL values	`array[array[1, NULL], NULL]`	two-dimensional `BIGINT` array with values `[1, NULL]` and `NULL`
two-dimensional array with different lengths	`array[array[1], array[1,2], array[1,2,3]]`	two-dimensional `BIGINT` array with values `[1]`, `[1,2]`, and `[1,2,3]`

Related Links

Matrix Functions and Operators

Tuple Functions and Operators

Math Functions and Operators

Array Data Transformation Functions

Data Types

Data Types for Data Pipelines

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Character and Binary Functions

All string functions support Unicode characters. Functions that transform character case are locale-sensitive.

These functions support the UTF-8 format:

- RTRIM
- LTRIM
- LEFT
- RIGHT
- TRIM
- TRANSLATE
- RPAD
- LPAD
- SUBSTRING

Index position values for character and string functions begin at position `1`. For example, the `"H"` in the string `"Hello"` is at position `1`.

ASCII

Returns the ASCII code value of the leftmost character of the character value.

The ASCII function only supports ASCII characters in the input argument.

Syntax

SQL

```
ASCII(char)
```

Example

```sql
SELECT ASCII('a');
```

*Output: * `97`

Example

```sql
SELECT ASCII('bc');
```

*Output: * `98`

BIT_LENGTH

Returns the length of the character value in bits.

Syntax

```sql
BIT_LENGTH(char)
```

Example

```sql
SELECT BIT_LENGTH('a');
```

*Output: * `8`

Example

```sql
SELECT BIT_LENGTH('ab');
```

*Output: * `16`

Example

```SQL
SELECT BIT_LENGTH('ab4');
```

*Output: * `24`

BTRIM

Alias for **TRIM**.

CHR

Converts an integer value to a string.

The value is first sign-extended to 8 bytes. Then, if it can be represented as 1 byte, a string is returned with that one byte.

Otherwise, a string is returned of length 2 bytes, 4 bytes, or 8 bytes with the bytes set to the big-endian representation of the integer, depending on how many high order zero bytes there are in the integer.

Syntax

```SQL
CHR(integer)
```

Example

```SQL
SELECT CHR(97);
```

*Output: * `a`

CHAR_LENGTH

Alias for **LENGTH**.

CHARACTER_LENGTH

Alias for **LENGTH**.

CONCAT

Concatenates two or more values, all of which must be binary, hash, or string data types.

For strings, as long as one argument is a character value, the other arguments are implicitly cast to a character value.

Syntax

```SQL
CONCAT(value1, value2 [, ...])
```

Arguments	Data Type	Description
`value1, value2 [, ...]`	`BINARY`, `HASH`, or `CHAR`	Two or more values to concatenate. If any argument is a character value, the others are also cast to a character value.

Example

```SQL
SELECT CONCAT('ocient',' ','data', ' ', 'warehouse');
```

*Output: * `ocient data warehouse`

Example

```sql
SELECT 'ocient'||' data warehouse';
```

*Output: * `ocient data warehouse`

ENDSWITH

Returns `true` if `x` ends with `y` and `false` otherwise.

Syntax

```sql
ENDSWITH(char1, char2)
```

Argument	Data Type	Description
`char1`	`CHAR`	A string to evaluate for whether it ends with `char2`.
`char2`	`CHAR`	A string to evaluate for whether `char1` ends with it as a substring.

Example

```sql
SELECT ENDSWITH('ocient data warehouse','warehouse');
```

*Output: * `true`

Example

```sql
SELECT ENDSWITH('ocient data warehouse','db');
```

*Output: * `false`

Example

```sql
SELECT ENDSWITH('tamaño','o');
```

*Output: * `true`

INITCAP

For each word in the specified string, capitalize the first character if it is alphabetic. The system defines words as alphanumeric strings separated by non-alphanumeric characters. The system converts all other alphabetic characters to lowercase.

Syntaxes

```sql
INITCAP(char_string)
INITCAP(char_string, delimiter)
```

Argument	Data Type	Description
`char_string`	`CHAR`	A string to evaluate for initial capitalization.
`delimiter`	`CHAR`	Optional. One or more non-alphanumeric characters that specify the delimiter between words. The characters include all the ASCII printable non-alphanumeric characters except backquotes `` ` `` and equals `=`. Non-ASCII characters are not included. ℹIf you specify alphabetic characters for the delimiter, the function converts them to lowercase. **Example** `sql SQL SELECT INITCAP('ocient'); ` *Output: * `Ocient`

INSTR

Returns the index position of the first occurrence where the character value `char_substring` appears in the character value `char` by ignoring the case.

Syntax

```SQL
INSTR(char, char_substring)
```

Argument	Data Type	Description
`char`	`CHAR`	A string to evaluate for the first index position where `char_substring` appears.
`char_substring`	`CHAR`	A string to evaluate for where it first appears in `char`.

Example

```SQL
SELECT INSTR('ocient dw dw', 'ocient');
```

*Output: * `1`

```SQL
SELECT INSTR('ocient dw dw', 'dw');
```

*Output: * `8`

JSON_EXTRACT_PATH_TEXT

Returns the value for the key-value pair referenced by a series of path elements in a JSON string.

Syntax

SQL

```
JSON_EXTRACT_PATH_TEXT (
    json_string, path [, path2 [, ...] ] [, null_if_invalid ] )
```

Argument	Data Type	Description
`json_string`	`CHAR`	A JSON string.
`path [, path2 [, ...] ]`	`CHAR`	One or more path elements in a JSON string. You must include at least one path element up to a maximum of five, meaning the function can extract from paths nested up to five levels deep in the JSON string. Path elements are case-sensitive. If a specified path element does not exist in the JSON string, the function returns `NULL`.
`null_if_invalid`	`BOOLEAN`	Optional. If you set this value to `TRUE`, the function returns `NULL` if the JSON string is invalid. If you set this value to `FALSE`, the function returns an error when the JSON string is invalid. The default value is `FALSE`.

Examples

Extract Values from Nested Paths

This example extracts the value based on two specified paths in the JSON string, `n4` and `n6`.

SQL

```
SELECT JSON_EXTRACT_PATH_TEXT(
    '{"n2":{"n3":1},"n4":{"n5":99,"n6":"circle"}}',
    'n4', 'n6'
    );
```

*Output: * `"circle"`

Return NULL from Invalid JSON

In this example, the query requests the same nested paths, but the JSON is invalid because all of the keys lack quotation marks.

The query returns `NULL` because the `null_if_invalid` argument equals `TRUE`.

SQL

```
SELECT JSON_EXTRACT_PATH_TEXT(
    '{n2:{n3:1},n4:{n5:99,n6:"circle"}}',
    'n4', 'n6',
    TRUE
    );
```

*Output: * `NULL`

LCASE

Alias for **LOWER**.

LEFT

Return the number of characters in the string equal of the value `integer`. If `integer` is negative, the function returns all but the last `integer` characters.

Syntax

SQL

```
LEFT(char, integer)
```

Argument	Data Type	Description
`char`	`CHAR`	A string to be modified by returning the number of characters equal to `integer`.
`integer`	`INT`	The number of characters to return from the `char` string. If `integer` is negative, the function returns all but the last `integer` characters.

Example

```sql
SELECT LEFT('ocient data warehouse', 8);
```

*Output: * `ocient d`

Example

```sql
SELECT LEFT('ocient data warehouse', -3);
```

*Output: * `ocient data wareho`

LENGTH

Alias for CHAR_LENGTH and CHARACTER_LENGTH.

Returns the length of the value. For character data types, this value is in terms of characters. For binary data types, this value is in terms of bytes.

Syntax

```sql
LENGTH(character_or_binary_value)
```

Example

```sql
SELECT LENGTH('ocient data warehouse');
```

*Output: * `21`

LOCATE

Alias for POSITION.

Returns the index position of the first occurrence of the character value `substring` in the character value `string`.

Optionally, you can also include an additional value `offset` to offset the LOCATE function by the specified number of spaces.

Index positions begin at `1`.

Syntax

SQL

```
LOCATE(substring, string [, offset] )
```

Argument	Data Type	Description
`substring`	`CHAR`	A substring to be found for its first position in string.
`string`	`CHAR`	A string to be evaluated for the first position of `substring`.
`offset`	`INT`	Optional. A number of index positions in `string` to offset the search for the `substring` value. This index position must be a positive integer, and it starts from the left of `string`.

Example

SQL

```
SELECT LOCATE('ware', 'ocient data warehouse');
```

*Output: * `13`

Example

```sql
SELECT LOCATE('e', 'ocient');
```

*Output: * `4`

Example

In this example, the index starts at position 5. This means the function skips the initial `'e'` in the string. Instead, it returns the second `e` at index position 15.

```sql
SELECT LOCATE('e', 'ocient database', 5);
```

*Output: * `15`

LOWER

Alias for LCASE.

Convert string to lowercase.

Syntax

```sql
LOWER(char)
```

Example

```sql
SELECT LOWER('Ocient');
```

*Output: * `ocient`

Example

```sql
SELECT LOWER('OCIENT');
```

*Output: * `ocient`

LPAD

Pad the input text to the specified length with the pad string on the left side. If text is longer than length, it is truncated to `length` characters. If the argument pad is not provided, the space character is used.

Syntax

```sql
LPAD(string, length [, pad_character] )
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be padded on its left side with additional characters, so that it equals the `length` value.
`length`	`INT`	The number of characters for `string` to total. If the `string` is already longer than `length`, the function truncates `string` to equal the `length` value.
`pad_character`	`CHAR`	Optional. A string of one or more characters to use to pad `string` to equal the `length` value. If not provided, `pad_character` uses a whitespace character to pad.

Example

```sql
SELECT LPAD('ocient data warehouse',30,'ab');
```

*Output: * `abababababaocient data warehouse`

Example

```sql
SELECT LPAD('ocient data warehouse',6);
```

*Output: * `ocient`

LTRIM

Removes leading blanks from the string value `string`.

Alternatively, you can specify a second string value `trim_character`. If you specify the `trim_character` value, the LTRIM function removes all leading instances of the `trim_character` value from the `string`.

Syntax

```sql
LTRIM(string [, trim_character] )
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be trimmed on its left side.
`trim_character`	`CHAR`	Optional. A string of one or more characters to be trimmed from the left side of `string`. Each character is trimmed individually, not as a cohesive substring. If you do not specify this argument, this value defaults to removing all whitespace from the left side of `string`.

Example

```sql
SELECT LTRIM(' ocient');
```

*Output: * `ocient`

Example

```
SQL

SELECT LTRIM('aaaaaocient','a');
```

*Output: * `ocient`

Example

In this example, all characters `'abeo '` are removed from the left side of the string.

```
SQL

SELECT LTRIM('aaeabe ocient','abeo ');
```

*Output: * `cient`

MD5

Returns the hexadecimal string (all lowercase) representing the md5 hash of `char`.

Syntax

```
SQL

MD5(char)
```

Example

```
SQL

SELECT md5('ocient');
```

*Output: * `438f03cf6e9ddf8793e02db25f2d2f88`

MID

Alias for **SUBSTRING**.

OCTET_LENGTH

Returns the length in bytes of a character or binary value.

Syntax

```sql
OCTET_LENGTH(value)
```

Example

```sql
SELECT OCTET_LENGTH('a');
```

*Output: * `1`

Example

```sql
SELECT OCTET_LENGTH('ocient');
```

*Output: * `6`

POSITION

Alias for **LOCATE**.

REGEXP_COUNT

Searches a string for all occurrences of a regular expression pattern. The function returns an integer representing the number of times the regular expression pattern occurs in the string.

Syntax

```
SQL

REGEXP_COUNT( string, pattern [, position [, parameters ] ] )
```

Argument	Data Type	Description
`string`	`CHAR`	The string to search, specified as a character string.
`pattern`	`CHAR`	The regular expression pattern to use in the search, specified as a character string.
`position`	`INT`	Optional. The position, specified as a positive integer, that represents the position within the `string` string where to begin the search, based on the number of characters. If this integer exceeds the number of characters in `string`, then the function returns `0`. The default value is `1`.
`parameters`	`CHAR`	Optional. Parameters, specified as a character string, that contains one or more characters representing regular expression options for pattern matching. Supported options are: `c` — Perform case-sensitive matching. The Ocient® System enables this type of matching by default. `i` — Perform case-insensitive matching. `p` — Interpret the pattern using the Perl Compatible Regular Expression (PCRE) dialect. For details, see **PCRE**. The Ocient System enables this interpretation by default.

Example

This example searches the `'ABABDaSGRESaB'` string for the count of occurrences of the `'AB'` string by ignoring the case sensitivity. The search starts at position `1`.

```
SQL

SELECT REGEXP_COUNT('ABABDaSGRESaB', 'AB', 1, 'i');
```

*Output: * `3`

REGEXP_INSTR

Searches a string using a regular expression pattern and returns an integer representing the start position or end position of the substring that matches.

The function returns `0` if no match is found.

The `REGEXP_INSTR` function is similar to the **POSITION** function, but it allows greater precision with regular expressions.

Syntax

SQL

```
REGEXP_INSTR( source_string, pattern [, position [, occurrence] [, option [, parame
```

Argument	Data Type	Description
`string`	`CHAR`	The string to search, specified as a character string. Note that SQL escape sequences for string literals can override regular expression escape sequences. For details, see **String Literals and Escape Sequences**.
`pattern`	`CHAR`	The regular expression pattern to use in the search, specified as a character string.
`position`	`INT`	Optional. A positive integer that represents the position within the `string` string where to begin the search, based on the number of characters. This value alters only the start position for the match, not the returned string position. If this integer exceeds the number of characters in `string`, then the function returns `0`. The default value is `1`.
`occurrence`	`INT`	Optional. A positive integer that represents the occurrence of a positive pattern match to return. The default value is `1`, which means the function returns the first substring that matches the regular expression pattern.
`option`	`INT`	Optional. An integer that specifies whether to return the position of the matching start position or the end position. Your choices are: * `0`: Returns the start position of the match. * `1`: Returns the end position of the match `+1`.

Argument	Data Type	Description
		The function treats any nonzero integer value as `1`. The default value is `0`.
`parameters`	`CHAR`	Optional. A string containing one or more characters that represents the regular expression options for pattern matching. Supported options are: `c` — Perform case-sensitive matching. The Ocient System enables this type of matching by default. `i` — Perform case-insensitive matching. `e` — Extract the substring using a regular expression subpattern. This subpattern is enclosed in parentheses in the regular expression. The function uses the full regular expression pattern for the match but returns only the first subpattern match. `p` — Interpret the pattern using the Perl Compatible Regular Expression (PCRE) dialect. For details, see **PCRE**. The Ocient System enables this interpretation by default.

Example

The query searches the website URL for the substring preceded by the `#` character. The function includes optional arguments to start the search at position `9` and match the first occurrence. The last specified optional argument directs the function to return the ending position of the matching substring.

SQL

```
SELECT REGEXP_INSTR(
    'https://docs.ocient.com/character-binary-functions#J28jB',
    '#\w+',
    9,
    1,
    1
    );
```

*Output: * `57`

REGEXP_REPLACE

Searches a string for all occurrences of a regular expression pattern. The function replaces every match occurrence of the pattern with a new string.

The `REGEXP_REPLACE` function is similar to the **REPLACE** and **TRANSLATE** functions, but it allows greater precision with regular expressions.

Syntax

SQL

```
REGEXP_REPLACE( string, pattern [, replace_string [ , position [, parameters ] ] ]
```

Argument	Data Type	Description
`string`	`CHAR`	The string to search, specified as a character string. Note that SQL escape sequences for string literals can override regular expression escape sequences. For details, see **String Literals and Escape Sequences**.
`pattern`	`CHAR`	The regular expression pattern to use in the search, specified as a character string.
`replace_string`	`CHAR`	Optional. A string to replace all occurrences of the regular expression pattern in `string`. This string can include references to capture groups in the regular expression `pattern`. To reference capture groups, use a `$` followed by the group number. For example, `$1` references the first capture group, and `$2` references the second. The default value is an empty string `''`.
`position`	`INT`	Optional. A positive integer that represents the position within the string to begin the search, based on the number of characters. If this integer exceeds the number of characters in `string`, then the function returns the original `string`. The default value is `1`.
`parameters`	`CHAR`	Optional. A string containing one or more characters representing regular expression options for pattern matching. Supported options are: `c` — Perform case-sensitive matching. The Ocient System enables this type of matching by default. `i` — Perform case-insensitive matching. `p` — Interpret the pattern using the Perl Compatible Regular Expression (PCRE) dialect. For details, see **PCRE**. The Ocient System enables this interpretation by default.

Examples

Replace Text Using a Substring

The query replaces the matching substring `'#J28jB'` with the characters `'#OFqD_'`. The search begins at position `9`.

```
SQL

SELECT REGEXP_REPLACE(
    'https://docs.ocient.com/character-binary-functions#J28jB',
    '#\w+',
    '#OFqD_',
    9
    );
```

*Output: * `/character-and-binary-functions`

Replace Text Using Multiple Capture Groups

This example uses three capture groups to take an unformatted phone number and convert it into the `(XXX) XXX-XXXX` format. The `replace_string` argument references each capture group as `$1`, `$2`, and `$3`.

```
SQL

SELECT REGEXP_REPLACE(
    '5558675309',
    '(\d{3})(\d{3})(\d{4})',
    '($1) $2-$3'
    );
```

*Output: * `(555) 867-5309`

REGEXP_SUBSTR

Returns one substring from a string that matches a specified regular expression pattern.

`REGEXP_SUBSTR` is similar to the **SUBSTR** function, but it allows greater precision with regular expressions.

If the pattern produces no matches, the function returns an empty string.

Syntax

```
SQL

REGEXP_SUBSTR( string, pattern [, position [, occurrence [, parameters ] ] ] )
```

Argument	Data Type	Description
`string`	`CHAR`	The string to search, specified as a character string. Note that SQL escape sequences for string literals can override regular expression escape sequences. For details, see **String Literals and Escape Sequences**.
`pattern`	`CHAR`	The regular expression pattern to use in the search, specified as a character string.
`position`	`INT`	Optional. A positive integer that represents the position within the string to begin the search, based on the number of characters. If this integer is greater than the number of characters in `string`, then the function returns the original `string`. The default value is `1`.
`occurrence`	`INT`	Optional. A positive integer that represents the occurrence of a positive pattern match to return. If this value exceeds the number of matches, then the function returns NULL. The default value is `1`, which means the function returns the first substring that matches the regular expression pattern.
`parameters`	`CHAR`	Optional. A string containing one or more characters representing regular expression options for pattern matching. Supported options are: `c` — Perform case-sensitive matching. The Ocient System enables this type of matching by default. `i` — Perform case-insensitive matching. `e` — Extract the substring using a regular expression subpattern. This subpattern is enclosed in parentheses in the regular expression. The function uses the full regular expression pattern for the match, but it returns only the first subpattern match. If there is no subexpression in the `pattern` argument, then the `e` parameter is ignored. `p` — Interpret the pattern using the Perl Compatible

Argument	Data Type	Description
		Regular Expression (PCRE) dialect. For details, see **PCRE**. The Ocient System enables this interpretation by default.

Example

The query returns the first occurrence of the `#J28jB` string by using the regular expression pattern `'#\w+'`. The search starts at position `9`.

SQL

```
SELECT REGEXP_SUBSTR(
    'https://docs.ocient.com/character-binary-functions#J28jB',
    '#\w+',
    9,
    1
    );
```

*Output: * `#J28jB`

REPEAT

Repeats the `char` string `num` times without spaces.

Syntax

SQL

```
REPEAT(char, num)
```

Example

Repeat the `a` string five times.

SQL

```
SELECT REPEAT('a', 5);
```

*Output: * `aaaaa`

REPLACE

Replaces all occurrences of `substr_to_remove` in the character value `string` with `substr_to_replace`.

Syntax

```
SQL

REPLACE(string, substr_to_remove, substr_to_replace)
```

Argument	Data Type	Description
`string`	`CHAR`	A string to alter by replacing all instances of `substr_to_remove` with `substr_to_replace`.
`substr_to_remove`	`CHAR`	A substring to remove from `string`, replacing all instances with the `substr_to_replace` value. If `substr_to_remove` is the empty string, the system returns `string`.
`substr_to_replace`	`CHAR`	A substring to replace all instances of `substr_to_remove`.

Example

```
SQL

SELECT REPLACE('abcabcabcabc', 'ab', '$');
```

*Output: * `$c$c$c$c`

REVERSE

Reverse the input string.

Syntax

```
SQL

REVERSE(char)
```

Example

```
SQL

SELECT REVERSE('abcd');
```

*Output: * `dcba`

RIGHT

Return the number of trailing characters in the string equal to the value `integer`.

Syntax

```
SQL

RIGHT(char, integer)
```

Argument	Data Type	Description
`char`	`CHAR`	A string to evaluate to return the number of trailing characters equal to `integer`.
`integer`	`INT`	The number of characters of return from the end of the `char` string. If `integer` is negative, the function returns all but the first `integer` characters.

Example

```
SQL

SELECT RIGHT('ocient data warehouse', 8);
```

*Output: * `arehouse`

Example

```sql
SELECT RIGHT('ocient data warehouse', -3);
```

*Output: * `ent data warehouse`

RPAD

Pad the input text to the specified length with the pad string on the right side. If text is longer than length, it is truncated to `length` characters. If the argument pad is not provided, the space character is used.

Syntax

```sql
RPAD(character_value_text, integral_value_length [, character_value_pad] )
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be padded on its right side with additional characters, so that it equals the `length` value.
`length`	`INT`	The number of characters for `string` to total. If the `string` is already longer than `length`, the function truncates `string` to equal the `length` value.
`pad_character`	`CHAR`	Optional. A string of one or more characters to use to pad `string` to equal the `length` value. If not provided, `pad_character` uses a whitespace character to pad.

Example

```sql
SELECT RPAD('ocient data warehouse',30);
```

*Output: * `ocient data warehouse`

Example

```sql
SELECT RPAD('ocient data warehouse',30,'ab');
```

*Output: * `ocient data warehouseabababababa`

RSUBSTRING

Returns the substring from the right side of a string, based on a specified length.

Syntax

```sql
RSUBSTRING(string, integer)
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be evaluated to return a substring based on the `integer` value.
`integer`	`INT`	The number of characters to return from the right side of the `string` value. This value must be positive.

Example

```sql
SELECT RSUBSTRING('ocient data warehouse',14);
```

*Output: * `data warehouse`

RTRIM

Removes trailing blanks from the string value `string`.

Alternatively, you can specify a second string value `trim_character`. If you specify the `trim_character` value, the RTRIM function removes all trailing instances of the `trim_character` value from the `string`.

Syntax

```
SQL

RTRIM(string [, trim_character] )
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be trimmed on its right side.
`trim_character`	`CHAR`	Optional. A string of one or more characters to be trimmed from the right side of `string`. Each character is trimmed individually, not as a cohesive substring. If you do not specify this argument, this value defaults to removing all whitespace from the right side of `string`.

Example

```
SQL

SELECT RTRIM('ocient ');
```

*Output: * `ocient`

Example

```
SQL

SELECT RTRIM('ocientaaaaaa','a');
```

*Output: * `ocient`

SHA1

Uses the [SHA-1](**https://en.wikipedia.org/wiki/SHA-1#:~:text=In%20cryptography%2C%20SHA%2D1%20(,rendered%20as%2040%20hexadecimal%20digits**.) cryptographic hash function to convert a string into a 40-character string representing the hexadecimal value of a 160-bit checksum.

Syntax

```
SQL

SHA1(string)
```

Argument	Data Type	Description
`string`	`CHAR`	A string to convert using the SHA-1 cryptographic hash function.

Example

```
SQL

SELECT SHA1('Ocient');
```

*Output: * `0c2a9a042b9f047f875c3414e7a4f4c53efbe082`

SPACE

Returns a string of repeated spaces equal to the number value, `repeat`. You can join this to another string by using the **CONCAT** function.

Syntax

```
SQL

SPACE(repeat)
```

Example

```sql
SELECT CONCAT(SPACE(10),'end');
```

*Output: * `end`

SPLIT_PART

Split the value `string` based on the `delimiter` value. The function returns a substring from the split operation based on the `index` value (starting from 1).

Syntax

```sql
SPLIT_PART(string, delimiter, index)
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be split based on the `delimiter` value.
`delimiter`	`CHAR`	The character to use as a delimiter in `string`.
`index`	`INT`	The index of the substring to return. The first index position starts at `1`.

Example

```sql
SELECT SPLIT_PART('id|name|address','|',1);
```

*Output: * `id`

Example

```sql
SELECT SPLIT_PART('id|name|address','|',2);
```

*Output: * `name`

Example

```SQL
SELECT SPLIT_PART('id,name,address',',',3);
```

*Output: * `address`

SPLIT_TO_ARRAY

Splits a string into an array of substrings.

Syntax

```SQL
SPLIT_TO_ARRAY(string, [ delimiter ])
```

Argument	Data Type	Description
`string`	`CHAR`	A string to split into an array of substrings.
`delimiter`	`CHAR`	Optional. The delimiter that divides the string. The default value is a comma (`,`).

Example

This example specifies `|` as the delimiter to split the input string.

```SQL
SELECT SPLIT_TO_ARRAY('AB|CD|EF', '|');
```

*Output: * `['AB','CD','EF']`

STARTSWITH

Returns `true` if `string` starts with `substring` and `false` otherwise.

Syntax

```
SQL

STARTSWITH(string, substring)
```

Argument	Data Type	Description
`string`	`CHAR`	A string to search.
`substring`	`CHAR`	A substring to find at the beginning of `string`.

Example

```
SQL

SELECT STARTSWITH('ocient','o');
```

*Output: * `true`

Example

```
SQL

SELECT STARTSWITH('ocient','c');
```

*Output: * `false`

STRPOS

Equivalent to using **LOCATE** as `LOCATE(substring, string)`. Note the reversed argument order.

Syntax

```sql
STRPOS(string, substring)
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be evaluated for the first position of `substring`.
`substring`	`CHAR`	A substring to be found for its first position in `string`.

Example

```sql
SELECT STRPOS('ocient data warehouse','house');
```

*Output: * `17`

SUBSTR

Alias for **SUBSTRING**.

SUBSTRING

Alias for SUBSTR and MID.

Returns the substring of a character or binary value that starts with the position specified by the second argument and that ends with the position specified by one less than the sum of the second and third arguments. When the sum of the second and third arguments is less than two, the function returns the empty string.

Syntax

```sql
SUBSTRING(string, start_position [, length] )
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be truncated based on the `start_position` and `end_position` values.
`start_position`	`INT`	The starting position to return a substring. The first index position starts at `1`.
`length`	`INT`	Optional. The number of characters from the `start_position` to include in the returned substring. If unspecified, SUBSTRING returns all characters after the `start_position`.

Example

SQL

```
SELECT SUBSTRING('ocient data warehouse',8);
```

*Output: *`data warehouse`

Example

SQL

```
SELECT SUBSTRING('ocient data warehouse',8,4);
```

*Output: *`data`

TO_CHAR

Converts a numeric, date, or timestamp value into a `CHAR` date type.

SQL

```
TO_CHAR(value, format)
```

Argument	Data Type	Description
`value`	`DATE`, `TIMESTAMP`, or all numeric types	A numeric, date, or timestamp value to be converted into a `CHAR` date type.
`format`	`CHAR`	The format used for the `CHAR` conversion. For information on data type formats, see the **Formatting Functions** page.

Example

SQL

```
SELECT TO_CHAR(20200610132514/1000000,'9999-99-99');
```

*Output: * `2020-06-10`

Example

SQL

```
SELECT TO_CHAR(20200610132514%1000000,'99:99:99');
```

*Output: * `13:25:14`

TRANSLATE

Replaces specified characters in a provided string with a separate set of characters.

Characters specified in the `char_to_remove` set are replaced with characters in the `char_to_replace` set based on the corresponding index position.

Syntax

SQL

```
TRANSLATE(string, char_to_remove, char_to_replace)
```

Argument	Data Type	Description
`string`	`CHAR`	A string to have specific characters transposed based on the `char_to_remove` and `char_to_replace` values.
`char_to_remove`	`CHAR`	The characters to be removed from `string`. Characters specified in the `char_to_remove` set are replaced with characters in the `char_to_replace` set based on the corresponding index position. If `char_to_remove` is longer than `char_to_replace`, occurrences of the extra characters are removed.
`char_to_replace`	`CHAR`	The characters to replace the removed characters in `char_to_remove`.

Example

SQL

```
SELECT TRANSLATE('abcdef','ace','ghi');
```

*Output: * `gbhdif`

TRIM

Alias for BTRIM.

Trim leading and trailing space characters from the string.

Alternatively, you can specify a second string value `trim_char`. If a `trim_char` value is provided, the `TRIM` function removes all leading and trailing instances of the `trim_char` value from the string.

Syntax

SQL

```
TRIM(string [, trim_char] )
```

Argument	Data Type	Description
`string`	`CHAR`	A string to be trimmed of leading and trailing space characters, or any other characters specified by the `trim_char` argument.
`trim_char`	`CHAR`	Optional. If specified, this removes an alternative substring from the start or end of the `string` value.

Example

SQL

```
SELECT TRIM(' trimmed string ');
```

*Output: * `trimmed string`

Example

SQL

```
SELECT TRIM('aaaaaaaatrimmed stringaaaaaaaaa', 'a');
```

*Output: * `trimmed string`

UCASE

Alias for **UPPER**.

UPPER

Alias for UCASE.

Convert string to upper case.

Syntax

```sql
UPPER(character_value)
```

Example

```sql
SELECT UPPER('ocient');
```

*Output: * `OCIENT`

Concatenate Operator(`||`)

The `||` operator concatenates two strings. If you specify a NULL string, the result is NULL.

`||` Syntax

```sql
string1 || string2
```

Argument	Data Type	Description
`string1`	`CHAR`	A string to concatenate.
`string2`	`CHAR`	A string to concatenate with the first string.

Examples

Concatenate Two Strings

Concatenate two strings.

```sql
SELECT 'hello ' || 'world';
```

Output: `'hello world'`

Concatenate a NULL String

Concatenate two strings, one of which is NULL.

```sql
SELECT NULL || 'world';
```

Output: `NULL`

Related Links

WHERE

Formatting Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Types

The Ocient® System supports a variety of data types for SQL functions and querying. For information on supported data types, see **Data Types for SQL Functions**. Data type requirements for a specific function can be found in **SQL Reference** section.

SQL statements, such as DDL and DCL commands, use generic data types numeric and string. For details, see **Data Types for SQL Statements**.

Data Types for SQL Functions

The following data types are supported in table columns and queries.

This table provides more information on the data types supported by the Ocient System. The example value column gives SQL input examples for the respective data types.

The data types in this table apply to SQL functions.

> The default column values for the Tuple, Matrix, and Array values in a `CREATE TABLE` statement require an alternate syntax than the examples shown in this table.
>
> For examples of how to specify these default values, see **Create a Table with All Data Types**.

Name	Description	System Limits	Example Value
ARRAY	An array of the specified type in the format `TYPE[]`. The `TYPE` type can be any supported data type besides ARRAY. For example, `INT[]` creates an array of integers.	Maximum memory allocation: 512MiB	`INT[]\(1,2,3)` `[]\()` can be used for empty arrays. For example: `BIGINT[]\()`
BIGINT	8-byte signed integer	Minimum value: `-9223372036854775808` Maximum value: `9223372036854775807`	`9876543210`

Name	Description	System Limits	Example Value
BINARY(N) or HASH(N)	Fixed-length binary array with length N	Maximum memory allocation: 512MiB For HASH(N), the maximum value is N repeats of `0xFF`, and the minimum value is N repeats of `0x00`.	BINARY(4): `'0x01234567'`, BINARY(3): `'0xabcdef'`
BOOLEAN	1-byte logical Boolean value	None	`TRUE`, `FALSE`
CHARACTER(N) or CHAR(N) or CHAR or CHARACTER	Variable-length character string. Length N is only for compatibility. The Ocient System does not use this argument.	Maximum memory allocation: 512MiB	CHAR(16): `'This is a string'`
DATE	4-byte calendar date (year, month, day)	Minimum value: `0001-01-01` Maximum value: `9999-12-31`	`'2020-02-02'`, `'2000-01-01'`
DECIMAL(P,S)	Exact numerical with precision P and scale S	Minimum value: `-999999999999999999999999999999999999` Maximum value: `999999999999999999999999999999999999`	`123.45`
DOUBLE	8-byte IEEE® double precision floating-point number	Minimum value: -1.7977 x 10^308 Maximum value: 1.7977 x 10^308	`3.141592`
FUNCTION	Lambda function or user-defined function	None	`(x int, y int) -> CASE WHEN x = y THEN 0 WHEN INT(COALESCE(x, 1000)) < INT(COALESCE(y, 1000)) THEN -1 ELSE 1 END`

Name	Description	System Limits	Example Value
INT	4-byte signed integer	Minimum value: `-2147483648` Maximum value: `2147483647`	`123456789`
IPV4	4-byte Internet Protocol version 4	Minimum value: `0.0.0.0` Maximum value: `255.255.255.255`	`'127.0.0.1'`
IP	16-byte Internet Protocol version 6 (can also hold IPV4)	Minimum value: `0000:0000:0000:0000:0000:0000:0000:0000` Maximum value: `ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff`	`'0123:4567:89ab:cdef:0123:4567:89ab:cdef'`, `'1111::0'`, `'127.0.0.1'`
LINESTRING	Geometric type composed of N points	Maximum memory allocation: 512MiB	`'LINESTRING(0 0,2 0)'`
MATRIX	A one or two-dimensional mathematical matrix used for matrix calculations.	Minimum value: None Maximum value: 16,384 elements	A Matrix[2][2]: `{ {0, 0}, {0, 0} }` A Matrix [2][4]: `{ {0, 0, 0, 0}, {0, 0, 0, 0} }`
POINT	Geometric point in 2 dimensions	None	`'POINT(-87.6410 41.8841)'`
POLYGON	Geometric type composed of N closed linestrings	Maximum memory allocation: 512MiB	`'POLYGON((0 0,2 0,5 5,0 2,0 0), (0 0,1 0,2 2,0 2,0 0))'`
FLOAT	4-byte IEEE single precision floating-point number	Minimum value: -3.4028 x 10^38 Maximum value: 3.4028 x 10^38	`2.718`
SMALLINT	2-byte signed integer	Minimum value: `-32768` Maximum value: `32767`	`32767`
TIME	8-byte time of day in nanoseconds	Minimum value: `00:00:00.000000000`	`'12:34:56.012345678'`

Name	Description	System Limits	Example Value
		Maximum value: `23:59:59.999999999`	
TIMESTAMP	8-byte date and time in nanoseconds with no associated time zone.	Minimum value: `1677-09-21 00:12:43.145224192` Maximum value: `2262-04-11 23:47:16.854775807`	`'2000-01-02 12:34:45'`
TINYINT or BYTE	1-byte signed integer	Minimum value: `-128` Maximum value: `127`	`127`
TUPLE<<type1, type2, ...>>	Tuple of elements of different types.	Maximum memory allocation: 512MiB	`tuple<<INT,INT>>(1,2)` An Array of Tuples: `tuple<<INT,INT>>[] (tuple<<INT,INT>>(1,2), tuple<<INT,INT>>(3,4))`
UUID	16-byte universally unique identifier	Minimum value: `00000000-0000-0000-0000-000000000000` Maximum value: `ffffffff-ffff-ffff-ffff-ffffffffffff`	`'01234567-89ab-cdef-1357-0123456789ab'`
VARBINARY(N)	Variable-length binary array with maximum length of N	Maximum memory allocation: 512MiB	VARBINARY(6): `'0xaabbccddeeff'`
VARCHAR(N)	Variable-length character string with maximum length of N	Maximum memory allocation: 512MiB	VARCHAR(4): `'This is a variable length string'`

Data Types for SQL Statements

DDL, DCL, and General SQL Syntax statements define parameter data types as either numeric or string. These data types are simplified terms that follow these rules:

- `numeric` — Any numeric value. This value defaults to the `BIGINT` data type if it is an integer with no decimal parts. If the number includes decimal parts, it defaults to `DOUBLE`.

- `string` — A series of one or more characters. This value defaults to the `VARCHAR` data type.

Data Type Considerations

Integer literals in SQL statements are of type BIGINT unless explicitly cast to another type.

Floating point literals in SQL statements are of type DOUBLE unless explicitly cast to another type.

TRUE and FALSE can be used in SQL statements as valid BOOLEAN literals.

Floating point values must be strictly numeric. Infinity and NaN values are unsupported.

The decimal type supports a maximum precision of 31 and a scale of one less than the precision.

Interval types can only be used within an expression. If the final result of an outermost expression is an interval type, the Ocient System automatically converts it to a BIGINT and the value loses the units information.

- “yyyy” : The year as a four-digit number.
- “MM” : The month, from 01 through 12.
- “dd” : The day of the month, from 01 through 31.
- “HH” : The hour, using a 24-hour clock from 00 to 23.
- “mm” : The minute, from 00 through 59.
- “ss” : The second, from 00 through 59.
- “FFFFFFFFF” : If non-zero, the billionths of a second in a date and time value.

Data Type Compatibility

The Ocient System casts data types to the highest common type when possible in expressions to produce an output of that type. These data-type promotions occur in these hierarchies. This promotion often applies to `COALESCE`, `CONCAT`, and `||`, and other operators.

Numeric Types

```SQL
DOUBLE
FLOAT
DECIMAL
BIGINT
INT
SMALLINT
TINYINT
```

Geospatial Types

```SQL
POLYGON
LINESTRING
POINT
```

Geospatial functions might have semantic differences when you execute these functions with different types that have higher precedence than the type promotion of its arguments.

Size Limits

Variable-length columns have limits to how large the value can be.

Data types with a 124 KiB size limit:

- TUPLE<<type1, type2, ...>> : Tuple value with a VARCHAR(N) or VARBINARY(N) type

Data types with a 512 MiB size limit:

- TYPE[]: The inner values of the array are subject to the size limits of the inner type.
- LINESTRING
- POLYGON
- VARBINARY(N)

- VARCHAR(N)

> The database ignores length of N for CHAR, VARCHAR, and VARBINARY. The database uses N for SQL conformance only.

> When you query data, if the database must analyze a value that exceeds 124 KiB at query execution time, the system increases memory usage. Large computed values are non-freeable during query execution, and the system cannot offload to temporary disk. Queries that operate on computed large values with many rows can cause the system to run out of memory and result in the killing of the query.

Array

Array is a SQL type container that stores multiple elements of the same SQL type. Arrays can have zero or any number of values. For instance, an INT array can contain many integers, including NULL. Arrays in Ocient can store any SQL type: IPV4, IP, BOOLEAN, BYTE, SHORT, INT, LONG, FLOAT, DOUBLE, TIME, TIMESTAMP, CHAR, DATE, BINARY, HASH, POINT, UUID, DECIMAL, and TUPLE.

Arrays have special functions and operators that are described in the following sections:

- **Array Functions**
- **Array Operators**

Tuple

Tuple is a SQL type container that is meant to store elements of different types. Ocient supports tuples using any of the supported SQL types: IPV4, IP, BOOLEAN, BYTE, SHORT, INT, BIGINT, FLOAT, DOUBLE, TIME, TIMESTAMP, CHAR, DATE, BINARY, HASH, POINT, UUID, and DECIMAL. In Ocient, tuples can either exist as single columns, SQL array elements, nested tuples, or tuples of arrays.

Tuple functions and operators are described in the following sections:

- **Tuple Functions**
- **Tuple Operators**

Matrix

Matrix is a SQL-type container. It is a fixed-size, one or two-dimensional array (mathematical matrix) of the same fixed-size SQL data types. Currently, Matrix only supports storing doubles. Matrix dimensions must be non-zero.

Matrix functions and operators are described in the following sections:

- **Matrix Functions**
- **Matrix Operators**

Geospatial Data Types

Ocient supports three different geospatial geographies: POLYGONS, LINESTRINGS, and POINTS.

- A `POLYGON` can be constructed with a closed LINESTRING or an outer shell and an array of inner rings.
- A `LINESTRING` represents a series of POINTS connected by line segments. It can be constructed with either LINESTRINGS or POINTS.
- A `POINT` represents a point in space defined by an (x, y)/(long, lat) coordinate pair.

Polygon semantics in the Ocient System follow the **OGC standard (ISO 19125-1) for LinearRings**, where the exterior and holes can be meaningfully oriented either counter-clockwise or clockwise. To conceptualize this, imagine walking along the exterior of the polygon in the order of its defined points, so everything to your left is the “interior” of the polygon. A counter-clockwise polygon is the typical orientation where, using an example of a state boundary, the interior of the state corresponds to the interior of the polygon. If you define the state boundary using the clockwise orientation, the interior of the polygon represents everything except the state.

ⓘ If a counter-clockwise exterior polygon has holes, the holes should be clockwise-oriented.

ⓘ Ensure that both the data and polygon literals are oriented in the way you intend when you use them in queries.

© 2026 Ocient, Inc.

Geography Equivalent

Many other geospatial implementations use an object-oriented approach to geospatial data types by implementing a `GEOGRAPHY` union type that might contain any combination of these types: `POINT`, `LINESTRING`, or `POLYGON`.

Ocient supports `GEOGRAPHY` slightly differently by using `POLYGON` as an implicit union type. This container type means unclosed `POLYGON` types with no holes are equivalent to `LINESTRING` types, and that a `POLYGON` might contain a single `POINT`. This definition of a `POLYGON` differs from other implementations, where such `POLYGON` types would be considered degenerate. During loading, Ocient is able to convert incoming `GEOGRAPHY` data to the appropriate `POLYGON` equivalent. Ocient geospatial functions appropriately handle these cases as needed.

Spatial Reference Identifier (SRID)

Similar to geography union types, many other geospatial implementations define an attribute table of column identifiers and an associated SRID. The system uses the SRID lookup to define semantics of functions such as `st_area`, which uses different calculations if the specified polygon is planar, defined on an ideal sphere, or on a spheroid. The identifiers also specify units of measure and the number of variables used to define a point in the reference system. However, Ocient only supports the storage of latitude and longitude coordinates for its geography types, and uses only GCS WGS 84 (EPSG code 4326) semantics. Rather than applying a transformation like you might in other implementations to change units or model accuracy, the geospatial measurement functions in the Ocient System allow you to specify the units of measure directly, as well as whether to use a faster idealized sphere model of the Earth, or a more accurate spheroidal model.

Ensure that you set the units and spheroidal model correctly when performing critical measurements. Also, be careful when you export data from other databases to Ocient so that the incoming data is in GCS WGS 84 format. Ensure that you use the Loading and Transformation pipeline to prepend or strip SRID definitions from EWKT or EWKB-formatted data. Internally, the Ocient System has no concept of SRIDs, however it does support parsing and emitting SRID-prepended data for compatibility with external tools. When the database parses SRID-prepended data, the system ignores the SRID component and parses the data as a GCS WGS 84 defined geography.

© 2026 Ocient, Inc.

Geospatial Functions

Each geospatial geography has a number of functions and operators that can be used in SQL queries to perform analyses. Some functions apply to specific geography types. For more function and operator specifics, see **Geospatial Functions**.

Related Links

Date and Time Functions

Character and Binary Functions

Query Ocient

CREATE TABLE SQL Statement Examples

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Date and Time Functions

Basic Date and Time Functions

ADD_MONTHS

Adds the specified number of months to the date. Equivalent to using a date and adding a `MONTH(value)`.

Syntax

```
SQL

ADD_MONTHS(time, int)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function modifies by adding a specified number of months.
`int`	`INT`	The number of months to add to the `time` value.

Example

```
SQL

SELECT ADD_MONTHS('2022-11-15 04:18:00',2);
```

Output: `2023-01-15`

CENTURY

Returns the number of centuries.

Syntax

```sql
CENTURY(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes to return the number of its centuries.

Example

In this example, the current date is `2023-01-03`.

```sql
SELECT CENTURY(CURRENT_DATE());
```

Output: `21`

CURDATE

Alias for CURRENT_DATE.

CURRENT_DATE

Alias for CURDATE.

Returns the current date in the format `YYYY-MM-DD`.

Syntax

```sql
CURRENT_DATE
```

Example

```sql
SELECT CURRENT_DATE;
```

Output: `2023-01-03`

CURRENT_TIME

Returns the current time as a TIME value (e.g., `hh:mm:ss.mm`).

Syntax

```sql
CURRENT_TIME
```

Example

```sql
SELECT CURRENT_TIME;
```

Output: `19:40:04`

CURRENT_TIMESTAMP

Alias for **NOW**.

Returns the current date and time as a TIMESTAMP value (e.g., `YYYY-MM-DD hh:mm:ss.mmm`).

Syntax

```sql
CURRENT_TIMESTAMP
```

Example

```sql
SELECT CURRENT_TIMESTAMP;
```

Output: `2023-01-03 11:00:03.316674397`

DATE_PART

Alias for **EXTRACT**.

DATE_TRUNC

Returns the date or timestamp entered, truncated to the specified precision.

Syntax

```sql
DATE_TRUNC(precision, time)
```

Argument	Data Type	Description
`precision`	`CHAR`	The timestamp unit used to truncate the returned value. For the `precision` input, acceptable values are: `NANOSECOND` `MICROSECOND` `MILLISECOND` `SECOND` `MINUTE` `HOUR` `DAY` `WEEK` `MONTH` `QUARTER` `YEAR` `DECADE` `CENTURY` `MILLENNIUM`
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function truncates to the specified precision.

Argument	Data Type	Description
		The `time` input supports precision values DAY or larger.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-03 11:00:59.114058182`.

SQL

```
SELECT DATE_TRUNC('NANOSECOND', CURRENT_TIMESTAMP());
```

Output: `2023-01-03 11:01:05.280975906`

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-01 11:00:59.114058181`.

SQL

```
SELECT DATE_TRUNC('MONTH', CURRENT_TIMESTAMP());
```

Output: `2023-01-01 00:00:00.000000000`

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-01 11:00:59.114058181`.

SQL

```
SELECT DATE_TRUNC('CENTURY', CURRENT_TIMESTAMP());
```

Output: `2001-01-01 00:00:00.000000000`

DAY

Alias for **DAY_OF_MONTH**.

DAY_OF_WEEK

Returns an integer, in the range of 1 to 7, that represents the day of the week. The value 1 is Sunday, and 7 is Saturday.

Syntax

```
SQL

DAY_OF_WEEK(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for its day of the week.

Example

```
SQL

SELECT DAY_OF_WEEK(CURRENT_DATE());
```

Output: `3`

DAY_OF_YEAR

Alias for DOY.

Returns an integer in the range 1 to 366 that represents the day of the year.

Syntax

```
SQL

DAY_OF_YEAR(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for its day of year.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-03 11:00:59.114058182`.

SQL

```
SELECT DAY_OF_YEAR(CURRENT_TIMESTAMP());
```

Output: `3`

DAY_OF_MONTH

Alias for DAY.

Extracts the day-of-month portion of a timestamp or date as an integer.

Syntax

SQL

```
DAY_OF_MONTH(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for its day of the month.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-03 11:00:59.114058182`.

SQL

```
SELECT DAY(CURRENT_TIMESTAMP());
```

© 2026 Ocient, Inc.

Output: `3`

DECADE

The decade is the year divided by 10.

Syntax

```
SQL

DECADE(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for its decade.

Example

This example uses `CURRENT_DATE` set to `2023-01-03`.

```
SQL

SELECT DECADE(CURRENT_DATE());
```

Output: `202`

DOW

Alias for **DAY_OF_WEEK**.

DOY

Alias for **DAY_OF_YEAR**.

EOMONTH

Returns the last day of the month using the specified timestamp or date. If you specify both arguments, this function returns the last day of the month for the resulting timestamp or date after the system adds the specified number of months to the first argument value.

Syntax

```SQL
EOMONTH(time [, int])
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for the last day of the month.
`int`	`INT`	Optional. An integer value that represents the number of months to add to the `time` value. If unspecified, the value defaults to 0.

Examples

This example returns the last day of the month for the current timestamp.

```SQL
SELECT EOMONTH(CURRENT_TIMESTAMP(),0);
```

Output: `2023-02-27`

This example returns the last day of the month in November in 2022 by using the EOMONTH function to add one month to the last day of October in 2022.

```SQL
SELECT EOMONTH(DATE('2022-10-31'), 1);
```

Output: `2022-11-30`

EPOCH

The number of seconds after 1970-01-01 00:00:00 UTC.

Syntax

```sql
EPOCH(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for the number of seconds after 1970-01-01 00:00:00 UTC.

Example

```sql
SELECT EPOCH(CURRENT_TIMESTAMP());
```

Output: `1672743901`

EXTRACT

Extract a component from a timestamp or date.

Syntax

```sql
EXTRACT(precision FROM time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for the component to be extracted.

© 2026 Ocient, Inc.

Argument	Data Type	Description

`precision`	`CHAR`	The timestamp unit to be extracted.
		For the `precision` input, acceptable values are:
		`NANOSECOND`
		`MICROSECOND`
		`MILLISECOND`
		`SECOND`
		`MINUTE`
		`HOUR`
		`DAY`
		`WEEK`
		`MONTH`
		`QUARTER`
		`YEAR`
		`DECADE`
		`CENTURY`
		`MILLENNIUM`

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-03 11:00:59.114058182`.

SQL

```
SELECT EXTRACT(HOUR FROM CURRENT_TIMESTAMP());
```

Output: `11.0`

HOUR

Extracts the hour portion of a timestamp as an integer.

Syntax

SQL

```
HOUR(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for its hour value.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-03 11:00:59.114058182`.

SQL

```
SELECT HOUR(CURRENT_TIMESTAMP());
```

Output: `11`

ISODOW

Extracts the day of the week based on ISO 8601, which ranges from Monday (1) to Sunday (7).

Syntax

SQL

```
ISODOW(timestamp or date)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for its day-of-week value.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-01-03 11:00:59.114058182`.

SQL

```
SELECT ISODOW(CURRENT_TIMESTAMP());
```

Output: `2`

ISDATE

Returns `TRUE` if the input argument can be successfully cast to a date, `FALSE` otherwise.

Syntax

```SQL
ISDATE(char)
```

Argument	Data Type	Description
`char`	`CHAR`	A string to cast as a `DATE`.

Example

```SQL
SELECT ISDATE('2023-01-03');
```

Output: `TRUE`

MAKEDATETIME

Returns a timestamp consisting of the specified date and time.

Syntax

```SQL
MAKEDATETIME(date, time)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`date`	`CHAR`	A date string for timestamp creation.
`time`	`CHAR`	A time string for timestamp creation.

Example

```
SQL

SELECT MAKEDATETIME('2022-11-02','01:24:58');
```

Output: `2022-11-02 01:24:58.000000000`

MILLISECOND

Extracts the millisecond portion of a timestamp as an integer.

Syntax

```
SQL

MILLISECOND(timestring)
```

Argument	Data Type	Description
`timestring`	`TIMESTAMP`	A time value, specified as a timestamp or date, which the function extracts for the millisecond value.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-22 13:59:25.286`.

```
SQL

SELECT MILLISECOND(CURRENT_TIMESTAMP());
```

Output: `286`

MINUTE

Extracts the minute portion of a timestamp or date as an integer.

Syntax

```sql
MINUTE(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function extracts for the minute value.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-22 14:00:40.764`.

```sql
SELECT MINUTE(CURRENT_TIMESTAMP());
```

Output: `0`

MONTH_NAME

Returns the calendar name in English of the month for the specified date.

Syntax

```sql
MONTH_NAME(time)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes its month name.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-22 13:59:25.286`.

SQL

```
SELECT MONTH_NAME(CURRENT_TIMESTAMP());
```

Output: `February`

MONTHS_BETWEEN

Returns the difference between the two dates or timestamps in months as a `DOUBLE`. The fractional months component is based on a 31-day month. If the two dates have the same day or are both the last day of the month, this function returns a whole number instead.

Syntax

SQL

```
MONTHS_BETWEEN(time1, time2)
```

Argument	Data Type	Description
`time1`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function compares to `time2` for the difference in months.
`time2`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function compares to `time1` for the difference in months.

Example

```sql
SELECT MONTHS_BETWEEN(CURRENT_TIMESTAMP(), ADD_MONTHS(CURRENT_TIMESTAMP(),11));
```

Output: `-11`

MONTH

Extracts the month portion of a timestamp or date as an integer.

Syntax

```sql
MONTH(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes for its month value.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-22 13:59:25.286`.

```sql
SELECT MONTH(CURRENT_TIMESTAMP());
```

Output: `2`

MSECS

The seconds field, including fractional parts. This is the same as `EXTRACT(MILLISECONDS FROM time)`.

Syntax

```sql
MSECS(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date. The function multiplies the seconds part of the value by 1,000.

Example

```sql
SELECT MSECS(CURRENT_TIMESTAMP());
```

Output: `37637.648`

NANOS_TO_TIMESTAMP

Convert a number of nanoseconds into a timestamp equivalent to the duration after the epoch time, 1970-01-01 00:00:00 UTC.

Syntax

```sql
NANOS_TO_TIMESTAMP(time)
```

Argument	Data Type	Description
`ns`	`INT`	A number of nanoseconds, specified as an integer, which the function converts into a timestamp equivalent to the duration after the epoch time.

Example

```sql
SELECT NANOS_TO_TIMESTAMP(10000000);
```

Output: `1969-12-31 16:00:00.010`

NEXT_DAY

Returns the closest date after a specified date that lies on a specific day of the week.

Syntax

```sql
NEXT_DAY(date, character)
```

Argument	Data Type	Description
`date`	`DATE`, `TIMESTAMP`, or `CHAR`	A time value, specified as a date, timestamp, or string, which the function uses to return the closest date for a specific day of the week. Can be any type that can be cast to `DATE`.
`character`	`CHAR`	A string representing a day of the week. The string must match the first three characters (case insensitive) of the English name of any day of the week. If this string does not match the prefix of any day, this function returns NULL.

Example

```sql
SELECT NEXT_DAY('2023-02-27', 'sat');
```

Output: `2023-03-04`

NOW

Alias for **CURRENT_TIMESTAMP**.

QUARTER

Returns an integer between 1 and 4 representing the quarter of the year in which the specified date falls.

Syntax

```
SQL

QUARTER(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function computes to return the quarter of the year.

Example

```
SQL

SELECT QUARTER(DATE('2023-02-08 09:00:00.000'));
```

Output: `1`

ROUND

Returns the specified date or timestamp, rounded to the specified precision. The precision argument behaves like **DATE_TRUNC**. The function rounds up values to the specified precision.

Syntax

```
SQL

ROUND(time, precision)
```

Argument	Data Type	Description
`precision`	`CHAR`	The timestamp unit used to round the returned value. For the `precision` input, acceptable values are: `NANOSECOND` `MICROSECOND` `MILLISECOND` `SECOND` `MINUTE` `HOUR` `DAY` `WEEK` `MONTH` `QUARTER` `YEAR` `DECADE` `CENTURY` `MILLENNIUM`
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function rounds to the specified precision.

Example

SQL

```
SELECT ROUND(CURRENT_TIMESTAMP() ,'HOUR');
```

Output: `2023-02-08 09:00:00.000`

SECOND

Extracts the seconds portion of a timestamp as an integer.

Syntax

SQL

```
SECOND(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `TIME`	A time value, specified as a timestamp or date, which the function uses to return the seconds value.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-28 13:17:30.243`.

SQL

```
SELECT SECOND(CURRENT_TIMESTAMP());
```

Output: `30`

TIMESTAMP_TO_NANOS

Convert timestamp into nanoseconds after epoch as `BIGINT`.

Syntax

SQL

```
TIMESTAMP_TO_NANOS(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function uses to return the nanoseconds after the epoch.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-28 13:20:12.373`.

SQL

```
SELECT TIMESTAMP_TO_NANOS(CURRENT_TIMESTAMP());
```

© 2026 Ocient, Inc.

Output: `1677619212373708904`

TO_TIMESTAMP

For information on using the TO_TIMESTAMP conversion function, see the **TO_TIMESTAMP** page.

USECS

Returns the seconds part of a time value, including fractional parts, as an integer. Same as `EXTRACT(MICROSECONDS FROM time)`.

Syntax

```
SQL

USECS(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date. The function multiplies the seconds part of the value by 1,000,000.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-28 13:32:17.438`.

```
SQL

SELECT USECS(CURRENT_TIMESTAMP());
```

Output: `17438041`

WEEK

Returns the ISO-8601 week number, as an integer, of the specified timestamp or date value. The week starts on Monday, and the first week of a year contains January 4 of that

year.

For details, see the **ISO week date** definition.

Syntax

```sql
WEEK(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function uses to return the week number.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-28 13:37:37.128`.

```sql
SELECT WEEK(CURRENT_TIMESTAMP());
```

Output: `9`

YEAR

Extracts the year portion of a timestamp or date as an integer.

Syntax

```sql
YEAR(time)
```

Argument	Data Type	Description
`time`	`TIMESTAMP` or `DATE`	A time value, specified as a timestamp or date, which the function uses to extract the year.

Example

This example uses `CURRENT_TIMESTAMP` set to `2023-02-28 13:37:37.128`.

SQL

```
SELECT YEAR(CURRENT_TIMESTAMP());
```

Output: `2023`

Advanced Date and Time Functions

DATEADD

This function adds a specified number value (as a signed integer) to a specified `datepart` of an input date value, and then returns that modified value.

The data type of the returned value for this function is dynamic. The return type depends on the argument supplied for `date`. If the value for `date` is a string literal `date`, `DATEADD` returns a `datetime` value. If another valid input data type is supplied for `date`, `DATEADD` returns the same data type. `DATEADD` raises an error if the string literal seconds scale exceeds three decimal place positions (.nnn) or if the string literal contains the time zone offset part.

Syntax

SQL

```
DATEADD(datepart, number, date)
```

Argument	Data Type	Description
`datepart`	`CHAR`	The part of the date to which DATEADD adds an integer number.

Argument	Data Type	Description
		See the Datepart Values table for a list of supported arguments.
`number`	`INT`	An expression that can resolve to an `INT` that `DATEADD` adds to the specified `datepart` value of `date`. `DATEADD` accepts user-defined variable values for number. `DATEADD` truncates a specified `number` value that has a decimal fraction. In this situation, it does not round the number value.
`date`	`TIME`, `TIMESTAMP`, `DATE`	A column expression, expression, string literal, or user-defined variable. A string literal value must resolve to a date or timestamp. Use four-digit years to avoid ambiguity issues.

datepart	**Abbreviations**
`year`	yy, yyyy
`quarter`	qq, q
`month`	mm, m
`dayofyear`	dy, y
`day`	dd, d
`week`	wk, ww
`weekday`	dw, w
`hour`	hh
`minute`	mi, n
`second`	ss, s
`millisecond`	ms
`microsecond`	mcs
`nanosecond`	ns

`DATEADD` does not accept user-defined variable equivalents for the `datepart` arguments.

Examples

This example shows each result from incrementing different `datepart` values by 1.

SQL

```
SELECT 'year', DATEADD(year,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'quarter', DATEADD(quarter,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'month', DATEADD(month,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'dayofyear', DATEADD(dayofyear,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'day', DATEADD(day,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'week', DATEADD(week,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'weekday', DATEADD(weekday,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'hour', DATEADD(hour,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'minute', DATEADD(minute,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'second', DATEADD(second,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'millisecond', DATEADD(millisecond,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'microsecond', DATEADD(microsecond,1, '2007-01-01 13:10:10.1111111')
    UNION ALL
SELECT 'nanosecond', DATEADD(nanosecond,1, '2007-01-01 13:10:10.1111111');
```

Output

© 2026 Ocient, Inc.

Text

```
week                                    2007-01-08 13:10:10.111111100
millisecond                             2007-01-01 13:10:10.112111100
second                                  2007-01-01 13:10:11.111111100
day                                     2007-01-02 13:10:10.111111100
minute                                  2007-01-01 13:11:10.111111100
dayofyear                               2007-01-02 13:10:10.111111100
quarter                                 2007-04-01 13:10:10.111111100
month                                   2007-02-01 13:10:10.111111100
hour                                    2007-01-01 14:10:10.111111100
weekday                                 2007-01-02 13:10:10.111111100
year                                    2008-01-01 13:10:10.111111100
nanosecond                              2007-01-01 13:10:10.111111101
microsecond                             2007-01-01 13:10:10.111112100

Fetched 13 rows
```

In this example, each statement increments `datepart` by a number large enough to increment the next higher unit of date or time.

SQL

```
SELECT 'quarter', DATEADD(quarter,4,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'month', DATEADD(month,13,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'dayofyear', DATEADD(dayofyear,365,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'day', DATEADD(day,365,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'week', DATEADD(week,5,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'weekday', DATEADD(weekday,31,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'hour', DATEADD(hour,23,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'minute', DATEADD(minute,59,'2007-01-01 01:01:01.1111111')
    UNION ALL
SELECT 'second', DATEADD(second,59,'2007-01-01 01:01:01.1111111');
```

© 2026 Ocient, Inc.

Output

```
Text
```

```
day                                    2008-01-01 01:01:01.111111100
weekday                                2007-02-01 01:01:01.111111100
second                                 2007-01-01 01:02:00.111111100
month                                  2008-02-01 01:01:01.111111100
quarter                                2008-01-01 01:01:01.111111100
dayofyear                              2008-01-01 01:01:01.111111100
week                                   2007-02-05 01:01:01.111111100
hour                                   2007-01-02 00:01:01.111111100
minute                                 2007-01-01 02:00:01.111111100

Fetched 9 rows
```

DATEDIFF

This function returns an `INT` representing the difference between two date or time values, in a specified date or time unit.

If only a time value is assigned to a date data type variable, `DATEDIFF` sets the value of the missing `datepart` to the default value: `1900-01-01`. If only a date value is assigned to a variable of a time or date data type, `DATEDIFF` sets the value of the missing time part to the default value: `00:00:00`. If either `startdate` or `enddate` have only a time part and the other only a `datepart`, `DATEDIFF` sets the missing time and `datepart` to the default values.

If `startdate` and `enddate` have different date data types, and one has more time parts or fractional seconds precision than the other, `DATEDIFF` sets the missing parts of the other to 0.

Syntax

```
SQL
```

```
DATEDIFF(datepart, startdate, enddate)
```

Argument	Data Type	Description
`datepart`	`CHAR`	The units in which the function returns the difference between the `startdate` and `enddate`. Commonly used `datepart` units include `month` or `second`. The `datepart` value cannot be specified in a variable, or in a quoted string like `'month'`. If `datepart` has a DATE value, and `startdate` and `enddate` are both assigned a TIME value, then the function returns `0`. See the Datepart Values table for a list of supported arguments.
`startdate`	`TIME`, `TIMESTAMP`, `DATE`	A starting date or time value to determine the difference from the `enddate` value. If only a time value is assigned, `DATEDIFF` sets the value of the missing `datepart` to the default value: `1900-01-01`
`enddate`	`TIME`, `TIMESTAMP`, `DATE`	An ending date or time value to determine the difference from the `startdate` value. If only a time value is assigned, `DATEDIFF` sets the value of the missing `datepart` to the default value: `1900-01-01`

© 2026 Ocient, Inc.

datepart	Abbreviations
`year`	yy, yyyy
`quarter`	qq, q
`month`	mm, m
`dayofyear`	dy, y
`day`	dd, d
`week`	wk, ww
`weekday`	dw, w
`hour`	hh
`minute`	mi, n
`second`	ss, s
`millisecond`	ms
`microsecond`	mcs
`nanosecond`	ns

For `millisecond`, the maximum difference between `startdate` and `enddate` is 24 days, 20 hours, 31 minutes, and 23.647 seconds.

For `second`, the maximum difference is 68 years, 19 days, 3 hours, 14 minutes, and 7 seconds.

Each specific `datepart` name and abbreviations for that `datepart` name return the same value.

Example

The example shows how `DATEDIFF` calculates the difference between the two values based on the various `datepart` values.

SQL

```
SELECT 'year', DATEDIFF(year, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0
    UNION ALL
SELECT 'quarter', DATEDIFF(quarter, '2004-05-31 23:59:59.9999999', '2006-01-01 00:0
    UNION ALL
SELECT 'month', DATEDIFF(month, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00
    UNION ALL
SELECT 'dayofyear', DATEDIFF(dayofyear, '2004-05-31 23:59:59.9999999', '2006-01-01
    UNION ALL
SELECT 'day', DATEDIFF(day, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.000
    UNION ALL
SELECT 'week', DATEDIFF(week, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0
    UNION ALL
SELECT 'hour', DATEDIFF(hour, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:00.0
    UNION ALL
SELECT 'minute', DATEDIFF(minute, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:
    UNION ALL
SELECT 'second', DATEDIFF(second, '2004-05-31 23:59:59.9999999', '2006-01-01 00:00:
```

Output

Text

```
day         580
quarter     7
year        2
dayofyear   580
month       20
hour        13897
week        83
minute      833761
second      50025601
```

Related Links

Time Zone Functions

Formatting Functions

© 2026 Ocient All Rights Reserved

Formatting Functions

Formatting Considerations

- The format string can be any combination of characters and format patterns for each function specified in the **Date and Timestamp Formatting Patterns** and **Date and Timestamp Formatting Modifiers** tables.
- Patterns can be uppercase or lowercase, but not mixed case. YYYY or yyyy matches YYYY, but not YyYy.
- Any characters in the format string that are not part of a pattern match any character in that position in TO_DATE, TO_NUMBER, and TO_TIMESTAMP.
- You can escape string literals that contain format patterns with quotes. For example: `YYYY"a literal part YYYY"`

Date and Time Formatting Functions

Date and Timestamp Formatting Patterns

Pattern	Description
Y,YYY	4 digit year with comma
YYYY	4 digit year
YYY	Last 3 digits of a year. Values between 000-599 are assumed to be the year 2XXX and 600-999 are assumed to be the year 1XXX.
YY	Last 2 digits of a year. Values between 00-69 are assumed to be the year 20XX and 70-99 are assumed to be 19XX.
Y	Last digit of a year. Assumed to be year 200X.
BC, bc, AD, or ad	An era indicator BC or AD in upper or lower case
B.C., b.c., A.D., or a.d.	An era indicator B.C. or A.D. in upper or lower case
CC	Century value 1-99. Is ignored unless YY, Y, IY, or I are set
MM	Month of the year value 01-12
MONTH	Upper case English month name

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Pattern	Description
Month	English Month name with first letter capitalized
month	lower case English month name
RM	Upper case month of year in Roman numerals
rm	Lower case month of year in Roman numerals
MON	3 letter upper case abbreviation of an English month name
Mon	3 letter abbreviation of an English month name with first letter capitalized
mon	3 letter lower case abbreviation of an English month name
W	Week of month 1-5. For example: Days 15-21 are in the third week of the month.
DD	Day of month 1-31
WW	Week of year 1-53
DDD	Day of year 1-366
D	Day of week 1-7 where Sunday is 1. This value is ignored outside of to_char.
IYYY	4 digit ISO-8601 week-numbered year. See **ISO week date** for more details.
IYY	Last 3 digits of an ISO-8601 week-numbered year. Values between 000-599 are assumed to be the year 2XXX and 600-999 are assumed to be the year 1XXX.
IY	Last 2 digits of an ISO-8601 week-numbered year. Values between 00-69 are assumed to be the year 20XX and 70-99 are assumed to be 19XX.
I	Last digit of an ISO-8601 week-numbered year. Assumed to be the year 200X.
IDDD	Day of an ISO-8601 week-numbered year 001-371
IW	Week of an ISO-8601 week-numbered year 01-53
ID	Day of an ISO-8601 week 01-07 where Monday is 1
HH or HH12	Hour of day 01-12
HH24	Hour of day 0-23
AM, am, PM, or pm	Meridiem indicator AM or PM in upper or lower case
A.M., a.m., P.M., or p.m.	Meridiem indicator A.M. or P.M. in upper or lower case
SSSS	Seconds after midnight 0-86399
MI	Minute of hour 0-59

Pattern	Description
SS	Second of minute 0-59
MS	Milliseconds 000-999
US	Microseconds 000000-999999
NS	Nanoseconds 000000000-999999999
J	Julian Day (days after November 24, 4714 BC at midnight)

The following modifiers can also be applied to a format pattern.

Date and Timestamp Formatting Modifiers

Modifier	Description
FM	Has no effect on TO_DATE and TO_TIMESTAMP. Will not be considered a literal, but will not otherwise affect the behavior.
TH	Upper case ordinal number suffix. For example: `'DDTH'`
th	Lower case ordinal number suffix. For example: `'DDth'`
FX	Global prefix. By default, TO_DATE and TO_TIMESTAMP collapse all spaces to a single space. If FX is specified, spaces are not collapses. FX must be at the beginning of the format string to be applied. For example: `'FX YYYY-MM-DD'`
TM	Has no effect on TO_DATE and TO_TIMESTAMP. Will not be considered a literal, but will not otherwise affect the behavior.
SP	Has no effect on TO_DATE and TO_TIMESTAMP. Will not be considered a literal, but will not otherwise affect the behavior.

Date and Time notes

- You can use time specifiers such as HH12 in `TO_DATE`. Their format will be validated, but their values will not affect the resulting date.
- ISO 8601 formats for dates cannot be mixed with traditional formats for years, months, and days in `TO_DATE` and `TO_TIMESTAMP`
- Values for MS, US, and NS are scaled up if they do not have leading zeros. `TO_TIMESTAMP('0.3', 'S.NS')` corresponds to 300000000 NS and `TO_TIMESTAMP('0.000000003', 'S.NS')` corresponds to 3 NS.

- Conflicting information for the same pattern will cause an exception. ex: `TO_DATE('05 31', 'DD DD')`
- Conflicting information across different patterns such as DD and DDD will generally give precedence to the pattern that appears higher in the table.

TO_TIMESTAMP

Converts a character value with the specified format to a `TIMESTAMP` type.

Syntax

```
SQL

TO_TIMESTAMP(character_value, character_format)
```

Arguments	Data Type	Description
`character_value`	`CHAR`	A string of a value to be converted to a `TIMESTAMP` type. The string must match the provided `character_format`.
`character_format`	`CHAR`	The format pattern to be used to convert the `character_value`. See the **Date and Timestamp Formatting Patterns** section for more information about accepted pattern values.

Examples

```
SQL

SELECT TO_TIMESTAMP('2022/12/01', 'YYYY/MM/DD');
```

Output: `2022-12-01 00:00:00.000`

```
SQL

SELECT TO_TIMESTAMP('2023-02-28 13:43:20.403', 'YYYY/MM/DD HH24:MI:SS');
```

Output: `2023-02-28 13:43:20.000`

TO_DATE

Converts a character value with the specified format to a `DATE` type.

Syntax

```
SQL

TO_DATE(character_value, character_format)
```

Arguments	Data Type	Description
`character_value`	`CHAR`	A string of a value to be converted to a `DATE` type. The string must match the provided `character_format`.
`character_format`	`CHAR`	The format pattern to be used to convert the `character_value`. See the **Date and Timestamp Formatting Patterns** section for more information about accepted pattern values.

Example

```
SQL

SELECT TO_DATE('2022/12/01', 'YYYY/MM/DD');
```

Output: `2022-12-01`

Number Formatting Functions

Number Formatting Patterns

Pattern	Description
9	A digit.
0	A digit. In TO_CHAR, includes leading zeros.
. or D	A decimal point.
, or G	A group/thousand separator.
PR	Negative value in braces. For example: `-1 = <1>`

Pattern	Description
S or SG	A + or - sign for positive or negative, respectively.
MI	A - sign if the number is negative.
PL	A + sign if the number is positive.
V	Shifts the decimal place by the number of digits after V in the format. Cannot be mixed with `.` or `D`. For example: `TO_NUMBER('32', '9V99')` => 0.32
RN	Roman numeral. Not supported by TO_NUMBER.
EEEE	Exponent for scientific notation. Not supported by TO_NUMBER.
L	Currency symbol.

Number Formatting Notes

- You cannot mix PR with other sign indicators.
- Any character that does not match the corresponding group in the format string is ignored. `TO_NUMBER('1&2', '999')` = 12.
- If PR is present, it must be after every 0 or 9 in the format string.
- Similar to TO_DATE and TO_TIMESTAMP, the FM is not considered a literal, but will not otherwise affect the behavior of TO_NUMBER.

TO_NUMBER

Converts a character value with the specified format to a `DECIMAL` type.

Syntax

SQL

```
TO_NUMBER(character_value, character_format)
```

Arguments	Data Type	Description
`character_value`	`CHAR`	A string of a value to be converted to a `DECIMAL` type. The string must match the provided `character_format`.
`character_format`	`CHAR`	The format pattern to be used to convert the `character_value`. See the **Number Formatting**

Arguments	Data Type	Description
		Patterns section for more information about accepted pattern values.

Example

```sql
SELECT TO_NUMBER('5230.87', '9999V99');
```

Output: `5230.87`

Related Links

Date and Time Functions

Time Zone Functions

Character and Binary Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

HyperLogLog Functions

The HyperLogLog (HLL) sketch family of functions provides an approximate count of the number of unique elements in one or more columns.

HLL functionality is similar to running a query using a `COUNT(DISTINCT col)` clause and uses the same mechanisms as the **aggregate function** `APPROX_COUNT_DISTINCT`.

The tradeoff for HLL sketches is accuracy. For further explanation of this tradeoff in accuracy, see the Apache® **DataSketches documentation**. Each sketch is an approximate and compact representation of the original data, and it introduces a small margin of error. The Ocient® HLL implementation is based on the algorithm outlined in the research paper **HyperLogLog in Practice**. This implementation uses `log2k = 11` by default, which provides a percent error of approximately 5% at the 95% confidence interval.

HLL Sketch Functions in the SQL Statement

Sketches are useful on aggregated tables that require estimates of distinct counts.

This example creates a table for the new sketch columns `ip_address_sketch` and `user_id_sketch` as part of a CREATE TABLE AS SELECT statement. The new table inserts values from the `master_agg_table` table.

© 2026 Ocient, Inc.

SQL

```sql
CREATE TABLE my_sketch_table (
   col_a INT,
   col_b VARCHAR(255),
   ip_address_sketch HLL_SKETCH(11),
   user_id_sketch HLL_SKETCH(11)
 )
AS (
    SELECT
    col_a,
    col_b,
    HLL_SKETCH_CREATE(ip_address) as ip_address_sketch,
    HLL_SKETCH_CREATE(user_id) as user_id_sketch
FROM
    master_agg_table
    GROUP by 1,2);
```

After the database creates the new table, you can query for the approximate count of distinct values.

This example merges the sketch columns by using an `HLL_SKETCH_UNION` function. After the database merges the columns, the database converts sketch values to a distinct count estimate by using the `HLL_SKETCH_GET_ESTIMATE` function.

SQL

```sql
SELECT
    HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(ip_address_sketch)) AS ip_address_sket
    HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(user_id_sketch)) AS user_id_sketch
FROM
    my_sketch_table;
```

HLL Accuracy with log2k Parameter

In addition to the default `log2k = 11` implementation, you can specify a `log2k` value in the range of [10, 16] when creating an HLL Sketch.

This parameter controls the number of buckets used in the HLL algorithm, which controls the accuracy of the sketch upper bounded by `1.04 / sqrt(k)`. The tradeoff with larger

`log2k` values is that the sketch size grows exponentially. See this table for specifics on how the precision values affect size and accuracy.

Precision	Uncompressed Size	95% CI
10	1032 B	±6.50%
11 (default)	2056 B	±4.60%
12	4104 B	±3.25%
13	8200 B	±2.30%
14	16392 B	±1.63%
15	32776 B	±1.15%
16	65544 B	±0.81%

By default, `log2k` values larger than 11 are compressed on disk using ZSTD compression.

> When creating or referencing a HLL Sketch column in a `CREATE TABLE`, `CTAS` or `IAS` statement, you can use the type alias `HLL_SKETCH(log2k)` instead of the internal `HASH((2^log2k) + 8)`.
>
> This type alias is supported in `EXPORT` statements as well.

Example

In this example, a CREATE TABLE statement creates two HLL Sketch columns with specified `log2k` precision values.

- `sketch_a` has a precision value of `10`, meaning it has low precision and low storage requirements.
- `sketch_b` has a precision value of `15`, meaning it has high precision and high storage requirements.

Note that the precision values specified in both HLL_SKETCH_CREATE statements match the precision values in the CREATE TABLE columns.

SQL

```
CREATE TABLE my_sketch_table (
    sketch_a HLL_SKETCH(10),
    sketch_b HLL_SKETCH(15)
 )
AS (
    SELECT
    HLL_SKETCH_CREATE(col_a, 10) as sketch_a,
    HLL_SKETCH_CREATE(col_b, 15) as sketch_b
FROM
    master_agg_table
    GROUP by 1,2);
```

Supported HLL Sketch Functions

HLL_SKETCH_CREATE

Creates an HLL sketch from the data on a specified aggregated column. Returns a `HASH((2^log2k) + 8)` data representation of the sketch that you can store in a separate column.

> The HLL algorithm depends on the hash value of its input. When merging multiple sketches, it is important that each sketch derives from the same type in order to get accurate results. `1::BIGINT` might hash differently than `1::INT`, and thus, the HLL algorithm might not view these two hashes as referring to the same value `1`.
>
> For accurate results, users should cast numeric columns or expressions to a consistent type before they are used in a `HLL_SKETCH_CREATE` function, especially if they will be merged later on.

Syntax

SQL

```
HLL_SKETCH_CREATE(agg_col, [ log2k ] )
```

Arguments	Data Type	Description
`agg_col`	All data types are supported.	An aggregated column for use when you create a sketch.

Arguments	Data Type	Description
`log2k`	Integral literal	Optional. This parameter controls the number of buckets in the HLL Sketch, which affects the accuracy and storage of the sketch. This value must be an integer literal in the range of [10, 16]. It cannot be a reference to a column. By default, `log2k` values larger than 11 are compressed on disk using ZSTD compression. For details about this parameter, see **HLL Accuracy with log2k Parameter**.

Example

This example uses `HLL_SKETCH_CREATE` as a window aggregate to generate 100 sketches that contain one value in each sketch.

SQL

```
SELECT MOD(c1, 9) AS m,
    HLL_SKETCH_CREATE(c1, 12) OVER (PARTITION BY c1) AS sketch
FROM sys.dummy100;
```

HLL_SKETCH_UNION (aggregate function)

Merges multiple sketches in a single column into a unified sketch. All sketches must have the same precision. This function is an aggregate function and operates on a column.

Syntax

SQL

```
HLL_SKETCH_UNION(agg_sketch_col)
```

Arguments	Data Type	Description
`agg_sketch_col`	`HLL_SKETCH<log2k>`	A column that contains multiple sketches to be merged into one sketch.

Example

This example performs two merges. First, this code uses `HLL_SKETCH_CREATE` as a window aggregate to generate 100 sketches that contain one value in each sketch. The code merges the 100 sketches into nine sketches, and then the final SELECT statement merges the nine sketches into a single sketch. Also, the example uses the `HLL_SKETCH_GET_ESTIMATE` function to retrieve the estimated distinct count from the merged sketches.

SQL

```
WITH hll_sketch AS (
    SELECT MOD(c1, 9) AS m,
        HLL_SKETCH_CREATE(c1) OVER (PARTITION BY c1) AS sketch
    FROM sys.dummy100
),
hll_merged AS (
    SELECT HLL_SKETCH_UNION(hll_sketch.sketch) AS sketch
    FROM hll_sketch
    GROUP BY m
)
SELECT HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(hll_merged.sketch))
FROM hll_merged;
```

Output: `97`

HLL_SKETCH_UNION (scalar function)

Merges two sketches to a new combined sketch. This function is a scalar function and operates row-wise. The scalar function merges two sketch columns with heterogeneous precisions into a sketch with the lower of the two precisions.

Syntax

SQL

```
HLL_SKETCH_UNION(sketch_col1, sketch_col2)
```

Arguments	Data Type	Description
`sketch_col1`	`HLL_SKETCH<log2k>`	A column of HLL sketches

Arguments	Data Type	Description
`sketch_col2`	`HLL_SKETCH<log2k>`	A column of HLL sketches

Example

This example creates two sketches from the dummy table. The first uses the default precision of 11, and the second uses a precision of 14. The second sketch is offset by 1000, so there are 2000 unique values being sketched overall.

The query merges the two sketches using the two argument scalar union, which combines them into a new sketch. This unified sketch has a precision of 11, the lower of the two input precisions.

The query then decodes the sketch, which has the expected result of 2000 ± 92.

SQL

```
SELECT HLL_SKETCH_GET_ESTIMATE(
        HLL_SKETCH_UNION(
            HLL_SKETCH_CREATE(c1),
            HLL_SKETCH_CREATE((c1 + 1000)::INT, 14)
        )
    ) as sketch_col
FROM sys.dummy1000;
```

Output: `2000 ± 92`

HLL_SKETCH_GET_ESTIMATE

The `HLL_SKETCH_GET_ESTIMATE` scalar function converts a sketch into a distinct count estimate of a sketch value. Returns the distinct count estimate as a `BIGINT`.

To return a distinct count across a column of sketch values, you should first merge the sketches by using the `HLL_SKETCH_UNION` function.

Syntax

```sql
HLL_SKETCH_GET_ESTIMATE(sketch)
```

Arguments	Data Type	Description
`sketch`	`HLL_SKETCH<log2k>`	A sketch value to convert to an approximate distinct count.

Example

```sql
WITH hll_sketch AS (
    SELECT MOD(c1, 9) as m,
        HLL_SKETCH_CREATE(c1) OVER (PARTITION BY c1) AS sketch
    FROM sys.dummy100
),
hll_merged AS (
    SELECT HLL_SKETCH_UNION(hll_sketch.sketch) AS sketch
    FROM hll_sketch
    GROUP BY m
)
SELECT HLL_SKETCH_GET_ESTIMATE(HLL_SKETCH_UNION(hll_merged.sketch))
FROM hll_merged;
```

Output: `97`

HLL_SKETCH_GET_ESTIMATE_BOUND

The `HLL_SKETCH_GET_ESTIMATE_BOUND` scalar function takes a `HLL_SKETCH` column or an integral `log2k` literal value and returns the resulting bounding 95-percent confidence interval error proportion as a `DOUBLE`. If you provide an integral `log2k` value, it must be an integral literal and cannot be a column.

Syntax

```sql
HLL_SKETCH_GET_ESTIMATE_BOUND(sketch_or_integral)
```

Arguments	Data Type	Description
`sketch_or_integral`	`HLL_SKETCH<log2k>` or integral literal	A sketch value, column, or an integral literal.

Example

```sql
SELECT HLL_SKETCH_GET_ESTIMATE_BOUND(HLL_SKETCH_CREATE(c1, 14))
FROM sys.dummy100;
```

Output: `0.0163`

HLL_SKETCH_TO_STRING

The `HLL_SKETCH_TO_STRING` scalar function takes a `HLL_SKETCH` column or value and returns a string summary of the sketch.

Syntax

```sql
HLL_SKETCH_TO_STRING(sketch)
```

Arguments	Data Type	Description
`sketch`	`HLL_SKETCH<log2k>`	A sketch value or column to summarize as a string.

Example

SQL

```
SELECT HLL_SKETCH_TO_STRING(HLL_SKETCH_CREATE(c1)) AS summary
FROM sys.dummy10000;
```

Output:

Text

```
### HLL SKETCH SUMMARY:
Log2k                   : 11
Lower bound             : 9540.000000
Estimate                : 10000
Upper bound             : 10460.000000
Number of values seen   : 10000
Minimum index           : 114
Value at minimum index  : 0
Maximum index           : 1892
Value at maximum index  : 14
```

This summary contains this information:

- The `log2k` value. For more information, see **HLL Accuracy with log2k**.
- The lower, estimate, and upper bound of the sketch.
- The number of values seen in the sketch.
- The minimum and maximum bucket indexes and their values in the sketch.

The output fields in this summary are for internal purposes, except for the `log2k` value.

Bibliography

Heule, Stefan, Marc Nunkesser, and Alex Hall. “HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm.” In *Proceedings of the EDBT 2013 Conference*. Genoa, Italy, 2013.

Related Links

Aggregate Functions

Generate Tables Using sys.dummy

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Geospatial Functions

The OcientAIQ™ Unified Data Platform has OcientGeo™ that supports three different geospatial geographies: `POLYGON`, `LINESTRING`, and `POINT`.

- A `POLYGON` can be constructed with a closed LINESTRING or an outer shell and array of inner rings.
- A `LINESTRING` represents a series of points connected by line segments. It can be constructed with either `LINESTRING` or `POINT` data types.
- A `POINT` represents a point in space defined by an (x, y) or (longitude, latitude) coordinate pair.

> Each `LINESTRING` or `POLYGON` value can be up to a maximum of 512 MB in size. This means a `LINESTRING` or `POLYGON` can contain approximately 32 million point values.

These supported geospatial data types require Well-Known Text (WKT) formatting. For formatting examples, see **Data Types** or **WKT Representation of Geometry**. The OcientGeo functionality reference also contains examples with supported formatting and syntax.

Each geospatial geography has a number of functions and operators that can be used in SQL queries to perform analyses. Some functions apply to specific geography types. For operators specific to geospatial data types, refer to **Geospatial Operators**.

OcientGeo uses the authalic radius of the Earth, 6371007.1810824 meters, as the reference for many measurement calculations.

> The authalic radius reference might cause slight differences in measurements compared to other geospatial systems that use a different radius definition.

Geospatial Operators

OcientGeo supports special operators that perform operations on two different geographies. These can be used in queries to compare geographies or calculate values based on the geographies.

© 2026 Ocient, Inc.

Operator	Syntax	Purpose
intersects	`geoA && geoB`	Returns `TRUE` if the bounding box of geography A intersects with the bounding box of B.
contained	`geoA @ geoB`	Returns `TRUE` if the bounding box of geography A is contained within the bounding box of B.
equal	`geoA ~= geoB`	Returns `TRUE` if the bounding box of geography A is the same as bounding box of B.
contains	`geoA ~ geoB`	Returns `TRUE` if the bounding box of geography A contains the bounding box of B.
distance	`geoA <-> geoB`	Returns the distance in meters between the two geography parameters.
bounding box distance	`geoA <#> geoB`	Returns the distance in meters between the bounding box of the two geography parameters.
centroid distance	`geoA <<->> geoB`	Returns the distance in meters between the centroids of the two geography parameters.

Geospatial Functions Overview

OcientGeo supports many geospatial functions that operate on POINT, LINESTRING, and POLYGON values. Each function, its expected inputs, and return type are described in the following section.

Geospatial Filtering

Queries that involve geospatial filtering can make use of functions in the **Spatial Relationships** section.

To optimize these queries, OcientGeo provides a `SPATIAL` index that you can apply to columns with geospatial data types. For details about the `SPATIAL` index, see **Secondary Indexes**.

Attribute Functions

Attribute functions return descriptive information on the specified data set.

- ST_COORDDIM
- ST_DIMENSION
- ST_GEOMETRYTYPE
- ST_ISEMPTY
- ST_MEMSIZE
- ST_NDIMS
- ST_NDIMENSION
- ST_NPOINTS
- ST_NUMPOINTS
- ST_SRID
- ST_X
- ST_XMAX
- ST_XMIN
- ST_Y
- ST_YMAX
- ST_YMIN

Conversion Functions

Conversion functions transform a specified operand to a different data type.

- ST_ASBINARY
- ST_ASWKB
- ST_ASGEOJSON
- ST_ASEWKT
- ST_ASLATLONTEXT
- ST_ASTEXT
- ST_ASWKT
- ST_GEOHASH

Linestring Constructors

LINESTRING constructors use geospatial data to create a LINESTRING object.

- ST_LINEFROMTEXT
- ST_LINEFROMGEOJSON
- ST_LINEFROMWKB
- ST_LINEFROMEWKT
- ST_LINESTRING
- ST_MAKELINE

Linestring Functions

LINESTRING functions can perform alterations or access descriptive information on LINESTRING objects.

- ST_ADDPOINT
- ST_ENDPOINT
- ST_LINEINTERPOLATEPOINT
- ST_LINELOCATEPOINT
- ST_LINESUBSTRING
- ST_POINTN
- ST_REMOVEPOINT
- ST_SETPOINT
- ST_STARTPOINT

Point Constructors

POINT constructors use geospatial data to create a POINT object.

- ST_CENTROID
- ST_GEOGPOINT
- ST_MAKEPOINT

- ST_POINT
- ST_POINTFROMEWKT
- ST_POINTFROMGEOHASH
- ST_POINTFROMGEOJSON
- ST_POINTFROMTEXT
- ST_POINTFROMWKB

Polygon Constructors

POLYGON constructors use geospatial data to create a POLYGON object.

- ST_FORCECCW
- ST_MAKEPOLYGON
- ST_POLYGON
- ST_POLYGONFROMGEOJSON
- ST_POLYGONFROMWKB
- ST_POLYGONFROMEWKT
- ST_POLYGONFROMTEXT
- ST_WHOLEEARTH

Spatial Measurement

Spatial measurement functions can perform basic calculations on geospatial data, such as measuring the distance between two POINT objects or the area of a POLYGON object.

- ST_AREA
- ST_ANGLE
- ST_AZIMUTH
- ST_DISTANCE
- ST_DISTANCESPHERE
- ST_DISTANCESPHEROID
- ST_EUCLIDEANDISTANCE3D

© 2026 Ocient, Inc.

- ST_HAUSDORFFDISTANCE
- ST_LENGTH
- ST_LENGTH2D
- ST_MINIMUMDISTANCETOSURFACE
- ST_MAXDISTANCE
- ST_PERIMETER
- ST_PERIMETER2D

Spatial Operators

Spatial operators perform geometry calculations on geospatial data to return a different type of geospatial data.

- ST_BOUNDINGDIAGONAL
- ST_BUFFER
- ST_CLOSESTPOINT
- ST_CONVEXHULL
- ST_DIFFERENCEARRAY
- ST_ENVELOPE
- ST_EXPAND
- ST_EXTERIORRING
- ST_FLIPCOORDINATES
- ST_FORCE2D
- ST_INTERIORRINGN
- ST_INTERSECTALL
- ST_INTERSECTIONARRAY
- ST_LONGESTLINE
- ST_MAKEENVELOPE
- ST_MINIMUMBOUNDINGCIRCLE
- ST_MULTIDIFFERENCEARRAY
- ST_MULTIINTERSECTIONARRAY

© 2026 Ocient, Inc.

- ST_MULTISYMDIFFERENCEARRAY
- ST_MULTIUNIONARRAY
- ST_NRINGS
- ST_NUMINTERIORRING
- ST_NUMINTERIORRINGS
- ST_POINTONSURFACE
- ST_PROJECT
- ST_REDUCEPRECISION
- ST_REMOVEREPEATEDPOINTS
- ST_REVERSE
- ST_SEGMENTIZE
- ST_SHORTESTLINE
- ST_SIMPLIFY
- ST_SIMPLIFYARRAY
- ST_SNAPTOGRID
- ST_SYMDIFFERENCEARRAY
- ST_UNIONARRAY

Spatial Relationships

Spatial relationship functions use arguments to test for different types of spatial relationships.

- ST_CLUSTERDBSCAN
- ST_CONTAINS
- ST_CONTAINSPROPERLY
- ST_COVERS
- ST_COVEREDBY
- ST_CROSSES
- ST_DISJOINT
- ST_DWITHIN

- ST_EQUALS
- ST_INTERSECTS
- ST_ISCCW
- ST_ISPOLYGONCCW
- ST_ISPOLYGONCW
- ST_ISCLOSED
- ST_ISRING
- ST_ISSIMPLE
- ST_ISVALID
- ST_OVERLAPS
- ST_POINTINSIDECIRCLE
- ST_RELATE
- ST_TOUCHES
- ST_WITHIN

Spatiotemporal Measurement

Spatiotemporal measurement functions can perform basic calculations on geospatial data paired with TIMESTAMP data.

- ST_DISTANCE
- ST_MAXDISTANCE
- ST_TOTALSECONDSININTERSECTION

Spatiotemporal Operators

Spatiotemporal operators perform calculations on geospatial data using an array of TIMESTAMP arguments.

- ST_LONGESTLINE
- ST_LINEGETALLTIMESATPOINT
- ST_LINEGETPOINTATTIME

- ST_LINEGETTIMEATPOINT
- ST_INTERSECTION
- ST_SHORTESTLINE

Related Videos

At the Whiteboard with Ocient: Geospatial Analytics

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Attribute Functions

OcientGeo™ attribute functions return descriptive information on the specified data set.

ST_COORDDIM

Alias for ST_NDIMS or ST_NDIMENSION.

Returns an `INTEGER` of the coordinate dimension of the specified geography.

Syntax

```SQL
ST_COORDDIM(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	The geospatial object used to calculate the dimensions.

Example

```SQL
SELECT ST_COORDDIM(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

Output: `2`

ST_DIMENSION

Returns an `INTEGER` that represents the dimension of the specified geography. This table describes the result values of the `ST_DIMENSION` function.

© 2026 Ocient, Inc.

ST_DIMENSION Result Value	Dimension Type
`-1`	Empty geography value
`0`	`POINT`
`1`	`LINESTRING`
`2`	`POLYGON`

The `ST_DIMENSION` function always returns `0` for non-empty `POINT` objects.

For non-empty `LINESTRING` and `POLYGON` objects, the `ST_DIMENSION` function returns the dimension value equal to the greatest dimension represented by their bounding points. For example, a `LINESTRING` or `POLYGON` object that contains only a single point returns `0`. Similarly, a `POLYGON` object can return `1` if it contains insufficient points to represent a closed polygon.

Syntax

SQL

```
ST_DIMENSION(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	The geospatial object used to calculate the dimensions.

Examples

In this example, the function evaluates a `LINESTRING` with an empty `POINT` value.

SQL

```
SELECT ST_DIMENSION(ST_LINESTRING('POINT EMPTY'));
```

Output: `-1`

In this example, the function evaluates a `LINESTRING` with a single `POINT` value.

```sql
SELECT ST_DIMENSION(ST_LINESTRING('ST_POINT(0 0)'));
```

Output: `0`

This example evaluates an empty `LINESTRING`.

```sql
SELECT ST_DIMENSION(ST_LINESTRING('LINESTRING EMPTY'));
```

Output: `-1`

In this example, the function again evaluates a `LINESTRING` with a single `POINT` value.

```sql
SELECT ST_DIMENSION(ST_LINESTRING('LINESTRING(0 0)'));
```

Output: `0`

In this example, the function evaluates a `LINESTRING` with multiple `POINT` values.

```sql
SELECT ST_DIMENSION(ST_LINESTRING('LINESTRING(0 0, 1 1)'));
```

Output: `1`

This example evaluates an empty `POLYGON`.

```sql
SELECT ST_DIMENSION(ST_POLYGON('POLYGON EMPTY'));
```

Output: `-1`

© 2026 Ocient, Inc.

This example evaluates a `POLYGON` with a single `POINT`.

```sql
SELECT ST_DIMENSION(ST_POLYGON('POLYGON((1 1))'));
```

Output: `0`

This example evaluates a `POLYGON` with two `POINT` values.

```sql
SELECT ST_DIMENSION(ST_POLYGON('POLYGON((0 0, 1 1))'));
```

Output: `1`

This example evaluates a `POLYGON` with multiple `POINT` values, enough to close the `POLYGON`.

```sql
SELECT ST_DIMENSION(ST_POLYGON('POLYGON((0 0, 1 0, 1 1, 0 0))'));
```

Output: `2`

ST_GEOMETRYTYPE

Returns a `STRING` that represents the geometry of the input geospatial value. Supported values are `ST_POINT`, `ST_LINESTRING`, and `ST_POLYGON`. This function enforces the same strict dimension types as the **ST_DIMENSION** function. If `geo` is NULL, then the function returns NULL.

Syntax

```sql
ST_GEOMETRYTYPE(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	A geospatial value to be identified by its dimension type.

Example

```
SQL

SELECT ST_GEOMETRYTYPE(ST_POINT(1, 2));
```

Output: `ST_POINT`

ST_ISEMPTY

Returns `TRUE` if the specified geography value is empty, such as `'POLYGON EMPTY'`. A NULL input value returns NULL.

Syntax

```
SQL

ST_ISEMPTY(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	A geospatial value to be examined if it is empty.

Example

```
SQL

SELECT ST_ISEMPTY(ST_POLYGON('POLYGON EMPTY'));
```

Output: `TRUE`

ST_MEMSIZE

Returns an `INTEGER` representing the number of bytes in memory required to store the specified geography.

Syntax

```
SQL

ST_MEMSIZE(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	A geospatial value to be measured by its data memory.

Example

```
SQL

SELECT ST_MEMSIZE(ST_POINT(1,1));
```

Output: `16`

ST_NDIMS or ST_NDIMENSION

Alias for **ST_COORDDIM**.

ST_NPOINTS or ST_NUMPOINTS

Returns an `INTEGER` representing the number of `POINT` values in a specified geography.

If the specified value is a `POLYGON`, this function counts the number of `POINT` values in both the exterior and any holes.

Syntax

```sql
ST_NPOINTS(geo)
```

```sql
ST_NUMPOINTS(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	A geospatial value to be examined for its count of `POINT` values.

Example

```sql
SELECT ST_NPOINTS(
   ST_POLYGON(
      'POLYGON((1 2, 1 3, 1 2))'));
```

Output: `3`

ST_SRID

Returns the EPSG code of the spatial reference identifier (SRID) of the input geography. All Ocient® geographies are of type GCS WGS 84, which is identified by the value 4326. There is no way to set a different SRID.

Syntax

```sql
ST_SRID(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	A geospatial value to be analyzed for its EPSG code.

Example

```
SQL

SELECT ST_SRID(ST_MAKEPOINT(1.1, 3.11));
```

Output: `4326`

ST_X

Returns the x value, or longitude, of the specified `POINT`. The returned value is a `DOUBLE` type.

Syntax

```
SQL

ST_X(point)
```

Argument	Data Type	Description
`point`	`POINT`	A geospatial point to be analyzed for its x value, the longitude.

Example

```
SQL

SELECT ST_X(ST_POINT(3, 5));
```

Output: `3`

ST_XMAX

Returns the maximum x value of the specified geography. The returned value is a `DOUBLE` type.

Syntax

```
SQL

ST_XMAX(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	One or more geospatial points to be analyzed for the maximum x value.

Example

In this example, the `ST_XMAX` function returns `2` because it is the largest x value of the specified points.

```
SQL

SELECT ST_XMAX(
        ST_POLYGON(ST_LINESTRING('LINESTRING(2 5, 1 2, 1 4, 0 3)')));
```

Output: `2`

ST_XMIN

Returns the minimum x value of the specified geography.

Syntax

```
SQL

ST_XMIN(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	One or more geospatial points to be analyzed for the minimum x value.

Example

In this example, the `ST_XMIN` function returns `0` because it is the smallest x value of the specified points.

SQL

```
SELECT ST_XMIN(
        ST_POLYGON(ST_LINESTRING('LINESTRING(2 5, 1 2, 1 4, 0 3)')));
```

Output: `0`

ST_Y

Returns the y value, or latitude, of the specified `POINT`. The returned value is a `DOUBLE` type.

Syntax

SQL

```
ST_Y(point)
```

Argument	Data Type	Description
`point`	`POINT`	A geospatial point to be analyzed for its y value, the latitude.

Example

SQL

```
SELECT ST_Y(ST_POINT(3, 5));
```

Output: `5`

ST_YMAX

Returns the maximum y value of the specified geography.

Syntax

```
SQL

ST_YMAX(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	One or more geospatial points to be analyzed for the maximum y value.

Example

In this example, the `ST_YMAX` function returns `5` because it is the largest y value of the specified points.

```
SQL

SELECT ST_YMAX(
        ST_POLYGON(ST_LINESTRING('LINESTRING(2 5, 1 2, 1 4, 0 3)')));
```

Output: `5`

ST_YMIN

Returns the minimum y value of the specified geography.

Syntax

```
SQL

ST_YMIN(geo)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, `POLYGON`	One or more geospatial points to be analyzed for the minimum y value.

Example

In this example, the `ST_YMIN` function returns `2` because it is the smallest y value of the specified points.

SQL

```
SELECT ST_YMIN(
        ST_POLYGON(ST_LINESTRING('LINESTRING(2 5, 1 2, 1 4, 0 3)')));
```

Output: `2`

Related Links

Geospatial Data Types

Linestring Constructors

Point Constructors

Polygon Constructors

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Conversion Functions

OcientGeo™ conversion functions transform a specified operand to a different data type.

ST_ASBINARY

Alias of ST_ASWKB.

Returns the well-known binary (WKB) representation of the specified geography.

Syntax

```
SQL

ST_ASBINARY(geo [, XDR_or_NDR ] )
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	The geospatial object used to calculate the dimensions.
`XDR_or_NDR`	`CHAR`	Optional. This argument determines whether the resulting WKB is in little endian (NDR) or big endian (XDR) format. If you specify this argument, it must be either `XDR` or `NDR`. Otherwise, it defaults to `NDR`.

Example

```
SQL

SELECT ST_ASBINARY(ST_POINT(2, 2), 'NDR');
```

Output: `01010000000000000000000040000000000000040`

ST_ASGEOJSON

Returns the GeoJSON representation of the specified geography using the **IETF standards**.

```
SQL

ST_ASGEOJSON( geo [, planar_conversion ] )
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	The geospatial object used to create a GeoJSON representation. ℹ The GeoJSON specification has certain requirements for a `LINESTRING` and `POLYGON`. In keeping with this specification, a `POLYGON` instance that represents a `LINESTRING` results in a `LINESTRING` GeoJSON instance. Similarly, a `POLYGON` instance that represents a `POINT` results in a `POINT` GeoJSON instance. A `LINESTRING` instance that represents a `POINT` results in a `POINT` GeoJSON instance. Unlike the GeoJSON specification, a specified `POLYGON` does not need to follow the Right-Hand Rule. If a specified `POLYGON` contains any holes that are not valid `POLYGON` objects, then the function omits those holes in the resulting GeoJSON instance.
`planar_conversion`	`BOOLEAN`	Optional. A Boolean value that determines if the geodesic edges of the specified geography are converted to planar representation before the function applies the GeoJSON format. IETF® standards state that all GeoJSON objects are to be interpreted as explicitly planar. This argument allows preservation of the original geodesic shape. If `planar_conversion` is specified `TRUE`, points will be added such that the resulting GeoJSON represents a planar shape within 10m of the original geospatial object. Additionally, if the original geospatial object crossed the international dateline, it will be split at the point where it crosses. This will generate a GeoJSON object of either `MultiLineString` or `MultiPolygon` type, depending on the original type of the geospatial object. If you do not specify this argument, the value defaults to `FALSE`.

Examples

This example converts a `POINT` value into a GeoJSON representation.

SQL

```
SELECT ST_ASGEOJSON(ST_POINT(2, 2), false);
```

Output:

Text

```
{"type":"Point","coordinates":[2,2]}
```

This example converts an empty `POINT` value.

SQL

```
SELECT ST_ASGEOJSON(ST_POINTFROMTEXT('POINT EMPTY'), false);
```

Output:

Text

```
{"type":"Point","coordinates":[]}
```

This example converts a `LINESTRING` value into a GeoJSON representation.

SQL

```
SELECT ST_ASGEOJSON(ST_LINESTRING(ST_POINT(1.234, 2),ST_POINT(3.456, 5));
```

Output:

Text

```
{"type":"LineString","coordinates":[[1.234,2],[3.456,5]]}
```

This example again converts a `LINESTRING` value.

© 2026 Ocient, Inc.

SQL

```
SELECT ST_ASGEOJSON(ST_LINESTRING(ARRAY[ST_POINT(1.234, 2)]);
```

Output:

Text

```
{"type":"Point","coordinates":[1.234,2]}
```

This example converts an empty `LINESTRING` value into a GeoJSON representation.

SQL

```
SELECT ST_ASGEOJSON(ST_LINEFROMTEXT('LINESTRING EMPTY'), false);
```

Output:

Text

```
{"type":"LineString","coordinates":[]}
```

This example converts a `POLYGON` value into a GeoJSON representation.

SQL

```
SELECT ST_ASGEOJSON(
    ST_POLYGON(
        ARRAY [ST_POINT(1.234, 2),ST_POINT(3.456, 5),ST_POINT(5.678,         2),ST_
        ARRAY [ARRAY[ST_POINT(2,2.5),ST_POINT(2.1,2.6),ST_POINT(2.2,2.5),ST_POINT(2
        ARRAY [ST_POINT(3,2.5),ST_POINT(3.1,2.6),ST_POINT(3.2,2.5),ST_POINT(3,2.5)]
```

Output:

Text

```
{"type":"Polygon","coordinates":[[[1.234,2],[3.456,5],[5.678,2],[1.234,2]],[[2,2.5]
```

This example again converts a `POLYGON` value.

SQL

```
SELECT ST_ASGEOJSON(
    ST_POLYGON(
        ARRAY [ST_POINT(1.234, 2),ST_POINT(3.456, 5),ST_POINT(5.678, 2),ST_POINT(1.
        ARRAY [ARRAY[ST_POINT(2,2.5),ST_POINT(2.1,2.6),ST_POINT(2.2,2.5),ST_POINT(3
        ARRAY [ST_POINT(3,2.5),ST_POINT(3.2,2.5),ST_POINT(3,2.5)]]), false);
```

Output:

Text

```
{"type":"Polygon","coordinates":[[[1.234,2],[3.456,5],[5.678,2],[1.234,2]]]}
```

This example converts a two-point `POLYGON`, which results in a `LINESTRING` type in the GeoJSON representation.

SQL

```
SELECT ST_ASGEOJSON(
        ST_POLYGON(
            ST_LINESTRING(ST_POINT(1.234, 2), ST_POINT(3.456, 5))),
        false);
```

Output:

Text

```
{"type":"LineString","coordinates":[[1.234,2],[3.456,5]]}
```

This example converts a single-point `POLYGON`, which results in a `POINT` type in the GeoJSON representation.

SQL

```
SELECT ST_ASGEOJSON(ST_POLYGON(ST_POINT(1.234, 2)), false);
```

Output:

```
Text
{"type":"Point","coordinates":[1.234,2]}
```

This example converts an empty `POLYGON` value into a GeoJSON representation.

```
SQL
SELECT ST_ASGEOJSON(ST_POLYGONFROMTEXT('POLYGON EMPTY'), false);
```

Output:

```
Text
{"type":"Polygon","coordinates":[]}
```

ST_ASLATLONTEXT

Returns a string that represents geographic coordinates of a specified `POINT` in the specified format.

```
SQL
ST_ASLATLONTEXT(point, [ output_format ] )
```

Argument	Data Type	Description
`point`	`POINT`	A POINT value used to calculate geographic coordinates. If `point` is NULL, then this function returns NULL.
`output_format`	`CHAR`	Optional. A string that includes option flags to set the format and decimal precision for the output of the coordinates. Option flags must be uppercase to be recognized. Accepted option flags are: `D` — Represents coordinate degrees. If you specify a format string for `output_format`, this value is required. If `M` is omitted, this option flag sets the decimal precision. `M` — Represents minutes of latitude or longitude

Argument	Data Type	Description
		distance. If `S` is omitted, this option flag sets the decimal precision. `S` — Represents seconds of latitude or longitude distance. `C` — Represents the cardinal direction. If you specify this value, the appropriate cardinal direction is represented as N/S/E/W. If you omit this value, coordinates that are south or west are represented as negative values while east or north remain positive values. For all option flags besides `C`, you can repeat the option characters and use them with decimal points to set the format and decimal precision. The number of characters after the decimal point indicates the precision, while the number of characters before the decimal indicates the total width of the formatted number, including the decimal POINT. If the formatted number is shorter than the requested width, the system pads the value with leading spaces. Characters other than `D`, `M`, `S`, `C`, and `.` are passed through as string characters. If you do not specify `output_format` or leave it as empty, the value defaults to the format `D°M''S.SSS"C`.

Example

SQL

```
SELECT ST_ASLATLONTEXT(ST_POINT(10.2342342,-2.32498), 'D.D degrees MM.MM minutes C'
```

Output: `'2.0 degrees 19.50 minutes S 10.0 degrees 14.05 minutes E'`

ST_ASTEXT

Alias of ST_ASWKT and ST_EWKT.

Returns the well-known text (WKT) representation of the specified geography.

If you use `ST_ASEWKT`, the database prepends the resultant string with `SRID=4326;`.

SQL

```
ST_ASTEXT(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object that is to be returned as a WKT value. If you specify a `LINESTRING` or `POLYGON` that contains a single `POINT`, the function returns the WKT representation as a `POINT`. If you specify a `POLYGON` that has an exterior that is not closed, the function returns the WKT representation as a `LINESTRING`.

Example

```
SQL

SELECT ST_ASTEXT(ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 3, 1 2)')));
```

Output: `POLYGON((1 2, 1 3, 1 3, 1 2))`

ST_ASEWKT

For usage, see **ST_ASTEXT**.

ST_ASWKT

Alias of **ST_ASTEXT**.

ST_ASWKB

Alias of **ST_ASBINARY**.

ST_GEOHASH

Returns a string that represents the geohash of the input `POINT`. This function also accepts an optional second argument to specify the length of the geohash result.

If you specify any argument as NULL, then the function returns NULL.

```
SQL

ST_GEOHASH(point [, precision ] )
```

Argument	Data Type	Description
`point`	`POINT`	A `POINT` value that is to be converted into a geohash.
`precision`	`INTEGER`	Optional. A numeric value that specifies the geohash precision, which determines the length of the output string. This number must be in the range of 1 to 20. If you do not specify this value, `precision` defaults to 20.

Example

```
SQL

SELECT ST_GEOHASH(ST_POINT(-126, 48));
```

Output: `c0w3hc0w3hf1s70w3hf1s70w3`

Related Links

Geospatial Data Types

Attribute Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Linestring Constructors

OcientGeo™ LINESTRING constructors use geospatial data to create a LINESTRING object.

ST_LINEFROMTEXT

Alias for ST_LINESTRING(CHAR).

Creates a `LINESTRING` from a specified `CHAR`. The `CHAR` must be a `LINESTRING` value in WKT format.

Syntax

```
SQL

ST_LINEFROMTEXT(char)
```

Argument	Data Type	Description
`char`	`CHAR`	A character value in WKT format to be used to create a `LINESTRING`.

Example

```
SQL

SELECT ST_LINEFROMTEXT('LINESTRING(1 2)');
```

Output: `LINESTRING(1.000000 2.000000)`

ST_LINEFROMWKB

Alias for ST_LINESTRING(BINARY).

Creates a `LINESTRING` from the specified `BINARY`. The `BINARY` value must be a `LINESTRING` in WKB format.

© 2026 Ocient, Inc.

Syntax

```
SQL

ST_LINEFROMWKB(binary)
```

Argument	Data Type	Description
`binary`	`BINARY`	A binary value in WKB format to be used to create a `LINESTRING`.

Example

```
SQL

SELECT ST_LINEFROMWKB(
        BINARY(
'0x0102000000020000000000000000000f03f0000000000000040000000000000084000000000000010
        )
    );
```

Output: `LINESTRING(1 2,3 4)`

ST_LINEFROMEWKT

Creates a `LINESTRING` from the specified `CHAR`.

The `CHAR` value must be a `LINESTRING` in EWKT format.

`ST_LINEFROMEWKT` is functionally the same as `ST_LINESTRING(CHAR)`.

The input string must include an `SRID=…;` value. However, the database ignores this component as all Ocient® geography types are SRID 4326.

Syntax

```
SQL

ST_LINEFROMEWKT('SRID=value;char')
```

Argument	Data Type	Description
`char`	`CHAR`	A `LINESTRING` value in WKT format to be used to create a `LINESTRING`. To function properly, you must prefix this argument with an SRID value in the same string.
`value`	`INTEGER`	Any integer value. This is required, however, the database ignores this component and defaults to SRID 4326.

Example

SQL

```
SELECT ST_LINEFROMEWKT('SRID=4326;LINESTRING(1 2)');
```

Output: `LINESTRING(1.000000 2.000000)`

ST_LINEFROMGEOJSON

Creates a `LINESTRING` represented by the specified GeoJSON. The specified GeoJSON can represent a GeoJSON `POINT` or `LINESTRING` type.

Valid GeoJSON formats follow IETF® **standards**, and you can generate them by using **ST_ASGEOJSON**. If you specify an invalid GeoJSON, the behavior of the function is undefined.

Syntax

SQL

```
ST_LINEFROMGEOJSON(geojson [, geodesic ])
```

Argument	Data Type	Description
`geojson`	`CHAR`	A GeoJSON value that represents a `POINT` or `LINESTRING` type that is used to create a `LINESTRING`. If `geojson` is NULL, the function returns NULL.
`geodesic`	`BOOLEAN`	Optional. A `BOOLEAN` value that determines if the specified `geojson` should be converted from a planar to a

Argument	Data Type	Description
		geodesic representation. If you specify `geodesic` as `TRUE`, the function adds points to the resulting `LINESTRING` such that it remains within 10 meters of the original planar line. If unspecified, `geodesic` defaults to `FALSE`. If `geodesic` is NULL, the function returns NULL.

Examples

In this example, the function converts an empty GeoJSON value of a `LINESTRING`.

SQL

```
SELECT ST_LINEFROMGEOJSON('{"type":"LineString","coordinates":[]}',false);
```

Output: `LINESTRING EMPTY`

In this example, the function converts a GeoJSON value of a `LINESTRING`.

SQL

```
SELECT ST_LINEFROMGEOJSON(
        '{"type":"LineString","coordinates":[[1,1],[1,5],[5,5],[5,1]]}',
        false);
```

Output: `LINESTRING(1 1, 1 5, 5 5, 5 1)`

ST_LINESTRING

Alias for ST_MAKELINE.

Creates a `LINESTRING` based on the specified inputs.

ST_LINESTRING(geoArray) [#st_linestring-geoarray]

Creates a `LINESTRING` from the specified array of either `LINESTRING` or `POINT` values.

Syntax

```sql
ST_LINESTRING(geo_array)
```

Argument	Data Type	Description
`geo_array`	`ARRAY` of geographies, either `LINESTRING` or `POINT` values.	An array of `LINESTRING` or `POINT` values to be used to create a unified `LINESTRING`.

Example

```sql
SELECT ST_LINESTRING(
        ST_LINESTRING [](ST_LINESTRING(ST_POINT(1, 2), ST_POINT(3, 4))));
```

Output: `LINESTRING(1.000000 2.000000, 3.000000 4.000000)`

ST_LINESTRING(binary)

Creates a `LINESTRING` from the specified `BINARY`. The `BINARY` value must be a `LINESTRING` in WKB format.

Syntax

```sql
ST_LINESTRING(binary)
```

Argument	Data Type	Description
`binary`	`BINARY`	A `BINARY` value to be used to create a `LINESTRING`. The `BINARY` value must be a `LINESTRING` in WKB format.

Example

```sql
SELECT ST_LINESTRING(BINARY('0x0102000000020000000000000000f03f000000000000004000
```

Output: `LINESTRING(1 2,3 4)`

ST_LINESTRING(char) [#st_linestring-char]

Creates a `LINESTRING` from the specified `CHAR`. The `CHAR` value must be a `LINESTRING` in WKT format.

Syntax

```sql
ST_LINESTRING(char)
```

Argument	Data Type	Description
`char`	`CHAR`	A `CHAR` value to be used to create a `LINESTRING`. The argument must be a `LINESTRING` in WKT format.

Example

```sql
SELECT ST_LINESTRING('LINESTRING(1 2)');
```

Output: `LINESTRING(1.000000 2.000000)`

ST_LINESTRING(geo1, geo2) [#st_linestring-geo1-geo2]

Creates a `LINESTRING` consisting of two separate geographic arguments, both of which must be either a `POINT` or `LINESTRING`. You can mix geographic types, meaning one argument can be a `POINT` while the other is a `LINESTRING`.

Syntax

```sql
ST_LINESTRING(geo1, geo2)
```

Argument	Data Type	Description
`geo1`, `geo2`	`POINT` or `LINESTRING`	Two separate geographic arguments to be consolidated into a single `LINESTRING`. Inputs can be the same type or one of each.

Example

```sql
SELECT ST_LINESTRING(ST_POINT(1, 2), ST_POINT(1,3));
```

Output: `LINESTRING(1.000000 2.000000, 1.000000 3.000000)`

ST_MAKELINE

Alias for **ST_LINESTRING**.

ST_MAKELINE(geo1, geo2)

Alias for **ST_LINESTRING(geo1, geo2)**.

ST_MAKELINE(CHAR)

Alias for **ST_LINESTRING(char)**.

ST_MAKELINE(geoArray)

Alias for **ST_LINESTRING(geoArray)**.

Related Links

Geospatial Data Types

Linestring Functions

Conversion Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Linestring Functions

OcientGeo™ LINESTRING functions can perform alterations or access descriptive information on LINESTRING objects.

ST_ADDPOINT

Adds a `POINT` to the given `LINESTRING` at the specified 0-indexed location.

Syntax

SQL

```
ST_ADDPOINT(geo_linestring, geo_point_to_add [, location ] )
```

Argument	Data Type	Description
`geo_linestring`	`LINESTRING`	A linestring to be altered by adding the `geo_point_to_add` at a specified location. The function throws an error if `geo_linestring` is any data type except for `LINESTRING` or NULL.
`geo_point_to_add`	`POINT`	A point value to be added to the `geo_linestring`.
`location`	`INTEGER`	Optional. Represents a numeric 0-indexed location in the `geo_linestring` where `geo_point_to_add` is appended. If you do not specify this value, the function defaults to appending the `geo_point_to_add` argument to the end of `geo_linestring`. A location value of -1 also appends to the end. The function throws an error if the `location` argument is outside the bounds of the `geo_linestring` or if it is a value of less than -1.

Example

© 2026 Ocient, Inc.

```sql
SELECT ST_ADDPOINT(
        ST_LINESTRING('LINESTRING(1 2, 1 3)'),
        ST_POINT(1, 4),
        1);
```

Output: `LINESTRING(1.000000 2.000000, 1.000000 4.000000, 1.000000 3.000000)`

ST_ENDPOINT

Returns the endpoint of a specified `LINESTRING`. The returned value is a `POINT`.

Syntax

```sql
ST_ENDPOINT(geo)
```

Argument	Data Type	Description
`geo`	`LINESTRING`	A line to be computed to return its endpoint. If this value is a `POLYGON`, `POINT`, or is empty, then the function returns NULL. If this value is a non-geospatial type, the function throws an error.

Example

```sql
SELECT ST_ENDPOINT(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'));
```

Output: `POINT(1,4)`

ST_LINEINTERPOLATEPOINT

Returns a `POINT` along a `LINESTRING` based on a specified fraction of its total length.

Syntax

```SQL
ST_LINEINTERPOLATEPOINT(line, fraction)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A line used to compute an interpolated point.
`fraction`	`DOUBLE`	A decimal value between `0` and `1` that represents a fraction of the total length of the `line`. The function returns a point at this fraction of the length. The function throws an error if this value is not between `0` and `1`.

Example

In this example, the function computes a point halfway (0.5) along the specified line.

```SQL
SELECT ST_LINEINTERPOLATEPOINT(ST_LINESTRING('LINESTRING(1 2, 2 1)'), 0.5);
```

Output: `POINT(1.5 1.5)`

ST_LINELOCATEPOINT

Similar to **ST_LINEINTERPOLATEPOINT**, this function computes a fraction based on where a specified `POINT` is located along the length of a specified `LINESTRING`.

Syntax

```SQL
ST_LINELOCATEPOINT(line, point)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A `LINESTRING` used to compute a fraction based on the interpolated `point` value.
`point`	`POINT`	A `POINT` used to calculate a fraction based on where it is located along the `line` value. If the function cannot interpolate this value from the `line`, the function returns `0.0`.

Example

In this example, the function computes a fraction that represents where the point (1.5, 1.5) lies along the line.

SQL

```
SELECT ST_LINELOCATEPOINT(
        ST_LINESTRING('LINESTRING(1 2, 2 1)'),
        ST_POINT(1.5, 1.5));
```

Output: `0.5`

ST_LINESUBSTRING

Returns a `LINESTRING` that is a substring of a specified line that starts and ends at the specified fractions of its total length.

Syntax

SQL

```
ST_LINESUBSTRING(line, start_fraction, end_fraction)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A `LINESTRING` used to compute a substring line based on the `start_fraction` and `end_fraction` values.
`start_fraction`	`DOUBLE`	A decimal value between `0` and `1` that represents a fractional point of the total length of the `line`. The

Argument	Data Type	Description
		function uses this fractional point as the start point for the substring. The function throws an error if the `start_fraction` is greater than the `end_fraction`
`end_fraction`	`DOUBLE`	A decimal value between `0` and `1` that represents a fractional point of the total length of `line`. The function uses this fractional point as the endpoint for the substring.

Example

In this example, the function generates a subsection of the provided linestring, spanning from the `start_fraction` value (0.5) that represents the halfway point of the original linestring, to the `end_fraction` value (1.0) that represents the endpoint.

SQL

```
SELECT ST_LINESUBSTRING(ST_LINESTRING('LINESTRING(1 2, 2 1)'), 0.5, 1.0);
```

Output: `LINESTRING(1.50 1.50, 2.0 1.0)`

ST_POINTN

Returns the `POINT` value at a specified index of the specified `LINESTRING`. If out of bounds, the function returns NULL.

Syntax

SQL

```
ST_POINTN(geo, index)
```

Argument	Data Type	Description
`geo`	`LINESTRING`	A `LINESTRING` that is used to return one of its point values. If you specify a `POINT` or `POLYGON` value, this function returns NULL.

Argument	Data Type	Description
`index`	`INTEGER`	An index location of one of the point values in the `geo` value. This is one-based, meaning the first index location starts at `1`. Negative values count backward from the end of the linestring. In other words, an index value of `-1` returns the last point value.

Example

SQL

```
SELECT ST_POINTN(
        ST_LINESTRING(
            ST_POINT [](ST_POINT(1, 2), ST_POINT(1, 3), ST_POINT(1, 4))),
        3);
```

Output: `ST_POINT(1,4)`

ST_REMOVEPOINT

Removes a `POINT` value at a specified index from the specified line.

Syntax

SQL

```
ST_REMOVEPOINT(geo, index)
```

Argument	Data Type	Description
`geo`	`LINESTRING`	A `LINESTRING` that is to be altered by removing one of its point values.
`index`	`INTEGER`	An index of one of the point values in the `geo` value. This is zero-based, meaning the first index location starts at `0`, and the last index is `length(geo)-1`.

Example

```sql
SELECT ST_REMOVEPOINT(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'), 1);
```

Output: `LINESTRING(1.0 2.0, 1.0 4.0)`

ST_SETPOINT

Replaces a `POINT` value in a specified `LINESTRING` at a specified index. The function returns the altered `LINESTRING` with the replaced point.

Syntax

```sql
ST_SETPOINT(geo, index, geo_point_to_replace)
```

Argument	Data Type	Description
`geo`	`LINESTRING`	A `LINESTRING` that is to be altered with a new point value.
`index`	`INTEGER`	An index location of one of the point values in the `geo` linestring. This is zero-based, meaning the first index location starts at `0`. Negative values count backward from the end of the `geo` value. In other words, an index value of `-1` returns the last point value.
`geo_point_to_replace`	`POINT`	A `POINT` value to replace the value at the specified `index` location. If this value is not a `POINT` type or NULL, the function throws an error.

Example

In this example, the function replaces the point value located at the `-2` index location, meaning it is the second from the last in the sequence.

```sql
SELECT ST_SETPOINT(
        ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'),
        -2,
        ST_POINT(1, 5));
```

Output: `LINESTRING(1.000000 2.000000, 1.000000 5.000000, 1.000000 4.000000)`

ST_STARTPOINT

Returns the starting `POINT` value of the line. If this value is empty, the function returns NULL. If you specify a `POLYGON`, `POINT`, or NULL, then the function returns NULL. If you specify a non-geography value, the function throws an error.

Syntax

```sql
ST_STARTPOINT(geo)
```

Argument	Data Type	Description
`geo`	`LINESTRING`	A line to be computed to return its starting point. If this value is a `POLYGON`, `POINT`, or is empty, then the function returns NULL. If this value is a non-geospatial type, the function throws an error.

Example

```sql
SELECT ST_STARTPOINT(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 4)'));
```

Output: `ST_POINT(1,2)`

Related Links

Geospatial Data Types

Linestring Constructors

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Point Constructors

OcientGeo™ POINT constructors use geospatial data to create a POINT object.

ST_CENTROID

Returns a `POINT` that is the geographic center of mass of a specified geospatial object.

The geographic center of mass is calculated by taking the average of all points on a three-dimensional sphere, projecting the resultant point onto the sphere, and converting it back to latitude and longitude coordinates.

Syntax

```
SQL

ST_CENTROID( geo [, use_spheroid ] )
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object that the function analyzes to compute its geometric center.
`use_spheroid`	`BOOLEAN`	Optional. If this argument is `TRUE`, the function computes the centroid using a spheroid model instead of a spherical model. If you do not specify this value, the value defaults to `FALSE`.

Example

```
SQL

SELECT ST_CENTROID(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

Output: `POINT(1.66 1.33)`

ST_GEOGPOINT

Alias for `ST_POLYGON(ST_LINESTRING(ARRAY(ST_POINT(lon, lat))))`.

Creates and returns a `POLYGON` geography with a single point, defined by the longitude and latitude specified for the function.

Syntax

```
SQL

ST_GEOGPOINT(lon, lat)
```

Argument	Data Type	Description
`lon`	`DOUBLE`	A longitude coordinate to be used to create a `POLYGON` with a single point.
`lat`	`DOUBLE`	A latitude coordinate to be used to create a `POLYGON` with a single point.

Example

```
SQL

SELECT ST_GEOGPOINT(41.882548,-87.63922);
```

Output: `POLYGON((41.882548 -87.63922))`

ST_MAKEPOINT

Alias for **ST_POINT**.

ST_POINT

Creates a `POINT` from the specified input arguments.

ST_POINT(binary) [#st_point-binary]

Alias for ST_POINTFROMWKB.

Creates a `POINT` from the specified `BINARY` value, which must represent a point in WKB format.

Syntax

```
SQL

ST_POINT(binary)
```

Argument	Data Type	Description
`binary`	`BINARY`	A binary value in WKB format that represents a `POINT`.

Example

```
SQL

SELECT ST_POINT(BINARY('0x0101000000000000000000f03f0000000000000040'));
```

Output: `POINT(1 2)`

ST_POINT(char) [#st_point-char]

Creates a `POINT` from a `CHAR` value that represents a point in WKT format.

Syntax

```
SQL

ST_POINT(char)
```

Argument	Data Type	Description
`char`	`CHAR`	A `CHAR` that represents a `POINT` in WKT format.

Example

```sql
SELECT ST_POINT('POINT(1 2)');
```

Output: `POINT(1 2)`

ST_POINT(lon, lat)

Creates a `POINT` defined with the values provided for the longitude and latitude coordinates.

Syntax

```sql
ST_POINT(lon, lat)
```

Argument	Data Type	Description
`lon`	`DOUBLE`	A `DOUBLE` value that represents a longitude coordinate.
`lat`	`DOUBLE`	A `DOUBLE` value that represents a latitude coordinate.

Example

```sql
SELECT ST_POINT(87.6386, 41.8843);
```

Output: `POINT(87.6386 41.8843)`

ST_POINTFROMEWKT

Alias for **ST_POINT**.

Creates a `POINT` using an EWKT-formatted `CHAR` as an input argument.

Syntax

```
SQL

ST_POINTFROMEWKT(srid_char)
```

Argument	Data Type	Description
`srid_char`	`CHAR`	A string that includes a Spatial Reference Identifier (SRID) value and an EWKT-formatted value that represents a `POINT`. Separate the values in the string by a semicolon using this format: `'SRID = value; point_value'` The input string must include the `SRID = value;` component. However, the database ignores this component as all Ocient® geography types are `SRID = 4326`.

Example

```
SQL

SELECT ST_POINTFROMEWKT('SRID=4326;POINT(1 2)');
```

Output: `POINT(1 2)`

ST_POINTFROMGEOJSON

Creates a `POINT` represented by the specified GeoJSON value as an input argument. A second optional `BOOLEAN` argument determines if the specified GeoJSON should be converted from a planar to a geodesic representation.

If you specify a NULL value for either argument, the function returns NULL.

Syntax

```
SQL

ST_POINTFROMGEOJSON(geojson [, geodesic ] )
```

Argument	Data Type	Description
`geo_json`	`STRING`	A string of a GeoJSON. Valid GeoJSON formats follow IETF® **standards**, which you can generate using the **ST_ASGEOJSON** function. If you specify an invalid GeoJSON, the behavior of the function is undefined. The function returns NULL if any argument is NULL.
`geodesic`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, the specified GeoJSON is converted from a planar to a geodesic representation. This does not affect POINT geospatial objects. The default value is `FALSE`.

Examples

In this example, the function captures a `POINT` from an empty GeoJSON representation.

SQL

```
SELECT ST_POINTFROMGEOJSON('{"type":"Point","coordinates":[]}',false);
```

Output: `POINT EMPTY`

In this example, the function captures a `POINT` from a GeoJSON representation.

SQL

```
SELECT ST_POINTFROMGEOJSON('{"type":"Point","coordinates":[1,1]}',false);
```

Output: `POINT(1 1)`

ST_POINTFROMGEOHASH

Creates a `POINT` represented by the specified geohash.

Syntax

```
SQL

ST_POINTFROMGEOHASH(geohash)
```

Argument	Data Type	Description
`geohash`	`STRING`	A string of a valid geohash. Valid geohashes contain only the characters `0123456789bcdefghjkmnpqrstuvwxyz`. If you specify an invalid geohash, the function throws an `INVALID_ARGUMENT` error. The function returns NULL if this argument is NULL.

Examples

```
SQL

SELECT ST_POINTFROMGEOHASH('c0w3hf1s70w3hf1s70w3');
```

Output: `POINT(-126.00000000000003 48.00000000000007)`

```
SQL

SELECT ST_POINTFROMGEOHASH('kzn21');
```

Output: `POINT(42.60498046875 -5.60302734375)`

ST_POINTFROMTEXT

Alias for **ST_POINT(char)**.

ST_POINTFROMWKB

Alias for **ST_POINT(binary)**.

Related Links

Geospatial Data Types

Conversion Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Polygon Constructors

OcientGeo™ POLYGON constructors use geospatial data to create a POLYGON object.

ST_FORCECCW

Creates a standardized polygon from an existing one. Ocient® defines standardized as the exterior being counterclockwise (CCW) and all holes being clockwise (CW) oriented.

Syntax

```SQL
ST_FORCECCW(polygon)
```

Argument	Data Type	Description
`polygon`	`POLYGON`	A `POLYGON` to be converted on its exterior counterclockwise and clockwise for any interior holes.

Example

```SQL
SELECT ST_FORCECCW(ST_POLYGON('POLYGON((0 0,0 1,1 1,1 0,0 0))'));
```

Output: `POLYGON((0 0,1 0,1 1,0 1,0 0))`

ST_MAKEPOLYGON

Alias for **ST_POLYGON**.

ST_POLYGON

Creates a `POLYGON`.

ST_POLYGON(geo)

Creates a `POLYGON` from the specified `POINT`, `POINT` array, `LINESTRING`, or `POLYGON` geography.

Syntax

```
SQL

ST_POLYGON(geo)
```

Argument	Data Type	Description
`geo`	`POLYGON`	A `POINT`, `POINT` array, `LINESTRING`, or `POLYGON` value to be converted to a `POLYGON` object.

Example

```
SQL

SELECT ST_POLYGON(
        ST_LINESTRING(
            ST_POINT []( 
                ST_POINT(1, 2),
                ST_POINT(1, 3),
                ST_POINT(1, 4),
                ST_POINT(1, 2))));
```

Output: `POLYGON((1 2,1 3,1 4,1 2))`

ST_POLYGON(char) [#st_polygon-char]

Alias for ST_MAKEPOLYGON(char).

Creates a `POLYGON` from the specified `CHAR` representing a `POLYGON` in WKT format.

Syntax

```sql
ST_POLYGON(char)
```

Argument	Data Type	Description
`char`	`CHAR` that represents a `POLYGON` in WKT format.	The `CHAR` argument must represent a `POLYGON` in WKT format.

Example

```sql
SELECT ST_POLYGON('POLYGON((1 2,1 3,1 4,1 2))');
```

Output: `POLYGON((1 2,1 3,1 4,1 2))`

ST_POLYGON(binary) [#st_polygon-binary]

Alias for ST_POLYGONFROMWKB(binary).

Creates a `POLYGON` from the specified `BINARY`. The `BINARY` value must represent a `POLYGON` in WKB format.

Syntax

```sql
ST_POLYGON(binary)
```

Argument	Data Type	Description
`binary`	`BINARY` that represents a `POLYGON` in WKB format.	A `BINARY` argument to be converted to a `POLYGON`. The `BINARY` value must represent a `POLYGON` in WKB format.

Example

SQL

```
SELECT ST_POLYGON(BINARY('0x0103000000010000000500000000000000000000000000000000000000
```

Output: `POLYGON((0 0,0 1,1 1,1 0,0 0))`

ST_POLYGON(outer_ring, inner_rings)

Alias for ST_MAKEPOLYGON(outerRing, innerRings).

Creates a `POLYGON` from the specified outer ring (that must be either a `POINT`, `POINT` array, or `LINESTRING`) and the array of interior rings (closed lines).

The interior of the `POLYGON` is on the left when traveling around the exterior ring. The interior of a hole is on the right when traveling around a hole.

Syntax

SQL

```
ST_POLYGON(outer_ring, inner_rings)
```

Argument	Data Type	Description
`outer_ring`	`POINT`, `POINT` array, or `LINESTRING`	A `POINT`, `POINT` array, or `LINESTRING` that represents the outer ring of the `POLYGON` object.
`inner_rings`	`ARRAY` of `POINT` arrays, or `ARRAY` of `LINESTRING` values	An `ARRAY` of `POINT` arrays or `LINESTRING` values that represents the inner rings or holes in the `POLYGON` object.

Example

SQL

```
SELECT ST_POLYGON(
        ST_LINESTRING(ST_POINT [](ST_POINT(1, 2))),
        ST_LINESTRING [](ST_LINESTRING(ST_POINT [](ST_POINT(1, 2)))));
```

Output: `POLYGON((1.000000 2.000000), (1.000000 2.000000))`

ST_POLYGONFROMWKB

Alias for **ST_POLYGON(binary)**.

ST_POLYGONFROMEWKT

Alias for the **ST_POLYGON** constructor.

Creates a `POLYGON` using an EWKT-formatted `CHAR` as an input argument.

Syntax

```
SQL

ST_POLYGONFROMEWKT(char)
```

Argument	Data Type	Description
`char`	`CHAR`	A string that includes an SRID value and an EWKT-formatted value that represents a `POLYGON`. Separate the values in the string by a semicolon in this format: `'SRID = value; polygon_value'` The input string must include the `SRID = value;` component. However, the database ignores this component as all Ocient geography types are `SRID = 4326`.

Example

```
SQL

SELECT ST_POLYGONFROMEWKT('SRID=4326;POLYGON((1 2,1 3,1 4,1 2))');
```

Output: `POLYGON((1 2,1 3,1 4,1 2))`

ST_POLYGONFROMGEOJSON

Creates a `POLYGON` represented by the specified GeoJSON value as an input argument.

Syntax

```
SQL

ST_POLYGONFROMGEOJSON(geojson [, geodesic ] )
```

Argument	Data Type	Description
`geo_json`	`STRING`	A string of a GeoJSON. Valid GeoJSON formats follow IETF® **standards**, which you can generate using the **ST_ASGEOJSON** function. If you specify an invalid GeoJSON, the behavior of the function is undefined. The function returns NULL if any argument is a NULL value.
`geodesic`	`BOOLEAN`	Optional. If you specify `geodesic` as `TRUE`, the function adds points to the resulting `POLYGON` such that it remains within 10 meters of the original planar polygon. If you do not specify this argument, the value defaults to `FALSE`.

Examples

In this example, the function produces an empty `POLYGON` value from the GeoJSON representation.

```
SQL

SELECT ST_POLYGONFROMGEOJSON('{"type":"Polygon","coordinates":[]}',false);
```

Output: `POLYGON EMPTY`

In this example, the function produces a `POLYGON` value from the GeoJSON representation.

```sql
SELECT ST_POLYGONFROMGEOJSON(
        '{"type":"Polygon","coordinates":[[[1,1],[1,5],[5,5],[5,1]]]}',
        false);
```

Output: `POLYGON((1 1, 1 5, 5 5, 5 1))`

ST_POLYGONFROMTEXT

Alias for **ST_POLYGON(char)**.

ST_WHOLEEARTH

Returns the database internal representation of the whole earth polygon. The **ST_ENVELOPE** function returns this `POLYGON` value, with the envelope set as the entire earth.

GIS specifications do not describe a canonical polygon for the whole earth, but this value serves as the Ocient-canonical polygon. The whole earth polygon is also equivalent to `ST_POLYGON('POLYGON((0 0,0 0,0 0,0 0')))`, which can have a separate meaning in other GIS implementations.

Syntax

```sql
ST_WHOLEEARTH()
```

Example

```sql
SELECT ST_WHOLEEARTH();
```

Output: `POLYGON((0.0 0.0, 0.0 0.0, 0.0 0.0, 0.0 0.0))`

Related Links

Geospatial Data Types

Conversion Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Spatial Measurement

OcientGeo™ spatial measurement functions can perform basic calculations on geospatial data, such as measuring the distance between two POINT objects or the area of a POLYGON object.

ST_AREA

Returns the area of the specified geospatial object in the specified unit of measurement.

Syntax

```SQL
ST_AREA(geo [, units] [, use_spheroid ] )
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be measured in the specified measurement unit.
`units`	`STRING`	Optional. Supported units are `'FEET'`, `'KILOMETERS'`, `'MILES'`, and `'METERS'`. The default value is `'METERS'`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this value to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

```SQL
SELECT ST_AREA(
        ST_POLYGON(ST_LINESTRING('LINESTRING(2 3, 2 4, 4 2, 2 3)')),
        'MILES');
```

© 2026 Ocient, Inc.

Output: `196932665.2369081`

ST_ANGLE

Calculates the angle between two lines. The lines are represented by three `POINT` values, including one point for each line (a, b), and a third point (c) representing the vertex where they intersect. The function calculates the spherical angle as formed by the two great circles defined by (a, c) and (b, c).

ST_ANGLE returns the value in radians, in the range `[0, 2pi]`, excluding the value `2pi`.

If either a or b is the same or exactly antipodal to c, the function returns NULL. If any argument is NULL, the function returns NULL.

Syntax

SQL

```
ST_ANGLE(endpoint_a, vertex_c, endpoint_b)
```

Argument	Data Type	Description
`endpoint_a`	`POINT`	A `POINT` that represents an endpoint for a line between its position and the `vertex_c` value.
`vertex_c`	`POINT`	A `POINT` that represents the intersection between two lines. The lines comprise the endpoints between `endpoint_a` and `vertex_c`, and `endpoint_b` and `vertex_c`.
`endpoint_b`	`POINT`	A `POINT` that represents an endpoint for a line between its position and the `vertex_c` value.

Example

SQL

```
SELECT ST_ANGLE(ST_POINT(1,0), ST_POINT(0,0), ST_POINT(1,1));
```

Output: `5.49778714378213`

ST_AZIMUTH

Returns the azimuth of the line from `point1` to `point2` in radians.

Returns NULL if either of the points are empty. Returns `0` if both of the points are the same.

Syntax

```
SQL

ST_AZIMUTH(point1, point2)
```

Argument	Data Type	Description
`point1`	`POINT`	An endpoint for a line to be calculated for its azimuth value.
`point2`	`POINT`	An endpoint for a line to be calculated for its azimuth value.

Example

```
SQL

SELECT ST_AZIMUTH(ST_POINT(1,1), ST_POINT(2,2));
```

Output: `0.785398163397448`

ST_DISTANCE

Returns the minimum distance between the specified arguments.

The `ST_DISTANCE` function returns the minimum distance in the specified unit of measurement. The `use_spheroid` optional argument enables the computation to use a spheroid model rather than a spherical one.

Syntax

```
SQL

ST_DISTANCE(geo1, geo2 [, units] [, use_spheroid] )
```

Argument	Data Type	Description
`geo1`	`POINT` , `LINESTRING` , or `POLYGON`	A geospatial object to be measured for its minimum distance to `geo2` .
`geo2`	`POINT` , `LINESTRING` , or `POLYGON`	A geospatial object to be measured for its minimum distance to `geo1` .
`units`	`STRING`	Optional. Supported units include `'FEET'` , `'KILOMETERS'` , `'MILES'` , and `'METERS'` . If you do not specify this argument, the value defaults to `'METERS'` .
`use_spheroid`	`BOOLEAN`	Optional. If you set this value to `TRUE` , this function uses a spheroid model instead of a spherical model. Defaults to `FALSE` .

Example

```
SQL

SELECT ST_DISTANCE(
        ST_LINESTRING('LINESTRING(2 3, 2 4)'),
        ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5)'),
        'METERS',
        true);
```

Output: `111050.10339481`

To use ST_DISTANCE as a spatiotemporal function to calculate the minimum simultaneous distance between two linestring-timestamp array pairs, see the **ST_DISTANCE** function in **Spatiotemporal Measurement**.

ST_DISTANCESPHERE

Returns the minimum distance between the specified arguments using a spherical computation.

This function is identical to ST_DISTANCE with the `use_spheroid` argument set to `FALSE`.

SQL

```
ST_DISTANCESPHERE(geo1, geo2 [, units] )
```

See **ST_DISTANCE** for usage example.

ST_DISTANCESPHEROID

Returns the minimum distance between the specified arguments using a spheroid computation.

This function is identical to ST_DISTANCE with the `use_spheroid` argument set to `TRUE`.

SQL

```
ST_DISTANCESPHEROID(geo1, geo2 [, units] )
```

See **ST_DISTANCE** for usage example.

ST_EUCLIDEANDISTANCE3D

Returns the minimum distance between two geospatial points with altitude values. The distance represents the length of a Euclidean line between the points.

Syntax

SQL

```
ST_EUCLIDEANDISTANCE3D(point1, altitude1, point2, altitude2 [, units] )
```

Argument	Data Type	Description
`point1`	`POINT`	A geospatial point that represents a Euclidean line with `point2`.
`altitude1`	`DOUBLE`	Distance from the surface of the Earth for `point1`. This argument uses meters, unless you specify a different unit of measurement with the `units` argument.
`point2`	`POINT`	A geospatial point that represents a Euclidean line with `point1`.
`altitude2`	`DOUBLE`	Distance from the surface of the Earth for `point2`. This argument uses meters, unless you specify a different unit of measurement with the `units` argument.
`units`	`VARCHAR`	Optional. This argument specifies the unit of measurement for the distance input arguments and the function output. Supported units are `'FEET'`, `'KILOMETERS'`, `'MILES'`, and `'METERS'`. The default value is `'METERS'`.

Example

SQL

```
SELECT ST_EUCLIDEANDISTANCE3D(
    ST_POINT(0,0),1000.0,
    ST_POINT(0,1),1000.0,
    'METERS');
```

Output: `111211.09372321835`

ST_HAUSDORFFDISTANCE

Returns the Hausdorff distance between two geographies in a specified measurement. If any argument is NULL or empty, the function returns NULL.

The two geographies can be `POINT`, `LINESTRING`, or `POLYGON` values. The system promotes these geographies to filled polygons. Behavior is undefined if the resulting polygons are not convex.

Syntax

SQL

```
ST_HAUSDORFFDISTANCE(geo1, geo2 [, units] [, use_spheroid] )
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be measured for its Hausdorff distance to `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be measured for its Hausdorff distance to `geo1`.
`units`	`STRING`	Optional. Supported units are `'FEET'`, `'KILOMETERS'`, `'MILES'`, and `'METERS'`. The default value is `'METERS'`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

SQL

```
SELECT ST_HAUSDORFFDISTANCE(
        ST_POLYGON('POLYGON((1 1, 2 2, 1 2, 1 1))'),
        ST_POLYGON('POLYGON((0 0, 2 1, 2 0, 0 0))'),
        'METERS',
        TRUE);
```

Output: `156899.5136831999`

ST_LENGTH

Returns the length of the specified line in the specified measurement unit.

Returns `0` for all geography types except for `LINESTRING`. To return the total length of a `POLYGON`, use **ST_PERIMETER**.

Syntax

SQL

```
ST_LENGTH(geo [, units] [, use_spheroid] )
```

Argument	Data Type	Description
`geo`	`LINESTRING`	A line that is to be measured for its length. The function accepts `POINT` and `POLYGON` data types, but returns `0`.
`units`	`STRING`	Optional. Supported units are `'FEET'`, `'KILOMETERS'`, `'MILES'`, and `'METERS'`. The default value is `'METERS'`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

SQL

```
SELECT ST_LENGTH(
        ST_LINESTRING('LINESTRING(1 2, 1 4)'),
        'METERS',
        false);
```

Output: `222390.10395618092`

ST_LENGTH2D

Alias for **ST_LENGTH**.

ST_MINIMUMDISTANCETOSURFACE

Calculates the shortest distance between any point along a Euclidean line segment in three-dimensional space and the surface of the Earth. Two input points and their altitudes define this line segment. The line can go above or below the surface of the Earth.

The function takes the difference of both points from the center of the Earth to calculate the distance between the nearest point of the line and the surface of the Earth.

The function returns the distance in meters or another unit of measurement if specified. If the returned distance is positive, the shortest distance is above the surface. If the result is negative, the shortest distance is below the surface.

Syntax

SQL

```
ST_MINIMUMDISTANCETOSURFACE(point1, altitude1, point2, altitude2 [, units] )
```

Argument	Data Type	Description
`point1`	`POINT`	A geospatial point that represents a Euclidean line with `point2`.
`altitude1`	`DOUBLE`	Distance from the surface of the Earth for `point1`. This argument uses meters, unless you specify a different unit of measurement with the `units` argument.
`point2`	`POINT`	A geospatial point that represents a Euclidean line with `point1`.
`altitude2`	`DOUBLE`	Distance from the surface of the Earth for `point2`. This argument uses meters, unless you specify a different unit of measurement with the `units` argument.
`units`	`VARCHAR`	Optional. This argument specifies the unit of measurement for the distance input arguments and the function output. Supported units are `'FEET'`, `'KILOMETERS'`, `'MILES'`, and `'METERS'`. The default value is `'METERS'`.

Examples

This example takes a line from two points and calculates the distance of the nearest point to the surface of the Earth.

© 2026 Ocient, Inc.

SQL

```
SELECT ST_MINIMUMDISTANCETOSURFACE(
    ST_POINT(0,0),1000.0,
    ST_POINT(0,1),1000.0,
    'METERS');
```

Output: `757.3734914669767`

In this example, the output is negative, meaning the smallest distance is actually below the surface of the Earth.

SQL

```
SELECT ST_MINIMUMDISTANCETOSURFACE(
    ST_POINT(0,0),1000.0,
    ST_POINT(0,5),1000.0,
    'METERS');
```

Output: `-5064.738915202208`

ST_MAXDISTANCE

Returns the maximum distance between the specified arguments.

ST_MAXDISTANCE returns the maximum distance in the specified unit of measurement. The `use_spheroid` optional argument enables the computation to use a spheroid model rather than a spherical one.

Syntax

SQL

```
ST_MAXDISTANCE(geo1, geo2 [, units] [, use_spheroid] )
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be measured for its maximum distance to `geo2`.

Argument	Data Type	Description
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be measured for its maximum distance to `geo1`.
`units`	`STRING`	Optional. Supported units are `'FEET'`, `'KILOMETERS'`, `'MILES'`, and `'METERS'`. The default value is `'METERS'`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

SQL

```
SELECT ST_MAXDISTANCE(
        ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 7, 1 2)')),
        ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5, 1 2)')),
        'METERS',
        false);
```

Output: `555975.2598904524`

> To use ST_MAXDISTANCE as a spatiotemporal function to calculate the maximum simultaneous distance between two linestring-timestamp array pairs, see the **ST_MAXDISTANCE** function in **Spatiotemporal Measurement**.

ST_PERIMETER

Returns the length of the exterior (outer ring) of the `POLYGON` in the specified unit of measurement.

Returns `0` for all geography types except for `POLYGON`. To return the length of a `LINESTRING`, use **ST_LENGTH**.

Syntax

```
SQL

ST_PERIMETER(geo, [, units] [, use_spheroid] )
```

Argument	Data Type	Description
`geo`	`POLYGON`	A `POLYGON` that is to be measured for its total perimeter length. The function accepts `POINT` and `LINESTRING` data types, but returns `0`.
`units`	`STRING`	Optional. Supported units are `'FEET'`, `'KILOMETERS'`, `'MILES'`, and `'METERS'`. The default value is `'METERS'`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

```
SQL

SELECT ST_PERIMETER(
        ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 3, 1 2)')),
        'METERS',
        false);
```

Output: `222390.1039561809`

ST_PERIMETER2D

Alias for **ST_PERIMETER**.

Related Links

Geospatial Data Types

Spatial Operators

Spatial Relationships

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Spatial Operators

OcientGeo™ spatial noperators perform geometry calculations on geospatial data to return a different type of geospatial data.

ST_BOUNDINGDIAGONAL

Returns the diagonal `LINESTRING` from the minimum point to the maximum point of the bounding box that **ST_ENVELOPE** returns.

Syntax

```
SQL

ST_BOUNDINGDIAGONAL(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be measured for the diagonal length of its bounding box.

Example

```
SQL

SELECT ST_BOUNDINGDIAGONAL(
        ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 3, 1 3, 1 2)'))
    );
```

Output: `LINESTRING(1 2,1 3)`

ST_BUFFER

Returns a geography that contains all points where the distance from the geography is less than or equal to the specified distance. The function simplifies the geometry before

© 2026 Ocient, Inc.

buffering unless you specify the `simplification_tolerance` argument with a zero or negative value.

The output of the function is an array of polygons. For POINT and LINESTRING data types, the returned array contains one polygon only.

Syntax

SQL

```
ST_BUFFER(geo, radius [, num_segments_per_quarter_circle]
          [, simplification_tolerance] [, maximum_radius_error]
          [, join_style] [, miter_limit]
          [, endcap_style] [, side_style])
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object, specified as a point, linestring, or polygon, which the function buffers.
`radius`	`DOUBLE`	Radius, specified as a double, that indicates the distance to buffer the geospatial object in meters. Negative values cause polygons to shrink. Also, negative values cause the result to be empty for points and linestrings.
`num_segments_per_quarter_circle`	`BIGINT`	Optional. The number of segments to use per quarter circle when parts of the buffer are round, specified as a BIGINT. Only use if the `join_style` or `endcap_style` value is set to `ROUND`. Defaults to `8`.
`simplification_tolerance`	`DOUBLE`	Optional. Value that the function passes to the ST_SIMPLIFYARRAY function prior to buffering. If you specify a zero or negative value, the function skips simplification of the geometry. The unit is meters. Defaults to `500`.
`maximum_radius_error`	`DOUBLE`	Optional. Maximum radius error, specified as a double. The returned buffer might be slightly larger than the `radius` value by this amount. This value is required because arcs of great circles cannot be parallel. Use

Argument	Data Type	Description
		this argument for linestrings and polygons only. Defaults to `10.0`.
`join_style`	`STRING`	Optional. The type of join to use at the corners where segments of the geometry meet, specified as a string. Supported styles are `ROUND`, `BEVEL`, and `MITER`. Use this argument for linestrings and polygons only. Defaults to `ROUND`.
`miter_limit`	`DOUBLE`	Optional. Miter limit, specified as a double. Use this argument only if the `join_style` value is set to `MITER`. A mitered corner is cut off at a distance of `miter_ratio * radius` from the corner point of the geometry. Use this argument for linestrings and polygons only. Defaults to `1.0`.
`endcap_style`	`STRING`	Optional. The type of end cap for use at the ends of linestrings, specified as a string. Supported styles are `ROUND`, `SQUARE`, and `FLAT`. Use this argument for linestrings only. Defaults to `ROUND`.
`side_style`	`STRING`	Optional. The side, specified as a string, that denotes the side for buffering the geometry. Supported styles are `BOTH`, `LEFT`, and `RIGHT`. Use this argument for linestrings only. Defaults to `BOTH`.

Example

SQL

```
SELECT ST_BUFFER(
        ST_POLYGON('POLYGON((0 0, 1 0, 1 1, 0 0))'), 2000, 8, 500.0, 10.0,
        'MITER', 2.0);
```

Output:

`[POLYGON((2.202700241764562E-18 -0.017986411844962975, 1.0179863625702694 -0.0179863616840571, 1.0179892379654352 1.031486286416526, 1.009550207729886 1.0349816974070565, -0.03498171178141835 -0.009548649645081482, -0.03148633724650228 -0.017986322870467 57, 2.202700241764562E-18 -0.017986411844962975))]`

ST_CLOSESTPOINT

Returns the two-dimensional `POINT` of one specified geospatial object that is closest to a second specified geospatial object. This function works with any geospatial type.

If either geospatial argument is empty, the function returns NULL.

Syntax

SQL

```
ST_CLOSESTPOINT(geo1, geo2 [, use_spheroid] )
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be computed to determine its closest `POINT` value relative to `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A second geospatial object to be compared to `geo1` to find the closest `POINT` value of `geo1`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this value to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

SQL

```
SELECT ST_CLOSESTPOINT(
        ST_POLYGON(ST_LINESTRING('LINESTRING(2 3, 2 4, 2 4, 2 3)')),
        ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5, 1 2)')));
```

Output: `POINT(2,3)`

ST_CONVEXHULL

Returns a `POLYGON` representing the convex hull of the input geometry.

The convex hull is the smallest convex geometry that encloses the input geometry. If the `geo` argument is empty, the function returns an empty `POLYGON`.

Syntax

```
SQL

ST_CONVEXHULL(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be computed for a `POLYGON` of its convex hull. If `geo` is a `POINT` or a single-point `LINESTRING` or `POLYGON`, the function returns a `POLYGON` containing only that point value. If `geo` is NULL, the function returns NULL.

Example

```
SQL

SELECT ST_CONVEXHULL(
        ST_POLYGON('POLYGON((0 0,1 0,1 1,0.5 0.5,0.3 0.3,0 1,0 0))'));
```

Output: `POLYGON((0.0 0.0, 1.0 0.0, 1.0 1.0, 0.0 1.0, 0.0 0.0))`

ST_DIFFERENCEARRAY

Returns an array containing any geospatial objects that are present in the first specified geospatial argument that are not found in the second geospatial argument.

Syntax

```
SQL

ST_DIFFERENCEARRAY(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2` to find any unique geospatial values.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A second geospatial object used to compare with `geo1` to find unique values. The function captures only objects that are unique to `geo1`, and not those unique to `geo2`.

Example

In this example, the function returns `LINESTRING(2.0 2.0, 3.0 3.0)` because that portion of the first `LINESTRING` argument is not present in the second argument.

SQL

```
SELECT ST_DIFFERENCEARRAY(
        ST_LINESTRING('LINESTRING(1 1, 2 2, 3 3)'),
        ST_LINESTRING('LINESTRING(1 1, 2 2)'));
```

Output: `[LINESTRING(2.0 2.0, 3.0 3.0)]`

ST_ENVELOPE

Returns a `POLYGON` that represents the minimum bounding box for the specified geography. If the specified argument is empty, the function returns an empty `POLYGON`.

A single `POINT` argument results in a single `POINT`.

Syntax

SQL

```
ST_ENVELOPE(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be computed for the vertices of its bounding box.

Example

SQL

```
SELECT ST_ENVELOPE(
        ST_POLYGON(ST_LINESTRING('LINESTRING(10 10, 0 0)')));
```

Output: `POLYGON((0 0,10 0,10 10,0 10,0 0))`

ST_EXPAND

Returns the bounding box of a specified geospatial value, which is expanded by a specified length. The unit of length is in meters.

Syntax

SQL

```
ST_EXPAND(geo1, units_to_expand)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be computed for the vertices of its bounding box, with the added `units_to_expand`.
`units_to_expand`	`DOUBLE`	An additional distance to expand the bounding box of `geo`. Specify this argument only in meters.

Example

```sql
ST_EXPAND(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'), 1.0);
```

Output: `POLYGON((0.99 0.99, 1.33 0.99, 1.66 0.99, 2.01 0.99, 2.00 2.00, 1.66 2.00, 1.33 2.00, 0.99 2.00, 0.99 0.99))`

ST_EXTERIORRING

Returns a `LINESTRING` that represents the exterior ring of a provided `POLYGON` value. If you specify a geography value that is not a `POLYGON`, the function returns NULL.

Syntax

```sql
ST_EXTERIORRING(polygon)
```

Argument	Data Type	Description
`polygon`	`POLYGON`	A `POLYGON` object to be computed for a `LINESTRING` of its perimeter.

Example

```sql
SELECT ST_EXTERIORRING(ST_POLYGON('POLYGON((1 1, 3 1, 2 2, 1 1))'));
```

Output: `LINESTRING(1.0 1.0, 3.0 1.0, 2.0 2.0, 1.0 1.0)`

ST_FLIPCOORDINATES

Returns a new geographic object with the X and Y coordinates switched using the specified argument. The return type is the same as the specified type.

Syntax

```
SQL

ST_FLIPCOORDINATES(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object for switching its X and Y coordinates.

Example

```
SQL

ST_FLIPCOORDINATES(ST_POINT('POINT(1 3)'));
```

Output: `POINT(3.0 1.0)`

ST_FORCE2D

Convert a geographic object into a two-dimensional geography. The output representations only have the X and Y coordinates.

The returned type is the same as the specified type.

Syntax

```
SQL

ST_FORCE2D(geo1)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object to be converted to a two-dimensional object.

Example

```sql
SELECT ST_FORCE2D(ST_POINT('POINT(1 1)'));
```

Output: `POINT(1.0 1.0)`

ST_INTERIORRINGN

Returns a `LINESTRING` representing the interior ring of the specified `POLYGON`, which is specified by its index. (1-indexed)

Returns NULL if the index is out of range, or if you specify a non-POLYGON geography.

Syntax

```sql
ST_INTERIORRINGN(polygon, index)
```

Argument	Data Type	Description
`polygon`	`POLYGON`	A `POLYGON` object to be computed for its interior ring.
`index`	`INTEGER`	An index that represents the specific interior ring of the `polygon` value to be captured. Interior-ring indexes are 1-based, meaning the first object has an index of `1`.

Example

```sql
SELECT ST_INTERIORRINGN(
        ST_POLYGON(
            ST_LINESTRING(ST_POINT [](ST_POINT(1, 2))),
            ST_LINESTRING [](ST_LINESTRING(ST_POINT [](ST_POINT(1, 2))))),
        1);
```

Output: `LINESTRING(1 2)`

ST_INTERSECTALL

Returns the intersection of all geographies in the specified array. All geographies in the array must be the same type. The ST_INTERSECTALL function ignores NULL values in the array. The output geography has the same type as the input geometries.

Syntax

```
SQL

ST_INTERSECTALL(array)
```

Argument	Data Type	Description
`array`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON`	An array of geographic objects, which all must be the same type.

Example

```
SQL

SELECT ST_INTERSECTALL(
        ST_POINT [](ST_POINT(0, 0), ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: `POINT[]`

ST_INTERSECTIONARRAY

Returns a geography that represents the point-set intersection of two geographies.

If the two geographies are of different types, the function returns an array of the smallest geography. For example, if `geo1` is a `POINT`, and `geo2` is a `POLYGON`, the function returns an array of `POINT` values.

Syntax

```sql
ST_INTERSECTIONARRAY(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object to be compared to `geo2` to find a set of intersections.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object to be compared to `geo1` to find a set of intersections.

Example

This example compares two polygons to find any intersections. Because both objects are `POLYGON` types, the result is also a `POLYGON` containing all intersection points.

```sql
SELECT ST_INTERSECTIONARRAY(
    ST_POLYGON(ST_LINESTRING(ST_POINT(1,2),ST_POINT(3,4))),
    ST_POLYGON(ST_LINESTRING(ST_POINT(1,2),ST_POINT(4,5))));
```

Output: `[POLYGON((1, 2))]`

ST_LONGESTLINE

Returns the longest `LINESTRING` between two specified geospatial arguments.

Syntax

```sql
ST_LONGESTLINE(geo1, geo2, use_spheroid)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for the determination of the longest `LINESTRING` relative to `geo2`.

Argument	Data Type	Description
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for the determination of the longest `LINESTRING` relative to `geo1`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this value to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

SQL

```
SELECT ST_LONGESTLINE(
    ST_MAKELINE('LINESTRING(0 3, 1 3)'),
    ST_MAKELINE('LINESTRING(2 3, 0 3)'));
```

Output: `LINESTRING(0.0 3.0, 2.0 3.0)`

> To use ST_LONGESTLINE as a spatiotemporal function to calculate the longest line between two points at a concurrent time, see the **ST_LONGESTLINE** function.

ST_MAKEENVELOPE

Alias for `ST_ENVELOPE(ST_LINESTRING(ST_POINT(xmin, ymin), ST_POINT(xmax, ymax)))`.

Returns a `POLYGON` with vertices representing the minimum bounding box for the specified coordinates.

Syntax

SQL

```
ST_MAKEENVELOPE(xmin, ymin, xmax, ymax)
```

Argument	Data Type	Description
`xmin`	`DOUBLE`	The minimum x value for the bounding box.

Argument	Data Type	Description
`ymin`	`DOUBLE`	The minimum y value for the bounding box.
`xmax`	`DOUBLE`	The maximum x value for the bounding box.
`ymax`	`DOUBLE`	The maximum y value for the bounding box.

Example

```sql
SELECT ST_MAKEENVELOPE(0, 0, 10, 10);
```

Output: `POLYGON((0.0 0.0, 10.0 0.0, 10.0 10.0, 0.0 10.0, 0.0 0.0))`

ST_MINIMUMBOUNDINGCIRCLE

Returns the smallest circle `POLYGON` that contains the specified geographic object.

The function requires an additional argument to define the number of segments for inclusion in each quarter of the returned circle.

Syntax

```sql
ST_MINIMUMBOUNDINGCIRCLE(geo, num_segs_per_qt_circ)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for the determination of the smallest possible circle `POLYGON` to encompass it.
`num_segs_per_qt_circ`	`INTEGER`	The number of segments to be included in each quarter of the returned circle `POLYGON`.

Example

```sql
SELECT ST_MINIMUMBOUNDINGCIRCLE(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'), 3);
```

Output: `POLYGON((1.49 2.91, 0.27 0.79, 2.72 0.79, 1.49 2.91))`

ST_MULTIDIFFERENCEARRAY

Returns an array of geographies representing the parts of the union of the geographies in the first array that do not intersect with the union of geographies in the second array.

This operation is not symmetric, meaning `ST_MULTIDIFFERENCEARRAY(array1, array2)` does not produce the same results as `ST_MULTIDIFFERENCEARRAY(array2, array1)` unless both arrays are identical.

The function ignores NULL types. For types besides NULL, all geographies in both arrays must be of the same type.

The output geographies are the same type as the input geographies.

Syntax

```sql
ST_MULTIDIFFERENCEARRAY(array1, array2)
```

Argument	Data Type	Description
`array1`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be compared to the geographic objects in `array2`. The function returns any values in this array that do not intersect with `array2`.
`array2`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be compared to the geographic objects in `array1`. The function returns only values that do not intersect in `array1`.

Example

```sql
ST_MULTIDIFFERENCEARRAY(
    POINT[](ST_POINT(0, 0), ST_POINT(1, 1)),
    POINT[](ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: `['POINT(0.0 0.0)']`

ST_MULTIINTERSECTIONARRAY

When you specify two arrays of geospatial objects, this function returns an array of any intersections. The function treats each specified array as a union of all its geospatial values.

All geographies in both arrays must be of the same type, not counting NULL values, which the function ignores.

The output geographies are the same type as the input geographies.

Syntax

```sql
ST_MULTIINTERSECTIONARRAY(array1, array2)
```

Argument	Data Type	Description
`array1`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be compared to the geographic objects in `array2` for any intersections. The data types in this array must match the data types in `array2` unless they are NULL.
`array2`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be compared to the geographic objects in `array1` for any intersections. The data types in this array must match the data types in `array1` unless they are NULL.

Example

```sql
SELECT ST_MULTIINTERSECTIONARRAY(
    POINT[](ST_POINT(0, 0), ST_POINT(1, 1)),
    POINT[](ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: `['POINT(1.0 1.0)']`

ST_MULTISYMDIFFERENCEARRAY

Alias for `ST_MULTIUNIONARRAY(array1, array2) - ST_MULTIINTERSECTIONARRAY(array1, array2)`.

When you specify two arrays of geospatial objects, this function returns an array of any geospatial values that do not intersect. The function treats each specified array as a union of all its geospatial values.

All geographies in both arrays must be of the same type, not counting NULL values, which the function ignores.

The output geographies are the same type as the input geographies.

Syntax

```sql
ST_MULTISYMDIFFERENCEARRAY(array1, array2)
```

Argument	Data Type	Description
`array1`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be compared to the geographic objects in `array2` for any values that do not intersect. The data types in this array must match the data types in `array2` unless they are NULL.
`array2`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be compared to the geographic objects in `array1` for any values that do not intersect.

Argument	Data Type	Description
		The data types in this array must match the data types in `array1` unless they are NULL.

Example

SQL

```
SELECT ST_MULTISYMDIFFERENCEARRAY(
    POINT[](ST_POINT(0, 0), ST_POINT(1, 1)),
    POINT[](ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: `['POINT(0.0 0.0)','POINT(2.0 2.0)']`

ST_MULTIUNIONARRAY

When you specify two arrays of geospatial objects, this function returns one array that represents a union of all geographies in both arrays.

All geographies in both arrays must be the same type. The function ignores NULL values in the arrays. The output geographies are the same type as the input geographies.

Syntax

SQL

```
ST_MULTIUNIONARRAY(array1, array2)
```

Argument	Data Type	Description
`array1`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be combined with `array2`. The data types in this array must match the data types in `array2` unless they are NULL.
`array2`	`ARRAY` of `POINT`, `LINESTRING`, or `POLYGON` values.	An array of geographic objects to be combined with `array1`. The data types in this array must match the data types in `array1` unless they are NULL.

Example

```SQL
SELECT ST_MULTIUNIONARRAY(
    POINT[](ST_POINT(0, 0), ST_POINT(1, 1)),
    POINT[](ST_POINT(1, 1), ST_POINT(2, 2)));
```

Output: `['POINT(1.0 1.0)','POINT(2.0 2.0)','POINT(0.0 0.0)']`

ST_NRINGS

Returns the number of rings of the specified `POLYGON`, including both interior and exterior rings. If you specify a geography object that is not a `POLYGON`, this function returns `0`.

Syntax

```SQL
ST_NRINGS(geo)
```

Argument	Data Type	Description
`geo`	`POLYGON`	An object to be computed to return its total number of rings.

Example

```SQL
SELECT ST_NRINGS(
        ST_POLYGON('POLYGON((1.0 2.0, 1.0 4.0, 1.0 5.0, 1.0 2.0))'));
```

Output: `1`

ST_NUMINTERIORRINGS or ST_NUMINTERIORRING

Returns the number of interior rings of the specified `POLYGON`. If you specify a geography object that is not a `POLYGON`, this function returns `0`.

Syntax

SQL

```
ST_NUMINTERIORRINGS(polygon)
```

SQL

```
ST_NUMINTERIORRING(polygon)
```

Argument	Data Type	Description
`polygon`	`POLYGON`	An object to be computed to return its total number of interior rings.

Example

SQL

```
SELECT ST_NUMINTERIORRINGS(
        ST_POLYGON(
            ST_LINESTRING(ST_POINT [](ST_POINT(1, 2))),
            ST_LINESTRING [](ST_LINESTRING(ST_POINT [](ST_POINT(1, 2))))));
```

Output: `1`

ST_POINTONSURFACE

Returns a `POINT` guaranteed to intersect the specified geospatial object.

Syntax

SQL

```
ST_POINTONSURFACE(geo1)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	An object to be computed for an intersection `POINT` value.

Example

SQL

```
SELECT ST_POINTONSURFACE(ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

Output: `POINT(1.0 1.0)`

ST_PROJECT

Returns a `POINT` by projecting a distance and an azimuth value from the specified starting `POINT` value. The distance must be in meters, and azimuth must be in radians.

Syntax

SQL

```
ST_PROJECT(point, distance, azimuth)
```

Argument	Data Type	Description
`point`	`POINT`	A starting point for locating a returned destination by using the `distance` and `azimuth` values.
`distance`	`DOUBLE`	A distance value in meters.
`azimuth`	`DOUBLE`	An azimuth value in radians.

Example

SQL

```
SELECT ST_PROJECT(ST_POINT(1, 3), 100, 100);
```

Output: `POINT(0.99 3.00)`

ST_REDUCEPRECISION

Returns a new geospatial object with all `POINT` values rounded to the specified decimal precision.

Syntax

```
SQL

ST_REDUCEPRECISION(geo, precision)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be rounded to the specified `precision` value.
`precision`	`DOUBLE`	A decimal value that defines the level of precision for returned values. For example, a precision of `0.01` returns values rounded to the hundredth decimal. Do not use digits other than `0` and `1` to represent the precision.

Example

```
SQL

SELECT ST_REDUCEPRECISION(ST_POINT('POINT(1.21 1.34)'), 0.1);
```

*Output***:** `POINT(1.2 1.3)`

ST_REMOVEREPEATEDPOINTS

Returns a new geospatial object with no repeated `POINT` values. The returned type is the same as the specified type.

Syntax

```sql
ST_REMOVEREPEATEDPOINTS(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for the removal of any duplicate `POINT` values.

Example

```sql
SELECT ST_REMOVEREPEATEDPOINTS(ST_LINESTRING('LINESTRING(1 2, 1 2, 2 1, 1 0)'));
```

Output: `LINESTRING(1.0 2.0, 2.0 1.0, 1.0 0.0)`

ST_REVERSE

Returns a new geospatial object with the vertexes reversed. The returned type is the same as the specified type.

Syntax

```sql
ST_REVERSE(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for vertex reversal.

Example

```sql
SELECT ST_REVERSE(ST_LINESTRING('LINESTRING(1 2,1 10)'));
```

Output: `LINESTRING(1.0 10.0, 1.0 2.0)`

ST_SEGMENTIZE

Returns a geospatial object that the function modifies to have no segment longer than the specified `max_segment_length` in meters.

Syntax

SQL

```
ST_SEGMENTIZE(geo, max_segment_length)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for segment modification.
`max_segment_length`	`DOUBLE`	The maximum length of a segment to modify the specified `geo` object. This value must be in METERS.

Example

SQL

```
SELECT ST_SEGMENTIZE(
        ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'),
        150000.0);
```

Output: `POLYGON((1.0 1.0, 2.0 1.0, 2.0 2.0, 1.49 1.50, 1.0 1.0))`

ST_SHORTESTLINE

Returns the shortest `LINESTRING` between two specified geospatial arguments.

Syntax

```sql
ST_SHORTESTLINE(geo1, geo2, use_spheroid)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for the determination of the shortest `LINESTRING` relative to `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for the determination of the shortest `LINESTRING` relative to `geo1`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

```sql
SELECT ST_SHORTESTLINE(
    ST_MAKELINE('LINESTRING(4 3, 10 3)'),
    ST_MAKELINE('LINESTRING(0 1, 0 3)'));
```

Output: `LINESTRING(0.0 3.0, 4.0 3.0)`

> To use ST_SHORTESTLINE as a spatiotemporal operator to calculate the shortest line between two points at a concurrent time, see the **ST_SHORTESTLINE** function in the Spatiotemporal section.

ST_SIMPLIFY

Returns a simplified version of the specified geography, which is either a `POINT` or `LINESTRING`.

The function simplifies the input geography by replacing nearly straight edges that contain multiple points with a single straight edge. The input geography does not change by more than the specified tolerance, which you specify as a `DOUBLE` in meters.

Syntax

```sql
ST_SIMPLIFY(geo, tolerance)
```

Argument	Data Type	Description
`geo`	`POINT` or `LINESTRING`	A geospatial object to be simplified.
`tolerance`	`DOUBLE`	A value that represents the maximum number of meters to simplify in the specified geography.

Example

```sql
SELECT ST_SIMPLIFY(
    ST_LINESTRING('LINESTRING(0 0, 0.05 0, 0.1 0, 0.15 0, 2 0)'), 1);
```

Output: `LINESTRING(0 0, 2 0)`

ST_SIMPLIFYARRAY

Returns a `POLYGON` array that represents a simplified version of the specified geography, which is either a `POINT`, `LINESTRING`, or `POLYGON`.

The function simplifies the input geography by replacing nearly straight edges that contain multiple points with a single straight edge. The input geography does not change by more than the specified tolerance, which you specify as a `DOUBLE` in meters.

Syntax

```sql
ST_SIMPLIFYARRAY(geo, tolerance)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be simplified.
`tolerance`	`DOUBLE`	A value that represents the maximum number of meters to simplify in the specified geography.

Example

In this example, ST_SIMPLIFYARRAY evaluates a `POLYGON` with an outer ring of 13 points and an inner ring with four points. The example sets a tolerance argument of 10,000 meters, meaning that is the maximum distance that the function simplifies.

SQL

```
SELECT ST_SIMPLIFYARRAY(
        ST_POLYGON(
            'POLYGON((0 0, 1 0, 1 0.49, 1.01 0.49, 1.01 0.51, 1 0.51, 1 1, 0 1, 0 0
            ),
        10000);
```

Output: `[POLYGON((0.0 0.5000000762285838, 1.0 0.500000076228584, 1.0 1.0, 0.0 1.0, 0.0 0.5000000762285838)), POLYGON((1.0 0.4998004375649452, 0.5 0.4, 0.0 0.4998004375649452, 0.0 0.0, 1.0 0.0, 1.0 0.4998004375649452))]`

As shown in the output, the ST_SIMPLIFYARRAY function has reduced the size of the outer `POLYGON` from 13 points to 5.

Note that the function actually increases the number of points in the inner ring, going from four to six. The increase is due to how close the inner ring is to the edges of the outer ring. Taken altogether, the function still reduced the total number of points in the `POLYGON`.

ST_SNAPTOGRID

Returns a new geography value with all `POINT` values rounded to the specified precision. This function is similar to **ST_REDUCEPRECISION** but it allows you to specify the precision for the x and y coordinates separately.

Syntax

```
SQL

ST_SNAPTOGRID(geo1, precision_x, precision_y)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be rounded to the specified precision values.
`precision_x`	`DOUBLE`	A decimal value representing the level of precision for the x value. For example, a precision of `0.01` returns values rounded to the hundredth decimal. Do not use digits other than `0` and `1` to represent the precision.
`precision_y`	`DOUBLE`	A decimal value representing the level of precision for the y value. For example, a precision of `0.1` returns values rounded to the tenth decimal. Do not use digits other than `0` and `1` to represent the precision.

Example

```
SQL

SELECT ST_SNAPTOGRID(
    ST_MAKEPOINT(1.1, 3.11), 1, 0.1);
```

Output: `POINT(1.0 3.1)`

ST_SYMDIFFERENCEARRAY

Returns a geographic array that contains the parts that are not common between two geographic objects, `geo1` and `geo2`.

Syntax

```sql
ST_SYMDIFFERENCEARRAY(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object to be compared to the geographic objects in `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object to be compared to the geographic objects in `geo1`.

Example

```sql
SELECT ST_SYMDIFFERENCEARRAY(
    ST_LINESTRING('LINESTRING(1 1, 2 2)'),
    ST_LINESTRING('LINESTRING(1 1, 2 2, 3 3)'));
```

Output: `[LINESTRING(2.0 2.0, 3.0 3.0)]`

ST_UNIONARRAY

Performs a union of the input geography values to produce a geographic array.

Syntax

```sql
ST_UNIONARRAY(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object to be unified with the geographic values in `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geographic object to be unified with the geographic values in `geo1`.

Example

```sql
SELECT ST_UNIONARRAY(
    ST_LINESTRING('LINESTRING(0 0, 1 1, 2 2)'),
    ST_LINESTRING('LINESTRING(1 1, 2 2, 3 3)'));
```

Output: `['LINESTRING(0.0 0.0, 1.0 1.0, 2.0 2.0, 3.0 3.0)']`

Related Links

Geospatial Data Types

Spatial Measurement

Spatial Relationships

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Spatial Relationships

OcientGeo™ spatial relationship functions use arguments to test for different types of spatial relationships.

ST_CLUSTERDBSCAN

Returns the cluster number for each input geography based on a two-dimensional implementation of the density-based spatial clustering of applications with noise (**DBSCAN**) algorithm.

This function is a window function because the resulting value for an individual row depends on other rows and not just itself. No frame specification is allowed.

This function supports sort orders, but it is unnecessary unless deterministic resolution of tie-breaking in cluster determination is required. This tie-breaking happens only in situations where a point on the boundary of a cluster is equally valid for multiple clusters. In these cases, the result depends on the sort order of the data. For this reason, you might want to sort the data to force the deterministic behavior by using an `ORDER BY` in the `OVER` clause.

The function returns NULL for any row not included in a cluster.

Syntax

```
SQL

ST_CLUSTERDBSCAN(geo, eps, min_neighbors)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be grouped into a cluster based on the DBSCAN algorithm.
`eps`	`DOUBLE`	A numeric argument that controls the separation between objects before they are considered a new cluster. The `eps` argument is in meters.

© 2026 Ocient, Inc.

Argument	Data Type	Description
`min_neighbors`	`BIGINT`	The minimum number of objects within the appropriate distance of each other that need to exist before it is considered a cluster.

Example

SQL

```
SELECT ST_CLUSTERDBSCAN(ST_POINT(1.5, 1.5), 1.0, 1) OVER();
```

Output: `1`

ST_CONTAINS

Returns `TRUE` if the first geographic argument, `geo1`, contains the second geographic argument, `geo2`.

`geo1` contains `geo2` only if all of these criteria are met:

- No `POINT` values of `geo2` are located at the exterior of `geo1`.
- At least one `POINT` in the interior of `geo2` lies in the interior of `geo1`.

Geographies do not contain their boundaries, therefore, all `POINT` values in a `LINESTRING` are interior, except for the endpoints, which are regarded as exterior boundaries.

If either `geo1` or `geo2` is NULL, then the function returns NULL.

Syntax

SQL

```
SELECT ST_CONTAINS(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2`. The `POINT` values in `geo1` are evaluated to determine if they are contained within `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1`. The `POINT` values in `geo2` are evaluated to determine if they are contained within `geo1`.

Example

SQL

```
SELECT ST_CONTAINS(
    ST_LINEFROMTEXT('LINESTRING(0 0, 1 1, 2 2)'),
    ST_POINT(1, 1));
```

Output: `true`

ST_CONTAINSPROPERLY

This function is similar to ST_CONTAINS, but it has stricter criteria.

Returns true if `geo2` lies entirely in the interior of `geo1` and does not intersect or touch the boundary or exterior points.

Returns NULL if either `geo1` or `geo2` is NULL.

Syntax

SQL

```
ST_CONTAINSPROPERLY(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2`. The `POINT` values in `geo1` are evaluated to determine if they are contained within `geo2`.

Argument	Data Type	Description
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1`. The `POINT` values in `geo2` are evaluated to determine if they are contained within `geo1`.

Examples

In this example, the first geographic object is a `POLYGON` representing a simple square (0 0, 1 0, 1 1, 0 1). The second geographic object is a small `LINESTRING` that spans from the point (0.25 0.25) to the point (0.75 0.75).

Because the line is located entirely within the polygon, never touching its boundaries, the function evaluates to `true`.

SQL

```
SELECT ST_CONTAINSPROPERLY(
        ST_MAKEPOLYGON(
            ST_MAKELINE(
                ARRAY [ST_POINT(0, 0), ST_POINT(1, 0), ST_POINT(1, 1), ST_POINT(0,
        ST_MAKELINE(ST_POINT(0.25, 0.25), ST_POINT(0.75, 0.75)));
```

Output: `TRUE`

The second geographic object is slightly altered in this example, where the line spans from (0 0) to (1 1). Both of these endpoints touch the boundaries of the polygon, so the function now evaluates to `false`.

SQL

```
SELECT ST_CONTAINSPROPERLY(
        ST_MAKEPOLYGON(
            ST_MAKELINE(
                ARRAY [ST_POINT(0, 0), ST_POINT(1, 0), ST_POINT(1, 1), ST_POINT(0,
        ST_MAKELINE(ST_POINT(0, 0), ST_POINT(1, 1)));
```

Output: `FALSE`

ST_COVERS

Returns `TRUE` if no `POINT` in `geo2` is outside of `geo1`.

Returns NULL if either `geo1` or `geo2` is NULL.

Syntax

SQL

```
ST_COVERS(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2`. The `POINT` values in `geo2` are evaluated to determine if they are contained within `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1`. The `POINT` values in `geo2` are evaluated to determine if they are contained within `geo1`.

Example

SQL

```
SELECT ST_COVERS(
        ST_MAKEPOLYGON(
            ST_MAKELINE(
                ARRAY [ST_POINT(0, 0), ST_POINT(1, 0), ST_POINT(1, 1), ST_POINT(0,
        ST_MAKELINE(ST_POINT(0, 0), ST_POINT(1, 1)));
```

Output: `TRUE`

ST_COVEREDBY

This function operates similarly to ST_COVERS, except the arguments are interchanged.

Returns `TRUE` if no `POINT` in `geo1` is outside of `geo2`.

© 2026 Ocient, Inc.

Returns NULL if either `geo1` or `geo2` is NULL.

Syntax

```
SQL

ST_COVEREDBY(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2`. The `POINT` values in `geo1` are evaluated to determine if they are contained within `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1` The `POINT` values in `geo1` are evaluated to determine if they are contained within `geo2`.

Example

```
SQL

SELECT ST_COVEREDBY(
    ST_POINT(1,1),
    ST_POLYGON('POLYGON((1 1, 2 1, 2 2, 1 1))'));
```

Output: `TRUE`

ST_CROSSES

Returns `TRUE` if two geospatial objects meet this criteria:

- The intersection of the geospatial interiors is not empty.
- The intersection is not equal to `geo1` or `geo2`.
- Neither geospatial object is a single `POINT`.

Returns NULL if either argument is NULL.

Syntax

```sql
ST_CROSSES(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2` to evaluate whether they cross.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1` to evaluate whether they cross.

Example

```sql
SELECT ST_CROSSES(
    ST_LINESTRING(ST_POINT(0,0), ST_POINT(2,2)),
    ST_LINESTRING(ST_POINT(1,0), ST_POINT(1,5)));
```

Output: `TRUE`

ST_DISJOINT

Returns `true` if the specified geographies have no intersection, including boundaries. Both geographic arguments can be different types.

This function is the opposite of the **ST_INTERSECTS** function.

Syntax

```sql
ST_DISJOINT(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2` to evaluate whether they intersect.

Argument	Data Type	Description
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1` to evaluate whether they intersect.

Example

SQL

```
SELECT ST_DISJOINT(
    ST_POLYGON(ST_LINESTRING('LINESTRING(2 3, 2 4, 2 4, 2 3)')),
    ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5, 1 2)')));
```

Output: `TRUE`

ST_DWITHIN

Returns `TRUE` if the geographies are within a specified distance in meters.

Syntax

SQL

```
ST_DWITHIN(geo1, geo2, distance_meters)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2` to evaluate whether they are within the specified distance `distance_meters`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1` to evaluate whether they are within the specified distance `distance_meters`.
`distance_meters`	`DOUBLE`	A numeric value that represents the number of meters to evaluate the distance between `geo1` and `geo2`.

Example

```sql
SELECT ST_DWITHIN(
    ST_POINT('POINT(1 1)'),
    ST_POINT('POINT(0 0)'),
    100);
```

Output: `FALSE`

ST_EQUALS

Returns `TRUE` if both geographies are spatially equal.

Syntax

```sql
ST_EQUALS(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2` to evaluate whether they are equal.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1` to evaluate whether they are equal.

Example

```sql
SELECT ST_EQUALS(
    ST_POLYGON(ST_LINESTRING('LINESTRING(10 10, 0 0, 0 0, 10 10)')),
    ST_POLYGON(ST_LINESTRING('LINESTRING(10 10, 0 0, 0 0, 10 10)')));
```

Output: `True`

ST_INTERSECTS

Returns `TRUE` if the specified geographies intersect, including boundaries.

This function is the opposite of the **ST_DISJOINT** function.

Syntax

```
SQL

ST_INTERSECTS(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2` to evaluate whether they intersect.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1` to evaluate whether they intersect.

Example

```
SQL

SELECT ST_INTERSECTS(
    ST_POLYGON(ST_LINESTRING('LINESTRING(2 3, 2 4, 2 4, 2 3)')),
    ST_POLYGON(ST_LINESTRING('LINESTRING(1 2, 1 4, 1 5, 1 2)')));
```

Output: `FALSE`

ST_ISCCW

Alias for **ST_ISPOLYGONCCW**.

ST_ISCLOSED

Returns `TRUE` if an input `LINESTRING` has starting and ending points that are equal.

If you specify a `POLYGON` or `POINT` value, the function returns `true` if the specified geography is not empty.

Syntax

```
SQL

ST_ISCLOSED(line)
```

Argument	Data Type	Description
`line`	`POINT`, `LINESTRING`, or `POLYGON`	If you specify a `LINESTRING` object, the function evaluates it to determine whether its starting and ending points are equal. If you specify a `POLYGON` or `POINT` value, the function returns `true` if the specified geography is not empty.

Example

```
SQL

SELECT ST_ISCLOSED(
    ST_LINESTRING('LINESTRING(1 2, 1 3, 1 2)'));
```

Output: `TRUE`

ST_ISPOLYGONCW

Alias for `BOOLNOT(ST_ISCCW(poly))`.

Returns `TRUE` if an input `POLYGON` has an exterior that is clockwise.

If the `POLYGON` argument is empty, this function returns `TRUE`. If the argument is NULL, this function returns NULL.

Syntax

```
SQL

ST_ISPOLYGONCW(poly)
```

Argument	Data Type	Description
`poly`	`POLYGON`	A `POLYGON` object to evaluate to determine whether its point values align in a clockwise direction.

Example

SQL

```
SELECT ST_ISPOLYGONCW(
    ST_POLYGON('POLYGON((0 0,1 0,1 1,0 1,0 0))'));
```

Output: `FALSE`

ST_ISPOLYGONCCW

Alias for ST_ISCCW.

Returns `TRUE` if an input `POLYGON` has an exterior that is counter-clockwise.

If the `POLYGON` argument is empty, this function returns `TRUE`. If the argument is NULL, this function returns NULL.

Syntax

SQL

```
ST_ISPOLYGONCCW(poly)
```

Argument	Data Type	Description
`poly`	`POLYGON`	A `POLYGON` object to evaluate to determine whether its point values align in a counter-clockwise direction.

Example

```sql
SELECT ST_ISPOLYGONCCW(
    ST_POLYGON('POLYGON((0 0,1 0,1 1,0 1,0 0))'));
```

Output: `TRUE`

ST_ISRING

Returns `TRUE` if the specified `LINESTRING` is closed and does not intersect itself. This function is equivalent to evaluating both **ST_ISCLOSED** and **ST_ISSIMPLE**.

If you specify a value other than `LINESTRING`, the function throws an error.

Syntax

```sql
ST_ISRING(geo)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A `LINESTRING` object to evaluate to determine whether its starting and ending points are equal and whether it has no intersections.

Example

```sql
SELECT ST_ISRING(
    ST_LINESTRING('LINESTRING(1 2, 2 4, 1 3, 1 2)'));
```

Output: `TRUE`

ST_ISSIMPLE

For `LINESTRING` values, this function returns `FALSE` if any line segments intersect anywhere besides the endpoints.

For `POLYGON` values, the function returns `FALSE` if either the exterior ring or any interior hole is not simple.

For `POINT` values, the function always returns `TRUE`.

Syntax

```
SQL

ST_ISSIMPLE(geo)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to evaluate to determine if any points intersect. For `LINESTRING` values, this function returns `FALSE` if any line segments intersect anywhere besides the endpoints. For `POLYGON` values, the function returns `FALSE` if either the exterior ring or any interior hole is not simple. For `POINT` values, the function always returns `TRUE`.

Example

```
SQL

SELECT ST_ISSIMPLE(
    ST_POLYGON('POLYGON((1 2, 3 4, 3 5, 1 2))'));
```

Output: `TRUE`

ST_ISVALID

Returns `TRUE` if the specified geospatial value is a well-formed and valid geography according to the **OGC standards**.

`LINESTRING` values are not valid if they are composed of more than one `POINT` with the same coordinates.

`POLYGON` values are valid if they are composed of one external ring (shell) and zero or more internal rings. A ring is a `LINESTRING` where only the first and last `POINT` intersect each other. All internal rings need to be contained inside the shell, and none of the rings should intersect with each other in more than one `POINT`.

Syntax

```
SQL

ST_ISVALID(geo1)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to evaluate to determine if it is a well-formed geography.

Example

```
SQL

SELECT ST_ISVALID(
    ST_LINESTRING('LINESTRING(1 2, 2 4)'));
```

Output: `TRUE`

ST_OVERLAPS

Returns `TRUE` if both geographic arguments are of the same dimension and intersect, but neither contains the other. Returns NULL if either `geo1` or `geo2` is NULL.

Syntax

```
SQL

ST_OVERLAPS(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for evaluation to determine if it is shaped similarly and intersects `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for evaluation to determine if it is shaped similarly and intersects `geo1`.

Example

SQL

```
SELECT ST_OVERLAPS(
    ST_LINESTRING(ST_POINT(0,0), ST_POINT(2,2)),
    ST_LINESTRING(ST_POINT(1,1), ST_POINT(3,3)));
```

Output: `TRUE`

ST_POINTINSIDECIRCLE

Returns `TRUE` if the geographic object is inside a circle centered at the specified point coordinates and with the specified radius.

Syntax

SQL

```
ST_POINTINSIDECIRCLE(geo, center_x, center_y, circle_radius)
```

Argument	Data Type	Description
`geo`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for evaluation to determine if it is contained within a circle of the specified dimensions.
`center_x`	`DOUBLE`	The x coordinate of the center of the circle to evaluate.
`center_y`	`DOUBLE`	The y coordinate of the center of the circle to evaluate.
`circle_radius`	`DOUBLE`	The radius in meters of the circle to evaluate.

© 2026 Ocient, Inc.

Example

```sql
SELECT ST_POINTINSIDECIRCLE(
    ST_POINT(1, 3), 1, 2, 3000000);
```

Output: `TRUE`

ST_RELATE

Returns the **DE-9IM** intersection string that represents the nature of the intersection with the specified geographies. If you specify an intersection pattern as the third argument, this function returns `TRUE` if the intersection of the geographies satisfies the pattern and `FALSE` otherwise.

Syntax

```sql
ST_RELATE(geo1, geo2, [, intersectionPattern])
```

Parameter	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to relate with `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to relate with `geo1`.
`intersectionPatter n`	`STRING`	Optional. Intersection pattern to query the nature of the intersection of `geo1` and `geo2`. Must contain exactly 9 characters, corresponding to the component (interior, boundary, or exterior) intersections considered by DE-9IM. Supported characters are: * `T` — Components intersect. * `F` — Components do not intersect. * `0` — Component intersection is zero-dimensional. * `1` — Component intersection is 1-dimensional. * `2` — Component intersection is 2-dimensional.

Parameter	Data Type	Description
		* `*` — Specify no restriction on the component intersection.

Examples

Determine the nature of the intersection between the point `(0.0, 0.0)` and the line from `-1 0` to `1 0`. This function returns the DE-9IM string `0FFFFF102`, which means that the boundaries do not intersect at all, but there is evidence of an intersection at the zero, one, and two-dimensional spaces.

SQL

```
SELECT ST_RELATE(ST_POINT(0.0, 0.0),
    ST_LINESTRING('LINESTRING(-1 0, 1 0)'));
```

*Output: * `0FFFFF102`

Determine whether the intersection between the point `(0.0, 0.0)` and the line from `-1 0` to `1 0` follows the DE-9IM pattern string `TF*FFF102`. This function returns `TRUE`, which means the nature of the intersection between these two geographies conforms to this pattern.

SQL

```
SELECT ST_RELATE(ST_POINT(0.0, 0.0),
    ST_LINESTRING('LINESTRING(-1 0, 1 0)'), 'TF*FFF102');
```

*Output: * `TRUE`

ST_TOUCHES

Returns `TRUE` if the only `POINT` values in common between the two geographic arguments lie in the union of their boundaries.

Syntax

```sql
ST_TOUCHES(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for evaluation to determine if it touches `geo2`.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object for evaluation to determine if it touches `geo1`.

Example

```sql
SELECT ST_TOUCHES(
    ST_LINESTRING('LINESTRING(1.0 3.0, 0.0 3.0)'),
    ST_LINESTRING('LINESTRING(1.0 3.0, 2.0 3.0)'));
```

Output: `TRUE`

ST_WITHIN

Returns `TRUE` if the geography specified by the first argument is fully inside the geography specified by the second argument.

`ST_WITHIN(geo1, geo2)` is equivalent to `ST_CONTAINS(geo2, geo1)`. For details, see **ST_CONTAINS**.

Syntax

```sql
ST_WITHIN(geo1, geo2)
```

Argument	Data Type	Description
`geo1`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo2` to evaluate whether it is fully inside this second geography.
`geo2`	`POINT`, `LINESTRING`, or `POLYGON`	A geospatial object to be compared to `geo1` to evaluate whether it contains the first geography.

Example

SQL

```
SELECT ST_WITHIN(
    ST_POINT(1, 1),
    ST_LINEFROMTEXT('LINESTRING(0 0, 1 1, 2 2)')
);
```

Output: `true`

Related Links

Geospatial Data Types

Spatial Measurement

Spatial Operators

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Spatiotemporal Measurement

OcientGeo™ spatiotemporal measurement functions can perform basic calculations on geospatial data that is paired with TIMESTAMP data.

ST_DISTANCE

Returns the two-dimensional interpolated minimum simultaneous distance between two `LINESTRING-TIMESTAMP` array pairs in the specified unit of measurement.

If any argument is NULL or empty, the function returns NULL. If either `LINESTRING-TIMESTAMP` array pair has a mismatched number of points and timestamps, the function returns an error.

Behavior is undefined if either timestamp array is not monotonically increasing.

SQL

```
ST_DISTANCE(geo1, ts_arr1, geo2, ts_arr2, units [, use_spheroid] )
```

Argument	Data Type	Description
`geo1`	`LINESTRING`	A geospatial object to be evaluated for its minimum cotemporal distance from `geo2`.
`ts_arr1`	`TIMESTAMP ARRAY`	A timestamp array that is paired with `geo1`.
`geo2`	`LINESTRING`	A geospatial object to be evaluated for its minimum cotemporal distance from `geo1`.
`ts_arr2`	`TIMESTAMP ARRAY`	A timestamp array that is paired with `geo2`.
`units`	`STRING`	The unit of measurement used for the minimum cotemporal distance. Accepted options include `'FEET'`, `'KILOMETERS'`, `'MILES'`, or `'METERS'`. The default value is `'METERS'`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model.

© 2026 Ocient, Inc.

Argument	Data Type	Description
		The default value is `FALSE`.

Example

SQL

```
SELECT ST_DISTANCE(
        ST_LINESTRING('linestring(0 0,0 12)'),
        TIMESTAMP []('
            '2020-07-04 07:00:00.000000000',
            '2020-07-04 11:00:00.000000000'),
        ST_LINESTRING('linestring(0 0,6 0)'),
        TIMESTAMP []('
            '2020-07-04 08:00:00.000000000',
            '2020-07-04 10:00:00.00000000'),
        'MILES',
        TRUE);
```

Output: `206.124`

> To use ST_DISTANCE as a spatial function to calculate the minimum distance between two points, see the **ST_DISTANCE** function in the Spatial Measurement section.

ST_MAXDISTANCE

Returns the two-dimensional interpolated maximum cotemporal distance between two `LINESTRING-TIMESTAMP` array pairs in the specified unit of measurement.

If any argument is NULL or empty, the function returns NULL. If either `LINESTRING-TIMESTAMP` array pair has a mismatched number of points and timestamps, the function returns an error.

Behavior is undefined if either timestamp array is not monotonically increasing.

SQL

```
ST_MAXDISTANCE(geo1, ts_arr1, geo2, ts_arr2, units [, use_spheroid] )
```

Argument	Data Type	Description
`geo1`	`LINESTRING`	A geospatial object to be evaluated for its maximum cotemporal distance from `geo2`.
`ts_arr1`	`TIMESTAMP ARRAY`	A `TIMESTAMP ARRAY` that is paired with `geo1`.
`geo2`	`LINESTRING`	A geospatial object to be evaluated for its maximum cotemporal distance from `geo1`.
`ts_arr2`	`TIMESTAMP ARRAY`	A `TIMESTAMP ARRAY` that is paired with `geo2`.
`units`	`STRING`	The unit of measurement used for the maximum cotemporal distance. Accepted options include `'FEET'`, `'KILOMETERS'`, `'MILES'`, or `'METERS'`. The default value is `'METERS'`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. The default value is `FALSE`.

Example

SQL

```
SELECT ST_MAXDISTANCE(
    ST_LINESTRING('linestring(0 0,0 12)'),
        TIMESTAMP[]('2020-07-04 07:00:00.000000000', '2020-07-04 11:00:00.000000000
    ST_LINESTRING('linestring(0 0,6 0)'),
        TIMESTAMP[]('2020-07-04 08:00:00.000000000', '2020-07-04 10:00:00.00000000'
    'MILES',
    TRUE);
```

Output: `743.83`

To use ST_MAXDISTANCE as a spatial function to calculate the maximum distance between two points, see the **ST_MAXDISTANCE** function in the Spatial Measurement section.

ST_TOTALSECONDSININTERSECTION

Returns the total number of seconds spent in the intersection result calculated by the spatiotemporal version of **ST_INTERSECTION**.

Execute this function only with the result from ST_INTERSECTION. If the input is an empty, correctly typed tuple, this function returns 0 seconds. If the input is NULL, this function returns NULL.

```
SQL

ST_TOTALSECONDSININTERSECTION(tpl)
```

Argument	Data Type	Description
`tpl`	Tuple of `LINESTRING-TIMESTAMP` pairs	A tuple of `LINESTRING-TIMESTAMP` pairs used to evaluate how long an intersection existed.

Example

```
SQL

SELECT ST_TOTALSECONDSININTERSECTION(
    ST_INTERSECTION(
        ST_LINESTRING('LINESTRING(1 1,2 2,1 1)'),
            TIMESTAMP[](
                '2020-07-04 07:00:00.000000000',
                '2020-07-04 11:00:00.000000000',
                '2020-07-04 12:00:00.000000000'),
        ST_LINESTRING('LINESTRING(2 2,1 1)')));
```

Output: `18000.0`

Related Links

Geospatial Data Types

Spatiotemporal Operators

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Spatiotemporal Operators

OcientGeo™ spatiotemporal operators perform calculations on geospatial data using an array of TIMESTAMP arguments.

ST_LONGESTLINE

With the specified two `LINESTRING-TIMESTAMP ARRAY` pairs, this function returns a two-point `LINESTRING` that represents the maximum distance between points at a concurrent time.

If the two `LINESTRING` values do not overlap temporally, this function returns NULL.

Syntax

SQL

```
ST_LONGESTLINE(linestring1, ts_arr1, linestring2, ts_arr2 [, use_spheroid] )
```

Argument	Data Type	Description
`linestring1`	`LINESTRING`	A geospatial object to be evaluated for its maximum cotemporal distance from `linestring2`.
`ts_arr1`	`TIMESTAMP ARRAY`	A timestamp array that is paired with `linestring1`. Behavior is undefined if this timestamp array is not monotonically increasing.
`linestring2`	`LINESTRING`	A geospatial object to be evaluated for its maximum cotemporal distance from `linestring1`.
`ts_arr2`	`TIMESTAMP ARRAY`	A timestamp array that is paired with `linestring2`. Behavior is undefined if this timestamp array is not monotonically increasing.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

SQL

```
SELECT ST_LONGESTLINE(
        ST_MAKELINE(ST_POINT(0, 0), ST_POINT(0, 12)),
        TIMESTAMP [](
            '2020-07-04 07:00:00.000000000',
            '2020-07-04 11:00:00.000000000'),
        ST_MAKELINE(ST_POINT(0, 0), ST_POINT(8, 0)),
        TIMESTAMP [](
            '2020-07-04 08:00:00.000000000',
            '2020-07-04 10:00:00.000000000'));
```

Output: `LINESTRING(0 9, 8 0)`

To use ST_LONGESTLINE as a spatial function to calculate the longest line between two points, see the **ST_LONGESTLINE** function.

ST_LINEGETALLTIMESATPOINT

Returns a timestamp array of all times when the specified `LINESTRING` value intersects the specified `POINT` value. If the line never intersects the point, then the function returns an empty timestamp array.

If the specified inputs are NULL, the function returns NULL.

This function cannot represent intervals of time where the line was stationary at a point, therefore, use this function only for mobile paths. If the path is stationary, use the **ST_INTERSECTION** function as it can represent intervals of time spent intersecting a stationary `LINESTRING`.

Syntax

SQL

```
ST_LINEGETALLTIMESATPOINT(line, timestamp_arr, point)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A geospatial `LINESTRING` to be evaluated for all time values in the `timestamp_arr` when it intersects the specified `point` value. If this value is empty, the function returns an empty timestamp array. This value must be the same size as the `timestamp_arr`.
`timestamp_arr`	`TIMESTAMP ARRAY`	A `TIMESTAMP ARRAY` that is paired with the `line` argument. `timestamp_arr` must be the same size as the `line` value. Behavior is undefined if the timestamp array is not monotonically increasing.
`point`	`POINT`	A geospatial point to be evaluated for the time values when the point intersects the `line` value.

Example

SQL

```
SELECT ST_LINEGETALLTIMESATPOINT(
        ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0), ST_POINT(0, 0)),
        TIMESTAMP []('2020-07-04 07:00:00.000000000',
            '2020-07-04 08:00:00.000000000',
            '2020-07-04 09:00:00.000000000'),
        ST_POINT(0, 0));
```

Output: `TIMESTAMP[]('2020-07-04 07:00:00.000000000', '2020-07-04 09:00:00.000000000')`

ST_LINEGETPOINTATTIME

Returns a `POINT` within the bounds of the specified `LINESTRING` that corresponds to the interpolated point at the specified `TIMESTAMP` value.

The specified `TIMESTAMP` value does not need to be an actual value explicitly included in the array. The function interpolates the corresponding point, which means that the function approximates the point coordinates from the surrounding values.

The `LINESTRING` and `TIMESTAMP` array must be the same size. If the timestamp array is not monotonically increasing, the behavior is undefined.

Syntax

```
SQL

ST_LINEGETPOINTATTIME(line, timestamp_arr, time_for_point)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A geospatial `LINESTRING` to be evaluated for an interpolated point, based on the `time_for_point` value. This value must be the same size as the `timestamp_arr`.
`timestamp_arr`	`TIMESTAMP ARRAY`	A `TIMESTAMP ARRAY` that is paired with the `line` argument. `timestamp_arr` must be the same size as the `line` value. Behavior is undefined if the timestamp array is not monotonically increasing.
`time_for_point`	`TIMESTAMP`	A time value to approximate a `POINT` value on the `line` argument, based on interpolating the values in `timestamp_arr`. If this time value precedes the first value in `timestamp_arr`, the function returns the first `POINT` on `line`. Similarly, if this time value comes after the last value in `timestamp_arr`, the function returns the last `POINT` on `line`.

Example

```
SQL

SELECT ST_LINEGETPOINTATTIME(
    ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0)),
    TIMESTAMP[]('2020-07-04 07:00:00.000000000', '2020-07-04 08:00:00.000000000'),
    TIMESTAMP('2020-07-04 07:30:00.000000000'));
```

Output: `POINT(2, 0)`

ST_LINEGETTIMEATPOINT

Returns the interpolated time of the specified `POINT` on the specified `LINESTRING` that is paired with a `TIMESTAMP ARRAY`.

The `LINESTRING` and `TIMESTAMP ARRAY` must be the same size.

Syntax

SQL

```
ST_LINEGETTIMEATPOINT(line, timestamp_arr, point_for_time)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A geospatial line to be evaluated for an interpolated time value based on the `point_for_time` value. This value must be the same size as the `timestamp_arr`.
`timestamp_arr`	`TIMESTAMP ARRAY`	A timestamp array that is paired with the `line` argument. `timestamp_arr` must be the same size as the `line` value. Behavior is undefined if the timestamp array is not monotonically increasing.
`point_for_time`	`POINT`	A `POINT` value to approximate a `TIMESTAMP` value on the `line` argument, based on interpolating the values in `timestamp_arr`. If `line` intersects this `POINT` value multiple times, then the function returns the earliest time.

Example

SQL

```
SELECT ST_LINEGETTIMEATPOINT(
    ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0)),
    TIMESTAMP[]('2020-07-04 07:00:00.000000000', '2020-07-04 08:00:00.000000000'),
    ST_POINT(2, 0));
```

Output: `TIMESTAMP('2020-07-04 07:30:00.000000000')`

© 2026 Ocient, Inc.

ST_INTERSECTION

Returns a tuple that represents the intersection of a spatiotemporal `LINESTRING` with a static geography.

If the specified static geography is a `POINT`, the returned value is a tuple that contains a paired `POINT` array and `TIMESTAMP` array. If the specified static geography is a `LINESTRING` or `POLYGON`, then the returned value is a `LINESTRING` array paired with a two-dimensional `TIMESTAMP` array.

Syntax

SQL

```
ST_INTERSECTION(line, timestamp_arr, intersection_point)
```

Argument	Data Type	Description
`line`	`LINESTRING`	A geospatial line to be evaluated for all intersections with the `intersection_point`. If this argument is empty, the function returns NULL.
`timestamp_arr`	`TIMESTAMP ARRAY`	A `TIMESTAMP ARRAY` that is paired with the `line` argument. `timestamp_arr` must be the same size as the `line` value. Behavior is undefined if the timestamp array is not monotonically increasing.
`intersection_point`	`POINT, LINESTRING`, or `POLYGON`	A geographic object to evaluate for all intersections with the `line` argument. If this value is empty, the function returns an empty tuple. If the value is NULL, the function returns NULL.

Example

SQL

```
SELECT ST_INTERSECTION(
    ST_LINESTRING('LINESTRING(1 1,2 2,3 3)'),
    TIMESTAMP[]('2020-07-04 07:00:00.000000000', '2020-07-04 11:00:00.000000000',
         '2020-07-04 12:00:00.000000000'),
    ST_POLYGON('POLYGON((2 2,3 2,3 3,3 2,2 2))'));
```

Output: `TUPLE(LINESTRING[](ST_LINESTRING('LINESTRING(2 2)'), ST_LINESTRING('LINESTRING(3 3)')), ARRAY[ARRAY[TIMESTAMP '2020-07-04 11:00:00.000000000'], ARRAY[TIMESTAMP '2020-07-04 12:00:00.000000000']])`

ST_SHORTESTLINE

When you specify two `LINESTRING-TIMESTAMP ARRAY` pairs, this function returns a `LINESTRING` with two points that represents the minimum distance between points at a concurrent time.

If the two linestrings do not overlap temporally, this function returns NULL.

Syntax

SQL

```
ST_SHORTESTLINE(linestring1, ts_arr1, linestring2, ts_arr2 [, use_spheroid ])
```

Argument	Data Type	Description
`linestring1`	`LINESTRING`	A geospatial object to be evaluated for its minimum cotemporal distance from `linestring2`.
`ts_arr1`	`TIMESTAMP ARRAY`	A timestamp array that is paired with `linestring1`.
`linestring2`	`LINESTRING`	A geospatial object to be evaluated for its minimum cotemporal distance from `linestring1`.
`ts_arr2`	`TIMESTAMP ARRAY`	A timestamp array that is paired with `linestring2`.
`use_spheroid`	`BOOLEAN`	Optional. If you set this argument to `TRUE`, this function uses a spheroid model instead of a spherical model. Defaults to `FALSE`.

Example

SQL

```
SELECT ST_SHORTESTLINE(
    ST_MAKELINE(ST_POINT(0, 0), ST_POINT(4, 0)),
    TIMESTAMP[]('2020-07-04 07:00:00.000000000', '2020-07-04 08:00:00.000000000'),
    ST_MAKELINE(ST_POINT(3, 0), ST_POINT(4, 8)),
    TIMESTAMP[]('2020-07-04 07:00:00.000000000', '2020-07-04 07:30:00.000000000'));
```

Output: `LINESTRING(0.09295649430902601 0, 3.0461764508256914 0.37184471667525754)`

To use ST_SHORTESTLINE as a spatial function to calculate the shortest line between two points, see the **ST_SHORTESTLINE** function.

Related Links

Geospatial Data Types

Spatiotemporal Measurement

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Machine Learning Model Functions

OcientML™ functionality supports these SQL functions for machine learning models.

> The scope of a machine learning model is the schema.

Supported Machine Learning Models

Supported machine learning models and reference material are divided into these categories.

> For functions to help organize data before training a model, see **Data Preparation**.

Regression Models

- **Simple Linear Regression**
- **Multiple Linear Regression**
- **Vector Autoregression**
- **Polynomial Regression**
- **Linear Combination Regression**
- **Nonlinear Regression**
- **Gradient Boosted Trees**
- **Regression Tree**

Classification Models

- **K Nearest Neighbor Classification**
- **Naive Bayes Classification**
- **Decision Tree**
- **Random Forest**
- **Logistic Regression**

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

- **Support Vector Machine**
- **Gradient Boosted Trees**

Clustering and Dimension Reduction Models

- **Principal Component Analysis**
- **K-Means Clustering**
- **Gaussian Mixture**
- **Linear Discriminant Analysis**

Ensemble Models

- **Bagging**
- **Boosting**
- **Stacking**

Other Models

- **Association Rules**
- **Feedforward Neural Network**

For a view of the full list of model options, see **Machine Learning Model Options**.

Execute a Query Using a Machine Learning Model

To create a machine learning model and manage the model, see **Machine Learning Models** for the corresponding syntax. After you create the model, you can execute a query using the model with this syntax.

Syntax

SQL

```
SELECT model_name ( expression [, ... ] ) FROM table_reference
```

© 2026 Ocient, Inc.

Parameter	Data Type	Description
`model_name`	identifier	The name of a machine learning model created using a `CREATE MODEL` SQL statement. The model name must be a valid identifier and reference an existing trained model.
`expression`	identifier string numeric	One or more expressions that serve as input features for the machine learning model evaluation. These expressions must match the expected input schema the model was trained on. Expressions can be any combination of literal values, column names, arithmetic expressions, and function invocations, with parentheses for grouping.
`table_reference`	identifier	The name of a table, view, or subquery that provides the data for model evaluation. The table must contain columns or computed expressions that match the expected input features of the model.

Example

Create a table with data for the model.

SQL

```
CREATE TABLE mldemo.mlr AS (SELECT a.c1 AS x1, b.c1 AS x2, 1 + 2*a.c1 + 3*b.c1 AS y
    FROM sys.dummy10 a, sys.dummy10 b);
Modified 100 rows
```

Create a multiple linear regression model based on the data.

SQL

```
CREATE MLMODEL mlr_model TYPE MULTIPLE LINEAR REGRESSION ON (SELECT * FROM mldemo.m
options('metrics' -> 'true');
Modified 0 rows
```

Execute a `SELECT` query against the multiple linear regression to see the actual and predicted values. Limit the result set to 10 rows.

SQL

```sql
SELECT x1, x2, y AS actual, mlr_model(x1, x2) AS predicted
    FROM mldemo.mlr
    LIMIT 10;
```

Output

Text

```
x1          x2          actual              predicted
-------------------------------------------------------------
6           1           16                  15.999999999999975
6           2           19                  18.999999999999975
6           3           22                  21.999999999999975
6           4           25                  24.999999999999975
6           5           28                  27.999999999999975
6           6           31                  30.999999999999975
6           7           34                  33.99999999999997
6           8           37                  36.99999999999997
6           9           40                  39.99999999999997
6           10          43                  42.99999999999997

Fetched 10 rows
```

Related Links

Machine Learning in Ocient

Machine Learning Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Preparation

Lag generator and vector assembler SQL functions simplify generating and organizing lagged features for time series data. Each function expands into standard SQL during query processing, so developers can reduce boilerplate and create feature sets with minimal code.

These functions fall into two main types:

- **Lag Generator Functions** — Create individual lag feature columns.
- **Vector Assembler Functions** — Group generated lag columns into vectors.

Developers can apply lag features across many Ocient®-supported models, including:

- Simple Linear Regression (univariate forecasting with lag inputs)
- Multiple Linear Regression (multivariate regression with lagged predictors)
- Polynomial Regression (regression with higher-order lagged terms)
- Linear Combination Regression (custom functions of lagged inputs)
- Nonlinear Regression (arbitrary functions that include lag variables)
- Feedforward Neural Networks (lagged features as input vectors)
- Vector Autoregression (VAR) (multivariate time series with structured lag vectors)
- Autoregression (univariate time series modeled through lagged values)
- Vector Valued Regression (multivariate dependent variables with lagged inputs)

The functions provide general-purpose utilities for time series feature engineering. In particular, VAR models benefit the most from lag functions because the models require large numbers of structured lag vectors.

Lag Generator Functions

Lag generator functions provide options for creating lagged columns for one or more variables in a single statement.

These functions follow the syntax rules of window aggregate functions as they define the window for the lag computation with an `OVER` and `ORDER BY` clause. For details, see

Window Aggregate Functions.

The window parameters of lag generator functions are represented in these examples by `<window_specification>`.

LAGS

Generates a series of lagged columns for a single variable in one statement. This function simplifies univariate time-series feature creation.

Syntax

SQL

```
LAGS(expression, start_lag, end_lag [, step])
    OVER (<window_specification>)
```

Argument	Type	Description
`expression`	ANY	The column or expression to lag.
`start_lag`	INTEGER	First lag to generate. This value must be less than the `end_lag` value.
`end_lag`	INTEGER	Last lag to generate. This value must be greater than the `start_lag` value.
`step`	INTEGER	Optional. Defines the gap between successive lags. Must be a positive integer. If unspecified, this value defaults to `1`.

Example

This query produces three lagged columns (`sales_lag1` , `sales_lag2` , `sales_lag3`) from the sales column.

SQL

```
SELECT
    LAGS(sales, 1, 3) OVER (ORDER BY day)
FROM store_sales;
```

Output

SQL

```
sales_lag1 | sales_lag2 | sales_lag3
-----------+------------+------------
NULL       | NULL       | NULL
100        | NULL       | NULL
120        | 100        | NULL
130        | 120        | 100
```

LAGS_ZEROFILL

Generates lagged columns for a single variable and replaces NULL values with `0`.

Syntax

SQL

```
LAGS_ZEROFILL(expression, start_lag, end_lag [, step])
    OVER (<window_specification>)
```

Argument	Type	Description
`expression`	ANY	The column or expression to lag.
`start_lag`	INTEGER	First lag to generate. This value must be less than the `end_lag` value.
`end_lag`	INTEGER	Last lag to generate. This value must be greater than the `start_lag` value.
`step`	INTEGER	Optional. Defines the gap between successive lags. Must be a positive integer. If unspecified, this value defaults to `1`.

Example

This query produces three lagged sales columns and replaces NULL values with 0.

SQL

```
SELECT
    LAGS_ZEROFILL(sales, 1, 3) OVER (ORDER BY day)
FROM store_sales;
```

Output

SQL

```
sales_lag1 | sales_lag2 | sales_lag3
-----------+------------+------------
0          | 0          | 0
100        | 0          | 0
120        | 100        | 0
130        | 120        | 100
```

MULTI_LAGS

Generates lagged columns for multiple variables at once. This function is intended primarily for creating features for multivariate time-series models.

Syntax

SQL

```
MULTI_LAGS(expr1, expr2, [ ,... ] start_lag, end_lag [, step])
    OVER (<window_specification>)
```

Argument	Type	Description
`expr1, expr2, [ ,... ]`	ANY	One or more columns or expressions to lag. Applies the same lag range and step to each input.
`start_lag`	INTEGER	First lag to generate. This value must be less than the `end_lag` value.
`end_lag`	INTEGER	Last lag to generate. This value must be greater than the `start_lag` value.
`step`	INTEGER	Optional. Defines the gap between successive lags. Must be a

Argument	Type	Description
		positive integer. If unspecified, this value defaults to `1`.

Example This query creates four lagged columns for both `interest_rate` and `gdp_growth`.

SQL

```
SELECT
    MULTI_LAGS(interest_rate, gdp_growth, 1, 4) OVER (ORDER BY quarter)
FROM economic_data;
```

Output

SQL

```
interest_rate_lag1 | interest_rate_lag2 | interest_rate_lag3 | interest_rate_lag4 |
-------------------+--------------------+--------------------+--------------------+
NULL               | NULL               | NULL               | NULL               |
2.5                | NULL               | NULL               | NULL               |
2.7                | 2.5                | NULL               | NULL               |
2.8                | 2.7                | 2.5                | NULL               |
```

MULTI_LAGS_ZEROFILL

Generates lagged columns for multiple variables and replaces NULL values with `0`.

Syntax

SQL

```
MULTI_LAGS_ZEROFILL(expr1, expr2, [ ,... ] start_lag, end_lag [, step])
    OVER (<window_specification>)
```

Argument	Type	Description
`expr1, expr2, [ ,... ]`	ANY	One or more columns or expressions to lag. Applies the same lag range and step to each input.

Argument	Type	Description
`start_lag`	INTEGER	First lag to generate. This value must be less than the `end_lag` value.
`end_lag`	INTEGER	Last lag to generate. This value must be greater than the `start_lag` value.
`step`	INTEGER	Optional. Defines the gap between successive lags. Must be a positive integer. If unspecified, this value defaults to `1`.

Example This query generates four lagged columns each for `x1`, `x2`, and `x3`, replacing missing values with 0.

SQL

```
SELECT
    MULTI_LAGS_ZEROFILL(x1, x2, x3, 1, 4) OVER (ORDER BY time_col)
FROM public.my_time_series_table;
```

Output

SQL

```
x1_lag1 | x1_lag2 | x1_lag3 | x1_lag4 | x2_lag1 | x2_lag2 | x2_lag3 | x2_lag4 | x3_
--------+---------+---------+---------+---------+---------+---------+---------+----
0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0
10      | 0       | 0       | 0       | 20      | 0       | 0       | 0       | 30
11      | 10      | 0       | 0       | 21      | 20      | 0       | 0       | 31
12      | 11      | 10      | 0       | 22      | 21      | 20      | 0       | 32
```

Vector Assembler Function

A vector assembler function provides options for grouping multiple lagged columns into structured vectors in a single statement. This function makes preparing input data for time series models easier, especially those that require organized lag structures, such as VAR.

© 2026 Ocient, Inc.

LAG_VECTORS

Groups lagged columns generated by the `MULTI_LAGS` or `MULTI_LAGS_ZEROFILL` functions into vector columns. Models such as VAR require these structured lag vectors.

The resulting vectors are named `lag_vector_` .

Syntax

SQL

```
LAG_VECTORS(expr1, expr2, [ ..., ] start_lag, end_lag [, step])
```

Argument	Type	Description
`expr1, expr2, [ ,... ]`	NUMERIC	Two or more lagged columns to group. Applies the same lag range and step to each input.
`start_lag`	INTEGER	First lag to assemble. This value must be less than the `end_lag` value.
`end_lag`	INTEGER	Last lag to assemble. This value must be greater than the `start_lag` value.
`step`	INTEGER	Optional. Defines the gap between successive lags. Must be a positive integer. If unspecified, this value defaults to `1` .

Examples

Assemble Lagged Columns into Vectors

This query uses `MULTI_LAGS_ZEROFILL` to generate lagged columns for `x1` , `x2` , and `x3` , then uses `LAG_VECTORS` to group them into `lag_vector1` through `lag_vector4` .

© 2026 Ocient, Inc.

SQL

```
SELECT
    {x1, x2, x3} AS current_vector,
    LAG_VECTORS(x1, x2, x3, 1, 4)
FROM (
    SELECT
        x1, x2, x3,
        MULTI_LAGS_ZEROFILL(x1, x2, x3, 1, 4) OVER (ORDER BY time_col)
    FROM public.my_time_series_table
) s;
e;
```

Output

SQL

```
current_vector | lag_vector1 | lag_vector2 | lag_vector3 | lag_vector4
---------------+-------------+-------------+-------------+-------------
{10, 20, 30}   | {0, 0, 0}   | {0, 0, 0}   | {0, 0, 0}   | {0, 0, 0}
{11, 21, 31}   | {10, 20, 30}| {0, 0, 0}   | {0, 0, 0}   | {0, 0, 0}
{12, 22, 32}   | {11, 21, 31}| {10, 20, 30}| {0, 0, 0}   | {0, 0, 0}
{13, 23, 33}   | {12, 22, 32}| {11, 21, 31}| {10, 20, 30}| {0, 0, 0}
```

Prepare Data for a VAR Model

This example shows how to prepare training data for a VAR model with three variables (`x1` , `x2` , `x3`) and four lags. The example demonstrates how the lag generator functions and vector assembler can simplify feature creation compared to writing out every LAG expression manually.

- `MULTI_LAGS_ZEROFILL` generates all individual lagged columns (`x1_lag1` … `x3_lag4`) in a single call, replacing NULL values with zero.
- `LAG_VECTORS` automatically groups these lagged columns into vector inputs (`lag_vector1` … `lag_vector4`), which the model requires.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL my_var_model
TYPE VECTOR_AUTOREGRESSION ON (
    SELECT
        {x1, x2, x3},
        LAG_VECTORS(x1, x2, x3, 1, 4)
    FROM (
        SELECT
            x1, x2, x3,
            MULTI_LAGS_ZEROFILL(x1, x2, x3, 1, 4) OVER (ORDER BY t)
        FROM public.my_time_series_table
    )
)
OPTIONS (
    'numVariables' -> '3',
    'numLags'      -> '4'
);
```

Prepare Data for a VAR Model with Manual Vector Assembly

This version of the univariate regression example uses repeated `LAG` invocations instead of the `LAGS` function. Although it produces the exact same set of lagged features (`sales_lag1` , `sales_lag2` , `sales_lag3`), the query requires writing each `LAG` expression manually. This makes the SQL longer, more repetitive, and harder to maintain compared to the concise single-line `LAGS` version.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL my_var_model_manual
TYPE VECTOR_AUTOREGRESSION ON (
    SELECT
        {x1, x2, x3},
        {x1_lag1, x2_lag1, x3_lag1},
        {x1_lag2, x2_lag2, x3_lag2},
        {x1_lag3, x2_lag3, x3_lag3},
        {x1_lag4, x2_lag4, x3_lag4}
    FROM (
        SELECT
            x1, x2, x3,
            COALESCE(LAG(x1, 1) OVER (ORDER BY t), 0) AS x1_lag1,
            COALESCE(LAG(x1, 2) OVER (ORDER BY t), 0) AS x1_lag2,
            COALESCE(LAG(x1, 3) OVER (ORDER BY t), 0) AS x1_lag3,
            COALESCE(LAG(x1, 4) OVER (ORDER BY t), 0) AS x1_lag4,
            COALESCE(LAG(x2, 1) OVER (ORDER BY t), 0) AS x2_lag1,
            COALESCE(LAG(x2, 2) OVER (ORDER BY t), 0) AS x2_lag2,
            COALESCE(LAG(x2, 3) OVER (ORDER BY t), 0) AS x2_lag3,
            COALESCE(LAG(x2, 4) OVER (ORDER BY t), 0) AS x2_lag4,
            COALESCE(LAG(x3, 1) OVER (ORDER BY t), 0) AS x3_lag1,
            COALESCE(LAG(x3, 2) OVER (ORDER BY t), 0) AS x3_lag2,
            COALESCE(LAG(x3, 3) OVER (ORDER BY t), 0) AS x3_lag3,
            COALESCE(LAG(x3, 4) OVER (ORDER BY t), 0) AS x3_lag4
        FROM public.my_time_series_table
    )
)
OPTIONS (
    'numVariables' -> '3',
    'numLags'      -> '4'
);
```

Related Links

Window Aggregate Functions

Machine Learning in Ocient

Machine Learning Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Regression Models

OcientML™ supports regression models that investigate the relationship between independent variables and at least one dependent variable or outcome. When trained, regression models can help establish the relationship between variables by estimating how one affects the other.

To create the model, use the `CREATE MLMODEL` syntax. For details, see **CREATE MLMODEL**.

Regression Model Common Options

The regression model types have many common options that provide similar functionality. This table describes the purpose of the more common options.

Option	Notes
`metrics`	The model supports the `metrics` option for all regression types. If this option is set to `true`, then the model also calculates the coefficient of determination (also known as the R^2 value), the root mean squared error (RMSE), and the adjusted R^2 value. If the option is not specified, then the model defaults this option to `false`.
`threshold`	Most regression model types support the `threshold` option, which enables soft-thresholding. This model type is the same as lasso regression for simple linear regression. However, this model type is not quite the same as lasso regression for other model types. If you specify a non-zero threshold, the model shifts coefficients towards zero by the amount of the threshold. If this shift causes the coefficient to switch from positive to negative or vice versa, the model sets the coefficient to zero. The system performs this coefficient adjustment as a final step before the system saves the model coefficients. Any reported coefficient of determination is based on the model prior to adjustment. The adjustment always causes the coefficient of determination to be smaller. However, this approach is useful as it prevents overfitting.
`gamma`	Some regression model types support the `gamma` option. The value of this option is the Tikhonov square matrix in the form: `{{a, b, c, ...}, {d, e, f, ...}, {g, h, i, ...}}`. If you do not specify this option, the model defaults the option to the `0` matrix, which equates to multiple linear regression without regularization. You can perform ridge regression by using a `gamma` value. This is a bit complicated for the polynomial and linear combination regression model types. The system computes these model types by using multiple linear regression after applying the functions to be linearly combined to the input

© 2026 Ocient, Inc.

Option	Notes
	arguments. In these cases, the system applies ridge regression after applying the model functions to the model input values.
`weighted`	Some regression model types also support weighted least squares. The model types do not support generalized least squares, where the weight matrix can be non-diagonal. To use weighted least squares, add the option `weighted` and set it to `true`. Then, add one more column to the result set used to create the model. This new column must appear last, after the dependent variable, and it specifies the weight for that sample.

ⓘ Model option names are case-sensitive.

Simple Linear Regression

Model Type: `SIMPLE LINEAR REGRESSION`

Trains a new machine learning model of type `<model_type>` on the result set returned by the SQL SELECT statement. After the database creates the model, `<model_name>` becomes an executable function in SQL SELECT statements.

SQL

```
CREATE MLMODEL model_name
    TYPE model_type ON (
        select_statement
)
    [ options( option_list ) ]
```

The result set used as input to the model must have two numeric columns. The first column is the independent variable (referred to as `x`). The second column is the dependent variable (referred to as `y`). The model finds the least squares best fit for `y = ax + b`.

The simple linear regression model type also supports the `yIntercept` option, which is the y-intercept of the resulting best-fit line. If you do not specify this option, the model does not force the y-intercept to be any particular value, and the model uses least squares to find the best value instead. If you force the y-intercept to be a particular value, the uses least squares to find the best fit with that constraint.

Model Options

Optional

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (r^2) and the root mean squared error (RMSE). This option defaults to `false`.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

`threshold` — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. This option defaults to `true`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a simple linear regression model with y-intercept `10`. Collect metrics for the model execution by setting the `metrics` option to `true`.

SQL

```
CREATE MLMODEL my_model
TYPE SIMPLE LINEAR REGRESSION ON (
  SELECT
    x1,
    y
  FROM public.my_table
)
options(
  'yIntercept' -> '10',
  'metrics' -> 'true'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the model takes a single numeric argument that represents `x` and returns `ax + b`.

SQL

```
SELECT my_model(col1) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.simple_linear_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Multiple Linear Regression

Model Type: `MULTIPLE LINEAR REGRESSION`

Multiple linear regression means that there is a vector of independent variables and the dependent variable is a scalar-valued function of the vector input, and that it is linear in all vector components.

The model uses the result set as an input. The result set has N columns that must be numeric. The first `N - 1` columns are the independent variables (it can be considered a single independent variable that is a vector). The last column is the dependent variable. The model finds the least squares best fit for `y = a1 * x1 + a2 * x2 + ... + b`, or in vector notation, `y = ax + b`, where `a` and `x` are vectors and the multiplication is a dot product.

Model Options

Optional

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (r^2), the adjusted coefficient of determination, and the root mean squared error (RMSE). This option defaults to `false`.

`threshold` — This option enables soft thresholding. If you specify this option, the option must be a positive numeric value. After the model calculates the coefficients, if any coefficients are greater than the threshold value, the model subtracts the threshold value from the coefficients. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to the coefficients. For any coefficients that are between the negative and positive threshold values, the model sets the coefficients to zero.

`weighted` — If you set this option to `true`, the model performs weighted least squares regression, where each sample has a weight or importance associated with it. In this case, there is an extra numeric column after the dependent variable that has the weight for the sample. This option defaults to `false`.

`gamma` — If you set this option, the option must be a matrix. The value represents a Tikhonov gamma matrix used for regularization. For details, see **Tikhonov regularization**. The model uses this option for ridge regression.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

© 2026 Ocient, Inc.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. This option defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a multiple linear regression model. Collect metrics for the model execution by setting the `metrics` option to `true`.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL my_model
TYPE MULTIPLE LINEAR REGRESSION ON (
  SELECT
    x1,
    x2,
    x3,
    y
  FROM public.my_table
)
options(
  'metrics' -> 'true'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the independent variables are provided to the model function execution, and the function returns the estimate of the dependent variable.

SQL

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.multiple_linear_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Vector Autoregression

Model Type: `VECTOR AUTOREGRESSION`

Vector auto-regression is a model that estimates the next value of multiple variables based on some number of lags of all variables, as a group. The model tries to build the

following:

```
Estimate <x1(t), x2(t)> based on x1(t-1), x2(t-1), x1(t-2), and x2(t-2)
```

In this example, `x1(t)` means the value of `x1` at time `t`, and `x1(t-1)` means the value of `x1` at time `t-1` (typically the previous sample time). The syntax `<x1(t), x2(t)>` demonstrates that the result of the model is a row vector that contains all the predictions of the model and that all predictions rely on all lags of all variables.

When you create a model, the input result set must have one more column than the number of lags. Each column must be a row vector of a size equal to the number of variables. The first column is the un-lagged values, e.g., `{{x1, x2, x3}}`. The second column is the first lag for all variables, e.g., `{{x1_lag1, x2_lag2, x3_lag3}}`.

For functions to help organize data before training a model, see **Data Preparation**.

Model Options

Required

`numVariables` — Specify this option as a positive integer for the number of variables in the model.

`numLags` — Specify this option as a positive integer for the number of lags in the model.

Optional

`metrics` — If you set this option to true, the function collects the metric for the coefficient of determination (r^2). This option defaults to false.

`threshold` — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any of them are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

© 2026 Ocient, Inc.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. This option defaults to `true`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a vector autoregression model with three variables and four lags on each one. Collect metrics for the model execution by setting the `metrics` option to `true`.

The example uses the **LAG_VECTORS** function to generate lagged columns of matrices for the vector autoregression model.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL my_model
TYPE VECTOR AUTOREGRESSION ON (
  SELECT
    {{x1, x2, x3}},
    LAG_VECTORS(x1, x2, x3, 1, 4) OVER(ORDER BY t)
  FROM public.my_table
)
OPTIONS(
  'metrics' -> 'true',
  'numVariables' -> '3',
  'numLags' -> '4'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the number of arguments you specify must be equal to the number of lags. Each of those arguments must be a lag vector that contains lagged values for all model variables at a specific lag period. The first argument to the model function (`lag_vector_1`) represents lag 1, the second argument (`lag_vector_2`) represents lag 2, and so on. The LAG_VECTORS function automatically generates these lag vectors with the appropriate structure.

In this example, lag 1 represents values from the previous row when ordered by time `t` , which means the model predicts the current values based on historical patterns from 1, 2, 3, and 4 rows back in the time-ordered sequence.

SQL

```
SELECT my_model(lag_vector_1, lag_vector_2, lag_vector_3, lag_vector_4) AS predicte
    {{x1, x2, x3}} AS actual
FROM (
    SELECT x1, x2, x3,
        LAG_VECTORS(x1, x2, x3, 1, 4) OVER(ORDER BY t)
    FROM my_table
);
```

© 2026 Ocient, Inc.

After you execute a model, you can find the details about the execution results in the `sys.vector_autoregression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Polynomial Regression

Model Type: `POLYNOMIAL REGRESSION`

Polynomial regression means that there is one to many independent variables and one dependent variable, and that should be modeled in terms of an nth-degree polynomial of the independent variables.

The `order` option must be set to a positive integer value that specifies the degree of the polynomial to use. If you specify the option `negativePowers` and set it to `true`, then the model also includes terms with negative exponents, but still with the restriction that the sum of the absolute value of the exponents in a term will be less than or equal to the value specified on the `order` option. Regardless of whether you use the `negativePowers` option, the model computes a coefficient for every possible term that meets this restriction. When you use the `negativePowers` option, the model contains many more terms. For example, a quadratic model over two independent variables has six terms, but when you use the `negativePowers` option, the model has 13 terms.

The result set you specify as input to the model has N columns, which must all be numeric. The first N-1 columns are the independent variables (it can be considered a single independent variable that is a vector). The last column is the dependent variable. The model finds the least squares best fit of a sum of all possible combinations of terms where the degree is less than or equal to the value of the order option. For example, with two independent variables (`x1` and `x2`) and order set to 2, the model is `y = a1*x1^2 + a2*x2^2 + a3*x1*x2 + a4*x1 + a5*x2 + b`.

Model Options

Required

`order` — This option is the degree of the polynomial and must be set to a positive integer.

Optional

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (r^2), the adjusted coefficient of determination, and the root mean squared error (RMSE). This option defaults to `false`.

`threshold` — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any of them are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

`weighted` — If you set this option to `true`, the model performs weighted least squares regression, where each sample has an associated weight. When weighted, there is an extra numeric column after the dependent variable that has the weight for the sample. This option defaults to `false`.

`negativePowers` — If you set this option to `true`, the model includes independent variables raised to negative powers. These variables are called Laurent polynomials. The model generates all possible terms such that the sum of the absolute value of the power of each term in each product is less than or equal to the order. For example, with two independent variables and the order set to 2, the model is: `y = a1*x1^2 + a2*x1^-2 + a3*x2^2 + a4*x2^-2 + a5*x1*x2 + a6*x1^-1*x2 + a7*x1*x2^-1 + a8*x1^-1*x2^-1 + a9*x1 + a10*x1^-1 + a11*x2 + a12*x2^-1 + b`.

`gamma` — If you specify this option, the value must be a matrix. The value represents a Tikhonov gamma matrix that is used for regularization. For details, see **Tikhonov regularization**.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. This option defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input

column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a polynomial regression model of degree 3. Collect metrics for the model execution by setting the `metrics` option to `true`.

SQL

```
CREATE MLMODEL my_model
TYPE POLYNOMIAL REGRESSION ON (
  SELECT
    x1,
    x2,
    x3,
    y
  FROM public.my_table
)
options(
  'order' -> '3',
  'metrics' -> 'true'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the independent variables are provided to the model function execution, and the function returns the estimate of the dependent variable.

```
SQL

SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.polynomial_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Linear Combination Regression

Model Type: `LINEAR COMBINATION REGRESSION`

Linear combination regression is a model built on top of m independent variables and a single dependent variable. But in this case, the function used to perform least-squares regression is a linear combination of functions that you specify. The general form is `y = c0 + c1 * f1(x1, x2, ...) + f2(x1, x2, ...) + ...`.

The model determines the number of independent variables based on the number of columns in the SQL statement on which the model is based. There is always a column for the dependent variable, and there can be a weight column (see the `weighted` option). So, the number of independent variables is either one or two less than the number of columns in the result of the input SQL statement. The number of user-specified functions for the model must be specified by defining `function1`, `function2`, and so on (keys in the options dictionary). As long as consecutive function key names exist, the model includes these names. The model always includes a constant term. The value strings for the `functionN` keys must be specified in SQL syntax and should refer to `x1, x2, ...` for the model input independent variables.

The result set you specify as input to the model has N columns, which must all be numeric. The first N-1 columns are independent variables (it can be considered a single independent variable that is a vector). The last column is the dependent variable. The model finds the least squares best fit for a model of the form `y = a1 * f1(x1, x2, ... xn) + a2 * f2(x1, x2, ... xn) + ... + an * fn(x1, x2, ... nx)`, where `f1, f2, ..., fn` are functions that are provided in a required option.

> The system validates the user-specified functions during prediction time. If the function definition is invalid, the model throws an error during this time.

Model Options

Required

`functionN` — You must specify the first function (`f1`) using a key named `'function1'`. Subsequent functions must use keys with names that use subsequent values of N. You must specify functions in SQL syntax and should use the variables `x1, x2, ..., xn` to refer to the 1st, 2nd, and nth independent variables, respectively. For example, `'function1' -> 'sin(x1 * x2 + x3)', 'function2' -> 'cos(x1 * x3)'`.

Optional

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (r^2), the adjusted coefficient of determination, and the root mean squared error (RMSE). This option defaults to `false`.

`threshold` — This option enables soft thresholding. If you specify this option, the option must be a positive numeric value. After the model calculates the coefficients, if any coefficients are greater than the threshold value, the model subtracts the threshold value from the coefficients. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to the coefficients. For any coefficients between the negative and positive threshold values, the model sets the coefficients to zero.

`weighted` — If you set this option to `true`, the model performs weighted least squares regression, where each sample has an associated weight or importance. When weighted, there is an extra numeric column after the dependent variable that represents the weight of the sample. This option defaults to `false`.

© 2026 Ocient, Inc.

`gamma` — If you specify this option, it must be a matrix. The value represents a Tikhonov gamma matrix used for regularization. For details, see **Tikhonov regularization**.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. This option defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a linear combination regression with two functions that exhibit a wave-like pattern.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL my_model
TYPE LINEAR COMBINATION REGRESSION
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'function1' -> 'sin(x1 * x2 + x3)',
  'function2' -> 'cos(x1 * x3)'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model after training, the independent variables are provided to the model function execution, and the function returns the estimate of the dependent variable.

SQL

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.linear_combination_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Nonlinear Regression

Model Type: `NONLINEAR REGRESSION`

Nonlinear regression finds the best-fit parameters of an arbitrary function using an arbitrary loss function. This model type essentially provides direct access to capabilities that both logistic regression and support vector machines rely on. The first `N - 1` columns are numeric independent variables, and the last column is the numeric dependent variable.

> If you specify a function or loss function with an invalid value, the model throws an error from the `SELECT` statements at training time.

Model Options

Required

`numParameters` — Specify this option as a positive integer. This value specifies the number of different parameters to optimize, i.e., how many different `aN` variables there are in the user-specified function.

`function` — Specify the name of the function to fit the data in SQL syntax. Use `a1, a2, …` to refer to the parameters for optimization. Use `x1, x2, …` to refer to the input features. The model does not allow some SQL functions. The model allows only scalar expressions that can be represented internally as postfix expressions. Most notably, the model does not allow some functions that are rewritten as CASE statements (like `least()` and `greatest()`). If your function is not allowed, the model displays an error message.

This model does not perform auto-scaling.

Optional

`metrics` — If you set this option to true, the model calculates the coefficient of determination (r^2), the adjusted r^2, and the root mean squared error (RMSE). However, the model calculates these quality metrics using the least squares loss function, and not the user-specified loss function, because these metrics really only make sense for least squares.

`lossFunction` — If you specify this option, this parameter dictates the nonlinear optimizer the loss function uses on a per-sample basis. Then, the actual loss function is the sum of this function applied to all samples. The model should use the variable `y` to refer to the dependent variable in the training data and the variable `f` to refer to the

computed estimate for a specified sample. The default is least squares, which could be specified as `(f-y)*(f-y)`.

`minInitParamValue` — A DOUBLE value setting the lower bound for random initial parameter values. The default value is -1.

`maxInitParamValue` — A DOUBLE value setting the upper bound for random initial parameter values. The default value is 1.

`ridgeCoefficient` — If you specify this option, the value must be a `double` data type. This option is the ridge coefficient for the loss function. The default behavior is that the function ignores this option, effectively setting this option to `0.0`.

`lassoCoefficient` — If you specify this option, the value must be a `double` data type. This option is the lasso coefficient for the loss function. The default behavior is that the function ignores this option, effectively setting this option to `0.0`.

`samplesPerThread` — If you specify this option, the value must be a positive integer. This parameter controls the target number of samples sent to each thread. Each thread independently computes a logistic regression model, and the models are all combined at the end. The option defaults to 1 million.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`finiteDifferenceH` — A `DOUBLE` value representing the step size (h) for approximating gradients using the finite difference method. The model uses this value only if analytical gradients are not active.

This value should generally be a small positive value, typically from 0.0001 (1e-4) to 0.0000001 (1e-7).

If you do not specify this value, the default is 0.00001 (1e-5).

`gradientClipThreshold` — A `DOUBLE` value representing the absolute value threshold for any calculated gradient component to be clipped (i.e., capped at the threshold value with

its original sign). This threshold helps prevent issues with exploding gradients in unstable loss landscapes.

Set this value to 0 or a negative value to disable gradient clipping.

If you do not specify this value, the default is `ie6`.

`learningRate` — A `DOUBLE` value representing the base learning rate for the Adam (Adaptive Moment Estimation) machine learning optimizer. Adam adapts this rate individually for each parameter during training. A common starting point for Adam is 0.001 (1e-3). Valid values must be positive and are generally in the range of 0.00001 (1e-5) to 0.01 (1e-2).

A higher learning rate (e.g., 0.01) can speed up training but can cause the optimizer to overshoot and miss optimal solutions, while a lower learning rate (e.g., 0.00001) ensures more stable and precise convergence but can make training much slower.

If you do not specify this option, the system automatically selects a learning rate and adjust it over the course of training using the 1Cycle learning schedule. Specifying a learning rate disables automatic adjustment and instead uses a fixed learning rate value.

`miniBatchSize` — An `INTEGER` value representing the number of data samples the model processes together before updating any model parameters. This value must be positive.

A higher value can speed up computation and efficiently use hardware, but it can require more memory, slow down learning, or reduce the ability of the model to generalize. A lower value allows for more frequent updates and can help the model generalize better, but it can make training noisier and less stable.

If you do not specify this value, the default is 16.

`numEpochs` — A positive INTEGER for the maximum number of training epochs (full passes over the data). The default is 200 for Adam optimization or 100 for Levenberg-Marquardt. Training can typically stop earlier than this threshold because of automatic early stopping.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model

executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a nonlinear regression that fits five parameters `a1, a2, a3, a4, a5` with two independent variables `x1, x2`.

SQL

```
CREATE MLMODEL my_model
TYPE NONLINEAR REGRESSION
ON (
  SELECT
    x1,
    x2,
    y1
  FROM public.my_table
)
options(
  'numParameters' -> '5',
  'function' -> 'a1 * sin(a2 * x1 + a3) + a4 + a5 * x2'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When the model executes, pass `N - 1` independent variables, and the model returns the estimate of the dependent variable.

```SQL
SELECT my_model(x1, x2) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.nonlinear_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Regression Tree

Model Type: `REGRESSION TREE`

Regression Tree is a supervised machine learning algorithm that predicts continuous numerical values by constructing a tree-like model, similar to how Decision Tree models work for classification. For details, see **Decision Tree (Classification)**.

A Regression Tree outputs continuous target values by learning simple decision rules inferred from the input features. The algorithm recursively splits the data set based on feature values that minimize prediction error (typically measured by mean squared or absolute error). Each internal node represents a decision based on a feature value, each branch represents the outcome of that decision, and each leaf node contains a predicted numerical value, typically the mean of the target values in that region.

The Regression Tree model is particularly effective for problems involving numerical prediction, such as price forecasting, demand estimation, or any scenario where the target variable is continuous rather than categorical.

Model Options

Optional

`continuousFeatures` — A set of INTEGER values specifying which input feature columns should be treated as continuous numeric variables for optimal tree splitting. The value must be a comma-separated list of feature column indexes (starting with 1) that correspond to numeric columns in your SELECT statement, excluding the target column. For example, `'continuousFeatures' -> '1,2'` would treat the first and second columns

of your SELECT statement as continuous. If not specified, all features are treated as categorical, which can impair results for numeric columns like prices, measurements, or scores.

`distinctCountLimit` — If you set this option, the value must be a positive integer. This value sets the limit for the number of distinct values a non-continuous feature and the label can contain. This option defaults to 256.

`maxCellsToFetch` — If you set this option, the value must be a positive integer. Controls the chunking behavior when fetching feature values during model training. The limit represents the maximum number of data cells (calculated as number of columns × number of rows) that can be fetched in a single operation, not a byte limit. When the expected data size exceeds this threshold, the algorithm switches to database-based processing using SQL queries instead of in-memory processing. This value defaults to 33,554,432 cells (calculated as 32 * 1024 * 1024).

`maxDepth` — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree (the maximum number of features to split on). The default is unspecified, which means there is no maximum depth.

`maxRows` — If you set this option, the value must be a positive integer. This option limits the number of rows used for model training by creating a snapshot table with only the specified number of rows from the input query. This option cannot be used with `noSnapshot -> true` (attempting to set both results in an `invalid argument` error during model creation). When this option is unspecified, the model trains using all rows from the input query.

`maxThreads` — If you set this option, the value must be a positive integer. This value indicates the maximum number of parallel threads to use while the model trains. The default value is 2.

`metrics` — If you set this option to `true`, the model also calculates the percentage of samples correctly classified by the model and saves this information in a catalog table. This option defaults to `false`.

`noSnapshot` — If you set this option to `true`, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training. This option defaults to `false`. In this case, the database creates and uses the intermediate table. Setting this option to `true` is useful when the training set is

fixed. If the training set is a table with modifications, set this option to `false`, as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to `false` because there is no guarantee that executing that query twice generates the same 100 rows each time.

`numSplits` — If you set this option, the value must be an integer greater than 1. This value sets the maximum number of binary branches a continuous feature can consider. The default value is 32.

`skipLimitCheck` — If you set this option to `true`, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one that is specified by the `distinctCountLimit` option. This option defaults to `false`.

`enableResplits` — A Boolean value that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This can capture more complex, range-specific relationships. If unspecified, this value defaults to true, meaning continuous features remain available for additional splits after use, allowing the tree to create more complex decision boundaries. When set to false, the model marks continuous features as exhausted after their first use, and they cannot be used again in subsequent splits in the same tree.

`resplitDepth` — An INTEGER value that sets the maximum depth at which tree nodes can be re-split during optimization. Controls how deep the algorithm searches for better split points. If unspecified, this value defaults to 6.

`resplitThreshold` — A DECIMAL value that sets the minimum improvement threshold required to trigger a re-split operation. Lower values (e.g. 0.01) allow more aggressive re-splitting but can increase training time. Higher values (e.g. 1.0) require larger improvements to trigger re-splits. If unspecified, this value defaults to 0.1.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

© 2026 Ocient, Inc.

`queryInternalParallelism` — If you set this option, the database appends the `USING PARALLELISM = <value>` clause to all intermediate SQL queries the model executes during training, where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `true`, the database prevents the deletion of any intermediate tables that it creates during model training. This option defaults to `false`.

Execute the Model

This example creates a regression tree model using three input features (`feature1`, `feature2`, `feature3`) to predict a continuous target variable. The model builds a single tree structure that performs predictions by following a series of binary decisions based on feature values.

The model uses these options:

- `'maxDepth' -> '5'` — Limits the tree to a maximum depth of 5 levels, controlling model complexity and preventing overfitting. A depth of 5 provides a good balance between model expressiveness and generalization capability.
- `'enableResplits' -> 'true'` — Allows the algorithm to reuse the same continuous feature multiple times along a single branch path with different threshold values.

SQL

```
CREATE MLMODEL my_reg_tree
TYPE REGRESSION TREE
ON (
    SELECT feature1, feature2, feature3, target
    FROM training_data
)
OPTIONS (
    'maxDepth' -> '5',
    'enableResplits' -> 'true'
);
```

Call the trained model in a `SELECT` query, passing in the feature columns as arguments.

```sql
SELECT my_reg_tree(feature1, feature2, feature3)
FROM scoring_data;
```

For the output of the model, see the `sys.regression_tree_models` system catalog table.

Gradient Boosted Trees

Model Type: `GRADIENT BOOSTED TREES`

Gradient Boosted Trees (GBT) is an ensemble machine learning algorithm that builds a sequence of decision trees where each new tree is trained to correct the prediction errors of the previous trees. Unlike the Random Forest model, which creates trees independently in parallel, GBT uses a sequential approach based on gradient descent optimization, where each tree learns to predict the residual errors (gradients) of the current ensemble. This iterative error-correction process allows the model to capture complex, non-linear patterns by progressively refining predictions through multiple weak learners.

The algorithm supports both regression and classification tasks through different loss functions (use the `lossFunction` option to toggle whether the model performs regression or classification tasks). For regression with squared error loss, trees directly predict residual errors, while classification with logistic loss maintains raw scores transformed through sigmoid or softmax functions.

Model Options

Required

`numChildren` — An INTEGER value representing the total number of trees to build sequentially. Each tree learns to correct the errors of the previous trees.

`learningRate` — A DECIMAL value between 0.0 and 1.0 that tunes how much the model learns from each successive child.

Optional

`lossFunction` — A string value that determines how the model calculates prediction errors and what type of problem it solves. Accepted values include:

- `'squared_error'` — Configures the model for regression tasks. Calculates errors as the squared difference between predicted and actual values. Use this for predicting continuous numeric values (e.g., prices, temperatures, quantities). The target column must contain numeric values. This is the default value.
- `'log_loss'` — Configures the model for classification tasks. This model uses logistic loss to calculate prediction errors for probability-based predictions.

`fractionSelected` — A DECIMAL value greater than 0.0 and less than or equal to 1.0 that specifies what fraction of the training rows to randomly select for each boosting iteration. When you specify a value, the algorithm uses `CEIL(fractionSelected * totalRows)` per iteration. If you do not specify this value, the model uses all rows in their original order.

`inputsPerChild` — An INTEGER greater than or equal to 1 that specifies the number of input features each boosting tree should use. This value cannot exceed the number of input features available in the data set. When you specify this value, the algorithm deterministically cycles through pre-enumerated feature subsets to ensure each tree uses exactly this many features. When you do not specify this value, the model uses all available features for each tree.

`maxDepth` — A positive INTEGER value that represents the maximum allowable depth of the decision trees. The default value is 3.

`maxThreads` — A positive INTEGER value that sets the maximum number of parallel threads to use for training each child decision tree. Parallel threads do not affect the sequential method of training each tree. The default value is 16.

`enableResplits` — A Boolean value that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This action can capture more complex, range-specific relationships. If you do not specify this value, the default value is `true`, meaning continuous features remain available for additional splits after use, which allows the tree to create more complex decision boundaries. When you set this value to `false`, the model marks continuous features as exhausted after their first use, and the model cannot use these features again in subsequent splits in the same tree.

`resplitDepth` — An INTEGER value that sets the maximum depth at which tree nodes can be re-split during optimization. Controls how deep the algorithm searches for better split points. If you do not specify this value, the default value is 6.

© 2026 Ocient, Inc.

`resplitThreshold` — A DECIMAL value that sets the minimum improvement threshold required to trigger a re-split operation. Lower values (e.g., 0.01) allow more aggressive re-splitting but can increase training time. Higher values (e.g., 1.0) require larger improvements to trigger re-splits. If you do not specify this value, the default value is 0.1.

`maxCellsToFetch` — An INTEGER value that determines the memory threshold to switch from training with system memory to training with SQL queries in the database. In-memory training is generally faster, but is limited by the available SQL Node memory. If the size of a training data subset exceeds this value, then the system performs training operations using SQL queries. The default value is 33,554,432 (calculated as `32 * 1024 * 1024`).

`metrics` — A Boolean value. If you set this value to `true`, the system calculates and stores final model metrics (`R²/RMSE` for regression or `Accuracy/LogLoss` for classification) on the training data.

`featureArray` — A Boolean value. If you set this value to `true`, the model expects all input features to be in a single ARRAY column instead of separate columns.

`continuousFeatures` — A set of INTEGER values that specify the input feature columns that the model should treat as continuous numeric variables for optimal tree splitting. The value must be a comma-separated list of feature column indexes (starting with 1) that correspond to numeric columns in your `SELECT` SQL statement, excluding the target column. For example, `'continuousFeatures' -> '1,2'` would treat the first and second columns of your `SELECT` statement as continuous. If you do not specify this value, the model treats all features as categorical, which can impair results for numeric columns like prices, measurements, or scores.

Execute the Model

This example creates a GBT model to predict house prices using a sequential ensemble approach. The model trains on four input features (`square_feet`, `bedrooms`, `bathrooms`, `age_years`) to predict the target variable (`price`), filtering out any records with missing price data to ensure clean training.

The model uses these options:

- `'numChildren' -> '100'` — Builds a sequence of 100 regression trees. Each tree learns to correct the prediction errors made by all previous trees in the sequence.

This relatively high number allows the model to capture complex patterns but requires careful tuning of the learning rate to prevent overfitting.

- `'learningRate' -> '0.1'` — Sets a conservative learning rate where each new tree contribution is scaled by 10 percent before being added to the ensemble. This moderate rate balances training speed with model stability.
- `'lossFunction' -> 'squared_error'` — Configures the model for regression by using squared error loss, which measures prediction accuracy as the squared difference between predicted and actual prices. This specification is the standard choice for continuous numeric predictions, ensuring the model optimizes for minimizing price prediction errors.

SQL

```
CREATE MLMODEL house_price_gbt
TYPE GRADIENT BOOSTED TREES
ON (
    SELECT square_feet, bedrooms, bathrooms, age_years, price
    FROM housing_data
    WHERE price IS NOT NULL
)
OPTIONS (
    'numChildren' -> '100',
    'learningRate' -> '0.1',
    'lossFunction' -> 'squared_error'
);
```

Call the trained model in a `SELECT` query, passing in the feature columns as arguments.

SQL

```
SELECT house_id,
    house_price_gbt(square_feet, bedrooms, bathrooms, age_years) AS predicted_price
FROM new_listings
WHERE square_feet IS NOT NULL
    AND bedrooms IS NOT NULL
    AND bathrooms IS NOT NULL
    AND age_years IS NOT NULL;
```

© 2026 Ocient, Inc.

For the output of the model, see the `sys.gradient_boosted_trees_models` system catalog table.

Related Links

Machine Learning in Ocient

Classification Models

Machine Learning Models

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Classification Models

OcientML™ supports classification models that involve understanding and grouping large data sets into preset categories or subpopulations. With the help of pre-classified training data sets, machine learning classification models leverage various algorithms to classify future data sets into respective and relevant categories.

To create the model, use the `CREATE MLMODEL` syntax. For details, see **CREATE MLMODEL**.

> Model option names are case-sensitive.

K-Nearest Neighbors Classification

Model Type: `K NEAREST NEIGHBORS`

K-nearest neighbors (KNN) is a classification algorithm, where the first `N - 1` inputs are the features, which must be numeric. The last input column is a label, which can be any data type.

There is no training step for KNN. Instead, when you create the model, the model saves a copy of all input data to a table, so that when the model is executed in a later SQL statement, a snapshot of the data the model is supposed to use is available. You can override both the weight function and the distance function.

> The system validates the `weight` and `distance` options during prediction time. If the values are invalid, the model throws an error during this time.

Model Options

Required

`k` — This option must be a positive integer that specifies how many closest points to use for classifying a new point.

© 2026 Ocient, Inc.

Optional

`distance` — If you specify this option, the value must be a function in SQL syntax for calculating the distance between a point used for classification and points in the training data set. This function should use the variables x1, x2, ... for the 1st, 2nd, ... features in the training data set, and p1, p2, ... for the features in the point for classification. If you do not specify this option, the option defaults to the Euclidean distance.

`metrics` — If you set this option to `true`, the model calculates the percentage of samples that are correctly classified by the model and saves this information in a catalog table. The default value is `false`.

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. The default value is `true`.

`weight` — If you specify this option, the value must be a function in SQL syntax for calculating the weight of a neighbor. The function should use the variable d for distance. By default, the distance is set to 1.0/(d+0.1), thus avoiding division by zero on exact inputs and still allowing neighbors to have some influence.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

© 2026 Ocient, Inc.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a KNN model with `8` closest points for classification and distance function `power(x1 - p1, 2) + power(x2 - p2, 2) + power(x3 - p3, 2)`.

SQL

```
CREATE MLMODEL my_model
TYPE K NEAREST NEIGHBORS
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'k' -> '8',
  'distance' -> 'power(x1 - p1, 2) + power(x2 - p2, 2) + power(x3 - p3, 2)'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, the model executes with `N - 1` features as input and returns a label. The model chooses the label from the class with the highest score. The model scores classes by summing the weights from the nearest k points in the training data.

```sql
SELECT my_model(x1, x2, x3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.k_nearest_neighbors_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Naive Bayes Classification

Model Type: `NAIVE BAYES`

Naive Bayes is a classification algorithm. The input is `N - 1` feature columns, and the last column is a label column. All columns can be any data type. The label column must be discrete. The feature columns can be discrete or continuous. When you use continuous feature columns, you must specify which columns are continuous (see options).

Naive Bayes works by assuming that all features are equally important in the classification and that there is no correlation between features. With those assumptions, the algorithm computes all frequency information and saves it in three tables that you create using SQL `SELECT` statements.

Model Options

Optional

`metrics` — If you set this option to `true`, the model calculates the percentage of samples correctly classified by the model and saves this information in a catalog table. This option defaults to `false`.

`continuousFeatures` — If you specify this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start at 1.

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. Defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a Naive Bayes model with feature indexes `1,3`.

SQL

```
CREATE MLMODEL my_model
TYPE NAIVE BAYES
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'continuousFeatures' -> '1,3'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, you specify `N - 1` feature input arguments, and the model returns the most likely class. The returned class is based on computing the class with the highest probability, given prior knowledge of the feature values. In other words, the class `y` has the highest value of `P(y | x1, x2, …, xn)`.

SQL

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.naive_bayes_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Decision Tree

Model Type: `DECISION TREE`

The decision tree is a classification model. The first `N - 1` input columns are features and can be any data type. All non-numeric features must be discrete and contain no more than the configured `distinctCountLimit` number of unique values. This limit is in place to prevent the internal model representation from growing too large. Numeric features are discrete by default and have the same limitation on the number of unique values, but they can be marked as continuous with the `continuousFeatures` option. For continuous features, the model builds the decision tree by dividing the values into two ranges instead of using discrete, unique values. The last input column is the label and can be any data type.

When you create the model, you specify all features first, and then specify the label as the last column in the result set.

You can use secondary indexes on discrete feature columns to greatly speed up the training of a decision tree model.

Ocient® supports a similar tree model for regression. For details, see **Regression Tree**.

Model Options

Optional

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`distinctCountLimit` — If you set this option, the value must be a positive integer. This value sets the limit for how many distinct values a non-continuous feature and the label can contain. This option defaults to 256.

`doPrune` — If you set this option to `true`, the model uses Pessimistic Error Pruning (PEP) to prune the tree after training. This option defaults to `false`.

`featureArrayElements` — If you set this option, the value must be a comma-separated list of the feature indexes. If you set the `featureArray` option to `true`, this list determines the elements of the arrays to use for training. By default, the model uses all elements.

© 2026 Ocient, Inc.

`maxCellsToFetch` — If you set this option, the value must be a positive integer. Controls the chunking behavior when fetching feature values during model training. The limit represents the maximum number of data cells (calculated as number of columns × number of rows) that can be fetched in a single operation, not a byte limit. When the expected data size exceeds this threshold, the algorithm switches to database-based processing using SQL queries instead of in-memory processing. This value defaults to 33,554,432 cells (calculated as 32 * 1024 * 1024).

`maxDepth` — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree (the maximum number of features to split on). The default is unspecified, which means there is no maximum depth.

`maxRows` — If you set this option, the value must be a positive integer. This option limits the number of rows used for model training by creating a snapshot table with only the specified number of rows from the input query. This option cannot be used with `noSnapshot -> true` (attempting to set both results in an `invalid argument` error during model creation). When this option is unspecified, the model trains using all rows from the input query.

`maxThreads` — If you set this option, the value must be a positive integer. This value indicates the maximum number of parallel threads to use while the model trains. The default value is 2.

`metrics` — If you set this option to `true`, the model also calculates the percentage of samples correctly classified by the model and saves this information in a catalog table. This option defaults to `false`.

`noSnapshot` — If you set this option to `true`, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training. This option defaults to `false`. In this case, the database creates and uses the intermediate table. Setting this option to `true` is useful when the training set is fixed. If the training set is a table with modifications, set this option to `false` as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to `false` because there is no guarantee that running that query twice generates the same 100 rows each time.

© 2026 Ocient, Inc.

`numSplits` — If you set this option, the value must be an integer greater than 1. This value sets the maximum number of binary branches a continuous feature can consider. The default value is 32.

`ROCNumSamples` — If you set the option, you must also set the `metrics` option. This positive integer indicates the number of samples for the model to use for the area under the ROC curve. The default value is 10.

`skipLimitCheck` — If you set this option to `true`, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one that is specified by the `distinctCountLimit` option. This option defaults to `false`.

`weighted` — If you set this option, the model considers weights for labels. If you set this option value to `true`, you must specify an additional column as a `double` in the training data for label weights. Rows with the same labels must have the same weights. If you set this value to `auto`, the model calculates weights automatically by weighting each label according to the ratio of the count of the most frequent label to the count of the specified label. As a result, the most frequent label has a weight of `1.0`, and the other label weights are higher. This option defaults to `false`, which means all labels have equal weight.

`splitMetric` — Controls which function is used to evaluate the quality of a split during tree construction. Supported options are:

- `gini_impurity` (default) — Measures impurity based on class distributions.
- `weighted_f1` — Uses class-frequency–weighted F1 score to guide splits.

`enableResplits` — A Boolean value that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This can capture more complex, range-specific relationships. If unspecified, this value defaults to true, meaning continuous features remain available for additional splits after use, allowing the tree to create more complex decision boundaries. When set to false, the model marks continuous features as exhausted after their first use, and they cannot be used again in subsequent splits in the same tree.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in

the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a decision tree model with feature indexes `1,3`.

SQL

```
CREATE MLMODEL my_model
TYPE DECISION TREE ON (
  SELECT
    c1,
    c2,
    c3,
    y1
  FROM public.my_table
)
options(
  'continuousFeatures' -> '1,3'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, you must specify the `N - 1` features as parameters. The model returns the expected label.

```
SQL

SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.decision_tree_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Random Forest

Model Type: `RANDOM FOREST`

Random forest is a classification model consisting of multiple decision trees. The result consists of the most common label among the tree results and an array of pairs of labels and their frequencies, sorted in descending order by frequency.

The model breaks ties in various ways depending on the type of label. For strings, the model uses lexicographic order (reverse of the usual alphabetic order), so `C` comes before `A`, for example. For Booleans, the model chooses `true` before `false`. For numeric types, the model chooses the largest number first.

When you call the model to make classification predictions, you can optionally use soft voting by adding an extra Boolean argument to the statement, e.g., `house(x, y, true)`. This extra argument must be a Boolean literal, either `true` (soft voting) or `false` (hard voting). If you do not specify this value, the default value is `false` (hard voting).

In soft voting, each decision tree reports a list of possible results and their confidence factors. The random forest model adds the confidence factors from the decision trees

and normalizes them to add up to 1.0. The model sorts the results by descending confidence, then by descending values.

In hard voting, the model does not account for confidence factors. Each decision tree makes its own class prediction as a vote. The model selects the prediction with the most votes from the trees.

Model Options

Required

`numChildren` — Number of child decision trees.

Optional

`bootstrap` — If you set this option to `true`, the model uses bootstrap sampling with replacement, meaning each tree in the random forest is trained on a random subset of the data (either the `rowsPerChild` or `fractionSelected` value sets the exact number of rows), and the same row can appear multiple times in each tree. If you set this option to `false`, this option does not use replacement, meaning each row can appear at most once per tree. The default value is `false`.

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1.

`distinctCountLimit` — If you set this option, the value must be a positive integer. This value limits how many distinct values a non-continuous feature and the label can contain. The default value is 256.

`featureSubsetStrategy` — If you set this option, the model passes this option directly to the child decision trees to specify how many features each tree should consider at each split from the still-available features. When this value is higher, the model has higher accuracy and lower variance, but it takes longer to train.

You can specify this option either as an integer (e.g., `4`, meaning consider up to four features at each split) or one of three string values:

- `all` — Each tree checks every feature.
- `sqrt` — Each tree checks up to the square root of the total number of features.

© 2026 Ocient, Inc.

- `one-third` — Each tree checks up to one-third of the total number of features.

The default value is `all`.

`fractionSelected` — The proportion of rows the model uses to train each decision tree. The value is a `double` that must be in the interval (0, 1]. You cannot set this option if you also set the `rowsPerChild` option to a positive value. The default behavior is that the model uses all available rows.

`inputsPerChild` — Number of features used to create each child decision tree. The default value is the number of features you specify for the forest divided by 3 and rounded up.

`maxCellsToFetch` — If you set this option, the value must be a positive integer. Controls the chunking behavior when fetching feature values during model training. The limit represents the maximum number of data cells (calculated as number of columns × number of rows) that can be fetched in a single operation, not a byte limit. When the expected data size exceeds this threshold, the algorithm switches to database-based processing using SQL queries instead of in-memory processing. This value defaults to 33,554,432 cells (calculated as 32 * 1024 * 1024).

`maxChildThreads` — An integer representing the maximum number of threads each child decision tree can use. If you do not specify this option, each child decision tree uses at most one thread, the default for decision trees.

`maxDepth` — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree. The default value is 3.

`maxThreads` — The maximum number of parallel threads to use while the model trains decision trees. This value must be a positive integer. The default value is 16.

`metrics` — If you set this option to `true`, the model also calculates the percentage of samples that are correctly classified by the model for the random forest and saves this information in a catalog table. This option is always set to `false` for the child trees.

`noSnapshot` — If you set this option to `true`, the data source must not change. In this case, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training a random forest. Child decision trees always have this option set to `true`, so the database does not create a

separate intermediate table for each decision tree. The default value is `false`. Setting this option to `true` is useful when the training set is fixed. If the training set is a table with modifications, set this option to `false` as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to `false` because there is no guarantee that running that query twice generates the same 100 rows each time.

`requiredFeatures` — A comma-separated list of integers as strings representing specific features where the first feature has the value `1`. The model uses these features in every decision tree in the forest. The default behavior is that the decision tree in the forest can train on any feature that is in the list.

`rowsPerChild` — If you set this option to a positive integer, the number represents the number of rows (from a random sample) to use for each decision tree. If you set this option to 0, each child uses all available rows. The default value is 0. You cannot set this option to a positive value if you also set the `fractionSelected` option.

`skipLimitCheck` — If you set this option to `true`, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one that is specified by the `distinctCountLimit` option. The default value is `false`.

`weighted` — If you set this option, the model considers weights for labels. If you set this option value to `true`, you must specify an additional column as a `double` in the training data for label weights. Rows with the same labels must have the same weights. If you set this value to `auto`, the model calculates weights automatically by weighting each label according to the ratio of the count of the most frequent label to the count of the specified label. As a result, the most frequent label has a weight of `1.0`, and the other label weights are higher. This option defaults to `false`, which means all labels have equal weight.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false` , the database deletes any intermediate tables created during model training. If you set this option to `true` , the database prevents the deletion of any intermediate tables created during model training. The default value is `false` .

Execute the Model

Create a random forest model with four child decision trees and two features randomly chosen by the model for each tree. The `training_table` table contains the training data set. Collect metrics for the model execution by setting the `metrics` option to `true` .

SQL

```
CREATE MLMODEL test_model
TYPE RANDOM FOREST
ON (
  SELECT *
  FROM public.training_table
)
options(
  'numChildren' -> '4',
  'inputsPerChild' -> '2'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` , `sys.machine_learning_model_options` , and `sys.random_forest_models` system catalog tables.

Execute this model using three columns `a`, `b`, and `c`, from `large_table` table that contains the whole data set.

SQL

```
SELECT test_model(a, b, c) FROM large_table;
```

Optionally, you can execute the model using soft voting by including an extra Boolean argument.

SQL

```
SELECT test_model(a, b, c, true) FROM large_table;
```

The output of the model is a tuple containing the most common label as the first element. The second element is an array of tuples, where the first element is the label and the second is the frequency. The model sorts the array by frequency from highest to lowest, where the sum of frequencies is 1. When two labels have the same frequency, the model breaks the tie according to lexicographic order.

To retrieve the most common label, use `[]` with an index of 1, `[1]`, for example, `test_model(a,b,c)[1]`.

After you execute a model, you can find the results in the output of the model function execution.

For details, see the description of the associated system catalog tables in the Machine Learning section in **System Catalog**.

Logistic Regression

Model Type: `LOGISTIC REGRESSION`

This model fits a logistic curve to the data across any number of classes greater than one.

The first `N - 1` inputs are features and must be numeric. Features can be one-hot encoded. The last input column is the class or label. You must have at least one non-NULL label in the result set for model creation. The model best fits the logistic curve using a negative log likelihood loss function. The model uses an algorithm that is a combination of particle swarm optimization, line search, and genetic algorithms to find the best-fit parameters.

Model Options

Optional

`metrics` — If you set this option to `true`, the model calculates the percentage of samples that are correctly classified by the model and saves this information in the `sys.logistic_regression_models` system catalog table. This option defaults to `false`.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. This option defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`numEpochs` — An `INTEGER` value representing the maximum number of epochs, or full passes during training through the entire data set. This value must be positive.

If you do not specify this value, the default is 20.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes

during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a logistic regression model. Collect metrics for the model execution by setting the `metrics` option to `true`.

SQL

```
CREATE MLMODEL my_model
TYPE LOGISTIC REGRESSION
ON (
  SELECT
    x1,
    x2,
    x3,
    y1
  FROM public.my_table
)
options(
  'metrics' -> 'true'
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute this model after training, you must specify the features as input and the label as the output. The label can be any data type.

SQL

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.logistic_regression_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Support Vector Machine

Model Type: `SUPPORT VECTOR MACHINE`

Support Vector Machine (SVM) essentially finds a hypersurface (the hypersurface is a curve in 2-dimensional space) that correctly splits the data into any number of classes greater than one and maximizes the margin around the hypersurface. By default, SVM finds a hyperplane to split the data (the hyperplane is a straight line in 2-dimensional space). SVM uses a hinge loss function to balance the two objectives of finding a hyperplane with a wide margin while minimizing the number of incorrectly classified points.

The first `N - 1` input columns are the features and must be numeric. The last column is the label and can be any arbitrary type. You must have at least one non-NULL label in the result set for model creation.

Model Options

Optional

`metrics` — If you set this option to true, the model also calculates the percentage of samples that are correctly classified by the model and saves this information in a catalog table. This option defaults to false.

`regularizationCoefficient` — If you specify this option, the value must be a valid floating-point number. This option is used to control the balance of finding a wide margin and minimizing incorrectly classified points in the loss function. When this value is larger (and positive), it makes having a wide margin around the hypersurface more important relative to the incorrectly classified points. Because of how Ocient implements SVM, the values for this parameter are likely different from the values used in other common SVM implementations. This option defaults to 1.0 / 1000000.0.

`functionN` — By default, SVM uses a linear kernel. If you use a different kernel, you must provide a list of functions that are summed together, just like with linear combination regression. You must specify the first function using a key named ‘function1’. Subsequent functions must use keys with names that use subsequent values of N. You must specify functions in SQL syntax and use the variables x1, x2, ..., xn to refer to the 1st, 2nd, and nth independent variables, respectively. You can specify the default linear kernel as: ‘function1’ → ‘x1’, ‘function2’ → ‘x2’, and so on. The model always adds a constant term equivalent to ‘functionN’ → ‘1.0’ that you do not need to specify explicitly.

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. Defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`numEpochs` — An `INTEGER` value representing the maximum number of epochs, or full passes during training through the entire data set. This value must be positive.

If you do not specify this value, the default is 200.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the

database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a support vector machine model.

SQL

```
CREATE MLMODEL my_model
TYPE SUPPORT VECTOR MACHINE ON (
  SELECT
    c1,
    c2,
    c3,
    y1
  FROM public.my_table
);
```

After you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, the `N - 1` features must be passed as parameters. The model returns the expected label.

SQL

```
SELECT my_model(col1, col2, col3) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.support_vector_machine_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Gradient Boosted Trees

Model Type: `GRADIENT BOOSTED TREES`

Gradient Boosted Trees (GBT) is an ensemble machine learning algorithm that builds a sequence of decision trees where each new tree is trained to correct the prediction errors of the previous trees. Unlike the Random Forest model, which creates trees independently in parallel, GBT uses a sequential approach based on gradient descent optimization, where each tree learns to predict the residual errors (gradients) of the current ensemble. This iterative error-correction process allows the model to capture complex, non-linear patterns by progressively refining predictions through multiple weak learners.

The algorithm supports both regression and classification tasks through different loss functions (use the `lossFunction` option to toggle whether the model performs regression or classification tasks). For regression with squared error loss, trees directly predict residual errors, while classification with logistic loss maintains raw scores transformed through sigmoid or softmax functions.

Model Options

Required

`numChildren` — An INTEGER value representing the total number of trees to build sequentially. Each tree learns to correct the errors of the previous trees.

`learningRate` — A DECIMAL value between 0.0 and 1.0 that tunes how much the model learns from each successive child.

Optional

`lossFunction` — A string value that determines how the model calculates prediction errors and what type of problem it solves. Accepted values include:

- `'squared_error'` — Configures the model for regression tasks. Calculates errors as the squared difference between predicted and actual values. Use this for predicting continuous numeric values (e.g., prices, temperatures, quantities). The target column must contain numeric values. This is the default value.
- `'log_loss'` — Configures the model for classification tasks. This model uses logistic loss to calculate prediction errors for probability-based predictions.

`fractionSelected` — A DECIMAL value greater than 0.0 and less than or equal to 1.0 that specifies what fraction of the training rows to randomly select for each boosting iteration. When you specify a value, the algorithm uses `CEIL(fractionSelected * totalRows)` per iteration. If you do not specify this value, the model uses all rows in their original order.

`inputsPerChild` — An INTEGER greater than or equal to 1 that specifies the number of input features each boosting tree should use. This value cannot exceed the number of input features available in the data set. When you specify this value, the algorithm deterministically cycles through pre-enumerated feature subsets to ensure each tree uses exactly this many features. When you do not specify this value, the model uses all available features for each tree.

`maxDepth` — A positive INTEGER value that represents the maximum allowable depth of the decision trees. The default value is 3.

`maxThreads` — A positive INTEGER value that sets the maximum number of parallel threads to use for training each child decision tree. Parallel threads do not affect the sequential method of training each tree. The default value is 16.

`enableResplits` — A Boolean value that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This action can capture more complex, range-specific relationships. If you do not specify this value, the default value is `true`, meaning continuous features remain available for additional splits after use, which allows the tree to create more complex decision boundaries. When you set this value to `false`, the model marks continuous features as exhausted after their first use, and the model cannot use these features again in subsequent splits in the same tree.

`resplitDepth` — An INTEGER value that sets the maximum depth at which tree nodes can be re-split during optimization. Controls how deep the algorithm searches for better split points. If you do not specify this value, the default value is 6.

`resplitThreshold` — A DECIMAL value that sets the minimum improvement threshold required to trigger a re-split operation. Lower values (e.g., 0.01) allow more aggressive re-splitting but can increase training time. Higher values (e.g., 1.0) require larger improvements to trigger re-splits. If you do not specify this value, the default value is 0.1.

`maxCellsToFetch` — An INTEGER value that determines the memory threshold to switch from training with system memory to training with SQL queries in the database. In-memory training is generally faster, but is limited by the available SQL Node memory. If the size of a training data subset exceeds this value, then the system performs training operations using SQL queries. The default value is 33,554,432 (calculated as `32 * 1024 * 1024`).

`metrics` — A Boolean value. If you set this value to `true`, the system calculates and stores final model metrics (`R²/RMSE` for regression or `Accuracy/LogLoss` for classification) on the training data.

`featureArray` — A Boolean value. If you set this value to `true`, the model expects all input features to be in a single ARRAY column instead of separate columns.

`continuousFeatures` — A set of INTEGER values that specify the input feature columns that the model should treat as continuous numeric variables for optimal tree splitting. The value must be a comma-separated list of feature column indexes (starting with 1) that correspond to numeric columns in your `SELECT` SQL statement, excluding the target column. For example, `'continuousFeatures' -> '1,2'` would treat the first and second columns of your `SELECT` statement as continuous. If you do not specify this value, the model treats all features as categorical, which can impair results for numeric columns like prices, measurements, or scores.

Execute the Model

This example creates a Gradient Boosted Trees classifier to predict whether a customer stops using a service. The model trains on four input features (`tenure_months`, `monthly_charges`, `total_charges`, `support_tickets`) to predict the target variable (`churn_label`), filtering out any records with missing labels to ensure clean training.

The model uses these options:

- `'numChildren' -> '100'` — Builds a sequence of 100 weak learners (regression trees). Each tree learns to correct the classification residuals (logistic loss) of all previous trees. A higher value can capture complex patterns, and you should pair it with an appropriate learning rate.

- `'learningRate' -> '0.1'` — Scales the contribution of each new tree by 10 percent, balancing convergence speed and generalization. This value represents a small learning rate, making training more conservative and stable.
- `'lossFunction' -> 'log_loss'` — Configures the model for classification through logistic loss. If you do not specify this value, the default value is `'squared_error'` (regression), so explicitly setting `'log_loss'` is mandatory for classifiers.

SQL

```
CREATE MLMODEL churn_gbt_classifier
TYPE GRADIENT BOOSTED TREES
ON (
    SELECT
        tenure_months,
        monthly_charges,
        total_charges,
        support_tickets,
        churn_label
    FROM customer_training
    WHERE churn_label IS NOT NULL
)
OPTIONS (
    'numChildren'   -> '100',
    'learningRate'  -> '0.1',
    'lossFunction'  -> 'log_loss'
);
```

Call the trained model in a `SELECT` query, passing in the feature columns as arguments.

© 2026 Ocient, Inc.

SQL

```
SELECT
    customer_id,
    churn_gbt_classifier(
        tenure_months,
        monthly_charges,
        total_charges,
        support_tickets
        )[1] AS predicted_label,
    churn_gbt_classifier(
        tenure_months,
        monthly_charges,
        total_charges,
        support_tickets
        )[2] AS class_probabilities
FROM customer_scoring
WHERE tenure_months IS NOT NULL
    AND monthly_charges IS NOT NULL
    AND total_charges IS NOT NULL
    AND support_tickets IS NOT NULL;
```

For classification, the model returns a tuple of:

- `element [1]` — The predicted class label
- `element [2]` — An array of (`class_label` , `probability`) pairs

For the output of the model, see the `sys.gradient_boosted_trees_models` system catalog table.

Related Links

Clustering and Dimension Reduction Models

Other Models

Machine Learning Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Clustering and Dimension Reduction Models

OcientML™ supports these machine learning models for clustering and dimension reduction.

To create the model, use the `CREATE MLMODEL` syntax. For details, see **CREATE MLMODEL**.

> Model option names are case-sensitive.

Principal Component Analysis

Model Type: `PRINCIPAL COMPONENT ANALYSIS`

Principal Component Analysis (PCA) is typically not used as a model on its own. PCA is most commonly used on the inputs to other models. PCA serves two purposes.

1. PCA normalizes all numeric feature data. Some types of models are sensitive to the scale of numeric features, and when different features have different scales, the results end up skewed. PCA normalizes all features to the same scale.
2. PCA is used for dimensionality reduction. PCA computes linear combinations of the original features to put the most signal into a smaller number of new features.

The input result set when creating a PCA model is N numeric columns, which are all features. There is no label or dependent variable. After you create a PCA model, the `sys.principal_component_analysis_models` system catalog table contains information on the percentage of the signal in each PCA feature. You can use this information to figure out how many of the output features to keep.

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

© 2026 Ocient, Inc.

Principal Component Analysis and Logistic Regression

You can also use PCA models as inputs to other models. For example, if you have three features and you want to use PCA to reduce the number to two features, you can execute the following SQL statements.

SQL

```
CREATE MLMODEL reduceTo2
TYPE PRINCIPAL COMPONENT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3
  FROM public.my_table
);
```

You can use this model as input for another model, for example, logistic regression.

SQL

```
CREATE MLMODEL binaryClass
TYPE LOGISTIC REGRESSION ON (
  SELECT
    reduceTo2(c1, c2, c3, 1),
    reduceTo2(c1, c2, c3, 2),
    label
  FROM …
);
```

To correctly use this model later, you must pass the original features through the PCA model when you execute the logistic regression model.

© 2026 Ocient, Inc.

```sql
SELECT
  binaryClass(
    reduceTo2(x1, x2, x3, 1),
    reduceTo2(x1, x2, x3, 2)
  )
FROM … ;
```

Similarly, to create a PCA analysis over four variables, execute this SQL statement:

```sql
CREATE MLMODEL my_model
TYPE PRINCIPAL COMPONENT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3,
    c4
  FROM public.my_table
);
```

Model Options

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes

during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

K-Means Clustering

Model Type: `KMEANS`

K-means is an unsupervised clustering algorithm. All of the columns in the input result set are features, and there is no label. All the input columns must be numeric. The algorithm finds `k` points such that all points are classified by the closest `k` points. Distance calculations are Euclidean by default.

Model Options

Required

`k` — This option must be a positive integer. The option specifies the algorithm for how many clusters to make.

Optional

`epsilon` — If you specify this option, the value must be a valid positive floating point value. When the maximum distance that a centroid moves from one iteration of the algorithm to the next is less than this value, the algorithm terminates. This parameter defaults to `1e-4`.

`normalize` — If you set this option to `true`, the model normalizes the data before the start of training. The default value is `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option,

the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a K-means model with `8` clusters.

SQL

```
CREATE MLMODEL my_model
TYPE KMEANS
ON (
  SELECT
    x1,
    x2,
    x3,
    x4
  FROM public.my_table
)
options(
  'k' -> '8'
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

Because there are no labels for clusters, when you execute this function after training with the same number (and same order) of features as input, the result is an integer that specifies the cluster to which the point belongs.

```sql
SELECT my_model(x1, x2, x3, x4) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.k_means_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Gaussian Mixture

Model Type: `GAUSSIAN MIXTURE MODEL`

Gaussian Mixture Models (GMMs) find the mixture of `N` Gaussian distributions that best fit the data. The most common use is unsupervised clustering, but there are some key differences compared to K-means clustering:

- GMMs can handle clusters that are not circular, i.e., clusters in the shape of ovals or higher-dimensional analogs with variances in different directions.
- Clusters can have an arbitrary rotation, i.e., they can have covariance.
- GMMs can handle instances when clusters are not as equally likely. If a point is right between two clusters, it is more likely to be more common in the training data.
- GMMs can show the probability of a new point belonging to each cluster instead of providing only a single cluster value.

© 2026 Ocient, Inc.

GMMs generally handle more complex data than K-means models, but they require more training and processing time during execution.

Model Options

Required

`numDistributions` — Must be a positive integer. This value specifies the number of clusters of Gaussian distributions for the model to make.

Optional

`epsilon` — If you specify this option, the value must be a valid positive floating point value. When the maximum distance that the entire best model moves in its n-dimensional space is less than this value, the algorithm terminates. This parameter defaults to 1e-8 (0.00000001 as a decimal number).

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. Defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the

database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a GMM with three clusters of Gaussian distributions.

SQL

```
CREATE MLMODEL gmm
TYPE GAUSSIAN MIXTURE MODEL
ON (
  SELECT
    x,
    y,
    z
  FROM my_table
)
options(
  'numDistributions' -> '3'
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute this function after training with the same number (and same order) of features as input, the result is an integer that specifies the probable cluster to which the point belongs.

SQL

```
SELECT gmm(x,y,z) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.gaussian_mixture_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Linear Discriminant Analysis

Model Type: `LINEAR DISCRIMINANT ANALYSIS`

Linear Discriminant Analysis (LDA) is a dimension-reduction technique similar to PCA. While PCA is unsupervised, LDA can classify and use class labels to find linear combinations of features that best separate the classes.

The input result set when you create an LDA model is `N` numeric columns that are all features. The last column is a class label and can be any data type. After you create an LDA model, the `sys.linear_discriminant_analysis_models` system catalog table contains information on the percentage of the signal that is in each LDA feature. You can use this information to figure out how many of the output features to keep.

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

Linear Discriminant Analysis and Logistic Regression

You can use LDA models as inputs to other models. For example, to reduce the number of features from three to two using LDA, you can execute these SQL statements.

SQL

```
CREATE MLMODEL reduceTo2
TYPE LINEAR DISCRIMINANT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3,
    label
  FROM public.my_table
);
```

You can use this model as input for another model, for example, logistic regression.

SQL

```
CREATE MLMODEL binaryClass
TYPE LOGISTIC REGRESSION ON (
  SELECT
    reduceTo2(c1, c2, c3, 1),
    reduceTo2(c1, c2, c3, 2),
    label
  FROM …
  );
```

To correctly use this model later, you must pass the original features through the LDA model when you execute the logistic regression model.

SQL

```
SELECT
  binaryClass(
    reduceTo2(x1, x2, x3, 1),
    reduceTo2(x1, x2, x3, 2)
  )
FROM … ;
```

Similarly, to create an LDA model over four variables, execute this SQL statement.

SQL

```
CREATE MLMODEL my_model
TYPE LINEAR DISCRIMINANT ANALYSIS ON (
  SELECT
    c1,
    c2,
    c3,
    c4,
    label
  FROM public.my_table
);
```

© 2026 Ocient, Inc.

Model Options

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. Defaults to `true`.

`featureArray` — If you set this option to `true`, the model expects only one array-type column as input instead of multiple columns of training data. Each array row in the input column must be of the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

When you execute the trained LDA model, you must provide the same original input features in the same order, followed by a positive integer argument that specifies the LDA component to return. The LDA component index starts at 1.

© 2026 Ocient, Inc.

SQL

```sql
SELECT
  my_model(col1, col2, col3, col4, 2) as component2,
  my_model(col1, col2, col3, col4, 3) as component3,
FROM public.my_table;
```

Bibliography

Bahmani, Bahman, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. “Scalable K-Means++.” Proceedings of the VLDB Endowment 5, no. 7 (2012): 622–33. **https://doi.org/10.14778/2180912.2180915**.

Related Links

Classification Models

Other Models

Machine Learning Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ensemble Models

Ensemble models improve predictive performance by combining the strengths of multiple individual machine learning models into a single, robust system. Instead of relying on one algorithm, ensemble techniques train a collection of base models and aggregate their predictions to reduce errors, lower variance, and prevent overfitting.

Bagging

Model Type: `BAGGING`

A bagging model trains multiple base models in parallel on random subsets of the training data and aggregates their predictions. Use the bagging model to reduce variance and prevent overfitting.

The bagging model utilizes child models embedded as options in JSON format (see `baseModels`).

The bagging model trains all child models independently in parallel. When aggregating all the child model predictions, the bagging model uses majority voting for classification and averaging for regression.

> A bagging model can use any model type that supports classification or regression.

Model Options

Required

`baseModels` — A JSON array defining the child models to train. Each object in the array specifies a model type, a count (how many of this model to create), and any options specific to the individual child model.

`baseModels` JSON Arguments

© 2026 Ocient, Inc.

Argument	Data Type	Description
`type`	string	The OcientML™ model type (e.g., `"DECISION TREE"`, `"MULTIPLE LINEAR REGRESSION"`). The selected model type should comply with the selected `taskType` (i.e., select a classification model if your `taskType` is `CLASSIFICATION`.
`count`	integer	The number of instances of this model type to train.
`options`	JSON object	JSON object containing options specific to that child model type, exactly as if you were creating that model directly. For a list of supported options, see the descriptions of model types.

`taskType` — Specifies the task type with values: `'CLASSIFICATION'` or `'REGRESSION'`.

Optional

`maxThreads` — An optional integer value that controls the maximum number of child models that the bagging model trains in parallel. This option does not affect the internal threading of each child. If supported, the `maxThreads` value of each child model controls the internal threading. The default value is 16.

`bootstrap` — An optional Boolean value. If you set this option to `true`, the model uses bootstrap sampling with replacement. Each child trains on a random sample where rows can be repeated. Requires you to set the `noSnapshot` option to `false`. If you set this option to `false`, the model uses sampling without replacement (each row appears at most one time per child). The default value is `false`.

`rowsPerChild` — An optional integer specifying the exact number of rows to use for each child model. `0` means use all available rows (only valid if you do not set the `fractionSelected` option). The default value is `0`.

`fractionSelected` — An optional double value between 0.0 and 1.0 that specifies the proportion of rows to use for each child model. This option is unusable if you set the `rowsPerChild` option to greater than 0. The default value is `1.0` (use all rows).

`inputsPerChild` — An optional integer that specifies the number of features (columns) to use for each child model. The default value is `CEIL(total_features / 3)`.

© 2026 Ocient, Inc.

`metrics` — An optional Boolean value. If you set this option to `true`, the model calculates and stores quality metrics on the training data after training completes. The default value is `false`.

For classification, the quality metrics include:

- Accuracy: The percentage of training rows where the model predicted the correct class.
- Area Under the Curve (AUC): A score from 0 to 1 that measures how well the model distinguishes between two different categories.

For regression, the quality metrics include:

- Root Mean Squared Error (RMSE): The square root of the average squared difference between the predicted and actual values.
- Adjusted R²**:** The coefficient of determination, adjusted for the number of predictors (features) in the model. This metric indicates how well the independent variables explain the variance in the dependent variable.

The default value is `false`.

`featureArray` — An optional Boolean value. If you set this option to `true`, the model expects only one array-type column as input, rather than multiple columns of training data. Each array row in the input column must be the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

Execute the Model

This example creates a bagging model that, in effect, functions as a random forest by training 50 decision trees for a classification task.

SQL

```
CREATE MLMODEL my_schema.bagging_classifier
TYPE BAGGING
OPTIONS (
    'taskType' = 'CLASSIFICATION',
    'baseModels' = '[
        {
            "type": "DECISION TREE",
            "count": 50,
            "options": {
                "maxDepth": 8,
                "splitMetric": "gini_impurity"
            }
        }
    ]',
    'bootstrap' = 'true',
    'fractionSelected' = '0.8',
    'maxThreads' = '16',
    'metrics' = 'true'
)
AS
SELECT
    feature1,
    feature2,
    feature3,
    label
FROM my_schema.training_data;
```

After the training completes, execute the bagging model.

SQL

```
SELECT
    my_schema.bagging_classifier(feature1, feature2, feature3)
FROM my_schema.scoring_data;
```

Boosting

Model Type: `BOOSTING`

A boosting model is an ensemble technique that trains multiple base models sequentially, where each new model attempts to correct the errors of the combined previous models. Boosting reduces bias and errors in supervised learning tasks.

The boosting model utilizes child models specified as options in JSON format (see the `baseModels` argument). The boosting model trains its child models one after another, with each child contributing to the final prediction, weighted by a set learning rate.

> A boosting model can use any model type that supports regression.

Model Options

Required

`baseModels` — A JSON array describing the child models to train. Each object in the array specifies a model type, the number of instances of this model to create, and other options that depend on the model type.

`baseModels` JSON Arguments

Argument	Data Type	Description
`type`	string	The OcientML model type (e.g., `"DECISION TREE"`, `"MULTIPLE LINEAR REGRESSION"`). The selected model type should comply with the selected `taskType` (i.e., select a classification model if your `taskType` is `CLASSIFICATION`.
`count`	integer	The number of instances of this model type to train sequentially.
`options`	JSON object	JSON object containing options specific to that child model type, similar to creating that model directly. For a list of supported options of a child model, see the individual descriptions of model types.

`taskType` — Specifies the task type with values: `'CLASSIFICATION'` or `'REGRESSION'`.

`learningRate` — A decimal value between 0.0 and 1.0 that tunes how much the model learns from each successive child. Lower values can lead to better generalization, but they require more trees (a higher `count` value in the `baseModels` JSON field).

© 2026 Ocient, Inc.

Optional

`lossFunction` — A string value that determines how the model calculates prediction errors and what type of problem it solves. Accepted values are:

- `'squared_error'` — Configures the model for regression tasks. Calculates errors as the squared difference between predicted and actual values. Use this value for predicting continuous numeric values (e.g., prices, temperatures, quantities). The target column must contain numeric values. The `lossFunction` option defaults to `squared_error` if you set the `taskType` option to `REGRESSION`.
- `'log_loss'` — Configures the model for classification tasks. This model uses logistic loss to calculate prediction errors for probability-based predictions. The `lossFunction` option defaults to `log_loss` if you set the `taskType` option to `CLASSIFICATION`.

`fractionSelected` — A double value specifying the proportion of rows (0.0 < n <= 1.0) to use for each child model. The default value is `1.0` (use all rows).

`inputsPerChild` — An integer value specifying the number of features (columns) to use for each child model. When you set this option, the algorithm deterministically cycles through feature subsets. If you do not set this option, the model uses all available features for each child.

`maxThreads` — An optional integer value that controls the maximum number of child models that the model trains in parallel. This option does not affect the internal threading of each child. If supported, the `maxThreads` value of each child model controls the internal threading. The default value is 16.

`metrics` — An optional Boolean value. If you set this option to `true`, this option calculates and stores quality metrics on the training data after training completes. The default value is `false`.

For classification, the quality metrics include:

- Accuracy: The percentage of training rows where the model predicted the correct class.
- Area Under the Curve (AUC): A score from 0 to 1 that measures how well the model distinguishes between two different categories.

For regression, the quality metrics include:

- Root Mean Squared Error (RMSE): The square root of the average squared difference between the predicted and actual values.
- Adjusted R²**:** The coefficient of determination, adjusted for the number of predictors (features) in the model. This metric indicates how well the independent variables explain the variance in the dependent variable.

`skipDropTable` — An optional Boolean value. If you set this option to `false`, the database deletes intermediate tables created during training. If you set this option to `true`, the database prevents deletion of intermediate tables (useful for debugging). The default value is `false`.

`continuousFeatures` — An optional comma-separated list of feature indexes (1-based) that are continuous numeric variables (i.e., not categorical or discrete variables).

`featureArray` — An optional Boolean value. If you set this option to `true`, the model expects only one array-type column as input, rather than multiple columns of training data. Each array row in the input column must be the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

Execute the Model

This example creates a boosting model (similar to Gradient Boosted Trees but using generic boosting logic) that trains 100 decision trees sequentially for a regression task.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL my_schema.boosting_regressor
TYPE BOOSTING
OPTIONS (
    'taskType' = 'REGRESSION',
    'baseModels' = '[
        {
            "type": "DECISION TREE",
            "count": 100,
            "options": {
                "maxDepth": 4
            }
        }
    ]',
    'learningRate' = '0.1',
    'lossFunction' = 'squared_error',
    'metrics' = 'true'
)
AS
SELECT
    feature1,
    feature2,
    feature3,
    label
FROM my_schema.training_data;
```

After training completes, execute the boosting model.

SQL

```
SELECT
    my_schema.boosting_regressor(feature1, feature2, feature3)
FROM my_schema.scoring_data;
```

Stacking

Model Type: `STACKING`

The stacking model combines multiple base models into a single, higher-accuracy ensemble. A stacking model can use any other machine learning model that supports

regression, classification, or clustering.

A stacking model can produce better accuracy than a single model by incorporating the strengths of different models and feature subsets.

Stacking trains two model levels:

- Level-zero models: one or more base models that train on the original features.
- Level-one model (meta-model): a single model that trains on the predictions of the level-zero models (plus any preserved features), and learns how to weight and combine them.

> A stacking model cannot use the Vector Autoregression, Feedforward Neural Network, or Association Rules models as level-zero or level-one models.

Model Options

Required

The stacking model utilizes other models embedded as options in JSON format (see `levelZeroModels` and `levelOneModel`). Both are required.

`levelZeroModels` — An array of JSON objects representing one or more base models. Internally, the system maps these models using the same strings as other machine learning model types. Any stacked models have the same features, requirements, and options as they would without stacking.

`levelOneModel` — A JSON object that defines the single meta-model that sits on top of all level-zero models in a stacking ensemble. The object specifies the model type, name, and any model-specific options for this level-one model, which is trained on the outputs of the level-zero models (and any preserved input columns) to produce the final prediction.

JSON strings for both `levelZeroModels` and `levelOneModel` share these arguments unless noted otherwise in the description.

© 2026 Ocient, Inc.

Argument	Data Type	Description
`type`	string	The machine learning model type to use for this child (for example, `"RANDOM FOREST"`, `"LOGISTIC REGRESSION"`, `"KMEANS"`). Must be a valid OcientML model type that supports regression, classification, or clustering.
`name`	string	Optional. A name for the child model. If you omit this argument, the system auto-generates a name by attaching a suffix to the parent model name.
`options`	JSON object	JSON object containing options specific to the chosen model type, exactly as if you were creating that model directly. For a list of options, see the individual descriptions of individual model types.
`ignoreColumns`	string	Optional. Used only for `levelZeroModels`. The value is a comma-separated list of 1-based column indices to exclude from this level-0 model training input. If you omit this argument, the model sees all available features, labels, and extra columns.
`extraCallArguments`	string	Advanced optional argument. An array of argument lists. Each element is an array of strings representing extra literal arguments appended when the system executes this child model in the level-one training SQL. Level-zero child models can use any number of argument lists. Level-one child models can use only one argument list.

Optional

`maxThreads` — An optional integer value that controls the maximum number of level-0 child models that the stacking model trains in parallel. This option does not affect the internal threading of each child. If supported, the `maxThreads` value of each child model controls the internal threading. A higher `maxThreads` value can reduce total training time for many level-0 models, but increases concurrent resource usage on the cluster. The default value is 16.

`noSnapshot` — An optional Boolean value that controls whether the stacking trainer first materializes the training query into an intermediate snapshot table. When you set the `noSnapshot` option to `false`, the model creates a temporary snapshot table from the input SQL and trains all level-0 and level-1 models against that fixed data set, ensuring consistent data even if the source changes. When you set the `noSnapshot` option to

`true`, the trainer reads the input query directly without creating a snapshot (which can be faster but requires the underlying data not to change).

If the query is a common table expression (i.e., it starts with a `WITH` clause), then stacking automatically forces the `noSnapshot` option value to `false` and logs a warning.

The default value is `false`.

`featureArray` — An optional Boolean value. If you set this option to `true`, the model expects only one array-type column as input, rather than multiple columns of training data. Each array row in the input column must be the same size. Regardless of whether you use this option, the model treats training data the same with labeling and weight scoring. The default value is `false`.

`hasLabelColumn` — Specifies whether the training data includes a label column (necessary for supervised tasks like classification or regression). If you set this option to `true`, the input query must return features followed by a label column. If you set this option to `false`, the model assumes the input contains only features (typically for unsupervised tasks, such as clustering or dimension reduction models). The default value is `true`.

`extraColumnCount` — A non-negative integer representing the number of columns present after the label column (e.g., for weights). Level-zero models do not use these columns as standard features, but you can pass these columns to the level-one model using the `extraCallArguments` argument logic. The default value is `0`.

`preservedColumnsForLevelOne` — A list of integers representing column indices (1-based) from the input data. The stacking model passes these specified columns to the level-one model. This option is useful if the level-one model needs to access raw features to improve the base model predictions.

If you do not specify this option, the default value is an empty list (i.e., no original features are preserved).

Execute the Model

This example demonstrates how a stacking model can embed level-zero and level-one child models. The JSON `OPTIONS` string includes two level-zero models (random forest

and gradient boosted trees) and produces the final prediction using the level-one model (logistic regression). Each models specifies its own options for how to run.

In the example, the random forest model (`rf_base`) uses 200 decision trees. This model explicitly ignores columns 3 and 4, training only on columns 1 and 2. In contrast, the gradient boosted trees model (`gbt_base`) uses 300 decision trees and all available columns.

The level-one model uses logistic regression to combine the outputs of the two base models. The model takes the predictions from `rf_base` and `gbt_base` as its inputs and produces the final classification result.

When you run the stacking model, the stacking model first executes the base models in parallel (up to the `maxThreads` value). The resulting predictions then go into the logistic regression meta-model to generate the final score.

© 2026 Ocient, Inc.

SQL

```
CREATE MLMODEL my_schema.stacking_classifier
TYPE STACKING
OPTIONS (
    -- Level-0 (Base) Models
    'levelZeroModels' = '[
        {
            "type": "RANDOM FOREST",
            "name": "rf_base",
            "options": {
                "numTrees": 200,
                "maxDepth": 12
            },
            "ignoreColumns": "3,4"
        },
        {
            "type": "GRADIENT BOOSTED TREES",
            "name": "gbt_base",
            "options": {
                "numChildren": 300,
                "learningRate": 0.05
            }
        }
    ]',

    -- Level-1 (Meta) Model
    'levelOneModel' = '{
        "type": "LOGISTIC REGRESSION",
        "name": "stacking_meta",
        "options": {
            "maxIterations": 200,
            "regularization": 0.1
        }
    }',

    -- Top-level STACKING options
    'maxThreads' = '16',
    'hasLabelColumn' = 'true'
)
AS
SELECT
    feature1,
    feature2,
```

```
    feature3,
    feature4,
    label
FROM my_schema.training_data;
```

After training completes, execute the stacking model.

SQL

```
SELECT
    my_schema.stacking_classifier(feature1, feature2, feature3, feature4)
FROM my_schema.scoring_data;
```

Related Links

Regression Models

Classification Models

Clustering and Dimension Reduction Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Other Models

OcientML™ supports other machine learning models for advanced analysis that include association rules and feedforward neural network.

To create the model, use the `CREATE MLMODEL` syntax. For details, see **CREATE MLMODEL**.

> Model option names are case-sensitive.

Association Rules

Model Type: `ASSOCIATION RULES`

The association rules model trains itself over rows of arrays to suggest other associated values. The model returns values commonly associated in a set that are absent in the provided value set. In practical use, this model is often used to help in retail transactions by suggesting related items or services for purchase based on similar past transactions.

Model Options

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false` , the database deletes any intermediate tables created during model training. If you set this option to `true` , the

database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create an association rules model. When you create the model, there should be a single input column, an array of any inner type.

SQL

```
CREATE MLMODEL my_model
TYPE ASSOCIATION RULES ON (
  SELECT
    array_agg(product)
  FROM public.my_table
  GROUP BY customer_id
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model, you must provide an array with the same inner type as the input column. The array represents the current state of the transaction.

You must also provide a second argument that is a positive integer that represents the association ranking of the item to return. The value of `1` indicates to add the first most-associated item to the transaction.

The third and fourth arguments are optional constant Boolean values.

If the third argument is set to `true`, the model adds the most associated item to the transaction based on other transactions that have the same items in the current transaction. Otherwise, only some of the items must be the same. The third argument defaults to `true`.

If the fourth argument is set to `true`, the model counts duplicate items in a transaction only once. Otherwise, the model counts duplicate items multiple times. To

specify the fourth argument, you must specify the third argument. The default value for the fourth argument is `true`, which means that duplicate items are counted one time.

SQL

```
SELECT my_model(array['sausage', 'mushrooms'], 1);
SELECT my_model(array['sausage', 'mushrooms'], 1, true, true);
```

The output from the model is a tuple `( v, p, c )`. The definitions of these tuple values are:

`v` — The associated item determined by the model based on your input arguments.

`p` — The proportion of arrays in the training data that contain the items of the input array and the associated item `v`, as compared to arrays in the training data that have the input items but not necessarily the associated item `v`.

`c` — The count of occurrences of the associated item `v` in the training data arrays based on your input arguments.

After you execute a model, you can find the details about the execution results in the `sys.association_rules_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Feedforward Neural Network

Model Type: `FEEDFORWARD NETWORK`

The feedforward network model is a neural network model where data moves from the inputs through hidden layers to the outputs. Feedforward neural networks are fully connected (sometimes named multilayer perceptrons). The number of inputs is determined by the number of columns in the input result set. Each input must be numeric. The last column in the input result set is the target variable. For models with one output, the column is also numeric. For models with multiple outputs, the result must be a 1xN matrix (a row vector).

Common uses of multiple output models are:

- Multi-class classification — Multiple outputs are one-hot encoded values that represent the class of the record. The model uses results with argmax to select the highest probability class.
- Probability modeling — Multiple output values represent probabilities between 0 and 1 that sum to 1.
- Multiple numeric prediction — Multiple output values represent different numeric values to predict against.

Model Options

Required

`hiddenLayers` — You must set this option to a positive integer that specifies how many hidden layers to use.

`hiddenLayerSize` — You must set this option to a positive integer that specifies the number of nodes in each hidden layer.

`outputs` — You must set this option to a positive integer that specifies the number of outputs.

`lossFunction` — This option specifies the loss function that all hidden layer nodes and all output layer nodes use. This function can be one of several predefined loss functions or a user-defined loss function.

The predefined loss functions are `squared_error` (regression), `vector_squared_error` (vector-valued regression), `log_loss` (binary classification with target values of 0 and 1), `logits_loss` (binary classification with target values of 0 and 1), `hinge_loss` (binary classification with target values of -1 and 1), and `cross_entropy_loss` (multi-class classification). If the value for this required parameter is none of these strings, the model assumes a user-defined loss function. The user-defined loss function specifies the per-sample loss. Then, the actual loss function is the sum of this function applied to all samples. The model should use the variable `y` to refer to the dependent variable in the training data, and the model should use the variable `f` to refer to the computed estimate for a specified sample.

© 2026 Ocient, Inc.

Optional

`activationFunction` — If you set this option, the values are `linear`, `relu` (rectified linear unit), `leakyrelu` (leaky rectified linear unit), `tanh` (hyperbolic tangent function), or `sigmoid` (fast sigmoid approximation). This option defaults to `relu`. This option affects all layers except the output layer.

`outputActivationFunction` — If you set this option, the values are `linear`, `relu` (rectified linear unit), `leakyrelu` (leaky rectified linear unit), `tanh` (hyperbolic tangent function), or `sigmoid` (fast sigmoid approximation). Different activation functions have different output ranges. The chosen activation function should match the dependent variable of your data. For example, if the dependent variable can be anything, then choose the `linear` value. If the dependent variable is always positive, then choose the `relu` value. If your outputs range from -1 to 1 or you perform hinge loss classification, `tanh` is a good option because the hyperbolic tangent function has the same range. But, if your outputs range from 0 to 1 or you perform log loss classification, `sigmoid` is a better choice for the same reason. This option defaults to `linear`. The option only sets the activation function for the output layer.

`metrics` — If you set this option to `true`, the model calculates the average value of the loss function.

`useSoftMax` — If you set this option to true, the model applies a softmax function to the output of the output layer before computing the loss function. This option defaults to `true` if the `lossFunction` is set to `cross_entropy_loss` and `false` otherwise.

`minInitParamValue` — If you set this option, the value must be a floating-point number. Sets the minimum for initial parameter values in the optimization algorithm. This option defaults to -1.

`maxInitParamValue` — If you set this option, the value must be a floating-point number. Sets the maximum for initial parameter values in the optimization algorithm. This option defaults to 1.

`normalize` — If you set this option to `true`, this option applies z-score normalization to inputs by default, storing means and standard deviation, and automatically applying them at inference. This option defaults to `true`.

© 2026 Ocient, Inc.

`finiteDifferenceH` — A `DOUBLE` value representing the step size (h) for approximating gradients using the finite difference method. The model uses this value only if analytical gradients are not active.

This value should generally be a small positive value, typically from 0.0001 (1e-4) to 0.0000001 (1e-7).

If you do not specify this value, the default is 0.00001 (1e-5).

`gradientClipThreshold` — A `DOUBLE` value representing the absolute value threshold for any calculated gradient component to be clipped (i.e., capped at the threshold value with its original sign). This threshold helps prevent issues with exploding gradients in unstable loss landscapes.

Set this value to 0 or a negative value to disable gradient clipping.

If you do not specify this value, the default is `ie6`.

`learningRate` — A `DOUBLE` value representing the base learning rate for the Adam (Adaptive Moment Estimation) machine learning optimizer. Adam adapts this rate individually for each parameter during training. A common starting point for Adam is 0.001 (1e-3). Valid values must be positive and are generally in the range of 0.00001 (1e-5) to 0.01 (1e-2).

A higher learning rate (e.g., 0.01) can speed up training but can cause the optimizer to overshoot and miss optimal solutions, while a lower learning rate (e.g., 0.00001) ensures more stable and precise convergence but can make training much slower.

If you do not specify this option, the system automatically selects a learning rate and adjust it over the course of training using the 1Cycle learning schedule. Specifying a learning rate disables automatic adjustment and instead uses a fixed learning rate value.

`numEpochs` — An `INTEGER` value representing the maximum number of epochs, or full passes during training through the entire data set. This value must be positive.

If you do not specify this value, the default is 200.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model

executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

Execute the Model

Create a neural network to perform multi-class classification for three possible classes. `y1`, `y2`, and `y3` are one-hot encoded outputs. If the values are 1 and the rest are 0, the value 1 denotes the class that the training data belongs to from the three classes.

SQL

```
CREATE MLMODEL my_model
TYPE FEEDFORWARD NETWORK
ON (
  SELECT
    x1,
    x2,
    {{y1,y2,y3}}
  FROM public.my_table
)
options(
  'hiddenLayers' -> '1',
  'hiddenLayerSize' -> '8',
  'outputs' -> '3',
  'lossFunction' -> 'cross_entropy_loss',
  'activationFunction' -> 'relu',
  'useSoftmax' -> 'true'
);
```

Also, after you create the model, you can see its details by querying the `sys.machine_learning_models` and `sys.machine_learning_model_options` system catalog tables.

When you execute the model later, pass `N - 1` input variables, and the model returns the estimate of the target variable. In the case of multiple outputs, the result is a 1xN matrix (a row vector). If the model uses multiple outputs to perform multi-class classification, use `argmax` to get the integer that represents the class.

```sql
SELECT argmax(my_model(x1, x2)) FROM my_table;
```

After you execute a model, you can find the details about the execution results in the `sys.feedforward_network_models` system catalog table.

For details, see the description of the associated system catalog tables in the Machine Learning section in the **System Catalog**.

Related Links

Machine Learning Model Functions

Machine Learning Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Machine Learning Model Options

This list of options contains model options for all models in the Ocient® System.

Association Rules

Model Options

Optional

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

Bagging

Model Options

Required

`baseModels` — This option specifies the children of the bagging model. You must specify this value as a JSON array, where each object in the array has the three fields `type` , `count` , and `options` .

`taskType` — This option specifies the type of task, which must be either `CLASSIFICATION` or `REGRESSION` depending on the type of model for training.

Optional

`ROCNumSamples` — If you set this option, you must specify a positive integer that represents the number of samples for the model to use when calculating the area under

© 2026 Ocient, Inc.

the ROC curve. You must also set the `metrics` option to `true`. The default value is the number of child models.

`bootstrap` — If you set this option to `true`, the model uses bootstrap sampling with replacement, meaning each child model trains on a random subset of the data (either the `rowsPerChild` or `fractionSelected` value sets the exact number of rows), and the same row can appear multiple times for each child. If you set this option to `false`, the model does not use replacement, meaning each row can appear at most one time per child. The default value is `false`.

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`fractionSelected` — If you set this option, the option represents the proportion of rows the model uses to train each child model. The value is a `double` that must be in the interval (0, 1]. You cannot set this option if you also set the `rowsPerChild` option to a positive value. The default behavior is that the model uses all available rows.

`inputsPerChild` — If you set this option, the option represents the number of features used to create each child model. The default value is the total number of features divided by 3 and rounded up.

`maxChildThreads` — If you set this option, the value must be an integer representing the maximum number of threads each child model can use. If a child accepts a `maxThreads` option, the model passes this value to the child.

`maxThreads` — If you set this option, the option represents the maximum number of parallel threads to use while the model trains. This value must be a positive integer. The default value is 16.

© 2026 Ocient, Inc.

`metrics` — If you set this option to `true`, the system calculates certain metrics depending on the value of the `taskType` option. If you set the `taskType` option to `CLASSIFICATION`, the metrics are the percentage of correctly classified rows and the area under the ROC curve. If you set the value to `REGRESSION`, the metrics are the root mean square error and the adjusted R-squared. The default value is `false`.

`noSnapshot` — If you set this option to `true`, the data source must not change. In this case, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training a random forest. Child decision trees always have this option set to `true`, so the database does not create a separate intermediate table for each decision tree. The default value is `false`. Setting this option to `true` can speed up training when the training set is fixed.

`requiredFeatures` — If you set this option, the value must be a comma-separated list of integers representing features starting at index 1. The bagging model passes these features down to every child. The default value is an empty list, meaning there is no required feature.

`rowsPerChild` — If you set this option to a positive integer, the number represents the number of rows (from a random sample) to use for each decision tree. If you set this option to 0, each child uses all available rows. The default value is 0. You cannot set this option to a positive value if you also set the `fractionSelected` option.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`weighted` — If you set this option, the system passes the value directly to child models that support it. The behavior depends on the model type of the target child.

Boosting

Model Options

Required

`baseModels` — This option specifies the children of the bagging model. You must specify this as a JSON array, where each object in the array has the three fields `type`, `count`, and `options`.

`learningRate` — A `decimal` value between 0.0 and 1.0 that tunes how much the model learns from each successive child.

`taskType` — This option specifies the type of task, which must be either `CLASSIFICATION` or `REGRESSION` depending on the type of model for training.

Optional

`ROCNumSamples` — If you set this option, you must specify a positive integer that represents the number of samples for the model to use when calculating the area under the ROC curve. You must also set the `metrics` option to `true`. The default value is `10`.

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`fractionSelected` — If you set this option, the option represents the proportion of rows the model uses to train each child model. The value is a `double` that must be in the interval (0, 1]. You cannot set this option if you also set the `rowsPerChild` option to a positive value. The default behavior is that the model uses all available rows.

`inputsPerChild` — If you set this option, the value must be an `integer` type that is greater than or equal to 1, which specifies the number of input features each boosting child should use. This value cannot exceed the number of input features available in the

data set. When you specify this value, the algorithm deterministically cycles through pre-enumerated feature subsets to ensure each child uses exactly the specified number of features. When you do not specify this value, the model uses all available features for each child.

`lossFunction` — If you set this option, the value represents the loss function used by the model. Accepted values are: `'squared_error'` and `'log_loss'`. When you set this value to `'squared_error'`, the model calculates errors as the squared difference between predicted and actual values. The target column must contain numeric values. This is the default value when the `taskType` option is set to `REGRESSION`. When you set this value to `'log_loss'`, the model calculates errors using logistic loss. This is the default value when the `taskType` option is `CLASSIFICATION`.

`maxThreads` — If you set this option, the option represents the maximum number of parallel threads to use while the model trains. This value must be a positive integer. The default value is 16.

`metrics` — If you set this option to `true`, the system calculates certain metrics depending on the value of the `taskType` option. If the `taskType` option is set to `CLASSIFICATION`, the metrics are the percentage of correctly classified rows and the area under the ROC curve. If the `taskType` option is set to `REGRESSION`, the metrics are the root mean square error and the adjusted R-squared. The default value is `false`.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Decision Tree

Model Options

Optional

`ROCNumSamples` — If you set this option, you must specify a positive integer that represents the number of samples for the model to use when calculating the area under the ROC curve. You must also set the `metrics` option to `true`. The default value is `10`.

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`distinctCountLimit` — If you set this option, the value must be a positive `integer` type. This value sets the limit for how many distinct values a non-continuous feature and label can contain. The default value is `256`.

`doPrune` — If you set this option to `true`, the model uses Pessimistic Error Pruning (PEP) to prune the tree after training. The default value is `false`.

`enableResplits` — If you set this option, it must be a `boolean` type that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This action can capture more complex, range-specific relationships. The default value is `true`, meaning that continuous features remain available for additional splits after use, thereby allowing the tree to create more complex decision boundaries. If you set this option to `false`, the model marks continuous features as exhausted after their first use, and the model cannot use them again in subsequent splits in the same tree.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`featureSubsetStrategy` — If you set this option, the option specifies how many features the decision tree should consider at each split from the still-available features. When this value is higher, the model has a higher accuracy and lower variance, but takes longer to train. You can specify this option either as an integer (e.g., `4`, meaning consider up to four features at each split) or one of these values: `all` (check every

feature), `sqrt` (check up to the square root of the number of total features), and `one-third` (check up to one-third of the number of total features). The default value is `all`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`maxCellsToFetch` — If you set this option, the value must be a positive integer. Controls the chunking behavior when fetching feature values during model training. The limit represents the maximum number of data cells (calculated as number of columns × number of rows) that can be fetched in a single operation, not a byte limit. When the expected data size exceeds this threshold, the algorithm switches to database-based processing using SQL queries instead of in-memory processing. This value defaults to 33,554,432 cells (calculated as 32 × 1024 × 1024).

`maxDepth` — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree (the maximum number of features to split on). The default is unspecified, which means there is no maximum depth.

`maxRows` — If you set this option, the value must be a positive integer. This option limits the number of rows used for model training by creating a snapshot table with only the specified number of rows from the input query. This option cannot be used with `noSnapshot -> true` (attempting to set both results in an `invalid argument` error during model creation). When this option is unspecified, the model trains using all rows from the input query.

`maxThreads` — If you set this option, the value must be a positive integer. This value indicates the maximum number of parallel threads to use while the model trains. The default value is 2.

`metrics` — If you set this option to `true`, the model also calculates the percentage of samples correctly classified by the model and saves this information in a catalog table. This option defaults to `false`.

`noSnapshot` — If you set this option to `true`, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training. This option defaults to `false`. In this case, the database

creates and uses the intermediate table. Setting this option to `true` is useful when the training set is fixed. If the training set is a table with modifications, set this option to `false` as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to `false` because there is no guarantee that running that query twice generates the same 100 rows each time.

`numSplits` — If you set this option, the value must be an integer greater than 1. This value sets the maximum number of binary branches a continuous feature can consider. The default value is 32.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`skipLimitCheck` — If you set this option to `true`, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one specified by the `distinctCountLimit` option. The default value is `false`.

`splitMetric` — If you set this option, the option controls which function the model uses to evaluate the quality of a split during tree construction. Supported options are: `gini_impurity` (measures impurity based on class distributions), `macro_f1` (uses the macro-averaged F1 score to guide splits), `micro_f1` (uses the micro-averaged F1 score to guide splits), and `weighted_f1` (uses the class-frequency-weighted F1 score to guide splits). The default value is `gini_impurity`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`suppressJIT` — If you set this option to `true`, the model suppresses just-in-time code generation.

`weighted` — If you set this option, the model considers weights for labels. If you set this option value to `true`, you must specify an additional column as a `double` in the training data for label weights. Rows with the same labels must have the same weights. If you set this value to `auto`, the model calculates weights automatically by weighting each label according to the ratio of the count of the most frequent label to the count of the specified label. As a result, the most frequent label has the weight `1.0` and the other label weights are higher. The default value is `false`, which means all labels have equal weight.

Feedforward Network

Model Options

Required

`hiddenLayerSize` — You must set this option to a positive `integer` type that specifies the number of nodes in each hidden layer.

`hiddenLayers` — You must set this option to a positive `integer` type that specifies how many hidden layers to use.

`lossFunction` — This option specifies the loss function that all hidden layer nodes and all output layer nodes use. This function can be one of several predefined loss functions or a user-defined loss function. The predefined loss functions are `squared_error` (regression), `vector_squared_error` (vector-valued regression), `log_loss` (binary classification with target values of 0 and 1), `logits_loss` (binary classification with target values of 0 and 1), `hinge_loss` (binary classification with target values of -1 and 1), and `cross_entropy_loss` (multi-class classification). If the value for this required option is none of these strings, the model assumes a user-defined loss function. The user-defined loss function specifies the per-sample loss. Then, the actual loss function is the sum of this function applied to all samples. The model should use the variable `y` to refer to the dependent variable in the training data, and the model should use the variable `f` to refer to the computed estimate for the specified sample.

`outputs` — You must set this option to a positive integer that specifies the number of outputs.

Optional

`activationFunction` — If you set this option, the values are `linear`, `relu` (rectified linear unit), `leakyrelu` (leaky rectified linear unit), `tanh` (hyperbolic tangent function), or `sigmoid` (fast sigmoid approximation). The default value is `relu`. This option affects all layers except the output layer.

`adamBeta1` — If you set this option, the option represents the value of β_1 in the Adam optimization algorithm. For higher values of this option, training is less noisy but takes longer to converge. The default value is `0.9`.

`adamBeta2` — If you set this option, the option represents the value of β_2 in the Adam optimization algorithm. For higher values of this option, training is less noisy but takes longer to converge. The default value is `0.99`.

`adamEpsilon` — If you set this option, the option represents the value of ε in the Adam optimization algorithm. For higher values of this option, training is more numerically stable but takes longer to converge. The default value is `1e-7`.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`finiteDifferenceH` — If you set this option, the value must be a `double` representing the step size (`h`) for approximating gradients using the finite difference method. The model uses this value only if analytical gradients are not active. This value should generally be a small positive value, typically from `0.0001` (`1e-4`) to `0.0000001` (`1e-7`). The default value is `0.00001` (`1e-5`).

`gradientClipThreshold` — If you set this option, the value must be a `double` that represents the gradient norm threshold for clipping. When the overall gradient norm exceeds this threshold, the system scales all gradient components uniformly to preserve direction. This operation prevents issues with exploding gradients in unstable loss landscapes. Set this value to 0 or a negative value to disable gradient clipping. The default value is `1000000` (`1e6`).

© 2026 Ocient, Inc.

`learningRate` — If you set this option, the value must be a `double` type representing the base learning rate for the Adam (Adaptive Moment Estimation) machine learning optimizer. Adam adapts this rate individually for each parameter during training. A common starting point for Adam is `0.001` (`1e-3`). Valid values must be positive and are generally in the range of `0.00001` (`1e-5`) to `0.01` (`1e-2`). A higher learning rate can speed up training, but can cause the optimizer to overshoot and miss optimal solutions. Conversely, a lower learning rate ensures more stable and precise convergence but can make training much slower. If you do not specify this option, the system automatically selects a learning rate and adjusts it during training using the 1Cycle learning schedule. Specifying a learning rate disables automatic adjustment and instead uses a fixed learning rate value.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`maxInitParamValue` — If you set this option, the value must be a floating-point number. Sets the maximum for initial parameter values in the optimization algorithm. The default value is `1` .

`metrics` — If you set this option to `true` , the model calculates the average value of the loss function.

`minInitParamValue` — If you set this option, the value must be a floating-point number. Sets the minimum for initial parameter values in the optimization algorithm. The default value is `-1` .

`normalize` — If you set this option to `true` , this option applies z-score normalization to inputs by default, storing means and standard deviation, and automatically applying them at inference. The default value is `true` .

`numEpochs` — If you set this option, the value must be a positive `integer` type representing the maximum number of epochs, or full passes, during training through the entire data set. If you do not specify this option, the default maximum is `200` , but training typically stops earlier due to automatic early stopping when the model has converged.

© 2026 Ocient, Inc.

`outputActivationFunction` — If you set this option, the values are `linear`, `relu` (rectified linear unit), `leakyrelu` (leaky rectified linear unit), `tanh` (hyperbolic tangent function), or `sigmoid` (fast sigmoid approximation). Different activation functions have different output ranges. The chosen activation function should match the dependent variable of your data. For example, if the dependent variable can be anything, then choose the `linear` value. If the dependent variable is always positive, then choose the `relu` value. If your outputs range from -1 to 1 or you perform hinge loss classification, `tanh` is a good option because the hyperbolic tangent function has the same range. But, if your outputs range from 0 to 1 or you perform log loss classification, `sigmoid` is a better choice for the same reason. This option defaults to `linear`. The option only sets the activation function for the output layer.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`randomSeed` — If you set this option, the value must be a positive `integer` type representing the seed for the random number generator the system uses for weight initialization. Setting this option makes model training deterministic (given the same data and options). If you do not specify this option or set it to `0`, the system uses a non-deterministic random seed.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`useSoftmax` — If you set this option to `true`, the model applies a softmax function to the output of the output layer before computing the loss function. The default value is `true` if you set the `lossFunction` to `cross_entropy_loss`, and `false` otherwise.

Gaussian Mixture Model

Model Options

Required

`numDistributions` — This option must be a positive `integer` type that specifies the number of clusters of Gaussian distributions for the model to make.

Optional

`epsilon` — If you specify this option, the value must be a valid positive floating point number. When the maximum distance that the entire best model moves in its n-dimensional space is less than this value, the algorithm terminates. The default value is `0.00000001` (`1e-8`).

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`maxIterations` — If you set this option, the option represents the maximum number of optimization iterations to train the model. For higher values of this option, the model is likelier to converge to the expected `epsilon`, but it might take longer to train. The default value is `100`.

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. Defaults to `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes

© 2026 Ocient, Inc.

during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Gradient Boosted Trees

Model Options

Required

`learningRate` — A `decimal` value between 0.0 and 1.0 that tunes how much the model learns from each successive child.

`numChildren` — An `integer` value representing the total number of trees to build sequentially. Each tree learns to correct the errors of the previous trees.

Optional

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`enableResplits` — If you set this option, the value must be a `boolean` type that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This action can capture more complex, range-specific relationships. The default value is `true`, meaning that continuous features remain available for additional splits after use, thereby allowing the tree to create more complex decision boundaries. If you set this option to `false`, the model marks continuous features as exhausted after their first use, and the model cannot use them again in subsequent splits in the same tree.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

© 2026 Ocient, Inc.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`fractionSelected` — If you set this option, the option represents the proportion of rows the model uses to train each child model. The value is a `double` that must be in the interval (0, 1]. You cannot set this option if you also set the `rowsPerChild` option to a positive value. The default behavior is that the model uses all available rows.

`inputsPerChild` — If you set this option, the value must be an `integer` type greater than or equal to 1 that specifies the number of input features each boosting tree should use. This value cannot exceed the number of input features available in the data set. When you specify this value, the algorithm deterministically cycles through pre-enumerated feature subsets to ensure each tree uses exactly the specified number of features. The default behavior is that the model uses all available features for each tree.

`lossFunction` — If you set this option, the option represents the loss function used and determines the type of task the model does. Accepted values are: `'squared_error'` and `'log_loss'`. When you set this value to `'squared_error'`, the model calculates errors as the squared difference between predicted and actual values. The target column must contain numeric values. This is the default value for regression tasks. When you set this value to `'log_loss'`, the model calculates errors using logistic loss. This is the default value for classification tasks.

`maxCellsToFetch` — If you set this value, the value must be an `integer` type that determines the memory threshold to switch from training with system memory to training with SQL queries in the database. In-memory training is generally faster, but is limited by the available SQL Node memory. If the size of a training data subset exceeds this value, then the system performs training operations using SQL queries. The default value is `33,554,432` (calculated as `32 * 1024 * 1024`).

`maxDepth` — If you set this value, the value must be a positive `integer` type that represents the maximum allowable depth of the child trees. The default value is `3`.

`maxThreads` — If you set this value, the value must be a positive `integer` type that sets the maximum number of parallel threads to use for training each child decision

tree. Parallel threads do not affect the sequential method of training each tree. The default value is `16`.

`metrics` — If you set this value to `true`, the system calculates and stores final model metrics (`R²/RMSE` for regression or `Accuracy/LogLoss` for classification) on the training data. The default value is `false`.

`numSplits` — If you set this option, the value must be an `integer` greater than 1. This value sets the maximum number of binary branches a continuous feature can consider. The default value is `32`.

`resplitDepth` — If you set this option, the value must be an `integer` type that sets the maximum depth at which tree nodes can be re-split during optimization. This option controls how deep the algorithm searches for better split points. The default value is `6`.

`resplitThreshold` — If you set this option, the value must be a `decimal` type that sets the minimum improvement threshold required to trigger a re-split operation. Lower values allow more aggressive re-splitting but can increase training time. Higher values require larger improvements to trigger re-splits. The default value is `0.1`.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Kmeans

Model Options

Required

`k` — This option must be a positive `integer` type that specifies how many clusters to make.

Optional

`epsilon` — If you specify this option, the value must be a valid positive floating point value. When the maximum distance that a centroid moves from one iteration of the algorithm to the next is less than this value, the algorithm terminates. The default value is `0.00000001` (`1e-8`).

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`lloydRounds` — If you set this option, the option represents the maximum number of iterations of the Lloyd algorithm to train the model after guessing the centroids. For higher values of this option, the model is more likely to be accurate but takes longer to train. The default value is `20`.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`normalize` — If you set this option to `true`, the model normalizes the data before the start of training. The default value is `true`.

`oversampling` — If you set this option, the option represents the number of candidate guesses for the model to choose in the parallel-round phase of k-means||. For higher values of this option, the model is more likely to be accurate but takes longer to train. The default value is `k`.

`parallelRounds` — If you set this option, the option represents the minimum number of parallel rounds for which the k-means|| algorithm runs. For higher values of this option, the model is more likely to be accurate but takes longer to train. The default value is `8`.

© 2026 Ocient, Inc.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

K Nearest Neighbors

Model Options

Required

`k` — This option must be a positive `integer` type that specifies how many closest points to use for classifying a new point.

Optional

`distance` — If you set this option, the value must be a function in SQL syntax for calculating the distance between a point used for classification and points in the training data set. This function should use the variables x1, x2, ... for the 1st, 2nd, ... features in the training data set, and p1, p2, ... for the features in the point for classification. The default value is the Euclidian distance function.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. The default value is `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`weight` — If you specify this option, the value must be a function in SQL syntax for calculating the weight of a neighbor. The function should use the variable d for distance. By default, the distance is set to `1.0 / (d + 0.1)`, thus avoiding division by zero on exact inputs and still allowing neighbors to have some influence.

Linear Combination Regression

Model Options

Required

`functionN` — You must specify the first function using a key named `'function1'`. Subsequent functions must use keys with names that use subsequent values of N. You must specify functions in SQL syntax and should use the variables `x1, x2, ..., xn` to refer to the 1st, 2nd, and nth independent variables, respectively. For example, `'function1' -> 'sin(x1 * x2 + x3)', 'function2' -> 'cos(x1 * x3)'`.

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of

the input array to use starting at index 1. The system uses all indexes of the input array by default.

`gamma` — If you set this option, the value must be a matrix. This value represents a Tikhonov gamma matrix used for regularization. For details, see **Tikhonov regularization**.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (R-squared), the adjusted coefficient of determination, and the root mean squared error (RMSE). The default value is `false`.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. The default value is `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`threshold` — This option enables soft thresholding. If you specify this option, the option must be a positive numeric value. After the model calculates the coefficients, if any coefficients are greater than the threshold value, the model subtracts the threshold value from the coefficients. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to the coefficients. For any coefficients between the negative and positive threshold values, the model sets the coefficients to zero.

© 2026 Ocient, Inc.

`weighted` — If you set this option to `true`, the model performs weighted least squares regression, where each sample has an associated weight or importance. When weighted, there is an extra numeric column after the dependent variable that represents the weight of the sample. The default value is `false`.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

Linear Discriminant Analysis

Model Options

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. The default value is `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all

rows in the input. The default value is `false`.

Logistic Regression

Model Options

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the model calculates the percentage of samples that are correctly classified by the model and saves this information in the `sys.logistic_regression_models` system catalog table. The default value is `false`.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. The default value is `true`.

`numEpochs` — If you set this option, the value must be a positive `integer` type representing the maximum number of IRLS iterations during training. The default value is `20`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Multiple Linear Regression

Model Options

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`gamma` — If you set this option, the option must be a matrix. The value represents a Tikhonov gamma matrix used for regularization. For details, see **Tikhonov regularization**. The model uses this option for ridge regression.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (R-squared), the adjusted coefficient of determination, and the root mean squared error (RMSE). The default value is `false`.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during

training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. The default value is `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`threshold` — If you set this option, the option enables soft thresholding. The value must be a positive number. After the model calculates the coefficients, if any coefficients exceed the threshold value, the model subtracts the threshold value from those coefficients. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to the coefficients. For any coefficients that are between the negative and positive threshold values, the model sets the coefficients to zero.

`weighted` — If you set this option to `true`, the model performs weighted least squares regression, where each sample has a weight or importance associated with it. In this case, the table contains an additional numeric column after the dependent variable, which contains the weight for the sample. The default value is `false`.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

Naive Bayes

Model Options

Optional

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the model calculates the percentage of samples correctly classified by the model and saves this information in a system catalog table. The default value is `false`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Nonlinear Regression

Model Options

Required

`function` — Specify the name of the function to fit the data in SQL syntax. Use `a1, a2, …` to refer to the parameters for optimization. Use `x1, x2, …` to refer to the input features. The model does not allow some SQL functions. The model allows only scalar expressions that can be represented internally as postfix expressions. Most notably, the model does not allow some functions that are rewritten as CASE statements (like

`least()` and `greatest()`). If your function is not allowed, the model displays an error message.

`numParameters` — Specify this option as a positive integer. This value specifies the number of different parameters to optimize, i.e., how many different `aN` variables there are in the user-specified function.

Optional

`adamBeta1` — If you set this option, the option represents the value of β_1 in the Adam optimization algorithm. For higher values of this option, training is less noisy but takes longer to converge. The default value is `0.9`.

`adamBeta2` — If you set this option, the option represents the value of β_2 in the Adam optimization algorithm. For higher values of this option, training is less noisy but takes longer to converge. The default value is `0.99`.

`adamEpsilon` — If you set this option, the option represents the value of ε in the Adam optimization algorithm. For higher values of this option, training is more numerically stable but takes longer to converge. The default value is `1e-7`.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`finiteDifferenceH` — If you set this option, the value must be a `double` type representing the step size (`h`) for approximating gradients using the finite difference method. The model uses this value only if analytical gradients are not active. This value should generally be a small positive number, typically from `0.0001` (`1e-4`) to `0.0000001` (`1e-7`). The default value is `0.00001` (`1e-5`).

`gradientClipThreshold` — If you set this option, the value must be a `double` that represents the gradient norm threshold for clipping. When the overall gradient norm exceeds this threshold, the system scales all gradient components uniformly to

preserve direction. This operation prevents issues with exploding gradients in unstable loss landscapes. Set this value to 0 or a negative value to disable gradient clipping. The default value is `1000000` (`1e6`).

`lassoCoefficient` — If you specify this option, the value must be a `double` data type. This option is the lasso coefficient for the loss function. The default behavior is the function ignores this option, effectively setting this option to `0.0` .

`learningRate` — If you set this option, the value must be a `double` type representing the base learning rate for the Adam (Adaptive Moment Estimation) machine learning optimizer. Adam adapts this rate individually for each parameter during training. A common starting point for Adam is `0.001` (`1e-3`). Valid values must be positive and are generally in the range of `0.00001` (`1e-5`) to `0.01` (`1e-2`). A higher learning rate can speed up training, but can cause the optimizer to overshoot and miss optimal solutions. Conversely, a lower learning rate ensures more stable and precise convergence but can make training much slower. If you do not specify this option, the system automatically selects a learning rate and adjusts it during training using the 1Cycle learning schedule. Specifying a learning rate disables automatic adjustment and instead uses a fixed learning rate value.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`lossFunction` — If you set this option, the option indicates to the nonlinear optimizer the loss function to use on a per-sample basis. Then, the actual loss function is the sum of this function applied to all samples. The model should use the variable `y` to refer to the dependent variable in the training data and the variable `f` to refer to the computed estimate for the specified sample. The default is the least squares function, which you can specify as `(f-y)*(f-y)` .

`maxInitParamValue` — If you specify this option, the value must be a floating-point number. This option sets the maximum for initial parameter values in the optimization algorithm. The default value is `1` .

`metrics` — If you set this option to `true` , the model calculates the coefficient of determination (R-squared), the adjusted R-squared, and the root mean squared error

(RMSE). However, the model calculates these quality metrics using the least squares loss function, and not the user-specified loss function, because these metrics only make sense for least squares. The default value is `false`

`minInitParamValue` — If you specify this option, the value must be a floating-point number. This option sets the minimum for initial parameter values in the optimization algorithm. The default value is `-1`.

`numEpochs` — If you set this option, the value must be a positive `integer` type representing the maximum number of epochs, or full passes, during training through the entire data set. If you do not specify this option, the default maximum value is `200` for Adam optimization or `100` for Levenberg-Marquardt, but training typically stops earlier due to automatic early stopping when the model has converged.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`randomSeed` — If you set this option, the value must be a positive `integer` type representing the seed for the random number generator the system uses for weight initialization. Setting this option makes model training deterministic (given the same data and options). If you do not specify this option or set it to `0`, the system uses a non-deterministic random seed.

`ridgeCoefficient` — If you specify this option, the value must be a `double` data type. This option is the ridge coefficient for the loss function. The default behavior is the function ignores this option, effectively setting this option to `0.0`.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

© 2026 Ocient, Inc.

Polynomial Regression

Model Options

Required

`order` — This option is the degree of the polynomial and must be set to a positive integer.

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`gamma` — If you specify this option, the value must be a matrix. The value represents a Tikhonov gamma matrix that is used for regularization. For details, see **Tikhonov regularization**.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (R-squared), the adjusted coefficient of determination, and the root mean squared error (RMSE). The default value is `false`.

`negativePowers` — If you set this option to `true`, the model includes independent variables raised to negative powers. These variables are named Laurent polynomials. The model generates all possible terms such that the sum of the absolute value of the power of each term in each product is less than or equal to the order. For example, with two independent variables and the order set to `2`, the model is: `y = a1*x1^2 + a2*x1^-2 + a3*x2^2 + a4*x2^-2 + a5*x1*x2 + a6*x1^-1*x2 + a7*x1*x2^-1 +`

© 2026 Ocient, Inc.

`a8*x1^-1*x2^-1 + a9*x1 + a10*x1^-1 + a11*x2 + a12*x2^-1 + b`. The default value is `false`.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. The default value is `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`threshold` — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any of them are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

`weighted` — If you set this option to `true`, the model performs weighted least squares regression, where each sample has an associated weight. When weighted, there is an extra numeric column after the dependent variable that has the weight for the sample. The default value is `false`.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

Principal Component Analysis

Model Options

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Random Forest

Model Options

Required

`numChildren` — Number of child decision trees.

Optional

`ROCNumSamples` — If you set the option, you must also set the `metrics` option. This positive integer indicates the number of samples for the model to use for the area under the ROC curve. The default value is the number of child decision trees.

`bootstrap` — If you set this option to `true`, the model uses bootstrap sampling with replacement, meaning the model trains each tree in the random forest on a random

subset of the data (either the `rowsPerChild` or `fractionSelected` option sets the exact number of rows), and the same row can appear multiple times in each tree. If you set this option to `false`, this option does not use replacement, meaning each row can appear at most once per tree. The default value is `false`.

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`distinctCountLimit` — If you set this option, the value must be a positive integer. This value limits how many distinct values a non-continuous feature and the label can contain. The default value is `256`.

`doPrune` — If you set this option to `true`, the model uses Pessimistic Error Pruning (PEP) to prune the tree after training. The default value is `false`.

`enableResplits` — If you set this option, the value must be a `boolean` type that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This action can capture more complex, range-specific relationships. The default value is `true`, meaning that continuous features remain available for additional splits after use, allowing the tree to create more complex decision boundaries. When you set this option to `false`, the model marks continuous features as exhausted after their first use, and the model cannot use them again in subsequent splits in the same tree. The model passes this option directly to the child decision trees.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`featureSubsetStrategy` — If you set this option, the model passes this option directly to the child decision trees. The option specifies how many features the child decision trees should consider at each split from the still-available features. When this value is higher, the model will have higher accuracy and lower variance, but takes longer to

train. You can specify this option either as an integer (e.g., `4`, meaning consider up to 4 features at each split) or one of the three possible string options: `all` (checks every feature), `sqrt` (checks up to the square root of the number of total features), and `one-third` (checks up to one-third of the number of total features). The default value is `all`.

`fractionSelected` — If you set this option, the option represents the proportion of rows the model uses to train each child model. The value is a `double` that must be in the interval (0, 1]. You cannot set this option if you also set the `rowsPerChild` option to a positive value. The default behavior is that the model uses all available rows.

`inputsPerChild` — If you set this option, the option specifies the number of features for the creation of each child decision tree. The default value is the number of features you specify for the forest divided by 3 and rounded up.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`maxCellsToFetch` — If you set this option, the model passes this option directly to the child decision trees. If you set this option, the value must be a positive integer. Controls the chunking behavior when fetching feature values during model training. The limit represents the maximum number of data cells (calculated as the number of columns × number of rows) that the system can fetch in a single operation, not a byte limit. When the expected data size exceeds this threshold, the algorithm switches to database-based processing using SQL queries instead of in-memory processing. The default value is `33,554,432` cells (calculated as `32 × 1024 × 1024`).

`maxChildThreads` — If you set this option, the value must be an `integer` type representing the maximum number of threads each child decision tree can use. The default value is `1`.

`maxDepth` — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree. The default value is `3`.

`maxThreads` — If you set this option, the option specifies the maximum number of parallel threads to use while the model trains decision trees. This value must be a positive integer. The default value is `16`.

© 2026 Ocient, Inc.

`metrics` — If you set this option to `true`, the model also calculates the percentage of samples that are correctly classified by the model for the random forest and saves this information in a system catalog table. The default value is `false`.

`noSnapshot` — If you set this option to `true`, the data source must not change. In this case, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training a random forest. Child decision trees always have this option set to `true`, so the database does not create a separate intermediate table for each decision tree. The default value is `false`. Setting this option to `true` is useful when the training set is fixed. If the training set is a table with modifications, set this option to `false` as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to `false` because there is no guarantee that running that query twice generates the same 100 rows each time.

`numSplits` — If you set this option, the value must be an `integer` type greater than 1. This value sets the maximum number of binary branches a continuous feature can consider. The default value is `32`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`requiredFeatures` — If you set this option, the option must be a comma-separated list of integers as strings representing specific features where the first feature has the value `1`. The model uses these features in every decision tree in the forest. The default behavior is that the decision tree in the forest can train on any feature in the list.

`rowsPerChild` — If you set this option to a positive integer, the number represents the number of rows (from a random sample) to use for each decision tree. If you set this option to `0`, each child uses all available rows. The default value is `0`. You cannot set this option to a positive value if you also set the `fractionSelected` option.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`skipLimitCheck` — If you set this option to `true`, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one that is specified by the `distinctCountLimit` option. This option defaults to `false`.

`splitMetric` — If you set this option, the option controls which function the model uses to evaluate the quality of a split during tree construction. Supported options are: `gini_impurity` (measures impurity based on class distributions), `macro_f1` (uses macro-averaged F1 score to guide splits), `micro_f1` (uses micro-averaged F1 score to guide splits), and `weighted_f1` (uses class-frequency-weighted F1 score to guide splits). The default value is `gini_impurity`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`weighted` — If you set this option, the model considers weights for labels. If you set this option value to `true`, you must specify an additional column as a `double` in the training data for label weights. Rows with the same labels must have the same weights. If you set this value to `auto`, the model calculates weights automatically by weighting each label according to the ratio of the count of the most frequent label to the count of the specified label. As a result, the most frequent label has a weight of `1.0`, and the other label weights are higher. This option defaults to `false`, which means all labels have equal weight.

Regression Tree

Model Options

Optional

`continuousFeatures` — If you set this option, the value must be a comma-separated list of the feature indexes that are continuous numeric variables. Indexes start with 1. In the default state, the model considers no features as continuous.

`distinctCountLimit` — If you set this option, the value must be a positive integer. This value sets the limit for the number of distinct values a non-continuous feature and the label can contain. This option defaults to `256`.

`enableResplits` — If you set this option, the value must be a `boolean` type that determines if the tree can reuse the same continuous feature multiple times along a single branch (e.g., split on `x1 < 7` and later `x1 < 3`). This action can capture more complex, range-specific relationships. The default value is `true`, meaning that continuous features remain available for additional splits after use, which allows the tree to create more complex decision boundaries. When you set this option to `false`, the model marks continuous features as exhausted after their first use, and the model cannot use them again in subsequent splits in the same tree.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`featureSubsetStrategy` — If you set this option, the option specifies how many features the regression tree should consider at each split from the still-available features. When this value is higher, the model has a higher accuracy and lower variance, but takes longer to train. You can specify this option either as an integer (e.g., `4`, meaning consider up to four features at each split) or one of these values: `all` (check every feature), `sqrt` (check up to the square root of the number of total features), and `one-third` (check up to one-third of the number of total features). The default value is `all`.

`maxCellsToFetch` — If you set this option, the value must be a positive integer. Controls the chunking behavior when fetching feature values during model training. The limit represents the maximum number of data cells (calculated as the number of columns × number of rows) that the system can fetch in a single operation, not a byte limit. When the expected data size exceeds this threshold, the algorithm switches to database-based processing using SQL queries instead of in-memory processing. The default value is `33,554,432` cells (calculated as `32 × 1024 × 1024`).

`maxDepth` — If you set this option, the value must be a positive integer. This value sets the maximum allowable depth of the decision tree (the maximum number of features to split on). The default is unspecified, which means there is no maximum depth.

© 2026 Ocient, Inc.

`maxRows` — If you set this option, the value must be a positive integer. This option limits the number of rows used for model training by creating a snapshot table with only the specified number of rows from the input query. This option cannot be used with `noSnapshot -> true` (attempting to set both results in an `invalid argument` error during model creation). When this option is unspecified, the model trains using all rows from the input query.

`maxThreads` — If you set this option, the value must be a positive integer. This value indicates the maximum number of parallel threads to use while the model trains. The default value is `2`.

`metrics` — If you set this option to `true`, the model also calculates the percentage of samples correctly classified by the model and saves this information in a system catalog table. The default value is `false`.

`noSnapshot` — If you set this option to `true`, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training. This option defaults to `false`. In this case, the database creates and uses the intermediate table. Setting this option to `true` is useful when the training set is fixed. If the training set is a table with modifications, set this option to `false`, as the decision tree trainer uses different data sets in different parts of the tree. Likewise, if the training set consists of a query that returns 100 rows, then set this option to `false` because there is no guarantee that executing that query twice generates the same 100 rows each time.

`numSplits` — If you set this option, the value must be an integer greater than 1. This value sets the maximum number of binary branches a continuous feature can consider. The default value is `32`.

`queryInternalParallelism` — If you set this option, the database appends the `USING PARALLELISM = <value>` clause to all intermediate SQL queries the model executes during training, where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`resplitDepth` — If you set this option, the value must be an `integer` type that sets the maximum depth at which tree nodes can be re-split during optimization. Controls how deep the algorithm searches for better split points. The default value is `6`.

`resplitThreshold` — If you set this option, the value must be a `decimal` type that sets the minimum improvement threshold required to trigger a re-split operation. Lower values (e.g., `0.01`) allow more aggressive re-splitting but can increase training time. Higher values (e.g., `1.0`) require larger improvements to trigger re-splits. The default value is `0.1`.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`skipLimitCheck` — If you set this option to `true`, the model skips cardinality checks that throw errors when columns have too many values. The limit that this option checks is the same one that you specify using the `distinctCountLimit` option. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Simple Linear Regression

Model Options

Optional

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the model collects quality metrics such as the coefficient of determination (R-squared) and the root mean squared error (RMSE). The default value is `false`.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. The default value is `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`threshold` — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

`yIntercept` — If you set this option, then the option must be a numeric value. The system forces the specific y-intercept (i.e., the model value when `x` is zero).

Stacking

Model Options

Required

`levelOneModel` — This option specifies the level-1 child models of the stacking model. You must specify this value as a JSON array, where each object in the array has the four fields `type` (required), `name`, `options`, `ignoreColumn`, and `extraCallArguments`.

`levelZeroModels` — This option specifies the level-0 child models of the stacking model. You must specify this value as a JSON array, where each object in the array has the four fields `type` (required), `name`, `options`, `ignoreColumn`, and `extraCallArguments`.

Optional

`extraColumnCount` — If you set this option, the value must be an `integer` type that specifies how many non-feature columns there are in the input data. The default value is `0`.

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`hasLabelColumn` — If you set this option, the value must be a `boolean` type that specifies whether the input data includes a label column. The default value is `true`.

`maxThreads` — If you set this option, the option specifies the maximum number of parallel threads to use while the model trains. This value must be a positive integer. The default value is `16`.

`noSnapshot` — If you set this option to `true`, the data source must not change. In this case, the database does not create an intermediate table that stores the result of the specified SQL statement, which the model uses for training a random forest. Child decision trees always have this option set to `true`, so the database does not create a separate intermediate table for each decision tree. The default value is `false`. Setting this option to `true` can speed up training when the training set is fixed.

`preservedColumnsForLevelOne` — If you set this option, this option specifies the columns from the original training data to pass as an input column to the level-1 model, in addition to the level-0 outputs. This value should be a comma-separated list of integers starting at 1. The default behavior is to preserve none of the columns.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Support Vector Machine

Model Options

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`functionN` — By default, SVM uses a linear kernel. If you use a different kernel, you must provide a list of functions that are summed together, just like with linear combination regression. You must specify the first function using a key named 'function1'. Subsequent functions must use keys with names that use subsequent values of N. You must specify functions in SQL syntax and use the variables x1, x2, ... , xn to refer to the 1st, 2nd, and nth independent variables, respectively. You can specify the default linear kernel as: 'function1' → 'x1', 'function2' → 'x2', and so on. The model always adds a constant term equivalent to 'functionN' → '1.0' that you do not need to specify explicitly.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the model also calculates the percentage of samples that are correctly classified by the model and saves this information in a catalog table. This option defaults to `false`.

`normalize` — If you set this option to `true`, the model automatically computes the mean and standard deviation of each feature and uses them to normalize the data during training. Defaults to `true`.

`numEpochs` — If you set this option, the value must be a positive `integer` type representing the maximum number of IRLS iterations during training. The default value is `20`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes

during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`regularizationCoefficient` — If you set this option, the value must be a valid floating-point number. Use this option to control the balance of finding a wide margin and minimizing incorrectly classified points in the loss function. A larger (and positive) value makes having a wide margin around the hypersurface more important relative to the incorrectly classified points. Because of how the system implements SVM, the values for this option are likely different than values used in other common SVM implementations. The default value is `1.0 / 1000000.0`.

`skipDropTable` — If you set this option to `false`, the database deletes any intermediate tables created during model training. If you set this option to `true`, the database prevents the deletion of any intermediate tables created during model training. The default value is `false`.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

Vector Autoregression

Model Options

Required

`numLags` — Specify this option as a positive integer for the number of lags in the model.

`numVariables` — Specify this option as a positive integer for the number of variables in the model.

Optional

`featureArray` — If you set this option to `true`, the model expects only one array-type input column instead of multiple columns of training data. Each array row in the input column must be the same size. The default value is `false`.

`featureArrayElements` — If you set this option, the `featureArray` option must be set to `true`. The value must be a comma-separated list of integers representing indexes of the input array to use starting at index 1. The system uses all indexes of the input array by default.

`loadBalance` — If you set this option, the database appends the `USING load_balance_shuffle = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified option value (`true` or `false`). The default value is unspecified. In this case, the database does not add this clause.

`metrics` — If you set this option to `true`, the function collects the metric for the coefficient of determination (R-squared). The default value is `false`.

`normalize` — If you set this option to `true`, the model uses auto-scaling to compute the mean and standard deviation of each input feature to normalize data during training, making training more numerically stable. The model then unscales parameters so the persisted model operates in the original units. The default value is `true`.

`queryInternalParallelism` — If you set this option, the database appends the `USING parallelism = <value>` clause to all intermediate SQL queries the model executes during training where `value` is the specified positive integer value. The default value is unspecified. In this case, the database does not add this clause.

`suppressArrayLengthCheck` — If you set this option, the `featureArray` option must be set to `true`. The system skips checking that the array length is the same size for all rows in the input. The default value is `false`.

`threshold` — This option enables soft thresholding. If you specify this option, then the option must be a positive numeric value. After the model calculates the coefficients, if any of them are greater than the threshold value, the threshold value is subtracted from them. If any coefficients are less than the negation of the threshold value, the model adds the threshold value to them. For any coefficients that are between the negative and positive threshold values, the model sets those coefficients to zero.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Math Functions and Operators

Math Operators

Ocient® supports addition, subtraction, multiplication, and division between any two numeric types. Division involving two integral types always results in an integral type.

Decimal Precision

These rules determine the precision `p` and scale `s` for the result of a specified expression that involves decimals `d1` and `d2`, with precisions `p1` and `p2`, and scales `s1` and `s2`:

Operation	Precision	Scale
addition, subtraction	max(`s1` , `s2`) + max(`p1` - `s1` , `p2` - `s2`) + 1	max(`s1` , `s2`)
multiplication	`p1` + `p2`	`s1` + `s2`
division	*See note.*	max(6, `s1` + `p2` + 1)

For division, the resulting precision is the minimum necessary to hold the whole number portion of the result. If that portion does not exist because the absolute value of the result is less than one, the resulting precision is one greater than the scale. For other operations, the result might still have a lower precision (but the same scale) than specified by these rules when the result would otherwise have a leading insignificant zero.

If the resulting precision exceeds 31, the database decreases both precision and scale until the precision is 31, except in division, where the minimal scale is always six. If the resulting precision still exceeds 31, the database truncates it to 31. The database returns an overflow error when the result exceeds the available integral portion of the result. When the database decreases the resulting scale, it truncates the excess fractional digits in the result.

Math Operators for Time Intervals

Ocient supports math operators for time interval values (hour, minute, second, etc.), including when used with time-series data types such as TIME, TIMESTAMP, and DATE. You can use addition, subtraction, multiplication, and division on time intervals like any other numeric value.

© 2026 Ocient, Inc.

Division operations can produce inaccurate results because the system truncates the precision of the result to a whole number. For details, see **Decimal Precision**.

Examples

Add Interval Value to Timestamp

This example uses the MINUTES cast function to add six minutes to the TIMESTAMP value returned from the TIMESTAMP cast function of the `'2023-04-05 12:15:05'` string.

SQL

```
SELECT TIMESTAMP('2023-04-05 12:15:05') + (MINUTES(6));
```

Output: `2023-04-05 12:21:05.000`

Add Fractional Interval Value to Timestamp

This example shows how division can produce an inaccurate result. The division operation in the example must produce a whole number. In other words, the system calculates two minutes as `5 / 2 = 2` instead of two and a half minutes as `5 / 2 = 2.5`.

SQL

```
SELECT TIMESTAMP('2023-04-05 12:15:05') + (MINUTES(5) / 2);
```

Output: `2023-04-05 12:17:05.000`

Math Functions

These scalar math functions operate on a single value and return a single value. They can appear anywhere where an ordinary value or condition can appear.

For a list of common math symbols supported in Ocient, see **Math Operators**.

ABS

Returns the absolute value of a specified floating-point number.

Syntax

```
SQL
ABS(x)
```

Example

```
SQL
SELECT ABS(-250.5);
```

Output: `250.5`

ACOS

Returns the inverse cosine of a specified floating-point number.

Syntax

```
SQL
ACOS(x)
```

Example

```
SQL
SELECT ACOS(0.25);
```

Output: `1.318116071652818`

ACOSH

Returns the hyperbolic arc-cosine of a specified floating-point number.

Syntax

```sql
ACOSH(x)
```

Example

```sql
SELECT ACOSH(2);
```

Output: `1.3169578969248166`

ASIN

Returns the inverse sine of a specified floating-point number.

Syntax

```sql
ASIN(x)
```

Example

```sql
SELECT ASIN(0.25);
```

Output: `0.25268025514207865`

ASINH

Returns the hyperbolic arc-sine of a specified floating-point number.

Syntax

```sql
ASINH(x)
```

Example

```sql
SELECT ASINH(2);
```

Output: `1.4436354751788103`

ATAN

Returns the inverse tangent of a specified floating-point number.

Syntax

```sql
ATAN(x)
```

Example

```sql
SELECT ATAN(0.25);
```

Output: `0.24497866312686414`

ATAN2

Returns the inverse tangent of two numeric, floating-point values.

Syntax

```sql
ATAN2(x, y)
```

Example

```sql
SELECT ATAN2(0.50, 1);
```

Output: `0.4636476090008061`

ATANH

Returns the hyperbolic arc-tangent of a specified floating-point number.

Syntax

```sql
ATANH(x)
```

Example

```sql
SELECT ATANH(0.25);
```

Output: `0.25541281188299536`

BICDF

The cumulative distribution function of the standard bivariate normal distribution, which returns the probability that a random sample of (A, B) is less than `x` and less than `y` (`A <= x and B <= y`). The correlation is set to `0` in the case where you specify two arguments only.

You can specify an optional `rho` coefficient to calculate the correlation.

For details, see the **bivariate cumulative distribution function**.

Syntax

```
SQL

BICDF(x, y [, rho])
```

Example

```
SQL

SELECT BICDF(1,2);
```

Output: `0.8222040420815764`

BIPDF

The probability density function of the standard bivariate normal distribution returns the relative likelihood that a random sample would equal the specified values. The correlation is set to `0` in the case where you specify two arguments only.

You can specify an optional `rho` coefficient to calculate the correlation.

For details, see the **bivariate probability density function**.

Syntax

```
SQL

BIPDF(x, y, [, rho])
```

Example

```
SQL

SELECT BIPDF(1,2);
```

Output: `0.013064233284684893`

BITFUNC

Performs a variety of bit operations.

The `char_func` can be any of these string literals: `'AND'`, `'OR'`, or `'XOR'`. This argument determines the bitwise operation that is performed on arguments `x` and `y`.

Syntax

SQL

```
BITFUNC(char_func, x, y)
```

Argument	Data Type	Description
`character_func`	`CHAR`	This argument determines the bitwise operation performed on arguments `x` and `y`. Can be any of these string literals: `'AND'`, `'OR'`, or `'XOR'`. To perform bit operations using a NOT operator, use the **BITNOT** function.
`x`	`INT`	A numeric value to be evaluated by the function.
`y`	`INT`	A numeric value to be evaluated by the function.

Example

SQL

```
SELECT BITFUNC('AND',11,12);
```

Output: `8`

Example

SQL

```
SELECT BITFUNC('OR',11,12);
```

Output: `15`

Example

```
SQL

SELECT BITFUNC('XOR',11,12);
```

Output: `7`

BITAND

Alias for the **BITFUNC** syntax `BITFUNC('AND', x, y)`.

BITNOT

Returns the bitwise negation of `integral_x`.

Syntax

```
SQL

BITNOT(integral_x)
```

Example

```
SQL

SELECT BITNOT(1);
```

Output: `-2`

BITOR

Alias for the **BITFUNC** syntax `BITFUNC('OR', x, y)`.

BITXOR

Alias for the **BITFUNC** syntax `BITFUNC('XOR', x, y)`.

BOOLFUNC

Performs a Boolean logical evaluation on arguments `x` and `y`. Numbers equal to `0` evaluate to `false`, and numbers not equal to `0` evaluate to `true`.

Syntax

SQL

```
BOOLFUNC(char_func, x, y)
```

Argument	Data Type	Description
`char_func`	`CHAR`	This argument determines the Boolean operation performed on arguments `x` and `y`. Can be any of these string literals: `'AND'`, `'OR'`, or `'XOR'`. To perform operations using a NOT operator, use the **BOOLNOT** function.
`x`	`INT`	A numeric value for evaluation by the function.
`y`	`INT`	A numeric value for evaluation by the function.

Example

SQL

```
SELECT BOOLFUNC('AND',0,1);
```

Output: `false`

Example

SQL

```
SELECT BOOLFUNC('OR',0,1);
```

Output: `true`

Example

```
SQL

SELECT BOOLFUNC('XOR',0,1);
```

Output: `true`

BOOLAND

Alias for the **BOOLFUNC** syntax `BOOLFUNC('AND', x, y)`.

BOOLNOT

Returns the logical negation of the **BOOLFUNC** function. Numbers equal to `0` evaluate to `true`, and numbers not equal to `0` evaluate to `false`.

Syntax

```
SQL

BOOLNOT(numeric_x)
```

Example

```
SQL

SELECT BOOLNOT(0);
```

Output: `true`

BOOLOR

Alias for the **BOOLFUNC** syntax `BOOLFUNC('OR', x, y)`.

BOOLXOR

Alias for the **BOOLFUNC** syntax `BOOLFUNC('XOR', x, y)`.

CBRT

Returns the cube root of the numeric value `x`.

Syntax

```
SQL

CBRT(x)
```

Example

```
SQL

SELECT CBRT(64);
```

Output: `4.0`

CDF

The cumulative distribution function of the standard normal distribution. Returns the probability that a random sample is less than or equal to the specified value.

For details about the cumulative distribution function, see **Cumulative distribution function**.

Syntax

```
SQL

CDF(x)
```

Example

```sql
SELECT CDF(1.5);
```

Output: `0.9331927987311419`

CEIL

Returns the nearest integer greater than or equal to `x`.

Syntax

```sql
CEIL(x)
```

Example

```sql
SELECT CEIL(23.45);
```

Output: `24.0`

CEILING

Alias for CEIL.

COS

Returns the cosine of `x`.

Syntax

```sql
COS(x)
```

Example

```sql
SELECT COS(2);
```

Output: `-0.4161468365471424`

COSH

Returns the hyperbolic cosine of `x`.

Syntax

```sql
COSH(x)
```

Example

```sql
SELECT COSH(2);
```

Output: `3.7621956910836314`

COT

Returns the cotangent of `x`.

Syntax

```sql
COT(x)
```

Example

```sql
SELECT COT(2);
```

Output: `-0.45765755436028577`

DEGREES

Returns the corresponding angle in degrees for `x` in radians.

Syntax

```sql
DEGREES(x)
```

Example

```sql
SELECT DEGREES(1.5);
```

Output: `85.94366926962348`

DIV

Returns the result of `x` divided by `y`. If `y` is zero, returns NULL.

Syntax

```sql
DIV(x, y)
```

Example

```sql
SELECT DIV(45, 9);
```

Output: `5`

ERF

The error function is used for measurements that follow a normal distribution. Use this function to find the probability that the error of a single measurement lies between [-a, a]. For details about the error function, see **Error function**.

Syntax

```sql
ERF(x)
```

Example

```sql
SELECT ERF(8);
```

Output: `1.0`

ERFC

The complement of the error function. `ERFC(x) = 1 - ERF(x)`.

Syntax

```sql
ERFC(x)
```

Example

```sql
SELECT ERFC(8);
```

Output: `1.1224297172982928E-29`

EXP

Returns the exponential of `x` (e raised to the power of `x`).

Syntax

```sql
EXP(x)
```

Example

```sql
SELECT EXP(1);
```

Output: `2.718281828459045`

FLOOR

Returns the nearest integer less than or equal to `x`.

Syntax

```sql
FLOOR(x)
```

Example

```sql
SELECT FLOOR(22.75);
```

Output: `22.0`

GAMMA

Compute the gamma function of the specified value. For details about the gamma function, see the **Gamma function**.

Syntax

```sql
GAMMA(x)
```

Example

```sql
SELECT GAMMA(9.0);
```

Output: `40319.99999999999`

HEXBINX

Returns the x-coordinate of the center of the nearest hexagonal bin to the point (`x` , `y`). Hexagonal bins have a side length of 1.

Syntax

```sql
HEXBINX(x, y)
```

Example

```sql
SELECT HEXBINX(5, 7);
```

Output: `4.5`

HEXBINY

Returns the y-coordinate of the center of the nearest hexagonal bin to the point (`x` , `y`). Hexagonal bins have a side length of 1.

Syntax

```sql
HEXBINY(x, y)
```

Example

```sql
SELECT HEXBINY(5, 7);
```

Output: `7.794`

IERF

The inverse of the ERF error function.

Syntax

```sql
IERF(x)
```

Example

```sql
SELECT IERF(0);
```

Output: `0.0`

IERFC

The inverse of the complement of the error function.

Syntax

```sql
IERFC(x)
```

Example

```sql
SELECT IERFC(1);
```

Output: `0.0`

LEAKYRELU

Returns the leaky rectified linear unit function of `x`.

If unspecified, the optional `numeric_value_slope` argument defaults to `0.01`.

Syntax

```sql
LEAKYRELU(x [, numeric_value_slope] )
```

Example

```sql
SELECT LEAKYRELU(-40);
```

Output: `-0.4`

LEFT_SHIFT

Returns `x` shifted to the left by `y` bits. If `y` is less than 0, then `x` shifts to the right by the absolute value. The function throws an out-of-range error if you attempt a shift into the sign bit.

Syntax

```sql
LEFT_SHIFT(x, y)
```

Example

```sql
SELECT LEFT_SHIFT(1,2);
```

Output: `4`

LN

Returns the natural logarithm of `x` (base e or 2.171828...).

Syntax

```sql
LN(x)
```

Example

```sql
SELECT LN(2);
```

Output: `0.6931471805599453`

LOG2

Returns the base 2 logarithm of `x`.

Syntax

```sql
LOG2(x)
```

Example

```sql
SELECT LOG2(2);
```

Output: `1.0`

LOG

Returns the base 10 logarithm of `x`.

The optional `base` argument specifies the numeral system to use. If unspecified, the function defaults to base 10.

Syntax

```sql
LOG([ base,] x)
```

Example

```sql
SELECT LOG(2);
```

Output: `0.3010299956639812`

LOG_GAMMA

The natural logarithm of the absolute value of the gamma function.

Syntax

```sql
LOG_GAMMA(x)
```

Example

```sql
SELECT LOG_GAMMA(10);
```

Output: `12.80182748008147`

MOD

Returns the remainder from `x` divided by `y`.

Syntax

```sql
MOD(x, y)
```

Example

```sql
SELECT MOD(12, 10);
```

Output: `2`

PI

Returns the constant value of π.

Syntax

```sql
PI()
```

Example

```sql
SELECT PI();
```

Output: `3.141592653589793`

PMOD

Returns the smallest non-negative equivalence class of `x` % `y`.

Syntax

```sql
PMOD(x, y)
```

Example

```sql
SELECT PMOD(12,10);
```

Output: `2`

POWER

Returns `x` raised to the power of `y`. The return type `FLOAT` is returned for a `FLOAT` input, `DECIMAL` for `DOUBLE` input, and `BIGINT` otherwise.

Syntax

```sql
POWER(x, y)
```

Example

```sql
SELECT POWER(2, 4);
```

Output: `16`

PROBIT

The inverse of the cumulative distribution function. The function accepts only decimal values between `0` and `1`.

For details about the probit function, see the **Probit model**.

Syntax

```sql
PROBIT(x)
```

Example

```sql
SELECT PROBIT(0.4);
```

Output: `-0.2533471031357998`

RADIANS

Returns the corresponding angle in radians for `x` in degrees.

Syntax

```sql
RADIANS(x)
```

Example

```sql
SELECT RADIANS(2);
```

Output: `0.03490658503988659`

RAND

Takes no argument and returns a random `DOUBLE` value in the range [0, 1).

Syntax

```sql
RAND()
```

Example

```sql
SELECT RAND();
```

Output: `0.6053105024272647`

RELU

Returns the rectified linear unit function of `x`.

Syntax

```sql
RELU(x)
```

Example

```sql
SELECT RELU(-1);
```

Output: `0.0`

RIGHT_SHIFT

Returns `x` shifted to the right by `y` bits. If `y` is less than 0, then `x` shifts to the left by the absolute value of `y` in bits. The function throws an out-of-range error if you attempt a shift into the sign bit.

Syntax

```sql
RIGHT_SHIFT(x, y)
```

Example

```sql
SELECT RIGHT_SHIFT(5,1);
```

Output: `2`

ROUND

Returns `x` rounded to the nearest integer.

Syntax

```sql
ROUND(x)
```

Example

```sql
SELECT ROUND(12.49);
```

Output: `12.0`

SIGN

Returns the positive (+1), zero (0), or negative (-1) sign of `x`.

Syntax

```sql
SIGN(x)
```

Example

```sql
SELECT SIGN(+12);
```

Output: `1`

Example

```sql
select SIGN(0);
```

Output: `0`

Example

```sql
SELECT SIGN(-12);
```

Output: `-1`

SIN

Returns the sine of `x`.

Syntax

```sql
SIN(x)
```

Example

```sql
SELECT SIN(2);
```

Output: `0.9092974268256817`

SINH

Returns the hyperbolic sine of `x`.

Syntax

SQL

```
SINH(x)
```

Example

SQL

```
SELECT SINH(2);
```

Output: `3.626860407847019`

SQRT

Returns the square root of `x`.

Syntax

SQL

```
SQRT(x)
```

Example

SQL

```
SELECT SQRT(81);
```

Output: `9`

© 2026 Ocient, Inc.

SQUARE

Returns the square of `x`.

Syntax

```sql
SQUARE(x)
```

Example

```sql
SELECT SQUARE(7);
```

Output: `49`

TAN

Returns the tangent of `x`.

Syntax

```sql
TAN(x)
```

Example

```sql
SELECT TAN(1.75);
```

Output: `-5.52037992250933`

TANH

Returns the hyperbolic tangent of `x`.

Syntax

```
SQL

TANH(x)
```

Example

```
SQL

SELECT TANH(1.75);
```

Output: `0.9413755384972874`

TO_BASE

Converts an integer value to its string representation in a specified base (radix).

Syntax

```
SQL

TO_BASE(value, base)
```

Argument	Data Type	Description
`value`	`BYTE`, `SMALLINT`, `INT`, `BIGINT`	The integer value to convert to the specified base. Can be positive or negative.
`base`	`BYTE`, `SMALLINT`, `INT`, `BIGINT`	The base (radix) for the conversion. Must be an integer between 2 and 36 (inclusive). For bases greater than 10, lowercase letters `(a-z)` represent digits 10 through 35 (e.g., `a` = 10, `b` = 11, and `z` = 35).

Examples

Convert to Binary (Base 2)

This example converts the decimal value `7` to its binary representation.

```sql
SELECT TO_BASE(7, 2);
```

Output: `111`

Convert to Octal (Base 8)

This example converts the decimal value `64` to its octal representation.

```sql
SELECT TO_BASE(64, 8);
```

Output: `100`

TRUNC

Returns `x` truncated to `y` decimal places.

Syntax

```sql
TRUNC(x,y)
```

Example

```sql
SELECT TRUNC(1.23456,3);
```

Output: `1.234`

TRUNCATE

Alias for TRUNC.

Related Links

Array Functions and Operators

Matrix Functions and Operators

Tuple Functions and Operators

Number Formatting Functions

Data Types

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Matrix Functions and Operators

In Ocient®, a matrix is a fixed-size, multi-dimensional array of `DOUBLE` values. Many OcientML™ models and feature representations are naturally expressed as matrices, making them a core building block for in-database machine learning.

Matrices help you:

- **Store dense numeric features** — Keep related numeric inputs together as a single matrix value instead of many scalar columns. This storage simplifies machine learning training and scoring queries.
- **Represent model internals and transformations** — Learned weights, intermediate layers, and linear transformations are often matrix-shaped. Keeping these as matrices in SQL lets more of the machine learning pipeline execute directly in Ocient.
- **Apply math and linear-algebra style operations in SQL** — Matrix functions and operators let you construct, inspect, and manipulate matrix values in the same queries that drive machine learning models and preprocessing steps.

For an overview of Ocient machine-learning capabilities, see **Machine Learning in Ocient**.

Matrix Usage

To define a matrix data type in SQL, use the syntax `MATRIX[rows][columns]`, where the bracketed numbers specify the number of matrix rows and columns.

Example

This example creates a basic table `matrix_example` with a matrix column `col_matrix` that contains matrices with two rows and three columns.

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE matrix_example (
    id INT,
    col_matrix MATRIX[2][3] NOT NULL
);
```

Insert data for two matrices into the matrix column.

SQL

```
INSERT INTO matrix_example (id, col_matrix) VALUES
    (1, {{1, 2, 3}, {4, 5, 6}}),
    (2, {{0, 0, 0}, {7, 8, 9}});
```

For more DDL examples that use matrices, see **CREATE TABLE SQL Statement Examples**.

For details about working with matrix values, see **Data Types**.

Matrix Functions

In the examples, `_R{}` and `_C{}` are vector notation, meaning `_R{ 1,2,3 }` is a row vector with values `1`, `2`, and `3`.

Function	Syntax	Purpose	Example
Constructor	make_matrix_ixj(i, j, e_00, ..., e_ij)	Creates an `ixj` matrix with elements `e_00`, `…`, `e_ij`	make_matrix_1x3(1,3,1,2,3)
Constructor (2)	matrix_from_text(string)	Creates a matrix from the specified string	matrix_from_text('{ {1,2,3}, {1,2,3} }')
+ (matrix addition)	matrix + matrix	Performs matrix addition	{ {1,2,3}, {4,5,6} } + { {4,5, {1,2,3} }
- (matrix subtraction)	matrix - matrix	Performs matrix subtraction	{ {1,2,3}, {4,5,6} } - { {4,5,6 {1,2,3} }

Function	Syntax	Purpose	Example
* (scalar multiplication)	numeric * matrix	Performs scalar multiplication	2 * { {4,5,6}, {1,2,3} }
* (matrix multiplication)	matrix * matrix	Performs matrix multiplication	{ {1,2,3}, {4,5,6} } * { {1,2}, {3,4}, {5,6} }
/ (scalar division)	matrix / numeric	Performs scalar division	{ {1,2,3}, {4,5,6} } / 2
transpose	transpose(matrix)	Returns the transpose of the matrix	transpose({ {1,2,3}, {4,5,6}
determinant	det(matrix)	Returns the determinant of the matrix as a double	det({ {1,2}, {3,4} })
inverse	inverse(matrix)	Returns the inverse of a square, invertible matrix	inverse({ {1,2}, {3,4} })
trace	matrix_trace(matrix)	Returns the trace of a square matrix as a double	matrix_trace({ {1,2,3}, {4,5,6 {7,8,9} })
vector sum	vector_sum(matrix)	Returns the sum of elements in a one-dimensional matrix/vector	vector_sum(_R3) vector_sum(_C3)
vector min	vector_min(matrix)	Returns the minimum of elements in a one-dimensional matrix/vector	vector_min(_R3) vector_min(_C3)
vector max	vector_max(matrix)	Returns the maximum of elements in a one-dimensional matrix/vector	vector_max(_R3) vector_max(_C3)
vector argmin	vector_argmin(matrix)	Returns the 1-based position (index) of the minimum element in a one-dimensional matrix or vector (row or column vector). The first	vector_argmin(_R0)

Function	Syntax	Purpose	Example
		element has index 1.	
vector argmax	vector_argmax(matrix)	Returns the 1-based position (index) of the maximum element of a one-dimensional matrix or vector. The first element has index 1.	vector_argmax(_C0)
softmax	softmax(matrix)	Returns the softmax of a one-dimensional matrix or vector.	softmax(_R1)
cross entropy loss	cross_entropy_loss(matrix, matrix)	Returns the cross entropy loss of two one-dimensional matrices or vectors.	cross_entropy_loss(_R0.8, _C0 cross_entropy_loss(_C0.8, _R0 cross_entropy_loss(_R0.8, _R0 cross_entropy_loss(_C0.8, _C0
log loss	log_loss(matrix, matrix)	Returns the log loss of two one-dimensional matrices or vectors.	log_loss(_R0.8, _C0) log_loss(_C0.8, _R0) log_loss(_R0.8, _R0) log_loss(_C0.8, _C0)
logits loss	logits_loss(matrix, matrix)	Returns the logits loss of two one-dimensional matrices or vectors.	logits_loss(_R0.8, _C0) logits_loss(_C0.8, _R0) logits_loss(_R0.8, _R0) logits_loss(_C0.8, _C0)
hinge loss	hinge_loss(matrix, matrix)	Returns the hinge loss of two one-dimensional matrices or vectors.	hinge_loss(_R0.8, _C0) hinge_loss(_C0.8, _R0) hinge_loss(_R0.8, _R0) hinge_loss(_C0.8, _C0)
dot product	dot(matrix, matrix)	Returns the dot product of two one-dimensional matrices/vectors	dot(_R3, _C3) dot(_C3, _R3) dot(_R3, _R3) dot(_C3, _C3)
magnitude	abs(matrix)	Returns the magnitude of one-	abs(_R3) abs(_C3)

Function	Syntax	Purpose	Example
		dimensional matrix/vector	
LUPQ decomp	LUPQ_decomp(matrix)	Returns LUPQ decomposition of a square matrix A as a tuple of 4 matrices where PAQ = LU	
LUP decomp	LUP_decomp(matrix)	Returns LUP decomposition of a square matrix as a tuple of 3 matrices	
QR decomp	qrdecomp(matrix)	Returns QR decomposition of a matrix as a tuple of 2 matrices	
SVD decomp	SVD_decomp(matrix)	Returns SVD decomposition of a matrix as a tuple of 3 matrices	
eigenvalues/vectors	eigen(matrix)	Returns eigenvalues and eigenvalues of a square matrix as a vector of pairs	
identity matrix	identity_matrix(unsigned int)	Returns an identity matrix of the specified dimension	identity_matrix(3)
NULL matrix	null_matrix(unsigned int, unsigned int)	Returns a NULL matrix of the specified (row, col) dimensions	null_matrix(2, 3)
zero matrix	zero_matrix(unsigned int, unsigned int)	Returns a zero matrix of the specified (row, col) dimensions	zero_matrix(2, 3)
matrix dimension	matrix_dim(matrix)	Returns the dimensions of the specified matrix as a tuple of (row, col) integers. When the input matrix is NULL, the	matrix_dim(_R{1, 2, 3})

Function	Syntax	Purpose	Example
		function returns NULL.	
Frobenius norm	frobenius(matrix)	Returns the Frobenius norm of a matrix. This norm is the matrix norm of an `M`` x ``N` matrix defined as the square root of the sum of the absolute squares of its elements. |a|^2 is the same as a^2, where a is a real number.	frobenius(matrix_from_text('{- {8,6}}'))

Matrix Operators

Operator	Syntax	Purpose
<	matrix < matrix	Less than comparison
>	matrix > matrix	Greater than comparison
<=	matrix <= matrix	Less than or equal to comparison
>=	matrix >= matrix	Greater than or equal to comparison
=	matrix = matrix	Equal to comparison
<>	matrix <> matrix	Not equal to comparison
[,]	matrix[rowIndex,colIndex]	Access the matrix at the specified row and column. 1-Indexed. Accessing a NULL matrix or using a NULL index returns NULL. Access out of bounds returns a NULL.
[:,:]	matrix[r1:c1,r2:c2]	Slice the matrix at the specified row range and column range. * The index starts at `1`. * Slicing a NULL matrix or using a NULL index returns NULL. * Start indices omitted or below `1` become `1`. End indices omitted or above the number of rows/columns become the number of rows/columns. Example — `matrix_10X10[:10,-1:20]` is equivalent to

Operator	Syntax	Purpose
		`matrix[1:10,1:10]`. * Ranges completely out of bounds throw an `INVALID_ARGUMENT` error. * Sequential slices like `matrix[1:1,1:1][1:1,1:1]` throw an error. * Slices cannot be combined with access. `matrix[1,1:10]` is not allowed, but `matrix_10X10[1:1,1:10]` is. * Trying to slice with non-constant indexes throws an error.

Operator Examples

Operator	SQL Predicate	Result
`<`	`{ {1,2,3}, {4,5,6} } < { {4,5,6}, {1,2,3} }`	true
	`{ {1,2,3}, {4,5,6} } < { {1,2,2}, {1,2,3} }`	false
	`{ {1,2,3}, {4,5,6} } < { {1,2,3}, {4,5,6} }`	false
`>`	`{ {4,5,6}, {4,5,6} } > { {1,2,3}, {7,8,9} }`	true
	`{ {1,2,3}, {4,5,6} } > { {4,5,6}, {1,2,3} }`	false
	`{ {1,2,3}, {4,5,6} } > { {1,2,3}, {4,5,6} }`	false
`<=`	`{ {1,2,3}, {4,5,6} } <= { {4,5,6}, {1,2,3} }`	true
	`{ {1,2,3}, {4,5,6} } <= { {1,2,2}, {1,2,3} }`	false
	`{ {1,2,3}, {4,5,6} } <= { {1,2,3}, {4,5,6} }`	true
`>=`	`{ {4,5,6}, {4,5,6} } >= { {1,2,3}, {7,8,9} }`	true
	`{ {1,2,3}, {4,5,6} } >= { {1,5,6}, {1,2,3} }`	false
	`{ {1,2,3}, {4,5,6} } >= { {1,2,3}, {4,5,6} }`	true
`=`	`{ {1,2,3}, {4,5,6} } = { {1,2,3}, {4,5,6} }`	true
	`{ {1,2,3}, {4,5,6} } = { {1,2,4}, {4,5,6} }`	false
`<>`	`{ {1,2,3}, {4,5,6} } <> { {1,2,3}, {4,5,6} }`	false
	`{ {1,2,3}, {4,5,6} } <> { {1,2,4}, {4,5,6} }`	true
`matrix_access_operator`	`{ {1,2,3}, {4,5,6} }[1,2]`	2.0
	`{ {1,2,3}, {4,5,6} }[NULL,2]`	NULL

© 2026 Ocient, Inc.

Operator	SQL Predicate	Result
`matrix_slice_operator`	`{ {1,2,3}, {4,5,6} }[1:,2:3]`	{ {2.0,3.0}, {5.0,6.0} }
	`{ {1,2,3}, {4,5,6} }[1:NULL,2:3]`	NULL

Bibliography

Golub, Gene H., and Charles F. Van Loan. *Matrix Computations*. 3rd ed, Johns Hopkins University Press, 1996.

Related Links

Array Functions and Operators

Math Functions and Operators

Tuple Functions and Operators

Data Types

Data Types for Data Pipelines

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Network Type Functions

The Ocient® System supports these functions for `IP` and `IPV4` data types.

IP

Casts the `IPV4` data type to the `IP` data type. The returned value is the IPv6-mapped IPv4 address.

Syntax

SQL

```
IP(ipv4_col)
```

Argument	Data Type	Description
`ipv4_col`	`CHAR` or `IPV4`	IP address in the `IPV4` data type to cast.

Example

Cast the IPv4 address `128.1.2.3` to the IPv6-mapped IPv4 address.

SQL

```
SELECT IP(IPV4('128.1.2.3'));
```

Output

Text

```
ip(ipv4(('128.1.2.3')))
-----------------------------------------------
128.1.2.3

Fetched 1 row
```

© 2026 Ocient, Inc.

IPV4

Returns the IPv4 portion of the IPv6 address.

You can only cast the IP data type to IPv4 if the IPv6 value has an IPv4 semantic equivalent. The database can convert these IPv6 address formats to IPv4:

- IPv4-compatible IPv6 `::x.x.x.x`
- IPv6-mapped IPv4 `::ffff:x.x.x.x`
- NAT64-prefixed `64:ff9b::x.x.x.x`

Syntax

```
SQL

IPV4(ip_col)
```

Argument	Data Type	Description
`ip_col`	`IP`	IPv6-mapped IP address in the `IP` data type to cast. If the IPv6 address is not IPv4-compatible, IPv6-mapped, or NAT64-prefixed, the cast returns an `INVALID DATA TYPE CONVERSION` error.

Example

Cast the IPv6-mapped IP address `::ffff:128.1.2.3` to the `IPV4` data type. Return the IPv4 portion of the address.

```
SQL

SELECT IPV4(IP('::ffff:128.1.2.3'));
```

Output

Text

```
ipv4(ip(('::ffff:128.1.2.3')))
------------------------------
128.1.2.3

Fetched 1 row
```

IS_IPV4

The `IS_IPV4` function tests whether the database can convert the `IP` value to the `IPV4` data type. The database can convert these IPv6 address formats to IPv4:

- IPv4-compatible IPv6 `::x.x.x.x`
- IPv6-mapped IPv4 `::ffff:x.x.x.x`
- NAT64-prefixed `64:ff9b::x.x.x.x`

The function returns a Boolean value (`true` or `false`) that indicates whether the conversion works from the specified IP address to the IPv4 address.

Syntax

SQL

```
IS_IPV4(ip_col)
```

Argument	Data Type	Description
`ip_col`	`IP`	Specified IP address for conversion to an IPv4 address.

Example

Determine whether the `::fffe:128.1.2.3` string is a valid IPv4 address. Use the `IP` function to cast the string as an IP address.

SQL

```
SELECT IS_IPV4(IP('::fffe:128.1.2.3'));
```

Output

Text

```
is_ipv4_convertible(ip(('::fffe:128.1.2.3')))
---------------------------------------------
false

Fetched 1 row
```

SUBNET

The `SUBNET` function computes the prefix from an `IP` or `IPV4` value and the size of the prefix. The function returns a value of the same type as the `ip_col` argument.

Syntax

SQL

```
SUBNET(ip_col, prefix_size)
```

Argument	Data Type	Description
`ip_col`	`IP` or `IPV4`	The IP address for the prefix calculation.
`prefix_size`	Integer	The size of the subnet network prefix. For `IPV4` addresses, the size must be in the range [0, 32], and for `IP` addresses, the size must be in the range [0, 128]. If the prefix exceeds these range values, the function returns an `INVALID ARGUMENT` error.

Example

Calculate the prefix for the NAT64-prefixed IP address `64:ff9b::128.2.3.4` with prefix size `98`. Use the `IP` function to cast the string as an IP address.

SQL

```
SELECT SUBNET(IP('64:ff9b::128.2.3.4'), 98);
```

Output

Text

```
subnet(ip(('64:ff9b::128.2.3.4')), (98))
--------------------------------------------
64:ff9b:0:0:0:0:8000:0

Fetched 1 row
```

Related Links

Data Types

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Scalar Data Conversion Functions

These functions convert one data type to another.

Scalar Data Conversion Functions

Function	Syntax	Purpose
BYTE	BYTE(numeric value)	Casts the argument to a value of type byte
	BYTE(character value)	Parses the string as a byte value
SMALLINT	SMALLINT(numeric value)	Casts the argument to a value of type smallint
	SMALLINT(character value)	Parses the string as a `smallint` value
INTEGER	INT(numeric value) or INTEGER(numeric value)	Casts the argument to a value of type integer
	INT(character value) or INTEGER(character value)	Parses the string as an integer value
BIGINT	BIGINT(numeric value)	Casts the argument to a value of type bigint
	BIGINT(character value)	Parses the string as a `bigint` value
	BIGINT(time value)	Creates a `bigint` with the number of milliseconds represented by the time argument
	BIGINT(timestamp value)	Creates a `bigint` with the number of milliseconds after the Unix® epoch corresponding to the timestamp argument
FLOAT	FLOAT(numeric value)	Casts the argument to a value of type float
	FLOAT(character value)	Parses the string as a float value
DOUBLE	DOUBLE(numeric value)	Casts the argument to a value of type double
	DOUBLE(character value)	Parses the string as a double value
DECIMAL	DECIMAL(numeric value)	Casts the argument to a value of type decimal

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Function	Syntax	Purpose
	DECIMAL(character value)	Parses the string and creates a decimal value
	DECIMAL(numeric value, integer value, integer value)	Casts the first argument to a decimal value with precision of the second argument and scale of the third argument
	DECIMAL(character value, integer value, integer value)	Parses the first argument to a decimal value with precision of the second argument and scale of the third argument
CHAR | STR	CHAR(numeric value)	Creates a string version of the numeric value
	CHAR(character value)	None
	CHAR(timestamp)	Creates a string representation of the timestamp
	CHAR(date)	Creates a string representation of the date
	CHAR(time)	Creates a string representation of the time
	CHAR(boolean)	Creates a string representation of the Boolean value
	CHAR(binary)	Creates a hexadecimal string representation of the binary value
	CHAR(ip)	Creates a string representation of the IPV4 or IPV6 value
	CHAR(ipv4)	Creates a string representation of the IPV4 value
	CHAR(hash)	Creates a string representation of the fixed-binary hash value
	CHAR(point)	Creates a string representation of the point value
	CHAR(uuid)	Creates a string representation of the Universally Unique IDentifier (UUID)
BINARY	BINARY(character value)	Parses a hexadecimal string (such as `0x54ab`) to create a binary value. Letters can be in either case.
	BINARY(binary)	None
	BINARY(hash)	Converts the fixed-length binary value (hash) to a variable-length binary value

Function	Syntax	Purpose
BOOLEAN	BOOLEAN(character value)	Parses a string to create a Boolean value. The function converts any of these string values to `TRUE` : `true` , `t` , `yes` , `y` , `on` , `1` . The function converts any of these string values to `FALSE` : `false` , `f` , `no` , `n` , `off` , `0` . The string parsing is case-insensitive. Any unsupported string values raise an error.
	BOOLEAN(numeric value)	Parses a numeric value to create a Boolean value. The function converts any non-zero value to `TRUE` . The function converts `0` to `FALSE` .
	BOOLEAN(boolean)	None
DATE	DATE(character value)	Parses a string in the form ‘YYYY-MM-DD’ to create a date. Extra characters are ignored.
	DATE(date)	None
	DATE(timestamp)	Truncates a timestamp to be a date value
HASH	HASH(char, length)	Creates a fixed-length binary value, with length <length>, from a string, i.e., 0x1234abcd. Zero extended if the string does not have enough bytes, truncated if it has too many.
	HASH(binary, length)	Creates a fixed-length binary value, with length <length>, from a variable-length binary value. Zero extended if the string does not have enough bytes, truncated if it has too many.
IP	IP(character value)	Creates a value of type IPV4 or IPV6 from a string
IPV4	IPV4(character value)	Parses a string containing an IP address and creates a value of type IPV4
	IPV4(ipv4)	no-op

© 2026 Ocient, Inc.

Function	Syntax	Purpose
TIME	TIME(character value)	Parses the string to create a time value. The string must be in the form `'HH:MM[.SSSSSSSSS]'`
	TIME(bigint)	Creates a time value representing a certain number of milliseconds after midnight
	TIME(time)	None
	TIME(timestamp)	Makes a time value from the time component of a timestamp
TIMESTAMP	TIMESTAMP(character value)	Parses a string and makes a timestamp value. The string must be in the format `'YYYY-MM-DD[ HH:MM][.SSSSSSSSS]'`.
	TIMESTAMP(date)	Makes a timestamp from the date by setting the time component to `00:00:00.0`
	TIMESTAMP(timestamp)	None
	TIMESTAMP(bigint value)	Creates a timestamp from a `bigint` representing the number of milliseconds after the Unix epoch (1/1/1970).
UUID	UUID(character value)	Parses the string and makes a UUID value. The string must be a valid UUID.
	UUID(uuid)	None
WEEKS	WEEKS(integral value)	Converts an integral value to an interval value of type weeks for date calculations.
	Weeks(weeks)	None
DAYS	DAYS(integral value)	Converts an integral value to an interval value of type days for date calculations.
	DAYS(days)	None
	DAYS(weeks)	Converts an interval value of type weeks to an interval value of type days representing the same time span.
MONTHS	MONTHS(integral value)	Converts an integral value to an interval value of type months for date calculations.
	MONTHS(months)	None
	MONTHS(years)	Converts an interval value of type years to an interval value of type months

© 2026 Ocient, Inc.

Function	Syntax	Purpose
		representing the same time span
YEARS	YEARS(integral value)	Converts an integral value to an interval value of type years
	YEARS(years)	None
HOURS	HOURS(integral value)	Converts an integral value to an interval value of type hours
	HOURS(hours)	None
	HOURS(days)	Converts an interval value of type days to an interval value of type hours representing the same time span.
	HOURS(weeks)	Converts an interval value of type weeks to an interval value of type hours representing the same time span.
MINUTES	MINUTES(integral value)	Converts an integral value to an interval value of type minutes
	MINUTES(minutes)	None
	MINUTES(hours)	Converts an interval value of type hours to an interval value of type minutes representing the same time span
	MINUTES(days)	Converts an interval value of type days to an interval value of type minutes representing the same time span
	MINUTES(weeks)	Converts an interval value of type weeks to an interval value of type minutes representing the same time span
SECONDS	SECONDS(integral value)	Converts an integral value to an interval value of type seconds
	SECONDS(seconds)	None
	SECONDS(minutes)	Converts an interval value of type minutes to an interval value of type seconds representing the same time span
	SECONDS(hours)	Converts an interval value of type hours to an interval value of type seconds representing the same time span
	SECONDS(days)	Converts an interval value of type days to an interval value of type seconds representing the same time span

Function	Syntax	Purpose
	SECONDS(weeks)	Converts an interval value of type weeks to an interval value of type seconds representing the same time span
MILLISECONDS	MILLISECONDS(integral value)	Converts an integral value to an interval value of type milliseconds
	MILLISECONDS(milliseconds)	None
	MILLISECONDS(seconds)	Converts an interval value of type seconds to an interval value of type milliseconds, representing the same time span
	MILLISECONDS(minutes)	Converts an interval value of type minutes to an interval value of type milliseconds representing the same time span
	MILLISECONDS(hours)	Converts an interval value of type hours to an interval value of type milliseconds representing the same time span
	MILLISECONDS(days)	Converts an interval value of type days to an interval value of type milliseconds representing the same time span
	MILLISECONDS(weeks)	Converts an interval value of type weeks to an interval value of type milliseconds representing the same time span
MICROSECONDS	MICROSECONDS(integral value)	Converts an integral value to an interval value of type microseconds
	MICROSECONDS(milliseconds)	Converts an interval value of type milliseconds to an interval value of type microseconds representing the same time span
	MICROSECONDS(seconds)	Converts an interval value of type seconds to an interval value of type microseconds representing the same time span
	MICROSECONDS(minutes)	Converts an interval value of type minutes to an interval value of type microseconds representing the same time span
	MICROSECONDS(hours)	Converts an interval value of type hours to an interval value of type microseconds representing the same time span
	MICROSECONDS(days)	Converts an interval value of type days to an interval value of type microseconds representing the same time span

© 2026 Ocient, Inc.

Function	Syntax	Purpose
	MICROSECONDS(weeks)	Converts an interval value of type weeks to an interval value of type microseconds representing the same time span
	MICROSECONDS(microseconds)	None
NANOSECONDS	NANOSECONDS(integral value)	Converts an integral value to an interval value of type nanoseconds
	NANOSECONDS(microseconds)	Converts an interval value of type microseconds to an interval value of type nanoseconds representing the same time span
	NANOSECONDS(milliseconds)	Converts an interval value of type milliseconds to an interval value of type nanoseconds representing the same time span
	NANOSECONDS(seconds)	Converts an interval value of type seconds to an interval value of type nanoseconds representing the same time span
	NANOSECONDS(minutes)	Converts an interval value of type minutes to an interval value of type nanoseconds representing the same time span
	NANOSECONDS(hours)	Converts an interval value of type hours to an interval value of type nanoseconds representing the same time span
	NANOSECONDS(days)	Converts an interval value of type days to an interval value of type nanoseconds representing the same time span
	NANOSECONDS(weeks)	Converts an interval value of type weeks to an interval value of type nanoseconds representing the same time span
	NANOSECONDS(nanoseconds)	None

Implicit Conversions

In certain situations, Ocient® SQL can automatically convert expressions to a specific data type without an explicit SQL statement.

Cast from Strings

You can cast string literals to any other data type by placing the name of the data type before the string literal.

Examples

Cast a Double

Cast the value `1.5` from a string to a floating-point value.

```
SQL
SELECT DOUBLE '1.5';
```

*Output: * `1.5`

Cast an Array

Cast the array `INT[0,1,2,3]` from a string to an array of integers.

```
SQL
SELECT INT[] 'INT[0,1,2,3]';
```

*Output: * `['0','1','2','3']`

Cast a Point Value

Cast the point value `POINT(0 0)` to the geospatial data type `POINT`. In this case, the cast keyword is `ST_POINT`, and the data type is `POINT`.

```
SQL
SELECT ST_POINT 'POINT(0 0)';
```

*Output: * `POINT(0.0 0.0)`

Cast from Date and Time Strings

You can cast date and time values to their interval value with the `INTERVAL` keyword.

Retrieve Interval Values from Various Date and Time Strings

Retrieve the interval value `5` from the `'5 DAYS'` string.

```
SQL
SELECT INTERVAL '5 DAYS';
```

*Output: * `5`

Retrieve the interval value `2` from the `'2 WEEKS'` string.

```
SQL
SELECT INTERVAL '2 WEEKS';
```

*Output: * `2`

Retrieve the interval value `10` from the `'10 SECONDS'` string.

```
SQL
SELECT INTERVAL '10 SECONDS';
```

Output: `10`

CAST Functions

These SQL statements allow casting an expression to any other data type.

CAST

Transforms an expression to any other data type.

This function operates similarly to the **:: Operator**.

Syntax

SQL

```
CAST[!] expr [AS] data_type
```

Argument	Data Type	Description
`expr`	Any	A column or value to transform to another data type.
`data_type`	String	Any of the Ocient-supported data types. For a full list, see **Data Types**.

As shown in the syntax examples, `CAST` can support the optional `!` operator.

The `!` operator returns a NULL value if the operation cannot cast to the specified data type. This behavior prevents situations where the system would normally raise an error.

Examples

Cast an Integer to Varchar

Transform the value `2` as a `VARCHAR` string.

SQL

```
SELECT CAST(2 AS VARCHAR);
```

Output: `'2'`

Cast a Varchar to Date

Transform the `'2020-02-02'` string to the `DATE` data type.

SQL

```
SELECT CAST('2020-02-02' AS DATE);
```

*Output: * `'2020-02-02'`

Cast an Incompatible Type with the `!` Operator

Transform the `'a'` string to an integer, or `INT` data type, using the `!` operator.

```sql
SELECT CAST!('a' AS INT);
```

Output: `NULL`

This operation generates NULL because the database cannot transform this value to an integer. Without adding this operator, the statement raises an error because the database cannot transform the input expression to a numeric type.

`::` Operator

Transforms an expression to any other data type.

This function operates similarly to the **CAST** function.

`::` Syntax

```sql
expr ::[!] data_type
```

Argument	Data Type	Description
`expr`	Any	A column or value to transform to another data type.
`data_type`	String	Any of the Ocient-supported data types. For a full list, see **Data Types**.

> As shown in the syntax examples, `::` can support the optional `!` operator.
>
> The `!` operator returns a NULL value if the operation cannot cast to the specified data type. This behavior prevents situations where the system would normally raise an error.

Examples

Cast an Integer to Float

Transform the `'2'` string to a floating-point number.

```sql
SELECT '2' :: FLOAT;
```

Output: `2.0`

Cast to an Incompatible Type with the `!` Operator

Transform the `'a'` string to an integer, or `INT` data type, using the `!` operator.

```sql
SELECT 'a' ::! INT;
```

Output: `NULL`

This operation generates NULL because the database cannot transform this value to an integer. Without adding this operator, the statement raises an error because the database cannot transform the input expression to a numeric type.

Related Links

Date and Time Functions

Character and Binary Functions

Math Functions and Operators

Formatting Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Time Zone Functions

⚠ The Ocient® cluster system administrator is responsible for keeping the OS time zone database (e.g., on Ubuntu®, `tzdata`) up to date. Outdated time zone database files might cause unexpected results due to IANA rule updates around name changes, DST observance, and UTC offsets for time zones.

ⓘ The `TIMESTAMP` data type has no associated time zone value. The database administrator tracks the associated time zone with a `TIMESTAMP` column, whether through an associated `time_zone` `VARCHAR` column or other documentation. Ocient recommends storing all `TIMESTAMP` data types in the UTC time zone to avoid ambiguity and using these functions to convert to arbitrary time zones. You can still store `TIMESTAMP` data types associated with an arbitrary time zone and use these functions to convert to UTC and from UTC to another time zone.

CONVERT_UTC_TIMESTAMP_TO_LOCAL

The function converts a timestamp from the UTC time zone to a specified local time zone.

This function can take the `timezone` argument as a `VARCHAR` literal, or the argument can be a column. When the argument is a literal, you can arbitrarily capitalize it. When the argument is a column, the values in the column must be capitalized according to the canonical **IANA time zone names,** such as `'America/Chicago'`.

Syntax

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(timestamp,timezone)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`timestamp`	`TIMESTAMP` or `VARCHAR`	Timestamp to convert that you specify as a `TIMESTAMP` or `VARCHAR` that you typecast in a timestamp format.
`timezone`	`VARCHAR`	`VARCHAR` that contains the full name of the target time zone.

Examples

Convert a Constant Time Zone Value

Convert timestamp `2022-11-06 02:09:00` to the local time in Chicago.

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 02:09:00','America/Chic
```

Output

Text

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(timestamp(('2022-11-06 02:09:00')), ('America/Chicag
--------------------------------------------------------------------------------
2022-11-05 21:09:00.000000000
```

Convert a Timestamp Value in a Column

Convert the timestamp in the `ts` column to the local time in Chicago using the `test_timezone` table.

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts,'America/Chicago') FROM test_timezone;
```

Output

Text

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, ('America/Chicago'))
-----------------------------------------------------
2023-01-09 17:23:12.173997444
```

Convert a Timestamp Value and Time Zone Name in Columns

Convert the timestamp in the `ts` column to the local time specified by the `timezone_name` column using the `test_timezone` table.

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts,timezone_name) FROM test_timezone;
```

Output

Text

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, timezone_name)
-------------------------------------------------------
2023-01-09 17:43:21.193229991
```

Convert a Timestamp Value Using Other Date and Time Functions

Use the `TO_TIMESTAMP` function to convert a numeric timestamp to the local time in Chicago.

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(TO_TIMESTAMP(1534587890), 'America/Chicago');
```

Output

Text

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(to_timestamp((1534587890)), ('America/Chicago'))
-----------------------------------------------
2018-08-18 05:24:50.000000000
```

Use a UTC timestamp from the output of the `CONVERT_LOCAL_TIMESTAMP_TO_UTC` function with Chicago time and convert it back to local time in Chicago.

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '202
    'America/Chicago');
```

Output

Text

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(CONVERT_LOCAL_TIMESTAMP_TO_UTC(timestamp(('2022-11-0
--------------------------------------------------------------------------------
2022-11-09 01:59:00.000000000
```

Extract the hour portion of the local Chicago time using the output of the `TO_TIMESTAMP` function on a numeric timestamp.

SQL

```
SELECT HOUR(CONVERT_UTC_TIMESTAMP_TO_LOCAL(TO_TIMESTAMP(1534587890), 'America/Chica
```

Output

Text

```
hour(CONVERT_UTC_TIMESTAMP_TO_LOCAL(to_timestamp((1534587890)), ('America/Chicago')
--------------------------------------------------------------------------------
5
```

Convert a Timestamp During Table Creation

Create a table using a `SELECT` statement that converts a timestamp in local Chicago time.

SQL

```
CREATE TABLE test_timezone_2 (ts timestamp, b int)
    AS (SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 02:09:00', 'Ame
        1 FROM sys.dummy1);

Modified 1 row

SELECT * FROM test_timezone_2;            1
```

Output

Text

```
ts_1                                        b_1
-------------------------------------------------------------------------------
2022-11-05 21:09:00.000000000
```

Convert a Timestamp During an Insert Operation

Insert a timestamp in local Chicago time into the `test_timezone_2` table.

SQL

```
INSERT INTO test_timezone_2 SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-1
    1 FROM sys.dummy1;

Modified 1 row
```

Convert a Timestamp in a Query Filter

Use timestamp, a numeric constant value in local Chicago time, as a filter on the column `ts` in the `test_timezone` table.

SQL

```
SELECT * FROM test_timezone
    WHERE ts=CONVERT_UTC_TIMESTAMP_TO_LOCAL(Timestamp '2022-11-06 14:09:00', 'Ameri
```

Output

Text

```
ts                                              b
-------------------------------------------------------
2022-11-06 08:09:00.000000000                   1
```

Use timestamp, a runtime value from the table column `ts` in local Chicago time, as a filter on the `test_timezone` table.

SQL

```
SELECT * FROM test_timezone
    WHERE CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago')='2022-11
```

Output

Text

```
ts                                              b
-------------------------------------------------------
2022-11-06 08:09:00.000000000                   1
```

Convert a Timestamp in the Grouping Part of the Query

Group the query results by a timestamp converted to local time in Chicago.

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago')
    FROM test_timezone GROUP BY CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'Amer
```

Output

© 2026 Ocient, Inc.

Text

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, ('America/Chicago'))
-----------------------------------------------------
2022-11-05 04:09:00.000000000
2022-11-09 02:09:00.000000000
2022-11-04 04:09:00.000000000
2022-11-06 02:09:00.000000000
```

Using a grouping, omit the `2022-11-10 02:09:00` timestamp using the `HAVING` clause.

SQL

```
SELECT CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago')
    FROM test_timezone GROUP BY CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'Amer
    HAVING CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts::timestamp, 'America/Chicago') != '202
```

Output

Text

```
CONVERT_UTC_TIMESTAMP_TO_LOCAL(ts, ('America/Chicago'))
-----------------------------------------------------
2022-11-06 02:09:00.000000000
2022-11-04 04:09:00.000000000
2022-11-05 04:09:00.000000000
```

CONVERT_LOCAL_TIMESTAMP_TO_UTC

The function converts a timestamp in a specified local time zone to the UTC time zone. In the case of ambiguous timestamps, the database returns the earliest of the two possible times in the UTC time zone. Ambiguous timestamps refer to the times when the DST time zone change happens, and 1 hour can transform to either 1 or +1 and 1 or -1. Ambiguous timestamps can also refer to non-existent times, such as `2023-03-12 02:01:00 'America/New_York'`, which are similarly converted to the earliest possible UTC time, `2023-03-12 07:00:00 'UTC'`, due to the skipping of hour `02:00:00` entirely with DST spring forward.

This function can take the `timezone` argument as a `VARCHAR` literal, or the argument can be a column. When the argument is a literal, you can arbitrarily capitalize it. When the argument is a column, the values in the column must be capitalized according to the canonical IANA time zone names, such as `'America/Chicago'`.

Syntax

SQL

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(timestamp,timezone)
```

Argument	Data Type	Description
`timestamp`	`TIMESTAMP` or `VARCHAR`	Timestamp to convert that you specify as a `TIMESTAMP` or `VARCHAR` that you typecast in a timestamp format.
`timezone`	`VARCHAR`	`VARCHAR` that contains the full name of the target time zone.

Examples

Convert a Constant Time Zone Value

Convert timestamp `2022-11-06 02:09:00` to Chicago UTC time.

SQL

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-06 02:09:00','America/Chic
```

Output

Text

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(timestamp(('2022-11-06 02:09:00')), ('America/Chicag
--------------------------------------------------------------------------------
2022-11-06 08:09:00.000000000
```

Convert a Timestamp Value in a Column

Convert the timestamp in the `ts` column to Chicago UTC time using the `test_timezone` table.

SQL

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts,'America/Chicago') FROM test_timezone;
```

Output

Text

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, ('America/Chicago'))
-------------------------------------------------------
2023-01-09 17:23:12.173997444
```

Convert a Timestamp Value and Time Zone Name in Columns

Convert the timestamp in the `ts` column to the UTC time specified by the `timezone_name` column using the `test_timezone` table.

SQL

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts,timezone_name) FROM test_timezone;
```

Output

Text

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, timezone_name)
-------------------------------------------------
2023-01-09 17:43:21.193229991
```

Convert a Timestamp Value Using Other Date and Time Functions

Use the `TO_TIMESTAMP` function to convert a numeric timestamp to the Chicago UTC time.

SQL

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(TO_TIMESTAMP(1534587890), 'America/Chicago');
```

© 2026 Ocient, Inc.

Output

```
Text

CONVERT_LOCAL_TIMESTAMP_TO_UTC(to_timestamp((1534587890)), ('America/Chicago'))
-------------------------------------------
2018-08-18 15:24:50.000000000
```

Use a local timestamp from the output of the `CONVERT_UTC_TIMESTAMP_TO_LOCAL` function with Chicago time and convert it back to UTC time.

```
SQL

SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(TO_TIMESTAMP(CONVERT_UTC_TIMESTAMP_TO_LOCAL(T
    'America/Chicago');
```

Output

```
Text

CONVERT_LOCAL_TIMESTAMP_TO_UTC(CONVERT_UTC_TIMESTAMP_TO_LOCAL(timestamp(('2022-11-0
-------------------------------------------------------------------------------
2022-11-06 01:59:00.000000000
```

Convert a Timestamp During Table Creation

Create a table using a `SELECT` statement that converts a timestamp in Chicago UTC time.

```
SQL

CREATE TABLE test_timezone (ts timestamp, b int)
    AS (SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-09 02:09:00', 'Ame
    1 FROM sys.dummy1);

Modified 1 row

SELECT * FROM test_timezone;
```

Output

Text

```
ts                                              b
-------------------------------------------------
2022-11-09 08:09:00.000000000                   1
```

Convert a Timestamp During an Insert Operation

Insert a timestamp in Chicago UTC time into the `test_timezone` table.

SQL

```
INSERT INTO test_timezone SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-
    1 from sys.dummy1;

Modified 1 row
```

Convert a Timestamp in a Query Filter

Use a UTC timestamp in Chicago as a numeric constant value to filter on the column `ts` in the `test_timezone` table.

SQL

```
SELECT * FROM test_timezone
    WHERE ts=CONVERT_LOCAL_TIMESTAMP_TO_UTC(Timestamp '2022-11-06 02:09:00', 'Ameri
```

Output

Text

```
ts                                              b
-------------------------------------------------
2022-11-06 08:09:00.000000000                   1
```

Use a timestamp in Chicago UTC time, a runtime value from the table column `ts`, as a filter on the `test_timezone` table.

SQL

```
SELECT * FROM test_timezone
    WHERE CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago')='2022-11
```

Output

Text

```
ts                                              b
------------------------------------------------------
2022-11-06 08:09:00.000000000                   1
```

Convert a Timestamp in the Grouping Part of the Query

Group the query results by a timestamp converted to UTC time in Chicago.

SQL

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago')
    FROM test_timezone GROUP BY CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'Amer
```

Output

Text

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, ('America/Chicago'))
-------------------------------------------------------
2022-11-09 14:09:00.000000000
2022-11-06 14:09:00.000000000
2022-11-04 14:09:00.000000000
2022-11-05 14:09:00.000000000
```

Using a grouping, omit the `2022-11-05 14:09:00` timestamp using the `HAVING` clause.

© 2026 Ocient, Inc.

SQL

```
SELECT CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago')
    FROM test_timezone GROUP BY CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'Amer
    HAVING CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts::timestamp, 'America/Chicago') != '202
```

Output

Text

```
CONVERT_LOCAL_TIMESTAMP_TO_UTC(ts, ('America/Chicago'))
-------------------------------------------------------
2022-11-09 14:09:00.000000000
2022-11-06 14:09:00.000000000
2022-11-04 14:09:00.000000000
```

Related Links

Data Query Language (DQL) Statement Reference

Date and Time Functions

Conversion Functions

CREATE TABLE

INSERT INTO TABLE

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Tuple Functions and Operators

Tuple Functions

Function	Syntax	Purpose
Constructor	tuple<<TYPE1,TYPE2, ..., TYPEn>>(e1, e2, ..., en)	Construct a tuple given elements. NULL is also supported as an element.
Implicit Constructor	tuple(e1, e2, ..., en)	Construct a tuple with the specified elements. Types of the tuple are inferred from the inner elements.
convert tuple to string	char(tuple)	Converts a tuple to its string representation.
convert string to tuple	string_to_tuple(str, format)	Converts the string representation of a tuple (e.g., 'tuple<<INT,BIGINT,CHAR>>(1,2,NULL)') into a tuple.
tuple casting	cast_to_tuple(tuple, format)	Converts a tuple into another tuple of a different type (e.g., cast_to_tuple(tuple<<int,int>>(1,2), 'TUPLE(BIGINT,DOUBLE)')).

Tuple Function Examples

Function	SQL statement	Result
type constructor	tuple<<INT,BIGINT>>(1,2)	tuple holding INT(1) and BIGINT(2)
type constructor for array of tuples	tuple<<INT,BIGINT>>[](tuple<<INT,BIGINT>>(1,2), tuple<<INT,BIGINT>>(3,4))	array holding 2 tuples: [(1,2), (1,2)]
char(<tuple value>)	char(tuple<<INT,BIGINT>>(1,2))	'tuple<<INT,BIGINT>>(1,2)'
string_to_array	string_to_array('tuple<<INT,BIGINT>>[tuple<<INT,BIGINT>>(1,2)]','ARRAY(TUPLE(INT,BIGINT))')	tuple<<INT,BIGINT>>[tuple<<INT,BIGINT>>(1,2)]

Tuple Operators

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Operator	Syntax	Description
access operator	tuple[element_index]	Returns the element stored in the specified index position. The index starts at `1`. Access out of bounds returns a NULL.
slice operator	tuple[left_index:right_index]	Returns the subtuple ranging from left_index to right_index inclusive. The index starts at `1`.

About the Tuple Slice Operator

- Each index starts at `1`.
- If the tuple is NULL or either index value is NULL, the result of the slicing is NULL.
- If you omit the left index, it is equivalent to using `1`. If you omit the right index, it is equivalent to `len(tuple)`. If you omit both, it returns the full tuple. `tuple<<int,int,int>>(1, 2, 3)[:]` is equivalent to `tuple<<int,int,int>>(1, 2, 3)`.
- Values below zero for the left index convert to `1`, and values beyond the tuple length for the right index convert to `len(tuple)`. `tupleVal[-1:4]` is equivalent to `tupleVal[1:4]`, and `tuple<<int,int,int>>(1, 2, 3)[1:6]` is equivalent to `tuple<<int,int,int>>(1, 2, 3)[1:3]`.
- Ranges completely out of tuple bounds throw an invalid argument error, e.g., `tuple<<int,int>>(1,2)[4:5]`.
- Sequential slices slice each dimension of tuples. `tuple<<tuple<<int,int,int>>,tuple<<int,int,int>>>>(tuple<<int,int,int>>(1, 2, 3), tuple<<int,int,int>>(4, 5, 6))[1:1][2:3]` = `tuple<<tuple<<int,int>>>>(tuple<<int,int>>(2, 3))`
- More than N sequential slices of an N-dimensional tuple, for example, three sequential slices on a two-dimensional tuple throw an error. `tuple<<int>>(1)[:][:]`
- You cannot combine slicing with the access operator. If you attempt such a combination, any access operator `[n]` converts to `[:n]`. `tupleVal[4][1:6]` = `tupleVal[:4][1:6]`.
- Slicing with non-constant indexes throws an error.
- Slicing a tuple of arrays such as `tuple<<int[],double[]>>(int[1,2,3],double[4,5,6])[1:1][2:3]` slices `[1:1]` on the tuple and `[2:3]` on the resulting inner arrays, resulting in `tuple<<int[]>>(int[2,3])`.

- Slicing a tuple with a mix of containers and primitives such as `tuple<<int,int[]>>(1,int[1,2,3])` throws an error if, at any given depth, the slice is not valid for one or more elements. Slicing the tuple with `[:][2:3]` fails because you cannot slice an integer, and the resultant tuple from `[:]` includes an integer. However, slicing with `[2:2][2:3]` returns `tuple<<int[]>>(int[2,3])` because the tuple resulting from the `[2:2]` slice contains only elements that the next slice range can slice.

Operator Examples

Operator	SQL Predicate	Result
access operator	tuple<<CHAR,BIGINT>>('SQL',2)[1]	'SQL'
access operator	tuple<<CHAR,BIGINT>>('SQL',2)[2]	NULL
slice operator	tuple<<CHAR,BIGINT,BIGINT>>('SQL',1,2)[1:2]	tuple<<CHAR,BIGINT>>('SQL',1)

Related Links

Array Functions and Operators

Matrix Functions and Operators

Math Functions and Operators

Data Types

Data Types for Data Pipelines

Query Ocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Window Aggregate Functions

Window functions operate on a set of rows and return a single value for each row from the underlying query. Like aggregate functions, window functions operate on a set of rows, but they do not reduce the number of rows returned by the query.

Unlike regular aggregate functions, window functions do not group rows into a single output row. Instead, the rows retain their separate identities, and the window function produces aggregate calculations that access more than just the current row in the query result.

The term window describes the set of rows on which the window function operates. A window function returns a value from the rows in a window.

The database uses the `ORDER BY` and `PARTITION BY` clauses to define the scope of the window.

Syntax

Window Aggregate functions follow this syntax with `window_function` representing any of the functions listed in **Supported Window Aggregate Functions**.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

SQL

```
window_function( arg1, arg2 [,...] [ IGNORE NULLS ] )
   OVER (
      [ PARTITION BY part_list ]
      [ ORDER BY order_list [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] ]
      [ <ROWS_or_RANGE_clause> ] )

<ROWS_or_RANGE_clause> ::=
{ ROWS | RANGE }
   { <window_frame_preceding>
   | BETWEEN { <window_frame_preceding> | <window_frame_following> }
   AND { <window_frame_preceding> | <window_frame_following> }
     }

<window_frame_preceding> ::=
{ UNBOUNDED PRECEDING | row_integer PRECEDING | CURRENT ROW }

<window_frame_following> ::=
{ UNBOUNDED FOLLOWING | row_integer FOLLOWING | CURRENT ROW }
```

Argument	Data Type	Description
`arg1, arg2 [,...]`	String	The arguments of the window function. The arguments depend on the type of window function.

Parameter	Data Type	Description
`part_list`	String	Specifies the column by which the rows are partitioned for the window function.
`order_list`	String	Specifies the column by which the rows are ordered.
`row_integer`	Numeric	An integer representing the number of rows to operate from the current row.

Window Aggregate Keywords

Window aggregate functions use these clauses and keywords to designate the parameters for the query and output.

PARTITION BY

This clause defines a set of rows within a result set. `PARTITION BY` operates similarly to a **GROUP BY** clause, except it does not reduce the number of rows in the result set.

ORDER BY

Sorts the result set in ascending or descending order based on one or more specified columns.

An `ORDER BY` clause must be included on all window functions to ensure your query results are deterministic. Some window functions require an `ORDER BY` clause. See the syntax for each function for specifics. For details, see **ORDER BY**.

ASC or DESC

Specifies whether the values in the specified column should be sorted in `ASC` ascending or `DESC` descending order. `ASC` is the default sort order.

NULLS FIRST, NULLS LAST, or IGNORE NULLS

Specifies if the database sorts NULL rows first `NULLS FIRST`, last `NULLS LAST`, or whether the database ignores NULLs `IGNORE NULLS`. The default is `NULLS LAST`.

ROWS or RANGE

Limit the rows within the partition by specifying start and end points within the partition. Specify a range of rows with respect to the current row by logical association using `RANGE` or physical association using `ROWS`.

The `RANGE` clause logically limits the rows within a partition by specifying a range of values with respect to the value in the current row. The system defines preceding or following rows based on the ordering in the `ORDER BY` clause. If you do not specify this clause, the default is ascending order.

The `ROWS` clause limits the rows within a partition by specifying a fixed number of rows preceding or following the current row.

© 2026 Ocient, Inc.

Specifying a window frame by the `RANGE` keyword with the current row `CURRENT ROW` keyword includes all rows with the same values as the current row in the `ORDER BY` clause.

`ROWS` and `RANGE` clauses can include these options.

Keyword Clause	Description
`UNBOUNDED PRECEDING`	Specifies that the window starts at the first row of the partition. This clause specifies a window starting point only.
`UNBOUNDED FOLLOWING`	Specifies that the window ends at the last row of the partition. This clause specifies a window endpoint only. For example: `RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING` This window frame starts with the current row and ends with the last row of the partition.
`row_integer PRECEDING`	The `row_integer` value specifies the number of rows or values to precede the current row. You cannot use the `PRECEDING` keyword in a `RANGE` clause.
`row_integer FOLLOWING`	The `row_integer` value specifies the number of rows or values to follow the current row. When you specify the `FOLLOWING` keyword as the window starting point, the endpoint must also use a `FOLLOWING` clause with a row value. For example: `ROWS BETWEEN 2 FOLLOWING AND 10 FOLLOWING` This window frame starts with the second row following the current row and ends with the tenth row following the current row. You cannot use the `FOLLOWING` keyword in a `RANGE` clause.
`CURRENT ROW`	When you specify the `CURRENT ROW` keyword in a `ROWS` clause, the window frame starts or ends at the current row. When you specify this keyword in a `RANGE` clause, the window frame includes all rows with equal values. For example: `ROWS BETWEEN 2 PRECEDING AND CURRENT ROW` This clause means that the window function operates on a frame of three rows in size, where the first two rows precede the current row (inclusive). You can specify the current row as either a starting or an end point.

Supported Window Aggregate Functions

These functions are supported for window frames. Some examples use the `sys.dummy` virtual table to populate data. For details, see **Generate Tables Using sys.dummy**.

You cannot use the `DISTINCT` keyword with window aggregation functions.

CUME_DIST

Returns a positive value less than one that represents a percentage of values that are less than or equal to the current value in the group.

Syntax

SQL

```
CUME_DIST ()
   OVER ( [ PARTITION BY part_list ] ORDER BY order_list )
```

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example shows the cumulative distribution values for a table with generated numbers from one to 10.

SQL

```
SELECT x, CUME_DIST()
    OVER (ORDER BY x) AS d
    FROM (SELECT c1 AS x FROM sys.dummy10);
```

Output

© 2026 Ocient, Inc.

Text

```
| "x" | "d" |
|-----|-----|
|   1 | 0.1 |
|   2 | 0.2 |
|   3 | 0.3 |
|   4 | 0.4 |
|   5 | 0.5 |
|   6 | 0.6 |
|   7 | 0.7 |
|   8 | 0.8 |
|   9 | 0.9 |
|  10 | 1.0 |
```

DELTA

Computes the finite difference between successive values of expression under the specified ordering. This is a backward difference, meaning the delta value for a specified row is the difference between the value of that row and the previous row.

The function requires two rows of input data to calculate a first-degree difference, three rows of input data to calculate a second-degree difference, and so on.

However, rather than returning NULL for the first few rows, the function returns the correct values, such that using DELTA of degree `n` followed by DELTA of degree `-n` cancel each other out to return the initial result (other than floating point error accumulation). In general, using this window function over larger result sets with large degrees worsens floating-point error accumulation.

Integer degree values in the range `[-3, 3]` are generally stable up to approximately 5 million rows. Fractional degree values in the range `[-3, 3]` are generally stable up to approximately 1 million rows. The stable degree range is wider over smaller result sets and narrower over larger result sets. For example, fractional and integer degrees are stable over the range `[-4, 4]` up to approximately 80,000 rows.

⚠ The function treats any NULL values in the input data as zero, which might cause unexpected results. Filter out NULL values before you use the window function.

© 2026 Ocient, Inc.

Syntax

```SQL
DELTA ( expression [, degree ])
   OVER (
      [ PARTITION BY part_list ]
      ORDER BY order_list )
```

Argument	Data Type	Description
`expression`	Numeric	The column or expression to use for the DELTA calculation.
`degree`	Numeric	Optional. If `degree` is set to a positive integer greater than 1, the window function calculates the higher-order finite difference without using nested `DELTA` calls. If negative, the calculation is an anti-difference (i.e., discrete integration). You can specify fractional values, either positive or negative, and these values mean fractional differences or sums. The default is `1`.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Examples

Calculate Difference Values Using Basic Numbers

This example shows the delta values for a table with generated numbers from one to 10.

```SQL
SELECT y, DELTA(y, 1)
    OVER (ORDER BY y) AS d
    FROM (SELECT DOUBLE(c1 * c1) AS y FROM sys.dummy10);
```

© 2026 Ocient, Inc.

Output

Text

```
| "y" | "d" |
|-----|-----|
| 1   | 1   |
| 4   | 3   |
| 9   | 5   |
| 16  | 7   |
| 25  | 9   |
| 36  | 11  |
| 49  | 13  |
| 64  | 15  |
| 81  | 17  |
| 100 | 19  |
```

Calculate Difference Values Using Partitioned Sales Data

This example uses the `DELTA` function with a table of regional sales data to find the difference in sales by region and year.

SQL

```
SELECT
    region,
    year,
    sales_amount,
    DELTA(sales_amount) OVER (PARTITION BY region ORDER BY year) AS sales_delta
FROM
    regional_sales
ORDER BY
    region,
    year;
```

Output

Text

```
| "region" | "year" | "sales_amount" | "sales_delta" |
|----------|--------|----------------|---------------|
| North    |   2018 |          12000 |         12000 |
| North    |   2019 |          15000 |          3000 |
| North    |   2020 |          17000 |          2000 |
| North    |   2021 |          16000 |         -1000 |
| North    |   2022 |          18000 |          2000 |
| South    |   2018 |          10000 |         10000 |
| South    |   2019 |          14000 |          4000 |
| South    |   2020 |          15000 |          1000 |
| South    |   2021 |          17000 |          2000 |
| South    |   2022 |          16000 |         -1000 |
```

DENSE_RANK

Assigns a rank to each row in the result set with equal values having the same number. There are no gaps between ranks.

Syntax

SQL

```
DENSE_RANK ()
   OVER ( [ PARTITION BY part_list ] ORDER BY order_list )
```

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example generates the rank for a series of ten random numbers. Use the **RAND** function to generate the random numbers.

SQL

```sql
WITH RandomNumbers AS (
    SELECT RAND() AS num
    FROM sys.dummy10
)
SELECT num, DENSE_RANK()
    OVER (ORDER BY num) AS n
    FROM RandomNumbers;
```

Output

Text

```
|          num           |  n  |
| ---------------------- | --- |
| 0.021744504292815926 | 1   |
| 0.11531389663905277  | 2   |
| 0.17818297230968436  | 3   |
| 0.3009914646699445   | 4   |
| 0.366713904443239    | 5   |
| 0.7801256114813641   | 6   |
| 0.8735727062571587   | 7   |
| 0.8969752860565523   | 8   |
| 0.9679736914744647   | 9   |
| 0.9910291449541664   | 10  |
```

DERIVATIVE

The function calculates the difference between successive values of two numeric expressions, representing a backward difference (e.g., the difference between the current and previous row). It can be considered as computing the derivative of one expression with respect to another.

The function must use an `ORDER BY` clause, and it cannot use a `PARTITION BY` clause.

`DERIVATIVE` requires two arguments, `expression` and `expression2`, which must be included in the `ORDER BY` clause.

The function computes the difference in `expression` values between consecutive rows and divides this by the difference in `expression2` values over the same rows. If `expression` represents some measured value and `expression2` is time, the result is the rate of change (derivative) of the value with respect to time.

The function can compute higher-degree derivatives (second, third, etc.). To calculate the `nth`-degree derivative, it uses a backward difference formula that compares the current row to the previous `n` rows.

For example:

- A second-degree derivative requires three rows of data: the current row, the previous row, and the one before the previous row.
- A third-degree derivative would require four rows, and so on.

Unlike some functions that return NULL for rows that lack sufficient preceding data, the `DERIVATIVE` function is designed to return correct values even for initial rows. This behavior ensures that if you apply a derivative of degree `n` followed by its inverse (derivative of degree `-n`), you recover the original values with minimal error (except for floating point inaccuracies).

In general, using this function over larger result sets with larger degrees worsens the floating-point error. Integer degree values in the range of three are generally stable up to approximately 5 million rows. Fractional degree values in the range of three are generally stable up to approximately 1 million rows. The stable degree range is wider over smaller result sets and narrower over larger result sets. For example, fractional and integer degrees are stable over the range `[-4, 4]` up to approximately 80,000 rows.

⚠ For higher-order calls, the sample points for `expression` must be equally spaced, or the function might return incorrect results. In other words, the DELTA executions between consecutive `expression2` values must always be the same. If this is not true, use some form of interpolation to create an evenly-spaced series of samples before you use the window function.

⚠ The function treats any NULL values in the input data for either `expression` or `expression2` arguments as zero, which might cause unexpected results. Filter out NULL values before you use the window function.

Syntax

SQL

```
DERIVATIVE ( expression [, degree ] )
   OVER ( ORDER BY expression2 )
```

Argument	Data Type	Description
`expression`	Numeric	The column or expression to use for the `DERIVATIVE` calculation compared to `expression2`.
`expression2`	Numeric	The column or expression to use for the `DERIVATIVE` calculation compared to `expression`.
`degree`	Numeric	Optional. If you set `degree` to a positive integer greater than one, the window function calculates the higher-order difference quotients (discrete derivatives) without using nested `DERIVATIVE` executions. If the `degree` argument is a negative integer, the calculation is a discrete anti-differentiation or integration. The default value is `1`.

Examples

Calculate the Derivative Using Basic Numbers

This example shows the derivative values for a basic table with values `x` and `y`.

SQL

```
SELECT x, y, DERIVATIVE(y, 1)
    OVER (ORDER BY x) AS d
    FROM (SELECT DOUBLE(2 * x + 1) AS y, x
        FROM (SELECT c1/2.0 AS x FROM sys.dummy10));
```

Output

Text

```
| "x" | "y"  | "d" |
|-----|------|-----|
| 0.5 |  2.0 | 4.0 |
| 1.0 |  3.0 | 2.0 |
| 1.5 |  4.0 | 2.0 |
| 2.0 |  5.0 | 2.0 |
| 2.5 |  6.0 | 2.0 |
| 3.0 |  7.0 | 2.0 |
| 3.5 |  8.0 | 2.0 |
| 4.0 |  9.0 | 2.0 |
| 4.5 | 10.0 | 2.0 |
| 5.0 | 11.0 | 2.0 |
```

Calculate the Antiderivative Using a Negative Degree

This example uses a negative degree to produce an antiderivative.

SQL

```sql
SELECT x, y, DERIVATIVE(y, -1)
    OVER (ORDER BY x) AS d
    FROM (SELECT DOUBLE(2 * x + 1) AS y, x
        FROM (SELECT c1/2.0 AS x FROM sys.dummy10));
```

Output

Text

```
| "x" | "y"  | "d"  |
|-----|------|------|
| 0.5 |  2.0 |  1.0 |
| 1.0 |  3.0 |  2.5 |
| 1.5 |  4.0 |  4.5 |
| 2.0 |  5.0 |  7.0 |
| 2.5 |  6.0 | 10.0 |
| 3.0 |  7.0 | 13.5 |
| 3.5 |  8.0 | 17.5 |
| 4.0 |  9.0 | 22.0 |
| 4.5 | 10.0 | 27.0 |
| 5.0 | 11.0 | 32.5 |
```

FIRST_VALUE

Returns the first value in the ordered result set.

Syntax

SQL

```
FIRST_VALUE ( expression )
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`expression`	Any	The column or expression value to use for the `FIRST_VALUE` calculation.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example uses a table with generated numbers from one to 10. Of this group, the function determines the first value from the ordering.

SQL

```
SELECT y, FIRST_VALUE(y)
    OVER (ORDER BY y) AS first_value
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

Text

```
| "y" | "first_value" |
|-----|---------------|
|   1 |             1 |
|   2 |             1 |
|   3 |             1 |
|   4 |             1 |
|   5 |             1 |
|   6 |             1 |
|   7 |             1 |
|   8 |             1 |
|   9 |             1 |
|  10 |             1 |
```

LAG

Returns a row backward from the current row (one row back unless you specify a different number value).

For marking repeated LAG calls on one or more columns, see **Data Preparation**.

Syntax

SQL

```
LAG (value_expr [, offset ])
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`value_expr`	Any	The column or expression to use for the `LAG` calculation.
`offset`	Numeric	Optional. This number determines how many rows back from the current row to return a value. The default value is `1`.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Examples

Calculate Lag Using Basic Numbers

This example uses a table with generated numbers from one to 10. Of this group, the `LAG` function returns the last value for the specified column `y`.

SQL

```
SELECT y, LAG(y)
    OVER (ORDER BY y) AS wf
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

Text

```
| "y" | "wf" |
|-----|------|
|   1 |      |
|   2 |    1 |
|   3 |    2 |
|   4 |    3 |
|   5 |    4 |
|   6 |    5 |
|   7 |    6 |
|   8 |    7 |
|   9 |    8 |
|  10 |    9 |
```

Calculate Lag Using an Offset

In this example, the offset value `5` changes the number of rows back from the current row from which to return a value.

SQL

```sql
SELECT y, LAG(y, 5)
    OVER (ORDER BY y) AS wf
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

© 2026 Ocient, Inc.

Text

```
| "y" | "wf" |
|-----|------|
|   1 |      |
|   2 |      |
|   3 |      |
|   4 |      |
|   5 |      |
|   6 |    1 |
|   7 |    2 |
|   8 |    3 |
|   9 |    4 |
|  10 |    5 |
```

LAST_VALUE

Returns the last value in the ordered result set.

Syntax

SQL

```
LAST_VALUE ( expression )
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`expression`	Any	The column or expression value to use for the `LAST_VALUE` calculation.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example uses a table with generated numbers from one to 10. Of this group, the function determines the last value from the ordering.

SQL

```
SELECT y, LAST_VALUE(y)
    OVER (ORDER BY y) AS last_value
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

Text

```
| "y" | "last_value" |
|-----|--------------|
|   1 |            1 |
|   2 |            2 |
|   3 |            3 |
|   4 |            4 |
|   5 |            5 |
|   6 |            6 |
|   7 |            7 |
|   8 |            8 |
|   9 |            9 |
|  10 |           10 |
```

LEAD

Returns a row forward from the current row (one row forward unless you specify a different number value).

Syntax

SQL

```
LEAD (value_expr [, offset ])
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`value_expr`	Any	The value to use for the `LEAD` calculation.
`offset`	Numeric	Optional. This number determines how many rows forward from the current row to return a value. The default value is `1`.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Examples

Calculate the Lead Value Using Basic Numbers

This example uses a table with generated numbers from one to 10. Of this group, the `LEAD` function returns the last value for the specified column `y`.

SQL

```
SELECT y, LEAD(y)
    OVER (ORDER BY y) AS lead_value
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

Text

```
| "y" | "lead_value" |
|-----|--------------|
|   1 |            2 |
|   2 |            3 |
|   3 |            4 |
|   4 |            5 |
|   5 |            6 |
|   6 |            7 |
|   7 |            8 |
|   8 |            9 |
|   9 |           10 |
|  10 |              |
```

Calculate the Lead Value Using an Offset

In this example, using a table of generated numbers with values from one to 10, specify the offset value `5` in the `LEAD` function to change the number of rows forward from the current row to return a value.

SQL

```sql
SELECT y, LEAD(y, 5)
    OVER (ORDER BY y) AS lead_value
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

© 2026 Ocient, Inc.

Text

```
| "y" | "lead_value" |
|-----|--------------|
|   1 |            6 |
|   2 |            7 |
|   3 |            8 |
|   4 |            9 |
|   5 |           10 |
|   6 |              |
|   7 |              |
|   8 |              |
|   9 |              |
|  10 |              |
```

NTH_VALUE

Returns the `n`-th value in the ordered result set.

Syntax

SQL

```
NTH_VALUE (expr, offset)
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`expr`	Any	The column or expression from which to retrieve a value.
`offset`	Numeric	A positive integer that indicates the row number, starting from the first row in the window. This integer must be positive.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

© 2026 Ocient, Inc.

Examples

Calculate the Fifth Value Using Bounded Rows

This example uses a table with generated numbers from one to 10. Of this group, the `NTH_VALUE` function returns the fifth value for the specified column `y`.

For the first four rows, the window contains fewer than five rows, so the function returns NULL because the fifth value is not yet available.

SQL

```
SELECT y, NTH_VALUE(y, 5)
    OVER (ORDER BY y) AS nth
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

Text

```
| "y" | "nth" |
|-----|-------|
|   1 |       |
|   2 |       |
|   3 |       |
|   4 |       |
|   5 |     5 |
|   6 |     5 |
|   7 |     5 |
|   8 |     5 |
|   9 |     5 |
|  10 |     5 |
```

Calculate the Fifth Value Using Unbounded Rows

To include future rows in the window frame, you can modify the window frame as shown in this example.

© 2026 Ocient, Inc.

SQL

```
SELECT y, NTH_VALUE(y, 5)
    OVER ( ORDER BY y ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS
FROM
    (SELECT c1 AS y FROM sys.dummy10);
```

Output

Text

```
| "y" | "nth" |
|-----|-------|
|   1 |     5 |
|   2 |     5 |
|   3 |     5 |
|   4 |     5 |
|   5 |     5 |
|   6 |     5 |
|   7 |     5 |
|   8 |     5 |
|   9 |     5 |
|  10 |     5 |
```

PERCENTILE

Returns the value corresponding to the specified percentile (0 ≤ `n` ≤ 1) within the group.

The set of values is treated as a continuous distribution, so it is possible that the computed value cannot appear in the result set.

Syntax

SQL

```
PERCENTILE (expr, percentile)
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`expr`	Any	The column or expression to use for the `PERCENTILE` calculation.
`percentile`	Float	A percentile (0 ≤ `n` ≤ 1) to determine the value to return by its position within the ordering of `expr`.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example finds the value at approximately 90 percent of the range of the expression values. For the source data, use a table of generated numbers from one to 10.

The value returns `9.1` because the window column starts at `1`. It would be `9` if the column started at `0`.

SQL

```
SELECT y, PERCENTILE(y, 0.9)
    OVER (ORDER BY y) AS percentile_value
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

© 2026 Ocient, Inc.

Text

```
| "y" | "percentile_value" |
|-----|--------------------|
|   1 |                9.1 |
|   2 |                9.1 |
|   3 |                9.1 |
|   4 |                9.1 |
|   5 |                9.1 |
|   6 |                9.1 |
|   7 |                9.1 |
|   8 |                9.1 |
|   9 |                9.1 |
|  10 |                9.1 |
```

PERCENT_RANK

Returns a percentage value for each row representing the percentage of values less than the current group, excluding the highest value. The highest value in a group is always 1.

Syntax

SQL

```
PERCENT_RANK ()
    OVER ( [ PARTITION BY part_list ] ORDER BY order_list )
```

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example generates the percent rank of the table with ten generated numbers from one to 10.

SQL

```
SELECT y, PERCENT_RANK()
    OVER (ORDER BY y) AS percent_rank
    FROM (SELECT c1 AS y FROM sys.dummy10);
```

Output

Text

```
| "y" |   "percent_rank"   |
|-----|--------------------|
|   1 |                0.0 |
|   2 | 0.1111111111111111 |
|   3 | 0.2222222222222222 |
|   4 | 0.3333333333333333 |
|   5 | 0.4444444444444444 |
|   6 | 0.5555555555555556 |
|   7 | 0.6666666666666666 |
|   8 | 0.7777777777777778 |
|   9 | 0.8888888888888888 |
|  10 |                1.0 |
```

RANK

Assigns a rank number to each row in the result set with equal values having the same number. There can be gaps between ranks. Results are similar to the ranking used in sporting events.

Syntax

SQL

```
RANK ()
    OVER ( [ PARTITION BY part_list ] ORDER BY order_list )
```

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.

Parameter	Data Type	Description
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example ranks a series of ten random numbers. Use the **RAND** function to generate the random numbers.

SQL

```
WITH RandomNumbers AS (
    SELECT RAND() AS num
    FROM sys.dummy10
)
SELECT num, RANK() OVER (ORDER BY num) AS n
FROM RandomNumbers;
```

Output

Text

```
|  num                |  n  |
| ------------------- | --- |
| 0.0003210869634629  | 1   |
| 0.03250125317575761 | 2   |
| 0.09413438896602266 | 3   |
| 0.130728439355569   | 4   |
| 0.376078282337459   | 5   |
| 0.5049826734614208  | 6   |
| 0.5642048500743658  | 7   |
| 0.5989180494879046  | 8   |
| 0.9077625673606013  | 9   |
| 0.9428422123895421  | 10  |
```

RATIO_TO_REPORT

Computes the ratio of a value to the sum of the set of values. For example, the `RATIO_TO_REPORT` function calculates the ratio of the salary of an employee relative to the total salaries in the department where the employee belongs.

Syntax

```
SQL

RATIO_TO_REPORT ( ratio_expression )
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`ratio_expression`	Numeric	A column or expression to calculate the ratio for each row compared to the total.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example shows a ratio of each salary of an individual employee to the total salaries.

```
SQL

SELECT name, salaries, RATIO_TO_REPORT(salaries)
    OVER (ORDER BY salaries) AS ratio
    FROM salaries;
```

Output

Text

```
|      name       |  salaries |        ratio        |
|-----------------|-----------|---------------------|
| Vasquez, Gustav |   70000.0 | 0.08484848484848485 |
| Ho, Chen        |   75000.0 | 0.09090909090909091 |
| Kiwi, Marie     |  110000.0 | 0.13333333333333333 |
| Smith, Roberts  |  140000.0 |  0.1696969696969697 |
| Porter, Polly   |  180000.0 | 0.21818181818181817 |
| Svenson, Olaf   |  250000.0 | 0.30303030303030304 |
```

ROW_NUMBER

Assigns a unique number to each row in the result set.

Syntax

SQL

```
ROW_NUMBER ()
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Examples

Determine the Unique Number of Each Row in a Table

This example assigns a row number to a basic table of fruit prices based on the price order.

SQL

```
SELECT fruit, price, ROW_NUMBER()
OVER (ORDER BY price) AS row_no
    FROM fruits;
```

Output

Text

```
|  "fruit"   | "price" |"row_no"|
|------------|---------|--------|
| banana     |     0.5 |      1 |
| kiwi       |    0.75 |      2 |
| orange     |     1.5 |      3 |
| mango      |    1.75 |      4 |
| apple      |    1.99 |      5 |
| strawberry |    3.99 |      6 |
```

Determine the Unique Number of Each Row Using a Partitioned Table

This example uses a table of sales data for two regions. The `ROW_NUMBER` function orders the sales team members by their region and sales amounts.

SQL

```
SELECT
    salesperson,
    region,
    sales_amount,
    SUM(sales_amount) OVER (PARTITION BY region) AS total_sales_by_region,
    ROW_NUMBER() OVER (PARTITION BY region ORDER BY sales_amount DESC) AS sales_ran
FROM
    sales;
```

Output

Text

```
| "salesperson" | "region" | "sales_amount" | "total_sales_by_region" | "sales_rank
|---------------|----------|----------------|-------------------------|------------
| Eve           | North    |           2500 |                    5000 |
| Bob           | North    |           1500 |                    5000 |
| Alice         | North    |           1000 |                    5000 |
| Dave          | South    |           3000 |                    5000 |
| Carol         | South    |           2000 |                    5000 |
```

ZSCORE

Calculates the z-score of the sample based on the standard deviation (similar to the **STDDEV** aggregate function).

The standard deviation assumes that the data is a sample of a larger set, meaning it corrects for bias.

If your data set is comprehensive and represents an entire group, use the **ZSCOREP** function instead.

Syntax

SQL

```
ZSCORE (expr)
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`expr`	Numeric	A column or expression to calculate the standard deviation for each row.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example calculates the standard deviation in a range of salaries.

SQL

```
SELECT name, salaries, ZSCORE(salaries)
OVER (ORDER BY salaries) AS std_dev
    FROM salaries;
```

Output

Text

```
|      name       | salaries |      std_dev       |
|-----------------|----------|--------------------|
| Vasquez, Gustav |  70000.0 |                    |
| Ho, Chen        |  75000.0 | 0.7071076862849652 |
| Kiwi, Marie     | 110000.0 | 1.1470789011835154 |
| Smith, Roberts  | 140000.0 | 1.2593337909146212 |
| Porter, Polly   | 180000.0 | 1.4100478781464227 |
| Svenson, Olaf   | 250000.0 |  1.634473657490547 |
```

ZSCOREP

Calculates the z-score of the sample based on the population standard deviation (similar to the **STDEVP** aggregate function).

The population standard deviation assumes that the data contains the entire population, meaning there is no correction for bias. Use it when your data set is comprehensive and represents the entire group (i.e. population) you are analyzing.

If your data set is a sample of a larger set, use the **ZSCORE** function.

Syntax

SQL

```
ZSCOREP (expr)
    OVER ( [ PARTITION BY part_list ] [ ORDER BY order_list ] )
```

Argument	Data Type	Description
`expr`	Numeric	A column or expression to calculate the standard deviation for each row.

Parameter	Data Type	Description
`part_list`	Any	Specifies the column by which the rows are partitioned for the window function.
`order_list`	Any	Specifies the column by which the rows are ordered for the window function.

Example

This example calculates the population standard deviation in a range of salaries.

SQL

```
SELECT name, salaries, ZSCOREP(salaries)
OVER (ORDER BY salaries) AS std_devp
    FROM salaries;
```

Output

Text

```
|      name       | salaries |      std_devp      |
|-----------------|----------|--------------------|
| Vasquez, Gustav |  70000.0 |                    |
| Ho, Chen        |  75000.0 | 1.0000012800024576 |
| Kiwi, Marie     | 110000.0 | 1.4048790013060097 |
| Smith, Roberts  | 140000.0 | 1.4541534063682966 |
| Porter, Polly   | 180000.0 | 1.5764814535323706 |
| Svenson, Olaf   | 250000.0 | 1.7904761837110903 |
```

Related Links

Math Functions and Operators

Data Query Language (DQL) Statement Reference

Generate Tables Using sys.dummy

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Other Functions and Expressions

The Ocient® System supports these miscellaneous functions and expressions.

CASE

`CASE` operates similarly to conditional scripting in other programming languages, allowing it to function like an if / then / else statement or as a switch statement.

`CASE` sets one or more `WHEN` conditions to be evaluated. When a condition is evaluated as true, the database applies the specified action in a `THEN` clause. If no matches are found, the database applies the action in the `ELSE` clause. If no `ELSE` clause is specified and no matches are found, the result is `NULL`.

This evaluation creates a new column where the values are expressions inside the `THEN` clause.

Syntax

SQL

```
--To use CASE like an if / then statement:

CASE <caseCondition> ELSE expression | NULL END

<caseCondition> ::=
WHEN search_condition | expression THEN expression | NULL

--To use CASE like a switch statement:
CASE switch_expression <switchCondition> ELSE expression | NULL END

<switchCondition> ::=
WHEN switch_value THEN expression | NULL
```

Arguments	Description
`search_condition`	A condition including a column name and a Boolean expression to evaluate.
`switch_expression`	A column name to be used to evaluate a `switch_value`.

© 2026 Ocient, Inc.

Arguments	Description
`switch_value`	A value to be evaluated for the `switch_expression`.
`expression`	An expression to return based on evaluating the `search_condition` or `switch_expression`.

Example

This example uses conditional logic with `CASE` to categorize individuals into different groups based on their ages.

SQL

```
SELECT age,
  CASE
    WHEN age < 13 THEN 'child'
    WHEN age < 19 THEN 'teen'
    ELSE 'adult'
  END AS age_grp
FROM sys.people
ORDER BY age;
```

Example

This example uses switch conditions to categorize age groups.

SQL

```
SELECT age,
  CASE age
    WHEN 13 THEN 'thirteen'
    WHEN 14 THEN 'fourteen'
    WHEN 15 THEN 'fifteen'
    WHEN 16 THEN 'sixteen'
    WHEN 17 THEN 'seventeen'
    WHEN 18 THEN 'eighteen'
    WHEN 19 THEN 'nineteen'
    ELSE 'not teens'
  END
FROM sys.people;
```

© 2026 Ocient, Inc.

COALESCE

Returns the first argument that is not NULL or NULL if all arguments are NULL. Arguments must be the same type, and the returned value is the most compatible type.

`COALESCE` can work with mixed-type numeric arguments, which means the lower precedence types are cast to the highest one. For example, when using `TIMESTAMP` and `DATE` data types, all values are cast to a timestamp.

Syntax

SQL

```
COALESCE(value1, value2 [,...])
```

Argument	Data Type	Description
`value [,...]`	All data types are supported. All values must be of the same type.	Values for comparison to find the value that is not NULL.

Example

SQL

```
SELECT COALESCE(null,null,13,null,8);
```

Output: `13`

COMMIT

Returns the most recent commit hash of the database to which the client is currently connected.

Syntax

SQL

```
COMMIT()
```

Example

```sql
SELECT COMMIT();
```

Output: `cd0d53e3aa`

CURRENT_DATABASE

Alias for **DATABASE**.

CURRENT_NODE

Returns the name of the SQL Node where the current query executes. The name of the node corresponds to the `name` column in the `sys.nodes` system catalog table.

Syntax

```sql
CURRENT_NODE()
```

Example

```sql
SELECT CURRENT_NODE();
```

Output: `sql0`

CURRENT_GROUPS

Returns the fully qualified names of groups to which the current user belongs. You can query the `sys.groups` system catalog table to view the group information.

Syntax

```sql
CURRENT_GROUPS()
```

Example

```sql
SELECT CURRENT_GROUPS();
```

Output: `['sso_users@system', 'analysts@system']`

CURRENT_NODE_ID

Returns the identifier of the SQL Node where the current query executes. The name of the node corresponds to the `id` column in the `sys.nodes` system catalog table.

Syntax

```sql
CURRENT_NODE_ID()
```

Example

```sql
SELECT CURRENT_NODE_ID();
```

Output: `f690c14a-4e4f-4143-8e5e-20c90b60e15c`

CURRENT_SCHEMA

Returns the name of the current schema.

Syntax

```sql
CURRENT_SCHEMA()
```

Example

```sql
SELECT CURRENT_SCHEMA();
```

Output: `schema@database`

CURRENT_SESSION_ID

Returns the identifier of the current session as a Universally Unique IDentifier (UUID) that matches the `id` column in the `sys.sessions` system catalog table.

Syntax

```sql
CURRENT_SESSION_ID()
```

Example

```sql
SELECT CURRENT_SESSION_ID();
```

Output: `6ffe8383-e894-44c0-91c5-38d5925f51ec`

CURRENT_SYSTEM

Returns the name of the system. You can set the name using the **ALTER SYSTEM RENAME TO** SQL statement.

If you never set the system name, the Ocient System initializes the name of the system as a random identifier, the UUID.

Syntax

```
SQL

CURRENT_SYSTEM()
```

Example

```
SQL

SELECT CURRENT_SYSTEM();
```

Output: `2677b88c-7f07-470c-be3e-e1bafddc1221`

CURRENT_USER

Returns the user for the current connection.

Syntax

```
SQL

CURRENT_USER()
```

Example

```
SQL

SELECT CURRENT_USER();
```

Output: `user@database`

DATABASE

Alias for CURRENT_DATABASE.

Returns the name of the database to which the client is currently connected.

Syntax

```SQL
DATABASE()
```

Example

```SQL
SELECT DATABASE();
```

Output: `test`

GREATEST

Returns the largest non-NULL value of all the arguments, or NULL if all the arguments are NULL. All values must be comparable to each other.

Syntax

```SQL
GREATEST(value [,...])
```

Argument	Data Type	Description
`value [,...]`	All data types are supported. All values must either be of the same type or comparable to each other.	A series of values to compare to find the largest value of the set.

Example

```sql
SELECT GREATEST(12,13,5,8);
```

Output: `13`

IF_NULL

Returns the first argument that is not NULL or NULL if all arguments are NULL. Arguments must be the same type, and the returned value is the most compatible type.

`IF_NULL` can work with mixed-type numeric arguments, which means the lower precedence types are cast to the highest one. For example, when using `TIMESTAMP` and `DATE` data types, all values are cast to a timestamp.

Syntax

```sql
IF_NULL(value1, value2)
```

Argument	Data Type	Description
`value [,...]`	All data types are supported. All values must be of the same type.	Values for comparison to find the value that is not NULL.

Example

```sql
SELECT IF_NULL(null, 5)
```

Output: `5`

LEAST

Returns the smallest non-NULL value of all arguments, or NULL if all arguments are NULL. All values must be comparable to each other.

Syntax

```SQL
LEAST(value [,...])
```

Argument	Data Type	Description
`value [,...]`	All data types are supported. All values must either be of the same type or comparable to each other.	A series of values to compare to find the smallest value of the set.

Example

```SQL
SELECT LEAST(12,13,5,8);
```

Output: `5`

MURMUR3

Returns a 32-bit **MurmurHash3 hash** of the input value as an INTEGER data type. If you specify a NULL input value, the function returns 0. This function is primarily useful for partitioning data.

Syntax

```SQL
MURMUR3(value)
```

Argument	Data Type	Description
`value`	All SQL data types are supported.	Specified value for the MurmurHash3 hash function.

Example

```
SQL

SELECT MURMUR3(null);
```

Output: `0`

Example

```
SQL

SELECT MURMUR3('fred');
```

Output: `331477181`

NULL_IF

Returns the NULL value if two arguments are equal; otherwise, returns the first argument.

Syntax

```
SQL

NULL_IF(value1, value2)
```

Example

```
SQL

SELECT NULL_IF(13,12);
```

Output: `13`

© 2026 Ocient, Inc.

Example

```sql
SELECT NULL_IF(13,13);
```

Output: `NULL`

RAND_UUID

Alias for **UUID_GENERATE**.

SHOW

The `SHOW` function lets you explore the database and its metadata for user-defined items. For details, see **Information Schema**.

Syntax

```sql
SHOW <item>;

-- <item> can be any of the following:
{   COLUMNS,
    DATA_TYPES,
    DATABASES,
    GROUPS,
    INDEXES,
    RESERVED_WORDS,
    SCHEMATA,
    TABLES,
    USERS,
    VIEWS }
```

Examples

This example uses SHOW to explore databases on the system.

SQL

```
SHOW DATABASES;
```

Output

SQL

```
database_name
created
-------------------------------------------------------------------------------
test
2022-12-19 19:52:34.576426348
```

This example explores table data.

SQL

```
SHOW TABLES;
```

Output

SQL

```
table_catalog
table_schema
table_name
table_type
is_insertable_into
created_at
-------------------------------------------------------------------------------
test
test_schema
test_table
BASE TABLE
YES
2022-12-20 14:45:13.858144104
```

UUID_GENERATE

Alias for **RAND_UUID**.

Generates a random UUID value (version 4).

For details about the UUID data type, see **Data Types**.

Syntax

```
SQL

UUID_GENERATE()
```

Example

Generate one UUID value.

```
SQL

SELECT UUID_GENERATE();
```

Output: `3f6b2b58-4c9a-4a32-9b6a-b2f9b8e2a41e`

TYPE_STRIP

Returns the SQL type of the specified value.

Syntax

```
SQL

TYPE_STRIP(value)
```

Argument	Data Type	Description
`value`	All data types are supported.	Specified value for the determination of the type. You can specify columns, literals, and expressions.

Examples

```sql
SELECT TYPE_STRIP(_R{1,2,3});
```

Output: `MATRIX[1][3]`

```sql
SELECT TYPE_STRIP(current_user());
```

Output: `CHAR`

```sql
SELECT TYPE_STRIP(NULL);
```

Output: `NULL`

```sql
SELECT TYPE_STRIP(NULL::INT[]);
```

Output: `INT[]`

VERSION

Returns the version of the database to which the client is currently connected.

Syntax

```sql
VERSION()
```

Example

```sql
SELECT VERSION();
```

Output: `22.0.0-20230105.2353`

ZN

If `x` is NULL, returns 0. Otherwise, returns `x`.

Syntax

```sql
ZN(x)
```

Example

```sql
SELECT ZN(null);
```

Output: `0`

Example

```sql
SELECT ZN(12);
```

Output: `12`

Related Links

Databases

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Definition Language (DDL) Statement Reference

DDL statements allow users to run commands and administrative operations on the Ocient® System. You can execute DDL statements over a database connection to manage nodes, storage spaces, tables and views, system configuration, users and groups, underlying storage segments, and more.

> ⚠ **Quoting Identifiers**
>
> In all DDL statements, identifiers not in double quotes must begin with a letter and can only contain letters, numbers, and underscores. Identifiers are the names of databases, tables, nodes, and so on. For details, see **Identifiers**. The Ocient System internally converts identifiers to lower-case. Identifiers in double quotes can contain any characters besides newline and carriage return, can begin with any character, and are not case-adjusted. To use an identifier with the same name as any keyword, it must be in double quotes.
>
> String literals must be enclosed in single quotes. A single quote within a string can be escaped as ”*. *A string literal can be preceded by e to enable additional escape sequences (for example: `e'\n'`).

DDL statements are supported for these categories:

Cluster and Node Management

- **CLUSTER**
  - CREATE CLUSTER
  - DROP CLUSTER
  - ALTER CLUSTER
    - ALTER CLUSTER ADD PARTICIPANTS
    - ALTER CLUSTER DROP PARTICIPANTS
    - ALTER CLUSTER ADD STORAGESPACE
    - ALTER CLUSTER REMOVE STORAGESPACE
    - ALTER CLUSTER ALTER CONFIG SET

© 2026 Ocient, Inc.

        - ALTER CLUSTER ALTER LOG LEVEL SET
        - ALTER CLUSTER RENAME
- **STORAGESPACE**
    - CREATE STORAGESPACE
    - DROP STORAGESPACE
    - ALTER STORAGESPACE RENAME
- **NODE**
    - DROP NODE
    - ALTER NODE
        - ALTER NODE ADD ROLE
        - ALTER NODE REMOVE ROLE
        - ALTER NODE ALTER CONFIG SET
        - ALTER NODE RENAME
        - ALTER NODE ALTER LOG LEVEL SET
        - ALTER NODE ALTER METRIC LEVEL
        - ALTER NODE SET ADDRESS
- **CONNECTIVITY POOL**
    - CREATE CONNECTIVITY_POOL
    - DROP CONNECTIVITY_POOL
    - ALTER CONNECTIVITY_POOL
        - ALTER CONNECTIVITY_POOL SET
        - ALTER CONNECTIVITY_POOL RENAME TO
        - ALTER CONNECTIVITY_POOL ADD PARTICIPANTS
        - ALTER CONNECTIVITY_POOL ALTER PARTICIPANT
        - ALTER CONNECTIVITY_POOL DROP PARTICIPANTS
        - ALTER CONNECTIVITY_POOL SET SSO INTEGRATION
        - ALTER CONNECTIVITY_POOL REMOVE SSO INTEGRATION

- SSO INTEGRATION
  - CREATE SSO INTEGRATION
  - DROP SSO INTEGRATION
  - ALTER SSO INTEGRATION
- **SYSTEM**
  - ALTER SYSTEM ALTER CONFIG SET
  - ALTER SYSTEM RENAME TO
  - ALTER SYSTEM ALTER METRIC LEVEL
  - ALTER SYSTEM ALTER SECURITY
  - ALTER SYSTEM SET DEFAULT STORAGESPACE

Schemas

- CREATE SCHEMA
- DROP SCHEMA
- ALTER SCHEMA RENAME

Databases

- CREATE DATABASE
- DROP DATABASE
- ALTER DATABASE
  - ALTER DATABASE RENAME
  - ALTER DATABASE SET SSO INTEGRATION
  - ALTER DATABASE ALTER SSO INTEGRATION
  - ALTER DATABASE REMOVE SSO INTEGRATION
  - ALTER DATABASE ALTER SECURITY

Tables

- CREATE TABLE
- CREATE TABLE AS SELECT (CTAS)
- CREATE TABLE AS SELECT USING LOADERS
- DROP TABLE
- ALTER TABLE
  - ALTER TABLE RENAME
  - ALTER TABLE RENAME COLUMN
  - ALTER TABLE ADD COLUMN
  - ALTER TABLE ALTER COLUMN COMPRESSION
  - ALTER TABLE ALTER REDUNDANCY
  - ALTER TABLE DROP COLUMN
  - ALTER TABLE STREAMLOADER_PROPERTIES
  - ALTER TABLE DISABLE INDEX
  - ALTER TABLE ENABLE INDEX
  - ALTER TABLE ENABLE RETENTION POLICY AGE
  - ALTER TABLE DISABLE RETENTION POLICY
- DELETE FROM TABLE
- EXPORT TABLE
- INSERT INTO TABLE
  - INSERT INTO TABLE USING LOADERS
- TRUNCATE TABLE

Views

- CREATE VIEW
- DROP VIEW
- ALTER VIEW RENAME
- ALTER VIEW AS
- EXPORT VIEW

Indexes

- CREATE INDEX
- DROP INDEX

Data Pipelines

- PIPELINE
  - CREATE PIPELINE
  - DROP PIPELINE
  - PREVIEW PIPELINE
  - START PIPELINE
  - STOP PIPELINE
  - ALTER PIPELINE
  - ALTER PIPELINE RENAME
  - EXPORT PIPELINE
  - CREATE PIPELINE FUNCTION
  - DROP PIPELINE FUNCTION

Distributed Tasks

- **TASK**
  - CREATE TASK
  - CANCEL TASK

Machine Learning Models

- CREATE MLMODEL
- ALTER MLMODEL
- DROP MLMODEL
- EXPORT MLMODEL

© 2026 Ocient, Inc.

- REFRESH MLMODEL

Users, Groups, and Service Classes

- **USER**
  - CREATE USER
  - DROP USER
  - ALTER USER
    - ALTER USER SET
- **GROUP**
  - CREATE GROUP
  - DROP GROUP
  - ALTER GROUP
    - ALTER GROUP USER
    - ALTER GROUP RENAME
    - ALTER GROUP SET SERVICE CLASS
    - ALTER GROUP ALTER SECURITY
- **SERVICE CLASS**
  - CREATE SERVICE CLASS
  - DROP SERVICE CLASS
  - ALTER SERVICE CLASS
    - ALTER SERVICE CLASS RENAME
    - ALTER SERVICE CLASS SET
    - ALTER SERVICE CLASS RESET
  - ALTER QUERY

Data Integrity and Storage

- DRAIN PAGES

- ALTER SEGMENT QUARANTINE

Query Analysis

- EXPLAIN
- EXPLAIN PIPELINE

Query Management

- CANCEL
- KILL

Statistics Cache Management

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Cluster and Node Management

Cluster and Node Management commands allow administrators to manage storage clusters, create and manage storage spaces, create and manage nodes, and apply configuration changes to nodes, clusters, or the overall system.

CLUSTER

CREATE CLUSTER

`CREATE CLUSTER` creates a new cluster in the current database. The cluster name must be distinct from the name of any existing tables in the database.

To create a cluster, the logged-in user must be a system-level user.

For details about creating clusters, see **Ocient Application Configuration**.

> ⚠ For Foundation clusters, if the system is configured with a single-cluster configuration, when you add a second Foundation cluster, the system automatically changes to a multi-cluster configuration. This change is irreversible and requires a full system restart to fully take effect.

Syntax

```sql
CREATE CLUSTER [ IF NOT EXISTS ] cluster_name <create_definition>

<create_definition> ::=
   TYPE [=] foundation
   | PARTICIPANTS [=] (node_name1, node_name2, ...)
   | STORAGESPACE [=] storage_space_name
```

Parameter	Data Type	Description
`cluster_name`	string	The name of the new cluster to create.

© 2026 Ocient, Inc.

Define Clusters (`<create_definition>`)

You must include these configuration settings when you create a new cluster.

Configuration	Data Type	Description
`TYPE`	string	The supported value is: `foundation` — Processes query operators and handles storage on the system.
`PARTICIPANTS`	string	A list of one or more node names included in the new cluster. The number of participant nodes in a cluster must be equal to or greater than the storage-space width. For details, see **CREATE STORAGESPACE**. Adding more than the minimum number of participants to a cluster overprovisions the cluster and allows replacing or dropping participant nodes. Nodes cannot be assigned as participants to multiple clusters.
`STORAGESPACE`	string	The name of the storage space for the cluster.

Examples

This example creates a Foundation cluster named `lts_cluster0` using the storage space named `my_storage_space` with three participant nodes.

SQL

```
CREATE CLUSTER lts_cluster0
    TYPE=foundation
    PARTICIPANTS=(node1,node2,node3)
    STORAGESPACE=my_storage_space;
```

DROP CLUSTER

`DROP CLUSTER` removes an existing cluster from the system.

To remove a cluster, you must be a system-level user.

Syntax

SQL

```
DROP CLUSTER [ IF EXISTS ] cluster_name
```

Parameter	Data Type	Description
`cluster_name`	string	The name of the specified cluster to remove.

Example

This example removes an existing cluster named `lts_cluster0`.

SQL

```
DROP CLUSTER lts_cluster0;
```

ALTER CLUSTER

ALTER CLUSTER ADD PARTICIPANTS

`ALTER CLUSTER ADD PARTICIPANTS` adds the specified nodes to an existing cluster in the system. This operation can overprovision the storage space if necessary.

> For the system to recognize newly added nodes, you must restart the cluster. To do this, you can follow the steps in **Add Foundation Nodes**.

Syntax

SQL

```
ALTER CLUSTER [ IF EXISTS ] cluster_name
    ADD PARTICIPANTS [=] ( node_name1 [,...] );
```

Parameter	Data Type	Description
`cluster_name`	string	The name of the specified cluster to alter.
`node_name1, [,...]`	string	The name of one or more nodes to add to the cluster.

Example

This example adds the node `node3` to a cluster named `example_cluster`, which already contains `node1` and `node2`. Following this operation, `example_cluster` consists of three nodes.

SQL

```
ALTER CLUSTER example_cluster ADD PARTICIPANTS (node3);
```

ALTER CLUSTER DROP PARTICIPANTS

`ALTER CLUSTER DROP PARTICIPANTS` drops nodes from an existing cluster.

This SQL statement fails for a Foundation cluster. Dropping the specified nodes brings the number of participants below the total width of the storage space associated with that cluster.

Syntax

SQL

```
ALTER CLUSTER [ IF EXISTS ] cluster_name
    DROP PARTICIPANTS [=] ( node_name1 [,...] );
```

Parameter	Data Type	Description
`cluster_name`	string	The name of the specified cluster to alter.
`node_name1, [,...]`	string	The name of one or more nodes to drop from the cluster.

Example

This example drops participant node `node3` from a cluster named `example_cluster`, which contains `node1`, `node2`, and `node3`. Following this operation, the participants in `example_cluster` consist of the nodes `node1` and `node2`.

SQL

```
ALTER CLUSTER example_cluster DROP PARTICIPANTS (node3);
```

© 2026 Ocient, Inc.

ALTER CLUSTER ADD STORAGESPACE

`ALTER CLUSTER ADD STORAGESPACE` links an existing storage space to a storage cluster, enabling tables in that storage space to use the Foundation Nodes of that cluster. A storage space must be associated with at least one storage cluster before you can create tables in the space.

Required Privileges

To add a storage space to a cluster, you must have system-level privileges.

Syntax

```
SQL

ALTER CLUSTER [ IF EXISTS ] cluster_name ADD STORAGESPACE storage_space_name
```

Parameter	Type	Description
`cluster_name`	string	The name of the storage cluster to modify.
`storage_space_name`	string	The name of an existing storage space to associate with the cluster.

Example

This example adds the storage space `analytics_space` to the cluster `lts_cluster_0`.

```
SQL

ALTER CLUSTER lts_cluster_0 ADD STORAGESPACE analytics_space;
```

ALTER CLUSTER REMOVE STORAGESPACE

`ALTER CLUSTER REMOVE STORAGESPACE` detaches a storage space from a storage cluster. This detaching prevents the creation of new tables in that storage space within the cluster.

The statement fails if:

- Tables still exist in the storage space within the cluster. Remove all associated tables before detaching the storage space.
- The storage space is currently set as the system default storage space. Reset the default by using **ALTER SYSTEM SET DEFAULT STORAGESPACE** before detaching the storage space.
- You are attempting to detach the core system storage space: `"systemStorageSpace"`.

Required Privileges

To detach a storage space from a cluster, you must have system-level privileges.

Syntax

```SQL
ALTER CLUSTER [ IF EXISTS ] cluster_name REMOVE STORAGESPACE storage_space_name
```

Parameter	Type	Description
`cluster_name`	string	The name of the storage cluster to modify.
`storage_space_name`	string	The name of the storage space to detach from the cluster.

Example

This example detaches the storage space `analytics_space` from the cluster `lts_cluster_0`.

```SQL
ALTER CLUSTER lts_cluster_0 REMOVE STORAGESPACE analytics_space;
```

ALTER CLUSTER ALTER CONFIG SET

`ALTER CLUSTER ALTER CONFIG SET` sets a configuration override for the configuration at the cluster scope. For an explanation of how configuration overrides work, see **Inspect the Current Configuration**.

If the specified path to the configuration setting does not exist, the system considers it as a new configuration and appends it to the collection of existing configurations. To reset a configuration override for a specific key, use the `RESET` keyword.

Syntax

SQL

```
-- To set a new configuration override:
ALTER CLUSTER [ IF EXISTS ] cluster_name
    ALTER CONFIG SET string_definition [=] string_value [,...]

-- To reset a configuration override:
ALTER CLUSTER [ IF EXISTS ] clusterName
    ALTER CONFIG RESET [ string_definition [,...] ]
```

Parameter	Data Type	Description
`cluster_name`	string	The name of the specified cluster to alter.
`string_definition`	string	The name of a configuration parameter to alter. Contact Ocient® support for details about altering configuration parameters.
`string_value`	string	The new value to set for the specified configuration parameter.

Examples

This example alters the base configuration for cluster `cluster_name` and sets the certificate for all nodes to a certificate named `cert_name.crt`.

SQL

```
ALTER CLUSTER cluster_name
    ALTER CONFIG SET 'certificateStore.cert' = 'cert_name.crt';
```

This example resets the certificate configuration for cluster `cluster_name`.

SQL

```
ALTER CLUSTER cluster_name
    ALTER CONFIG RESET 'certificateStore.cert';
```

This example alters the configuration at the cluster scope for cluster `cluster_name` and service role `lts`.

SQL

```
ALTER CLUSTER cluster_name
    ALTER CONFIG SET 'lts.numLevels' = '3';
```

ALTER CLUSTER ALTER LOG LEVEL SET

`ALTER CLUSTER cluster_name ALTER LOG LEVEL SET` sets the log level for a particular logger for all nodes in the provided cluster. For an explanation of how configuration overrides work, see **Inspect the Current Configuration**.

Syntax

SQL

```
-- To set a new configuration override:
ALTER CLUSTER cluster_name
    ALTER LOG LEVEL SET [ <logger_parameters> [,...] ]

<logger_parameters> ::=
    { 'ALL' | logger_name [=] logging_level }

-- To reset a configuration override:
ALTER CLUSTER cluster_name ALTER LOG LEVEL RESET [logger_name [, ...] ]
```

Parameter	Data Type	Description
`cluster_name`	string	The name of the specified cluster to alter.

Logger Parameters (`<logger_parameters>`)

Configuration	Data Type	Description
`logger_name`	string	Use `ALL` for the root logging configuration. Otherwise, the `logger_name` parameter can be any string specifying a logger name for all nodes in the provided cluster.
`logger_level`	string	The level of severity to be assigned to the `logger_name`. `logger_level` must be one of the following options: `'EDEBUG' | 'DEBUG' | 'VERBOSE' | 'INFO' | 'WARN' | 'ERROR'` See the **Log Monitoring** page for more details on log levels. To reset a logger to the default log level (`INFO`), use the `RESET` keyword.

Examples

This example alters the log level at the cluster scope. This sets the query log level for all nodes in the cluster to `DEBUG`.

SQL

```
ALTER CLUSTER cluster_name
    ALTER LOG LEVEL SET 'query' = 'DEBUG';
```

This example resets the log level for the query logger for the entire cluster.

SQL

```
ALTER CLUSTER cluster_name
    ALTER LOG LEVEL RESET 'query';
```

ALTER CLUSTER RENAME TO

Rename an existing cluster in the system.

Syntax

SQL

```
ALTER CLUSTER old_cluster_name RENAME TO new_cluster_name
```

Parameter	Data Type	Description
`old_cluster_name`	string	The name of the specified cluster to alter.
`new_cluster_name`	string	The new name to assign to the cluster.

Example

This example renames an existing cluster named `lts-cluster-0` to `lts-cluster-1`.

SQL

```
ALTER CLUSTER "lts-cluster-0" RENAME TO "lts-cluster-1";
```

STORAGESPACE

CREATE STORAGESPACE

`CREATE STORAGESPACE` creates a new storage space. The name of the storage space must be distinct from the name of any existing storage spaces in the system.

To create a storage space, you must possess the `CREATE STORAGESPACE` privilege for the current system.

> It is recommended that the total width of a production system be smaller than the number of Foundation Nodes in the cluster so that loading can continue even in the event of a node outage.

Syntax

SQL

```
CREATE STORAGESPACE [ IF NOT EXISTS ] storage_space_name
    [ WIDTH [=] width_integer], [ PARITY_WIDTH[=] pw_integer ]
```

Parameter	Data Type	Description
`storage_space_name`	string	The name of the new storage space. A storage space name must begin with a letter followed by letters, numbers, and underscores.

Parameter	Data Type	Description
`width_integer`	integer	The total number of nodes to use in each segment group as it is written to disk. Width cannot be greater than the number of nodes in the storage cluster.
`pw_integer`	integer	The number of parity coding bits to use for each segment group. This number must be less than `width`. The specified storage space can still complete queries even if nodes are disabled as long as the number of disabled nodes does not exceed the `parity_width`. For example, a storage space with a `width = 5` and a `parity_width = 2` can have up to two nodes become disabled, and the system still uses the three remaining nodes to execute queries. However, if a third node goes down, the system can no longer process query operations.

Example

This example creates a new storage space named `ocient`.

SQL

```
CREATE STORAGESPACE ocient WIDTH = 10, PARITY_WIDTH = 2;
```

DROP STORAGESPACE

`DROP STORAGESPACE` removes an existing storage space, along with all associated tables and views. The SQL statement fails if one or more storage clusters are linked to the storage space.

To remove a storage space, you must possess the `DROP STORAGESPACE` privilege for the storage space. Note that this action cannot be undone.

Syntax

SQL

```
DROP STORAGESPACE [ IF EXISTS ] storage_space_name [, ...]
```

Parameter	Data Type	Description
`storage_space_name`	string	The name of a storage space to drop. You can drop multiple storage spaces by specifying additional storage space names and separating each with commas.

Example

Remove an existing storage space named `ocient_ss`.

SQL

```
DROP STORAGESPACE ocient_ss;
```

Remove multiple storage spaces.

SQL

```
DROP STORAGESPACE ocient_ss1, ocient_ss2;
```

ALTER STORAGESPACE

`ALTER STORAGESPACE RENAME` renames an existing storage space.

To rename a storage space, the logged-in user must be a system-level user.

Syntax

SQL

```
ALTER STORAGESPACE [ IF EXISTS ] old_storage_space_name RENAME TO new_storage_space
```

Parameter	Data Type	Description
`old_storage_space_name`	string	The name of a storage space to rename.
`new_storage_space`	string	The new name for the specified storage space.

Example

To rename an existing storage space named `teststoragespace` to `storage-space-1`.

```
SQL
ALTER STORAGESPACE "teststoragespace" RENAME TO "storage-space-1";
```

NODE

DROP NODE

`DROP NODE` removes an existing node from the system.

A node cannot be removed from the system if it is still part of a cluster or one of the two remaining Administration Nodes.

Syntax

```
SQL
DROP NODE [ IF EXISTS ] node_name [, ...]
```

Parameter	Data Type	Description
`node_name`	string	The name of a node to drop. You can drop multiple nodes by specifying additional node names and separating each with commas.

Example

Remove a node named `example_node`.

```
SQL
DROP NODE example_node;
```

Remove multiple nodes.

SQL

```
DROP NODE example_node1, example_node2;
```

ALTER NODE

ALTER NODE RENAME TO

`ALTER NODE RENAME` renames an existing node.

To rename a node, you must be a system-level user.

> ⚠ Renaming a node can cause a mismatch between the name of the node and the hostname of the node. Furthermore, you have to update users and scripts to use the new name of the node. Use this statement only to reverse these situations that can arise naturally (for example, you must replace a node with a node that has a different name).
>
> If you have a DNS configuration for the node, you must reconfigure the node address after renaming the node by using the **ALTER NODE SET ADDRESS** SQL statement.

Syntax

SQL

```
ALTER NODE [ IF EXISTS ] old_node_name RENAME TO new_node_name
```

Parameter	Data Type	Description
`old_node_name`	string	The name of a node to rename.
`new_node_name`	string	The new name for the specified node.

Example

This example renames an existing node named `sql0` to `sql1`.

SQL

```
ALTER NODE sql0 RENAME TO sql1;
```

ALTER NODE ADD ROLE

`ALTER NODE ADD ROLE` adds a role to a node. Each node can have one or more roles that it performs.

To add a role, you must be a system-level user.

Syntax

SQL

```
ALTER NODE [ IF EXISTS ] node_name ADD ROLE [ IF NOT EXISTS ] <node_role>

<node_role> ::=
{ sql | streamloader | operatorvm | admin | health }
```

Parameter	Data Type	Description
`node_name`	string	The name of a node to alter.

Node Roles (`<node_role>`)

This table explains the supported roles for nodes.

Role	Description	Notes
`streamloader`	The `streamloader` role allows the node to continually import a large amount of data into the database.	When adding the `streamloader` role, the node must be restarted in order for the change to take effect.
`sql`	The `sql` role accepts a SQL statement, parses the statement, and compiles the statement.	You can remove this role only if there is at least one other SQL Node present.
`operatorvm`	The `operatorvm` role manages database query plans and executes queries.	Added or removed automatically together with the `sql` or `lts` role.
`lts`	The nodes with the `lts` role store data on disk in segments in a column-oriented fashion. The `lts` role is associated with the Foundation Nodes on the system.	Can only be added or removed by adding or removing the node as a participant in a Foundation cluster.

© 2026 Ocient, Inc.

Role	Description	Notes
`admin`	The `admin` role accepts the administrative protocol and updates system metadata.	Can only be removed if there are at least two other Administrative Nodes.
`health`	A node with the `health` role is responsible for maintaining performance-related counters and statistics.	Present initially on every node. Cannot be removed.

Example

Add SQL role to the node `sql1`.

SQL

```
ALTER NODE sql1 ADD ROLE sql;
```

ALTER NODE REMOVE ROLE

`ALTER NODE REMOVE ROLE` removes a role from a node. For a list of roles, see **Node Roles**.

Some node roles cannot be removed using an `ALTER NODE` SQL statement, including `health`, `lts`, and `operatorvm`.

To remove a role, you must be a system-level user.

> The node must be restarted in order for this change to take effect.

> If you remove a SQL role from a SQL Node, the database removes the node from the connectivity pool. You must restart all SQL Nodes so that the other SQL Nodes understand that the node is no longer a SQL Node.

> When you remove the `admin` role, you must also remove the `/var/opt/ocient/metadataStorage.raft` file.

Syntax

```sql
ALTER NODE [ IF EXISTS ] node_name REMOVE ROLE [ IF EXISTS ]
    { sql | streamloader | admin }
```

Parameter	Data Type	Description
`node_name`	string	The name of a node to alter.

Example

This example removes the `sql` role from a node.

```sql
ALTER NODE my_node REMOVE ROLE sql;
```

ALTER NODE ALTER CONFIG SET

`ALTER NODE ALTER CONFIG SET` sets a configuration override for the configuration at the node scope. For an explanation of how configuration overrides work, see **Inspect the Current Configuration**.

If the specified path to the configuration setting does not exist, it will be considered a new configuration and appended to the collection of existing configurations. To reset a configuration override for a specific key, use the `RESET` keyword.

Syntax

```sql
-- To set a new configuration override:
ALTER NODE [ IF EXISTS ] node_name
    ALTER CONFIG SET string_definition [=] string_value [,...]

-- To reset a configuration override:
ALTER NODE [ IF EXISTS ] node_name ALTER CONFIG RESET string_definition [,...]
```

Parameter	Data Type	Description
`node_name`	string	The name of the specified node to alter.
`string_definition`	string	The name of a configuration parameter to alter. Contact Ocient Support for details about altering configuration parameters.
`string_value`	string	The new value to be set for the specified configuration parameter.

Examples

This example sets the certificate name for this node to `cert_name.crt`.

SQL

```
ALTER NODE node2 ALTER CONFIG SET 'certificateStore.cert' = 'cert_name.crt';
```

This example resets the certificate configuration for the node `node2`.

SQL

```
ALTER NODE node2 ALTER CONFIG RESET 'certificateStore.cert';
```

This example alters the configuration at the node scope for node `node2` for service role `lts`.

SQL

```
ALTER NODE node2 ALTER CONFIG SET 'lts.numLevels' = '3';
```

ALTER NODE ALTER LOG LEVEL SET

`ALTER NODE node_name ALTER LOG LEVEL SET` sets the log level for a particular logger on the specified node. For an explanation of how configuration overrides work, see **Inspect the Current Configuration**.

The logger name specified can be any string specifying a logger name, or ALL for the root logging configuration. See the **Log Configuration** for more details.

To reset a logger to the default log level (INFO), use the `RESET` keyword.

Syntax

SQL

```
-- To set a new configuration override:
ALTER NODE [ IF EXISTS ] node_name ALTER LOG LEVEL SET { <logger_parameters> [,...]

<logger_parameters> ::=
    'ALL' | logger_name [=] { 'EDEBUG' | 'DEBUG' | 'VERBOSE' | 'INFO' | 'WARN' | 'E

-- To reset a configuration override:
ALTER NODE [ IF EXISTS ] node_name ALTER LOG LEVEL RESET [ logger_name [,...] ]
```

Parameter	Data Type	Description
`node_name`	string	The name of the specified node to alter.
`<logger_parameters>`	string	Use `ALL` for the root logging configuration. Otherwise, the `logger_name` parameter can be any string specifying a logger name assigned to a severity level. See the **Log Configuration** page for more details. To reset a logger to the default log level (`INFO`), use the `RESET` keyword.

Examples

This example alters the log level at the node scope. The example sets the query log level for this particular node to `DEBUG` .

SQL

```
ALTER NODE node_name ALTER LOG LEVEL SET 'query' = 'DEBUG';
```

This example resets log level for the query logger on the specified node.

SQL

```
ALTER NODE node_name ALTER LOG LEVEL RESET 'query';
```

ALTER NODE ALTER METRIC LEVEL

Alters the reporting level of various internal metrics on a specific node. For details about internal metrics, see **Statistics Monitoring**.

To set metrics reporting on all nodes across the system, see **ALTER SYSTEM ALTER METRIC LEVEL**.

Syntax

SQL

```
ALTER NODE [ IF EXISTS ] node_name ALTER METRIC LEVEL
    { SET <set_options> | RESET <reset_options> }

<set_options> ::=
    [ LIKE | REGEX ] metric_to_alter [ ,... ] [=] { 'INFO' | 'DEBUG' }

<reset_options> ::=
    { [ LIKE | REGEX ] metric_to_reset [ ,... ] | ALL }
```

Parameter	Data Type	Description
`node_name`	string	The name of a node to alter.

Set Metric Levels `<set_options>`

The `SET` SQL statement assigns one or more specified metrics to the specified reporting level.

To set levels for a range of statistics, you can use wildcard characters with the `LIKE` keyword. Alternatively, you can use regular expressions with the `REGEX` keyword.

Parameter	Data Type	Description
`metric_to_alter`	string	The names of one or more metrics to alter for logging. Enclose any metric names in single quotes. The supported logging level values for metrics are `INFO` and `DEBUG`. Either keyword must be in single quotes.

Reset Metric Levels `<reset_options>`

The `RESET` SQL statement reverts one or more specified metrics to the default log level.

You can use wildcard characters with the `LIKE` keyword. Alternatively, you can use regular expressions with the `REGEX` keyword.

Use the `ALL` keyword to revert all metrics to the default log level.

Parameter	Data Type	Description
`metric_to_reset`	string	The names of one or more metrics to reset by reverting their log level to the default. To revert all metrics, use the `ALL` keyword.

Examples

In this example, the `ALTER NODE` SQL statement sets the `resultCache.queries` metric on the node `oc1` to the `DEBUG` level.

SQL

```
ALTER NODE oc1 ALTER METRIC LEVEL
    SET 'resultCache.queries' = 'DEBUG';
```

This example uses the `LIKE` wildcard keyword to capture any metrics prefixed by `localStorageService.device.smart.` and assign them to the `INFO` level.

SQL

```
ALTER NODE oc1 ALTER METRIC LEVEL
    SET LIKE 'localStorageService.device.smart.%' 'INFO';
```

Similar to the previous example, this SQL statement uses regular expressions to capture any metrics prefixed by `localStorageService.device.smart.` .

SQL

```
ALTER NODE oc1 ALTER METRIC LEVEL
    SET REGEX 'localStorageService.device.smart.*' 'INFO';
```

© 2026 Ocient, Inc.

This example alters the level of multiple metrics.

```
SQL

ALTER NODE oc1 ALTER METRIC LEVEL
    SET 'resultCache.queries' = 'DEBUG',
        'resultCache.data' = 'INFO',
        LIKE 'localStorageService.device.smart.%' 'INFO';
```

This example resets the same metrics.

```
SQL

ALTER NODE oc1 ALTER METRIC LEVEL
    RESET 'resultCache.queries',
          'resultCache.data',
          LIKE 'localStorageService.device.smart.%';
```

ALTER NODE SET ADDRESS

`ALTER NODE node_name SET ADDRESS ip_address` changes the internal IP address or hostname for any node. You must restart the system after you change the address. If you change the address of an Administrator Node, then you must update the `bootstrap.conf` file for every node with the new address.

For details about this file, see **Ocient System Bootstrapping**.

Syntax

```
SQL

ALTER NODE [ IF EXISTS ] node_name SET ADDRESS node_address
```

Parameter	Data Type	Description
`node_name`	string	The name of a node to alter.
`node_address`	string	The address for a node. This address can be an internal IP address (e.g., `111.11.11.111`) or a DNS hostname with the node address matching the node name (e.g., `sql1`).

Examples

This example changes the IP address to `111.11.11.111` for the node named `my_node` .

SQL

```
ALTER NODE my_node SET ADDRESS '111.11.11.111';
```

This example changes the SQL Node `sql1` to use the DNS address `sql1` .

SQL

```
ALTER NODE sql1 SET ADDRESS 'sql1';
```

CONNECTIVITY POOL

CREATE CONNECTIVITY_POOL

`CREATE CONNECTIVITY_POOL` creates a connectivity pool from participant SQL Nodes for a client connection. Connectivity pools enable you to change the IP address of the client connection. Client redirects happen only within connectivity pools. For details about connectivity pools and managing them, see **Manage the Network Configuration of an Ocient System**.

All SQL Nodes must be part of at least one connectivity pool. SQL Nodes only listen using the values set for the `listen_ip` and `listen_port` parameters.

The database uses the `source_ip` , `source_port` , and `priority` parameters when multiple connectivity pools exist that a client can use for the connection. In this case, the database uses the connectivity pool with the highest priority.

The database commits all changes made from these SQL statements after you restart all the SQL Nodes.

For information on all connectivity pools on the system, query the system catalog tables for **Connectivity Pools**. To see the connectivity pool assignment for each node, query the system catalog table for **connectivity pool participants**.

Syntax

SQL

```
CREATE CONNECTIVITY_POOL [IF NOT EXISTS] pool_name <create_definition>

<create_definition> ::=
    SOURCE_ADDRESS [=] source_ip
    | [ SOURCE_PORT [=] source_port ]
    | PRIORITY [=] priority
    | [ SSO INTEGRATION [=] sso_name ]
    | PARTICIPANTS (
        (NODE [=] node_name
         LISTEN_ADDRESS [=] listen_ip
         LISTEN_PORT [=] listen_port
         ADVERTISED_ADDRESS [=] advertised_ip
         [ ADVERTISED_PORT [=] advertised_port ]
         [ OPENAPI_PORT [=] openapi_port ), ...)
```

Parameter	Data Type	Description
`pool_name`	string	The name of the connectivity pool.
`source_ip`	string	The source IP address, which is the IP address in CIDR notation of the client that connects to the SQL Node.
`source_port`	integer	Optional. The source port number is the port number of the client. This parameter defaults to none, which means any port number.
`priority`	integer	The priority of the connection. A higher number indicates a higher priority.
`sso_name`	string	The name of an SSO integration to use for the connectivity pool. For details, see **SSO INTEGRATION**.

Parameter	Data Type	Description
`node_name`	string	The name of the SQL Node.
`listen_ip`	string	The IP address for listening. **i** Connectivity pools do not support dynamic IP addresses. If the IP address of your SQL Node changes, you must manually update it by using an **ALTER CONNECTIVITY_POOL ALTER PARTICIPANT** SQL statement.
`listen_port`	integer	The port number for listening.
`advertised_ip`	string	The IP address to return to the client.
`advertised_port`	integer	Optional. The port number to return to the client. If you specify NULL for this parameter, then the database uses the value of the `listen_port` parameter.
`openapi_port`	integer	The specified port for HTTP Query API-compliant specifications and endpoints.

Examples

Create the connectivity pool named `test_pool` with source address `1.2.3.4/32` and priority `1` for one SQL Node named `sql0`. The node has the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address to return to the client.

SQL

```
CREATE CONNECTIVITY_POOL IF NOT EXISTS test_pool
    SOURCE_ADDRESS '1.2.3.4/32'
    PRIORITY 1
    PARTICIPANTS (
        (NODE sql0
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'));
```

Create the connectivity pool named `test_pool` with source address `1.2.3.4/32`, port number `44`, and priority `1` for one SQL Node named `sql0`. The node has the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

SQL

```
CREATE CONNECTIVITY_POOL IF NOT EXISTS test_pool
    SOURCE_ADDRESS '1.2.3.4/32'
    SOURCE_PORT 44
    PRIORITY 1
    PARTICIPANTS (
        (NODE sql0
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050));
```

Create the connectivity pool named `test_pool` with source address `1.2.3.4/32`, port number `44`, priority `2`, and three participant SQL Nodes: `sql0`, `sql1`, and `sql2`. The nodes have the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

SQL

```
CREATE CONNECTIVITY_POOL IF NOT EXISTS test_pool
    SOURCE_ADDRESS '1.2.3.4/32'
    SOURCE_PORT 44
    PRIORITY 2
    PARTICIPANTS (
        (NODE sql0
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050),
        (NODE sql1
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050),
        (NODE sql2
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050));
```

© 2026 Ocient, Inc.

DROP CONNECTIVITY_POOL

`DROP CONNECTIVITY_POOL` removes the specified connectivity pool.

> All nodes must belong to a connectivity pool. The execution of this SQL statement does not permit the removal of a pool such that a node becomes an orphan.

Syntax

```
SQL

DROP CONNECTIVITY_POOL [ IF EXISTS ] pool_name
```

Parameter	Data Type	Description
`pool_name`	string	The name of the connectivity pool to remove.

Example

Remove the connectivity pool named `test_pool`.

```
SQL

DROP CONNECTIVITY_POOL test_pool;
```

ALTER CONNECTIVITY_POOL

ALTER CONNECTIVITY_POOL SET

`ALTER CONNECTIVITY_POOL SET` sets the metadata of a connectivity pool that includes the source IP address, source port number, priority, and node participants.

Syntax

SQL

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name SET <connectivity_pool_definition>

<connectivity_pool_definition> ::=
    SOURCE_ADDRESS [=] source_ip |
    SOURCE_PORT [=] source_port |
    PRIORITY [=] priority |
    PARTICIPANTS(
        (NODE [=] node_name
         LISTEN_ADDRESS [=] listen_ip
         LISTEN_PORT [=] listen_port
         ADVERTISED_ADDRESS [=] advertised_ip
         [ADVERTISED_PORT [=] advertised_port] ), ...)
```

For parameter definitions, see **CREATE CONNECTIVITY POOL**.

To directly edit the `LISTEN_ADDRESS`, `LISTEN_PORT`, `ADVERTISED_ADDRESS`, or `ADVERTISED_PORT` parameters, you must either remove the participants and add them back with new values for these parameters using the `ALTER CONNECTIVITY_POOL DROP PARTICIPANTS` and `ALTER CONNECTIVITY_POOL ADD PARTICIPANTS` statements, respectively, or use the `ALTER CONNECTIVITY_POOL SET PARTICIPANTS` statement.

Examples

This example sets the priority to `2` of the connectivity pool named `test_pool`.

SQL

```
ALTER CONNECTIVITY_POOL test_pool SET PRIORITY = 2;
```

This example sets the participants of the connectivity pool named `test_pool` to the list of participants with SQL Nodes `sql1` and `sql2`. When you execute this statement, the database resets the prior list of node participants to the new list. The nodes have the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

SQL

```
ALTER CONNECTIVITY_POOL test_pool
    SET PARTICIPANTS(
        (NODE sql1
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050),
         (NODE sql2
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050));
```

View the node identifiers of the participants of the `test_pool` connectivity pool by querying the node identifier `node_id` in the `sys.connectivity_pool_participants` and `sys.connectivity_pools` system catalog tables.

SQL

```
SELECT cpp.node_id
FROM sys.connectivity_pool_participants cpp
INNER JOIN sys.connectivity_pools cp
ON cpp.id = cp.id
WHERE cp.name = 'test_pool';
```

ALTER CONNECTIVITY_POOL RENAME TO

`ALTER CONNECTIVITY_POOL RENAME TO` SQL statement renames the existing connectivity pool.

Syntax

SQL

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name RENAME TO new_pool_name
```

Parameter	Data Type	Description
`pool_name`	string	The name of the existing connectivity pool.
`new_pool_name`	string	The new name of the connectivity pool.

Examples

Rename the `test_pool` connectivity pool to `new_test_pool`.

SQL

```
ALTER CONNECTIVITY_POOL test_pool RENAME TO new_test_pool;
```

ALTER CONNECTIVITY_POOL ADD PARTICIPANTS

`ALTER CONNECTIVITY_POOL ADD PARTICIPANTS` adds one or more nodes to the participant list of an existing connectivity pool.

For information on all connectivity pools on the system, query the system catalog tables for **Connectivity Pools**. To see the connectivity pool assignment for each node, query the system catalog table for **connectivity pool participants**.

Syntax

SQL

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
    ADD PARTICIPANTS(
        (NODE [=] node_name
         LISTEN_ADDRESS [=] listen_ip
         LISTEN_PORT [=] listen_port
         ADVERTISED_ADDRESS [=] advertised_ip
         [ADVERTISED_PORT [=] advertised_port] ), ...)
```

Examples

Add one node `sql5` to the participant list of the connectivity pool `test_pool`. The node has the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

SQL

```
ALTER CONNECTIVITY_POOL test_pool
    ADD PARTICIPANTS(
        NODE sql5
        LISTEN_ADDRESS '111.1.1.1'
        LISTEN_PORT 4050
        ADVERTISED_ADDRESS 'localhost'
        ADVERTISED_PORT 4050);
```

Add two nodes `sql6` and `sql7` to the participants list of the connectivity pool `test_pool`. The nodes have the IP address `111.1.1.1` and port number `4050` for listening. Specify the local IP address and port number `4050` to return to the client.

SQL

```
ALTER CONNECTIVITY_POOL test_pool
    ADD PARTICIPANTS(
        (NODE sql6
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050),
         (NODE sql7
         LISTEN_ADDRESS '111.1.1.1'
         LISTEN_PORT 4050
         ADVERTISED_ADDRESS 'localhost'
         ADVERTISED_PORT 4050));
```

ALTER CONNECTIVITY_POOL ALTER PARTICIPANT

`ALTER CONNECTIVITY_POOL ALTER PARTICIPANT` modifies the configurations of one SQL Node assigned to the existing connectivity pool.

This SQL statement can alter only one configuration parameter of a participant node at a time.

For information on all connectivity pools on the system, query the system catalog tables for **Connectivity Pools**. To see the connectivity pool assignment for each node, query the system catalog table for **connectivity pool participants**.

Syntax

```
SQL
```

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
    ALTER PARTICIPANT node_name
    SET <connectivity_pool_definition>

<connectivity_pool_definition> ::=
    { LISTEN_ADDRESS [=] listen_ip
    | LISTEN_PORT [=] listen_port
    | ADVERTISED_ADDRESS [=] advertised_ip
    | ADVERTISED_PORT [=] advertised_port
    | OPENAPI_PORT [=] openapi_port }
```

Parameter	Data Type	Description
`pool_name`	string	The name of the connectivity pool.
`node_name`	string	The name of the SQL Node to alter.
`listen_ip`	string	The IP address for listening.
`listen_port`	numeric	The port number for listening.
`advertised_ip`	string	The IP address to return to the client.
`advertised_port`	numeric	The port number to return to the client. If you specify NULL for this parameter, then the database uses the value of the `listen_port` parameter.
`openapi_port`	numeric	The specified port for HTTP Query API-compliant specifications and endpoints.

Example

Modify the SQL Node `sql0` to use the new `listen_port` value `4051`.

```
SQL
```

```
ALTER CONNECTIVITY_POOL cp_test
    ALTER PARTICIPANT sql0
    SET listen_port 4051;
```

ALTER CONNECTIVITY_POOL DROP PARTICIPANTS

`ALTER CONNECTIVITY_POOL DROP PARTICIPANTS` removes one or more nodes from the participant list of an existing connectivity pool. If you remove the last node of a connectivity pool, the Ocient System automatically removes the connectivity pool.

For information on all connectivity pools on the system, query the system catalog tables for **Connectivity Pools**. To see the connectivity pool assignment for each node, query the system catalog table for **connectivity pool participants**.

> All nodes must belong to a connectivity pool. The execution of this SQL statement does not permit the removal of a node such that it becomes an orphan.

Syntax

SQL

```
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
    DROP PARTICIPANTS (node_name, ...)
```

Examples

Remove one node `sql5` from the connectivity pool `test_pool`.

SQL

```
ALTER CONNECTIVITY_POOL IF EXISTS test_pool DROP PARTICIPANTS sql5;
```

Remove nodes `sql6` and `sql7` from the connectivity pool `test_pool`.

SQL

```
ALTER CONNECTIVITY_POOL test_pool DROP PARTICIPANTS (sql6, sql7);
```

ALTER CONNECTIVITY_POOL SET SSO INTEGRATION

`ALTER CONNECTIVITY_POOL SET SSO INTEGRATION` assigns a connectivity pool to use a specific SSO integration.

Connectivity pools look for the specified SSO integration name for each database you are trying to connect to. If the database does not support that SSO integration name, it uses the default SSO.

Syntax

```SQL
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
    SET SSO INTEGRATION sso_name
```

Parameter	Data Type	Description
`pool_name`	string	The name of the existing connectivity pool.
`sso_name`	string	The new name of the SSO integration.

Example

```SQL
ALTER CONNECTIVITY_POOL cp_test SET SSO INTEGRATION sso_test;
```

ALTER CONNECTIVITY_POOL REMOVE SSO INTEGRATION

`ALTER CONNECTIVITY_POOL REMOVE SSO INTEGRATION` removes an assigned SSO integration.

Syntax

```SQL
ALTER CONNECTIVITY_POOL [ IF EXISTS ] pool_name
    REMOVE SSO INTEGRATION sso_name
```

Parameter	Data Type	Description
`pool_name`	string	The name of the existing connectivity pool.
`sso_name`	string	The name of the existing SSO integration.

© 2026 Ocient, Inc.

Example

This example removes the SSO integration `sso_test` from the connectivity pool `cp_test`.

```
SQL

ALTER CONNECTIVITY_POOL cp_test REMOVE SSO INTEGRATION sso_test;
```

SSO INTEGRATION

CREATE SSO INTEGRATION

Creates a new SSO integration protocol, which can connect to databases as a database integration (see **ALTER DATABASE SET SSO INTEGRATION**) or connect using a connectivity pool (see **ALTER CONNECTIVITY_POOL SET SSO INTEGRATION**).

Syntax

```
SQL

CREATE SSO INTEGRATION [IF NOT EXISTS ] sso_name
    PROTOCOL sso_protocol <sso_property> [, ... ]

<sso_property> ::=
    -- literal or string value
    property_name = value |
    -- list of values
    property_name = [ value [, ...] ] |
    -- map of values
    property_name = { key = value [, ...] }
```

Parameter	Type	Description
`sso_name`	string	The identifier of the SSO integration to create.
`sso_protocol`	string	Supported values are: `oidc` , `openid` , and `openid_connect` . All these values are aliases of each other and use the OAuth 2.0 Authorization.

Parameter	Type	Description
`property_name`	string	One or more properties to include in the SSO integration. Each SSO property must be specified either as a literal, list, or map. See the SSO Properties table for the requirements for each supported SSO parameter. SSO properties also have these rules: You can place any string key or value between double quotations (e.g., `"default_group"`). String values that contain characters other than `[a-zA-z] | [0-9] | '_'` require double quotations (e.g., `"this.is.a.complex-$tring"`). Using a NULL value clears the existing configuration of a list or map property (e.g., `user_claim_ids = NULL`).

SSO Properties

Name	Required	Value Type	Default Value	Description
`issuer`	Yes	string	None	The **complete U** for the OAuth 2.(and OpenID Connect Authorization Server. This property value is the expected `\"iss\"` claim i access tokens validated by the database.
`client_id`	Yes	string	None	The **client identifier** as registered with t OpenID Provider.
`client_secret`	Yes	string	None	The **client secr** as registered wit the OpenID Provider. This property is requi for some SSO workflows.
`default_group`	Yes	string	None	The group users assigned if no group is specifie the `group_claim_map`

Name	Required	Value Type	Default Value	Description
				`gs` property for the OpenID Provider.
`redirect_host`	No	string	Depends on the connector (pyocient or JDBC).	Specifies the hos name during SS redirection.
`redirect_ssl`	No	Boolean	Depends on the connector (pyocient or JDBC).	Set to `true` to enable SSL callb during SSO redirection (i.e., redirect uses `https`). Set to `false` to disable (i.e., redirect uses `http`).
`disabled`	No	Boolean	`false`	Set to `true` to disable the OIDC integration for maintenance temporarily. ⚠ If you set this property to `true` all authentication requests using th connection fail.
`enable_id_token_authentication`	No	Boolean	`false`	Set this property `true` if the identifier token a contains the authorization tok In most circumstances, t option is necessa only for machine to-machine connections with user interaction, such as a server using a script to connect to an Ocient System.
`user_id_claims`	No	list of strings (e.g., value [, ...]	[“email”]	Set the identifier token claims use to identify users audit trails. If yo do not set a valu

Name	Required	Value Type	Default Value	Description
				the system uses `"email"` claim i is present, otherwise it uses `["iss", "sub"]`
`additional_scopes`	No	list of strings (e.g., value [, ...]	[]	Specifies additio scopes to reques when executing Authorization Co Flow.
`additional_audienc es`	No	list of strings (e.g., value [, ...]	[]	Specifies additio audiences to acc when validating tokens. This property is usefu for authorization servers without t token exchange capability. Each additional audience is a cas sensitive URL fro a provider, simila to the `issuer` value.
`groups_claim_ids`	No	list of strings (e.g., value [, ...]	[]	Specifies the tok claims that can b used to map the user to a Databa group. If you specify the `groups_claim_ma ngs` property, y must also specify the `groups_claim_id` property.
`groups_claim_mappi ngs`	No	map of strings (e.g., key = value [, ...])		Specifies mappir from the Provide group to the Database group.
`roles_claim_ids`	No	list of strings (e.g., value [, ...]	[]	Specifies the tok claims that can b used to map the user to a Databa role. If you speci the

Name	Required	Value Type	Default Value	Description
				`roles_claim_map` `gs` property, yo must also specify the `roles_claim_ids` property.
`roles_claim_mappin` `gs`	No	map of strings (e.g., key = value [, ...])		Specifies mappir from the Provide role to the Database role.
`allowed_groups`	No	list of strings (e.g., value [, ...]	[]	Specifies a list of external identity provider groups that are permitte to authenticate through this SSO integration. If yo specify the `allowed_groups` property, only us who are membe of the specified identify provider groups can acce the Ocient Syste If this property is empty or you do not specify it, the group-based filtering is disabl and the system allows all authenticated us from the provide unless they are explicitly blocked Group names mu match exactly as they appear in th claims of the identify provider group.
`allowed_roles`	No	list of strings (e.g., value [, ...]	[]	Defines a list of external identity provider roles th are authorized to access the Ocier System through

Name	Required	Value Type	Default Value	Description
				SSO integration. Only users assig to the specified roles in the exter provider can authenticate. If this property is empty or you do not specify it, rol based access control is disable for this integratic Role names mus correspond exac to the role claims provided by the identity provider the authenticatic response.
`blocked_groups`	No	list of strings (e.g., value [, ...]	[]	Specifies a list of external identity provider groups that are explicitly denied access through this SSO integration. The system blocks us who are member of any specified group in the external identity provider from authenticating, regardless of oth permissions they have. Setting thi property takes precedence over the `allowed_groups` property if a use belongs to both a allowed group ar a blocked group. Group names mu match exactly as they appear in th claims of the

Name	Required	Value Type	Default Value	Description
				identify provider group.
`blocked_roles`	No	list of strings (e.g., value [, ...]	[]	Defines a list of external identity provider roles th are explicitly prohibited from accessing the Ocient System through this SSO integration. The system denies access to users assigned to any the specified role overriding any other access permissions. Setting this property takes precedence over the `allowed_roles` property in cases where a user is i both an allowed role and a blocke role. Role names must correspond exactly to the ro claims of the identity provider group.
`allow_offline_acce ss`	No	Boolean	`false`	When the `allow_offline_a ss` property is `true`, Ocient requests offline access from the OpenID Connect identity provider including `access_type=off e` and `prompt=consent` the authorizatior URL.

Example

This example creates an SSO integration protocol with the specified OpenID issuer `"https://accounts.google.com"`, client identifier, and default group. Each of these parameters is required to create a new SSO integration. The example assumes preset values for the client identifier and default group.

```
SQL

CREATE SSO INTEGRATION sso_test PROTOCOL oidc
    issuer = "https://accounts.google.com",
    client_id = example_database_app_id,
    default_group = example_database_group;
```

DROP SSO INTEGRATION

Drops an SSO integration protocol.

Syntax

```
SQL

DROP SSO INTEGRATION [ IF EXISTS ] sso_name
```

Parameter	Type	Description
`sso_name`	string	The identifier of the SSO integration to drop.

Example

This SQL statement drops the SSO integration protocol `sso_test`.

```
SQL

DROP SSO INTEGRATION sso_test;
```

ALTER SSO INTEGRATION

Alters an SSO integration protocol by renaming it or modifying its properties.

```SQL
ALTER SSO INTEGRATION sso_name [ IF EXISTS ] {
    RENAME TO new_name |
    SET <sso_property> [, ... ]
    }

<sso_property> ::=
    -- literal or string value
    property_name = value |
    -- list of values
    property_name = [ value [, ...] ] |
    -- map of values
    property_name = { key = value [, ...] }
```

Parameter	Type	Description
`sso_name`	string	The identifier of the SSO integration to alter.
`new_name`	string	The new name of the SSO integration.
`property_name`	string	One or more properties to include in the SSO integration. Each SSO property must be specified either as a literal, list, or map. See the **SSO Properties** table to see the requirements for each supported SSO parameter. SSO properties also have these rules: You can place any string key or value between double quotations (e.g., `"default_group"`). String values that contain characters other than `[a-zA-z] | [0-9] | '_'` require double quotations (e.g., `"this.is.a.complex-$tring"`). Using a NULL value clears the existing configuration of a list or map property (e.g., `user_claim_ids = NULL`).

Examples

Alter SSO Integration Properties

This example alters the SSO integration to use different properties, changing the OpenID client identifier and the default group to `"group2"`. The example assumes a preset value for the client identifier.

SQL

```
ALTER SSO INTEGRATION sso_test SET
    client_id = different_database_app_id,
    default_group = "group2";
```

Rename an SSO Integration

This example renames the SSO integration to `sso_test_2`.

SQL

```
ALTER SSO INTEGRATION sso_test RENAME TO sso_test_2;
```

SYSTEM

ALTER SYSTEM ALTER CONFIG SET

`ALTER SYSTEM ALTER CONFIG SET` sets an override for the configuration at the system scope. For an explanation of how configuration overrides work, see **Inspect the Current Configuration**.

If the specified path to the configuration setting does not exist, it is considered a new configuration and appended to the collection of existing configurations. To reset a configuration override for a specific parameter, use the `RESET` keyword.

This configuration override is set in the base node configuration by default. To apply the override to a specific service role, prefix the key with the service role name.

Syntax

SQL

```
-- To set a new configuration override:
ALTER SYSTEM
    ALTER CONFIG SET parameter_name [=] parameter_value [,...]

-- To reset a configuration override:
ALTER SYSTEM
    ALTER CONFIG RESET [ parameter_name [,...] ]
```

Parameter	Data Type	Description
`parameter_name`	string	The name of a configuration parameter to alter. This parameter should be a string literal (i.e., enclosed in single quotes). Contact Ocient Support for details about altering configuration parameters.
`parameter_value`	any	The new value to be set for the specified configuration parameter. The data type of this value depends on the configuration parameter. This can be a string literal that can be cast to the data type of the configuration parameter. For example, the string literal `'true'` is equivalent to the Boolean value `true`.

Examples

This example alters the base configuration of the system scope. The example sets the certificate name for each node in the entire system to `cert_name.crt`.

SQL

```
ALTER SYSTEM ALTER CONFIG SET 'certificateStore.cert' = 'cert_name.crt';
```

This example resets the certificate configuration for the entire system.

SQL

```
ALTER SYSTEM ALTER CONFIG RESET 'certificateStore.cert';
```

This example alters the configuration for the service role `sql`. The `purgeCacheFrequency` setting determines the frequency in seconds to purge the result-set cache of stale values.

SQL

```
ALTER SYSTEM ALTER CONFIG SET 'sql.purgeCacheFrequency' = 600;
```

ALTER SYSTEM ALTER LOG LEVEL

`ALTER SYSTEM ALTER LOG LEVEL SET` sets the log level for a particular logger for all nodes in the system. For an explanation of how configuration overrides work, see **Inspect the Current Configuration**.

The logger name specified can be any string specifying a logger name, or `ALL` for the root logging config. See the **Log Configuration** for more details.

Syntax

SQL

```
-- To set a new configuration override:
ALTER SYSTEM ALTER LOG LEVEL SET { <logger_parameters> [,...] }

<logger_parameters> ::=
    'ALL' | logger_name [=] { 'EDEBUG' | 'DEBUG' | 'VERBOSE' | 'INFO' | 'WARN' | 'E

-- To reset a configuration override:
ALTER SYSTEM ALTER LOG LEVEL RESET [logger_name [, ...]]
```

Parameter	Data Type	Description
`<logger_parameters>`	string	Use `ALL` for the root logging config. Otherwise, the `logger_name` parameter can be any string specifying a logger name assigned to a severity level. See the **Log Configuration** page for more details. To reset a logger to the default log level (`INFO`), use the `RESET` keyword.

Example

This example alters the log level at the system scope.

SQL

```
ALTER SYSTEM ALTER LOG LEVEL SET 'ALL' = 'DEBUG';
```

ALTER SYSTEM RENAME TO

`ALTER SYSTEM RENAME TO` sets the system name. The system name appears in the `sys.system_information` system catalog table. You can also see the current system name by using the **CURRENT_SYSTEM** function.

Syntax

SQL

```
ALTER SYSTEM RENAME TO system_name
```

Parameter	Data Type	Description
`system_name`	string	The name of the Ocient System.

Example

SQL

```
ALTER SYSTEM RENAME TO "productionSystem";
```

ALTER SYSTEM ALTER METRIC LEVEL

Alters the reporting level of various internal metrics across the system, including all nodes. For details about internal metrics, see **Statistics Monitoring**.

To set metrics reporting for individual nodes, see **ALTER NODE ALTER METRIC LEVEL**.

Syntax

SQL

```
ALTER SYSTEM ALTER METRIC LEVEL
    { SET <set_options> | RESET <reset_options> }

<set_options> ::=
    [ LIKE | REGEX ] metric_to_alter [ ,... ] [=] { 'INFO' | 'DEBUG' }

<reset_options> ::=
    { [ LIKE | REGEX ] metric_to_reset [ ,... ] | ALL }
```

Set Metric Levels `<set_options>`

The `SET` SQL statement assigns one or more specified metrics to the specified reporting level. You can assign metrics to `INFO` or `DEBUG` levels.

To set levels for a range of statistics, you can use wildcard characters with the `LIKE` keyword. Alternatively, you can use regular expressions with the `REGEX` keyword.

Parameter	Data Type	Description
`metric_to_alter`	string	The names of one or more metrics to alter for logging. Enclose any metric names in single quotes. The supported logging levels for metrics are `INFO` and `DEBUG`. Either keyword must be in single quotes.

Reset Metric Levels `<reset_options>`

The `RESET` SQL statement reverts one or more specified metrics to the default reporting level.

You can use wildcard characters with the `LIKE` keyword. Alternatively, you can use regular expressions with the `REGEX` keyword.

Use the `ALL` keyword to revert all metrics to the default level.

Parameter	Data Type	Description
`metric_to_reset`	string	The names of one or more metrics to reset by reverting their level to the default. To revert all metrics, use the `ALL` keyword.

Examples

In this example, the `ALTER SYSTEM` SQL statement sets the `resultCache.queries` metric to the `DEBUG` level.

SQL

```
ALTER SYSTEM ALTER METRIC LEVEL
    SET 'resultCache.queries' = 'DEBUG';
```

© 2026 Ocient, Inc.

This example uses the `LIKE` wildcard to capture any metrics prefixed by `localStorageService.device.smart.` and assign them to the `INFO` level.

SQL

```
ALTER SYSTEM ALTER METRIC LEVEL
    SET LIKE 'localStorageService.device.smart.%' 'INFO';
```

Similar to the previous example, this SQL statement uses regular expressions to capture any metrics prefixed by `localStorageService.device.smart.`.

SQL

```
ALTER SYSTEM ALTER METRIC LEVEL
   SET REGEX 'localStorageService.device.smart.*' 'INFO';
```

This example alters the level of multiple metrics.

SQL

```
ALTER SYSTEM ALTER METRIC LEVEL
    SET 'resultCache.queries' = 'DEBUG',
        'resultCache.data' = 'INFO',
        LIKE 'localStorageService.device.smart.%' 'INFO';
```

This example resets the same metrics.

SQL

```
ALTER SYSTEM ALTER METRIC LEVEL
    RESET 'resultCache.queries',
          'resultCache.data',
          LIKE 'localStorageService.device.smart.%';
```

ALTER SYSTEM ALTER SECURITY

Set security settings using the `ALTER SYSTEM ALTER SECURITY` SQL statement. Replace `<security_setting>` with the security setting and `<value>` with the value.

You can inspect the current values of system configuration settings using the `sys.config` and `sys.node_config` system catalog tables.

Syntax

```
SQL

ALTER SYSTEM ALTER SECURITY <security_setting> [=] <value>
```

Parameter	Data Type	Description
`security_setting`	string	The security setting with values: * `password_minimum_length` * `password_complexity_level` * `password_no_repeat_count` * `password_lifetime_days` * `password_invalid_attempt_limit` For details about these values, see **Database Password Security Settings**.
`value`	numeric	An integer to represent one of the security settings. For details about this value, see **Database Password Security Settings**.

Examples

Set a Security Setting for All Databases

Set the password lifetime to 30 days for the entire system.

```
SQL

ALTER SYSTEM ALTER SECURITY password_lifetime_days = 30;
```

Set Multiple Security Settings

Set multiple security settings in a single SQL statement. In this case, set the password lifetime to 30 days and the minimum password character length to 12. As with other similar SQL statements, the `=` character is optional.

SQL

```
ALTER SYSTEM ALTER SECURITY
    password_lifetime_days 30,
    password_minimum_length 12;
```

ALTER SYSTEM SET DEFAULT STORAGESPACE

Sets the specified storage space as the default for creating new tables.

`CREATE TABLE` SQL statements automatically use the default storage space unless you specify a different storage space (see **Table Options**). After you create a table, you cannot alter its assigned storage space. Statements such as `DROP STORAGESPACE` and `ALTER CLUSTER REMOVE STORAGESPACE` throw an error if they attempt to drop the default storage space of the system.

To remove the default storage space, use the optional `RESET` keyword. The `RESET` keyword alters the system to have no default storage space. Until you set a storage space as the new default, `CREATE TABLE` statements throw an error if they do not specify a storage space for the new table.

Required Privileges

To use this SQL statement, you must have the `ALTER` privilege for the system. For details about privileges, see **Data Control Language (DCL) Statement Reference**.

Syntax

SQL

```
ALTER SYSTEM { SET | RESET } DEFAULT STORAGESPACE [ storage_space_name ]
```

Parameter	Data Type	Description
`storage_space_name`	string	The name of a storage space to set as the system default. This parameter is required when you use the `SET` keyword to define a default storage space. Do not include this parameter when you use the `RESET` keyword to remove the default storage space.

Examples

Set a Default Storage Space

This example sets `ss_0` as the system default storage space.

SQL

```
ALTER SYSTEM SET DEFAULT STORAGESPACE ss_0;
```

Remove the Default Storage Space

This example removes any default storage space.

SQL

```
ALTER SYSTEM RESET DEFAULT STORAGESPACE;
```

Related Links

Other Functions and Expressions

CREATE TABLE SQL Statement Examples

Database Password Security Settings

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Databases

This group of DDL SQL statements allows database administrators to manage databases. Database administrators can create and modify databases, including SSO authentication settings. You can view information about databases using the `sys.databases` system catalog table.

For information on database components, see the pages on **Schemas**, **Tables**, **Views**, and **Indexes**.

CREATE DATABASE

`CREATE DATABASE` creates a new database. The database name must be distinct from the name of any existing database in the system.

To create a database, you must have the `CREATE DATABASE` privilege for the current system.

Syntax

SQL

```
CREATE DATABASE [ IF NOT EXISTS ] database_name
```

Parameter	Type	Description
`database_name`	string	A unique identifier for the database. The system generates an error if you specify a duplicate name. `system` is a reserved database name and can neither be created nor dropped.

Example

Create a new database named `ocient`.

© 2026 Ocient, Inc.

SQL

```
CREATE DATABASE ocient;
```

DROP DATABASE

`DROP DATABASE` removes an existing database. This SQL statement also disconnects all users currently connected to the database.

To remove a database, you must have the `DROP DATABASE` privilege for the current database.

You cannot drop a database while it has any `PIPELINE` in a running status.

> ⚠ The `DROP DATABASE` SQL statement removes the existing database and all created users, tables, and views.
>
> This action cannot be undone.

Syntax

SQL

```
DROP DATABASE [ IF EXISTS ] database_name [, ...]
```

Parameter	Type	Description
`database_name`	string	An identifier for the database to be dropped. You can drop multiple databases by specifying additional database names and separating each with commas. `system` is a reserved database name and can neither be created nor dropped.

Example

Remove an existing database named `ocient`.

SQL

```
DROP DATABASE ocient;
```

ALTER DATABASE

ALTER DATABASE RENAME

`ALTER DATABASE RENAME` renames an existing database.

To rename a database, you must have the `ALTER DATABASE` privilege for the database.

Syntax

SQL

```
ALTER DATABASE old_database_name RENAME TO new_database_name
```

Parameter	Type	Description
`old_database_name`	string	The old identifier of the database for the rename.
`new_database_name`	string	The new identifier of the database for the rename.

Example

Rename an existing database named `oracle` to `ocient`.

SQL

```
ALTER DATABASE oracle RENAME TO ocient;
```

ALTER DATABASE SET SSO INTEGRATION

`ALTER DATABASE SET SSO INTEGRATION` configures the database to authenticate using an external SSO provider. This SSO integration is the default for connections unless you use a connectivity pool or specify a different provider.

To set a connection, you must be a system-level user or a database administrator and have an open connection to the database.

This SQL statement is an alias for **ALTER DATABASE ALTER SSO INTEGRATION**.

See **Configuring the Ocient Database** for details about configuring SSO protocols.

> If your Ocient® System is version 25.0 or later, you can create multiple SSO integrations for each database. An SSO integration assigned to the database by the `ALTER DATABASE` SQL statement is the primary SSO connection, unless you connect to the database with a connectivity pool that has a different SSO integration assigned to it.

Syntax

SQL

```
ALTER DATABASE database
    ALTER SSO INTEGRATION sso_name
```

Parameter	Type	Description
`database`	string	The identifier of the database for configuration.
`sso_name`	string	The identifier of the SSO integration to use.

Example

This example sets an example database to use the SSO integration named `sso_test`.

SQL

```
ALTER DATABASE example_database
    SET SSO INTEGRATION sso_test;
```

ALTER DATABASE ALTER SSO INTEGRATION

`ALTER DATABASE ALTER SSO CONNECTION` configures the database to authenticate using an external SSO provider. This SSO integration is the default for connections unless you use a connectivity pool or specify a different provider.

To alter a connection, you must be a system-level user or a database administrator and have an open connection to the database.

This SQL statement is an alias for **ALTER DATABASE SET SSO INTEGRATION**.

See **Configuring the Ocient Database** for details about configuring SSO protocols.

> If your Ocient System is version 25.0 or later, you can create multiple SSO integrations for each database. An SSO integration assigned to the database by the `ALTER DATABASE` SQL statement is the primary SSO connection, unless you connect to the database with a connectivity pool that has a different SSO integration assigned to it.

Syntax

SQL

```
ALTER DATABASE database
    ALTER SSO INTEGRATION sso_name
```

Parameter	Type	Description
`database`	string	The identifier of the database for configuration.
`sso_name`	string	The identifier of the SSO integration to use.

Example

This example alters an example database to use the SSO integration named `sso_test`.

SQL

```
ALTER DATABASE example_database
    ALTER SSO INTEGRATION sso_test;
```

ALTER DATABASE REMOVE SSO INTEGRATION

`ALTER DATABASE REMOVE SSO INTEGRATION` removes an existing SSO integration as the default connection protocol for the database. This action effectively undoes the **ALTER DATABASE ALTER SSO INTEGRATION** SQL statement.

To remove a connection, you must be a system-level user or a database administrator.

Syntax

```
SQL

ALTER DATABASE database REMOVE SSO INTEGRATION
```

Parameter	Type	Description
`database`	string	The identifier of the database for deletion.

Example

Remove the default connection from the database named `example_database`.

```
SQL

ALTER DATABASE example_database REMOVE SSO INTEGRATION;
```

ALTER DATABASE ALTER SECURITY

Sets the security settings at the database level using the `ALTER DATABASE ALTER SECURITY` SQL statement. Replace `<security_setting>` with the security setting and `<value>` with the value.

Syntax

```
SQL

ALTER DATABASE database ALTER SECURITY <security_setting> [=] <value>
```

Parameter	Data Type	Description
`database`	string	The identifier of the database for setting security settings.
`security_setting`	string	The security setting with values: * `password_minimum_length` * `password_complexity_level` * `password_no_repeat_count`

Parameter	Data Type	Description
		* `password_lifetime_days` * `password_invalid_attempt_limit` For details about these values, see **Database Password Security Settings**.
`value`	numeric	An integer to represent one of the security settings. For details about this value, see **Database Password Security Settings**.

Example

Set the password lifetime to 20 days for the database `example_db`.

SQL

```
ALTER DATABASE example_db
    ALTER SECURITY password_lifetime_days = 20;
```

Related Links

Core Elements of an Ocient System

Data Query Language (DQL) Statement Reference

Database Password Security Settings

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Schemas

This group of DDL SQL statements allows database administrators to manage schema design. Database administrators can create, modify, and drop schemas. You can view information about the schema using the `sys.schemas` system catalog table.

For information on other database components, see the pages on **Databases**, **Tables**, **Views**, and **Indexes**.

Implicit and Explicit Schemas

The Ocient® System creates implicit schema objects when you execute the `CREATE` or `ALTER` SQL statements on a table, view, or machine learning model in the database, and there is no destination schema for those objects. For example, in the `ALTER TABLE schema1.table1 RENAME to schema2.table1` statement, if `schema2` does not exist in the database, the system creates `schema2` as an implicit schema object.

The system creates explicit schema objects when you execute a `CREATE SCHEMA` SQL statement or promote an implicit schema using the `ALTER SCHEMA` SQL statements or the `GRANT` SQL statement on the schema.

To execute SQL statements on objects or view the metadata of objects in an explicit schema, you must have the `VIEW SCHEMA` privilege and the basic required privileges for the statement. For an implicit schema, you do not need any additional privileges.

To create an object, if the schema is implicit, then you need the `CREATE object ON DATABASE` privilege, where `object` can be a table, view, or machine learning model. If the schema is explict, then you need the `CREATE object ON SCHEMA` privilege.

You can check the disposition of the schema using the `is_implicit` column in the `sys.schemas` system catalog table.

> After promotion, you cannot demote an explicit schema back to an implicit one.

CREATE SCHEMA

`CREATE SCHEMA` defines an explicit schema. To use the optional `OR REPLACE` SQL statement, the schema must not contain other objects.

Required Privileges

Users with the `CREATE SCHEMA` privilege on the database can create an explicit schema.

After you create an explicit schema, ensure to specify the privileges for each user. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

SQL

```
CREATE [ OR REPLACE ] SCHEMA [ IF NOT EXISTS ] schema_name
```

Parameter	Type	Description
`schema_name`	string	An identifier for the schema to create. The name must be distinct from any existing schema in the database unless you specify the `REPLACE` keyword. `sys`, `sysgdc`, `syslookup`, and `information_schema` are reserved schema names and can neither be created nor dropped.

Example

Define the `sandbox` schema.

SQL

```
CREATE SCHEMA sandbox;
```

ALTER SCHEMA RENAME

`ALTER SCHEMA RENAME` renames an existing schema. If you use this statement with an implicit schema, the system promotes the implicit schema to an explicit schema.

Required Privileges

To rename a schema, you must have the `VIEW` and `ALTER` privileges for the schema.

The Ocient System requires these privileges if this statement includes a change to the schema:

- `VIEW` and `ALTER` privilege on the current schema
- `CREATE SCHEMA` privilege on the database

Syntax

```
SQL

ALTER SCHEMA [ IF EXISTS ] old_schema_name RENAME TO new_schema_name
```

Parameter	Type	Description
`old_schema_name`	string	The old identifier of the schema for rename.
`new_schema_name`	string	The new identifier of the schema for rename. The name must be distinct from the name of any existing schema in the system.

Example

Rename an existing schema named `sandbox` to `ocient`.

```
SQL

ALTER SCHEMA sandbox RENAME TO ocient;
```

DROP SCHEMA

`DROP SCHEMA` removes an existing schema. The Ocient System automatically removes an implicit schema when you remove all its contents. However, an explicit schema persists even without any objects.

If the existing schema contains objects and you do not specify the `CASCADE` keyword, the system cannot remove this schema. Otherwise, the system removes all objects inside the existing schema.

Required Privileges

If the schema is explicit, you must have the `DROP SCHEMA` privilege. Otherwise, if the schema is implicit, the schema must not contain any objects, or if you specify the `CASCADE` keyword, you must have the `VIEW` and `DROP` privileges for all objects in the schema to remove it.

Syntax

SQL

```
DROP SCHEMA [ IF EXISTS ] schema_name [, ...] [ CASCADE ]
```

Parameter	Type	Description
`schema_name`	string	An identifier for the schema to remove. You can drop multiple schemas by specifying additional schema names and separating each with commas. `sys`, `sysgdc`, `syslookup`, and `information_schema` are reserved schema names and can neither be created nor dropped.

Example

Remove an existing schema named `ocient`.

SQL

```
DROP SCHEMA ocient;
```

Related Links

Core Elements of an Ocient System

Data Query Language (DQL) Statement Reference

Database Password Security Settings

System Catalog

© 2026 Ocient All Rights Reserved

Tables

This group of DDL SQL statements allows database administrators to manage tables. Database administrators can create, modify, drop, and export tables. You can also truncate segments on a table. Additional statements are available for unique settings like segment redundancy, compression settings, and streamloader properties. You can view information about tables using the `sys.tables` system catalog table.

For information on other database components, see the pages on **Databases**, **Schemas**, **Views**, and **Indexes**.

CREATE TABLE

Creates a new table in the current database.

The table name must be distinct from the name of any existing tables in the database unless the `REPLACE` keyword is specified. To use `REPLACE` in the `CREATE TABLE` statement, you must have `DELETE` privileges.

By default, columns are nullable unless otherwise specified.

For faster query results, you can define one TimeKey® for the table, which must be a timestamp, date, or time column, with a specified bucket resolution. Tables with a specified TimeKey can perform query operations faster, especially if they involve time filtering.

You can specify a Clustering Key composed of one or more fixed-length columns. Designating columns as cluster keys that are frequently referenced in queries can greatly improve performance.

For details about defining TimeKeys and clustering indexes, see **TimeKeys and Clustering Keys**.

See the **Data Types** section for table-supported data types.

Required Privileges: You must have the `CREATE TABLE` privilege for the current database.

© 2026 Ocient, Inc.

For examples, see **CREATE TABLE SQL Statement Examples**.

Syntax

SQL

```
CREATE [ OR REPLACE ] TABLE [ IF NOT EXISTS ] table_name
    [ ( <column_definition> [, ...] | <clustering_definition> ) ]
    [ [ WITH ] <create_option> [, ... ] ]
    [ AS (query) ]

<column_definition> ::=
   column_name [ data_type ] [ <timekey_definition> ] [ <column_constraint> ] [, ..

<timekey_definition> ::=
  TIME KEY BUCKET (bucket_granularity, bucket_value)

<column_constraint> ::=
   | [ NOT ] NULL
   | DEFAULT expression
   | COMMENT comment
   | COMPRESSION GDC [ (compression_value), EXISTING schema_name ]
   | COMPRESSION [ compression_scheme ]

<clustering_definition> ::=
    CLUSTERING KEY key_name (ck_col1, ck_col2 [, ...])
        [, INDEX index_name (idx_col1, idx_col2 [, ...])
        [, INDEX ... ] ]

 <create_option> ::=
   INDEX index_name (index_column) [ USING index_type ]
   | RETENTION POLICY AGE retention_granularity retention_value
   | STORAGESPACE storage_space_name
   | SEGMENTSIZE segment_value
   | REDUNDANCY segment_part (redundancy_scheme)
   | STREAMLOADER_PROPERTIES streamloader_json
   | COMMENT '<string>'
```

Parameter	Type	Description
`table_name`	string	A unique identifier for the table. `table_name` must be distinct from the name of any

Parameter	Type	Description
		existing tables in the database unless the `REPLACE` keyword is specified.
`query`	string	A `SELECT` query used to load data into the newly created table. For details, see **CREATE TABLE AS SELECT** .

Column Definition (`<column_definition>`)

The parameters listed here are required for defining each column in a table.

Parameter	Type	Description
`column_name`	string	An identifier for a column to be included in the newly created table.
`data_type`	string	The data type of a specified column. For a list of supported data, see **Data Types**.

Clustering Key and Index Definition (`<clustering_definition>`)

The parameters listed here are required for defining a Clustering Key or clustering indexes on a table.

For details about how to apply clustering columns, see **Clustering Key**.

SQL

```
CLUSTERING KEY key_name (ck_col1, ck_col2 [, ...])
    [, INDEX index_name (idx_col1, idx_col2 [, ...])
    [, INDEX ... ] ]
```

Parameter	Type	Description
`key_name`	string	An identifier for the Clustering Key.
`ck_col1, ck_col2 [, ...]`	string	A series of specific columns comprising the Clustering Key. Clustering Key columns must not be nullable. Specify `NOT NULL` in the column definition for the respective columns. For details, see **Column Definition**.

Parameter	Type	Description
		No limit exists on the number of columns for the Clustering Key.
`index_name`	string	Optional. An identifier for a clustering index. You must include the definition of columns that comprise the index using the `idx_col1, idx_col2 [, ...]` parameter.
`idx_col1, idx_col2 [, ...]`	string	Optional. A series of specific columns comprising a clustering index. You can apply clustering indexes only to columns included in the Clustering Key. Specify any number of columns in any specified order. You must include the identifier for the index using the `index_name` parameter.

TimeKey Definition (`<timekey_definition>`)

This syntax example is for a single TimeKey column, which can be included in the column definition of a `CREATE TABLE` statement. For the full syntax, see **CREATE TABLE**.

For details about using TimeKeys, see **TimeKeys**.

SQL

```
column_name [ data_type ] TIME KEY BUCKET (bucket_granularity, bucket_value [, NOIN
```

Parameter	Type	Description
`column_name`	string	An identifier for the column.
`data_type`	string	Optional. The data type of the column. For the TimeKey column, this column should be DATE or TIMESTAMP type. The TimeKey column can also support INT or BIGINT type, but these data types do not use a `bucket_value` argument.
`bucket_granularity`	numeric	The granularity of the TimeKey column, based on the specified TIME type in `bucket_value` .
`bucket_value`	string	The TIME type to parse the TimeKey column. Supported values include: `[ DAY, HOUR, MINUTE, SECOND ]`

Parameter	Type	Description
		For example, `BUCKET(1, DAY)` sets the time-bucket granularity to a fixed width of one day.

Column Constraint (`<column_constraint>`)

The parameters listed here include constraints and other configurations for individual columns. For best performance, one column with date or time data in each table should be defined as the `TIME KEY` with a specified `bucket_value`.

For details about TimeKey columns, see **TimeKeys**.

Parameter	Type	Description
`expression`	data type of the column	If you specify this parameter as a constraint, the parameter sets the default value for the column. The value can be a literal or a constant, deterministic expression. Default expressions cannot use non-deterministic functions such as `RAND()`. The system evaluates the expression at the time of table creation and stores the resulting value. If the expression result does not match the column data type, the system automatically attempts type coercion to convert it. **Literals Usage** — The value can be a literal enclosed in quotes or an unquoted numeric literal. For example, this `CREATE TABLE` SQL statement includes a default value for its `UUID` column. `CREATE TABLE example_table ( col1 UUID NOT NULL DEFAULT '00000000-0000-0000-0000-000000000000' );` The supported geospatial data types `POINT`, `LINESTRING`, and `POLYGON` require WKT formatting. For formatting examples, see the **Well-known text representation of geometry**. **Expressions Usage** — The value can also be an expression that evaluates to a constant, such as an arithmetic operation or a function execution. For example, this `CREATE TABLE` statement uses two default expressions. The column `col1` uses a math equation that evaluates to `104`. The column `col2` concatenates two strings using the `CONCAT` function to create the new string `'HEYYOU'`.

Parameter	Type	Description
		`CREATE TABLE example_table ( col1 INT DEFAULT 4 + (5 * 20), col2 VARCHAR DEFAULT CONCAT('HEY', 'YOU') );`
`comment`	string	An optional comment for the column.
`compression_value`	numeric	An integer value of either `1`, `2` or `4` is used to define the storage of `COMPRESSION GDC`. The compression of the values is defined as follows: - A `compression_value` of 1 can hold up to 255 unique values. - A `compression_value` of 2 can hold up to 65535 unique values. - A `compression_value` of 4 can hold millions of unique values. For tuple columns, compression is specified for each tuple value rather than with other constraints. The example here defines a tuple column with GDC compression only for the first inner `VARCHAR` value. `my_tuple TUPLE<<INT, VARCHAR(255) COMPRESSION GDC(1), VARCHAR(255)>>` It is not recommended to use compression on a column that will contain more than one million unique values.
`schema_name`	string	Fully qualified column name or a column name if you specify a column in the same table. If `EXISTING` is specified as a compression constraint, the compression GDC reuses the system lookup table for the `schema_name` column, rather than creating a new one. The existing column must be in the same database as the new column. `schema_name` should include the schema, table, and column names, each separated by periods and enclosed in double-quotation marks, for example: `"schema_name"."table_name"."column_name"` If an existing column is specified, any new column that uses the existing column's system lookup table must be deleted before the existing column can be deleted.
`compression_schem e`	string	The type of compression used for the column. Supported values include: `[ COMPRESSION NONE | COMPRESSION ZSTD | COMPRESSION DYNAMIC ]` If no compression setting is specified, the compression defaults to `COMPRESSION DYNAMIC` for fixed-length columns and `COMPRESSION NONE` for variable-length columns. `COMPRESSION ZSTD` applies to `VARCHAR` columns as well as other data types. For the `COMPRESSION DYNAMIC` setting, the Ocient® System applies LZ4 compression only if the column data is dynamically determined to be compressible. For fixed-length columns, `COMPRESSION DYNAMIC` applies delta-delta compression.

Parameter	Type	Description
		For details about Ocient-supported compression schemes, see **Table Compression Options**.

Create Option (`<create_option>`)

The parameters listed here include various options to configure table storage space, segments, redundancy, streamloading, and indexes.

Parameter	Type	Description
`index_name`	string	An identifier for the index. The name must be distinct from the name of any existing index on the table.
`index_column`	string	The name of the column for the index. Identical indexes on the same column are not allowed. A column can only have multiple indexes if they are of different types or parameters.
`index_type`	string	Optional. One of these supported secondary index types: `INVERTED | HASH | NGRAM | SPATIAL | ZONE_MAP` If unspecified, this value defaults to an index type based on its data type. For details, see **Index Type Requirements and Defaults**.
`retention_granularity`	numeric	A number that represents the amount of time before the system deletes data in the table. You must pair this number with a specific time type, represented by the `retention_value` parameter. For details about setting up retention policies, see **Table Retention Policies**.
`retention_value`	string	The unit of time that represents how long the table retains data. Supported values are: `WEEK`, `DAY`, `HOUR`, `MINUTE`, `SECOND`, `MILLISECOND` You must pair this number with the amount of time, represented by the `retention_granularity` parameter. For example, `1 DAY` schedules the table retention to a granularity of one day.
`storage_space_name`	string	An identifier for the `STORAGESPACE`.

Parameter	Type	Description
`segment_value`	numeric	A value to define the size of the segment.
`segment_part`	string	Specifies the segment part redundancy of the table. Supported values include: `\{ DATA | MANIFEST | INDEX | SUMMARY_STATS | STATS }` These settings are defined as follows: `DATA` : The actual data for the table. `MANIFEST` : Header information stored about the data, which describes how to locate any specified cluster of rows within the data. `INDEX` : The index of the data used for quicker lookups and better query performance. `SUMMARY_STATS` : A collection of statistics on the data that includes compression, row count, and average column size. `STATS` : Used in the optimizer, the probability density function and combinable distinct estimators used to make better optimizations to query plans.
`redundancy_scheme`	string	The redundancy scheme. Supported values include: `\{ COPY | PARITY }` These settings are defined as follows: `COPY` : A copy of the bytes is stored throughout the storage cluster to ensure redundancy. This option uses more storage but is faster during rebuilds and node outages. `PARITY` : Using the parity encoding specified on the storage cluster, this option uses parity bits to ensure redundancy for the data. This option uses less storage but is slower during rebuilds and node outages.
`streamloader_json`	string	A JSON string that defines the streamloader parameters. For details, see **ALTER TABLE STREAMLOADER PROPERTIES**.

Example

This example creates a new table in the current database and schema named `trades`.

The table uses the TIMESTAMP column `created_at` as the TimeKey, with the granularity set at 1 hour. The columns `ticker_symbol` and `t_type` are defined as the table clustering keys. The example also includes a streamloader property `pageQueryExclusionDuration` to delay how soon data pages that were recently added can be included in query results.

SQL

```
CREATE TABLE trades (
    id UUID NOT NULL,
    ticker_symbol VARCHAR(255) NOT NULL COMPRESSION GDC(2),
    t_type VARCHAR(255) NOT NULL COMPRESSION GDC(1),
    raw_ticker_data VARCHAR(255),
    created_at TIMESTAMP NOT NULL TIME KEY BUCKET(1, HOUR),
    array_of_tuples tuple<<varchar(2040), byte, bigint, double, timestamp, date, ti
    CLUSTERING INDEX idx_ticker_symbol_type (ticker_symbol, t_type),
    INDEX idx_type (t_type)
)
    STORAGESPACE ocient_storage,
    REDUNDANCY DATA (PARITY),
    REDUNDANCY MANIFEST (COPY),
    STREAMLOADER_PROPERTIES '{
    "pageQueryExclusionDuration" : "2700s"
}';
```

CREATE TABLE AS SELECT (CTAS)

CTAS provides the ability to create and load a new table from the result of a query on one or more existing tables. The first column of the query result maps to the first column of the new table definition, the second column maps to the second column of the new table, and so on. The new table is available for querying after it has been created, and the entire result set from the query has been loaded into the table. When you receive a response to the CTAS SQL statement, the load is complete, and the table is ready.

When you create a table from a `SELECT` SQL statement, the schema for the table can be automatically determined based on the query results. You can override this behavior with an alternative schema, provided the query results can automatically be cast to the target column types. CTAS also supports all syntax options for the new table. CTAS does not support default values and explicit nullable definitions on the column of the table.

CTAS statements support secondary and prefix indexes.

To create a table, you must have both the `CREATE TABLE` privilege for the current database and the `SELECT` privilege on all referenced tables and views.

For syntax and parameter information, see **CREATE TABLE**.

Default Table Definitions

By default, a new table created with a CTAS statement retains column names, data types, and nullable definitions from the queried table. You can override this configuration with alternate table definitions in the CTAS statement.

> ⚠ Tables created by a CTAS statement do not inherit some table definitions from the original table, including the following:
>
> - TimeKey
> - Clustering Key and Clustering indexes
> - Secondary indexes
> - Column compression
> - Optional table configurations (see **create option**)
>
> To include these table definitions, you must explicitly specify them in the CTAS statement.

Examples

These CTAS examples select columns from the `original_table` table that this `CREATE TABLE` statement defines. This table contains these columns:

- `col_int` — Non-nullable integer with the default value `123456789`
- `col_bigint` — Non-nullable 8-byte signed integer
- `col_id` — Non-nullable integer
- `col_point` — Non-nullable point with the default value `POINT(0 0)`
- `col_timestamp` — Non-nullable TimeKey with granularity set at 1 day
- `col_varchar` — Variable-length character string with a maximum length of 255 characters and Zstandard compression

The table has a Clustering Key `ck` using the `col_bigint` and `col_id` columns. It also has two secondary indexes: a `hash` index `idx_01` on the `col_varchar` column and a `spatial` index on the `idx_02` column.

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE original_table (
    col_int INT NOT NULL DEFAULT 123456789,
    col_bigint BIGINT NOT NULL,
    col_id INT NOT NULL,
    col_point POINT NOT NULL DEFAULT 'POINT(0 0)',
    col_timestamp TIMESTAMP TIME KEY BUCKET (1,DAY) NOT NULL,
    col_varchar VARCHAR(255) COMPRESSION ZSTD,
    CLUSTERING KEY ck (col_bigint, col_id)
)
WITH
    INDEX idx_01 (col_varchar) USING HASH,
    INDEX idx_02 (col_point) USING SPATIAL;
```

CTAS Using All Columns from a Base Table

This example shows a basic CTAS statement that inherits most of its table definition from the `original_table` table.

SQL

```
CREATE TABLE basic_ctas AS (
    SELECT * FROM original_table );
```

The new `basic_ctas` table includes all the columns and data types from the original table definition. However, it does not include the segment keys, indexes, or the compression on the `col_varchar` column. The `EXPORT TABLE` SQL statement shows the differences in the `basic_ctas` table.

SQL

```
EXPORT TABLE basic_ctas;
```

Output

Text

```
CREATE TABLE basic_ctas (
    "col_int" INT NOT NULL,
    "col_bigint" BIGINT NOT NULL,
    "col_id" INT NOT NULL,
    "col_point" POINT NOT NULL,
    "col_timestamp" TIMESTAMP NOT NULL,
    "col_varchar" VARCHAR(536870912) COMPRESSION DYNAMIC NULL
)
REDUNDANCY cde (PARITY),
REDUNDANCY column_metadata (COPY),
REDUNDANCY data (PARITY),
REDUNDANCY index (PARITY),
REDUNDANCY pdf (PARITY),
REDUNDANCY skip_lists (COPY),
REDUNDANCY stats (PARITY),
REDUNDANCY summary_stats (PARITY),
REDUNDANCY table_of_contents (COPY),
STORAGESPACE "ss0",
SEGMENTSIZE 4;
```

In this output, the table options for `REDUNDANCY`, `STORAGESPACE`, and `SEGMENTSIZE` are all default table settings.

CTAS Using a Full Table Definition

This example CTAS statement includes a more detailed table definition. The definition includes new columns for the TimeKey, Clustering Key, and secondary indexes. The example also makes various changes from the `original_table` schema:

- Different column default value
- Different compression scheme (dynamic compression)
- New TimeKey granularity of 1 hour
- Three columns in the Clustering Key
- Different secondary index types (`NGRAM` and `SPATIAL`)

© 2026 Ocient, Inc.

SQL

```sql
CREATE TABLE complex_ctas (
    col_amt INT NOT NULL,
    col_phone BIGINT NOT NULL COMPRESSION DYNAMIC,
    col_id INT NOT NULL,
    col_point POINT NOT NULL DEFAULT 'POINT(0 0)',
    col_timestamp TIMESTAMP TIME KEY BUCKET (1,HOUR) NOT NULL,
    col_varchar VARCHAR(255) COMPRESSION DYNAMIC,
    CLUSTERING KEY ck (col_amt, col_phone, col_id)
)
WITH
    INDEX idx_01 (col_varchar) USING NGRAM(3),
    INDEX idx_02 (col_point) USING SPATIAL
AS (
    SELECT * FROM original_table );
```

CTAS Using a Subset of Table Columns

This example selects only a subset of columns, `col_int`, `col_bigint`, and `col_id`, from the original table to insert into the new `subset` table. The example also specifies alternate table options for `REDUNDANCY` and `SEGMENTSIZE`.

SQL

```sql
CREATE TABLE subset (
    col_amt INT NOT NULL,
    col_phone BIGINT NOT NULL,
    col_id INT NOT NULL
)
WITH
    REDUNDANCY cde (PARITY),
    SEGMENTSIZE 3
AS (
    SELECT
        col_int,
        col_bigint,
        col_id
FROM
    original_table );
```

Due to limitations of the JDBC API, the reported modified row count might not be accurate for tables larger than two billion rows.

CTAS Using Transformations on Columns

This example performs various transformation functions on the original columns as it selects them for the new table. These include:

- `col_int_multiply` column is the multiplication of the `col_int` values by 10.
- `col_month_add` column is the result of adding three months to each `col_timestamp` column value.
- `col_year` column is the extraction of the year value from each `col_timestamp` column value.
- `col_substring` column contains the first three characters from each `col_varchar` column value.

SQL

```
CREATE TABLE calc_table (
    col_int_multiply INT NOT NULL,
    col_month_add TIMESTAMP NOT NULL,
    col_year INT NOT NULL,
    col_substring VARCHAR(255)
)
AS (
    SELECT
        col_int * 10,
        ADD_MONTHS(col_timestamp, 3),
        YEAR(col_timestamp),
        SUBSTRING(col_varchar, 1, 3)
FROM
    original_table );
```

CTAS USING LOADERS

Specify one or more Loader Nodes for executing the CTAS SQL statement. If you do not use this option, the Ocient System uses all Loader Nodes that are live to execute the SQL statement.

This statement is useful for managing loading operations, particularly when balancing multiple loads of different sizes and resource requirements. Alternatively, this statement can also help simplify small batch loads by sourcing the data from a single Loader Node.

Syntax

Parameter	Type	Description
`streamloader`	string	A unique name for the Loader Node. Identify the names of Loader Nodes from the `sys.nodes` table by using this query: `SELECT name FROM sys.nodes;` If the name of the Loader Node contains special characters, you must enclose it in quotes, such as `"stream-loader1"`.
`query`	string	A `SELECT` query that defines values or a table and any of its columns to use for data in the specified `table_name` table.

For the query to execute successfully, the specified names of the Loader Nodes must:

- Identify nodes that are live.
- Identify nodes that have the Loader role.

Examples

Create a table named `my_schema.my_ctas_table_2` with a clustering index named `idx` on the `int_col` column with the values in the `int_col` column in the table named `my_schema.my_table`. Use the Loader Node named `stream-loader1` to execute this SQL statement.

SQL

```
CREATE TABLE my_schema.my_ctas_table_2 (
   CLUSTERING INDEX idx (int_col)
 )
 USING LOADERS "stream-loader1"
 AS (SELECT int_col FROM my_schema.my_table);
```

In this example, execute the same CTAS SQL statement with two Loader Nodes named `stream-loader2` and `stream-loader3`.

```sql
CREATE TABLE my_schema.my_ctas_table_2 (
   CLUSTERING INDEX idx (int_col)
 )
 USING LOADERS "stream-loader2","stream-loader3"
 AS (SELECT int_col FROM my_schema.my_table);
```

DROP TABLE

`DROP TABLE` removes one or more existing tables in the current database, along with all associated views.

> This action cannot be undone.

To remove a table, the logged-in user must be a system-level user or have the `DELETE TABLE` privileges for the table.

Syntax

```sql
DROP TABLE [ IF EXISTS ] table_name [, ...]
```

Parameter	Type	Description
`table_name`	string	A unique identifier for the table. You can drop multiple tables by specifying additional table names and separating each with commas.

Examples

This example drops an existing table in the current database and schema named `employees`.

```sql
DROP TABLE employees;
```

In this example, drop two tables in the current database and schema named `employees` and `departments`.

```
SQL

DROP TABLE employees, departments;
```

When you drop multiple tables, and none of them exist in the database, the database returns an error for each missing table. Use the `IF EXISTS` statement to convert the error to a warning. If you execute the `DROP TABLE` statement and only some of the tables exist while other tables are missing, the database drops the existing tables and returns warnings for each missing table.

ALTER TABLE

ALTER TABLE RENAME

`ALTER TABLE RENAME` renames an existing table.

Required Privileges

To rename a table, you must have the `ALTER TABLE` privilege for the table.

The Ocient System requires these privileges if this statement includes a change to the schema:

- `VIEW` privilege on the current schema of the table
- `VIEW TABLE` and `CREATE TABLE` privileges on the target schema (if the schema already exists)
- `CREATE TABLE` privilege on the database (if the schema does not exist)

Syntax

```
SQL

ALTER TABLE [ IF EXISTS ] old_table_name RENAME TO new_table_name
```

Parameter	Type	Description
`old_table_name`	string	The name of the table to alter.
`new_table_name`	string	The new name to replace `old_table_name`.

Example

This example renames an existing table in the current database and schema named `us.employees` to `mid_west_employees`.

SQL

```
ALTER TABLE us.employees RENAME TO mid_west_employees;
```

This example renames an existing table in the current database named `us.employees` to `north_america.employees`.

SQL

```
ALTER TABLE us.employees RENAME TO north_america.employees;
```

ALTER TABLE RENAME COLUMN

`ALTER TABLE RENAME COLUMN` renames an existing column.

To rename a column, you must have the `ALTER TABLE` privilege for the table.

Syntax

SQL

```
ALTER TABLE [ IF EXISTS ] table_name RENAME COLUMN old_column_name TO new_column_na
```

Parameter	Type	Description
`table_name`	string	A unique identifier for the table.
`old_column_name`	string	The name of the table column to alter.

Parameter	Type	Description
`new_column_name`	string	The new column name to replace `old_column_name`.

Example

This example renames an existing column `name` in the table `employees` in the current database and schema to `first_name`.

SQL

```
ALTER TABLE employees RENAME COLUMN name TO first_name;
```

ALTER TABLE ADD COLUMN

`ALTER TABLE ADD COLUMN` adds a new column to the table.

To add a column, you must have the `ALTER TABLE` privilege for the table.

New columns must either be nullable or specify a default value.

For a defined list of column parameters, see **Column Definition**. For constraints, see **Column Constraint**.

Syntax

SQL

```
ALTER TABLE [ IF EXISTS ] table_name ADD COLUMN <column_definition>;

<column_definition> ::=
   column_name [ data_type ] [ <column_constraint> [, ...] ]

<column_constraint> ::=
   TIME KEY BUCKET(bucket_granularity [, bucket_value ] )
   | [ NOT ] NULL
   | DEFAULT literal
   | COMMENT comment
   | COMPRESSION GDC [ (compression_value), EXISTING schema_name ]
   | COMPRESSION [ compression_scheme ]
```

© 2026 Ocient, Inc.

Parameter	Type	Description
`table_name`	string	A unique identifier for the table.

Examples

This example adds a BIGINT column to the `employees` table in the current database and schema with the default value of `0`.

SQL

```
ALTER TABLE employees ADD COLUMN
    new_column BIGINT
        NOT NULL
        DEFAULT 0;
```

This example adds a column that is nullable.

SQL

```
ALTER TABLE employees ADD COLUMN
    new_column BIGINT
    NULL;
```

ALTER TABLE ALTER COLUMN COMPRESSION

`ALTER TABLE ALTER COLUMN COMPRESSION` alters an existing column in the table to change its compression scheme.

Supported compression schemes are `COMPRESSION NONE`, `COMPRESSION DYNAMIC`, and `COMPRESSION ZSTD`.

> Altering the compression setting of a column only affects compression for data loaded after you execute the SQL statement.

For details about Ocient-supported compression schemes, see **Table Compression Options**.

Syntax

SQL

```
ALTER TABLE [ IF EXISTS ] table_name ALTER COLUMN column_name
    SET COMPRESSION [ compression_scheme ];
```

Parameter	Type	Description
`table_name`	string	The name of the table containing the column to alter.
`column_name`	string	The name of the column to alter.
`compression_scheme`	string	Supported values for the compression schemes are: `COMPRESSION NONE` specifies no compression applied. `COMPRESSION DYNAMIC` applies only if the column data is dynamically determined to be compressible. `COMPRESSION ZSTD` applies to VARCHAR columns as well as other data types. For this option only, you can specify these additional parameters: `compression_level` — This value signifies how much compression the data receives. The default value is `0`. The full range of values is from `-7` through `15`. The database uses less memory when this value is lower, whereas more memory when this value is larger. Larger values provide better compression. `dictionary_size` — Dictionary size specified as a positive integer that signifies the size of the shared compression dictionary in bytes. The default value is `32768` (32K). The full range of values is from `4096` (4K) through `1048576` (1MB). This value denotes the amount of memory consumed during segment generation. In general, larger values provide better compression but use more memory.

Examples

This example alters the compression scheme for the column `employee_name` in the table `employees` in the current database and schema.

SQL

```
ALTER TABLE employees ALTER COLUMN employee_name
    SET COMPRESSION LZ4;
```

This example alters the compression scheme to Zstandard for the column `employee_name` in the table `employees` in the current database and schema.

SQL

```
ALTER TABLE employees ALTER COLUMN employee_name
    SET COMPRESSION ZSTD
        compression_level=5,
        dictionary_size=32768;
```

ALTER TABLE ALTER REDUNDANCY

`ALTER TABLE ALTER REDUNDANCY` alters the segment part redundancy for future segments of an existing table.

Note that altering a segment part redundancy setting only affects data loaded after applying the SQL statement.

SQL

```
ALTER TABLE [ IF EXISTS ] table_name ALTER REDUNDANCY segment_part (redundancy_sche
```

Parameter	Type	Description
`table_name`	string	The name of the table that you want to alter.
`segment_part`	string	Specifies the segment part redundancy of the table. Supported values include: `\{ DATA | MANIFEST | INDEX | PDF | CDE | STATS }` These settings are defined as follows: `DATA` : The actual data for the table. `MANIFEST` : Header information stored about the data, which describes how to locate any given cluster of rows within the data. `INDEX` : The index of the data used for quicker lookups and better query performance. `STATS` : Used in the optimizer, the probability density function and combinable distinct estimators used to make better optimizations to query plans.
`redundancy_scheme`	string	The redundancy scheme. Supported values include: `\{ COPY | PARITY }` These settings are defined as follows: `COPY` — The system stores a copy of the bytes throughout the storage cluster to ensure redundancy. This option uses more storage but is faster during rebuilds and node outages. `PARITY` — The system uses the parity encoding specified

Parameter	Type	Description
		on the storage cluster, and uses parity bits to ensure redundancy for the data. This option uses less storage but is slower during rebuilds and node outages.

Example

This example alters the `STATS` part to `COPY` redundancy.

SQL

```
ALTER TABLE employees ALTER REDUNDANCY STATS (COPY);
```

ALTER TABLE DROP COLUMN

`ALTER TABLE DROP COLUMN` drops an existing column from the table.

You cannot remove the TimeKey column and the clustering key columns from the table.

When you remove a column, the database does not remove or free any actual data.

SQL

```
ALTER TABLE [ IF EXISTS ] table_name DROP COLUMN column_name [ IF EXISTS ]
```

Parameter	Type	Description
`table_name`	string	The name of the table to alter.
`column_name`	string	The name of the column to drop.

Example

This example removes a column named `address` from the table `employees`.

SQL

```
ALTER TABLE employees DROP COLUMN address;
```

ALTER TABLE STREAMLOADER_PROPERTIES [#alter-table-streamloader_properties]

`ALTER TABLE STREAMLOADER_PROPERTIES` resets the table streamloader properties to the provided string. The properties string must be in valid JSON format.

The database registers streamloader changes dynamically. Therefore, you do not need to restart nodes or take other actions for the changes to take effect.

Any properties not specified in the string default to the system-wide setting.

SQL

```
ALTER TABLE [ IF EXISTS ] table_name STREAMLOADER_PROPERTIES streamloader_json
```

Parameter	Type	Description
`table_name`	string	The name of the table to alter.
`streamloader_json`	string	The streamloader properties to alter. See this table for a list of all supported properties.

Configuring Streamloader Properties

`STREAMLOADER_PROPERTIES` is a field on the table metadata that must be written as a JSON string in order to be read properly. The database can dynamically render any changes to `STREAMLOADER_PROPERTIES` with the `ALTER TABLE` SQL statement. You can set Loader Node properties for a new table as a parameter in the `CREATE TABLE` SQL statement. You do not need to restart the database node for the changes to take effect.

Per-Table Streamloader Properties

Parameter	Data type	Description
`pageQueryExclusionDuration`	An integer in nanoseconds (ns) or a string with the suffix `ns`, `us`, `ms`, or `s` appended. For example: `"10s"` = 10 seconds, `"1000us"` = 1,000 microseconds	Per-table configuration for the time interval for pages that should be excluded from queries. The database excludes pages with time column values that are greater than the duration of the query. A value of `0` means the database does not exclude any pages.

Parameter	Data type	Description
		By default, this value is set to `0` if not specified.

Example

This example sets the Loader Node properties of the table `employees` to `{"pageQueryExclusionDuration" : "30s"}`. This means that any pages added less than 30 seconds ago will not be included in query results.

SQL

```
ALTER TABLE employees
    STREAMLOADER_PROPERTIES '{
        "pageQueryExclusionDuration" : "30s"
        }';
```

ALTER TABLE DISABLE INDEX

The `ALTER TABLE DISABLE INDEX` statement instructs future queries not to use the specified indexes, but existing segments and new segments continue to have the index available in case you enable the index again.

All secondary indexes except for secondary clustering key indexes can be disabled. Trying to disable other types of indexes generates an error.

You can specify the index by name or UUID.

Syntax

SQL

```
ALTER TABLE [ IF EXISTS ] table_name DISABLE INDEX
    { index_name_or_uuid | IN (index_name_or_uuid [,...]) }
```

Parameter	Type	Description
`table_name`	string	The name of the table to alter.

Parameter	Type	Description
`index_name_or_uuid`	string	The name or UUID of the index to disable. If you specify the index by name, the name must match an existing index. If you specify a UUID instead, it does not have to match an existing index. Therefore, you can disable a dropped index using its UUID, which ensures it is not used within old segments that were loaded with the index. You can get a list of index names and UUIDs in your database by referencing the **sys.indexes** table in the system catalog.

Examples

This example disables an existing index named `current_idx` on the table `employees`.

SQL

```
ALTER TABLE employees DISABLE INDEX current_idx;
```

This example disables an existing or dropped index with the UUID `5c15d8de-36fa-4055-9bdc-3f1750aaeea0`.

SQL

```
ALTER TABLE employees DISABLE INDEX '5c15d8de-36fa-4055-9bdc-3f1750aaeea0';
```

This example disables both indexes `current_idx` and `other_idx` on the table `employees`.

SQL

```
ALTER TABLE employees DISABLE INDEX IN (current_idx, other_idx);
```

ALTER TABLE ENABLE INDEX

The `ALTER TABLE ENABLE INDEX` statement reverts the operation performed by the `ALTER TABLE DISABLE INDEX` statement.

Syntax

```sql
ALTER TABLE [ IF EXISTS ] table_name ENABLE INDEX index_name_or_uuid
```

Parameter	Type	Description
`table_name`	string	The name of the table to alter.
`index_name_or_uuid`	string	The name or UUID of the index to enable. If you specify the index by name, the name must match an existing index. If you specify a UUID instead, it does not have to match an existing index. Therefore, you can disable a dropped index using its UUID, which ensures it is not used within old segments that were loaded with the index. You can get a list of index names and UUIDs in your database by referencing the **sys.indexes** table in the system catalog.

Examples

This example enables an existing index named `current_idx` on the table `employees`.

```sql
ALTER TABLE employees ENABLE INDEX current_idx;
```

This example enables an existing or dropped index with the UUID `5c15d8de-36fa-4055-9bdc-3f1750aaeea0`.

```sql
ALTER TABLE employees ENABLE INDEX '5c15d8de-36fa-4055-9bdc-3f1750aaeea0';
```

This example enables both indexes `current_idx` and `other_idx` on the table `employees`.

```sql
ALTER TABLE employees ENABLE INDEX IN (current_idx, other_idx);
```

ALTER TABLE ENABLE RETENTION POLICY AGE

This `ALTER TABLE` SQL statement enacts a new retention policy on the specified table. A table can have only one retention policy. For details about retention policies, see **Table Retention Policies**.

Required Privileges

You must have the `ALTER` and `DELETE` privileges for the specified table.

> ⚠ Enacting a new retention policy on an existing table already loaded with data might cause the system to delete many rows.

Syntax

SQL

```
ALTER TABLE [ IF EXISTS ] table_name ENABLE RETENTION POLICY AGE
    retention_granularity retention_value
```

Parameter	Type	Description
`table_name`	string	The name of the table to alter.
`retention_granularity`	numeric	A number that represents the amount of time before the system deletes data in the table. You must pair this number with a specific time type, represented by the `retention_value` parameter. For details about setting up retention policies, see **Table Retention Policies**.
`retention_value`	string	The unit of time that represents how long the table retains data. Supported values are: `WEEK`, `DAY`, `HOUR`, `MINUTE`, `SECOND`, `MILLISECOND` You must pair this number with the amount of time, represented by the `retention_granularity` parameter. For example, `1 DAY` schedules the table retention to a granularity of one day.

Example

This example creates a new retention policy for `my_table` that deletes any rows older than 1 day.

```
SQL

ALTER TABLE my_table ENABLE RETENTION POLICY AGE 1 DAY;
```

ALTER TABLE DISABLE RETENTION POLICY

This `ALTER TABLE` SQL statement disables a retention policy on the specified table. For details about retention policies, see **Table Retention Policies**.

Required Privileges

You must have the `ALTER TABLE` system privilege and the `ALTER` privilege for the specified table.

Syntax

```
SQL

ALTER TABLE [ IF EXISTS ] table_name DISABLE RETENTION POLICY
```

Parameter	Type	Description
`table_name`	string	The name of the table to alter.

Example

This example removes the retention policy for `my_table`.

```
SQL

ALTER TABLE my_table DISABLE RETENTION POLICY;
```

EXPORT TABLE

`EXPORT TABLE` shows the `CREATE TABLE` statement for an existing table in the current database.

To export a table, you must have the `SELECT TABLE` privilege for the table.

```
SQL

EXPORT TABLE table_name
```

Parameter	Type	Description
`table_name`	string	The name of the table that you want to export.

Example

This example exports an existing table in the current database and schema named `trades`.

```
SQL

EXPORT TABLE trades;
```

Output

SQL

```
CREATE TABLE "admin@system"."trade_test" (
    "id" UUID NOT NULL,
    "ticker_symbol" VARCHAR(1048576) COMPRESSION GDC(2) NOT NULL,
    "t_type" VARCHAR(1048576) COMPRESSION GDC(1) NOT NULL,
    "raw_ticker_data" VARCHAR(1048576) COMPRESSION DYNAMIC NULL,
    "created_at" TIMESTAMP TIME KEY BUCKET(1, HOUR) NOT NULL,
    "array_of_tuples" TUPLE<<VARCHAR(1048576) COMPRESSION DYNAMIC,TINYINT,BIGINT,DO
    CLUSTERING INDEX "idx_ticker_symbol_type" ("ticker_symbol", "t_type"),
    INDEX "idx_type" ("t_type")
)
REDUNDANCY cde (PARITY),
REDUNDANCY manifest (COPY),
REDUNDANCY pdf (PARITY),
REDUNDANCY stats (PARITY),
REDUNDANCY column_metadata (COPY),
REDUNDANCY index (PARITY),
REDUNDANCY summary_stats (PARITY),
REDUNDANCY skip_lists (COPY),
REDUNDANCY data (PARITY),
STORAGESPACE "storage",
SEGMENTSIZE 4,
STREAMLOADER_PROPERTIES '{"pageQueryExclusionDuration" : "30s"}';
CREATE INDEX "new_idx" ON "admin@system"."trade_test" ("raw_ticker_data") USING HAS
```

Related Links

Core Elements of an Ocient System

Data Query Language (DQL) Statement Reference

Database Password Security Settings

Data Control Language (DCL) Statement Reference

Data Manipulation Language (DML) Statement Reference

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Views

This group of DDL SQL statements allows database administrators to manage views. Database administrators can create, modify, drop, and export views. You can view information about views using the `sys.views` system catalog table.

For information on database components, see the pages on **Databases**, **Schemas**, **Tables**, and **Indexes**.

CREATE VIEW

`CREATE VIEW` creates a new view in the current database or replaces an existing view. For view creation, the name of the view must be distinct from the name of any existing views in the database.

To create a view, you must have both the `CREATE VIEW` privilege for the current database and the `SELECT` privilege on all directly referenced tables and views in the query.

You must have `DROP` privileges on the view. You also must have `CREATE VIEW` privileges on the database.

Syntax

```
SQL

CREATE [ OR REPLACE ] VIEW [ IF NOT EXISTS ] view_name AS query
```

Parameter	Type	Description
`view_name`	string	A distinct identifier used to name the view.
`query`	string	A `SELECT` query that defines the data from a table used to create the view.

Example

This example creates an existing view in the current database and schema named `option_trades`.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

SQL

```
CREATE VIEW option_trades
    AS SELECT * FROM trades WHERE type = 'OPTIONS';
```

DROP VIEW

`DROP VIEW` removes one or more existing views in the current database.

To remove a view, you must have the `DROP VIEW` privilege for the view.

Syntax

SQL

```
DROP VIEW [ IF EXISTS ] view_name [, ...]
```

Parameter	Type	Description
`view_name`	string	The identifier of the view to drop. You can drop multiple views by specifying additional view names and separating each with commas.

Example

This example drops an existing view in the current database and schema named `star_employees`.

SQL

```
DROP VIEW star_employees;
```

This example drops two views named `star_employees` and `bad_employees`.

SQL

```
DROP VIEW star_employees, bad_employees;
```

ALTER VIEW RENAME

`ALTER VIEW RENAME` renames an existing view.

Required Privileges

To rename a view, you must have the `ALTER VIEW` privilege for the view.

The Ocient® System requires these privileges if this statement includes a change to the schema:

- `VIEW` privilege on the current schema of the view
- `VIEW VIEW` and `CREATE VIEW` privileges on the target schema (if the schema already exists)
- `CREATE VIEW` privilege on the database (if the schema does not exist)

Syntax

```
SQL

ALTER VIEW [ IF EXISTS ] old_view_name RENAME TO new_view_name
```

Parameter	Type	Description
`old_view_name`	string	The identifier of the view to rename.
`new_view_name`	string	The new name for the specified view.

Examples

This example renames an existing view in the current database and schema named `star_employees` to `star_mid_west_employees`.

```
SQL

ALTER VIEW star_employees RENAME TO star_mid_west_employees;
```

This example renames an existing view in the current database named `us.star_employees` to `us.star_mid_west_employees`.

```sql
ALTER VIEW us.star_employees RENAME TO us.star_mid_west_employees;
```

ALTER VIEW AS

`ALTER VIEW AS` modifies an inner query of the existing view.

To modify the query for an existing view, you must be a system-level user or have the `ALTER VIEW` privilege for the view. You must also have the `SELECT` privilege on all referenced tables and views in the new query.

Syntax

```sql
ALTER VIEW [ IF EXISTS ] view_name AS query
```

Parameter	Type	Description
`view_name`	string	The identifier for the view to alter.
`query`	string	A `SELECT` query that defines the new data from a table used to alter the view.

Example

This example alters a view in the current database and schema named `star_employees`.

```sql
ALTER VIEW star_employees AS SELECT * FROM sys.tables;
```

EXPORT VIEW

`EXPORT VIEW` shows the `CREATE VIEW` statement for an existing view in the current database.

To export a view, the logged-in user must be a system-level user or have the `READ VIEW` right for the view.

Syntax

```
SQL

EXPORT VIEW view_name
```

Parameter	Type	Description
`view_name`	string	The identifier for the view to export.

Example

This example exports an existing view in the current database and schema named `students`.

```
SQL

EXPORT VIEW students;
```

Related Links

Core Elements of an Ocient System

Data Query Language (DQL) Statement Reference

Database Password Security Settings

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Indexes

This group of DDL SQL statements allows database administrators to manage indexes. Database administrators can create and drop indexes. You can view information about indexes using the `sys.indexes` system catalog table.

For information on other database components, see the pages on **Databases**, **Schemas**, **Tables**, and **Views**.

CREATE INDEX

`CREATE INDEX` creates a new secondary index.

Indexes help optimize database queries when created on columns that are frequently referenced. For more information on how Ocient® indexes operate, see **Secondary Indexes**.

Creating an index does not trigger re-indexing of existing segments. Only segments generated after the `CREATE INDEX` is issued contain the new index.

Indexes can be created on columns containing various different data types as long as the requirements are met. Please note that depending on the data type, the system can assign different index types by default if you decline to specify which index type to use. The name must be distinct from the name of any existing index on the table.

> You can apply indexes regardless of whether they have GDC compression.

Syntax

SQL

```
CREATE INDEX [ IF NOT EXISTS ] index_name ON table (column_name) [ USING <index_typ

<index_type> ::=
   INVERTED | HASH | NGRAM [ (n_value) ] | SPATIAL | ZONE_MAP
```

© 2026 Ocient, Inc.

Parameter	Type	Description
`index_name`	string	An identifier for the index to create. The name must be distinct from the name of any existing index on the table.
`table`	string	The name of the table for the index.
`column_name`	string	The name of the column for the index. Identical indexes on the same column are not allowed. A column can only have multiple indexes if they are of different types or parameters.
`n_value`	integer	Optional. When used with an NGRAM index, this numeric value specifies the character length of the substrings to be indexed. If unspecified, this value defaults to `3`.

Index Types (`<index_type>`)

Ocient supports four index types alongside the clustering index: `INVERTED`, `HASH`, `NGRAM`, and `SPATIAL`.

An index notionally stores a mapping of a column value to the rows that contain that value, and the index type differentiates the format in which the column values are stored and accessed.

Unless an index type is explicitly specified with a `USING` clause, the data type of a column determines a default index type that the system creates. For information on index type defaults, see **Index Type Requirements and Defaults**.

For container data types (e.g., arrays and tuples), the index stores the internal elements of the container, and is used on predicates that target the internal values. However, a mapping of NULL column values is generally stored for both scalar and container data types, so the index can always be used for `column IS NULL` predicates.

Index Type	Primary Data Types	Primary Usage Description
`INVERTED`	Fixed-length numeric columns.	Stores whole column value internally, meaning its storage size is approximately the same as the width of the data type. Supports lookups using strict equality

Index Type	Primary Data Types	Primary Usage Description
		or range comparisons. For details, see the **INVERTED index** section.
`HASH`	Variable-length character columns	Stores a hash of the indexed column value rather than the full value. Primarily used for exact comparisons. For details, see the **HASH index** section.
`NGRAM`	Variable-length character columns	Stores substrings equal in size to its `n_value`. Storage requirements can greatly vary depending on column data size, width and cardinality. Supports exact string comparison and filters including `LIKE`, `NOT LIKE`, `SIMILAR TO` and `NOT SIMILAR TO`. For details, see the **NGRAM index** section.
`SPATIAL`	Geospatial columns (`POINT`, `LINESTRING`, `POLYGON`)	Groups geographic objects for bounding-box filtering. For details, see the **SPATIAL index** section.
`ZONE_MAP`	Fixed-length columns (INT, BIGINT, SMALLINT, FLOAT, DOUBLE, BINARY, DECIMAL, HASH, CHAR, IP / IPV4, TIME, TIMESTAMP, DATE, UUID)	Supports range and exact matches on a whole segment level. The system stores the minimum and maximum values for the column per-segment.

For further description and examples of the index types, see **Secondary Indexes**.

Examples

This example creates an index named `new_idx` on the `address` column of the table. Because `address` is a `VARCHAR` column, this index defaults to the HASH index type.

SQL

```
CREATE INDEX new_idx ON employees (address);
```

This example creates an index of type NGRAM on the `address` column. As the NGRAM has no specified `n_value`, it defaults to indexing substrings of three characters long.

```SQL
CREATE INDEX ngram_address_idx ON employees (address) USING NGRAM;
```

This example creates an index on a component of the `tuple_col` column. As this column is of data type `INT`, the index defaults to using the `INVERTED` type.

```SQL
CREATE INDEX tuple_index ON employees (tuple_col[1]);
```

This example creates an index on a component of the `point_col` column. As this column is of data type `POINT`, the index defaults to using the `SPATIAL` type.

```SQL
CREATE INDEX spatial_index ON employees (point_col)
```

DROP INDEX

`DROP INDEX` drops a secondary index on a table.

After an index is dropped, new segments that are generated do not contain the new index. However, no existing segments will be altered. This means that until a segment is rebuilt, you can still use the removed index internally, and the system does not reclaim the storage space the removed index occupied.

Syntax

```SQL
DROP INDEX [ IF EXISTS ] index_name ON table_name
```

Parameter	Type	Description
`index_name`	string	An identifier for the index to drop.
`table_name`	string	The name of the table with the index to drop.

Example

This example drops the index named `new_idx` on the `employees` table.

SQL

```
DROP INDEX new_idx ON employees;
```

Related Links

Core Elements of an Ocient System

Data Query Language (DQL) Statement Reference

Database Password Security Settings

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Pipelines

Data pipelines enable you to load data from your chosen source into the OcientAIQ™ Unified Data Platform. You can preview a data pipeline using the `PREVIEW PIPELINE` SQL statement to test the creation statement and any required data transformation. Then, you can create a data pipeline using the `CREATE PIPELINE` statement. To manage the load, use the `START PIPELINE` statement to start the load and the `STOP PIPELINE` statement to stop it. You can evolve the schema of a data pipeline using the `ALTER PIPELINE` statement. You can also rename a data pipeline using the `ALTER PIPELINE RENAME` statement. To see the full definition of a created pipeline, use the `EXPORT PIPELINE` statement. When you finish with the load, you can use the `DROP PIPELINE` statement to remove the data pipeline. You can create user-defined data pipeline functions using the `CREATE PIPELINE FUNCTION` statement and remove the function using the `DROP PIPELINE FUNCTION` statement. Also, you can administer privileges for data pipelines and data pipeline functions. For details, see **Data Control Language (DCL) Statement Reference**.

CREATE PIPELINE

`CREATE PIPELINE` defines a data pipeline that you can execute with the `START PIPELINE` SQL statement. Specify the type of load, data source, and data format.

You must have the `ALTER` privilege on the pipeline. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

SQL

```
CREATE [ OR REPLACE ] [ BATCH | CONTINUOUS | TRANSACTIONAL ] PIPELINE [ IF NOT EXIS
    [ <advanced_pipeline_options> ]
    [ BAD_DATA_TARGET <kafka_bad_data_target> ]

    SOURCE ( <s3_source> | <filesystem_source> | <hdfs_source> | <kafka_source> )
        [ MONITOR ( <sqs_monitor> | <kafka_monitor> ) ]
    [ LOOKUP lookup_source <lookup_options> ]  [ ... ]

    EXTRACT
        ( <delimited_extract_options> | <asn1_extract_options> |
        <avro_extract_options> | <json_extract_options> |
        <parquet_extract_options> | <binary_extract_options> |
        <xml_extract_options> )
        [ <general_extract_options> ]

    [ INSERT ] INTO destination_table_name

    SELECT
        expression as col_alias_target,
        expression2 as col_alias_target2,
        ...
    [ WHERE filter_expression ]

    [ [ INSERT ] INTO destination_table_name_n
    SELECT
        expression_n as col_alias_target_n,
        expression2_n as col_alias_target2_n,
        ...
    [ WHERE filter_expression_n ] ] [ ... ]

/*********************/
/* ADVANCED OPTIONS */
/*********************/
advanced_pipeline_options ::=
    [ CORES processing_cores ]
    [ PARTITIONS file_partitions ]
    [ BATCH_SIZE number_of_rows ]
    [ RECORD_NUMBER_FORMAT record_number_format ]

/****************************/
/* BAD_DATA_TARGET OPTIONS */
/****************************/
```

```
kafka_bad_data_target ::=
    KAFKA
        BOOTSTRAP_SERVERS bootstrap_servers
        TOPIC topic_name
        [ CONFIG config_option ]

/******************/
/* SOURCE OPTIONS */
/******************/
s3_source ::=
    S3
        <file_based_source_options>
        BUCKET bucket_name
        ( FILTER | FILTER_GLOB | FILTER_REGEX | OBJECT_KEY ) specifiers
        [ REGION region ]
        [ ENDPOINT endpoint ]
        [ ENABLE_PATH_STYLE_ACCESS enable_path_style_access ]
        [ ACCESS_KEY_ID access_key_credentials ]
        [ SECRET_ACCESS_KEY secret_key_credentials ]
        [ SESSION_TOKEN session_token ]
        [ ROLE_ARN role_arn ]
        [ ASSUME_ROLE_CONFIG assume_role_config]
        [ MAX_CONCURRENCY parallel_connections ]
        [ READ_TIMEOUT num_seconds ]
        [ REQUEST_DEPTH num_requests ]
        [ REQUEST_RETRIES num_retries ]
        [ HEADERS headers ]

filesystem_source ::=
    FILESYSTEM
        <file_based_source_options>
        ( FILTER | FILTER_GLOB | FILTER_REGEX ) specifiers

hdfs_source ::=
    HDFS
        <file_based_source_options>
        ( FILTER | FILTER_GLOB | FILTER_REGEX ) specifiers
        ENDPOINT endpoint
        [ CONFIG hdfs_config ]

kafka_source ::=
    KAFKA
        BOOTSTRAP_SERVERS bootstrap_servers
        TOPIC topic_name
```

```
        [ WRITE_OFFSETS write_offsets ]
        [ CONFIG config_option ]
        [ AUTO_OFFSET_RESET ( 'latest' | 'earliest' )

file_based_source_options ::=
    [ PREFIX prefix ]
    [ COMPRESSION_METHOD 'gzip' ]
    [ SORT_BY ( 'filename' | 'created' | 'modified' )
        [ SORT_DIRECTION ( ASC | DESC ) ] ]
    [ SORT_REWRITE sort_rewrite ]
    [ START_FILENAME start_filename ]
    [ END_FILENAME end_filename ]
    [ START_CREATED_TIMESTAMP start_created_timestamp ]
    [ END_CREATED_TIMESTAMP end_created_timestamp ]
    [ START_MODIFIED_TIMESTAMP start_modified_timestamp ]
    [ END_MODIFIED_TIMESTAMP end_modified_timestamp ]

/******************/
/* MONITOR OPTIONS */
/******************/
sqs_monitor ::=
    SQS
        <general_monitor_options>
        QUEUE_URL queue_url
        [ REGION region ]
        [ ENDPOINT endpoint ]
        [ ACCESS_KEY_ID access_key_id ]
        [ SECRET_ACCESS_KEY secret_access_key ]

kafka_monitor ::=
    KAFKA
        <general_monitor_options>
        BOOTSTRAP_SERVERS bootstrap_servers
        TOPIC topic
        AUTO_OFFSET_RESET ( 'latest' | 'earliest' )
        [ GROUP_ID group_id ]
        [ MESSAGE {
            FILENAME file_selector
            TIMESTAMP timestamp_selector
            SIZE size_selector
        } ]
        [ CONSUMER {
            MAX_MESSAGES max_messages
            TIMEOUT consumer_timeout
```

```
        } ]

general_monitor_options ::=
    [ POLLING_INTERVAL polling_interval ]
    [ BATCH {
        MAX_FILES max_files
        TIMEOUT batch_timeout
        LOOKBACK lookback
     } ]

/********************/
/* LOOKUP OPTIONS  */
/********************/
lookup_options ::=
    CONNECTION_TYPE connection_type
    CONNECTION_STRING connection_string
    LOOKUP_SCHEMA lookup_schema
    LOOKUP_TABLE lookup_table
    [ REFRESH_PERIOD refresh_period ]
    [ CONFIG config_json ]

/********************/
/* EXTRACT OPTIONS */
/********************/
general_extract_options ::=
    [ CHARSET_NAME charset_name ]
    [ COLUMN_DEFAULT_IF_NULL column_default_if_null ]
    [ NULL_STRINGS null_strings ]
    [ TRIM_WHITESPACE trim_whitespace ]
    [ VALIDATE_CHARACTERS validate_characters ]
    [ REPLACE_INVALID_CHARACTERS replace_invalid_characters ]
    [ REPLACEMENT_CHARACTER replacement_character ]

delimited_extract_options ::=
    FORMAT ( 'delimited' | 'csv' )
        [ COMMENT_CHAR comment_char ]
        [ EMPTY_FIELD_AS_NULL empty_field_as_null ]
        [ ESCAPE_CHAR escape_char ]
        [ FIELD_DELIMITER field_delimiter ]
        [ FIELD_OPTIONALLY_ENCLOSED_BY enclosure_char ]
        [ HEADERS delimited_headers ]
        [ NUM_HEADER_LINES num_header_lines ]
        [ NUM_FOOTER_LINES num_footer_lines ]
        [ RECORD_DELIMITER record_delimiter ]
```

```
        [ SKIP_EMPTY_LINES skip_empty_lines ]
        [ OPEN_ARRAY open_array_char ]
        [ CLOSE_ARRAY close_array_char ]
        [ ARRAY_ELEMENT_DELIMITER array_delimiter ]
        [ OPEN_OBJECT open_object_char ]
        [ CLOSE_OBJECT close_object_char ]
        [ STRIP_ARRAY_ELEMENT_QUOTES strip_array_element_quotes ]
        [ STRIP_FIELD_QUOTES strip_field_quotes ]
        [ TRIM_ARRAY_ELEMENTS trim_array_elements]

asn1_extract_options ::=
    FORMAT 'asn.1'
        SCHEMA {
            URL url_asn1
            RECORD_TYPE record_type
        }

avro_extract_options ::=
    FORMAT 'avro'
        [ SCHEMA {
            [ INLINE inline_string ]
            [ INFER_FROM 'sample_file' ]
            [ URL url_avro ]
            [ CONFIG config_avro ]
            [ SUBJECT subject ]
            [ SCHEMA_REGISTRY_ID_LOCATION ( 'header_or_value' | 'value' | 'none' )
        } ]

binary_extract_options ::=
    FORMAT 'binary'
        RECORD_LENGTH length_in_bytes
        [ ENDIANNESS ( 'big' | 'little' ) ]
        [ AUTO_TRIM_PADDING auto_trim_padding ]
        [ PADDING_CHARACTER padding_character ]

json_extract_options ::=
    FORMAT 'json'

parquet_extract_options ::=
    FORMAT 'parquet'
        SCHEMA {
            INFER_FROM infer_from
        }
```

```
xml_extract_options ::=
    FORMAT 'xml'
```

Pipeline Identity and Naming

The name of a data pipeline is unique in an Ocient® System. Reference the pipeline in other SQL statements like `START PIPELINE`, `STOP PIPELINE`, and `DROP PIPELINE` using the name of the pipeline. The SQL statement throws an error if a pipeline with the same name already exists unless you specify the `IF NOT EXISTS` option. You can rename a pipeline with the `ALTER PIPELINE RENAME` SQL statement.

Update a Pipeline

You can update a pipeline with the `OR REPLACE` clause in the `CREATE PIPELINE` SQL statement. Use this clause when you want to continue loading from the current place in a continuous load, but you need to modify transformations or other settings. If you specify the `OR REPLACE` clause and the pipeline already exists, the database replaces the original pipeline object with the options specified in the new `CREATE OR REPLACE PIPELINE` statement. When you replace an existing pipeline, the pipeline retains its current position in the source data so that data is not duplicated when the pipeline is resumed with a `START PIPELINE` SQL statement. First, you must stop a pipeline before executing the `CREATE OR REPLACE` SQL statement.

Data Pipeline Modes

You can define pipelines in either batch `BATCH`, continuous `CONTINUOUS`, or transactional `TRANSACTIONAL` mode.

- File sources (e.g., `s3`, `filesystem`) support `BATCH` and `CONTINUOUS` modes. File-based loads default to `BATCH` mode if you do not specify this keyword.
- Apache® Kafka® only supports `CONTINUOUS` mode. Loads with a Kafka source default to `CONTINUOUS` mode if you do not specify this keyword.
- When you execute the `START PIPELINE` SQL statement using a data pipeline in the `BATCH` mode, the system creates a static list of files with the `PENDING` status in the `sys.pipeline_files` system catalog table. With the `CONTINUOUS` mode, the monitor

appends new incoming files to the list of files in the `sys.pipeline_files` system catalog table.

- With the `CONTINUOUS` mode, you can apply filters for the data pipeline. The system only uses files in the consumed messages if the filenames match the filters. For this mode, these options are invalid:
  - PREFIX
  - SORT_BY
  - SORT_DIRECTION
  - SORT_REWRITE
  - START_CREATED_TIMESTAMP
  - END_CREATED_TIMESTAMP
  - START_MODIFIED_TIMESTAMP
  - END_MODIFIED_TIMESTAMP
- The `TRANSACTIONAL` mode enables the data pipeline to roll back the interim results of the load if an error occurs during load execution. The system makes all data visible only after the load completes successfully. This mode supports all options that the `BATCH` mode does. For details, see **Transactional Data Pipelines**.

Required Options

You must define certain options in every pipeline. For these options, there is no default value. Optional options need to be present only if you use specific functions. A pipeline must contain the `SOURCE`, `EXTRACT FORMAT`, and `INTO table_name ... SELECT` statements.

- **SOURCE Options**
- **EXTRACT Options**
- **Data Formats for Data Pipelines**
- **Transform Data in Data Pipelines**

These options depend on each other: `IF NOT EXISTS` and `OR REPLACE` SQL statements are mutually exclusive.

SELECT Statement and Data Transformation

Use the `INTO table_name ... SELECT` SQL statement in the `CREATE PIPELINE` SQL statement to specify how to transform the data. The `SELECT` statement includes a set of one or more expressions and a target column name in the form `expression as column_name`. The `expression` part in the statement contains a source field reference (e.g., `$1` or `$my_field.subfield`) and, optionally, the transformation function you want to apply.

If your data is nested in arrays, you can use special transformation functions, such as the **EXPLODE_OUTER** function, to expand the data into individual rows.

For details about data transformation and supported transformation functions, see **Transform Data in Data Pipelines**.

For details about data types and casting, see **Data Types for Data Pipelines** and **Data Formats for Data Pipelines**.

You can also specify metadata values, such as filename, to load in the `SELECT` SQL statement. For details, see **Load Metadata and File-Based Partitioned Data in Data Pipelines**.

This is an example of a transformation statement snippet.

SQL

```
...
FORMAT json
INTO public.orders
SELECT
    TIMESTAMP(BIGINT($created_timestamp)) as ordertime,
    metadata('filename') as source_filename,
    $order_number as ordernumber,
    $customer.first_name as fname,
    LEFT($customer.middle_initial,1) as minitial,
    $customer.last_name as lname,
    $postal_code as postal_code,
    $promo_code as promo_code,
    $order_total as ordertotal,
    DECIMAL($tax,8,2) as tax,
    CHAR[]($line_items[].product_name) as product_names,
    CHAR[]($line_items[].sku) as skus
```

© 2026 Ocient, Inc.

Optionally, you can filter the load by using the `WHERE` clause in the form `WHERE filter_expression`. This expression should evaluate to a Boolean value or NULL and can include more than one filter expression. The system loads rows that contain data matching the filter criteria when the expression evaluates to `true`. The system does not load the rows when the expression evaluates to `false` or NULL. You can include any transform in the `WHERE` clause as in the `SELECT` clause.

This is an example of a filter snippet that loads order data for customers whose last name starts with the letter `a`. Use the `COALESCE` function to return only non-NULL values.

SQL

```
...
FORMAT json
INTO public.orders
SELECT
    TIMESTAMP(BIGINT($created_timestamp)) as ordertime,
    metadata('filename') as source_filename,
    $order_number as ordernumber,
    $customer.first_name as fname,
    LEFT($customer.middle_initial,1) as minitial,
    $customer.last_name as lname,
    $postal_code as postal_code,
    $promo_code as promo_code,
    $order_total as ordertotal,
    DECIMAL($tax,8,2) as tax,
    CHAR[]($line_items[].product_name) as product_names,
    CHAR[]($line_items[].sku) as skus
WHERE
    STARTSWITH(COALESCE($customer.last_name, ''),'a')
```

NULL and Default Value Handling

The Ocient System has specific ways to handle NULL and default values in the data pipeline. Consider this information when you prepare data to load and write the `CREATE TABLE` and `CREATE PIPELINE` SQL statements.

To load NULL values, insert the NULL value in the data. In this case, if you specify `NOT NULL` for the target column in the `CREATE TABLE` statement, the data pipeline fails to load.

If you omit a column in the `SELECT` SQL statement, the data pipeline loads the default value for the column. If you do not specify the default value, the pipeline loads a NULL value. If you specify `NOT NULL` for the target column, the data pipeline also fails to load.

You can use the `DEFAULT` keyword to load a default value. In this case, if the column does not have a default value, the pipeline fails to load. If the pipeline loads a NULL into a column with a specified default value, you can use `COALESCE(<value>, DEFAULT)` to insert the default value instead of the NULL value, where `<value>` is the NULL column. You can modify the load of one column at a time in this way.

This table describes the data pipeline behavior for NULL or omitted column values.

Value in the Data Pipeline	Nullable Target Column	Default Value in Target Column	Resulting Data Pipeline Behavior
NULL	No	Value might or might not be set.	The pipeline uses the default value if the default exists and you specify the `COLUMN_DEFAULT_IF_NULL` option. Otherwise, the pipeline fails.
NULL	Yes	Value might or might not be set.	The pipeline uses the default value if the default exists and you specify the `COLUMN_DEFAULT_IF_NULL` option. Otherwise, the pipeline uses the NULL value.
Omitted column in the `SELECT` SQL statement	NULL value might or might not be set.	Yes	The pipeline uses the default value.
Omitted column in the `SELECT` SQL statement	No	No	The pipeline fails.
Omitted column in the `SELECT` SQL statement	Yes	No	The pipeline uses the NULL value.

Required Privileges

You must have the `CREATE PIPELINE` privilege on the underlying database and the `VIEW` privilege on each table in the pipeline definition to execute the `CREATE PIPELINE` SQL

statement. The table must already exist.

See the `START PIPELINE` SQL statement for the required privileges to execute a pipeline.

For details, see **Data Control Language (DCL) Statement Reference**.

Examples

Load JSON Data from Kafka

This example loads JSON data from Kafka using the `CREATE PIPELINE` SQL statement. Use the bootstrap server `192.168.0.1:9092` using the Kafka topic `orders`. Load data into the `public.orders` table. Specify the data to load as these JSON selectors:

- Identifier `$id`
- User identifier `$user_id`
- Product identifier `$product_id`
- Subtotal amount `$subtotal`
- Tax `$tax`
- Total amount `$total`
- Discount amount `$discount`
- Created time `$created_at`
- Quantity `$quantity`

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE orders_pipeline
    SOURCE KAFKA
        BOOTSTRAP_SERVERS '192.168.0.1:9092'
        TOPIC 'orders'
    EXTRACT
        FORMAT json
INTO public.orders
SELECT
    $id as id,
    $user_id as user_id,
    $product_id as product_id,
    $subtotal as subtotal,
    $tax as tax,
    $total as total,
    $discount as discount,
    $created_at as created_at,
    $quantity as quantity;
```

For a complete tutorial, see **Data Pipeline Load of JSON Data from Kafka**.

Load Delimited Data from S3

This example loads delimited data in CSV format from Amazon® Web Services℠ (AWS℠) S3. Use the `https://s3.us-east-1.amazonaws.com` endpoint with the `ocient-docs` bucket and path `metabase_samples/csv/orders.csv`. Denote path-style access using the `ENABLE_PATH_STYLE_ACCESS` option set to `true`. Specify one header line with the `NUM_HEADER_LINES` option. Load data into the `public.orders` table. Specify the data to load using the column numbers:

- Idenfier
- User identifier
- Product identifier
- Subtotal amount
- Tax
- Total amount
- Discount amount

- Created time
- Quantity

SQL

```
CREATE PIPELINE orders_pipeline
    SOURCE S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/csv/orders.csv'
        ENABLE_PATH_STYLE_ACCESS true
    EXTRACT
        FORMAT csv
        NUM_HEADER_LINES 1
INTO public.orders
SELECT
    $1 as id,
    $2 as user_id,
    $3 as product_id,
    $4 as subtotal,
    $5 as tax,
    $6 as total,
    $7 as discount,
    $8 as created_at,
    $9 as quantity;
```

For a complete tutorial, see **Data Pipeline Load of CSV Data from S3**.

Continuous File Load of Delimited Data in CSV Files

Create a data pipeline that uses continuous file loading with delimited data in CSV files. Specify to use 32 partitions and 16 cores with the `PARTITIONS` and `CORES` options, respectively. The source is S3 with the `http://endpoint.ocient.com` endpoint and `cs_data` bucket. Specify the filter `'*.csv'` to find all files with a filename that matches a glob pattern without subdirectories, for example, `data.csv`. The system filters filenames with subdirectories such as `data/data_sample.csv`.

For continuous file loading, specify the `MONITOR` option to use the Kafka monitor. Specify the `test-broker:9092` bootstrap server, `cfl_kafka_ten_adtech_flat_small` Kafka topic, and reset the offset to the smallest offset using the `earliest` value of the

`AUTO_OFFSET_RESET` option. Set the client group identifier to `84079bf1-bdc4-4b10-ba12-41ba6b17dffe`.

The format is `csv` with the record delimiter as the newline character `\n`. Load data in the `public.ad_sessions` table.

The `SELECT` statement identifies the columns to load by number. There are 39 fields in the CSV data. For each column, transform each column using cast functions. For details, see **Scalar Data Conversion Functions** for each function. See **Date and Time Functions** for the `TO_TIMESTAMP` function.

SQL

```
CREATE CONTINUOUS PIPELINE test_continuous_pipeline
    PARTITIONS 32
    CORES 16
    SOURCE S3
        ENDPOINT 'http://endpoint.ocient.com/'
        BUCKET 'cs_data'
        FILTER '*.csv'
        MONITOR kafka
            BOOTSTRAP_SERVERS 'test-broker:9092'
            TOPIC 'cfl_kafka_ten_adtech_flat_small'
            AUTO_OFFSET_RESET 'earliest'
            GROUP_ID '84079bf1-bdc4-4b10-ba12-41ba6b17dffe'
    EXTRACT
        FORMAT csv
        RECORD_DELIMITER '\n'
    INTO public.ad_sessions
SELECT
    TO_TIMESTAMP(CHAR($1), 'yyyy-MM-dd HH:mm:ss.SSSSSS', 'java') AS event_date_time
    CHAR($2)                    AS device_model,
    TINYINT($8)                 AS device_user_age,
    BOOLEAN($10)                AS device_ad_tracking_disabled,
    BINARY($11)                 AS device_mac,
    INT($14)                    AS ip_zip,
    FLOAT($19)                  AS ip_zip_latitude,
    DOUBLE($20)                 AS ip_zip_longitude,
    BIGINT($21)                 AS session_id,
    SMALLINT($32)               AS session_response_latency,
    DECIMAL($34, 10, 1)         AS session_transaction_revenue,
    CHAR($39)                   AS session_app_name
;
```

Transactional Load of Delimited Data in CSV Files

Create a data pipeline that uses transactional file loading with delimited data in CSV files. The source is S3 with the `http://endpoint.ocient.com` endpoint and `cs_data` bucket. Specify the filter `'*.csv'` to find all files with a filename that matches a glob pattern without subdirectories, for example, `data.csv`. The system filters filenames with subdirectories such as `data/data_sample.csv`.

The format is `csv`. Load data in the `public.ad_sessions` table.

The `SELECT` statement identifies the columns to load by number. For each column, transform each column using cast functions. For details, see **Scalar Data Conversion Functions** for each function. See **Date and Time Functions** for the `TO_TIMESTAMP` function.

SQL

```
CREATE TRANSACTIONAL PIPELINE test_continuous_pipeline
    SOURCE S3
        ENDPOINT 'http://endpoint.ocient.com/'
        BUCKET 'cs_data'
        FILTER '*.csv'
    EXTRACT
        FORMAT csv
    INTO public.ad_sessions
SELECT
    TO_TIMESTAMP(CHAR($1), 'yyyy-MM-dd HH:mm:ss.SSSSSS', 'java') AS event_date_time
    CHAR($2)                    AS device_model,
    TINYINT($8)                 AS device_user_age,
    BOOLEAN($10)                AS device_ad_tracking_disabled,
    BINARY($11)                 AS device_mac,
    INT($14)                    AS ip_zip,
    FLOAT($19)                  AS ip_zip_latitude,
    DOUBLE($20)                 AS ip_zip_longitude,
    BIGINT($21)                 AS session_id,
    SMALLINT($32)               AS session_response_latency,
    DECIMAL($34, 10, 1)         AS session_transaction_revenue,
    CHAR($39)                   AS session_app_name
;
```

SOURCE Options

File-Based Source Options

Options that apply to both the `S3`, `FILESYSTEM`, and `HDFS` sources.

Option Key	Default	Data Type	Description
FILTER | FILTER_GLOB | FILTER_REGEX	None	STRING	The expression for filtering files in the directory or child directories to load. If you do not specify the `PREFIX` option, this pattern applies

Option Key	Default	Data Type	Description
			to the full path to the file, except for the Bucket for S3-compatible file sources. If you specify the `PREFIX` option, this pattern applies to the file path portion after the specified prefix value. Paths include a leading forward slash. For `FILESYSTEM` sources, this option is required. For `S3` sources, one of the `FILTER`, `FILTER_GLOB`, `FILTER_REGEX`, or `OBJECT_KEY` keys is required. Specify one of these options: * `FILTER_GLOB` or `FILTER` — Regular Unix® filename patterns. Supported options include: - `*` indicates unlimited wildcard characters except for the path character, `/`. - `**` indicates wildcard characters, including path separators. - `?` indicates a single wildcard character. * `FILTER_REGEX` — Regular expression patterns to filter files. Some common patterns include: - `.` matches any character. - `*` matches any 0 or more of the preceding character. - `+` matches a single character. - `[135]` matches any one character in the set. - `[1-5]` matches any one character in the range. - `(a|b)` matches a or b. **ℹ** When you use the FILTER or FILTER_GLOB options with a continuous data pipeline, the system supports only basic

Option Key	Default	Data Type	Description

			globbing, and extended, range globbing is not supported.
			
			FILTER_GLOB Examples:
			List all CSV files in subdirectories named `2024` that are in any subdirectory of the `trades` directory, which is in the root of the bucket.
			`FILTER_GLOB = '/trades/*/2024/*.csv'`
			List all CSV files in the bucket in all subdirectories.
			`FILTER_GLOB = '**/*.csv'`
			
			FILTER_REGEX Examples:
			List all `json.gz` files in the bucket that contain the name `orders` anywhere in the path.
			`FILTER_REGEX = '.*orders.*\.json\.gz'`
			List all files in the bucket in the root path `metrics` or `values` where the date string on the filename is 2000 to 2004.
			`FILTER_REGEX = '/(metrics|values)/.*200[0-4].*'`

PREFIX	`NULL`	STRING or ARRAY OF STRINGS	Optional. Specify a prefix within which to apply the filter. When you specify a list of prefixes, the system applies the filter to each one, and the results are unioned together. Prefixes must be paths to directories, meaning they end with a forward slash. For `FILESYSTEM` sources, paths should be absolute, meaning they begin with a forward slash. For `S3` sources, use

Option Key	Default	Data Type	Description

			prefixes to reduce the search space when listing objects within the specified `BUCKET`. For maximum performance, use the `PREFIX` option whenever possible, especially when combined with the `FILTER_REGEX` option.
			
			ℹ This option is not available for continuous data pipelines.
			
			Examples:
			List all CSV files in the `2024` subdirectory of `files`.
			`PREFIX '/files/2024/'`
			`FILTER '**/*.csv'`
			List all JSON files of subdirectory `2024/09/` in the `orders` directory.
			`PREFIX '/data/orders/2024/09/'`
			`FILTER_REGEX '.*/orders/.*json'`
			List all JSON files in multiple subdirectories.
			`PREFIX ['/data/orders/2024/09/', '/data/orders/2024/10/']`

COMPRESSION_METHOD	`NULL`	STRING	Optional. The method the Ocient System uses to decompress file data. The only supported option is `gzip`. To load uncompressed data, omit the COMPRESSION_METHOD option. ℹ The COMPRESSION_METHOD option is only applicable to file-based sources. Kafka pipelines automatically decompress data based on the Kafka topic configuration

Option Key	Default	Data Type	Description
SORT_BY	`filename`	STRING	Optional. The sort criteria for sorting the file list before the load. Supported options are: * `filename` — Sort files lexicographically by the filename. * `created` — Sort files based on the file created timestamp. * `modified` — Sort files based on the file modified timestamp. Sorting files such that the Ocient System sorts the data within the files according to the TimeKey® column in the target table can lead to better query performance. **i**This option is not available for continuous data pipelines.
SORT_DIRECTION	`ASC`	STRING	Optional. The sort direction, either `ASC` ascending or `DESC` descending, determines the sort order. If you specify this option, you must also specify the SORT_BY option. **i**This option is not available for continuous data pipelines.
SORT_REWRITE	`NULL`	STRING	Optional. This option is a sort comparator, allowing files to be renamed during the sort operation using capture groups from the filter regular expression. If you specify this option, you must also specify the `FILTER_REGEX` option. If you do not specify this

Option Key	Default	Data Type	Description
			option, the system uses the sorting specified by the SORT_BY option. ℹ This option is not available for continuous data pipelines. Example: `FILTER_REGEX '(\d+)_(\d+)_(\d+)-order.json' SORT_REWRITE '\3-\1-\2.json'` This specification renames the file from `5_6_1999-orders.json` to `1999_5_6.json` by switching the order of the date fields.
START_FILENAME	`NULL`	STRING	Optional. The filename string that is the lower bound to filter for files lexicographically for batch pipelines. (inclusive) Use the full path of the file, such as `'/dir/load_file.json'`. If the file is located in the top-most directory, start the path with a slash `/`, such as `'/load_file.json'`. You can use this option without the END_FILENAME option. If you specify the END_FILENAME option, the value of the START_FILENAME option must be smaller than the value for the END_FILENAME option lexicographically.
END_FILENAME	`NULL`	STRING	Optional. The filename string that is the upper bound to filter for files lexicographically for batch pipelines. (inclusive)

Option Key	Default	Data Type	Description
			Use the full path of the file, such as `'/dir/load_file.json'`. If the file is located in the top-most directory, start the path with a slash `/`, such as `'/load_file.json'`. You can use this option without the START_FILENAME option. If you specify the END_FILENAME option, the value of the START_FILENAME option must be smaller than the value for the END_FILENAME option lexicographically.
START_CREATED_TIMESTAMP	`NULL`	TIMESTAMP-formatted STRING	Optional. The ISO-8601-compliant date or date time that is used as the lower bound (inclusive) to filter files by created timestamp for batch pipelines. The time zone should match the file metadata. If you specify both, the value for the START_CREATED_TIMESTAMP option must be before the value for the END_CREATED_TIMESTAMP option. **i**This option is not available for continuous data pipelines.
END_CREATED_TIMESTAMP	`NULL`	TIMESTAMP-formatted STRING	Optional. The ISO-8601-compliant date or date time that is used as the lower bound (inclusive) to filter files by created timestamp for batch pipelines. The time zone should match the file metadata.

Option Key	Default	Data Type	Description
			If you specify both, the value for the START_CREATED_TIMESTAMP option must be before the value for the END_CREATED_TIMESTAMP option. **i**This option is not available for continuous data pipelines.
START_MODIFIED_TIMESTAMP	`NULL`	TIMESTAMP-formatted STRING	Optional. The ISO-8601-compliant date or date time that is used as the lower bound (inclusive) to filter files by modified timestamp for batch pipelines. The time zone should match the file metadata. If you specify both, the value for the START_MODIFIED_TIMESTAMI option must be before the value for the END_MODIFIED_TIMESTAMP option. **i**This option is not available for continuous data pipelines.
END_MODIFIED_TIMESTAMP	`NULL`	TIMESTAMP-formatted STRING	Optional. The ISO-8601-compliant date or date time that is used as the lower bound (inclusive) to filter files by modified timestamp for batch pipelines. The time zone should match the file metadata. If you specify both, the value for the START_MODIFIED_TIMESTAMI option must be before the

Option Key	Default	Data Type	Description
			value for the END_MODIFIED_TIMESTAMP option. **i**This option is not available for continuous data pipelines.

S3 Source Options

You can apply these options to data sources of the `SOURCE S3` type, which include S3 and S3-compatible services.

Option Key	Default	Data Type	Description
BUCKET	None	STRING	The name of the bucket in AWS S3.
OBJECT_KEY	`NULL`	STRING or ARRAY OF STRINGS	Optional. Specify an S3 object key(s) to load. You cannot use this option if you specify `PREFIX`, `FILTER`, `FILTER_GLOB`, or `FILTER_REGEX` keys are specified. If you do not specify one of the `FILTER`, `FILTER_GLOB`, or `FILTER_REGEX` keys, this option is required. Object keys do not include a leading forward slash. Specifying individual objects can be faster than filtering, since the Ocient System avoids listing all the files in a large directory. Object keys must not include any of the special characters: `*?{}[]` **i**This option is not available for continuous data pipelines.

Option Key	Default	Data Type	Description

			Examples:
			Load a single object from the designated bucket at the specified object key.
			`OBJECT_KEY 'order_data/jsonl/orders_20251101.jsonl'`
			Load a list of objects from the designated bucket at the specified object keys.
			`OBJECT_KEY ['order_data/jsonl/orders_20251101.jsonl', 'order_data/jsonl/orders_20251201.jsonl']`

ACCESS_KEY_ID	`''`	STRING	Optional. The access key identification for AWS credentials. If you specify this option, you must also specify the `SECRET_ACCESS_KEY` option. The Ocient System uses anonymous credentials when you specify an empty value for this option.
SECRET_ACCESS_KEY	`''`	STRING	Optional. The secret key for AWS credentials. If you specify this option, you must also specify the ACCESS_KEY_ID option. The Ocient System uses anonymous credentials when you specify an empty value for this option.
SESSION_TOKEN	`NULL`	STRING	Optional. Temporary credentials can be made with a combination with the existing access key identifier and secret key, and an additional session token. You must specify the ACCESS_KEY_ID and SECRET_ACCESS_KEY

Option Key	Default	Data Type	Description
			options when you use this option.
ROLE_ARN	`NULL`	STRING	Optional. The Amazon® Resource Name (ARN) of the role to retrieve temporary credentials. The STS client used to retrieve the temporary credentials specifies the **region** loaded from `DefaultAwsRegionProviderChain` and **credentials** loaded from `DefaultCredentialsProvider`. You cannot use this option with the ACCESS_KEY_ID and SECRET_ACCESS_KEY options.
ASSUME_ROLE_CONFIG	`{}`	STRING	Optional. Additional configuration used to configure the `StsAssumeRoleCredentialsProvider` or `AssumeRoleRequest` used to retrieve and refresh temporary credentials with the ROLE_ARN option. This option is a JSON-formatted string that you can only specify at the same time as the ROLE_ARN option. The supported configuration options are: `"asyncCredentialUpdateEnabled"`, `"prefetchTimeSeconds"`, `"staleTimeSeconds"` for `StsAssumeRoleCredentialsProvider` and `"durationSeconds"`, `"externalId"`, `"policy"`, `"roleSessionName"`, `"serialNumber"`, `"tokenCode"` for `AssumeRoleRequest`.

Option Key	Default	Data Type	Description
REGION	`'us-east-1'`	STRING	Optional. The region that the Ocient System uses for AWS access. If you specify the `ENDPOINT` option, the system ignores this option.
ENDPOINT	`NULL`	STRING	Optional. The endpoint URI for the S3-compatible service API. (e.g., `https://s3.us-east-2.amazonaws.com`) When unspecified, this option defaults to `https://s3.REGION.amazonaws.com`. If you provide this option, the Ocient System ignores the `REGION` option.
ENABLE_PATH_STYLE_ACCESS	`NULL`	BOOLEAN	Optional. Whether to use path-style access, where the path includes the bucket name in the URL. For example, `https://s3.us-east-1.amazonaws.com/bucket-name/key-name`. The `false` value denotes virtual-hosted style access, where the URL contains the bucket name as part of the domain name. For example, `https://bucket-name.s3.us-east-1.amazonaws.com/key-name`. When unspecified, this option defaults to `true` if the `ENDPOINT` option is specified and `false` otherwise.
HEADERS	`NULL`	STRING	Optional.

Option Key	Default	Data Type	Description
			The headers to send with every request. This option is a JSON-formatted string. Represent the chosen header names as keys with corresponding values as scalars or lists of scalars. The system converts scalars that are not strings to strings. During the load, the system maps each element in a list to the header name represented by the corresponding key. Examples: `HEADERS '{"x-amz-request-payer": "requester"}'` returns this header for each request: `x-amz-request-payer: requester`. `HEADERS '{"header-name": ["list", "of", "values"]}'` returns this header for each request: `header-name: list, of, values`.
MAX_CONCURRENCY	50	INTEGER	Optional. Determines the number of parallel connections the Ocient System uses to communicate with the AWS S3 service. ⚠ This option does not require modification in most cases. Contact Ocient Support to modify these values.
READ_TIMEOUT	0 (unlimited timeout)	TIME INTERVAL	Optional. The time until a read operation times out. ⚠ This option does not require modification in most cases. Contact Ocient

Option Key	Default	Data Type	Description
			Support to modify these values.
REQUEST_DEPTH	500	INTEGER	Optional. The upper boundary of requests that the Ocient System handles concurrently. ⚠ This option does not require modification in most cases. Contact Ocient Support to modify these values.
REQUEST_RETRIES	10	INTEGER	Optional. Number of times the AWS SDK retries failing requests before the Ocient System throws an error. ⚠ This option does not require modification in most cases. Contact Ocient Support to modify these values.

S3 Credentials Hierarchy

The S3 source configuration supports this hierarchy to obtain S3 credentials:

- Level 1: Data pipeline configuration (set by using the `ACCESS_KEY_ID` and `SECRET_ACCESS_KEY` options)
- Level 2: **AWS SDK Default Credential Provider Chain** (set by using the instructions provided in the web page)
- Level 3: Anonymous access (set by default)

If the Ocient System does not obtain the credentials at a level, the system tries a lower level.

> ⓘ You can choose the level to store the credentials, where Level 1 is the highest. Higher levels take precedence for credential storage.

FILESYSTEM Source Options

No options exist specific to the file system source (`SOURCE FILESYSTEM`) except for general **file-based source options**.

> When you load data using `SOURCE FILESYSTEM` , the files must be addressable from all of your Loader Nodes. The Ocient System uses the specified path in the pipeline PREFIX and FILTER options to select the files to load.
>
> A shared view of the files you want to load must be available to all Loader Nodes involved in a pipeline. For example, you can use a Network File System (NFS) mount available to all Loader Nodes at a common path on each node.

Example

This example `CREATE PIPELINE` SQL statement snippet contains a `FILESYSTEM` source and filters to all CSV files in the `/tmp/sample-data/` directory on each of the Loader Nodes.

SQL

```
CREATE PIPELINE ...
    SOURCE FILESYSTEM
        PREFIX '/tmp/sample-data/'
        FILTER '**/*.csv'
    EXTRACT
        FORMAT delimited
INTO public.orders
SELECT
    $1 as username,
    $2 as subtotal,
    ...
```

HDFS Source Options

You can specify these options for Apache® Hadoop® Distributed File System (HDFS) data sources (`SOURCE HDFS`).

For advanced options, see **Data Pipeline Load of JSON Data from HDFS**.

Option Key	Default	Data Type	Description
ENDPOINT	None	STRING	Required. The hostname and port number of the namenode server. For example, `'my-hdfs-server:1234'`. The namenode server manages the file system and regulates access to data.
CONFIG	None	JSON-formatted STRING	Optional. The JSON string that contains HDFS **configuration properties** to use during the load. The properties are primarily for authentication. You can also use these properties to configure advanced properties such as connection timeouts, parallelism, and network chunk sizes. **Example:** `CONFIG '{"dfs.client.use.datanode.hostname": true, "dfs.client.socket-timeout": 300000}'`

KAFKA Source Options

You can apply these options to Kafka data sources (`SOURCE KAFKA`).

> For compression, you do not need to specify a compression option in Kafka-based pipelines, because the Ocient System handles the compression type automatically. Records produced to the Kafka broker with a `compression.type` setting or with the `compression.type` set on the topic automatically decompress when the loading process consumes the records. The loading process uses built-in headers in Kafka to determine the required decompression during extraction.

Option Key	Default	Data Type	Description
BOOTSTRAP_SERVERS	None	STRING	A comma-delimited list of `IP:port` pairs that contain the IP addresses and the associated port numbers of the Kafka Brokers. You can also use a

Option Key	Default	Data Type	Description
			hostname instead of the IP address. **Example:** `BOOTSTRAP_SERVERS = '198.51.100.1:9092,198.51.100.2:9092'`
TOPIC	None	STRING	The name of the Kafka topic indicates where to consume records.
WRITE_OFFSETS	`true`	BOOLEAN	Optional. Indicates whether the Kafka consumer should write its durably-made record offsets to the Kafka Broker.
CONFIG	`'{ "group.id": "<systemId>__<databaseName>__<pipelineName>" }'`	JSON-formatted STRING	Optional. The **consumer configuration** that the Kafka consumers should use. Frequently specified configurations include `group.id` and `security.protocol`. These configurations are fixed and cannot be overridden: `{ "enable.auto.commit": false, "key.deserializer": "org.apache.kafka.common.serialization.ByteArrayDeserializer", "value.deserializer": "org.apache.kafka.common.serialization.ByteArrayDeserializer" }`
AUTO_OFFSET_RESET	`'latest'`	STRING	Optional. Determine which action to take for the Kafka configuration when there is no initial offset in the offset store or the specified offset is out of range. Supported values are:

Option Key	Default	Data Type	Description
			`'earliest'` — Automatically reset the offset value to the smallest value. `'latest'` — Automatically reset the offset value to the largest value.
GROUP_ID	`'<systemId>__<databaseName>__<pipelineName>'`	STRING	Optional. Client group identifier for the Kafka source configuration.

For the consumer configuration, to create a secure connection from the Kafka consumer to a Kafka Broker, set the `"security.protocol"` key along with any SSL or SASL keys.

If a certificate file is required, you must add it to the truststore used by the JVM® on all Loader Nodes. The truststore path must be identical on all Loader Nodes. The Kafka configuration can reference this truststore path.

If you specify the `ssl.certificate.location` or `ssl.ca.location` consumer configuration, you must specify both of these configurations. Otherwise, the system throws an error. For example: `CONFIG '{"auto.offset.reset": "earliest","ssl.certificate.location":"/etc/blab/file1.txt","ssl.ca.location":"/etc/blab/file2.txt"}'`

Continuous File Loading Source Options

Specify these options for data pipelines that use `FILESYSTEM` or `S3` sources with the `CONTINUOUS` mode. For a Kafka source, do not use these options.

General File Monitor Options

Option Key	Default	Data Type	Description
MONITOR	None	STRING	The type of monitor. Valid values are: `sqs` or `kafka`.
POLLING_INTERVAL	10 SECONDS	TIME INTERVAL	Optional. Unit of time to consume the event topic. Valid value range: 10 to 120 seconds.

Option Key	Default	Data Type	Description
MAX_FILES	100	INTEGER	Optional. In the `BATCH` syntax, the maximum number of pending files to collect before starting a batch. Valid value range: 10 to 2000.
TIMEOUT	1 MINUTE	TIME INTERVAL	Optional. In the `BATCH` syntax, the time to wait before starting a batch. Valid value range: 10 seconds to 30 minutes.
LOOKBACK	1 DAY	TIME INTERVAL	Optional. In the `BATCH` syntax, the unit of time to look back for file deduplication. Valid value range: 0 to 48 hours.

SQS Monitor Options

Use these options when you use `MONITOR sqs` for Amazon® SQS.

Option Key	Default	Data Type	Description
QUEUE_URL	None	STRING	The URL of the target queue. For example: `http://localhost:32769/000000000000/queue1`
REGION	`NULL`	STRING	Optional. The region for the SQS operation for signing requests. When unspecified, the region is inferred from `QUEUE_URL` by looking for `sqs.REGION` or `sqs-fips.REGION`. If the region cannot be inferred, it defaults to `us-east-1`.
ENDPOINT	`NULL`	STRING	Optional. The endpoint URL for the client. For example: `http://localhost:32769` When unspecified, defaults to

Option Key	Default	Data Type	Description
			`https://sqs[-fips].REGION.amazonaws.com` if `REGION` was inferred or specified. Otherwise, this option is required.
ACCESS_KEY_ID	`''`	STRING	Optional. Access key identifier for SQS authentication. If you specify this option, you must also specify the `SECRET_ACCESS_KEY` option. The Ocient System uses anonymous credentials when you specify an empty value for this option.
SECRET_ACCESS_KEY	`''`	STRING	Optional. The secret access key for SQS authentication.

Kafka Monitor Options

Use these options when you use `MONITOR kafka` for Kafka.

Option Key	Default	Data Type	Description
BOOTSTRAP_SERVERS	None	STRING	The bootstrap servers for the Kafka consumer configuration. The string contains a list of brokers as a comma-separated list of the broker hostnames or broker hostnames and port number combinations, in the format `hostname:port`.
TOPIC	None	STRING	The name of the Kafka topic indicates where to consume records.
AUTO_OFFSET_RESET	`'latest'`	STRING	Optional. Action to take for the Kafka configuration when there is no initial offset in the offset store or the chosen offset is out of range: `'earliest'` — Automatically reset the offset to the smallest

Option Key	Default	Data Type	Description
			offset. `'latest'` — Automatically reset the offset to the largest offset.
GROUP_ID	`'<systemId>__<dataBaseName>__<pipelineName>_monitor'`	STRING	Optional. Client group identifier for the Kafka configuration.
CONFIG	`'{ "enable.auto.commit": false, "group.id": "<systemId>__<dataBaseName>__<pipelineName>_monitor", "max.poll.interval.ms": "600000", "heartbeat.interval.ms": "6000", "session.timeout.ms": "600000" }'`	JSON-formatted STRING	Optional. The **consumer configuration** that the Kafka consumers should use. This option is a JSON-formatted string. Certain values within this configuration are fixed, whereas the Ocient System provides other values with a default value that you can modify.
MAX_MESSAGES	100	INTEGER	Optional. In the `CONSUMER` syntax, the maximum number of messages to poll each time.
TIMEOUT	1000 MILLISECONDS	TIME INTERVAL	Optional. In the `CONSUMER` syntax, the operation timeout that controls how long the consume request waits for the response. The valid range is: 0 milliseconds to 1 hour.
FILENAME	`$"Records"[1].s3.object.key` corresponds to a `s3:ObjectCreated:Put` event.	SELECTOR	Optional. In the `MESSAGE` syntax, defines the JSON selector for the filename of an incoming message in a monitor. The default value works for `s3:ObjectCreated:*` events. If you override this default configuration, you can parse custom JSON messages. The format must still adhere to S3 standards. For details, see **Amazon Simple Storage Service Event message structure**. The message must be

Option Key	Default	Data Type	Description
			in the JSON format. If you specify this option, then you must specify the other two options in the `MESSAGE` syntax.
TIMESTAMP	`$"Records"[1]."eventTime"` corresponds to a `s3:ObjectCreated:Put` event.	SELECTOR	Optional. In the `MESSAGE` syntax, defines the JSON selector for the last modification timestamp of the file of an incoming message in a monitor. The default value works for `s3:ObjectCreated:*` events. If you override this default configuration, you can parse custom JSON messages. The format must still adhere to S3 standards. For details, see **Amazon Simple Storage Service Event message structure**. The message must be in the JSON format. If you specify this option, then you must specify the other two options in the `MESSAGE` syntax.
SIZE	`$"Records"[1].s3.object.size` corresponds to a `s3:ObjectCreated:Put` event.	SELECTOR	Optional. In the `MESSAGE` syntax, defines the JSON selector for the file size of an incoming message in a monitor. The default value works for `s3:ObjectCreated:*` events. If you override this default configuration, you can parse custom JSON messages. The format must still adhere to S3 standards. For details, see **Amazon Simple Storage Service Event message structure**. The message must be in the JSON format. If you specify this option, then you must specify the other two options in the `MESSAGE` syntax.

The same Kafka `CONFIG` option override considerations apply. For details, see **KAFKA Source Options**.

LOOKUP Options

You can optionally look up data in a table from an external database. You can include this table in the `SELECT` statement of the `CREATE PIPELINE` SQL statement and perform join operations on its columns. To use the `LOOKUP` keyword, specify the external source name `lookup_source`.

You can look up data in multiple external databases. In this case, use the `LOOKUP` keyword with the source name and corresponding options for each database.

You must provide the appropriate JDBC JAR file for the JDBC connection to external databases. For tables in the Ocient System, this JAR file is not needed.

> ⚠ The Ocient System executes the JDBC code as trusted code. Only provide trusted JAR files to the loading process. The `streamloader.extractorEngineParameters.configurationOption.engine.external.jdbc.jarRootDirectory` configuration parameter specifies the location of the JAR file. For details, see **Configuration Settings for Data Pipelines**.

Option Key	Default	Data Type	Description
CONNECTION_TYPE	None	STRING	The type of connection. Only `'jdbc'` is supported.
CONNECTION_STRING	None	STRING	The string for connection to the external database. For example: `'jdbc:sqlite:/path/to/file'`
LOOKUP_SCHEMA	None	STRING	The schema of the table in the external database.
LOOKUP_TABLE	None	STRING	The name of the table in the external database.
REFRESH_PERIOD	30	TIME INTERVAL	Optional. The time between lookup cache refreshes.
CONFIG	`NULL`	JSON-formatted STRING	Optional. The JSON string that contains additional configuration options. For example: `'{"force_refresh_if_null": false}'` Supported JSON keys are:

Option Key	Default	Data Type	Description
			`force_refresh_if_null`, specified as a Boolean that defaults to `true`. When you set this value to `true`, the system triggers a lookup cache refresh for each value missing from the lookup table (to account for a concurrent load). This cache refresh can have a noticeable performance impact. As such, if you expect that some values cannot result in successful lookups, then set this key to the `false` value. `load_timeout_seconds`, specified as a long integer with no default value. This option is the timeout, in seconds, for a lookup cache refresh. `driver_class`, specified as a string with no default value. This option is the fully qualified JDBC driver class name. If you specify this option, the system throws an error when the system cannot find this driver or the driver cannot load into the process. If the JAR file for the connection is a JDBC 4 file, the system automatically loads the driver class. Otherwise, you must specify the fully qualified class name (from `Class::getName` in Java®) using this option, such as `'{"driver_class": "org.test.example.Driver"}'`.

EXTRACT Options

General Extract Options

You can specify these options on any of the allowed `FORMAT` types.

Option Key	Default	Data Type	Description
FORMAT	None	STRING	Specifies the format of the to load. Supported values

Option Key	Default	Data Type	Description

			`delimited`
			`csv`
			`json`
			`binary`
			`asn.1`
			`parquet`
			`xml`

CHARSET_NAME	Binary format:
	`ibm1047`
	All other formats:
	`utf-8`
		STRING	Optional.
			
			Specifies the character se decoding data from the sc records into character dat this character set when yc load data into VARCHAR columns or when you app `CHAR` transformation functions.
			Defaults to `utf-8` for all formats except BINARY, w defaults to `ibm1047`.
			You can configure the defa value for `BINARY` formatt data using a SQL stateme such as:
			`ALTER SYSTEM ALTER CONFI`
			`'sql.pipelineParameters.`
			`t.binary.defaultCharset'`
			`'ibm1047'`

COLUMN_DEFAULT_IF_NULL	`false`	BOOLEAN	Optional. Specifies whether pipeline should load the column de value when the result of a series of transforms is NU If you set this option to `f` (the default), the pipeline NULL values into columns If you set this option to `t` the pipeline loads the defi default value of the colum when the result of the execution of the transform on the column is NULL.
NULL_STRINGS	Delimited or CSV format: `['null', 'NULL']`	ARRAY OF STRINGS	Optional. Specifies string values tha should represent a NULL v

Option Key	Default	Data Type	Description
	JSON format: `[]`		when extracted from the s records. Use this in `csv`, `delimit` and `json` formats to con specific values to NULL ins of requiring individual trar function calls to `NULL_IF` those values. This option applies to sou data immediately after extraction. If the result of transformations is one of t strings, you must use `NUL` to transform to NULL with specified string values.
TRIM_WHITESPACE	`false`	BOOLEAN	Optional. Specifies whether to trim whitespace from the begi and end of each string fie during extraction. In this c whitespace is a space, tab carriage return, or linefee character. If the trimmed field value matches one of the values specified by the `NULL_STR` option, the system conver value in the field to `NULL` you set this value to `true` the `EMPTY_FIELD_AS_NULL` option to `true`, the syste converts any field that co only whitespace to `NULL` This option does not trim of strings. To trim elements in arrays can use the `TRIM` transfo function or the `TRANSFORM` function with a Lambda expression that uses the function.
VALIDATE_CHARACTERS	`false`	BOOLEAN	Optional. Specifies whether to dete invalid characters based c

Option Key	Default	Data Type	Description
			encoding type during extraction. If the system detects invalid characters system throws a file-level for file loads or a record-le error for Kafka loads. This option is not supporte ASN.1, Apache® Parquet™ binary formats.
REPLACE_INVALID_CHARACTERS	`false`	BOOLEAN	Optional. Specifies whether to repla invalid characters based c encoding type with the replacement character `U+FFFD`. This option is not supporte ASN.1, Parquet, and binar formats.
REPLACEMENT_CHARACTER	`�`	STRING	Optional. Specifies the character th indicates invalid data whe set the REPLACE_INVALID_CHARA option to `true`. The defa value is the replacement character `U+FFFD`. This option is not supporte ASN.1, Parquet, and binar formats.

For ASN.1, Apache® Avro™, and Parquet extract options, the SCHEMA option is a `MAP` type with a `NULL` default value. This option is optionally specified depending on the format. SCHEMA specifies the syntax of options for schema retrieval.

ASN.1 Extract Options

You can specify these options for ASN.1 data record extraction (`FORMAT ASN.1`).

For details about ASN.1-formatted data, see **Load ASN.1 Data**.

Option Key	Default	Data Type	Description
URL	None	STRING	In the `SCHEMA` syntax, specify the URL to the ASN.1 schema file in the `.asn` format.
RECORD_TYPE	None	STRING	In the `SCHEMA` syntax, specify the fully qualified name of the record type to parse. The name is the root-level ASN.1 type to extract from each DER-encoded record. (e.g., `Example.PersonnelRecord`)

Avro Extract Options

You can specify these options for Avro data record extraction. The `SCHEMA` options are all optional for object container files (OCF).

For Kafka loads, the schema configuration must include either the `URL` option or the `INLINE` option, but not both.

For file-based loads, the schema configuration can include:

- Either the `INLINE` option or the `INFER_FROM` option, but not both of these options
- Neither the `INLINE` nor the `INFER_FROM` options

For details about loading Avro-formatted data, see **Load Avro Data**.

Option Key	Default	Data Type	Description
URL	None	STRING	The URL of the schema registry.
CONFIG	None	STRING	The configuration string that includes authentication for accessing the schema registry.
SUBJECT	`<topic_name>-value`, where `<topic_name>` is the name of the source Kafka topic	STRING	The custom subject for the schema registry.

Option Key	Default	Data Type	Description
SCHEMA_REGISTRY_ID_LOCATION	`header_or_value`	STRING	For Kafka loads, the location to store the schema identifier for the schema registry. Supported values are: `'header_or_value'` — Specifies that the system searches for the schema Universally Unique IDentifier (UUID) in the `__value_schema_id` Kafka header. If the system finds the UUID, the system uses that identifier. Otherwise, the system derives the identifier from the starting bytes of the Kafka message value. `'value'` — Specifies that the system derives a schema identifier or UUID from the starting bytes of the Kafka message value. `'none'` — For inline schemas, specifies that there is no embedded schema identifier to skip.
INLINE	None	STRING	The JSON specification of the schema declaration as a string. Specify either a JSON string or object.
INFER_FROM	None	STRING	For file-based loads, specifies that the system infers the schema from files. Supported value is: `'sample_file'` — Use a sample file.

BINARY Extract Options

You can apply these options to binary data record extraction (`FORMAT BINARY`).

For details about BINARY-formatted data, see **Load Binary Data**.

> The general option `CHARSET_NAME` has a different default value for `FORMAT BINARY` .

© 2026 Ocient, Inc.

Option Key	Default	Data Type	Description
RECORD_LENGTH	None	INTEGER	Specifies the fixed size in bytes of each record in the source data. The Ocient System splits the binary data into binary chunks according to this length value and processes them individually.
ENDIANNESS	`big`	STRING	Optional. Specifies the endianness used to interpret multi-byte sequences in various transforms of binary data. Accepted values are `'big'` and `'little'`.
AUTO_TRIM_PADDING	`true`	BOOL	Optional. Determines if padding characters should be trimmed after decoding the binary data into string data. If you set this option to `TRUE`, the Ocient System trims all instances of the `PADDING_CHARACTER` value from the end of a string after the system decodes the string from `BINARY` type.
PADDING_CHARACTER	(space)	STRING	Optional. The padding character from the string after the system decodes the string from `BINARY` type. You can change the default padding character for `BINARY` formatted data using a SQL statement such as: `ALTER SYSTEM ALTER CONFIG SET 'sql.pipelineParameters.extract.binaryFormat.defaultPaddingCharacter' = '*'`

The Ocient System trims the default padding character of a space from the end of the text data in binary data.

Delimited and CSV Extract Options

You can specify these options for delimited and CSV format data record extraction (`FORMAT DELIMITED` or `FORMAT CSV` data formats, which are aliases).

For details about working with delimited and CSV data, see **Load Delimited and CSV Data**.

Option Key	Default	Data Type	Description
FIELD_DELIMITER	`,`	STRING or ARRAY OF STRINGS	Optional. Specifies a character or list of possible characters for delimiting fields. The default value sets the field delimiter to only a comma. The value must be one byte. The Ocient System automatically interprets the values you specify as C-style escaped strings. This means you do not need to specify an escape string (`e'some value'`) is not required to specify control characters. This differs from the default string behavior in Ocient. For details, see **String Literals and Escape Sequences**. `FIELD_DELIMITER` and `FIELD_DELIMITERS` are aliases. **Examples:** Use a tab character: `FIELD_DELIMITER = '\t'` Use a pipe character: `FIELD_DELIMITER = '|'` Use either a pipe or a comma character: `FIELD_DELIMITER = [',', '|']`

Option Key	Default	Data Type	Description

RECORD_DELIMITER	`['\r\n', '\n']`	STRING
		or
		ARRAY OF
		STRINGS
			Optional.
			
			Specifies the string or an array of strings for delimiting records. The file is split into individual records using this character during processing. Common values include `'\r\n'` and `'\n'`. The value must be one or two bytes.
			The Ocient System automatically interprets the values you specify as C-style escaped strings. An escape string (`e'some value'`) is not required to specify control characters. This differs from the default strings behavior in Ocient.
			The system chooses the first specified delimiter and uses that delimiter for the rest of the data. Data with mixed delimiters is not supported.
			For details, see **String Literals and Escape Sequences**.
			`RECORD_DELIMITER` and `RECORD_DELIMITERS` are aliases.
			Examples:
			Use a linefeed character:
			`RECORD_DELIMITER = '\n'`
			Use a carriage return and linefeed character sequence:
			`RECORD_DELIMITER = '\r\n'`

NUM_HEADER_LINES	`0`	INTEGER	Optional. Specifies the number of

Option Key	Default	Data Type	Description
			header lines, typically 0 or 1. The Ocient System skips this number of lines and does not load them as data during file processing. Use this option when your data includes a row of header values.
NUM_FOOTER_LINES	`0`	INTEGER	Optional. Specifies the number of footer lines, typically 0 or 1. The Ocient System skips this number of lines starting from the end of the file and does not load them as data during file processing. Use this option when your data includes a row of footer values.
OPEN_ARRAY	`[`	STRING	Optional. Specifies the character that indicates the start of an array in a `csv` or `delimited` field. Use this option to parse array data types. Specify the `CLOSE_ARRAY` option also when using this option. Set this option to NULL or `''` to turn off the detection of these control characters. If you set this option to either of these characters, the system also turns off the detection of these characters for the `CLOSE_ARRAY` option. **Example:** Convert source data

Option Key	Default	Data Type	Description

			such as `val1,"[1,2,3]",val2` to an array when referenced as `$2[]`.
			`OPEN_ARRAY '['`
			`CLOSE_ARRAY ']'`
			`ARRAY_ELEMENT_DELIMITER ','`

CLOSE_ARRAY	`]`	STRING	Optional.
			
			Specifies the character that indicates the end of an array in a `csv` or `delimited` field. Use this option to parse array data types. Specify the `OPEN_ARRAY` option also when using this option.
			Set this option to NULL or `''` to turn off the detection of these control characters. If you set this option to either of these characters, the system also turns off the detection of these characters for the `OPEN_ARRAY` option.
			Example:
			Convert source data such as `val1,"{1,2,3}",val2` to an array when referenced as `$2[]`.
			`CLOSE_ARRAY '}'`
			`OPEN_ARRAY '{'`
			`ARRAY_ELEMENT_DELIMITER ','`

ARRAY_ELEMENT_DELIMITER	`,`	STRING	Optional. Specifies the character that separates values in an array within a `csv` or `delimited` field. Use this option to parse array data types.

Option Key	Default	Data Type	Description
			Set this option to NULL or `''` to turn off the detection of these control characters. **Example:** Convert source data such as `val1,"[1;2;3]",val2` to an array when referenced as `$2[]`. `ARRAY_ELEMENT_DELIMITER = ';'`
OPEN_OBJECT	`{`	STRING	Optional. Specifies the character that indicates the start of a tuple in a field. Use this option to parse tuple data types. Specify the `CLOSE_OBJECT` option also when using this option. Set this option to NULL or `''` to turn off the detection of these control characters. If you set this option to either of these characters, the system also turns off the detection of these characters for the `CLOSE_OBJECT` option.
CLOSE_OBJECT	`}`	STRING	Optional. Specifies the character that indicates the end of a tuple in a field. Use this option to parse tuple data types. Specify the `OPEN_OBJECT` option also when using this option. Set this option to NULL or `''` to turn off the detection of these control characters. If

Option Key	Default	Data Type	Description
			you set this option to either of these characters, the system also turns off the detection of these characters for the `OPEN_OBJECT` option.
EMPTY_FIELD_AS_NULL	`true`	BOOLEAN	Optional. Specifies whether the Ocient System should extract an empty source field as NULL or a missing value. When this option is set to `true`, the Ocient System treats empty fields as NULL. Otherwise, the system treats fields as a missing value. For string-type fields, a missing value is equivalent to an empty string. Define an empty field as two consecutive delimiters (e.g., the second field is empty in `abc,,xyz`). If a field is explicitly an empty string as indicated by quote characters, the Ocient System treats the field as an empty string, not an empty field (e.g., the second field is an empty string in `abc,"",xyz`). The `CHAR($1)` transformation function handles both NULLs and empty strings and passes them through. ⚠ Beware that the `NULL_IF($1, '')` transformation function directly loads a NULL for both NULLs *and *missing values. This

Option Key	Default	Data Type	Description
			transformation can override the behavior of `EMPTY_FIELD_AS_NULL`.

FIELD_OPTIONALLY_ENCLOSED_BY	`"`	STRING	Optional.
			
			Also known as the “quote character,” this option specifies the character for optionally enclosing fields. Fields enclosed by this character can include delimiters, the enclosure character, or the escape character. Set this option to NULL or `''` to turn off the detection of these control characters.
			Examples:
			Use a double quote character:
			`FIELD_OPTIONALLY_ENCLOSED_BY = '"'`
			Use a single quote character:
			`FIELD_OPTIONALLY_ENCLOSED_BY = ''''`

ESCAPE_CHAR	`"`	STRING	Optional. Specifies the escape character within fields enclosed by the `FIELD_OPTIONALLY_ENCLOSED_BY` option. Use this option to escape the enclosure character or escape character. Set this option to NULL or `''` to turn off the detection of these control characters. **Examples:** Use a double quote as the escape character. `ESCAPE_CHAR = '"'` Use a single quote as the escape character.

Option Key	Default	Data Type	Description
			When you specify the escape character, you often have to use an escape sequence. This action follows standard SQL rules. `ESCAPE_CHAR = ''''`
SKIP_EMPTY_LINES	`false`	BOOLEAN	Optional. Specifies whether or not to skip empty lines.
COMMENT_CHAR	`NULL`	STRING	Optional. Specifies the character used to comment out a record in the source file. The load skips records where the first character of a record is equal to this character. Set this option to NULL or `''` to turn off the detection of these control characters. Example: `COMMENT_CHAR '#'`
HEADERS	`NULL`	ARRAY OF STRINGS	Optional. Specifies the header labels associated with each column in a delimited file. This array of values corresponds to the columns in order from left to right. Use these labels in the `CREATE PIPELINE SELECT` SQL statement to refer to column values. For example, if you specify `HEADERS ['col1', 'col2', col3']`, you can refer to the first column as `$"col1"` instead of `$1` in the `SELECT` statement.

Option Key	Default	Data Type	Description
STRIP_ARRAY_ELEMENT_QUOTES	`false`	BOOLEAN	Optional. When you set this option to `true`, the system removes quote characters from array elements, so that, for example, `["str"]` becomes `[str]`.
STRIP_FIELD_QUOTES	`true`	BOOLEAN	Optional. When you set this option to `true`, the system removes quote characters from the string, so that, for example, `"str"` becomes `str`.
TRIM_ARRAY_ELEMENTS	`false`	BOOLEAN	Optional. When you set this option to `true`, the system removes whitespace characters from the array elements, so that, for example, `" str"` becomes `"str"`.

JSON Extract Options

No options exist for JSON data record extraction (`FORMAT JSON`).

For details about JSON-formatted data, see **Load JSON Data**.

PARQUET Extract Options

You can specify these options for Parquet data record extraction (`FORMAT ``PARQUET`).

For details about Parquet-formatted data, see **Load Parquet Data**.

⚠ When you use the `FORMAT PARQUET` option with an AWS S3 Source, the `ENDPOINT` option is required.

Option Key	Default	Data Type	Description
SCHEMA INFER_FROM	`sample_file`	STRING	In the `SCHEMA` syntax, specify how to infer the Parquet schema. The supported value is: `sample_file` — Infer from a random file. **Example:** `SCHEMA (INFER_FROM sample_file)`

XML Extract Options

No options exist for the XML format extraction (`FORMAT XML`).

For details about XML-formatted data, see **Load XML Data**.

Bad Data Targets

Bad data represents records that the Ocient System could not load due to errors in the transformations or invalid data in the source records. You can provide options for a bad data target that the Ocient System uses during pipeline execution to capture the records that are not loaded. The original bytes that the pipeline tried to load are captured in the bad data target along with the metadata about the error, such as the error message or source.

Kafka is the only supported bad data target.

Kafka Bad Data Target

When you use Kafka as a bad data target, the Ocient System produces the original bytes of the source record into the Kafka topic of your choice. The Ocient System includes the metadata about the record in the header of the record as it is sent to Kafka. You can configure the Kafka topic on your Kafka Brokers using the retention and partition settings of your choice.

⚠ In the event that the Kafka Broker is unreachable when the Ocient System attempts to produce a bad data record to the bad data target, the system logs an error on the Loader Node, and the pipeline continues.

Example

This example `CREATE PIPELINE` SQL statement snippet contains a bad data target definition using the `BAD_DATA_TARGET` option.

SQL

```
CREATE PIPELINE ...
    BAD_DATA_TARGET
        KAFKA
        TOPIC 'orders_errors'
        BOOTSTRAP_SERVERS '111.11.111.1:9092,111.11.111.2:9092'
        CONFIG '{"compression.type": "gzip"}'
    SOURCE
        ...
    EXTRACT
        ...
INTO public.orders
SELECT
    $order.billing.name as username,
    $order.subtotal as subtotal,
    ...
```

Kafka Bad Data Target Options

Option Key	Default	Data Type	Description
BOOTSTRAP_SERVERS	None	STRING	A comma-delimited list of `IP:port` pairs that contain the IP addresses of the Kafka Brokers and the associated port number. **Example:** `BOOTSTRAP_SERVERS = '111.11.111.1:9092,111.11.111.2:9092'`
TOPIC	None	STRING	The name of the Kafka topic where the Ocient System should produce bad data records.
CONFIG	`'{ "compression.type" : "none" }'`	JSON-formatted STRING	Optional. The **producer configuration** that the Kafka producer should use.

Advanced Pipeline Tuning Options

You can use pipeline tuning options to control the parallelism or batching dynamics of your pipelines. This tuning can throttle the resources used on a pipeline or increase parallel processing across Loader Nodes. These options are advanced settings that might require a detailed understanding of the underlying mechanics of the loading infrastructure in the Ocient System to employ. Due to the inherent nature of each source type, the behavior of these options can differ between file-based and Kafka-based loads. All these options are optional.

| **Option Key** | **Default** | **Data Type** | **Description** | | --- | --- | --- | --- | | CORES | The maximum number of CPU cores available on each Loader Node. | INTEGER | Maximum number of processing threads that the Ocient System uses during execution on each Loader Node. The system creates this number of threads on each Loader Node.
The Ocient System automatically determines the default value by finding the number of cores of a Loader Node. You can use this option for performance tuning.
The calculation for maximum parallelism of a pipeline is: `number_of_loaders * CORES`.
About Kafka Partitions and Parallelism
For Kafka Loads, this option determines the number of Kafka Consumers created on each Loader Node.
For Kafka Pipelines, the recommendation is that `number_of_loaders * CORES` equals the number of Kafka topic partitions.
If this number exceeds the number of Kafka topic partitions, the work might spread unevenly across Loader Nodes.
If this number is less than the number of Kafka topic partitions, some Kafka Consumers might receive uneven amounts of work. In this case, use a value for `number_of_loaders * CORES` that is an even divisor of the number of Kafka topic partitions to avoid a skew in the rates of processing across partitions. | | PARTITIONS | Equal to the value of CORES. | INTEGER | Specifies the number of partitions over which to split the file list. Not applicable to Kafka loads.
The Ocient System automatically sets a default value based on the configured value for the `CORES` option. You can use this option for performance tuning.
The number of partitions determines how many buckets of work the Ocient System generates for each batch of files processed on a Loader Node. The pipeline processes this number of partitions in parallel using the specified number of cores.
If you specify fewer partitions than cores, some cores are not fully utilized, and resources are wasted. If you specify more partitions than cores, the Ocient System divides partitions in a round-robin fashion over the available cores. | | BATCH_SIZE | A dynamic value,

determined by the Ocient System for each pipeline to maximize performance. | INTEGER | Number of rows in the batch to load at one time.
The Ocient System automatically calculates a dynamic value depending on the table columns and the utilization of internal buffers to transfer records to the database backend. You can use this option to turn off the dynamic adjustments for performance tuning.
⚠ Only change this setting in rare cases where loading performance is slower than expected, and you have a large record size. If this setting is improperly set, pipelines might fail with out-of-memory exceptions.
You can configure the default value (for the batch payload target) using a SQL statement such as:
`ALTER SYSTEM ALTER CONFIG SET 'streamloader.extractorEngineParameters.configurationOption.osc.batch.payload.target ' '65536'` | | RECORD_NUMBER_FORMAT | For file loads that do not use the **EXPLODE_OUTER** function, the default is `[19, 45, 0]`.
For Kafka loads that do not use the `EXPLODE_OUTER` function, the default is `[0, 64, 0]`.
For loads that use the `EXPLODE_OUTER` function, the default is the specified load-type-specific default value with 13 subtracted from the record index bits. The system adds these bits to the bits for rows within a record.
For example, the default for file loads that use this function is `[19, 32, 13]`. | ARRAY | The 64-bit record number for each record of the load. This number uniquely identifies a row within its partition.
The format is an array with three values in the format`[, , ]`
The file index bits “ value is the number of bits used to represent the file index within a partition.
The record index bits “ value is the number of bits used to represent the record index within a file.
The rows per record index bits “ is the number of bits used to represent the row within a record. The system uses this value with the `EXPLODE_OUTER` function.
These three values must sum to 64.
Example
`RECORD_NUMBER_FORMAT= [10, 54, 0]`
Set the number of file index bits to `10` and the number of record index bits to `54`, allowing up to 2^10 files and 2^54 records per file. The system does not support the `EXPLODE_OUTER` function in this configuration because the rows per record index bits are `0`. | |

DROP PIPELINE

`DROP PIPELINE` removes an existing pipeline in the current database. You cannot remove a pipeline that is running.

You must have the DROP privilege on the pipeline to execute this SQL statement. For details, see **Data Control Language (DCL) Statement Reference**.

> ⚠ When you drop a pipeline, the Ocient System also removes the associated system catalog information, such as pipeline errors, events, files, partitions, and metrics.

Syntax

```
SQL

DROP PIPELINE [ IF EXISTS ] pipeline_name [, ...]
```

Parameter	Data Type	Description
`pipeline_name`	string	The name of the specified pipeline to remove. You can drop multiple pipelines by specifying additional pipeline names, separated by commas.

Examples

Remove Existing Data Pipeline

Remove an existing pipeline named `ad_data_pipeline`.

```
SQL

DROP PIPELINE ad_data_pipeline;
```

Remove Existing Data Pipeline by Checking for Existence

Remove an existing pipeline named `ad_data_pipeline` or return a warning if the Ocient System does not find the pipeline in the database.

```sql
DROP PIPELINE IF EXISTS ad_data_pipeline;
```

PREVIEW PIPELINE

`PREVIEW PIPELINE` enables you to view the results of loading data for a specific `CREATE PIPELINE` SQL statement without creating a whole data pipeline and without storing those results in the target table. Using this SQL statement, you can iterate quickly and modify the syntax as needed to achieve your expected results. After you confirm your expected results, you can use the same syntax in the body of the `CREATE PIPELINE` statement with the appropriate source.

A table must exist in the database to serve as the target of your `PREVIEW PIPELINE` statement. This table ensures the pipeline matches the column types of the target table. However, the execution of this statement does not load data into the target table.

Preview Sources

The `SOURCE INLINE` source type is available only for the `PREVIEW PIPELINE` SQL statement. You cannot create a data pipeline with inline source data. The inline source data limit is 1,000 rows.

Other source types defined in the `CREATE PIPELINE` statement, `S3`, `KAFKA`, and `FILESYSTEM`, are compatible with the `PREVIEW PIPELINE` statement. The extract options vary by the source type to mirror the `CREATE PIPELINE` statement. The database returns 10 records by default.

Preview Error Handling

Pipeline-level errors cause the `PREVIEW PIPELINE` SQL statement to fail. The Ocient System returns an error and no result set. However, the Ocient System accumulates record-level errors that occur during the execution of this statement in a single warning that the system returns along with the result set. Each line of the warning describes a record-level error in human-readable form or as a JSON blob, depending on the value of the `SHOW_ERRORS_AS_JSON` option. Rows or columns that encounter record-level errors have `NULL` values in the result set.

Preview Limitations

Limitations of this SQL statement are:

- Before executing a `PREVIEW PIPELINE` SQL statement, you must create a table for the Ocient System to have context for the preview.
- The maximum number of rows a `PREVIEW PIPELINE` SQL statement can return is 1,000 rows.
- The `COLUMN_DEFAULT_IF_NULL` option from the `CREATE PIPELINE` SQL statement has no effect on the `PREVIEW PIPELINE` SQL statement.
- The `PREVIEW PIPELINE` SQL statement does not honor the assignment of a service class based on text matching.
- These source options are not supported:
  - `START_FILENAME`
  - `END_FILENAME`
- When you execute two duplicate `PREVIEW PIPELINE` statements for a specific Kafka topic, the two statements share a consumer group. If the topic is small, one or both of the result sets might only be a partial result.
- For multiple tables, you can preview only one table at a time. You must specify the name of the table you want to preview using the `FOR` keyword.
- Previewing a continuous data pipeline is not supported.

Syntax

© 2026 Ocient, Inc.

SQL

```
PREVIEW PIPELINE pipeline_name
    [ MODE mode ]
    [ SHOW_ERRORS_AS_JSON show_errors_as_json ]
    SOURCE [ INLINE ] (inline_string | <s3_source> | <filesystem_source> | <kafka_s
        [ LIMIT limit ]
    EXTRACT
        FORMAT csv
        RECORD_DELIMITER record_delimiter
        FIELD_DELIMITERS ['delim1', 'delim2', ...]
        [ INTERMEDIATE_VALUES intermediate_values ]
[ INSERT ] INTO created_tablename
SELECT
    preview_column_formula AS preview_column_name, ...
[ WHERE filter_expression ]
[ [ INSERT ] INTO created_tablename_n
SELECT
    preview_column_formula AS preview_column_name, ...
[ WHERE filter_expression ] ] [ ,... ]
[ FOR created_tablename_n ]
```

Though this syntax shows the CSV format, you can also use the `PREVIEW PIPELINE` statement with the other formats.

Parameter	Data Type	Description
`pipeline_name`	string	The name of the specified data pipeline for the preview.
`created_tablename`	identifier	The identifier for the name of the table that you create before executing the `PREVIEW PIPELINE` SQL statement.
`created_tablename_n`	identifier	For multiple tables, the identifier for the name of another table that you create before executing the `PREVIEW PIPELINE` SQL statement. Use the `FOR` keyword to specify which table content to preview.
`preview_column_formula`	identifier	The identifier for the formula of the data to load. For example, for the data in the first field of the inline source, use `$1`. If you need to add a transformation, you can use functions to transform data, such as `CONCAT($1, $2)`, to load the concatenation of the first two fields in the inline source data.

Parameter	Data Type	Description
`preview_column_name`	identifier	The name of the column in the target table.

SQL Statement Options

Option Key	Default	Data Type	Description
SOURCE INLINE	Required	STRING	The string that contains data for the preview of the data pipeline load. For example, source data can be `'oci,ent,ocient|ware,house,warehouse'`, where `|` is the record delimiter and `,` is the field delimiter. For special characters, such as `\t`, use an escape sequence such as `e'oci,ent,oci\tent|ware,house,ware\thouse'`. This option supports these text-based formats: CSV, JSON, and XML.
MODE	`'transform'`	STRING	Indicates whether to perform a validation of the PREVIEW PIPELINE SQL statement. Valid values are: `'validate'` and `'transform'`. Set this option to `'validate'` for checking that the creation of the data pipeline succeeds. If the pipeline is valid, the statement produces no output; otherwise, it returns an error. Set this option to `'transform'` to retrieve a preview of the results of the pipeline.

Option Key	Default	Data Type	Description
SHOW_ERRORS_AS_JSON	`false`	BOOLEAN	Indicates whether to show errors. Values are `true` or `false`. If the value is `true`, the Ocient System returns record-level errors as JSON blobs rather than human-readable messages.
LIMIT	10	INTEGER	The number of rows, specified as an integer, to return in the preview results for sources with many rows. The default value is 10 rows.
INTERMEDIATE_VALUES	`false`	BOOLEAN	Indicates whether to capture intermediate values during a transformation sequence. Values are `true` or `false`. If the value is `true`, the Ocient System appends an extra column to the result set. Each value in the column contains a JSON blob that describes the intermediate values processed for each column after each transformation.

You must specify at least one column name in the `SELECT` part of the syntax. The name of the specified column must match the name of the column in the created table. The number of columns in the `SELECT` part can be less than those in the created table.

For definitions of other extract options, see the `CREATE PIPELINE` SQL statement options in **CREATE PIPELINE**.

Examples

Preview Pipeline Using CSV Format

Preview the load of two rows of data. First, create a table to serve as the context for the load. The `previewload` table contains three columns with these data types: string, integer, and Boolean.

SQL

```
CREATE TABLE previewload (col1 VARCHAR, col2 INT, col3 BOOLEAN);
```

Create the preview pipeline `testpipeline` with this data: `'hello,2,true|bye,3,false'`. Specify the CSV extract format, `|` record delimiter, and the `,` field delimiter. Load the data without transformation.

SQL

```
PREVIEW PIPELINE testpipeline
    SOURCE INLINE 'hello,2,true|bye,3,false'
    EXTRACT
        FORMAT CSV
        RECORD_DELIMITER '|'
        FIELD_DELIMITERS [',']
INTO previewload
SELECT
    $1 AS col1,
    $2 AS col2,
    $3 AS col3;
```

Output

Text

```
col1                                              col2       col3
\-----------------------------------------------------------------
hello                                             2          true
bye                                               3          false

Fetched 2 rows
```

Delete the `previewload` table.

SQL

```
DROP TABLE previewload;
```

© 2026 Ocient, Inc.

Preview Pipeline Using CSV Format with Escape Characters

Preview the load of two rows of data. First, create a table to serve as the context for the load. The `previewload` table contains three columns with these data types: string, integer, and Boolean.

SQL

```
CREATE TABLE previewload (col1 VARCHAR, col2 INT, col3 BOOLEAN);
```

Create the preview pipeline `testpipeline` with this data: `'hello\tworld,2,true|bye\tworld,3,false'`. Specify the CSV extract format, `|` record delimiter, and `,` field delimiter. Load the data without transformation. In this case, the data contains the special character `\t`. You must escape the character by using the escape sequence `e`.

SQL

```
PREVIEW PIPELINE testpipeline
    SOURCE INLINE e'hello\tworld,2,true|bye\tworld,3,false'
    EXTRACT
        FORMAT CSV
        RECORD_DELIMITER '|'
        FIELD_DELIMITERS [',']
    INTO previewload
    SELECT
        $1 AS col1,
        $2 AS col2,
        $3 AS col3;
```

Output

Text

```
col1                                                col2       col3
\-------------------------------------------------------------------
hello   world                                         2        true
bye     world                                         3        false

Fetched 2 rows
```

Delete the `previewload` table.

SQL

```
DROP TABLE previewload;
```

Preview Pipeline Using CSV Format with Transformation

Create a table to serve as the context for the load. The `previewload` table contains three string columns.

SQL

```
CREATE TABLE previewload (col1 VARCHAR, col2 VARCHAR, col3 VARCHAR);
```

Create the preview pipeline `testpipeline` with this data: `'hello,world|bye,world'`. Specify the CSV extract format, `|` record delimiter, and the `,` field delimiter. Load the data with a transformation to concatenate the two strings and return the result in the third column.

SQL

```
PREVIEW PIPELINE testpipeline
    SOURCE INLINE 'hello,world|bye,world'
    EXTRACT
        FORMAT CSV
        RECORD_DELIMITER '|'
        FIELD_DELIMITERS [',']
    INTO previewload
    SELECT
        $1 AS col1,
        $2 AS col2,
        CONCAT($1,$2) AS col3;
```

Output

Text

```
col1                                                col2
\-------------------------------------------------------------------------------
hello                                               world
bye                                                 world

Fetched 2 rows
```

The third column contains the concatenated result of the first two columns.

Delete the `previewload` table.

SQL

```
DROP TABLE previewload;
```

Preview Pipeline Using the Kafka Source

Create the `previewload` table with these columns:

- `id` — Non-NULL integer
- `salut` — Non-NULL string

- `name` — Non-NULL string
- `surname` — Non-NULL string
- `zipcode` — Non-NULL integer
- `age` — Non-NULL integer
- `rank` — Non-NULL integer

SQL

```
CREATE TABLE previewload (
    id INT NOT NULL,
    salut VARCHAR(3) NOT NULL,
    name VARCHAR(10) NOT NULL,
    surname VARCHAR(10) NOT NULL,
    zipcode INT NOT NULL,
    age INT NOT NULL,
    rank INT NOT NULL);
```

Create the preview pipeline `test_small_kafka_simple_csv`. Specify the `ddl_csv` topic. Indicate that the Kafka consumer should not write its durably-made record offsets to the Kafka Broker by using the `WRITE_OFFSETS` option set to `false`. Specify the bootstrap server as `servername:0000` and configuration options as `"auto.offset.reset": "earliest"` by using the `BOOTSTRAP_SERVERS` and `CONFIG` options, respectively. Limit the returned results to three rows by using the `LIMIT` option. Specify the CSV extract format and `\n` record delimiter by using the `FORMAT` and `RECORD_DELIMITER` extract options, respectively.

© 2026 Ocient, Inc.

SQL

```
PREVIEW PIPELINE test_small_kafka_simple_csv
    SOURCE KAFKA
        TOPIC 'ddl_csv'
        WRITE_OFFSETS false
        BOOTSTRAP_SERVERS 'servername:0000'
        CONFIG '{"auto.offset.reset": "earliest"}'
        LIMIT 3
    EXTRACT
        FORMAT csv
        RECORD_DELIMITER '\n'
    INTO previewload
    SELECT
        INT($1) AS id,
        CHAR($2) AS salut,
        CHAR($3) AS name,
        CHAR($4) AS surname,
        INT($5) AS zipcode,
        INT($6) AS age,
        INT($7) AS rank;
```

Text

```
id          salut                                                name
\-------------------------------------------------------------------------------
105         Mr                                                   Jmhsuxofspx
101         Mr                                                   Ijmmtbddkyh
109         Mr                                                   Bigohpwfwmr

Fetched 3 rows
```

START PIPELINE

`START PIPELINE` begins the execution of the specified data pipeline that extracts data and loads it into the target tables specified by the `CREATE PIPELINE` SQL statement.

When you execute the `START PIPELINE` SQL statement, the Ocient System creates a static list of files in the `sys.pipeline_files` system catalog table and marks them with the `PENDING` status. After the system assigns a file to an underlying task, the system marks the file as `QUEUED`. After the system verifies that the file exists, the system marks the file as `LOADING` to signify that a Loader Node has started reading the source data. Finally, upon successfully loading the file, the system transitions the status of the file to the terminal status `LOADED`.

A Kafka pipeline never enters the `COMPLETED` state in the `information_schema.pipeline_status` view. Instead, the pipeline remains running after you start the pipeline until you stop it or the pipeline reaches the specified error limit using the `ERROR LIMIT` option.

You must have the EXECUTE privilege on the pipeline and the INSERT privilege on any table that is a target in the pipeline. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

SQL

```
START PIPELINE pipeline_name
    [ ERROR
        [ LIMIT <integer_value> ]
        [ FILE_ERROR (FAIL | SKIP_MISSING_FILE | TOLERATE) ] ]
    [ USING LOADERS <loader_names> ]
    [ ON COMPLETION (NO_FLUSH | FLUSH_AND_WAIT | FLUSH_AND_RETURN) ]
```

Parameter	Data Type	Description
`pipeline_name`	string	The name of the specified data pipeline.

SQL Statement Options

Option Key	Default	Data Type	Description
ERROR LIMIT `<integer_value>`	0	INTEGER	Error log option that determines the number of record-level errors that can occur during the execution of a pipeline that the load tolerates before the whole pipeline execution

Option Key	Default	Data Type	Description
			fails. `<integer_value>` is a number greater than or equal to -1. When you set `<integer_value>` to -1, the load tolerates an unlimited number of record-level errors. By default, continuous pipelines tolerate an unlimited number of record-level errors, whereas batch pipelines tolerate zero errors. With multiple target tables, record-level errors are specific to a table. For example, if a transformation for loading one table succeeds but another transformation for loading another table fails, the row corresponding to the successful transformation loads to the first table but not the other.
ERROR FILE_ERROR `<error_action>`	`FAIL`	STRING	For pipelines that load data from S3 or local file sources, this error configuration option determines how to treat unrecoverable file-level errors. Examples of unrecoverable file-level errors are: The file is listed when the pipeline starts, but is missing later during the load. The Gzip file is corrupted and cannot be decompressed. The file cannot be downloaded from the source. Record-level error that is not tolerable occurs when tokenizing or transforming data in the file. `<error_action>` can be one of these keywords: `FAIL` — Fail the whole pipeline because of a file-level error. `SKIP_MISSING_FILE` — Only tolerate errors that occur due to missing files. If a file exists in the list when the pipeline starts but is missing later during the load, skip the file

Option Key	Default	Data Type	Description
			and continue with the next file. `TOLERATE` — Tolerate all unrecoverable file-level errors. In this mode, the load also tolerates an unlimited number of record-level errors. The `FAILED`, `SKIPPED`, and `LOADED_WITH_ERRORS` file statuses appear in the `sys.pipeline_files` system catalog tables, respectively, and indicate how the pipeline handled the file error.
USING LOADERS `loader_names`	`NULL`	LIST OF STRINGS	Specify one or more names of Loader Nodes as a comma-separated list for executing the `START PIPELINE` SQL statement. If you do not use this option, the Ocient System uses all of the Loader Nodes that are active to execute the pipeline. You can find node names in the `sys.nodes` system catalog table.
ON COMPLETION `<completion_mode>`	`NO_FLUSH`	STRING	Completion type option that specifies the behavior when the pipeline finishes loading. This option determines when the remaining pages are converted into Segments. `NO_FLUSH` — Do not force a flush of pages. Rely on watermarks and timeouts to trigger final conversion to Segments. `FLUSH_AND_WAIT` — Trigger a flush of pages, initiating final conversion to Segments. The pipeline blocks and waits for the conversion to Segments to complete before marking the pipeline as `COMPLETED`. `FLUSH_AND_RETURN` — Trigger a flush of pages, initiating the final conversion to Segments. The Ocient System marks the pipeline as `COMPLETED` immediately following the flush without waiting for conversion to Segments to complete.

For the query to execute successfully, the specified node names must identify nodes that have:

- `ACTIVE` operational status
- `streamloader` role

When you execute the `START PIPELINE` SQL statement, the Ocient System creates a static list of files only for batch pipelines and a dynamic list for continuous pipelines in the `sys.pipeline_files` system catalog table.

Examples

Start an existing pipeline named `ad_data_pipeline` with default settings.

SQL

```
START PIPELINE ad_data_pipeline;
```

Start an existing pipeline named `ad_data_pipeline` with error tolerance (tolerate 10 errors before aborting the pipeline). For details about error tolerance, see **Error Tolerance in Data Pipelines**.

SQL

```
START PIPELINE ad_data_pipeline
    ERROR LIMIT 10;
```

Data pipelines log a message for each pipeline error to the `sys.pipeline_errors` system catalog table, even if you do not specify the ERROR option. Use `BAD_DATA_TARGET` settings to capture the original source data.

Start an existing pipeline named `ad_data_pipeline` using the Loader Node named `stream-loader1`.

SQL

```
START PIPELINE ad_data_pipeline
    USING LOADERS "stream-loader1";
```

To resume a pipeline with file loading, see **Data Pipeline Behavior Considerations**.

To restart a Kafka data pipeline, see **Data Pipeline Behavior Considerations**.

For pipeline dependencies, see **Data Pipeline Behavior Considerations**.

STOP PIPELINE

`STOP PIPELINE` stops the execution of the pipeline and its associated tasks. After you stop a pipeline, you can execute the `START PIPELINE` SQL statement on the pipeline to run the pipeline again. Regardless, the load deduplicates any records previously loaded in the same pipeline.

You must have the EXECUTE privilege on the pipeline to execute this SQL statement. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

```
SQL

STOP PIPELINE pipeline_name
```

Parameter	Data Type	Description
`pipeline_name`	string	The name of the specified data pipeline.

Example

Stop an existing pipeline named `ad_data_pipeline`.

```
SQL

STOP PIPELINE ad_data_pipeline;
```

You can see the status of the parent tasks in the `sys.tasks` system catalog table and see the status of the child tasks in the `sys.subtasks` system catalog table.

ALTER PIPELINE

`ALTER PIPELINE` evolves the schema of the data pipeline in the `SELECT` clause. For pipelines with multiple tables, specify each `SELECT` clause for tables in the existing pipeline. The `WHERE` clause must match the corresponding clause of the existing pipeline. Use the `FORCE` keyword to change a non-backward-compatible schema or add, modify, or remove the `WHERE` clause. For details about schema evolution, see **Load Avro Data**.

> To execute this SQL statement, the data pipeline must not be running. You can use the `STOP PIPELINE` SQL statement to stop the execution of the data pipeline.

You must have the ALTER privilege on the pipeline to execute this SQL statement. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

SQL

```
ALTER PIPELINE [ IF EXISTS ] pipeline_name
[ INSERT ] INTO table_name
SELECT
    $field_name AS field_name [ , ... ]
[ WHERE ... ]
[ INSERT INTO table_nameN SELECT ... ]
[ FORCE ]
```

Parameter	Data Type	Description
`pipeline_name`	string	The name of the specified pipeline to alter.

Examples

These examples use this table and data pipeline definition. You must stop the execution of the data pipeline before performing schema evolution operations.

Create the `users` table with these columns:

- `id` — Universally Unique IDentifier (UUID) of the user

- `firstname` — First name of the user
- `lastname` — Last name of the user
- `birthyear` — Year of birth
- `groups` — List of groups where the user belongs

SQL

```
CREATE TABLE users(
    id         UUID NOT NULL,
    firstname  VARCHAR(255) NOT NULL,
    lastname   VARCHAR(255) NOT NULL,
    birthyear  INT,
    groups     VARCHAR(255)[] NOT NULL DEFAULT 'char[]'
);
```

Assume you have user data in Avro format in multiple files in the `/data/users` directory. Create the `users_pipeline` data pipeline for the Avro files containing user data `*.avro`. The schema configuration instructs the system to infer from all files using the `INFER_FROM` option.

SQL

```
CREATE PIPELINE users_pipeline
    SOURCE filesystem
        FILTER '/data/users/*.avro'
    EXTRACT
        FORMAT avro
        SCHEMA {
            INFER_FROM all_files
        }
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    $birthyear  AS birthyear;
```

Start the data pipeline.

SQL

```
START PIPELINE users_pipeline;
```

Schema Evolution with a Column Addition

Stop the data pipeline.

SQL

```
STOP PIPELINE users_pipeline;
```

Change the schema of an existing pipeline named `users_pipeline` to add the `groups` column. Access the array of strings within the column.

SQL

```
ALTER PIPELINE users_pipeline
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    $birthyear  AS birthyear,
    $groups[]   AS groups;
```

Start the data pipeline again.

SQL

```
START PIPELINE users_pipeline;
```

Check Whether the Data Pipeline to Change Exists for Column Addition

Stop the data pipeline.

SQL

```
STOP PIPELINE users_pipeline;
```

© 2026 Ocient, Inc.

Change the schema of an existing pipeline named `users_pipeline` to add the `groups` column. Use the `IF EXISTS` keywords to check whether the data pipeline exists. Access the array of strings within the column.

SQL

```
ALTER PIPELINE IF EXISTS users_pipeline
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    $birthyear  AS birthyear,
    $groups[]   AS groups;
```

Start the data pipeline again.

SQL

```
START PIPELINE users_pipeline;
```

Schema Evolution with a Column Removal

Stop the data pipeline.

SQL

```
STOP PIPELINE users_pipeline;
```

Change the schema of an existing pipeline named `users_pipeline` to remove the `birthyear` column.

© 2026 Ocient, Inc.

SQL

```
ALTER PIPELINE users_pipeline
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname;
```

Start the data pipeline again.

SQL

```
START PIPELINE users_pipeline;
```

Schema Evolution with a Column Data Type Modification

Stop the data pipeline.

SQL

```
STOP PIPELINE users_pipeline;
```

Change the schema of an existing pipeline named `users_pipeline` to narrow the `INT` data type to a `SMALLINT` type for the `birthyear` column using the `SMALLINT` casting function.

SQL

```
ALTER PIPELINE users_pipeline
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    SMALLINT($birthyear)  AS birthyear;
```

Start the data pipeline again.

SQL

```
START PIPELINE users_pipeline;
```

Schema Evolution with the Addition of a Filter Condition

Stop the data pipeline.

SQL

```
STOP PIPELINE users_pipeline;
```

Change the schema of an existing pipeline named `users_pipeline` to filter the year of birth to be greater than 1950 using the `WHERE` clause. Add the `FORCE` keyword to instruct the data pipeline to evolve the schema with the filter condition.

SQL

```
ALTER PIPELINE users_pipeline
INTO users
SELECT
    $id         AS id,
    $firstname  AS firstname,
    $lastname   AS lastname,
    $birthyear  AS birthyear,
WHERE
    $birthyear > 1950
FORCE;
```

Start the data pipeline again.

SQL

```
START PIPELINE users_pipeline;
```

ALTER PIPELINE RENAME

`ALTER PIPELINE RENAME TO` SQL statement changes the name of the pipeline object, while retaining its identifier, options, and other metadata. The Ocient System reflects this change in the `sys.pipelines` system catalog table. Then, you must use the new name when you refer to the pipeline in SQL statements.

You must have the ALTER privilege on the pipeline to execute this SQL statement. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

SQL

```
ALTER PIPELINE [ IF EXISTS ] pipeline_original_name RENAME TO pipeline_new_name
```

Parameter	Data Type	Description
`pipeline_original_name`	string	The name of the existing data pipeline.
`pipeline_new_name`	string	The new name of the data pipeline.

Example

Rename an existing pipeline named `ad_data_pipeline` to `renamed_pipeline`.

SQL

```
ALTER PIPELINE ad_data_pipeline RENAME TO renamed_pipeline;
```

EXPORT PIPELINE

`EXPORT PIPELINE` returns the `CREATE PIPELINE` SQL statement used to create the pipeline object. You can use the output of this statement to recreate an identical pipeline when you remove the original pipeline.

The execution of this statement censors sensitive S3 values like `ACCESS_KEY_ID` and `SECRET_ACCESS_KEY` and Kafka **Consumer Configuration** password-type fields. The database replaces them with `*`.

To execute this statement, you must have the VIEW privilege on the pipeline and any table the pipeline targets. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

```
SQL

EXPORT PIPELINE pipeline_name
```

Parameter	Data Type	Description
`pipeline_name`	string	The name of the specified data pipeline.

Example

Export an existing pipeline in the database `ad_data_pipeline`.

```
SQL

EXPORT PIPELINE ad_data_pipeline;
```

CREATE PIPELINE FUNCTION

`CREATE PIPELINE FUNCTION` enables you to define a function for loading data. Define the function behavior using the Apache® Groovy™ language. For details bout this language, see **The Apache Groovy Programming Language**.

Function arguments and output are strongly typed and immutable.

You can test the execution of your function using the `PREVIEW PIPELINE` SQL statement.

> The Ocient System does not support the overload of function names.

Syntax

SQL

```
CREATE [ OR REPLACE ] PIPELINE FUNCTION [ IF NOT EXISTS ]
    function_name( input_argument [, ...] )
    LANGUAGE GROOVY
    RETURNS output_argument_definition
    IMPORTS [ library_name [, ...] ]
    AS $$
        groovy_declaration
    $$
```

Parameter	Type	Description
`function_name`	string	A unique identifier for the data pipeline function.
`input_argument`	string	The name of one or more input arguments of the function. Specify data types for input arguments according to the support data types defined in **Data Types for Data Pipelines**. For the data type declaration, use `NOT NULL` where applicable for maximum performance.
`output_argument_definition`	string	The type definition of the output from the function.
`library_name`	string	The name of one or more Java libraries. You can include libraries by using the `IMPORTS` clause or specifying the fully-qualified class (e.g., `java.lang.Integer`) path in the source definition.
`groovy_declaration`	string	The Groovy definition of the function.

Install and Enable Third-Party Libraries

You can use the default list of supported third-party libraries or additional third-party libraries that you install.

Supported Libraries

Data pipeline functions can import classes from the default list of supported third-party libraries. This table provides the resources for each supported library package.

Library Package	Resource
`java.lang.*`	**java.lang**
`java.util.*`	**java.util**
`java.nio.ByteBuffer.*`	**ByteBuffer**
`groovy.json.*`	**groovy.json**
`groovy.xml.*`	**groovy.xml**
`groovy.yaml.*`	**groovy.yaml**
`org.apache.groovy.datetime.extensions.*`	**org.apache.groovy.datetime.extensions**
`org.apache.groovy.dateutil`	**org.apache.groovy.dateutil.extensions**
`com.ocient.streaming.data.types.*`	**Data Types for User-Defined Data Pipeline Functions**

Additional Libraries

You can install and enable additional third-party libraries to import for use in your data pipeline functions.

You must install the JAR package on all Loader Nodes in the `/opt/ocient/current/lib/extractorengine_udt` folder.

Then, add the fully qualified class name in the function import list as part of the `library_name` parameter. For example, to reference the `ByteBuffer` class from the `com.fastbuffer` package, specify `com.fastbuffer.ByteBuffer` in the `library_name` parameter and use the class in the Groovy definition as `var x = new com.fastbuffer.ByteBuffer()`.

Groovy Data Type Mapping

For the Groovy definition, the Ocient System maps its SQL data type to the corresponding Groovy data type. Your Groovy code should use the Groovy data type defined in this table for any input arguments and output.

© 2026 Ocient, Inc.

SQL Data Type	Groovy Data Type
`BIGINT`	`java.lang.Long`
`BINARY(N) or HASH(N)`	`byte[]`
`BOOLEAN`	`java.lang.Boolean`
`CHAR(N) or VARCHAR(N)`	`java.lang.String`
`DATE`	`java.time.LocalDate`
`DECIMAL(P,S)`	`com.ocient.streaming.data.types.Decimal`
`DOUBLE`	`java.lang.Double`
`INT`	`java.lang.Integer`
`IPV4`	`java.net.Inet4Address`
`IP`	`java.net.Inet6Address`
`ST_POINT`	`com.ocient.streaming.data.types.gis.STPoint`
`ST_LINESTRING`	`com.ocient.streaming.data.types.gis.STLinestring`
`ST_POLYGON`	`com.ocient.streaming.data.types.gis.STPolygon`
`FLOAT`	`java.lang.Float`
`SMALLINT`	`java.lang.Short`
`TIME`	`com.ocient.streaming.data.types.Time`
`TIMESTAMP`	`com.ocient.streaming.data.types.Timestamp`
`BYTE`	`java.lang.Byte`
`TUPLE<<type1, type2, …>>`	`com.ocient.streaming.data.types.OcientTuple`
`TYPE[]`	`java.util.List<TYPE>`
`UUID`	`java.util.UUID`
`VARBINARY(N)`	`byte[]`
`VARCHAR(N)`	`java.lang.String`

Example

Create the `sort_function` data pipeline function to sort an array of integers. The function has two input arguments: `value`, a non-NULL array of integers, and `ascending`,

the sort order. The function returns a non-NULL array of integers. If `value` is empty, the function throws an error.

The function imports these Java libraries:

- `java.lang.Integer`
- `java.util.ArrayList`
- `java.util.Collections`
- `java.util.Comparator`
- `java.util.List`

Define the Groovy code. Because the input arguments do not change, the example Groovy code first copies the `value` argument, sorts the copied list according to the sort order, and returns the sorted array.

© 2026 Ocient, Inc.

SQL

```
CREATE PIPELINE FUNCTION
    sort_function( value INT[] NOT NULL, ascending BOOLEAN NOT NULL)
    LANGUAGE GROOVY
    RETURNS INT[] NOT NULL
    IMPORTS [
        'java.lang.Integer',
        'java.util.ArrayList',
        'java.util.Collections',
        'java.util.Comparator',
        'java.util.List'
        ]
    AS $$
        /* Throw an error if the array is empty */
        if (value.isEmpty()){
            throw new PipelineFunctionException("Unexpected empty array");
        }

        /* Make a copy of the list. */
        List<Integer> sorted = new ArrayList<>((List<Integer>)value);

        /* Sort the array elements according to the specified order. */
        sorted.sort(ascending ? Comparator.naturalOrder() : Comparator.reverseOrder

        /* Return the sorted array. */
        return sorted;
    $$;
```

View the creation information about the `sort_function` function using the `sys.pipeline_functions` system catalog table. This statement returns the function name, return type, argument names, data types of the arguments, and the list of imported libraries.

SQL

```
SELECT name, return_type, argument_names, argument_types, imported_libraries
FROM sys.pipeline_functions;
```

DROP PIPELINE FUNCTION

`DROP PIPELINE FUNCTION` removes an existing pipeline function.

You must have the DROP privilege on the pipeline function to execute this SQL statement. For details, see **Data Control Language (DCL) Statement Reference**.

Syntax

```
SQL

DROP PIPELINE FUNCTION [ IF EXISTS ] function_name [, ...]
```

Parameter	Data Type	Description
`function_name`	string	The name of the specified data pipeline function to remove. You can drop multiple pipelines by specifying additional function names, separated by commas.

Examples

Remove the Existing Pipeline Function

Remove an existing pipeline function named `sort_function`.

```
SQL

DROP PIPELINE FUNCTION sort_function;
```

Remove an Existing Pipeline Function by Checking for Existence

Remove an existing pipeline function named `sort_function` or return a warning if the Ocient System does not find the function in the database.

```
SQL

DROP PIPELINE FUNCTION IF EXISTS sort_function;
```

Related Links

Load Data

Data Pipeline Load of CSV Data from S3

Data Pipeline Load of Parquet Data from S3

Transform Data in Data Pipelines

Data Pipeline Behavior Considerations

Load Metadata and File-Based Partitioned Data in Data Pipelines

Data Control Language (DCL) Statement Reference

Identifiers

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Transactional Data Pipelines

Transactional data pipelines in Ocient® load data in batches and automatically roll back all uncommitted data when a failure occurs. The `CREATE TRANSACTIONAL PIPELINE` SQL statement supports a `START FOREGROUND` clause that creates a pipeline, immediately starts it, and blocks the SQL session until the pipeline reaches an inactive state — either `COMPLETED` or `FAILED`. This approach provides a single-command batch loading from within a SQL session.

> **Info:** The `START FOREGROUND` clause is only valid for `TRANSACTIONAL` pipelines. The Ocient System rejects this clause on `CONTINUOUS` and `BATCH` pipelines.

CREATE TRANSACTIONAL PIPELINE

The `CREATE TRANSACTIONAL PIPELINE` SQL statement creates a new transactional data pipeline in the current database or replaces an existing one when you specify the `OR REPLACE` keywords. When you include the `START FOREGROUND` clause, the statement also starts the pipeline immediately and blocks until the pipeline reaches an inactive state.

```
CREATE [OR REPLACE] TRANSACTIONAL PIPELINE pipeline_name
    SOURCE source_type source_options
    EXTRACT extract_options
    INTO table_name
    SELECT column_list
    [START FOREGROUND
        [ERROR (LIMIT n, FILE_ERROR {FAIL | SKIP_MISSING_FILE | TOLERATE})]
        [USING LOADERS loader_name [, ...]]
        [ON COMPLETION {NO_FLUSH | FLUSH_AND_WAIT | FLUSH_AND_RETURN}]
    ]
;
```

Parameter	Type	Description
`pipeline_name`	Identifier	The name of the pipeline to create. The name must be distinct from any existing pipeline names in the current database.

© 2026 Ocient, Inc.

Parameter	Type	Description
`source_type`	Identifier	The type of data source (for example, `S3` or `HDFS`). See **CREATE PIPELINE** for supported values.
`source_options`	Varies	Source-specific connection and configuration settings. See **CREATE PIPELINE** for details.
`extract_options`	Varies	Format and parsing settings for the source data. See **CREATE PIPELINE** for details.
`table_name`	Identifier	The target table for loading data.
`column_list`	String	The list of columns for the load.
`n`	INTEGER	The maximum number of errors the system permits before the pipeline transitions to the `FAILED` status.
`loader_name`	Identifier	The name of a Loader Node to use for this pipeline run. Separate multiple Loader Node names with commas in the `USING LOADERS` clause.

The `ERROR` clause controls how the system behaves with errors. The `LIMIT` keyword limits the number of errors. The `FILE_ERROR` keyword denotes the behavior when a source file is missing or unreadable. The `FAIL` value causes the pipeline to fail immediately. The `SKIP_MISSING_FILE` value silently skips missing files. And the `TOLERATE` value records the error and continues loading.

The `ON COMPLETION` clause controls segment generation after the pipeline completes. The default value is `NO_FLUSH`. The `FLUSH_AND_WAIT` value denotes that the system flushes segments synchronously before returning. The `FLUSH_AND_RETURN` value denotes that the system initiates a flush and returns immediately.

START FOREGROUND Clause

The `START FOREGROUND` clause is not valid in combination with the `IF NOT EXISTS` clause. When you omit the `START FOREGROUND` clause, the `CREATE TRANSACTIONAL PIPELINE` statement creates the pipeline without starting it, and all existing behavior is unchanged.

The `ERROR`, `USING LOADERS`, and `ON COMPLETION` sub-clauses within the `START FOREGROUND` clause behave identically to the same clauses as in the `START PIPELINE` SQL statement.

Blocking Behavior

When you execute the `CREATE TRANSACTIONAL PIPELINE ... START FOREGROUND` SQL statement, the SQL session blocks for the full duration of pipeline execution. The session does not return until the pipeline reaches an inactive state (`COMPLETED` or `FAILED`).

If you disconnect from the SQL session during pipeline execution, for example, by pressing Ctrl+C at the command line, the pipeline continues running independently. The disconnection does not stop or cancel the pipeline. After the pipeline reaches an inactive state, the Ocient System records the final status and all associated events in the `sys.pipelines` and `sys.pipeline_events` system catalog tables.

Queuing Behavior

If all pipeline concurrency slots are occupied when you execute the statement, the system places the pipeline in a `QUEUED` state. The SQL session remains blocked while the pipeline is queued and continues to block after a concurrency slot opens, until the pipeline itself reaches an inactive state. You can monitor the pipeline status in the `sys.pipelines` system catalog table from a separate session during this time.

Error Behavior

When the pipeline reaches the `COMPLETED` state, the statement returns successfully. When the pipeline reaches the `FAILED` or `STOPPED` state, the statement throws a `SQLException` error that contains the status message of the pipeline.

The `ERROR LIMIT` and `FILE_ERROR` keywords control failure thresholds in the same way as in the `START PIPELINE` statement. When the number of errors exceeds the `LIMIT` value, the pipeline transitions to the `FAILED` state.

Transactional Rollback

On failure, the system automatically rolls back the transactional storage scope. The system does not commit any data from the failed pipeline run to the target table. This rollback behavior is identical to any other transactional pipeline failure and requires no special handling.

© 2026 Ocient, Inc.

Pipeline Lifecycle

Pipelines started with the `START FOREGROUND` clause follow the same lifecycle as any other pipeline. The system records all lifecycle events, including `CREATED`, `RUNNING`, `COMPLETED`, and `FAILED` states, in the `sys.pipeline_events` table. The `STARTED` event includes the start options used for the run, providing a complete audit trail. After the pipeline reaches an inactive state, the pipeline object persists in the `sys.pipelines` system catalog table with full metadata and metrics.

Examples

Create a Transactional Pipeline with an Error Limit

This statement creates a transactional pipeline that loads CSV files from Amazon® Web Services℠ (AWS℠) S3, tolerates up to 100 errors, and blocks until loading completes or the system exceeds the error limit.

```
CREATE OR REPLACE TRANSACTIONAL PIPELINE my_pipeline
    SOURCE S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/csv/orders.csv'
        ENABLE_PATH_STYLE_ACCESS true
    EXTRACT FORMAT CSV
        FIELD_DELIMITER ','
        NUM_HEADER_LINES 1
    INTO my_schema.my_table
    SELECT
        $1 as id,
        $2 as user_id,
        $3 as product_id
    START FOREGROUND
        ERROR LIMIT 100, FILE_ERROR TOLERATE
        ON COMPLETION FLUSH_AND_WAIT;
```

Create a Transactional Pipeline Using a Specific Loader Node

This statement creates a transactional pipeline using a specific Loader Node and returns as soon as the pipeline completes or fails, with no segment flush.

```
CREATE OR REPLACE TRANSACTIONAL PIPELINE sales_pipeline
    SOURCE S3
        ENDPOINT 'https://s3.us-east-1.amazonaws.com'
        BUCKET 'ocient-docs'
        FILTER 'metabase_samples/csv/orders.csv'
        ENABLE_PATH_STYLE_ACCESS true
    EXTRACT FORMAT CSV
        FIELD_DELIMITER ','
        NUM_HEADER_LINES 1
    INTO sales.transactions
    SELECT
        $1 as id,
        $2 as user_id,
        $3 as product_id
    START FOREGROUND
        USING LOADERS loader_node_1
        ON COMPLETION NO_FLUSH;
```

Related Links

START PIPELINE CREATE PIPELINE DROP PIPELINE sys.pipelines sys.pipeline_events Transactions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Distributed Tasks

You can manage distributed tasks in the Ocient® System. These tasks are processes that execute in the background to keep the system running smoothly and efficiently. These DDL commands create tasks to rebuild damaged or missing segment data and check data integrity.

For information on managing tasks, see **Manage Distributed Tasks**.

CREATE TASK

Launches a new task.

Privileges

To create any task, you must have the System Administrator role, which has the `UPDATE` privileges on the system.

Syntax

SQL

```
CREATE TASK [ task_name ]
    TYPE { rebuild | check_disk | rebalance }
    [ LOCATION { SYSTEM | CLUSTER cluster_name | NODE node_name } ]
    [ OPTIONS task_option_map [, ... ] ]
```

To create a task, you must specify a task type `TYPE`:

- **rebuild** — Reconstruct data segments by using erasure-coded data in the same segment group.
- **check_disk** — Verify segments on a node by computing checksums.
- **rebalance** — Distribute data evenly across disks in the system.

You can also optionally specify a location using the `LOCATION` keyword to execute the task. Supported values include:

- `SYSTEM` — Targets all segments in the system. If you do not specify the `LOCATION` keyword, `SYSTEM` is the default value.
- `CLUSTER` — Targets a specific cluster.
- `NODE` — Targets a specific node.

Parameter	Data Type	Description
`task_name`	String	Optional. The identifier for the task.
`TYPE`	String	Required. Specifies the type of the distributed task to create. Supported tasks are: `rebuild` — Rebuild task `check_disk` — Check disk task `rebalance` — Rebalance task Task names are case sensitive.
`node_name`	String	This argument is required only if your task targets a node with the `LOCATION NODE` keywords. The name of a specific node for the task to target.
`cluster_name`	String	This argument is required only if your task targets a cluster with the `LOCATION CLUSTER` keywords. The name of a specific cluster for the task to target.
`task_option_map`	Key-value pairs	Optional. A comma-separated list of task-specific options in key-value pair format `key = value`.

`rebuild` Task

Performing a rebuild segment operation reconstructs data segments by using erasure-coded data in the same segment group. This task allows the system to restore segments that are in the damaged `DAMAGED` or missing `MISSING` status to full performance. For details on rebuild tasks, see **Guide to Rebuilding Segments**.

The `rebuild` task supports only the `SYSTEM` and `CLUSTER` locations. If you attempt the rebuild task on the node, the system returns an error.

To rebuild a single node, use the cluster location with the `LOCATION CLUSTER` keywords. Include the name of the node you want to rebuild as one of the key-value options.

`rebuild` Options (`task_option_map`)

The rebuild task supports these options. You must specify these options as a comma-separated list of key-value pairs.

Option Key	Value Type	Description
`node`	String	Limits the rebuild operation to only segments on the specified node.
`segment_groups`	List	Limits the rebuild operation to the specified segment group identifiers. Specify this option as a bracketed list, e.g., `[ id1, id2, ... ]`.
`table_id`	String	Limits the rebuild to segment groups belonging to a specific table identifier.

Examples

Rebuild All Segments on the System

This example performs a system-wide rebuild task.

SQL

```
CREATE TASK TYPE rebuild LOCATION SYSTEM;
```

Rebuild Segments on the Specific Table

This example rebuilds all segments on a system for the specified table with the table identifier `529a2e9d-d06c-46cd-a93b-624d3bed1c08`.

SQL

```
CREATE TASK TYPE rebuild OPTIONS table_id = '529a2e9d-d06c-46cd-a93b-624d3bed1c08';
```

Rebuild Segments on the Specific Cluster

This example rebuilds segments on the Foundation cluster `my_lts_cluster1`.

SQL

```
CREATE TASK TYPE rebuild LOCATION CLUSTER my_lts_cluster1;
```

Rebuild Segments on the Specific Node

This example rebuilds segments on the Foundation Node `my_node1` in the Foundation Cluster `my_lts_cluster1`.

SQL

```
CREATE TASK TYPE rebuild LOCATION CLUSTER my_lts_cluster1 OPTIONS node = 'my_node1'
```

Rebuild the Specific Segment Group

This example rebuilds segments in the segment group with the identifier `53` in the Foundation Cluster `my_lts_cluster1`.

SQL

```
CREATE TASK TYPE rebuild LOCATION CLUSTER my_lts_cluster1 OPTIONS segment_groups =
```

Create a Named Task

This example creates the rebuild task named `my_rebuild_task` in the Foundation Cluster `my_lts_cluster1`.

SQL

```
CREATE TASK my_rebuild_task TYPE rebuild LOCATION CLUSTER my_lts_cluster1;
```

`check_disk` Task

The `check_disk` task verifies the integrity of data segments on storage nodes by computing checksums. The task compares stored checksums with recomputed ones to detect disk corruption.

`check_disk` Options (`task_option_map`)

The `check_disk` task supports these options. You must specify these options as a comma-separated list of key-value pairs.

Option Key	Value Type	Description
`fix`	Boolean	If you set this option value to `true`, the task modifies the storage cluster state based on the checksum result. For example, if the task finds a segment that fails the checksum, the task sets its segment state to `DAMAGED`. Otherwise, the task reports any segments that fail the checksum, but it does not alter segment states. The default value is `false`.
`sample`	Boolean	If you set this option value to `true`, the task checks only a sample of data in each segment. The default value is `false`.
`only_unhealthy`	Boolean	If you set this option value to `true`, the task checks only segments that are not in the `INTACT` state. The default value is `false`.
`storage_ids`	List	This option limits the `check_disk` task to only a specific list of storage identifiers. Specify these storage identifiers as a bracketed list, e.g., `[ 'id1', 'id2', ... ]`.

Examples

Perform Checksum on the Specific Node

This example computes the checksum on all segments on storage node `my_node1`.

SQL

```
CREATE TASK TYPE check_disk LOCATION NODE 'my_node1';
```

Perform a Checksum on the Specific Cluster

This example computes a checksum for a sample of blocks in each segment of the storage cluster `my_lts_cluster1` using the `sample = true` key-value pair.

SQL

```
CREATE TASK TYPE check_disk LOCATION CLUSTER my_lts_cluster1 OPTIONS sample = true;
```

Compute Checksum on the Specific Storage Identifier

This example computes a checksum for the segment with the storage identifier `529a2e9d-d06c-46cd-a93b-624d3bed1c08` on the storage node `my_node1`.

SQL

```
CREATE TASK TYPE check_disk LOCATION NODE 'my_node1' OPTIONS storage_ids = [ '529a2
```

`rebalance` Task

Rebalancing evenly distributes data across disks in the system. Execute this task if data is not properly balanced across the system, such as after adding nodes or drives.

For details on performing a `rebalance` task, see **Expand and Rebalance System**.

> The `rebalance` task supports only `SYSTEM` and `CLUSTER` locations. If you attempt the rebalance task on a node, the system returns an error.

Example

This example performs a rebalance task across all foundation clusters.

SQL

```
CREATE TASK TYPE rebalance;
```

CANCEL TASK

Cancels a running task.

Privileges

To cancel a task, you must be assigned the System Administrator role.

Syntax

```
SQL

CANCEL TASK ('task_name' | 'task_id')
```

Parameter	Data Type	Description
`task_name`	string	The name specified on task creation. This applies only if an optional name was created for the task.
`task_id`	string	The unique identifier for the task in the system tables. For rebuild tasks, you can find the unique identifier in the **sys.result_cache** table. For check disk tasks, you can find the unique identifier in the **sys.subtasks** table.

Example

Cancel a task named `my_task`.

```
SQL

CANCEL TASK 'my_task';
```

DROP TASK

Drops orphaned or completed tasks from the system.

The task identifier must be a task Universally Unique IDentifier (UUID). The statement does not accept task names because names are not globally unique. A name becomes reusable after its prior task reaches a terminal status, which makes name-based lookup ambiguous. The terminal task statuses are `FAILED`, `CANCELLED`, `INVALID`, `COMPLETE`, and `QUIESCED`.

By default, when you do not specify any optional keywords, the `DROP TASK` SQL statement drops a single leaf task. The database throws an error if the task has children or is not in a terminal status.

Privileges

To drop a task, you must be assigned the System Administrator role.

If you have the System Administrator role, you can drop any task. If you are not an administrator, you can drop COALESCE tasks for which you have `CREATE` privileges or pipeline tasks for which you have `EXECUTE` privileges.

Syntax

SQL

```
DROP TASK [ IF EXISTS ] uuid [ CASCADE ] [ FORCE ]
```

Parameter	Data Type	Description
`uuid`	string	The task UUID, which is the unique identifier for the task in the system tables. The statement does not accept task names.

The optional keywords are:

- `IF EXISTS` — Prevents the statement from returning an error when the task is not found.
- `CASCADE` — Drops the entire subtree rooted at the specified task. All tasks in the subtree must be in a terminal status.
- `FORCE` — Skips the terminal-status checks. You can combine the `FORCE` with `CASCADE` keywords to force the drop of an entire subtree regardless of task status.

Generally, terminal tasks automatically roll off the system, so executing the `DROP TASK` SQL statement is not necessary. However, the `FORCE` and `CASCADE` keywords can help you drop orphaned tasks from the system in the event of an internal lockup. Be aware that this statement can unintentionally delete the state of in-progress tasks.

Examples

Drop a Single Leaf Task

This example drops a single leaf task using its UUID.

```
SQL
DROP TASK '529a2e9d-d06c-46cd-a93b-624d3bed1c08';
```

Drop a Task Only If It Exists

This example drops a task and does not return an error if the system cannot find it.

```
SQL
DROP TASK IF EXISTS '529a2e9d-d06c-46cd-a93b-624d3bed1c08';
```

Drop a Task and Its Subtree

This example drops the specified task and all tasks in its subtree. All tasks in the subtree must be in a terminal status.

```
SQL
DROP TASK '529a2e9d-d06c-46cd-a93b-624d3bed1c08' CASCADE;
```

Force the Drop of a Task Subtree

This example drops the specified task and its entire subtree, regardless of the task status.

```
SQL
DROP TASK '529a2e9d-d06c-46cd-a93b-624d3bed1c08' CASCADE FORCE;
```

Related Links

System Catalog

Guide to Rebuilding Segments

© 2026 Ocient All Rights Reserved

Machine Learning Models

OcientML™ functionality enables you to create a machine learning model, rename the model, export the syntax for the model creation, retrain the model, execute a query against the model, and drop the model.

CREATE MLMODEL

Train a new machine learning model of type `<model type>` on the result set returned by the SQL SELECT statement. After the database creates the model, `<model name>` becomes a callable function in SQL SELECT statements.

Syntax

```SQL
CREATE [ OR REPLACE ] MLMODEL <model_name> TYPE <model_type>
    ON( <SQL SELECT statement> ) [options(<option_list>)]
```

model_name

Parameter	Data Type	Description
`model name`	`VARCHAR`	The name of the model to create.

model_type

Parameter	Data Type	Description
`model type`	`VARCHAR`	The type of machine learning model to create.

These models are supported. You can find full descriptions of each model in **Regression Models**, **Classification Models**, **Clustering and Dimension Reduction Models**, **Ensemble Models**, or **Other Models**.

- `SIMPLE LINEAR REGRESSION`

© 2026 Ocient, Inc.

- `MULTIPLE LINEAR REGRESSION`
- `POLYNOMIAL REGRESSION`
- `LINEAR COMBINATION REGRESSION`
- `VECTOR AUTOREGRESSION`
- `KMEANS`
- `KNN (K Nearest Neighbors)`
- `LOGISTIC REGRESSION`
- `NAIVE BAYES`
- `NONLINEAR REGRESSION`
- `FEEDFORWARD NETWORK`
- `PRINCIPAL COMPONENT ANALYSIS`
- `LINEAR DISCRIMINANT ANALYSIS`
- `SUPPORT VECTOR MACHINE`
- `DECISION TREE`
- `GAUSSIAN MIXTURE MODEL`
- `ASSOCIATION RULES`
- `GRADIENT BOOSTED TREES`
- `REGRESSION TREE`
- `BAGGING`
- `BOOSTING`
- `STACKING`

option_list

Parameter	Data Type	Description
`option_list`	`VARCHAR`	The options for the specified machine learning model that is specified as a comma-separated list in the format: `<option name 1> -> <value 1>, <option name 2> -> <value 2>`, and so on. Names and values must be all enclosed in single quotes and are case sensitive with the exception that Boolean values can be `true`, `false`, `TRUE`, or `FALSE`. Refer to the respective model for the full options list. Example options list:

Parameter	Data Type	Description
		`options( 'yIntercept' -> '10', 'metrics' -> 'true' )`

The SQL SELECT statement that serves as the basis for the model must return rows that fit the specified requirements of the model. For example, in multiple linear regression, the first N columns are the independent variables, and the last column is the dependent variable.

> You cannot create a machine learning model with an existing schema and name combination.

Example

Assume you created the `mldata` table that contains the data for the model. Then, you can create the `my_model` machine learning model based on that data.

SQL

```
CREATE MLMODEL my_model
TYPE SIMPLE LINEAR REGRESSION ON (
  SELECT
    x1,
    y
  FROM mldata
)
options(
  'yIntercept' -> '10',
  'metrics' -> 'true'
);
```

ALTER MLMODEL RENAME

Rename a machine learning model. Use the `IF EXISTS` clause to ignore any models that do not exist.

Required Privileges

To rename a view, you must have the `ALTER MLMODEL` privilege for the model.

The Ocient® System requires these privileges if this statement includes a change to the schema:

- `VIEW` privilege on the current schema of the model
- `VIEW MLMODEL` and `CREATE MLMODEL` privileges on the target schema (if the schema already exists)
- `CREATE MLMODEL` privilege on the database (if the schema does not exist)

Syntax

```
SQL

ALTER MLMODEL [IF EXISTS] <model name> RENAME TO <new model name>
```

Parameter	Data Type	Description
`model name`	`VARCHAR`	The name of the model to rename.
`new model name`	`VARCHAR`	The new name of the model.

Example

```
SQL

ALTER MLMODEL my_model RENAME TO my_model_slr;
```

DROP MLMODEL

Drop a machine learning model. Use the `IF EXISTS` clause to ignore any models that do not exist.

Syntax

```
SQL

DROP MLMODEL [IF EXISTS] model_name [, ...]
```

Parameter	Data Type	Description
`model name`	`VARCHAR`	The name of the model to drop. You can drop multiple models by specifying additional names and separating each with commas.

Example

Drop the machine learning model `my_model`.

SQL

```
DROP MLMODEL my_model;
```

Drop multiple machine learning models.

SQL

```
DROP MLMODEL my_model1, my_model2;
```

EXPORT MLMODEL

Return the SQL statement that can recreate the machine learning model.

Syntax

SQL

```
EXPORT MLMODEL <model name>
```

Parameter	Data Type	Description
`model name`	`VARCHAR`	The name of the model to create.

Example

```sql
EXPORT MLMODEL my_model;
```

Output

```sql
export
-------------------------------------------
CREATE MLMODEL "my_schema"."my_model" TYPE simple linear regression ON(select x, y

Fetched 1 row
```

The output SQL statement includes the schema explicitly.

REFRESH MLMODEL

Retrain a machine learning model without changing any model options.

Syntax

```sql
REFRESH MLMODEL <model name>
```

Parameter	Data Type	Description
`model name`	`VARCHAR`	The name of the model to retrain.

Example

```sql
REFRESH MLMODEL my_model;
```

Related Links

Machine Learning in Ocient

Machine Learning Model Functions

Regression Models

Classification Models

Clustering and Dimension Reduction Models

Other Models

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Users, Groups, and Service Classes

Users, Groups, and Service Classes are managed using DDL SQL statements that allow administrators to create, drop, or alter each type of database object. These commands also allow administrators to associate service classes with groups and control service class settings for workload management.

USER

CREATE USER

`CREATE USER` creates a new user, which is scoped to the database of the active connection processing the DDL statement. This means that usernames can be reused across different databases.

Syntax

SQL

```
CREATE USER [ IF NOT EXISTS ] fully_qualified_username <user_definition> [, ...] ]

<user_definition> ::=
    PASSWORD [=] 'password_string' |
    [ FIRST_NAME [=] 'first_name_string' ] |
    [ LAST_NAME [=] 'last_name_string' ] |
    [ EMAIL [=] 'email_string'
```

Parameter	Data Type	Description
`fully_qualified_username`	string	A fully qualified username (FQUN) that follows the pattern `user_name@database_name` and uniquely identifies a user in Ocient®. An FQUN can reference a user associated with a different database than the one you are connected to. For FQUN examples, see the table below.

If you specify only a `user_name` without a `database_name`, Ocient assumes the user is associated with the database of the active connection.

© 2026 Ocient, Inc.

user id	user name	database name	fully qualified user name (FQUN)
000	alice	db1	alice@db1
001	alice	db2	alice@db2
002	jimmy	system	jimmy@system

Define User Parameters (`<user_definition>`)

You can apply the following optional parameters when creating a new user object.

Parameter	Data Type	Description
`password_string`	string	A password for the user account. This parameter is required.
`first_name_string`	string	Optional. The first name of the user.
`last_name_string`	string	Optional. The last name of the user.
`email_string`	string	Optional. The email of the user.

Example

This example creates a new user named `jmack@test-database` with the password `pass1234`.

SQL

```
CREATE USER
    "jmack@test-database" PASSWORD = 'pass1234',
    FIRST_NAME = 'Johnny',
    LAST_NAME = 'Mack';
```

DROP USER

`DROP USER` removes an existing user from the system.

To remove a user, you must possess the `DROP USER` privilege for the user.

© 2026 Ocient, Inc.

SQL

```
DROP USER [ IF EXISTS ] fully_qualified_username [,...]
```

Parameter	Data Type	Description
`fully_qualified_username`	string	A fully qualified username (FQUN) that follows the pattern `user_name@database_name` and uniquely identifies a user in Ocient. You can drop multiple users by specifying additional usernames and separating each with commas. An FQUN can reference a user associated with a different database than the one you are connected to.

If you specify only a `user_name` without a `database_name`, Ocient assumes the user is associated with the database of the active connection.

Example

This example removes a user named `jmack@test-database`.

SQL

```
DROP USER "jmack@test-database";
```

This example drops multiple users.

SQL

```
DROP USER "jmack@test-database", "jcollins@test-database";
```

ALTER USER

Change the status of a user using the `ALTER USER` SQL statement.

Required Privileges

To change the state of a user, you must have `ALTER` privileges on the user or one of these roles:

- System Administrator Role
- Security Administrator Role
- Administrator Role on the database of the user

> Users cannot change their own state.

Syntax

```
SQL

ALTER USER <user_name> { ENABLE | DISABLE | PASSWORD_EXPIRED }
```

Parameter	Data Type	Description
`user_name`	string	The name of the user.

Examples

Enable a User

Enable a user named `example_user`.

```
SQL

ALTER USER example_user ENABLE;
```

Disable a User

Disable a user named `example_user`.

```
SQL

ALTER USER example_user DISABLE;
```

Expire the Password of a User

Change the `example_user` user to the `password_expired` state. This action requires the user to update their password on their next successful login.

```
SQL

ALTER USER example_user PASSWORD_EXPIRED;
```

ALTER USER SET

Changes the password of an existing user.

Syntax

```
SQL

ALTER USER [ IF EXISTS ] fully_qualified_username SET PASSWORD = password_string
```

Parameter	Data Type	Description
`fully_qualified_username`	string	A fully qualified username (FQUN) that follows the pattern `user_name@database_name` and uniquely identifies a user in Ocient. An FQUN can reference a user associated with a different database than the one you are connected to.
`password_string`	string	A password for the user account. There are no restrictions on the string used for a password.

If you specify only a `user_name` without a `database_name`, Ocient assumes the user is associated with the database of the active connection.

Example

This example changes the password of user `"jmack@test-database"` to `'newpass'`.

```
SQL

ALTER USER "jmack@test-database" SET PASSWORD = 'newpass';
```

© 2026 Ocient, Inc.

GROUP

CREATE GROUP

`CREATE GROUP` creates a new group. The name must be distinct from the name of any existing group in the database.

```
SQL

CREATE GROUP [ IF NOT EXISTS ] group_name
```

Parameter	Data Type	Description
`group_name`	string	A group name unique from any existing group in the database.

Example

This example creates a new group named `group1`.

```
SQL

CREATE GROUP "group1";
```

DROP GROUP

`DROP GROUP` removes an existing group from the system.

To remove a group, you must possess the `DROP GROUP` privilege for the group.

Syntax

```
SQL

DROP GROUP [ IF EXISTS ] group_name [, ...]
```

Parameter	Data Type	Description
`group_name`	string	The identifier of the group to drop. You can drop multiple groups by specifying additional

Parameter	Data Type	Description
		group names, separated by commas.

Example

This example removes a group named `group1`.

SQL

```
DROP GROUP "group1";
```

This example removes multiple groups.

SQL

```
DROP GROUP "group1", "group2";
```

ALTER GROUP

ALTER GROUP USER

`ALTER GROUP` adds or removes users from the group using the `USER` keyword.

Syntax

SQL

```
ALTER GROUP [ IF EXISTS ] groupname { ADD | DROP } USER username [, ...]
```

Parameter	Data Type	Description
`group_name`	string	The identifier of the group to alter.
`username`	string	A username that uniquely identifies a user in the specified group.

If you specify only a `user_name` without a `database_name`, Ocient assumes the user is associated with the database of the active connection.

Example

This example adds `user1` to an existing group named `group1`.

```
SQL

ALTER GROUP "group1" ADD USER "user1";
```

ALTER GROUP RENAME

`ALTER GROUP` renames an existing group by including the `RENAME TO` keyword.

Syntax

```
SQL

ALTER GROUP [ IF EXISTS ] group_name RENAME TO new_group_name
```

Parameter	Data Type	Description
`group_name`	string	The identifier of the group to alter.
`new_group_name`	string	The new name for the group.

Example

This example renames an existing group `group1` to `group2`.

```
SQL

ALTER GROUP "group1" RENAME TO "group2";
```

ALTER GROUP SET SERVICE CLASS

Set a service class for the specified group.

Syntax

© 2026 Ocient, Inc.

```sql
ALTER GROUP group_name SET SERVICE CLASS sc_name
```

Parameter	Data Type	Description
`group_name`	string	The identifier of the group to alter.
`sc_name`	string	The new name of the service class to assign to the group. To unset a service class from a group and restore defaults, specify “DEFAULT” as the service class name.

Example

```sql
ALTER GROUP "wlm_test" SET SERVICE CLASS "high_priority";
```

ALTER GROUP ALTER SECURITY

Set the security settings at the group level using the `ALTER GROUP ALTER SECURITY` SQL statement. Replace `<security_setting>` with the security setting and `<value>` with the value.

Syntax

```sql
ALTER GROUP group_name ALTER SECURITY <security_setting> [=] <value>
```

Parameter	Data Type	Description
`group_name`	string	The identifier of the group for setting security settings.
`security_setting`	string	The security setting with values: * `password_minimum_length` * `password_complexity_level` * `password_no_repeat_count` * `password_lifetime_days` * `password_invalid_attempt_limit` For details about these values, see **Database Password Security Settings**.

Parameter	Data Type	Description
`value`	numeric	An integer to represent one of the security settings. For details about this value, see **Database Password Security Settings**.

Example

Set the password lifetime to 10 days for the group `example_group`.

SQL

```
ALTER GROUP example_group
    ALTER SECURITY password_lifetime_days = 10;
```

SERVICE CLASS

CREATE SERVICE CLASS

A service class defines a set of limits on various system parameters.

The Ocient System applies service classes per group. By default, all groups are in the `DEFAULT` service class, which has no limits.

If a user belongs to multiple groups with different service classes, the system uses the first service class alphabetically by service class name.

Syntax

SQL

```
CREATE SERVICE CLASS [ IF NOT EXISTS ] sc_name [ <service_class_definition> [, ...] ]

<service_class_definition> ::=
    MAX_TEMP_DISK_USAGE [=] percentage |
    MAX_ELAPSED_TIME [=] time |
    MAX_CONCURRENT_QUERIES [=] queries |
    MAX_ROWS_RETURNED [=] rows |
    MINIMIZE_QUERY_DEBUG_RECORDS [=] minimize_query_debug_records |
    SCHEDULING_PRIORITY [=] priority |
    CACHE_MAX_BYTES [=] cache_max_bytes |
    CACHE_MAX_TIME [=] cache_max_time |
    MAX_ELAPSED_TIME_FOR_CACHING [=] max_elapsed_time_for_caching |
    MAX_COLUMNS_IN_RESULT_SET [=] max_columns_in_result_set |
    PRIORITY_ADJUSTMENT_FACTOR [=] priority_adjustment_factor |
    PRIORITY_ADJUSTMENT_TIME [=] priority_adjustment_time |
    MIN_PRIORITY [=] priority |
    MAX_PRIORITY [=] priority |
    STATEMENT_TEXT { LIKE | REGEX } [=] statement_text_pattern
```

Parameter	Data Type	Description
`sc_name`	string	The identifier of the service class.

Define Service Classes (`<service_class_definition>`)

When you create a new service class, the Ocient System sets any omitted service class definitions to the maximum or least restrictive value. Each definition should be comma-separated. When you alter a service class definition, the system modifies only the definition you specify.

See the **Workload Management Walkthrough** for more details.

Parameter	Description	Values
`MAX_TEMP_DISK_USAGE`	Limit the percentage of temporary disk space used by the total service class relative to the amount of remaining free space. If all running queries for a particular	A percentage integer from 0-100.

Parameter	Description	Values
	service class exceed the given percentage, queries will fail.	
`MAX_ELAPSED_TIME`	Limit query runtime. The database process kills any queries that exceed this limit.	-1 for unlimited, or a positive integer for the number of seconds.
`MAX_CONCURRENT_QUERIES`	Limit the maximum number of queries that can run concurrently for a given service class. The database process queues additional queries.	-1 for unlimited, or a positive integer for the number of queries.
`MAX_ROWS_RETURNED`	Limit number of rows returned by a query. The database process kills queries that exceed this limit.	-1 for unlimited, or a positive integer for the number of rows.
`MINIMIZE_QUERY_DEBUG_RECORDS`	Stop the load of the `sys.completed_operator_instances` system catalog table and the generation of specific debug logs associated with the query.	Boolean `true` for minimizing the capture of metadata and logs, and `false` or unset (default) for capturing this data.
`SCHEDULING_PRIORITY`	Limit initial effective query priority.	Any decimal value between the `MIN_PRIORITY` and `MAX_PRIORITY`.
`CACHE_MAX_BYTES`	Maximum number of bytes in a result set if the result set can be stored in the cache. This value is the number of bytes in the wire-protocol representation, or what is sent to a client.	-1 for none, or a positive integer for the number of bytes.
`CACHE_MAX_TIME`	Maximum time rows are cached.	-1 for none, or a positive integer for the number of seconds.
`MAX_ELAPSED_TIME_FOR_CACHING`	Maximum elapsed time for caching. If you set this parameter, queries that exceed maximum elapsed time return no results, but continue to run in the background until either the query reaches the maximum elapsed time for caching or the query successfully completes and caches the result set. For details, see documentation.	-1 for none, or a positive integer for the number of seconds.
`MAX_COLUMNS_IN_RESULT_SET`	Limit number of columns returned by a query. The database process kills queries that exceed this limit.	-1 for unlimited, or a positive integer for the number of columns.

Parameter	Description	Values
`PRIORITY_ADJUSTMENT_FACTOR`	Percentage amount to adjust query priority. You can override this limit at the query level or the session level. Formula is: `(new_priority = current_priority * priority_adjustment_factor)`	0 to disable dynamic priority adjustment (default), or a positive double value that is greater than 0. Values less than 1 decrease priority over time until the priority reaches the service class minimum priority, while values greater than 1 increase priority over time until the priority reaches the service class maximum priority.
`PRIORITY_ADJUSTMENT_TIME`	Time period that indicates the frequency for the adjustment of query priority during execution of the query. You can override this limit at the query level or the session level.	0 to disable dynamic priority adjustment (default), or a positive integer (unsigned 32-bit integer) for the number of seconds that is greater than 0.
`MIN_PRIORITY`	Limit the minimum query priority. The current effective priority cannot be smaller than this value.	0 (default) or any decimal value (double) greater than 0.
`MAX_PRIORITY`	Limit the maximum query priority. The current effective priority cannot be greater than this value.	-1 (default) that indicates infinity or any decimal value (double) greater than 0.
`STATEMENT_TEXT`	The system uses the specified service class for any queries that match the specified text pattern. The system uses service classes with statement text before any service classes that lack statement text. If there are multiple service classes with statement text available, the system attempts to match them in alphabetical order. This matching allows you to set up service classes so that the system can assign certain query types automatically to higher-priority service classes.	A string that uses either LIKE or REGEX pattern matching. For example, `STATEMENT_TEXT REGEX ".*my_table.*|.*your_table.*"` would match any queries that include the strings `my_table` or `your_table`.
`LOW_LATENCY`	Optional settings that enable you to manage low-latency service classes.	Boolean `true` or `false`, or a Key-Value setting map.

Parameter	Description	Values
`MEMORY_OPTIMAL_STRATEGY`	Direct the Ocient System to reduce query memory usage, even if the operation might increase execution time. Contact Ocient Support when you change this parameter.	Boolean `true` instructs the system to apply stricter scheduling and execution rules to minimize memory consumption and avoid excessive concurrent usage. Otherwise, setting this parameter to `false` instructs the system to disable such strategies, even if the service class default enables them. By default, this parameter is unset. If you leave this parameter unset, the system reverts to higher-level default behavior or other service class settings.

Example

This example creates a service class named `sc_name`.

SQL

```
CREATE SERVICE CLASS sc_name
    MAX_TEMP_DISK_USAGE = 80,
    MAX_ELAPSED_TIME = 100,
    MAX_CONCURRENT_QUERIES = 10,
    MAX_ROWS_RETURNED = 100,
    SCHEDULING_PRIORITY = 5.0,
    CACHE_MAX_BYTES = 1000,
    CACHE_MAX_TIME = 25,
    MAX_ELAPSED_TIME_FOR_CACHING = 50;
```

DROP SERVICE CLASS

Remove an existing service class from the system.

When dropping a service class, you can use the `FORCE` flag to drop a service class and unset it from any dependent groups. If `FORCE` is not used, the function throws an error if there are dependent groups.

Syntax

```
SQL

DROP SERVICE CLASS sc_name [ IF EXISTS ] [ FORCE ]
```

Parameter	Data Type	Description
`sc_name`	string	The identifier of the service class.

Example

```
SQL

DROP SERVICE CLASS "my_sc";
```

ALTER SERVICE CLASS

ALTER SERVICE CLASS RENAME

Rename a service class.

Syntax

```
SQL

ALTER SERVICE CLASS [ IF EXISTS ] sc_name RENAME TO new_name
```

Parameter	Data Type	Description
`sc_name`	string	The old identifier of the service class.
`new_name`	string	The new identifier for the `sc_name` service class.

Example

```
SQL

ALTER SERVICE CLASS "my_sc" RENAME TO "my_new_sc";
```

ALTER SERVICE CLASS SET

Alter a service class with new parameter definitions by using the `SET` keyword.

Syntax

SQL

```
ALTER SERVICE CLASS [ IF EXISTS ] sc_name
   SET <service_class_definition> [, ...]

<service_class_definition> ::=
  MAX_TEMP_DISK_USAGE [=] percentage |
  MAX_ELAPSED_TIME [=] time |
  MAX_CONCURRENT_QUERIES [=] queries |
  MAX_ROWS_RETURNED [=] rows |
  SCHEDULING_PRIORITY [=] priority |
  CACHE_MAX_BYTES [=] cache_max_bytes |
  CACHE_MAX_TIME [=] cache_max_time |
  MAX_ELAPSED_TIME_FOR_CACHING [=] max_elapsed_time_for_caching |
  MAX_COLUMNS_IN_RESULT_SET [=] max_columns_in_result_set |
  PRIORITY_ADJUSTMENT_FACTOR [=] priority_adjustment_factor |
  PRIORITY_ADJUSTMENT_TIME [=] priority_adjustment_time |
  MIN_PRIORITY [=] priority |
  MAX_PRIORITY [=] priority |
  STATEMENT_TEXT { LIKE | REGEX } [=] statement_text_pattern
```

Parameter	Data Type	Description
`sc_name`	string	The identifier of the service class.

For the service class definition parameters `<service_class_definition>`, see **Define Service Classes**.

Examples

This example changes the value of `MAX_ROWS_RETURNED` to `51` on the service class named `sc_name`.

```sql
ALTER SERVICE CLASS "sc_name" SET MAX_ROWS_RETURNED = 51;
```

This example makes dynamic priority adjustments at the service class level for a priority adjustment time of 15 seconds, priority adjustment factor of 0.75, minimum priority of 2.0, and maximum priority of 5.0.

```sql
ALTER SERVICE CLASS "sc_name" SET PRIORITY_ADJUSTMENT_TIME = 15;
ALTER SERVICE CLASS "sc_name" SET PRIORITY_ADJUSTMENT_FACTOR = 0.75;
ALTER SERVICE CLASS "sc_name" SET MIN_PRIORITY = 2.0;
ALTER SERVICE CLASS "sc_name" SET MAX_PRIORITY = 5.0;
```

This example sets the service class `sc_name` to handle queries matching the regular expression `.*my_table.*`.

```sql
ALTER SERVICE CLASS "sc_name" SET STATEMENT_TEXT REGEX ".*my_table.*";
```

ALTER SERVICE CLASS RESET

Alter a service class by restoring default values for one or more specified settings.

Syntax

SQL

```
ALTER SERVICE CLASS [ IF EXISTS ] sc_name RESET <service_class_definition> [, ...]

<service_class_definition> ::=
  NAME [=] name |
  MAX_TEMP_DISK_USAGE [=] percentage |
  MAX_ELAPSED_TIME [=] time |
  MAX_CONCURRENT_QUERIES [=] queries |
  MAX_ROWS_RETURNED [=] rows |
  SCHEDULING_PRIORITY [=] priority |
  CACHE_MAX_BYTES [=] cache max bytes |
  CACHE_MAX_TIME [=] cache max time |
  MAX_ELAPSED_TIME_FOR_CACHING [=] max elapsed time for caching |
  MAX_COLUMNS_IN_RESULT_SET [=] max columns in result set |
  PRIORITY_ADJUSTMENT_FACTOR [=] priority adjustment factor |
  PRIORITY_ADJUSTMENT_TIME [=] priority adjustment time |
  MIN_PRIORITY [=] priority |
  MAX_PRIORITY [=] priority |
  STATEMENT_TEXT { LIKE | REGEX } [=] statement_text_pattern
```

Parameter	Data Type	Description
`sc_name`	string	The identifier of the service class.

For descriptions of the `<service_class_definition>` parameters, see **Defining Service Classes**.

Example

This example resets the value of `MAX_ROWS_RETURNED` on the service class named `sc_name`.

SQL

```
ALTER SERVICE CLASS "sc_name" RESET MAX_ROWS_RETURNED;
```

ALTER QUERY

Set the priority of a SQL query that is running. This priority takes effect immediately. The priority must remain within service class priority limits. The Ocient System overrides other preset priorities for this query, including any existing dynamic priority adjustments.

Syntax

```
SQL

ALTER QUERY query_id SET PRIORITY [=] priority
```

Parameter	Data Type	Description
`query_id`	numeric	The identifier of the SQL query that is running.
`priority`	numeric	The priority of the specified SQL query.

Example

This example sets the priority of the SQL query `34566` to `2`.

```
SQL

ALTER QUERY 34566 SET PRIORITY 2;
```

Related Links

Manage Users, Groups, and Roles

Data Control Language (DCL) Statement Reference

Database Password Security Settings

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Integrity and Storage

Ocient® provides several capabilities to check on the quality of data on a node and to quarantine segment groups that are problematic for any reason. Also, you can manage pages by converting them to segments. Related to data integrity is the ability to rebuild tasks. For details, see **Distributed Tasks**.

ALTER SEGMENT QUARANTINE

`ALTER SEGMENT QUARANTINE` isolates segment groups from segment generation or queries. Use this SQL statement if segments are causing unexpected results or crashes. The segments can be un-quarantined by setting `quarantine_level` to `none`.

Syntax

SQL

```
-- To quarantine a single segment group:
ALTER SEGMENT QUARANTINE FOR { <quarantine_level> }
    WHERE SEGMENT_GROUP_ID = seg_id

-- To quarantine multiple segment groups:
ALTER SEGMENT QUARANTINE FOR { <quarantine_level> }
    WHERE SEGMENT_GROUP_ID in ( seg_id [,...] )

<quarantine_level> ::=
segment_generation | query | all | none
```

Parameter	Data type	Description
`seg_id`	numeric	The identification numbers for segment groups. Depending on your `ALTER SEGMENT QUARANTINE` statement, the Ocient System quarantines the specified segment groups for having an existing quarantine removed. For details about finding abnormal segments, see the **Guide to Rebuilding Segments**.

© 2026 Ocient, Inc.

Using Quarantine Levels (`<quarantine_level>`)

You can place quarantines at these segment levels.

Level	Description
`segment_generation`	Isolates the specified segments from segment generation. This statement prevents corrupt data from causing problems while still leaving the data available for future analysis.
`query`	Isolates segment groups from subsequent queries. Data that is quarantined at the query level is not returned by database queries. This can be useful to prevent corrupt data from causing query problems while leaving it available for future analysis.
`all`	Isolates the specified segments from segment generation and subsequent queries.
`none`	Removes any quarantines from the specified segments.

Examples

This example quarantines a single segment group from `segment_generation`.

SQL

```
ALTER SEGMENT QUARANTINE FOR segment_generation WHERE SEGMENT_GROUP_ID = 123456789;
```

This example quarantines multiple segment groups from `segment_generation`.

SQL

```
ALTER SEGMENT QUARANTINE FOR segment_generation WHERE SEGMENT_GROUP_ID IN (1,2,3,4,
```

This example removes a quarantine from a single-segment group.

SQL

```
ALTER SEGMENT QUARANTINE FOR none WHERE SEGMENT_GROUP_ID = 123456789;
```

This example removes a quarantine from multiple segment groups.

© 2026 Ocient, Inc.

```sql
ALTER SEGMENT QUARANTINE FOR none WHERE SEGMENT_GROUP_ID IN (1,2,3,4,5);
```

This example quarantines at the `query` level.

```sql
ALTER SEGMENT QUARANTINE FOR query WHERE SEGMENT_GROUP_ID = 123456789;
```

This example removes a quarantine at the `query` level.

```sql
ALTER SEGMENT QUARANTINE FOR none WHERE SEGMENT_GROUP_ID = 123456789;
```

This example quarantines at both the query and segment-group levels.

```sql
ALTER SEGMENT QUARANTINE FOR all WHERE SEGMENT_GROUP_ID = 123456789;
```

DRAIN PAGES

The `DRAIN PAGES` SQL statement instructs the Ocient System to immediately convert buffered pages into segments on one or more Loader Nodes. During normal operation, the system converts pages to segments automatically based on internal thresholds and timeouts, so the `DRAIN PAGES` statement is usually needed only in special circumstances. For example, draining pages can be useful before performing planned maintenance on Loader Nodes.

The SQL statement targets all Loader Nodes by default. You can optionally scope the operation to a specific storage scope or table, and you can restrict it to a subset of Loader Nodes.

© 2026 Ocient, Inc.

⚠ The `DRAIN PAGES` statement forces the immediate restructuring of buffered data into durable segments, which can affect system resource utilization and loading performance. Contact Ocient Support before using this SQL statement.

Required Privileges

- To drain all pages or drain pages for a specific storage scope, you must have `UPDATE` privileges on the system.
- To drain pages for a specific table, you must have `UPDATE` privileges on the system or `DELETE` privileges on the database that contains the table.

Syntax

SQL

```
DRAIN PAGES
    [ FOR { SCOPE UUID = scope_uuid
          | TABLE UUID = table_uuid } ]
    [ WITH LOADERS loader_name [, loader_name [, ...] ] ]
```

Parameter	Type	Description
`scope_uuid`	string	The Universally Unique IDentifier (UUID) of a storage scope. When you specify this value, the system converts only pages that belong to this storage scope. This value is mutually exclusive with the `table_uuid` value. To find storage scope UUIDs, query the **sys.storage_scopes** system catalog table.
`table_uuid`	string	The UUID of a table. When you specify this value, the system converts only pages that belong to this table. This value is mutually exclusive with the `scope_uuid` value. To find table UUIDs, query the **sys.tables** system catalog table.
`loader_name`	string	The name of a Loader Node. When you specify one or more loader names, the system sends the drain request only to those nodes. By default, the system sends the request to all Loader Nodes.

Examples

Drain All Pages

This example converts all buffered pages to segments across every Loader Node.

```
SQL

DRAIN PAGES;
```

Drain Pages for a Storage Scope

This example converts only pages associated with the storage scope identified by the UUID `a1b2c3d4-e5f6-7890-abcd-ef1234567890`.

```
SQL

DRAIN PAGES
    FOR SCOPE UUID = 'a1b2c3d4-e5f6-7890-abcd-ef1234567890';
```

Drain Pages for a Table

This example converts only pages associated with the table identified by the UUID `12345678-abcd-ef01-2345-6789abcdef01`.

```
SQL

DRAIN PAGES
    FOR TABLE UUID = '12345678-abcd-ef01-2345-6789abcdef01';
```

Drain Pages on Specific Loader Nodes

This example sends the drain request to the `loader-node-1` and `loader-node-2` Loader Nodes.

```
SQL

DRAIN PAGES
    WITH LOADERS 'loader-node-1', 'loader-node-2';
```

Drain Pages for a Table on a Specific Loader Node

This example converts only pages associated with the table identified by the UUID `12345678-abcd-ef01-2345-6789abcdef01` on the `loader-node-1` Loader Node.

SQL

```
DRAIN PAGES
    FOR TABLE UUID = '12345678-abcd-ef01-2345-6789abcdef01'
    WITH LOADERS 'loader-node-1';
```

Related Links

Errors and Warnings

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Query Analysis

You can analyze queries by commands that instruct Ocient® to return plan-related information for a database administrator or analyst to evaluate. The results of these commands are vital in performance optimization activities on an Ocient System.

Ocient supports SQL DDL commands to terminate running queries and distributed tasks. These commands provide administrative control over query execution and are distinct from programmatic cancellation methods in JDBC and pyocient drivers.

EXPLAIN

`EXPLAIN` shows the optimized query plan. The database optimizes the query normally, but it does not execute. Instead, the database returns the query plan in the specified format.

Syntax

```sql
EXPLAIN [ PROTO | JSON | DEBUG ] query
```

Parameter	Data type	Description
`query`	string	A valid SELECT statement on a database table.

This table defines the explain plan formats.

Format	Description
PROTO	Includes the same information as the JSON format, but it is presented in a human-readable format.
JSON	The default option. Returns the optimized query plan in JSON format.
DEBUG	The DEBUG format is more compact but does not include some information about each operator.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Examples

This example shows the optimized plan for the query `SELECT * FROM names` in JSON format.

SQL

```
EXPLAIN SELECT * FROM names;
```

This EXPLAIN example generates a lengthy output showing the execution plan for the query. This execution plan shows how the database scans the tables referenced by the statement and calculates the memory usage and total execution cost.

This example shows the optimized plan for the query `SELECT * FROM names` in DEBUG format, which produces a more compact execution plan.

SQL

```
EXPLAIN DEBUG SELECT * FROM names;
```

Output

Text

```
ROOT_OP @ (0x7eae71515c40), ID: 94921393, md: 2, Out (1): time_x |
  REORDER_OP @ (0x7f4c19ee1c40), ID: 94921392, md: 0, Out (1): time_x | no-op reord
    RENAME_OP @ (0x7f49b1bc2c40), ID: 94921390, md: 2, Out (1): time_x | Old2New: (
      GATHER_OP @ (0x7f49b7886a40), ID: 94921389, md: 2, Out (1): loc.time_x |
        GATHER_OP @ (0x7f49b78866c0), ID: 94921388, md: 2, Out (1): loc.time_x |
          RANDOM_SHUFFLE_OP @ (0x7f486825ba40), ID: 94921394, md: 2, Out (1): loc.t
            INDEX_OP @ (0x7f49b792f040), ID: 94921387, md: 1, Out (1): loc.time_x |
```

EXPLAIN PIPELINE

`EXPLAIN PIPELINE` shows the pipeline the system uses for compiling a query. The system compiles the pipeline for the specified segments, and if no segment is specified, it picks random segments with the tables.

To explain a pipeline, you must be a system-level user or possess the `SELECT TABLE` privilege for the table used in the query.

Syntax

SQL

```
-- To show pipeline for a random segment:
EXPLAIN PIPELINE query

-- To show pipeline for a specific segment:
EXPLAIN PIPELINE FOR SEGMENT 'storage_id' ON query

-- To show pipeline for multiple specific segments:
EXPLAIN PIPELINE FOR SEGMENT IN ('storage_id' [, ...]) ON query
```

Parameter	Data type	Description
`query`	string	A valid SELECT statement on a database table.
`storage_id`	string	A valid segment storage identifier.

Examples

This example shows the pipeline on the query `SELECT * FROM names` for a random segment.

SQL

```
EXPLAIN PIPELINE SELECT * FROM names;
```

This example shows the pipeline for the segment with the identifier `529a2e9d-d06c-46cd-a93b-624d3bed1c08`.

SQL

```
EXPLAIN PIPELINE FOR SEGMENT '529a2e9d-d06c-46cd-a93b-624d3bed1c08'
    ON SELECT * FROM names;
```

This example shows the pipeline for segments with these identifiers: `529a2e9d-d06c-46cd-a93b-624d3bed1c08`, `378eac93-2e3d-4ce6-bf1f-d8b7a78fe585`.

SQL

```
EXPLAIN PIPELINE FOR SEGMENT IN (
    '529a2e9d-d06c-46cd-a93b-624d3bed1c08',
    '378eac93-2e3d-4ce6-bf1f-d8b7a78fe585')
    ON SELECT * FROM names;
```

All the examples generate lengthy output that includes the execution plan for the query for the specified segments. This execution plan shows how the database scans the tables referenced by the statement and calculates the memory usage and total execution cost.

Related Links

SQL Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Query Management

Query management statements let you monitor and control running queries in an Ocient® System. For information about system tables containing query data, see **Monitoring**.

CANCEL

Cancels a running query based on its specific query identifier.

The system cancels query execution at the next available pause point, such as between batches of data processing. If the execution of the `CANCEL` SQL statement fails, you can force the query to terminate immediately by using the **KILL** SQL statement.

To view active queries and their identifiers, see the **sys.queries** table.

A `CANCEL` statement can return these statuses.

Status Message	Description
`OK`	The system successfully terminated the query and restored the resources assigned to its execution.
`ID_NOT_FOUND`	The specified query identifier does not exist in the system. The missing identifier can indicate that the query finished execution before the system could abort it.
`NOT_AUTHORIZED`	The user does not have `UPDATE` privilege for the system level to terminate queries.
`NO_CONNECTION`	The system lost connection to the database while trying to terminate the query.
`NETWORK_COMMS_ERRO R`	Cannot communicate across the distributed system to terminate the query.
`INTERNAL_ERROR`	The system encountered an unexpected error.

Required Privileges

Users must have system-level `UPDATE` privileges to cancel queries.

Syntax

```sql
CANCEL query_id
```

Parameter	Type	Description
`query_id`	string	The Universally Unique IDentifier (UUID) of the query to cancel. Must be a valid UUID string enclosed in single quotes.

Examples

Cancel a specific query with the `9a9eb7e3-3440-4814-871f-7f0584ac2fd8` identifier.

```sql
CANCEL '9a9eb7e3-3440-4814-871f-7f0584ac2fd8';
```

KILL

Kills a running query identified by its UUID. This SQL statement performs a forceful termination of the query, aborting execution regardless of its current stage.

If you want to attempt a more gradual query termination that stops processing at the next pause point, use the **CANCEL** SQL statement.

To view active queries and their identifiers, see the **sys.queries** table.

Status Message	Description
`OK`	The system successfully terminated the query and restored the resources assigned to its execution.
`ID_NOT_FOUND`	The provided query identifier does not exist in the system. The missing identifier can indicate that the query finished execution before the system could abort it.
`NOT_AUTHORIZED`	The user does not have `UPDATE` privilege for the system level to terminate queries.
`NO_CONNECTION`	The system lost connection to the database while trying to terminate the query.

Status Message	Description
`NETWORK_COMMS_ERRO R`	Cannot communicate across the distributed system to terminate the query.
`INTERNAL_ERROR`	The system encountered an unexpected error.

Required Privileges

Users must have system-level `UPDATE` privileges to kill queries.

Syntax

SQL

```
KILL query_id
```

Arguments

Parameter	Type	Description
`query_id`	String	The UUID of the query to kill. Must be a valid UUID string enclosed in single quotes.

Example

Kill a specific query based on the UUID `9a9eb7e3-3440-4814-871f-7f0584ac2fd8`.

SQL

```
KILL '9a9eb7e3-3440-4814-871f-7f0584ac2fd8';
```

Related Links

System Catalog

Data Query Language (DQL) Statement Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Statistics Cache Management

System administrators can invalidate internal statistics caches used by the SQL optimizer. The optimizer then immediately sends new requests to Foundation Nodes to repopulate statistics for the underlying data in tables.

The Ocient® System calculates two primary types of statistics. The system calculates Cardinality Distinct Estimates (CDEs) by performing a union operation on the information stored in the table on Foundation Nodes. Also, the system calculates Probability Density Functions (PDFs) by sampling the table data local to the Foundation Node. Then, the system performs a union operation on the CDEs and PDFs on SQL Nodes when stored in the statistics cache. Notably, the union operation for the PDF is not associative, and its implementation is not deterministic. The union operation for the CDE is associative, commutative, and deterministic.

The use of these SQL statements in system administration settings is rare. Do not use these statements for day-to-day maintenance of Ocient clusters. Mostly, you can use these commands when large amounts of data have been loaded or removed from the Ocient System quickly, and the system administrator wants to recalculate statistics immediately rather than wait for the configured statistics cache refresh timer to reflect the changes. These syntaxes help to stabilize the optimizer for new data distributions quickly.

Invalidate Optimizer Statistics Cache

`INVALIDATE STATS` clears the entire optimizer statistics cache on all SQL Nodes.

If a table has not undergone any row delta after the last statistics cache refresh, the Ocient System does not recalculate statistics by sampling the underlying table. PDF statistics might still change slightly due to non-deterministic union operations. If the table has undergone a row change, the Ocient System recalculates statistics from scratch.

Query performance might be affected until the statistics cache repopulates, specifically for optimization time. For tables that have undergone a row delta, the optimizer temporarily uses the old statistics file again until the recalculated file is available.

Syntax

```sql
INVALIDATE STATS
```

Example

Clear the entire optimizer statistics cache on all SQL Nodes.

```sql
INVALIDATE STATS;
```

Invalidate All Base Statistics

The `INVALIDATE STATS FILES` SQL statement forcibly invalidates the CDE and PDF files of the Foundation Node for all tables regardless of changes in the row delta, which triggers the system to recalculate the statistics.

The optimizer statistics cache continues to use its current value until the Foundation Nodes have finished recalculating the new statistics, after which the cache replaces the table entry with the new statistics. This operation should have minimal impact on query optimization times.

Depending on the size of the system, the system might take minutes to hours to recalculate all statistics on the system fully, and for new statistics to populate in the optimizer cache.

Syntax

```sql
INVALIDATE STATS FILES
```

Example

Invalidate all statistics

```sql
INVALIDATE STATS FILES;
```

Invalidate All Base Statistics For a Specific Table

The `ALTER table_name INVALIDATE STATS` SQL statement forcibly invalidates the CDE and PDF files of the Foundation Node for the specified table, regardless of changes in the row delta, which triggers the system to recalculate the statistics.

The optimizer statistics cache continues to use its current value until the Foundation Nodes have finished recalculating the new statistics, after which the cache replaces the table entry with the new statistics. This operation should have minimal impact on query optimization times.

Syntax

```sql
ALTER TABLE table_name INVALIDATE STATS
```

Parameter	Data Type	Description
`table_name`	string	The name of the table.

Example

Force the system to recalculate the statistics for the table named `test_table`.

```sql
ALTER TABLE test_table INVALIDATE STATS;
```

Related Links

Plan Optimization

Query Performance Optimizations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Control Language (DCL) Statement Reference

In order to assign privileges to users, administrators can use Data Control Language (DCL) statements. You can write and execute DCL statements like any other SQL statements against the Ocient® System and databases. DCL statements allow you to quickly and easily grant or revoke privileges from users. DCL statements work off two main concepts: Grants and Revokes. Statements that begin with `GRANT` give the associated privileges to a user or group. Statements that begin with `REVOKE` remove those privileges from a user or group.

A full list of privileges can be found in the **Ocient Privileges Reference**.

Supported DCL SQL statements are:

- GRANT PRIVILEGE
- REVOKE PRIVILEGE
- GRANT ROLE
- REVOKE ROLE

> When you create an object in the system, database, or schema, the Ocient System automatically grants you all valid privileges for that type of object. For example, if you create a table in a database, the system grants you all privileges that apply to the created table. For details, see **Table Privileges**.
>
> For non-SSO users, the system assigns creator privileges to the user. For SSO users, the system follows specific criteria to assign the privileges. To understand the criteria the system uses, see **User Access Control and Workload Management with SSO**.

Grant Privileges

The `GRANT` statement grants privileges to a user or group. The privileges that you can grant in the `GRANT` statement map to the tables and differ per object. The `WITH GRANT OPTION` keywords specify that the grantee can also grant privileges on that object.

To grant privileges, you must have:

© 2026 Ocient, Inc.

- `VIEW` privileges for the specified user or group being granted new privileges.
- `SYSAUTH` privileges over the system or database object that the specified user or group is being granted privileges on.

Syntax

SQL

```
GRANT {
    { CREATE { <systemObjects> }
        | VIEW { SCHEMA | TABLE | VIEW | QUERIES | REDACTED QUERIES }
        | ALTER { SCHEMA | TABLE | VIEW }
        | DROP { SCHEMA | TABLE | VIEW }
        | CANCEL QUERIES
        | SELECT | DELETE | SYSAUTH | USE | ALL }
        ON SYSTEM
    | { CREATE { <databaseObjects> }
        | VIEW { SCHEMA | TABLE | VIEW  | QUERIES | REDACTED QUERIES }
        | ALTER { SCHEMA | TABLE | VIEW }
        | DROP { SCHEMA | TABLE | VIEW }
        | CANCEL QUERIES
        | SELECT | ALTER | DROP | DELETE | SYSAUTH | USE | ALL }
        ON DATABASE object_name
    | { CREATE { <schemaObjects> | MLMODEL } | VIEW { <schemaObjects> }
        | ALTER { <schemaObjects> }
        | DROP { <schemaObjects> }
        | VIEW | SELECT | ALTER | DROP | DELETE | SYSAUTH | INSERT | ALL }
        ON SCHEMA object_name
    | { VIEW | ALTER | DROP | SYSAUTH | ALL }
        ON { GROUP | USER } object_name
    | { SELECT | DROP | SYSAUTH | ALL }
        ON MLMODEL object_name
    | { VIEW | EXECUTE | ALTER | DROP | SYSAUTH | ALL }
        ON PIPELINE object_name
    | { VIEW | DROP | SYSAUTH | ALL }
        ON PIPELINE FUNCTION object_name
    | { VIEW | SELECT | ALTER | DROP | SYSAUTH | ALL }
        ON VIEW object_name
    | { VIEW | SELECT | ALTER | LOAD | DROP | DELETE | SYSAUTH | INSERT | ALL }
        ON TABLE object_name
    | { VIEW | SELECT | ALL }
        ON TABLE sys.system_catalog_table
 }
    TO { { GROUP | USER } object_name | PUBLIC } [ WITH GRANT OPTION ]

<systemObjects> ::=
    DATABASE | TABLE | VIEW | USER | GROUP
<databaseObjects> ::=
    SCHEMA | TABLE | VIEW | MLMODEL | PIPELINE | PIPELINE FUNCTION | USER | GROUP
```

```
<schemaObjects> ::=
    TABLE | VIEW
```

When you grant any privilege to a user on a system or database object other than the `VIEW` privilege, the database implicitly grants the `VIEW` privilege on the specified object to the user, as well as associated object types (e.g., `DROP VIEW ON DATABASE` also grants `VIEW VIEW ON DATABASE` and `VIEW ON DATABASE`).

Examples

Grant Privilege on a Table

This example grants privileges for the `SELECT` SQL statement on a table to a trusted group.

SQL

```
GRANT SELECT ON TABLE company_data TO GROUP trusted_employees;
```

Grant Privilege on a Database

This example grants the `SELECT` privilege for all tables and views on the `database_name` database to a trusted group.

SQL

```
GRANT SELECT ON DATABASE database_name TO GROUP trusted_employees;
```

Grant Privilege on the System

This example grants the `SELECT` privilege for all tables and views on all databases in the system to a trusted group.

SQL

```
GRANT SELECT ON SYSTEM TO GROUP trusted_employees;
```

For examples of granting privileges for individual users, see **Grant Role Membership**.

Revoke Privileges

The `REVOKE` statement revokes privileges from a user or group.

To revoke privileges, you must have:

- `VIEW` privileges for the specified user or group having their privileges revoked.
- `SYSAUTH` privileges over the system or database object from which the specified user or group has their privileges revoked.

© 2026 Ocient, Inc.

SQL

```
REVOKE [ GRANT OPTION FOR ] {
    { CREATE { <systemObjects> }
        | VIEW { SCHEMA | TABLE | VIEW | QUERIES | REDACTED QUERIES }
        | ALTER { SCHEMA | TABLE | VIEW }
        | DROP { SCHEMA | TABLE | VIEW }
        | CANCEL QUERIES
        | SELECT | DELETE | SYSAUTH | USE | ALL }
        ON SYSTEM
    | { CREATE { <databaseObjects> }
        | VIEW { SCHEMA | TABLE | VIEW | QUERIES | REDACTED QUERIES }
        | ALTER { SCHEMA | TABLE | VIEW }
        | DROP { SCHEMA | TABLE | VIEW }
        | CANCEL QUERIES
        | SELECT | ALTER | DROP | DELETE | SYSAUTH | USE | ALL }
        ON DATABASE object_name
    | { CREATE { <schemaObjects> | MLMODEL } | VIEW { <schemaObjects> }
        | ALTER { <schemaObjects> }
        | DROP { <schemaObjects> }
        | VIEW | SELECT | ALTER | DROP | DELETE | SYSAUTH | INSERT | ALL }
        ON SCHEMA object_name
    | { VIEW | ALTER | DROP | SYSAUTH | ALL }
        ON { GROUP | USER } object_name
    | { SELECT | DROP | SYSAUTH | ALL }
        ON MLMODEL object_name
    | { VIEW | EXECUTE | ALTER | DROP | SYSAUTH | ALL }
        ON PIPELINE object_name
    | { VIEW | DROP | SYSAUTH | ALL }
        ON PIPELINE FUNCTION object_name
    | { VIEW | SELECT | ALTER | DROP | SYSAUTH | ALL }
        ON VIEW object_name
    | { VIEW | SELECT | ALTER | LOAD | DROP | DELETE
        | SYSAUTH | INSERT | ALL }
        ON TABLE object_name
    | { VIEW | SELECT | ALL }
        ON TABLE sys.system_catalog_table
  }
    FROM { [ USER | GROUP ] object_name | PUBLIC }

<systemObjects> ::=
    DATABASE | TABLE | VIEW | USER | GROUP
<databaseObjects> ::=
    SCHEMA | TABLE | VIEW | MLMODEL | PIPELINE | PIPELINE FUNCTION | USER | GROUP
```

```
<schemaObjects> ::=
    TABLE | VIEW
```

When you revoke the `VIEW` privilege on a system or database object, the database implicitly revokes all privileges on the specified object from the user.

If you revoke the `VIEW` privilege on an associated object type, the system revokes the privileges for that object type. For example, grant the `DROP TABLE` privilege to the user by using the `GRANT DROP TABLE ON DATABASE` SQL statement. Due to implicit granting by the system, this action causes the user to have `DROP TABLE`, `VIEW TABLE`, and `VIEW` privileges. If you revoke the `VIEW TABLE` privilege using the `REVOKE VIEW TABLE ON DATABASE` SQL statement, then the system revokes the privileges for the associated type. In this case, the user only retains the `VIEW` privilege on the database.

Example

This example revokes the SELECT privilege on a table from an untrusted group.

SQL

```
REVOKE SELECT ON TABLE company_data FROM GROUP untrusted;
```

For examples of revoking privileges for individual users, see **Revoking Role Membership**.

Ocient Privileges Reference

These tables describe the privilege options on each object in Ocient and the allowed privileges.

As of version 23.0, Ocient DCL has replaced the `TRUNCATE` privilege with the `DELETE` privilege.

Any `TRUNCATE` privileges, whether manually granted or assigned as part of a user role, should automatically convert to `DELETE` privileges following a system upgrade to version 23.0 or later.

Ocient System Privileges

Privilege	Description
CREATE [DATABASE, USER, GROUP]	The privilege to create the specified object type. On creation, the creating user inherits all privileges on the object. In the case of the DATABASE object, this is enforced by granting the creating user the newly created database administrator role as described in the Roles section.
CREATE [SCHEMA, TABLE, VIEW]	The privilege to create the specified object type on any database where the user can connect. On creation, the creating user inherits all privileges on the object. In order to apply the privilege, the user has to be connected to a user-defined database.
DROP [SCHEMA, TABLE, VIEW]	The privilege to drop the specified object type on any database where the user can connect. In order to apply the privilege, the user has to be connected to a user-defined database.
ALTER [SCHEMA, TABLE, VIEW]	The privilege to modify the specified object type on any database where the user can connect. In order to apply the privilege, the user has to be connected to a user-defined database.
VIEW [SCHEMA, TABLE, VIEW]	The privilege to see and read information of the specified object type within the system.
VIEW QUERIES	The privilege to see the full SQL statement and metadata for all queries within the system.
VIEW REDACTED QUERIES	The privilege to see the metadata for all queries with a fully redacted SQL statement within the system.
CANCEL QUERIES	The privilege to cancel or kill running queries within the system.
SELECT	The privilege to read data on any table or view where the user has the VIEW privilege.
SYSAUTH	The privilege to grant and revoke all privileges on the system.
USE	The ability to connect to the system.
DELETE	The privilege to delete data from and truncate tables within the system.

Database Privileges

Privilege	Description
CREATE [SCHEMA, TABLE, VIEW, MLMODEL, PIPELINE, PIPELINE FUNCTION, USER, GROUP]	The privilege to create the specified object type. The user who creates the object inherits all privileges on the object.
DROP [SCHEMA, TABLE, VIEW]	The privilege to drop the specified object type on this database.

Privilege	Description
ALTER [SCHEMA, TABLE, VIEW]	The privilege to modify the specified object type on this database.
VIEW [SCHEMA, TABLE, VIEW]	The privilege to see and read information of the specified object type within the database.
VIEW QUERIES	The privilege to see the full SQL statement and metadata for all queries in this database.
VIEW REDACTED QUERIES	The privilege to see the metadata for all queries with a fully redacted SQL statement in this database.
CANCEL QUERIES	The privilege to cancel or kill running queries in this database.
SELECT	The privilege to read data on any table or view, where the user has the VIEW privilege, in the database.
USE	Ability to connect to the database. Users created in this database are implicitly granted this privilege.
ALTER	The ability to issue ALTER DATABASE statements on this database.
DROP	The ability to drop this database.
SYSAUTH	The ability to grant and revoke all privileges on the database.
DELETE	The privilege to delete data from and truncate tables in the database.

Schema Privileges

Privilege	Description
CREATE [TABLE | VIEW | MLMODEL]	The privilege to create the specified object type.
DROP [TABLE | VIEW]	The privilege to drop the specified object type within the schema.
ALTER [TABLE | VIEW]	The privilege to modify the specified object type within the schema.
VIEW [TABLE | VIEW]	The privilege to see and read information of the specified object type within the schema.
SELECT	The privilege to read data on any table or view, where the user has the VIEW privilege, within the schema.
ALTER	The ability to modify the schema.

Privilege	Description
DROP	The privilege to drop the schema and its contents.
VIEW	The ability to see and read information about the schema.
SYSAUTH	The ability to grant and revoke all privileges on the schema.
INSERT	The privilege to insert data into tables in the schema.
DELETE	The privilege to delete data from and truncate tables in the schema.

Granting privileges using DCL statements promotes an implicit schema to an explicit schema. For details, see **Schemas**.

View Privileges

Privilege	Description
VIEW	Ability to see and read schema information about the view.
SELECT	Ability to read data from the view.
ALTER	Ability to modify the view.
DROP	Ability to drop the view.
SYSAUTH	Ability to grant privileges on the view.

Table Privileges

Privilege	Description
VIEW	Ability to see and read schema information about the table.
SELECT	Ability to read data from the table.
ALTER	Ability to modify the table.
LOAD	Ability to load data into the table.
DROP	Ability to drop the table.
DELETE	Ability to delete and truncate the table.
SYSAUTH	Ability to grant and revoke privileges on the table.

© 2026 Ocient, Inc.

Privilege	Description
INSERT	Ability to insert data into the table.

Machine Learning Model Privileges

Privilege	Description
ALTER	Ability to rename the model.
SELECT	Ability to read data and descriptive data from the model.
DROP	Ability to drop the model.
SYSAUTH	Ability to grant and revoke privileges on the model.

Data Pipeline Privileges

Privilege	Description
VIEW	See and read information schema about a pipeline.
EXECUTE	Start and stop the execution of a pipeline.
ALTER	Rename and replace a pipeline.
DROP	Drop a pipeline.
SYSAUTH	Grant and revoke privileges on a pipeline.

Data Pipeline Function Privileges

Privilege	Description
VIEW	See and read information schema about a data pipeline function in the system catalog tables.
DROP	Drop a data pipeline function.
SYSAUTH	Grant and revoke privileges on a data pipeline function.

User Privileges

Privilege	Description
VIEW	Ability to read information about the user.
ALTER	Ability to modify the user.
DROP	Ability to drop the user.
SYSAUTH	Ability to grant rights and revoke privileges on the user.

Group Privileges

Privilege	Description
VIEW	Ability to read information about the group.
ALTER	Ability to alter the group.
DROP	Ability to drop the group.
SYSAUTH	Ability to grant privileges on the group.

System Catalog and Object or View Visibility

The `sys.privileges` table in the system catalog exposes the privileges in the system. This table displays this information:

- Timestamp of the grant
- Grantor
- Grantee
- Privilege granted
- Object type
- Object id
- Grantable

You can grant and revoke System Catalog `VIEW` and `SELECT` privileges just like ordinary tables. By default, everyone has `VIEW` and `SELECT` privileges on all system catalog tables.

© 2026 Ocient, Inc.

You can see the objects within the system catalog tables only if you have sufficient privileges on those objects. These objects require a `VIEW` or `SELECT` privilege or membership in a group or role to provide visibility. When you execute queries against a system catalog table, you do not see or know the existence of objects to which you do not have access.

Views offer a similar functionality because the database does not check privileges to the underlying tables and views after you create a view. You can create a table with sensitive information and restrict visibility by creating a view on top of the table with only certain rows or columns. Someone else with the `SELECT` privilege to the view can query the view, even without any privileges to the underlying table.

Roles

Similar to groups, users can inherit privileges by being granted one of the predefined roles in Ocient. The names of and privileges assigned to roles in Ocient are predefined by the system. Roles can be applicable to the Ocient System or one of the user-defined databases. The roles in Ocient are as follows:

System Roles

- **Security Administrator** — Can read, create, and modify all users and groups.
- **System Administrator** — Can read, create, and modify system objects such as tables, clusters, databases, etc. Can see and delete any queries system-wide.
- **System Analyst** — Read-only access to the entire system. Can see all queries in the system.

Database Roles

- **Database Administrator** — Can read, create, and modify database objects such as tables, clusters, databases, etc. This role can also create users for the database. Can see and delete all queries in the database.
- **Database Analyst** — Read-only rights to database objects and data within the database. Can see all queries in the database.
- **Public** — Every user who has access to the database has the Public role by default. You can grant other privileges to this role. This role allows the creation of a schema

using the `CREATE SCHEMA` SQL statement and allows the viewing of other roles by default.

For specifics on each Ocient role, see **Default Role Privileges**.

Grant Role Membership

Grants a role to a user or group.

Syntax

```
SQL

GRANT ROLE role_name TO { USER | GROUP } user_or_group_name
```

Parameter	Data Type	Description
`role_name`	string	The name of a role to be granted to the specified user or group. Enclose this string in quotes.
`user_or_group_name`	string	The name of a user or group to be granted the specified role.

Example

This example grants `user1` the system administrator role.

```
SQL

GRANT ROLE "System Administrator" TO USER user1;
```

Revoke Role Membership

Revokes a role from a user or group.

Syntax

SQL

```
REVOKE ROLE role_name FROM { USER | GROUP } user_or_group_name
```

Parameter	Data Type	Description
`role_name`	string	The name of a role to be revoked for the specified user or group. Enclose this string in quotes.
`user_or_group_name`	string	The name of a user or group to have the specified role revoked.

Example

This example revokes the system administrator role from `user1`.

SQL

```
REVOKE ROLE "System Administrator" FROM USER user1;
```

Default Role Privileges

Security Administrator Privileges

Target	Privileges
System	SYSAUTH, CREATE [USER]
Schema	SYSAUTH
Database	SYSAUTH, CREATE [USER, GROUP]
View	SYSAUTH
Table	SYSAUTH
Machine Learning Model	SYSAUTH
Data Pipeline	SYSAUTH
Data Pipeline Function	SYSAUTH
User	VIEW, ALTER, DROP, SYSAUTH

© 2026 Ocient, Inc.

Target	Privileges
Group	VIEW, ALTER, DROP, SYSAUTH

System Administrator Privileges

Target	Privileges
System	USE, CREATE [DATABASE, USER], VIEW QUERIES, CANCEL QUERIES
Schema	VIEW, ALTER, DROP
Database	- USE - ALTER - DROP - CREATE [SCHEMA, TABLE, VIEW, MLMODEL, USER, GROUP, PIPELINE, PIPELINE FUNCTION]
View	VIEW, SELECT, ALTER, DROP
Table	VIEW, SELECT, ALTER, LOAD, DROP, DELETE, INSERT
Machine Learning Model	SELECT, DROP
Data Pipeline	VIEW, EXECUTE, ALTER, DROP
Data Pipeline Function	VIEW, DROP
User	VIEW, ALTER, DROP
Group	VIEW, ALTER, DROP

System Analyst Privileges

Target	Privileges
System	USE, VIEW QUERIES
Schema	VIEW
Database	USE
View	VIEW, SELECT
Table	VIEW, SELECT
Machine Learning Model	SELECT

Target	Privileges
Data Pipeline	VIEW
Data Pipeline Function	VIEW
User	VIEW
Group	VIEW

Database Administrator Privileges

Target	Privileges
Schema	VIEW, ALTER, DROP, SYSAUTH
Database	- USE - ALTER - DROP - CREATE [SCHEMA, TABLE, VIEW, MLMODEL, USER, GROUP, PIPELINE, PIPELINE FUNCTION] - VIEW QUERIES, CANCEL QUERIES - SYSAUTH
View	VIEW, SELECT, ALTER, DROP, SYSAUTH
Table	VIEW, SELECT, ALTER, LOAD, DROP, DELETE, INSERT, SYSAUTH
Machine Learning Model	SELECT, DROP, SYSAUTH
Data Pipeline	VIEW, EXECUTE, ALTER, DROP, SYSAUTH
Data Pipeline Function	VIEW, DROP, SYSAUTH
User	VIEW, ALTER, DROP, SYSAUTH
Group	VIEW, ALTER, DROP, SYSAUTH

Database Analyst Privileges

Target	Privileges
Schema	VIEW
Database	USE, VIEW QUERIES
View	VIEW, SELECT

© 2026 Ocient, Inc.

Target	Privileges
Table	VIEW, SELECT
Machine Learning Model	SELECT
Data Pipeline	VIEW
Data Pipeline Function	VIEW
User	VIEW
Group	VIEW

Public Privileges

Target	Privileges
Database	CREATE SCHEMA

Query Visibility

The visibility of queries depends on which database you are logged into and your assigned privileges. Users with no additional VIEW QUERIES or VIEW REDACTED QUERIES privileges can view only queries that they submit themselves.

The VIEW QUERIES privilege allows access to queries for all users in these system catalog tables:

- `sys.queries`
- `sys.completed_queries`
- `sys.plans`
- `sys.active_query_scheduler_info`
- `sys.all_operator_instances`
- `sys.completed_operator_instances`
- `sys.op_inst_debug_info`

The VIEW REDACTED QUERIES privilege allows access to queries for all users in these system catalog tables, but with a redacted SQL statement for queries from other users:

- `sys.queries`
- `sys.completed_queries`

For the initial grant of query privileges, the grantor must have the `Database Administrator` or `System Administrator` role. Use the `WITH GRANT OPTION` keywords to allow a user to further grant this privilege to other users and groups.

Login Impacts on Query Visibility

Along with having role privileges, users also must be logged into the appropriate database to view queries:

- If the user logs in to the `SYSTEM` database, the user can see all queries from all databases.
- If the user logs in to a database other than `SYSTEM`, the user can see queries in only that database.

Related Links

Database Administration

Manage Users, Groups, and Roles

Object-Type Level Privileges Management

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Manipulation Language (DML) Statement Reference

DML statements let you add, remove, and manage the data stored in tables. You can write and execute DML statements like any other SQL statements in the Ocient® System. The SELECT SQL statement is sometimes considered a DML statement, and the Ocient System includes it in the Data Query Language (DQL). For details, see the **Data Query Language (DQL) Statement Reference**.

Supported DML SQL statements are:

- DELETE FROM TABLE
- INSERT INTO TABLE
- TRUNCATE TABLE

DELETE FROM TABLE

Removes rows from the specified table.

You can use the `WHERE` clause to specify the rows to remove. If a `DELETE` SQL statement lacks the `WHERE` clause, then the database deletes all rows in the table.

To use this statement, you must have the `DELETE` privilege for the table.

For details and examples, see **Remove Records from an Ocient System**.

> `DELETE` actions cannot be undone.
>
> If a `DELETE` operation fails during execution, the database rolls back the changes and returns to its original state.

> Due to limitations of the JDBC API, the reported modified row count might not be accurate for `DELETE` operations that are larger than two billion rows.

Syntax

```sql
DELETE FROM table_name [ WITH cte ] [ WHERE <filter_clause> ]
```

Parameter	Type	Description
`table_name`	string	The name of the table, specified as a string, indicates where to delete rows.
`cte`	string	A common table expression that defines temporary data for the `DELETE` statement. For details about using common table expressions, see **WITH**.
`<filter_clause>`	None	A logical combination of predicates that filter the rows to delete based on one or more columns. For details, see the **WHERE** clause. The `DELETE` SQL statement removes all rows from a table if you do not include the `WHERE` clause.

Examples

Delete Rows from the Table with Filter Criteria

This `DELETE` SQL statement removes all rows in the `movies` table that have a budget of less than `10000`.

```sql
DELETE FROM movies WHERE budget < 10000;
```

Delete Rows from the Table Using a Common Table Expression

This example uses a common table expression using the `WITH` keyword to find rows representing all transactions that occurred before 2022 that are less than $100. The `DELETE` SQL statement receives the results from the common table expression. Then, the database executes this statement to delete the corresponding rows.

SQL

```
DELETE FROM transactions
    WITH old_transactions AS (
        SELECT
            transaction_id
        FROM
            transactions
        WHERE transaction_date < '2022-01-01'
            AND amount < 100
    )
    WHERE transaction_id IN (
        SELECT transaction_id
        FROM old_transactions );
```

INSERT INTO TABLE

`INSERT INTO` inserts rows into a table in the current database using literal values, column references, function executions, computed expressions, or column default values.

This SQL statement requires the `INSERT` privilege for the relevant table.

> Due to limitations of the JDBC API, the reported modified row count might not be accurate for insert operations that are larger than two billion rows.

Syntax

SQL

```
INSERT INTO table_name [ ( col1, col2 [, ...] ) ]
    [ WITH cte ] { query | [ DEFAULT VALUES | VALUES [ <rows_to_insert> ] }

<rows_to_insert> ::=
    ( row1_exp1, row1_exp2 [, ...] ),
    ( row2_exp1, row2_exp2 [, ...] )
    [, ...]
```

© 2026 Ocient, Inc.

Using `DEFAULT VALUES` inserts a single row where each target column is populated with its column defaults (as defined in the column definition) instead of an explicit `VALUES` list.

For table columns that do not each have a defined default value, the inserted row is NULL. If the column has no default and also has the `NOT NULL` constraint, the insert operation generates an error.

Parameter	Type	Description
`table_name`	string	The name of the table for insertion.
`col1, col2 [, ... ]`	string	A list of specific columns to insert specific values. This column list defaults to all columns in the table if you do not specify any column names. The `INSERT` statement can use a subset of the table columns. Any columns not included in the statement are populated with their column default value in their column definition. If the column definition does not specify a default, the inserted row is NULL. If the column has no default and also has the `NOT NULL` constraint, the insert operation generates an error.
`cte`	string	A common table expression that defines temporary data for the `INSERT` statement. For details about using common table expressions, see **WITH**.
`query`	string	A `SELECT` query that defines values or a table and any of its columns that should be inserted into the specified `table_name`.
`row1_exp1, row1_exp2 [, ...]`	string	The expressions to insert into columns in the table. This list must match the number of columns specified in the INSERT statement. Similarly, each expression must match the data type of the column that corresponds to its position. Expressions can be any of the following: **L****iterals**: `1`, `3.14`, ‘ `abc`’, `DATE '2024-01-01'`, etc. **Column references:** If your `INSERT` statement includes a common table expression using a `WITH` clause, you can reference columns from the separate table in that clause. **Function executions**: `ABS(-5)`, `NOW()`, `ST_DISTANCE`, etc.

Parameter	Type	Description
		Computed expressions: `price * quantity`, `COALESCE(x, 0)`, etc. **Column default values**: Use the keyword `DEFAULT` to insert a default value specified in the column definition. If the column definition does not specify a default, the inserted row is NULL. If the column has no default and also has the `NOT NULL` constraint, the insert operation generates an error.

Examples

Insert Values from One Column

This example inserts the columns from `system.table_b` into `system.table_a`.

SQL

```
INSERT INTO system.table_a SELECT * FROM system.table_b;
```

Insert Values from Multiple Columns

This example inserts the column `system.table_b.id_col_b` into `system.table_a.id_col_a` and `system.table_b.int_col_b` into `system.table_a.int_col_a`.

SQL

```
INSERT INTO system.table_a (id_col_a, int_col_a)
    SELECT id_col_b, int_col_b FROM system.table_b;
```

Insert Literal Values

Create a table with product, quantity, and date of sale information with these non-nullable columns:

- `product` — Product identifier
- `quantity` — Quantity of the product sold
- `sale_date` — Date of sale

© 2026 Ocient, Inc.

SQL

```
CREATE TABLE sales (
    product INT NOT NULL,
    quantity INT NOT NULL,
    sale_date DATE NOT NULL
);
```

Insert three rows of literal values that represent different sales.

SQL

```
INSERT INTO sales (product, quantity, sale_date)
    VALUES
        (1, 10, '2023-01-15'),
        (2, 5, '2023-01-20'),
        (1, 8, '2023-02-05');
```

Insert Values Using a Common Table Expression

In this example, a common table expression performs calculations on the `sales` table before inserting rows into the `monthly_sales_summary` table.

The example uses the `monthly_sales_summary` table created by this `CREATE TABLE` statement with these non-nullable columns:

- `product_id` — Product identifier
- `month` — Month part of the date
- `total_quantity` — Total quantity of the product

SQL

```
CREATE TABLE monthly_sales_summary (
    "product_id" INT NOT NULL,
    "month" DATE NOT NULL,
    "total_quantity" INT NOT NULL
);
```

© 2026 Ocient, Inc.

The common table expression following the `WITH` keyword extracts the month from the sale date `sale_date` and calculates the sum of the quantity sold `total_quantity` of the product from the `sales` table before inserting this data. Then, the `INSERT` SQL statement specifies to insert the data into the `monthly_sales_summary` table.

SQL

```sql
INSERT INTO monthly_sales_summary (product_id, month, total_quantity)
WITH monthly_totals AS (
    SELECT
        product,
        DATE_TRUNC('month', sale_date) AS month,
        SUM(quantity_sold) AS total_quantity
    FROM
        sales
    GROUP BY
        product,
        DATE_TRUNC('month', sale_date)
)
SELECT
    product,
    month,
    total_quantity
FROM
    monthly_totals;
```

Insert Columns Using Default Values

This code utilizes the `customers` table with these columns:

- `customer_id` — Customer identifier
- `name` — Customer name
- `status` — Customer status with the default `active` status
- `created_at` — Created date

© 2026 Ocient, Inc.

SQL

```sql
CREATE TABLE customers (
    customer_id INT,
    name VARCHAR(100),
    status VARCHAR(20) DEFAULT 'ACTIVE',
    created_at TIMESTAMP
);
```

Use the `DEFAULT VALUES` keyword to insert one row of default values into the table. For columns that lack a defined default value, the operation inserts a NULL row.

SQL

```sql
INSERT INTO customers DEFAULT VALUES;
```

The resulting row contains all NULL values except for the `status` column, which has the `active` default value.

SQL

```sql
SELECT * FROM customers;
```

Output

SQL

```
   | customer_id | name  | status | created_at |
   | ----------- | ----- | ------ | ---------- |
   |             |       | ACTIVE |            |
```

Alternatively, you can insert default values by using the `DEFAULT` keyword as one of the row values in the `INSERT` statement.

© 2026 Ocient, Inc.

SQL

```
INSERT INTO customers (customer_id, name, status, created_at)
    VALUES (1, 'Alice', DEFAULT, NULL);
```

Output

SQL

```
   | customer_id | name  | status | created_at |
   | ----------- | ----- | ------ | ---------- |
   |             |       | ACTIVE |            |
   | 1           | Alice | ACTIVE |            |
```

INSERT INTO TABLE USING LOADERS

Specify one or more Loader Nodes for executing the `INSERT INTO` SQL statement. If you do not use this option, the Ocient System uses all Loader Nodes that are live to execute the SQL statement.

This statement is useful for managing loading operations, particularly when balancing multiple loads of different sizes and resource requirements. Alternatively, this statement can also help simplify small batch loads by sourcing the data from a single Loader Node.

Syntax

SQL

```
INSERT INTO TABLE table_name USING LOADERS streamloader [, ... ]
     query
```

Parameter	Type	Description
`streamloader`	string	A unique name for the Loader Node. Identify the names of Loader Nodes from the `sys.nodes` table by using this query: `SELECT name FROM sys.nodes;` If the name of the streamloader contains special characters, you must enclose it in quotes, such as `"stream-loader1"`.

For the query to execute successfully, the specified names must:

- Identify nodes that are live.
- Identify nodes that have the Loader role.

Examples

This example inserts the column `system.table_b.id_col_b` into `system.table_a.id_col_a` and `system.table_b.int_col_b` into `system.table_a.int_col_a`. Use the Loader Node named `stream-loader1` to execute this SQL statement.

SQL

```
INSERT INTO system.table_a (id_col_a, int_col_a)
    USING LOADERS "stream-loader1"
    SELECT id_col_b, int_col_b FROM system.table_b;
```

In this example, execute the same SQL statement with two Loader Nodes named `stream-loader2` and `stream-loader3`.

SQL

```
INSERT INTO system.table_a (id_col_a, int_col_a)
    USING LOADERS "stream-loader2","stream-loader3"
    SELECT id_col_b, int_col_b FROM system.table_b;
```

TRUNCATE TABLE

`TRUNCATE TABLE` removes some or all records from an existing table in the current database. The system deletes the truncated data, but the table and its schema remain intact in the system even if all data is deleted. If the entire table is truncated, Global Dictionary Compression tables remain in place.

To truncate a table, you must have the `DELETE` privilege for the table.

To remove a subset of rows from a table, you can use the **DELETE FROM TABLE** SQL statement.

For details and examples of using `TRUNCATE`, see **Remove Records from an Ocient System**.

⚠ This action cannot be undone and results in data loss.

Syntax

SQL

```
TRUNCATE TABLE table_name

TRUNCATE TABLE table_name WHERE segment_group_id = <ID>

TRUNCATE TABLE table_name WHERE segment_group_id in (<ID>, ...)
```

Parameter	Type	Description
`table_name`	string	The name of the table to truncate.
`segment_group_id`	numeric	Identifier of the segment group.

Examples

This example truncates an existing table in the current database and schema named `students`.

SQL

```
TRUNCATE TABLE students;
```

This example truncates an existing table in the current database named `us.students`.

```
SQL

TRUNCATE TABLE us.students;
```

This example truncates a single segment group from an existing table in the current database named `students`.

```
SQL

TRUNCATE TABLE students WHERE segment_group_id = 123456789;
```

This example truncates a number of segment groups from an existing table in the current database named `us.students`.

```
SQL

TRUNCATE TABLE us.students WHERE segment_group_id IN (1,2,3,4,5);
```

Related Links

Data Query Language (DQL) Statement Reference

Data Definition Language (DDL) Statement Reference

Transactional Control Language (TCL) Statement Reference

Commands Supported by the Ocient JDBC CLI Program

Ocient Python Module (pyocient)

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Query Language (DQL) Statement Reference

The OcientAIQ™ Unified Data Platform supports querying using the SQL syntax that follows ANSI SQL standards.

Ocient SQL Syntax

This code block shows the general syntax to perform SQL querying in Ocient® and the order in which commands should go.

For specific descriptions and syntax for the commands, see the respective SQL statement sections on this page.

Syntax

SQL

```
[ WITH ... ]
SELECT ...
    [ EXCEPT(...) ]
[ FROM ...
    [ JOIN ... ] ]
[ WHERE ... ]
[ GROUP BY ...
    [ HAVING ... ] ]
[ ORDER BY ... ]
[ LIMIT ... ]
[ OFFSET ... ]
[ INTERSECT ... ]
[ EXCEPT ... ]
[ UNION ... ] ]
[ USING ... ]
[ TRACE ... ]
[ TAG ... ]
```

Default Schema

© 2026 Ocient, Inc.

Ocient identifies every table using a database and schema. For example, the fully qualified path to the `movies` table is `cinema.adventure.movies`, where `cinema` is the database and `adventure` is the schema.

When you do not fully qualify a table name, the Ocient System uses a default schema. When you first log into the system, the default schema is your fully qualified username. You can change the default schema using the **SET SCHEMA** command.

Querying SQL Statement Reference

Ocient supports the following SQL statements.

WITH

Assigns a name to a common table expression, allowing an auxiliary query to be used in the main query. This helps break complex queries into smaller parts.

Syntax

SQL

```
WITH
    <cte_name> [ ( <cte_column> [ ,... ] ) ]
    AS ( <sub_query> )
<select_query>
```

Parameters

Parameter	Description
`<cte_name>`	A name for the common table expression populated with the result set from the `sub_query` statement.
`<cte_column>`	Optional. This is a list of column names for common table expressions populated with the `sub_query` result set. If no column names are provided, the column names are the same as the table referenced in the `sub_query` statement.
`<sub_query>`	An auxiliary query that collects data for a common table expression. `sub_query` can use regular query commands, including ORDER BY, LIMIT, OFFSET, UNION, INTERSECT, EXCEPT, WHERE, GROUP BY, and HAVING. See the respective SQL statement reference sections for usage rules.

Parameter	Description
`<select_query>`	The main SELECT query. Note that the `FROM` statement in the main `SELECT` query must also include the `cte_name` if the query uses its values.

Example

In this example, the subquery calculates the average budget for all rows in the `movies` table. The main query uses that average to find all movies that spent more.

SQL

```
WITH avg_budget_table (average_budget) AS (
    SELECT AVG(budget)
    FROM movies
)
SELECT title,
    budget,
    revenue
FROM movies,
    avg_budget_table
WHERE budget > avg_budget_table.average_budget;
```

Output

title	budget	revenue
CGI Why	237000000	2787965087
Titania	200000000	1845034188
Merchandise Vehicle 5	200000000	1066969703
The Tentpole	220000000	1519557910
Pirates of Palm Springs	140000000	655011224
Spyman 16	200000000	1108561013
Frigid	150000000	1274219009
Fury 7	190000000	1506249360
Superhero 23	250000000	1084939099

© 2026 Ocient, Inc.

title	budget	revenue
Triassic World	150000000	1513528810
Iron Chef 3	200000000	1215439994
The Last Airman	150000000	318502923

SELECT

Initiates a query statement or a subquery clause within other statements.

You can query the data of tables where you have the SELECT privilege.

For information on using SELECT as a subquery for filtering or ordering results, see the **WHERE** and **HAVING** sections.

For information on using SELECT as a subquery for a common table expression, see the **WITH** section.

> `SELECT *` queries that lack a `FROM` clause automatically reference the `sys.dummy1` table.
>
> For example, `SELECT *;` is the same as `SELECT * FROM sys.dummy1;` .

Syntax

SQL

```
SELECT [ ALL | DISTINCT ]
    [ * [ EXCEPT ( column_name [ , ... ] ) ]
    | <select_list_entry> [ , ... ] ]
    [ <from_clause> ]
```

Parameters

Parameter	Description
`ALL`	`SELECT` and `SELECT ALL` are both the same. Returns all valid data rows from the database that meet the criteria of your query.

Parameter	Description
`DISTINCT`	`SELECT DISTINCT` returns only unique rows that do not match other rows based on the criteria of your query.
`*`	Returns all columns from the specified tables in the result set for the query.
`EXCEPT`	When used with `*`, `EXCEPT` allows specific columns to be excluded from the query results.
`column_name`	The name of one or more columns from the specified table that you want to exclude (using `EXCEPT`) from your query results.

<select_list_entry>

The `select_list_entry` defines a column or expression to include in your query result set.

Syntax

SQL

```
<select_list_entry> ::=
    column_name | expression [ AS new_name ]
```

Parameters

Parameter	Description
`column_name`	The name of one or more columns from the specified table that you want to include in your query results.
`expression`	One or more expressions that you want to include in your query results. Expressions can be any combination of literal values, column names, arithmetic expressions, parentheses, and function calls.
`new_name`	An alias, treated as an identifier, for an alternative name for the column or the results of the expression.

<from_clause>

For information on the `<from_clause>`, see the **FROM** documentation.

Examples

Using SELECT *

This example uses `SELECT *` to return all the columns in the `movies` table.

SQL

```
SELECT * FROM movies LIMIT 5;
```

Output

movie_id	title	budget	popularity	release_date
211672	Minnows	74000000	875.581305	2015-06-17
24	Billy the Killy	30000000	79.754966	2003-10-10
19995	CGI Why	237000000	150.437577	2009-12-10
37724	Spyman 16	200000000	93.004993	2012-10-25
24428	The Tentpole	220000000	144.448633	2012-04-25

Using SELECT * EXCEPT

This example uses `SELECT * EXCEPT` to exclude certain columns from the result set.

SQL

```
SELECT *
EXCEPT (movie_id, runtime, vote_average, vote_count)
FROM movies
LIMIT 5;
```

Output

title	budget	release_date	revenue
Swords & Scabbards	94000000	2003-12-01	1118888979
Billy the Killy	30000000	2003-10-10	180949000
Merchandise Vehicle 5	200000000	2010-06-16	1066969703
CGI Why	237000000	2009-12-10	2787965087

© 2026 Ocient, Inc.

title	budget	release_date	revenue
Space Odyssey 6000	10500000	1968-04-10	68700000

Using Lateral Column Aliases

Ocient SQL queries support lateral column aliases, meaning you can immediately reuse aliases for calculations in the same query as new inputs. Hence, you can simplify queries that normally require subqueries and common table expressions.

These examples use the `products` table with these columns:

- `product_id` — Product identifier as an integer
- `product_name` — Product name as a string
- `price` — Price as a floating point number

Create this table using the `CREATE TABLE` SQL statement.

SQL

```sql
CREATE TABLE products (
    product_id INT,
    product_name VARCHAR(100),
    price DECIMAL(10, 2)
    );
```

Insert four records into the `products` table.

SQL

```sql
INSERT INTO products (product_id, product_name, price) VALUES
    (1, 'Laptop', 1000.00),
    (2, 'Tablet', 500.00),
    (3, 'Smartphone', 800.00),
    (4, 'Monitor', 300.00);
```

Create a query to determine prices after discounts and taxes by using a common table expression subquery `DiscountedPrices`. Use the `WITH` keyword to create the subquery.

© 2026 Ocient, Inc.

SQL

```sql
WITH DiscountedPrices AS (
    SELECT
        product_id,
        product_name,
        price,
        price * 0.90 AS discounted_price,
        price * 0.90 * 1.05 AS total_price_after_tax
    FROM
        products
)
SELECT
    product_id,
    product_name,
    price,
    discounted_price,
    total_price_after_tax
FROM
    DiscountedPrices;
```

Lateral aliases allow the same calculations to be packaged in a single query.

This simpler query is essentially the same as the longer common table expression example, but the logic is condensed because you can reference the `discounted_price` alias immediately to calculate the `total_price_after_tax` value in the same query.

SQL

```sql
SELECT
    product_id,
    product_name,
    price,
    price * 0.90 AS discounted_price,
    discounted_price * 1.05 AS total_price_after_tax
FROM
    products;
```

FROM

Specifies the table or view to use in a `SELECT` statement.

Syntax

```SQL
<select_clause> FROM
    { table_name | ( <sub_query> ) | ( <values_clause> ) } [ ,... ]
    [ <join_clause> [ ,... ] ]
```

Parameters

Parameter	Description
`<select_clause>`	A query statement using the `SELECT` SQL statement. For details, see the **SELECT** section.
`table_name`	The names of one or more tables or views that you want to query. If you specify multiple tables in the `FROM` list, separated by commas, the effect is the same as if you explicitly perform a **CROSS JOIN** on all of those sources.
`<sub_query>`	One or more subqueries using the `SELECT` SQL statement. A subquery generates a table from which the `select_clause` query references data. Each subquery must be enclosed in parentheses with an optional correlation name. For details, see the **SELECT** section.
`<values_clause>`	Use the VALUES keyword to define a table with data. For example, the `SELECT * FROM (VALUES (1, NULL), (2, 5))` SQL statement selects all the data from a table defined by the specified values as `(1, NULL)` for the first row with two columns, and `(2, 5)` for the second row.
`<join_clause>`	A `JOIN` clause used to retrieve data from two or more tables for your query. For details, see the **JOIN** section.

JOIN

Combines rows from multiple tables so they can be accessed by a query.

Syntax

```SQL
<select_query>
    FROM <table_reference1> <join_operation> <table_reference2>
        ON table1_column <boolean_operator> table2_column
```

Parameter

Parameter	Description
`<select_query>`	A valid `SELECT` query. For details, see **SELECT**.
`<table_reference1>`	Either a table name or a full `SELECT` in parentheses `table_reference1` takes precedence for returning rows for `LEFT OUTER JOIN`, `SEMI JOIN`, and `ANTI JOIN`. For specific rules, see **Types of Join Operations**.
`<table_reference2>`	Either a table name or a full `SELECT` in parentheses function. `table_reference2` takes precedence for returning rows for `RIGHT OUTER JOIN`. For specific rules, see **Types of Join Operations**.
`table1_column`	A column from either `table_reference` The `JOIN` operation uses this column to match rows with `table2_column` to combine data for the result set.
`<boolean_operator>`	Join conditions can use any Boolean expression, including `=`, `!=`, `<`, `>`, `=>`, and `<=`.
`table2_column`	A column from the `table_reference` not used for `table1_column` The `JOIN` operation uses this column to match rows based on the `table1_column` to combine data for the result set.

Types of Join Operations (`<join_operation>`) [#types-of-join-operations]

Ocient supports the following types of `JOIN` operations.

Syntax

SQL

```
<join_operation> ::=
    { INNER JOIN
    | LEFT [ OUTER ] JOIN
    | RIGHT [ OUTER ] JOIN
    | FULL [ OUTER ] JOIN
    | CROSS JOIN
    | SEMI JOIN
    | ANTI JOIN }
```

JOIN Type Descriptions

JOIN Type	Description
`INNER JOIN`	Returns all rows that exist in both tables. This join type is the default type of join.
`LEFT OUTER JOIN`	Returns all rows from the first table regardless of whether there are matching rows in the second table. `LEFT JOIN` performs the same operation as `LEFT OUTER JOIN`.
`RIGHT OUTER JOIN`	Returns all rows from the second table regardless of whether there are matching rows in the first table. `RIGHT JOIN` performs the same operation as `RIGHT OUTER JOIN`.
`FULL OUTER JOIN`	Returns all matched and unmatched rows. `FULL JOIN` performs the same operation as `FULL OUTER JOIN`.
`CROSS JOIN`	Returns the Cartesian product of the joined tables. Each row in the first table is joined with all rows in the second table. The result set contains the number of rows in the first table multiplied by the number of rows in the second table.
`SEMI JOIN`	Returns the rows from the first table that have at least one match in the second table. No columns from the second table are available in the query.
`ANTI JOIN`	Returns the rows from the first table that have no matches in the second table. No columns from the second table are available in the query.

© 2026 Ocient, Inc.

By default, `JOIN` statements that involve subqueries can operate laterally if necessary. This behavior allows subqueries to reference joined columns from the preceding items included in the `FROM` clause.

For example, both parts of this join operation use subqueries that reference table `x`. The `LATERAL` keyword is optional; the join operation acts the same regardless of whether it is included.

Text

```
SELECT \* FROM
    (
    SELECT
        1 AS cx1,
        2 AS cx2) AS x
INNER JOIN LATERAL (
    SELECT
        x.cx1 AS cy1,
        x.cx2 AS cy2) AS y
ON
    x.cx1 = y.cy1;
```

Lateral joins are primarily useful when a cross-referenced column is necessary for computing the rows to join. A common application is providing an argument value for a set-returning function.

Examples

These examples join two tables:

- `games` — A table of video game titles and their genre_ids. Note that some games have a NULL value assigned to their genre_id.

game_name	genre_id
Star Battle 6000	9
Dwarf Simulator	NULL
2002 Futbol	11
Space Attack!	9
Fantasy Fool	8

game_name	genre_id
Blasto the Squirrel	5
Trucker Quest	7
Block, Stack, and Cry!	6
Basketball Day	11
Pizza Mutant	1
Italian Plumber	5
Zombies!	1
Space Ninjas	12
Moon Mayor	10
Spreadsheet Hero	NULL
Zoom Tycoon	12
Porch Poacher	6

- `genre` — A table of video game genres, which are identified by IDs.

genre_name	id
Racing	7
Puzzle	6
Adventure	2
Simulation	10
Shooter	9
Misc	4
Action	1
Sports	11
Role-Playing	8
Platform	5
Strategy	12
Fighting	3

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

INNER JOIN

This example uses an `INNER JOIN` operation to capture only the rows that exist in both tables. Games with NULL values for their `genre_id` are eliminated from the result set.

SQL

```
SELECT game_name,
    genre_name
FROM video_games.game
    INNER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
Fantasy Fool	Role-Playing
Blasto the Squirrel	Platform
Italian Plumber	Platform
Porch Poacher	Puzzle
Space Ninjas	Strategy
Basketball Day	Sports
2002 Futbol	Sports
Moon Mayor	Simulation
Block, Stack, and Cry!	Puzzle
Zombies!	Action
Pizza Mutant	Action
Space Attack!	Shooter
Trucker Quest	Racing
Zoom Tycoon	Strategy

LEFT OUTER JOIN

This example uses a `LEFT OUTER JOIN` operation to capture all rows from the left table (game), even if they have no matching row in the right table (genre).

© 2026 Ocient, Inc.

SQL

```sql
SELECT game_name,
    genre_name
FROM video_games.game
    LEFT OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
Fantasy Fool	Role-Playing
Space Ninjas	Strategy
Zoom Tycoon	Strategy
Zombies!	Action
Pizza Mutant	Action
Trucker Quest	Racing
Blasto the Squirrel	Platform
2002 Futbol	Sports
Basketball Day	Sports
Italian Plumber	Platform
Block, Stack, and Cry!	Puzzle
Star Battle 6000	Shooter
Dwarf Simulator	NULL
Moon Mayor	Simulation
Porch Poacher	Puzzle
Spreadsheet Hero	NULL
Space Attack!	Shooter

RIGHT OUTER JOIN

This example uses a `RIGHT`` OUTER JOIN` operation to capture all rows from the right table (genre), even if they have no matching row in the left table (game).

© 2026 Ocient, Inc.

SQL

```sql
SELECT game_name,
    genre_name
FROM video_games.game
    RIGHT OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
NULL	Adventure
Trucker Quest	Racing
Pizza Mutant	Action
Moon Mayor	Simulation
Fantasy Fool	Role-Playing
Space Attack!	Shooter
NULL	Misc
Star Battle 6000	Shooter
Space Ninjas	Strategy
Porch Poacher	Puzzle
Block, Stack, and Cry!	Puzzle
Blasto the Squirrel	Platform
Italian Plumber	Platform
Zoom Tycoon	Strategy
NULL	Fighting
Zombies!	Action
2002 Futbol	Sports
Basketball Day	Sports

FULL OUTER JOIN

This example uses a `FULL OUTER JOIN` operation to capture all rows from both tables, even if they do not match.

SQL

```sql
SELECT game_name,
    genre_name
FROM video_games.game
    FULL OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
Basketball Day	Sports
2002 Futbol	Sports
Zombies!	Action
Blasto the Squirrel	Platform
Moon Mayor	Simulation
Trucker Quest	Racing
NULL	Misc
Star Battle 6000	Shooter
Space Attack!	Shooter
Spreadsheet Hero	NULL
NULL	Adventure
NULL	Fighting
Italian Plumber	Platform
Pizza Mutant	Action
Dwarf Simulator	NULL
Fantasy Fool	Role-Playing
Block, Stack, and Cry!	Puzzle
Space Ninjas	Strategy

© 2026 Ocient, Inc.

game_name	genre_name
Zoom Tycoon	Strategy
Porch Poacher	Puzzle

CROSS JOIN

This example uses a `CROSS JOIN` operation to capture every possible combination of rows from both tables, regardless of whether they match. Note that, unlike other `JOIN` operations, `CROSS JOIN` does not require an `ON` statement.

SQL

```
SELECT game_name,
    genre_name
FROM video_games.genre
    CROSS JOIN video_games.game;
```

Output

ⓘ As the result set for this CROSS JOIN example is more than 200 rows, the results are abbreviated.

genre_name	genre_id
Spreadsheet Hero	Role-Playing
Spreadsheet Hero	Misc
Spreadsheet Hero	Sports
Spreadsheet Hero	Platform
Spreadsheet Hero	Strategy
Spreadsheet Hero	Shooter
Spreadsheet Hero	Fighting
Spreadsheet Hero	Action
Spreadsheet Hero	Simulation
Spreadsheet Hero	Racing

© 2026 Ocient, Inc.

genre_name	genre_id
Spreadsheet Hero	Puzzle
Spreadsheet Hero	Adventure
Space Attack!	Role-Playing
Space Attack!	Misc
Space Attack!	Sports
Space Attack!	Platform
Space Attack!	Strategy
Space Attack!	Shooter
Space Attack!	Fighting
Space Attack!	Action
Space Attack!	Simulation
Space Attack!	Racing
Space Attack!	Puzzle
Space Attack!	Adventure
...	...

SEMI JOIN

This example uses a `SEMI JOIN` operation to capture genre names that have at least one match in the games table. Rows are only included once, even if there are multiple matches.

SQL

```
SELECT genre_name,
    id
FROM video_games.genre
    SEMI JOIN video_games.game ON genre.id = game.genre_id;
```

Output

genre_name	genre_id
Simulation	10
Shooter	9
Racing	7
Strategy	12
Puzzle	6
Platform	5
Action	1
Sports	11
Role-Playing	8

ANTI JOIN

This example uses an `ANTI JOIN` operation to capture any rows in the game table that do not match any rows in the genre table.

SQL

```sql
SELECT game_name
FROM video_games.game
    ANTI JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name
Spreadsheet Hero
Dwarf Simulator

WHERE

Filters rows based on a specified condition.

Syntax

```sql
<select_query>
    WHERE { column_name <filter_condition> filter_value |
    [ NOT ] EXISTS ( <sub_query> ) }
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`column_name`	A column used for grouping that is evaluated by the `filter_condition`.
`filter_value`	A value used to evaluate the specified `column_name`. For details, see the **<filter_condition>** section.
`<sub_query>`	A subquery with a filter condition that follows the `EXISTS` clause. For details, see the **EXISTS** section.

<filter_condition>

A logical combination of predicates used to evaluate the referenced `column_name` based on the `filter_value`.

Syntax

SQL

```
<filter_condition> ::=
  { =
    | ==
    | <>
    | !=
    | [ NOT ] EQUALS
    | <
    | <=
    | >
    | >=
    | [ NOT ] IN
    | [ NOT ] LIKE
    | [ NOT ] SIMILAR TO
    | BETWEEN
    | FOR SOME
    | FOR ALL
    | IS [ NOT ] NULL
    | IS [ NOT ] DISTINCT FROM
  }
```

Definitions

Operator	Description	Example
`=` , `==` , `EQUALS`	Equal to.	`Author = 'Alcott'`
`<>` , `!=`	Not equal to	`Dept <> 'Sales'`
`>`	Greater than	`Hire_Date > '2012-01-31'`
`<`	Less than	`Bonus < 50000.00`
`>=`	Greater than or equal	`Dependents >= 2`
`<=`	Less than or equal	`Rate <= 0.05`
`[ NOT ] LIKE`	Begins with a character pattern to match. The pattern is enclosed in parentheses and can include the wildcard characters `%` , representing zero, one or multiple characters, and `_` representing a single character. To have a literal `%` or `_` that does not act as a wildcard on the right-hand side of a `LIKE` or `NOT`	`Full_Name LIKE 'Will%'`

Operator	Description	Example
	`LIKE` expression, put a backslash before the character.	
`[ NOT ] SIMILAR TO`	For usage information, see the **SIMILAR TO Operator** section.	`'abc' SIMILAR TO 'abc'`
`BETWEEN value1 AND value2`	Requires two values. The filter evaluates to TRUE if the specified column is within the range between the two values. `BETWEEN` bounds are inclusive. Values can be times, numbers, or strings.	`Sales BETWEEN 1000 AND 5000`
`FOR_SOME()`	Evaluates an array to determine if at least one value meets the filter criteria. In the example, the `FOR_SOME()` operator would return all rows of the column `col_int_array` where at least one value is greater than 10. For details, see **Array Filters**.	`FOR_SOME(col_int_array) > 10` `FOR_SOME(col_int_array) > 10`

`FOR_ALL()`	Evaluates all values of an array to determine if all meet the filter criteria.
	In the example, the `FOR_ALL()` operator would return any rows of the column `col_int_array` where all values are greater than 10.
	For details, see **Array Filters**.
		`FOR_ALL(col_int_array) > 10`
		
		`FOR_ALL(col_int_array) = 1`

`[ NOT ] IN`	Equal to one of multiple possible values	`DeptCode IN (101, 103, 209)`
`IS [ NOT ] NULL`	Compare to `NULL` (missing data)	`Address IS NOT NULL`
`IS [ NOT ] DISTINCT FROM`	Compares the equality of two expressions for a Boolean result, including comparisons that involve `NULL` values. This operator ensures reliable comparisons in scenarios where `NULL` values need to be treated as significant data points. Normally, `NULL = NULL` evaluates to `false` because `NULL` represents an unknown value. However, the statement `a IS NOT DISTINCT FROM b` treats `NULL` as a comparable value and returns `true` when both values are `NULL` or when `a = b`. Inversely, `a IS DISTINCT FROM b` returns `true` if the values are different or if one value is `NULL` while the other is not `NULL`.	`Debt IS NOT DISTINCT FROM Receivables`

EXISTS

An `EXISTS` clause used in a `WHERE` statement evaluates if a subquery returns any rows.

© 2026 Ocient, Inc.

For each row that the database computes in the outer query, if the subquery returns at least one row, the `EXISTS` clause evaluates to `true`. In this outcome, the outer query returns its row.

If the subquery returns zero rows, the `EXISTS` clause evaluates to `false`, and the outer query excludes those rows.

The `NOT EXISTS` clause performs the opposite Boolean logic. In this case, the outer query returns rows only when the subquery has no matches.

Examples

These examples use two tables, one for company departments and the other for employees assigned to those departments.

Create the `departments` table for department data.

SQL

```
CREATE TABLE departments (
    id INT,
    name VARCHAR(50)
);
```

Create the `employees` table for employee data.

SQL

```
CREATE TABLE employees (
    id INT,
    name VARCHAR(50),
    department_id INT
);
```

Insert department data for three departments, `HR`, `IT`, and `Marketing`, into the `departments` table.

© 2026 Ocient, Inc.

SQL

```sql
INSERT INTO departments (id, name) VALUES
    (1, 'HR'),
    (2, 'IT'),
    (3, 'Marketing');
```

Insert employee data for four employees, `Alice`, `Bob`, `Charlie`, and `David`, into the `employees` table.

SQL

```sql
INSERT INTO employees (id, name, department_id) VALUES
    (1, 'Alice', 1),
    (2, 'Bob', 2),
    (3, 'Charlie', 2),
    (4, 'David', 4);
```

Find Employees Belonging to an Existing Department

This example uses an `EXISTS` clause to find any names from the `employees` table who are assigned to a department listed in the `departments` table.

SQL

```sql
SELECT name
FROM employees AS e
WHERE EXISTS (
    SELECT *
    FROM departments AS d
    WHERE e.department_id = d.id
);
```

The query returns all employees except `David`, who is assigned to a department that does not exist in the `departments` table.

Output

Text

```
Bob
Charlie
Alice
```

Find Departments That Have Employees

This query returns departments that have at least one employee. The subquery uses `SELECT 1` to check whether any matching rows exist in the `employees` table. The output is the same if the query uses `SELECT *` instead.

SQL

```sql
SELECT name
FROM departments AS d
WHERE EXISTS (
    SELECT 1
    FROM employees AS e
    WHERE e.department_id = d.id
);
```

The query results exclude the `Marketing` department because no employees belong to it.

Output

Text

```
HR
IT
```

Filter Departments Based on an Uncorrelated Table

In this example, the outer query filters the `departments` table where `id != 3` (excluding the `Marketing` department). The subquery checks if there is at least one employee with a department identifier less than `4`.

© 2026 Ocient, Inc.

The query is uncorrelated because the subquery does not reference the `departments` table.

SQL

```
SELECT *
FROM departments d
WHERE d.id != 3
AND EXISTS (
    SELECT *
    FROM employees e
    WHERE e.department_id < 4
);
```

Output

Text

```
HR
IT
```

Find Employees Without a Valid Department

This example uses a `NOT EXISTS` clause. The example returns only employees who are assigned to a department identifier `department_id` not listed in the `departments` table.

SQL

```
SELECT name
FROM employees AS e
WHERE NOT EXISTS (
    SELECT *
    FROM departments AS d
    WHERE e.department_id = d.id
);
```

*Output: * `David`

SIMILAR TO Operator

`SIMILAR TO` is a keyword that extends the `LIKE` operator, adding more features for match filtering, including many metacharacters used in regular expressions.

`%` and `_` both act as wildcard operators, but other supported metacharacters match traditional regular expressions.

Syntax

```
SQL

WHERE string1 SIMILAR TO string2
```

This table describes the metacharacters supported by the `SIMILAR TO` keyword.

Metacharacter	Description
`%`	Repeated wildcard, matches any characters zero or more times. Equivalent usage to `LIKE`.
`_`	Wildcard, matches exactly one character. Equivalent usage to `LIKE`.
`|`	Alteration, meaning either of two alternatives (`a|b` represents `a` or `b`).
`*`	Repetition of zero or more times.
`+`	Repetition of one or more times.
`?`	Repetition of zero or one times.
`{m}`	Repetition of exactly `m` times.
`{m,n}`	Repetition between `m` and `n` times (inclusive).
`{m,}`	Repetition of at least `m` times.
`()`	Logical grouping.
`[]`	A character class, equivalent to character classes in regular expressions. Example: `[a-z]` is any lowercase English letter.
`^`	Beginning of the line anchor and the negation in character classes.
`$`	End of the line anchor.

`SIMILAR TO` also supports escaping these metacharacters with `\` and certain escape sequences supported by C family languages.

Metacharacter	Description
`\a`	The alert/bell character.
`\b`	The backspace character.
`\B`	A single `\` character. Equivalent to `\\`.
`\f`	Form feed.
`\t`	Horizontal tab.
`\v`	Vertical tab.
`\xy`	`xy` are octal digits that represent the character with the numeric value `xy`.
`\s`	Matches any whitespace.
`\S`	Matches any non-whitespace.
`\m`, `\M`, or `\y`	The boundary of a word. Equivalent to `\b` in normal regular expressions.
`*`	The star character.

Array Filters

A filter expression can use the functions `FOR_SOME()` and `FOR_ALL()` to apply a predicate against all values of an input array.

Array filter functions have the following rules:

- They can only evaluate array types.
- They can go on either the left or right side of a Boolean comparison expression, but not both sides at the same time.
- They must directly use a Boolean comparison operator.
- They cannot use the `SOME` and `ALL` SQL keywords.

Filter Behavior with Empty Arrays

Array filter functions have unique behavior when evaluating empty arrays.

- `FOR_ALL()` evaluates an empty array as `TRUE`.
- `FOR_SOME()` evaluates an empty array as `FALSE`.

© 2026 Ocient, Inc.

If empty arrays must be evaluated for a different result, you can use the `ARRAY_LENGTH` function to specify a minimum array length. For example, this statement would evaluate an array as `TRUE` only if it is not empty and all values matched `%ocient%`:

```
SQL

ARRAY_LENGTH(array_col) > 0 AND FOR_ALL(array_col) LIKE '%ocient%
```

For details about array functions, see the **Array Functions and Operators** page.

Filter Behavior with NULL rows

Filtering with array functions can yield different results depending on whether the array row is NULL or whether the values in the array are NULL.

If `FOR_SOME()` or `FOR_ALL()` evaluates a NULL row (the row itself is NULL, not that the array contains NULL values), then the result is always `FALSE`.

To check an array for the presence of NULL values, you must use the `IS NULL` operator. For example:

```
SQL

FOR_SOME(array_col) IS NULL
```

Array comparison operators, such as `@>`, `<@`, and `&&`, do not adhere to Boolean logic for NULL values. For details, see the **Array Functions and Operators** page.

GROUP BY

Groups rows with the same values into summary rows, based on a specified aggregate function.

Syntax

```sql
SELECT <select_list_clause> FROM table_name
    GROUP BY { column_name | expression | integer } [ , ... ]
    [ HAVING <filter_condition> ]
```

Parameters

Parameter	Description
`<select_list_clause>`	A list of one or more columns for the `SELECT` query. To use `GROUP BY` to summarize the query result set, at least one of the columns in `column_list` must include an aggregate function, such as `SUM()`, `COUNT()`, or `MAX()`. For a list of supported functions, see the **Aggregate Functions** page.
`table_name`	The table to use for the query.
`column_name`	The name of one or more columns to use to group the result set. If you specify multiple columns, the result set is grouped by each unique combination of values from the columns.
`expression`	Any combination of literal values, column names, arithmetic expressions, parentheses, and function calls.
`integer`	An integer representing the position of the columns referenced in the `column_list`. The first position starts at `1`.
`<filter_condition>`	A `HAVING` clause that filters the aggregated groups based on a specified condition. For details, see the **<filter_condition>** section.

Example

This example uses a movie database to calculate the total amount spent on movie production per year.

```sql
SELECT YEAR(release_date),
    SUM(budget)
FROM movies
GROUP BY 1
ORDER BY 1 ASC;
```

Output

year(release_date)	sum(budget)
1968	10500000
1979	31500000
1981	18000000
1982	28000000
1992	14000000
1997	200000000
2003	264000000
2009	237000000
2010	350000000
2012	670000000
2013	350000000
2015	414000000

HAVING

Filters aggregated rows based on a specified condition.

`HAVING` operates in a `GROUP BY` statement by setting a filter for the rows to be aggregated and grouped.

For information on using `GROUP BY` in a query statement, see **GROUP BY**.

Syntax

SQL

```
GROUP BY column_name HAVING <filter_condition>
```

Parameters

Parameter	Description
`column_name`	A column used for grouping that is evaluated by the `filter_condition`.
`<filter_condition>`	A logical combination of Boolean predicates. For information on supported logical predicates, see **<filter_condition>**.

Example

This example calculates the total amount spent on movie production per year. The `HAVING` clause filters out any rows that do not have a sum of at least $100 million.

SQL

```
SELECT YEAR(release_date),
    SUM(budget)
FROM movies
GROUP BY 1
HAVING SUM(budget) > 100000000
ORDER BY 1 ASC;
```

Output

year(release_date)	sum(budget)
1997	200000000
2003	264000000
2009	237000000
2010	350000000
2012	670000000
2013	350000000
2015	414000000

ORDER BY

Sorts the result set in ascending or descending order based on one or more specified columns.

If you specify multiple columns, they are sorted hierarchically from left to right.

Syntax

SQL

```
ORDER BY { column_position | column_name } [ ASC | DESC ]
    [ NULLS FIRST | NULLS LAST ] [ , ... ]
```

Parameters

Parameter	Data Type	Description
`column_position`	Integer	The position of a column to be used for the `ORDER BY` sorting. Column positions start at `1`.
`column_name`	String	The name of a column to be used for the `ORDER BY` sorting.
`ASC | DESC`	String	Optional. Specifies whether to sort the column in ascending (`ASC`) or descending (`DESC`) order. If unspecified, defaults to `ASC`.
`NULLS FIRST | NULLS LAST`	String	Optional. `NULLS FIRST` means NULL values go at the start of the result set. `NULLS LAST` means NULL values go at the end of the result set. If unspecified, defaults to `NULLS FIRST`.

Example

In this example, the `ORDER BY` statement orders the movies from newest to oldest.

SQL

```
SELECT release_date, title
FROM movies
ORDER BY release_date DESC;
```

Output

release_date	title
2015-06-17	Minnows
2015-06-09	Triassic World
2015-04-01	Fury 7
2013-11-27	Frigid
2013-04-18	Iron Chef 3
2012-10-25	Spyman 16
2012-07-16	Superhero 23
2012-04-25	The Tentpole
2010-06-30	The Last Airman
2010-06-16	Merchandise Vehicle 5
2009-12-10	CGI Why
2003-12-01	Swords & Scabbards
2003-10-10	Billy the Killy
2003-07-09	Pirates of Palm Springs
1997-11-18	Titania
1992-08-07	Bang Bang Western
1982-06-25	Blade Walker
1981-06-12	Raiders of the Jungle Temple
1979-08-15	Apocalypse None
1968-04-10	Space Odyssey 6000

LIMIT

Limits the number of returned rows from a query to a specified amount.

The returned rows are nondeterministic unless you use an **ORDER BY** clause.

Syntax

```SQL
<select_query> LIMIT limit_number
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`limit_number`	The number of rows to return from the query, specified as a positive integer or any constant scalar expression that evaluates to a positive integer.

Examples

Limit Returned Rows Using a Number

This example uses `LIMIT` to restrict the number of returned movie titles to only three.

```SQL
SELECT title FROM movies LIMIT 3;
```

Output

```SQL
Minnows
CGI Why
Billy the Killy
```

Limit Returned Rows Using an Expression

The `LIMIT` SQL statement can also accept expressions. This query uses the `1+2` expression with the `sys.dummy` table to create a table of three incrementing integers. For details, see **Generate Tables Using sys.dummy**.

© 2026 Ocient, Inc.

```SQL
SELECT c1 FROM sys.dummy10 LIMIT 1+2;
```

Output

```SQL
1
2
3
```

OFFSET

Skips a specified number of rows from the result set.

The returned rows are nondeterministic unless you use an **ORDER BY** clause.

Syntax

```SQL
<select_query> OFFSET offset_number
```

Parameters

Parameter	Description
`<select_query>`	A valid SELECT query.
`offset_number`	The number of rows to skip when returning results from the query, specified as a positive integer or any constant scalar expression that evaluates to a positive integer.

Examples

Skip Rows From the Result Set Using a Number

In this example, the `ORDER BY` statement orders the results chronologically. As a result, the `OFFSET` clause removes the oldest six movies from the result set.

SQL

```sql
SELECT release_date,
    title
FROM movies
ORDER BY release_date OFFSET 6;
```

Output

release_date	title
2003-07-09	Pirates of Palm Springs
2003-10-10	Billy the Killy
2003-12-01	Swords & Scabbards
2009-12-10	CGI Why
2010-06-16	Merchandise Vehicle 5
2010-06-30	The Last Airman
2012-04-25	The Tentpole
2012-07-16	Superhero 23
2012-10-25	Spyman 16
2013-04-18	Iron Chef 3
2013-11-27	Frigid
2015-04-01	Fury 7
2015-06-09	Triassic World
2015-06-17	Minnows

Skip Rows From the Result Set Using an Expression

The `OFFSET` SQL statement can also accept expressions. This query uses the expression `3+4` with the `sys.dummy` table to create a column of incrementing integers by skipping the first seven out of 10 rows. For details, see **Generate Tables Using sys.dummy**.

```sql
SELECT c1 FROM sys.dummy10 OFFSET 3+4;
```

Output

```sql
8
9
10
```

INTERSECT

Returns any rows that match between two separate `SELECT` queries. To use `INTERSECT`, the two queries must be compatible, meaning they must return the same number of columns and have similar data types.

By default, the SQL statement eliminates duplicate rows unless the query includes the optional `ALL` keyword. Using the `DISTINCT` keyword is the same as this default behavior.

Syntax

```sql
<select_query1> INTERSECT [ ALL | DISTINCT ] <select_query2>
```

Parameters

Parameter	Description
`<select_query1>`	A valid SELECT query.
`<select_query2>`	A valid SELECT query.

Example

This example has two `SELECT` queries with an `INTERSECT` statement. The full query retrieves movies that had a budget of at least $200 million, but also earned more than$ 200million, butalsoearnedmorethan1 billion in revenues.

SQL

```sql
SELECT *
FROM movies
WHERE revenue > 1000000000
INTERSECT
SELECT *
FROM movies
WHERE budget > 20000000;
```

Output

movie_id	title	budget	popularity	release_date
597	Titania	200000000	100.025899	1997-11-18
19995	CGI Why	237000000	150.437577	2009-12-10
49026	Superhero 23	250000000	112.31295	2012-07-16
10193	Merchandise Vehicle 5	200000000	59.995418	2010-06-16
211672	Minnows	74000000	875.581305	2015-06-17
122	Swords & Scabbards	94000000	123.630332	2003-12-01
37724	Spyman 16	200000000	93.004993	2012-10-25
109445	Frigid	150000000	165.125366	2013-11-27
168259	Furious 7	190000000	102.322217	2015-04-01
68721	Iron Chef 3	200000000	77.68208	2013-04-18
135397	Triassic World	150000000	418.708552	2015-06-09
24428	The Tentpole	220000000	144.448633	2012-04-25

EXCEPT

© 2026 Ocient, Inc.

The `EXCEPT` keyword returns the result set of a first `SELECT` query minus any matching rows from a second `SELECT` query.

`EXCEPT` requires the two queries to be compatible, meaning they must return the same number of columns and have similar data types. By default, the result set eliminates duplicate rows unless the query includes the optional `ALL` keyword. Using the `DISTINCT` keyword is the same as this default behavior.

The `EXCEPT` keyword can also be used to exclude specific columns from a `SELECT *` query. For information on that alternate usage, see the **SELECT** syntax and example.

Syntax

SQL

```
<select_query1> EXCEPT [ ALL | DISTINCT ] <select_query2>
```

Parameters

Parameter	Description
`<select_query1>`	A valid SELECT query that is compared to `select_query2`. The results of an EXCEPT statement are the rows in `select_query1` that do not match any row in `select_query2`.
`<select_query2>`	A valid SELECT query that is compared to `select_query1`.

Example

In this example, the first `SELECT` statement queries all rows in the `movies` table. The `EXCEPT` clause and the second `SELECT` statement eliminate from the results any rows with movies that had a budget greater than `20000000`.

SQL

```
SELECT *
FROM movies
EXCEPT
SELECT *
FROM movies
WHERE budget > 20000000;
```

Output

title	budget	popularity	release_date	revenue
Bang Bang Western	14000000	37.380435	1992-08-07	159157447
Space Odyssey 6000	10500000	86.201184	1968-04-10	68700000
Raiders of the Jungle Temple	18000000	68.159596	1981-06-12	389925971

UNION

Returns the combined result set of two or more `SELECT` queries.

By default, `UNION` eliminates duplicate rows from the result set unless you specify the `ALL` keyword. Using the `DISTINCT` keyword is the same as this default behavior.

Syntax

SQL

```
<select_query1> UNION [ ALL | DISTINCT ] <select_query2> [ ... ]
```

Parameters

Parameter	Data Type	Description
`select_query1`	String	A valid `SELECT` query that is combined with `select_query2`.
`select_query2`	String	A valid `SELECT` query that is combined with `select_query1`.

Example

This example uses `UNION` to merge two separate queries for identical columns into the same result set.

© 2026 Ocient, Inc.

SQL

```sql
SELECT title,
    popularity,
    vote_average
FROM movie
WHERE popularity > 800
UNION
SELECT title,
    popularity,
    vote_average
FROM movies.movie
WHERE vote_average > 7.5;
```

Output

title	popularity	vote_average
Raiders of the Jungle Temple	68.159596	7.7
Minnows	875.581305	6.4
Apocalypse None	49.973462	8
Superhero 23	112.31295	7.6
Billy the Killy	79.754966	7.7
Bang Bang Western	37.380435	7.7
Blade Walker	94.056131	7.9
Swords & Scabbards	123.630332	8.1
Merchandise Vehicle 5	59.995418	7.6
Space Odyssey 6000	86.201184	7.9

USING

Overrides various system configurations for processing a specified query.

The `USING` keyword is required for only the first query override, not for subsequent ones.

For descriptions of the supported query configurations, see the parameter table below.

© 2026 Ocient, Inc.

Syntax

```SQL
<select_query>
USING
    [ SCHEDULING_PRIORITY = priority_value ]
    [ MAX_ROWS_RETURNED = max_rows ]
    [ MAX_ELAPSED_TIME = max_elapsed_time ]
    [ MAX_TEMP_DISK_USAGE = max_temp_disk_usage ]
    [ CACHE_MAX_TIME = cache_max_time ]
    [ CACHE_MAX_BYTES = cache_max_bytes ]
    [ SERVICE CLASS service_class ]
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`priority_value`	An optional clause that specifies the priority at which the query should run. The priority value is a floating-point number, which indicates the priority compared to the other queries. The priority value does not specify any percentage of resources that the query takes or any specific degree of difference between queries. If unspecified, defaults to `SCHEDULING_PRIORITY = 1.0`.
`max_rows`	An optional clause that limits the number of rows returned by a query. If a query exceeds this number of rows, the system kills it. For example, `max_rows_returned = 10` kills any queries returning more than ten rows.
`max_elapsed_time`	An optional clause that limits how long a query can run. If a query exceeds this time limit in seconds, the system kills it. For example, `max_elapsed_time = 10` kills any queries taking longer than 10 seconds.
`max_temp_disk_usage`	An optional clause that limits the percentage of temporary disk used by a query. If a query exceeds this temporary disk limit, the system kills it. For example, `max_temp_disk_usage = 10` kills any queries taking more than 10% of temporary disk space.
`cache_max_time`	An optional clause that affects whether a specified query uses results from the cache rather than executing the query. If there is a cached result with the same query text, executed less than `cache_max_time` seconds ago, the database uses that cached result. The default value is 0, and thus by default, the database does not return any cached results. The cache considers all SQL Nodes, and if a potential cached result is only

Parameter	Description
	available on a different SQL Node, the database redirects the query to that node. If you specify the force attribute on the connection, which disables load balancing and redirection, the database considers only cached results on the current node.
`cache_max_bytes`	An optional clause that controls whether the database stores the results of any specified query in the cache. The system stores the results of all queries executed using a service class specifying this attribute in the cache if the result size is smaller than this value. You can determine the size of a result set in bytes by querying the `bytes_returned` field of the `completed_queries` virtual table. The default value is 0, and thus by default, the database does not cache any results. The database caches results in memory on the SQL Nodes. These results are not cached if there is insufficient memory available.
`service_class`	An optional clause at the end of a query that sets a specific service class to run that query. For details, see **CREATE SERVICE CLASS**.

Example

SQL

```
SELECT * FROM sys.dummy10
USING
    SCHEDULING_PRIORITY = 1.0
    MAX_ROWS_RETURNED = 10
    MAX_ELAPSED_TIME = 10
    MAX_TEMP_DISK_USAGE = 10;
```

Output

SQL

```
c1
-----------
1
2
3
4
5
6
7
8
9
10
```

TRACE

An optional clause that executes the query, but discards the original result set. Instead, `TRACE` returns a result set of tracing data that describes the execution of the query.

Append the `TRACE` SQL statement to the end of a query. The statement can include optional parameters to control its frequency and level of detail.

To understand the results of this statement, see **TRACE Results**.

Trace queries have the same effect to the workload as a query run without the clause.

Contact Ocient Support for guidance in using the TRACE SQL statement.

Syntax

SQL

```
<select_query> TRACE
    [ FREQUENCY frequency_int ]
    [ RESOLUTION resolution_int ]
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`frequency_int`	Controls how often the database samples trace events during query execution in milliseconds. If unspecified, defaults to 500 milliseconds.
`resolution_int`	Determines the level of detail captured for a trace or event. If the resolution of a trace is greater than the resolution of an event, the database records the event. If unspecified, defaults to 100 milliseconds.

TRACE Results

Each row of the trace output provides some information about what an operator instance has done in the time between the previous trace sample and the most current sample.

Column	Description
`plan_parent_id`	The UUID of the parent of the operator instance in the query plan. If an operator has multiple parents, this field specifies the UUID of one of the parents.
`plan_op_id`	The UUID of the operator in the query plan.
`operator_type`	The type of operator that the database uses to sample trace events.
`node_id`	The UUID of the node where this operator executes.
`code_id`	The identifier of the VM core where this operator instance executes. Core identifiers are unique for each node.
`op_id`	The identifier of the operator instance. Identifiers are unique for each node.
`time`	The time in milliseconds when the database collects the information for the trace event relative to the start of the query.
`group`	Group related trace events together.
`sample`	The sample of trace event data. For possible values, see the following table.
`value`	The value of trace event data.

Sample and Value Columns

The `sample` and `value` columns describe what occurred in a specified trace period.

Sample	Description
`ROWS_IN`	The number of rows an operator instance has read from its children operators.
`ROWS_OUT`	The number of rows the operator instance returns.
`BLOOM_FILTERED_ROWS`	The number of rows that this operator instance discards by using Bloom filters.
`SCHEDULE_CYCLE`	The number of cycles in the schedule for this operator instance.
`OOM_CYCLE`	The number of out-of-memory cycles in the schedule for this operator.
`INITIALIZE`	The time when the database initializes the operator.
`FINALIZE`	The time when the database finalizes the operator.

Example

```sql
SELECT * FROM sys.dummy10 TRACE;
```

Output

plan_parent_id	plan_op_id	operator_type	node_id	core_id
77041d2b-f7ca-49d5-a3f5-4b75b0782ed1	498690cf-4a9b-4c1f-affa-417f31fddc2d	RENAME_OPERATOR	08ca7b05-1e4d-455f-9f46-2174ed048d33	4
77041d2b-f7ca-49d5-a3f5-4b75b0782ed1	498690cf-4a9b-4c1f-affa-417f31fddc2d	RENAME_OPERATOR	08ca7b05-1e4d-455f-9f46-2174ed048d33	4
77041d2b-f7ca-49d5-a3f5-4b75b0782ed1	498690cf-4a9b-4c1f-affa-417f31fddc2d	RENAME_OPERATOR	08ca7b05-1e4d-455f-9f46-2174ed048d33	4

TAG

An optional clause that adds one or more tags to the SQL query. You can also add tags to a common table expression in the `WITH` clause. After you define tags, you can find them in the `sys.queries` and `sys.completed_queries` system catalog tables.

Syntax

```
SQL

<select_query> <tag> [ ... ]

<tag> ::= TAG <tag_identifier>
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`tag_identifier`	Identifier of the tag for the query. Enclose the identifier in double quotes if it contains any special characters, such as spaces.

Examples

SQL Query with One Tag

Select the count of a table with one row and tag the query with the name `count1`.

```
SQL

SELECT COUNT(*) FROM sys.dummy1 TAG count1;
```

SQL Query with Multiple Tags

Select the device model and tag this query with two names `device_model` and `phones`.

```
SQL

SELECT device_model
FROM system.adtech_flat
LIMIT 1
TAG device_model
TAG phones;
```

Common Table Expression with One Tag

Define a common table expression with the `average_budget` tag. You can also add another tag for the whole query `overall_budget`.

SQL

```
WITH avg_budget_table (average_budget) AS (
    SELECT AVG(budget)
    FROM movies
    TAG average_budget
)
SELECT title,
    budget,
    revenue
FROM movies,
    avg_budget_table
WHERE budget > avg_budget_table.average_budget
TAG overall_budget;
```

String Literals and Escape Sequences

All string literals in SQL statements must be enclosed in single quotes. To use a single quote within a string, you can use another single quote as an escape, `''`*. *

For other escape sequences, include an `e` character before the string literal, i.e., before the opening single quote. This directs the system to recognize escape sequences in the string literal. This means:

- All single `\` characters in the string now escape themselves.
- Any subsequent character after the `\` is also escaped if it matches an escape sequence (see the table).

If you use escape sequences, be prepared that you might need to alter strings that include `\`, such as directory paths.

Supported Escape Sequences

Escape Sequence	Description
`''`	Single quotation mark
`\"`	Double quotation mark

Escape Sequence	Description
`\n`	Newline character
`\r`	Carriage return character
`\f`	Form feed character (i.e., page break)
`\b`	Backspace
`\\`	Backslash
`\t`	Tab

Examples

These examples show how the system interprets strings with and without escape sequences.

String Without an Escape Sequence

This example selects the simple string literal `'\my\directory\path'` that does not use escape sequences.

```
SQL

SELECT '\my\directory\path';
```

*Output: *`\my\directory\path`

String With an Escape Sequence

This example selects the simple string literal `'\my\directory\path'` and uses an escape sequence. The result omits the backslashes.

```
SQL

SELECT e'\my\directory\path';
```

*Output: *`mydirectorypath`

String With an Escape Sequence to Retain the Backslashes

This example uses the `'\\my\\directory\\path'` string with an escape sequence to include backslashes in the result.

SQL

```
SELECT e'\\my\\directory\\path';
```

*Output: * `\my\directory\path`

Escape Sequences with Regular Expressions

> Be careful using escape sequences with strings that also use regular expressions. Escape sequences in SQL syntax override those in regular expressions.

In this example, the **REGEXP_SUBSTR** function uses the regular expression `\w+` to match any word characters.

SQL

```
SELECT REGEXP_SUBSTR(
    'abcdefghijklmnopqrstuvwxyz',
    '\w+'
    );
```

*Output: * `abcdefghijklmnopqrstuvwxyz`

The function behaves differently if the regular expression is an escape sequence because it overrides the `\` character. The regular expression searches only for the character `w`.

SQL

```
SELECT REGEXP_SUBSTR(
    'abcdefghijklmnopqrstuvwxyz',
    e'\w+'
    );
```

*Output: * `w`

Related Links

SQL Reference

Data Control Language (DCL) Statement Reference

Data Definition Language (DDL) Statement Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Join Operations

Join operations in the Ocient® System deliver high performance without requiring manual fine-tuning of queries and tables. This topic explains how the Ocient System handles joins to help you better understand performance characteristics and make informed design choices.

For syntax and information about supported join types, see the **JOIN** reference.

Join Optimization in Ocient

Before starting any join operation, the Ocient System automatically determines the optimal plan for combining tables. The system determines the best order for joining tables to reduce processing and storage demands while increasing performance. This process includes:

- **Optimizing I/O with Early Filtering** — The execution engine uses various techniques to dynamically reduce input and output (I/O) for join operations by filtering unneeded values.
- **Determining join order** — The optimizer automatically reorders tables for efficiency, usually from smallest to largest in terms of row count, regardless of how you order tables in the SQL statement.
- **Selecting the join strategy** — The system can override user-specified join types with more performant alternatives. For example, the system can convert an `OUTER JOIN` to an `INNER JOIN` when the result remains semantically equivalent.

As a result, you generally do not need to fine-tune `JOIN` queries or configure table definitions to designate a specific table order. The optimizer ensures efficient execution plans even if you do not specify optimal table or join conditions.

> Join performance does not generally benefit from other table configurations for query optimization, such as TimeKey®, Clustering Keys, secondary indexes, or compression.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

How Ocient Joins Tables

The Ocient System executes all `JOIN` operations within SQL statements as either hash or product joins:

Hash Joins

- The system uses hash joins automatically whenever your `JOIN` predicate includes at least one equality comparison (e.g., `ON a.column_1 = b.column_2`).
- The system splits and shuffles data so the hash join executes in parallel. This shuffling enables great scalability and performance, even with large tables.

Product Joins

- If your `JOIN` predicate does not include an equality condition, the system uses a product join (a cartesian join with a filter). For example, the `JOIN` predicate `ON a.column_1 > b.column_2` uses a product join because it is not an equality condition.
- In a product join, the system broadcasts the smaller table of the join to all compute nodes, and then compares every row according to your condition.
- Product joins perform more slowly when operating on large tables, so it is generally better to use an equality comparison in your `JOIN` SQL statements when possible.

Join Storage Considerations

The performance of SQL join operations is limited by the available system memory, specifically the size of the huge pages configured on the server. During the execution of hash joins, the system typically loads the smaller table of the join into memory.

Spill to Disk

If the memory required for the join operation exceeds what is physically available, the system spills intermediate data to disk, utilizing swap or temporary files. This reliance on disk I/O can significantly degrade query performance compared to in-memory processing, but it ensures that the system can complete the query without failure.

Temporary Storage Requirements

When intermediate results or hash tables exceed available memory, the join operation adjusts to using temporary disk space approximately equal to the combined sizes of the participating tables (both the build and probe sides).

Resource Contention

Because joins are resource-intensive, heavy join workloads can affect overall system throughput and potentially impact the performance of concurrent queries, particularly in systems without strict workload management configurations.

Join Best Practices

These best practices help you design join queries that balance performance, scalability, and maintainability. While the Ocient optimizer handles most join operations automatically, applying these techniques can further reduce overhead and improve query efficiency. Many of these tips are common for OLAP database administration and reflect proven approaches across analytical systems.

Moderate Denormalization

In columnar analytical databases, moderate denormalization is a recommended best practice. This denormalization means selectively embedding low-cardinality, frequently used dimension attributes into a fact table to reduce the number of joins. For example, duplicating a product category column in a sales fact table can remove the need for a separate join to the product dimension, while still leaving high-cardinality tables normalized.

Small dimension tables (less than 1 million rows) can also benefit from Global Dictionary Compression (GDC). With GDC, the system can share dictionaries across tables, improving both performance and storage efficiency.

Query Design Considerations

Beyond general join strategies, small design choices in your SQL statements can impact performance significantly. These practices help ensure that the optimizer can make efficient decisions and avoid unnecessary overhead during execution.

Use Window Aggregate Functions Instead of Self-Joins

In many cases, window aggregate functions can replace self-joins, especially for time period comparisons.

Window Aggregate Example (Most Performant)

In this example, the `LAG` function retrieves the revenue from the previous month within the same result set, allowing you to calculate month-over-month changes without requiring a self-join.

SQL

```sql
SELECT
    month,
    revenue,
    LAG(revenue) OVER (ORDER BY month) AS prior_month_revenue,
    revenue - LAG(revenue) OVER (ORDER BY month) AS month_change
FROM monthly_sales;
```

You can obtain the same results by using a self-join statement in a query, but this is less efficient.

Self-Join Example (Less Performant)

The query scans the `monthly_sales` table twice, once as `curr` and again as `prev`. This approach produces the same result as the window function but requires more overhead.

SQL

```sql
SELECT
    curr.month,
    curr.revenue,
    prev.revenue AS prior_month_revenue,
    curr.revenue - prev.revenue AS month_change
FROM monthly_sales curr
JOIN monthly_sales prev
    ON curr.month = prev.month + INTERVAL '1' MONTH;
```

© 2026 Ocient, Inc.

Match Data Types for Join Keys

Always ensure that join keys share the same data type. Mismatched types force the system to apply implicit type conversions, which can slow down execution and interfere with query optimization.

Materialize Calculations Before Joins

Avoid using functions or expressions directly in join conditions. If you apply a calculation at join time (for example, casting a TIMESTAMP to a DATE), the system must evaluate that function for every row, which can result in losing the ability to use statistics or efficient hash keys. A better method is to pre-compute the value and store it as a separate column during loading or as part of a staging step. This approach reduces per-row processing costs and keeps joins predictable.

Materializing Calculations Example

Materializing calculations before joins using a common table expression separates data transformation from the join itself, so the join operates on clean, precomputed keys. This approach reduces workload during the join, enables better use of indexes and statistics, and yields simpler, more predictable query plans.

SQL

```sql
WITH summarized AS (
    SELECT customer_id, SUM(purchase_amount) AS total_spent
    FROM purchases
    GROUP BY customer_id
)
SELECT c.region, s.total_spent
FROM customers c
JOIN summarized s
    ON c.customer_id = s.customer_id;
```

Related Links

Data Query Language (DQL) Statement Reference

Window Aggregate Functions

© 2026 Ocient, Inc.

Query Performance Optimizations

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

SQL Syntax Conventions

Syntax Conventions

The Ocient® documentation uses the following conventions for defining SQL statements in syntax blocks, unless otherwise noted.

Syntax Symbol	Definition	Example
KEYWORD	SQL statement names, SQL statement option names, stored procedure names, utility names, utility flags, and other SQL keywords are in uppercase.	`SELECT * FROM <my_table> LIMIT 100;` SELECT, FROM, and LIMIT are all SQL keywords. For that reason, they are in uppercase.
`{}`	Curly braces indicate that you must choose at least one of the enclosed options. Do not include the curly braces in your SQL statement.	`INSERT INTO table_name [( col1 [, col2 ] [,...] ) ] \{ \<select_statement> | VALUES ( [ val1 [, val2] [,...] ) }` This syntax example has two options to define what data to add using `INSERT INTO`: either provide specific values or use an embedded `SELECT` query. These options are enclosed in curly braces because the user must use one of these options to successfully run an `INSERT INTO` SQL statement.
`[]`	Brackets mean the enclosed functionality is optional, not necessary to run the specified SQL statement. Do not include the brackets in your SQL query. When used with a list of options, the brackets mean that you can choose one of the listed options.	`CREATE CLUSTER [ IF NOT EXISTS ] cluster_name ( \<create_definition> [, ...] )` This syntax example encloses `IF NOT EXISTS` in brackets because it is optional for running the `CREATE CLUSTER` SQL statement.
`|`	The vertical pipe acts as a delimiter for option lists, meaning you can select only one of the options.	`EXPLAIN [ PROTO | JSON | DEBUG ] \<query>` This example shows three options that the `EXPLAIN` SQL statement can use. The vertical pipe indicates that each option is exclusive, meaning only one can be used.
`[ , ... ]`	The ellipsis means you can choose to add as many of the indicated options or parameters as you like, separating your choices with commas to be	`INSERT INTO <table_name> [ ( <column_name> [, ...] ) ] <query>` The INSERT INTO statement uses the ellipsis `[, ...]` to indicate that an infinite number of additional `<column_name>` values can be specified.

© 2026 Ocient, Inc.

Syntax Symbol	Definition	Example
	typed as part of the SQL statement.	
`<syntax_subsection>`	For complex syntax blocks, portions are set apart with angle brackets `<>` to denote there is a separate subsection for a particular portion of the syntax. Syntax blocks with these subsections usually include complicated options that might distract from the core functionality of the syntax of the SQL statement. The names of syntax subsection are written in lowercase with underscores between the words. In the syntax block, the `::=` symbol signifies the subsection.	`CREATE TABLE [ IF NOT EXISTS | OR REPLACE ] table_name [ (\<create_definition> [, ...] ) ] [ [ WITH ] \<create_option> [, ... ] ] [ AS (query) ]` `\<create_definition> ::= \ <column_definition> | \<index_definition>` `\<create_option> ::= STORAGESPACE storage_space_name | SEGMENTSIZE segment_value | REDUNDANCY segment_part (redundancy_scheme) | STREAMLOADER_PROPERTIES streamloader_json | CLUSTERING [KEY | INDEX] clustering_index (clustering_column [, ...]) | INDEX index_name (index_column [, ...]) [USING index_type]`  `\<column_definition> ::= column_name [data_type] [\<column_constraint> [, ...]]`  `\<index_definition> ::= [CLUSTERING] KEY index_name (index_column [, ...])`  `\<column_constraint> ::= TIME KEY BUCKET(bucket_value [, DAY, HOUR, MINUTE, SECOND]) | [NOT] NULL | DEFAULT literal | COMMENT comment | COMPRESSION GDC [(compression_value), EXISTING schema_name] | MAX SIZE (size_value) | [NO] COMPRESSION [compression_scheme]` In this example, CREATE TABLE uses subsections for complex options, such as its column definition, options, and column constraints. Including all of these in one block would be difficult for a user to parse, especially if their goal was only to create a basic table. Note that each subsection is defined with the name of the subsection in angle brackets followed by `::=`.

© 2026 Ocient, Inc.

For the definition of a function syntax, the text refers to input arguments of the function as arguments. For the definition of SQL statements, such as DDL statements, the text refers to input terms as parameters.

Rules for Column and Table Names

Names for tables and columns in Ocient have these rules:

- Names must begin with an alphabet letter.
- Names can contain letters, numbers, and underscores.
- Column names enclosed in double quotes can contain any characters except for newlines and carriage returns. To use double quotation marks in a name, put two double quotation marks back to back.
- Unless names are enclosed in double quotes, they are treated as case-insensitive.

Related Links

SQL Reference

Data Control Language (DCL) Statement Reference

Data Definition Language (DDL) Statement Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Identifiers

In the Ocient® System, identifiers are lexical tokens that name specific entities in the database. For example, a table name is an identifier.

In most cases, you do not have to quote an identifier. However, the identifier must meet these two conditions:

- Start with a letter followed by zero or more letters, digits, or underscores.
- Must not be a reserved word. For the full list of reserved words, see **Reserved Words**.

If the identifier does not meet either of these conditions, then enclose the identifier in double quotes `""` to create a delimited identifier. SQL is case-insensitive, and the database normalizes identifiers to lowercase. For example, you must add double quotes to the `1start_with_digit`, `hy-phen`, and `mixedCase` identifiers so that the database can parse them correctly: `"1start_with_digit"`, `"hy-phen"`, and `"mixedCase"`.

The system uses identifiers to identify these objects:

- Database name
- Schema name
- Function name
- Table name
- Table alias name
- Column name
- Column alias name
- Index name
- View name
- Task name
- Node names
- Machine learning model name

© 2026 Ocient, Inc.

- Connectivity pool name
- Connectivity pool participant name
- Storage space name
- Cluster name
- Service class name
- Single sign-on (SSO) protocol
- Tag name
- JSON key
- Username
- Group name
- Common table expression
- Subexpression name
- Data pipeline name
- Data pipeline type

Related Links

SQL Syntax Conventions

SQL Reference

Machine Learning Model Functions

Connectivity Pools

Users, Groups, and Service Classes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Generate Tables Using sys.dummy

`sys.dummy` is a virtual table that the OcientAIQ™ Unified Data Platform generates with a specific number of rows. This table is a handy tool to quickly create tables for testing without scripting extensive DDL statements.

When you execute `sys.dummy`, the database creates a table with a single column, which is automatically populated with incremental integers.

`sys.dummy` follows similar rules to a table reference. The SQL statement must specify a non-negative integer that represents the number of rows for the SQL statement to generate in the table.

In this syntax block, the `{N}` parameter represents the number of rows for `sys.dummy` to generate.

> `SELECT *` queries that lack a `FROM` clause automatically reference the `sys.dummy1` table.
>
> For example, `SELECT *;` is the same as `SELECT * FROM sys.dummy1;`.

Syntax

SQL

```
sys.dummy{N}
```

Parameter	Description
`{N}`	The number of rows for `sys.dummy` to generate. The minimum numeric value for `{N}` is `0`, and the maximum is `9223372036854775807`. This parameter must not have any leading or trailing characters. If `{N}` does not adhere to these limits, the database returns an error. Do not specify curly brackets with this parameter in queries.

Example

Create a table with 10 rows.

© 2026 Ocient, Inc.

```sql
SELECT * FROM sys.dummy10;
```

*Output: *

```sql
c1
---
1
2
3
4
5
6
7
8
9
10
```

Related Links

Data Query Language (DQL) Statement Reference

SQL Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Transactional Control Language (TCL) Statement Reference

The Ocient® System supports multi-statement transactions. By default, a connection uses autocommit mode, in which the database commits each statement immediately. To group several statements into a single transaction, disable autocommit with **SET AUTOCOMMIT**, and then end the transaction with **COMMIT** or **ROLLBACK**.

> The HTTP Query API does not support transactions. To use transactions, connect to the Ocient System with the Ocient JDBC driver or the pyocient module.

> The Apache® Spark™ Connector supports transactions and performs the commit and rollback actions automatically. When you use the Spark Connector, you do not need to issue COMMIT or ROLLBACK statements.

SET AUTOCOMMIT

Enables or disables autocommit mode for the connection. When autocommit is enabled, the database commits each statement immediately. This mode is the default. When autocommit is disabled, the database groups subsequent statements into a transaction that you complete with the **COMMIT** or **ROLLBACK** statements. Valid values are:

- `ON` — Enables autocommit mode, in which the database commits each statement immediately. This value is the default.
- `OFF` — Disables autocommit mode. The database groups subsequent statements into a transaction until you enter the COMMIT or ROLLBACK statement.

When you change autocommit from `OFF` to `ON`, the database commits any pending transactions.

After you disable autocommit, ensure that you execute SQL statements that are supported by transactions. Otherwise, the database throws an error. For a list of supported statements, see **Transactions**.

© 2026 Ocient, Inc.

Syntax

```
SQL

SET AUTOCOMMIT { ON | OFF }
```

Example

This example disables autocommit, inserts two rows into a table, and then commits the transaction to permanently save the rows.

Disable autocommit.

```
SQL

SET AUTOCOMMIT OFF;
```

Create the public.txn_demo table with two columns: a number and a string.

```
SQL

CREATE TABLE public.txn_demo (
    id INT,
    term VARCHAR(255));
```

Insert two rows into the `public.txn_demo` table and commit these two transactions.

```
SQL

INSERT INTO public.txn_demo VALUES (1, 'alpha');
INSERT INTO public.txn_demo VALUES (2, 'beta');
COMMIT;
```

Drop the table.

SQL

```
DROP TABLE public.txn_demo;
```

COMMIT

Commits the current transaction and permanently saves all changes made after the transaction started. This statement applies only when you disable autocommit with **SET AUTOCOMMIT**. When autocommit is enabled, COMMIT has no effect.

Syntax

SQL

```
COMMIT
```

Example

This example disables autocommit, inserts two rows into the `public.txn_demo` table, and then commits the transaction to permanently save the rows.

Disable the autocommit.

SQL

```
SET AUTOCOMMIT OFF;
```

Create the public.txn_demo table with two columns: a number and a string.

SQL

```
CREATE TABLE public.txn_demo (
    id INT,
    term VARCHAR(255));
```

Insert two rows into the `public.txn_demo` table and commit these two transactions.

```sql
INSERT INTO public.txn_demo VALUES (1, 'alpha');
INSERT INTO public.txn_demo VALUES (2, 'beta');
COMMIT;
```

Drop the table.

```sql
DROP TABLE public.txn_demo;
```

ROLLBACK

Rolls back the current transaction and discards all changes made after the transaction started. This statement applies only when you disable autocommit with **SET AUTOCOMMIT**. When autocommit is enabled, ROLLBACK has no effect.

Syntax

```sql
ROLLBACK
```

Example

This example disables autocommit, inserts two rows in a table, and then rolls back the transaction to discard the inserted rows.

Disable the autocommit.

```sql
SET AUTOCOMMIT OFF;
```

Create the `public.txn_demo` table with two columns: a number and a string.

```sql
CREATE TABLE public.txn_demo (
    id INT,
    term VARCHAR(255));
```

Insert two rows into the `public.txn_demo` table, then roll back both transactions.

```sql
INSERT INTO public.txn_demo VALUES (3, 'gamma');
INSERT INTO public.txn_demo VALUES (4, 'delta');
ROLLBACK;
```

Drop the table.

```sql
DROP TABLE public.txn_demo;
```

Related Links

Data Manipulation Language (DML) Statement Reference

Data Query Language (DQL) Statement Reference

Data Definition Language (DDL) Statement Reference

Commands Supported by the Ocient JDBC CLI Program

Ocient Python Module (pyocient)

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Errors and Warnings

The Ocient® System produces these error and warning codes as part of its operation. Refer to these codes for their description.

You can also execute this SQL statement using the `sys.sql_messages` system catalog table to view the codes and descriptions in the database: `SELECT * FROM sys.sql_messages`.

Error Codes

The error codes are negative values that split into ranges by number such as -100s, -200s, and so on.

-1xx Codes	Description
-100	Unknown error occurred.
-101	This is an internal error that indicates that a client (driver or connector) received an invalid response from the server.
-102	Internal error that indicates a corruption of the serialized result set sent from the server to the client.
-103	Internal error that indicates an unexpected condition has happened.
-2xx Codes	**Description**
-200	A client attempted a connection with a malformed connection string.
-201	The connection to the server failed.
-202	The handshake with the server failed.
-203	The connection to the server was unexpectedly closed.
-204	A network communications error has occurred.
-205	The client tried to connect to a database with a specified name, but no database with that name exists.
-206	The target node is offline.
-3xx Codes	**Description**
-300	An argument to a driver method execution was invalid.

© 2026 Ocient, Inc.

-1xx Codes	Description
-301	A driver method executed, but the input argument object was already closed.
-302	The cursor was not positioned in a valid row.
-303	An attempt was made to create a new result set, but the previous result set for the connection was still open.
-304	The user and password combination was incorrect.
-305	A statement that was not a query was passed to executeQuery().
-306	A statement that was a query was passed to executeUpdate().
-307	A NULL output buffer was passed to a driver method, where a NULL buffer is not allowed.
-308	A driver execution was made that is not implemented.
-313	An invalid parameter marker number was specified.
-316	No database connection exists.
-317	Invalid data was detected for a parameter marker.
-318	The cursor was not in the required state.
-319	An invalid buffer length was specified.
-320	There is currently no result set.
-321	A NULL indicator pointer was required but not specified.
-322	A connection attribute that is not supported was specified in SQLBrowseConnect. The four supported attributes are: “DRIVER”, “DSN”, “UID”, “PWD”.
-323	An invalid descriptor record index was specified. Valid indices start at 1 and increase to the size specified by executing SQLSetDescField with the SQL_DESC_COUNT field.
-324	An invalid or inconsistent descriptor field type was specified in SQLSetDescRec.
-325	An invalid descriptor field was specified in SQLSetDescField.
-326	The target handle passed to SQLCopyDesc was an IRD.
-327	An implicit descriptor handle that was not the original was the specified value in the execution of SQLSetStmtAttr with the SQL_ATTR_APP_PARAM_DESC or SQL_ATTR_APP_ROW_DESC attribute.
-328	Divide by zero.
-329	Invalid floating point operation that resulted in NaN or Inf.

-1xx Codes	Description
-330	The action conflicts with the current state of the database.
-331	The unconditional DELETE FROM statement is rewritten using the TRUNCATE statement. In this case, the database does not create a plan that would appear in the results from the execution of the EXPLAIN statement.
-332	Already in use. Used mostly in DDL when a resource or option is already assigned to the object.
-333	Optional value was substituted.
-334	This driver is unknown or the version support has been removed.
-335	The execution of a client method was invalid.
-336	Cannot execute the new SQL statement as the system is processing a prior statement using the same connection in the background.
-4xx Codes	**Description**
-400	One of the referenced columns does not exist.
-401	The requested data type conversion was invalid.
-402	Unknown data type.
-403	A column name was ambiguous in the context where it was used.
-404	Two incompatible data types were specified as return types in a case statement.
-405	Join leads to a duplicate column name.
-406	Invalid POINT conversion from NULL.
-407	Column value is larger than internal limits.
-5xx Codes	**Description**
-500	Generic syntax error.
-501	The SQL statement is not valid or not supported in the context where it was used.
-502	A duplicate common table expression name was detected.
-503	The number of columns does not match the SELECT statement.
-504	A negative limit was specified.
-505	A negative offset was specified.
-506	An incorrect number of arguments was specified in a function.
-507	Aggregation of a constant value is invalid.

-1xx Codes	Description
-508	An argument to a function had an incorrect data type.
-509	An invalid comparison was detected.
-510	Aggregation was used in a context where it is not allowed (possibly something in a WHERE clause that belongs in a HAVING clause).
-511	A HAVING clause did not have any aggregation in it.
-512	An ORDER BY clause referenced a column by position and that position was out of range.
-513	An expression of type list cannot be returned in a result set.
-514	You cannot have an expression of type list within another list.
-515	A list expression contains incompatible types.
-516	The function does not exist.
-517	If the result of an expression is a time interval type, the result must be cast to an integral type.
-518	An invalid matrix literal was specified.
-519	A matrix literal had incorrect dimensions.
-520	An invalid query priority was specified.
-521	A column reference in a SELECT statement is invalid due to aggregation.
-522	Columns for the first and second SELECT statements for the UNION, EXCEPT, or INTERSECT statement are not compatible.
-523	An expression in the CASE condition is not allowed.
-524	An expression in the JOIN condition is not allowed.
-525	Aggregation is not allowed in the PARTITION BY or ORDER BY clauses of a windowed aggregate.
-526	A non-constant value was found where a constant was expected in an argument of a windowed aggregate.
-527	Range-based framing does not allow the PRECEDING or FOLLOWING keywords.
-528	ORDER BY with a column ordinal is not allowed in a windowed aggregate.
-529	A frame specification is not allowed if no ORDER BY clause is specified in a windowed aggregate.
-530	UNBOUNDED PRECEDING is only allowed on the starting frame bound.
-531	UNBOUNDED FOLLOWING is only allowed on the ending frame bound.

© 2026 Ocient, Inc.

-1xx Codes	Description
-532	The ending frame bound cannot be less than the starting frame bound.
-533	No frame specification is allowed for this particular windowed aggregate.
-534	A window function requires a positive integer argument.
-535	The specified window function requires an ORDER BY statement in the OVER() clause.
-536	Invalid aggregation.
-537	Invalid expression.
-538	A CASE expression within a CASE expression is not allowed.
-539	Window function was used in a context where it is not allowed (perhaps in a predicate of a WHERE or HAVING clause).
-6xx Codes	**Description**
-600	A referenced table does not exist.
-601	The specified database already exists.
-602	The specified table already exists.
-603	The specified view already exists.
-604	The specified view was not found.
-605	The specified database was not found.
-606	The specified storage space was not found.
-607	The specified storage space already exists.
-608	The specified user does not exist.
-609	The specified user already exists.
-610	The specified password for the new user is invalid.
-611	The specified group name already exists.
-612	The specified group does not exist.
-613	The specified connection already exists.
-614	The specified connection was not found.
-615	The referenced translation does not exist.
-616	A translation with that name exists.

-1xx Codes	Description
-617	The specified user was not found in this group.
-618	The provided storage space has no storage cluster assigned.
-619	The referenced table is a system table and not allowed for the export operation.
-620	The referenced pipeline does not exist.
-621	A pipeline with that name exists.
-622	The pipeline is invalid.
-623	The pipeline is unable to be correctly compiled.
-624	The referenced task does not exist.
-625	An index with that type and on that column already exists.
-626	An index with that name already exists.
-627	The referenced table ID no longer exists.
-628	A connectivity pool with that name already exists.
-629	Cannot find a connectivity pool with the specified name.
-630	Cannot drop the last connectivity pool from a node.
-638	Could not find a SCHEMA with that name (Schema does not exist).
-7xx Codes	**Description**
-700	The user does not have read authorization on a referenced table.
-701	The user does not have authorization to create a database.
-702	The user does not have modify authorization on a referenced table.
-703	The user does not have authorization to create a view.
-704	The user does not have modify authorization on a referenced database.
-705	The user does not have modify authorization on a referenced view.
-706	The user does not have authorization to drop a referenced table.
-707	The user does not have authorization to drop a referenced row.
-708	The user does not have authorization to drop a referenced view.
-709	The user does not have authorization to drop a referenced database.
-710	The user does not have authorization to create a table.

-1xx Codes	Description
-711	The user does not have authorization to create a storage space.
-712	The user does not have authorization to drop a referenced storage space.
-713	The user does not have authorization to create a user.
-714	The user does not have authorization to drop a user.
-715	The user does not have the authority to create a group.
-716	The user does not have the authority to drop the group.
-717	The user does not have the authority to create a connection.
-718	The user does not have the authority to drop the connection.
-719	The user is not authorized to take this action.
-720	The user does not have the authority to modify the group.
-721	The user does not have the authority to execute a plan.
-722	The user does not have the authority to execute an inline plan.
-723	The user does not have the authority to create a connectivity pool.
-724	The user does not have the authority to drop the connectivity pool.
-725	The user does not have the authority to modify the connectivity pool.
-726	Unable to drop some objects in the request.
-728	The user does not have the authority to drop the SCHEMA (Objects exist in the schema).
-733	Session has expired. Refresh is necessary.
-8xx Codes	**Description**
-800	An MLMODEL with that name was not found.
-801	An MLMODEL with that name already exists.
-802	The user does not have the authority to create an MLMODEL.
-803	The user does not have the authority to drop the MLMODEL.
-804	Unable to create an MLMODEL due to a singular matrix.
-805	Unable to create an MLMODEL because the training set is empty.
-806	Unable to perform LUP decomposition on this singular matrix.
-807	The user does not have the authority to modify the MLMODEL.

© 2026 Ocient, Inc.

-1xx Codes	Description
-9xx Codes	**Description**
-900	Operation was canceled or aborted.
-901	I/O error.
-902	Scalar Subquery Cardinality Violation.
-903	Numeric value out of range.
-904	Plan compilation error.
-905	TKT limit reached.
-906	Out of memory.
-907	Segment not available.
-908	System is still initializing.
-909	Out of temporary disk space.
-910	Operation was automatically killed due to drop in the client connection.
-911	Failed to acquire OSN (Ownership Sequence Number). Data for querying is no longer available to the storage cluster.
-10xx Codes	**Description**
-1000	The privilege target already has the privilege.
-1001	The user does not have the authority to grant this privilege.
-1002	The privilege target does not have that privilege.
-1003	The referenced role does not exist.
-1004	The provided object ID was not found.
-1005	Create task failure.
-1006	Task already in progress.
-1007	Could not find the supplied service class.
-1008	Supplied service class definition is invalid.
-11xx Codes	**Description**
-1100	No available service classes for the user have available query slots.
-1101	The specified node is not found.
-1102	Segment not found.

-1xx Codes	Description
-1111	Cluster not found.
-1112	A cluster with that name exists.
-1120	Query exceeded service class time limit.
-1121	Query exceeded session time limit.
-1122	Query exceeded query time limit.
-1123	Query exceeded service class row limit.
-1124	Query exceeded session row limit.
-1125	Query exceeded query row limit.
-1126	Query exceeded service class temporary disk limit.
-1127	Query exceeded session temporary disk limit.
-1128	Query exceeded query temporary disk limit.
-1129	Too many columns in result set for the service class.
-1130	Too many columns in result set for the session.
-1131	Too many columns in result set for the query.
-12xx Codes	**Description**
-1200	The security integration is disabled.
-1201	The security integration does not exist.
-1202	The request failed due to a network communications error.
-1203	The Single Sign On request identifier could not be found.
-1204	No session associated with the connection exists.
-1205	The supplied security token is invalid.
-1206	The supplied security token signature does not match the expected value.
-1207	The supplied security token has a signature fingerprint from an unknown entity.
-1208	The supplied security token has expired.
-1220	A network communications error occurred.
-1221	The OpenID provider rejected the request that indicates an internal server error.
-1222	The OpenID provider returned an invalid response that indicates an error with the provider.

© 2026 Ocient, Inc.

-1xx Codes	Description
-1223	The request_uri in the Authorization Request returns an error or contains invalid data.
-1224	The OpenID provider does not support use of the request parameter.
-1225	The OpenID provider does not support use of the request_uri parameter.
-1226	The OpenID provider discovery document is invalid.
-1227	The database OpenID configuration is invalid.
-1228	The OpenID ID or access token has expired.
-1229	The ID or access token was signed by a key not known to the OpenID provider.
-1230	The signing key returned by the OpenID provider is invalid.
-1231	The token contains an invalid claim.
-1232	The token was issued by an unknown OpenID provider.
-1233	The token was issued for an alternate audience.
-1234	The token signature is invalid, which implies the token was manipulated enroute to the database.
-1235	The specified ID token is invalid.
-1236	The specified Access token is invalid.
-1237	The refresh token returned from the provider is invalid.
-1238	The authentication server has timed out while waiting for client authentication.
-1250	The client is not authorized to request an authorization code using this method.
-1251	The resource owner or authorization server denied the request.
-1252	The authorization server does not support obtaining an authorization code using this method.
-1253	The requested scope is invalid, unknown, or malformed.
-1254	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.
-1255	The authorization server encountered an unexpected condition that prevented it from fulfilling the request.
-1256	The authorization server is currently unable to handle the request due to a temporary overloading or maintenance of the server.
-1257	Cannot lock table.

© 2026 Ocient, Inc.

-1xx Codes	Description
-1258	Cannot unlock table.
-1259	Cannot delete from a join or a view with multiple tables.
Other	**Description**
-9999	Undefined exception.

Warning Codes

The warning codes are positive values.

Warning Code	Description
+300	Data received by a JDBC driver was truncated.
+301	The driver is below the recommended threshold for the warning version.
+900	A query returned an empty result set.
+901	The query is estimated to have a long execution time.
+1113	The referenced object does not exist.
+1114	An object with that name exists.
+1115	Not authorized to take this action.
+1116	Restart the node for changes to take effect.
+1117	Complete system restart required for changes to take effect.
+1118	Statement contains invalid syntax.
+1119	Unable to drop some objects in the request.

Related Links

SQL Reference

Data Definition Language (DDL) Statement Reference

Data Integrity and Storage

Query Analysis

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Pipeline Loading Errors

This table captures error codes and their corresponding error messages that you can encounter when you load data using data pipelines in the Ocient® System. Also, the table contains the definitions of the parameters in the error messages, descriptions that explain the error messages, and actions you can take to resolve the errors.

Code	Error Message	Message Parameters	Description
OLD00	Key `key` for class `class` must be set to a non-NULL value. Details: `details`	`key` : The key of the missing property. `class` : The class that has the missing property. `details` : Additional information about the error.	This error occurs when the loading process reads the pipeline definition missing a required property.
OLD01	Value `value` is not acceptable with the key `key` for class `class` . Expected `example` . Details: `details`	`value` : The value that you set for the specified pipeline definition key. `key` : The pipeline definition key with an incorrect value. `class` : The class that contains the property. `example` : An example or description of the correct value. `details` : Additional information about the error.	This error occurs when the loading process reads the pipeline definition that contains a key with an incorrect value.
OLD02	No class with name `class` exists in the pipeline builder packages. Pipeline cannot run.	`class` : The class that cannot be found.	This error occurs when the loading process reads the pipeline definition that contains an unknown class.

© 2026 Ocient, Inc.

Code	Error Message	Message Parameters	Description
OLE00	Failed to extract record with index `index` from `sourceName`. Details: `details`	`index` : The index of the record (index starts at 1) in the source where the error occurred. `sourceName` : The name of the source resource where the error occurred. `details` : Additional information about the error.	This error occurs when the extraction of a record fails.
OLE01	`dataExtract` cannot be extracted from `sourceData`. Details: `details`	`dataExtract` : The data to extract. `sourceData` : The source data for the extraction. `details` : Additional information about the error	This error occurs when the extraction of data from a record fails.
OLE02	The source data has encountered invalid data or character sets. If invalid data is expected in the load, turn off the VALIDATE_CHARACTERS option or use the REPLACE_INVALID_CHARACTERS option instead.	None	This error occurs when the pipeline has the VALIDATE_CHARACTERS option turned on and the load encounters invalid data.
OLM00	The `message` is too large. `modification`	`message` : The type of message that is too large. `modification` : An example or description of a modification to limit the payload size of the message.	This error occurs when an internal communication fails because the message size exceeds the maximum size set by the system.

Code	Error Message	Message Parameters	Description
OLO00	Failed to connect to `sourceType` source because the specified `locationType` `location` was not found. Details: `details`	`sourceType` : The type of the source. `locationType` : The type of the data location argument that was not found. `location` : The value of the data location argument that was not found. `details` : Additional information about the error.	This error occurs when the pipeline cannot find the source data (e.g., cannot find the configured bucket or file path).
OLO01	Failed to connect to `sourceType` source because the access credentials are unauthorized. Details: `details`	`sourceType` : The type of source where access is denied. `details` : Additional information about the error.	This error occurs when the pipeline cannot connect to the source due to invalid credentials.
OLO10	File list is empty at the `sourceType` source provided. The locations searched are: `location` . Details: `details`	`sourceType` : The type of the source. `location` : The value of the data location argument that was not found.	This error occurs when the pipeline cannot find the source data (e.g., the file path is empty).
OLO50	File `fileName` was not found.	`fileName` : The name of the missing file.	This error occurs when the pipeline is configured to fail on missing files, and it cannot find a file in the list.
OLS00	Failed to load transformed value `value` to sink after `transformName` . Details: `details`	`value` : The value that the Loader Node could not load to the sink. `transformName` :	This error occurs when the Loader Node cannot load a transformed value to the sink. For example,

Code	Error Message	Message Parameters	Description
		The name of the executed transform. `details` : Additional information about the error.	the transformed value might be too large for the sink column.
OLS97	No table with name `tableName` exists in sink at `ipAddress` : `port` . The available tables are: `existingTables` .	`tableName` : The fully specified name of the specified table. `ipAddress` : The IP address for the Loader Node of the Ocient sink. `port` : The external TCP port for the Loader Node of the Ocient sink. `existingTables` : The list of existing tables in the same database and schema as the specified table.	This error occurs when no table with the specified name exists in the sink, but the database and schema of the table do exist.
OLS98	The column name `columnName` does not exist. The columns available to the pipeline from the `tableName` table are: `existingColumns` .	`columnName` : The name of the column that was not found. `tableName` : The table that was searched. `existingColumns` : The columns in the table that are available to the pipeline.	This error occurs when the target table in the sink does not contain all the columns specified in the pipeline or a column specified as a transform target is not declared in the sink definition of the pipeline. (The database ignores deleted columns. This error does not occur when you load data to a deleted column.)
OLS99	Failed to connect to sink at `ipAddress` : `port` . Check that a streamloader at that address is reachable.	`ipAddress` : The IP address for the Loader Node of the Ocient sink. `port` : The external TCP port for the Loader	This error occurs when the pipeline cannot connect to its Ocient sink using a Loader Node.

Code	Error Message	Message Parameters	Description
		Node of the Ocient sink.	
OLT00	Cannot perform operation `operationName` because the inputs are `inputType`. Expected inputs are `expectedType`.	`operationName`: The name of the operation that cannot be performed because its inputs are the wrong type. (usually the name of a transform) `inputType`: The type of inputs that the operation received. `expectedType`: The type of inputs that the operation expects to receive.	This error occurs when transforms in a column are not composed correctly.
OLT01	Failed to transform value `value` using `transformName`. Details: `details`	`value`: The value that was not transformed. `transformName`: The name of the failed transform. `details`: Additional information about the error.	This error occurs when a transformation fails.
OLT02	Value `value` is not an acceptable `parameterName` for `transformName`. Expected `example`.	`value`: The incorrect value for the specified parameter. `parameterName`: The name of the specified parameter. `transformName`: The name of the transform that is not configured correctly. `example`: An example or description of the correct value.	This error occurs when the parameter of the transform is not configured correctly.

Code	Error Message	Message Parameters	Description
OLT03	Failed to transform value `value` using `transformName` because the value is `valueDescription`. Expected `example`.	`value`: The value that was not transformed. `transformName`: The name of the failed transform. `valueDescription`: A feature or description of the incorrect value for this transform. `example`: An example or description of the correct value.	This error occurs when a transformation fails because the data is not correct for the transform.
OLT04	Failed to match value `value` to format `format` in `transformName` because `valuePart` at position `valuePartPosition` does not match pattern `pattern`. Expected `example`.	`value`: The value that was not parsed with the transform format. `format`: The transform format. `transformName`: The name of the failed transform. `valuePart`: The part of the value that was unmatched. `valuePartPosition`: The position (index starts at 1) of the part unmatched in the value. `pattern`: The pattern in the transform format that did not match part of the value. `example`: An example or description of the correct value.	This error occurs when a transformation fails because the value does not match the specified transform format.
OLTM0	Failed to transform `value` value using `transformName`. `details`.	`value`: The value that was not transformed. `transformName`: The name of the	This error occurs when an EXPLODE transformation fails.

Code	Error Message	Message Parameters	Description
		failed transform. `details` : Additional information about the error.	
OL998	An error occurred in library `library` : `details`	`library` : The name of the library where the error occurred. `details` : Additional information about the error.	This error occurs when the Ocient System detects an error from an internal library.
OL999	An error occurred in class `class` : `details`	`class` : The class that threw the exception. `details` : Additional information about the error.	This error occurs when the Ocient System detects a general error.

Related Links

Errors and Warnings

Monitor Data Pipelines

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Reserved Words

Keywords in Ocient® are specific words that the database can interpret either as SQL syntax or unquoted identifiers. You cannot use a subset of them, named reserved words, as unquoted identifiers without introducing parsing ambiguities in the database. As a result, when you use these words as an identifier, you must surround the reserved word with double quotes.

You can find these words in the sys.reserved_words system catalog table.

A

ADD | ALL | ALTER | AND | ANTI | ANY | ARRAY | AS

B

BALANCE | BETWEEN

C

CACHE | CALL | CANCEL | CASE | CAST | CLASS | CLUSTERING | COLUMN | COLVECTOR | COMMENT | COMPRESSION | COUNT | CREATE | CROSS | CURRENT | CURRENT_DATABASE | CURRENT_DATE | CURRENT_SCHEMA | CURRENT_TIME | CURRENT_TIMESTAMP | CURRENT_USER

D

DATA | DATABASE | DEFAULT | DELETE | DISABLE | DISTINCT | DROP

E

ELSE | ENABLE | EQUALS | ERROR | EXCEPT | EXECUTE | EXISTING | EXISTS | EXPLAIN | EXPORT | EXTRACT

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

F

FALSE | FIRST | FOLLOWING | FROM | FULL | FUNCTION

G

GDC | GRANT

H

HAVING

I

IGNORE | ILIKE | IN | INDEX | INNER | INSERT | INTERSECT | INTERVAL | INTO | INVALIDATE | IS

J

JOIN

K

KILL

L

LAST | LATERAL | LEFT | LIKE | LOAD | LOADERS

M

MATRIX | MLMODEL | MOVE

N

NODE | NOT | NULL | NULLS

O

OFFSET | ON | OPTION | OPTIONS | ORDER | OUTER | OVER

P

PARTITION | PRAGMA | PRECEDING | PROCEDURE

R

RANGE | REDUNDANCY | REFRESH | REGEX | REMOVE | RENAME | RESET | REVOKE | RIGHT | ROLE | ROW | ROWS | ROWVECTOR

S

SEGMENT | SELECT | SEMI | SERVICE | SET | SHOW | SIMILAR | SOME | START | STOP | STORAGESPACE | SUMMARY_STATS | SYSAUTH

T

TABLE | TAG | TASK | THEN | TO | TRACE | TRUE | TRUNCATE | TYPE

U

UNBOUNDED | UNION | USE | USING

V

VIEW

W

WHEN | WHERE | WITH

Z

© 2026 Ocient, Inc.

Related Links

SQL Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

System Catalog Reference

The Ocient® System Catalog comprises a set of virtual tables and views. Together, these present administrators with metadata that describes the state of the data stored in the system, activity on the system, and the configuration of the system. The `information_schema` schema contains views that present information joined from multiple system catalog tables.

- **System Catalog** — Listing and description of all system catalog tables included in the system.
- **Information Schema** — Listing and description of all `information_schema` schema views and `SHOW` commands defined by default.

To explore the system catalog tables, use sample queries for various topics as described in **Discover Insights From System Catalog Tables**.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

System Catalog

The system catalog exposes read-only virtual tables that contain metadata about the system.

Alphabetical List of Tables

- sys.active_operator_instances
- sys.active_query_scheduler_info
- sys.addendum_directories
- sys.all_operator_instances
- sys.association_rules_models
- sys.average_bb_sizes
- sys.average_column_sizes
- sys.bagging_models
- sys.boosting_models
- sys.channel_endpoint_parameters
- sys.clusters
- sys.column_cardinalities
- sys.column_distributions
- sys.columns
- sys.columns_compression_info
- sys.completed_operator_instances
- sys.completed_queries
- sys.compute_configurations
- sys.config
- sys.connectivity_pool_participants
- sys.connectivity_pools
- sys.constraints

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

- sys.current_osn
- sys.data_usage
- sys.databases
- sys.decision_tree_models
- sys.degraded_segment_groups
- sys.feedforward_network_models
- sys.function_signatures
- sys.functions
- sys.gaussian_mixture_models
- sys.global_map_table_info
- sys.gradient_boosted_trees_models
- sys.group_roles
- sys.groups
- sys.index_advisor_enabled
- sys.index_advisors
- sys.index_columns
- sys.index_recommendations
- sys.index_usage
- sys.indexes
- sys.k_means_models
- sys.k_nearest_neighbors_models
- sys.linear_combination_regression_models
- sys.linear_discriminant_analysis_models
- sys.load_errors
- sys.load_events
- sys.load_metrics
- sys.locations
- sys.locks
- sys.logistic_regression_models

© 2026 Ocient, Inc.

- sys.lts_cluster_info
- sys.machine_learning_model_options
- sys.machine_learning_models
- sys.mcp_tools
- sys.merge_eligibilities
- sys.merge_policies
- sys.metric_levels
- sys.multiple_linear_regression_models
- sys.naive_bayes_models
- sys.network_interface_models
- sys.network_interface_usage_types
- sys.node_clusters
- sys.node_config
- sys.node_network_interfaces
- sys.node_status
- sys.nodes
- sys.nonlinear_regression_models
- sys.oidc_integrations
- sys.oidc_sessions
- sys.op_inst_debug_info
- sys.orphaned_segments
- sys.osn_acquisitions
- sys.pipeline_errors
- sys.pipeline_errors_historical
- sys.pipeline_events
- sys.pipeline_events_historical
- sys.pipeline_files
- sys.pipeline_files_historical
- sys.pipeline_files_pending

© 2026 Ocient, Inc.

- sys.pipeline_functions
- sys.pipeline_metrics
- sys.pipeline_metrics_historical
- sys.pipeline_metrics_info
- sys.pipeline_partitions
- sys.pipeline_tables
- sys.pipeline_tasks
- sys.pipelines
- sys.plans
- sys.polynomial_regression_models
- sys.principal_component_analysis_models
- sys.privileges
- sys.procedures
- sys.queries
- sys.random_forest_models
- sys.regression_tree_models
- sys.reserved_words
- sys.result_cache
- sys.retention_policies
- sys.rights
- sys.roles
- sys.sample_segment_io_pipelines
- sys.scheduled_tasks
- sys.schemas
- sys.security_integrations
- sys.security_settings
- sys.segment_batch_events
- sys.segment_directories
- sys.segment_generation_events

© 2026 Ocient, Inc.

- sys.segment_group_transfers
- sys.segment_groups
- sys.segment_part_inventory
- sys.segment_part_redundancy_info
- sys.segment_parts
- sys.segments
- sys.segments_compression_info
- sys.service_classes
- sys.service_classes_for_user
- sys.service_role_channel_endpoints
- sys.service_role_status
- sys.service_roles
- sys.sessions
- sys.simple_linear_regression_models
- sys.sql_messages
- sys.stacking_models
- sys.stats_files
- sys.stop_osn_reap_requests
- sys.storage_capacity
- sys.storage_device_files
- sys.storage_device_metrics
- sys.storage_device_status
- sys.storage_scopes
- sys.storage_spaces
- sys.storage_used
- sys.stored_leaf_segment_part_inventory
- sys.stored_segments
- sys.subtasks
- sys.support_vector_machine_models

© 2026 Ocient, Inc.

- sys.system_information
- sys.system_table_columns
- sys.system_tables
- sys.table_cardinalities
- sys.tables
- sys.tasks
- sys.tkt_table_clustering_column_indices
- sys.tkt_table_info
- sys.unhealthy_segments
- sys.user_groups
- sys.user_mappings
- sys.user_roles
- sys.users
- sys.vector_autoregression_models
- sys.view_columns
- sys.views
- sys.vl_columns_compression_info

Alphabetical List of Categories

- Configuration
- Connectivity Pools
- Databases
- Loading
- Machine Learning
- Model Context Protocol
- Monitoring
- Network
- Pipelines
- Security

- Security Integrations
- Statistics
- Storage
- System
- System Information
- User Management
- Workload Management

Catalog Table Details

Configuration

sys.config

This table contains all configuration overrides applied on the system.

Column Name	Column Type	Column Description
scope_type	CHAR	Type of the scope.
scope_id	UUID	Universally Unique IDentifier (UUID) of the storage scope where this override applies.
key	CHAR	String that represents the configuration parameter.
value	CHAR	Value for the configuration parameter key.

sys.node_config

This table contains the effective node configuration for all nodes in the system.

Column Name	Column Type	Column Description
node_id	UUID	Universally Unique IDentifier (UUID) of the node where this configuration exists (sys.nodes).
key	CHAR	String that represents the configuration parameter.
value	CHAR	Value for the configuration parameter key.

© 2026 Ocient, Inc.

Connectivity Pools

sys.connectivity_pool_participants

This table contains the connectivity pool participants and their information.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the connectivity pool that contains the participant.
node_id	UUID	UUID of the participant node.
created_at	TIMESTAMP	Timestamp that indicates the creation of the connectivity pool participant.
updated_at	TIMESTAMP	Timestamp that indicates the last update of the connectivity pool participant.
listen_address	CHAR	Address where the participant node listens.
listen_port	INT	Port number where the participant node listens.
advertised_address	CHAR	Address that the participant node advertises to the client.
advertised_port	INT	Optional port number that the participant node advertises to the client.
openapi_port	INT	Optional port number where the API, written according to the OpenAPI Specification (OAS), listens.

sys.connectivity_pools

This table contains the connectivity pools and their information (excluding participants).

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the connectivity pool.
name	CHAR	Name of the connectivity pool.
created_at	TIMESTAMP	Timestamp that indicates the creation of the connectivity pool.
updated_at	TIMESTAMP	Timestamp that indicates the last update of the connectivity pool.
source_address	CHAR	Source address in CIDR notation for the connectivity pool.

Column Name	Column Type	Column Description
source_port	INT	Optional source port of the connectivity pool.
priority	INT	Priority of the connectivity pool.
sso_integration_name	CHAR	Name of the default SSO security integration for the connectivity pool.

Databases

sys.columns

This table contains all columns in each table in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the column.
name	CHAR	Name of the column.
data_type	CHAR	Data type of the column (INT, CHAR, BOOLEAN, etc.).
position	INT	Position of the column in its table.
table_id	UUID	UUID of the table that contains this column (sys.tables).
nullable	BOOLEAN	Specifies whether the values in this column can be NULL.
default_expression	CHAR	Default value of this column when the value is unspecified.
description	CHAR	Description of the column.
type_hint	CHAR	Type hint for the expected type of the column.
ordinal	LONG	Ordinal of the column with respect to its table.
gdc_type	CHAR	Type of GDC applied to this column.
potential_index	BOOLEAN	Specifies whether the column is a potential index.
system	BOOLEAN	Specifies whether the column is a system column.

sys.constraints

This table contains the constraints for the database.

Column Name	Column Type	Column Description
id	UUID	The Universally Unique IDentifier (UUID) of the constraint.
table_id	UUID	The UUID of the table (sys.tables).
name	CHAR	The name of the constraint that has to be unique within an individual table.
constraint_type	CHAR	The type of the constraint (PRIMARY_KEY, UNIQUE, FOREIGN_KEY).
column_names	ARRAY(CHAR)	The names of the columns where the constraint applies.
referenced_constraint_id	UUID	The UUID of the referenced constraint. This value is NULL for non-foreign key constraints.
referenced_table_id	UUID	The UUID of the referenced table. This value is NULL for non-foreign key constraints.
referenced_column_names	ARRAY(CHAR)	The names of the columns where the referenced constraint applies. This value is NULL for non-foreign key constraints.

sys.databases

This table contains all databases defined in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the database.
name	CHAR	Name of the database.
created_at	TIMESTAMP	Timestamp that specifies when the database was created.
user_can_view_all_queries	BOOLEAN	Current user can view all queries made in this database

sys.functions

This table contains all of the user-defined functions in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the user-defined function.
schema	CHAR	Schema of the user-defined function.
name	CHAR	Name of the user-defined function.
sql_expression	CHAR	For SQL user-defined functions, this is the expression that defines the function.
database_id	UUID	UUID of the database of this user-defined function (sys.databases).
created_at	TIMESTAMP	Timestamp of when this user-defined function was created.
updated_at	TIMESTAMP	Timestamp of when this user-defined function was last updated.
language	CHAR	Language that the user-defined function is written in.
function_type	CHAR	For non-SQL user-defined functions, the type of function being defined.
filename	CHAR	For non-SQL user-defined functions, the name of the file containing the user-defined function code.
function_name_in_external_code	CHAR	For non-SQL user-defined functions, the name of the Java or Python function or method that defines the user-defined function.
return_type	CHAR	For non-SQL user-defined functions, the return type of the user-defined function.

sys.global_map_table_info

This table contains the global dictionary compression (GDC) details associated with each compressed column.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
column_id	UUID	UUID of the column (sys.columns).

Column Name	Column Type	Column Description
compressed_size	INT	Compressed column size (in bytes).
max_count	LONG	Maximum amount of unique column values allowed in the column.
current_count	LONG	Current amount of unique column values in the column.

sys.index_advisor_enabled

This table contains all tables and databases where the index advisor is currently enabled.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the table or database.
name	CHAR	Name of the table or database.
object_type	CHAR	Determines whether this object is a database or table.

sys.index_advisors

This table contains raw index advisor information.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table for the potential index (sys.tables).
column_id	UUID	UUID that represents the column (sys.columns).
table_name	CHAR	Name of the table.
column_name	CHAR	Name of the column.
index_type	CHAR	Type of the potential index.
query_id	UUID	UUID that represents the query.
query_time	TIMESTAMP	A UNIX timestamp that represents when the query executed.
sql	CHAR	The SQL statement to use for creating this index.

© 2026 Ocient, Inc.

sys.index_columns

This table contains the index settings applied to each column.

Column Name	Column Type	Column Description
index_id	UUID	Universally Unique IDentifier (UUID) that represents the index (sys.indexes).
column_id	UUID	UUID that represents the column (sys.columns).
ordinal	INT	Ordinal of the column in the index structure.
tuple_element	CHAR	Specifies the name of the component column that the index applies to if the index exists on a tuple column.
ascending	BOOLEAN	Indicates whether the values in this column are built in ascending order.
column_ordinal	LONG	Ordinal position of this column in the source table.

sys.index_recommendations

This table contains index recommendations.

Column Name	Column Type	Column Description
table_name	CHAR	Name of the table.
column_name	CHAR	Name of the column.
sql	CHAR	The SQL statement to use for creating this index.
usage_count	LONG	Number of times this index would be used.

sys.index_usage

This table contains index usage information.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table where the index is located.
index_id	UUID	UUID of the index.
segment_id	UUID	UUID of the segment with the index.

Column Name	Column Type	Column Description
query_id	UUID	UUID of the query that could have used the index.
was_used	BOOLEAN	Whether or not the index was used.

sys.indexes

This table contains all indexes defined on tables.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the index.
name	CHAR	Name of the index.
index_type	CHAR	Type of the index.
table_id	UUID	UUID of the table for the index (sys.tables).
index_use	CHAR	Intended use for the index by the system.
ngram_size	INT	Size of each N-gram (in bytes).
enabled	BOOLEAN	Whether the index is enabled on this table.

sys.locations

This table contains all of the external table providers.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the location.
name	CHAR	Name of the location.
remote_info	CHAR	Remote information that defines how to talk to the external table provider.
database_id	UUID	UUID of the database of this location (sys.databases).
created_at	TIMESTAMP	Timestamp that represents when this location was created.
updated_at	TIMESTAMP	Timestamp that represents when this location was last updated.

© 2026 Ocient, Inc.

sys.procedures

This table contains all of the stored procedures in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the stored procedure.
schema	CHAR	Schema of the stored procedure.
name	CHAR	Name of the stored procedure.
database_id	UUID	UUID of the database of this stored procedure (sys.databases).
created_at	TIMESTAMP	Timestamp of when this stored procedure was created.
updated_at	TIMESTAMP	Timestamp of when this stored procedure was last updated.
language	CHAR	Language that the stored procedure is written in.
filename	CHAR	For Java® and Python® stored procedures, the name of the file that the database should load when it needs this stored procedure.
function_name	CHAR	For Java® and Python® stored procedures, the name of the function that executes the stored procedure.
body	CHAR	For SQL stored procedures, the SQL statements to execute when the database runs this stored procedure.
returns_result	BOOLEAN	For SQL stored procedures, the system sets this value to `true` if the stored procedure returns a result set. The default value is `false`.

sys.tables

This table contains all tables defined in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
name	CHAR	Name of the table.
schema	CHAR	Name of the schema.

Column Name	Column Type	Column Description
database_id	UUID	UUID of the database (sys.databases).
storage_space_id	UUID	UUID of the storage space (sys.storage_spaces).
maximum_segment_size_gib	INT	Maximum size of a segment for the table in GiB.
description	CHAR	Detailed description of the table.
streamloader_property_string	CHAR	Additional properties assigned to control the stream loading behavior for the table.
lts_property_string	CHAR	Additional properties assigned to control the Foundation role (formerly “lts role”) behavior for the table.
created_at	TIMESTAMP	Timestamp that represents when the table was created.
altered_at	TIMESTAMP	Timestamp that represents when the table’s structure was last modified via DDL.
rolehostd_version	CHAR	The rolehostd version in which this table was created
software_compatible_version	INT	The rolehostd version this table is compatible with.
creator_id	UUID	The UUID of the user or group who created the table (sys.users).

sys.tkt_table_clustering_column_indices

This table contains the columns defined as part of a clustering index.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
column_id	UUID	UUID of the column (sys.columns).

sys.tkt_table_info

This table contains the columns that are defined as the time key on each user-defined table and the size of the time bucket in use.

Column Name	Column Type	Column Description
table_id	UUID	The Universally Unique IDentifier (UUID) of the table (sys.tables).
time_column_id	UUID	The UUID of the column that represents the time column in the table (sys.columns).
time_bucket_width	LONG	The width of the time bucket for segmenting data in nanoseconds or days.
time_bucket_width_interval	CHAR	The readable interval for the width of the time bucket for segmenting data.
retention_policy	LONG	The interval for the retention policy for rolling off data in nanoseconds or days.
retention_policy_interval	CHAR	The readable interval of the retention policy for rolling off data.

sys.user_mappings

This table contains user mappings for locations.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the user mapping.
name	CHAR	Name of the user mapping.
local_userid	CHAR	Local user identifier.
remote_userid	CHAR	Remote user identifier.
location_id	UUID	UUID of the location for this user mapping.
database_id	UUID	UUID of the database for this user mapping (sys.databases).
created_at	TIMESTAMP	Timestamp that represents when this user mapping was created.
updated_at	TIMESTAMP	Timestamp that represents when this user mapping was last updated.
remote_password	CHAR	Password for the remote user identifier.

sys.view_columns

This table contains all columns in each view in the system.

Column Name	Column Type	Column Description
id	UUID	The Universally Unique IDentifier (UUID) of the column.
name	CHAR	The name of the column.
data_type	CHAR	The data type of the column.
view_id	UUID	The UUID of the view this column comes from.
nullable	BOOLEAN	Whether or not this column is nullable.
default_expression	CHAR	The default expression of this column, if it exists.
ordinal	LONG	The ordinal of this column.
global_dictionary_compression_table_id	UUID	UUID of the table with the GDC column (sys.tables).

sys.views

This table contains all user-defined database views created on the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the view.
name	CHAR	Name of the view.
database_id	UUID	UUID of the database (sys.databases).
schema	CHAR	Schema name where the view exists.
query	CHAR	Query used to generate the view content.
description	CHAR	Detailed description of the view.
global_dictionary_compression_table_id	UUID	UUID of the table with the GDC column (sys.tables).
created_at	TIMESTAMP	Timestamp that represents the date and time for the creation of the view.

Column Name	Column Type	Column Description
updated_at	TIMESTAMP	Timestamp that represents the date and time for the last update of the view.
creator_id	UUID	The UUID of the user or group who created the view (sys.users/sys.groups).

Loading

sys.load_errors

The `load_errors` table shows information for load errors in the system.

Column Name	Column Type	Column Description
node_id	UUID	The unique identifier of the Loader Node.
table_id	UUID	If the event is specific to a table, the unique identifier of the table.
error_message	VARCHAR	The message that provides more details about the error.
error_timestamp	TIMESTAMP	The timestamp when this error occurred.
load_id	UUID	If the event is specific to a pipeline or query, the Universally Unique IDentifier (UUID) of the pipeline or query.

sys.load_events

The `load_events` table shows information for the load events in the system.

Column Name	Column Type	Column Description
node_id	UUID	The unique identifier of the Loader Node.
table_id	UUID	If the event is specific to a table, the unique identifier of the table.
event_type	VARCHAR	The type of the event.
event_timestamp	TIMESTAMP	The timestamp when this event occurred.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
load_id	UUID	If the event is specific to a pipeline or query, the unique identifier of the pipeline or query.
batch_id	UUID	If the event is specific to segment generation, this column is the unique identifier for the batch of pages specified by the system for segment generation.
event_message	VARCHAR	The message that provides more details about the event.

sys.load_metrics

The `load_metrics` table shows information for the load metrics in the system.

Column Name	Column Type	Column Description
node_id	UUID	The Universally Unique IDentifier (UUID) of the Loader Node (sys.nodes).
table_id	UUID	If the metric is specific to a table, the UUID of the table (sys.tables).
load_id	UUID	If the metric is specific to a data pipeline or query, the UUID of the data pipeline or query.
name	VARCHAR	The type of the metric.
component	VARCHAR	The component of the metric.
value	BIGINT	The value of the metric.
updated_at	TIMESTAMP	Timestamp that represents when this metric was updated.

sys.segment_batch_events

This table contains information about segment batches during segment generation.

Column Name	Column Type	Column Description
batch_id	UUID	The Universally Unique IDentifier (UUID) for the batch.
table_id	UUID	The UUID for the table.
num_partitions	BIGINT	The number of partitions in this batch.
bucket_min	BIGINT	The size in bytes of the smallest bucket in this batch.

Column Name	Column Type	Column Description
bucket_max	BIGINT	The size in bytes of the largest bucket in this batch.
scope_id	UUID	The UUID for the scope of this batch.
batch_start_time	TIMESTAMP	The start time of the batch analysis.
batch_end_time	TIMESTAMP	The end time of the batch analysis, including completion and partitioning.
batch_flush_reason	VARCHAR	The reason for the trigger of this batch analysis.
size	BIGINT	The total size of this batch, in bytes.

sys.segment_generation_events

This table contains information about individual segments during segment generation.

Column Name	Column Type	Column Description
batch_id	UUID	The Universally Unique IDentifier (UUID) for the batch.
table_id	UUID	The UUID for the table.
ida_offset	BIGINT	The index number of the partition in this batch.
start_time	TIMESTAMP	The start time of segment analysis.
end_time	TIMESTAMP	The end time of segment analysis, including completion and partitioning.
size	BIGINT	The total size of this segment, in bytes.
num_rows	BIGINT	The total number of rows in this segment.

Machine Learning

sys.association_rules_models

This table contains the name of the snapshot table for an association rules model.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).

Column Name	Column Type	Column Description
num_arguments	INT	Number of arguments in the machine learning model data.
table_name	CHAR	Name of the snapshot table.

sys.bagging_models

This table contains all model parameters for Bagging Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
aggregate_expression	CHAR	The vote expression that retrieves information from the child models.
num_features	INT	Number of arguments in the machine learning model data.
num_children	INT	Number of child models used by the bagging model.
correctly_classified	DOUBLE	Percentage of training data that the bagging model correctly classifies.
area_under_roc	DOUBLE	Area under the ROC curve of the bagging model.
rmse	DOUBLE	Root mean square error of the regression.
adjusted_r2	DOUBLE	Adjusted R^2 value of the regression.

sys.boosting_models

This table contains all model parameters for Boosting Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
expression	CHAR	The SQL expression that defines the boosting model.
num_arguments	INT	Number of arguments in the machine learning model data.

Column Name	Column Type	Column Description
learning_rate	DOUBLE	Learning rate.
classes	ARRAY(CHAR)	Class names.
loss_function	CHAR	Loss function.
initial_predictions	ARRAY(DOUBLE)	Initial predictions.
correctly_classified	DOUBLE	Fraction of training data that the model correctly classifies.
area_under_roc	DOUBLE	Area under the ROC curve of the model.
rmse	DOUBLE	Root mean square error of the model.
adjusted_r2	DOUBLE	Adjusted R^2 of the model.

sys.decision_tree_models

This table contains all model parameters for Decision Tree Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
case_statement	CHAR	The case statement that defines the decision tree.
case_statement_confidence	CHAR	The case statement that defines the decision tree and returns confidences.
num_arguments	INT	Number of arguments in the machine learning model data.
correctly_classified	DOUBLE	Fraction of training data that the decision tree model correctly classifies.
area_under_roc	DOUBLE	Area under the ROC curve of the decision tree model.
argument_types	ARRAY(CHAR)	Argument type names.
output_type	CHAR	Output type name.
disable_jit	BOOLEAN	Determines whether to disable JIT for the decision tree. Set this value to `true` to disable JIT. The default value is `false`.

sys.feedforward_network_models

This table contains all model parameters for Feedforward Neural Network Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the feedforward network model.
num_arguments	INT	Number of arguments in the machine learning model data.
average_loss	DOUBLE	Average value of the loss function.

sys.gaussian_mixture_models

This table contains all model parameters for Gaussian Mixture Models.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the Gaussian Mixture Model.
num_distributions	INT	Number of distributions in the Gaussian Mixture Model.
average_loss	DOUBLE	Average value of the loss function.

sys.gradient_boosted_trees_models

This table contains all model parameters for Gradient Boosted Trees Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).

Column Name	Column Type	Column Description
expression	CHAR	The SQL expression that defines the gradient boosted tree.
num_arguments	INT	Number of arguments in the machine learning model data.
r2	DOUBLE	Coefficient of determination.
rmse	DOUBLE	Root mean square error of the model.
adjusted_r2	DOUBLE	Adjusted R^2 of the model.
correctly_classified	DOUBLE	Fraction of training data that the gradient boosted trees model correctly classifies.
initial_predictions	ARRAY(DOUBLE)	Initial predictions.
learning_rate	DOUBLE	Learning rate.
loss_function	CHAR	Loss function.
classes	ARRAY(CHAR)	Class names.

sys.k_means_models

This table contains all model parameters for K-Means Clustering Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
centroids	ARRAY(DOUBLE)	Values that represent the centroid values of each cluster.
num_arguments	INT	Number of arguments in the machine learning model data.
means	ARRAY(DOUBLE)	Means of each of the feature columns in the data.
standard_deviations	ARRAY(DOUBLE)	Standard deviations of each of the feature columns in the data.

sys.k_nearest_neighbors_models

This table contains the number of arguments and the target table for K-Nearest Neighbors Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
num_arguments	INT	Number of arguments in the machine learning model data.
table_name	CHAR	Name of the target table for the K Nearest Neighbors (KNN) model.
correctly_classified	DOUBLE	Percentage of training data that is correctly classified by the KNN model.
means	ARRAY(DOUBLE)	Means of each of the feature columns in the data.
standard_deviations	ARRAY(DOUBLE)	Standard deviations of each of the feature columns in the data.

sys.linear_combination_regression_models

This table contains all model parameters for Linear Combination Regression Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of model coefficients associated with the regression.
y_intercept	DOUBLE	y-intercept of the regression.
coefficient_of_determination	DOUBLE	R^2 value of the regression, if calculated, otherwise NULL.
num_arguments	INT	Number of arguments in the machine learning model data.
rmse	DOUBLE	Root mean square error of the regression.
adjusted_r2	DOUBLE	Adjusted R^2 value of the regression.

sys.linear_discriminant_analysis_models

This table contains all model parameters for Linear Discriminant Analysis (LDA) Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the LDA model.
num_features	INT	Number of features returned by the LDA model.
importance	ARRAY(DOUBLE)	Importance of each of the output features that represent the data.

sys.logistic_regression_models

This table contains all model parameters for Logistic Regression Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the logistic regression model.
correctly_classified	DOUBLE	Percentage of training data that is correctly classified by the logistic regression.
num_arguments	INT	Number of arguments in the machine learning model data.
zero_case	CHAR	The case where the logistic regression uses a value of 0.
one_case	CHAR	The case where the logistic regression uses a value of 1.
classes	ARRAY(CHAR)	Target classes for the logistic regression.

sys.machine_learning_model_options

This table contains options defined for each machine learning model. The database stores these options as key-value pairs.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	UUID of the machine learning model (sys.machine_learning_models).
machine_learning_model_key	CHAR	Machine learning model configuration option key.
machine_learning_model_value	CHAR	Machine learning model configuration option value.

sys.machine_learning_models

This table contains the machine learning models that are defined in the database.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the machine learning model.
name	CHAR	Name of the machine learning model.
schema	CHAR	Schema of the machine learning model.
machine_learning_model_type	CHAR	Type of the machine learning model.
database_id	UUID	UUID of the database that contains the machine learning model definition (sys.databases).
on_select	CHAR	The SELECT SQL statement used to create this model.
creator_id	UUID	The UUID of the user or group who created the machine learning model (sys.users/sys.groups).

sys.multiple_linear_regression_models

This table contains the y-intercept and coefficient of determination values for the multiple linear regression machine learning models.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
y_intercept	DOUBLE	y-intercept of the regression.

Column Name	Column Type	Column Description
coefficient_of_determination	DOUBLE	R^2 value of the regression, if calculated, otherwise NULL.
rmse	DOUBLE	Root mean square error of the regression.
adjusted_r2	DOUBLE	Adjusted R^2 value of the regression.
slopes	ARRAY(DOUBLE)	Slopes for each variable in the regression.

sys.naive_bayes_models

This table contains all model parameters for Naive Bayes Classification Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
num_arguments	INT	Number of arguments in the machine learning model data.
result_probability_table	CHAR	Name of the internal result probability table.
feature_result_matrix_table	CHAR	Name of the internal feature result matrix table.
correctly_classified	DOUBLE	Percentage of training data that is correctly classified by the Naive Bayes model.

sys.nonlinear_regression_models

This table contains all model parameters for Nonlinear Regression Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the nonlinear regression model.
coefficient_of_determination	DOUBLE	R^2 value of the regression, if calculated, otherwise NULL.

Column Name	Column Type	Column Description
num_arguments	INT	Number of arguments in the machine learning model data.
rmse	DOUBLE	Root mean square error of the regression.
adjusted_r2	DOUBLE	Adjusted R^2 value of the regression.

sys.polynomial_regression_models

This table contains all model parameters for Polynomial Regression Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of model coefficients associated with the polynomial regression model.
y_intercept	DOUBLE	y-intercept of the regression.
coefficient_of_determination	DOUBLE	R^2 value of the regression, if calculated, otherwise NULL.
num_arguments	INT	Number of arguments in the machine learning model data.
rmse	DOUBLE	Root mean square error of the regression.
adjusted_r2	DOUBLE	Adjusted R^2 value of the regression.

sys.principal_component_analysis_models

This table contains all model parameters for Principal Component Analysis (PCA) Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the PCA model.
num_features	INT	Number of features returned by the PCA model.

Column Name	Column Type	Column Description
importance	ARRAY(DOUBLE)	Importance of each of the output features that represents the data.
means	ARRAY(DOUBLE)	Means of each of the feature columns in the data.
standard_deviations	ARRAY(DOUBLE)	Standard deviations of each of the feature columns in the data.

sys.random_forest_models

This table contains all model parameters for Random Forest Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
vote_expression	CHAR	The vote expression that retrieves information from the child decision trees.
vote_expression_confidence	CHAR	The vote expression that retrieves information from the child decision trees and returns confidences.
num_arguments	INT	Number of arguments in the machine learning model data.
num_children	INT	Number of child trees in the forest.
correctly_classified	DOUBLE	Percentage of training data that the random forest model correctly classifies.
area_under_roc	DOUBLE	Area under the ROC curve of the random forest model.

sys.regression_tree_models

This table contains all model parameters for Regression Tree Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).

Column Name	Column Type	Column Description
case_statement	CHAR	The case statement that defines the regression tree.
num_arguments	INT	Number of arguments in the machine learning model data.
r2	DOUBLE	Coefficient of determination.
rmse	DOUBLE	Root mean square error.
adjusted_r2	DOUBLE	R^2.

sys.simple_linear_regression_models

This table contains the slope, y-intercept, and coefficient of determination values for single linear regression machine learning models.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
slope	DOUBLE	Slope of the regression.
y_intercept	DOUBLE	The y-intercept of the regression.
coefficient_of_determination	DOUBLE	R^2 value of the regression, if calculated, otherwise NULL.
rmse	DOUBLE	Root mean square error of the regression.
adjusted_r2	DOUBLE	Adjusted R^2 value of the regression.

sys.stacking_models

This table contains all model parameters for Stacking Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
num_arguments	INT	Number of arguments in the machine learning model data.

sys.support_vector_machine_models

This table contains all model parameters for Support Vector Machine (SVM) Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the SVM model.
correctly_classified	DOUBLE	Percentage of training data that is correctly classified by the SVM model.
num_arguments	INT	Number of arguments in the machine learning model data.
negative_case	CHAR	Case when the SVM model classifies data as negative.
positive_case	CHAR	Case when the SVM model classifies data as positive.
classes	ARRAY(CHAR)	Target classes for the SVM model.

sys.vector_autoregression_models

This table contains all model parameters for Vector Autoregression Models in the system.

Column Name	Column Type	Column Description
machine_learning_model_id	UUID	Universally Unique IDentifier (UUID) of the machine learning model (sys.machine_learning_models).
coefficients	ARRAY(DOUBLE)	List of coefficients associated with the vector autoregression model.
coefficient_of_determination	DOUBLE	R^2 value of the autoregression, if calculated, otherwise NULL.

Model Context Protocol

sys.mcp_tools

This table contains all of the model context protocol (MCP) tools in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the MCP tool.
schema	CHAR	Schema of the MCP tool.
name	CHAR	Name of the MCP tool.
argument_names	ARRAY(CHAR)	The argument names of the MCP tool, specified as an array.
argument_types	ARRAY(CHAR)	The argument types of the MCP tool, specified as an array.
argument_nullability	ARRAY(BOOLEAN)	Whether the argument types for the MCP tool are nullable, specified as an array.
language	CHAR	Programming language of the MCP tool.
description	CHAR	Description of the MCP tool.
toolset	CHAR	Toolset of the MCP tool.
definition	CHAR	Function definition of the MCP tool.
creator_id	UUID	The UUID of the user or group who created the MCP tool (sys.users/sys.groups).
created_at	TIMESTAMP	The timestamp when this MCP tool was created.

Monitoring

sys.active_operator_instances

This table contains information about operator instances that are active in the database.

Column Name	Column Type	Column Description
database_name	CHAR	The name of the database associated with the query to which the operator belongs.
user_name	CHAR	The username associated with the query to which the operator belongs.
query_id	UUID	A unique identifier of the query to which the operator belongs.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
operator_id	UUID	A unique identifier of the finalized operator.
node_id	UUID	The unique identifier of the node that stores the information about this operator.
silo_id	LONG	The unique identifier of the silo that stores the information about this operator.
core_id	LONG	The unique identifier of the virtual machine core that stores the information about this operator.
op_runtime_id	LONG	The unique identifier for the operator instance that the system assigns at runtime.
op_instance_type	CHAR	The name of the type for this operator instance.
bloom_filtered_rows	LONG	The rows filtered by Bloom filters.
rows_received	LONG	The number of rows fetched by the operator instance.
rows_processed	LONG	The number of rows processed by this operator instance during execution.
rows_emitted	LONG	The number of rows returned by the operator instance.
blocks_emitted	LONG	The number of data blocks returned by the operator instance.
first_block_receive_time	TIMESTAMP	The timestamp that specifies the return of the first data block.
first_block_emitted_time	TIMESTAMP	The timestamp that specifies the return of the first data block.
num_cycles	LONG	The number of cycles that run on this operator instance.

Column Name	Column Type	Column Description
num_oom_cycles	LONG	The number of run cycles that an operator received for processing out-of-memory issues.
num_failed_oom_cycles	LONG	The number of out-of-memory cycles received that failed to do any work.
num_no_work_oom_cycles	LONG	The number of out-of-memory cycles received that reported they had no work to do.
run_count	LONG	The number of times that the scheduler has reviewed this operator and `isRunnable` is set to `true`.
no_work_cycles	LONG	The number of times that the read and write cycle executed and performed no work.
backed_up_cycles	LONG	The number of times a cycle stopped because a backup of the parent exists.
leaf_branch_blocked_cycles	LONG	The number of times a cycle stopped on a leaf operator because of multi-child scheduling heuristics.
num_eof_cycles	LONG	The number of end-of-file cycles received.
normal_run_time_in_ms	DOUBLE	The time, in milliseconds, of the normal run for the operator instance.
dispatch_queue_time_in_ms	DOUBLE	The time spent, in milliseconds, processing incoming message events for the operator instance.
oom_run_time_in_ms	DOUBLE	The time, in milliseconds, of the out-of-memory issue for the operator instance.
max_hp_mem_usage	LONG	The maximum heap memory usage by the operator instance.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
num_blocks_written	LONG	The number of data blocks written to temporary disk.
num_blocks_read	LONG	The number of data blocks read from temporary disk.
num_fragments_written	LONG	The number of written memory fragments.
num_fragments_read	LONG	The number of read memory fragments.
cycles_over_1_sec	LONG	The count of cycles that ran over 1 second for this operator instance.
cycles_over_3_sec	LONG	The count of cycles that ran over 3 seconds for this operator instance.
long_dispatch_queue_events	LONG	The count of dispatch queue events processed by the operator with a runtime greater than 0.5 seconds.
total_bin_stream_allocated_bytes	LONG	The total amount of memory allocated in output blocks for variable-length column data.
total_bin_stream_used_bytes	LONG	The total amount of memory used in output blocks for variable-length column data.
total_col_stream_allocated_bytes	LONG	The total amount of memory allocated in output blocks for fixed-length column data.
total_col_stream_used_bytes	LONG	The total amount of memory used in output blocks for fixed-length column data.
blocked_received	LONG	The total number of input data blocks received by this operator instance.
blocked_created	LONG	The total number of non-trivial output data blocks created by this operator instance.
received_block_bin_stream_unused_space_bytes	LONG	The total amount of wasted space across all binary streams in received data

Column Name	Column Type	Column Description
		blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive.
received_block_col_stream_unused_space_bytes	LONG	The total amount of wasted space across all column streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive.
mandatory_alloc_attempts	LONG	The count of cycles where the scheduler enabled mandatory allocation.
mandatory_alloc_success_count	LONG	The count of cycles where the scheduler enabled mandatory allocation and this operator broke an out-of-memory deadlock.
max_num_pending_blocks	LONG	The maximum number of pending blocks across all partitions during the lifetime of this operator.
max_num_consecutive_backup_cycles	LONG	The maximum number of consecutive scheduler cycles where this operator has a backup. The scheduler can use this information to help detect certain resource deadlocks.
longest_time_since_unblocked_cycle	LONG	The maximum time between successful attempts to run this operator from the scheduler. The scheduler can use this information to help

Column Name	Column Type	Column Description
		detect certain resource deadlocks.
compiled_level	LONG	The level within a hierarchy of distinct operators to execute. A single plan operator can be compiled into this hierarchy.
additional_json	CHAR	Additional statistics for an operator instance in JSON format.
num_bloom_filters_sent	LONG	The number of Bloom filters the operator sent to its peers.
num_bloom_filters_received	LONG	The number of Bloom filters the operator received from its peers.
num_key_lists_sent	LONG	The number of potential index join key lists the operator sent to its peers.
num_key_lists_received	LONG	The number of potential index join key lists the operator received from its peers.
max_hash_table_rows	LONG	The maximum number of rows present in any hash table used by this operator instance.
plan_operator_estimated_output_cardinality	LONG	The number of rows the optimizer estimated for this plan operator to emit.
num_delayed_exceptions_generated	LONG	The number of rows this operator generated with delayed exception information.
num_pending_blocks_spilled	LONG	The number of queued pending data blocks this operator spilled to disk.
max_normal_cycle_time_ms	DOUBLE	The time, in milliseconds, of the longest normal run cycle for the operator instance.
max_oom_cycle_time_ms	DOUBLE	The time, in milliseconds, of the longest out-of-memory

Column Name	Column Type	Column Description
		run cycle for the operator instance.
num_rows_received_per_plan_child	ARRAY(LONG)	The number of rows this operator received from each of its children in the plan.
initialization_time	TIMESTAMP	The timestamp that represents when the operator began work.
num_cycles_blocked_by_temp_disk	LONG	The count of cycles that the operator spent waiting for input and output from the temporary disk.
num_cycles_blocked_by_external_pending_blocks	LONG	The count of cycles that the operator spent waiting for input and output from external pending blocks.
blobstore_callback_time_ms	DOUBLE	The time, in milliseconds, the system spends processing callbacks for temporary disk input and output for the operator instance.
num_bounding_box_lists_sent	LONG	The number of geospatial bounding box list filters the operator sent to its peers.
num_bounding_box_lists_received	LONG	The number of geospatial bounding box list filters the operator received from its peers.
target_rows_per_cycle	LONG	The number of rows this operator expects to process in a single cycle without exceeding time limits.
applied_dynamic_filter_stats	CHAR	Additional statistics about dynamic pushdown filters in JSON format.
is_runnable	BOOLEAN	Whether the operator has available data to process.
current_num_pending_blocks	LONG	The current number of pending blocks across all partitions of this operator.

sys.active_query_scheduler_info

This table contains information about how the scheduler schedules each query to execute within the virtual machine (VM).

Column Name	Column Type	Column Description
query_id	UUID	The Universally Unique IDentifier (UUID) of the query that the scheduler currently schedules to execute.
node_id	UUID	The UUID of the node that executes this query.
silo_id	LONG	The identifier of the silo that executes this query on its associated node.
core_id	LONG	The identifier of the VM core that executes this query on its associated silo.
virtual_runtime	DOUBLE	A unitless fairness metric the scheduler uses to determine the query to schedule next. Comparisons of this value are only meaningful across concurrently executing queries with identical `node_id`, `silo_id`, `core_id`, and `current_priority` values.
current_priority	DOUBLE	The current scheduling priority of this query on its VM core.
total_cycles	LONG	The number of times this VM core has attempted to schedule this query.
successful_cycles	LONG	The number of times this VM core has attempted to schedule this query and completed some amount of work.
no_work_cycles	LONG	The number of times this VM core has attempted to schedule this query and no operators were prepared to do work.
failed_cycles	LONG	The number of times this VM core has attempted to schedule this query, some operators attempted to do work, and all operators failed to complete any amount of work.
failed_oom_cycles	LONG	The number of times this VM core has attempted to schedule this query, some operators attempted to do work, all operators failed to complete any

Column Name	Column Type	Column Description
		amount of work, and at least one operator on the query has reported that it is out of memory.
num_greedy_cycles	LONG	The number of times this VM core has attempted to schedule this query, which is the exclusive highest priority query currently managed by this VM core.
num_try_chilled_ops_cycles	LONG	The number of times this VM core attempted to schedule this query and polled operators that have been marked as `chilled`, meaning they are unlikely to be prepared to do work.
num_successful_try_chilled_ops_cycles	LONG	The number of times this VM core attempted to schedule this query, polled operators that have been marked ‘chilled’, and successfully completed some amount of work.
total_op_inst_cycle_time	DOUBLE	The approximate total CPU time, in milliseconds, that the operators in this query have been scheduled to execute on this VM core.
total_op_inst_dispatch_queue_time	DOUBLE	The approximate total CPU time, in milliseconds, that operators associated with this query have spent processing incoming message events.
total_runtime	DOUBLE	The approximate execution time, in milliseconds, that this VM core has managed this query.
total_bytes_spilled	LONG	The approximate total temporary disk amount that the VM core uses during query execution.

sys.all_operator_instances

This table contains information about operator instances that have been created by the virtual machine.

Column Name	Column Type	Column Description
database_name	VARCHAR	The name of the database associated with the query to which the operator belongs.

Column Name	Column Type	Column Description
user_name	VARCHAR	The username associated with the query to which the operator belongs.
query_id	UUID	A Universally Unique IDentifier (UUID) of the query to which the operator belongs.
operator_id	UUID	The UUID of the finalized operator.
node_id	UUID	The UUID of the node that stores information about this operator.
silo_id	BIGINT	The identifier of the silo that stores information about this operator.
core_id	BIGINT	The identifier of the virtual machine core that stores information about this operator.
op_runtime_id	BIGINT	The identifier for the operator instance that the system assigns at runtime.
op_instance_type	VARCHAR	The name of the type for this operator instance.
bloom_filtered_rows	BIGINT	The rows filtered by Bloom filters.
rows_received	BIGINT	The number of rows fetched by the operator instance.
rows_processed	BIGINT	The number of rows processed by this operator instance during execution.
rows_emitted	BIGINT	The number of rows returned by the operator instance.
blocks_emitted	BIGINT	The number of data blocks returned by the operator instance.
finalization_time	TIMESTAMP	The timestamp that represents when the operator completed all work.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
first_block_receive_time	TIMESTAMP	The timestamp that specifies the receipt of the first data block.
first_block_emitted_time	TIMESTAMP	The timestamp that specifies the return of the first data block.
num_cycles	BIGINT	The number of cycles that run on this operator instance.
num_oom_cycles	BIGINT	The number of run cycles that an operator received for processing out-of-memory issues.
num_failed_oom_cycles	BIGINT	The number of out-of-memory cycles received that failed to do any work.
num_no_work_oom_cycles	BIGINT	The number of out-of-memory cycles received with no work to do.
run_count	BIGINT	The number of times that the scheduler has reviewed this operator and `isRunnable` is set to `true`.
no_work_cycles	BIGINT	The number of times that the read and write cycle executed and performed no work.
backed_up_cycles	BIGINT	The number of times a cycle stopped because a backup of the parent exists.
leaf_branch_blocked_cycles	BIGINT	The number of times a cycle stopped on a leaf operator because of multi-child scheduling heuristics.
num_eof_cycles	BIGINT	The number of end-of-file cycles received.
normal_run_time_in_ms	DOUBLE	The time, in milliseconds, of the normal run for the operator instance.
dispatch_queue_time_in_ms	DOUBLE	The time, in milliseconds, spent processing incoming

Column Name	Column Type	Column Description
		message events for the operator instance.
oom_run_time_in_ms	DOUBLE	The time, in milliseconds, of the out-of-memory issue for the operator instance.
max_hp_mem_usage	BIGINT	The maximum heap memory usage by the operator instance.
num_blocks_written	BIGINT	The number of data blocks written to temporary disk.
num_blocks_read	BIGINT	The number of data blocks read from temporary disk.
num_fragments_written	BIGINT	The number of written memory fragments.
num_fragments_read	BIGINT	The number of read memory fragments.
cycles_over_1_sec	BIGINT	The count of cycles that ran over 1 second for this operator instance.
cycles_over_3_sec	BIGINT	The count of cycles that ran over 3 seconds for this operator instance.
long_dispatch_queue_events	BIGINT	The count of dispatch queue events processed by the operator with a runtime greater than 0.5 seconds.
total_bin_stream_allocated_bytes	BIGINT	The total amount of memory allocated in output blocks for variable-length column data.
total_bin_stream_used_bytes	BIGINT	The total amount of memory used in output blocks for variable-length column data.
total_col_stream_allocated_bytes	BIGINT	The total amount of memory allocated in output blocks for fixed-length column data.
total_col_stream_used_bytes	BIGINT	The total amount of memory used in output blocks for fixed-length column data.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
blocked_received	BIGINT	The total number of input data blocks received by this operator instance.
blocked_created	BIGINT	The total number of non-trivial output data blocks created by this operator instance.
received_block_bin_stream_unused_space_bytes	BIGINT	The total amount of wasted space across all binary streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive.
received_block_col_stream_unused_space_bytes	BIGINT	The total amount of wasted space across all column streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive.
mandatory_alloc_attempts	BIGINT	The count of cycles where the scheduler enabled mandatory allocation.
mandatory_alloc_success_count	BIGINT	The count of cycles where the scheduler enabled mandatory allocation and this operator broke an out-of-memory deadlock.
max_num_pending_blocks	BIGINT	The maximum number of pending blocks across all partitions during the lifetime of this operator.
max_num_consecutive_backup_cycles	BIGINT	The maximum number of consecutive scheduler cycles

Column Name	Column Type	Column Description
		where this operator has a backup. The scheduler can use this information to help detect certain resource deadlocks.
longest_time_since_unblocked_cycle	BIGINT	The maximum time between successful attempts to run this operator from the scheduler. The scheduler can use this information to help detect certain resource deadlocks.
compiled_level	BIGINT	The level within a hierarchy of distinct operators to execute. The system can compile a single plan operator into this hierarchy.
additional_json	VARCHAR	Additional statistics for an operator instance in JSON format.
num_bloom_filters_sent	BIGINT	The number of Bloom filters the operator sent to its peers.
num_bloom_filters_received	BIGINT	The number of Bloom filters the operator received from its peers.
num_key_lists_sent	BIGINT	The number of potential index join key lists the operator sent to its peers.
num_key_lists_received	BIGINT	The number of potential index join key lists the operator received from its peers.
max_hash_table_rows	BIGINT	The maximum number of rows present in any hash table used by this operator instance.
plan_operator_estimated_output_cardinality	BIGINT	The number of rows the optimizer estimated for this plan operator to emit.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
num_delayed_exceptions_generated	BIGINT	The number of rows this operator generated with delayed exception information.
num_pending_blocks_spilled	BIGINT	The number of queued pending data blocks this operator spilled to disk.
max_normal_cycle_time_ms	DOUBLE	The time, in milliseconds, of the longest normal run cycle for the operator instance.
max_oom_cycle_time_ms	DOUBLE	The time, in milliseconds, of the longest out-of-memory run cycle for the operator instance.
num_rows_received_per_plan_child	BIGINT[]	The number of rows this operator received from each of its children in the plan.
initialization_time	TIMESTAMP	The timestamp that represents when the operator began work.
num_cycles_blocked_by_temp_disk	BIGINT	The count of cycles that the operator spent waiting for input and output from the temporary disk.
num_cycles_blocked_by_external_pending_blocks	BIGINT	The count of cycles that the operator spent waiting for input and output from external pending blocks.
blobstore_callback_time_ms	DOUBLE	The time, in milliseconds, the system spends processing callbacks for temporary disk input and output for the operator instance.
num_bounding_box_lists_sent	BIGINT	The number of geospatial bounding box list filters the operator sent to its peers.
num_bounding_box_lists_received	BIGINT	The number of geospatial bounding box list filters the operator received from its peers.

Column Name	Column Type	Column Description
target_rows_per_cycle	BIGINT	The number of rows this operator expects to process in a single cycle without exceeding time limits.
applied_dynamic_filter_stats	VARCHAR	Additional statistics about dynamic pushdown filters in JSON format.
is_runnable	BOOLEAN	Whether the operator has available data to process.
current_num_pending_blocks	BIGINT	The current number of pending blocks across all partitions of this operator.
is_active	BOOLEAN	Whether the operator is currently active or has finalized.

sys.completed_operator_instances

This table contains information about operator instances that have finalized in the virtual machine.

Column Name	Column Type	Column Description
database_name	VARCHAR	The database associated with the query to which the operator belongs.
user_name	VARCHAR	The user associated with the query to which the operator belongs.
query_id	UUID	A unique identifier of the query to which the operator belongs.
operator_id	UUID	A unique identifier of the finalized operator.
node_id	UUID	The unique identifier of the node that stores the information about this operator.
silo_id	BIGINT	The unique identifier of the silo that stores the

Column Name	Column Type	Column Description
		information about this operator.
core_id	BIGINT	The unique identifier of the virtual machine core that stores the information about this operator.
op_runtime_id	BIGINT	The unique identifier for the operator instance that the system assigns at runtime.
op_instance_type	VARCHAR	The name of the type for this operator instance.
bloom_filtered_rows	BIGINT	The rows filtered by Bloom filters.
rows_received	BIGINT	The number of rows fetched by the operator instance.
rows_processed	BIGINT	The number of rows processed by this operator instance during execution.
rows_emitted	BIGINT	The number of rows returned by the operator instance.
blocks_emitted	BIGINT	The number of data blocks returned by the operator instance.
finalization_time	TIMESTAMP	The timestamp that represents when the operator completed all work.
first_block_receive_time	TIMESTAMP	The timestamp that specifies the receipt of the first data block.
first_block_emitted_time	TIMESTAMP	The timestamp that specifies the return of the first data block.
num_cycles	BIGINT	The number of cycles that run on this operator instance.
num_oom_cycles	BIGINT	The number of run cycles that an operator received for processing out-of-memory issues.

Column Name	Column Type	Column Description
num_failed_oom_cycles	BIGINT	The number of out-of-memory cycles received that failed to do any work.
num_no_work_oom_cycles	BIGINT	The number of out-of-memory cycles received that reported they had no work to do.
run_count	BIGINT	The number of times that the scheduler has reviewed this operator and `isRunnable` is set to `true`.
no_work_cycles	BIGINT	The number of times that the read and write cycle executed and performed no work.
backed_up_cycles	BIGINT	The number of times a cycle stopped because a backup of the parent exists.
leaf_branch_blocked_cycles	BIGINT	The number of times a cycle stopped on a leaf operator because of multi-child scheduling heuristics.
num_eof_cycles	BIGINT	The number of end-of-file cycles received.
normal_run_time_in_ms	DOUBLE	The time, in milliseconds, of the normal run for the operator instance.
dispatch_queue_time_in_ms	DOUBLE	The time spent, in milliseconds, processing incoming message events for the operator instance.
oom_run_time_in_ms	DOUBLE	The time, in milliseconds, of the out-of-memory issue for the operator instance.
max_hp_mem_usage	BIGINT	The maximum heap memory usage by the operator instance.
num_blocks_written	BIGINT	The number of data blocks written to temporary disk.

Column Name	Column Type	Column Description
num_blocks_read	BIGINT	The number of data blocks read from temporary disk.
num_fragments_written	BIGINT	The number of written memory fragments.
num_fragments_read	BIGINT	The number of read memory fragments.
cycles_over_1_sec	BIGINT	The count of cycles that ran over 1 second for this operator instance.
cycles_over_3_sec	BIGINT	The count of cycles that ran over 3 seconds for this operator instance.
long_dispatch_queue_events	BIGINT	The count of dispatch queue events processed by the operator with a runtime greater than 0.5 seconds.
total_bin_stream_allocated_bytes	BIGINT	The total amount of memory allocated in output blocks for variable-length column data.
total_bin_stream_used_bytes	BIGINT	The total amount of memory used in output blocks for variable-length column data.
total_col_stream_allocated_bytes	BIGINT	The total amount of memory allocated in output blocks for fixed-length column data.
total_col_stream_used_bytes	BIGINT	The total amount of memory used in output blocks for fixed-length column data.
blocked_received	BIGINT	The total number of input data blocks received by this operator instance.
blocked_created	BIGINT	The total number of non-trivial output data blocks created by this operator instance.
received_block_bin_stream_unused_space_bytes	BIGINT	The total amount of wasted space across all binary streams in received data blocks (includes over-allocations as well as projected columns and dead

Column Name	Column Type	Column Description
		memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive.
received_block_col_stream_unused_space_bytes	BIGINT	The total amount of wasted space across all column streams in received data blocks (includes over-allocations as well as projected columns and dead memory). This number is NULL unless configuration parameters allow recording this statistic for each block because the calculation is expensive.
mandatory_alloc_attempts	BIGINT	The count of cycles where the scheduler enabled mandatory allocation.
mandatory_alloc_success_count	BIGINT	The count of cycles where the scheduler enabled mandatory allocation and this operator broke an out-of-memory deadlock.
max_num_pending_blocks	BIGINT	The maximum number of pending blocks across all partitions during the lifetime of this operator.
max_num_consecutive_backup_cycles	BIGINT	The maximum number of consecutive scheduler cycles where this operator has a backup. The scheduler can use this information to help detect certain resource deadlocks.
longest_time_since_unblocked_cycle	BIGINT	The maximum time between successful attempts to run this operator from the scheduler. The scheduler can use this information to help detect certain resource deadlocks.

Column Name	Column Type	Column Description
compiled_level	BIGINT	The level within a hierarchy of distinct operators to execute. A single plan operator can be compiled into this hierarchy.
additional_json	VARCHAR	Additional statistics for an operator instance in JSON format.
num_bloom_filters_sent	BIGINT	The number of Bloom filters the operator sent to its peers.
num_bloom_filters_received	BIGINT	The number of Bloom filters the operator received from its peers.
num_key_lists_sent	BIGINT	The number of potential index join key lists the operator sent to its peers.
num_key_lists_received	BIGINT	The number of potential index join key lists the operator received from its peers.
max_hash_table_rows	BIGINT	The maximum number of rows present in any hash table used by this operator instance.
plan_operator_estimated_output_cardinality	BIGINT	The number of rows the optimizer estimated for this plan operator to emit.
num_delayed_exceptions_generated	BIGINT	The number of rows this operator generated with delayed exception information.
num_pending_blocks_spilled	BIGINT	The number of queued pending data blocks this operator spilled to disk.
max_normal_cycle_time_ms	DOUBLE	The time, in milliseconds, of the longest normal run cycle for the operator instance.
max_oom_cycle_time_ms	DOUBLE	The time, in milliseconds, of the longest out-of-memory run cycle for the operator instance.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
num_rows_received_per_plan_child	BIGINT[]	The number of rows this operator received from each of its children in the plan.
initialization_time	TIMESTAMP	The timestamp that represents when the operator began work.
num_cycles_blocked_by_temp_disk	BIGINT	The count of cycles that the operator spent waiting for input and output from the temporary disk.
num_cycles_blocked_by_external_pending_blocks	BIGINT	The count of cycles that the operator spent waiting for input and output from external pending blocks.
blobstore_callback_time_ms	DOUBLE	The time, in milliseconds, the system spends processing callbacks for temporary disk input and output for the operator instance.
num_bounding_box_lists_sent	BIGINT	The number of geospatial bounding box list filters the operator sent to its peers.
num_bounding_box_lists_received	BIGINT	The number of geospatial bounding box list filters the operator received from its peers.
target_rows_per_cycle	BIGINT	The number of rows this operator expects to process in a single cycle without exceeding time limits.
applied_dynamic_filter_stats	VARCHAR	Additional statistics about dynamic pushdown filters in JSON format.

sys.completed_queries

This table contains information about queries that finished execution in the database.

Column Name	Column Type	Column Description
query_id	UUID	A Universally Unique IDentifier (UUID) of the executed query.
user	VARCHAR	The name of the user that executed the query.
database_name	VARCHAR	The name of the database that runs this query.
database_id	UUID	The unique identifier of the database this query ran in.
sql	VARCHAR	The executed SQL statement.
sql_text_length	BIGINT	The length of the SQL statement.
referenced_tables	VARCHAR[]	The tables and views referenced by the query.
total_time	BIGINT	The total time in milliseconds the query took to run. This time includes generation_time, optimization_time, and execution_time.
generation_time	BIGINT	The time in milliseconds it took to generate the query before any processing happened.
optimization_time	BIGINT	The time in milliseconds the optimizer processed the query and built an optimized plan for the query.
execution_time	BIGINT	Time in milliseconds the query was executed by the database on the LTS nodes to return the query result.
timestamp_start	TIMESTAMP	A timestamp that represents the point in time when the query entered the system.

Column Name	Column Type	Column Description
parsing_time	BIGINT	The time in milliseconds the query was parsing.
cache_lookup_time	BIGINT	The time in milliseconds the query spent in lookup up matching cached result sets.
validation_time	BIGINT	The time in milliseconds the query was validating.
plangen_time	BIGINT	The time in milliseconds the query plan was being generated.
queue_time	BIGINT	The time in milliseconds the query was queued in the system before any processing happened.
timestamp_optimization_start	TIMESTAMP	A timestamp that represents the point in time when optimization of the query started.
timestamp_execution_start	TIMESTAMP	A timestamp that represents the point in time when execution of the query started.
tree_probe_time	BIGINT	The time in milliseconds taken by the VM tree probe during query execution.
vm_initialization_time	BIGINT	Time in milliseconds the query was initializing on all participating nodes during execution.
timestamp_first_byte_sent	TIMESTAMP	A timestamp that represents the point in time when the first byte of the result set from the query was returned to the application.
timestamp_complete	TIMESTAMP	A timestamp that represents the point in time when the execution of the query completed from the client’s perspective.

Column Name	Column Type	Column Description
timestamp_execution_complete	TIMESTAMP	A timestamp that represents the point in time when the internal execution of the query completed.
awaiting_client_eof_fetch_time	BIGINT	Time in milliseconds that represents the difference between when the client fetched the eof for the query and when the eof was queued internally.
rows_returned	BIGINT	The number of rows returned by the query.
bytes_returned	BIGINT	The number of bytes returned by the query.
rows_inserted	BIGINT	The number of rows that the query inserts. This value is NULL when the database does not insert any rows. The value is 0 when the database does not identify any rows to insert from a `CREATE TABLE AS SELECT` or `INSERT AS SELECT` SQL statement.
rows_deleted	BIGINT	The number of rows that the query deletes. This value is NULL when the database does not delete any rows. The value is 0 when the database does not identify any rows to delete from a `DELETE FROM TABLE` SQL statement.
update_subquery_execution_time	BIGINT	The time in milliseconds taken by the virtual machine (VM) to execute a subquery that identifies the rows for insertion or deletion. This value is NULL when you execute a SQL statement that is not one of these statements: `CREATE TABLE AS SELECT`, `INSERT AS SELECT`, or `DELETE FROM TABLE`.

Column Name	Column Type	Column Description
transferred_bytes_per_second	BIGINT	The transfer rate in bytes per second for the result set of the query.
code	INT	The SQL code returned at the end of the query.
state	VARCHAR	The SQL state returned at the end of the query.
reason	VARCHAR	A message associated with the SQL code and SQL state.
initial_priority	DOUBLE	Priority based on the service class, session, and query-level limits.
initial_effective_priority	DOUBLE	initial_priority adjusted for total cost and memory usage.
final_effective_priority	DOUBLE	Final dynamically adjusted priority.
cost_estimate	DOUBLE	The cost estimate of the optimizer.
concurrency_service_class_name	VARCHAR	Name of the service class this query executes in.
concurrency_service_class_id	UUID	The unique identifier of the service class used for the query.
priority_adjust_factor	DOUBLE	The percentage amount by which the database adjusts the priority.
priority_adjust_time	INT	How frequently the database adjusts the priority during query execution.
cached_query	BOOLEAN	Flag that indicates whether the result was returned from the result set cache.
resultset_cached	BOOLEAN	Flag that indicates whether the result of the query was stored in the result set cache.

Column Name	Column Type	Column Description
temp_disk_consumed	BIGINT	The approximate total temporary disk usage, in bytes, during query execution.
protocol_version	VARCHAR	The version of the client-server protocol for this connection.
client_ip	VARCHAR	The IP address of the client that executed the query.
client_name	VARCHAR	The name of the client that executed the query.
client_session_id	VARCHAR	The session id of the client that executed the query.
num_client_threads	INT	The maximum number of client threads determined by the server.
node_id	UUID	The unique identifier of the node that stores the information about this query.
participating_nodes	VARCHAR[]	The nodes that participated in the execution of this query.
ocient_db_version	VARCHAR	The Ocient® database version used to run this query.
ocient_db_commit	VARCHAR	The Ocient database git commit used to run this query.
ocient_db_dirtied_commit	VARCHAR	The dirtied Ocient database git commit used to run this query.
max_temp_disk_usage	INT	The maximum temporary disk usage for this query, expressed as a percentage.
max_elapsed_time	INT	The maximum elapsed time for this query, in seconds.
max_rows_returned	BIGINT	The maximum number of rows this query can return.

Column Name	Column Type	Column Description
num_root_operator_instances	INT	The number of created root operator instances.
approx_system_peak_vm_node_heap_mem_bytes	BIGINT	The max observed heap memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.
approx_system_peak_vm_cluster_heap_mem_bytes	BIGINT	The max observed heap memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query.
approx_system_peak_vm_node_huge_mem_bytes	BIGINT	The max observed huge page memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.
approx_system_peak_vm_cluster_huge_mem_bytes	BIGINT	The max observed huge page memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query.
approx_system_peak_sql_node_heap_mem_bytes	BIGINT	The max observed heap memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.
approx_system_peak_sql_node_huge_mem_bytes	BIGINT	The max observed huge page memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.
driver_version	VARCHAR	The version of the client driver.
join_order_optimization_time	BIGINT	The time, in milliseconds, the join order of the query

Column Name	Column Type	Column Description
		was optimized.
join_order_optimization_algorithm	VARCHAR	The algorithm used for optimizing the join order of the query.
tags	VARCHAR[]	The list of tags specified for the query.
default_schema	VARCHAR	The default schema associated with the client that executed the query.
transaction_id	UUID	The transaction scope UUID for statements executed inside an explicit transaction (with autocommit set to `OFF`). This value is NULL for autocommit statements.
transaction_prefix_len	INT	The number of high-order bits of transaction scope UUID that identify the transaction group. This value is NULL for autocommit statements.

sys.metric_levels

This table contains settings for metric levels.

Column Name	Column Type	Column Description
match	CHAR	Metric name match.
match_type	CHAR	The type of match for the name of the metric.
level	CHAR	Metric level.
node_id	UUID	The unique node identifier for this setting. This setting might be NULL for system-wide metrics.

sys.node_status

This table contains the status of nodes in the system and the core node monitoring metrics.

Column Name	Column Type	Column Description
node_id	UUID	The Universally Unique IDentifier (UUID) of the node.
operational_status	CHAR	The operational status of the node. Values are ACTIVE, STARTING, STOPPING, ERROR, UNKNOWN, or UNREACHABLE.
loadavg_1	DOUBLE	Average load on the node over the last 1 minute. Represents the average number of processes in the system run queue.
loadavg_5	DOUBLE	Average load on the node over the last 5 minutes. Represents the average number of processes in the system run queue.
loadavg_15	DOUBLE	Average load on the node over the last 15 minutes. Represents the average number of processes in the system run queue.
max_rss	LONG	Maximum resident set size (RSS) of the process on the node in bytes.
heap	LONG	Heap memory allocated on the node in bytes.
software_start	TIMESTAMP	Timestamp for the last start of the Ocient software on the node.
system_start	TIMESTAMP	Timestamp for the last start of the node hardware.

sys.op_inst_debug_info

This table contains information about the debug information for the operator instance.

Column Name	Column Type	Column Description
operator_id	UUID	The Universally Unique IDentifier (UUID) of the operator
node_id	UUID	The unique identifier of the node that stores the information about this operator.
silo_id	LONG	The unique identifier of the silo that stores the information about this operator.
vm_core_id	LONG	The unique identifier of the virtual machine core that stores the information about this operator.
runtime_id	INT	The process-wide unique identifier of this operator instance.
key	CHAR	The debug information (debugInfo) key.

Column Name	Column Type	Column Description
value	CHAR	The debugInfo value.
op_inst_type	CHAR	Type of the operator Instance.
query_id	UUID	The UUID of the query that owns the operator.

sys.osn_acquisitions

This table contains information about client identifiers for Ownership Storage Number (OSN) acquisitions.

Column Name	Column Type	Column Description
node_id	UUID	The Universally Unique IDentifier (UUID) of the node.
client_id	UUID	The query identifier of the OSN acquisition.
osn	LONG	The locked OSN.
acquisition_time	TIMESTAMP	Timestamp that represents when the OSN was acquired.

sys.plans

This table contains information about the plan for each SQL statement. Not all SQL statements have a corresponding plan (e.g., some DDL statements, SQL statements that have failed parsing or optimization).

Column Name	Column Type	Column Description
query_id	UUID	The unique identifier of the SQL query that executed.
user	VARCHAR	The name of the user that executed the SQL query.
database_name	VARCHAR	The name of the database that executes this SQL query.
timestamp_start	TIMESTAMP	A timestamp that represents the point in time when the query entered the system.
plan	VARCHAR	The plan for the SQL query.

sys.queries

This table contains information about queries that the database is currently executing.

Column Name	Column Type	Column Description
query_id	UUID	A Universally Unique IDentifier (UUID) of the query that is currently running.
user	CHAR	The name of the user that executes the query.
database_name	CHAR	The name of the database that runs this query.
database_id	UUID	A unique identifier of the database this query is running in.
status	CHAR	The statuses represent how the virtual machine (VM) tracks a specific query. The statuses are: AWAITING_SLOT (The query waits for a service class slot for optimization or execution.), OPTIMIZING (The Ocient System optimizes the query. If the system cannot optimize the query, the query skips this status and changes to the QUEUED status.), QUEUED (The system initializes the query in the VM.), RUNNING (The system executes the query in the VM, either by compiling the result from the Foundation Nodes or by fetching the results from the cache.), and FINALIZING (The query is complete.).
sql	CHAR	The SQL statement that is executing.
sql_text_length	LONG	The length of the SQL statement.
referenced_tables	ARRAY(CHAR)	The tables and views referenced by the query.
total_time	LONG	The total time in milliseconds the query has

Column Name	Column Type	Column Description
		run. This time includes generation_time, optimization_time, and execution_time.
generation_time	LONG	The time in milliseconds the query was generated or is still generating. This number might increase if generation is in progress.
optimization_time	LONG	The time in milliseconds the optimizer processed or is still processing the query. This number might be 0 if optimization of the query has not started yet. This number might increase if optimization is in progress.
execution_time	LONG	Time in milliseconds the query executes. This number might be 0 if execution of the query has not started yet. This number might increase if execution of the query is still in progress.
timestamp_start	TIMESTAMP	A timestamp that represents the point in time when the query entered the system.
parsing_time	LONG	Time in milliseconds the query was parsing or is still parsing. This number might be 0 if parsing of the query has not started yet. This number might increase if the query is still parsing.
cache_lookup_time	LONG	The time in milliseconds the query spent in lookup up matching cached result sets.
validation_time	LONG	Time in milliseconds the query was validating or is still being validated. This number might be 0 if validation of the query has

Column Name	Column Type	Column Description
		not started yet. This number might increase if the query is still validating.
plangen_time	LONG	Time in milliseconds the query plan was being generated or is still generating. This number might be 0 if generation of the query plan has not started yet. This number might increase if the query plan is still generating.
queue_time	LONG	Time in milliseconds the query was queued or is still queuing. This number might be 0 if queuing of the query has not started yet. This number might increase if the query is still in the queue.
timestamp_optimization_start	TIMESTAMP	A timestamp that represents the point in time when optimization of the query started. This value might be 0 if optimization has not started yet.
timestamp_execution_start	TIMESTAMP	A timestamp that represents the point in time when execution of the query started. This value might be 0 if execution has not started yet.
tree_probe_time	LONG	Time in milliseconds the query was in the VM tree probe stage during execution. This number might be 0 if VM tree probe has not started yet. This number might increase if the probe is still happening.
vm_initialization_time	LONG	Time in milliseconds the query was initializing on all participating nodes during execution. This number might be 0 if the VM has

Column Name	Column Type	Column Description
		not begin initializing the query for execution yet.
timestamp_first_byte_sent	TIMESTAMP	A timestamp that represents the point in time when the first byte of the result set from the query was returned to the application. This value might be 0 if execution of the query has not started yet or no bytes have been transferred yet.
rows_returned	LONG	The number of rows returned by the query. This value might be 0 if no rows have been transferred so far.
bytes_returned	LONG	The number of bytes returned by the query. This value might be 0 if no rows have been transferred so far.
rows_inserted	LONG	The number of rows that the query inserts. This value is NULL when the database does not insert any rows. The value is 0 when the database does not identify any rows to insert from a `CREATE TABLE AS SELECT` or `INSERT AS SELECT` SQL statement.
rows_deleted	LONG	The number of rows that the query deletes. This value is NULL when the database does not delete any rows. The value is 0 when the database does not identify any rows to delete from a `DELETE FROM TABLE` SQL statement.
update_subquery_execution_time	LONG	The time in milliseconds taken by the VM to execute a subquery that identifies the rows for insertion and deletion. This value is NULL

Column Name	Column Type	Column Description
		when you execute a SQL statement that is not one of these statements: `CREATE TABLE AS SELECT`, `INSERT AS SELECT`, or `DELETE FROM TABLE`.
initial_priority	DOUBLE	Priority based on the service class, session, and query level limits.
initial_effective_priority	DOUBLE	initial_priority adjusted for total cost and memory usage.
effective_priority	DOUBLE	Current dynamically adjusted priority.
estimated_time	INT	The estimate of the optimizer for the time in milliseconds to process the query.
estimated_result_rows	LONG	The estimate of the optimizer for the number of rows in the result set.
estimated_result_size	LONG	The estimate of the optimizer for the size of the result set in bytes.
concurrency_service_class_name	CHAR	Name of the service class this query executes in.
concurrency_service_class_id	UUID	The unique identifier of the service class used for the query.
priority_adjust_factor	DOUBLE	The percentage amount by which the database adjusts the priority.
priority_adjust_time	INT	How frequently the database adjusts the priority during query execution.
cached_query	BOOLEAN	Flag that indicates whether the result is returned from the result set cache.
temp_disk_consumed	LONG	The approximate total temporary disk usage, in

Column Name	Column Type	Column Description
		bytes, during query execution.
server_ip	IPV4	The IP address of the SQL node that executed the query.
protocol_version	CHAR	The version of the client-server protocol for this connection.
driver_version	CHAR	The version of the client driver.
client_ip	CHAR	The IP address of the client that executed the query.
client_name	CHAR	The name of the client that executed the query.
client_session_id	CHAR	The session id of the client that executed the query.
num_client_threads	INT	The maximum number of client threads determined by the server.
node_id	UUID	The unique identifier of the node that provided the information about this query.
participating_nodes	ARRAY(CHAR)	The nodes that are currently known to be participating in the execution of this query. If the query is not in the RUNNING state, this list might not include all nodes that participate.
max_temp_disk_usage	INT	The maximum temporary disk usage for this query, expressed as a percentage.
max_elapsed_time	INT	The maximum elapsed time for this query, in seconds.
max_rows_returned	LONG	The maximum number of rows this query can return.
num_root_operator_instances	INT	The number of created root operator instances.

Column Name	Column Type	Column Description
approx_system_peak_vm_node_heap_mem_bytes	LONG	The max observed heap memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.
approx_system_peak_vm_cluster_heap_mem_bytes	LONG	The max observed heap memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query.
approx_system_peak_vm_node_huge_mem_bytes	LONG	The max observed huge page memory in use during execution on a single non-SQL vm node. The memory in use is not necessarily associated with this query.
approx_system_peak_vm_cluster_huge_mem_bytes	LONG	The max observed huge page memory in use during execution totaled across all non-SQL vm nodes. The memory in use is not necessarily associated with this query.
approx_system_peak_sql_node_heap_mem_bytes	LONG	The max observed heap memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.
approx_system_peak_sql_node_huge_mem_bytes	LONG	The max observed huge page memory in use during execution on the sql node. The memory in use is not necessarily associated with this query.
join_order_optimization_time	LONG	The time, in milliseconds, the join order of the query was optimized.
join_order_optimization_algorithm	CHAR	The algorithm used for optimizing the join order of the query.

Column Name	Column Type	Column Description
tags	ARRAY(CHAR)	The list of tags specified for the query.
default_schema	CHAR	The default schema associated with the client that executed the query.
transaction_id	UUID	The UUID of the transaction where this statement executes. This value is NULL when the statement is not part of an explicit transaction. For `COMMIT` and `ROLLBACK` SQL statements, the value is the scope for the commit or rollback.
transaction_prefix_len	INT	The transaction scope prefix length. This value is NULL when the statement is not part of an explicit transaction.

sys.result_cache

This table contains queries that have cached results in the system.

Column Name	Column Type	Column Description
node_id	UUID	The Universally Unique IDentifier (UUID) of the node that holds the cache results.
query_id	UUID	The UUID of the query.
database_id	UUID	The database of the query.
service_class_id	UUID	The UUID of the service class of the query.
timestamp	LONG	The timestamp in seconds after the UNIX epoch when the result set entered the cache.
time	CHAR	The human-readable time when the result set entered the cache.
time_remaining	LONG	The amount of time remaining in seconds before the cached result set expires.
referenced_tables	ARRAY(CHAR)	All referenced tables in this query.

sys.retention_policies

The `retention_policies` table shows the tables that have a retention policy enabled in the system.

Column Name	Column Type	Column Description
table_id	UUID	The Universally Unique IDentifier (UUID) of the table.
retention_interval	VARCHAR	The retention interval for the retention policy.
execution_interval	VARCHAR	The execution interval for the retention policy.

sys.sample_segment_io_pipelines

This table contains information about a sample of segment I/O processing plans used for the database table I/O during query execution.

Column Name	Column Type	Column Description
completed_time	TIMESTAMP	The time when this segment pipeline was completed.

Column Name	Column Type	Column Description
database_name	VARCHAR	The database associated with the query to which the I/O operator belongs.
user_name	VARCHAR	The name of the user associated with the query to which the I/O operator belongs.
query_id	UUID	The Universally Unique IDentifier (UUID) of the query to which the I/O operator belongs.
operator_uuid	UUID	The UUID of the I/O operator in the query plan.
operator_id	BIGINT	An identifier of the I/O operator in the VM node that owns it.
node_id	UUID	The UUID of the node on which the I/O operator was executed.
silo_id	BIGINT	The identifier of the silo on which this I/O operator was executed.
segment_storage_id	UUID	The UUID for the segment associated with this I/O pipeline.
table_id	UUID	The UUID for the database table where this segment belongs.
segment_bounds_lower_inclusive	BIGINT	The lower bound (inclusive) on the row numbers processed in this pipeline.
segment_bounds_upper_exclusive	BIGINT	The upper bound (exclusive) on the row numbers processed in this pipeline.
pipeline	VARCHAR	A text representation of this segment I/O pipeline.

sys.scheduled_tasks

This table contains the details of scheduled tasks in the system.

Column Name	Column Type	Column Description
id	UUID	The scheduled task identifier.
admin_owner_id	UUID	The identifier of the node that is managing the scheduled task.
task_type	CHAR	The type of the task.
execution_interval_ms	LONG	The interval, in milliseconds, between task executions.

sys.schemas

This table contains the details of schemas in the system.

Column Name	Column Type	Column Description
id	UUID	The Universally Unique IDentifier (UUID) of the schema.
name	CHAR	The name of the schema.
database_id	UUID	The UUID of the database for the schema.
created_at	TIMESTAMP	The timestamp that represents when the schema was created.
updated_at	TIMESTAMP	The timestamp that represents when the schema was updated.
creator_id	UUID	The UUID of the creator for the schema.
is_implicit	BOOLEAN	Denotes the implicit creation of this schema after the creation of another object.

sys.service_role_status

This table contains the status of each service role on the system.

Column Name	Column Type	Column Description
node_id	UUID	The Universally Unique IDentifier (UUID) of the node.
service_role_id	UUID	The UUID of the service role (sys.service_roles).
service_role_name	CHAR	The human-readable name of the service role on the node.
status	CHAR	The current status of the role on this node.

sys.storage_device_metrics

This table contains monitoring metrics collected on each storage device that is controlled by Ocient.

Column Name	Column Type	Column Description
id	CHAR	Unique and persistent serial number of the drive. This serial number is not a Universally Unique IDentifier (UUID).
key	CHAR	Name of the measured statistic on the drive.
value	CHAR	Value of the named statistic on the drive.
transient	BOOLEAN	When the value is true, the value indicates that it is transient in nature and is reset when the node is restarted.
node_id	UUID	The UUID of the node that contains the storage device.

sys.storage_device_status

This table contains real-time information about the storage devices for each online node.

Column Name	Column Type	Column Description
node_id	UUID	The Universally Unique IDentifier (UUID) of the node that contains the storage device.
device_id	INT	Identifier of the drive.
role_name	CHAR	The name of the role or roles active on the node that contains the storage device.
id	CHAR	Unique and persistent serial number of the drive. This serial number is not a UUID.
pci_address	CHAR	PCI address where the drive is mounted.
device_model	CHAR	Model information about the drive.
manufacturer	CHAR	Manufacturer information about the drive.
firmware_version	CHAR	Firmware version that is active on the drive.
capacity	LONG	Capacity, in bytes, of the drive.
utilization	LONG	Utilization, in bytes, of the drive.
endurance_percentage_used	DOUBLE	Value from 0 to 1 that represents an estimate of the percentage of the drive

Column Name	Column Type	Column Description
		used, as it applies to drive endurance.
device_status	CHAR	Human-readable version of the drive status. Values are `ACTIVE`, `FAILED`, or `NOT PRESENT`.
assigned_drive_slots	ARRAY(INT)	List of drive slots assigned to this device. The list might be empty if no slots are assigned to a device.
encryption_drive_locking_supported	BOOLEAN	Indicates whether encryption drive locking (e.g., OPAL) is supported on the device.
encryption_drive_locking_enabled	BOOLEAN	Indicates whether encryption drive locking (e.g., OPAL) is enabled on the device.
encryption_drive_locking_status	CHAR	Indicates the status of drive locking (e.g., OPAL): `LOCKED` or `UNLOCKED`. If drive locking is not supported, this value is `UNLOCKED`.
silo_id	LONG	The identifier of the silo where this device is located.

sys.subtasks

This table contains the status and details of subtasks in the system.

Column Name	Column Type	Column Description
id	UUID	The task identifier.
name	CHAR	The name of the task.
task_type	CHAR	The type of the task.
execution_type	CHAR	The type of the task execution.
location_type	CHAR	The type of the location where the task must run.
location_id	CHAR	The location where the task must run.
task_options	CHAR	The task arguments.
start_time	TIMESTAMP	The start time of the task.
end_time	TIMESTAMP	The end time of the task.
last_poll_time	TIMESTAMP	The last time that the task was polled for a status.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
duration	LONG	The duration, in milliseconds, of the task.
status	CHAR	The current status of the task.
details	CHAR	The details or results of the status for the current task.
task_owner_id	UUID	The identifier of the node that is running the task.
admin_owner_id	UUID	The identifier of the node that is monitoring the task.
parent_task_id	UUID	The identifier of the parent task for the current task.
root_task_id	UUID	The identifier of the root task for the current task.
database_id	UUID	The database identifier.
parallelization	CHAR	The parallelization type of the task.
state	BINARY	The internal state of the task.
scope	UUID	The identifier of the scope for the current task.

sys.tasks

The `tasks` table shows the root tasks that have been started in the system.

Column Name	Column Type	Column Description
id	UUID	Task identifier
name	VARCHAR	Task name
task_type	VARCHAR	Task type
execution_type	VARCHAR	Task execution type
location_type	VARCHAR	Location type where the task must run
location_id	VARCHAR	Location where the task must run
task_options	VARCHAR	Task arguments
start_time	TIMESTAMP	Task start time
end_time	TIMESTAMP	Task end time
last_poll_time	TIMESTAMP	Last time the task was polled for status
duration	BIGINT	Task duration in milliseconds
status	VARCHAR	Current task status

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
details	VARCHAR	Current task status details or results
task_owner_id	UUID	Identifier of the node that is running the task
admin_owner_id	UUID	Identifier of the node that is monitoring the task
parallelization	VARCHAR	Parallelization type of the task
scope	UUID	Identifier of the scope of the task

Network

sys.channel_endpoint_parameters

This table contains details for the endpoints defined on the nodes in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the channel endpoint.
node_id	UUID	UUID of the node where the endpoint is located (sys.nodes).
name	CHAR	Name of the channel endpoint.
endpoint_type	CHAR	Type of the channel endpoint (FULL or EXTERNAL).
port	INT	Port that the channel endpoint is using.
ip_address	CHAR	IP address of the channel endpoint.
context_type	CHAR	Context type of the channel endpoint
listener_type	CHAR	Listener type of the channel endpoint
initiator_type	CHAR	Initiator type of the channel endpoint.

sys.network_interface_models

This table contains the network devices registered on the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the NIC.

Column Name	Column Type	Column Description
manufacturer	CHAR	Manufacturer of the NIC.
model_name	CHAR	Model name of the NIC.
pci_id	CHAR	PCI identifier.
pci_subsys_id	CHAR	PCI Subsystem identifier.
driver_type	CHAR	Driver type of NIC.
bits_per_second	LONG	Bits per second.

sys.network_interface_usage_types

This table contains the uses of each network device in the system.

Column Name	Column Type	Column Description
network_interface_id	UUID	Universally Unique IDentifier (UUID) of the network interface (sys.network_interfaces).
network_interface_model_id	UUID	UUID of the network interface model (sys.network_interface_models).
network_type	CHAR	Usage type of NIC (ADMIN, DATA, EXTERNAL, LOCAL_HSI).

sys.node_network_interfaces

This table contains the network interfaces on each node.

Column Name	Column Type	Column Description
network_interface_model_id	UUID	Universally Unique IDentifier (UUID) of the network interface model (sys.network_interface_models).
network_interface_id	UUID	UUID of the network interface.
node_id	UUID	UUID of the node where this interface exists (sys.nodes).
interface_type	CHAR	Type of interface (PHYSICAL, BOND_SLAVE, BOND_MASTER).
address	CHAR	MAC address of the NIC.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
name	CHAR	Name of the NIC.
netmask	CHAR	Netmask of the NIC.
gateway	CHAR	Gateway of the NIC.
pci_address	CHAR	PCI address of the NIC.
master_interface_name	CHAR	Master interface name of the NIC.
bond_type	CHAR	Network bond type of the NIC.

sys.service_role_channel_endpoints

This table contains the channel endpoint parameters that are enabled for each network type and service role.

Column Name	Column Type	Column Description
service_role_id	UUID	The Universally Unique IDentifier (UUID) of the service role of this endpoint (sys.service_roles).
channel_endpoint_parameters_id	UUID	UUID of the parameters for this endpoint (sys.channel_endpoint_parameters).
network_type	CHAR	Network type for this endpoint (ADMIN, DATA, EXTERNAL, LOCAL_HSI).

Pipelines

sys.pipeline_errors

While running a pipeline, you might experience errors. These errors are often due to a mismatch between the expectation of the pipeline and the source data. System or networking errors can also occur while the pipeline is running. The sys.pipeline_errors system catalog table lists extraction, transformation, and pipeline errors.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines) that loads this partition.

Column Name	Column Type	Column Description
extractor_task_id	UUID	UUID of the task associated with this pipeline run (sys.subtasks).
error_index	BIGINT	Error indexes start at 1 and increase monotonically as the Ocient System finds them.
error_type	VARCHAR	The type of error. Values are EXTRACTION, TRANSFORMATION, and PIPELINE_ERROR.
error_code	VARCHAR	This 5-character code identifies the type of error.
source_name	VARCHAR	For file loads, this value is the absolute path of the file that includes the S3 bucket. For Kafka loads, the value is the topic name and partition number.
error_message	VARCHAR	The error message.
partition_id	VARCHAR	The identifier of the partition. For file loads, the value is the stream source identifier where the file is located. For Apache® Kafka® loads, this value is the partition number.
record_number	BIGINT	The one-based index of the processed record in the source file or Kafka partition defined in the `source_name` column. This index does not account for headers or skipped lines.
record_offset	BIGINT	The one-based offset in bytes of the record data in the source file for file loads or the Kafka offset for Kafka loads. This offset is -1 for files with non-row-based storage formats (such as Apache® Parquet™).
field_index	INT	The zero-based index of the expression in the `SELECT` clause of the `CREATE PIPELINE` SQL statement.
column_name	VARCHAR	The name of the column in the target table.
created_at	TIMESTAMP	Timestamp that represents when this error was created.

sys.pipeline_errors_historical

This view shows pipeline errors for pipelines that have been dropped from the system. The view is the historical equivalent of sys.pipeline_errors.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline that loads this partition.
extractor_task_id	UUID	UUID of the task associated with this pipeline run (sys.subtasks).

Column Name	Column Type	Column Description
error_index	BIGINT	Error indexes start at 1 and increase monotonically as the Ocient System finds them.
error_type	VARCHAR	The type of error. Values are `EXTRACTION` (extraction), `TRANSFORMATION` (transformation), and `PIPELINE_ERROR` (pipeline error).
error_code	VARCHAR	This 5-character code identifies the type of error.
source_name	VARCHAR	For file loads, this value is the absolute path of the file that includes the Amazon® Web Services℠ (AWS℠) S3 bucket. For Apache® Kafka® loads, the value is the topic name and partition number.
error_message	VARCHAR	The error message.
partition_id	VARCHAR	The identifier of the partition. For file loads, the value is the stream source identifier where the file is located. For Kafka loads, this value is the partition number.
record_number	BIGINT	The index of the processed record in the source file or Kafka partition defined in the `source_name` column. This index starts at 1. The index does not account for headers or skipped lines.
record_offset	BIGINT	The offset, in bytes, of the record data in the source file for file loads or the Kafka offset for Kafka loads. The offset starts at 1. This offset is -1 for files with non-row-based storage formats (such as Apache® Parquet™).
field_index	INT	The index of the expression in the `SELECT` clause of the `CREATE PIPELINE` SQL statement. This index starts at 0.
column_name	VARCHAR	The name of the column in the target table.
created_at	TIMESTAMP	Timestamp that represents when this error was created.

sys.pipeline_events

While your pipeline is running, the pipeline generates events in the sys.pipeline_events system catalog table to mark significant checkpoints when something has occurred. For details about the lists of files, see the sys.pipeline_files system catalog table. Pipeline events include messages from many different tasks. These events are the background processes that execute across different Loader Nodes during pipeline operation. For details about tasks, you can query the sys.tasks and sys.subtasks system catalog tables.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines).
task_id	UUID	UUID of the task associated with this pipeline run (sys.subtasks).
user_id	UUID	The user who initiated the event (sys.users).
event_type	VARCHAR	The type of the event. You can generate events with these types: CREATED, REPLACED, RENAMED, STARTED, STOPPING, and STOPPED. The Ocient System generates events with these types automatically: FILE_LISTING_STARTED, FILE_LISTING_GENERATED, FILE_LISTING_COMPLETED, COMPLETED, FAILED, EXTRACTION_STARTED, EXTRACTION_COMPLETED, EXTRACTION_FAILED, EXTRACTION_RESOURCE_EXHAUSTED, SKIPPING_PARTIALLY_LOADED_FILE, and KAFKA_PARTITION_REBALANCE.
event_message	VARCHAR	Message that provides more details about the event.
event_timestamp	TIMESTAMP	Timestamp that represents when this event occurred.

sys.pipeline_events_historical

This view shows pipeline events for pipelines that have been dropped from the system. The view is the historical equivalent of sys.pipeline_events.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline.
task_id	UUID	UUID of the task associated with this pipeline run (sys.subtasks).
user_id	UUID	The user who initiated the event (sys.users).
event_type	VARCHAR	The type of the event. You can generate events with these types: CREATED, REPLACED, RENAMED, STARTED, STOPPING, and STOPPED. The Ocient System generates events with these types automatically: FILE_LISTING_STARTED, FILE_LISTING_GENERATED, FILE_LISTING_COMPLETED, COMPLETED, FAILED, EXTRACTION_STARTED, EXTRACTION_COMPLETED, EXTRACTION_FAILED, EXTRACTION_RESOURCE_EXHAUSTED,

Column Name	Column Type	Column Description
		SKIPPING_PARTIALLY_LOADED_FILE, and KAFKA_PARTITION_REBALANCE.
event_message	VARCHAR	Message that provides more details about the event.
event_timestamp	TIMESTAMP	Timestamp that represents when this event occurred.

sys.pipeline_files

For file-based loads, the sys.pipeline_files system catalog table contains one row for each file that shows the status of the file.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines).
extractor_task_id	UUID	The unique identifier of the task requested to load this file (sys.subtasks).
name	VARCHAR	Name of the file.
create_timestamp	TIMESTAMP	Timestamp that represents when this file is created.
modified_timestamp	TIMESTAMP	Timestamp that represents when this file is last modified.
size	BIGINT	Size of the file.
status	VARCHAR	The status of the file that updates throughout the process of a pipeline load. Values are: PENDING (listed file but not assigned to an extractor task), QUEUED (assigned to an extractor task but not confirmed the existence of the file), LOADING (confirmed existence of the file and began loading), LOADED (loaded data with complete success), LOADED_WITH_ERRORS (loaded data with at least one error), FAILED (failed to load), and SKIPPED (skipped loading the file).
file_index	BIGINT	Index of the file.

sys.pipeline_files_historical

This view shows pipeline files (terminal and pending) for pipelines that have been dropped from the system. The view is the historical equivalent of sys.pipeline_files.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline.
extractor_task_id	UUID	The unique identifier of the task requested to load this file (sys.subtasks).
name	VARCHAR	Name of the file.
create_timestamp	TIMESTAMP	Timestamp that represents when this file is created.
modified_timestamp	TIMESTAMP	Timestamp that represents when this file is last modified.
size	BIGINT	Size of the file.
status	VARCHAR	The status of the file that updates throughout the process of a pipeline load. Values are: PENDING (system listed the file but has not assigned it to an extractor task), QUEUED (system has assigned to an extractor task but has not confirmed the existence of the file), LOADING (system has confirmed the existence of the file and began loading), LOADED (system has loaded data with complete success), LOADED_WITH_ERRORS (system has loaded data with at least one error), FAILED (system has failed to load), and SKIPPED (system has skipped loading the file).
file_index	BIGINT	Index of the file.

sys.pipeline_files_pending

For file-based loads, the sys.pipeline_files system catalog table contains one row for each file that shows the status of the file.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines).
name	VARCHAR	The name of the file.
create_timestamp	TIMESTAMP	The timestamp that represents when this file is created.
modified_timestamp	TIMESTAMP	The timestamp that represents when this file is last modified.
size	BIGINT	The size of the file.
file_index	BIGINT	The index of the file.
file_transaction_id	BIGINT	The identifier of the file transaction that produced this file.

sys.pipeline_functions

This table contains all of the pipeline functions in the system.

Column Name	Column Type	Column Description
id	UUID	The Universally Unique IDentifier (UUID) of the pipeline function.
name	CHAR	The name of the pipeline function.
database_id	UUID	The UUID of the database for this pipeline function (sys.databases).
language	CHAR	The language used to write the pipeline function.
return_type	CHAR	The return type of the pipeline function.
return_type_nullable	BOOLEAN	The nullable property of the return type for the pipeline function.
definition	CHAR	The definition of the pipeline function.
creator_id	UUID	The UUID of the user or group who created the pipeline function (sys.users/sys.groups).
created_at	TIMESTAMP	The timestamp when this pipeline function was created.
altered_at	TIMESTAMP	The timestamp when this pipeline function was last updated.
argument_names	ARRAY(CHAR)	The argument names of the pipeline function, specified as an array.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
argument_types	ARRAY(CHAR)	The argument types of the pipeline function, specified as an array.
argument_nullability	ARRAY(BOOLEAN)	Whether the argument types for the pipeline function are nullable, specified as an array.
imported_libraries	ARRAY(CHAR)	The import statements for the pipeline function, specified as an array.

sys.pipeline_metrics

Find metrics that measure the performance and activity of a pipeline in the sys.pipeline_metrics view. This table contains one row for each metric at a specified point in time. Metrics are either instantaneous or incremental. Instantaneous metrics reflect the current value of a counter at the time of the metric collection. Incremental metrics increase over time and reflect the cumulative value for a specified counter. The first snapshot of a metric appears when it is initialized. Afterward, the system updates the metric every 10 seconds. Each metric has a scope that defines its uniqueness. Do not aggregate metrics across different scopes. The scope columns are pipeline_id, extractor_task_id, partition_id, and sink_index. If the value of any of these columns is NULL, the scope applies to all values in that dimension.

Column Name	Column Type	Column Description
pipeline_id	UUID	UUID of the pipeline (sys.pipelines) loading this partition.
extractor_task_id	UUID	UUID of the task associated with this pipeline run (sys.subtasks).
partition_id	VARCHAR	The partition identifier. For file-based loads, this identifier is the stream source identifier. For Kafka loads, this identifier is the partition number.
sink_index	BIGINT	The one-based index of the table in the `CREATE PIPELINE` SQL statement or NULL for metrics that are not table-specific.
name	VARCHAR	The name of the metric.
value	BIGINT	The value of the metric.
updated_at	TIMESTAMP	Timestamp that represents when this metric was updated.

© 2026 Ocient, Inc.

sys.pipeline_metrics_historical

This view shows pipeline metrics for pipelines that have been dropped from the system. The view is the historical equivalent of sys.pipeline_metrics.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline loading this partition.
extractor_task_id	UUID	UUID of the task associated with this pipeline run (sys.subtasks).
partition_id	VARCHAR	The partition identifier. For file-based loads, this identifier is the stream source identifier. For Apache® Kafka® loads, this identifier is the partition number.
sink_index	BIGINT	The index of the table in the `CREATE PIPELINE` SQL statement or NULL for metrics that are not table-specific. This index starts at 1.
name	VARCHAR	The name of the metric.
value	BIGINT	The value of the metric.
updated_at	TIMESTAMP	Timestamp that represents when this metric was updated.

sys.pipeline_metrics_info

Find the full list of available metrics in the sys.pipeline_metrics_info built-in view. This static table provides metric units, type, and descriptions.

Column Name	Column Type	Column Description
name	VARCHAR	Name of the metric.
units	VARCHAR	Units of the metric.
metric_type	VARCHAR	Type of the metric. One of INCREMENTAL or INSTANTANEOUS.
description	VARCHAR	Description of the metric.

sys.pipeline_partitions

For Apache® Kafka® partition-based loads, the sys.pipeline_partitions system catalog table contains one row for each partition, which contains the current offsets and record

counts.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines) loading this partition.
extractor_task_id	UUID	UUID of the task associated with this pipeline run (sys.subtasks).
group_id	VARCHAR	The group identifier (group.id) of the pipeline that loads this partition.
topic_name	VARCHAR	The name of the Kafka topic where this partition belongs.
partition_number	BIGINT	The partition number on the Kafka topic.
max_offset	BIGINT	The current maximum offset on this partition.
last_committed_durable_offset	BIGINT	The last committed offset on the partition that was indicated as durable by the Ocient System. This offset might lag slightly behind the actual durability of tables.
lag	INT	The number of records yet to be processed in a partition, calculated as the difference between the maximum offset and the last committed durable offset.
records_processed	INT	The number of records processed (but not necessarily made durable) for this partition.
records_failed	INT	The number of failed records for this partition.
updated_at	TIMESTAMP	The timestamp when the Ocient System last updated the details of this partition.

sys.pipeline_tables

This table contains all of the table identifiers currently associated with pipeline objects in the system.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline (sys.pipelines).
table_id	UUID	UUID of the table (sys.tables).

Column Name	Column Type	Column Description
table_index	INT	The one-based index of the table in the `CREATE PIPELINE` SQL statement.

sys.pipeline_tasks

The Ocient System implements the internals of a pipeline using distributed tasks. Each pipeline has a unique pipeline identifier pipeline_id. Every time that a START PIPELINE SQL statement is executed, the Ocient System generates a run_pipeline task that serves as the main coordinator task. This task automatically partitions the source data into smaller units and creates a run_extractor task for each one. Then, the Ocient System sends each run_extractor task and executes it on a Loader Node. In the sys.pipeline_tasks system catalog table, you can see the run_pipeline and run_extractor tasks.

Column Name	Column Type	Column Description
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline.
task_id	UUID	Task identifier
execution_type	VARCHAR	The type of the task.
parent_task_id	UUID	Identifier of the parent task of the task
location_type	VARCHAR	The type of location where the task must run.
location_id	VARCHAR	The identifier for the location where the task must run.
task_owner_id	UUID	Identifier of the node that is running the task
admin_owner_id	UUID	Identifier of the node that is monitoring the task
status	VARCHAR	Current task status
details	VARCHAR	The details or results of the status for the current task.
start_time	TIMESTAMP	The start time of the task.
end_time	TIMESTAMP	The end time of the task.
duration	BIGINT	Task duration in milliseconds

sys.pipelines

This table contains all of the pipelines in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the pipeline.
name	CHAR	Name of the pipeline.
database_id	UUID	UUID of the database of the pipeline (sys.databases).
loading_mode	CHAR	Specifies whether the pipeline runs in a one-time (BATCH), one-time transactional (TRANSACTIONAL), or continuous (CONTINUOUS) way.
source_type	CHAR	Source of the data (S3, KAFKA, FILESYSTEM).
data_format	CHAR	Structure of the data (CSV, DELIMITED, JSON, BINARY, PARQUET, XML, ASN1, AVRO).
created_at	TIMESTAMP	Timestamp that indicates when this pipeline was created.
altered_at	TIMESTAMP	Timestamp that indicates when this pipeline was last updated.
creator_id	UUID	The UUID of the user or group who created the pipeline (sys.users/sys.groups).
status	CHAR	Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).
status_message	CHAR	Status message of the pipeline.
task_id	UUID	Universally Unique IDentifier (UUID) of the task (sys.tasks).
rolehostd_version	CHAR	The rolehostd version in which this table was created
pending_files_count	LONG	The number of files with the `PENDING` status (sys.pipeline_files).
queued_files_count	LONG	The number of files with the `QUEUED` status (sys.pipeline_files).
loading_files_count	LONG	The number of files with the `LOADING` status (sys.pipeline_files).
loaded_files_count	LONG	The number of files with the `LOADED` status (sys.pipeline_files).
loaded_with_errors_files_count	LONG	The number of files with the `LOADED_WITH_ERRORS` status (sys.pipeline_files).

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
failed_files_count	LONG	The number of files with the `FAILED` status (sys.pipeline_files).
skipped_files_count	LONG	The number of files with the `SKIPPED` status (sys.pipeline_files).
transaction_id	UUID	The UUID of the transaction scope if this pipeline runs in a one-time transactional way.

Security

sys.security_settings

This table contains security settings.

Column Name	Column Type	Column Description
database_id	UUID	The database for this setting. This might be NULL for system-wide settings
group_id	UUID	The group for this setting when the setting is for a group
password_minimum_length	INT	Minimum password length.
password_complexity_level	INT	Password complexity requirement.
password_lifetime_days	INT	Password lifetime in days.
password_no_repeat_count	INT	Password repetition count.
password_invalid_attempt_limit	INT	Invalid password attempt limit.

Security Integrations

sys.oidc_integrations

This table enumerates the configuration for OpenID Connect Single Sign-On integrations (i.e. OKTA®). You can modify this configuration using the ALTER DATABASE <database> ALTER SSO INTEGRATION oidc DDL statement where <database> is the name of your database.

Column Name	Column Type	Column Description
security_integration_id	UUID	The id of this OpenID Connect (OIDC) Single Sign-On integration.
database_id	UUID	The id of the database using this Single Sign-On authorization to this integration.
disabled	BOOLEAN	When true, all new connections made using this Single Sign-On integration will fail.
default_group	CHAR	The group which all users of this Single Sign-On intergation are granted membership to.
issuer	CHAR	The complete URL for the OAuth 2.0 / OpenID Connect Authorization Server. This is the expected “iss” claim in an access token.
client_id	CHAR	The provider-supplied ID of this Single Sign-On integration.
public_client	BOOLEAN	When true, PKCE is used in place of a client authentication method
enable_id_token_authentication	BOOLEAN	When true, a fat ID Token is sufficient to connect to the database.
user_id_claims	ARRAY(CHAR)	The id token claim(s) used to identify users in audit trails.
additional_scopes	ARRAY(CHAR)	Request scopes for authorization requests.
additional_audiences	ARRAY(CHAR)	Additional token audience to accept when validating tokens. This is useful for Authorization Servers without a token exchange capability.
groups_claim_ids	ARRAY(CHAR)	The token claims that can be used to map the user to a Database group. You must provide `groups_claim_ids` if `groups_claim_mappings` is provided.
groups_claim_mappings	CHAR	The mappings from Provider group => Database group.
roles_claim_ids	ARRAY(CHAR)	The token claims that can be used to map the user to a Database role. You must provide `roles_claim_ids` if `roles_claim_mappings` is provided.
roles_claim_mappings	CHAR	The mappings from Provider role => Database role.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
blocked_groups	ARRAY(CHAR)	Provider groups that are restricted from connecting to the database.
blocked_roles	ARRAY(CHAR)	Provider roles that are restricted from connecting to the database.
allowed_groups	ARRAY(CHAR)	Provider groups that are allowed to connect to the database.
allowed_roles	ARRAY(CHAR)	Provider roles that are allowed to connect to the database.
enable_debug_flow	BOOLEAN	
redirect_host	CHAR	Specifies the redirect host used in authorization requests.
redirect_ssl	BOOLEAN	Whether to use SSL for redirect URIs.
request_scopes	ARRAY(CHAR)	All request scopes for authorization requests.
scopes_on_refresh	BOOLEAN	Whether to include the scope list in the initial refresh-token request. The default value is `false`.
allow_offline_access	BOOLEAN	Whether to include `access_type=offline` and `prompt=consent` in the authorization URL. The default value is `false`.

sys.oidc_sessions

The sys.oidc_sessions table contains information about active database connections established using an OpenID Connect (oidc) Single Sign-On integration.

Column Name	Column Type	Column Description
id	UUID	The unique identifier for this session.
user	CHAR	The Fully Qualified User Name (FQUN) of this user.
database_id	UUID	The id of the database associated with this session.
security_integration_id	UUID	The id of the OpenID Connect (OIDC) Single Sign-On integration that authorized.
has_access_token	BOOLEAN	True if the identity provider supplied an Access Token.
has_refresh_token	BOOLEAN	True if the identity provider supplied a Refresh Token.

Column Name	Column Type	Column Description
compact_id_token	CHAR	The JWS compact serialization form of the ID Token.
id_token_json	CHAR	The ID Token claims.

sys.security_integrations

This table contains the OIDC security integrations installed on the system.

Column Name	Column Type	Column Description
id	UUID	UUID of the OIDC security integration.
database_id	UUID	UUID of the database associated with this security integration (sys.databases).
integration_type	CHAR	The type of the security integration (SSO, SCIM, LDAP, etc.).
table_descriptor	CHAR	An indirect pointer to the table where you can view the objects of this integration.
sso_integration_name	CHAR	SSO integration name, if the security integration is of type SSO
is_database_default	BOOLEAN	When true, the security integration is used as the default SSO integration for the database.

sys.sessions

This table contains information about active client connections.

Column Name	Column Type	Column Description
id	UUID	The unique identifier for this session.
user	CHAR	The Fully Qualified User Name (FQUN) of this user.
sso_protocol	CHAR	The Single Sign-On (SSO) protocol used to authorize the connection request. Null if basic password authentication was used. Tables exists for each SSO protocol using the following naming scheme: “<protocol>_sessions”. For example, all sessions created using the OpenID Connect (oidc) protocol will be listed in the “sys.oidc_sessions” table.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
database_id	UUID	The id of the database associated with this session.
default_schema	CHAR	The default schema used for queries submitted by this user.
created_at_timestamp	LONG	The number of seconds from epoch the session was established.
duration	LONG	The number of seconds since the session was established.
expires_at_timestamp	LONG	The number of seconds from epoch the session expires.
refresh_expires_at_timestamp	LONG	The number of seconds from epoch the refresh token expires.
is_renewable	BOOLEAN	When the value is `true`, the database attempts to extend the session when it expires.
client_ip	CHAR	The IP address of the client.
client_name	CHAR	The human-readable name of the client.
client_session_id	CHAR	An opaque identifier provided by client implementations. Used to correlate events across application boundaries.
protocol_version	CHAR	The version of the client-server protocol for this connection.
driver_version	CHAR	The version of the client driver.
tls_enabled	BOOLEAN	True if TLS is enabled for this client connection.
node_id	UUID	The unique identifier of the node the client is connected to.
connectivity_pool_id	UUID	The unique identifier of the connectivity pool the client is connected on.
group_names	CHAR	Names of all groups in the session.
role_names	CHAR	Names of all roles in the session.
active_service_class_names	CHAR	Names of all active service classes in the session.
inactive_service_class_names	CHAR	Names of all inactive service classes in the session.

© 2026 Ocient, Inc.

Statistics

sys.average_bb_sizes

This table contains the average size of bounding boxes in the database for geospatial data types.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
column_id	UUID	UUID of the column (sys.columns).
avg_width	DOUBLE	Average width of bounding boxes in this column.
avg_height	DOUBLE	Average height of bounding boxes in this column.

sys.average_column_sizes

This table contains the average size of each column in the database.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
column_id	UUID	UUID of the column (sys.columns).
size	DOUBLE	Average size of the column.

sys.column_cardinalities

This table contains estimates of the number of unique values in each column in the database. These estimates might not reflect recent loading or deletion activity.

Column Name	Column Type	Column Description
table_id	UUID	The Universally Unique IDentifier (UUID) of the table (sys.tables).
column_id	UUID	UUID of the column (sys.columns).
cardinality	LONG	The estimate of the number of unique values in this column.

Column Name	Column Type	Column Description
stats_type	CHAR	Represents the type of values, either ‘COLUMN’ or ‘INNER_ARRAY’.
num_included_rows	LONG	The estimate of the total number of non-NULL rows (where this column value is not NULL) in the table that are included in this distinct estimate.
num_excluded_rows	LONG	The estimate of the total number of rows in the table that are not included in this distinct estimate.

sys.column_distributions

This table contains some statistical characteristics of the calculated Kernel Density Distribution of data in the database columns.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
column_id	UUID	UUID of the column (sys.columns).
bandwidth	DOUBLE	Smoothing parameter for the distribution.
minimum_value	DOUBLE	Minimum value found in the distribution.
maximum_value	DOUBLE	Maximum value found in the distribution.
stats_type	CHAR	Represents the type of values in the distribution, either ‘TABLE’ or ‘INNER_ARRAY’.
num_samples	LONG	The number of sampled row values included in the Kernel Density Distribution.

sys.columns_compression_info

This table contains column-level compression statistics for fixed-length columns. The system calculates statistics from the column data across all segments.

Column Name	Column Type	Column Description
node_id	UUID	Universally Unique IDentifier (UUID) of the node that owns the segments included in these statistics.

Column Name	Column Type	Column Description
table_id	UUID	UUID of the table of this column (sys.tables).
ordinal	INT	Ordinal of the column.
is_deltadelta_enabled	BOOLEAN	Specifies whether deltadelta compression is enabled.
is_rle_enabled	BOOLEAN	Specifies whether RLE compression scheme is enabled.
is_ne_enabled	BOOLEAN	Specifies whether the NULL elimination compression scheme is enabled.
raw_size	LONG	Size of the data in bytes before compression.
compressed_size	LONG	Size of the data in bytes after compression.
num_deltadelta_blocks	LONG	Number of data blocks compressed using the deltadelta compression.
num_rle_blocks	LONG	Number of data blocks compressed using the RLE (Run-Length Encoding) compression scheme.
num_nerle_blocks	LONG	Number of data blocks compressed using a combination of the RLE and NE (NULL Elimination) compression schemes.
num_uncompressed_blocks	LONG	Number of uncompressed data blocks.
num_total_blocks	LONG	Total number of data blocks.
id	UUID	UUID of the column
name	CHAR	Name of the column

sys.segments_compression_info

This table contains segment-level compression statistics. The system calculates statistics from the data within the specified segment. All block statistics apply only to fixed-length columns within the segment. Non-block specific statistics (e.g., size information) include both fixed and variable-length columns.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table relevant to this segment (sys.tables).
segment_id	UUID	UUID for the segment.

Column Name	Column Type	Column Description
num_deltadelta_blocks	LONG	Number of data blocks compressed using deltadelta compression
num_rle_blocks	LONG	Number of data blocks compressed using RLE (Run-Length Encoding) compression scheme
num_nerle_blocks	LONG	Number of data blocks compressed using a combination of RLE and NE (Null Elimination) compression schemes
num_uncompressed_blocks	LONG	Number of uncompressed data blocks
num_total_blocks	LONG	Total number of data blocks
raw_size	LONG	Size of the data part before any compression and without any parity data
compressed_size	LONG	Size of the data part after compression and without any parity data + Size of the manifest part
compressed_data_part_size	LONG	Size of the data part after compression and without any parity data
num_cluster_keys	LONG	Number of unique cluster keys in the segment
num_time_buckets	LONG	Number of time buckets in the segment

sys.stats_files

This table contains information about the on-disk cache for each Foundation Node of the local table probability density functions (PDFs) and array PDFs.

Column Name	Column Type	Column Description
node_id	UUID	The Universally Unique IDentifier (UUID) of the node. References id in sys.nodes.
table_id	UUID	The UUID of the table. References id in sys.tables.
file_id	UUID	The UUID of the file.
file_type	CHAR	The type of this file. Values are ARRAY_PDF, TABLE_PDF, or TABLE_CDE.
file_size	LONG	Size in bytes of the file.
column_ordinal	LONG	Ordinal of the column where the stats file applies. The ordinal references sys.columns for the specified table. The value is NULL when file_type is TABLE_PDF.

Column Name	Column Type	Column Description
updated_at	TIMESTAMP	Timestamp of the last update to this file.
is_stale	BOOLEAN	Whether any rows are present on disk that were not present when this stats file was last updated
rows_computed_from	LONG	Number of rows present on the node when this stats file was last updated
marked_stale_at	TIMESTAMP	Timestamp of this file being marked stale.

sys.table_cardinalities

This table contains estimates of the number of rows in the individual tables in the database. These estimates might not reflect recent loading or deletion activity.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
cardinality	LONG	Estimate of the number of rows in this table.

sys.vl_columns_compression_info

This table contains column-level compression statistics for variable-length columns. The system calculates statistics from the column data across all segments.

Column Name	Column Type	Column Description
node_id	UUID	Universally Unique IDentifier (UUID) of the node that owns the segments included in these statistics.
table_id	UUID	UUID of the table (sys.tables).
ordinal	INT	Ordinal of the column.
is_compression_enabled	BOOLEAN	Whether compression is enabled.
raw_size	LONG	Size of the data in bytes before compression.
compressed_size	LONG	Size of the data in bytes after compression.
num_lz4_rows	LONG	Number of lz4 compressed rows.
num_uncompressed_rows	LONG	Number of uncompressed rows.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
num_null_rows	LONG	Number of NULL rows.
num_total_rows	LONG	Total number of rows.
id	UUID	UUID of the column
name	CHAR	Name of the column

Storage

sys.addendum_directories

This table contains all addendum directories with details about their associated directories and metadata.

Column Name	Column Type	Column Description
id	UUID	The identifier of the addendum directory within the storage cluster state.
associated_directory_group_id	CHAR	The group identifier of the associated directory.
associated_directory_ida_offset	INT	The IDA offset (the logical segment in a segment directory group) of the associated directory.
parent_segment_group_id	CHAR	The segment group identifier of the associated parent segment (sys.segment_groups).
parent_ida_offset	INT	The IDA offset of the associated parent segment.
segment_part_inventory_name	UUID	The name of the segment part inventory for the addendum part. This name references the id column of the segment_part_inventory system catalog table.
segment_part_inventory_storage_id	CHAR	The storage identifier of the segment part inventory.
num_deleted_rows	LONG	If the segment part inventory houses a delete part, this number is the number of rows subtracted from the parent segment.

Column Name	Column Type	Column Description
is_replica_for_ida_offset	INT	The IDA offset of the parent segment that this segment part inventory replicates.

sys.current_osn

This table contains the current Ownership Sequence Number (OSN) for each cluster on the system.

Column Name	Column Type	Column Description
start_osn	CHAR	The start OSN for the cluster.
end_osn	CHAR	The end OSN for the cluster.
cluster_id	UUID	The Universally Unique IDentifier (UUID) of the cluster.

sys.data_usage

This table contains information about the amount of queryable data for each table and file type.

Column Name	Column Type	Column Description
segment_type	VARCHAR	The type of the segment.
database_name	VARCHAR	The name of the database for the table.
schema	VARCHAR	The schema of the table.
name	VARCHAR	The name of the table.
num_segments	BIGINT	The number of segments of a specific segment type.
total_rows	BIGINT	The total number of rows for segments of a specific segment type.
avg_rows_per_segment	DOUBLE	The average amount of rows per segment for a segment of a specific segment type.
total_size_gb	DOUBLE	The total size in gigabytes for all segments of a specific segment type.
avg_size_mb	DOUBLE	The average size in megabytes for segments of a specific segment type.

© 2026 Ocient, Inc.

sys.degraded_segment_groups

This table contains information about degraded segment groups in addition to the clusters and tables where they are located.

Column Name	Column Type	Column Description
cluster_id	UUID	The unique identifier of the cluster.
cluster_name	VARCHAR	The name of the cluster.
segment_group_status	VARCHAR	The segment group status.
database_name	VARCHAR	The name of the database for the table.
schema	VARCHAR	The schema of the table.
name	VARCHAR	The name of the table.
degraded_segment_groups	BIGINT	The total number of damaged segment groups in a cluster.

sys.merge_eligibilities

This table computes and displays a count of segment groups and segment directory groups eligible for merging for each table in the Ocient System.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
database_name	VARCHAR	Name of the database for the table.
schema	VARCHAR	Name of the schema.
name	VARCHAR	Name of the table.
segment_groups_merge_eligible	BIGINT	The count of segment groups eligible for merging in the table.

sys.merge_policies

This table computes and displays the effective merge policy (enabled or disabled) for each table in the system. The system bases the policy on the default configuration and any overrides that are present for each table.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
database_name	VARCHAR	Name of the database for the table.
schema	VARCHAR	Name of the schema.
name	VARCHAR	Name of the table.
table_lts_property_string	VARCHAR	Additional properties assigned to control the Foundation role (formerly “LTS role”) behavior for the table.
effective_table_merge_policy	VARCHAR	The effective merge policy applied to this table based on the value of the global `feature_directory_merge` feature flag and an optional table merge policy override.

sys.orphaned_segments

This table contains information about orphaned segments in the cluster.

Column Name	Column Type	Column Description
storage_id	UUID	The Universally Unique IDentifier (UUID) for the orphaned segment.
owner	LONG	The identifier of the owner for this orphaned segment.
segment_type	CHAR	The type of the segment (TKT, PAGE, etc.).
node_id	UUID	The UUID for this setting, which might be NULL for system-wide metrics.

sys.segment_directories

This table contains all segment directories with details about parents, associated buckets, and metadata.

Column Name	Column Type	Column Description
segment_group_id	CHAR	The identifier of the segment group (sys.segment_groups).
ida_offset	INT	The unique index of the segment.

Column Name	Column Type	Column Description
storage_id	UUID	The storage identifier of the segment directory.
owner	LONG	The owner identifier of this stored segment.
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
begin_time	LONG	Start time of the bucket of this segment directory in the time bucket column of its table.
end_time	LONG	End time of the bucket of this segment directory in the time bucket column of its table.
depth	INT	The depth of the segment group. A non-zero value indicates a segment directory group.
parent_segment_group_id	CHAR	The parent identifier of the segment directory group (sys.segment_groups).
parent_ida_offset	INT	The unique index of the parent segment, if it exists.
parent_storage_id	UUID	The storage identifier of the parent segment directory, if it exists.
parent_owner	LONG	The owner of the parent storage identifier. A zero value indicates an unowned storage identifier. Whereas a non-zero value indicates a segment owned after rebuild.
row_count	LONG	If the directory depth equals 1, then this count is the number of rows in the base leaf segment (not factoring in any DELETE SQL statements). If the depth is greater than 1, then this count is the total number of rows spanned by the intermediate directory.
created_time	TIMESTAMP	Specifies the time, in nanoseconds, when this segment group was created.
rolehostd_version	CHAR	The version of the rolehostd binary at the time of the segment group generation.

sys.segment_group_transfers

This table contains all segment group transfers between clusters.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the segment group transfer.
src_cluster_id	UUID	UUID of the source cluster (sys.clusters).
dst_cluster_id	UUID	UUID of the destination cluster (sys.clusters).
status	CHAR	The status of the transfer.
segment_group_ids	ARRAY(CHAR)	List of the identifiers of segment groups in this transfer (sys.segment_groups).
src_committed_osn	CHAR	The Ownership Storage Number (OSN) in which the segment groups are considered fully transferred from the source cluster.
dst_committed_osn	CHAR	The OSN in which the segment groups are considered fully transferred from the destination cluster.

sys.segment_groups

This table contains all segment groups stored on the cluster with details about the ownership, associated table, data structure, and other metadata.

Column Name	Column Type	Column Description
id	CHAR	The identifier of the segment group.
cluster_id	UUID	Universally Unique IDentifier (UUID) of the internal cluster of this segment group.
segment_type	CHAR	Type of the segment (TKT, PAGE, etc.).
status	CHAR	The availability and health status of the segment group.
primary_owner	UUID	For a replicated segment group, the UUID of the node that serves the segment (sys.nodes).
loader_id	UUID	The streamloader node that wrote the segment group (sys.nodes).
table_id	UUID	UUID of the table (sys.tables).
scope_id	UUID	UUID of the storage scope (sys.storage_scopes).
block_size	LONG	Disk block size used to store segments in this group in bytes.

Column Name	Column Type	Column Description
begin_time	LONG	Inclusive minimum bucket value, specified as an integer, for all rows in this segment group. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and you can convert or compare it to other dates or times with the appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds after the epoch.
end_time	LONG	Inclusive maximum bucket value, specified as an integer, for all rows in this segment group. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and you can convert or compare it to other dates or times with the appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds after the epoch.
begin_timestamp	TIMESTAMP	Inclusive minimum bucket value, specified as a TIMESTAMP, for all rows in this segment group. If the bucket column is a unitless integer, this column is NULL.
end_timestamp	TIMESTAMP	Inclusive maximum bucket value, specified as an integer, for all rows in this segment group. If the bucket column is a unitless integer, this column is NULL.
coding_algorithm	CHAR	Coding algorithm used to erasure code this group (NO_CODING, PQ_PARITY, REED_SOLOMON).
coding_block_size	INT	The unit size that is erasure coded in bytes.
coding_threshold	INT	The number of coding blocks required to rebuild all blocks in a coding line.
coding_width	INT	The number of coding blocks in a coding line.
replication	INT	The number of replicas of each segment in the group.
parity_cycle	INT	Parity cycle that the system calculates by multiplying by the coding_width to return the number of segments in the segment group.
created_time	TIMESTAMP	Specifies the time, in nanoseconds, when this segment group was created.
rolehostd_version	CHAR	The version of the rolehostd binary at the time of the segment group generation.
commit_hash	CHAR	The commit hash of the rolehostd binary at the time of the segment group generation.
timestamp	CHAR	The build timestamp or commit timestamp of the rolehostd binary at the time of the segment group

Column Name	Column Type	Column Description
		generation.
build_user	CHAR	The build user of the rolehostd binary at the time of the segment group generation.
depth	INT	The depth of the segment directory group. A non-zero value indicates a segment directory group.
removal_type	CHAR	The method by which a segment group was removed.
visibility	CHAR	The visibility of the segment group.

sys.segment_part_inventory

This table contains information about the inventory of markers for segments. For example, a marker denotes whether the database can delete a specific segment part. One segment can have multiple inventories, whereas each inventory contains only one marker.

Column Name	Column Type	Column Description
id	UUID	The universally unique identifier (UUID) of the inventory that stores the markers for segments.
ida_offset	INT	The IDA offset (the logical part of the segment) denotes the position within the segment.
parent_segment_group_id	CHAR	The identifier of the segment group where the parent segment is located.
parent_ida_offset	INT	The IDA offset where the parent segment is located within the segment group.
num_deleted_rows	LONG	The number of deleted rows.
operation_id	UUID	The UUID of the query where the Ocient System creates the inventory (the `DELETE` SQL statement). The system shares this UUID across all inventories created in the same operation.
subsuming_type	CHAR	The type of the marker within the inventory.
node_id	UUID	The UUID of the node that contains the latest segment with the part.

sys.segment_part_redundancy_info

This table contains the redundancy strategy for each table and the segment part type.

Column Name	Column Type	Column Description
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
part_name	CHAR	Name of the segment part.
redundancy_type	CHAR	The redundancy strategy for this part (COPY, PARITY).

sys.segment_parts

This table contains the segment parts associated with each segment group. These segment parts belong to queryable segments. The table does not display parts that belong to quarantined segments or segments not yet activated.

Column Name	Column Type	Column Description
segment_group_id	CHAR	The identifier of the segment group (sys.segment_groups).
ida_offset	INT	Unique index of the segment. The original copy of this segment part belongs to this segment.
name	CHAR	Name of the segment part.
part_type	CHAR	Partial identifier of a segment part (DATA, INDEX, MANIFEST, STATS).
size	LONG	Size of the segment part in bytes.
segment_ida_offset	INT	Unique index of a segment within a segment group.
segment_lba_offset	LONG	Offset of this segment part within the Logical Block Address (LBA) on disk.

sys.segments

This table contains the individual segments stored in each storage cluster including basic information about the segment and the segment group where the segment belongs. The table contains queryable segments only. Quarantined segments or segments not yet activated do not appear in this table (they are present in the sys.stored_segments table).

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
segment_group_id	CHAR	The identifier of the segment group (sys.segment_groups).
segment_type	CHAR	Type of the segment (TKT or PAGE).
ida_offset	INT	Unique index of the segment in the segment group.
row_count	LONG	Count of accessible rows in the segment. If the count is unknown, the value is NULL.
storage_id	UUID	Universally Unique IDentifier (UUID) for the stored segment (sys.stored_segments.storage_id).
table_id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
segment_size	LONG	Size of the segment in bytes.
root_segment_directory_group_id	CHAR	The identifier of the root-level segment directory group to which this segment belongs. For root-level directories and non-subsumed segments, this value is the same as the `segment_group_id` column.
root_segment_directory_group_ida_offset	INT	The unique index of the directory in the root-level directory group to which this segment belongs.
begin_time	LONG	Inclusive minimum bucket value, specified as an integer, for all rows in this segment. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and you can convert or compare it to other dates or times with the appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds after the epoch.
end_time	LONG	Inclusive maximum bucket value, specified as an integer, for all rows in this segment. If the bucket column is a date or time value, this column has the same unit of measure as defined by the TimeKey column and you can

Column Name	Column Type	Column Description
		convert or compare it to other dates or times with the appropriate conversion functions. When you specify a TIMESTAMP TimeKey, the value represents nanoseconds after the epoch.
begin_timestamp	TIMESTAMP	Inclusive minimum bucket value, specified as a TIMESTAMP, for all rows in this segment. If the bucket column is a unitless integer, this column is NULL.
end_timestamp	TIMESTAMP	Inclusive maximum bucket value, specified as an integer, for all rows in this segment. If the bucket column is a unitless integer, this column is NULL.
owner	LONG	Owner identifier of this stored segment.
num_deleted_rows	LONG	Number of rows deleted from this segment.
page_size_with_replication	LONG	Specifies the replication of stored pages.

sys.stop_osn_reap_requests

This table contains the list of requests to stop reaping the Ownership Sequence Number (OSN). When the system adds a request to this table, the storage cluster stops reaping OSNs until all requests are removed.

Column Name	Column Type	Column Description
request_id	UUID	The Universally Unique IDentifier (UUID) of the request.
created_at	TIMESTAMP	The timestamp that represents when the request was created.
description	CHAR	The description of the request.
cluster_id	UUID	The UUID of the cluster.

sys.storage_capacity

This table contains information about storage capacity for each node.

Column Name	Column Type	Column Description
node_id	UUID	Universally Unique IDentifier (UUID) of the node (sys.nodes).
capacity_bytes	LONG	Approximate storage capacity of the node in bytes.
cluster_id	UUID	UUID of the cluster (sys.clusters).

sys.storage_device_files

This table contains the list of all files for each storage device.

Column Name	Column Type	Column Description
node_id	UUID	The unique identifier of the node.
device_id	INT	The numeric identifier of the drive.
storage_id	UUID	The unique identifier of the file.
owner	LONG	Owner of the storage identifier. The zero value indicates an unowned storage identifier. A non-zero value indicates a segment owned after the rebuild.
segment_type	CHAR	The type of the segment (TKT, PAGE, etc.).
parent_id	UUID	The unique identifier of the parent file (for addendum parts).
normalized_drive_slot	INT	Theoretical drive slot, if applicable, that was computed from the segment group identifier. The Ocient System expects to allocate the file to this slot.
abnormal_placement	BOOLEAN	Specifies whether the Ocient System places the segment abnormally. If this value is `true`, the segment is not located on the drive slot that was computed from the segment group identifier.
segment_group_id	CHAR	The identifier of the segment group (sys.segment_groups).

sys.storage_scopes

This table contains the storage scopes defined in the system with the number of rows, page groups, and segment groups in each storage scope.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the storage scope.
row_count	LONG	The number of rows that have been currently loaded onto the scope.
num_page_groups	LONG	The number of page groups currently present in the scope.
num_tkt_segment_groups	LONG	Number of TKT segment groups currently present in the scope.
scope_prefix_len	INT	The prefix length of the scope.
transaction_timeout	LONG	The transaction timeout of the scope.
creation_timestamp	TIMESTAMP	The timestamp when the scope was created.
last_activity_timestamp	TIMESTAMP	The timestamp when the scope was last updated.
cluster_id	UUID	UUID of the cluster (sys.clusters).

sys.storage_spaces

This table contains information for all defined storage spaces.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the storage space.
name	CHAR	Name of the storage space.
is_system_storage_space	BOOLEAN	Is this storage space the system storage space
block_size	INT	Unit size that is erasure coded in bytes.
total_width	INT	The coding width. (The N in an M of N parity configuration.)
parity_width	INT	Within a coding line, the number of blocks dedicated to storing parity information.
parity_type	CHAR	The methodology used to compute parity blocks (P+Q, XOR, REED SOLOMON, REPLICATION, NONE).
parity_cycles	INT	Parity cycles calculated by multiplying by total_width to return the number of segments in the segment group.

Column Name	Column Type	Column Description
page_replication	INT	The total number of page replicas, which includes the original page.

sys.storage_used

This table contains information about storage utilization for each node.

Column Name	Column Type	Column Description
node_id	UUID	Universally Unique IDentifier (UUID) of the node (sys.nodes).
table_id	UUID	UUID of the table (sys.tables).
used_bytes	LONG	Approximate storage utilization of the table on the node in bytes.
cluster_id	UUID	UUID of the cluster (sys.clusters).

sys.stored_leaf_segment_part_inventory

This table contains the physical location for the inventory of markers for all segments.

Column Name	Column Type	Column Description
segment_group_id	CHAR	The identifier of the segment group (sys.segment_groups).
cluster_id	UUID	Universally Unique IDentifier (UUID) of the cluster (sys.clusters).
ida_offset	INT	Index of the segment in the segment group.
storage_id	UUID	UUID of the stored segment.
owner	LONG	Owner identifier of the stored segment.
root_directory_group_id	CHAR	Root segment directory identifier, if it has been subsumed.
node_id	UUID	UUID of the node (sys.nodes).
start_osn	CHAR	The starting Ownership Sequence Number (OSN) for the latest OSN range of the inventory of markers.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
end_osn	CHAR	The ending OSN for the latest OSN range of the inventory of markers.
kind	CHAR	The kind of segment (DISK, VIRTUAL).
visibility	CHAR	The visibility of the inventory of markers.
toc_name	UUID	UUID of the inventory of markers. A NULL UUID denotes the main inventory of markers of the segment.
row_count	LONG	The number of rows in the base leaf segment (not factoring in any DELETE SQL statements).

sys.stored_segments

This table contains details for each segment that is saved to disk.

Column Name	Column Type	Column Description
segment_group_id	CHAR	The identifier of the segment group (sys.segment_groups).
cluster_id	UUID	Universally Unique IDentifier (UUID) of the cluster (sys.clusters).
ida_offset	INT	Index of the segment in the segment group.
storage_id	UUID	UUID of this stored segment.
owner	LONG	Owner identifier of this stored segment.
node_id	UUID	UUID of the node (sys.nodes).
status	CHAR	The current status of the segment (MISSING, INTACT, etc.).
kind	CHAR	The kind of segment (DISK, VIRTUAL).
start_osn	CHAR	The start OSN for the latest OSN range of the segment.
end_osn	CHAR	The end OSN for the latest OSN range of the segment.
visibility	CHAR	The visibility of the stored segment. Valid values are VISIBLE (segment is queryable), QUARANTINED (segment is not queryable), HIDDEN (segment is hidden), and INVALID (segment is invalid). During segment generation, the Ocient System can make segments hidden or invalid for a short period before the segments become queryable.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
abnormal_placement	BOOLEAN	Whether the segment is placed abnormally.
row_count	LONG	Count of rows in the segment.
segment_size	LONG	Size of the segment in bytes.

sys.unhealthy_segments

This table contains information about nodes with damaged, missing, or invalid segments.

Column Name	Column Type	Column Description
node_id	UUID	The unique identifier of the node.
node_name	VARCHAR	The name of the node.
status	VARCHAR	The status of the stored segment. Values include INTACT, MISSING, and DAMAGED.
num_unhealthy_segments	BIGINT	The total number of unhealthy segments in the node.

System

sys.clusters

This table contains all clusters defined in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the cluster (sys.clusters.id).
name	CHAR	Name of the cluster.
cluster_type	CHAR	Type of the cluster.
storage_space_ids	ARRAY(UUID)	UUIDs of the storage spaces used in this cluster (sys.storage_spaces.id).

sys.compute_configurations

This table contains the compute configurations for nodes and clusters in the system.

Column Name	Column Type	Column Description
cluster_id	UUID	Universally Unique IDentifier (UUID) of the cluster that this node belongs to (sys.clusters.id).
node_id	UUID	UUID of the node (sys.nodes.id).
level	LONG	Execution level for this node or cluster.
csn	LONG	Compute sequence number.

sys.locks

This table contains information of the currently queued and granted lock requests.

Column Name	Column Type	Column Description
request_id	UUID	The unique identifier of the lock.
owner_identifier	CHAR	The unique identifier of the owner of the lock. Should be of format `<node uuid with lowercase chars>.<locking system>.<locking reason>.<process uuid>`
lock_scope_id	CHAR	An identifier on the scope of the lock, should be of format `<system>.<type>.<target unique identifier>`
lock_type	CHAR	Whether the lock is to READ or WRITE on the elements in its scope.
status	CHAR	The status of the lock, QUEUED means that it is waiting to be accepted, GRANTED means it is currently active.
create_time	LONG	The creation time, in milliseconds, of the lock relative to the underlying RAFT Consensus Log. This value is not a UNIX timestamp.
last_refresh_time	LONG	The time, in milliseconds, when the lock was last refreshed relative to the underlying RAFT Consensus Log. This value is not a UNIX timestamp
priority_id	UUID	The identifier for lock prioritization.
create_timestamp	TIMESTAMP	The timestamp that represents when the lock was created on the Raft leader.
last_refresh_timestamp	TIMESTAMP	The timestamp that represents when the lock was most recently refreshed on the Raft leader.

© 2026 Ocient, Inc.

sys.lts_cluster_info

This table contains the storage clusters defined in the system.

Column Name	Column Type	Column Description
cluster_id	UUID	Universally Unique IDentifier (UUID) of the storage cluster (sys.clusters.id).
storage_space_ids	ARRAY(UUID)	UUIDs of the storage spaces used in this cluster (sys.storage_spaces.id).

sys.node_clusters

This table contains nodes that are members of each cluster.

Column Name	Column Type	Column Description
cluster_id	UUID	Universally Unique IDentifier (UUID) of the cluster that this node belongs to.
node_id	UUID	UUID of the node.
ordinal	INT	Ordinal of this node.

sys.nodes

This table contains all nodes defined in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the node.
unique_node_number	INT	Ordinal number assigned to the node.
name	CHAR	Name of the node.
status	CHAR	Status of the node (ACCEPTED, UNACCEPTED, INVALID).
hostname	CHAR	Hostname of the node.
software_version	CHAR	Version of the software running on this node.
kernel_version	CHAR	Version of the kernel running on this node.
system_version	CHAR	Version of Linux® running on this node.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
system_memory	LONG	Amount of available system memory in bytes.
sockets	INT	Number of CPU sockets.
cores_per_socket	INT	Number of cores per CPU socket.
hugepages_1gb	INT	Number of configured 1GB huge pages.
hugepages_2mb	INT	Number of configured 2MB huge pages.
hyperthreaded	BOOLEAN	Whether the node is hyperthreaded.

sys.service_roles

This table contains the service roles defined on each node.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the service role.
service_role_type	CHAR	Type of the service role.
node_id	UUID	UUID of the node. (sys.nodes.id)
level	CHAR	Execution level of the node.

System Information

sys.function_signatures

This table contains all function signatures defined in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the function signature.
name	CHAR	Name of the function.
return_type	CHAR	Return type of the function.
arg_types	ARRAY(CHAR)	The types of all arguments in the function.
function_type	CHAR	Type of the function.

sys.reserved_words

This table contains all reserved words defined in Ocient.

Column Name	Column Type	Column Description
reserved_word	CHAR	A word that is a reserved keyword for use by Ocient.

sys.sql_messages

This table contains information about the SQL errors and warnings that the Ocient System can produce.

Column Name	Column Type	Column Description
name	CHAR	SQL error or warning name.
code	INT	SQL error or warning code.
state	CHAR	SQL error or warning state.
reason	CHAR	Description of the SQL error or warning.
is_error	BOOLEAN	Whether or not this SQL message is an error or warning.

sys.system_information

This table contains information about the Ocient System.

Column Name	Column Type	Column Description
id	UUID	Universally Unique Identifier (UUID) for the overall system.
name	CHAR	The name of the system. This system assigns this name randomly using a UUID by default. You can change the name using the `ALTER SYSTEM RENAME` SQL statement.
current_compatible_software_version	INT	The minimum software version for the system compatibility.
allowed_compatible_software_version	INT	The software version that prevents the use of features in the newest version of the system to keep backward compatibility with the specified version in

Column Name	Column Type	Column Description
		the `current_compatible_software_version` column.
feature_directory_merge	CHAR	The value of the `directory merge` feature flag for segment directory groups.

sys.system_table_columns

This table contains all columns for tables available in the system catalog.

Column Name	Column Type	Column Description
id	UUID	The Universally Unique IDentifier (UUID) of this column (sys.columns).
name	CHAR	The name of this column.
data_type	CHAR	The data type of the column.
nullable	BOOLEAN	Whether or not this column is nullable.
ordinal	LONG	The ordinal of this column.
schema_name	CHAR	The name of the schema to which this column belongs.
table_name	CHAR	The name of the table to which this column belongs.
description	CHAR	Description of the column.

sys.system_tables

This table contains all tables available in the system catalog.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the table (sys.tables).
schema	CHAR	Schema of the table.
name	CHAR	Name of the table.
table_type	CHAR	Type of the table.
category	CHAR	Category of the table.

Column Name	Column Type	Column Description
description	CHAR	Description of the table.

User Management

sys.group_roles

This table contains roles that belong to each group.

Column Name	Column Type	Column Description
group_id	UUID	Universally Unique IDentifier (UUID) of the group (sys.groups).
role_id	UUID	UUID of the role (sys.roles).

sys.groups

This table contains information, including the service class, related to each group.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the group.
name	CHAR	Name of the group.
database_id	UUID	UUID of the database (sys.databases).
service_class_id	UUID	UUID of the service class of this group (sys.service_classes).

sys.privileges

This table contains information related to the privileges granted in Ocient.

Column Name	Column Type	Column Description
timestamp	TIMESTAMP	Timestamp of the grant given.
grantor	CHAR	The user who granted this privilege.
grantee	CHAR	The user who received this privilege.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
privilege	CHAR	The privilege for the grant.
privilege_target	CHAR	The type of object to which the privilege applies. This value is NULL if the privilege applies to the granted object.
object_type	CHAR	The type of object on which this privilege was granted.
object_id	UUID	Universally Unique IDentifier (UUID) of the object.
grantable	BOOLEAN	Whether the user can grant this privilege to another user.

sys.rights

This table contains information related to the rights granted in Ocient.

Column Name	Column Type	Column Description
entity_type	CHAR	Owner of the right (user, group, role).
entity_id	UUID	Universally Unique IDentifier (UUID) of the owner.
target_type	CHAR	Type of the target.
target_id	UUID	UUID of the target.
value	CHAR	The right being given (CREATE, READ, UPDATE, DELETE, SECURITY).

sys.roles

This table contains roles in the system.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the role.
database_id	UUID	UUID of the database of this role (sys.databases).
name	CHAR	Name of this role.
description	CHAR	Detailed description of this role.

sys.user_groups

This table contains the users that belong to groups in Ocient.

Column Name	Column Type	Column Description
user_id	UUID	Universally Unique IDentifier (UUID) of the user (sys.users).
group_id	UUID	UUID of the group where the user belongs (sys.groups).

sys.user_roles

This table contains roles associated with each user.

Column Name	Column Type	Column Description
user_id	UUID	Universally Unique IDentifier (UUID) of the user (sys.users).
role_id	UUID	UUID of the role (sys.roles).

sys.users

This table contains information about users in Ocient.

Column Name	Column Type	Column Description
id	UUID	Universally Unique IDentifier (UUID) of the user.
user_name	CHAR	Username.
database_id	UUID	UUID of the database.
first_name	CHAR	First name of the user.
last_name	CHAR	Last name of the user.
email	CHAR	Email address of the user.
password_updated_at	TIMESTAMP	Timestamp that represents the date and time for the last time the password for this user was updated
state	CHAR	State.
invalid_login_attempts	INT	Number of invalid login attempts after the last successful login.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
password_days_remaining	INT	Number of days remaining until the password expires. If you do not configure the password expiration, this value is NULL.

Workload Management

sys.service_classes

This table shows information about the service classes used for workload management in Ocient. For more details on workload management, see the corresponding section of the user documentation.

Column Name	Column Type	Column Description
id	UUID	The ID of the service class.
database_id	UUID	The ID of the database that has the service class.
name	CHAR	The name of the service class.
max_temp_disk_usage	LONG	The maximum temporary disk usage for a query running with this service class.
max_elapsed_time	LONG	The maximum time, in seconds, that a query running with this service class can run.
max_concurrent_queries	LONG	The maximum number of queries that can run concurrently with this service class.
max_rows_returned	LONG	The maximum number of rows a query running with this service class can return.
scheduling_priority	DOUBLE	The initial priority for a query running with this service class.
cache_max_bytes	LONG	The maximum number of bytes in a result set, if it is eligible for cache storage, for a query running with this service class.
cache_max_time	LONG	The maximum amount of time, in seconds, that the system caches rows for a query running with this service class.
max_elapsed_time_for_caching	LONG	The maximum amount of time, in seconds, that a query running with this service class can run and have its result set cached. This value must be higher than the `max_elapsed_time` value.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
max_columns_in_result_set	LONG	The maximum number of columns allowed in the result set of a query running with this service class.
priority_adjustment_factor	DOUBLE	The amount that the system adjusts the query priority in each time interval, as specified by the `priority_adjustment_time` value, for a query running with this service class.
priority_adjustment_time	LONG	The frequency, in seconds, of each priority adjustment for a query running with this service class.
min_priority	DOUBLE	The minimum value of the adjusted priority for a query running with this service class.
max_priority	DOUBLE	The maximum value of the adjusted priority for a query running with this service class.
statement_text	CHAR	Specifies the comparison string to use for matching statement text to queries. This string takes the format specified by the statement_text_matcher_type field.
statement_text_matcher_type	CHAR	Specifies the type of string comparison to use for matching statement text to queries. This value must be LIKE or REGEX and is required if you specify a ‘statement_text’ field.
half_parallelism	BOOLEAN	Determines whether the VM uses half of the available cores for each operator in this query rather than the maximum possible parallelism. Setting this value can reduce overhead and latency for queries, but it significantly decreases data throughput. So, updating this value should be rare outside of Ocient Support work.
load_balance_shuffle	BOOLEAN	Determines whether the optimizer includes a load-balancing network operator above I/O for each query. This field can add latency to queries, but it will likely increase data throughput. If unset, the choice is left to the optimizer. So, updating this value should be rare outside of Ocient Support work.
parallelism	INT	Determines the number of parallel cores to use for each operator in the VM. This value must be nonzero. If unset, the choice is left to the VM. So, updating this value should be rare outside of Ocient Support work.

© 2026 Ocient, Inc.

Column Name	Column Type	Column Description
minimize_query_debug_records	BOOLEAN	This value determines whether the optimizer and VM should attempt to limit debug metrics and logging for queries associated with this service class. Limiting query observability may improve performance for workloads running many queries per second.
memory_optimal_strategy	BOOLEAN	This value determines whether the optimizer and VM should attempt to lower the memory requirements of a query regardless of potential execution time penalties. So, updating this value should be rare outside of Ocient Support work.
is_system_managed	BOOLEAN	This value indicates whether or not the service class is a system managed service class.

sys.service_classes_for_user

The `service_classes_for_user` table shows groups that each user belongs to and the service class for each group.

Column Name	Column Type	Column Description
user_name	VARCHAR	Username.
group_name	VARCHAR	Name of the group.
service_class_name	VARCHAR	The name of the service class.

Related Links

Information Schema

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Information Schema

The system catalog exposes a set of system views in the `information_schema` schema that contain metadata about the Ocient® System. Most of these views follow the SQL standard with additional views specific to the system.

Alphabetical List of Views

- `information_schema.columns`
- `information_schema.data_types`
- `information_schema.databases`
- `information_schema.geometry_columns`
- `information_schema.groups`
- `information_schema.index_recommendations`
- `information_schema.indexes`
- `information_schema.information_schema_catalog_name`
- `information_schema.nodes`
- `information_schema.pipeline_status`
- `information_schema.pipeline_status_historical`
- `information_schema.pipeline_table_metrics`
- `information_schema.pipelines`
- `information_schema.pipelines_historical`
- `information_schema.reserved_words`
- `information_schema.schemata`
- `information_schema.table_privileges`
- `information_schema.table_storage`
- `information_schema.tables`
- `information_schema.transactional_pipeline_status`
- `information_schema.users`

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

- `information_schema.views`

System View Descriptions

information_schema.columns

The `columns` view shows information for all columns in the system.

Column Name	Column Type	Column Description
table_catalog	VARCHAR	Name of the database.
table_schema	VARCHAR	Name of the schema.
table_name	VARCHAR	Name of the table.
column_name	VARCHAR	Name of the column.
ordinal_position	BIGINT	Ordinal of the column with respect to its table.
data_type	VARCHAR	Data type of the column (INT, CHAR, BOOLEAN, etc.).
is_nullable	VARCHAR	Specifies whether the values in this column can be NULL.
column_default	VARCHAR	The default expression of this column, if it exists.

information_schema.data_types

The `data_types` view shows all data types in the system.

Column Name	Column Type	Column Description
data_type	VARCHAR	Data type.

information_schema.databases

The `databases` view shows all databases where the current user has access.

Column Name	Column Type	Column Description
database_name	VARCHAR	Name of the database.
created_at	TIMESTAMP	Timestamp that specifies when the database was created.

© 2026 Ocient, Inc.

information_schema.geometry_columns

The `geometry_columns` view shows information about all geometry type columns in the system.

Column Name	Column Type	Column Description
f_table_catalog	VARCHAR	Name of the database.
f_table_schema	VARCHAR	Name of the schema.
f_table_name	VARCHAR	Name of the table.
f_geometry_column	VARCHAR	Name of the column.
coord_dimension	INT	The coordinate dimension.
srid	INT	The ID of the spatial reference system.
type	VARCHAR	Data type of the column (POINT, LINESTRING, POLYGON, etc.).

information_schema.groups

The `groups` view shows all groups in the system.

Column Name	Column Type	Column Description
database_name	VARCHAR	Name of the database.
group_name	VARCHAR	Name of the group.

information_schema.index_recommendations

This view contains index recommendations.

Column Name	Column Type	Column Description
table_name	VARCHAR	Name of the table.
column_name	VARCHAR	Name of the column.
sql	VARCHAR	SQL statement to create or drop the recommended index.

information_schema.indexes

The `indexes` view shows all indexes used in the system.

Column Name	Column Type	Column Description
table_catalog	VARCHAR	Name of the database.
table_schema	VARCHAR	Name of the schema.
table_name	VARCHAR	Name of the table.
index_name	VARCHAR	Name of the index.
index_type	VARCHAR	Type of the index.
column_name	VARCHAR	Name of the column.

information_schema.information_schema_catalog_name

The `information_schema_catalog_name` view shows the current database.

Column Name	Column Type	Column Description
catalog_name	VARCHAR	Name of the database.

information_schema.nodes

The `nodes` view shows all nodes defined in the system.

Column Name	Column Type	Column Description
name	VARCHAR	Name of the node.
service_role_types	VARCHAR[]	The type of service roles on this node.
operational_status	VARCHAR	The operational status of the node. Values are ACTIVE, STARTING, STOPPING, ERROR, UNKNOWN, or UNREACHABLE.
software_version	VARCHAR	Version of the software running on this node.

information_schema.pipeline_status

To view dynamic information about the status of the pipeline, query from information_schema.pipeline_status, or you can use the SHOW PIPELINE_STATUS

command.

Column Name	Column Type	Column Description
database_name	VARCHAR	Name of the database where the pipeline was created.
pipeline_name	VARCHAR	The name of this pipeline.
table_names	VARCHAR[]	An array of fully-qualified table names where the pipeline loads data.
status	VARCHAR	Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).
status_message	VARCHAR	The corresponding event message of the last event that the Ocient System sees for this pipeline from the event_message column of the sys.pipeline_events system catalog table.
duration_seconds	INT	Duration, in seconds, of how long the pipeline has been running.
files_processed	BIGINT	The number of files that have been processed for file-based loads. (includes files with these statuses: LOADED, LOADED_WITH_ERRORS, and SKIPPED)
files_failed	BIGINT	The number of files that have failed and have a file status of FAILED for file-based loads.
files_remaining	BIGINT	The number of remaining files for file-based loads. (includes these file statuses: PENDING, QUEUED, and LOADING)
files_total	BIGINT	The total number of files for file-based loads.
fraction_complete	FLOAT	The estimated fraction completion as a value from 0.0 to 1.0 for file-based batch or transactional loads. The calculation is files processed or failed divided by the total files.
records_processed	BIGINT	The number of records that the pipeline has processed. The pipeline attempts to process and load a record. This number can be greater than the value of records_loaded due to the deduplication of records that the system processes twice or where record-level errors occur.
records_loaded	BIGINT	The number of records that have successfully loaded into the system. This number reflects the records in the system and does not include records that failed to load or were eliminated as duplicates.
records_failed	BIGINT	Number of records that have failed.

information_schema.pipeline_status_historical

To view information about the status of dropped pipelines, query from information_schema.pipeline_status_historical. This system view is the historical equivalent of information_schema.pipeline_status.

Column Name	Column Type	Column Description
database_name	VARCHAR	Name of the database where the pipeline was created.
pipeline_name	VARCHAR	The name of this pipeline.
status	VARCHAR	Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).
status_message	VARCHAR	The corresponding event message of the last event that the Ocient System sees for this pipeline from the `event_message` column of the `sys.pipeline_events` system catalog table.
duration_seconds	INT	Duration, in seconds, of how long the pipeline ran.
files_processed	BIGINT	The number of files that have been processed for file-based loads. (includes files with these statuses: LOADED, LOADED_WITH_ERRORS, and SKIPPED)
files_failed	BIGINT	The number of files that have failed and have a file status of FAILED for file-based loads.
files_remaining	BIGINT	The number of remaining files for file-based loads. (includes PENDING, QUEUED, and LOADING status files)
files_total	BIGINT	The total number of files for file-based loads.
fraction_complete	FLOAT	The estimated fraction completion as a value from 0.0 to 1.0 for file-based batch or transactional loads. The calculation is files processed or failed divided by the total files.
records_processed	BIGINT	The number of records that the pipeline has processed.
records_loaded	BIGINT	The number of records that have successfully loaded into the system.
records_failed	BIGINT	Number of records that have failed.
pipeline_id	UUID	Universally Unique IDentifier (UUID) of the pipeline.

information_schema.pipeline_table_metrics

To view dynamic information about the metrics specific to each of the target tables of the pipeline, query from the information_schema.pipeline_table_metrics view.

Column Name	Column Type	Column Description
database_name	VARCHAR	The name of the database where the pipeline was created.
pipeline_name	VARCHAR	The name of this pipeline.
table_name	VARCHAR	The fully-qualified name of the table where the pipeline loads data.
records_processed	BIGINT	The number of records that the pipeline has processed for loading into the table. The pipeline attempts to process and load a record. This number can be greater than the value of the records_loaded column due to the deduplication of records that the system processes twice or where the record-level errors occur.
records_loaded	BIGINT	The number of records the Ocient System has successfully loaded into the table. This number reflects the records in the table and does not include records that the pipeline failed to load or eliminated as duplicates.
records_failed	BIGINT	The number of records the pipeline failed to load into the table. This number does not include records the pipeline failed to load to all tables, such as failed records during extraction.
processing_duration	BIGINT	Duration, in milliseconds, indicating how long the pipeline has been processing records for the table.
loading_duration	BIGINT	Duration, in milliseconds, indicating how long the pipeline has been loading records into the table.
bytes_processed	BIGINT	Estimate of the amount of source data, in bytes, that the pipeline has processed for the table.

information_schema.pipelines

To view static information that stays the same after the pipeline has been created, query from information_schema.pipelines, or you can use the SHOW PIPELINES command. The columns in the information_schema.pipelines view derive from the underlying sys.pipelines system catalog table.

Column Name	Column Type	Column Description
database_name	VARCHAR	Name of the database where the pipeline was created.

Column Name	Column Type	Column Description
pipeline_name	VARCHAR	Name of the pipeline.
loading_mode	VARCHAR	Specifies whether the pipeline runs in a one-time (BATCH), one-time transactional (TRANSACTIONAL), or continuous (CONTINUOUS) way.
source_type	VARCHAR	Source of the data (S3, KAFKA, FILESYSTEM).
status	VARCHAR	Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).
data_format	VARCHAR	Structure of the data (DELIMITED, CSV, JSON, PARQUET).
table_names	VARCHAR[]	An array of fully-qualified table names where the pipeline loads data.
created_at	TIMESTAMP	Timestamp of when the pipeline was created.
altered_at	TIMESTAMP	Timestamp that indicates when the pipeline was last altered.
creator_id	UUID	Universally Unique IDentifier (UUID) of the creator of the pipeline.
transaction_id	UUID	The UUID of the transaction scope if this pipeline runs in a one-time transactional way.

information_schema.pipelines_historical

To view static information about pipelines that have been dropped from the system, query from the `information_schema.pipelines_historical` system view. This view is the historical equivalent of the `information_schema.pipelines` view.

Column Name	Column Type	Column Description
database_name	VARCHAR	Name of the database where the pipeline was created.
pipeline_name	VARCHAR	The name of this pipeline.
loading_mode	VARCHAR	Specifies whether the pipeline runs in a one-time (BATCH), one-time transactional (TRANSACTIONAL), or continuous (CONTINUOUS) way.
source_type	VARCHAR	Source of the data (S3, KAFKA, FILESYSTEM).
status	VARCHAR	Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).

Column Name	Column Type	Column Description
data_format	VARCHAR	Structure of the data (CSV, DELIMITED, JSON, BINARY, PARQUET, XML, ASN1, AVRO).
created_at	TIMESTAMP	Timestamp that indicates when this pipeline was created.
altered_at	TIMESTAMP	Timestamp that indicates when this pipeline was last updated.
creator_id	UUID	The Universally Unique IDentifier (UUID) of the user or group who created the pipeline (sys.users/sys.groups).
transaction_id	UUID	The UUID of the transaction scope if this pipeline runs in a one-time transactional way.
dropped_at	TIMESTAMP	Timestamp that indicates when the pipeline was dropped.
rolehostd_version	VARCHAR	The rolehostd version in which this pipeline was created.
pipeline_id	UUID	UUID of the pipeline.

information_schema.reserved_words

The `reserved_words` view shows all reserved words in the system.

Column Name	Column Type	Column Description
reserved_word	VARCHAR	A word that is a reserved keyword for use by Ocient.

information_schema.schemata

The `schemata` view shows information for all schemas in the system.

Column Name	Column Type	Column Description
schema_name	VARCHAR	Name of the schema.
catalog_name	VARCHAR	Name of the database.

information_schema.table_privileges

The `table_privileges` view shows information for all table privileges in the system.

Column Name	Column Type	Column Description
grantor	VARCHAR	The user who granted this privilege.
grantee	VARCHAR	The user who received this privilege.
table_catalog	VARCHAR	Name of the database.
table_schema	VARCHAR	Name of the schema.
table_name	VARCHAR	Name of the table.
privilege_type	VARCHAR	The privilege for the grant.
is_grantable	VARCHAR	Whether the user can grant this privilege to another user.

information_schema.table_storage

The `tables_storage` view shows information about the storage used by tables.

Column Name	Column Type	Column Description
table_catalog	VARCHAR	Name of the database.
table_schema	VARCHAR	Name of the schema.
table_name	VARCHAR	Name of the table.
row_count	BIGINT	The number of rows in the table.
deleted_row_count	BIGINT	The number of deleted rows in the table.
segment_count	BIGINT	The number of segments that constitute the table.
size	BIGINT	The size of the table in bytes.

information_schema.tables

The `tables` view shows information for all tables and views in the system, including user-defined and system objects.

Column Name	Column Type	Column Description
table_catalog	VARCHAR	Name of the database.
table_schema	VARCHAR	Name of the schema.

Column Name	Column Type	Column Description
table_name	VARCHAR	Name of the table.
table_type	VARCHAR	Type of the table.
is_insertable_into	VARCHAR	Whether an INSERT command can target this table.
created_at	TIMESTAMP	Timestamp that represents when the table was created.
creator_id	UUID	Universally Unique IDentifier (UUID) of the creator of the table.

information_schema.transactional_pipeline_status

To view information about the status of transactional pipelines (both active and dropped), query from information_schema.transactional_pipeline_status. This system view combines information_schema.pipeline_status and information_schema.pipeline_status_historical, filtered to pipelines with a non-NULL `transaction_id` column.

Column Name	Column Type	Column Description
transaction_id	UUID	The Universally Unique IDentifier (UUID) of the transaction scope for this pipeline.
database_name	VARCHAR	Name of the database where the pipeline was created.
pipeline_name	VARCHAR	The name of this pipeline.
table_names	VARCHAR[]	An array of fully-qualified table names where the pipeline loads data. This value is NULL for dropped pipelines.
status	VARCHAR	Status of the pipeline (RUNNING, STOPPED, COMPLETED, FAILED).
status_message	VARCHAR	The corresponding event message of the last event that the Ocient System sees for this pipeline from the `event_message` column of the `sys.pipeline_events` system catalog table.
duration_seconds	INT	Duration, in seconds, of how long the pipeline has been running.
files_processed	BIGINT	The number of files that have been processed for file-based loads. (includes files with these statuses: LOADED, LOADED_WITH_ERRORS, and SKIPPED)
files_failed	BIGINT	The number of files that have failed and have a file status of FAILED for file-based loads.

Column Name	Column Type	Column Description
files_remaining	BIGINT	The number of remaining files for file-based loads. (includes PENDING, QUEUED, and LOADING status files)
files_total	BIGINT	The total number of files for file-based loads.
fraction_complete	FLOAT	The estimated fraction completion as a value from 0.0 to 1.0 for file-based batch or transactional loads. The calculation is files processed or failed divided by the total files.
records_processed	BIGINT	The number of records that the pipeline has processed.
records_loaded	BIGINT	The number of records that have successfully loaded into the system.
records_failed	BIGINT	Number of records that have failed.
pipeline_id	UUID	UUID of the pipeline.

information_schema.users

The `users` view shows all users in the system.

Column Name	Column Type	Column Description
database_name	VARCHAR	Name of the database.
user_name	VARCHAR	Username of the user.

information_schema.views

The `views` view shows information for all user-defined and system views in the system.

Column Name	Column Type	Column Description
table_catalog	VARCHAR	Name of the database.
table_schema	VARCHAR	Name of the schema.
table_name	VARCHAR	Name of the table.
view_definition	VARCHAR	Query used to generate the view content.
created_at	TIMESTAMP	Timestamp that represents the date and time for the creation of the view.

Column Name	Column Type	Column Description
updated_at	TIMESTAMP	Timestamp that represents the date and time for the last update of the view.
creator_id	UUID	Universally Unique IDentifier (UUID) of the creator of the view.

SHOW Commands

The `SHOW` commands are additional commands you can use to retrieve metadata about the system. These commands are tightly coupled with the `information_schema` schema and offer a more user-friendly syntax alternative. Sometimes, unlike their `information_schema` counterparts, `SHOW` commands only return user-defined objects.

SHOW Command	Description
SHOW COLUMNS	This statement returns columns from user-defined tables and views where the current user has access.
SHOW DATA_TYPES	This statement returns all data types in the system.
SHOW DATABASES	This statement returns all databases where the current user has access.
SHOW GEOMETRY_COLUMNS	This statement returns all geometry type columns where the current user has access.
SHOW GROUPS	This statement returns all groups in the system where the current user has read access.
SHOW INDEXES	This statement returns all indexes on user-defined tables where the current user has access.
SHOW NODES	This statement returns all nodes in the system where the current user has read access.
SHOW PIPELINE_STATUS	This statement returns the status of all pipelines in the system where the current user has read access.
SHOW PIPELINES	This statement returns all pipelines in the system where the current user has read access.
SHOW RESERVED_WORDS	This statement returns all reserved words in the system.
SHOW SCHEMATA	This statement returns all user-defined schemas where the current user has access.
SHOW SYSTEM TABLES	This statement returns all system tables in the `sys` or `information_schema` schemas.

© 2026 Ocient, Inc.

SHOW Command	Description
SHOW TABLE_PRIVILEGES	This statement returns all table privileges for tables where the current user has access.
SHOW TABLES	This statement returns all user-defined tables where the current user has access.
SHOW USERS	This statement returns all users in the system where the current user has read access.
SHOW VIEWS	This statement returns all user-defined views where the current user has access.

Related Links

SQL Reference **Database Administration** **System Catalog**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Discover Insights From System Catalog Tables

You can query the Ocient® System catalog tables to discover important information on system objects such as databases, segments, tables, and indexes. These statistics can be important for monitoring your system to ensure it operates as intended.

However, querying these tables can be challenging because generating insights about your system can involve writing complex queries that involve merging, grouping, and filtering multiple system catalog tables.

Here, you can find examples of useful queries that you can use as templates for discovering insights about your system.

> Most of these example queries use the `data_type_coverage` table available with the Ocient Simulator. For details, see **Ocient Simulator**.

Segment System Catalog Tables

For specific column definitions of the system catalog tables referenced in these examples, see the **Storage** section of the System Catalog page.

Segment Size

Check the size of segments and return insights about the segment storage allocation and the number of table rows contained in each segment.

© 2026 Ocient, Inc.

SQL

```sql
SELECT current_timestamp,
    sg.segment_type,
    t.name,
    (COUNT(*)) AS num_segments,
    (SUM(s.row_count)) AS total_rows,
    (AVG(s.row_count)) AS avg_rows_per_segment,
    (DOUBLE(SUM(s.segment_size)) / 1000000000) AS total_size_gb,
    (MAX(s.segment_size / 1000000)) AS max_size,
    (MIN(s.segment_size / 1000000)) AS min_size,
    (AVG(s.segment_size) / 1000000) AS avg_size_mb,
    (STDEV(DOUBLE(s.segment_size) / 1000000)) AS stdev_size_mb
FROM
    sys.segment_groups AS sg
    JOIN sys.tables t ON t.id = sg.table_id
    JOIN sys.segments s ON s.segment_group_id = sg.id
GROUP BY
    sg.segment_type, t.name
ORDER BY
    total_size_gb DESC,
    total_rows DESC,
    sg.segment_type,
    t.name;
```

Output

Text

```
|  "current_timestamp()"  | "segment_type" |       "name"       | "num_segments" |
|-------------------------|----------------|--------------------|----------------|
| 2024-07-31 16:23:22.863 | TKT_SEGMENT    | data_type_coverage |              3 |
```

For information on the system catalog tables queried in this example, see **sys.segments** and **sys.tables**.

Page and Segment Information

Check the details for pages and segments in the system.

SQL

```
WITH cte AS (
SELECT
    COALESCE(v.schema, t.schema) AS schema,
    COALESCE(v.name, t.name) AS name,
    stored_segments.status,
    stored_segments.kind,
    SUM(CASE WHEN segments.segment_type = 'PAGE' THEN 1 ELSE 0 END) AS pagecount,
    SUM(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN 1 ELSE 0 END) AS segc
    SUM(CASE WHEN segments.segment_type = 'PAGE' THEN segments.row_count ELSE 0 EN
    SUM(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN segments.row_count EL
    MIN(CASE WHEN segments.segment_type = 'PAGE' THEN sg.begin_time ELSE NULL END)
    MAX(CASE WHEN segments.segment_type = 'PAGE' THEN sg.end_time ELSE NULL END) A
    MIN(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN sg.begin_time ELSE NU
    MAX(CASE WHEN segments.segment_type = 'TKT_SEGMENT' THEN sg.end_time ELSE NULL
FROM
    sys.segments
JOIN sys.stored_segments ON
    segments.segment_group_id = stored_segments.segment_group_id
    AND segments.ida_offset = stored_segments.ida_offset
JOIN sys.segment_groups sg ON
    sg.id = segments.segment_group_id
JOIN sys.tables AS t ON
    t.id = sg.table_id
LEFT JOIN sys.views AS v ON
    v.global_dictionary_compression_table_id = t.id
WHERE
    stored_segments.end_osn = 'OSN_INFINITY'
GROUP BY
    1,
    2,
    stored_segments.status,
    stored_segments.kind )
SELECT
    schema,
    name,
    status,
    kind,
    pagecount,
    segcount,
    page_rows,
    seg_rows,
```

```
    page_rows + seg_rows AS tot_rows,
    CEILING(page_rows /(pagecount +.000001)) AS rows_per_page,
    CEILING(seg_rows /(segcount +.000001)) AS rows_per_seg,
    to_timestamp(minpagetime / 1000000000) AS minpagetime,
    to_timestamp(maxpagetime / 1000000000) AS maxpagetime,
    to_timestamp(minsegtime / 1000000000) AS minsegtime,
    to_timestamp(maxsegtime / 1000000000) AS maxsegtime,
    NOW()
FROM
    cte
ORDER BY
    schema,
    name;
```

Output

Text

```
| "schema" |       "name"       | "status" | "kind" | "pagecount" | "segcount" | '
|----------|--------------------|----------|--------|-------------|------------|--
| loading  | data_type_coverage | INTACT   | DISK   |           0 |          3 |
```

Storage Spaces on a System

Check all storage spaces on your system.

SQL

```
SELECT *
FROM sys.storage_spaces;
```

Output

Text

```
|                 "id"                 |       "name"       | "is_system_storage
|--------------------------------------|--------------------|-------------------
| ce9f9c2e-d3fe-4101-a2b2-6f7f2580327b | ss0                | false
| "7277656e-6465-6c20-7761-732068657265" | systemstoragespace | true
```

Query Management System Catalog Tables

For specific column definitions of the system catalog tables referenced in these examples, see the **Monitoring** section of the System Catalog page.

Running Queries

Check all queries currently running on the system.

The output of the `sys.queries` table depends on what database you are logged into and your system role. For details, see **Query Visibility**.

SQL

```
SELECT
    queries.query_id,
    queries.initial_priority,
    queries.effective_priority,
    queries.user,
    queries.total_time,
    queries.status,
    queries.rows_returned,
    queries.bytes_returned
FROM
    sys.queries
WHERE
    queries.sql NOT LIKE '%sys.queries%';

SELECT
    q.query_id,
    q.user,
    q.sql,
    q.status,
    "database_name",
    q.bytes_returned
FROM
    sys.queries q;
```

Output

Text

```
|               "query_id"               |    "user"    |      "sql"       | "stat
|----------------------------------------|--------------|------------------|------
| "66b27dde-17df-403a-82d7-65e34b6fd07f" | admin@system |  "example_query" | RUNNI
```

Recent Queries

Check the most recent queries completed by the system.

The output of the `sys.completed_queries` table depends on what database you are logged into and your system role. For details, see **Query Visibility**.

SQL

```sql
SELECT user,
    timestamp_start,
    total_time,
    state,
    reason,
    sql
FROM sys.completed_queries
ORDER BY timestamp_start DESC
LIMIT 10;
```

Output

Text

```
|    "user"    |       "time_start"        | "total_time" | "state" |
|--------------|---------------------------|--------------|---------|-------------
| admin@system | "2024-08-06T21:51:29.284" |           84 | "00000" | The operatic
| admin@system | "2024-08-06T21:45:33.952" |           58 | "00000" | The operatic
| admin@system | "2024-08-06T21:41:04.521" |          205 | "00000" | The operatic
| admin@system | "2024-08-06T21:38:19.740" |           90 | "00000" | The operatic
| admin@system | "2024-08-06T21:37:35.181" |            5 | "42703" | The referenc
| admin@system | "2024-08-06T21:37:33.710" |            5 | "42703" | The referenc
| admin@system | "2024-08-06T21:36:03.692" |           11 | "42703" | The referenc
| admin@system | "2024-08-06T21:35:51.301" |            4 | "42703" | The referenc
| admin@system | "2024-08-06T21:35:10.331" |            5 | "42703" | The referenc
| admin@system | "2024-08-06T21:34:42.970" |          208 | "00000" | The operatic
```

Database and Table System Catalog Tables

For specific column definitions of the system catalog tables used in these examples, see the **Databases** section of the System Catalog page.

Database Sizes

Check the total size of all databases on your system.

SQL

```sql
SELECT TO_CHAR(SUM(sp.size), '999,999,999,999,999') AS totsize,
    d.name
    FROM sys.segment_parts AS sp
    JOIN sys.segment_groups AS sg
        ON sp.segment_group_id = sg.id
    JOIN sys.tables AS t
        ON sg.table_id = t.id
    JOIN sys.databases AS d
        ON t.database_id = d.id
 GROUP BY d.name
 ORDER BY 1 DESC;
```

Output

Text

```
| "totsize" | "name" |
|-----------|--------|
| 204276004 | test   |
```

Table Sizes

Check the total size of all tables on the system.

SQL

```sql
WITH
cte AS (
    SELECT
        sg.id AS segment_group_id,
        sp.name AS segment_name,
        sp.part_type AS partition_type,
        sp.size AS segment_size,
        sg.segment_type,
        sg.status,
        t.schema,
        t.name
    FROM sys.segment_parts AS sp
    INNER JOIN sys.segment_groups AS sg ON
        sp.segment_group_id = sg.id
    INNER JOIN sys.tables AS t ON
        sg.table_id = t.id ),
ctefinal AS (
    SELECT schema, name, SUM(segment_size) AS total_table_size
    FROM cte
    GROUP BY schema, name )
SELECT SCHEMA, name, total_table_size, SUM(total_table_size)
    OVER( ORDER BY total_table_size DESC ) AS running_total
    FROM ctefinal
    ORDER BY total_table_size DESC;
```

Output

Text

```
| "schema" |       "name"       | "total_table_size" | "running_total" |
|----------|--------------------|--------------------|-----------------|
| loading  | data_type_coverage |          204276004 |       204276004 |
```

Table Statistics

This query provides a breakdown of table statistics and partition information.

© 2026 Ocient, Inc.

SQL

```sql
WITH
cte1 AS (
    SELECT
        sg.id AS segment_group_id,
        sp.name AS segment_name,
        sp.part_type AS partition_type,
        sp.size AS segment_size,
        sg.segment_type,
        sg.status,
        t.schema,
        t.name
    FROM sys.segment_parts AS sp
    INNER JOIN sys.segment_groups AS sg ON
        sp.segment_group_id = sg.id
    INNER JOIN sys.tables AS t ON
        sg.table_id = t.id ),
cte AS (
    SELECT
        COUNT(*) AS num_segments,
        SUM(segment_size) AS partition_size,
        partition_type,
        SUM(SUM(segment_size)) OVER(PARTITION BY schema, name) AS total_table_size
        SUM(COUNT(*)) OVER(PARTITION BY schema, name) AS total_table_segments,
        schema,
        name
    FROM cte1
    GROUP BY
        partition_type,
        schema,
        name )
SELECT
    name,
    num_segments,
    partition_size,
    partition_type,
    CAST(FLOAT(partition_size)/(total_table_size + 1) * 100
        AS DECIMAL(6,3)) AS percent_of_total,
    total_table_size,
    total_table_segments
FROM cte
ORDER BY
```

```
    name,
    partition_size DESC;
```

Output

Text

```
|       "name"       | "num_segments" | "partition_size" |    "partition_type"
|--------------------|----------------|------------------|-----------------------
| data_type_coverage |              3 |        113397760 | parity_data
| data_type_coverage |              3 |         75599872 | data
| data_type_coverage |              3 |           172032 | skip_lists
| data_type_coverage |              3 |           172032 | copy_skip_lists
| data_type_coverage |              3 |            90112 | parity_stats
| data_type_coverage |              3 |            59210 | stats
| data_type_coverage |              3 |            12288 | table_of_contents
| data_type_coverage |              3 |            12288 | parity_index
| data_type_coverage |              3 |            12288 | copy_table_of_contents
| data_type_coverage |              3 |             8192 | index
| data_type_coverage |              3 |             4096 | parity_summary_stats
| data_type_coverage |              3 |              906 | column_metadata
| data_type_coverage |              3 |              906 | copy_column_metadata
| data_type_coverage |              3 |              710 | summary_stats
```

For information on the system catalog tables queried in this example, see **sys.segment_groups** and **sys.segment_parts**.

Compression

Check the information on the compression for each column in the specified table.

SQL

```sql
SELECT r."table",
    c.NAME,
    r.is_rle_enabled,
    r.is_ne_enabled,
    r.raw_size,
    r.compressed_size,
    r.is_deltadelta_enabled,
    r.num_deltadelta_blocks,
    r.num_rle_blocks,
    r.num_nerle_blocks,
    r.num_uncompressed_blocks,
    r.num_total_blocks
FROM sys.columns c,
    ( SELECT t.name AS "table",
        t.id,
        cci.is_deltadelta_enabled,
        cci.is_rle_enabled,
        cci.is_ne_enabled,
        cci.raw_size,
        cci.compressed_size,
        cci.num_deltadelta_blocks,
        cci.num_rle_blocks,
        cci.num_nerle_blocks,
        cci.num_uncompressed_blocks,
        cci.num_total_blocks,
        cci.ordinal
        FROM sys.tables AS t,
            sys.columns_compression_info AS cci
        WHERE t.id = cci.table_id
            AND t.name = 'my_table' ) AS r
WHERE r.id = c.table_id
    AND r.ordinal = c.ordinal;
```

Output

Text

```
|      "table"       |    "name"     | "is_rle_enabled" | "is_ne_enabled" | "raw_s
|--------------------|---------------|------------------|-----------------|-------
| data_type_coverage | col_double    | true             | true            |     90
| data_type_coverage | col_int       | true             | true            |     50
| data_type_coverage | col_tinyint   | true             | true            |     20
| data_type_coverage | col_date      | true             | true            |     50
| data_type_coverage | col_decimal   | true             | true            |    110
| data_type_coverage | col_ip        | true             | true            |    170
| data_type_coverage | col_point     | true             | true            |    170
| data_type_coverage | col_timestamp | true             | true            |     90
| data_type_coverage | col_bigint    | true             | true            |     90
| data_type_coverage | col_binary    | true             | true            |     30
| data_type_coverage | col_boolean   | true             | true            |     20
| data_type_coverage | col_ipv4      | true             | true            |     50
| data_type_coverage | col_time      | true             | true            |     90
| data_type_coverage | col_float     | true             | true            |     50
| data_type_coverage | col_uuid      | true             | true            |    170
| data_type_coverage | col_smallint  | true             | true            |     30
```

For information on the system catalog table queried in this example, see **sys.columns_compression_info**.

Compression for the Entire Table

This query summarizes the total compression of the specified table. In the example, replace the table name and schema name accordingly.

SQL

```
SELECT SUM(raw_size),
    SUM(compressed_size)
FROM (
    SELECT t.name,
        c.name AS colname,
        co.raw_size,
        co.compressed_size,
        SUM(co.compressed_size) OVER(PARTITION BY t.schema, t.name) AS table_size,
        100 -((100 * co.compressed_size) / co.raw_size) AS col_compress_percent,
        DECIMAL(
           FLOAT(co.compressed_size) * 100 / SUM(co.compressed_size)
           OVER(PARTITION BY t.schema, t.name),
           3,
           1
           ) AS percent_of_table
    FROM (
        SELECT table_id,
            ordinal,
            raw_size,
            compressed_size
        FROM sys.columns_compression_info
        UNION ALL
        SELECT table_id,
            ordinal,
            raw_size,
            compressed_size
        FROM sys.vl_columns_compression_info
      ) AS co
    JOIN sys.tables AS t ON co.table_id = t.id
    JOIN sys.columns AS c ON c.table_id = t.id
        AND co.ordinal = c.ordinal
    WHERE t.name = 'data_type_coverage'
        AND t.schema = 'loading'
    ORDER BY co.ordinal
    ) AS abc;
```

Output

Text

```
| "sum(raw_size)" | "sum(compressed_size)" |
|-----------------|------------------------|
|        73788914 |               73592306 |
```

For information on the system catalog table queried in this example, see **sys.columns_compression_info**.

Index Size

Check the total size of each index across the different segment groups.

In this example, the table has three secondary indexes added before loading data.

SQL

```
SELECT p.name AS uuid,
    p.part_type,
    i.name AS index_name,
    SUM(p.size) AS sum_size
FROM (
    SELECT *
    FROM sys.segment_parts
    WHERE part_type LIKE 'index'
    ) AS p
    INNER JOIN sys.indexes i ON p.name = i.id
GROUP BY 1,2,3
ORDER BY 1,2;
```

Output

Text

```
|                 "uuid"                 | "part_type"  | "index_name" | "sum_size
|----------------------------------------|--------------|--------------|----------
| "669cbcdf-772f-4c97-b250-0cc9e573cd7b" | index        | idx01        |    252723
| cccf6f5e-96a8-4621-b947-f318d120f6a0   | index        | idx03        |     18022
| d75ae71a-207e-4723-a86c-ecffc798c7d0   | index        | idx02        |    319078
```

For information on the system catalog tables queried in this example, see **sys.indexes** and **sys.segment_parts**.

Column Sizes

This query finds the size of all columns for the specified table. Before running this query on your system, replace `table_name` in the example.

© 2026 Ocient, Inc.

SQL

```sql
WITH cte AS (
    SELECT
        table_id,
        ordinal,
        compressed_size
    FROM
        sys.columns_compression_info
    UNION ALL
    SELECT
        table_id,
        ordinal,
        compressed_size
    FROM
        sys.vl_columns_compression_info )
SELECT
    ( SELECT t.name
        FROM sys.tables AS t
        WHERE t.id = cte.table_id ) AS table_name,
    ordinal,
    compressed_size,
    ( SELECT name
        FROM sys.columns AS c
        WHERE c.table_id = cte.table_id
            AND c.ordinal = cte.ordinal ) AS name,
    DOUBLE(compressed_size) / SUM(compressed_size) OVER() AS percent_of_total
FROM cte
WHERE
    table_id IN (
        SELECT id
        FROM sys.tables
        WHERE name LIKE 'table_name' )
ORDER BY compressed_size;
```

Output

Text

"table_name"	"ordinal"	"compressed_size"	"name"	"percent_
data_type_coverage	25	8192		0.0001(
data_type_coverage	21	8192		0.0001(
data_type_coverage	2	102400	col_boolean	0.0013534
data_type_coverage	18	204800	col_tinyint	0.0027068
data_type_coverage	19	266240		0.003518
data_type_coverage	4	294912	col_date	0.0038978
data_type_coverage	12	307200	col_smallint	0.004060
data_type_coverage	1	307200	col_binary	0.004060
data_type_coverage	10	507904	col_ipv4	0.006713
data_type_coverage	7	507904	col_float	0.006713
data_type_coverage	8	507904	col_int	0.006713
data_type_coverage	3	699370	col_char	0.009243
data_type_coverage	23	783147	col_varbinary	0.010350
data_type_coverage	6	905216	col_double	0.011964
data_type_coverage	17	905216	col_timestamp	0.011964
data_type_coverage	16	905216	col_time	0.011964
data_type_coverage	0	905216	col_bigint	0.011964
data_type_coverage	5	1105920	col_decimal	0.014617
data_type_coverage	13	1712128	col_point	0.022629
data_type_coverage	22	1712128	col_uuid	0.022629
data_type_coverage	11	1712128	col_ip	0.022629
data_type_coverage	20	1784388		0.02358
data_type_coverage	24	1788957	col_varchar	0.023644
data_type_coverage	14	8500000	col_linestring	0.11234
data_type_coverage	9	21477958	col_int_array	0.28387
data_type_coverage	15	27739482	col_polygon	0.3666

This output is based on a query using the table `data_type_coverage`.

Column Cardinality

This query finds the cardinality of each column in the specified table. Knowing the cardinality is useful for determining which compression scheme to use for a column.

SQL

```sql
SELECT t.schema,
    t.name,
    c.name,
    c.ordinal,
    cc.cardinality
FROM sys.column_cardinalities AS cc
    LEFT JOIN sys.tables AS t ON cc.table_id = t.id
    LEFT JOIN sys.columns AS c ON cc.column_id = c.id
WHERE t.name LIKE 'data_type_coverage'
ORDER BY c.ordinal;
```

Output

Text

```
| "schema" |       "name"       |    "name_1"    | "ordinal" | "cardinality" |
|----------|--------------------|----------------|-----------|---------------|
| loading  | data_type_coverage | col_bigint     |         0 |         98551 |
| loading  | data_type_coverage | col_binary     |         1 |         48048 |
| loading  | data_type_coverage | col_boolean    |         2 |             2 |
| loading  | data_type_coverage | col_char       |         3 |         95206 |
| loading  | data_type_coverage | col_date       |         4 |         29634 |
| loading  | data_type_coverage | col_decimal    |         5 |         20059 |
| loading  | data_type_coverage | col_double     |         6 |         53546 |
| loading  | data_type_coverage | col_float      |         7 |         75680 |
| loading  | data_type_coverage | col_int        |         8 |         99319 |
| loading  | data_type_coverage | col_int_array  |         9 |       5000855 |
| loading  | data_type_coverage | col_int_array  |         9 |         99899 |
| loading  | data_type_coverage | col_ipv4       |        10 |         99908 |
| loading  | data_type_coverage | col_ip         |        11 |         99875 |
| loading  | data_type_coverage | col_smallint   |        12 |         51227 |
| loading  | data_type_coverage | col_point      |        13 |        100000 |
| loading  | data_type_coverage | col_linestring |        14 |        100000 |
| loading  | data_type_coverage | col_polygon    |        15 |         99153 |
| loading  | data_type_coverage | col_time       |        16 |         99902 |
| loading  | data_type_coverage | col_timestamp  |        17 |         99829 |
| loading  | data_type_coverage | col_tinyint    |        18 |           256 |
| loading  | data_type_coverage | col_uuid       |        22 |         99892 |
| loading  | data_type_coverage | col_varbinary  |        23 |         68834 |
| loading  | data_type_coverage | col_varchar    |        24 |         93258 |
| loading  | data_type_coverage |                |           |         10410 |
| loading  | data_type_coverage |                |           |             2 |
| loading  | data_type_coverage |                |           |             1 |
| loading  | data_type_coverage |                |           |         93333 |
| loading  | data_type_coverage |                |           |             1 |
```

For information on the system catalog table queried in this example, see **sys.column_cardinalities**.

Cluster and Node System Catalog Tables

For specific column definitions of the system catalog tables referenced in these examples, see the **System** section of the System Catalog page.

Nodes Present on a System

Check the status of all nodes in the system.

SQL

```
SELECT name,
    status
FROM sys.nodes;
```

Output

Text

```
|   "name"    | "status" |
|-------------|----------|
| foundation2 | ACCEPTED |
| sql         | ACCEPTED |
| loader      | ACCEPTED |
| foundation1 | ACCEPTED |
| foundation0 | ACCEPTED |
```

Nodes Roles on a System

Check the roles of each node in the system.

© 2026 Ocient, Inc.

SQL

```sql
SELECT
    n.name, n.status,
    STRING_AGG(DISTINCT c.name, ', ' ORDER BY c.name) in_clusters,
    STRING_AGG(DISTINCT sr.service_role_type, ', ' ORDER BY sr.service_role_type)
FROM
    sys.nodes n
    LEFT JOIN sys.node_clusters nc ON nc.node_id = n.id
    LEFT JOIN sys.clusters c ON c.id = nc.cluster_id
    LEFT JOIN sys.service_roles sr ON sr.node_id = n.id
GROUP BY
    n.name, n.status
ORDER BY
    n.name;
```

Output

Text

```
|   "name"    | "status" |    "in_clusters"    |      "with_roles"       |
|-------------|----------|---------------------|-------------------------|--------
| foundation0 | ACCEPTED | "foundation_cluster |  foundation_cluster-vm" | "health
| foundation1 | ACCEPTED | "foundation_cluster |  foundation_cluster-vm" | "health
| foundation2 | ACCEPTED | "foundation_cluster |  foundation_cluster-vm" | "health
| loader      | ACCEPTED |                     | "health                 |  strean
| sql         | ACCEPTED | initial-vm          | "admin                  |  health
```

Node Status

Check the operational status, software versions, and assigned roles of your nodes. Possible status values include `ACTIVE`, `STARTING`, `STOPPING`, `ERROR`, `UNKNOWN`, or `UNREACHABLE`.

© 2026 Ocient, Inc.

SQL

```
SELECT name,
    operational_status,
    software_version,
    ARRAY_AGG(service_role_type)
FROM sys.node_status AS ns
    LEFT JOIN sys.nodes AS n ON ns.node_id = id
    LEFT JOIN sys.service_roles AS sr ON sr.node_id = n.id
GROUP BY name,
    operational_status,
    software_version;
```

Output

Text

```
|    name     | operational_status | software_version | array_agg(service_role_typ
|-------------|--------------------|------------------|---------------------------
| loader      | Active             |           24.0.0 | ['health','streamloader']
| foundation2 | Active             |           24.0.0 | ['health','lts','operatory
```

Foundation Nodes on a Specific Cluster

Check for all Foundation Nodes on your system clusters.

SQL

```
SELECT n.name,
    r.name
FROM sys.nodes AS n,
    ( SELECT *
        FROM sys.clusters AS c,
            sys.node_clusters AS nc
        WHERE c.cluster_type = 'Foundation'
            AND c.id = nc.cluster_id ) AS r
WHERE r.node_id = n.id;
```

Output

Text

```
| foundation2 | foundation_cluster |
|-------------|--------------------|
| foundation0 | foundation_cluster |
| foundation1 | foundation_cluster |
```

Drive Status

Check the status of all drives assigned to nodes in the system.

SQL

```sql
SELECT
    n.name AS node_name,
    s.node_id,
    s.id AS serial_number,
    s.pci_address,
    s.device_status,
    s.device_model
FROM
    sys.nodes AS n
JOIN sys.storage_device_status AS s
    ON n.id = s.node_id;
```

Output

Text

```
| "node_name" |               "node_id"                |            "serial_numbe
|-------------|----------------------------------------|-------------------------
| foundation1 | "6a96b307-57bc-477f-952c-e6cab38a7922" | "72062649-396c-4424-9da5-
| foundation1 | "6a96b307-57bc-477f-952c-e6cab38a7922" | ee2f0013-c7b7-4750-833e-3
| loader      | "83f0ecf9-9a31-4a56-85b9-7eb610447185" | ff3186fb-502f-417c-bcab-8
| loader      | "83f0ecf9-9a31-4a56-85b9-7eb610447185" | "11ff09e2-a6c1-4f76-ae96-
| foundation0 | "9330a0b3-b3b7-4503-949c-043b196c0cc4" | "6156962f-fcf6-4299-bbd7-
| foundation0 | "9330a0b3-b3b7-4503-949c-043b196c0cc4" | e6a4b24f-0d49-4704-b7ce-1
| sql         | "1198a86f-549d-4736-ac31-1ba54a2b02e7" | "37203a6c-9deb-47cf-8194-
| sql         | "1198a86f-549d-4736-ac31-1ba54a2b02e7" | "0921d62f-a31c-4681-9a56-
| foundation2 | babadc9c-f2a3-4512-ab9a-bab3fd88a544   | "842881df-b71c-4cfc-847b-
| foundation2 | babadc9c-f2a3-4512-ab9a-bab3fd88a544   | fc0e490a-a295-491d-b6a6-8
```

Roles, Privileges, and Service Class System Catalog Tables

For specific column definitions of the system catalog tables referenced in these examples, see the **User Management** section of the System Catalog page.

Group Assignment for Users

Check for all users on the system and their corresponding assigned group.

SQL

```sql
SELECT g.name AS group_name,
    r.user_name
FROM sys.groups AS g,
    ( SELECT u.user_name,
        ug.group_id
        FROM sys.users AS u,
            sys.user_groups AS ug
        WHERE u.id = ug.user_id ) AS r
ORDER BY g.name ASC;
```

Output

Text

```
| group_name | user_name |
|------------|-----------|
| DBA        | Humphrey  |
```

User Roles

Check for all users on the system and their assigned role.

SQL

```sql
SELECT rl.name AS role_name,
    r.user_name
FROM sys.roles AS rl,
    ( SELECT u.user_name,
        ur.role_id
        FROM sys.users AS u,
            sys.user_roles AS ur
        WHERE u.id = ur.user_id ) AS r
ORDER BY rl.name ASC;
```

Output

Text

```
|       role_name         | user_name |
|-------------------------|-----------|
| system administrator    | Humphrey  |
| system analyst          | Grimey    |
| database administrator | Charles   |
| database analyst        | Edna      |
```

User Privileges

Check the privileges for the specific table name. Before running on your system, replace `username` in the example.

SQL

```
SELECT t.schema AS db_name,
    t.name AS table_name,
    r.privilege
FROM sys.tables AS t,
    ( SELECT object_type,
        object_id,
        privilege
        FROM sys.PRIVILEGES
        WHERE grantee = 'username' ) AS r
WHERE r.object_id = t.id
ORDER BY t.schema, t.name ASC;
```

Output

Text

```
|  db_name |     table_name     |  privilege  |
|----------|--------------------|-------------|
| loading  | data_type_coverage | VIEW        |
| loading  | data_type_coverage | SELECT      |
```

Privileges for the Specific Table

Check the granted privileges across all database objects of the specific type, such as `GROUP`, `TABLE`, or `DATABASE`. In this example, the query checks for `TABLE` privileges.

SQL

```
SELECT p.grantee,
    t.schema,
    t.name,
    p.privilege
FROM sys.tables AS t,
    sys.privileges AS p
WHERE object_type = 'TABLE'
    AND p.object_id = t.id;
```

© 2026 Ocient, Inc.

Output

Text

```
| admin@system | admin@system |   example_table    | DELETE   |
|--------------|--------------|--------------------|----------|
| admin@system | admin@system | example_table      | DROP     |
| admin@system | admin@system | example_table      | SYSAUTH  |
| admin@system | admin@system | example_table      | LOAD     |
| admin@system | admin@system | example_table      | INSERT   |
| admin@system | admin@system | example_table      | ALTER    |
| admin@system | admin@system | example_table      | SELECT   |
| admin@system | admin@system | example_table      | VIEW     |
```

Groups Assigned to Service Classes

Check for the service class assignment for all groups.

SQL

```
SELECT g.name,
    sc.name
FROM sys.groups AS g,
    sys.service_classes AS sc
WHERE g.service_class_id = sc.id;
```

Output

Text

```
|  group   |    sc_name    |
|----------|---------------|
| analyst  | high_priority |
| dbadmin  | low_priority  |
```

Service Class Settings

Check for the settings for the specified service class. Before running on your system, replace the `high priority` service class name in the example.

For details about service class settings, see **Workload Management and Service Classes**.

SQL

```
SELECT *
FROM sys.groups AS g,
    sys.service_classes AS sc
WHERE g.service_class_id = sc.id
    AND sc.name = 'high_priority';
```

Output

Text

```
| id                                   | name     | database_id
|--------------------------------------|----------|-------------------------------
| e9c92a1b-32a8-4f88-ab72-485ff8b24f53 | analyst | e80010d7-6f26-438c-8461-11af309
```

Service Classes for the Specific User

Check for the service class of the specific user. Before running on your system, replace the `jmack@test` username in this example with the specific username that you want to query.

SQL

```
SELECT r2.name,
    sc.name,
    r2.user_name
FROM sys.service_classes AS sc,
    ( SELECT g.name,
        g.service_class_id,
        r.user_name
        FROM sys.groups AS g,
            ( SELECT u.user_name,
                ug.group_id
                FROM sys.users AS u,
                    sys.user_groups AS ug
                WHERE u.id = ug.user_id
                    AND u.user_name = 'jmack@test' ) AS r ) AS r2;
```

Output

Text

```
|  group  |  service_class  |  user_name  |
|---------|-----------------|-------------|
| analyst | Default         | jmack@test  |
| analyst | high_priority   | jmack@test  |
```

Data Pipelines

Pipeline Events

While your pipeline is running, the pipeline generates events in the `sys.pipeline_events` system catalog table to mark significant checkpoints when something has occurred. In this example, you execute the `CREATE PIPELINE` and `START PIPELINE` SQL statements on the pipeline named `my_pipeline` and let the pipeline complete. As expected, the `CREATED`, `STARTED`, and `COMPLETED` events appear. For details about the lists of files, see the `sys.pipeline_files` system catalog table.

Pipeline events include messages from many different tasks. These events are the background processes that execute across different Loader Nodes during pipeline

operation. For details about tasks, you can query the `sys.tasks` and `sys.subtasks` system catalog tables.

SQL

```
SELECT *
FROM sys.pipeline_events
ORDER BY event_timestamp;
```

Output

SQL

```
+--------------------------------------+--------------------------------------+---
| pipeline_id                          | task_id                              | us
|--------------------------------------+--------------------------------------+---
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      | dc
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      | dc
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      |
| d8848467-9262-4bd6-84a7-1380c95f8b8b |                                      |
| d8848467-9262-4bd6-84a7-1380c95f8b8b | 61007c85-6744-42e7-9401-66e1a594e990 |
| d8848467-9262-4bd6-84a7-1380c95f8b8b | 61007c85-6744-42e7-9401-66e1a594e990 |
| d8848467-9262-4bd6-84a7-1380c95f8b8b | 49c17f79-0108-4b05-aae5-9ae9a13abaca |
+--------------------------------------+--------------------------------------+---
Fetched 7 rows
```

Pipeline Metrics

Find metrics that measure the performance and activity of a pipeline in the `sys.pipeline_metrics` system catalog table. This table contains one row for each metric at a specified point in time. A common use of this data is to plot metric values over time to examine the behavior of a pipeline.

Metric Types

Metrics are either instantaneous or incremental:

- Instantaneous metrics reflect the current value of a counter at the time of the metric collection. These values can increase or decrease based on the current state

or rate of the metric.

- Incremental metrics increment over time and reflect the cumulative value for a specified counter.

Example

The first snapshot of a metric appears when it is initialized. Afterward, the metric is updated every 10 seconds. The `updated_at` column shows the metric collection time.

For example, if you are interested in the number of transformed record bytes, you can execute this query. The value increases over time as the pipeline runs.

SQL

```
SELECT *
FROM sys.pipeline_metrics
WHERE name = 'count.record.bytes.transformed'
ORDER BY updated_at;
```

Output

SQL

```
+--------------------------------------+--------------------------------------+---
| pipeline_id                          | extractor_task_id                    | pa
|--------------------------------------+--------------------------------------+---
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |
| 0f51f1c0-f251-4407-8393-0f9bcbf18f28 | 034cf49d-364b-42b5-ad0a-37b7c237f16e |
+--------------------------------------+--------------------------------------+---
Fetched 5 rows
```

Metric Scope

Each metric has a scope that defines its uniqueness. Be careful not to aggregate metrics across different scopes.

The scope columns are `pipeline_id`, `extractor_task_id`, `partition_id`, and `sink_index`. If the value of any of these columns is NULL, the scope applies to all values in that dimension.

For example, because the `partition_id` and `sink_index` columns are NULL for the metric `count.record.bytes.transformed`, its scope applies to the pipeline indicated by the `pipeline_id` and `extractor_task_id` across all sinks and partitions.

An `extractor_task_id` is the internal identifier used when the Ocient System executes a pipeline process on a Loader Node. A pipeline might have many of these processes across time and loaders.

Example

Retrieve the most recent snapshot for each metric and aggregate the values to produce a single value for a pipeline.

SQL

```
SELECT pipeline_id, name, SUM(most_recent_value) AS total_value
FROM (
    SELECT DISTINCT * FROM (
        SELECT pipeline_id, extractor_task_id, partition_id, name,
        first_value (value) OVER
            (PARTITION BY pipeline_id, extractor_task_id, partition_id, name
             ORDER BY updated_at DESC) AS most_recent_value
        FROM sys.pipeline_metrics
        WHERE name LIKE 'count%'
    )
)
GROUP BY pipeline_id, name;
```

Output

SQL

```
+--------------------------------------+--------------------------------+---------
| pipeline_id                          | name                           |   total_
|--------------------------------------+--------------------------------+---------
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.file.total               |
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.record.bytes.transformed |      33
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.file.processed           |
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.record.sent              |        1
| 0c7b32f1-37ca-43f7-96b6-69c266c978a5 | count.record.durable           |        1
+--------------------------------------+--------------------------------+---------
Fetched 5 rows
```

This output contains a single row for each metric for each pipeline. If you run a pipeline with 64 files and 100,000 records, the output might look like this. This table explains each column in the output.

Related Links

System Catalog

Data Integrity and Storage

Query Analysis

Data Definition Language (DDL) Statement Reference

Cluster and Node Management

Users, Groups, and Service Classes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Connection Driver Reference

Ocient® includes multiple ways to connect and issue statements or queries to the database. The most common drivers used for communicating with Ocient are the JDBC driver and the pyocient module. Many third-party tools use the JDBC driver for business intelligence, visualization, and data exploration. These drivers can also be used in custom programs by leveraging the programming language's appropriate mechanisms to use the driver, for example, using a JDBC driver directly in Java®.

- **JDBC Manual**
  - **Data Extract Tool**
  - **Commands Supported by the Ocient JDBC CLI Program**
  - **JDBC Classes and Methods**
  - **JDBC Spark Connector**
  - **OCGraph Java Library**
- **Ocient Python Module: pyocient**
  - **OCGraph Python Library**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

JDBC Manual

The JDBC Driver and command-line interface (CLI) enable you to connect to Ocient using a JDBC connection. Ensure that you meet the prerequisites before using the Ocient JDBC Driver. Then, invoke the CLI program, configure options, and connect to a database using the driver.

You can also use the data extract tool to extract a result set to delimited files in the target location. For details about data extracting, see **Data Extract Tool**.

For a list of commands available in the JDBC CLI, see **Commands Supported by the Ocient JDBC CLI Program**.

Prerequisites

This software is required for the JDBC driver.

Software	Version
Ocient	Use the latest Ocient System version.
Operating System (OS)	, , or . Use the latest version of each OS.

See the **Version Compatibility**.

Driver Features

The Ocient JDBC connector supports these features as of the current version.

Unicode Support	UTF-8
Isolation Levels	Ocient does not support transactions at this time.
Data Types	Supports all Ocient **Data Types**.
Security / Encryption	Uses SSL/TLS to connect to the Ocient System. TLS protocol is available as a **JDBC Configuration Option**.

© 2026 Ocient, Inc.

Invoke the Ocient JDBC CLI Program

If your system meets all the necessary prerequisites, you can run the JDBC CLI by using the Ocient JDBC JAR file. To do this, follow these steps.

1. Go to the Ocient **repository** for all JDBC versions. For more information on which version to pick, see the **Version Compatibility** page.

2. For the JDBC version you want to use, download the JAR file with dependencies. This JAR file follows the format: `ocient-jdbc4-<version number>-jar-with-dependencies.jar`.

   Move this JAR file to the directory where your Ocient System is installed.

3. From the shell terminal, run this command to launch the JDBC CLI.

   Shell

   ```
   java -classpath <path_to_jar_with_dependencies> com.ocient.cli.CLI [<username>
   ```

   This example runs JDBC version 2.10.

   Shell

   ```
   java -classpath ./ocient-jdbc4-2.10-jar-with-dependencies.jar com.ocient.cli.C
   ```

4. After launching, the JDBC CLI prompts you to enter your username and password.

   Shell

   ```
   Username: admin@system
   Password: admin
   ```

   The interface changes to the Ocient CLI.

```
Shell

Ocient> _
```

5 Connect to your system from the JDBC using a connection string.

Assuming the standard port `4050`, a self-signed certificate, and the SQL Node IP address `10.10.1.1`, you can connect to the system database with the following connecting string.

```
Shell

Ocient> connect to jdbc:ocient://10.10.1.1:4050/system;
```

The CLI responds with a connection message.

```
Shell

Connected to jdbc:ocient://10.10.1.1:4050/system
Ocient> _
```

Now that you are connected to your system, you can execute any queries or commands.

For Java version 1.8.0_144, **download** and install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8.

JDBC CLI Configuration Options

The JDBC CLI reads a configuration file consisting of key-value pairs located at `~/.ocient-cli-configuration` with this format.

Shell

```
key1=value1
key2=value2
...
keyN=valueN
```

The JDBC CLI supports these options.

Option	Description	Default Value
`cliIdleTimeoutMinutes`	Configures the idle timeout for the CLI. The CLI rejects subsequent commands and exits the process after `cliIdleTimeoutMinutes` minutes of inactivity.	`0` if left unspecified or the configuration file does not exist.
`performance`	Enables different levels of query result output for measuring benchmarks by using different performance options. For details, see the **PERFORMANCE** command.	`off`
`printUuid`	Accepts values `on` or `off`. If you set the `printUuid` option to `on`, the system prints the query identifier for each query that you execute in the CLI.	`off`
`timing`	Enables or disables reporting the execution time of each query. For details, see the **TIMING** command.	`off`

For supported commands, see **Commands Supported by the Ocient JDBC CLI Program**.

Run a Transaction in the JDBC CLI

The Ocient JDBC CLI supports multi-statement transactions using SQL statements. To begin a transaction, execute the `SET AUTOCOMMIT OFF` SQL statement, and then end the transaction with the `COMMIT` or `ROLLBACK` statements. A query within the transaction reads uncommitted rows that the same connection has inserted. This example disables autocommit, inserts two rows into the `public.txn_demo` table, counts the rows in the table, and commits the transaction. Next, the example inserts two more rows in the table and counts the rows. Then, the example runs a `ROLLBACK` command to roll back the changes, and counts the rows again.

SQL

```
Ocient> SET AUTOCOMMIT OFF;
Ocient> INSERT INTO public.txn_demo VALUES (1, 'alpha');
Ocient> INSERT INTO public.txn_demo VALUES (2, 'beta');
Ocient> SELECT count(*) AS n_in_txn FROM public.txn_demo;
Ocient> COMMIT;

Ocient> INSERT INTO public.txn_demo VALUES (3, 'gamma');
Ocient> INSERT INTO public.txn_demo VALUES (4, 'delta');
Ocient> SELECT count(*) AS n_in_txn FROM public.txn_demo;
Ocient> ROLLBACK;
Ocient> SELECT count(*) FROM public.txn_demo;
```

Use the Ocient JDBC Driver in Java Programs

First, you must load the Ocient driver class with this statement in a Java program using the JDBC driver.

Java

```
Class.forName("com.ocient.jdbc.JDBCDriver");
```

The driver class is located in the JDBC driver JAR file named `ocient-jdbc4.jar` and must be available in the CLASSPATH defined for the program.

Connect Using JDBC

The Ocient JDBC driver supports connection properties that can be supplied using the **CONNECT** command in the JDBC CLI or as a properties object passed in a Java application.

JDBC URL Example

This connection string includes various connection properties that trail the login credentials.

Shell

```
CONNECT TO 'jdbc:ocient://db.example.com:6432/salesdb;user=admin@system;password=ad
```

Java DriverManager Example

Alternatively, you can use the DriverManager class to provide connection properties if you are using a JDBC connection as part of a Java application.

Java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class OcientConnectionExample {
    public static void main(String[] args) throws SQLException {
        // Base JDBC URL: host, port, and database
        String url = "jdbc:ocient://db.example.com:6432/salesdb";

        // JDBC connection properties
        Properties props = new Properties();
        props.setProperty("user", "analytics_user");            // Ocient username
        props.setProperty("password", "StrongPassw0rd!");       // Ocient password
        props.setProperty("logLevel", "INFO");                  // Driver logging v
        props.setProperty("networkTimeout", "15000");           // 15s network time
        props.setProperty("enableBulkLoad", "true");            // Use bulk load fo
        props.setProperty("bulkLoadThreshold", "50000");        // Min rows for bul
        try (Connection conn = DriverManager.getConnection(url, props)) {
        }
    }
}
```

Supported JDBC Connection Properties

The JDBC driver supports these connection parameters.

Parameter	Description
`bulkLoadChunkSize`	The number of rows to include in each JSON data file (chunk) uploaded. The default value is `60000`.

Parameter	Description
`bulkLoadCleanupOnFailure`	If you set this parameter value to `true`, the driver deletes the temporary files and pipelines even if the load fails. The default value is `true`.
`bulkLoadFailOnError`	If you set this parameter value to `true`, `executeBatch()` fails if any `INSERT` operation fails. If you set this parameter value to `false`, `executeBatch()` falls back to a standard multi-row `INSERT` operation if any INSERT operation fails. The default value is `false`.
`bulkLoadLoaderNodesCacheSeconds`	The number of seconds to cache the list of active Loader Nodes the system discovers during bulk load. When you set this parameter to a value greater than `0`, the driver caches the Loader Node list for the specified duration and forwards the same value as a server-side `CACHE_MAX_TIME` hint on the lookup query. This cache is -wide and is not specific to individual connections. Do not enable this parameter if your application connects to multiple Ocient clusters from the same JVM. The default value is `0` (disabled).
`bulkLoadMode`	The bulk load transport mode. Set this parameter to `ssh` to stage data using SSH/SFTP to Loader Nodes, or set it to `s3` to stage data in an S3-compatible object store. The default value is `ssh`.
`bulkLoadPollIntervalSeconds`	The number of seconds to wait between polling the `sys.pipelines` system catalog table for load status. The default value is `2`.
`bulkLoadS3AccessKeyId`	The access key identifier for authenticating to the S3 endpoint. This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3ApiCallTimeoutMs`	The timeout, in milliseconds, for individual S3 API calls. The default value is `120000` (two minutes).
`bulkLoadS3Bucket`	The S3 bucket name for staging bulk load data files. This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3EnablePathStyleAccess`	If you set this parameter to `true`, the driver uses path-style access for S3 requests (e.g., `http://endpoint/bucket/key` instead of `http://bucket.endpoint/key`). This parameter is required for S3-compatible services such as MinIO or Ceph. The default value is `true`.
`bulkLoadS3Endpoint`	The S3-compatible endpoint URL for bulk load staging (e.g., `https://s3.us-east-1.amazonaws.com`). This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3MultipartPartSize`	The size, in bytes, of each part in a multipart upload. The minimum value is `5242880` (5 MiB). The default value is `10485760` (10 MiB).
`bulkLoadS3MultipartThreshold`	The size, in bytes, above which the driver uses multipart upload instead of a single PUT request. The default value is `10485760` (10 MiB).
`bulkLoadS3Prefix`	The key prefix for staged objects in S3. The default value is `ocient-bulk-load/`.

Parameter	Description
`bulkLoadS3Region`	The AWS region for the S3 endpoint. The default value is `us-east-1`.
`bulkLoadS3SecretAccessKey`	The secret access key for authenticating to the S3 endpoint. This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3UploadConcurrency`	The number of data chunks to upload to S3 in parallel. Increasing this value can improve staging throughput for large batches. The default value is `1`.
`bulkLoadSshChannelTimeoutSeconds`	The number of seconds to wait for an SFTP channel to open on an established SSH connection during bulk load. If the channel does not open within this duration, the driver throws an error. The default value is `30`.
`bulkLoadSshConnectTimeoutSeconds`	The number of seconds to wait for the SSH connection and authentication to a Loader Node during bulk load. If the system does not establish a connection within this duration, the driver throws an error. The default value is `10`.
`bulkLoadSshKeyPath`	The absolute path to the password-less private SSH key file. The default value is `~/.ssh/id_rsa`.
`bulkLoadSshHostKeyVerification`	Controls how the JDBC driver verifies an SSH server host key for bulk load connections. Supported values are: `acceptAll` (default) — The driver accepts all host keys. The connector does not check the server host key against the `known_hosts` file and does not output warning logs. `strict` — The driver accepts only hosts with keys present in the `known_hosts` file (see the `bulkLoadSshKnownHostsPath` parameter). The driver rejects unknown hosts or hosts with changed keys. `acceptNew` — The driver accepts new hosts and adds their keys to the `known_hosts` file, but rejects hosts with keys changed from the data in `known_hosts`.
`bulkLoadSshKnownHostsPath`	The path to the `known_hosts` file, which contains SSH verification keys. The system uses this path only if the `bulkLoadSshHostKeyVerification` parameter is set to `strict` or `acceptNew`. The default path is `~/.ssh/known_hosts`.
`bulkLoadSshUser`	The SSH username to use when connecting to Loader Nodes.
`bulkLoadThreshold`	The minimum number of rows in a batch group to trigger a bulk load. The default value is `25000`.
`bulkLoadUseAtomicPipeline`	If you set this parameter to `true`, bulk load uses an atomic pipeline that creates, starts, and monitors the pipeline in a single blocking operation `(CREATE TRANSACTIONAL PIPELINE ... START FOREGROUND)`. This mode requires the Ocient System version 27.1 or later. Set this parameter to `false` to use the legacy create, start, monitor, and

Parameter	Description
	drop functionality when connecting to older servers. The default value is `true`.
`defaultSchema`	Default schema.
`enableBulkLoad`	Enables the high-speed bulk load feature. Set this parameter value to `true` to enable this feature. Otherwise, set this parameter to `false` to leave the feature disabled. The default value is `false`.
`force`	If set to true (case-sensitive), this parameter disables load-balancing for the connection.
`handshake`	Specifies the handshake protocol used for the connection. Supported options include: `"CBC", "GCM", "SSO"` `"GCM"` — (Galois/Counter Mode). This is the default encryption and is the recommended password encryption algorithm. `"CBC"` — (Cipher Block Chaining) for password encryption. `"SSO"` — Single Sign-On.
`identityprovider`	An SSO integration established in the database. For details, see **CREATE SSO INTEGRATION**.
`logfile`	The filename to use for JDBC tracing.
`loglevel`	If set to ERROR (warnings and errors only) or DEBUG (verbose tracing) and logfile is also set, JDBC tracing is enabled. This parameter is case-sensitive.
`longQueryThreshold`	Estimated query runtime in milliseconds before deeper query optimization runs. `0` — Use database server default. `-1` — Never run deeper optimization.
`maxRows`	Maximum allowed result set size in the number of rows.
`maxRowsPerInsertBatch`	The maximum number of rows to combine into a single multi-row `INSERT` SQL statement when executing a batch load. If the total number of rows in a batch group exceeds this limit, the driver splits the group into multiple sub-statements. The default value is 128,000 rows.
`maxTempDisk`	Maximum allowed temp disk usage as a percentage (0 - 100).
`maxThreadsPerResultSet`	The maximum number of threads the client uses to fetch rows from the server per Result Set, as defined in the **official JDBC documentation**. In this context, a thread represents a logical connection to the database. Applications that create and operate on Statement objects concurrently might find value in setting this parameter. Defaults to `0`, which effectively creates an unbounded Result Set thread pool.
`maxTime`	Maximum allowed runtime of a query in seconds before it is canceled on the server.

Parameter	Description
`networkTimeout`	Network connection timeout in milliseconds. If unspecified, this defaults to 10000 milliseconds.
`parallelism`	Limits a query to running on a specified number of cores on each CPU.
`password`	The password for the user.
`priority`	Query priority. This sets the priority for queries to run on the server side. If unspecified, this defaults to `1.0`.
`priorityAdjustFactor`	The default query priority adjustment value. For details, see **SET ADJUSTFACTOR**.
`priorityAdjustTime`	The default frequency to adjust the query priority. For details, see **SET ADJUSTTIME**.
`serviceClassName`	Specifies the name of the service class to use for the database session.
`sparkMode`	When you set this parameter to `true`, the driver enables -specific JDBC behavior intended to improve compatibility with the Ocient Spark connector. For details, see **JDBC Spark Connector**. This parameter defaults to `false` if you are connecting directly using the Ocient JDBC driver (e.g., connecting with the `DriverManager` class or CLI). Otherwise, if you are using the Spark connector (catalog or `.format("ocient")`), the default is `true`.
`ssoNumericAddress`	Specifies the SSO callback URL as `127.0.0.1`. When this value is `false`, the URL is `localhost`. The default value is `false`.
`ssoOAuthCodeCallbackPort`	If `ssoOAuthFlow=authorizationCode`, this parameter specifies the port that the SSO authorization uses. If unspecified, the default value is `7050`.
`ssoOAuthFlow`	This parameter is only applicable if SSO authorization is enabled. Forces the driver to use either the “authorizationCode” or “deviceGrant” flow to establish a Single Sign-On session. If this parameter is not provided, the System uses the “authorizationCode” flow when a web browser is available to the client and the “deviceGrant” flow when a web browser is not available to the client.
`ssoDebugMode`	When set to `true`, the system records additional log messages related to SSO.
`ssoOktaNativeTokenPath`	The path to the Native Token (AES-256GCM JWE). The path is relative to your home directory (i.e. ’~/’).
`ssoSslCallback`	Specifies whether to use HTTPS instead of HTTP for the SSO callback URL. The default value is `false`.
`ssoTimeoutSeconds`	The number of seconds before the SSO connection request times out. The default value is `60`.

Parameter	Description
`statementPooling`	ℹ This parameter has been removed as of Ocient JDBC version 2.104 and later. When set to `ON`, recently used statements are cached. Set to `OFF` when using a third-party application that implements statement pooling
`tls`	Can be set to `unverified` or `on`. Enables SSL/TLS encryption for the connection.
`timeoutMillis`	Number of milliseconds before cancellable operations are timed out and killed by the driver. 0 means no timeout. Default: 0
`user`	The identifier of the user.

JVM System Properties

The Ocient JDBC driver supports system properties that control driver-wide behavior. Unlike **connection properties**, JVM system properties apply globally to all connections within the JVM. To set JVM properties, use the `-D` flag from the `java` command line. The system reads these properties once when the driver initializes and cannot change them at runtime.

Example

This example launches the JDBC CLI with the memory throttle disabled. There is no space between `-D` and the property name.

Shell

```
java -Dcom.ocient.jdbc.rs.disable-memory-throttle=true \
  -classpath ocient-jdbc4-jar-with-dependencies.jar com.ocient.cli.CLI
```

Supported JVM System Properties

Property	Type	Default	Description
`com.ocient.jdbc.rs.disable-memory-throttle`	Boolean	`false`	Controls the heap-based result-set fetch throttle. When you set this property to `false` (default), the driver pauses new `FETCH_DATA` requests when the projected heap usage exceeds

Property	Type	Default	Description
			90% of `Runtime.maxMemory()`, waiting for memory to free before resuming. This action prevents the `OutOfMemoryError` error in most cases. When you set it to `true`, the driver does not throttle on heap usage. If memory is genuinely exhausted, the JVM throws the `OutOfMemoryError` error instead of the driver waiting.

JVM system properties are not part of the JDBC connection URL or the `Properties` object the system passes to `DriverManager.getConnection()`. You must set them at the `java` command line.

Set the `com.ocient.jdbc.rs.disable-memory-throttle` property to `true` only when you have verified that your application has sufficient heap headroom or when the throttle interferes with the expected workload behavior. Disabling the throttle can cause the `OutOfMemoryError` error if the JVM heap is exhausted.

JDBC Bulk Loading

For very large batches, the Ocient JDBC driver provides a high-speed bulk load path. When you enable bulk loading, the driver bypasses the standard multi-row insert operation and instead stages the batch data in the **JSON Lines** format and loads it through a temporary, system-generated Ocient pipeline. The driver supports two transport modes for staging data.

- **SSH/SFTP (default)** — The driver opens an SSH connection to a Loader Node and writes JSON data files directly to the node file system using SFTP.
- **S3** — The driver uploads JSON data files to an S3-compatible bucket. The Ocient System then reads the staged files from S3 through standard pipeline mechanics. This mode eliminates the requirement for SSH access between the client and the Loader Nodes.

Set the `bulkLoadMode` connection parameter to select the transport mode. The default value is `ssh`.

Configuration

Follow these steps to configure bulk loading in your JDBC driver.

Enable Bulk Loading

The JDBC driver disables bulk loading by default. To enable it, you must meet these conditions.

- Set the `enableBulkLoad` connection parameter to `true`.
- Use a parameterized `INSERT` statement that uses placeholders for values, e.g., `INSERT INTO customers (id, name, status) VALUES (?, ?, ?)`. For details, see **Set Up Parameterized Insert Statements**.
- The total number of rows in the batch group meets or exceeds the `bulkLoadThreshold` parameter.

> For recommended configuration settings for workloads, see **Bulk Loading Best Practices**.

Choose a Transport Mode

Set the `bulkLoadMode` connection parameter to select how the driver stages data for the pipeline. The supported values are `ssh` (default) and `s3`.

SSH Mode Configuration

SSH mode requires passwordless SSH access from the client to all Loader Nodes in the cluster. To point the driver to your private SSH key file, set the `bulkLoadSshKeyPath` connection parameter. The client application must be able to read this file.

The driver automatically discovers available Loader Nodes by querying the **sys.nodes** and **sys.service_roles** system catalog tables.

S3 Mode Configuration

The S3 mode stages data in an S3-compatible object store (such as S3 or) and does not require SSH access to the Loader Nodes. When you select the S3 mode, you must provide the following required connection parameters.

Parameter	Description
`bulkLoadS3Endpoint`	The S3-compatible endpoint URL (e.g., `https://s3.us-east-1.amazonaws.com`).
`bulkLoadS3Bucket`	The S3 bucket name for staging data files.
`bulkLoadS3AccessKeyId`	The access key identifier for authenticating to the S3 endpoint.
`bulkLoadS3SecretAccessKey`	The secret access key for authenticating to the S3 endpoint.

You can also set optional S3 parameters to control the region, key prefix, path-style access, multipart upload behavior, upload concurrency, and API call timeout. For the full list of S3 parameters and their defaults, see the **Supported JDBC Connection Properties** table.

The driver uploads JSON data chunks to the staging bucket, generates a `CREATE TRANSACTIONAL PIPELINE SOURCE S3` SQL statement that points the Ocient System to the staged objects, and monitors the pipeline to completion. After the pipeline finishes, the driver deletes the staged objects from S3 and drops the pipeline. On failure, this cleanup occurs only when you set the `bulkLoadCleanupOnFailure` parameter to `true` (the default).

> ⚠ The S3 access key identifier and secret access key are embedded in the pipeline DDL so that the Ocient System can read the staged objects. The pipeline is transactional and the driver drops it immediately after use. The Spark connector redacts credentials from its logs.

Data Type Mapping

The JDBC driver supports all standard scalar types.

For complex types, Java SQL STRUCT (`java.sql.Struct`) types load as Ocient TUPLE types, and Java ARRAY (`java.sql.Array`) types load as Ocient ARRAY types.

Set Up Parameterized `INSERT` Statements

These steps demonstrate how to use a single parameterized `INSERT` statement using the Java `PreparedStatement` class. The parameterized statement binds different values for each row.

1 Create the `PreparedStatement` object `ps` with `?` placeholders.

```java
String sql = "INSERT INTO customers (id, name, status) VALUES (?, ?, ?)";
PreparedStatement ps = conn.prepareStatement(sql);
```

2 Bind parameter values by position.

```java
ps.setLong(1, 123L);         // First ?
ps.setString(2, "Alice");    // Second ?
ps.setString(3, "ACTIVE");   // Third ?
```

3 Execute a single insert.

```java
ps.executeUpdate();
```

Or, add multiple rows as a batch.

© 2026 Ocient, Inc.

Java

```
// First row
ps.setLong(1, 123L);
ps.setString(2, "Alice");
ps.setString(3, "ACTIVE");
ps.addBatch();

// Second row
ps.setLong(1, 124L);
ps.setString(2, "Bob");
ps.setString(3, "INACTIVE");
ps.addBatch();

// Send them together
int[] results = ps.executeBatch();
```

4 Close the resources.

Java

```
ps.close();
```

Connection Encryption (SSL/TLS)

The JDBC driver can use SSL/TLS to connect to Ocient, causing all traffic to be encrypted. Specify the `tls` property on the connect statement to enable TLS support. The `tls` property supports these values.

unverified Traffic on the connection is encrypted, but no verification is done on the certificate received from the Ocient System.

on Traffic is encrypted, and the JDBC client must be able to verify the certificate received from the Ocient System.

The TLS `on` mode requires that the client knows the **Certificate Authority** that signed the certificate provided by the Ocient System. Typically, this mode requires either that

the certificate is signed by a well-known certificate authority, or the Certificate Authority certificate has been imported into the truststore of the Java system. The Java `keytool` utility is used to manipulate a Java truststore.

Secure Connections Using TLS discusses how you can configure user-defined certificates for the Ocient System.

Sample Java Program Using the Ocient JDBC Driver

This sample program demonstrates how to utilize the Ocient JDBC driver to establish a connection to a database, construct a prepared SQL statement, execute the query, and iterate through the result set.

Java

```
public class OcientJDBCExample {
     public static void main(final String args[]) {
          Class.forName("com.ocient.jdbc.JDBCDriver");
          Properties props = new Properties();
          props.setProperty("user", "username");
          props.setProperty("password", "pwd");
          props.setProperty("force", "true");
          String url = "jdbc:ocient://192.168.121.82:4050/db";
          Connection conn = DriverManager.getConnection(url, props);
          PreparedStatement pstmt = conn.prepareStatement(
                              "select l_orderkey from tpch.lineitem where l_linenum
          Pstmt.setInt(1, 4);
          ResultSet rs = pstmt.executeQuery();
          while(rs.next()){
               // do something with row
          }
          rs.close();
          pstmt.close();
          conn.close();
          return;
     }
}
```

For supported classes and methods, see **JDBC Classes and Methods**.

Run a Transaction Using the Ocient JDBC Driver

To run a multi-statement transaction with the Ocient JDBC driver in a Java program, disable the autocommit mode using `setAutoCommit(false)`, and then call the `commit()` or `rollback()` methods. For transactions, autocommit mode is on by default. Ensure that you execute SQL statements that are supported by transactions. Otherwise, the database throws an error. For a list of supported statements, see **Transactions**. This example code connects to a sample database and creates the `public.txn_demo` table. Then, the code inserts two rows and displays the row count in the table. The code commits the INSERT statements and re-runs the row count. This code inserts two more rows and displays a row count. Then, the example rolls back the transaction and displays a row count. Finally, the code enables the autocommit mode.

© 2026 Ocient, Inc.

Java

```java
import java.sql.*;
import java.util.Properties;

public class TxnApi {
  static void show(Connection c, String label) throws SQLException {
    try (Statement s = c.createStatement();
         ResultSet rs = s.executeQuery("SELECT count(*) AS n FROM public.txn_demo")) {
      rs.next();
      System.out.println(label + " -> row count = " + rs.getInt("n"));
    }
  }
  public static void main(String[] args) throws Exception {
    String url = "jdbc:ocient://db.example.com:4050/salesdb";
    Properties p = new Properties();
    p.setProperty("user", "admin@system");
    p.setProperty("password", "admin");
    p.setProperty("tls", "unverified");
    try (Connection c = DriverManager.getConnection(url, p)) {

      // DDL must run outside a transaction (autocommit on)
      try (Statement s = c.createStatement()) {
        s.execute("DROP TABLE IF EXISTS public.txn_demo");
        s.execute("CREATE TABLE public.txn_demo (id INT, v VARCHAR(64))");
      }

      // Transaction 1: multiple inserts and read-your-writes, then commit
      c.setAutoCommit(false);
      try (Statement s = c.createStatement()) {
        s.executeUpdate("INSERT INTO public.txn_demo VALUES (1, 'alpha')");
        s.executeUpdate("INSERT INTO public.txn_demo VALUES (2, 'beta')");
        show(c, "T1 in-txn (read-your-writes, expect 2)");
      }
      c.commit();
      show(c, "T1 after COMMIT (expect 2)");

      // Transaction 2: inserts, then rollback
      try (Statement s = c.createStatement()) {
        s.executeUpdate("INSERT INTO public.txn_demo VALUES (3, 'gamma')");
        s.executeUpdate("INSERT INTO public.txn_demo VALUES (4, 'delta')");
        show(c, "T2 in-txn (expect 4)");
      }
      c.rollback();
```

```
        show(c, "T2 after ROLLBACK (expect 2)");

        c.setAutoCommit(true);
      }
    }
}
```

Related Links

Connect Using JDBC

Data Extract Tool

Commands Supported by the Ocient JDBC CLI Program

JDBC Classes and Methods.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Data Extract Tool

The data extract tool is a part of the JDBC driver to unload data. You can execute the tool directly from the JDBC CLI. The tool extracts a result set to delimited or Apache® Parquet™ files in the target location. To invoke the JDBC CLI, see the **JDBC Manual**.

> To use the data extract tool, you must have JDBC version 2.63 or higher.

Supported Data Extract Formats

The data extract tool supports unloading result sets into files in specific formats. Supported extract formats are:

- CSV — Outputs result sets as text files with fields separated by a chosen delimiter.
- Parquet — Outputs result sets as Parquet files.

General Command Structure

Here is the general structure of an extract command.

```SQL
EXTRACT TO <location_type> [OPTIONS([param=value [,...]])] AS <query>
```

The command is case-insensitive. Each extract command must start with `EXTRACT TO`. The location type `location_type` must be either `LOCAL` for the local machine or `S3` for Amazon® Web Services℠ (AWS℠) S3. You can enclose additional options within a pair of parentheses following the keyword `OPTIONS`. The location type is required and there are required options for each location type. For the `LOCAL` location type, the options must define the file prefix `file_prefix`, and for `S3`, the options must define the file prefix `file_prefix`, bucket `bucket`, and endpoint `endpoint`. Next, the query follows the keyword `AS`.

This example is a simple general command structure.

© 2026 Ocient, Inc.

```sql
EXTRACT TO LOCAL OPTIONS(
  file_prefix="/home/user/out/data_",
  file_extension=".csv"
)
AS SELECT c1 FROM sys.dummy10;
```

For supported options, see **Data Extract Options**.

Specify Options, Quoting, and Escaping Quotes

Here is the general format of options.

```sql
key1 = value1, key2 = value2, ... , keyN = valueN
```

You need to follow certain guidelines when you specify options. Keys (option names) can only consist of alphanumeric characters and are unquoted. Values can be either quoted (with the reserved character `"`) or unquoted. If values are unquoted, they can only contain alphanumeric characters. If the value has a non-alphanumeric character, you must quote it with the reserved character `"`. Note that the single quote character does not work.

```sql
OPTIONS(file_prefix = "/path/to/dir/result", header_mode = none, file_extension = "
```

To use the reserved quote character `"` as an argument, you must escape it with the backslash character `\`. To use `\` as an argument, you must escape it with another `\`. This code illustrates both of these scenarios.

```sql
OPTIONS(field_optionally_enclosed_by = "\"", escape = "\\")
```

Data Extract Options

Use these optional options with the `EXTRACT TO` syntax and the `OPTIONS` keyword to configure the behavior of the extract.

General Extract Options

This table describes optional options that apply to both the `LOCAL` and `S3` location types.

Option	Description	Default
`FILE_TYPE`	The type of the output file for extraction. Supports extraction to a delimited `.csv` file or Parquet `.parquet` file.	`DELIMITED`
`FILE_PREFIX`	Dictates the prefix used on the results. When extracting to `LOCAL`, this is the prefix used to determine the path of the results. This value can be a relative or full path. When extracting to S3, this value is the prefix for the key. In either case, the system adds additional file numbers and file extensions to generate the complete filename.	`results-`
`FILE_PREFIX_EXISTS`	Determines the behavior if the path specified by the `FILE_PREFIX` option already exists. Supported values are: `FAIL` and `OVERWRITE`. The `FAIL` value throws an error, whereas `OVERWRITE` deletes the contents of the path.	`'FAIL'`
`FILE_EXTENSION`	The file extension specified for each output file.	`.csv`
`MAX_ROWS_PER_FILE`	If you set this option to a non-zero value, the system splits the results into files with the specified maximum number of rows per file.	`NULL`
`COMPRESSION`	Compression type to use for a delimited extract. Supported compression types are: `NONE` — No compression `GZIP` — GZip compression `BZIP2` — bzip2 compression `XZ` — xz compression	`NONE`
`RECORD_DELIMITER`	Delimiter to use between records. This supports Java® strings, so special characters can be input using escape characters. UTF-16: `\u[utf-16 value]` or Octal `\[octal value]`.	`\n`
`FIELD_DELIMITER`	Delimiter to use between fields within a record. This supports Java strings, so special characters	`,`

Option	Description	Default
	can be input using escape characters. UTF-16: `\u[utf-16 value]` or Octal `\[octal value]`.	
`HEADER_MODE`	Dictates how to manage headers in result files. Supported values are `NONE`, `ALL_FILES`, and `FIRST_FILE`. `NONE` — The tool writes all output files without an additional header. `ALL_FILES` — The tool adds column names as a header in the first row of each output file. Each file has at most `MAX_ROWS_PER_FILE` + 1 total rows. `FIRST_FILE` — The tool adds column names as a header in the first row of the first output file. The tool does not add the header to subsequent files. Each file has at most `MAX_ROWS_PER_FILE` total rows, inclusive of the header in the first file.	`NONE`
`NULL_FORMAT`	Format string to use for writing NULL values to the output files.	`""` (empty string)
`ENCODING`	Encoding used when writing out data to files.	The default character set of the system, as determined by the **Oracle documentation**.
`ESCAPE`	Character used for escaping quoted fields. Set this to the NULL character `\0` to indicate that the escape character is not specified.	`"`
`FIELD_OPTIONALLY_ENCLOSED_BY`	Sometimes, you need to surround fields in a character. For example, the field might have a literal comma. Generally, this character is also known as the quote character. Set this option to the NULL character `\0` to indicate that the quote character is not specified.	`"`
`BINARY_FORMAT`	The format with which to encode the BINARY data type. Supports `UTF-8`, `Hexadecimal`, and `Base64`.	`Hexadecimal`
`COMPRESSION_BLOCK_SIZE`	The number of bytes that comprise each block to be compressed; larger blocks result in better compression at the expense of more RAM usage when compressing.	`4194304`
`COMPRESSION_LEVEL`	An integer value [-1, 9]. Use `-1` for the GZip default compression level, `0` for no compression, or a value [1-9] where 1 indicates fastest compression and 9 indicates best compression.	`1`

Option	Description	Default
`NUM_COMPRESSION_THREADS`	The number of threads to use for compression.	$(number of cores * 2)
`NUM_FETCH_QUERIES`	The number of parallel queries to execute in the database for data extraction.	`15`
`ESCAPE_UNQUOTED_VALUES`	Dictates whether to write escape sequences in unquoted values. Only applicable when `FIELD_DELIMITER` is set to `,`.	`false`
`INPUT_ESCAPED`	Dictates whether the input is already escaped. When this option is set to true, the tool does not add escape sequences, and data is written without changes to the output file. Only applicable when `FIELD_DELIMITER` is set to `,`. Ensure that data is properly escaped, otherwise the extract might produce invalid CSV data.	`false`
`PARTITION_MODE`	The strategy for partitioning the data. Supported values are: `NONE`, `KEY`, and `RANGE`. When you set this option to `NONE`, the tool uses standard extraction. When you set this option to `KEY`, the tool creates subdirectories for each unique value specified by the `PARTITION_COLUMNS` option. When you set this option to `RANGE`, the tool splits the data into the number of queries specified by the `NUM_FETCH_QUERIES` option based on the range of values specified in the `PARTITION_COLUMNS` option.	`NONE`
`PARTITION_COLUMNS`	The comma-separated list of columns to use for partitioning data when you set the `PARTITION_MODE` option to `KEY` or `RANGE`. The `RANGE` value only allows a single column. See **File Naming Conventions** for the file path structure for multiple partitioning columns when using the `KEY` value.	`NULL`
`QUOTE_ALL_FIELDS`	Dictates whether all written fields are enclosed with quotes. When this option is set to true, the tool encloses all fields with the `FIELD_OPTIONALLY_ENCLOSED_BY` character.	`false`
`SUCCESS_MARKER`	Identifies a successful completion of the extract. If you set this option to `true`, the tool creates a file with the `_SUCCESS` suffix in the root output directory when the extract of the entire job completes successfully.	`true`
`TARGET_FILE_SIZE_MB`	Specifies the size in megabytes for the target output file. The data extract tool splits the output	`NULL`

Option	Description	Default
	into files of approximately this size. The tool ignores this option if you set the `MAX_ROWS_PER_FILE` option.	
`TRANSLATE_CHARACTERS_MODE`	Character Mode to use for translating characters. Supported values are `CHAR` and `HEX`. The tool performs character translation only if you specify `TRANSLATE_CHARACTERS_FROM` and `TRANSLATE_CHARACTERS_TO`. The tool replaces the Nth character in `TRANSLATE_CHARACTERS_FROM` with the Nth character in `TRANSLATE_CHARACTERS_TO` in the extracted records. When `TRANSLATE_CHARACTERS_MODE` is set to CHAR, `TRANSLATE_CHARACTERS_FROM`, and `TRANSLATE_CHARACTERS_TO` must be equal-length strings of UTF-8 characters. For example: `TRANSLATE_CHARACTERS_MODE="CHAR"`, `TRANSLATE_CHARACTERS_FROM="àëï"`, `TRANSLATE_CHARACTERS_TO="aei"` When `TRANSLATE_CHARACTERS_MODE` is set to `HEX`, `TRANSLATE_CHARACTERS_FROM`, and `TRANSLATE_CHARACTERS_TO` must be comma-separated lists of hexadecimal UTF-8 code points with the same number of list elements. For example: `TRANSLATE_CHARACTERS_MODE="HEX"`, `TRANSLATE_CHARACTERS_FROM="c3a0,c3ab,c3af"`, `TRANSLATE_CHARACTERS_TO="61,65,69"`	`CHAR`
`TRANSLATE_CHARACTERS_FROM`	Sequence of UTF-8 characters in the source data to translate to a corresponding character in the `TRANSLATE_CHARACTERS_TO` option. See the `TRANSLATE_CHARACTERS_MODE` option for the expected format.	`""`
`TRANSLATE_CHARACTERS_TO`	Sequence of UTF-8 characters to use as a replacement for the characters included in `TRANSLATE_CHARACTERS_FROM`. See the `TRANSLATE_CHARACTERS_MODE` option for the expected format.	`""`
`TRIM_TRAILING_ZEROS`	Dictates whether to trim trailing zeros from numeric input fields.	`false`
`PARQUET_COMPRESSION`	Compression type to use for a Parquet extract. Supported compression types are: `NONE` — No compression `ZSTD` — ZSTD compression `SNAPPY` — Snappy compression `GZIP` — GZip compression	`SNAPPY`

Option	Description	Default
`PARQUET_ROW_GROUP_SIZE_BYTES`	The size in bytes for row groups within a Parquet output file.	`536870912` (512 MB)

> If you do not set either the `MAX_ROWS_PER_FILE` or the `TARGET_FILE_SIZE_MB` options, the data extract tool generates one output file for each partition. For a query without partitions, the tool generates a single output file.

S3 Extract Options

This table describes the required options that apply only to the `S3` location type. The data extract tool ignores these options when you use the `LOCAL` location type.

Option	Description
`BUCKET`	S3 bucket to use.
`ENDPOINT`	Endpoint for S3 upload. For details, see the Amazon® documentation for **specifying endpoints**.

This table describes optional options that apply only to the `S3` location type.

Option	Description	Default
`AWS_KEY_ID`	AWS Key ID. If empty, the CLI uses the Java AWS SDK default credentials provider chain documented **here**.	`""`
`AWS_SECRET_KEY`	AWS Secret Key. If empty, the CLI uses the Java AWS SDK default credentials provider chain documented **here**.	`""`
`REGION`	S3 region to upload to. Ignored when extracting to LOCAL.	`US_EAST_2`
`PATH_STYLE_ACCESS`	Whether **path style access** should be used to access a bucket.	`true` for JDBC version 3.4.1 and `false` for JDBC version 3.4.0 and earlier

File Naming Conventions

When you use the data extract tool, the tool produces a number of files. The Ocient® System determines the path of these files using multiple factors:

- File prefix as specified by the `FILE_PREFIX` option
  - If the `PARTITION_MODE` option is not set to `NONE`, the file prefix must be a directory and end with a forward slash. The system creates subdirectories in the Apache® Hive™ style for each partition (for example, for the `KEY` partition: `/tmp/extract/col1=val1/col2=val2` or for the `RANGE` partition: `/tmp/extract/val1<=col1<val2/`). If the `PARTITION_MODE` option is set to `NONE`, the file prefix can be a directory or a proper prefix (for example, `/tmp/extract/results_`).
- Partition mode as specified by the `PARTITION_MODE` option
- Maximum number of rows, or target size in MB, for each file as specified by the `MAX_ROWS_PER_FILE` option
  - If you do not set a maximum number of rows per file and no target file size, this part of the convention is `0`. However, if the `MAX_ROWS_PER_FILE` or `TARGET_FILE_SIZE_MB` options are set, then the system outputs rows to a single file until that limit or the lower of the two limits is reached. Then, the system generates another file with an incremented file number. The file number starts from `0`.
- Format or file extension
  - This convention applies only if the `FILE_TYPE` option is set to `DELIMITED`. For the `PARQUET` type, the file extension is always `.parquet`, even when you specify any type of compression using the `PARQUET_COMPRESSION` option.
- Compression extension
  - If you specify the `COMPRESSION` option for an extract with the `DELIMITED` type, then the tool adds the compression suffix.

Examples

These examples show you how to use the data extract tool for extracting data to the local machine, extracting CSV data, and extracting Parquet data. Most of these examples assume the database already contains tables with data. For loading data, see **Load Data**.

Local Extract to the Default Path

This example extracts the results of the `SELECT c1 FROM sys.dummy10` SQL query to the local machine at the relative path `result-0.csv`. The SQL query returns 10 rows with incremental numbers starting at 1. The `OPTIONS(...)` syntax is only required when you specify additional options.

SQL

```
EXTRACT TO LOCAL
AS SELECT c1 FROM sys.dummy10;
```

`sys.dummy` creates a virtual table with the specified number for rows. For details, see **Generate Tables Using sys.dummy**.

Local CSV Extract to the Specified Path

This example extracts the results of the `SELECT c1 FROM sys.dummy10` SQL query to the local machine at the absolute path `/home/user/out/data_0.csv`. The SQL query returns 10 rows with incremental numbers starting at 1.

SQL

```
EXTRACT TO LOCAL OPTIONS(
  file_prefix="/home/user/out/data_",
  file_extension=".csv"
)
AS SELECT c1 FROM sys.dummy10;
```

Local Parquet Extract with Partition by Key

This example partitions customer data into a Parquet file by the `country` column. This partitioning creates a Hive-style directory structure on the local machine. Use the file path `/home/user/customer_data/` with the `KEY` value for the partition strategy. Specify the target file size to be `256` MB.

The SQL query selects the identifier, name, email, and country from the `customers` table.

SQL

```
EXTRACT TO LOCAL OPTIONS(
    FILE_PREFIX = '/home/user/customer_data/',
    FILE_TYPE = PARQUET,
    PARTITION_MODE = 'KEY',
    PARTITION_COLUMNS = 'country',
    TARGET_FILE_SIZE_MB = 256
) AS SELECT id, name, email, country FROM customers;
```

The resulting file structure has these file paths.

```
/home/user/customer_data/country=US/part-0000.parquet
/home/user/customer_data/country=CA/part-0000.parquet
...
```

S3 Parquet Extract with Compression

This example extracts all sales data from the `today_sales` table to a single Parquet file on S3 using ZSTD compression. Use the S3 bucket named `my-analytics-bucket` with the file path `daily_reports/report.parquet` using the endpoint `s3.us-east-1.amazonaws.com`.

SQL

```
EXTRACT TO S3 OPTIONS(
    BUCKET = 'my-analytics-bucket',
    FILE_PREFIX = 'daily_reports/report.parquet',
    ENDPOINT = 's3.us-east-1.amazonaws.com',
    FILE_TYPE = PARQUET,
    PARQUET_COMPRESSION = 'ZSTD'
) AS SELECT * FROM today_sales;
```

S3 Parquet Extract with Partitioning by Range

This example extracts large event data to S3 by splitting the `event_id` column into `30` parallel queries using the `RANGE` partition strategy. Specify the maximum file size in `1024` megabytes. Override the default row group size using the `PARQUET_ROW_GROUP_SIZE_BYTES` option to specify `268435456` bytes (256 MB).

© 2026 Ocient, Inc.

Use the S3 bucket `iot-data` with the file path `events/2025-09-25/` at endpoint `s3.us-east-1.amazonaws.com`.

The SQL query selects the event identifier, sensor identifier, sensor reading, and timestamp from the `events` table.

SQL

```
EXTRACT TO S3 OPTIONS(
    BUCKET = 'iot-data',
    FILE_PREFIX = 'events/2025-09-25/',
    ENDPOINT = 's3.us-east-1.amazonaws.com',
    FILE_TYPE = PARQUET,
    PARTITION_MODE = 'RANGE',
    PARTITION_COLUMNS = 'event_id',
    NUM_FETCH_QUERIES = 30,
    TARGET_FILE_SIZE_MB = 1024,
    PARQUET_ROW_GROUP_SIZE_BYTES = 268435456
) AS SELECT event_id, sensor_id, sensor_reading, ts FROM events;
```

Related Links

Connect Using JDBC

JDBC Manual

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Commands Supported by the Ocient JDBC CLI Program

Commands are not case-sensitive in the JDBC CLI program.

For DDL statements, see **Data Definition Language (DDL) Statement Reference**.

> All commands, except QUIT, must end with a semicolon at the command line.

> You can cancel any query that does not update data by pressing CTRL+C.

The Ocient JDBC driver supports multi-statement transactions. By default, a connection uses the autocommit mode, in which the database commits each statement immediately. To group several statements into a single transaction, disable autocommit with **SET AUTOCOMMIT**, and then end the transaction with **COMMIT** or **ROLLBACK**.

> The HTTP Query API does not support transactions. To use transactions, connect to the Ocient System with the JDBC driver or the pyocient module.

> The Connector supports transactions and performs the commit and rollback actions automatically. When you use the Spark Connector, you do not need to issue COMMIT or ROLLBACK commands.

CANCEL

Cancels the current query that is still running on the system.

Syntax

Shell

```
CANCEL uuid;
```

© 2026 Ocient, Inc.

Parameter	Data Type	Description
uuid	String	A query Universally Unique IDentifier (UUID) that is currently running. Use **LIST ALL QUERIES**, or search the `sys.queries` table to find the UUID. You must enclose the UUID in single quotes, such as `'c87b506f-66d6-4987-8bf7-1fd4a06b64b1'`.

COMMIT

Commits the current transaction and permanently saves all changes made after the transaction started. This command applies only when you disable autocommit with **SET AUTOCOMMIT**. When autocommit is enabled, COMMIT has no effect.

Syntax

Shell

```
COMMIT [ TRANSACTION ];
```

Entering the COMMIT command with no other arguments commits the current transaction on the connection. This action is equivalent to calling the `commit()` method on the JDBC Connection object.

CONNECT

Connects to a SQL Node. For details on the connection steps, see **Connect Using JDBC**.

Syntax

Shell

```
CONNECT TO <url> [ USER username USING password ]
```

ⓘ The `<url>` parameter must follow this format:

`jdbc:ocient://node_address1:port1[,node_address2:port2[,...]]/database_name[;property=value[;...]]`

If you do not specify the `username` and `password`, then the database uses the default credentials specified at startup.

JDBC URL parameters

Parameter	Description
`node_address`	A hostname or IP address. The database attempts to connect using each `DNSName:Port` pair in the comma-delimited list from left to right until a successful connection happens.
`port`	Standard JDBC port number is `4050`. For details about ports, see **Ports**.
`database_name`	The identifier used for your database.

Additional JDBC URL Parameters

The JDBC URL can include these optional parameters in the key-value format `property=value`. Separate each parameter pair by semicolons.

URL parameters are case-insensitive.

To authenticate using Single Sign-On, set the `handshake` parameter to `SSO` and leave the `user` and `password` parameters empty. For example:
`jdbc:ocient://DNSName:Port[,IP2:Port2,...]/databaseName;handshake=SSO;user=;password=;`

Parameter	Description
`bulkLoadChunkSize`	The number of rows to include in each JSON data file (chunk) uploaded. The default value is `60000`.
`bulkLoadCleanupOnFailure`	If you set this parameter value to `true`, the driver deletes the temporary files and pipelines even if the load fails. The default value is `true`.
`bulkLoadFailOnError`	If you set this parameter value to `true`, `executeBatch()` fails if any `INSERT` operation fails. If you set this parameter value to `false`, `executeBatch()` falls back to a standard multi-row `INSERT` operation if any INSERT operation fails. The default value is `false`.
`bulkLoadLoaderNodesCacheSeconds`	The number of seconds to cache the list of active Loader Nodes the system discovers during bulk load. When you set this parameter to a value greater than `0`, the driver caches the Loader Node list for the specified duration and forwards the same value as a server-side `CACHE_MAX_TIME` hint on the lookup

Parameter	Description
	query. This cache is -wide and is not specific to individual connections. Do not enable this parameter if your application connects to multiple Ocient clusters from the same JVM. The default value is `0` (disabled).
`bulkLoadMode`	The bulk load transport mode. Set this parameter to `ssh` to stage data using SSH/SFTP to Loader Nodes, or set it to `s3` to stage data in an S3-compatible object store. The default value is `ssh`.
`bulkLoadPollIntervalSeconds`	The number of seconds to wait between polling the `sys.pipelines` system catalog table for load status. The default value is `2`.
`bulkLoadS3AccessKeyId`	The access key identifier for authenticating to the S3 endpoint. This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3ApiCallTimeoutMs`	The timeout, in milliseconds, for individual S3 API calls. The default value is `120000` (two minutes).
`bulkLoadS3Bucket`	The S3 bucket name for staging bulk load data files. This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3EnablePathStyleAccess`	If you set this parameter to `true`, the driver uses path-style access for S3 requests (e.g., `http://endpoint/bucket/key` instead of `http://bucket.endpoint/key`). This parameter is required for S3-compatible services such as MinIO or Ceph. The default value is `true`.
`bulkLoadS3Endpoint`	The S3-compatible endpoint URL for bulk load staging (e.g., `https://s3.us-east-1.amazonaws.com`). This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3MultipartPartSize`	The size, in bytes, of each part in a multipart upload. The minimum value is `5242880` (5 MiB). The default value is `10485760` (10 MiB).
`bulkLoadS3MultipartThreshold`	The size, in bytes, above which the driver uses multipart upload instead of a single PUT request. The default value is `10485760` (10 MiB).
`bulkLoadS3Prefix`	The key prefix for staged objects in S3. The default value is `ocient-bulk-load/`.
`bulkLoadS3Region`	The AWS region for the S3 endpoint. The default value is `us-east-1`.
`bulkLoadS3SecretAccessKey`	The secret access key for authenticating to the S3 endpoint. This parameter is required when you set the `bulkLoadMode` parameter to `s3`.
`bulkLoadS3UploadConcurrency`	The number of data chunks to upload to S3 in parallel. Increasing this value can improve staging throughput for large batches. The default value is `1`.
`bulkLoadSshChannelTimeoutSeconds`	The number of seconds to wait for an SFTP channel to open on an established SSH connection during bulk load. If the channel does not open within this duration, the driver throws an error. The default value is `30`.
`bulkLoadSshConnectTimeoutSeconds`	The number of seconds to wait for the SSH connection and authentication to a Loader Node during bulk load. If the system does not establish a connection within this duration, the driver throws an error. The default value is `10`.

Parameter	Description
`bulkLoadSshKeyPath`	The absolute path to the password-less private SSH key file. The default value is `~/.ssh/id_rsa`.
`bulkLoadSshHostKeyVerification`	Controls how the JDBC driver verifies an SSH server host key for bulk load connections. Supported values are: `acceptAll` (default) — The driver accepts all host keys. The connector does not check the server host key against the `known_hosts` file and does not output warning logs. `strict` — The driver accepts only hosts with keys present in the `known_hosts` file (see the `bulkLoadSshKnownHostsPath` parameter). The driver rejects unknown hosts or hosts with changed keys. `acceptNew` — The driver accepts new hosts and adds their keys to the `known_hosts` file, but rejects hosts with keys changed from the data in `known_hosts`.
`bulkLoadSshKnownHostsPath`	The path to the `known_hosts` file, which contains SSH verification keys. The system uses this path only if the `bulkLoadSshHostKeyVerification` parameter is set to `strict` or `acceptNew`. The default path is `~/.ssh/known_hosts`.
`bulkLoadSshUser`	The SSH username to use when connecting to Loader Nodes.
`bulkLoadThreshold`	The minimum number of rows in a batch group to trigger a bulk load. The default value is `25000`.
`bulkLoadUseAtomicPipeline`	If you set this parameter to `true`, bulk load uses an atomic pipeline that creates, starts, and monitors the pipeline in a single blocking operation `(CREATE TRANSACTIONAL PIPELINE ... START FOREGROUND)`. This mode requires the Ocient System version 27.1 or later. Set this parameter to `false` to use the legacy create, start, monitor, and drop functionality when connecting to older servers. The default value is `true`.
`defaultSchema`	Default schema.
`enableBulkLoad`	Enables the high-speed bulk load feature. Set this parameter value to `true` to enable this feature. Otherwise, set this parameter to `false` to leave the feature disabled. The default value is `false`.
`force`	If set to true (case-sensitive), this parameter disables load-balancing for the connection.
`handshake`	Specifies the handshake protocol used for the connection. Supported options include: `"CBC", "GCM", "SSO"` `"GCM"` — (Galois/Counter Mode). This is the default encryption and is the recommended password encryption algorithm.

Parameter	Description
	`"CBC"` — (Cipher Block Chaining) for password encryption. `"SSO"` — Single Sign-On.
`identityprovider`	An SSO integration established in the database. For details, see **CREATE SSO INTEGRATION**.
`logfile`	The filename to use for JDBC tracing.
`loglevel`	If set to ERROR (warnings and errors only) or DEBUG (verbose tracing) and logfile is also set, JDBC tracing is enabled. This parameter is case-sensitive.
`longQueryThreshold`	Estimated query runtime in milliseconds before deeper query optimization runs. `0` — Use database server default. `-1` — Never run deeper optimization.
`maxRows`	Maximum allowed result set size in the number of rows.
`maxRowsPerInsertBatch`	The maximum number of rows to combine into a single multi-row `INSERT` SQL statement when executing a batch load. If the total number of rows in a batch group exceeds this limit, the driver splits the group into multiple sub-statements. The default value is 128,000 rows.
`maxTempDisk`	Maximum allowed temp disk usage as a percentage (0 - 100).
`maxThreadsPerResultSet`	The maximum number of threads the client uses to fetch rows from the server per Result Set, as defined in the **official JDBC documentation**. In this context, a thread represents a logical connection to the database. Applications that create and operate on Statement objects concurrently might find value in setting this parameter. Defaults to `0`, which effectively creates an unbounded Result Set thread pool.
`maxTime`	Maximum allowed runtime of a query in seconds before it is canceled on the server.
`networkTimeout`	Network connection timeout in milliseconds. If unspecified, this defaults to 10000 milliseconds.
`parallelism`	Limits a query to running on a specified number of cores on each CPU.
`password`	The password for the user.
`priority`	Query priority. This sets the priority for queries to run on the server side. If unspecified, this defaults to `1.0`.
`priorityAdjustFactor`	The default query priority adjustment value. For details, see **SET ADJUSTFACTOR**.
`priorityAdjustTime`	The default frequency to adjust the query priority. For details, see **SET ADJUSTTIME**.

Parameter	Description
`serviceClassName`	Specifies the name of the service class to use for the database session.
`sparkMode`	When you set this parameter to `true`, the driver enables -specific JDBC behavior intended to improve compatibility with the Ocient Spark connector. For details, see **JDBC Spark Connector**. This parameter defaults to `false` if you are connecting directly using the Ocient JDBC driver (e.g., connecting with the `DriverManager` class or CLI). Otherwise, if you are using the Spark connector (catalog or `.format("ocient")`), the default is `true`.
`ssoNumericAddress`	Specifies the SSO callback URL as `127.0.0.1`. When this value is `false`, the URL is `localhost`. The default value is `false`.
`ssoOAuthCodeCallbackPort`	If `ssoOAuthFlow=authorizationCode`, this parameter specifies the port that the SSO authorization uses. If unspecified, the default value is `7050`.
`ssoOAuthFlow`	This parameter is only applicable if SSO authorization is enabled. Forces the driver to use either the “authorizationCode” or “deviceGrant” flow to establish a Single Sign-On session. If this parameter is not provided, the System uses the “authorizationCode” flow when a web browser is available to the client and the “deviceGrant” flow when a web browser is not available to the client.
`ssoDebugMode`	When set to `true`, the system records additional log messages related to SSO.
`ssoOktaNativeTokenPath`	The path to the Native Token (AES-256GCM JWE). The path is relative to your home directory (i.e. ’~/’).
`ssoSslCallback`	Specifies whether to use HTTPS instead of HTTP for the SSO callback URL. The default value is `false`.
`ssoTimeoutSeconds`	The number of seconds before the SSO connection request times out. The default value is `60`.
`statementPooling`	**i** This parameter has been removed as of Ocient JDBC version 2.104 and later. When set to `ON`, recently used statements are cached. Set to `OFF` when using a third-party application that implements statement pooling
`tls`	Can be set to `unverified` or `on`. Enables SSL/TLS encryption for the connection.
`timeoutMillis`	Number of milliseconds before cancellable operations are timed out and killed by the driver. 0 means no timeout. Default: 0
`user`	The identifier of the user.

The JDBC driver also supports system properties that apply globally across all connections. To set these properties, use the `-D` flag at JVM startup. For details, see **JVM System Properties**.

DESCRIBE TABLE

Lists the columns of the specified table and its associated data types.

Syntax

```
Shell
DESCRIBE TABLE [schema.]table [verbose]
```

Parameter	Description
`table`	A table identifier.
`schema`	A qualifying schema name for the specified table. If a qualifying schema is not specified for the table, the default name schema is assumed.
`verbose`	If specified, the Ocient System prints the full column metadata for the specified table as returned by GetColumns.

DESCRIBE VIEW

Returns the query text used to create the specified view.

Syntax

```
Shell
DESCRIBE VIEW [schema.]view [verbose]
```

Parameter	Description
`view`	A view identifier.

Parameter	Description
`schema`	A qualifying schema name for the specified table. If a qualifying schema is not specified for the table, the default name schema is assumed.
`verbose`	If specified, the Ocient System prints the full column metadata for the specified table as returned by GetColumns.

EXTRACT

Extracts a result set in delimited files to a specified location. For details about using this command, see **Data Extract Tool**.

Syntax

SQL

```
EXTRACT TO location
   { LOCAL | S3 }
   [ OPTIONS ( [ param=value [ ,... ] ] ) ] AS query
```

GET JDBC VERSION

Returns the JDBC version of the driver.

Syntax

Shell

```
GET JDBC VERSION
```

GET SCHEMA

Retrieves the default name schema for tables.

Syntax

Shell

```
GET SCHEMA
```

KILL

Issues a hard kill command to the virtual machine to terminate a query running on the system. Use only if `CANCEL` fails.

Syntax

Shell

```
KILL uuid
```

Parameter	Description
`uuid`	A query UUID that is currently running. Use **LIST ALL QUERIES**, or search the `sys.queries` table to find the UUID.

LIST ALL QUERIES

List all the currently executing queries on the database. This is the equivalent of executing `SELECT * FROM sys.queries`.

Syntax

Shell

```
LIST ALL QUERIES
```

LIST INDEXES

Lists the indexes on the specified table and its columns.

Syntax

```
Shell
LIST INDEXES [schema.]table [verbose]
```

Parameter	Description
`table`	A table identifier.
`schema`	A qualifying schema name for the specified table. If a qualifying schema is not specified for the table, the default name schema is assumed.
`verbose`	If specified, the Ocient System prints the full column metadata for the specified table as returned by getIndexInfo.

LIST SYSTEM TABLES

Lists all system catalog tables in the database.

Syntax

```
Shell
LIST SYSTEM TABLES [verbose]
```

Parameter	Description
`verbose`	If verbose is specified, the full table metadata, as returned by getTables, is printed.

LIST TABLES

Lists all tables (in all schemata) in the database.

Syntax

```
Shell
LIST TABLES [verbose]
```

Parameter	Description
`verbose`	If verbose is specified, the full table metadata, as returned by getTables, is printed.

LIST VIEWS

Lists all views (in all schemata) in the database.

Syntax

Shell

```
LIST VIEWS [verbose]
```

Parameter	Description
`verbose`	If verbose is specified, the full table metadata, as returned by getViews, is printed.

OUTPUT GIS KML

Outputs the next query in KML format to the specified filename. This flag affects only the next query.

This syntax prints out all GIS types to the KML file, and non-GIS types are added to the description of elements in the same row. You can upload this KML file to certain visualization tools.

Syntax

Shell

```
OUTPUT GIS KML <filename>
```

Parameter	Description
`filename`	A filename for the KML output.

OUTPUT NEXT QUERY

Output the next query in CSV format to the specified filename. This flag only affects the next query. All subsequent queries print out their results.

Add `APPEND` to add output to the specified file instead of overwriting it.

Syntax

Shell

```
OUTPUT NEXT QUERY filename [APPEND]
```

Parameter	Description
`filename`	A filename for the CSV output.

PERFORMANCE

Enables different levels of query result output for measuring benchmarks by using different performance options.

Syntax

Shell

```
PERFORMANCE [ NETWORK | DATABASE | OFF ]
```

Parameter	Description
`OFF`	This is the default mode. The system executes a query through all normal stages, including the database, client network, and client processing.

Parameter	Description
`NETWORK`	The system executes a query through the database and client network, but discards rows before they are processed by the client (nothing is printed to stdout). Use this mode when you think the client network is hindering performance.
`DATABASE`	The system executes a query through the database only. The database does not send any matching rows to the client. Use this mode when you think the database query execution is affecting performance.

`PERFORMANCE ON` mode has been removed as of JDBC version 3.0.

QUIT

Closes the command-line interface.

Syntax

Shell

```
QUIT
```

REPORT LOG ON

Enables -based metrics collection and redirects reporting events to the configured appenders.

The `REPORT LOG ON` command requires JDBC driver version 2.88 or later.

The Log4j configuration determines the destination for report logs. The Ocient logger name is `com.ocient.util.ReportLogger`. For details about configuring Log4j, see **the Log4j documentation**.

The default Log4j configuration routes report logs to the `RollingFileAppender`. The default configuration includes these settings:

Properties

- `fileName="logs/${sys:ocient.jdbc.sessionId:-unknown}/report.log"`
- `filePattern="logs/${sys:ocient.jdbc.sessionId:-unknown}/%d{yyyy-MM-dd}-%i-report.log.gz`

Rollover Policies

- `TimeBasedTriggeringPolicy`
- `SizeBasedTriggeringPolicy (100MB)`

You can retrieve the default configuration from the `ocient-jdbc4` jar file by using an archive utility like **unzip**. For example:

Shell

```
unzip -p <path-to-ocient-jdbc4> log4j2.xml
```

Syntax

Shell

```
REPORT LOG ON [ interval ]
```

Parameter	Description
`interval`	Optional. Sets the Log4j reporting interval in seconds. If unspecified, defaults to `30`.

REPORT LOG OFF

Disables file-based metrics collection.

> The `REPORT LOG OFF` command requires JDBC driver version 2.88 or later.

Syntax

```
Shell
REPORT LOG OFF
```

ROLLBACK

Rolls back the current transaction and discards all changes made after the transaction started. This command applies only when you disable autocommit with **SET AUTOCOMMIT**. When autocommit is enabled, ROLLBACK has no effect.

Syntax

```
Shell
ROLLBACK [ TRANSACTION ];
```

Entering ROLLBACK with no other arguments rolls back the current transaction on the connection. This action is equivalent to calling the `rollback()` method on the JDBC Connection object.

SELECT

Retrieves the appropriate result set of a SQL `SELECT` statement.

Syntax

```
Shell
SQL select statement
```

SET AUTOCOMMIT

Enables or disables the autocommit mode for the connection. When autocommit is enabled, the database commits each statement immediately. This mode is the default. When autocommit is disabled, the database groups statements into a transaction that you complete with the **COMMIT** or **ROLLBACK** commands.

© 2026 Ocient, Inc.

When you change autocommit from `OFF` to `ON`, the database commits any pending transactions.

Syntax

```
Shell
SET AUTOCOMMIT { ON | OFF }
```

Enables or disables autocommit mode with values `ON` or `OFF`, respectively. When you enter the `ON` value, the database commits each statement immediately. This value is the default. When you enter the `OFF` value, the database groups subsequent statements into a transaction until you enter the COMMIT or ROLLBACK command.

SET MAXROWS

Sets the maximum number of rows allowed in a result set. Applies only to the statement object that executed the command.

Syntax

```
Shell
SET MAXROWS num_of_rows
```

Parameter	Description
`num_of_rows`	The maximum number of rows allowed in a result set. If a query returns more rows than the specified value, the result set is truncated by `num_of_rows`. To disable this setting, you can either set the maximum rows to 0 or use the `RESET` keyword (e.g., `SET MAXROWS RESET;`).

SET ADJUSTFACTOR

Adjusts the query priority by a specified percentage amount at every interval set by the **SET ADJUSTTIME** command.

The `SET ADJUSTFACTOR` command requires JDBC driver version 2.52 or later.

Syntax

```
SET ADJUSTFACTOR priorityAdjustFactor
```

Parameter	Description
`priorityAdjustFactor`	A percentage amount to raise or lower the query priority. Accepted values are in the range of [-1,1]. At every interval set by the `priorityAdjTime`, the query priority is multiplied by `1 + priorityAdjustFactor` to get a new priority value.

SET ADJUSTTIME

Modifies how frequently the query priority is adjusted during the course of execution.

The `SET ADJUSTTIME` command requires JDBC driver version 2.52 or later.

Syntax

```
SET ADJUSTTIME priorityAdjTime
```

Parameter	Description
`priorityAdjTime`	The frequency for adjusting the query priority in seconds.

SET MAXTEMPDISK

Sets the maximum percentage of temporary disk used by queries. Applies only to the statement object that executed the command.

Syntax

```
SET MAXTEMPDISK percentage
```

Parameter	Description
`percentage`	The maximum percentage of temporary disk used by queries. Any queries exceeding this threshold are killed. To disable this setting, use the `RESET` keyword (e.g., `SET MAXTEMPDISK RESET;`).

SET MAXTIME

Sets the maximum query time. Applies only to the statement object that executed the command.

Syntax

```
SET MAXTIME seconds
```

Parameter	Description
`seconds`	The maximum query time. Queries longer than this are killed. To disable this setting, you can either set the maximum time to 0 or use the `RESET` keyword (e.g., `SET MAXTIME RESET;`).

SET PARALLELISM

Sets a limit on the number of cores on each CPU that can be allocated to running the query. Applies only to the statement object that executed the command.

The `parallelism` value limits a query to running on the specified number of cores on each CPU.

Syntax

```
Shell

SET PARALLELISM parallelism
```

Parameter	Description
`parallelism`	Limits a query to running on a specified number of cores on each CPU. To disable this setting, you can either set the `parallelism` value to -1 or use the `RESET` keyword (e.g., `SET PARALLELISM RESET;`).

SET PRIORITY

Sets the scheduling priority to use for queries. Applies only to the statement object that executed the command.

Syntax

```
Shell

SET PRIORITY priority
```

Parameter	Description
`priority`	The scheduling priority to use for queries. Set from 0 to 100. To disable this setting, you can either set the priority to -1 or use the `RESET` keyword. (e.g., `SET PRIORITY RESET;`).

SET SCHEMA

Changes the default schema for tables. Applies only to the statement object that executed the command.

Syntax

```
Shell

SET SCHEMA schema
```

Parameter	Description
`schema`	The default schema for new tables.

SET SERVICECLASS

Limits a query to using the specified service class. Applies only to the statement object that executed the command.

Syntax

Shell

```
SET SERVICECLASS service_class_name
```

Parameter	Description
`service_class_name`	The service class that a query is limited to using. To disable this setting, you can use the `RESET` keyword. (e.g., `SET SERVICECLASS RESET;`).

SOURCE

Reads and executes commands found in the specified file.

Syntax

Shell

```
SOURCE filename
```

Parameter	Description
`filename`	A valid filename or path.

TIMING

Enables or disables reporting the execution time of each query.

Syntax

```shell
TIMING { ON | OFF }
```

Related Links

JDBC Manual

System Catalog

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

JDBC Classes and Methods

This page lists Java® methods that are currently not supported by the Ocient® JDBC Driver for use in Java programs. The Ocient JDBC driver supports methods not listed on this page.

See the linked reference Java 8 documentation pages for details on the functionality of each class.

Java Class	Unsupported Methods or Comments
DriverManager	There are no unsupported methods. Get a connection with getConnection(). getConnection(String, Properties) The string is a URL and must be of the form: `jdbc:ocient://IP:Port/databaseName[;property=value;…]` For description of supported properties, see **JDBC Connection Properties**.
Connection	createSQLXML() prepareCall() releaseSavepoint() rollback() setNetworkTimeout() setSavepoint()
Statement	clearBatch() closeOnCompletion() getGeneratedKeys() getMoreResults() setCursorName() setEscapeProcessing() setMaxFieldSize() setPoolable()
PreparedStatement	getParameterMetaData() setRef() setRowId() setSQLXML() setUnicodeStream() setURL()
ResultSet	absolute() afterLast() beforeFirst() cancelRowUpdates() deleteRow() first() getCursorName() getRef() getRowId() getSQLXML() getUnicodeStream() getURL() insertRow() isLast() last() moveToCurrentRow() moveToInsertRow() previous() refreshRow() relative() updateArray() updateAsciiStream() updateBigDecimal() updateBinaryStream() updateBlob() updateBoolean() updateByte() updateBytes() updateCharacterStream() updateClob() updateDate() updateDouble() updateFloat() updateInt() updateLong() updateNCharacterStream() updateNClob() updateNString() updateNull() updateObject() updateRef() updateRow() updateRowId() updateShort() updateSQLXML() updateString() updateTime() updateTimestamp()
ResultSetMetaData	All methods are supported.
DatabaseMetaData	All methods are supported.

Related Links

© 2026 Ocient, Inc.

Connect Using JDBC

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

JDBC Spark Connector

The Ocient® Apache® Spark™ connector is a **Spark DataSourceV2** implementation that adapts an Ocient System to operate as a first-class source and sink for Spark workloads. Built on top of the Ocient JDBC driver, the connector allows Spark to read from and write to Ocient tables using Spark APIs and SQL statements.

The connector implements Spark catalog and table interfaces so you can register Ocient as a catalog (for `CREATE TABLE`, `INSERT`, `SELECT`, and `SHOW TABLES` SQL statements) or use it for ad-hoc reads and writes.

Key Features

The Ocient Spark connector includes these key features:

- Read Pushdown — The connector accelerates reads by pushing column selection, filters (including on nested fields), aggregations, and queries that only need the first N rows down to the Ocient System while still letting Spark validate the final results.
- Read Partitioning — The connector parallelizes reads by splitting data into multiple Spark partitions based on a partition column. For details, see **Read Partitioning Options**.
- **DataFrame** Write Behavior and Save Modes — The connector controls how it writes DataFrames to Ocient tables by honoring Spark save modes to append (`Append`), truncate-and-replace (`Overwrite`), or fail on existing tables (`ErrorIfExists`).
- Catalog Support — The connector exposes Ocient as a Spark catalog so you can use standard Spark SQL directly on an Ocient System.

Prerequisites

To use the Ocient Spark connector, your system must meet these software requirements.

Software	Version
Ocient	Use Ocient System version 26.1 or later.

© 2026 Ocient, Inc.

Software	Version
Operating System (OS)	Windows®, Linux®, or macOS®. Use the latest version of each OS.
Apache Spark	Version 3.5 or later.
Java®	Version 8 or later.
Ocient JDBC driver	Version 4.0 or later.

Additionally, you must have the `SELECT`, `INSERT`, `CREATE`, and `DELETE` user privileges for the specified database. For details, see **Data Control Language (DCL) Statement Reference**.

Ocient Spark Connector Setup and Initial Use

To start working with the Ocient Spark Connector, register the connector. Then, you can start executing SQL statements.

Connector Registration

For best results, first register the connector as a catalog in Spark.

To register the connector, edit the `spark-defaults.conf` file in your Spark install to include these lines. Replace the `username` and `password` fields with your Ocient System credentials.

Text

```
spark.sql.catalog.ocient_cat=com.ocient.spark.v2.DefaultSource
spark.sql.catalog.ocient_cat.url=jdbc:ocient://host:port/db
spark.sql.catalog.ocient_cat.user=<username>
spark.sql.catalog.ocient_cat.password=<password>
```

Use SQL Statements

After registration, the Spark connector lets you treat your Ocient System like any other Spark catalog. The connector routes SQL operations through the catalog implementation.

Execute the Spark command `USE` to switch to your Ocient catalog and schema for SQL statements. In this case, use the `ocient_cat` catalog and `my_schema` schema.

SQL

```
USE ocient_cat.my_schema;
```

Subsequent commands default to your Ocient catalog and schema, so you no longer need to reference them.

This example creates a new table `my_new_table` with identifier `id`, name `name`, event timestamp `event`, and the structure of an integer and string `nested_date`.

SQL

```
CREATE TABLE my_new_table (
    id BIGINT,
    name VARCHAR,
    event_time TIMESTAMP,
    nested_data STRUCT<a: INT, b: STRING>
);
```

Insert a row into the new table.

SQL

```
INSERT INTO my_new_table VALUES
    (1,'foo', '2025-01-01 12:00:00', (100, 'bar'));
```

Read the row from the table.

SQL

```
SELECT * FROM my_new_table WHERE id = 1;
```

List the table.

SQL

```
SHOW TABLES;
```

Drop the table.

SQL

```
DROP TABLE my_new_table;
```

Use Scala DataFrames

The Ocient Spark connector integrates directly with the Spark DataFrame API, so you can read from and write to Ocient tables using familiar Spark patterns. After you configure the Ocient catalog, you can reference fully qualified table names, and the connector handles all JDBC connectivity and type mapping.

The examples in this section use **Scala** to interact with an Ocient catalog.

Examples

Write from Spark to Ocient

This example takes an existing Spark DataFrame `df` and writes its rows into an Ocient table `my_table`.

Scala

```
df.write.saveAsTable("ocient_cat.my_schema.my_table")
```

Write from Ocient to Spark

This example reads from the Ocient table `my_table` and writes its rows into a new Spark DataFrame `df2`.

Scala

```
val df2 = spark.table("ocient_cat.my_schema.my_table")
```

Ad Hoc Usage

The Ocient Spark connector supports ad-hoc reads and writes using the Spark `.format("ocient")` method. This method is useful for brief operations, but it cannot use the Spark catalog system to create, drop, or list tables.

For example, this Spark command reads an Ocient table and creates the DataFrame `df` from its contents. Substitute `jdbc_connection` with the JDBC connection string for the database, the `username` and `pwd` values for your Ocient username and password, and `my_schema` and `my_table` with the schema and table name for the table to read.

Scala

```
val df = spark.read
    .format("ocient")
    .option("url", "jdbc_connection")
    .option("user", "username")
    .option("password", "pwd")
    .option("dbtable", "my_schema.my_table")
    .load()
```

This command takes the DataFrame `df` and appends its contents into an Ocient table.

Scala

```
df.write
    .format("ocient")
    .option("url", "jdbc_connection")
    .option("user", "username")
    .option("password", "pwd")
    .option("dbtable", "my_schema.my_table")
    .mode("append")
    .save()
```

Bulk Loading Best Practices

Use these recommended OS and Spark settings to get reliable performance and avoid inconsistent writes when using the Ocient JDBC bulk loader with Spark.

For details on bulk loading, see **JDBC Bulk Loading**.

Linux SSH Configuration

Increase the SSH connection capacity on Loader Nodes:

- Set `MaxStartups 1024` in the OS `sshd_config` configuration file on the `loader/SSH` endpoint hosts that accept SSH connections from the bulk loader.
- Restart the SSH service to apply the updated `sshd_config` configuration. For example, on an Ubuntu® system, run `sudo systemctl restart ssh`.

Spark Configuration

Edit the `spark-defaults.conf` configuration file to include these settings:

- `spark.task.maxFailures = 1` — This configuration prevents Spark from retrying failed tasks and potentially duplicating writes.
- `spark.speculation = false` — This configuration prevents Spark from launching speculative duplicate tasks that can re-run writes against Ocient.

Configuration Options

You can set specific configurations for the Ocient Spark connector through standard Spark options:

- Set globally using Spark configuration: Add options to your `spark-defaults.conf` file or your cluster Spark settings (e.g., `spark.sql.catalog.ocient_cat.url=...`).
- Set options per job or per operation: Use Spark (`.option()`) or command-line (`--conf`) statements to set options for one-time usage.

The connector passes most of these settings through to the underlying Ocient JDBC driver as connection properties, but the system interprets a few directly by the connector to shape the generated SQL.

Connection Options

These options control how the connector establishes a JDBC connection to Ocient and identify which table or query Spark should use. All options are for both read and write operations.

Option	Default	Description
`url`	None	Required. The Ocient JDBC URL. If you do not specify the `sparkMode` setting, the connector automatically sets `sparkMode=true`.
`user`	None	Required. The Ocient username.
`password`	None	Required. The password of the user.
`dbtable`	None	Required for adhoc commands using `.format("ocient")`. This option is the Ocient schema and table name (for example, `schema.table`).
`maskPassword`	1	Optional. Determines whether passwords are exposed in Spark connector logs. Supported values are `0` or `1`. If this option is set to `1`, Spark connector logs mask password fields. Otherwise, Spark connector logs include password fields.

Read Partitioning Options

These options control how Spark splits a read into multiple partitions based on a column range, affecting parallelism and data distribution during Ocient table scans. All options are for read operations only. If you do not specify any of these options, you have only one partition.

Option	Default	Description
`numPartitions`	1	Optional. The number of Spark partitions to create for reading.
`partitionColumn`	None	Optional. This is a numeric, date, or timestamp column to use for partitioning the read. If you specify the `partitionColumn` option, you can also define a value range to partition using the `lowerBound` and `upperBound` options. If you do not specify a range, the connector automatically uses the full range of values.
`lowerBound`	None	Optional. The minimum value of the range for the `partitionColumn` option. If you use this option, you must also include the `upperBound` option.

Option	Default	Description
`upperBound`	None	Optional. The maximum value of the range for the `partitionColumn` option. If you use this option, you must also include the `lowerBound` option.
`ocient.minRowsPerPartition`	1	Optional. Minimum target number of rows per Spark partition for the read. The connector uses this as a hint to avoid creating many tiny partitions. This option guarantees that each planned partition covers at least this many estimated rows, where possible.

Read Performance Options

These options tune how efficiently the connector fetches rows from Ocient during reads, including JDBC fetch size and Ocient System internal parallelism. All options are for read operations only.

Option	Default	Description
`fetchSize`	0	The JDBC `fetchSize` in rows for read operations. When you set `fetchSize = N`, the connector asks Ocient to send up to `N` rows per network fetch call, which can reduce round-trips for large result sets. The default is 0, which lets the driver choose an appropriate fetch size. This option behaves the same way as the Spark standard JDBC `fetchsize` option. For details, see the **Spark documentation**.
`ocient.parallelism`	1	Controls Ocient internal parallel execution level for read queries. When you set `ocient.parallelism = N`, the connector appends the `USING PARALLELISM N` clause to `SELECT` SQL statements so that Ocient executes each query with `N` internal workers. This option is separate from the `numPartitions` option, which controls the number of Spark tasks that run in parallel.

Write Performance Options

These options tune how efficiently Spark writes data to Ocient. All options apply to write operations only.

Option	Default	Description
`batchsize`	`4800000`	Controls how many rows Spark sends to Ocient in each JDBC batch during a write operation. When you set `batchsize = N`, the connector groups up to `N` rows per insertion batch, which can significantly improve write throughput for large DataFrame writes. The default is 4,800,000 rows. This option behaves the same way as the Spark standard JDBC `batchsize` option. For details, see the **Spark documentation**.
`connectRetryMaxRetries`	`1`	The number of times the connector retries opening a JDBC connection on the write path after the initial TCP connect returns the `connection refused` message (typically because the SQL Node is at capacity). Set this option to `0` to disable the retry. Only a TCP `connection refused` message triggers a retry. Connect timeouts and all other connect failures stay fatal. The retry runs before the connector sends any request, so the connector never executes the same batch twice.
`connectRetryDelayMs`	`60000`	The delay, in milliseconds, between connect-refused retries on the write path. The default is 60,000 milliseconds (60 seconds). This option has no effect when you set the `connectRetryMaxRetries` option to `0`.

The Spark connector interprets the `connectRetryMaxRetries` and `connectRetryDelayMs` options directly and does not forward them to the JDBC driver as connection properties. These options apply only to the write path and do not affect reads.

Data Type Mapping

The Ocient Spark connector supports all Ocient primitive types and complex types (such as array or tuple).

When Spark creates a table, the connector writes the full Spark logical type into an Ocient `TYPE HINT` clause on each column. For example, for a column that uses an Ocient TUPLE type and maps to a Spark STRUCT type, the connector generates column DDL that includes a type hint such as `TYPE HINT 'STRUCT<myCol: STRING, another: INT>'`.

During a read operation, the connector parses this `TYPE_HINT` field to reconstruct the original Spark schema, including nested field names. If the connector does not find the hint (e.g., for a pre-existing table), the connector maps Ocient TUPLE types to Spark STRUCT types with default field names (`_1`, `_2`, etc.).

Data Types

This table shows the Spark data types that correspond to the equivalent Ocient types. The table also lists whether each Spark type supports round-trips, meaning you can write the type from Spark to Ocient (creating the table) and then read it back into Spark while preserving the original Spark type and structure.

Spark Type	Ocient Type (in `CREATE TABLE` SQL Statement)	Round-trip
`StringType`	`VARCHAR`	Yes
`LongType`	`BIGINT`	Yes
`IntegerType`	`INTEGER`	Yes
`ShortType`	`SMALLINT`	Yes
`ByteType`	`TINYINT`	Yes
`DoubleType`	`DOUBLE`	Yes
`FloatType`	`FLOAT`	Yes
`DecimalType(p,s)`	`DECIMAL(p,s)`	Yes
`BooleanType`	`BOOLEAN`	Yes
`BinaryType`	`VARBINARY`	Yes
`DateType`	`DATE`	Yes
`TimestampType`	`TIMESTAMP`	Yes
`TimestampNTZType`	`TIMESTAMP`	Yes (with `TYPE_HINT`)
`ArrayType`	`ElementType[]`	Yes (with `TYPE_HINT`)
`StructType`	`TUPLE<<...>>`	Yes (with `TYPE_HINT`)
`MapType`	`TUPLE<<key, val>>[]`	Yes (with `TYPE_HINT`)

Spark 4.0 Types

When you run the connector on Spark 4.0 or later, the connector detects these additional Spark types at runtime using reflection and maps them to the corresponding Ocient types and `TYPE_HINT` values, without introducing a compile-time dependency on Spark 4.0 APIs.

Spark 4.0 Type	Ocient Type (in `CREATE TABLE` SQL Statement)	`TYPE_HINT`
`IntervalYearMonthType`	`INTEGER`	`SPARK_INTERVAL_YEAR_MONTH`
`IntervalDayTimeType`	`BIGINT`	`SPARK_INTERVAL_DAY_TIME`
`isVariantType`	`VARCHAR`	`SPARK_VARIANT`

Configure Logging

The Ocient Spark connector logs diagnostic messages through **Apache Log4j 2** under the `com.ocient.spark.v2` namespace. By default, the connector inherits the root logger level configured in your Spark environment.

To change the connector log level, add the following lines to the `$SPARK_HOME/conf/log4j2.properties` file.

Text

```
logger.ocient.name  = com.ocient.spark.v2
logger.ocient.level = WARN
```

Replace `WARN` with one of these supported levels: `DEBUG`, `INFO`, `WARN`, `ERROR`, or `OFF`.

Related Links

Connect Using JDBC

JDBC Manual

JDBC Classes and Methods.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved

OCGraph Java Library

The `OCGraph` Java® class in the Ocient® System implements a set of graph operations (similar to Apache® Spark™ GraphX) directly on tables using JDBC SQL. The overall structure of a graph comprises at least two database tables: one for vertices (nodes) and one for edges (directed links). The OCGraph library runs graph algorithms and transformations by executing SQL over these tables.

Getting Started

To use the OCGraph library of methods, you must have access to an installed Ocient System and the JDBC driver. For details, see **JDBC Manual**.

Use JDBC for connectivity and the `OCGraph` class for graph APIs. The Java library depends on a valid `java.sql.Statement` class to connect to the Ocient System. The class builds and executes SQL using that statement and does not manage connections itself.

Import the `OCGraph` Class

Specify the `com.ocient.jdbc.graph.OCGraph` class in the `import` declaration.

Java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import com.ocient.jdbc.graph.OCGraph;
```

Data Model Requirements

Database tables that use the `OCGraph` library must adhere to a GraphX table structure. In addition to the listed requirements, tables can include other columns.

Table	Description	Requirements
Vertices table	A table with one row per vertex (node). This table typically represents the anchor for graph	The table must contain the `id BIGINT NOT NULL` column

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Table	Description	Requirements
	algorithms and transforms. Many methods join edges to vertices by the `id` column.	definition as the unique vertex identifier.
Edges table	A table with one row per directed edge (relationship). Each row is a directed edge from a source vertex to a destination vertex.	The table must contain these column definitions: `srcid BIGINT NOT NULL`, `destid BIGINT NOT NULL`

Subgraph and Filtering

Use a subgraph or various filters to restrict a graph to relevant vertices and edges. These methods create filtered copies or masked intersections, preserving schema and optional indexes for performance.

subgraph

Creates filtered vertex and edge tables using vertex and triplet predicates, retaining only edges with endpoints that remain after vertex filtering. The method creates the requested indexes and performs best-effort cleanup in the event of failure.

Syntax

Java

```
subgraph(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    resultEdgesTable,
    vertexFilter,
    edgeFilter,
    [ resultVerticesIndexes [ , ... ] ],
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the filtered vertices table to create.
`resultEdgesTable`	String	Name of the filtered edges table to create.
`vertexFilter`	String	SQL predicate to filter the vertices (without the `WHERE` keyword). Example: `status = 'ACTIVE' AND score > 0`
`edgeFilter`	String	SQL predicate that the system evaluates in a triplet context using the aliases `a` (source vertex), `b` (edge), and `c` (destination vertex). Example: `b.weight > 0.5 AND a.country = c.country`
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL.

Example

Create an active customer subgraph that includes only purchases exceeding $50 where the source and destination share a region.

© 2026 Ocient, Inc.

Java

```
OCGraph.subgraph(
    "sales",
    "customers",
    "purchases",
    "sales",
    "customers_active",
    "purchases_active",
    "status = 'ACTIVE' AND score > 0",
    "b.amount > 50 AND a.region = c.region",
    new ArrayList<>(List.of("id","region")),
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

filterVertices

Creates a filtered subgraph by selecting vertices that match a predicate while retaining only edges with endpoints that are in the filtered vertex set.

Syntax

Java

```
filterVertices(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    resultEdgesTable,
    vertexFilter,
    [ resultVerticesIndexes [ , ... ] ],
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.

Argument	Data Type	Description
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the result vertices table. The name must not conflict with the names of input tables.
`resultEdgesTable`	String	Name of the edges table to create.
`vertexFilter`	String	SQL predicate for vertices (without the `WHERE` keyword). Example: `status = 'ACTIVE'`
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Filter US customers and retain edges with endpoints that remain in the filtered vertex set.

Java

```
OCGraph.filterVertices(
    "sales", "customers", "purchases",
    "sales", "customers_us", "purchases_us",
    "country = 'US'",
    new ArrayList<>(List.of("id")),
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

filterEdges

Creates a filtered edges table by selecting edges that match a predicate.

Syntax

Java

```
filterEdges(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultEdgesTable,
    edgeFilter,
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputEdgesTable`	String	Input edges table (must have `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultEdgesTable`	String	Name of the edges table to create.
`edgeFilter`	String	SQL predicate on edges (without the `WHERE` keyword). Example: `weight > 0.5 AND type = 'ACTIVE'`
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and database connection.

Example

This example demonstrates how to create a filtered edges table from an existing `purchases` edge set by keeping only edges that meet a business rule (`weight > 0.5` and `ACTIVE`). Then, the `filterEdges` method indexes the result on the `srcid` and `destid` columns for faster lookups.

Java

```
OCGraph.filterEdges(
    "sales", "purchases",
    "sales", "purchases_filtered",
    "weight > 0.5 AND type = 'ACTIVE'",
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

mask

Creates a masked subgraph by intersecting two graphs. Vertices intersect if the vertex identifier is present in both graphs. Edges intersect when the `srcid` and `destid` values are present in both graphs.

The method copies rows that intersect from the graph defined by the arguments `inputVerticesTable` and `inputEdgesTable`, including any attributes.

You can optionally create indexes on the result subgraph tables.

Syntax

Java

```
mask(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    otherSchema,
    otherVerticesTable,
    otherEdgesTable,
    resultSchema,
    resultVerticesTable,
    resultEdgesTable,
    [ resultVerticesIndexes [ , ... ] ],
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the primary input tables.
`inputVerticesTable`	String	The primary vertices table to intersect (must have an `id` column). The masked subgraph created by this method copies rows that intersect from this table.
`inputEdgesTable`	String	The primary edges table to intersect (must have `srcid` and `destid` columns). The masked subgraph created by this method copies rows that intersect from this table.
`otherSchema`	String	Schema containing the second graph.
`otherVerticesTable`	String	The second vertices table to intersect.
`otherEdgesTable`	String	The second edges table to intersect.
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create.
`resultEdgesTable`	String	Name of the edges table to create.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Create a masked subgraph by intersecting two graphs. The example copies vertices and edges that are present in both graphs, along with the remaining endpoints.

Java

```
OCGraph.mask(
    "sales", "customers", "purchases",
    "ref",   "customers_ref", "purchases_ref",
    "sales", "customers_masked", "purchases_masked",
    new ArrayList<>(List.of("id")),
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

Transformations

Construct new vertex or edge tables by computing derived columns, reversing direction, or aggregating duplicates. These methods do not change the original inputs. Instead, the methods materialize new results.

mapVertices

Creates a new vertices table with the identifier `id` and computed columns. Use the `resultColumnExpressions` argument to calculate additional columns. This method can also add indexes before inserting data.

Syntax

Java

```
mapVertices(
    inputSchema,
    inputVerticesTable,
    resultSchema,
    resultVerticesTable,
    resultColumnExpressions [ , ... ],
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create.
`resultColumnExpressions`	ArrayList	One or more SQL expressions defining result columns beyond `id`. Use the `AS alias_name` keyword for stable names.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Create vertices by adding derived columns, `name_upper` and `is_vip`, and generate indexes for the `id` and `name_upper` columns.

Java

```
OCGraph.mapVertices(
    "sales", "customers",
    "sales", "customers_enriched",
    new ArrayList<>(List.of(
        "UPPER(name) AS name_upper",
        "CASE WHEN score > 1000 THEN true ELSE false END AS is_vip"
    )),
    new ArrayList<>(List.of("id","name_upper")),
    stmt
);
```

mapEdges

Creates a new edges table with `srcid`, `destid`, and computed columns. Expressions should refer to input edge columns by their original names, and each computed expression should include an `AS alias` keyword.

Syntax

```
Java

mapEdges(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultEdgesTable,
    [ resultColumnExpressions [ , ... ] ],
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputEdgesTable`	String	Input edges table (must have `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultEdgesTable`	String	Name of the edges table to create.
`resultColumnExpressions`	ArrayList	Optional. SQL expressions for additional edge columns (besides the `srcid` and `destid` columns). Use the `AS alias_name` keyword for stable names.
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Create a new edges table with two columns, `discounted_amount` and `big_txn`, and generate indexes for the `srcid` and `destid` columns.

Java

```
OCGraph.mapEdges(
    "sales", "purchases",
    "sales", "purchases_enriched",
    new ArrayList<>(List.of(
        "amount * 0.9 AS discounted_amount",
        "CASE WHEN amount > 100 THEN 1 ELSE 0 END AS big_txn"
    )),
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

mapTriplets

Creates a new edges table with computed columns that reference a triplet made of: `a` (source vertex), `b` (edge), and `c` (destination vertex). The output automatically includes `b.srcid` and `b.destid` columns.

Syntax

Java

```
mapTriplets(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultEdgesTable,
    [ resultColumnExpressions [ , ... ] ],
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column). Used as the `a` and `c` vertices.

Argument	Data Type	Description
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns). This method uses this argument as the `b` edge.
`resultSchema`	String	A writable schema to create the result edges table.
`resultEdgesTable`	String	Name of the edges table to create.
`resultColumnExpressions`	ArrayList	Optional. SQL expressions that can reference `a`, `b`, or `c`. Use the `AS alias_name` keyword for stable names.
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Create a new triplet table from the vertices and edges tables with the `amount` and `same_country` columns, and generate indexes for the `src` and `destid` columns.

Java

```
OCGraph.mapTriplets(
    "sales", "customers", "purchases",
    "sales", "purchases_triplets",
    new ArrayList<>(List.of(
        "b.amount AS amount",
        "CASE WHEN a.country = c.country THEN 1 ELSE 0 END AS same_country"
    )),
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

reverseEdges

Creates a new edges table with the `srcid` and `destid` columns reversed, preserving other columns. Use this method to traverse a graph in the opposite direction.

Syntax

Java

```
reverseEdges(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultEdgesTable,
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultEdgesTable`	String	Name of the reversed edges table to create.
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Transform edge direction by reversing the `srcid` and `destid` columns. The example also creates indexes for these columns.

Java

```
OCGraph.reverseEdges(
    "sales", "purchases",
    "sales", "purchases_reversed",
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

© 2026 Ocient, Inc.

groupEdges

Groups duplicate rows of the `srcid` and `destid` columns, producing one row for each unique pair of values in a new edges table. This method performs aggregations based on one or more SQL expressions.

Syntax

Java

```
groupEdges(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultEdgesTable,
    resultColumnExpressions [ , ... ],
    [ resultEdgesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the grouped tables.
`resultEdgesTable`	String	Name of the grouped edges table to create.
`resultColumnExpressions`	ArrayList	One or more aggregate expressions for the `srcid` and `destid` columns. For details about SQL aggregations, see **Aggregate Functions**. Use the `AS alias_name` keyword for stable names.
`resultEdgesIndexes`	ArrayList	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Create a new edge table that includes SQL aggregations for counting unique transactions `txn_count` and total sums `total_amount`. Also, this method generates indexes for the `src` and `destid` columns.

Java

```
OCGraph.groupEdges(
    "sales", "purchases",
    "sales", "purchases_grouped",
    new ArrayList<>(List.of(
        "COUNT(*) AS txn_count",
        "SUM(amount) AS total_amount"
    )),
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

Triplets

Produce triplet representations that are made of `a` (source vertex), `b` (edge), and `c` (destination vertex), either as a logical view or a materialized table for downstream queries.

createTripletsView

Creates a view that combines the edge table with the source and destination vertex attributes. This view is useful for analyzing relationships without having to repeatedly join tables. The view includes these columns:

- All original edge columns (including the `srcid` and `destid` columns).
- All source-vertex columns except `id`. Source-vertex column names have the `src_` prefix.
- All destination-vertex columns except `id`. Destination-vertex column names have the `dest_` prefix.

Use the **createTripletsTable** method if you want to create a materialized table with indexes instead of a view.

Syntax

Java

```
createTripletsView(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultTripletsView,
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A schema to create the view.
`resultTripletsView`	String	Name of the triplets view to create. The name must not conflict with the names of input tables.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Create a triplets view to inspect edges with joined source and destination vertex attributes.

Java

```
OCGraph.createTripletsView(
    "sales", "customers", "purchases",
    "sales", "triplets_v",
    stmt
);
```

createTripletsTable

Creates a materialized table that combines the edge table with the source and destination vertex attributes. This table is useful for analyzing relationships without having to repeatedly join tables. The created table includes these columns:

- All original edge columns (including the `srcid` and `destid` columns).
- All source-vertex columns except `id`. Source-vertex column names have the `src_` prefix.
- All destination-vertex columns except `id`. Destination-vertex column names have the `dest_` prefix.

Use the **createTripletsView** method if you want to create a view instead of a new table.

Syntax

Java

```
createTripletsTable(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultTripletsTable,
    [ resultTripletsIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A schema to create the triplets table.
`resultTripletsTable`	String	Name of the triplets table to create. The name must not conflict with the names of input tables.
`resultTripletsIndexes`	ArrayList	Optional. Columns to index in the result table. Specify an empty list for none.

Argument	Data Type	Description
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Create a new table for triplets. Generate indexes for the `src_id` and `dest_id` columns.

Java

```
OCGraph.createTripletsTable(
    "sales", "customers", "purchases",
    "sales", "triplets_t",
    new ArrayList<>(List.of("src_id","dest_id")),
    stmt
);
```

Degrees

Compute degree metrics for each vertex from the edges table. These methods produce small vertex tables suitable for joins and analytics.

inDegrees

Computes how many edges point to each vertex in an edge table by counting how many times each unique `destid` value appears. The result table has two columns: `id` (the destination vertex) and `in_degree` (the count).

Syntax

© 2026 Ocient, Inc.

Java

```
inDegrees(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input table.
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A schema to create the in-degree table.
`resultVerticesTable`	String	Name of the vertices table to create (columns include `id` and `in_degree`).
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Compute in-degrees per vertex and generate an index on the `id` column.

Java

```
OCGraph.inDegrees(
    "sales", "purchases",
    "sales", "customers_in_degree",
    new ArrayList<>(List.of("id")),
    stmt
);
```

outDegrees

Computes how many edges originate from each vertex in an edge table by counting how many times each unique `srcid` value appears. The result table has two columns: `id` (the source vertex) and `out_degree` (the count).

Syntax

```java
outDegrees(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input table.
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	Schema to create the out-degree table.
`resultVerticesTable`	String	Name of the vertices table to create (columns include `id` and `out_degree`).
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Compute the out-degree count for each vertex and generate an index on the `id` column.

© 2026 Ocient, Inc.

Java

```
OCGraph.outDegrees(
    "sales", "purchases",
    "sales", "customers_out_degree",
    new ArrayList<>(List.of("id")),
    stmt
);
```

degrees

Computes the total degrees (in-degrees and out-degrees) for each vertex in an edge table by counting how many times each unique `srcid` and `destid` value appears. The result table has two columns: `id` (the destination or source vertex) and `degree` (the count).

Syntax

Java

```
degrees(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input table.
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A schema to create the degree table.
`resultVerticesTable`	String	Name of the vertices table to create (columns include `id` and `degree`).
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Argument	Data Type	Description
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Compute total degrees for each vertex and generate an index on the `id` column.

Java

```
OCGraph.degrees(
    "sales", "purchases",
    "sales", "customers_degree",
    new ArrayList<>(List.of("id")),
    stmt
);
```

Vertex Extraction and Joins

Build vertex sets from edges and combine vertex attributes across tables. These methods are useful for shaping vertex properties and consolidating features.

fromEdges

Builds a vertices table from an edges table by extracting the unique source and destination identifiers.

This method can optionally compute additional columns using SQL expressions by referencing the unique identifier as `ids.id`.

The created table always contains the `id` column with one additional column per expression.

Syntax

Java

```
fromEdges(
    inputSchema,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultColumnExpressions [ , ... ] ],
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input table.
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A schema to create the result vertices table.
`resultVerticesTable`	String	Name of the result vertices table (always includes `id`).
`resultColumnExpressions`	ArrayList	Optional. Specify a list of SQL expressions that reference `ids.id` to add additional columns.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Create a vertices table from edge endpoints and add a `bucket` column that assigns each vertex to one of 10 buckets. Generate an index for the `id` and `bucket` columns.

© 2026 Ocient, Inc.

Java

```
OCGraph.fromEdges(
    "sales", "purchases",
    "sales", "customers_from_edges",
    new ArrayList<>(List.of(
        "ids.id % 10 AS bucket"
    )),
    new ArrayList<>(List.of("id","bucket")),
    stmt
);
```

joinVertices

Merges two vertices tables by retaining every row from a primary table (`inputVerticesTable`) and selectively updating rows that also appear in the modification table (`modificationVerticesTable`). The merged table includes all vertices from the primary table that do not appear in the modification table.

For vertices that appear in both tables, the method must include a list of expressions (`resultAttributeExpressions`) in the same column order for every non-identifier column in the merged result table. These SQL expressions can add computations to columns or simply add aliases if no changes are needed. Each expression can reference columns from the primary table (using alias `a`) or from the modification table (using alias `b`).

Syntax

Java

```
joinVertices(
    inputSchema,
    inputVerticesTable,
    modificationSchema,
    modificationVerticesTable,
    resultSchema,
    resultVerticesTable,
    resultAttributeExpressions [ , ... ],
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the primary input table.
`inputVerticesTable`	String	Input vertices table (with the alias `a`). This table must have an `id` column.
`modificationSchema`	String	A schema for the modification vertices table.
`modificationVerticesTable`	String	Modification vertices table (with the alias `b`). This table must have an `id` column.
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create.
`resultAttributeExpressions`	ArrayList	A list of SQL expressions that define the non-identifier columns of the joined result. Each expression can reference the left (`inputVerticesTable`) vertex as `a` and the right (`modificationVerticesTable`) vertex as `b`. You must end every expression with an explicit alias using `AS alias_name` so the output column names are stable.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Merge vertex attributes and generate indexes for the `id` and `status` columns. This example includes two SQL expressions to update the `status` and `score` columns based on the modification vertex table using the `COALESCE` SQL reference function.

© 2026 Ocient, Inc.

Java

```
OCGraph.joinVertices(
    "sales", "customers",
    "sales", "customers_updates",
    "sales", "customers_merged",
    new ArrayList<>(List.of(
        "COALESCE(b.new_status, a.status) AS status",
        "COALESCE(b.score_delta + a.score, a.score) AS score"
    )),
    new ArrayList<>(List.of("id","status")),
  stmt
);
```

innerJoinVertices

Performs an inner join on two vertex tables using an equality comparison `a.id = b.id`. The result table automatically includes the `id` column from the first table.

The method must include a list of SQL expressions (`resultAttributeExpressions`) in the same column order for every non-identifier column in the merged result table. These SQL expressions can add computations to columns, or simply add aliases if no changes are needed. Each expression can reference columns from the primary table using the alias `a` or from the modification table using the alias `b`.

Syntax

Java

```
innerJoinVertices(
    inputSchema,
    inputVerticesTable,
    otherSchema,
    otherVerticesTable,
    resultSchema,
    resultVerticesTable,
    resultAttributeExpressions [ , ... ],
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A schema for the left vertices table.
`inputVerticesTable`	String	Input vertices table for the left side of the join (must have an `id` column).
`otherSchema`	String	A schema for the right vertices table.
`otherVerticesTable`	String	The other vertices table for the right side of the join (must have an `id` column).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create.
`resultAttributeExpressions`	ArrayList	A list of SQL expressions that define the non-identifier columns of the joined result. Each expression can reference the left (`inputVerticesTable`) vertex as `a` and the right (`otherVerticesTable`) vertex as `b` . You must end every expression with an explicit alias using `AS alias_name` to ensure the output column names are stable.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Create an inner join between two vertex tables and generate an index on the `id` column.

Java

```
OCGraph.innerJoinVertices(
    "sales", "customers",
    "sales", "profiles",
    "sales", "customers_joined",
    new ArrayList<>(List.of(
        "a.id AS id",
        "a.status AS status",
        "b.tier AS tier"
    )),
    new ArrayList<>(List.of("id")),
    stmt
);
```

outerJoinVertices

Performs a left outer join between two vertices tables using an equality comparison `a.id = b.id`. The result table includes all rows from the left table. For left-table rows that have no match in the right table, any expression that reads columns from the right table with the alias `b` evaluates to NULL (while expressions that only read the table with the alias `a` remain non-NULL as usual).

The method must include a list of SQL expressions (`resultAttributeExpressions`) in the same column order for every non-identifier column in the merged result table. These SQL expressions can add computations to columns or simply add aliases if no changes are needed. Each expression can reference columns from the primary table using the alias `a` or from the modification table using the alias `b`.

Syntax

Java

```
outerJoinVertices(
    inputSchema,
    inputVerticesTable,
    otherSchema,
    otherVerticesTable,
    resultSchema,
    resultVerticesTable,
    resultAttributeExpressions [ , ... ],
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables (aliased as `a`).
`inputVerticesTable`	String	Left vertices table (must have an `id` column).
`otherSchema`	String	Schema of the right vertices table (with the alias `b`).
`otherVerticesTable`	String	Right vertices table (must have an `id` column).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create.
`resultAttributeExpressions`	ArrayList	A list of SQL expressions that define the non-ID columns of the joined result. Each expression can reference the left (`inputVerticesTable`) vertex as `a` and the right (`otherVerticesTable`) vertex as `b`. You must end every expression with an explicit alias using `AS alias_name` to ensure the output column names are stable.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Perform a left outer join on two vertices tables and generate an index on the `id` column.

Java

```
OCGraph.outerJoinVertices(
    "sales", "customers",
    "sales", "profiles",
    "sales", "customers_joined",
    new ArrayList<>(List.of(
        "a.id AS id",
        "a.status AS status",
        "b.tier AS tier"
    )),
    new ArrayList<>(List.of("id")),
    stmt
);
```

collectNeighbors

For each vertex in a table, this method collects information on neighbors (identifier and any attributes) as an array of tuples. For a specified direction (`IN` , `OUT` , or `BOTH`), the method aggregates tuples representing each neighboring vertex into an array.

The direction types are:

- `IN` — Neighbors with edges pointing to the vertex (edges where `destid = id`).
- `OUT` — Neighbors that the vertex points to (edges where `srcid = id`).
- `BOTH` — Union of `IN` and `OUT` with neighbors from incoming (`destid = id`) and outgoing (`srcid = id`) edges.

The result table has the columns `id` (the vertex identifier) and `neighbors` (an array of tuples representing each neighbor).

If an error occurs after table creation, the method drops the result table.

Syntax

© 2026 Ocient, Inc.

Java

```
collectNeighbors(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultTable,
    direction,
    [ resultIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultTable`	String	Name of the neighbors collection table (with the `id` and `neighbors` columns).
`direction`	EdgeDirection	Specifies the direction of traversal. Supported values are: `IN` — Neighbors that have edges pointing to the vertex (edges where `destid = id`). For example, if an edge `5` points to `10`, then for `id=10`, neighbor `5` is included. `OUT` — Neighbors that the vertex points to (edges where `srcid = id`). For example, if an edge `5` points to `10`, then for `id=5`, neighbor `10` is included. `BOTH` — The union of `IN` and `OUT`. This traversal includes neighbors from edges pointing to `id` and edges from `id`.
`resultIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

© 2026 Ocient, Inc.

Example Collect incoming neighbors for each vertex and generate an index on the `id` column. The `direction` argument set to `IN` collects neighbors pointing to `id`.

Java

```
OCGraph.collectNeighbors(
    "sales", "customers", "purchases",
    "sales", "neighbors_in",
    OCGraph.EdgeDirection.IN,
    new ArrayList<>(List.of("id")),
    stmt
);
```

collectEdges

For each vertex in a table, this method collects an array of adjacent edge rows based on the specified direction. The result table has two columns: `id` (the vertex identifier) and `edges` (an array of tuples, each tuple containing all columns from the edges table for a connected edge).

The `direction` types are:

- `IN` — Edges pointing to the vertex (edges where `destid = id`).
- `OUT` — Edges originating from the vertex (edges where `srcid = id`).
- `BOTH` — Union of `IN` and `OUT` that includes edges from incoming (`destid = id`) and outgoing (`srcid = id`) directions. This direction retains duplicates.

If an error occurs after table creation, the method drops the result table.

Syntax

© 2026 Ocient, Inc.

Java

```
collectEdges(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultTable,
    direction,
    [ resultIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultTable`	String	Name of the edge collection table (with the `id` and `edges` columns).
`direction`	EdgeDirection	Specifies the direction of traversal. Supported values are: `IN` — Neighbors that have edges pointing to the vertex (edges where `destid = id`). For example, if an edge `5` points to `10`, then for `id=10`, neighbor `5` is included. `OUT` — Neighbors that the vertex points to (edges where `srcid = id`). For example, if an edge `5` points to `10`, then for `id=5`, neighbor `10` is included. `BOTH` — The union of `IN` and `OUT`. This traversal includes neighbors from edges pointing to `id` and edges from `id`.
`resultIndexes`	ArrayList	Optional. Columns to index. Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

© 2026 Ocient, Inc.

Example Collect outgoing edges for each vertex and generate an index for the `id` column. The example sets the `direction` to `OUT` to collect edges from `id`.

Java

```
OCGraph.collectEdges(
    "sales", "customers", "purchases",
    "sales", "outgoing_edges",
    OCGraph.EdgeDirection.OUT,
    new ArrayList<>(List.of("id")),
    stmt
);
```

Algorithms

High-level graph algorithms that iterate over the graph structure to produce labels, components, or counts.

labelPropagation

Executes the **Label Propagation Algorithm** (LPA) to assign community labels to vertices.

Each vertex starts with its own identifier as its label. For a number set by the `maxIterations` argument, each vertex updates its label to the most frequent label among its neighbors. The algorithm determines ties by choosing the smallest label. The algorithm uses temporary tables for intermediate results and drops these tables when the process completes or if it fails. Isolated vertices retain their initial label. The final table stores `id` and `label` columns and can include indexes.

Syntax

© 2026 Ocient, Inc.

Java

```
labelPropagation(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    maxIterations,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create (columns include `id` and `label`).
`maxIterations`	numeric	Maximum number of iterations (must be `1` or greater).
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Run label propagation for 10 iterations and assign labels to vertices. Generate an index on the `id` column.

Java

```
OCGraph.labelPropagation(
    "sales", "customers", "purchases",
    "sales", "lpa_labels",
    10,
    new ArrayList<>(List.of("id")),
    stmt
);
```

connectedComponents

Identifies the connected components of an undirected graph. This algorithm configures a Pregel computation in which each vertex initially sets its component label equal to its own identifier `id`.

In each iteration, vertices send their component label to neighbors. Each vertex updates based on the aggregated minimum value of its current component label and any received values. The process repeats until no more updates occur.

The result table maps each vertex `id` to its final `component` label.

Syntax

Java

```
connectedComponents(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).

Argument	Data Type	Description
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example Compute connected components for a maximum of 20 iterations and generate an index on the `id` column.

Java

```
OCGraph.connectedComponents(
    "sales", "customers", "purchases",
    "sales", "components",
    20,
    new ArrayList<>(List.of("id")),
    stmt
);
```

stronglyConnectedComponents

Computes strongly connected components (SCC) in a directed graph. This method runs a recursive algorithm that partitions vertices into subsets where every vertex is reachable from other vertices in the same subset.

This method uses recursive partitioning. The algorithm selects a pivot (typically the minimum identifier `id`), computes its predecessor set (vertices that can reach the pivot), and its descendant sets (vertices reachable from the pivot). Then, the method identifies the SCC as their intersection, removes that SCC from the graph, and recurses on the remainder until all vertices have been assigned to an SCC. The output contains columns for the `id` and `component` identifiers (the minimum `id` in the SCC).

The method creates temporary tables in the result schema to store intermediate results. This method drops these tables when the computation completes or fails. The final result

table contains two columns: `id` (vertex identifier) and `component` (the minimum vertex identifier in its SCC subset).

Syntax

Java

```
stronglyConnectedComponents(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create (columns include `id` and `component`).
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Compute the SCC and generate an index on the `id` column.

Java

```
OCGraph.stronglyConnectedComponents(
    "sales", "customers", "purchases",
    "sales", "scc",
    new ArrayList<>(List.of("id")),
    stmt
);
```

TriangleCount

TriangleCount identifies all 3-cycles (triangles) in the graph and counts how many distinct triangles each vertex participates in.

The algorithm first builds a canonical, undirected edge set by ensuring `srcid < destid` and removing duplicates to prevent double-counting. If your input edges are already canonicalized and deduplicated, use `TriangleCount.runPreCanonicalized` to skip preprocessing for faster performance.

The method then counts triangles (`a` , `b` , `c`) where `a < b < c` by intersecting neighbor lists and aggregates per-vertex participation to produce a result table with the `id` and `triangle_count` columns.

`run` syntax

Java

```
TriangleCount.run(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

`runPreCanonicalized` syntax

Java

```
TriangleCount.runPreCanonicalized(
    inputSchema,
    inputVerticesTable,
    canonicalEdgesTable,
    resultSchema,
    resultVerticesTable,
    [ resultVerticesIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns). For `runPreCanonicalized` execution, the method assumes this table is already canonicalized (e.g., `srcid < destid`) and deduplicated.
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create (columns include `id` and `triangle_count`).
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Examples

Count Triangles Using `run`

Canonicalize the raw edges internally, count unique triangles, and write per-vertex triangle counts with an index on the `id` column.

Java

```
OCGraph.TriangleCount.run(
    "sales",
    "customers",
    "purchases",
    "sales",
    "triangle_counts",
    new ArrayList<>(List.of("id")),
    stmt
);
```

Count Triangles Using `runPreCanonicalized`

Use a pre-canonicalized, deduplicated edge table to count triangles and write per-vertex triangle counts with an index on the `id` column.

Java

```
OCGraph.TriangleCount.runPreCanonicalized(
    "sales",
    "customers",
    "purchases_canonical",
    "sales",
    "triangle_counts",
    new ArrayList<>(List.of("id")),
    stmt
);
```

pregel

Provides a generic vertex-centered iteration framework for custom graph algorithms, similar to the **Pregel model**.

Each iteration updates vertex states by sending messages along edges and then aggregating these messages to compute new states. The algorithm continues iterating until it reaches convergence (no state changed or no messages produced) or a specified iteration cap.

The algorithm uses multiple specified SQL expressions.

© 2026 Ocient, Inc.

Syntax

Java

```
pregel(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    initializerExpr,
    [ sendToSourceExpr ],
    [ sendToDestExpr ],
    aggregateExpr,
    updaterExpr,
    maxIterations,
    [ resultVerticesIndexes [ , ... ] ],
   stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create (columns include `id` and `result`).
`initializerExpr`	String	A SQL expression to compute the initial state for each vertex (e.g., `CASE WHEN type = 'seed' THEN 1.0 ELSE 0.0 END`).
`sendToSourceExpr`	String	Optional. Defines the message sent to the source vertex of an edge, referencing the current vertex states `a.state` (source) and `c.state` (destination), and any edge attributes in `b`. Example: `CASE WHEN a.country = c.country THEN 1 ELSE 0 END`
`sendToDestExpr`	String	Optional. Defines the message sent to the destination vertex of an edge, referencing the current vertex states

Argument	Data Type	Description
		`a.state` (source) and `c.state` (destination), and any edge attributes in `b`. Example: `CASE WHEN a.country = c.country THEN 1 ELSE 0 END`
`aggregateExpr`	String	A SQL aggregation to combine messages per vertex (e.g., `SUM(msg)`). For a list of supported aggregations, see **Aggregate Functions**.
`updaterExpr`	String	A SQL expression to compute the next state from the current state and aggregated messages. For example, this code updates the state to the minimum current state and the aggregated message (similar to the **connectedComponent** method): `LEAST(a.state, COALESCE(m.aggregated_message, a.state)) AS state`
`maxIterations`	numeric	Maximum number of iterations (must be either `-1` or a value of `1` or greater). If this value is `-1`, the Pregel algorithm has no limit, and it runs until convergence. A value of `0` causes the algorithm to return an error.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Run a simple Pregel computation summing incoming edge amounts into the vertex state for 10 iterations at most, and generate an index on the `id` column.

Java

```
OCGraph.pregel(
    "sales", "customers", "purchases",
    "sales", "pregel_result",
    "0 AS state",
    "b.amount",
    null,
    "SUM(msg) AS aggregated_message",
    "state + COALESCE(aggregated_message, 0) AS state",
    10,
    new ArrayList<>(List.of("id")),
    stmt
);
```

Paths and Ranking

These methods include the shortest-path and PageRank algorithms.

shortestPaths

Computes the shortest distance from every vertex to each set of landmark vertices using an iterative relaxation algorithm. The algorithm resembles **Bellman-Ford** but simultaneously handles multiple destinations.

Each landmark starts at distance `0` and all others at positive infinity. On each iteration, the algorithm examines every edge and checks whether traveling through the connected neighbor would yield a shorter route to a landmark. If a shorter route exists, the algorithm updates the distance of the source vertex. The process stops when no distances improve, or the algorithm reaches the maximum number of iterations.

After the process finishes, the algorithm writes a result table with the `srcid`, `destid`, and `distance` columns.

Syntax

Java

```
shortestPaths(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultTable,
    landmarks [ , ... ],
    [ edgeWeightColumn ],
    maxIterations,
    [ resultIndexes [ , ... ] ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result table.
`resultTable`	String	Name of the distances result table. This table has the `srcid`, `destid`, and `distance` columns.
`landmarks`	ArrayList	One or more landmark vertex identifiers. This list must be non-empty and contain no NULLs.
`edgeWeightColumn`	String	Optional. The name of an edge weight column. If you do not specify this argument, the default value is `1.0`.
`maxIterations`	numeric	Maximum number of relaxation iterations. This value must be `1` or greater.
`resultIndexes`	ArrayList	Optional. The list of columns to index in the result table (`srcid`, `destid`, and `distance` columns). Specify an empty list for none.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

© 2026 Ocient, Inc.

Example

Compute distances from landmarks and generate indexes on the `src` and `dest` columns.

Java

```
OCGraph.shortestPaths(
    "sales", "customers", "purchases",
    "sales", "distances",
    new ArrayList<>(List.of(1L, 42L)),
    null, // unweighted
    10,
    new ArrayList<>(List.of("srcid","destid")),
    stmt
);
```

staticPageRank

Computes **PageRank** scores for each vertex over a fixed number of iterations.

The algorithm follows the standard PageRank formula with a reset probability (`resetProb`) and uses common table expressions to calculate contributions from incoming edges and redistribute rank from dangling nodes.

The algorithm supports two variants:

- Standard PageRank — All vertices start with rank `1.0/N` , where `N` is the number of vertices. Specify this variant if `personalizationSrcId` is `null` .
- Personalized PageRank — The specified vertex starts with a rank of `1.0` , while others start with a rank of `0.0` . Specify this variant if `personalizationSrcId` is a vertex identifier.

After running PageRank for a fixed number of iterations, the method writes a result vertices table containing all original vertex columns with a new PageRank scoring column.

Syntax

Java

```
staticPageRank(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    numIterations,
    resetProb,
    [ resultVerticesIndexes [ , ... ] ],
    [ personalizationSrcId ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create (columns include all vertex columns and a `pagerank` column).
`numIterations`	numeric	Number of iterations to run. This value must be 1 or greater.
`resetProb`	numeric	A value between `0.0` and `1.0` that controls the damping factor for how PageRank random surfer moves across vertices. A high value (e.g., `0.85`) puts more emphasis on link structure, encouraging the algorithm to move from each vertex to a neighbor. This value means PageRank scores tend to concentrate around well-linked regions. A lower score (e.g., `0.50`) allows the algorithm to ignore edges and instead jump to a vertex chosen from a base distribution. For standard PageRank, this base is uniform over all vertices. For personalized PageRank, the base is biased toward the specified vertex. This behavior creates more uniform scoring with less sensitivity to link topology.

Argument	Data Type	Description
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`personalizationSrcId`	Long	Determines whether PageRank uses the standard or the personalized variant. For personalized mode, specify a vertex identifier. PageRank scoring starts with this vertex set at `1.0`, while all other vertices start at `0.0`. For standard mode, specify `null`. All vertices start with rank `1.0/N`, where `N` is the number of vertices.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

Example

Run fixed-iteration PageRank and generate an index on the `id` column. This example uses a high reset probability `resetProb` of `0.85` to ensure the ranking concentrates on highly linked regions.

Java

```
OCGraph.staticPageRank(
    "sales", "customers", "purchases",
    "sales", "pagerank_static",
    10,
    0.85,
    new ArrayList<>(List.of("id")),
    null,
    stmt
);
```

dynamicPageRank

Computes PageRank scores until convergence based on a specified threshold value (`tolerance`). Unlike the **staticPageRank** method, this algorithm runs iterations until the sum of absolute differences between ranks in successive iterations is less than or equal to the `tolerance` value. The algorithm handles personalization similarly to

`staticPageRank`. At each iteration, the method uses the PageRank formula, collects rank values, and redistributes them.

The algorithm supports two variants:

- Standard PageRank — All vertices start with rank `1.0/N`, where `N` is the number of vertices. Specify this variant if `personalizationSrcId` is `null`.
- Personalized PageRank — The specified vertex starts with a rank of `1.0`, while others start with a rank of `0.0`. Specify this variant if `personalizationSrcId` is a vertex identifier.

After running PageRank until the system reaches the `tolerance` threshold, the method writes a vertices table containing all the original vertex columns with a new PageRank scoring column.

Syntax

Java

```
dynamicPageRank(
    inputSchema,
    inputVerticesTable,
    inputEdgesTable,
    resultSchema,
    resultVerticesTable,
    tolerance,
    resetProb,
    [ resultVerticesIndexes [ , ... ] ],
    [ personalizationSrcId ],
    stmt
)
```

Argument	Data Type	Description
`inputSchema`	String	A non-empty schema containing the input tables.
`inputVerticesTable`	String	Input vertices table (must have an `id` column).
`inputEdgesTable`	String	Input edges table (must have the `srcid` and `destid` columns).

Argument	Data Type	Description
`resultSchema`	String	A writable schema to create the result tables.
`resultVerticesTable`	String	Name of the vertices table to create (columns include all vertex columns and `pagerank`).
`tolerance`	numeric	The convergence threshold that stops iterations when the rank changes become negligible. After each iteration, the algorithm measures the maximum change in any rank of a vertex. If that change is below the specified tolerance, the algorithm considers the computation converged and stops early. A high tolerance value (e.g., `0.001`) is good for quick exploratory runs that generate an approximate ranking. A low tolerance value (e.g., `0.000001`) generates a precise ranking, but requires a higher compute cost.
`resetProb`	numeric	A value between `0.0` and `1.0` that controls the damping factor for how the PageRank random surfer moves across vertices. A high value (e.g., `0.85`) puts more emphasis on link structure, encouraging the algorithm to move from each vertex to a neighbor. The high value means PageRank scores tend to concentrate around well-linked regions. A lower score (e.g., `0.50`) allows the algorithm to ignore edges and instead jump to a vertex chosen from a base distribution. For standard PageRank, this base is uniform over all vertices. For personalized PageRank, the base is biased toward the specified vertex. This behavior creates more uniform scoring with less sensitivity to link topology.
`resultVerticesIndexes`	ArrayList	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`personalizationSrcId`	Long	Determines whether PageRank uses the standard or the personalized variant. For personalized mode, specify a vertex identifier. PageRank scoring starts with this vertex set at `1.0`, while all other vertices start at `0.0`. For standard mode, specify `null`. All vertices start with rank `1.0/N`, where `N` is the number of vertices.
`stmt`	java.sql.Statement	JDBC statement for executing SQL. You must have a valid statement and a database connection.

© 2026 Ocient, Inc.

Example

Run dynamic PageRank to convergence. This example uses a low `tolerance` value of `1.0e-6`, which generates high-precision rankings but requires more computing resources.

Java

```
OCGraph.dynamicPageRank(
    "sales", "customers", "purchases",
    "sales", "pagerank_dynamic",
    1.0e-6,
    0.85,
    new ArrayList<>(List.of("id")),
    null,
    stmt
);
```

Bibliography

Pregel: A System for Large-Scale Graph Processing.” Accessed November 18, 2025. **https://research.google/pubs/pregel-a-system-for-large-scale-graph-processing/**.

Related Links

JDBC Manual

OCGraph Python Library

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ocient Python Module: pyocient

The Ocient® pyocient Python® module supports direct connections to the Ocient System through a command line interface (CLI) as well as Python programs that follow the **Python Database API Specification 2.0.** To download, install, and use the pyocient CLI, see **Connect Using pyocient**.

The pyocient package has these supported features and functionality.

Prerequisites

To use the pyocient client, you should have this software with the corresponding version installed.

Software	Version
Ocient	All Ocient versions are supported.
OS	Windows®, Linux®, or macOS®. Use the latest version of each OS system.
Python	Version 3.8 or later.

Module Features

The pyocient connector supports these features as of the current version.

Unicode Support	UTF-8
Isolation Levels	Read committed. The pyocient module supports multi-statement transactions using the connection **autocommit** attribute and the **commit()** and **rollback()** methods.
Data Types	Supports all Ocient **Data Types**.
Security / Encryption	GCM, CBC, and SSO protocols are supported. TLS protocol is optional as a setting.

The pyocient module supports multi-statement transactions. By default, a connection uses autocommit mode, in which the database commits each statement immediately. To

group several statements into a single transaction, set the connection **autocommit** attribute to `False`, and then complete the transaction with **commit()** or **rollback()**.

> The HTTP Query API does not support transactions. To use transactions, connect to the Ocient System with the pyocient module or the Ocient JDBC driver.

> The Apache® Spark™ Connector supports transactions and performs the commit and rollback actions automatically. When you use the Spark Connector, you do not need to call `commit()` or `rollback()` methods.

pyocient CLI

You can make a connection by invoking the pyocient module in a command-line interface with a connection string using this syntax.

Syntax

Shell

```
$ pyocient [named_arguments] dsn [sql]
```

For details on the parts of a connection string, see these sections.

- **Connecting Via a DSN**
- **Using SQL Queries in the CLI**
- **Using Named Arguments**
- **Additional pyocient CLI Commands**

Connect Using a DSN

Required. A Data Source Name (DSN) supplies the user credentials, host, port, and database to establish a connection. This string can also include various optional parameters.

Syntax

Shell

```
ocient://<user>:<password>@<host>:[<port>][/<database>][?<parameters>[&...]]
```

A DSN consists of these arguments.

Argument	Definition
`<user>`	Required. An Ocient username.
`<password>`	Required. The password associated with your username for connection to Ocient.
`<host>`	Required. The Ocient host address. You can specify multiple hosts if they are separated by commas. The system attempts to connect to multiple hosts in order, starting with the leftmost host. For example, `host1` is the first for the connection attempt in this case: `ocient://someone:somepassword@host1,host2:4051/mydb`
`<port>`	The port number used by your host. The default is `4050`.
`<database>`	The name of the database for the connection. The default is `system`.
`<parameters>`	One or more optional parameters for the connection in the format `parameter=value`. Supported parameters include: `tls`: TLS protocol security. You can set the value to `off`, `unverified`, or `on`. The default is `off`. `force`: Values are `true` or `false`. If you set this value to `true`, this argument forces the connection to stay on this server. The default is `false`. `handshake`: Security handshake protocol. Supported values include `cbc`. By default, the system does not use `cbc`. `identityprovider`: An SSO integration established in the database. For details, see **CREATE SSO INTEGRATION**. `ssoOAuthFlow`: This parameter is only applicable if SSO authorization is enabled. Forces the driver to use either the `authorizationCode` or `deviceGrant` flow to establish a Single Sign-On session. If you do not specify this parameter, the system uses the `authorizationCode` flow when a web browser is available to the client and the `deviceGrant` flow when a web browser is not available to the client.

DSN Example

When you successfully connect with a DSN, your command line switches to interactive mode.

Shell

```
$ pyocient ocient://user:password@myhost:4050/mydb?tls=on&force=true
Ocient Database™
System Version: 22.0, Client Version 1.0.13
>
```

In interactive mode, you can enter SQL commands and queries directly at the command line. This example shows a simple `SELECT` query that you enter at the command line.

Shell

```
> SELECT * FROM my_table LIMIT 1;
[
    {
        "time_x": "2022-06-15T13:31:42.824000",
        "col_001": 1.0,
        "col_002": 2.0,
        "col_003": 3,
        "col_004": 4,
        "keyid": 5
    }
]
```

SQL Queries in the Connection String

Instead of executing queries and commands in the CLI, you can also include SQL scripts directly in your connection string.

A connection string can contain multiple queries as long as each is delineated by semicolons.

You must enclose any SQL statements in a connection string in quotes and specify the statements after the DSN portion.

Example

This example shows a SQL query embedded in a pyocient connection string. The query processes and returns its values without launching the interactive mode.

Shell

```
$ pyocient ocient://user:password@myhost:4050/mydb?tls=on&force=true "SELECT * FROM

[
    {
        "time_x": "2022-06-15T13:31:42.824000",
        "col_001": 1.0,
        "col_002": 2.0,
        "col_003": 3,
        "col_004": 4,
        "keyid": 5
    }
]
```

Use Optional Named Arguments

You can add these optional arguments to a connection string.

Shell

```
[-h] [-o OUTFILE | -n] [-c CONFIGFILE | --noconfig]
[-i INFILE] [-l {critical,error,warning,info,debug}]
[--logfile LOGFILE] [-t] [--format {json | table | csv}]
[--nocolor]
```

Named Argument Definitions

Named Argument	Definition
`-o | --outfile OUTFILE`	pyocient records all SQL output to a specified output file. You must also include the file path to the output file, enclosed in quotation marks. The Ocient System saves output values to the file after query completion.
`-n | --noout`	Prevent all output results.
`-c | --configfile CONFIGFILE`	Direct pyocient to use a configuration file. You must include the path location and name of the configuration file, enclosed in quotation marks.
`--noconfig`	Direct pyocient not to use a configuration file.

Named Argument	Definition
`-i | --infile INFILE`	Direct pyocient to use an input file that contains SQL statements.
`-l | --loglevel LOGLEVEL`	Sets the logging level. Logging level options include: `CRITICAL | ERROR | WARNING | INFO | DEBUG` Defaults to `CRITICAL`. For details, see **Log Monitoring**.
`--logfile LOGFILE`	Direct pyocient to record log events in a specified log file. You must include the path location and name of the log file, enclosed in quotation marks.
`-t, --time`	pyocient outputs the execution time for all queries and commands.
`-f, --format`	Specifies the output format for all SQL queries. Format options include: `JSON | TABLE | CSV` `JSON` — JavaScript® Object Notation. `TABLE` — A table rendered in ASCII characters. `CSV` — Comma-separated values format. The default value is `JSON`.
`--nocolor`	When you use pyocient in interactive mode, the interface does not color SQL keywords.

Additional pyocient CLI Commands

When running in a CLI, pyocient recognizes these commands in addition to the SQL statements and queries.

Command	Description
`CONNECT TO <dsn> USER <username> USING <password>;`	Allows switching to a different DSN or user during an interactive session. The DSN must be enclosed in quotation marks. Otherwise, the command follows the normal format explained in **Connect Using a DSN**. You can specify the username and password either in the DSN string or with the `USER` and `USING` keywords.
`SOURCE '<file>';`	Executes the input statements from the specified file. The filename must be in single quotes.
`SET FORMAT TABLE;`	Sets the output format. Format options include `JSON`, `TABLE`, and `CSV`. Defaults to `JSON`. `JSON` — JavaScript Object Notation. `TABLE` — A table rendered in ASCII characters. `CSV` — Comma-separated values format.

© 2026 Ocient, Inc.

Command	Description
`QUIT;`	Quits the CLI. Alternatively, you can exit the interface by using `Ctrl + D`.

Run a Transaction in the pyocient CLI

The pyocient CLI runs two scripts on a single connection with the `-i` argument. To run multiple statements in one transaction, use the `SET AUTOCOMMIT OFF` statement, and then end the transaction with the `COMMIT` or `ROLLBACK` statement. Because each invocation uses one connection, include all statements of the transaction in a single input file.

The `txn_commit.sql` example script inserts multiple rows into the `public.txn_demo` table, confirms the row count with a query that reads its own uncommitted writes, and then commits the transaction.

SQL

```sql
SET AUTOCOMMIT OFF;
INSERT INTO public.txn_demo VALUES (1, 'alpha');
INSERT INTO public.txn_demo VALUES (2, 'beta');
SELECT count(*) AS n_in_txn FROM public.txn_demo;
COMMIT;
```

The `txn_rollback.sql` example script inserts multiple rows into the `public.txn_demo` table, confirms the row count, and then rolls back the transaction, discarding the inserted rows. The example confirms the discard using another row-count query.

SQL

```sql
SET AUTOCOMMIT OFF;
INSERT INTO public.txn_demo VALUES (3, 'gamma');
INSERT INTO public.txn_demo VALUES (4, 'delta');
SELECT count(*) AS n_in_txn FROM public.txn_demo;
ROLLBACK;
SELECT count(*) AS n_after_rollback FROM public.txn_demo;
```

Run each script with the `-i` argument.

Shell

```
pyocient "ocient://user:password@myhost:4050/mydb?tls=unverified" -i txn_commit.sql
pyocient "ocient://user:password@myhost:4050/mydb?tls=unverified" -i txn_rollback.s
```

pyocient API

This Python database API conforms to the **Python Database API Specification 2.0**, and you can use it to access the Ocient System.

You can also execute this module as a main function, in which case it acts as a primitive CLI for the database.

When you execute this module as a main function, the API takes a connection string in the DSN format. The connection string can also include one or more query strings that execute. For information on using a DSN to connect, see **Connect Using a DSN**.

The API returns output in JSON format by default.

The database returns any warnings and sends them to the Python warnings module. By default, that module sends warnings to standard output, however, you can change the behavior by using that module.

pyocient.Connection

Constructor

Python

```
class pyocient.Connection(
    dsn=None, user=None, password=None,
    host=None, database=None, tls=None,
    handshake=None, identityprovider=None,
    force=None, configfile=None, session=None)
```

A connection to Ocient. This class is normally constructed by executing the module `connect()`, but you can also construct it directly.

You can specify connection parameters using the `dsn` argument, other keyword arguments, or a mix of both. If the system receives multiple arguments for the same parameters, the keyword arguments take precedence and override the `dsn` value.

Definitions for the connection parameters are listed in **Connect Using a DSN section**. Other parameters are:

- `configfile` — The name of a configuration file in .ini format, where each section either uses default connection settings or a pattern that matches the host or database. Sections are matched in order, so highly detailed sections should precede less detailed sections.
- `session` — A string of an SSO security token.

This code provides an example of an `.ini` configuration file.

INI

```ini
[DEFAULT]
tls = unverified

; This matches the specific host and database.
[foo.ocient.com/somedb]
user = panther
password = pink

; This matches the specific host.
[foo.ocient.com]
user = panther
password = pink

; This matches any host in the ocient.com domain.
[*.ocient.com]
user = tom
password = jerry
database = mice

; This matches any host in the ocient.com domain using the default database.
[*.ocient.com]
user = tom
password = jerry
```

The currently supported parameter is:

`tls` — The values are `on`, `off`, or `unverified` in the DSN. You can also set TLS by using `Connection.TLS_NONE`, `Connection.TLS_UNVERIFIED`, or `Connection.TLS_ON` as a keyword parameter.

Connection Object Methods

autocommit

Controls the autocommit mode for the connection. When you set this method to `True`, the database commits each statement immediately. This mode is the default value. When you set this attribute to `False`, the database groups statements into a transaction that you complete by calling **commit()** or **rollback()**.

When you change the value from `False` to `True`, the database commits any pending transaction.

This example disables autocommit mode for a connection.

Python

```
connection.autocommit = False
```

close()

Close the connection. Any subsequent queries on the closed connection fail.

commit()

Commit the pending transaction and permanently save all changes made after the transaction started. This method has an effect only when the `autocommit` method value is `False`.

cursor()

Return a new cursor object for this connection.

rollback()

Roll back the pending transaction and discard all changes made after the transaction started. This method has an effect only when the `autocommit` method value is `False`.

pyocient.Cursor

Constructor

```python
class pyocient.Cursor(conn)
```

A database cursor that can manage a query and its returned results.

Cursors are normally created by executing `cursor()` on a connection, but you can also create them directly by providing a connection.

Cursor Object Methods

close()

Close this cursor. The current result set is closed, but you can reuse the cursor for subsequent `execute()` calls.

execute(operation, parameters=None) [#execute]

Prepare and execute a database operation (query or command).

Parameters might be provided as a mapping and are bound to variables in the operation. Variables are specified in Python extended format codes, for example: `WHERE name=%(name)`

executemany(operation, parameterlist)

Prepare a database operation (query or command) and then execute it against all parameter sequences or mappings found in the sequence `parameterlist`.

Parameters might be provided as a mapping and are bound to variables in the operation. Variables are specified in Python extended format codes, for example: `WHERE name=%(name)`

© 2026 Ocient, Inc.

fetchone()

Fetch the next row of a query result set. The function returns a single row or `None` when no more data rows are available.

fetchmany(size=None) [#fetchmany]

Fetch the next set of rows of a query result and return a sequence of sequences as a list of tuples. The database returns an empty sequence when no more rows are available.

The number of rows to fetch per call is specified by the `size` parameter. If this parameter is not specified, the array size of the cursor determines the number of rows to fetch. The method tries to fetch as many rows as indicated by the size parameter. If this is not possible due to the specified number of rows not being available, the function returns fewer rows.

fetchall()

Fetch all (remaining) rows of a query result and return them as a sequence of sequences (e.g., a list of tuples). Note that the array size attribute of the cursor can affect the performance of this operation.

arraysize

This read-and-write attribute specifies the number of rows to fetch at a time with **fetchmany**. The default value is `1`, meaning the attribute fetches a single row at a time.

tables(schema='%', table='%')

Retrieve the database tables. By default, the database returns all tables.

system_tables(table='%')

Retrieve the database system tables.

views(view='%')

Retrieve the database views.

columns(schema='%', table='%', column='%')

Retrieve the database columns.

indexes(schema='%', table='%')

Retrieve the database indexes.

getTypeInfo()

Retrieve the database type information.

fetchval()

The fetchval() convenience method returns the first column of the first row if there are results, otherwise it returns `None`.

resolve_new_endpoint(new_host, new_port)

Handles mapping to a secondary interface based on the secondary interface mapping saved on this connection.

Arguments:

```
Text
new_host[string]: the new host to be remapped
new_port[int]: the new port to be remapped
```

Returns:

```
Text
[tuple(string, int)]: The actual endpoint to connect to in the format: (host, port)
```

redirect(new_host, new_port)

Redirects to the proper secondary interface with a new endpoint.

Arguments:

© 2026 Ocient, Inc.

```
new_host[string]: the new host to be redirected
new_port[int]: the new port to be redirected
```

Returns:

```
[Connection]: A new connection.
```

refresh()

Refresh the session associated with this connection. The server returns a new server session identifier and security token.

setinputsizes(sizes)

Used before an **execute** call to set memory sizes for the specified operation.

For details on usage, see the **Python API documentation**.

setoutputsize(size [, column])

Used before an **execute** call to set a column buffer size for fetches of columns with large data types, such as LONG, BLOB, etc.

For details on usage, see the **Python API documentation**.

Type Objects and Constructors

These constructors create objects to hold special values to comply with defined types in a database schema. When you pass these objects to the cursor methods, the module detects the proper type of input parameter and binds it accordingly.

For details, see the Python documentation on **Type Objects and Constructors**.

Date(*year*, *month*, *day*)

This function constructs an object that holds a date value.

Time(*hour*, *minute*, *second*)

This function constructs an object that holds a time value.

Timestamp(*year*, *month*, *day*, *hour*, *minute*, *second*)

This function constructs an object that holds a timestamp value.

DateFromTicks(*ticks*)

This function constructs an object that holds a date value from the specified ticks value (number of seconds after the epoch; see the documentation of **the standard Python time module** for details).

TimeFromTicks(*ticks*)

This function constructs an object that holds a time value from the specified ticks value (number of seconds after the epoch; see the documentation of the standard Python time module for details).

TimestampFromTicks(*ticks*)

This function constructs an object that holds a timestamp value from the specified ticks value (number of seconds after the epoch; see the documentation of the standard Python time module for details).

Binary(*string*)

This function constructs an object capable of holding a binary (long) string value.

STRING type

This type object describes columns in a database that are string-based (e.g., `CHAR`).

BINARY type

This type object describes (long) binary columns in a database (e.g., `LONG`, `RAW`, `BLOB`).

NUMBER type

This type object describes numeric columns in a database.

DATETIME type

This type object describes date and time columns in a database.

ROWID type

This type object describes the “Row ID” column in a database.

Exceptions

Error

Exception that is the base class of all other error exceptions.

```
Text
exception Error(reason, sql_state='58005', vendor_code=-100)
```

Warning

Exception that is the base class of all other warning exceptions.

```
Text
exception Warning(reason='', sql_state='00000', vendor_code=0)
```

InterfaceError

Exception raised for errors related to the database interface rather than the database itself.

```
Text
exception InterfaceError(reason, sql_state='58005', vendor_code=-100)
```

DatabaseError

Exception raised for errors that are related to the database.

Text

```
exception DatabaseError(reason, sql_state='58005', vendor_code=-100)
```

InternalError

Exception raised when the database encounters an internal error, e.g., the cursor is not valid anymore.

Text

```
exception InternalError(reason, sql_state='58005', vendor_code=-100)
```

OperationalError

Exception raised for errors related to the operation of the database and not necessarily under the control of the programmer, e.g., an unexpected disconnect occurs, the data source name is not found, etc.

Text

```
exception OperationalError(reason, sql_state='58005', vendor_code=-100)
```

ProgrammingError

Exception raised for programming errors, e.g., table not found, syntax error in the SQL statement, wrong number of parameters specified, etc.

Text

```
exception ProgrammingError(reason, sql_state='58005', vendor_code=-100)
```

IntegrityError

Exception raised when the relational integrity of the database is affected, e.g., a foreign key check fails.

© 2026 Ocient, Inc.

```
exception IntegrityError(reason, sql_state='58005', vendor_code=-100)
```

DataError

Exception raised for errors due to problems with the processed data like division by zero, numeric value out of range, etc.

```
exception DataError(reason, sql_state='58005', vendor_code=-100)
```

NotSupportedError

Exception raised when a method or database API that is not supported by the database is used.

```
exception NotSupportedError(reason, sql_state='58005', vendor_code=-100)
```

Optional DB API Extensions

Cursor.rownumber

This read-only attribute should provide the current 0-based index of the cursor in the result set or `None` if the index cannot be determined.

You can see the index as the index of the cursor in a sequence (the result set). The next fetch operation fetches the row indexed by **.rownumber** in that sequence.

Connection.Error, Connection.ProgrammingError, etc.

All exception classes defined by the DB API standard should be exposed on the **Connection** objects as attributes (in addition to being available at module scope).

These attributes simplify error handling in multi-connection environments.

© 2026 Ocient, Inc.

Cursor.connection

This read-only attribute returns a reference to the **Connection** object on which the cursor was created.

The attribute simplifies writing polymorph code in multi-connection environments.

Cursor.__iter__()

Return self to make cursors compatible with the Python iteration protocol.

Transaction pyocient API Example

To run multiple statements in a single transaction with the pyocient API, set the connection **autocommit** attribute to `False`, and then call the **commit()** or **rollback()** methods. For transactions, autocommit mode is on by default. Ensure that you execute SQL statements that are supported by transactions. Otherwise, the database throws an error. For a list of supported statements, see **Transactions**.

The example code establishes a connection and creates the `txn_demo` table. Then, the example inserts two rows, commits the transaction, and verifies the row count. This example continues to insert two more rows and verify the row count. The code then calls the rollback method and verifies the row count to ensure that the rollback action is successful. The example resets the autocommit mode for other transactions.

© 2026 Ocient, Inc.

Python

```python
import pyocient

conn = pyocient.connect(
    host="myhost:4050",
    user="user", password="password",
    database="mydb", tls="unverified",
)

def count(label):
    cur = conn.cursor()
    cur.execute("SELECT count(*) AS n FROM public.txn_demo")
    print(f"{label} -> {cur.fetchone()[0]}")
    cur.close()

# DDL must run outside a transaction (autocommit is on by default)
cur = conn.cursor()
cur.execute("DROP TABLE IF EXISTS public.txn_demo")
cur.execute("CREATE TABLE public.txn_demo (id INT, v VARCHAR(64))")
cur.close()

# Transaction 1: execute multiple inserts, then commit
conn.autocommit = False
cur = conn.cursor()
cur.execute("INSERT INTO public.txn_demo VALUES (1, 'alpha')")
cur.execute("INSERT INTO public.txn_demo VALUES (2, 'beta')")
count("T1 in-txn (read-your-writes, expect 2)")
conn.commit()
count("T1 after commit() (expect 2)")

# Transaction 2: execute multiple inserts, then rollback
cur.execute("INSERT INTO public.txn_demo VALUES (3, 'gamma')")
cur.execute("INSERT INTO public.txn_demo VALUES (4, 'delta')")
count("T2 in-txn (expect 4)")
conn.rollback()
count("T2 after rollback() (expect 2)")

conn.autocommit = True   # commits any open work
conn.close()
```

Related Links

pyocient PyPI

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

OCGraph Python Library

The `ocient_graph` module brings a programming model (similar to Apache® Spark™ GraphX) to the Ocient® System directly from Python® using the `pyocient` driver. The module treats a graph as two relational tables, one for vertices (nodes) and one for edges (directed links). This module provides a composable API for graph transformations, neighborhood analytics, and iterative algorithms (e.g., Pregel, PageRank).

The API validates inputs, avoids destructive changes by materializing results into new tables, supports optional indexing for performance, and follows Ocient SQL conventions. The package installs separately from `pyocient` and exposes a Python-native interface that mirrors the Java® library. For details, see **OCGraph Java Library**.

Installation

Use `pyocient` for connectivity and `ocient_graph` for graph APIs. The graph library is a separate package that depends on `pyocient`. For a tutorial about installing and using `pyocient`, see **Ocient Python Module: pyocient**.

Install and Import

Install the `ocient_graph` module.

Shell

```
pip install ocient_graph
```

Import the module.

Python

```
from pyocient import connect
from ocient_graph import (
    subgraph,
    collect_neighbors,
    EdgeDirection,
)
```

© 2026 Ocient, Inc.

Data Model Requirements

Database tables that use the OCGraph Python library must adhere to this structure. In addition to the listed requirements, tables can include other columns.

Table	Description	Requirements
Vertices table	A table with one row per vertex (node). This table typically represents the anchor for graph algorithms and transforms. Many methods join edges to vertices by the `id` column.	The table must contain the `id BIGINT NOT NULL` column definition as the unique vertex identifier.
Edges table	A table with one row per directed edge (relationship). Each row is a directed edge from a source vertex to a destination vertex.	The table must contain these column definitions: `srcid BIGINT NOT NULL`, `destid BIGINT NOT NULL`

Subgraph and Filtering

Use a subgraph or various filters to restrict a graph to relevant vertices and edges. These functions create filtered copies or masked intersections, preserving schema and optional indexes for performance.

subgraph

Creates filtered vertex and edge tables using vertex and triplet predicates, retaining only edges with endpoints that remain after vertex filtering. The function creates the requested indexes and performs best-effort cleanup in the event of failure.

Syntax

© 2026 Ocient, Inc.

Python

```
subgraph(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    result_edges_table,
    vertex_filter,
    edge_filter,
    [ result_vertices_indexes [ , ... ] ],
    [ result_edges_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the filtered vertices table to create.
`result_edges_table`	str	Name of the filtered edges table to create.
`vertex_filter`	str	SQL predicate to filter the vertices (without the `WHERE` keyword). Example: `status = 'ACTIVE' AND score > 0`
`edge_filter`	str	SQL predicate that the system evaluates in a triplet context using the aliases `a` (source vertex), `b` (edge), and `c` (destination vertex). This predicate does not require a `WHERE` keyword. Example: `b.amount > 50 AND a.region = c.region`
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Argument	Data Type	Description
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example Create an active customer subgraph that includes only purchases exceeding $50 where the source and destination share a region.

Python

```
subgraph(
    connection,
    "sales",
    "customers",
    "purchases",
    "sales",
    "customers_active",
    "purchases_active",
    "status = 'ACTIVE' AND score > 0",
    "b.amount > 50 AND a.region = c.region",
    ["id","region"],
    ["srcid","destid"],
)
```

filter_vertices

Creates a filtered subgraph by selecting vertices that match a predicate while retaining only edges with endpoints that are in the filtered vertex set.

Syntax

Python

```
filter_vertices(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    result_edges_table,
    vertex_filter,
    [ result_vertices_indexes [ , ... ] ],
    [ result_edges_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the result vertices table. The name must not conflict with the names of input tables.
`result_edges_table`	str	Name of the edges table to create. The name must not conflict with the names of input tables.
`vertex_filter`	str	SQL predicate to filter the vertices (without the `WHERE` keyword). Example: `status = 'ACTIVE' AND score > 0`
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example Filter US customers and retain edges with endpoints that remain in the filtered vertex set.

Python

```
filter_vertices(
    connection,
    "sales", "customers", "purchases",
    "sales", "customers_us", "purchases_us",
    "country = 'US'",
    ["id"],
    ["srcid","destid"],
)
```

filter_edges

Creates a filtered edges table by selecting edges that match a predicate.

Syntax

Python

```
filter_edges(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_edges_table,
    edge_filter,
    [ result_edges_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty achema containing the input edges table.
`input_edges_table`	str	Input edges table (must have `srcid` and `destid` columns).

Argument	Data Type	Description
`result_schema`	str	A writable schema to create the result tables.
`result_edges_table`	str	Result edges table name. The name must not conflict with the names of input tables.
`edge_filter`	str	SQL predicate on edges (without the `WHERE` keyword). Example: `weight > 0.5 AND type = 'ACTIVE'`
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example This example demonstrates how to create a filtered edges table from an existing `purchases` edge set by keeping only edges that meet a business rule (`weight > 0.5` and `ACTIVE`). Then, the `filterEdges` method indexes the result on the `srcid` and `destid` columns for faster lookups.

Python

```
filter_edges(
    connection,
    "sales", "purchases",
    "sales", "purchases_filtered",
    "weight > 0.5 AND type = 'ACTIVE'",
    ["srcid", "destid"],
)
```

mask

Creates a masked subgraph by intersecting two graphs. Vertices intersect if the vertex identifier is present in both graphs. Edges intersect when the `srcid` and `destid` values are present in both graphs.

The function creates a masked subgraph from rows that intersect with each other. The function copies rows that intersect from the graph defined by the arguments `input_vertices_table` and `input_edges_table`, including any attributes.

You can optionally create indexes on the result subgraph tables.

Syntax

Java

```
mask(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    other_schema,
    other_vertices_table,
    other_edges_table,
    result_schema,
    result_vertices_table,
    result_edges_table,
    [ result_vertices_indexes [ , ... ] ],
    [ result_edges_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	The primary vertices table to intersect (must have an `id` column). The masked subgraph created by this function copies rows that intersect from this table.
`input_edges_table`	str	The primary edges table to intersect (must have `srcid` and `destid` columns). The masked subgraph created by this function copies rows that intersect from this table.
`other_schema`	str	Schema containing the second graph.
`other_vertices_table`	str	The second vertices table to intersect.
`other_edges_table`	str	The second edges table to intersect.
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create.

Argument	Data Type	Description
`result_edges_table`	str	Name of the edges table to create.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example

Create a masked subgraph by intersecting two graphs. The example copies vertices and edges that are present in both graphs, along with the remaining endpoints.

Java

```
mask(
    connection,
    "sales", "customers", "purchases",
    "ref",   "customers_ref", "purchases_ref",
    "sales", "customers_masked", "purchases_masked",
    ["id"],
    ["srcid", "destid"],
)
```

Transformations

Construct new vertex or edge tables by computing derived columns, reversing direction, or aggregating duplicates. These functions do not change the original inputs. Instead, the functions materialize new results.

map_vertices

Creates a new vertices table with the identifier `id` and computed columns. Use the `result_column_expressions` argument to calculate additional columns. This function can also add indexes before inserting data.

Syntax

Python

```
map_vertices(
    connection,
    input_schema,
    input_vertices_table,
    result_schema,
    result_vertices_table,
    result_column_expressions,
    [ result_vertices_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input vertices table.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`result_schema`	str	A schema to create the result vertices table.
`result_vertices_table`	str	Name of the result vertices table. The name must not conflict with the names of input tables.
`result_column_expressions`	List[str]	One or more SQL expressions defining result columns beyond `id`. Use the `AS alias_name` keyword for stable names.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Create a new vertices table with two new columns, `name_upper` and `is_vip`, and generate indexes for the `id` and `name_upper` columns.

© 2026 Ocient, Inc.

Python

```
map_vertices(
    connection,
    "sales", "customers",
    "sales", "customers_enriched",
    [
      "UPPER(name) AS name_upper",
      "CASE WHEN score > 1000 THEN true ELSE false END AS is_vip",
    ],
    ["id","name_upper"],
)
```

map_edges

Creates a new edges table with `srcid`, `destid`, and any additional computed columns. Expressions should refer to input edge columns by their original names, and each computed expression should include an `AS alias` keyword.

Syntax

Python

```
map_edges(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_edges_table,
    [ result_column_expressions [, ... ] ],
    [ result_edges_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input edges table.
`input_edges_table`	str	Input edges table (must have `srcid` and `destid` columns).

Argument	Data Type	Description
`result_schema`	str	A schema to create the result edges table.
`result_edges_table`	str	Name of the result edges table. The name must not conflict with the names of input tables.
`result_column_expressions`	List[str]	Optional. SQL expressions for additional edge columns (besides the `srcid` and `destid` columns). Use the `AS alias_name` keyword for stable names.
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example

Create a new edges table with two columns, `discounted_amount` and `big_txn`, and generate indexes for the `srcid` and `destid` columns.

Python

```
map_edges(
    connection,
    "sales", "purchases",
    "sales", "purchases_enriched",
    [
      "amount * 0.9 AS discounted_amount",
      "CASE WHEN amount > 100 THEN 1 ELSE 0 END AS big_txn",
    ],
    ["srcid","destid"],
)
```

map_triplets

Creates a new edges table with computed columns that reference `a` (source vertex), `b` (edge), and `c` (destination vertex). The output automatically includes `b.srcid` and `b.destid` columns.

Syntax

Python

```
map_triplets(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_edges_table,
    [ result_column_expressions [, ... ] ],
    [ result_edges_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column). Used as the `a` and `c` vertices.
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns). This function uses this argument as the `b` edge.
`result_schema`	str	A writable schema to create the result edges table.
`result_edges_table`	str	Name of the result edges table. The name must not conflict with the names of input tables.
`result_column_expressions`	List[str]	Optional. SQL expressions that can reference `a` (source vertex), `b` (edge), and `c` (destination vertex). Use the `AS alias_name` keyword for stable names.
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example Create a new triplet table from the vertices and edges tables with the `amount` and `same_country` columns, and generate indexes for the `src` and `destid` columns.

Python

```
map_triplets(
    connection,
    "sales", "customers", "purchases",
    "sales", "purchases_triplets",
    [
      "b.amount AS amount",
      "CASE WHEN a.country = c.country THEN 1 ELSE 0 END AS same_country",
    ],
    ["srcid","destid"],
)
```

reverse_edges

Creates a new edges table with the `srcid` and `destid` columns reversed, preserving other columns. Use this function to traverse a graph in the opposite direction.

Syntax

Python

```
reverse_edges(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_edges_table
    [ result_edges_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input edges table.
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the reversed edges table.

Argument	Data Type	Description
`result_edges_table`	str	Name of the reversed edges table to create. The name must not conflict with the names of input tables.
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example Transform edge direction by reversing the `srcid` and `destid` columns. The example also creates indexes for these columns.

Python

```
reverse_edges(
    connection,
    "sales", "purchases",
    "sales", "purchases_reversed",
    ["srcid","destid"],
)
```

group_edges

Groups duplicate rows of the `srcid` and `destid` columns, producing one row for each unique pair of values in a new edges table. This function performs aggregations based on one or more SQL expressions.

Syntax

Python

```
group_edges(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_edges_table,
    [ result_column_expressions [, ... ] ],
    [ result_edges_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input edges table.
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the grouped tables.
`result_edges_table`	str	Name of the grouped edges table. The name must not conflict with the names of input tables.
`result_column_expressions`	List[str]	One or more aggregate expressions for the `srcid` and `destid` columns. For details about SQL aggregations, see **Aggregate Functions**. Use the `AS alias_name` keyword for stable names.
`result_edges_indexes`	List[str]	Optional. Columns to index in the result edges table (e.g., `srcid` and `destid` columns). Specify an empty list for none.

Example Create a new edge table that includes SQL aggregations for counting unique transactions `txn_count` and total sums `total_amount`. Also, this function generates indexes for the `src` and `destid` columns.

Python

```
group_edges(
    connection,
    "sales", "purchases",
    "sales", "purchases_grouped",
    [
      "COUNT(*) AS txn_count",
      "SUM(amount) AS total_amount",
    ],
    ["srcid","destid"],
)
```

Triplets

Produce triplet representations that are made of `a` (source vertex), `b` (edge), and `c` (destination vertex), either as a logical view or a materialized table for downstream queries.

create_triplets_view

Creates a view that combines the edge table with the source and destination vertex attributes. This view is useful for analyzing relationships without having to repeatedly join tables. The view includes these columns:

- All original edge columns (including the `srcid` and `destid` columns).
- All source-vertex columns except `id`. Source-vertex column names have the `src_` prefix.
- All destination-vertex columns except `id`. Destination-vertex column names have the `dest_` prefix.

Use the **create_triplets_table** function instead if you want to create a materialized table with indexes instead of a view.

Syntax

Python

```
create_triplets_view(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_triplets_view
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input vertices and edges tables.

Argument	Data Type	Description
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A schema to create the view.
`result_triplets_view`	str	Name of the triplets view to create. The name must not conflict with the names of input tables.

Example Create a triplets view to inspect edges with joined source and destination vertex attributes.

Python

```
create_triplets_view(
    connection,
    "sales", "customers", "purchases",
    "sales", "triplets_v",
)
```

create_triplets_table

Creates a materialized table that combines the edge table with the source and destination vertex attributes. This table is useful for analyzing relationships without having to repeatedly join tables. The created table includes these columns:

- All original edge columns (including the `srcid` and `destid` columns).
- All source-vertex columns except `id`. Source-vertex column names have the `src_` prefix.
- All destination-vertex columns except `id`. Destination-vertex column names have the `dest_` prefix.

Use the **create_triplets_view** function if you want to create a view instead of a new table.

Syntax

Python

```
create_triplets_table(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_triplets_table,
    [ result_triplets_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A schema to create the triplets table.
`result_triplets_table`	str	Name of the triplets table to create. The name must not conflict with any input names.
`result_triplets_indexes`	List[str]	Optional. Columns to index in the result table. Specify an empty list for none.

Example Create a new table for triplets. Generate indexes for the `src_id` and `dest_id` columns.

Python

```
create_triplets_table(
    connection,
    "sales", "customers", "purchases",
    "sales", "triplets_t",
    ["srcid","destid"],
)
```

Degrees

Compute degree metrics for each vertex from the edges table. These functions produce small vertex tables suitable for joins and analytics.

in_degrees

Computes how many edges point to each vertex in an edge table by counting how many times each unique `destid` value appears. The result table has two columns: `id` (the destination vertex) and `in_degree` (the count).

Syntax

Python

```
in_degrees(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_vertices_table,
    [ result_vertices_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input table.
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A schema to create the in-degree table.
`result_vertices_table`	str	Name of the vertices table to create (columns include `id` and `in_degree`).
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Compute in-degrees per vertex and generate an index on the `id` column.

Python

```
in_degrees(
    connection,
    "sales", "purchases",
    "sales", "customers_in_degree",
    ["id"],
)
```

out_degrees

Computes how many edges originate from each vertex in an edge table by counting how many times each unique `srcid` value appears. The result table has two columns: `id` (the source vertex) and `out_degree` (the count).

Syntax

Python

```
out_degrees(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_vertices_table,
    [ result_vertices_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input table.
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A schema to create the out-degree table.
`result_vertices_table`	str	Name of the vertices table to create (columns include `id` and `out_degree`).

Argument	Data Type	Description
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Compute the out-degrees count for each vertex and generate an index on the `id` column.

Python

```
out_degrees(
    connection,
    "sales", "purchases",
    "sales", "customers_out_degree",
    ["id"],
)
```

degrees

Computes the total degrees (in-degrees and out-degrees) for each vertex in an edge table by counting how many times each unique `srcid` and `destid` value appears. The result table has two columns: `id` (the destination or source vertex) and `degree` (the count).

Syntax

Python

```
degrees(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_vertices_table,
    [ result_vertices_indexes [, ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input table.
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A schema to create the degree table.
`result_vertices_table`	str	Name of the vertices table to create (columns include `id` and `out_degree`).
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Compute total degrees for each vertex and generate an index on the `id` column.

Python

```
degrees(
    connection,
    "sales", "purchases",
    "sales", "customers_degree",
    ["id"],
)
```

Vertex Extraction and Joins

Build vertex sets from edges and combine vertex attributes across tables. These functions are useful for shaping vertex properties and consolidating features.

from_edges

Builds a vertices table from an edges table by extracting the unique source and destination identifiers.

This function can optionally compute additional columns using SQL expressions by referencing the unique identifier as `ids.id`.

The created table always contains the `id` column with one additional column per expression.

Syntax

Python

```
from_edges(
    connection,
    input_schema,
    input_edges_table,
    result_schema,
    result_vertices_table,
    [ result_column_expressions [ , ... ] ],
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input table.
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A schema to create the result vertices table.
`result_vertices_table`	str	Name of the result vertices table (always includes `id`).
`result_column_expressions`	List[str]	Optional. Specify a list of SQL expressions that reference `ids.id` to add additional columns.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Create a vertices table from edge endpoints and add a `bucket` column that assigns each vertex to one of 10 buckets. Generate an index for the `id` and `bucket` columns.

© 2026 Ocient, Inc.

Python

```
from_edges(
    connection,
    "sales", "purchases",
    "sales", "customers_from_edges",
    ["ids.id % 10 AS bucket"],
    ["id","bucket"],
)
```

join_vertices

Merges two vertices tables by retaining every row from a primary table (`input_vertices_table`) and selectively updating rows that also appear in the modification table (`modification_vertices_table`). The merged table includes all vertices from the primary table that do not appear in the modification table.

For vertices that appear in both tables, the function must include a list of expressions (`resultAttributeExpressions`) in the same column order for every non-identifier column in the merged result table. These SQL expressions can add computations to columns, or simply add aliases if no changes are needed. Each expression can reference columns from the primary table (using alias `a`) or from the modification table (using alias `b`).

Syntax

Python

```
join_vertices(
    connection,
    input_schema,
    input_vertices_table,
    modification_schema,
    modification_vertices_table,
    result_schema,
    result_vertices_table,
    result_attribute_expressions [ , ... ],
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the primary input table.
`input_vertices_table`	str	Input vertices table (with the alias `a`). This table must have an `id` column.
`modification_schema`	str	A schema for the modification vertices table.
`modification_vertices_table`	str	Modification vertices table (with the alias `b`). This table must have an `id` column.
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create.
`result_attribute_expressions`	List[str]	A list of SQL expressions that define the non-identifier columns of the joined result. Each expression can reference the left (`input_vertices_table`) vertex as `a` and the right (`modification_vertices_table`) vertex as `b`. You must end every expression with an explicit alias using `AS alias_name` so the output column names are stable.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Merge vertex attributes and generate indexes for the `id` and `status` columns. This example includes two SQL expressions to update the `status` and `score` columns based on the modification vertex table using the `COALESCE` SQL reference function.

Python

```
join_vertices(
    connection,
    "sales", "customers",
    "sales", "customers_updates",
    "sales", "customers_merged",
    [
      "COALESCE(b.new_status, a.status) AS status",
      "COALESCE(b.score_delta + a.score, a.score) AS score",
    ],
    ["id","status"],
)
```

inner_join_vertices

Performs an inner join on two vertex tables using an equality comparison `a.id = b.id`. The result table automatically includes the `id` column from the first table.

The function must include a list of SQL expressions (`result_attribute_expressions`) in the same column order for every non-identifier column in the merged result table. These SQL expressions can add computations to columns, or simply add aliases if no changes are needed. Each expression can reference columns from the primary table using the alias `a` or from the modification table using the alias `b`.

Syntax

Python

```
inner_join_vertices(
    connection,
    input_schema,
    input_vertices_table,
    other_schema,
    other_vertices_table,
    result_schema,
    result_vertices_table,
    result_attribute_expressions [ , ... ],
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A schema for the left vertices table.
`input_vertices_table`	str	Input vertices table for the left side of the join (must have an `id` column).
`other_schema`	str	A schema for the right vertices table.
`other_vertices_table`	str	The other vertices table for the right side of the join (must have an `id` column).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create.
`result_attribute_expressions`	List[str]	A list of SQL expressions that define the non-identifier columns of the joined result. Each expression can reference the left (`input_vertices_table`) vertex as `a` and the right (`other_vertices_table`) vertex as `b` . You must end every expression with an explicit alias using `AS alias_name` to ensure the output column names are stable.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Create an inner join between two vertex tables and generate an index on the `id` column.

Python

```
inner_join_vertices(
    connection,
    "sales", "customers",
    "sales", "profiles",
    "sales", "customers_joined",
    [
      "a.id AS id",
      "a.status AS status",
      "b.tier AS tier",
    ],
    ["id"],
)
```

outer_join_vertices

Performs a left outer join between two vertices tables using an equality comparison `a.id = b.id`. The result table includes all rows from the left table. For left-table rows that have no match in the right table, any expression that reads columns from the right table with the alias `b` evaluates to NULL (while expressions that only read the table with the alias `a` remain non-NULL as usual).

The method must include a list of SQL expressions (`result_attribute_expressions`) in the same column order for every non-identifier column in the merged result table. These SQL expressions can add computations to columns, or simply add aliases if no changes are needed. Each expression can reference columns from the primary table using the alias `a` or from the modification table using the alias `b`.

Syntax

© 2026 Ocient, Inc.

Python

```
outer_join_vertices(
    connection,
    input_schema,
    input_vertices_table,
    other_schema,
    other_vertices_table,
    result_schema,
    result_vertices_table,
    result_attribute_expressions [ , ... ],
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A schema for the left vertices table.
`input_vertices_table`	str	Left vertices table (must have an `id` column).
`other_schema`	str	A schema for the right vertices table (with the alias `b`).
`other_vertices_table`	str	Right vertices table (must have an `id` column).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create.
`result_attribute_expressions`	List[str]	A list of SQL expressions that define the non-identifier columns of the joined result. Each expression can reference the left (`input_vertices_table`) vertex as `a` and the right (`other_vertices_table`) vertex as `b`. You must end every expression with an explicit alias using `AS alias_name` to ensure the output column names are stable.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Perform a left outer join on two vertices tables and generate an index on the `id` column.

Python

```
outer_join_vertices(
    connection,
    "sales", "customers",
    "sales", "profiles",
    "sales", "customers_joined",
    [
      "a.id AS id",
      "a.status AS status",
      "b.tier AS tier",
    ],
    ["id"],
)
```

collect_neighbors

For each vertex in a table, this function collects information on neighbors (identifier and any attributes) as an array of tuples. For a specified direction (`IN` , `OUT` , or `BOTH`), the function aggregates tuples representing each neighboring vertex into an array.

The direction types are:

- `IN` — Neighbors with edges pointing to the vertex (edges where `destid = id`).
- `OUT` — Neighbors that the vertex points to (edges where `srcid = id`).
- `BOTH` — Union of `IN` and `OUT` with neighbors from incoming (`destid = id`) and outgoing (`srcid = id`) edges.

The result table has the columns `id` (the vertex identifier) and `neighbors` (an array of tuples representing each neighbor).

If an error occurs after table creation, the function drops the result table.

Syntax

Python

```
collect_neighbors(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_table,
    direction,
    [ result_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_table`	str	Name of the neighbors collection table (with the `id` and `neighbors` columns).
`direction`	EdgeDirection	Specifies the direction of traversal. Supported values are: `IN` — Neighbors that have edges pointing to the vertex (edges where `destid = id`). For example, if an edge `5` points to `10`, then for `id=10`, neighbor `5` is included. `OUT` — Neighbors that the vertex points to (edges where `srcid = id`). For example: If an edge `5` points to `10`, then for `id=5`, neighbor `10` is included. `BOTH` — The union of `IN` and `OUT`. This traversal includes neighbors from edges pointing to `id` and edges from `id`.
`result_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Collect incoming neighbors for each vertex and generate an index on the `id` column. The `direction` argument set to `IN` collects neighbors pointing to `id`.

© 2026 Ocient, Inc.

Python

```
collect_neighbors(
    connection,
    "sales", "customers", "purchases",
    "sales", "neighbors_in",
    EdgeDirection.IN,
    ["id"],
)
```

collect_edges

For each vertex in a table, this function collects an array of adjacent edge rows based on the specified direction. The result table has two columns: `id` (the vertex identifier) and `edges` (an array of tuples, each tuple containing all columns from the edges table for a connected edge).

The `direction` types are:

- `IN` — Edges pointing to the vertex (edges where `destid = id`).
- `OUT` — Edges originating from the vertex (edges where `srcid = id`).
- `BOTH` — Union of `IN` and `OUT` that includes edges from incoming (`destid = id`) and outgoing (`srcid = id`) directions. This direction retains duplicates.

If an error occurs after table creation, the function drops the result table.

Syntax

Python

```
collect_edges(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_table,
    direction,
    [ result_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_table`	str	Name of the edge collection table (with the `id` and `edges` columns).
`direction`	EdgeDirection	Specifies the direction of traversal. Supported values are: `IN` — Neighbors that have edges pointing to the vertex (edges where `destid = id`). For example, if an edge `5` points to `10`, then for `id=10`, neighbor `5` is included. `OUT` — Neighbors that the vertex points to (edges where `srcid = id`). For example, if an edge `5` points to `10`, then for `id=5`, neighbor `10` is included. `BOTH` — The union of `IN` and `OUT`. This traversal includes neighbors from edges pointing to `id` and edges from `id`.
`result_indexes`	List[str]	Optional. Columns to index. Specify an empty list for none.

Example Collect outgoing edges per vertex. The example sets the `direction` to `OUT` to collect edges from `id`.

Python

```
collect_edges(
    connection,
    "sales", "customers", "purchases",
    "sales", "outgoing_edges",
    EdgeDirection.OUT,
    ["id"],
)
```

© 2026 Ocient, Inc.

Algorithms

High-level graph algorithms that iterate over the graph structure to produce labels, components, or counts.

label_propagation

Executes the **Label Propagation Algorithm** (LPA) to assign community labels to vertices.

Each vertex starts with its own identifier as its label. For a number set by the `maxIterations` argument, each vertex updates its label to the most frequent label among its neighbors. The algorithm determines ties by choosing the smallest label. The algorithm uses temporary tables for intermediate results and drops these tables when the process completes or if it fails. Isolated vertices retain their initial label. The final table stores `id` and `label` columns and can include indexes.

Syntax

Python

```
label_propagation(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    max_iterations,
    result_vertices_indexes,
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).

Argument	Data Type	Description
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create (columns include `id` and `label`).
`max_iterations`	int	Maximum number of iterations (must be `1` or greater).
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Run label propagation for 10 iterations and assign labels to vertices. Generate an index on the `id` column.

Python

```
label_propagation(
    connection,
    "sales", "customers", "purchases",
    "sales", "lpa_labels",
    10,
    ["id"],
)
```

connected_components [#connected_components]

Identifies the connected components of an undirected graph. This algorithm configures a Pregel computation in which each vertex initially sets its component label equal to its own identifier `id`.

In each iteration, vertices send their component label to neighbors. Each vertex updates based on the aggregated minimum value of its current component label and any received values. The process repeats until no more updates occur.

The result table maps each vertex `id` to its final `component` label.

Syntax

Python

```
connected_components(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Compute connected components for a maximum of 20 iterations and generate an index on the `id` column.

Python

```
connected_components(
    connection,
    "sales", "customers", "purchases",
    "sales", "components",
    20,
   ["id"],
)
```

strongly_connected_components

Computes strongly connected components (SCC) in a directed graph. This function runs a recursive algorithm that partitions vertices into subsets where every vertex is reachable from other vertices in the same subset.

This function uses recursive partitioning. The algorithm selects a pivot (typically the minimum identifier `id`), computes its predecessor set (vertices that can reach the pivot), and its descendant sets (vertices reachable from the pivot). Then, the function identifies the SCC as their intersection, removes that SCC from the graph, and recurses on the remainder until all vertices have been assigned to an SCC. The output contains columns for the `id` and `component` identifiers (the minimum `id` in the SCC).

The function creates temporary tables in the result schema to store intermediate results. This function drops these tables when the computation completes or fails. The final result table contains two columns: `id` (vertex identifier) and `component` (the minimum vertex identifier in its SCC subset).

Syntax

Python

```
strongly_connected_components(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create (columns include `id` and `component`).
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Compute the SCC and generate an index on the `id` column.

Python

```
strongly_connected_components(
    connection,
    "sales", "customers", "purchases",
    "sales", "scc",
    ["id"],
)
```

TriangleCount

TriangleCount identifies all 3-cycles (triangles) in the graph and counts how many distinct triangles each vertex participates in.

The algorithm first builds a canonical, undirected edge set by ensuring `srcid < destid` and removing duplicates to prevent double-counting. If your input edges are already canonicalized and deduplicated, use `TriangleCount.run_pre_canonicalized` to skip preprocessing for faster performance.

The function then counts triangles (`a` , `b` , `c`) where `a < b < c` by intersecting neighbor lists and aggregates per-vertex participation to produce a result table with the `id` and `triangle_count` columns.

`run` syntax

Python

```
TriangleCount.run(
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    [ result_vertices_indexes [ , ... ] ],
)
```

`run_pre_canonicalized` syntax

Python

```
TriangleCount.run_pre_canonicalized(
    connection,
    input_schema,
    input_vertices_table,
    canonical_edges_schema,
    canonical_edges_table,
    result_schema,
    result_vertices_table,
    [ result_vertices_indexes [ , ... ] ]
)
```

© 2026 Ocient, Inc.

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns). For `runPreCanonicalized` execution, the function assumes this table is already canonicalized (e.g., `srcid < destid`) and deduplicated.
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create (columns include `id` and `triangle_count`).
`result_vertices_indexes`	list[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Examples

Count Triangles Using `run`

Canonicalize the raw edges internally, count unique triangles, and write per-vertex triangle counts with an index on the `id` column.

Python

```
TriangleCount.run(
    connection,
    "sales",
    "customers",
    "purchases",
    "sales",
    "triangle_counts",
    ["id"],
)
```

Count Triangles Using `run_pre_canonicalized`

Use a pre-canonicalized, deduplicated edge table to count triangles and write per-vertex triangle counts with an index on the `id` column.

Python

```
TriangleCount.run_pre_canonicalized(
    connection,
    "sales",
    "customers",
    "sales",
    "purchases_canonical",
    "sales",
    "triangle_counts",
    ["id"],
)
```

pregel

Provides a generic vertex-centered iteration framework for custom graph algorithms, similar to the **Pregel model**.

Each iteration updates vertex states by sending messages along edges and then aggregating these messages to compute new states. The algorithm continues iterating until it reaches convergence (no state changes or no messages produced) or a specified iteration cap.

The algorithm uses multiple specified SQL expressions.

Syntax

Python

```
pregel(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    initializer_expr,
    send_to_source_expr,
    send_to_dest_expr,
    aggregate_expr,
    updater_expr,
    max_iterations,
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create (columns include `id` and `result`).
`initializer_expr`	str	A SQL expression to compute the initial state for each vertex (e.g., `CASE WHEN type = 'seed' THEN 1.0 ELSE 0.0 END`).
`send_to_source_expr`	str	Optional. Defines the message sent to the source vertex of an edge, referencing the current vertex states `a.state` (source) and `c.state` (destination), and any edge attributes in `b`. Example: `CASE WHEN a.country = c.country THEN 1 ELSE 0 END`

Argument	Data Type	Description
`send_to_dest_expr`	str	Optional. Defines the message sent to the destination vertex of an edge, referencing the current vertex states `a.state` (source) and `c.state` (destination), and any edge attributes as `b.<edge_column>`. Example: `CASE WHEN a.country = c.country THEN 1 ELSE 0 END`
`aggregate_expr`	str	A SQL aggregation to combine messages per vertex (e.g., `SUM(msg)`). For a list of supported aggregations, see **Aggregate Functions**.
`updater_expr`	str	A SQL expression to compute the next state from the current state and aggregated messages. For example, this code updates the state to the minimum current state and the aggregated message (similar to the **connected_components** function): `LEAST(a.state, COALESCE(m.aggregated_message, a.state)) AS state`
`max_iterations`	int	Maximum number of iterations (must be either `-1` or a value of `1` or greater). If this value is `-1`, the Pregel algorithm has no limit, and it runs until convergence. A value of `0` causes the algorithm to return an error.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.

Example Run a simple Pregel computation summing incoming edge amounts into the vertex state for 10 iterations at most, and generate an index on the `id` column.

Python

```
pregel(
    connection,
    "sales", "customers", "purchases",
    "sales", "pregel_result",
    "0 AS state",
    "b.amount",
    None,
    "SUM(msg) AS aggregated_message",
    "state + COALESCE(aggregated_message, 0) AS state",
    10,
    ["id"],
)
```

Paths & Ranking

These functions include the shortest-path and PageRank algorithms.

shortest_paths

Computes the shortest distance from every vertex to each set of landmark vertices using an iterative relaxation algorithm. The algorithm resembles **Bellman-Ford** but simultaneously handles multiple destinations.

Each landmark starts at distance `0` and all others at positive infinity. On each iteration, the algorithm examines every edge and checks whether traveling through the connected neighbor would yield a shorter route to a landmark. If a shorter route exists, the algorithm updates the distance of the source vertex. The process stops when no distances improve or the algorithm reaches the maximum number of iterations.

After the process finishes, the algorithm writes a result table with the `srcid`, `destid`, and `distance` columns.

Syntax

Python

```
shortest_paths(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_table,
    landmarks[ , ... ],
    edge_weight_column,
    max_iterations,
    [ result_vertices_indexes [ , ... ] ]
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result table.
`result_table`	str	Name of the distances result table. This table has the `srcid`, `destid`, and `distance` columns.
`landmarks`	List[int]	One or more landmark vertex identifiers. This list must be non-empty and contain no NULLs.
`edge_weight_column`	Optional[str]	Optional. The name of an edge weight column. If you do not specify this argument, the default value is `1.0`.
`max_iterations`	int	Maximum number of relaxation iterations. This value must be `1` or greater.
`result_indexes`	List[str]	Optional. The list of columns to index in the result table (`srcid`, `destid`, and `distance` columns). Specify an empty list for none.

Example Compute distances from landmarks and generate indexes on the `src` and `dest` columns.

Python

```
shortest_paths(
    connection,
    "sales", "customers", "purchases",
    "sales", "distances",
    [1, 42],
    None,
    10,
    ["srcid","destid"],
)
```

static_page_rank

Computes **PageRank** scores for each vertex over a fixed number of iterations.

The algorithm follows the standard PageRank formula with a reset probability (`reset_prob`) and uses common table expressions to calculate contributions from incoming edges and redistribute ranks from dangling nodes.

The algorithm supports two variants:

- Standard PageRank — All vertices start with rank `1.0/N`, where `N` is the number of vertices. Specify this variant if `personalizationSrcId` is `null`.
- Personalized PageRank — The specified vertex starts with a rank of `1.0`, while others start with a rank of `0.0`. Specify this variant if `personalizationSrcId` is a vertex identifier.

After running PageRank for a fixed number of iterations, the function writes a result vertices table containing all original vertex columns with a new PageRank scoring column.

Syntax

Python

```
static_page_rank(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    num_iterations,
    reset_prob,
    [ result_vertices_indexes [ , ... ] ],
    personalization_src_id,
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.
`result_vertices_table`	str	Name of the vertices table to create (columns include all vertex columns and a `pagerank` column).
`num_iterations`	int	Number of iterations to run. This value must be 1 or greater.
`reset_prob`	float	A value between `0.0` and `1.0` that controls the damping factor for how PageRank random surfer moves across vertices. A high value (e.g., `0.85`) puts more emphasis on link structure, encouraging the algorithm to move from each vertex to a neighbor. This value means PageRank scores tend to concentrate around well-linked regions. A lower score (e.g., `0.50`) allows the algorithm to ignore edges and instead jump to a vertex chosen from a base distribution. For standard PageRank, this base is uniform over all vertices. For personalized PageRank, the

Argument	Data Type	Description
		base is biased toward the specified vertex. This behavior creates more uniform scoring with less sensitivity to link topology.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`personalization_src_id`	Optional[int]	Determines whether PageRank uses the standard or the personalized variant. For personalized mode, specify a vertex identifier. PageRank scoring starts with this vertex set at `1.0`, while all other vertices start at `0.0`. For standard mode, specify `None`. All vertices start with rank `1.0/N`, where `N` is the number of vertices.

Example Run fixed-iteration PageRank and generate an index on the `id` column. This example uses a high reset probability `reset_prob` of `0.85` to ensure the ranking concentrates on highly linked regions.

Python

```
static_page_rank(
    connection,
    "sales", "customers", "purchases",
    "sales", "pagerank_static",
    10,
    0.85,
    ["id"],
    None,
)
```

dynamic_page_rank

Computes PageRank scores until convergence based on a specified threshold value (`tolerance`). Unlike the **static_page_rank** function, this algorithm runs iterations until the sum of absolute differences between ranks in successive iterations is less than or equal to the `tolerance` value. The algorithm handles personalization similarly to `static_page_rank`. At each iteration, the function uses the PageRank formula, collects rank values, and redistributes them.

The algorithm supports two variants:

- Standard PageRank — All vertices start with rank `1.0/N`, where `N` is the number of vertices. Specify this variant if `personalizationSrcId` is `null`.
- Personalized PageRank — The specified vertex starts with a rank of `1.0`, while others start with a rank of `0.0`. Specify this variant if `personalizationSrcId` is a vertex identifier.

After running PageRank until the system reaches the `tolerance` threshold, the function writes a vertices table containing all the original vertex columns with a new PageRank scoring column.

Syntax

Python

```
dynamic_page_rank(
    connection,
    input_schema,
    input_vertices_table,
    input_edges_table,
    result_schema,
    result_vertices_table,
    tolerance,
    reset_prob,
    [ result_vertices_indexes [ , ... ] ],
    personalization_src_id,
)
```

Argument	Data Type	Description
`connection`	pyocient.Connection	An active database connection using the `pyocient` module.
`input_schema`	str	A non-empty schema containing the input tables.
`input_vertices_table`	str	Input vertices table (must have an `id` column).
`input_edges_table`	str	Input edges table (must have the `srcid` and `destid` columns).
`result_schema`	str	A writable schema to create the result tables.

Argument	Data Type	Description
`result_vertices_table`	str	Name of the vertices table to create (columns include all vertex columns and `pagerank`).
`tolerance`	float	The convergence threshold that stops iterations when the rank changes become negligible. After each iteration, the algorithm measures the maximum change in any rank of a vertex. If that change is below the specified tolerance, the algorithm considers the computation converged and stops early. A high tolerance value (e.g., `0.001`) is good for quick exploratory runs that generate an approximate ranking. A low tolerance value (e.g., `0.000001`) generates a precise ranking, but requires a higher compute cost.
`reset_prob`	float	A value between `0.0` and `1.0` that controls the damping factor for how the PageRank random surfer moves across vertices. A high value (e.g., `0.85`) puts more emphasis on link structure, encouraging the algorithm to move from each vertex to a neighbor. The high value means PageRank scores tend to concentrate around well-linked regions. A lower score (e.g., `0.50`) allows the algorithm to ignore edges and instead jump to a vertex chosen from a base distribution. For standard PageRank, this base is uniform over all vertices. For personalized PageRank, the base is biased toward the specified vertex. This behavior creates more uniform scoring with less sensitivity to link topology.
`result_vertices_indexes`	List[str]	Optional. Columns to index in the result vertices table (e.g., `id`). Specify an empty list for none.
`personalization_src_id`	Optional[int]	Determines whether PageRank uses the standard or the personalized variant. For personalized mode, specify a vertex identifier. PageRank scoring starts with this vertex set at `1.0`, while all other vertices start at `0.0`. For standard mode, specify `None`. All vertices start with rank `1.0/N`, where `N` is the number of vertices.

Example Run dynamic PageRank to convergence and generate an index on the `id` column. This example uses a low `tolerance` value of `1.0e-6`, which generates high-

precision rankings but requires more computing resources.

```python
dynamic_page_rank(
    connection,
    "sales", "customers", "purchases",
    "sales", "pagerank_dynamic",
    1.0e-6,
    0.85,
    ["id"],
    None,
)
```

Bibliography

Pregel: A System for Large-Scale Graph Processing.” Accessed November 18, 2025. **https://research.google/pubs/pregel-a-system-for-large-scale-graph-processing/**.

Related Links

OCGraph Java Library

Ocient Python Module: pyocient

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Ingest Data with Legacy LAT Reference

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

Loading and Transformation (LAT) Overview

This section provides an overview of how loading works in Ocient along with step-by-step examples to illustrate the key aspects of the loading process. A separate LAT Reference provides a detailed explanation of the options for each data format and data source in Ocient.

The key elements in the Loading system are:

- Data Source — an origin source for data such as Amazon® Web Services℠ (AWS℠) S3, Apache® Kafka®, or Apache® Hadoop® Distributed File System (HDFS). Sources can be file or streaming in nature.
- Data Type Extraction — the format for data extraction from the source. Examples include JSON, CSV, fixed width.
- Transformations — the functions used to cleanse, route, and transform the incoming data.
- Indexer — operated by the `streamloader` role on a Loader Node, the Indexer collects transformed records, stores in replicated pages, and converts into segments on the Foundation Nodes.

Reference:

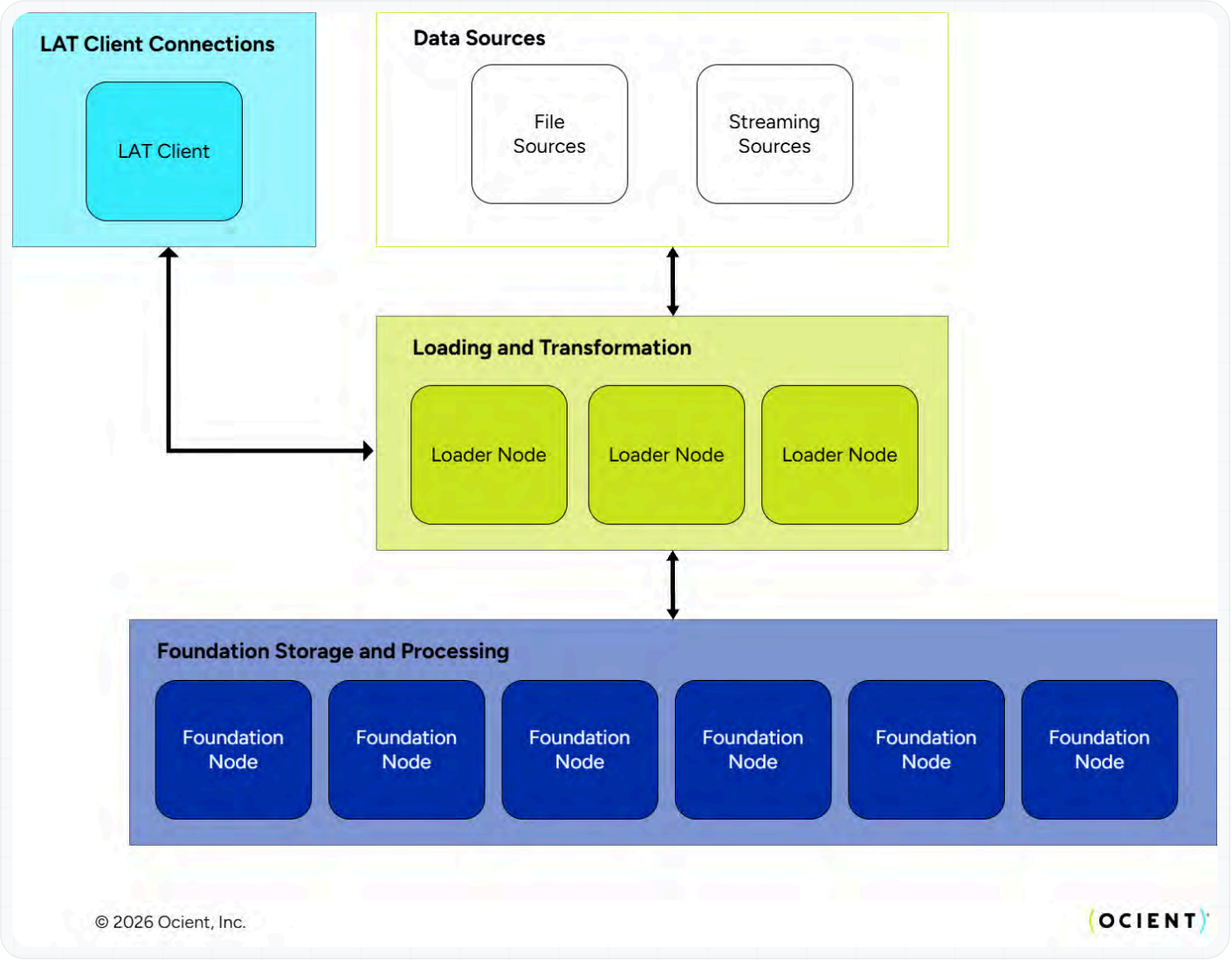
- **LAT Data Types in Loading**
- **LAT Pipeline Configuration**
- **LAT Client Command Line Interface**
- **LAT Metrics**
- **LAT Advanced Topics**
- **LAT Packaging and Installation**

Loading Overview

The LAT is a service within an Ocient System that is responsible for fetching data from file or streaming sources, extracting records, transforming them, and routing them to Ocient tables. The LAT runs as a standalone Java® process on Loader Nodes. It runs many parallel workers on high-

© 2026 Ocient, Inc.

core count servers to deliver the required parallelism for high-throughput loading. This diagram shows that a set of Loader Nodes sit between the data sources and the Foundation Nodes to control the loading flow and deliver high performance across many parallel workers.



The LAT loads data from two general types of data sources:

- Streaming Sources — continuous loading of data from an ordered stream such as Kafka
- File Sources — a discrete batch of data loaded from files until complete, typically from a file system or object storage solution like AWS S3

The main difference between these modes is whether the task continues indefinitely or ends when the batch of files has been loaded. In both cases, the LAT uses identical transformation and loading processes so that users can manage both with the same mental model and configuration files. Most data types are supported identically in streaming and batch processing modes, however some formats are not suitable for streaming due to their structure.

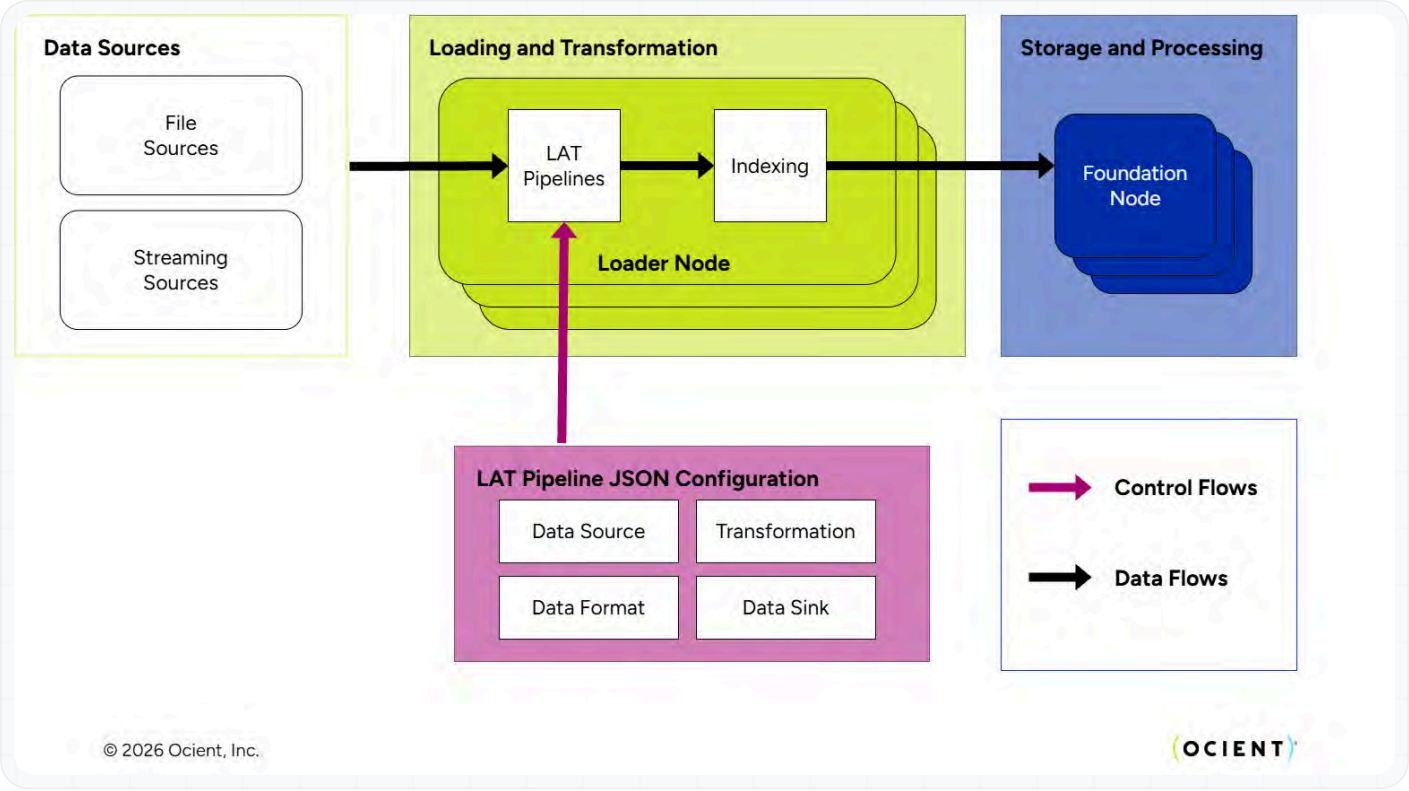
A complete list of data sources and supported data formats is in **LAT Data Types in Loading**.

While loading, the LAT coordinates with the Ocient System to manage backpressure of loading new records. New rows accumulate into a storage structure called a “page” which is a column-oriented storage mechanism. These pages are replicated across Foundation Nodes so that the specified redundancy is maintained in the event of a system outage. When enough pages have accumulated in a given time period (as defined by the TimeKey® bucket), the Indexer will convert the pages into segment structures and place these on the Foundation Nodes for permanent storage as Segment Groups that span the width of the storage space.

This process is transparent to analysts issuing queries. Rows are federated during queries so that pages and Segments are seamlessly presented in result sets.

Pipelines Overview

The process and configuration that defines how data is loaded is referred to as a “pipeline.” A “pipeline” defines the end-to-end data flow in a loading task including the Source type (e.g., Kafka), the record Extraction for the data type (e.g., CSV extraction), the transformations on each record (e.g., JSON value extraction, data exploding, string concatenation, flattening), and where transformed data should be loaded in Ocient tables. As shown in this diagram, a user sets each of these sections in the pipeline file to control the LAT.



The LAT pipelines are managed through an HTTP API on the Loader Nodes or through a Command Line Interface (CLI) that provides a convenient way to work with the HTTP API. Because most Ocient Systems include many Loader Nodes, the CLI coordinates pipelines across the specified set of Loader Nodes.

The LAT Client issues the command to the Loader Node to create the new pipeline with the pipeline configuration file. When started, the LAT uses the pipeline to execute a highly parallelized loading task across all Loader Nodes and Foundation Nodes to load the configured tables.

Detailed instructions for configuring a pipeline are found in **LAT Pipeline Configuration**.

Time Ordering

The Ocient System contains timeseries data. Loading in the data warehouse is considerably faster when data is presented in an ordered time sequence according to the TimeKey in a table. For most streaming sources, this is typically accomplished automatically with limited “out of order” data by the nature of the queue.

However, for file based sources, it is not uncommon for data to be collected in folders in a manner that could appear haphazard to the loader. For this reason, it is critical that the `sort_type` setting for file based sources be used correctly to inform the LAT how files should be ordered.

Read more about file sorting in **Load From a File Source**.

Exactly-Once Guarantees

Loading in Ocient is designed to ensure exactly once processing of data. The LAT operates independent streams of data that each has a monotonically increasing row ID tied to an individual row in a File-based load or a partition offset in a Kafka-based load. Exactly-once processing is made possible through coordination between the loading components around a “Durability Horizon” that represents the highest record durably stored on non-volatile storage in each independent stream. In the event of a node outage or a replay of the data, the Durability Horizon automatically removes duplicate records in an efficient manner.

It is important to understand how the LAT determines the unique row ID for different source types to ensure that data is loaded correctly. This is described in more detail in the LAT Reference Documentation.

Learn more about the underlying approach used in **Loading Characteristics and Concepts**.

Dynamic Schema Changes

Ocient also supports schema changes on tables while loading data in a continuously streaming pipeline. Ocient maintains a table version with each running pipeline that serves as a loading contract until the load task complete. This ensures that existing pipelines are not interrupted when columns are added or removed from a table that is receiving new data. Ocient continues loading records using the original table version even after a change has been made to the tables. These

dynamic schema changes from `ALTER TABLE` commands allow flexible updates to tables while complex loading processes are active. Pipelines can then be updated to add or remove data elements and match the altered schema, reducing the burden of coordination across systems.

Loading Examples

The following examples walk through a simple loading example using the LAT for a streaming load off of Kafka and a file load off of S3. These examples are very similar due to the way the LAT uses a common language for all transformation and loading regardless of data source or data type. The final set of examples outline some more complex transformations.

Examples:

- **LAT Load JSON Data from Kafka**
- **LAT Load CSV Data from S3**
- **LAT Load JSON Data from S3**
- **LAT Advanced Loading and Transformations**

For more detailed information about LAT settings, see the **LAT Pipeline Configuration**.

Related Links

LAT Overview

LAT Data Types in Loading

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Load JSON Data from Kafka

⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

A common setup for streaming data into Ocient is to send JSON documents to Apache® Kafka® and then transform each document into rows in one or more different tables. Ocient's Loading and Transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial will guide users through a simple example load using a small set of data in JSON format. The data in this example is created from a test set for the Metabase℠ Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. A Kafka cluster is operational and can be reached by the Ocient Loader Nodes.
2. An Ocient System is installed and configured with an active storage cluster (See the **Ocient Application Configuration** guide).
3. The Ocient Loader Nodes are running the latest Loading and Transformation version which is configured to connect to Kafka for stream loading.
4. A default "sink" for the Ocient Loader Nodes is configured on the system.
5. The **LAT Client Command Line Interface** is installed.
6. The test data for this Tutorial can be found at the following S3 addresses.

ⓘ You must be logged into Amazon® Web Services℠ (AWS℠) to download these files.

```
https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl
https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/products.jsonl
https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/people.jsonl
https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/reviews.jsonl
```

© 2026 Ocient, Inc.

Step 1: Create a New Database

To begin, you are going to load four example tables in a database. First, connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then run the following DDL command:

```sql
CREATE DATABASE metabase;
```

Step 2: Create Tables

To create tables in the new database, first connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), then run the following DDL commands:

SQL

```sql
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);

CREATE TABLE public.people (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  address VARCHAR(255),
  email VARCHAR(255),
  password VARCHAR(255),
  name VARCHAR(255),
  city VARCHAR(255),
  longitude DOUBLE,
  state VARCHAR(255),
  source VARCHAR(255),
  birth_date DATE,
  zip VARCHAR(255),
  latitude DOUBLE,
  CLUSTERING INDEX idx01 (id)
);

CREATE TABLE public.products(
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  ean VARCHAR(255),
  title VARCHAR(255),
  category VARCHAR(255) COMPRESSION GDC(2) NOT NULL,
  vendor VARCHAR(255),
  price DOUBLE,
  rating DOUBLE,
  CLUSTERING INDEX idx01 (category)
);
```

© 2026 Ocient, Inc.

```
CREATE TABLE public.reviews (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  product_id INT NOT NULL,
  reviewer VARCHAR(255),
  rating INT,
  body VARCHAR(255),
  CLUSTERING INDEX idx01 (product_id)
);
```

Now, the database tables are created, and you can begin loading data.

Step 3: Create a Data Pipeline

Data pipelines are created using a simple loading configuration that is submitted to the Transformation Nodes to start loading. Each Kafka topic is routed to one or more Ocient tables, and each column is the result of a transformation applied to the source document.

First, inspect the data that you load. Each document has a format similar to the following example.

JSON

```
/* orders */
{"id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total'
{"id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total
...

/* products */
{"id": 1, "ean": "1018947080336", "title": "Rustic Paper Wallet", "category": "Gi
{"id": 2, "ean": "7663515285824", "title": "Small Marble Shoes", "category": "Doc
{"id": 3, "ean": "4966277046676", "title": "Synergistic Granite Chair", "category
...

/* people */
{"id": 1, "address": "9611-9809 West Rosedale Road", "email": "borer-hudson@yahoc
{"id": 2, "address": "101 4th Street", "email": "williamson-domenica@yahoo.com",
{"id": 3, "address": "29494 Anderson Drive", "email": "lina.heaney@yahoo.com", "p
...

/* reviews */
{"id": 1, "product_id": 1, "reviewer": "christ", "rating": 5, "body": "Ad perspic
{"id": 2, "product_id": 1, "reviewer": "xavier", "rating": 4, "body": "Reprehende
{"id": 3, "product_id": 1, "reviewer": "cameron.nitzsche", "rating": 5, "body": '
...
```

As you can see, this is similar to the target schema, but will require some transformation. Most transformations are identical to functions already in Ocient's SQL dialect. To route data to your tables, you must create a pipeline.json file that has the following structure:

© 2026 Ocient, Inc.

JSON

```
{
    "version": 2,
    "workers": 4,
    "pipeline_id": "pipeline-metabase",
    "source": {
        "type": "kafka",
        "kafka": {
          "bootstrap.servers": "127.0.0.1:9092",
          "auto.offset.reset": "earliest"
        }
    },
    "transform": {
        "topics": {
            "orders": {
                "tables": {
                    "metabase.public.orders": {
                        "columns": {
                            "id": "id",
                            "user_id": "user_id",
                            "product_id": "product_id",
                            "subtotal": "subtotal",
                            "tax": "tax",
                            "total": "total",
                            "discount": "discount",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                            "quantity": "quantity"
                        }
                    }
                }
            },
            "people": {
                "tables": {
                    "metabase.public.people": {
                        "columns": {
                            "id": "id",
                            "address": "address",
                            "email": "email",
                            "password": "password",
                            "name": "name",
                            "city": "city",
                            "longitude": "longitude",
```

```
                            "state": "state",
                            "source": "source",
                            "birth_date": "birth_date",
                            "zip": "zip",
                            "latitude": "latitude",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                        }
                    }
                }
            },
            "reviews": {
                "tables": {
                    "metabase.public.reviews": {
                        "columns": {
                            "id": "id",
                            "product_id": "product_id",
                            "reviewer": "reviewer",
                            "rating": "rating",
                            "body": "body",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                        }
                    }
                }
            },
            "products": {
                "tables": {
                    "metabase.public.products": {
                        "columns": {
                            "id": "id",
                            "ean": "ean",
                            "title": "title",
                            "category": "category",
                            "vendor": "vendor",
                            "price": "price",
                            "rating": "rating",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                        }
                    }
                }
            }
        }
    }
}
```

Step 4: Using the Loading and Transformation CLI

With a pipeline.json file ready to go, you can test this pipeline. To test, use the LAT CLI. For these examples, you can assume that two LATs are configured and will set them using an environment variable.

First, configure the LAT CLI to use the hosts of your Loading and Transformation service. You can add these to every CLI command as a flag, but for simplicity you can also set them as environment variables. From a command line, run the following command replacing the IP addresses with the IP addresses of your LAT processes:

Shell

```
export LAT_HOSTS="https://10.0.0.1:8443,https://10.0.0.2:8443"
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://` .

Next, check on the status of the LAT:

Shell

```
lat_client pipeline status
```

Example response:

Bash

```
10.0.0.1:8443: Stopped
10.0.0.2:8443: Stopped
```

Success! This confirms that you can reach the LAT from your CLI. If the status is “Running” it means a pipeline is already executing a pipeline. You are next going to update and start your new pipeline.

© 2026 Ocient, Inc.

This example uses secure connections. If you receive an SSL Error when testing, your service cannot be configured to use TLS or you might need to use the `--no-verify` flag if certificate validation fails.

Step 5: Test the Transformation

The CLI supports previewing a transformation with an example document and the pipeline file. This makes it easy to test your transformations.

First, save an example document to your file system to use for this test. For this demo, you can download an example file from **https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl** and save it to `~/orders.jsonl`.

Next, make sure the pipeline.json file that you created is stored at `~/pipeline.json`.

Now that both files are available, run the CLI to preview the results. You can pass the preview command the topic name, the pipeline file, and the sample record file. The response contains the transformed data tied to the destination table and a list of any error records.

Shell

```
lat_client preview --topic orders --pipeline ~/pipeline.json --records ~/orders.j
```

Example response:

JSON


```
{
    "tableRecords": {
        "metabase.public.orders": [
            {
                "id": 1,
                "user_id": 1,
                "product_id": 14,
                "subtotal": 37.65,
                "tax": 2.07,
                "total": 39.72,
                "discount": null,
                "created_at": 1549921227892000000,
                "quantity": 2
            },
            {
                "id": 2,
                "user_id": 1,
                "product_id": 123,
                "subtotal": 110.93,
                "tax": 6.1,
                "total": 117.03,
                "discount": null,
                "created_at": 1526371444580000000,
                "quantity": 3
            },
            {
                "id": 3,
                "user_id": 1,
                "product_id": 105,
                "subtotal": 52.72,
                "tax": 2.9,
                "total": 49.2,
                "discount": 6.42,
                "created_at": 1575670968544000000,
                "quantity": 2
            }
        ]
    },
    "recordErrors": []
}
```


© 2026 Ocient, Inc.

You can see that the data is transformed and the columns to which each transformed value will be mapped. If there are issues in the values, these will appear in the `recordErrors` object. You can quickly update your pipeline.json file and preview again. Now, you can inspect different documents to confirm that various states of data cleanliness like missing columns, null values, and special characters are well handled by your transformations.

Step 6: Configure and Start the Data Pipeline

With a tested transformation, the next step is to set up and start the data pipeline.

First, configure the pipeline using the `pipeline create` command. This validates and creates the pipeline, but will not take effect until you start the pipeline:

Shell

```
lat_client pipeline create --pipeline ~/pipeline.json
```

Example response:

Bash

```
10.0.0.1:8443: Created
10.0.0.2:8443: Created
```

> In cases where there is an existing pipeline operating, it is necessary to stop the pipeline and remove the original pipeline before creating and starting the new pipeline.

Now that the pipeline has been created on all LAT Nodes, you can start the LAT by running the `pipeline start` commands:

Shell

```
lat_client pipeline start
```

Example responses:

Bash

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

Step 7: Confirm that Loading is Operating Correctly

With your pipeline in place and running, data will immediately begin loading off of the Kafka topics that are configured in the pipeline. If you do not have data in the Kafka topics yet, now would be a good time to start producing data into the topics.

Producing Test Data into Kafka:

For test purposes, **kafkacat** is a helpful utility that makes it easy to product records into a topic. For example, if you have a file of sample data `orders.jsonl` in a JSONL format (newline delimited JSON records), you can run the following command to send those records into your Kafka broker:

Shell

```
kafkacat -b <broker_ip_address>:9092 -t <topic_name> -T -P -l orders.jsonl
```

Assuming your broker is running at `10.0.0.3` and you want to send data into the four topics defined in your `pipeline.json` definition, you can run:

Shell

```
kafkacat -b 10.0.0.3:9092 -t orders -T -P -l orders.jsonl
kafkacat -b 10.0.0.3:9092 -t products -T -P -l products.jsonl
kafkacat -b 10.0.0.3:9092 -t people -T -P -l people.jsonl
kafkacat -b 10.0.0.3:9092 -t reviews -T -P -l reviews.jsonl
```

Each of these commands will push the entire JSONL file of messages into Kafka with one record per line. As these are produced into Kafka, your running pipeline will begin loading them into Ocient.

© 2026 Ocient, Inc.

Observing Loading Progress:

With data in Kafka, our pipeline will begin loading data immediately and streaming any new data into Ocient. To observe this progress, you can monitor the metrics endpoint of the Loading and Transformation Nodes. This can be done manually from a command line or from a tool like Grafana®. For this example, you can run a `curl` command against the endpoint and review the result. For details on metrics, see the **LAT Metrics** Documentation.

Command:

CURL

```
curl https://127.0.0.1:8443/v2/metrics/lat:type=pipeline
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://` .

Example response:

JSON


```
{
  "request": {
  "mbean": "lat:type=pipeline",
  "type": "read"
  },
  "value": {
  "partitions": [
    {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 18759,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 18759,
        "partition": "table_orders-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 18759,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 88,
        "records_errors_total": 0,
        "offsets_buffered": 18759,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
    },
    {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 2499,
```

```
    "records_buffered": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "source_bytes_buffered": 0,
    "records_errors_transformation": 0,
    "offsets_processed": 2499,
    "partition": "table_people-0",
    "records_filter_accepted": 1,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,
    "offsets_pushed": 2499,
    "pushes_unacknowledged": 0,
    "invalid_state": 0,
    "bytes_pushed": 223,
    "records_errors_total": 0,
    "offsets_buffered": 2499,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
  },
  {
    "offsets_durable": 1,
    "pushes_errors": 0,
    "pushes_attempts": 1,
    "rows_pushed": 1,
    "offsets_written": 199,
    "records_buffered": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "source_bytes_buffered": 0,
    "records_errors_transformation": 0,
    "offsets_processed": 199,
    "partition": "table_products-0",
    "records_filter_accepted": 1,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,
    "offsets_pushed": 199,
    "pushes_unacknowledged": 0,
    "invalid_state": 0,
```

```
        "bytes_pushed": 145,
        "records_errors_total": 0,
        "offsets_buffered": 199,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
      },
      {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 1111,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 1111,
        "partition": "table_reviews-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 1111,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 184,
        "records_errors_total": 0,
        "offsets_buffered": 1111,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
      }
  ],
  "paused": 1,
  "bytes_buffered": 0,
  "workers": 20
  },
  "timestamp": 1626970368,
  "status": 200
}
```

Check Row Counts in Tables:

To confirm that you are seeing results in the target tables, you can also run some simple queries to check row counts. Depending on the streamloader role settings, the time for records to become queryable can vary from a few seconds to minutes:

Example Queries:

SQL

```
Ocient> SELECT count(*) FROM public.orders;
count(*)
------------------------
18760
```

SQL

```
Ocient> SELECT count() FROM public.people;
count()
-----------------------
2500
```

SQL

```
Ocient> SELECT count(*) FROM public.products;
count(*)
--------------------
200
```

SQL

```
Ocient> SELECT count(*) FROM public.reviews;
count(*)
--------------------
1112
```

Success! Now you can explore the data in these four tables with any Ocient SQL queries. If more data is pushed into these topics, your pipeline is still running and will automatically load all new data.

Check Errors

In this example, all rows load successfully. However, a successful load does not always happen, and you can inspect errors using the LAT Client. Whenever the LAT process fails to parse a file correctly or fails to transform or load a record, the LAT process records an error. The LAT Client includes the `lat_client pipeline errors` command that reports the latest errors on the pipeline.

A full error log is available on the Loader Nodes. These logs report all bad records and the reason that the load fails.

> When you load a pipeline from Kafka, the load might route errors to an error topic on the Kafka broker instead of the logs. The LAT Client does not contain the errors sent to the error topic. You can inspect these errors with Kafka utilities instead.

This LAT Client command displays a maximum of 100 error messages.

```
lat_client pipeline errors --max-errors 100 --only-error-messages

|-------------------------------------------------|
| exception_message                               |
|-------------------------------------------------|
| Column name: time1. Message: Failed to evaluate  |
| expression. Cause:                              |
| java.time.format.DateTimeParseException         |
|-------------------------------------------------|
| Column name: time1. Message: Failed to evaluate  |
| expression. Cause:                              |
| java.time.format.DateTimeParseException         |
|-------------------------------------------------|
```

The errors indicate that there is an issue parsing the `time1` column. Options exist on the `pipeline errors` command to return JSON and to restrict the response to specific

components of the error detail that includes a reference to the source location of this record.

The following command returns JSON that is delimited with newline characters. You can pass the JSON output to `jq` or a file. The JSON includes the source topic or file group, the filename where the error occurred, the offset that indicates the line number or Kafka offset, and the exception message that aids in troubleshooting and identifying the incorrect record in the source data. You can use the `log_original_message` pipeline setting to provide direct access to the parsed source record for errors when appropriate.

```
lat_client pipeline errors --max-errors 100 --json

{"time": "2022-05-17T16:53:50.387386+00:00", "topic": "calcs", "partition": 0, "s
{"time": "2022-05-17T16:53:50.404684+00:00", "topic": "calcs", "partition": 0, "s
```

Related Links

LAT Overview

LAT Data Types in Loading

LAT Advanced Topics

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Load CSV Data from S3

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

A common setup for batch loading files into Ocient is to load from a bucket on Amazon® Web Services℠ (AWS℠) S3 with time partitioned data. In many instances, a batch load is performed on a recurring basis to load new files. The LAT transforms each document into rows in one or more different tables. Ocient's Loading and Transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial will guide users through a simple example load using a small set of data in CSV format. The data in this example is created from a test set for the Metabase℠ Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active storage cluster (See the **Ocient Application Configuration** guide).
3. Loading and Transformation is installed on the Loader Nodes.
4. A default "sink" for the Ocient Loader Nodes is configured on the system.
5. The **LAT Client Command Line Interface** is installed.

Step 1: Create a New Database

To begin, load two example tables in a database. First, connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then run the following DDL command:

© 2026 Ocient, Inc.

SQL

```
CREATE DATABASE metabase;
```

Step 2: Create Tables

To create tables in the new database, first connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), then run the following DDL commands:

SQL

```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);

CREATE TABLE public.products(
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  ean VARCHAR(255),
  title VARCHAR(255),
  category VARCHAR(255) COMPRESSION GDC(2) NOT NULL,
  vendor VARCHAR(255),
  price DOUBLE,
  rating DOUBLE,
  CLUSTERING INDEX idx01 (category)
);
```

Now, the database tables are created and you can begin loading data.

Step 3: Create a Data Pipeline

Data pipelines are created using a simple loading configuration that is submitted to the Transformation Nodes to start loading. File Groups designate a batch of files to load. Each File Group is routed to one or more Ocient tables, and each column is the result of a transformation applied to the source document.

First, inspect the data that you plan to load. Each document has a format similar to the following example:

Text

```
/* orders */
id,user_id,product_id,subtotal,tax,total,discount,created_at,quantity
1,1,14,37.65,2.07,39.72,null,2019-02-11T21:40:27.892Z,2
2,1,123,110.93,6.1,117.03,null,2018-05-15T08:04:04.580Z,3
3,1,105,52.72,2.9,49.2,6.42,2019-12-06T22:22:48.544Z,2
...

/* products */
id,ean,title,category,vendor,price,rating,created_at
1,1018947080336,Rustic Paper Wallet,Gizmo,"Swaniawski, Casper and Hilll",29.46,4.
2,7663515285824,Small Marble Shoes,Doohickey,Balistreri-Ankunding,70.08,0,2019-04
3,4966277046676,Synergistic Granite Chair,Doohickey,"Murray, Watsica and Wunsch",
...
```

This is similar to the target schema created in Step 2, but it will require some transformation. Most transformations are identical to functions already in Ocient's SQL dialect. To route data to the tables, you need to create a pipeline.json file that has the following structure:

© 2026 Ocient, Inc.

JSON

```
{
    "version": 2,
    "workers": 4,
    "source": {
        "compression": "none",
        "type": "s3",
        "endpoint": "https://s3.us-east-1.amazonaws.com",
        "bucket": "ocient-docs",
        "file_groups": {
            "orders": {
                "prefix": "metabase_samples/csv",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**orders*.csv",
                "sort_type": "lexicographic"
            },
            "products": {
                "prefix": "metabase_samples/csv",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**products*.csv",
                "sort_type": "lexicographic"
            }
        }
    },
    "extract": {
        "orders": {
            "record_type": "delimited",
            "headers": [
                "id",
                "user_id",
                "product_id",
                "subtotal",
                "tax",
                "total",
                "discount",
                "created_at",
                "quantity"
            ],
            "null_strings": [
                "NULL",
                "N/A"
            ],
```

```
            "record_delimiter": "\r\n",
            "num_header_lines": 1
        },
        "products": {
            "record_type": "delimited",
            "headers": [
                "id",
                "ean",
                "title",
                "category",
                "vendor",
                "price",
                "rating",
                "created_at"
            ],
            "record_delimiter": "\r\n",
            "num_header_lines": 1
        }
    },
    "transform": {
        "file_groups": {
            "orders": {
                "tables": {
                    "metabase.public.orders": {
                        "columns": {
                            "id": "id",
                            "user_id": "user_id",
                            "product_id": "product_id",
                            "subtotal": "subtotal",
                            "tax": "tax",
                            "total": "total",
                            "discount": "discount",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                            "quantity": "quantity"
                        }
                    }
                }
            },
            "products": {
                "tables": {
                    "metabase.public.products": {
                        "columns": {
                            "id": "id",
```

```
                        "ean": "ean",
                        "title": "title",
                        "category": "category",
                        "vendor": "vendor",
                        "price": "price",
                        "rating": "rating",
                        "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                    }
                }
            }
        }
    }
}
```

The two interesting parts of this pipeline.json file are the way the file groups and the extraction settings are defined.

First, note that each file group sets the S3 endpoint, a bucket, a prefix used for filtering the considered files, and then a file matcher. In this example, there is only a single file, but if there were many files matching the pattern `**orders*.csv` then they would all be part of the file group. You do not need S3 credentials because this is a public bucket, but if it were private, you can supply credentials in a few different ways.

Next, note the extract section. Here, the example specifies a delimited format. If it were compressed (e.g., gzip), you could specify compression. The example also specifies the headers to associate with each column in the CSV data. Delimited extracts can specify different record delimiters (e.g., `\n`), specify field delimiters (e.g., `|`, `\t`, `,`), define how to handle empty fields or to trim whitespace, and specify strings that should be considered `NULL`. While not used in this file, an example of null strings is provided that would turn the string literal “NULL” or “N/A” into a database `NULL`.

The final parameter supplied in the example is the sort type for the file load. This informs the LAT how you would like data to be ordered when loading. The ideal sort organizes files in time order according to the defined TimeKey®. This makes more efficient segments and is much faster to load. This example uses the lexicographic sort which orders according to the characters in the file name. Other sort types are

available to use file modified time or to extract the timestamp for sorting from the file path or file name.

Step 4: Using the Loading and Transformation CLI

With a pipeline.json file ready to go, you can test this pipeline. To test, use the LAT CLI. For these examples, assume that two LATs are configured and set using an environment variable.

First, configure the LAT CLI to use the hosts of the Ocient Loading and Transformation service. You can add these to every CLI command as a flag, but for simplicity you can also set them as environment variables. From a command line, run the following command replacing the IP addresses with the IP addresses of your LAT processes:

Shell

```
export LAT_HOSTS="https://10.0.0.1:8443,https://10.0.0.2:8443"
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://`.

Next, check on the status of the LAT:

Shell

```
lat_client pipeline status
```

Example response:

Bash

```
10.0.0.1:8443: Stopped
10.0.0.2:8443: Stopped
```

Success! This confirms that you can reach the LAT from your CLI. If the status is “Running” it means a pipeline is already executing a pipeline. In the next step, you will

update and start the new pipeline.

This example uses secure connections. If you receive an SSL Error when testing, your service cannot be configured to use TLS or you might need to use the `--no-verify` flag if certificate validation fails.

Step 5: Test the Transformation

The CLI supports previewing a transformation with an example document and the pipeline file. This makes it easy to test your transformations.

First, save an example document to your file system to use for this test. For this demo, you can download an example file from **https://ocient-docs.s3.amazonaws.com/metabase_samples/csv/orders.csv** and save it to `~/orders.csv`.

Next, make sure the pipeline.json file that you created is stored at `~/pipeline.json`.

Now that both files are available, you can run the CLI to preview the results. Pass the preview command the topic name, the pipeline file, and the sample record file. The response contains the transformed data tied to the destination table and a list of any error records.

Similar to how you can preview records on a Apache® Kafka® topic for file loads, you can supply any one `file_groups` created in the extract section to preview the transformations.

Shell

```
lat_client preview --topic orders --pipeline ~/pipeline.json --records ~/orders.c
```

Example response:

© 2026 Ocient, Inc.

JSON


```
{
    "tableRecords": {
        "metabase.public.orders": [
            {
                "id": 1,
                "user_id": 1,
                "product_id": 14,
                "subtotal": 37.65,
                "tax": 2.07,
                "total": 39.72,
                "discount": null,
                "created_at": 1549921227892000000,
                "quantity": 2
            },
            {
                "id": 2,
                "user_id": 1,
                "product_id": 123,
                "subtotal": 110.93,
                "tax": 6.1,
                "total": 117.03,
                "discount": null,
                "created_at": 1526371444580000000,
                "quantity": 3
            },
            {
                "id": 3,
                "user_id": 1,
                "product_id": 105,
                "subtotal": 52.72,
                "tax": 2.9,
                "total": 49.2,
                "discount": 6.42,
                "created_at": 1575670968544000000,
                "quantity": 2
            }
        ]
    },
    "recordErrors": []
}
```


© 2026 Ocient, Inc.

You can see that the data is transformed and the columns to which each transformed value will be mapped. If there are issues in the values, these will appear in the `recordErrors` object. You can quickly update the pipeline.json file and preview again. Now, you can inspect different documents to confirm that various states of data cleanliness like missing columns, null values, and special characters are well handled by your transformations.

Step 6: Configure and Start the Data Pipeline

With a tested transformation, the next step is to setup and start the data pipeline.

First, configure the pipeline using the `pipeline create` command. This validates and creates the pipeline, but will not take effect until you start the pipeline:

Shell

```
lat_client pipeline create --pipeline ~/pipeline.json
```

Example response:

Bash

```
10.0.0.1:8443: Created
10.0.0.2:8443: Created
```

> In cases where there is an existing pipeline operating, it is necessary to stop the pipeline and remove the original pipeline before creating and starting the new pipeline.

Now that the pipeline has been created on all LAT Nodes, you can start the LAT by running the `pipeline start` commands:

Shell

```
lat_client pipeline start
```

Example responses:

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

Step 7: Confirm that Loading is Operating Correctly

With your pipeline in place and running, data will immediately begin loading from the S3 file groups that you defined. If there were many files per file group, the LAT would first sort the files, then partition them for the fastest loading based on the sorting criteria you provided.

Observing Loading Progress

With the pipeline running, data immediately begins to load into Ocient. To observe this progress, you can use the `pipeline status` command from the LAT Client or monitor the LAT metrics endpoint of the Loader Nodes.

You can check the status with this command by using the `--list-files` flag to include a summary of the files included in the load.

```shell
lat_client pipeline status --list-files
```

Example responses:

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running

Pipeline Files Processed: 0
Pipeline Error Count: 0
Pipeline Files Remaining: 2

orders
  Files Processed: 0
  Error Count: 0
  Files Remaining: 1

products
  Files Processed: 0
  Error Count: 0
  Files Remaining: 1

orders

Status         Filename
processing     metabase_samples/csv/orders.csv

In Process Files:
processing     metabase_samples/csv/orders.csv

products

Status         Filename
processing     metabase_samples/csv/products.csv

In Process Files:
processing     metabase_samples/csv/products.csv
```

You can monitor the LAT metrics endpoint manually at the command line. Or, you can use a tool like Grafana® to retrieve metrics. For this example, run the curl command against the endpoint and review the result. For details on metrics, see the **LAT Metrics** Documentation.

Command:

CURL

```
curl https://127.0.0.1:8443/v2/metrics/lat:type=pipeline
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://` .

Example response:

JSON


```
{
  "request": {
  "mbean": "lat:type=pipeline",
  "type": "read"
  },
  "value": {
  "partitions": [
    {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 18759,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 18759,
        "partition": "table_orders-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 18759,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 88,
        "records_errors_total": 0,
        "offsets_buffered": 18759,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
    },
    {
    "offsets_durable": 1,
    "pushes_errors": 0,
    "pushes_attempts": 1,
    "rows_pushed": 1,
    "offsets_written": 199,
```


© 2026 Ocient, Inc.

```
    "records_buffered": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "source_bytes_buffered": 0,
    "records_errors_transformation": 0,
    "offsets_processed": 199,
    "partition": "table_products-0",
    "records_filter_accepted": 1,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,
    "offsets_pushed": 199,
    "pushes_unacknowledged": 0,
    "invalid_state": 0,
    "bytes_pushed": 145,
    "records_errors_total": 0,
    "offsets_buffered": 199,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
    }
  ],
  "paused": 1,
  "bytes_buffered": 0,
  "workers": 20
  },
  "timestamp": 1626970368,
  "status": 200,
}
```


Check Row Counts in Tables

To confirm that you are seeing results in the target tables, you can also run some simple queries to check row counts. Depending on the streamloader role settings, the time for records to become queryable can vary from a few seconds to minutes:

Example Queries:

© 2026 Ocient, Inc.

SQL

```
Ocient> SELECT count(*) FROM public.orders;
COUNT(*)
--------------------
18760
```

SQL

```
Ocient> SELECT count() FROM public.products;
COUNT()
--------------------
200
```

Now you can explore the data in these four tables with any Ocient SQL queries.

Check Errors

In this example, all rows load successfully. However, a successful load does not always happen, and you can inspect errors using the LAT Client. Whenever the LAT process fails to parse a file correctly or fails to transform or load a record, the LAT process records an error. The LAT Client includes the `lat_client pipeline errors` command that reports the latest errors on the pipeline.

A full error log is available on the Loader Nodes. These logs report all bad records and the reason that the load fails.

> When you load a pipeline from Kafka, the load might route errors to an error topic on the Kafka broker instead of the logs. The LAT Client does not contain the errors sent to the error topic. You can inspect these errors with Kafka utilities instead.

This LAT Client command displays a maximum of 100 error messages.

© 2026 Ocient, Inc.

```
lat_client pipeline errors --max-errors 100 --only-error-messages

|-------------------------------------------------|
| exception_message                               |
|-------------------------------------------------|
| Column name: time1. Message: Failed to evaluate  |
| expression. Cause:                              |
| java.time.format.DateTimeParseException         |
|-------------------------------------------------|
| Column name: time1. Message: Failed to evaluate  |
| expression. Cause:                              |
| java.time.format.DateTimeParseException         |
|-------------------------------------------------|
```

The errors indicate that there is an issue parsing the `time1` column. Options exist on the `pipeline errors` command to return JSON and to restrict the response to specific components of the error detail that includes a reference to the source location of this record.

The following command returns JSON that is delimited with newline characters. You can pass the JSON output to `jq` or a file. The JSON includes the source topic or file group, the filename where the error occurred, the offset that indicates the line number or Kafka offset, and the exception message that aids in troubleshooting and identifying the incorrect record in the source data. You can use the `log_original_message` pipeline setting to provide direct access to the parsed source record for errors when appropriate.

```
lat_client pipeline errors --max-errors 100 --json

{"time": "2022-05-17T16:53:50.387386+00:00", "topic": "calcs", "partition": 0, "s
{"time": "2022-05-17T16:53:50.404684+00:00", "topic": "calcs", "partition": 0, "s
```

Related Links

LAT Overview

LAT Data Types in Loading

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Load JSON Data from S3

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

A common setup for batch loading files into Ocient is to load from a bucket on Amazon® Web Services℠ (AWS℠) S3 with time partitioned data. In many instances, a batch load is performed on a recurring basis to load new files. The LAT transforms each document into rows in one or more different tables. Ocient's Loading and Transformation capabilities use a simple SQL-like syntax for transforming data. This tutorial will guide users through a simple example load using a small set of data in JSONL (newline delimited JSON) format. The data in this example is created from a test set for the Metabase℠ Business Intelligence tool.

Prerequisites

This tutorial assumes that:

1. The Ocient System has network access to S3 from the Loader Nodes.
2. An Ocient System is installed and configured with an active storage cluster (See the **Ocient Application Configuration** guide).
3. Loading and Transformation is installed on the Loader Nodes.
4. A default "sink" for the Ocient Loader Nodes is configured on the system.
5. The **LAT Client Command Line Interface** is installed.

Step 1: Create a New Database

To begin, you are going to load two example tables in a database. First, connect to a SQL Node using the **Commands Supported by the Ocient JDBC CLI Program**. Then run the following DDL command:

© 2026 Ocient, Inc.

SQL

```
CREATE DATABASE metabase;
```

Step 2: Create Tables

To create tables in the new database, first connect to that database (e.g., `connect to jdbc:ocient://sql-node:4050/metabase`), then run the following DDL commands:

SQL

```
CREATE TABLE public.orders (
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  subtotal DOUBLE,
  tax DOUBLE,
  total DOUBLE,
  discount DOUBLE,
  quantity INT,
  CLUSTERING INDEX idx01 (user_id, product_id)
);

CREATE TABLE public.products(
  created_at TIMESTAMP TIME KEY BUCKET(1, DAY) NOT NULL,
  id INT NOT NULL,
  ean VARCHAR(255),
  title VARCHAR(255),
  category VARCHAR(255) COMPRESSION GDC(2) NOT NULL,
  vendor VARCHAR(255),
  price DOUBLE,
  rating DOUBLE,
  CLUSTERING INDEX idx01 (category)
);
```

Now, the database tables are created and you can begin loading data.

© 2026 Ocient, Inc.

Step 3: Create a Data Pipeline

Data pipelines are created using a simple loading configuration that is submitted to the Transformation Nodes to start loading. File Groups designate a batch of files to load. Each File Group is routed to one or more Ocient tables, and each column is the result of a transformation applied to the source document.

First, let's inspect the data you plan to load. Each document has a format similar to the following example:

JSON

```
/* orders */
{"id": 1, "user_id": 1, "product_id": 14, "subtotal": 37.65, "tax": 2.07, "total'
{"id": 2, "user_id": 1, "product_id": 123, "subtotal": 110.93, "tax": 6.1, "total
...

/* products */
{"id": 1, "ean": "1018947080336", "title": "Rustic Paper Wallet", "category": "Gi
{"id": 2, "ean": "7663515285824", "title": "Small Marble Shoes", "category": "Doc
{"id": 3, "ean": "4966277046676", "title": "Synergistic Granite Chair", "category
...
```

As you can see, this is similar to your target schema, but will require some transformation. Most transformations are identical to functions already in Ocient's SQL dialect. To route data to your tables, you need to create a pipeline.json file that has the following structure:

JSON

```
{
    "version": 2,
    "workers": 4,
    "source": {
        "type": "s3",
        "endpoint": "https://s3.us-east-1.amazonaws.com",
        "bucket": "ocient-docs",
        "compression": "none",
        "file_groups": {
            "orders": {
                "prefix": "metabase_samples/jsonl",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**orders*.jsonl",
                "sort_type": "lexicographic"
            },
            "products": {
                "prefix": "metabase_samples/jsonl",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**products*.jsonl",
                "sort_type": "lexicographic"
            }
        }
    },
    "transform": {
        "file_groups": {
            "orders": {
                "tables": {
                    "metabase.public.orders": {
                        "columns": {
                            "id": "id",
                            "user_id": "user_id",
                            "product_id": "product_id",
                            "subtotal": "subtotal",
                            "tax": "tax",
                            "total": "total",
                            "discount": "discount",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                            "quantity": "quantity"
                        }
                    }
                }
```

```
            },
            "products": {
                "tables": {
                    "metabase.public.products": {
                        "columns": {
                            "id": "id",
                            "ean": "ean",
                            "title": "title",
                            "category": "category",
                            "vendor": "vendor",
                            "price": "price",
                            "rating": "rating",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                        }
                    }
                }
            }
        }
    }
}
```

The most interesting part of this pipeline.json file is the way it defines the file groups. Note that each sets the S3 endpoint, a bucket, a prefix used for filtering the considered files, and then a file matcher. In this case you only have a single file, but if there were many files matching the pattern `**orders*.jsonl` then they would all be part of the file group.

The final parameter that you supplied is the sort type for the file load. This informs the LAT how you would like data to be ordered when loading. The ideal sort is in time order according to the defined TimeKey®. This makes more efficient segments and is much faster to load. In this case, you used the lexicographic sort which orders according to the characters in the file name. Other sort types are available to use file modified time or to extract the timestamp for sorting from the file path or file name.

Step 4: Using the Loading and Transformation CLI

With a pipeline.json file ready to go, you can test this pipeline. To test, use the LAT CLI. For these examples, assume that two LATs are configured and set using an

environment variable.

First configure the LAT CLI to use the hosts of the Ocient Loading and Transformation service. You can add these to every CLI command as a flag, but for simplicity you can also set them as environment variables. From a command line, run the following command replacing the IP addresses with the IP addresses of your LAT processes:

Shell

```
export LAT_HOSTS="https://10.0.0.1:8443,https://10.0.0.2:8443"
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://` .

Next, check on the status of the LAT:

Shell

```
lat_client pipeline status
```

Example response:

Bash

```
10.0.0.1:8443: Stopped
10.0.0.2:8443: Stopped
```

This confirms that you can reach the LAT from your CLI. If the status is “Running” it means a pipeline is already executing a pipeline. Next, you are going to update and start your new pipeline.

This example uses secure connections. If you receive an SSL Error when testing, your service might not be configured to use TLS or you might need to use the `--no-verify` flag if the certificate validation fails.

Step 5: Test the Transformation

The CLI supports previewing a transformation with an example document and the pipeline file. This makes it easy to test your transformations.

First, save an example document to your file system to use for this test. For this demo, you can download an example file from **https://ocient-docs.s3.amazonaws.com/metabase_samples/jsonl/orders.jsonl** and save it to `~/orders.jsonl`.

Next, make sure the pipeline.json file that you created is stored at `~/pipeline.json`.

Now that both files are available, you can run the CLI to preview the results. Pass the preview command the topic name, the pipeline file, and the sample record file. The response contains the transformed data tied to the destination table and a list of any error records.

Similar to how you can preview records on a Apache® Kafka® topic for file loads, you can supply any one of the topics you created as file groups to preview the transformations.

Shell

```
lat_client preview --topic orders --pipeline ~/pipeline.json --records ~/orders.j
```

Example response:

JSON


```
{
    "tableRecords": {
        "metabase.public.orders": [
            {
                "id": 1,
                "user_id": 1,
                "product_id": 14,
                "subtotal": 37.65,
                "tax": 2.07,
                "total": 39.72,
                "discount": null,
                "created_at": 1549921227892000000,
                "quantity": 2
            },
            {
                "id": 2,
                "user_id": 1,
                "product_id": 123,
                "subtotal": 110.93,
                "tax": 6.1,
                "total": 117.03,
                "discount": null,
                "created_at": 1526371444580000000,
                "quantity": 3
            },
            {
                "id": 3,
                "user_id": 1,
                "product_id": 105,
                "subtotal": 52.72,
                "tax": 2.9,
                "total": 49.2,
                "discount": 6.42,
                "created_at": 1575670968544000000,
                "quantity": 2
            }
        ]
    },
    "recordErrors": []
}
```


© 2026 Ocient, Inc.

You can see that the data is transformed and the columns to which each transformed value will be mapped. If there are issues in the values, these will appear in the `recordErrors` object. You can quickly update your pipeline.json file and preview again. Now, you can inspect different documents to confirm that various states of data cleanliness like missing columns, null values, and special characters are well handled by your transformations.

Step 6: Configure and Start the Data Pipeline

With a tested transformation, the next step is to setup and start the data pipeline.

First, you must configure the pipeline using the `pipeline create` command. This validates and creates the pipeline, but will not take effect until you start the pipeline:

Shell

```
lat_client pipeline create --pipeline ~/pipeline.json
```

Example response:

Bash

```
10.0.0.1:8443: Created
10.0.0.2:8443: Created
```

> In cases where there is an existing pipeline operating, it is necessary to stop the pipeline and remove the original pipeline before creating and starting the new pipeline.

Now that the pipeline has been created on all LAT Nodes, you can start the LAT by running the `pipeline start` commands:

Shell

```
lat_client pipeline start
```

Example responses:

```bash
10.0.0.1:8443: Running
10.0.0.2:8443: Running
```

Step 7: Confirm that Loading is Operating Correctly

With your pipeline in place and running, data will immediately begin loading from the S3 file groups you defined. If there were many files per file group, the LAT would first sort the files, then partition them for the fastest loading based on the sorting criteria you provided.

Observing Loading Progress

With the pipeline running, data immediately begins to load into Ocient. To observe this progress, you can use the `pipeline status` command from the LAT Client or monitor the LAT metrics endpoint of the Loader Nodes.

You can check the status with this command by using the `--list-files` flag to include a summary of the files included in the load.

```shell
lat_client pipeline status --list-files
```

Example responses:

Bash

```
10.0.0.1:8443: Running
10.0.0.2:8443: Running

Pipeline Files Processed: 0
Pipeline Error Count: 0
Pipeline Files Remaining: 2

orders
  Files Processed: 0
  Error Count: 0
  Files Remaining: 1

products
  Files Processed: 0
  Error Count: 0
  Files Remaining: 1

orders

Status        Filename
processing    metabase_samples/jsonl/orders.jsonl

In Process Files:
processing    metabase_samples/jsonl/orders.jsonl

products

Status        Filename
processing    metabase_samples/jsonl/products.jsonl

In Process Files:
processing    metabase_samples/jsonl/products.jsonl
```

Command:

CURL

```
curl https://127.0.0.1:8443/v2/metrics/lat:type=pipeline
```

If your LAT is running without TLS configured, replace the port number of your LAT Hosts with 8080 and the protocol with `http://`.

Example response:

JSON


```
{
  "request": {
  "mbean": "lat:type=pipeline",
  "type": "read"
  },
  "value": {
  "partitions": [
    {
        "offsets_durable": 1,
        "pushes_errors": 0,
        "pushes_attempts": 1,
        "rows_pushed": 1,
        "offsets_written": 18759,
        "records_buffered": 0,
        "records_errors_column": 0,
        "records_errors_deserialization": 0,
        "source_bytes_buffered": 0,
        "records_errors_transformation": 0,
        "offsets_processed": 18759,
        "partition": "table_orders-0",
        "records_filter_accepted": 1,
        "records_errors_row": 0,
        "records_filter_rejected": 0,
        "records_errors_generic": 0,
        "producer_send_attempts": 0,
        "offsets_pushed": 18759,
        "pushes_unacknowledged": 0,
        "invalid_state": 0,
        "bytes_pushed": 88,
        "records_errors_total": 0,
        "offsets_buffered": 18759,
        "complete": 0,
        "offsets_end": 1,
        "producer_send_errors": 0
    },
    {
    "offsets_durable": 1,
    "pushes_errors": 0,
    "pushes_attempts": 1,
    "rows_pushed": 1,
    "offsets_written": 199,
```


© 2026 Ocient, Inc.

```
    "records_buffered": 0,
    "records_errors_column": 0,
    "records_errors_deserialization": 0,
    "source_bytes_buffered": 0,
    "records_errors_transformation": 0,
    "offsets_processed": 199,
    "partition": "table_products-0",
    "records_filter_accepted": 1,
    "records_errors_row": 0,
    "records_filter_rejected": 0,
    "records_errors_generic": 0,
    "producer_send_attempts": 0,
    "offsets_pushed": 199,
    "pushes_unacknowledged": 0,
    "invalid_state": 0,
    "bytes_pushed": 145,
    "records_errors_total": 0,
    "offsets_buffered": 199,
    "complete": 0,
    "offsets_end": 1,
    "producer_send_errors": 0
    }
  ],
  "paused": 1,
  "bytes_buffered": 0,
  "workers": 20
  },
  "timestamp": 1626970368,
  "status": 200
}
```


Check Row Counts in Tables

To confirm that you are seeing results in the target tables, you can also run some simple queries to check row counts. Depending on the streamloader role settings, the time for records to become queryable can vary from a few seconds to minutes:

Example Queries:

SQL

```
Ocient> SELECT count(*) FROM public.orders;
COUNT(*)
----------------------
18760
```

SQL

```
Ocient> SELECT count() FROM public.products;
COUNT()
--------------------
200
```

Now you can explore the data in these four tables with any Ocient SQL queries.

Check Errors

In this example, all rows load successfully. However, a successful load does not always happen, and you can inspect errors using the LAT Client. Whenever the LAT process fails to parse a file correctly or fails to transform or load a record, the LAT process records an error. The LAT Client includes the `lat_client pipeline errors` command that reports the latest errors on the pipeline.

A full error log is available on the Loader Nodes. These logs report all bad records and the reason that the load fails.

> When you load a pipeline from Kafka, the load might route errors to an error topic on the Kafka broker instead of the logs. The LAT Client does not contain the errors sent to the error topic. You can inspect these errors with Kafka utilities instead.

This LAT Client command displays a maximum of 100 error messages.

```
lat_client pipeline errors --max-errors 100 --only-error-messages

|-------------------------------------------------|
| exception_message                               |
|-------------------------------------------------|
| Column name: time1. Message: Failed to evaluate  |
| expression. Cause:                              |
| java.time.format.DateTimeParseException         |
|-------------------------------------------------|
| Column name: time1. Message: Failed to evaluate  |
| expression. Cause:                              |
| java.time.format.DateTimeParseException         |
|-------------------------------------------------|
```

The errors indicate that there is an issue parsing the `time1` column. Options exist on the `pipeline errors` command to return JSON and to restrict the response to specific components of the error detail that includes a reference to the source location of this record.

The following command returns JSON that is delimited with newline characters. You can pass the JSON output to `jq` or a file. The JSON includes the source topic or file group, the filename where the error occurred, the offset that indicates the line number or Kafka offset, and the exception message that aids in troubleshooting and identifying the incorrect record in the source data. You can use the `log_original_message` pipeline setting to provide direct access to the parsed source record for errors when appropriate.

```
lat_client pipeline errors --max-errors 100 --json

{"time": "2022-05-17T16:53:50.387386+00:00", "topic": "calcs", "partition": 0, "s
{"time": "2022-05-17T16:53:50.404684+00:00", "topic": "calcs", "partition": 0, "s
```

Related Links

LAT Overview

LAT Data Types in Loading

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Advanced Loading and Transformations

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

This guide presents examples of some more advanced transformation topics such as exploding data, loading multiple tables from a single topic, working with arrays, and accessing nested data.

Loading Multiple Tables from a Topic

The **LAT Load JSON Data from Kafka** guide shows an example of loading each topic into its own table. However, each topic can route data to multiple tables and many Apache® Kafka® topics can stream load simultaneously to one or more tables in a single pipeline.

Here is an example `pipeline.json` file that will load data into two separate tables:

© 2026 Ocient, Inc.

JSON

```
{
    "version": 2,
    "pipeline_id": "pipeline-metabase",
    "source": {
        "type": "kafka",
        "kafka": {
          "bootstrap.servers": "127.0.0.1:9092",
        }
    },
    "transform": {
        "topics": {
            "products": {
                "tables": {
                    "metabase.public.products": {
                        "columns": {
                            "id": "id",
                            "ean": "ean",
                            "title": "title",
                            "category": "category",
                            "vendor": "vendor",
                            "price": "price",
                            "rating": "rating",
                            "created_at": "to_timestamp(created_at, 'yyyy-MM-dd\\
                        }
                    },
                    "metabase.public.product_ratings": {
                        "columns": {
                            "created_at": "now()",
                            "product_id": "id",
                            "rating": "rating",
                        }
                    }
                }
            }
        }
    }
}
```

A few key points here.

1. You are loading data from a Kafka topic called `products`.
2. That topic is loading into a table called `database.schema.table`.
3. Most columns are simply mapped from the source JSON document into, however some are transformed from the original data.
    a. `created_at` is parsed from a string using the `to_timestamp` function
    b. `created_at` for `product_ratings` is set dynamically to the value of the `now()` function

Loading Nested JSON Data

As you can see from the transformation, the syntax is similar to SQL, but the inner language used to traverse the JSON document and manipulate arrays can be unfamiliar. The approach used to select a property from the source JSON is to follow the tree with dot notation. For example:

Given a series of two documents like:

© 2026 Ocient, Inc.

JSON


```
{
  "country" : "US",
  "states" : ["AR", "VA", "MD", "IL"],
  "languages" : [
    {
      "name" : "English",
      "locale" : "en-US",
      "dialects" : ["Northern American English", "Southeast Super-Regional Englis
    },
    {
      "name" : "Spanish",
      "locale" : "es-US",
      "dialects" : ["Mexican Spanish", "Caribbean Spanish", "Central American Spa
    }
  ],
  "capital" : {
    "city" : "Washington",
    "region" : "District of Columbia"
  },
  "founded_in" : 1776
}

{
  "country" : "CA",
  "provinces" : ["BC", "ON", "QC", "MB"],
  "languages" : [
    {
      "name" : "English",
      "locale" : "en-CA"
    },
    {
      "name" : "French",
      "locale" : "fr-CA"
    }
  ],
  "capital" : {
    "city" : "Ottawa",
    "region" : "Ontario"
  },
```


© 2026 Ocient, Inc.

```
  "founded_in" : 1867
}
```

You can access a property like the country simply by referencing the property:

JSON

```
/* Ocient transformations are in this format */
{
  "column_name" : "transformation"
}

/* to apply the country property to the country_code column */
{
  "country_code" : "country"
}
```

You can access nested properties easily as well:

JSON

```
/* sets the column capital_city to values like "Ottawa" */
{
  "capital_city" : "capital.city"
}
```

If you want to join the capital sub-properties together, you can easily do so:

JSON

```
/* sets the column capital_city to values like "Ottawa, Ontario" */
{
  "capital_city" : "concat(capital.city, ',', capital.region)"
}
```

Note that these two documents are not fully cleansed, and you want to capture the states or provinces array and assign it to a common destination column, regions. You can do this with a coalesce operator `||`:

JSON

```
/* sets the column regions to ["AR", "VA", "MD", "IL"] for the US and ["BC", "ON"
{
  "regions" : "states || provinces"
}
```

Loading Array Data

Where arrays are encountered, special array notation is used. The benefit of this syntax is that properties in objects in arrays can also be accessed to return an array.

Another useful capability is the flattening of arrays. In the event that the values returned from a transformation includes nested arrays, you can flatten it to a single dimension array by appending `[]` to the end of the transformation expression. For example:

JSON

```
/* For the US, this transformation would return a structure like: [ ["", ""], ["
{
  "regions" : "languages.dialects"
}

/* You can flatten this to return this one-dimensional array with structure like:
{
  "regions" : "languages.dialects[]"
}
```

Exploding Array Data

In some cases, a single record from a loading source should represent more than one row in the target table. This is referred to as exploding the data. This allows nested array data to be expanded and to be associated with the corresponding values from the record.

For example, if you would like to “explode” the data in the languages array into a single row per value to populate a standalone “country_languages” table, you could do

the following for the `transform` section of your `pipeline.json` file:

JSON

```
{
  "transform": {
    "tables": {
      "my_database.my_schema.country_languages": {
        "columns": {
          "country_name": "country",
          "language_name": "EXPLODE(languages[].name)",
          "dialects": "EXPLODE(languages[].dialects)",
          "locale": "EXPLODE(languages[].locale)"
        }
      }
    }
  }
}
```

This would result in the following output. Note that instead of the original 2 records, this result will produce four rows into the new database table. The “exploded” values are unique, but the records inherit their parent values like `country` from the record when the explode occurs:

Example Rows:

JSON


```
{
  "country" : "US",
  "language_name" : "English",
  "dialects" : ["Northern American English", "Southeast Super-Regional English",
  "locale": "en-US"
}
{
  "country" : "US",
  "language_name" : "Spanish",
  "dialects" : ["Mexican Spanish", "Caribbean Spanish", "Central American Spanish
  "locale": "es-US"
}
{
  "country" : "CA",
  "language_name" : "English",
  "dialects" : NULL,
  "locale" : "en-CA"
}
{
  "country" : "CA",
  "language_name" : "French",
  "dialects" : NULL,
  "locale" : "fr-CA"
}
```


If multiple columns in the transformation include an `EXPLODE`, they results will be "zipped" together. If the arrays are different lengths, the number of rows will match the longest array. The shorter array will populate the end of its rows with NULL. This is sometimes referred to as an "Outer Explode."

Transforming and Joining String Data

One common requirement is to split a string on a token to make an array. This is accomplished using transformation functions in the LAT.

Assuming a data input with a space separated string:

© 2026 Ocient, Inc.

JSON

```
{ "states": "AL AK AZ AR CA CO CT DE" }
```

You can extract an array of state values in a few ways using the `tokenize` function. Tokenize has a number of flags to control how the pattern should be interpreted. For example `'q'` indicates the pattern is a literal and `'i'` indicates case insensitive regular expression.

JSON

```
/* You can use a literal space to split and load ["AL", "AK", "AZ", "AR", "CA", "
{
  "states_col" : "tokenize(states, ' ', 'q')"
}

/* You can use a regular expression for whitespace to split and load ["AL", "AK",
{
  "states_col" : "tokenize(states, '\\s', '')"
}
```

For more complex situations, you can split on any regular expression such as cases where values are wrapped with double quotes and might contain spaces. For this, you can use the expression `(?=(([^\\'\"]*[\\'\"]){2})*[^\\'\"]*$)` that leverages a zero-width positive lookahead assertion. Note that because this regex will be presented in a JSON document, certain characters must be escaped within the JSON string. For example, in JSON:

JSON

```
{
  "my_column" : "tokenize(states, ' (?=(([^\\'\"]*[\\'\"]){2})*[^\\'\"]*$)', '')'
}
```

This would actually pass the following regular expression to the tokenize function when the backslash escape sequences are processed:

Text

```
(?=(([^'"]*['"]){2})*[^'"]*$)
```

Similarly, note that double backslashes are required to provide escape characters in the regular expression. For example, `'\\s'` would actually pass the regular expression `\s` to the tokenize function.

Similar to splitting on a token, the LAT can easily join an array into a string with the `join` function. With source data that includes an array like the following:

JSON

```
{ "state_names": ["Alabama", "Alaska", "Arizona", "Arkansas", "California"] }
```

You can use the join function to produce this string:

JSON

```
/* join takes a string delimiter and will produce "Alabama, Alaska, Arizona, Arka
{
    "state_names_col": "join(', ', state_names)"
}
```

Related Links

LAT Overview

LAT Data Types in Loading

LAT Advanced Topics

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Overview

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

The Ocient Loading and Transformation (LAT) is a service used for transforming and streaming data into Ocient. The LAT transforms records from a configured Source using the **JMESPath** transformation language and streams records into Ocient tables.

LAT Benefits

- Exactly-once processing from a Source
- Selecting data and arrays from nested JSON data
- Transforming records and routing into one or more tables
- Scalar data transformation (e.g., concatenate strings, uppercasing, rounding)
- Dynamic schema change handling

LAT Data Flow

The LAT consumes JSON records, transforms them, and loads them into one or more Ocient tables. This process is referred to as a “pipeline”, with a single pipeline running per LAT instance. The pipeline is described by a JSON-based configuration that instructs the LAT how to consume records from a Source, how to transform the records, and which Ocient table to route the records to. As records are consumed, the transformations described in the configuration are applied to each record before forwarding the resulting records to Ocient.

Each pipeline configuration can map a record to 1 or more Ocient Tables, each table with its own transformation configuration. LAT can optionally be configured to write records which were unable to be processed to an Apache® Kafka® error topic.

© 2026 Ocient, Inc.

Multiple LAT instances can be used in tandem to further parallelize a pipeline. The LAT makes use of the Kafka Consumer library to ensure that each Kafka Partition is only assigned to a single instance. To use multiple instances, configure each LAT instance with the same pipeline configuration. The LAT instances will automatically coordinate and distribute the load across the instances.

The LAT is controlled by an HTTP endpoint and a command line interface (CLI) is supplied to perform common actions such as starting, stopping, or updating the LAT pipelines.

Related Links

Connect to Ocient

Ingest Data with Legacy LAT Reference

Related Links

At the Whiteboard with Ocient: Loading and Transformation

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Data Types in Loading

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

After extracting from source files or topics, data can be transformed and mapped to target columns in Ocient tables.

LAT automatically binds many primitive source data types to Ocient column data types and automatically converts some source types to the target column type. Different source data types automatically bind to target columns using different conventions.

For example, a string in JSON such as `'123'` that loads into an `INT` column properly binds to the corresponding integer value `123`. Similarly, an integer like `456` can automatically bind to an `INT` column as `456`, a `FLOAT` column as `456.0`, and a `VARCHAR` column as `'456'`. Finally, a string including Well-Known Text (WKT) representation of a `POINT` automatically binds into an `ST_POINT` column.

Empty String Handling

When you attempt to load an empty string into a target column with LAT, most data types raise an error and the record does not load into the target table. The exceptions are `varchar`, `decimal`, `binary`, and `array` column types. In those cases, the data warehouse loads these values:

- `varchar`: An empty string
- `decimal`: 0.0 (with the appropriate precision)
- `binary`: An empty binary blob
- `array`: An empty array

> ⓘ When you extract data from text-based formats like delimited data, the data warehouse loads empty data as an empty string unless you use the `empty_as_null` setting.

NULL Handling

© 2026 Ocient, Inc.

If you do not provide a value for a column in the target table, this table explains what value the data warehouse stores in the column. For the purposes of default value handling, these are equivalent:

- Pipeline is missing a column.
- Pipeline includes a column but the value in the JSON or the source data evaluates to `null`.
- Pipeline includes a column but the field is missing on the source record (e.g., `a.b` does not exist in a JSON record).

To transform a NULL or missing value into a default in the pipeline, use the transformation functions. For example: `not_null(a.b, 'my_default')`

Ocient Column Definition	Database Behavior When Column is Missing or Column Value Evaluates to NULL
NOT NULL with default	Default
NOT NULL with no default	Record fails
Nullable with default	Default
Nullable with no default	NULL

Data Type Binding

Refer to the **Data Types** for detailed information about each data type. These tables describe the automatic binding conventions of each data type.

`BIGINT`

Valid JSON Bind Types:	Number, String
Note:	Number and String can be either integer or floating point values and are converted to 64-bit signed integers. Decimals are dropped. '+','-' can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent.
Example:	{ "number_bigint": 1.0, "string_bigint": "1"

Valid JSON Bind Types:	**Number, String**
	} -> 1, 1

`BINARY`

Valid JSON Bind Types:	**String**
Note:	Converted as UTF-8 to binary: each character in the string taking its binary representation.

Example:	{
	 “string_binary”: “hello”
	 }
	 -> BINARY(01101000 01100101 01101100 01101100 01101111)

`BOOLEAN`

Valid JSON Bind Types:	**Boolean, String**
Note:	For a value of true, a string must be one of (case insensitive): true, t, yes, on, y, 1. For a value of false, a string must be one of (case insensitive): false, f, no, off, n, 0.

Example:	{
	 “boolean”: true,
	 “string_0”: “yes”,
	 “string_1”: “FALSE”
	 }
	 -> true, true, false

`DATE`

Valid JSON Bind Types:	**String, Number**
Note:	If String, must be in format “YYYY-MM-DD”. If Number must be an EPOCH_DAY. Use the built-in TO_DATE function to convert a string with a different date format to an EPOCH_DAY and bind.

Valid JSON Bind Types:	**String, Number**

Example:	{
	“string_date”: “2021-11-04”,
	“epoch_date”: 0
	}
	-> DATE(2021-11-04), DATE(1970-01-01)

`DECIMAL`

Valid JSON Bind Types:	**Number, String**
Valid Logical Types:	Decimal
Note:	Number and String can be either integer or floating point values. Precision and scale for the loaded data are defined by the database table definition. The target column must support enough digits both before and after the decimal point to accommodate decimal values extracted from Fixed Width Binary or other record formats that support decimal data. If insufficient precision or scale exists in the target column, the data will fail to load.

Example:	{
	“number_decimal”: 1.0,
	“string_decimal”: “1.1”
	}
	-> DECIMAL(1.0), DECIMAL(1.1)

`DOUBLE` , `DOUBLE PRECISION`

Valid JSON Bind Types:	**Number, String**
Note:	Number and String can be either integer or floating point values.

Example:	{
	“number_double”: 1.0,
	“string_double”: “1”
	}
	-> 1.0, 1.0

`HASH`

Valid JSON Bind Types:	**String**
Note:	Converted as hexadecimal to binary: each character in the string must be a valid hexadecimal digit /[0-9a-f]+/i.

Example:	{
	“hex_string”: “0a1b”
	}
	-> HASH(0000 1010 0001 1011)

`INT`

Valid JSON Bind Types:	**Number, String**
Note:	Number and String can be either integer or floating point values and are converted to 32-bit signed integers. Decimals are dropped. ’+’,’-’ can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent.

Example:	{
	“number_int”: 1.0,
	“string_int”: “1”
	}
	-> 1, 1

`IPV4`

Valid JSON Bind Types:	**String**
Note:	Source data must be in the form: [0-255].[0-255].[0-255].[0-255]

Example:	{
	“string_ipv4”: “192.168.0.1”
	}
	-> IPV4(192.168.0.1)

`IP`

Valid JSON Bind Types:	**String**
Note:	Source string can be in IPV4 or IPV6 format. IPV6 can appear in full form as show in this example or in condensed form. Condensed form can use :: notation (e.g., 2001:cdba::3257:9652) and leading zeros in IPV6 hextets can also be omitt (e.g., 2001:0db8:0000:0000:0000:0000:3257:9652 → 2001:db8:0:0:0:0:3257:96 IPV6 can also appear as a string of 32 hex characters without colons (e.g., 2001cdba00000000000000032579652).

Example:	{
	“string_ip_ipv4”: “192.168.0.1”,
	“string_ip_ipv6”: “2001:cdba:0000:0000:0000:0000:3257:9652”
	}
	-> IP(192.168.0.1), IP(2001:cdba:0000:0000:0000:0000:3257:9652)

`LINESTRING`, `ST_LINESTRING`

Valid JSON Bind Types:	**String**
Note:	Must be in the WKT format “LINESTRING(LONG LAT, ...)” or “LINESTRING EMPTY”

Example:	{
	“string_st_linestring”: “LINESTRING(-71.064544 42.28787, -90.444444 89.562993)”
	}
	-> ST_LINESTRING(-71.064544 42.28787, -90.444444 89.562993)

`MATRIX`

Valid JSON Bind Types:	**Array[Array], String**
Note:	Arrays must be two-dimensional with the same rows and columns as specified in the table. String must be in JSON two-dimensional array format. Each element will be cast according the the element type’s rules in this section.

Example:	{
	“int_matrix”: [[0, 1, 2],[3, 4, 5],[6, 7, 8]],
	“string_matrix”: “[[0,1],[2,3]]”
	}
	-> MATRIX[[0,1,2],[3,4,5],[6,7,8]], MATRIX[[0,1],[2,3]]

`POINT`, `ST_POINT`

Valid JSON Bind Types:	**String**
Note:	Must be in the format “POINT(LONG LAT)“

Example:	{
	“string_st_point”: “POINT(-71.064544 42.28787)”
	}
	-> ST_POINT(-71.064544, 42.28787)

`POLYGON`, `ST_POLYGON`

> Ensure that polygons are oriented in your intended way when you load them as polygon orientation matters in Ocient semantics. If you intend all polygons to be oriented counter-clockwise, use the **st_forcepolygonccw** transform function to enforce counter-clockwise semantics.

Valid JSON Bind Types:	**String**
Note:	Must be in the WKT format “POLYGON((LONG LAT, ...))” or “POLYGON((LONG LAT, ...), (LONG LAT, ...))” or “POLYGON EMPTY”

Example:	{
	“string_st_polygon”: “POLYGON((-87.62 41.87, -89.40 43.07, -87.906 43.04, -87.62 41.87))”
	}
	-> ST_POLYGON((-87.62 41.87, -89.40 43.07, -87.906 43.04, -87.62 41.87))

`REAL`, `FLOAT`, `SINGLE PRECISION`

Valid JSON Bind Types:	**Number, String**
Note:	Number and String can be either integer or floating point values. Precision and scale are inherited from the table definition.

Example:	{
	“number_float”: 1.0,
	“string_float”: “1”
	}
	-> 1.0, 1.0

`SMALLINT`

Valid JSON Bind Types:	**Number, String**
Note:	Number and String can be either integer or floating point values and are converted to 16-bit signed integers. Decimals are dropped. '+','-' can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent.

Example:	{
	"number_smallint": 1.0,
	"string_smallint": "1"
	}
	-> 1, 1

`TIME`

Valid JSON Bind Types:	**Number, String**
Note:	String must be in the format: HH:MM:SS[.SSSSSSSSS]. Number must represent nanoseconds since midnight. Use the TO_TIME transformation function to convert an arbitrary string format to nanoseconds and bind.

Example:	{
	"number_time": 1000000000,
	"string_time": "07:33:10.0001"
	}
	-> TIME(00:00:01.0000), TIME(07:33:10.0001)

`TIMESTAMP`

Valid JSON Bind Types:	**Number, String**
Note:	String must be in the format: YYYY-MM-DD HH:MM:SS[.SSSSSSSSS]. Number must represent nanoseconds since epoch. Use the TO_TIMESTAMP transformation function to convert an arbitrary string format to nanoseconds and bind.

Valid JSON Bind Types:	**Number, String**

Example:	{
	“number_timestamp”: 1000000000,
	“string_timestamp”: “2021-11-04 07:33:10.0001”
	}
	-> TIMESTAMP(1970-01-01 00:00:01.0000), TIMESTAMP(2021-11-04 07:33:10.0001)

`TINYINT`, `BYTE`

Valid JSON Bind Types:	**Number, String**
Note:	Number and String can be either integer or floating point values and are converted to 8-bit signed integers. Decimals are dropped. ’+’,’-’ can be used at the beginning of a string to indicate sign. String overflows result in errors, while numeric overflows are silent.

Example:	{
	“number_smallint”: 1.0,
	“string_smallint”: “1”
	}
	-> 1, 1

`TUPLE`

Valid JSON Bind Types:	**Array, String**
Note:	String must be in JSON array format “[e0,e1,e2]”. Each element will be cast to the tuple element type at that element’s index from the Ocient table according to the rules for that element type in this section.

Example:	{
	“double_int”: [1.0, 1],
	“string_array”: [“Ocient”, [0, 1]],
	“string_tuple”: “[1.0, 1]”,
	“tuple_tuple”: [[1.0, 1],[“Ocient”, 2]]
	}
	-> TUPLE(DOUBLE, INT)(1.0, 1), TUPLE(STRING, INT[])(“Ocient”, INT[0, 1]), TUPLE(DOUBLE, INT)(1.0, 1), TUPLE(TUPLE(DOUBLE, INT), TUPLE(VARCHAR, INT)((1.0, 1), (“Ocient”, 1))

`TYPE[]`

Valid JSON Bind Types:	**Array, String**
Note:	String must be in JSON array format “[e0,e1,e2]”. Each element will be cast to the array element type from the Ocient table according to the rules for that element type in this section.

Example:	{
	“int_array”: [0,1,2],
	“2d_array”: [[0, 1, 2],[3, 4]],
	“string_array”: “[0,1,2]”,
	“tuple_array”: [[0, 1.0, “Ocient”],[1, 2.0, “rocks”]]
	}
	-> INT[0,1,2], INT[][INT[0,1,2],INT[3,4]], INT[0,1,2] TUPLE(INT, FLOAT, VARCHAR)[(0, 1.0, “Ocient”), (1, 2.0. “rocks”)]

`UUID`

Valid JSON Bind Types:	**String**
Note:	Source data must be a valid hyphen-separated UUID.

Example:	{
	“string_uuid”: “40e6215d-b5c6-4896-987c-f30f3678f608”
	}
	-> UUID(40e6215d-b5c6-4896-987c-f30f3678f608)

`VARCHAR`, `CHAR`

Valid JSON Bind Types:	**String, Number, Boolean**
Note:	Number and Boolean types are converted to their string representations.

Example:	{
	“number_string”: 1000,
	“boolean_string”: true,
	“string_string”: “string”
	}
	-> “1000”, “true”, “string”

Load Geospatial Data

Geospatial data can be loaded from a variety of formats into Ocient. The Ocient System requires special considerations when you load geospatial data into Ocient including:

- Coordinate System
- Supported Geospatial Loading Formats
- Geospatial Type Conversion
- POINT Normalization
- POLYGON Normalization

Coordinate System

Ocient uses Spatial Reference System ID (SRID) 4326 for all internal geospatial data types. SRID 4326 is defined in the **WGS84 Standard**. Before you load the source data, all data should have the SRID 4326 data type. The loading operation ignores any source data that contains the coordinate system SRID information.

Supported Geospatial Loading Formats

Well-Known Text

The primary data format for loading geospatial data in Ocient is the Well-Known Text (WKT) format. This string format is a portable format that can represent points, lines, and polygons.

String data in WKT format can automatically load into `POINT`, `LINESTRING`, and `POLYGON` Ocient column types. In addition, arrays of strings in WKT format can automatically load into array columns of a suitable geospatial data type. The Ocient System assumes raw strings are in WKT format when the system loads these strings into geospatial column types.

Each `LINESTRING` or `POLYGON` value can be up to a maximum of 512 MB in size. This means a `LINESTRING` or `POLYGON` can contain approximately 32 million point values.

- WKT strings are whitespace insensitive, so `POINT(10 20)` is equivalent to `POINT (10 20)`.
- WKT strings are case insensitive.
- Extended Well-Known Text (EWKT) is not supported. To load EWKT data, load the substring after the EWKT semicolon as the WKT format.

Well-Known Binary and Extended Well-Known Binary

The other data formats that can load into Ocient are Well-Known Binary (WKB) and Extended Well-Known Binary (EWKB) formats. The Ocient System requires an explicit transformation function for both WKB and EWKB formats when you load data and convert it into the target column type.

You can use the function `st_geomfromewkb` to transform WKB and EWKB data.

In both cases, the Ocient System represents the WKB or EWKB string data as hexadecimal string data. The hexadecimal string can be one of these formats:

- Only hexadecimal digits: `00000000014000000000000004010000000000000`
- Prefixed with `0x` : `0x00000000014000000000000004010000000000000`
- Prefixed with `\x` : `\x00000000014000000000000004010000000000000`

Geometry Conversion

Ocient provides the automatic conversion of geometries from simpler types to more complex types. The conversion allows the load of a LINESTRING column with a POINT, or a POLYGON column with a POINT or LINESTRING. This capability also allows the load of arrays of POLYGON data from the source arrays of a combination of POINT, LINESTRING, and POLYGON data, which represents a geometry collection.

POINT Normalization

During loading, Ocient automatically performs normalization of POINT data into a regular format used within Ocient. The Ocient System performs the following operations on POINT data during the load:

- Constrain longitude to [-180, 180) and latitude to [-90, 90]; wrap around invalid coordinates using correct geographical handling.
- Snap points near the pole to the pole.
- Set longitude of points on the pole to 0
- Remove signed zeros from coordinates, so -0 becomes 0.

POLYGON Normalization

Ocient follows a counterclockwise rotation convention to indicate the outer ring of a POLYGON. The inner ring follows a clockwise rotation. If the Ocient System loads a POLYGON with a clockwise outer ring, the system indicates that the POLYGON is outside of the given ring.

> ⚠ In some cases, the POLYGON load can lead to unexpected results when the source data follows a different polygon rotation convention.

To account for this, the LAT provides the `st_forcepolygonccw` function that forces a counterclockwise rotation of the outer polygon ring and a clockwise rotation of the inner polygon ring when applicable.

Geospatial Transformation Functions

The LAT has a select set of transformation functions to construct geospatial types and manipulate them during the load. You can find the supported geospatial transformation functions in **LAT Transformation Functions**.

Related Links

Ingest Data with Legacy LAT Reference

Understanding Data Types

Geospatial Functions

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Pipeline Configuration

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

A pipeline configuration is a JSON file that fully describes the necessary elements to run a pipeline, including

- **Source** - the source location from which a pipeline should read records to process
- **Extract** - how a pipeline should extract data from the source
- **Transform** - how a pipeline should transform incoming records
- **Sink** - the destination where a pipeline should write transformed rows

The JSON file is a list of keys with (possibly nested) values. The available key-value pairs are documented in these sections.

Pipeline

Top level configuration for a pipeline.

Required keys:

- **version**
- **source**
- **extract**
- **transform**

The following is an example of the structure of a pipeline configuration.

© 2026 Ocient, Inc.

JSON

```
{
    "version": 2,
    "source": {
        "type": "kafka"
        // kafka source configuration
    },
    "sink": {
        "type": "ocient"
        // ocient sink configuration
    },
    "extract": {
        // extract configuration
    },
    "transform": {
        "topics": {
            "my-topic": {
                "tables": {
                    "my-table": {
                        "columns": {
                            "my-col1": "record_field_1",
                            "my-col2": "record_field_2"
                        }
                    }
                }
            }
        }
    }
}
```

Configuration

`version`

The pipeline's version. Required value is `2`.

Type:	`int`
Required:	Yes
Default:	

`pipeline_id`

A unique identifier for this pipeline. Allowed characters are `a-z`, `A-Z`, `0-9`, `_`, and `-`.

The `pipeline_id` is used to uniquely identify a pipeline. It is used for a few purposes:

Deduplication scope. See **Understanding Deduplication**.

1. For Apache® Kafka® loads, the consumer `group.id` is set to `ocient-lat-[pipeline_id]`.

> For most loads from File Sources, it is advisable to leave the `pipeline_id` unset when creating a pipeline using the LAT client. The client will assign the pipeline a random UUID.

Type:	`string`
Required:	No
Default:	LAT client randomly generated UUID.

`workers`

The number of workers this pipeline should use for processing records.

Type:	`int`
Required:	No
Default:	`DEFAULT_NUM_WORKERS` in service configuration

`log_original_records`

Add original records to the error log when errors occur. When this setting is set to `true`, LAT writes data extracted from the source to the error log and in some error messages. By default, this setting is `false` and source data is not written to the error log nor included in error messages.

In order to enable this setting, the **LAT_ALLOW_LOG_ORIGINAL_RECORDS** service configuration must also be enabled.

This configuration only affects pipelines that do not use an **error_topic**.

`seek_on_rebalance`

Whether to seek a newly assigned partition to the latest known durable record prior to resuming processing. Disabling this behavior should typically be reserved for test scenarios and is only supported for Kafka loading.

Type:	`boolean`
Required:	No
Default:	`true`

`continue_on_unrecoverable_error`

Whether to allow workers to continue processing when they encounter an ordinarily unrecoverable error.

Type:	`boolean`
Required:	No
Default:	`false`

`single_file_mode`

Enable the single file mode. This mode is designed for a specific use case where there are few files but each file size is large. When using this mode, only a single file is processed at a time, so **Common File Source Configuration** must be equal to `1` and only one **Common File Source Configuration** can be defined. The single file is processed in parallel by the number of workers defined by the pipeline `workers` setting. There is no need to enable this in common use cases.

Type:	`boolean`
Required:	No

Type:	`boolean`
Default:	`false`

> ⚠ A known limitation exists with the LAT metrics when you use the single file mode. Metrics returned from the `lat_client pipeline status` command might not display the expected count of files processed, processing, and so on. However, the load still displays the correct `PROCESSING` and `COMPLETED` statuses.

`error_topic`

A Kafka Topic to write records which cannot be processed. If absent, error records will be logged to the error log file without additional processing.

> ⓘ This configuration is only available if a Kafka Source is configured for the pipeline. The configuration for that source will apply to the Kafka Producer for this topic.

Type:	`string`
Required:	No
Default:	`null`

`polling_duration`

Maximum duration to block while polling for new records from a Source, in milliseconds.

Type:	`int`
Required:	No
Default:	`1000`

`source`

Source configuration section. See **LAT Source Configuration** for nested configuration.

Type:	`object`
Required:	Yes
Default:	

`sink`

Sink configuration section. See **LAT Sink Configuration** for inline configuration details. `sink` cannot be set if `sink_name` is set. `sink` can be omitted if a default sink is defined as an **External Sink Configuration**.

Type:	`object`
Required:	No
Default:	`null`

`sink_name`

The name of an externally configured sink. `sink_name` cannot be set if **sink** is set. See **External Sink Configuration**. `sink_name` can be omitted if a default sink is defined as an **External Sink Configuration**.

Type:	`string`
Required:	No
Default:	`null`

`extract`

Extract configuration section. See **LAT Extract Configuration** for nested configuration.

Type:	`object`
Required:	No
Default:	Defaults to JSON record type with default settings

`transform`

Transform configuration section. See **LAT Transform Configuration** for nested configuration.

Type:	`object`
Required:	Yes
Default:	

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Source Configuration

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

source

A source configuration object.

Required keys:

- **source.type**

`source.type` [#source-type]

Type of source to use in the pipeline.

Type:	**string**
Required:	Yes
Default:	

Allowed values:

- `kafka` : see **Kafka Configuration** for additional configuration.
- `s3` : see **Load from a File Source** and **S3 Source Configuration** for additional configuration.
- `local` : see **Load from a File Source** for additional configuration.

Load From a File Source

Currently, LAT supports loading files from an Amazon® Web Services℠ (AWS℠) S3 instance (like AWS S3 or IBM® Cloud® Object Storage) or from a local file system.

© 2026 Ocient, Inc.

Common configuration for all File Sources will be listed in **Common File Source Configuration**, followed by sections describing source-specific configurations.

File Sources are defined by creating “file groups” that represent a logical set of files to be loaded. Each file group is given a `file_group_name` that corresponds to the file group name in the transform section of the pipeline configuration. Each file group has settings to select the specific files that should be loaded.

The LAT supports loading from individual files with extensions such as `.jsonl` and `.csv`. GZIP compression is supported. However, files using the TAR archive format, LZOP compression, or ZIP compression are not supported.

> ⚠ If you load files with unsupported archive or compression formats, the load might stop or produce unexpected results.

The performance of loading into Ocient is greatly improved when records are presented in a well ordered time sequence. This sequence allows more efficient creation of segments and sorting of records into TimeKey® buckets. For this reason, the LAT has options to define how files in a file group should be sorted prior to loading.

File Group Filtering and Sorting Example

If you want to load all the files under a certain directory, the file group configuration is basic:

JSON

```
"my_file_group": {
    "prefix": "/dir1",
    "file_matcher_syntax": "glob",
    "file_matcher_pattern": "**",
    "sort_type": "lexicographic"
}
```

This file group configuration will select all files under `/dir1` (including those under its subdirectories) and sort them lexicographically.

However, if a user needs to selectively choose, filter, and sort the files selected, this process provides flexible options for doing so.

For an individual file group in a file type load, there are multiple steps to select, rename, filter, and sort the files retrieved. The files selected occur through this sequence:

1. Prefix filtering
2. File matching
3. File renaming
4. Timestamp extract pattern matching (only `extract_timestamp`)
5. Range matching
6. Final list sorting

For example, given these files from a local file system:

Text

```
/dir1/file.json
/dir1/files1/year=2021/month=01/day=01/files2/10-00-00.csv
/dir1/files1/year=2021/month=01/day=01/file.json
/dir1/files1/year=2021/month=01/day=01/files3/11-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/12-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/14-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/13-00-00.json
```

And a `file_group` configuration:

JSON

```
"my_group": {
    "prefix": "/dir1/files1/",
    "file_matcher_syntax": "regex",
    "file_matcher_pattern": "/dir1/files1/year=(\\d{4})/month=(\\d{2})/day=(\\d{2})
    "rename_format": "{1}-{2}-{3}-{4}-{5}-{6}",
    "sort_type": "extract_timestamp",
    "path_timestamp_pattern": "yyyy-MM-dd-HH-mm-ss",
    "start_time": "2021-01-01T13:00:00",
    "stop_time": "2021-01-01T14:00:01"
}
```

Step 1: Prefix Filtering

Prefix filtering occurs first and includes only files that are in the matching prefix paths. The prefix is the path part following the bucket for S3 types, and it is the path to files in a local file type load.

JSON

```
"prefix": "/dir1/files1/"
```

Result: `/dir1/file.json` is filtered out because it is not in the `prefix`.

Step 2: File Matching

Next, file matching uses a pattern to match files that should be included. File matcher patterns apply to the full path of the file including any prefix defined in the prior step.

JSON

```
"file_matcher_syntax": "regex"
"file_matcher_pattern": "/dir1/files1/year=(\\d{4})/month=(\\d{2})/day=(\\d{2})/fil
```

Result: `/dir1/files1/year=2021/month=01/day=01/files2/10-00-00.csv` is filtered out because it does not end in `.json`. `/dir1/files1/year=2021/month=01/day=01/file.json` is filtered out because it is not in the subdirectory `files1`.

Step 3: File Renaming

File renaming is a step that can be useful in a case where the selected files have disparate file names that would make it difficult to extract timestamps from. Files are only renamed internally to the LAT to facilitate the file selection process; they are *not* actually renamed locally or on S3.

By unifying each file name into a consistent format, this step should make it easier to use the `extract_timestamp` sort type, or even to `lexicographically` sort on some parameter within the file name itself.

JSON

```
"file_matcher_pattern": "/dir1/files1/year=(\\d{4})/month=(\\d{2})/day=(\\d{2})/fil
"rename_format": "{1}-{2}-{3}-{4}-{5}-{6}",
```

© 2026 Ocient, Inc.

Result: The files that match the `file_matcher_pattern` regular expression are renamed using the `rename_format`.

Current Files at this Step:

```Text
/dir1/files1/year=2021/month=01/day=01/files3/11-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/12-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/14-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/13-00-00.json
```

Renamed file list that enters the next step:

```Text
2021-01-01-11-00-00
2021-01-01-12-00-00
2021-01-01-14-00-00
2021-01-01-13-00-00
```

Step 4: Timestamp Extract Pattern Matching

A filter is applied when the sort_type is `extract_timestamp`. The `extract_timestamp` sort type extracts timestamps from the file's filepath. Either a `path_timestamp_pattern` or a `file_timestamp_pattern` must be set. If a filename does not match the set pattern, it will be filtered out.

```JSON
"sort_type": "extract_timestamp"
"path_timestamp_pattern": "yyyy-MM-dd-HH-mm-ss",
```

Result: In this example, a `path_timestamp_pattern` is set, which is a `DateTimeFormatter` pattern that will extract a timestamp from the starting from the beginning of the filename, used for the next step. No files are filtered here because the rename step above was able to unify filename formats in a way that the `path_timestamp_pattern` could match on all the files.

© 2026 Ocient, Inc.

Internal to the LAT, the files are associated with their datetimes, which will be used in the next step:

```
Text

filename -> datetime extracted in ISO 8601 format
2021-01-01-11-00-00 -> 2021-01-01T11:00:00Z
2021-01-01-12-00-00 -> 2021-01-01T12:00:00Z
2021-01-01-14-00-00 -> 2021-01-01T14:00:00Z
2021-01-01-13-00-00 -> 2021-01-01T13:00:00Z
```

There are potential pitfalls with using the `extract_timestamp` `sort_type`, described in the **LAT Source Configuration** section.

Step 5: Range Matching

Range matching occurs next based on the chosen `sort_type` algorithm.

- For `extract_timestamp` or `metadata`, a timestamp is associated with each file for sorting. A start and end time can be optionally provided to limit files to select.
- For `lexicographic`, the filename itself is used for sorting. A start and end file name can be provided to limit the files that are selected.

```
Text

"sort_type": "extract_timestamp"
"path_timestamp_pattern": "yyyy-MM-dd-HH-mm-ss",
"start_time": "2021-01-01T13:00:00"
"stop_time": "2021-01-01T14:00:01"t
```

Result: The file `2021-01-01-12-00-00` is filtered out because it is not in the start and end time ranges.

Step 6: Final List Sorting

Finally, the fully filtered file list contains two files from your original list. These are then sorted according to the `sort_type` algorithm and partitioned across the workers for loading.

© 2026 Ocient, Inc.

Remaining files before sorting:

```
2021-01-01-14-00-00
2021-01-01-13-00-00
```

Final list - sorted, renamed files:

```
2021-01-01-13-00-00
2021-01-01-14-00-00
```

(original file names):

```
/dir1/files1/year=2021/month=01/day=01/files2/13-00-00.json
/dir1/files1/year=2021/month=01/day=01/files2/14-00-00.json
```

Example S3 Source Configuration

> The names defined for `file_groups` (“my_file_group” in this example) should match the `file_groups` used in the transform configuration section.

© 2026 Ocient, Inc.

JSON

```
{
    "source": {
        "type": "s3",
        "endpoint": "http://some.endpoint/",
        "bucket": "my_bucket",
        "file_groups": {
            "my_file_group": {
                "prefix": "some/prefix/",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**.json",
                "sort_type": "extract_timestamp",
                "path_timestamp_pattern": "'dir1/files1/'yyyy'-'MM'-'dd'T'HH':'mm':
            }
        },
        "compression": "gzip"
    },
    "transform" : {
        "file_groups" : {
            "my_file_group" : {
                "tables" : {
                    ...
                }
            }
        }
    }
}
```

Example Local File Source Configuration

ⓘ The names defined for `file_groups` (“my_file_group” in this example) should match the `file_groups` used in the transform configuration section.

JSON

```
{
    "source": {
        "type": "local",
        "file_groups": {
            "my_file_group": {
                "prefix": "/path/to/data/directory/",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**.json",
                "sort_type": "extract_timestamp",
                "path_timestamp_pattern": "'dir1/files1/'yyyy'-'MM'-'dd'T'HH':'mm':
            }
        },
        "compression": "gzip"
    },
    "transform" : {
        "file_groups" : {
            "my_file_group" : {
                "tables" : {
                    ...
                }
            }
        }
    }
}
```

Common File Source Configuration

`source.file_groups`

An object that maps file group names to their corresponding configuration objects.

Type:	**object**
Required:	Yes
Default:	

`source.file_groups.<file_group_name>`

The configuration object for a file group.

Type:	object
Required:	Yes
Default:	

`source.file_groups.<file_group_name>.prefix`

For an S3 Source, the `prefix` is used to get a subset of S3 objects from the bucket. A more specific prefix can improve time required to list files in S3 sources.

For a Local File Source, `prefix` is an absolute or relative path from the working directory of the LAT. In either case, the path can be a file or a directory. If the path is a directory, LAT tries to load all files recursively under this path. When looking for files, LAT follows the symbol links. LAT only loads regular files that are readable and not hidden. Note that shell-specific expansions like `~` are not supported.

You can include multiple prefixes for a single file group if you specify this property as an array of strings. In that case, the LAT loads all files that are under any of the prefixes in that array. The system loads files that match more than one prefix within a single file group only once.

> ⚠ For a Local File Source, the path used for prefix is the path on the server where the LAT is running, which might not be the same machine where you are running the LAT Client Command Line Interface to create your pipeline.

Type:	string or string array
Required:	No
Default:	`""`

`source.file_groups.<file_group_name>.file_matcher_syntax`

The `file_matcher_syntax` defines the type of syntax used by the `file_matcher_pattern` setting, which includes matching files in the file group. The pattern is applied to the fully qualified file name in the list of files found under the `prefix`.

See the `file_matcher_pattern` description for examples and complete syntax details for `glob` and `regex` options.

Type:	**string**
Required:	No, although it is not valid to have only one of `file_matcher_syntax` or `file_matcher_pattern` be provided
Default:	`glob`

Allowed values:

- `glob` - a simplified pattern matching system based on wildcards
- `regex` - a regular expression syntax

`source.file_groups.<file_group_name>.file_matcher_pattern`

The `pattern` used to select files from the `prefix`-filtered list. Files that match the pattern are included in the file group. The **fully qualified file name** (path and filename under the prefix) are matched. Files that do not match are excluded from the file group.

This pattern is also used in the case of LAT **file renaming** - if a `rename_format` is provided, the LAT will attempt to internally rename files using capture groups in the `file_matcher_pattern`. If capture groups are provided in the `file_matcher_pattern`, they will be extracted and placed into the `rename_format` in order to generate a renamed file. Capture groups do *not* need to be named; the captured values are assigned to the `rename_format` sequentially, i.e. the first capture group in the `file_matcher_pattern` will be placed in `{1}` in the `rename_format`.

ⓘ Files are only renamed internally to the LAT to facilitate the file selection process; they are *not* actually renamed locally or on S3.

ⓘ This matched file list can be further filtered using an `extract_timestamp's` pattern and the start/end ranges of any file group type algorithm. See an expanded example in the **File Group Filtering and Sorting example** section.

ⓘ A double backslash within the regular expression is necessary to generate valid escape characters for the `pipeline.json`. For example, the capture group `(\d{4})` needs to be escaped as `(\\d{4})`.

Type:	string
Required:	No, although it is not valid to have only one of `file_matcher_syntax` or `file_matcher_pattern` be provided
Default:	`**`

Example

With a `file_matcher_syntax` of `glob` and `file_matcher_pattern` of `**.json`, all files (after `prefix`-filtering) that end in `.json` will be selected.

> A `**` in the glob based pattern ensures that any pattern including directory boundaries are matched. A pattern `*` does not match the directory character `/`.

Example

With a `file_matcher_syntax` of `regex`, a `file_matcher_pattern` of `auctions.*-2021.json` would use a regular expression to ensure that files such as `auctions-12-01-2021.json` and `auctions-12-02-2021.json` would be selected.

See `getPathMatcher()` for complete syntax details of the `glob` and `regex` options.

Common Matcher Patterns:

Matcher Syntax	Pattern	Meaning
glob	`*`	Matches zero or more characters without crossing directory boundaries
glob	`**`	Matches zero or more characters crossing directory boundaries
glob	`?`	Matches exactly one character of a name component
glob	`[]`	Matches any of the characters in the bracket expression (e.g., [abc]). Supports ranges.
regex	`.`	Matches any character
regex	`\d`	Matches any digit
regex	`\D`	Matches any non-digit
regex	`[]`	Matches any of the characters in the bracket expression (e.g., [abc]). Supports ranges.

© 2026 Ocient, Inc.

Matcher Syntax	Pattern	Meaning
regex	`*`	Matches the preceding character zero or more times
regex	`+`	Matches the preceding character one or more times
regex	`?`	Matches the preceding character once or not at all

`source.file_groups.<file_group_name>.rename_format`

A Java® `MessageFormat` string. Captured groups from the `file_matcher_pattern` will be placed into the `rename_format` string's format elements corresponding to the order they were captured. The renamed file is used for subsequent sorting of files. When using `lexicographic` sort, the renamed file is used instead of the original filename. For `extract_timestamp` sorting, the `path_timestamp_pattern` is applied against the renamed file. This is best elucidated with an example:

Text

```
filename: 20210102.json
file_matcher_pattern: (\\d{4})(\\d{2})(\\d{2}).*
rename_format: "file-{1}-{3}-{2}"
"sort_type": "extract_timestamp"
"path_timestamp_pattern": "'file-'yyyy'-'MM'-'dd"

Renames the file to: "file-2021-02-01"
The "path_timestamp_pattern" extracts an ISO-8601 datetime of 2021-02-01T00:00:00Z
```

Files are only renamed internally to the LAT to facilitate the file selection process; they are *not* actually renamed locally or on S3.

Type:	**string**
Required:	No
Default:	None - if `rename_format` is not provided, files will not undergo a rename step.

`source.file_groups.<file_group_name>.sort_type`

Defines the sorting algorithm used for sorting the files selected for loading in this file group by time. Files should be sorted in time order for best loading performance and the

creation of efficient Ocient segments. `sort_type` can be `extract_timestamp`, `metadata`, or `lexicographic`, and different settings apply to each of these selections.

Type:	**string**
Required:	Yes
Default:	`none`

Allowed values:

- `extract_timestamp`: extract the timestamp from the file name or file path information. Either `path_timestamp_pattern` or `file_timestamp_pattern` is required when choosing this `sort_type`.
- `metadata`: extract the timestamp from the source file's metadata. This option uses the file last modified time when sorting the list.
- `lexicographic`: sort files based on the alphanumeric sort of files in the file group using the full path and filename as the sorting key.

> The system breaks the tie among files in a lexicographic way by using their fully qualified original file names.

`source.file_groups.<file_group_name>.path_timestamp_pattern`

A Format string with `DateTimeFormatter` patterns used to extract a datetime from the path portion of a file's fully-qualified filename. This timestamp is used with the `extract_timestamp` sort type to order the selected files in the file group prior to loading. It is attempted on every file's fully-qualified filename under the `prefix`; files that do not match the pattern will be skipped and the remaining subset will be used for sorting.

Type:	**string**
Required:	One of `path_timestamp_pattern` or `file_timestamp_pattern` is required for an `extract_timestamp` sort type file group
Default:	`none`

`source.file_groups.<file_group_name>.file_timestamp_pattern`

A Format string with `DateTimeFormatter` patterns used to extract a datetime from the base filename of a file's fully-qualified filename. This timestamp is used with the `extract_timestamp` sort type to order the selected files in the file group prior to loading. It is attempted on every file's base filename under the `prefix`; files that do not match the pattern will be skipped and the remaining subset will be used for sorting.

Type:	**string**
Required:	One of `path_timestamp_pattern` or `file_timestamp_pattern` is required for an `extract_timestamp` sort type file group
Default:	`none`

`source.file_groups.<file_group_name>.start_time`

An ISO-8601 compliant Date or Datetime, used as the lower bound to filter for files in a `extract_timestamp` or `metadata` sort type file group. Datetimes are UTC unless a timezone is provided. Inclusive.

Type:	**string**
Required:	No
Default:	`Instant.MIN`

`source.file_groups.<file_group_name>.stop_time`

An ISO-8601 compliant Date or Datetime, used as the upper bound to filter for files in a `extract_timestamp` or `metadata` sort type file group. Datetimes are UTC unless a timezone is provided. Exclusive.

Type:	**string**
Required:	No
Default:	`Instant.MAX`

*Extract Timestamp Default Values *

With an `extract_timestamp` `sort_type`, if a certain time unit is not provided, they will default to their respective value in `1970-01-01T00:00:00.000000`.

Example:

```
Text

Filename: `July03.json`
`path_timestamp_pattern` or `file_timestamp_pattern`: `MMMMdd'.json'`
```

The full timestamp extracted from this filename will be `1970-07-03T00:00:00.000000 UTC`.

Potential Pitfalls

Pitfall 1: This default value should be considered when a `start_time` or `stop_time` is provided but a certain time unit is not present within the filename.

Example:

```
Text

Capture files ranging from July 1st through July 3rd:

Filenames: July01.json, July02.json, July03.json
"file_timestamp_pattern": "MMMMdd'.json'"
"start_time": "1970-07-01"
"stop_time": "1970-07-04"
```

Note that 1970 was used as the year in the `start_time` and `stop_time`, but the system could not extract a year from the filenames. An alternative strategy is to use the File Renaming step to insert missing dates or times.

Pitfall 2: Only applicable to using the `extract_timestamp` `sort_type` with `start_time` and `stop_time`: Consider the time unit granularities between your pattern and start/stop times.

Example:

© 2026 Ocient, Inc.

Text

```
Filenames: dir1/2021/01/01/00:00:00.json, dir1/2021/01/01/01:00:00.json, dir1/2021/
"path_timestamp_pattern": "'dir1/'yyyy'/'MM'/'dd'/'"
"start_time": "2021-01-01T00:00:00"
"stop_time": "2021-01-01T04:00:00"
```

Given this configuration, one might expect that the files that are hour 4 and later would be filtered out. However, because the pattern does not extract the times from the filenames, each file's extracted datetime in ISO 8601 format is `2021-01-01T:00:00:00UTC`, so no files are outside the range and filtered. Make sure to extract as much information about the datetime as necessary.

`source.file_groups.<file_group_name>.start_file`

A string, used as the lower bound to filter for files in a `lexicographic` sort type file group. Inclusive.

Type:	**string**
Required:	No
Default:	`none`, but a `lexicographic` sort type file group will not check for a lower bound if this is not present.

`source.file_groups.<file_group_name>.stop_file`

A string, used as the upper bound to filter for files in a `lexicographic` sort type file group. Inclusive.

Type:	**string**
Required:	No
Default:	`none`, but a `lexicographic` sort type file group will not check for an upper bound if this is not present.

`source.file_groups.<file_group_name>.compression`

The compression method for the files in this file group. If this value is set, it will override `source.compression`; if not set, it will inherit `source.compression`. See

`source_compression` for available options.

Type:	**string**
Required:	No
Default:	`null`

`source.file_groups.<file_group_name>.bucket`

This setting is available only for S3 sources.

If specified, this setting overrides the source-wide bucket for this file group. This allows loading from multiple buckets at once within the same pipeline.

If a `bucket` is specified for every file group, you can omit the `source.bucket`.

Type:	**string**
Required:	No
Default:	`null`

`source.partitions`

The number of partitions for each file group. For example, if there are 2 file groups and `partitions` is set to 32, then a total of 64 partitions will exist, with 32 for each file group. The partition index is 0-based for each file group. In the above example, each file group will have partitions `[0, 31]`.

⚠ When `partitions` is specified for a load, its value must not change for the lifetime of this load, otherwise records cannot load correctly.

ⓘ When using the LAT Client, `partitions` will be automatically assigned by the client. If it is set in the source configuration provided by the user, it will be overwritten.

© 2026 Ocient, Inc.

Type:	int
Required:	No
Default:	Set by the LAT client

`source.partitions_assigned`

A 2-element array indicating the minimum and maximum partition indices that this LAT Node owns, inclusive on both ends. For example, `[0, 15]` means that the node owns Partitions 0, 1, 2, ..., 15. The first element should be greater than or equal to 0. The second element should be greater than or equal to the first element, and less than the number of partitions.

> When using the LAT Client, `partitions_assigned` will be automatically assigned by the client. If it is set in the source configuration provided by the user, it will be overwritten.

Type:	int array
Required:	No
Default:	Set by the LAT client

`source.compression`

The compression method for the files. Currently, only `none` and `gzip` are supported. `none` means that the files are not compressed. Note that this value sets the default compression method for all the files from this source, and it can be overridden by `source_file_groups_compression` per file group.

Type:	string
Required:	No
Default:	`none`

`source.chunk_size`

The size of a chunk when fetching data. For example, if you have a 40 MB file and the chunk size is 16 MB, you can issue 3 requests sequentially to get the file. The unit is MB.

The value must be greater than 0.

> The chunk size should be larger than or equal to the maximum record size.

Type:	**int**
Required:	No
Default:	`16`

`source.buffer_size`

The total buffer size for *all* assigned partitions. This value will be divided by the total number of assigned partitions to calculate each partition's buffer size. For example, if this value is 4096, `source.partitions` is 16, `source.partitions_assigned` is `[0, 3]`, and you have 2 file groups, then you have a total of (3-0+1)*2=8 assigned partitions, and each partition will get 4096/8=512 MB as the buffer size. The unit is MB. The value must be greater than 0.

> The buffer size *per partition* must be at least twice the value of `source.chunk_size`.

Type:	**int**
Required:	No
Default:	`4096`

`source.max_fetch_concurrency`

The maximum concurrency when fetching files for each partition. Setting this configuration too high can result in thread contention. The value must be greater than 0.

Type:	**int**
Required:	No
Default:	`2`

© 2026 Ocient, Inc.

S3 Source-Specific Configuration

`source.endpoint`

The endpoint for the S3 instance, usually starting with `http` or `https`. It can be an IP address or a domain. For example, for AWS S3 in the `us-east-2` region, endpoint would be `https://s3.us-east-2.amazonaws.com`. The S3 Source supports Virtual-hosted-style access and Path-style access. The `s3://` protocol based access is not supported. For more details, see **Methods for Accessing an S3 Bucket**

Type:	**string**
Required:	Yes
Default:	

`source.region`

The region for the S3 instance. This is mostly used for AWS S3 instances.

Type:	**string**
Required:	No
Default:	`us-east-2`

`source.bucket`

The bucket from which to get S3 objects.

You can override the bucket on a per-file group basis by setting the `source.file_groups.<file_group_name>.bucket`.

This setting is required unless it is specified for every file group.

Type:	**string**
Required:	No
Default:	

`source.path_style_access`

Whether to force path style access for the S3 instance. If set to `false`, LAT will try to use the virtual-hosted style (using DNS subdomains), and falls back to the path style access.

AWS S3 is deprecating path style access according to **this post**.

Type:	**boolean**
Required:	No
Default:	`false`

`source.access_key_id`

The access key for the S3 instance. This should be used together with the `source.secret_access_key` setting. If either of them is absent, LAT will default to the next item in the credentials hierarchy. See **S3 Credentials Hierarchy**.

Type:	**string**
Required:	No
Default:	`null`

`source.secret_access_key`

The access secret for the S3 instance. This should be used together with the `source.access_key_id` setting. If either of them is absent, LAT will default to the next item in the credentials hierarchy. See **S3 Credentials Hierarchy**.

Type:	**string**
Required:	No
Default:	`null`

`source.session_token`

Temporary credentials can be made with a combination with the existing secret ID and secret key, and an additional session token. Note that this is only used when both `source.access_key_id` and `source.secret_access_key` are specified.

Type:	string
Required:	No
Default:	`null`

`source.retries`

Configures the number of retries that will be attempted when download from S3 source fails.

Type:	int
Required:	No
Default:	Default attempts specified in AWS SDK.

`source.backoff_strategy_base_delay_seconds`

Configures the base delay for the backoff strategy of the S3 client in seconds.

Type:	int
Required:	No
Default:	1

`source.backoff_strategy_max_backoff_seconds`

Configures the maximum backoff delay for the backoff strategy of the S3 client in seconds.

Type:	int
Required:	No
Default:	For the default delay, see the **AWS SDK documentation**.

`source.max_pending_connection_acquires`

Configures the maximum number of pending acquires allowed by the Netty client.

Type:	int
Required:	No
Default:	For the default number of acquires, see the **AWS SDK documentation.**

`source.netty_read_timeout_seconds`

Configures the read timeout, in seconds, of the Netty client. When you set this value to zero, the system disables the read timeout. The LAT configures read timeouts to be tried again by the S3 Client.

Type:	int
Required:	No
Default:	For the default timeout value, see the **AWS SDK documentation.**

`source.connection_timeout_seconds`

Configures the amount of time in seconds for the Netty client to wait when initially establishing a connection before giving up and timing out. The LAT configures connection timeouts to be retryable by the S3 Client.

Type:	int
Required:	No
Default:	For the default timeout value, see the **AWS SDK documentation.**

`source.connection_acquisition_timeout_seconds`

Configures the amount of time in seconds for the Netty client to wait when acquiring a connection from the pool before giving up and timing out. The LAT configures connection acquisition timeouts to be retryable by the S3 Client.

Type:	int
Required:	No
Default:	For the default timeout value, see the **AWS SDK documentation.**

`source.requester_pays`

Configures whether or not the requester (i.e. the LAT user) should be charged for downloading data from the S3 Requester Pays buckets. This configuration should be set to `true` whenever you request from a Requester Pays bucket, or else the request fails and the bucket owner is charged for the request.

Type:	**boolean**
Required:	No
Default:	`falsed`

S3 Credentials Hierarchy

The S3 source configuration supports the following hierarchy to obtain S3 credentials. If the LAT does not obtain the credentials at a level, the LAT tries the lower level.

- Level 1: Pipeline configuration
- Level 2: **AWS SDK Default Credential Provider Chain**
- Level 3: (Default) Anonymous access

> You can choose the level to store the credentials where Level 1 is the highest. Higher levels take precedence for credential storage.

Load from a Kafka Source

The Apache® Kafka® Source allows LAT to connect to a Kafka cluster and process records from one or more topics. You can reference each Kafka topic in the **LAT Transform Configuration** and can load into one or more tables.

In addition to the LAT specific configuration, this Source supports configuration passthrough to the underlying **Kafka library** which this Source uses. There are some library configurations which are not allowed, and will be noted in this table.

Example Kafka Source Configuration

JSON

```
{
    "source": {
        "type": "kafka",
        "kafka": {
            "bootstrap.servers": "127.0.0.1:9092",
            "auto.offset.reset": "earliest"
        }
    }
}
```

Kafka Configuration

`source.end_offsets_polling_duration`

Frequency with which to poll for end offsets for lag calculation, in milliseconds.

Type:	**int**
Required:	No
Default:	30000

`source.kafka`

Kafka consumer configuration object. See **ConsumerConfig** for details.

Type:	**object**
Required:	Yes
Default:	

`source.kafka`

Required Keys:

- `kafka.bootstrap.servers`
    - This key is required for connection information to the Kafka servers.

Common Keys:

- `kafka.auto.offset.reset`
  - This key is commonly set to `earliest` to consume all of the data in an existing Kafka topic.

Disallowed Kafka Configuration

The following Kafka consumer configuration options are not allowed:

- `kafka.enable.auto.commit`
  - LAT will always internally set this configuration to `false`; it is critical for correct operation to do so.
- `kafka.key.deserializer`
  - LAT will always internally configure deserialization
- `kafka.value.deserializer`
  - LAT will always internally configure deserialization

If any of the disallowed Kafka library configurations are set, a warning will be logged and their configurations will be ignored.

Defaulted Kafka Configuration

- `kafka.group.id`
  - If unset. LAT will configure `group.id` internally using the value of `ocient-lat-[$pipeline_id]`. Most users should leave this unset unless they want to explicitly control the `group.id` that is being used.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Sink Configuration

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

The Sink Configuration controls the destination of data in the LAT Pipeline.

`sink`

A sink configuration object.

Required keys:

- **sink.type**

`sink.type` [#sink-type]

Type of sink to use in the pipeline.

Type:	**string**
Required:	Yes
Default:	

Allowed values:

- `ocient` : see **Ocient Sink** for additional configuration.
- `file` : see **File Sink** for additional configuration.

Ocient Sink

The Ocient Sink allows LAT to connect to an Ocient cluster to write rows to one or more tables.

Required keys:

© 2026 Ocient, Inc.

- **sink.remotes**

`sink.remotes` [#sink-remotes]

Array of one or more Ocient Loader Nodes, in `host:port,...` format

Type:	**string[]**
Required:	Yes
Default:	

`sink.batch_records`

Number of records to buffer per partition before flushing records to Ocient

Type:	**int**
Required:	No
Default:	1000

`sink.batch_duration`

Time based flushing parameter, in milliseconds. Records will flush to Ocient after this duration has elapsed with no new activity, even if fewer than `batch_records` records have been processed.

Type:	**int**
Required:	Yes
Default:	30000

`sink.idle_partition_polling_period`

Time based polling parameter, in milliseconds. This Sink will periodically poll the remote for progress on write durability for idle partitions.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Type:	**int**
Required:	No
Default:	60000

`sink.request_timeout`

Request timeout when communicating with Ocient remotes, in milliseconds.

Type:	**int**
Required:	No
Default:	300000

`sink.request_backoff`

Duration to delay after a failed request to an Ocient remote prior retrying, in milliseconds.

Type:	**int**
Required:	No
Default:	1000

`sink.request_jitter`

Additional duration to delay after a failed request to an Ocient remote prior retrying, in milliseconds. The total delay incurred prior to a given retry is `request_backoff + rand(0, request_jitter)`.

Type:	**int**
Required:	No
Default:	5000

`sink.high_watermark`

High watermark memory point, in bytes. The LAT will stop pushing new rows to memory buffers. It will not resume pushing rows into the memory buffers until `low_watermark` is reached.

Type:	**int**
Required:	No
Default:	1000000000

`sink.low_watermark`

Low watermark memory point, in bytes. After reaching `high_watermark`, the LAT will begin pushing rows to memory buffers again when this memory level is reached.

Type:	**int**
Required:	No
Default:	500000000

`sink.storage_scope_id`

UUID of the storage scope that rows will be associated with. The scope with the given UUID must already exist in the target cluster.

Type:	**string**
Required:	No
Default:	`null`

`sink.skip_page_replication`

A Boolean value to determine whether to omit page replicas for the specified storage scope. This is ignored if `sink.storage_scope_id` is not specified or has already been seen by the remotes.

© 2026 Ocient, Inc.

Type:	**boolean**
Required:	No
Default:	`false`

`sink.netty_event_loop_group_threads`

The number of threads in the Netty event loop group used to communicate with remotes.

Type:	**int**
Required:	No
Default:	`1`

Example Ocient Sink Configuration

JSON

```
{
    "sink": {
        "type": "ocient",
        "remotes": ["loader0:5050", "loader1:5050", "loader2:5050"]
    }
}
```

File Sink

A Sink type for **testing** LAT pipelines that writes the transformed data to local **JSONL** files.

Required keys:

- **sink.location**

sink.location [#sink-location]

An absolute or relative path to the location that the sink should write files to.

Type:	**String**
Required:	Yes
Default:	

Example File Sink Configuration

JSON

```
{
    "sink": {
        "type": "file",
        "location": "out/"
    }
}
```

External Sink Configuration

Rather than including a sink directly within the pipeline, it is also possible to configure a pipeline to use a sink that is specified externally. Sinks can be managed (created, deleted, and more) using the **LAT Client Command Line Interface**. A sink must exist before a pipeline can use it.

There are three ways to configure a pipeline to use a sink.

1. If a sink is included directly within a pipeline (using the **LAT Sink Configuration**), it will be used.
2. If a sink is not specified within the pipeline, you can specify a **sink_name** that corresponds to a sink previously created using the LAT Client.
3. If neither `sink` nor `sink_name` is specified in a pipeline, the default sink will be used. If a default sink has not been created using the LAT Client, a pipeline must specify either a `sink` or a `sink_name`.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved

LAT Extract Configuration

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

An extract configuration object. This object defines the way that data is extracted from the source records. The extract definition is applied to a specific topic or file group as defined in the source. Keys are file group names or topic names and values are **Extract Settings**.

> ⓘ For file based loads, this should match a file_group_name defined in the `source.file_groups` section. For Apache® Kafka® based loads this should match a `topic` defined in Kafka.

Required keys:

- None - Extract defaults to JSON for all topics or file_groups unless provided

Example Extract

In the following example, two file groups are defined. The file groups defined in the source section are referenced in the extract section to define how data is extracted from the file group. For Kafka based loads, the topic names are used in the extract section to define extraction if required.

© 2026 Ocient, Inc.

JSON

```
{
    ...,
    "source": {
        "type": "s3",
        "endpoint": "http://some.endpoint/",
        "bucket": "my_bucket",
        "file_groups": {
            "delimited_user_data": {
                "prefix": "crm/users/",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**.csv",
                "sort_type": "lexicographic"
            },
            "delimited_order_data": {
                "prefix": "crm/orders/",
                "file_matcher_syntax": "glob",
                "file_matcher_pattern": "**.json",
                "sort_type": "lexicographic"
            }
        }
    },
    "extract": {
        "delimited_user_data": {
            "record_type": "delimited",
            "headers": ["id", "username", "last_login_at", null, "first_name", "las
            "field_delimiter": "|",
            "trim_whitespace": true,
            "skip_blank_lines": true,
            "field_optionally_enclosed_by": "\"",
            "null_strings": ["NULL", "N/A"]
        },
        "delimited_order_data": {
            "record_type": "delimited",
            "num_header_lines": 1,
            "exclude_columns": ["user_id", "order_total"],
            "field_delimiter": ","
        }
    },
    "transform": {
        ...
```

```
    }
}
```

Extract Settings

`extract.<file_group_name>.record_type`

Similar to Source and Sink config, `record_type` instructs the Pipeline to instantiate a specific extraction implementation for a given type of source record such as Delimited Records (e.g., CSV, TSV), JSON Records, or Fixed Width Binary Records.

Type:	**string**
Required:	No
Default:	json

Allowed values:

- `json` : see **JSON Extract** for additional configuration.
- `delimited` : see **Delimited Extract** for additional configuration.
- `fixed_width_binary` : see **Fixed Width Binary Extract** for additional configuration.

Shared Extract Configuration

`extract.<file_group_name>.null_strings`

`null_strings` instructs the extraction process to convert a string literal such as *NA* to a `null` value in the transformation layer. This allows different null string values to be provided. `null_strings` must be an array of strings. If `trim_whitespace` is set to true and a string value is trimmed to become a `null_string`, that value converts to `null`.

Type:	**string[]**
Required:	No
Default:	

`extract.<file_group_name>.empty_as_null`

The `empty_as_null` setting specifies whether empty fields should be treated as null. An empty field is defined as a string field containing no characters (its length is 0). When set to false, empty fields will be treated literally as empty strings. When set to true, empty fields will be treated as `null` in the transformation and loading process. If `trim_whitespace` is set to true and `empty_as_null` is set to true, a field that consists of only whitespace will also be converted to `null`.

Type:	**boolean**
Required:	No
Default:	True for the Delimited and FWB extract types, False for a JSON extract type

`extract.<file_group_name>.trim_whitespace`

The `trim_whitespace` setting specifies whether whitespace should be trimmed from the beginning and end of each string field as it is extracted.

Type:	**boolean**
Required:	No
Default:	False

`extract.<file_group_name>.encoding`

The `encoding` setting specifies the file encoding to use. Encodings should use standard Java® encodings defined in the **Java Internationalization Guide**.

Type:	**string**
Required:	No
Default:	UTF8

`extract.<file_group_name>.replace_invalid_characters`

`replace_invalid_characters` defines whether invalid characters based on the encoding type should be replaced with a replacement character (i.e. `U+FFFD`). If this setting is false and invalid characters are encountered, the record will not be loaded and a record extraction error will be logged.

Type:	**boolean**
Required:	No
Default:	False

JSON Extract

The JSON Extract type allows LAT to extract JSON data from the source records. There are no additional configuration settings available for JSON beyond the common Extract settings.

Example JSON Extract Configuration

In this example JSON extract configuration, the `replace_invalid_characters` setting is used to convert any invalid UTF-8 characters with the UTF-8 replacement character. The file group name assigned to this file group was defined as “json_user_data” in the `source` section of the pipelines. When transforming this data, you can reference columns by the JSON object keys.

JSON

```
{
    "extract": {
        "json_user_data" : {
            "record_type": "json",
            "replace_invalid_characters": true
        }

    },
    ...
}
```

Delimited Extract

The Delimited Extract type allows the LAT to extract delimited data from the source records. The delimiter character is configurable to support CSV, TSV, or other delimited formats like pipe and semicolon. Fixed width files are a separate extraction type.

Notes on Delimited Files

- Delimited files can have additional comment rows that are prefixed by a `#` character. These lines are not processed and will be skipped during loading.
- A maximum number of 1024 columns can be loaded in a single delimited file.
- You can enclose fields with an optional enclosure character. In these cases, the enclosure character is not retained in the loaded data.
- You can skip a configurable number of header lines during loading to prevent column headers from appearing as loaded data.

The system parses headers from the beginning of each file if you configure **num_header_lines** and do not configure **headers**.

The system automatically names columns if you do not configure **num_header_lines** and **headers**. The columns are named “0","0","1”, and so on consecutively instead.

- You can define multiple file groups. This allows different delimited extraction settings on each file group.
- The default record delimiter is a `\n` newline character. If source data differs, this should be overridden.

Records with improperly escaped “quote characters” or malformed “quoting” of column data will not parse. See the **num_header_lines** setting for more details.

- Control characters such as the NULL byte `0x0` are automatically processed as whitespace when parsing delimited records.

Delimited Configuration

`extract.<file_group_name>.field_delimiter`

The `field_delimiter` specifies the string that should function as a column delimiter. The field_delimiter character applies to any unquoted and unescaped text when extracting records from the source file. This can be a single character or a multi-character string delimiter. Only one field_delimiter string is allowed per extract.

Type:	**string**
Required:	No
Default:	,

`extract.<file_group_name>.record_delimiter`

The `record_delimiter` specifies the string that indicates the end of a record. The record_delimiter character applies to any unquoted and unescaped text when extracting records from the source file. This can be a single character or a two-character string delimiter. Only one `record_delimiter` string is allowed per extract.

> The default is a standard newline character `\n`. Depending on the file, typical settings for this delimiter are a newline `\n`, a carriage return `\r` or a carriage return and newline `\r\n`.

Type:	**string; maximum 2 characters**
Required:	No
Default:	\n

`extract.<file_group_name>.field_optionally_enclosed_by`

Also referred to as the “quote character,” the `field_optionally_enclosed_by` settings specifies the character that is optionally used to enclose a field or column. The data between enclosure characters can contain delimiter characters. The enclosure character itself must be escaped inside of an enclosed field. Common values are `"` to enclose fields by double quotes. Because this setting defines an optional enclosure, not all fields in the record need to be enclosed. The `escape` character used to escape the enclosure character is separately configured and defaults to `"` per the RFC-4180 standard.

Type:	string; maximum 1 character
Required:	No
Default:	”

Example:

In the following pipe delimited record, the double quote character can be specified so that the final column will be `this field has a | in it` when loaded into the database. If no enclosure character is specified, this record would yield four columns instead of three.

Text

```
a|b|"this field has a | in it"
```

`extract.<file_group_name>.escape`

The `escape` settings specifies the character that is used as an escape occurrences of the enclosing character used to value so that the following character is escaped. This is typically used in enclosed columns to escape the enclosure character defined as `field_optionally_enclosed_by`.

Type:	string; maximum 1 character
Required:	No
Default:	“

`extract.<file_group_name>.num_header_lines`

The `num_header_lines` setting defines the number of header lines in the source. The system reads these header lines but does not load them as data.

If you do not specify **headers**, the system uses the final line within this section to name the columns in each file. Then, the system uses these names in the Transform section to refer to column values. The system names columns for each file, so the names available in the Transform section might change if the file header changes.

⚠ This setting only applies to file-based source types.

Type:	**int**
Required:	No
Default:	0

`extract.<file_group_name>.headers`

The `headers` setting defines the header labels associated with each column in a delimited file. This array of values is associated with the columns in order from left to right. The strings assigned to the column names are used in the Transform section to refer to column values. Any explicitly supplied headers that are an empty string `""` or a `null` literal will be skipped during extraction. This can have performance benefits if all columns are not needed.

Type:	**string[]**
Required:	No
Default:	null

`extract.<file_group_name>.include_columns`

The `include_columns` setting restricts extraction to the columns with the specified names. If you do not need all columns during transformation, you might improve performance when you use the `include_columns` setting.

If there are no headers and you specify this setting, the system names columns based on their original positions before other columns are filtered out. For example, if there are four columns and `$0` and `$3` are included, the system names these columns `$0` and `$3` during transformation, not `$0` and `$1`.

If you specify the `exclude_columns` setting, you cannot specify this setting.

© 2026 Ocient, Inc.

Type:	string[]
Required:	No
Default:	null

`extract.<file_group_name>.exclude_columns`

The `exclude_columns` setting causes the LAT to not extract any of the specified columns. If you do not need all columns during transformation, you might improve performance when you use the `exclude_columns` setting.

If there are no headers and you specify this setting, the system names columns based on their original positions before other columns are filtered out. For example, if there are four columns and `$1` and `$2` are excluded, the system names these columns `$0` and `$3` during transformation, not `$0` and `$1`.

If you specify the `include_columns` setting, you cannot specify this setting.

Type:	string[]
Required:	No
Default:	null

`extract.<file_group_name>.skip_blank_lines`

The `skip_blank_lines` setting specifies whether lines with only whitespace should be skipped or processed.

Type:	boolean
Required:	No
Default:	False

`extract.<file_group_name>.skip_comment_lines`

The `skip_comment_lines` setting specifies whether lines with a leading comment character should be skipped or processed.

Type:	**boolean**
Required:	No
Default:	true

`extract.<file_group_name>.comment_character`

The `comment_character` setting specifies which character represents the start of a comment line.

Type:	**string; maximum 1 character**
Required:	No
Default:	’#‘

Example Delimited Extract Configuration

In this example extract configuration, a pipe delimited file type with five column names is provided. The file group name assigned to this file group was defined as “delimited_user_data” in the `source` section of the pipelines. Whitespace will be trimmed from each field and blank lines will be skipped instead of reporting errors. Fields with special characters can be escaped by `"`. Any literal values of `NULL` or `N/A` are converted to `null`. When transforming this data, columns can be referred to by `id`, `username`, `last_login_at`, `first_name`, and `last_name`. The fourth column is listed as `null` in the header, so it will be skipped during extraction. Because `num_header_lines` is not specified, the files are assumed to have no headers.

© 2026 Ocient, Inc.

JSON

```
{
    "extract": {
        "delimited_user_data": {
            "record_type": "delimited",
            "headers": ["id", "username", "last_login_at", null, "first_name", "las
            "field_delimiter": "|",
            "trim_whitespace": true,
            "skip_blank_lines": true,
            "include_columns": ["id", "username", "last_name"],
            "field_optionally_enclosed_by": "\"",
            "null_strings": ["NULL", "N/A"]
        },
        ...
    },
    ...
}
```

Fixed Width Binary Extract

The Fixed Width Binary (FWB) Extract type allows the LAT to extract data from binary source records that have a fixed number of bytes per column. FWB source records typically come from sources like mainframe systems like the IBM® Z® series that use Cobol Copybooks to produce character, integer, floating point, and decimal data in various byte formats. The FWB Extract type requires users to specify the byte offset, width in bytes, and the type of extraction to use on the indicated bytes for each column. The FWB type also allows users to specify a record length in bytes rather than a character record delimiter. FWB files are typically extracted using `encodings` such as `cp500` and `cp037`.

Fixed Width Binary Data Types

The following data types are supported when extracting Fixed Width Binary data. All data types require specification of `offset` and `width` to locate the bytes to extract in each record.

Data Type	Description	Supported Settings
string	Represents bytes of character data following the encoding defined in the extract settings. Bytes are extracted into a string for transformation and loading.	`encoding` - inherited
integer	Represents a signed integer. Uses the `endianness` setting to determine how bytes should be interpreted. The `integer` type can be used for 1, 2, 4, or 8 bytes of data. These correspond to mainframe concepts of a Byte, Half-word, Full-word, and Double-word.	`endianness` - inherited
unsigned_integer	Represents a unsigned integer. Uses the `endianness` setting to determine how bytes should be interpreted. Can be used for 1, 2, 4, or 8 bytes of data. The largest integer data type in the Ocient System is a signed BIGINT, so the maximum value for an `unsigned_integer` is 2^63-1.	`endianness` - inherited
float	Represents a floating point number as defined by the IEEE®-754 floating point specification. Can be used for 4 or 8 bytes of data. 4 bytes represents a single precision floating point number, and 8 bytes represents a double precision floating point number.	
packed_decimal	A Packed Binary Coded Decimal format for decimal precision that packs two integral values in each byte and encodes the sign in the final byte. The number of decimal points represented by the data is supplied with a separate extraction setting. The sign is encoded in the 4 least significant bits of the final byte. A value of `0xD` or `0xB` indicates a negative sign. Any other value including the common values of `0xF`, `0xE`, `0xC` or `0xA` indicating positive or unsigned.	`endianness` - inherited `decimals` - number of digits to place the right of the decimal point `sign_format` - defaults to 'trailing' byte format indicating that the sign is encoded in the most significant four bits of the final byte; provided for future extensibility.
zoned_decimal	A Zoned Binary Coded Decimal format for decimal precision that packs two integral values in each byte and encodes the sign in the final byte. The number of decimal points represented by the data is supplied with a separate extraction setting. The sign is encoded in the 4 most significant bits of the final byte. A value of `0xD`, `0xB`, `0x7` indicates a negative sign. Values of `0xF`, `0xE`, `0xC` or `0xA` indicate positive or unsigned.	`endianness` - inherited `decimals` - number of digits to place the right of the decimal point `sign_format` - defaults to 'trailing' byte format indicating that the sign is encoded in the most significant four bits of the final byte; provided for future extensibility.

© 2026 Ocient, Inc.

Fixed Width Binary Configuration

`extract.<file_group_name>.record_width`

The `record_width` setting specifies the total number of bytes in each record. Files are read serially, and the next record is assumed to start on the following byte. The `record_width` can be greater than the number of bytes used in extracting the individual columns.

Type:	**integer**
Required:	Yes
Default:	

`extract.<file_group_name>.padding_character`

The `padding_character` setting specifies a character in a fixed width file type that should be removed from extracted character data. After extracting a character field, any leading or trailing occurrences of this character will be stripped from the ends of the character data. Typical examples are the space character or an underscore.

Type:	**string**
Required:	No
Default:	” ” (in the specified encoding)

`extract.<file_group_name>.endianness`

The `endianness` setting specifies whether the extraction of numeric fields should use Big Endian or Little Endian byte ordering to process extracted bytes. `endianness` impacts integer, packed_decimal, and zoned_decimal column types.

Type:	**string**
Required:	No

Type:	string
Default:	big

Allowed values:

- `big` : Big Endian
- `little` : Little Endian

`extract.<file_group_name>.columns`

A extract columns configuration object. A collection of columns and their associated extraction configuration. Keys are column names and values are **Column Extracts configuration**.

Required keys:

- **extract.columns.<columns>**

Column Extracts

Column extract configurations for Fixed Width Binary records.

`extract.<file_group_name>.columns.<column>`

A column extract configuration object keyed by a column name. A column's value is defined as an extract definition. The extract definition will indicate the offset and width to locate the bytes to extract, the data type to use in the extraction, and any additional parameters about the bytes to extract.

Required keys:

- offset
- width
- data_type

Optional keys:

- decimals

- sign_format

`extract.<file_group_name>.columns.<column>.offset`

Number of bytes to skip from the beginning of the record before beginning to extract bytes for this column. Note that `offset` is relative to the beginning of the record, so arbitrary offsets can be used including overlapping sets of bytes with other columns.

Type:	**integer**
Required:	Yes
Default:	

`extract.<file_group_name>.columns.<column>.width`

Number of bytes to extract from the `offset` location.

Type:	**integer**
Required:	Yes
Default:	

`extract.<file_group_name>.columns.<column>.data_type`

Data type to extract from the bytes that are specified by `offset` and `width`. See **Fixed Width Binary Data Types** for reference on available data types.

Type:	**string**
Required:	Yes
Default:	string

Allowed values:

- `string`
- `integer`

- `unsigned_integer`
- `float`
- `packed_decimal`
- `zoned_decimal`

Note that each `data_type` has specific settings that are allowed to control the extraction to that `data_type`. Some of these are inherited from the overall extract configuration, while others are defined on each column.

`extract.<file_group_name>.columns.<column>.decimals`

The number of numeric values that are placed to the right of the decimal point in a `zoned_decimal` or `packed_decimal` data type. For example, if `decimals` is set to `2` and the data in a `zoned_decimal` represents the numerals `12345`, this would create the decimal `123.45` in the extracted value.

> `decimals` configures the `scale` of the extracted decimal and the total number of digits present in the source data determines the precision. If insufficient scale or precision exists in the target Ocient column, data will fail to load.

Type:	**integer**
Required:	No
Default:	0

Applies to:

- `packed_decimal`
- `zoned_decimal`

`extract.<file_group_name>.columns.<column>.sign_format`

In decimal types, the sign for the number can be encoded in different ways. Currently, only the `trailing` format is supported which requires the sign to be encoded in 4 bits of the last byte in the column data. This setting is provided for future extensibility. The

values that indicate a positive or negative sign are listed in **Fixed Width Binary Data Types**.

Type:	**string**
Required:	No
Default:	trailing

Allowed values:

- `trailing` - the sign is encoded in the most significant 4 bits of the last byte in the binary coded decimal.

Applies to:

- `packed_decimal`
- `zoned_decimal`

Fixed Width Binary Extract Example

The following example illustrates extracting fixed width binary data from a file group that was given the name “order_files” in the `source` section of the pipeline. The file has 3100 bytes per record and is only extracting a few columns.

JSON

```
{
    "extract": {
        "order_files": {
            "record_type": "fixed_width_binary",
            "record_width": 3100,
            "endianness": "big",
            "columns": {
                "order_id": {
                    "offset": 0,
                    "width": 4,
                    "data_type": "integer"
                },
                "username": {
                    "offset": 4,
                    "width": 20,
                    "data_type": "string"
                },
                "order_total": {
                    "offset": 24,
                    "width": 10,
                    "data_type": "packed_decimal",
                    "decimals": 2
                },
                "sales_tax_total": {
                    "offset": 34,
                    "width": 8,
                    "data_type": "zoned_decimal",
                    "decimals": 3
                }
            }
        }
    }
}
```

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Transform Configuration

⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

`transform`

A transform configuration object.

Required keys:

- Transform Configuration source only: **transform.topics**
- File sources only: **transform.file_groups**

`transform.topics` [#transform-topics]

A collection of topics and their associated configuration. Keys are topic names and values are **topic configuration objects** . Each `<topic>` set as a key in this object represents an Apache® Kafka® topic defined in Kafka.

ⓘ When loading from an `s3` or `local` File Source type, `transform.file_groups` should be used instead of `transform.topics` . Each key in the `file_group` must match a `file_group_name` defined in the Source section of the pipeline configuration.

Type:	**object**
Required:	Yes
Default:	

`transform.database` [#transform-database]

A database to be used to fully qualify any table names that are not fully qualified.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

For example, if `transform.database` is set to `myDatabase`, a table name of the form `schema.table` will become `myDatabase.schema.table`.

Type:	**string**
Required:	No
Default:	null

`transform.schema`

A schema to be used to fully qualify any table names that are not fully qualified. Specifying this property requires that **transform.database** is specified.

For example, if `transform.database` is set to `myDatabase` and `transform.schema` is set to `mySchema`:

- A table name of the form `table` will become `myDatabase.mySchema.table`
- A table name of the form `schema.table` will become `myDatabase.schema.table`
- A table name of the form `database.schema.table` will stay as `database.schema.table`

Type:	**string**
Required:	No
Default:	null

Kafka Load Transform Example

JSON

```
{
    ...
    "transform": {
        "topics": {
            "topic_1": {
                "tables": {
                    ...
                }
            },
            "topic_2": {
                "tables": {
                    ...
                }
            }
        }
    }
}
```

File Based Load Transform Example

Unlike Kafka, File loads define file groups in the source section of the pipeline configuration. The “file_groups” defined in the source and transform sections must match.

© 2026 Ocient, Inc.

JSON

```
{
    "source": {
        "type": "s3",
        ...
        "file_groups": {
            "file_group_1": {
                ...
            },
            "file_group_2": {
                ...
            },
        }
    }
    "transform": {
        "file_groups": {
            "file_group_1": {
                "tables": {
                    ...
                }
            },
            "file_group_2": {
                "tables": {
                    ...
                }
            }
        }
    }
}
```

Topics

Topic configuration objects.

Required keys:

- **transform.topics.<topic>.tables**

> For file based loads, `topics` are replaced by `file_groups`, but all other settings are equivalent.

`transform.topics.<topic>.filter`

A record filter to apply at the topic level. See **LAT Record Filtering** for details.

Type:	**string**
Required:	Yes
Default:	

`transform.topics.<topic>.tables`

A collection of tables and their associated configuration. Keys are table names and values are **columns configuration**.

Type:	**object**
Required:	Yes
Default:	

Tables

Table configuration objects.

Required keys:

- **transform.topics.<topic>.tables.<table>.columns**

`transform.topics.<topic>.tables.<table>.filter`

A record filter to apply at the table level. See **LAT Record Filtering** for details.

Type:	**string**
Required:	Yes
Default:	

`transform.topics.<topic>.tables.<table>.columns`

A collection of columns and their associated configurations. Keys are table names and values are **Columns**.

Columns

Column transformation configurations.

Required keys:

- **transform.topics.<topic>.tables.<table>.columns.<column>**

`transform.topics.<topic>.tables.<table>.columns.<column>`

A column transformation keyed by a column name. A column's value is defined as a transformation expression. The expression will query the record and return a value that is loaded into the associated column. The grammar of these expressions uses **JMESPath** enhanced with some custom Ocient transformations and User Defined Transformations (UDTs).

Type:	**string**
Required:	Yes
Default:	

Complex Transform Example

JSON

```
{
    "transform": {
        "database": "adtechdb",
        "schema": "adtech",
        "topics" : {
            "dsp.Auctions" : {
                "tables" : {
                    "auction" : {
                        "columns" : {
                            "auctionid" : "auctionId",
                            "created" : "created",
                            "bidder_nodeid" : "bidder.nodeId",
                            "deals_dealid" : "deals[].dealId"
                        }
                    },
                    "trafficSource" : {
                        "columns" : {
                            "auctionid" : "auctionId",
                            "created" : "created",
                            "sourcesellerid" : "trafficSource.sourceSellerId",
                            "domainname" : "trafficSource.domainName",
                            "contextualdataset_externalentryid" : "trafficSource.si
                            "dimensions_hw" : "join('x', [dimensions.height, dimens
                        }
                    },
                    "feedback" : {
                        "columns" : {
                            "auctionid" : "auctionId",
                            "created" : "created",
                            "received" : "received",
                            "campaignid" : "EXPLODE(auctionResponseFeedbacks[].camp
                            "won" : "EXPLODE(auctionResponseFeedbacks[].won)",
                            "winningbid" : "EXPLODE(auctionResponseFeedbacks[].winn
                            "sourcelossreasoncode" : "EXPLODE(auctionResponseFeedba
                        }
                    }
                }
            }
        }
    }
}
```

© 2026 Ocient, Inc.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Data Transformation with JMESPath

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

Within a transformation each column is configured with a JMESPath expression describing where to extract the column's value and any transformations to apply to that value.

For information on common JMESPath functions and filtering tools, see the **Standard JMESPATH Functions** section.

For a list of functions to modify data during loading, see the **LAT Transformation Functions** page.

Information on customizing your own transformation functions can be found on the **LAT User-Defined Transformations** page.

For expressions to select elements from an array matching certain criteria, see the **LAT Record Filtering** page.

Notes/Warnings

JSON Keys with Numbers and Special Characters

You must escape JSON keys that begin with a number or special character in the pipeline configuration.

- For a JSON record: `{ "0_123" : "value" }` transform this record as follows to write "value" to "my_col" --- `{ "my_col" : "\"0_123\"" }`
- For a JSON record: `{ "a-col" : 123 }` transform this record as follows to write "value" to "my_col" --- `{ "my_col" : "\"a-col\"" }`

Case Sensitivity

© 2026 Ocient, Inc.

- The Ocient System normalizes database objects (e.g., table names, column names, schemas) in LAT Transformations according to the same rules as the database unless explicitly escaped with quotes.
  - The Ocient System normalizes all names to lowercase unless explicitly escaped. For example, the transformation: `{ "myCol" : "value" }` normalizes to `mycol` and matches a database column `mycol` that was not escaped at creation time.
  - If you create the database column as an escaped literal such as `"myCol"` then the LAT transformation must be specified as: `{ "\"myCol\"" : "value" }` to correctly map to the column.
- Transformation functions are case sensitive, so represent them by using all lower case letters (e.g., `st_geomfromewkb(my_column)`).

Literals

- You must escape literal values with a backtick. This applies to numbers, booleans, and empty arrays (otherwise the Ocient System interprets the value as a JSON transformation operator).

Right:

JSON

```
{
    "column_1": "`0`",
    "column_2": "`1`",
    "column_3": "`true`",
    "column_4": "`[]`"
}
```

Wrong:

JSON

```
{
    "column_1": "0",
    "column_2": "1",
    "column_3": "true",
    "column_4": "[]"
}
```

- You can write string literals with a single quote.

JSON

```
{ "column_1": "'example_value'" }
```

Loading Numbers in Exponential Notation

When you need to load numbers from exponential notation (e.g. `1.20E-05`), you might require a conversion to load them into the target Ocient column or to perform transformations on the data. Use the `to_number` transformation to load this into a column of type `decimal` or to convert to a numeric value for further transformations like multiplication.

Example:

A source record in a CSV file might have a number like `7.00E-05` that appears as a string in the field named `source_field`.

JSON

```
{ "column_1": "to_number(source_field)" }
```

This JSON code converts the string in exponential notation into the double `0.00007` that is now suitable for loading into decimal, double, or float columns, or for use in mathematical transformation functions.

Flatten, Array Access, and Pipes

For general syntax help for arrays, projections, and pipe expressions, see **JMESPath website**. Flattening arrays, accessing entire arrays, and using pipe expressions are three important operations when you work with nested data in LAT transformations.

- The flatten operator (e.g., `[]`) flattens an array one level. This also removes NULL values from the array.

> ⚠ This can cause unintended consequences if multiple arrays need to remain aligned for data processing.

- When you work with nested arrays, you can maintain array hierarchy. For example, you can replace an array index such as `*` with `[*]` to maintain the hierarchy.
- You can use the pipe operator (e.g., `a.b[*] | [0]`) to stop the projection of an expression into the prior expression.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Transformation Functions

⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

LAT supports many transformation functions to modify data during loading. The functions include standard JMESPath Functions and Ocient-specific transformation functions.

ⓘ Transformation functions in LAT are case sensitive. The standard syntax for transformation function calls is all lower case characters.

Standard JMESPath Functions

The following standard JMESPath Functions are supported in LAT Transformation expressions. For details about using these functions, see **JMESPath Functions**.

JMESPath Function	Function Signature	Example
abs	number abs(number $value)	abs(num_1)
avg	number avg(array[number] $elements)	avg(array_1)
contains	boolean contains(array|string *subject*, *any* subject, anysearch)	contains(array_1, 'test')
ceil	number ceil(number $value)	ceil(num_1)
ends_with	boolean ends_with(string *subject*, *string* subject, stringprefix)	ends_with(str_1, 'ing')
floor	number floor(number $value)	floor(num_1)
join	string join(string *glue*, *array\[string]*glue, array\[string]stringsarray)	join(',', array_1)
keys	array keys(object $obj)	keys(obj_1)
length	number length(string|array|object $subject)	length(array_1)
map	array[any] map(expression→any→any expr, array[any] elements)	map(&to_string(@), array_1)

JMESPath Function	Function Signature	Example
max	number max(array[number]|array[string] $collection)	max(array_1)
max_by	max_by(array elements, expression→number|expression→string expr)	max_by(array_1, &age)
merge	object merge([object *argument, [, object $…]])	merge(obj_1, obj_2, obj_3)
min	number min(array[number]|array[string] $collection)	min(array_1)
min_by	min_by(array elements, expression→number|expression→string expr)	min_by(array_1, &age)
not_null	any not_null([any *argument*\[, *any*argument\[, any…]])	not_null(val_1, val_2, val_3)
reverse	array reverse(string|array $argument)	reverse(str_1)
sort	array sort(array[number]|array[string] $list)	sort(array_1)
sort_by	sort_by(array elements, expression→number|expression→string expr)	sort_by(array_1, &age)
starts_with	boolean starts_with(string *subject*, *string* subject, stringprefix)	starts_with(str_1, ‘chi’)
sum	number sum(array[number] $collection)	sum(array_1)
to_array	array to_array(any $arg)	to_array(val_1)
to_string	string to_string(any $arg)	to_string(val_1)
type	string type(array|object|string|number|boolean|null $subject)	type(val_1)
values	array values(object $obj)	values(obj_1)

Other Transformation Functions

The following transformation functions are also available in LAT transformation expressions.

add

Returns the sum of the two arguments.

Signature:	`number add(number $a, number $b)`
Arguments:	- `a` : first argument - `b` : second argument
Example:	`add(int_col_0, int_col_1)`

`array_cap`

Caps the length of an array to a maximum number of elements.

Signature:	`array array_cap(Array[any] $array_1, number $max_elements)`
Arguments:	- `array_1` : the array to cap - `max_elements` : number of elements to cap the array at
Example:	`array_cap(array_1, 1000)`

`array_cat`

Concatenate two arrays, returning an array. Accepts an array or null value, returning an empty array if both parameters are null.

Signature:	`array array_cap(array[any] $array_1, array[any] $array_2)`
Arguments:	- `array_1` : first array - `array_2` : second array concatenated to first
Example:	`array_cat(array_1, array_2)`

`concat`

Concatenates two strings or converts arguments into strings and concatenates them into a string.

© 2026 Ocient, Inc.

Signature:	`string concat(any $a, any $b)`
Arguments:	- `a` : first argument (automatically converted to string when not specified as a string) - `b` : second argument (automatically converted to string when not specified as a string)
Example:	`concat("ocient", `10`)`

`divide`

Returns the floating-point division of the two arguments.

Signature:	`number divide(number $a, number $b)`
Arguments:	- `a` : first argument - `b` : second argument
Example:	`divide(int_col_0, int_col_1)`

`hash_code`

Returns an integer hash code for a string, number, or Boolean.

Signature:	`number hash_code(string|number|boolean $val)`
Arguments:	`val` : the value to hash

Example:	`hash_code(null) → null`
	`hash_code("Chicago") → -1884315574`
	`hash_code("test123") → -1422501792`
	`hash_code(`12345`) ->`
	`12345` @@HTML_TABLE_TOKEN_3@@ `hash_code(`123.45`) ->`
	`-1936584703` @@HTML_TABLE_TOKEN_4@@ `hash_code(`true`) ->`
	`3` @@HTML_TABLE_TOKEN_5@@ `hash_code(`false`) -> 1`

`if`

Returns one of the other arguments based on the truth value of the first. When the first argument is true, return the second argument. When false, return the third.

A NULL input into the first argument will be evaluated as false. Both branches are eagerly evaluated.

Signature:	`any if(boolean $b, any $if_true, any $if_false)`
Arguments:	- `b` : boolean value that determines which of the options to return - `if_true` : value returned if b is true - `if_false` : value returned if b is false
Example:	`if(value > `5`, "North America", "Europe")`

`json`

Parses a string into a JSON value for use in further JMESPath operations. All values other than string are returned unmodified. Strings containing null, boolean, number, array, and object types are supported.

Signature:	`object json(string $val)`
Arguments:	`val` : string to parse
Example:	`json("{ \"key\": 10 }")`

`lazy_if`

Returns one of the other arguments based on the truth value of the first. When the first argument is true, return the second argument. When false, return the third. The second and third arguments are both expressions. Branches are lazily evaluated.

Signature:	`any lazy_if(boolean $b, expression $if_true, expression $if_false, any data)`
Arguments:	- `b` : boolean value that determines which of the options to return - `if_true:` expression evaluated and returned if true - `if_false` : expression evaluated and returned if false - `data` : information used to evaluate either expression
Example:	`if(value > 5, &"North America", &abs(@), 1.0)`

`left`

Returns the `num_chars` leftmost characters of the string column. If null, null is returned. If the index is greater than the length of the string, the entire string is returned.

Signature:	`string left(string $val, number $num_chars)`
Arguments:	- `a` : string column - `num_chars` : number of chars to use from the left of a string column.
Example:	`left("ocient", '3')`

`lower_case`

Returns lower case version of a string.

Signature:	`string lower_case(string $a)`
Arguments:	`a` : string to lower case
Example:	`lower_case("OCIENT")`

`lpad`

Pads an input string to a specified length with a padding string added to the left side.

Signature:	`string lpad(string $val, number $length, string $padding)`
Arguments:	- `val` : the input string. This cannot be null. - `length` : the length you want after padding. If the input string is longer than this length, it is truncated to length characters - `padding` : the padding string. If no padding string is provided, the input string is padded with spaces.
Example:	`lpad(string_1, '3', '_')`

`ltrim`

Removes leading contiguous instances of a set of characters in a given string.

Signature:	`string ltrim(string $val, string $strip_chars)`
Arguments:	- `val` : the input string to trim - `strip_chars` : set of characters that should be removed (default is ' ' if the value is null or not specified)

Example:	`ltrim(null, ' ') → null`
	`ltrim(' Chicago ') → 'Chicago '`
	`ltrim(' Chicago ', ' ') → 'Chicago '`
	`ltrim(' Chicago ', null) → 'Chicago '`
	`ltrim('123test123', '321') → 'test123'`

`map`

An override of JmesPath's built in map which maps a given expression into elements of an array. This override makes the small adjustment that if the array itself is null, then a null is returned.

Signature:	`array map(expression $expr, array[any] array_1)`
Arguments:	- `expr` : expression the expression to map - `array_1` : array[any] the array to which the expression will be mapped
Example:	`map(&if(@ == 1, true, false), [1,2,1,1,2]) → [true, false, true, true, false]`

`matches`

Returns Boolean for whether this input matches the pattern.

Flags can be:

- `s` → Pattern.DOTALL
- `m` → Pattern.MULTILINE
- `i` → Pattern.CASE_INSENSITIVE
- `x` → Pattern.COMMENTS
- `q` → Pattern.LITERAL

© 2026 Ocient, Inc.

Signature:	`boolean matches(string $val, string $pattern, string $flags)`
Arguments:	- `val` : value to match - `pattern` : regex to match against - `args` : regex flags
Example:	`matches(col_1, ".*mytest.*", 'i')`

`millis_to_timestamp`

Converts an integer number of milliseconds since the Epoch to an Ocient timestamp column.

Signature:	`number millis_to_timestamp(number $t)`
Arguments:	`t` : value to convert to an Ocient timestamp
Example:	`millis_to_timestamp(my_millis)`

`multiply`

Returns the floating-point multiplication of the two arguments.

Signature:	`number multiply(number $a, number $b)`
Arguments:	- `a` : first argument - `b` : second argument
Example:	`multiply(int_col_0, int_col_1)`

`nanos_to_timestamp`

Converts an integer number of nanoseconds since the Epoch to an Ocient timestamp column.

Signature:	`number nanos_to_timestamp(number $t)`
Arguments:	- `t` : value to convert to an Ocient timestamp
Example:	`nanos_to_timestamp(my_nanosecs)`

`normalize_space`

Removes leading and trailing whitespace and replaces connected whitespace with a single space.

Signature:	`string normalize_space(string $val)`
Arguments:	`val` : string to normalize
Example:	`normalize_space(col_1)`

`now, current_timestamp`

Returns the current timestamp in nanos. This will correctly load into an Ocient timestamp column

Signature:	`number now() | number current_timestamp()`
Arguments:	None
Example:	`now(), current_timestamp()`

`null_if, nullif`

Returns null if the first argument matches any of the following arguments. Can be applied to multiple columns or to compare a column with a constant. Can also be used with more than two arguments in which case returns null if a matches any of the subsequent arguments. Returns the value the first argument if it does not match any other arguments.

Signature:	`any null_if(any $val, any $c0, [, any $…])`
Arguments:	- `val` : value to compare - `c0` : required value to compare against - `…` : optional multiple values to compare against
Example:	`null_if(col_a, "NULL", "NONE")`

`parse_array`

Given a string, parses it into an array of strings. Handles nested arrays as well.

Signature:	`array parse_array(string $array_string, string $bracket_char, string $delim_char, int $dimensions)`
Arguments:	- `array_string` : the string of the array to be parsed - `bracket_char` : the opening bracket character to use - `delim_char` : the delimiter character to use - `dimensions` : the dimension of this array
Example:	`parse_array("[ 1, 2, 3]", "[", ",", `1`) ⇒ [1, 2, 3]` @@HTML_TABLE_TOKEN_0@@ `parse_array("[[1,2],[3], []]", '[', ',', `2`) ⇒ [[1,2], [3], []]` @@HTML_TABLE_TOKEN_1@@ `parse_array("{ \"cat\", \"hat\" }", "{", ",", `1`) ⇒ ["cat", "hat"]`

`record_uuid`

Returns string that represents a unique identifier for the record in a given pipeline for a given file_group or topic. This return value will be consistent over multiple calls in the pipeline's transformation configuration for the same source record. Can be used as a surrogate key when loading multiple tables from the same source record in a pipeline to facilitate joins when no unique identifier exists on the source record.

Signature:	`string record_uuid()`
Arguments:	None
Example:	`record_uuid() -> "6064fc40-9961-4f5e-b74d-090458e4a609"`

`replace`

Returns string with text matching the regex pattern replaced by the replacement text.

Flags can be:

- `s` → Pattern.DOTALL
- `m` → Pattern.MULTILINE
- `i` → Pattern.CASE_INSENSITIVE
- `x` → Pattern.COMMENTS
- `q` → Pattern.LITERAL

Signature:	`string replace(string $val, string $pattern, string $replacement_text, string $flags)`
Arguments:	- `val` : value to search for replacement - `pattern` : section to replace - `replacement_text` : replacement string - `args` : regex flags
Example:	`replace(col_1, '[oO]ld', 'new', 'q')`

`right`

Returns the `num_chars` rightmost characters of the string column. If null, null is returned. If the index is greater than the length of the string, the entire string is returned.

Signature:	`string right(string $a, number $num_chars)`
Arguments:	- `a` : string column - `num_chars` : number of chars to use from the right of a string column.
Example:	`right("ocient", 3)`

`rpad`

Pads an input string to a specified length with a padding string added to the right side.

Signature:	`string rpad(string $val, number $length, string $padding)`
Arguments:	- `val` : the input string. This cannot be null. - `length` : the length you want after padding. If the input string is longer than

Signature:	`string rpad(string $val, number $length, string $padding)`
	this length, it is truncated to length characters - `padding` : the padding string. If no padding string is provided, the input string is padded with spaces.
Example:	`rpad(string_1, `3`, '_')`

`rtrim`

Removes trailing contiguous instances of a set of characters in a given string.

Signature:	`string rtrim(string $val, string $strip_chars)`
Arguments:	**-** `val` : the input string to trim - `strip_chars` : set of characters that should be removed (default is ' ' if the value is null or not specified)

Example:	`rtrim(null, ' ') → null`
	`rtrim(' Chicago ') → ' Chicago'`
	`rtrim(' Chicago ', ' ') → ' Chicago'`
	`rtrim(' Chicago ', null) → ' Chicago'`
	`rtrim('123test123', '321') → '123test'`

`sample`

Returns true or false based on the given sample rate. The sample rate must be a number. Commonly used inside of `IF` statements. An exception will be thrown if the sample rate is outside of [0.0, 1.0].

Signature:	`boolean sample(number $rate)`
Arguments:	`rate` : rate to return true. Must be between [0.0, 1.0].
Example:	`sample(`0.5`)`

`st_forcepolygonccw`

Forces a polygon to have a counterclockwise rotation of the outer polygon ring and a clockwise rotation of the inner polygon ring. You can use this function to convert polygons extracted from systems that interpret polygon rotation differently.

Signature:	`geometry st_forcepolygonccw(string $wkt_geometry)`
Arguments:	`wkt_geometry` : WKT string representing the polygon to rotate. Also accepts WKT of POINT or LINESTRING and returns those types unmodified.
Example:	`st_forcepolygonccw('POLYGON(…)')`

st_geomfromewkb

Returns a geometry from an Extended Well-Known Binary (EWKB) and Well-Known Binary (WKB) representations of a geometry. You can load the result of this function into POINT, LINESTRING, and POLYGON column types.

Signature:	`geometry st_geomfromewkb(string ewkb)`
Arguments:	`wkb` : hex encoded string Extended WKB or WKB representation of a geometry. EWKB or WKB can be standalone hexadecimal characters or can be prefixed with either 0x or \x.

Example:	`st_geomfromewkb('0103…')`
	`st_geomfromewkb('0x0103…')`
	`st_geomfromewkb('\x0103…')`

st_point

Constructs an POINT from numeric coordinates that represent the longitude and latitude of the point.

Signature:	`geometry st_point(float longitude, float latitude)`
Arguments:	- `longitude` : floating point longitude of the point to construct - `latitude` : floating point latitude of the point to construct
Example:	`st_point(float_col_1, float_col_2)`

substring

Returns the substring beginning at `start_index` (with indexes beginning at 1) to the end of the string, or to `num_chars` characters.

Signature:	`string substring(string $val, number $start_index, [number $num_chars])`
Arguments:	`val` : string to substring `start_index` : index to start at. First character is index 1. `num_chars` : optional. The maximum number of chars to return
Example:	`substring(str_col, \`10\`)` @@HTML_TABLE_TOKEN_0@@ `substring(str_col, \`10\`, \`5\`)` @@HTML_TABLE_TOKEN_1@@ `substring('Chicago Bears', \`3\`) → 'icago Bears'` @@HTML_TABLE_TOKEN_2@@ `substring('Chicago Bears', \`3\`, \`4\`) → 'icag'`

`substring_after`

Returns the string after the first occurrence of the needle.

Signature:	`string substring_after(string $val, string $needle)`
Arguments:	- `val` : string to substring - `needle` : delimiter to use to designate where to return subsequent substring.
Example:	`substring_after('my_text', '_') → 'text'`

`substring_before`

Returns the string before the first occurrence of the needle. Both parameters are literal strings.

Signature:	`string substring_before(string $val, string $needle)`
Arguments:	- `val` : string to substring - `needle` : delimiter to use to designate where to return prior substring.
Example:	`substring_before('my_text', '_') → 'my'`

`subtract`

Returns the difference of the two arguments.

Signature:	`number subtract(number $a, number $b)`
Arguments:	- `a` : first argument - `b` : second argument
Example:	`subtract(int_col_0, int_col_1)`

`to_array_length`

Turns any JSON object into an array with N copies of that object.

Signature:	`array[any] to_array_length(any $val, number $n)`
Arguments:	- `val` : value to copy into an array - `n` : number of copies to make
Example:	`to_array_length(value_col, `3`)`

`to_binary`

Converts a string to binary data for loading into an Ocient binary or hash column.

If `mode` is `'hex'`, `data` is parsed as a sequence of hexadecimal digits. Note that the sequence of digits must not begin with `0x`.

Otherwise, `mode` must be the name of a character encodings listed **here**, and `data` is encoded in that encoding.

Signature:	`binary to_binary(string $data, string $mode)`
Arguments:	- `data` : data to convert to binary - `mode` : either ‘hex’ or the name of a standard Java® character encoding
Example:	`to_binary(str_col, 'utf-16')`

`to_date`

Converts a string to a date for loading into an Ocient date column. Format string follows these **formatting rules**.

Signature:	`number to_date(string $date, string $format_string)`
Arguments:	- `date` : date string to convert - `format_string` : format string to use for conversion
Example:	`to_date(str_col, 'yyyy-MM-dd')`

`to_time`

Converts a string into a time of day for loading into an Ocient time column. Format string follows these **formatting rules**.

Signature:	`number to_time(string $time, string $format_string)`
Arguments:	- `time` : time string to convert - `format_string` : format string to use for conversion
Example:	`to_time(str_col, 'HH:mm:ss.nnnnnnnnn')`

`to_timestamp`

Converts a string to timestamp for loading into an Ocient timestamp column. Format string follows these **formatting rules**.

Signature:	`number to_timestamp(string $timestamp, string $format_string)`
Arguments:	- `timestamp` : timestamp string to convert - `format_string` : format string to use for conversion
Example:	`to_timestamp(str_col, 'yyyy-MM-dd HH:mm:ss.SSS')`

`to_tuple`

Works on tuples that get tokenized at the first nesting level by a given delimiter. Nested tuples, arrays or objects are considered entirely. For each tokenized tuple element, a transformation function can be specified that is applied to the tuple element. Tokenized elements are always treated as strings, requiring additional transformation when being treated as numbers.

Signature:	`array to_tuple(string $tuple_string, string $bracket_char, string $delim_char [, expression $transformation_function …])`
Arguments:	**-** `tuple_string` : the string of the tuple to be parsed. Must be encapsulated in the given bracket character and its closing peer. Empty tuple values are treated as null values. If the whole argument is NULL, the function returns NULL ignoring all given expressions in transformation_function... - `bracket_char` : the bracket character to use for parsing and result generation - `delim_char` : the delimiter character to use for parsing and result generation. The supported delimiters are: '(', '[' and '{' - `avg(…)` - `keys(…)` - `map(…)` - `sum(…)` - `value(…)` - `type(…) - partially for boolean and object types`
Example:	- `to_tuple((-1,2), '(', ',', &abs(to_number(@)), &@) → '(1.0,2)'` - `to_tuple((,'8',2), '(', ',', &if(@ != null, @, 'missing'), &if(length(@) > 0, @, 'missing'), &to_number(@) ) → '("missing","'8'",2)'`

`tokenize`

Returns array of strings after splitting on the regex pattern.

Flags can be:

- `s` → Pattern.DOTALL
- `m` → Pattern.MULTILINE
- `i` → Pattern.CASE_INSENSITIVE
- `x` → Pattern.COMMENTS
- `q` → Pattern.LITERAL

Signature:	`array[string] tokenize(string $val, string $pattern, string $flags)`
Arguments:	**-** `val` : string to tokenize - `pattern` : token regex pattern - `flags` : regex flags
Example:	`tokenize(col_1, ''), 'i` ')

`translate`

Replaces characters according to a map of characters. If the to_chars string is shorter than the from_chars string, the function removes the characters that are not mapped.

Signature:	`string translate(string $val, string $from_chars, string $to_chars)`
Arguments:	- `val` : string to translate - `from_char` : from character map - `to_chars` : to character map
Example:	`translate('Lorem Ipsum', 'mnop', '1234') → 'L3re1 I4su1'`

`trim`

Removes leading and trailing contiguous instances of a set of characters in a given string.

Signature:	`string trim(string $val, string $strip_chars)`
Arguments:	- `val` : the input string to trim - `strip_chars` : set of characters that should be removed (default is ' ' if the value is null or not specified)

Example:	`trim(null, ' ') → null`
	`trim(' Chicago ') → 'Chicago'`
	`trim(' Chicago ', ' ') → 'Chicago'`
	`trim(' Chicago ', null) → 'Chicago'`
	`trim('123test123', '321') → 'test'`

`truncate`

Truncates a decimal number to have a specified number of digits after its decimal point.

If `decimal` is null, this function returns null.

> ⚠ During the transformation stage, the LAT stores numbers as integers or floating point numbers by default. However, floating point numbers are inexact. For this reason, this function might return unexpected results. For example, `truncate(`0.29`, `2`)` evaluates to `0.28` , not `0.29` (because `0.29` is really `0.28999999...`).

Signature:	`number truncate(number $decimal, number $decimal_places)`
Arguments:	**-** `decimal` : the number to truncate - `decimal_places` : the number of digits to keep after the decimal point
Example:	`truncate(decimal_col, '2')`

`tuple_element_transformation`

Works on an element of a given array and applies the specified expression to that element. The function returns an array or null if specified.

Signature:	`array tuple_element_transformation(array $array_values, number $array_index, expression $transformation_function)`
Arguments:	**-** `array_values` : an array of values. If the whole argument is NULL, the function returns NULL. - `array_index` : a number pointing to an element in the array. The first index of the array is 0 - `transformation_function` : a transformation functions to apply to array element specified by array_index.
Example:	`tuple_element_transformation([-1.0,2,3, 'O', abs(@)) -> [1.0,2,3] ]`

`unique`

Return all unique values in an array.

Signature:	`array[any] unique(array[any] $vals)`
Arguments:	`vals` : array to return unique values from, removing all duplicates.
Example:	`unique(vals_col)`

`upper_case`

Returns upper case version of string.

Signature:	`string upper_case(string $a)`
Arguments:	`a` : string to upper case
Example:	`upper_case("ocient")`

`width_bucket`

Returns the bucket number of the queried value in a histogram starting at min, ending at max, and consisting of num_buckets count of buckets.

- The values are inclusive on the lower bound and exclusive of the upper bound so that all buckets are the same width.
- Also note that requesting 100 buckets will actually get you 102 possible buckets. Bucket 0 through 101. Where 0 captures all values below the minimum and 101 captures all values at the maximum and beyond.

Signature:	`number width_bucket(number $val, number $min, number max, number $num_buckets)`
Arguments:	- `val` : value to determine the bucket number for - `min` : min value of histogram - `max` : max value of histogram - `num_buckets` : number of buckets in histogram
Example:	`width_bucket(my_col, `0`, `100`, `20`)`

`zip`

Zips N one-dimensional arrays into a single two-dimensional array with each inner array containing N elements.

The resulting outer array will have a length equal to the max length of any input array. For input arrays smaller than the max length, null elements will be placed in the inner arrays.

Signature:	`array[any] zip(array[any] $a0, array[any] $a1, [, array[any] $…])`
Arguments:	- `a0` : array to zip - `a1` : array to zip

Signature:	`array[any] zip(array[any] $a0, array[any] $a1, [, array[any] $…])`
	- `…` : Optional additional arrays to zip
Example:	`zip(array_col0, array_col1, array_col2)`

Special Functions

EXPLODE

Performs an explosion on all indicated columns, producing one row per value in the exploded array. This can be applied to multiple columns in the same table. Non-exploded columns are held constant and exploded columns are zipped together in array order. If multiple exploded columns have different array lengths, the record is exploded into N records where N is the max length. Columns with array lengths less than N will have `null` values after each of their values are exploded.

You can explode a maximum of 8,192 array elements for each source record.

Usage

`EXPLODE(array[any] arr)` can be used like any other JMESPath function on a JSON Array type with a few limitations:

- `EXPLODE` can only be used as the outermost function in a JMESPath expression
- `EXPLODE` only works on a single dimension. Multidimensional explode is not supported.

Examples

Single-Column EXPLODE

Transform Configuration:

JSON

```
{
    "transform": {
        "topics" : {
            "topic0" : {
                "tables" : {
                    "table0" : {
                        "columns" {
                            "col0": "EXPLODE(array0)",
                            "col1": "integer0"
                        }
                    }
                }
            }
        }
    }
}
```

Input:

JSON

```
{
    "array0": [0, 1, 2, 3],
    "integer0": 4
}
```

Output:

JSON


```
[
    {
        "col0": 0,
        "col1": 4
    },
    {
        "col0": 1,
        "col1": 4
    },
    {
        "col0": 2,
        "col1": 4
    },
    {
        "col0": 3,
        "col1": 4
    }
]
```


Multi-Column EXPLODE

Transform Configuration:

JSON

```
{
    "transform": {
        "topics" : {
            "topic0" : {
                "tables" : {
                    "table0" : {
                        "columns" : {
                            "col0": "EXPLODE(array0)",
                            "col1": "EXPLODE(array1)",
                            "col2": "integer0"
                        }
                    }
                }
            }
        }
    }
}
```

Input:

JSON

```
{
    "array0": [0, 1, 2, 3],
    "array1": [0, 1, 2],
    "integer0": 4
}
```

Output:

JSON


```
[
    {
        "col0": 0,
        "col1": 0,
        "col2": 4
    },
    {
        "col0": 1,
        "col1": 1,
        "col2": 4
    },
    {
        "col0": 2,
        "col1": 2,
        "col2": 4
    },
    {
        "col0": 3,
        "col1": null,
        "col2": 4
    }
]
```


Note that in the 4th record `col1` has a `null` value because `array1` only has 3 values while `array0` has 4.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT User-Defined Transformations

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

In addition to built-in transformations, users can extend LAT by writing custom user-defined transformations (UDTs).

> ⓘ To bootstrap UDT development, contact Ocient Support.

UDT Interface

A UDT is implemented as a **JMESPath Java Function**. A Function is comprised of three elements:

1. **name** - the function's name determines how it can be called in a user pipeline e.g., `negate`
2. **arguments** - what inputs the function expects to be called with
3. **body** - the function's implementation, including return value

UDT names must be globally unique. LAT validates UDT names on startup to ensure that no two UDT names collide with each other and that no UDT name collides with an LAT builtin transformation.

This example function demonstrates these concepts:

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Java

```java
// negate(number) - takes an input number and changes its sign
// function name - derived from the class name, NegateFunction -> negate
public class NegateFunction extends BaseFunction {
    public NegateFunction() {
        super(
            // function arguments; this function takes exactly one argument of ty
            ArgumentConstraints.typeOf(JmesPathType.NUMBER));
      }

    // function body
    protected <T> T callFunction(Adapter<T> runtime, List<FunctionArgument<T>> ar
        double original = runtime.toNumber(arguments.get(0).value()).doubleValue(
        return runtime.createNumber(original * -1);
    }
}
```

UDT Packaging

One or more UDTs can be packaged in a jar file for consumption by LAT. In addition to the UDT implementations themselves, the jar file must contain a services file containing a mapping of all the UDTs the jar should expose for use by LAT. This service file must be located at `META-INF/services/io.burt.jmespath.function.Function` in the resulting jar, and its contents should be as follows:

Text

```
com.ocient.myudt.NegateFunction
```

ⓘ The file at the path `META-INF/services/io.burt.jmespath.function.Function` should include the fully qualified name of one or more UDTs contained within this jar to be used by LAT. Each UDT should be specified on its own line in the file.

Dependencies

If the UDT has dependencies on third party libraries, those libraries must be packaged in the same jar file. This is commonly referred to as a “fat jar”, “uber jar”, or “jar with dependencies.” Instructions for building such a jar file with Maven™ can be found **here**.

UDT Deployment

LAT can be configured to search for UDT jars in a particular directory. See **LAT Source Configuration** for details. LAT can read multiple UDTs across multiple jar files.

On startup, LAT will read all available UDTs into its configuration. To update the set of UDTs available to LAT, such as to add an additional UDT jar, a process restart is required.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Record Filtering

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

Incoming records can optionally be filtered by a JMESPath expression added with an additional filter key in a pipeline configuration.

JSON

```
{
    "transform": {
        "topics": {
            "my_topic": {
                "filter": "<jmespath-expression>",
                "tables": {
                    "my_table": {
                        "filter": "<jmespath-expression>",
                        "columns": {}
                    }
                }
            }
        }
    }
}
```

Records that pass through the filter (the filter expression evaluates to true) are processed as normal, while records that fail the filter (the filter expression evaluates to false) are dropped. Note that filters can be applied at the topic or table levels. A record needs to pass through all configured filters in order to be loaded to a table.

The filter JMESPath expression must evaluate to a `boolean` value. Examples of this are JMESPath comparators or functions that return a `boolean` like `contains`, `starts_with`, and `ends_with`.

© 2026 Ocient, Inc.

Example

Pipeline:

JSON

```
{
    "transform": {
        "topics": {
            "topic0": {
                "filter": "integer0 == `2`",
                "tables": {
                    "table0": {
                        "filter": "integer1 == `3`",
                        "columns": {
                            "col0": "integer0",
                            "col1": "integer1"
                        }
                    }
                }
            }
        }
    }
}
```

Input:

JSON

```
[
    {
        "integer0": 0,
        "integer1": 3
    },
    {
        "integer0": 2,
        "integer1": 1
    },
    {
        "integer0": 2,
        "integer1": 3
    }
]
```

Output:

JSON

```
[
    {
        "integer0": 2,
        "integer1": 3
    }
]
```

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Service Configuration

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

The LAT service is configurable using Service Configuration.

This configuration is derived from three sources, with the following precedence order:

1. Environment Variables
2. Properties File
3. Code Defaults

Configuration File Backup

When you install a new LAT instance or upgrade the existing instance, LAT creates a backup file of the `/etc/lat/lat.conf` as `/etc/lat/override.conf` and creates a new version of the `/etc/lat/lat.conf` file. The `/etc/lat/overrirde.conf` file does not control the LAT configuration. If you made prior changes to `lat.conf`, you must review `override.conf` and add any custom values into the revised `lat.conf`.

Environment Variables

`LAT_DEFAULT_WORKERS`

Default number of workers used by a newly configured pipeline.

Environment Variable:	**LAT_DEFAULT_WORKERS**
Property:	lat.default.workers
Default:	Server core count

`LAT_SHUTDOWN_DURATION`

© 2026 Ocient, Inc.

Duration, in milliseconds, to wait prior to termination on shutdown.

Environment Variable:	**LAT_SHUTDOWN_DURATION**
Property:	lat.shutdown.duration
Default:	120000

`LAT_UDT_PATH`

Directory containing zero or more user defined transformation (UDT) jars. Ensure that the specified path can be accessed by the LAT (e.g., CentOS® does not allow the LAT to access the home directory of an administrative user).

Environment Variable:	**LAT_UDT_PATH**
Property:	lat.udt.path
Default:	null

`LAT_API_PORT`

REST API and Metrics port.

Environment Variable:	**LAT_API_PORT**
Property:	lat.api.port
Default:	8080

`LAT_DATA_PATH`

Path to the LAT data directory. This directory is used to store persistent data related to pipelines (`pipeline/`) and sinks (`sink/`). In almost all cases, using the **LAT Client Command Line Interface** is preferred over editing the files in this directory manually. Ensure that the specified path can be accessed by the LAT (e.g., CentOS does not allow the LAT to access the home directory of an administrative user).

Environment Variable:	LAT_DATA_PATH
Property:	lat.data.path
Default:	.lat-data

`LAT_AUTO_START`

On LAT startup, automatically starts running a Pipeline if one is present. This only supports pipelines with Apache® Kafka® sources; File sources will not automatically start even if this configuration is set to `true`.

Environment Variable:	LAT_AUTO_START
Property:	lat.auto.start
Default:	false

`LAT_ALLOW_LOG_ORIGINAL_RECORDS`

Allow the original records that are being loaded to be logged when errors occur. This setting must be enabled for `log_original_records` to be enabled.

This setting only affects pipelines that log errors to the error log file.

Environment Variable:	LAT_ALLOW_LOG_ORIGINAL_RECORDS
Property:	lat.allow.log.original.records
Default:	false

`LAT_MAX_RETRIEVABLE_ERRORS`

A limit on the number of errors that you can retrieve using the `pipeline errors` command of the LAT client.

Environment Variable:	LAT_MAX_RETRIEVABLE_ERRORS
Property:	lat.max.retrievable.errors

Environment Variable:	**LAT_MAX_RETRIEVABLE_ERRORS**
Default:	1000

`LAT_API_PROXY_HOST`

API proxy host for OKTA® authorization requests.

Environment Variable:	**LAT_API_PROXY_HOST**
Property:	lat.api.proxy.host
Default:	null

`LAT_API_PROXY_PORT`

API proxy port for Okta authorization requests.

Environment Variable:	**LAT_API_PROXY_PORT**
Property:	lat.api.proxy.PORT
Default:	null

`LAT_API_OAUTH_DOMAIN`

Okta authorization domain.

Environment Variable:	**LAT_API_OAUTH_DOMAIN**
Property:	lat.api.oauth.domain
Default:	null

`LAT_API_OAUTH_ENDPOINTS`

A list of LAT API endpoints that Okta should apply authentication on. To override, provide a comma-delimited list, e.g., `/endpoint1,/endpoint2,/endpoint3`.

Environment Variable:	LAT_API_OAUTH_ENDPOINTS
Property:	lat.api.oauth.endpoints
Default:	null

`LAT_API_OAUTH_SERVER`

Okta authorization server.

Environment Variable:	LAT_API_OAUTH_SERVER
Property:	lat.api.oauth.server
Default:	null

`LAT_API_CERT_PATH`

API cert file path. Ensure that the specified path can be accessed by the LAT (e.g., CentOS does not allow the LAT to access the home directory of an administrative user).

Environment Variable:	LAT_API_CERT_PATH
Property:	lat.api.cert.path
Default:	null

`LAT_DNS_CACHE_TTL`

The time-to-live (TTL) in seconds that the LAT should use for its DNS cache. This configuration is only supported as an Environment Variable or as Default.

Environment Variable:	LAT_DNS_CACHE_TTL
Property:	N/A
Default:	30

`LAT_MEMORY`

Maximum JVM® heap memory. This configuration is only supported as an Environment Variable or as Default when running the LAT through the provided package service file or through the provided `lat.sh` script. If directly running the Jar JVM heap memory should be set through command line arguments.

Environment Variable:	**LAT_MEMORY**
Property:	N/A
Default:	32G

`LAT_MEMORY_DIRECT`

Maximum JVM direct memory. This configuration is only supported as an Environment Variable or as Default when running the LAT through the provided package service file or through the provided `lat.sh` script. If directly running the Jar JVM heap memory should be set through command line arguments.

Environment Variable:	**LAT_MEMORY_DIRECT**
Property:	N/A
Default:	32G

Override Environment Variables

The recommended way to manage the production service configuration when LAT is running as a daemon is using environment variables or the `/etc/systemd/system/lat.service.d/override.conf` file.

To override LAT environment variables as a `systemd` service, there are two recommended commands that edit the `etc/systemd/system/lat.service.d/override.conf` file.

1. Run `sudo vim etc/systemd/system/lat.service.d/override.conf` or `sudo systemctl edit lat`.
2. Insert the environment variables to override in the file. Save and exit the editor.

Configuration File Example:

```shell
[Service]
Environment=LAT_MEMORY_DIRECT=200G
Environment=LAT_MEMORY=200G
```

Related Links

Ingest Data with Legacy LAT Reference

LAT Packaging and Installation

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Metrics

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

Configuration

LAT metrics are exposed using the LAT REST API.

Endpoints

LAT exposes two metrics endpoints; a ***partitions*** endpoint, and a ***pipeline*** endpoint.

Partition Metrics

The partitions endpoint exposes individual metrics for every partition that is participating in the current LAT instance. For Apache® Kafka® loads, partitions correspond to Kafka's partitions. For File Source based loads, partitions correspond to the independent file sets created within each File Group for parallel processing.

The partitions metrics endpoint can be accessed by issuing a curl request as follows:

CURL

```
curl http://127.0.0.1:8080/v2/metrics/read/lat:type=partitions
```

A response for this request will look as follows, with one entry per partition in the partitions array:

© 2026 Ocient, Inc.

JSON


```
{
    "request": {
        "mbean": "lat:type=partitions",
        "type": "read"
    },
    "value": {
        "partitions": [
            {
                "offsets_durable": 0,
                "pushes_errors": 0,
                "pushes_attempts": 0,
                "rows_pushed": 0,
                "offsets_written": 0,
                "records_buffered": 0,
                "records_errors_column": 0,
                "records_errors_deserialization": 0,
                "records_errors_transformation": 0,
                "offsets_processed": 0,
                "lag": 0,
                "partition": "topic-0",
                "records_filter_accepted": 0,
                "records_errors_row": 0,
                "records_filter_rejected": 0,
                "records_errors_generic": 0,
                "producer_send_attempts": 0,
                "offsets_pushed": 0,
                "pushes_unacknowledged": 0,
                "invalid_state": 0,
                "bytes_pushed": 0,
                "errors_partition": 0,
                "records_errors_total": 0,
                "offsets_buffered": 0,
                "complete": 0,
                "offsets_end": 0,
                "producer_send_errors": 0
            }
        ]
    },
    "timestamp": 1642497992,
    "status": 200
}
```


© 2026 Ocient, Inc.

Partition Metrics Definitions

Metrics definitions are as follows:

Metric	Description
`partition`	Topic partition pair (e.g., `mytopic-0`)
`records_errors_deserialization`	Total number of processed records which failed to deserialize
`records_errors_transformation`	Total number of processed records which failed to transform
`records_errors_column`	Total number of processed records which failed to bind transformed values to columns
`records_errors_row`	Total number of processed records which failed during builder row advancement/other builder errors
`records_errors_generic`	Total number of processed records which failed for any other reason
`records_errors_total`	Sum of all record errors for this topic
`records_buffered`	Total number of processed records which have not yet been pushed. Should always be modulo configured buffer size
`records_filter_rejected`	Number of records rejected by the topic filter.
`records_filter_accepted`	Number of records accepted by the topic filter.
`offsets_processed`	For Kafka loading, records are processed if their offsets are strictly less than the value of `offsets_processed`. For file loading, this represents the index of the most recently processed file. If there is an error during the processing of a record/file, this metric is still updated according to that record/file's offset. When a file load is complete, this offset becomes equal to offsets_end. Can decrease due to reprocessing.
`offsets_written`	For Kafka loading, records are written if their offsets are strictly less than the value of offsets_written. For file loading, this represents the index of the most recently written file. This metric is only updated when the record/file is processed and the write is successful. Note that a successful write does not mean the record/file is durable. Can decrease due to reprocessing.

© 2026 Ocient, Inc.

Metric	Description
`offsets_buffered`	For Kafka loading, this represents the most recently processed record which was processed to completion and is waiting to be pushed into the OcientAIQ™ Unified Data Platform. For file loading, this represents the index of the most recently processed file which was processed to completion and is waiting to be pushed into the Ocient data warehouse.
`offsets_pushed`	For Kafka loading, this represents the highest offset of the batch of rows most recently pushed to the Ocient data warehouse. For file loading, this represents the highest file index of the batch of rows most recently pushed to the Ocient data warehouse. Can decrease due to reprocessing.
`offsets_end`	For Kafka loading, this represents the end offset of the partition. For file loading, this represents the file count of the partition. Should never decrease.
`offsets_durable`	For Kafka loading, records are made durable if their offsets are strictly less than the value of offsets_durable. For file loading, this represents the most durable file index. Should never decrease.
`producer_send_attempts`	Number of Kafka error topic producer send attempts
`producer_send_errors`	Number of Kafka error topic producer send errors
`bytes_pushed`	Number of bytes pushed into the Ocient data warehouse
`rows_pushed`	Number of rows pushed into the Ocient data warehouse
`pushes_attempts`	Number of attempts to push record batches into the Ocient data warehouse for this partition
`pushes_errors`	Number of attempts to push record batches into the Ocient data warehouse which resulted in error
`pushes_unacknowledged`	Number of attempts to push record batches into the Ocient data warehouse for which no response has yet been received
`invalid_state`	Number of times a code path was reached in LAT which is erroneous
`complete`	Whether or not the partition has any records left to process at the moment; this status can change often in a Kafka load but will likely not change from complete to incomplete in a file load
`errors_partition`	The number of times the LAT failed to fetch records for a particular partition

Metric	Description
`lag`	Calculated as offsets_end - offsets_durable; In a Kafka load, lag represents the number of unprocessed records. In a file load, lag represents the number of unprocessed files. Lag becomes zero when the complete status is true.

Pipeline Metrics

The pipeline endpoint exposes aggregate metrics for the LAT instance. There are three categories of pipeline metrics:

Pipeline Metric Category	Description
`"pipeline" aggregate`	Monotonically-increasing metrics that act as lifetime counters for the running pipeline. They are independent of the current set of partitions. For instance, `rows_pushed` is the total number of rows ever pushed for this pipeline.
`"partitions" aggregate`	Metrics that depend on the current set of active partitions for the running pipeline. For instance, `total_lag` is the summed lag across all currently active partitions for the pipeline.
`pipeline-specific`	Pipeline only metrics that are not aggregates.

The Pipeline Metrics endpoint can be accessed by issuing a curl request as follows:

Shell

```
curl http://127.0.0.1:8080/v2/metrics/read/lat:type=pipeline
```

A response for this request will look as follows:

© 2026 Ocient, Inc.

JSON


```
{
    "request": {
        "mbean": "lat:type=pipeline",
        "type": "read"
    },
    "value": {
        "pipeline": {
            "pushes_errors": 0,
            "pushes_attempts": 0,
            "rows_pushed": 0,
            "records_errors_column": 0,
            "records_errors_deserialization": 0,
            "records_errors_transformation": 0,
            "bytes_pushed": 0,
            "errors_partition": 0,
            "records_filter_accepted": 0,
            "records_errors_total": 0,
            "records_errors_row": 0,
            "records_filter_rejected": 0,
            "records_errors_generic": 0,
            "producer_send_attempts": 0,
            "producer_send_errors": 0
        },
        "partitions": {
            "max_lag": 0,
            "total_offsets_buffered": 0,
            "avg_lag": 0.0,
            "total_pushes_unacknowledged": 0,
            "min_lag": 0,
            "total_lag": 0,
            "total_invalid_state": 0,
            "total_offsets_pushed": 0,
            "total_records_buffered": 0,
            "total_complete": 0,
            "total_offsets_processed": 0
        },
        "paused": 0,
        "bytes_buffered": 0,
        "complete": 0,
        "workers": 32,
        "lat_version": "3.0.0"
    },
```


© 2026 Ocient, Inc.

```
    "timestamp": 1642497956,
    "status": 200
}
```

Individual metrics such as `lag`, `offsets_buffered`, `pushes_errors`, etc. are defined within the **Partition Metrics Definitions** section.

However, metrics specific *only* to the pipeline endpoint are defined as follows:

Metric	Description
`paused`	0 if processing is active across all workers, 1 otherwise.
`bytes_buffered`	Global allocated memory in bytes.
`complete`	1 if the pipeline has finished the entire loading process, 0 otherwise.
`workers`	The number of active workers.
`lat_version`	The version of the running LAT server.

Related Links

Ingest Data with Legacy LAT Reference

System Information REST Endpoints

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Log4J Configuration

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

The LAT uses Apache® Log4J 2™ for **logging**. Users can configure their logging setup however they would like. The LAT packages come with example `log4j2.xml` files that can be used directly, or modified by the user. This XML code in the `log4j2.xml` file shows an example configuration.

© 2026 Ocient, Inc.

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" strict="true" monitorInterval="30" packages="biz.pal
  <Appenders>
    <RollingRandomAccessFile name="HttpLog" append="true" fileName="logs/http.log
      <PatternLayout pattern="%m%n" />
      <Policies>
        <SizeBasedTriggeringPolicy size="100 MB" />
      </Policies>
      <DefaultRolloverStrategy>
        <Delete basePath="logs">
          <IfFileName glob="http.*.log.gz" />
          <IfAccumulatedFileSize exceeds="5 GB" />
        </Delete>
      </DefaultRolloverStrategy>
    </RollingRandomAccessFile>
    <RollingRandomAccessFile name="LATLog" append="true" fileName="logs/lat.log"
      <PatternLayout pattern="%d{ISO8601} %-5p [%t] - %m%n" />
      <Policies>
        <SizeBasedTriggeringPolicy size="100 MB" />
      </Policies>
      <DefaultRolloverStrategy>
        <Delete basePath="logs">
          <IfFileName glob="lat.*.log.gz" />
          <IfAccumulatedFileSize exceeds="20 GB" />
        </Delete>
      </DefaultRolloverStrategy>
    </RollingRandomAccessFile>
    <RollingRandomAccessFile name="ErrorLog" append="true" fileName="logs/error.l
      <PatternLayout pattern="%m%n" />
      <Policies>
        <SizeBasedTriggeringPolicy size="100 MB" />
      </Policies>
      <DefaultRolloverStrategy fileIndex="nomax">
        <Delete basePath="logs">
          <IfFileName glob="error.*.log.gz" />
          <IfAccumulatedFileSize exceeds="20 GB" />
        </Delete>
      </DefaultRolloverStrategy>
    </RollingRandomAccessFile>
    <!--
    <Gelf name="Graylog" host="udp:graylog-dev" port="12201" version="1.1" extrac
```

```
      <Field name="process_name" literal="lat" />
      <Field name="log_level" pattern="%-5p" />
      <Field name="thread" pattern="%t" />
    </Gelf>
    -->
  </Appenders>
  <Loggers>
    <Logger name="org.apache.kafka" level="info" />
    <Logger name="com.ocient.lat.sink.ocient.Binders.Binder" level="info" />
    <!-- HttpChannel at debug so that jetty preserves the stacktrace when logging
    <Logger name="org.eclipse.jetty.server.HttpChannel" level="debug" />
    <Logger name="org.eclipse.jetty.server.RequestLog">
      <AppenderRef ref="HttpLog" />
    </Logger>
    <Logger name="com.ocient.lat.sink.file.FileErrorSink" level="info" additivity
      <AppenderRef ref="ErrorLog" />
      <!--
      <AppenderRef ref="Graylog" />
      -->
    </Logger>
    <Logger name="software.amazon.awssdk" level="info" />
    <Logger name="io.netty" level="info" />
    <Root level="info">
      <AppenderRef ref="LATLog" />
      <!--
      <AppenderRef ref="Graylog" />
      -->
    </Root>
  </Loggers>
</Configuration>
```

Relevant Loggers:

- `org.apache.kafka`: Apache® Kafka® logger.
- `com.ocient.lat.sink.ocient.Binders.Binder`: LAT binder logger. Can be verbose, recommended to keep at `info` level.
- `org.eclipse.jetty.server.HttpChannel` and `org.eclipse.jetty.server.RequestLog`: Jetty logs for the HTTP endpoint.
- `com.ocient.lat.Worker`: Main worker logger containing important pipeline logs.

Valid log levels are (in order of least to most verbose):

- `error`
- `warn`
- `info`
- `debug`
- `trace`

Log4j can be configured to log different packages at different levels as shown in the example. Here, the example logs the `com.ocient.streaming.client` package at the trace level, but the remainder of the code is logged at info level.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Packaging and Installation

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

The Ocient LAT supports multiple types of packages for deployment and installation:

- Debian® `.deb` package
- CentOS® `.rpm` package
- Raw `.tar.gz` tar folder
- Direct .jar
- Docker® container

Contact Ocient Support for the latest package for a deployment method.

Prerequisites

- Java® 17 or greater
  - Can either be system-level Java, or installed somewhere accessible by the user.

> ⚠ The LAT must run on Ocient Loader Nodes where you installed the Ocient RPM to provide access to numerical libraries, ensuring the correct loading of geospatial data types. If this condition is not met, some geospatial objects might load inaccurately.

Debian/CentOS Packages

1. Install/Upgrade

Debian: `sudo dpkg -i lat_$VERSION_all.deb`

CentOS: `sudo rpm -i lat-$VERSION-1.noarch.rpm`

2. Setup Java

© 2026 Ocient, Inc.

If the default system Java is version 17 or greater, no action is required. If not, edit the service file to contain a `JAVA_HOME` environment variable pointing to a Java version 17 or greater installation.

1. Run: `systemctl edit lat`
2. Insert the following

Shell

```
[Service]
Environment=JAVA_HOME="PATH_TO_JAVA_HOME"
```

3. Configure

Modify the **LAT Source Configuration** and **LAT Log4J Configuration** (see **Package Contents** for file locations).

> It is recommended to use the `.conf` file for most LAT configuration. However, `LAT_MEMORY` and `LAT_MEMORY_DIRECT` can only be set by using environment variable. For these use the same process as setting `JAVA_HOME` as detailed in step 2.

4. Start/Stop

Start: `systemctl start lat`

Stop: `systemctl stop lat`

Package Contents

- Jar: `/opt/lat/lat-$VERSION.jar`
- Service File: `/usr/lib/systemd/system/lat.service`
- Configuration: `/etc/lat/`
  - Server Configuration: `lat.conf`
  - Logging Configuration: `log4j2.xml`
- Logs: `/var/log/lat/lat.log`
- LAT data directory: `/opt/lat/.lat-data`

Tar Folder

1. Extract file in your preferred location

Run: `tar -xzf lat-$VERSION.tar.gz`

2. Setup Java

If the default system Java is version 17 or greater, no action is required. If not, set the `$JAVA_HOME` environment variable to a Java installation with version 17 or greater:

```
export JAVA_HOME="PATH_TO_JAVA_HOME"
```

3. Configure

Modify the **LAT Source Configuration** and **LAT Log4J Configuration** (see **Folder Contents** for file locations).

> It is recommended to use the `.conf` file for most LAT configuration. However, `LAT_MEMORY` and `LAT_MEMORY_DIRECT` can only be set using environment variable. For these use the same process as setting `JAVA_HOME` as detailed in step 2.

4. Start/Stop

Start: `./lat.sh [path]`, where `path` is an optional parameter for the path to a server configuration file (e.g., `lat.conf`). If `path` is not present, it will default to `config/lat.conf`.

Stop: Stop the running `lat.sh` script with `ctrl-c`.

Folder Contents

- Jar: `lat-$VERSION.jar`
- Configuration: `config/`
  - Server Configuration: `lat.conf`
  - Logging Configuration: `log4j2.xml`
- Start Script: `lat.sh`
- LAT data directory: `.lat-data`

© 2026 Ocient, Inc.

Jar Direct

The LAT is a Java jar that can be deployed on any machine running Java 17+.

To run the LAT use the following command line statement:

```Shell
java [options] -Dlog4j.configurationFile=<log4j-config-path> -jar <lat-jar-path> [l
```

- `log4j-config-path` : REQUIRED : path to the log4j configuration file
- `lat-jar-path` : REQUIRED : path to the LAT jar
- `lat-config-path` : OPTIONAL : path to a LAT service configuration file

Recommended Java Command Line Options

See the **Java command line documentation** for options.

Max Direct Memory Size

Option:	**-XX:MaxDirectMemorySize**
Recommended:	32G
Example:	-XX:MaxDirectMemorySize=32G

Minimum Heap Size

Option:	**-Xms**
Recommended:	32G
Example:	-Xms=32G

Maximum Heap Size

© 2026 Ocient, Inc.

Option:	-Xmx
Recommended:	32G
Example:	-Xmx=32G

Example Java Command Line

Shell

```
java \
  -Dlog4j.configurationFile=config/log4j2.xml \
  -XX:MaxDirectMemorySize=32G \
  -Xms32G \
  -Xmx32G \
  -jar lat-[VERSION NUMBER].jar \
  config/lat.conf
```

Docker Container

Ocient provides a Docker container for the LAT for testing workloads.

> The container is not recommended for production workloads and should be used for testing only.

Configuration

The only mode of configuring the LAT within a Docker container is by using the service configuration environment variables. A `lat.conf` file is not supported.

See **LAT Service Configuration** for the list of supported environment variables.

It is not recommended to override the following environment variables.

- `LAT_DATA_PATH`
- `LAT_API_PORT`

Logging

Within the Docker container the LAT logs `stdout`. The user can use the `docker logs ...` command to view the LAT logs.

The following environment variables are available to modify the log level of various loggers. See **LAT Log4J Configuration** for descriptions of the loggers and log levels.

`ROOT_LOG_LEVEL`

Logger:	**root**
Default:	debug

`BINDERS_LOG_LEVEL`

Logger:	**com.ocient.lat.sink.ocient.Binders.Binder**
Default:	info

`HTTP_LOG_LEVEL`

Logger:	**org.eclipse.jetty.server.HttpChannel**
Default:	debug

`KAFKA_LOG_LEVEL`

Logger:	**org.apache.kafka**
Default:	info

Java

The `JAVA_OPTS` environment variable is available for controlling JVM® settings. Container defaults to default JVM settings.

See the **Java command line documentation** for detail on the command line options.

Example:

```shell
JAVA_OPTS="-XX:MaxDirectMemorySize=32G -Xms32G -Xmx32G"
```

Data Volumes

The LAT in the docker container writes persistent data to `/lat/data`. It is recommended to mount a volume to this path so that the data can outlive the container instance. The **Error Log File** is also written to this path if configured. If viewing this error file is necessary, a volume must be used.

Do not use the `LAT_DATA_PATH` configuration to override the data path inside the docker container. It will cause the LAT to write data and error logs into different locations and will require two volumes.

If using the Local File Source, the user must mount the data directory as a bind mount inside the container. It is recommended to mount to a location under the `/lat/` directory. When configuring the local file source the bind mount path should be configured with a file group.

Ports

By default the LAT API listens on port `8080`. It is not recommended to override the `LAT_API_PORT` environment variable and instead use Docker's port mapping functionality.

Examples

Docker

Shell

```
docker run --rm --name lat \
  --env ROOT_LOG_LEVEL=info \
  --env JAVA_OPTS="-XX:MaxDirectMemorySize=32G -Xms32G -Xmx32G" \
  -p 8080:8080 \
  --mount source=lat-data,target=/lat/data \
  lat:2.0.0
```

Docker Compose

YAML

```
---
version: '3.9'
services:
  lat:
    image: lat:2.0.0
    ports:
      - "8080:8080"
    restart: on-failure
    volumes:
      - lat-data:/lat/data
    environment:
      JAVA_OPTS: "-XX:MaxDirectMemorySize=32G -Xms32G -Xmx32G"
      ROOT_LOG_LEVEL: info

volumes:
  lat-data:
```

Related Links

Install an Ocient System

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Advanced Topics

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

Addition of an LAT Instance to the Loading Path

Use these steps to add an LAT instance to the loading path:

1. Stop the LAT process on all Loader Nodes using `sudo systemctl stop lat`.
2. Copy `/opt/lat/.lat-data/pipeline/pipeline.json` from one of the Loader Nodes to the new node.
3. On the new node, execute this command: `sudo chown lat:lat /opt/lat/.lat-data/pipeline/pipeline.json`
4. On the new node, execute this command: `sudo chmod 644 /opt/lat/.lat-data/pipeline/pipeline.json`
5. Start the LAT process on all Loader Nodes using `sudo systemctl start lat`.

Removal of an LAT Instance from the Loading Path

Use these steps to remove an LAT instance from the loading path:

1. Stop the LAT process on all Loader Nodes using `sudo systemctl stop lat`.
2. Delete the `/opt/lat/.lat-data/pipeline/pipeline.json` from the chosen node.
3. Start the LAT process on all Loader Nodes using `sudo systemctl start lat`.
4. For file loading, rebalance the load across the remaining nodes. For details, see the `pipeline rebalance` command in **LAT Client Command Line Interface**.

For a load using Apache® Kafka®, shut down the service on the chosen node and delete the configuration.

© 2026 Ocient, Inc.

Dynamic Schema Changes

The loading system in Ocient is designed to support dynamic schema changes. The LAT allows data to load continuously even while database tables are altered (e.g., ADD COLUMN, DROP COLUMN). In this way dynamic changes can occur on the database and the pipeline can be updated later to begin streaming the revised data through the pipeline into the revised tables. The typical flow for updating a table is to first execute a DDL Command such as "`ALTER TABLE … ADD COLUMN …`" and then when it is completed to execute the update on the LAT Pipeline to include (or remove) the altered column in the transformation.

Error Sink

The LAT provides 2 ways to view loading, transformation, or binding errors that occur during a pipeline.

Kafka Error Topic

If a Kafka source is used, failed records can optionally be routed to a configurable **error_topic**. This topic will contain the records that fail and their failure reason/exception. The topic can be used to gain insight into why a given record failed to load. Each entry in the topic will contain the following:

- Value[byte[]]: A byte array containing the record itself.
- Headers:
  - topic: The topic from which the record originated.
  - partition: The partition from which the record originated.
  - offset: The partition offset from which the record originated.
  - state: The state of the record (the location where the record encountered an error).
  - exception: The exception associated with the error
  - exception_message: The exception message, if it exists.

Error Log File

If you use a File source, or if the Error Topic configuration is not set, the LAT sends errors to a dedicated error log file. If you use the default `log4j2` configuration, this error log can be found alongside the rest of the LAT logs in `error.log` and accessed using the LAT Client with the **Subcommands**.

If you use a custom `log4j2` configuration, your appender configuration should look similar to this code.

XML

```
<RollingRandomAccessFile name="ErrorLog" append="true" fileName="logs/error.log"
    <PatternLayout pattern="%m%n" />
    <Policies>
        <SizeBasedTriggeringPolicy size="100 MB" />
    </Policies>
    <DefaultRolloverStrategy fileIndex="nomax">
        <Delete basePath="logs">
            <IfFileName glob="error.*.log.gz" />
            <IfAccumulatedFileSize exceeds="20 GB" />
        </Delete>
    </DefaultRolloverStrategy>
</RollingRandomAccessFile>
```

If the **LAT_ALLOW_LOG_ORIGINAL_RECORDS** service configuration and the **log_original_records** pipeline configuration are both set to `true`, the error log file includes a JSON representation of the record that caused an error.

⚠ If you use a custom `log4j2` configuration, you must still use a `RollingRandomAccessFileAppender` named `ErrorLog`. Also, `PatternLayout` must still be `%m%n`, because the errors API can retrieve errors. If you do not use `RollingRandomAccessFileAppender`, the LAT does not start. If `PatternLayout` is not correct, the errors API does not work.

Understanding Deduplication

The LAT loads rows from data sources in an exactly-once fashion. This is made possible by row level deduplication for LAT pipelines and the ability to replay records from a source. In short, if the same records are replayed through the LAT, there are specific

scenarios that guarantee that no duplicate records will be persisted into the Ocient tables.

For a deeper understanding this works, a few key concepts are explained here:

- Partitioning Data
- The Durability Horizon
- Deduplication Scope

Partitioning Data

To deliver high throughput loading, the LAT partitions the data source into independent sets of data. These are then loaded in parallel across all LAT instances. Each partition is considered a well-ordered sequence of rows that is replayable.

Some sources like Kafka natively support the concept of partitions and have a native record ID as part of their protocol.

Others, like a batch of files from Amazon® Web Services™ (AWS™) S3, require LAT to partition the data on its own and assign a record ID. For file loading, LAT establishes a record ID based on the sorted list of files in the file group and the row of each record within the files.

⚠ Altering the list of files in the target directory on the source system can change the record ID. This will impair the ability of the LAT to properly deduplicate rows if a file loading pipeline is stopped and restarted.

The Durability Horizon

Within a partition, each record is assigned a unique record ID. This ID is monotonically increasing within a partition. As data is loaded through the LAT and into Ocient's page stores, this data is said to become "Durable" meaning that in the event of a node shutdown, the data would be preserved on non-volatile storage. At this point the record is no longer in memory, but stored on disk in a redundant fashion. The "Durability Horizon" is the largest record ID that has become Durable on each partition of data. If a previously loaded record were replayed it is recognized as a duplicate of the original record and ignored. New records are loaded and the Durability Horizon is increased.

Deduplication Scope

Deduplication is constrained based on a few settings in the LAT pipeline configuration. Each pipeline ID is considered an independent loading task. Additionally, each topic or file group is considered an independent data set. As a result, deduplication does not apply between different pipelines or different topics and file groups even if they are loading the same underlying files.

Records are deduplicated when all of the following are true:

1. The pipeline ID matches
2. The topic name or file group name matches

When this is true, any record with a record ID less than or equal to the current Durability Horizon will be considered a duplicate and ignored. When the record ID has progressed higher than the Durability Horizon, then new data will begin loading into the database.

If no `pipeline_id` is set, the LAT Client will automatically assign a randomly generated ID, resulting in no deduplication across different pipelines. In the event that you want deduplication between multiple pipelines, the `pipeline_id` should be copied from the previous pipeline and explicitly set. The topic/file groups must also be the same to ensure deduplication.

> ⚠ When updating an existing file load pipeline to select a different group of files by altering start/stop time or another filter, be sure to use a new pipeline ID (best accomplished by avoiding setting an explicit pipeline ID when creating the pipeline with the client). Otherwise, unexpected results can occur such as rows in a new file being considered a duplicate.

V1 → V2 Migration

LAT V2 modified the way the deduplication scope is calculated. In V1 only the topic name was used for scope calculation. Therefore, to maintain deduplication in a pipeline that is being upgraded from V1 to V2 **must be explicitly set to (the empty string)** upon creation.

© 2026 Ocient, Inc.

Related Links

Ingest Data with Legacy LAT Reference

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

LAT Client Command Line Interface

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

The LAT Client can be used to interact with a running LAT instance. It supports subcommands for interacting with pipelines, and previewing transformations.

The LAT Client is distributed in the form of a Python® wheel file. Contact Ocient Support for the wheel that corresponds to the LAT version.

Prerequisites

- Python >= 3.8
- pip3 >= 20.2.3
  - If `pip install` fails, try upgrading `pip`.
- Wheel Python package
  - `pip install wheel`

Install

It is recommended to install the wheel in a Python virtual environment to avoid conflicts with globally installed Python packages. For the install command, replace `$VERSION` with the latest version of the LAT client.

Steps:

1. Create the virtual environment. `python3 -m venv venv`
2. Activate the virtual environment. `source venv/bin/activate`
3. Install the wheel. `pip install lat_client-$VERSION-py3-none-any.whl`
4. Run commands with `lat_client COMMAND ARGS`
5. When a new terminal is opened, repeat step 2 to activate the environment and gain access to the `lat_client` command.

© 2026 Ocient, Inc.
Ocient Proprietary and Confidential

Usage

Get help on the command line:

```shell
lat_client --help
lat_client <subcommand> --help
```

Common Arguments

Some arguments are available on all subcommands. For convenience, most of them can also be set using an environment variable.

`--no-verify`

Skip certificate validation when connecting to LAT. Ignored if using http scheme

Example:	`--no-verify`

`--hosts (LAT_HOSTS)`

One or more LAT hosts to orchestrate. Valid domain names or IP addresses can be used.

Example:	`--hosts http://192.168.0.1:8080 http://192.168.0.2:8081`
Environment:	`export LAT_HOSTS="http://10.4.0.1:8080,http://10.4.0.2:8081"`

`--oauth-domain (LAT_OAUTH_DOMAIN)`

OKTA® OAuth domain to use for token acquisition.

Example:	`--oauth-domain https://dev-12345678.okta.com`
Environment:	`export LAT_OAUTH_DOMAIN="https://dev-12345678.okta.com"`

`--oauth-server (LAT_OAUTH_SERVER)`

Okta OAuth authorization server to use for token acquisition.

© 2026 Ocient, Inc.

Example:	`--oauth-server abcdef000ghijklm111`
Environment:	`export LAT_OAUTH_SERVER="abcdef000ghijklm111"`

`--client-id (LAT_CLIENT_ID)`

Okta client id to use for token acquisition.

Example:	`--client-id 12345678`
Environment:	`export LAT_CLIENT_ID="12345678"`

`--client-secret (LAT_CLIENT_SECRET)`

Okta client secret to use for token acquisition.

Example:	`--client-secret abc123`
Environment:	`export LAT_CLIENT_SECRET="abc123"`

`--oauth-http-proxy (LAT_OAUTH_HTTP_PROXY)`

HTTP proxy URL to use for token acquisition. Authentication credentials can be passed in proxy URL.

Example:	`--oauth-http-proxy http://user:pass@some.proxy.com`
Environment:	`export LAT_OAUTH_HTTP_PROXY="http://user:pass@some.proxy.com"`

Subcommands

`pipeline create`

Create a new pipeline.

For most use cases, it is advisable to leave the `pipeline_id` unset when creating a pipeline. The client will set it to a random UUID to prevent deduplication across different pipelines. In the event that you want deduplication between pipelines, the `pipeline_id`

should be copied from the previous pipeline and included in the new pipeline. The **LAT Transform Configuration** must also be the same to ensure deduplication.

When the client is used to create a pipeline with a file source, the client will make adjustments to the source configuration such that partitions are assigned evenly across nodes.

First, the client will get the number of workers from the pipeline. If one is not set, it will use the minimum configured `lat.default.workers` instead. Then, it will set `partitions = workers * num_nodes`. Finally, it will set `partitions_assigned = [workers * node_index, workers * (node_index + 1) - 1]` for each node.

Arguments:	`--pipeline` **: path to the pipeline configuration .json file**
Example:	`lat_client pipeline create --pipeline /home/user/my_new_pipeline.json`

`pipeline get`

Get the configuration for an existing pipeline.

If the configured pipelines are identical, print the pipeline, otherwise an explanation of the inconsistency will be provided.

For pipelines with a file source, `partitions_assigned` is ignored when checking if pipelines are identical. Additionally, the client will validate that all partitions are assigned, and that no partition is assigned more than once.

Example:	`lat_client pipeline get`

`pipeline update`

Update the configuration for an existing pipeline.

The new pipeline can only make changes to subfields in `transform`, *except* for any `topic` / `file_group` names. All other subfields of `transform` are allowed to change, including to the `table` and `column` fields.

For pipelines with a file source, partition assignments will be copied from the existing pipeline.

If the pipeline was running prior to the update, successful completion of this command will automatically restart the pipeline.

Arguments:	`--pipeline` **: path to the pipeline configuration .json file**
Example:	`lat_client pipeline update --pipeline /home/user/my_new_pipeline.json`

`pipeline delete`

Delete an existing pipeline.

Unless the `--force` flag is used, a pipeline must be stopped, or deletion will fail.

Arguments:	**-** `--force` **: Force a pipeline to delete regardless of running status** **-** `--skip-validation` **: Delete a pipeline regardless of cluster consistency**
Example:	lat_client pipeline delete —skip-validation

`pipeline start`

Start the configured pipeline.

Prior to starting the pipeline, the pipeline start subcommand will validate that all specified hosts are configured with an identical pipeline, except for pipelines with a file source, which must have different `partitions_assigned` such that each partition is assigned exactly once across all hosts.

Example:	`lat_client pipeline start`

`pipeline stop`

Stop the configured pipeline.

Example:	`lat_client pipeline stop`

`pipeline status`

Retrieve status of the pipeline. The valid pipeline statuses are `STOPPED`, `RUNNING`, `COMPLETED`, and `FAILED`. When the pipeline is `FAILED`, the file statuses will remain in processing.

Arguments:	**`--list-files`: Lists selected files in their sorted order for each file group, along with file statuses (`completed`, `processing`, `not_started`). Output is summarized to be human readable. The system truncates large file lists.**
Arguments:	`--list-all-files`: Lists all selected files in their sorted order for each file group, along with file statuses (`completed`, `processing`, `not_started`).
Example:	`lat_client pipeline status`

`pipeline errors`

Retrieve errors that occur while the current pipeline runs.

Arguments:	**- `--json`: output errors as lines of JSON rather than in the default human-readable format** **- `--max-errors` MAX_ERRORS: an upper limit on the number of errors to retrieve (default is 100)** **- `--only-records`: only show records (not error messages or other information)** **- `--only-error-messages`: only show error messages (not records or other information)** **- `--no-records`: show all information except records**
Example:	`lat_client pipeline errors --no-records --max-errors 10`

`pipeline rebalance`

Rebalances partitions evenly to all provided LAT Nodes.

> This subcommand only applies to pipelines running file sources.

This command is meant to be used in the case of a node outage during a file load. LAT file loading does not support automatic partition re-balancing and manual intervention is required. The flow is as follows:

1. LAT Node goes offline.
2. LAT operator rebalances the partitions from the offline node onto the online nodes using the client. The operator should use the rebalance command and omit the offline node from the hosts argument.
3. LAT Node comes back online.
4. LAT operator rebalances partitions using the client to include all nodes including the newly online node. The operator should use the rebalance command and include all online nodes in the hosts argument.

Example: `lat_client pipeline rebalance`

`sink create`

Create a new sink configuration.

> The sink configuration file for this subcommand should match the same format as the **LAT Sink Configuration**. For example:
>
> Text
>
> ```
> {  "type": "ocient",  "remotes": ["1.2.3.4:5050"]}
> ```

Arguments:

- `--sink` **: path to the sink configuration .json file**
- `--name` **: Name of the sink to create**
- `--default` **: set this as the default sink**

Example: `lat_client sink create --sink /home/user/my_sink_config.json --name my-sink-name-1 --default`

`sink delete`

Delete a sink configuration. The sink configuration must not be part of a created or running pipeline.

Arguments:	`--name` **: Name of the sink to delete**
Example:	`lat_client sink delete --name my-sink-name-1`

`sink list`

List all configured sinks.

Example:	`lat_client sink list`

`sink get`

Get a sink configuration by id.

Arguments:	`--name` **: Name of the sink to get configuration for.**
Example:	`lat_client sink get --name my-sink-name-1`

`preview`

Preview a transformation.

At most one of `--transform` or `--pipeline` can be provided. If neither is provided, the host will attempt to use the transformation configured in its pipeline.

You can specify the `--extract` or `--pipeline` option. If you specify none of these options, the host uses the JSON Extractor by default. For details about record and extractor types, see the **LAT Extract Configuration**.

You must specify the `--topic` or `--file-group` option, which should match the `topic` or `file_group` key in the specified `transform` section.

© 2026 Ocient, Inc.

Arguments:	**- `--topic` name of the topic the records are associated with - `--file-group` name of the file group the records are associated with - `--records` path to a file of records to transform. Record formats can be of Delimited Records (e.g., CSV, TSV), JSON Records, or Fixed Width Binary Records. - `--extract` [Optional] path to a .json file containing the extract section of a pipeline definition to use for extraction. - `--transform` [Optional] path to a .json file containing the transform section of a pipeline definition to use for transformation. - `--pipeline` [Optional] path to a .json file containing a pipeline to use for transformation and extraction, if present.**
Example without pipeline:	`lat_client preview --topic test_topic --records ./data/my_records --extract /home/user/my_extract.json --transform /home/user/my_transform.json`
Example using pipeline:	`lat_client preview --topic test_topic --records ./data/my_records --pipeline /home/user/my_pipeline.json`

Common Workflows

Check on Status of LAT Pipelines

Shell

```
$ lat_client pipeline status
10.4.0.1:8080: STOPPED
10.4.0.2:8080: STOPPED
10.4.0.3:8080: STOPPED

$ lat_client pipeline status --hosts https://10.4.0.1:8080
10.4.0.1:8080: STOPPED
```

Updating an Existing LAT Pipeline

Shell

```
lat_client pipeline update --pipeline /home/user/my_new_pipeline.json
```

If the pipeline was running prior to the update, successful completion of this command will automatically restart the pipeline.

If unsuccessful, the CLI will report an error with an explanation of what is wrong with the command. Common issues are invalid JSON, missing a required column. An unsuccessful update of the pipeline config does not impact actively running pipelines.

Restart the Pipeline

Shell

```
## First, stop the running pipelines. To send custom hosts, use --hosts IP1:port IP
$ lat_client pipeline stop
Stopped: 10.4.0.1:8080, 10.4.0.2:8080, 10.4.0.3:8080

## Then, start the pipelines.
$ lat_client pipeline start
Started:
10.4.0.1:8080, 10.4.0.2:8080, 10.4.0.3:8080
```

Check Multiple LAT Nodes to See If the Pipeline Configurations Are Compatible

Shell

```
## hosts with compatible pipelines
$ lat_client pipeline get --hosts https://10.4.0.1:8080 https://10.4.0.2:8080 https
{
    "pipeline_id": "91326229-5fc6-4542-99ce-87bdcb00a978",
    "version": 2,
    "source": { ... },
    "sink": { ... },
    "transform": { ... }
}

## hosts with incompatible pipeline
$ lat_client pipeline get --hosts https://10.4.0.1:8080 https://10.4.0.2:8080 https
Hosts responded with incompatible pipelines:
  b5a6e8647cd3a28d4a5bd07cd1c491ce: 10.4.0.1:8080, 10.4.0.2:8080
  160f270e39d7a51a431c841613e4dc4e: 10.4.0.3:8080
```

The CLI will compare the MD5 hash of the pipeline configurations on all nodes and respond that all pipelines match or are inconsistent.

© 2026 Ocient, Inc.

LAT Client Command Line Interface Troubleshooting

If the certificate authorities on the system running the Python client (LAT Client) need to be updated, an error can occur. This type of error message might appear.

Shell

```
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify f
```

You can run this command to resolve the error.

Shell

```
pip install -upgrade certifi
```

The root cause of this error can be a connection to either the LAT Server over SSL or an authentication to Okta to obtain an access token over SSL. You can run the same command in both cases to resolve the issue.

When the SSL certificate is self-signed on the LAT Server, you can use the `--no-verify` flag when you connect to the LAT Server without verifying the SSL Certificate.

Related Links

Ingest Data with Legacy LAT Reference

Install an Ocient System

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Install LAT Client in a Disconnected Network

> ⚠ Data Pipelines are now the preferred method for loading data into the Ocient® System. For details, see **Load Data**.

Operate the LAT Client in a disconnected network using these steps to install the LAT Client.

Requirements

On the Red Hat® Enterprise Linux® (RHEL®) version 8.0, the Python® version 3.6.8 specifies the default `python3` requirement and the LAT Client requires Python version 3.8 or greater.

Contact Ocient Support for the LAT Client Python wheel file.

Install the LAT Client Python wheel by first gathering the RPMs for Python 3.11 on a system with access to the appropriate software repository `python3.11` and its dependencies:

- `mpdecimal`
- `python3.11-libs`
- `python3.11-pip-wheel`
- `python3.11-setuptools-wheel`

Use your regular path for the Python packages.

Gather the `lat_client` and its dependencies (`.whl` files). Execute this download command on a machine with Python 3.11 and `pip` installed.

Shell

```
$ pip download lat_client-3.1.2-py3-none-any.whl
```

© 2026 Ocient, Inc.

This command downloads these dependencies.

```shell
lat_client
  aiohttp==3.9.3
    aiosignal==1.3.1
      frozenlist==1.4.1
    attrs==23.2.0
    frozenlist==1.4.1
    multidict==6.0.5
    yarl==1.9.4
      idna==3.6
      multidict==6.0.5
  more-itertools==10.2.0
  pydantic==2.6.4
    annotated-types==0.6.0
    pydantic_core==2.16.3
      typing_extensions==4.10.0
    typing_extensions==4.10.0
  rich==13.7.1
    markdown-it-py==3.0.0
      mdurl==0.1.2
    Pygments==2.17.2
  wheel==0.43.0
```

Copy the RPMs and wheel files to the machine in the disconnected network.

Install the LAT Client

Install Python 3.11 on the machine in the disconnected network where `<python3.11 RPM>` is the path to the Python 3.11 RPM. `<mpdecimal rpm>` is the path to a dependency of Python 3.11. You can list more dependencies in this way using spaces between each one.

```shell
dnf install <python3.11 RPM> <mpdecimal rpm>
```

After Python 3.11 is installed, create a `venv` to install the LAT Client and its dependencies.

Shell

```
$ /usr/bin/python3.11 -m venv venv
```

Enter the `venv`.

Shell

```
$ source venv/bin/activate
```

Install `lat_client` and its dependencies in the `venv`. Specify wheel files `<wheels>` as paths to wheel files separated by spaces.

Shell

```
(venv)$ pip install lat_client-3.1.2-py3-none-any.whl <wheels>
```

Check that the LAT Client runs.

Shell

```
(venv)$ lat_client --version
3.1.2
```

Related Links

LAT Client Command Line Interface

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Version Compatibility

Recommended JDBC Driver Versions

This table describes the recommended versions of the JDBC driver and Java® for each Ocient® System release.

Ocient System Version	JDBC Driver Version	Java Version
27.1	4.2.0	8 or later
27.0	4.1.0	8 or later
26.1	3.7.0 and 4.0.0	8 or later
25.5	3.6.7, 3.6.6, 3.6.5, and 3.6.4	8 or later
25.2	3.5.1	8 or later
25.1	3.4.1	8 or later
25.0	3.3.1	8 or later
24.3 and 24.4	3.2.x	11 or later
23.1, 24.0, 24.1, and 24.2	3.1.7	8 or later
23.0	3.0	8 or later

JDBC versions 3.2.x are not compatible with Java 8. Version 3.3.0 is compatible.

JDBC API Compatibility

This table describes the specifications for each version of the JDBC API and whether it is supported by the latest Ocient JDBC driver.

JDBC API Version	Requirement	Support as of JDBC 4.0.0
4.3	Enhanced DatabaseMetaData to determine if Sharding is supported	Supported

© 2026 Ocient, Inc.

JDBC API Version	Requirement	Support as of JDBC 4.0.0
4.3	Added the method drivers to DriverManager to return a Stream of the currently loaded and available JDBC drivers	Supported
4.3	Added support to Statement for enquoting literals and simple identifiers	Supported
4.3	Clarified the Java SE version that methods were deprecated	Supported
4.3	Added Sharding support	Not Supported
4.3	Enhanced Connection to be able to provide hints to the driver that a request, an independent unit of work, is beginning or ending	Not Supported
4.2	Added JDBCType enum and SQLType interface	Supported
4.2	Added support for large update counts	Supported
4.2	DatabaseMetaData methods to return the maximum Logical LOB size and if Ref Cursors are supported	Supported
4.2	Support for REF CURSORS in CallableStatement	Not Supported
4.1	Allow Connection, ResultSet and Statement objects to be used with the try-with-resources statement	Supported
4.1	Support added to CallableStatement and ResultSet to specify the Java type to convert to using the getObject method	Supported
4.1	Added support to Connection to specify a database schema, abort and timeout a physical connection.	Supported
4.1	DatabaseMetaData methods to return PseudoColumns and if a generated key is always returned	Not Supported
4.1	Added support to close a Statement object when its dependent objects have been closed	Not Supported
4.1	Support for obtaining the parent logger for a Driver, DataSource, ConnectionPoolDataSource and XADataSource	Not Supported
4.0	auto java.sql.Driver discovery — no longer need to load a java.sql.Driver class using Class.forName	Supported
4.0	National Character Set support added	Supported
4.0	Support added for the SQL:2003 XML data type	Not Supported

© 2026 Ocient, Inc.

JDBC API Version	Requirement	Support as of JDBC 4.0.0
4.0	SQLException enhancements — Added support for cause chaining; New SQLExceptions added for common SQLState class value codes	Not Supported
4.0	Enhanced Blob/Clob functionality — Support provided to create and free a Blob/Clob instance as well as additional methods added to improve accessibility	Supported
4.0	Support added for accessing a SQL ROWID	Not Supported
4.0	Support added to allow a JDBC application to access an instance of a JDBC resource that has been wrapped by a vendor, usually in an application server or connection pooling environment.	Supported
4.0	Availability to be notified when a PreparedStatement that is associated with a PooledConnection has been closed or the driver determines is invalid	Not Supported
3.0	Pooled statements — reuse of statements associated with a pooled connection	Supported
3.0	Savepoints — allow a transaction to be rolled back to a designated savepoint	Not Supported
3.0	Properties defined for ConnectionPoolDataSource — specify how connections are to be pooled	Not Supported
3.0	Metadata for parameters of a PreparedStatement object	Not Supported
3.0	Ability to retrieve values from automatically generated columns	Not Supported
3.0	Ability to have multiple ResultSet objects returned from CallableStatement objects open at the same time	Not Supported
3.0	Ability to identify parameters to CallableStatement objects by name as well as by index	Not Supported
3.0	ResultSet holdability — ability to specify whether cursors should be held open or closed at the end of a transaction	Not Supported
3.0	Ability to retrieve and update the SQL structured type instance that a Ref object references	Not Supported
3.0	Ability to programmatically update BLOB, CLOB, and ARRAY values.	Supported
3.0	Addition of the java.sql.Types.DATALINK data type — allows JDBC drivers access to objects stored outside a data source	Not Supported

© 2026 Ocient, Inc.

JDBC API Version	Requirement	Support as of JDBC 4.0.0
3.0	Addition of metadata for retrieving SQL type hierarchies	Not Supported
2.1	Scrollable result sets—using new methods in the ResultSet interface that allow the cursor to be moved to a particular row or to a position relative to its current position	Not Supported
2.1	Batch updates	Supported
2.1	Programmatic updates—using ResultSet updater methods	Not Supported
2.1	New data types—interfaces mapping the SQL3 data types	Supported
2.1	Custom mapping of user-defined types (UDTs)	Not Supported
2.1	Miscellaneous features, including performance hints, the use of character streams, full precision for java.math.BigDecimal values, additional security, and support for time zones in date, time, and timestamp values.	Supported
2.0	Pooled connections — allowing connections to be used and reused	Supported
2.0	Distributed transactions — allowing a transaction to span diverse DBMS servers	Not Supported
2.0	The DataSource interface as a means of making a connection. The Java Naming and Directory Interface™ (JNDI) is used for registering a DataSource object with a naming service and also for retrieving it.	Not Supported
2.0	RowSet technology — providing a convenient means of handling and passing data	Not Supported

Recommended LAT Versions

This table describes the recommended Ocient Loading and Transformation (LAT) versions for each Ocient System release.

Ocient System Version	Minimum Version	Recommended Version
27.1	3.0	3.6.2

LAT Version 3.0.0 and greater is only compatible with Version 22.0 and greater of the Ocient System due to a change in the calculations of empty bounding boxes on geospatial data types.

Related Links

Release Notes

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Release Notes

Release notes are available for the OcientAIQ™ Unified Data Platform and the associated clients, JDBC and pyocient. For the corresponding release notes, see these links.

Products	Release Notes
OcientAIQ Unified Data Platform	**OcientAIQ Unified Data Platform Release Notes**
JDBC	**JDBC Release Notes**
pyocient	**pyocient Release Notes**

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

OcientAIQ Unified Data Platform Release Notes

27.1

Release Highlights

- Data Extract Tool —
  - Added the extraction of Apache® Parquet™ files to the data extract tool. For details, see **Data Extract Tool**.
- Data Pipeline Loading —
  - Added the `CREATE TRANSACTIONAL PIPELINE` SQL statement with the optional `START FOREGROUND` clause. This statement creates a transactional pipeline, starts it immediately, and blocks the SQL session until the pipeline completes or fails. On failure, the Ocient® System automatically rolls back all uncommitted data. For details, see **Transactional Data Pipelines**.
- JDBC Bulk Loading —
  - Added the S3-compatible object store as an alternative bulk load transport mode. You can now set the `bulkLoadMode` connection parameter to `s3` to stage data through an S3-compatible endpoint instead of SSH or SFTP to Loader Nodes. For details, see **JDBC Bulk Loading** and **JDBC Connection Properties**.

Features

- Transaction Support —
  - The Ocient System now supports transactions in the database and in the JDBC and pyocient drivers. A transaction groups one or more SQL statements into a single unit of work that the database either commits or discards as a whole. Transactions let you apply a set of related changes together and roll them back if any statement fails. For details, see **Transactions**. For syntaxes, see **Transactional Control Language (TCL) Statement Reference**.
- Distributed Tasks —

© 2026 Ocient, Inc.

    - Added the `DROP TASK` SQL statement to drop orphaned or completed tasks from the system. For details, see **Distributed Tasks**.
- System Catalog Tables Additions —
    - `sys.pipeline_errors_historical`
    - `sys.pipeline_events_historical`
    - `sys.pipeline_files_historical`
    - `sys.pipeline_metrics_historical`
- System Catalog Column Additions —
    - `transaction_id` and `transaction_prefix_len` columns in both of the `sys.queries` and `sys.completed_queries` system catalog tables
    - `transaction_id` column in the `sys.pipelines` system catalog table
- Information Schema Views Additions —
    - `information_schema.pipeline_status_historical`
    - `information_schema.pipelines_historical`
    - `information_schema.transactional_pipeline_status`

Version Compatibility

- The `array_to_string` SQL function no longer adds quotes around values of VARCHAR arrays.

27.0

Release Highlights

- Data Pipeline Loading —
    - Added the new **ALTER PIPELINE** SQL statement for schema evolution. You can now modify the SELECT clause of a data pipeline for columns, filters, and transformations without recreating the pipeline.
- Machine Learning Model Updates —

- Added ensemble models with the Bagging, Boosting, and Stacking models. For details, see **Ensemble Models**.

Features

- Data Pipeline Loading —
  - Added the `line_number` metadata key for loading the line number of the file. For details, see **Load Metadata and File-Based Partitioned Data in Data Pipelines**.
  - Added these extract options for delimited formats and CSV:
    - STRIP_FIELD_QUOTES
    - STRIP_ARRAY_ELEMENT_QUOTES
    - TRIM_ARRAY_ELEMENTS
    - For details, see **Delimited and CSV Extract Options**.
- Machine Learning Model Updates —
  - Added the regression version of the Decision Tree model. For details, see **Regression Tree**.
  - Added `featureSubsetStrategy` and `maxCellsToFetch` options to the Random Forest model. For details, see **Random Forest**.
- Data Storage —
  - Added the `DRAIN PAGES` SQL statement to manually migrate data from pages to segments. For details, see the **DRAIN PAGES** statement.
  - Added the `NOINDEX` option for time bucket clauses in `CREATE TABLE` statements. This option prevents a TimeKey® from impacting storage ordering. For details, see **TimeKeys and Clustering Keys**.
  - Added the `ALTER SYSTEM SET DEFAULT STORAGESPACE` SQL statement to enable default storage spaces for tables. For details, see the **ALTER SYSTEM SET DEFAULT STORAGESPACE** statement.
  - Added the `ALTER CLUSTER ADD STORAGESPACE` and `ALTER CLUSTER REMOVE STORAGESPACE` SQL statements to link storage spaces to clusters. For details, see the **ALTER CLUSTER ADD STORAGESPACE** and **ALTER CLUSTER REMOVE STORAGESPACE** statements.

- CREATE TASK syntax —
  - Added the `LOCATION` and `OPTIONS` keywords to the `CREATE TASK` SQL statement for easier task creation. For details, see **Distributed Tasks**.
- Aggregate Functions —
  - Added these functions to assess the performance of machine learning models:
    - `ACCURACY_SCORE`
    - `COEFFICIENT_OF_DETERMINATION`
    - `CONFUSION_MATRIX`
    - `F1_SCORE`
    - `MEAN_ABSOLUTE_ERROR`
    - `MEAN_ABSOLUTE_PERCENTAGE_ERROR`
    - `PRECISION_SCORE`
    - `RECALL_SCORE`
    - `ROC_AUC_SCORE`
    - For details, see **Aggregate Functions**.
- Scalar Data Conversion Functions —
  - Updated the `BOOLEAN` function to accept new string and numeric values for casting. For details, see **Scalar Data Conversion Functions**.
- Default Expressions —
  - Expanded valid column defaults from exclusively literals to any constant, deterministic expression that the system casts to the target column. For details, see **Column Constraint**.
- Values Expressions
  - You can now use value expressions with the `VALUES` keyword in the `FROM` clause. For details, see the **FROM** clause syntax.
- System Catalog Column Additions —
  - `target_rows_per_cycle` and `applied_dynamic_filter_stats` in the `sys.active_operator_instances` system catalog table

- `default_schema` in the `sys.completed_queries` system catalog table
- `begin_timestamp` and `end_timestamp` columns in the `sys.segments` system catalog table
- `begin_timestamp` and `end_timestamp` columns in the `sys.segment_groups` system catalog table
- `scope_prefix_len`, `transaction_timeout`, `creation_timestamp`, and `last_activity_timestamp` in the `sys.storage_scopes` system catalog table
- `default_schema` in the `sys.queries` system catalog table
- `password_days_remaining` column in the `sys.users` system catalog table
- For details, see **System Catalog**.

Version Compatibility

- The Ocient® System now supports non-literal expressions (e.g., arithmetic and deterministic functions) with the `DEFAULT` keyword to specify a default value. The system validates default values and ensures consistent casting. For example, you must now specify a TIMESTAMP default value as `TIMESTAMP DEFAULT 0` or `TIMESTAMP DEFAULT '1970-01-01 00:00:00.00000000'`.

26.1

Release Highlights

- Data Pipeline Loading —
  - The Ocient® System now loads data in the Apache® Avro™ format. For details, see **Load Avro Data**. The new `FORMAT avro` extract options are in the syntax of the **CREATE PIPELINE** SQL statement.
- Array Functions —
  - Added the **ARRAY_CAT_DISTINCT** and **ARRAY_DISTINCT** array functions for concatenation and deduplication, respectively.
  - Added the **ARRAY_CONCAT_AGG** sorted aggregate function for concatenation.
- Index Types —

    - Added the `ZONE_MAP` index type to store the range of values present in a segment for the indexed column, enabling queries to skip over entire segments with no rows matching a compatible filter. For details, see the **ZONE_MAP Index Type**.
- HTTP Query API —
    - Added the **Ocient HTTP Query API** endpoint library to enable execution of SQL statements.
- Google® Looker™ Studio Integration —
    - Added the connector for integrating the Ocient System with Looker Studio. For details, see **Looker Connector**.

Features

- Data Pipeline Loading Changes —
    - The Ocient System now loads data from Apache® Hadoop® Distributed File System (HDFS) sources. The new source options are in the syntax of the **CREATE PIPELINE** SQL statement. For details, see **Data Pipeline Load of JSON Data from HDFS**.
    - The Ocient System can now load data from HDFS. For the source syntax and options, see the **CREATE PIPELINE** SQL statement. For the tutorial, see **Data Pipeline Load of JSON Data from HDFS**.
    - The data pipeline loading functionality supports the `TO_CHAR` function for converting `VARBINARY` types. For details, see **Character Data Transformation Functions**.
    - Updated the `METADATA` function with:
        - New `key` metadata key to specify the Apache® Kafka® bytes of the Kafka key for the record. For details, see **Load Metadata and File-Based Partitioned Data in Data Pipelines**.
        - New `METADATA('hive_partition',search_string)` syntax for retrieving a value from a key-value pair in the filename metadata using Apache® Hive™ naming standards with a filter set in the file path.
    - Added the `NUM_FOOTER_LINES` option for delimited and CSV formats to skip lines for the footer of the file. For details, see **Delimited and CSV Extract**

Options.

    - The file-based source `START_TIMESTAMP` and `END_TIMESTAMP` options have been removed for data pipelines.

- General SQL Syntax —
    - Added the `TAG` keyword to add a tag to the SQL query. For details, see the **TAG** syntax.
- Table Retention Policies —
    - Added functionality to add retention policies to table definition statements. For details, see **Table Retention Policies**.
- Schema Database Object —
    - Added the schema database object and its corresponding privileges. For details, see **Data Control Language (DCL) Statement Reference**.
    - You can create, modify, and remove a schema by using the `CREATE SCHEMA`, `ALTER SCHEMA RENAME`, and `DROP SCHEMA` SQL statements, respectively. For details, see **Schemas**.
- Cross-Database SSO Authentication —
    - Enabled SSO-authenticated users to access multiple databases using a single identity provider configuration. For details, see **Set Up Cross-Database SSO Integration**.
    - Added the **CURRENT_GROUPS** function that returns the fully qualified names of groups in the database.
- Query Privileges —
    - Added the VIEW QUERIES, VIEW REDACTED QUERIES, and CANCEL QUERIES privileges in the database. For details, see **Data Control Language (DCL) Statement Reference**.
- System and Database Privileges —
    - Added the DELETE privilege to the system and database objects. For details, see **Data Control Language (DCL) Statement Reference**.
- Machine Learning Model Updates —

- Added `featureArray` option to all models except Simple Linear Regression, Vector Autoregression, Association Rules, and Feedforward Neural Network models.
- Added the `normalize` option.
- Added new options `bootstrap` and `maxChildThreads` to the Random Forest model.
- Added new options `learningRate`, `numEpochs`, `miniBatchSize`, `gradientClipThreshold`, and `finiteDifferenceH`. These options replace `popSize`, `initialIterations`, `subsequentIterations`, `momentum`, `gravity`, `lossFuncNumSamples`, `numGAAttempts`, `maxLineSearchIterations`, `minLineSearchStepSize`, and `consistentSample` options for Nonlinear Regression, Logistic Regression, Support Vector Machine, and Feedforward Neural Network models.
- Added new models **Gradient Boosted Trees** and **Regression Tree**.

- Data Type Updates —
  - Array and tuples without specified data types, such as `ARRAY[]`, `ARRAY[NULL]`, and `TUPLE(NULL, NULL)`, are now valid expressions. The Ocient System treats these types as `CHAR[]` for arrays and `TUPLE<<CHAR,CHAR>>(NULL, NULL)` for tuples.
- System Catalog Table Changes —
  - The `sys.pipeline_files` table no longer contains the `stream_source_id` column. The table now contains the `file_index` column.
  - The `sys.columns` table has these new columns:
    - `system` — Boolean value that specifies whether the column is internal
    - `type_hint` — expected type of the column
  - With k-nearest neighbors models, the `machine_learning_models` table returns the `K_NEAREST_NEIGHBORS` value for the `machine_learning_model_type` column.
  - All authenticated users can now view these tables:
    - `sys.nodes`
    - `sys.service_roles`
    - `sys.node_status`

    - `sys.channel_endpoint_parameters`
  - The `encrypted` column has been removed from the `sys.channel_endpoint_parameters` table.

Version Compatibility

- Decimal values use rounding instead of truncation when you cast types from DECIMAL or STRING to a lower precision and scale.
- The system automatically grants the DELETE privilege to the creator of database and schema objects for objects created in version 26.1 and later.

26.0

Internal updates only.

25.5

Version Compatibility

- The `||` operator behavior has changed. If you specify a NULL value for concatenation, then the result is NULL.

25.2

Features

- Data Pipeline Changes —
  - Added the `WHERE` filter keyword to the `SELECT` SQL statement in the `CREATE PIPELINE` statement. For details, see **CREATE PIPELINE**.
  - Added new `HEADERS` Amazon® Web Services℠ (AWS℠) S3 source option for the `CREATE PIPELINE` SQL statement to define the headers to send with every request.
  - Added the `MODE` option to the `PREVIEW PIPELINE` SQL statement. This option indicates whether to perform a validation of the `PREVIEW PIPELINE` statement. For details, see **PREVIEW PIPELINE**.

- Added the `streamloader.extractorEngineParameters.configurationOption.filesystem.access.directories` configuration setting to denote where you can load server file system data. For details, see **Configuration Settings for Data Pipelines**.
- Updated the `METADATA` function with a new metadata key and syntax. For details, see **Load Metadata and File-Based Partitioned Data in Data Pipelines**:
  - The `source_record_id` metadata key indicates a unique source row identifier that is generated by the Ocient System during loading.
  - New `METADATA('hive_partition',search_string)` syntax that uses a search string as a key for searching the filename metadata to return the value from the key-value pair as a string.
- Added these transformation functions. For details, see **Transform Data in Data Pipelines**:
  - `IF` — Evaluate an expression for `true` or `false`.
  - `CASE WHEN` — Return a result value based on whether an expression is `true`.
  - `ELEMENT_AT` — Returns the element value of the array or tuple at the specified index.
  - `MAP_KEYS` — Returns the keys in the specified JSON string.
  - `MAP_VALUES` — Returns the values in the specified JSON string.
  - `REDUCE` — Applies a merge function to a starting value and all elements in the array, and then reduces the array to a single value.
- System Catalog Table Changes —
  - Added Boolean parameter `scopes_on_refresh` to the `sys.oidc_integrations` system catalog table to include or not include scopes when submitting a refresh token request.

25.1

Features

- New data pipeline transformation function `ARRAY_CONTAINS` determines whether an array contains the specified value, including NULLs.
- System Catalog Table Changes —
  - The `record_number` column of the `sys.pipeline_errors` system catalog table is now the index, which starts at 1, of the processed record relative to the source file (or Apache® Kafka® partition) where the record was extracted.
  - The `record_offset` column of the `sys.pipeline_errors` is now the offset of 1 for the processed record relative to the start of the source file for files with contiguous records, -1 for files with non-contiguous records (i.e., Apache® Parquet™), and the Kafka offset for Kafka loads.

25.0

Release Highlights

- Load Data into the Ocient® System — The new way to load data into the Ocient System is to use Data Pipeline functionality. This functionality supports loading data from multiple formats such as CSV, JSON, and Apache® Parquet™. Data pipelines support data transformation during the load. You can load multiple data formats, including geospatial data. This functionality supports these new syntaxes:
  - `CREATE PIPELINE` SQL statement to create a data pipeline.
  - `DROP PIPELINE` statement to remove a data pipeline.
  - `PREVIEW PIPELINE` statement to view the results of a data load before creating the data pipeline.
  - `START PIPELINE` statement to start the execution of the data load.
  - `STOP PIPELINE` statement to stop the execution of the data load.
  - `ALTER PIPELINE RENAME` statement to rename a data pipeline.
  - `EXPORT PIPELINE` statement to return the `CREATE PIPELINE` SQL statement used to create the specified data pipeline.
  - `CREATE OR REPLACE PIPELINE FUNCTION` statement to define your own data pipeline function for loading data.
  - `DROP PIPELINE FUNCTION` statement to remove the data pipeline function.

- For details, see **Load Data** and **Data Pipelines**.

Features

- Java® Runtime Environment — The Ocient System installation requires Java® 21, where the `openjdk-21-jre-headless` package is recommended.
- Persistent Data — Data persists in the system storage space of the Ocient System in system catalog tables such as the `sys.completed_queries` table. For details, see **Storage Space Settings**.
- Default Compression Scheme — The default compression scheme for variable-length columns is `NONE`.
- Set more granular privileges for database and system objects. For details, see **Object-Type Level Privileges Management** and **Data Control Language (DCL) Statement Reference**.
- The `SHOW SYSTEM TABLES` SQL statement displays the system catalog tables.
- System Catalog Table Changes —
  - `information_schema.tables` has a column `table_type` with two new types: `SYSTEM TABLE` for system catalog tables and `SYSTEM VIEW` for information schema tables. `BASE TABLE` and `VIEW` still exist for user-defined tables and views.
  - `sys.system_tables` has new columns: `schema`, `type`, and `description`.
  - `sys.segment_groups` has the new `depth` and `visibility` columns.
  - `sys.stored_segments` has the new `abnormal_placement` column.
  - `sys.completed_queries` contains persistent data. The system experiences a small loading delay between completing the execution of a SQL query and when the corresponding results appear in the table.
  - The definition for the `sys.queries` table has been updated to remove these columns:
    - `time_start`
    - `time_optimization_start`
    - `time_execution_start`
    - `time_first_byte_sent`

- The definition for the `sys.queries` table has been updated to change the data type of these columns from `LONG` to `TIMESTAMP`:
  - `timestamp_start`
  - `timestamp_optimization_start`
  - `timestamp_execution_start`
  - `timestamp_first_byte_sent`
- The definition for the `sys.locks` table has been updated for these columns:
  - Renamed `createTime` to `create_time`.
  - Renamed `lastRefreshTime` to `last_refresh_time`.
  - Renamed `priortyId` to `priority_id`.
- Updated column descriptions for these tables:
  - `sys.segment_directories`
  - `sys.addendum_directories`
  - `sys.segment_groups`
  - `sys.segment_parts`
  - `sys.segments`
  - `sys.stored_segments`
  - `sys.service_role_status`
- Updated the data type for the `segment_group_ids` column in the `sys.segment_group_transfers` table from `ARRAY(LONG)` to `ARRAY(CHAR)` because segment group identifiers are unsigned.
- Renamed the `service_roles_id` column in the `sys.service_role_channel_endpoint` table to `service_role_id` to align with the naming in the `sys.service_role_status` table.

Version Compatibility

- `sys.built_in_views` has been removed. The built-in views have been removed from `sys.built_in_views`, `sys.views`, and `information_schema.views`. The view information now appears in the `sys.system_tables` system catalog table.

`information_schema.views` includes user-defined views and only information schema views.

- `sys.views` no longer has a `view_type` column.

24.0

Release Highlights

- Better SQL Errors: Improved error messages for poorly constructed SQL queries.
- Enhance Build Metadata: Updated Ocient® package file names and build information for clarity.
- Improved Tracing: Enabled using the `TRACE` keyword in SQL queries to profile query performance. For details, see the **TRACE** keyword.
- Large Blobstore Support: Enabled more than 2TiB of data to spill per disk for high-density drive situations.
- Large Drive Support: Added support for drives up to 15.36TB in size per node.
- Rebalance System: Added the `REBALANCE` task, which enables optimization of query efficiency by transferring data around the system until nodes are roughly balanced in terms of data volume per node. For details, see **Rebalance System**.
- System Storage Space: Added support for multiple storage spaces and enabled the creation of a system storage space for internal Ocient System data.
- Workload Management Usability:
  - Added support for assigning service classes to queries based on query text. For details, see the **CREATE SERVICE CLASS** SQL statement.
  - Added support for changing the priority for queries. For details, see the **ALTER QUERY** SQL statement.

Features

- [DB-19266]: Network Configuration —
  - All nodes must now belong to a connectivity pool. Manage connectivity pools using these new SQL statements. For details, see **CONNECTIVITY POOL**.
    - `CREATE CONNECTIVITY_POOL` to create a connectivity pool.

    - `DROP CONNECTIVITY_POOL` to drop a connectivity pool.
    - `ALTER CONNECTIVITY_POOL SET` to set the metadata of a connectivity pool.
    - `ALTER CONNECTIVITY_POOL RENAME TO` to rename a connectivity pool.
    - `ALTER CONNECTIVITY_POOL ADD PARTICIPANTS` to add nodes to a connectivity pool.
    - `ALTER CONNECTIVITY_POOL DROP PARTICIPANTS` to remove nodes from a connectivity pool.
  - The `ALTER NODE SET ADDRESS` SQL statement changes the internal IP address for a node. For details, see **ALTER NODE SET ADDRESS**.
  - You can now configure the network of an Ocient System. For details, see **Manage the Network Configuration of an Ocient System**.
  - Redirects now occur only within connectivity pools. When you upgrade an Ocient System, you must first configure a connectivity pool.
- [DB-28986]: System Catalog Table Updates —
  - Renamed `client_version` to `protocol_version` in the `sys.queries` and `sys.completed_queries` system catalog tables.
  - Added `driver_version` column to the `sys.queries` and `sys.completed_queries` tables.
- [DB-29788]: Regular Expression Functions —
  - Added new functions that use regular expression search patterns. The new functions are:
    - **REGEXP_COUNT**
    - **REGEXP_INSTR**
    - **REGEXP_REPLACE**
    - **REGEXP_SUBSTR**

Version Compatibility

- The `bootstrap.conf` no longer supports `highspeedAddress` as an advanced system configuration option. For other options, see **Node Bootstrapping Reference**.

- The default behavior for the `UNNEST` function no longer uses the `NULL_INPUT` clause for a multi-item `SELECT` list. If you would like to utilize the default behavior from version 23.0 and prior, you may do so by changing `ALTER SYSTEM ALTER CONFIG SET sql.unnestLegacySelectListBehavior = 'true'`.

Feature Removal

- Removed `COMPRESSION LZ4` from the compression options. This compression scheme remains in use as part of `COMPRESSION DYNAMIC` for variable-length columns.
- Removed the table-valued function `REPLACEMENT_JOIN`. For creating compressed lookup tables, see **Global Dictionary Compression**.
- Particle swarm optimization functionality has been removed from the Ocient System.
- ODBC connection has been removed from the Ocient System.

23.0

Release Highlights

- All machine learning functionality is available to use. For details, see **Machine Learning Model Functions** and **Machine Learning in Ocient** to get started.
- Delete Syntax: Enabled the deletion of individual rows in the database.
- Integrations: Added drivers and support for the following third-party applications:
  - DBeaver
  - Tableau®

Features

- [DB-13607]: Delete Syntax — Added the SQL DELETE statement syntax that enables the deletion of individual rows in the database.
- [DB-18020]: Large Geospatial Types — Increased the size of `LINESTRING` and `POLYGON` geospatial data types to 512 MB. For details, see **Load Geospatial Data**.

- [DB-19048]: Geospatial Index — Added the SPATIAL index type for indexing geospatial data. For details, see **SPATIAL Index Type**.
- [DB-18280]: Connectors Refresh — Added integration with DBeaver and Tableau®. For details, see **DBeaver Integration** and **Tableau Integration**.
- [DB-20412]: Multi-Cluster Loading and Cluster of Clusters — Added support for loading and working with multiple clusters. For details, see **Multiple Storage Clusters for Loading Data**.
- [DB-21609]: Machine Learning Model Updates —
  - Added the **Gaussian Mixture Model**.
  - Added the **Association Rules Model**.
  - Added the **Linear Discriminant Analysis Model**.

Version Compatibility

- Large Geospatial Types are not backwards compatible with earlier releases. For details, see **Version Compatibility**.
- The database data control language (DCL) denotes user role privileges to remove data using the `DELETE` keyword instead of `TRUNCATE`.

22.0

Release Highlights

- HyperLogLog (HLL): Added HLL sketch functionality.
- Information Schema: Added the `information_schema` schema that shows system metadata.
- Integrations: Added drivers and support for the following third-party applications:
  - Metabase℠
  - SQLAlchemy
  - Apache® Superset®

Features

- [DB-13603]: Information Schema — Added the information_schema schema that shows system metadata in an accessible format.
- [DB-20484]: Superset Integration — Merged sqlalchemy-ocient driver into Superset repository, allowing Superset to support Ocient® database connections.
- [DB-20484]: SQLAlchemy Integration — Published sqlalchemy-ocient driver to PyPI.
- [DB-21011]: EXCEPT Clause — Added EXCEPT clause so that SELECT * queries can explicitly omit columns from results.
- [DB-21769]: Metabase Integration — Added Ocient as a Metabase partner driver, allowing Metabase to access Ocient databases out-of-the-box.
- [DB-23030]: JDBC Packaging — Removed OpenJump dependency from ocient-jdbc4.
- [DB-23175]: Time Zone Adjustment Support — Added various improvements to time zone functionality, including:
  - Support for daylight savings adjustment based on time zone.
  - Added time zone functions CONVERT_UTC_TIMESTAMP_TO LOCAL and CONVERT_LOCAL_TIMESTAMP_TO_UTC. For more information, see **Time Zone Functions**.
  - Enhanced performance for time zone conversion.
- [DB-23177]: Push-Down Aggregation to the I/O Layer — Under certain conditions, the system pushes aggregation to the I/O operator for better efficiency and performance.
- [DB-23299]: HLL Sketch Functionality — Added support for variable log2k HLL sketch algorithm and associated functions. For details, see the **HLL Functions page**.
- [DB-23745]: Implement Evacuate Node — Evacuate node is a tool to move all segments off of a node in a system that is overprovisioned to the other nodes in the cluster. This tool is useful when you replace drives or a node.
- [DB-19888]: Machine Learning Model Updates —
  - The Ocient System scopes machine learning models to schemas. The system assigns the `pre_v22_mlmodel` schema to any model you created prior to version 22.0.
  - The `sys.multiple_linear_regression_slopes` system catalog table has been removed.

- Rename machine learning models using ALTER MLMODEL.
- New DDL commands:
  - CREATE OR REPLACE
  - REFRESH
  - EXPORT
- For details about the new syntaxes, see **Machine Learning Model Functions**.

LAT Features

- [LAT-1469]: Manual Configuration of LAT Endpoints — Enabled manual configuration of LAT endpoints for OAuth with OKTA®.
- [LAT-1475]: Enablement of Stopping Load Processing During Error Condition — Added default behavior to stop processing during file loading in the event of an unrecoverable error when the system extracts records from a file. For details, see **continue_on_unrecoverable_error**.
- [LAT-1476]: Enablement of LAT Service in Installation — Enabled LAT Service in `systemd` by default upon installation completion. This update reflects a change in the default behavior during installation.
- [LAT-1477]: LAT Version for Metrics — Exposed `lat_version` in the metrics.
- [LAT-1557]: Support for Loading Multiple S3 Buckets — Added LAT functionality to load data from multiple S3 buckets simultaneously within the same pipeline.

Version Compatibility

- Information Schema — Views created prior to Version 22.0 do not have column data appearing in the information_schema. You can drop and recreate these views to populate column data.
- LAT — Version 3.0.0 and greater is only compatible with Version 22.0 and greater of the Ocient system. For details, see **Version Compatibility**.

21.0

Release Highlights

The Ocient System now supports the following operating systems:

- Ubuntu® 20.04
- Debian 11
- RHEL 8

Other highlights include:

- Whole column compression: Added Zstandard (ZSTD) compression for fixed and variable length columns.
- Check system configuration: Added `precheck` and `postcheck` commands to check system configuration before and after installation.
- Workload management dynamic priority: Enabled the adjustment of the query priority dynamically at the session, service class, and query levels.
- Ability to quiesce node: Added process for graceful node shutdown.

Features

- [DB-18636]: ZSTD Compression - Added a new whole-column compression scheme (ZSTD) that can be enabled for fixed and variable length columns.
- [DB-18990]: Improved Stats Storage And Usage - Various improvements have been added to speed up the fetching of statistics by the optimizer and ensure it gets up-to-date statistics. These changes primarily center around probability density functions being stored as pre-aggregated stats files instead of on a per-segment basis.
- [DB-20190]: Distributed Tasks - Added `check_disk` task type and new vtables `sys.subtasks`, `sys.tasks`, and `sys.rebuild_tasks` for monitoring tasks. Remove `CHECK DATA` command.
- [DB-19117]: Metadata - Added `participating_nodes` to the `sys.queries` and `sys.completed_queries` virtual tables.
- [DB-18633]: Graceful Node Shutdown - Added quiesce process for graceful node shutdown.
- [DB-18061]: LCK Deprecation - Added new disk data format that is smaller and also improves performance of some index based queries.

- [DB-19414]: Range Query Improvement - Improved performance of range queries by utilizing the inverted secondary index.
- [DB-20168]: Geospatial Function Expansion - Added these geospatial scalar functions.
    - Measurement Functions
        - ST_ANGLE
        - ST_DISTANCESPHERE
        - ST_DISTANCESPHEROID
        - ST_LENGTH2D
        - ST_HAUSDORFFDISTANCE
    - Analytic and Property Functions
        - ST_DIMENSION
        - ST_GEOHASH
        - ST_SRID
        - ST_ISPOLYGONCW
        - ST_ISPOLYGONCCW
    - To String and Binary Functions
        - ST_ASWKT
        - ST_ASWKB
        - ST_ASEWKT
    - Geography Simplification Function
        - ST_SIMPLIFY
    - Constructor Functions
        - ST_POINTFROMGEOHASH
        - ST_GEOGPOINT
        - ST_MAKEPOLYGONORIENTED
        - ST_POINT_FROMEWKT
        - ST_LINESTRING_FROMEWKT

    - ST_POLYGON_FROMEWKT
    - ST_MAKEENVELOPE
- Additionally, you can construct ST_POLYGON types directly from a POINT[] without going through an intermediate ST_LINESTRING.

Keywords

Added these new keywords as reserved words in the Ocient system.

- ANALYSIS
- AUTOREGRESSION
- BAYES
- CANCEL
- COMPONENT
- DECISION
- DISABLE
- DISABLE_STATS_FILE_UPDATES
- ENABLE
- FEEDFORWARD
- INSERT
- KMEANS
- KNN
- LOGISTIC
- MACHINE
- MOVE
- NAIVE
- NETWORK
- NONLINEAR
- PRINCIPAL
- REPLACE
- SOURCE

- SUPPORT
- TREE
- VECTOR
- ZSTD

20.0

Release Highlights

- CREATE TABLE AS SELECT SQL Statement: Extract, load, and transform (ELT) workflow functionality to extract data and load it into a new database table by using the query results from a SELECT SQL statement. The tables you create using the CREATE TABLE AS SELECT SQL statement have some indexing limitations in version 20.0. For details, see the “About Create Table As Select (CTAS)” section of the Ocient user documentation.
- INSERT INTO SQL Statement: ELT workflow functionality to extract data and insert it into an existing database table using the INSERT INTO SQL statement.
- N-gram Indexes: Full index on VARCHAR, VARCHAR arrays, and VARCHAR tuple components for efficient queries using the LIKE SQL statement.
- Large VARCHAR [DB-16142]: Support VARCHAR columns up to 1GB in size.
- Ocient Simulator: An instance of the Ocient system for data loading and functional testing.
- Single Sign-On (SSO): Authenticate access to Ocient through an external SSO server and assign SSO users to groups in Ocient.

Feature Removal

The ALTER ROLE DDL command has been removed. You can make all changes using the ALTER CONFIG SQL statement. To alter a role, prefix the key with the role name followed by a dot.

The following system tables have been added:

- average_bb_sizes
- linear_combination_regression_models

- node_config
- node_status
- sso_connections
- storage_device_status

The following system tables have been removed:

- hugepage_configurations
- memory_module_models
- node_memory_modules
- oidc_integrations
- oidc_sessions
- polynomial_regression_models
- security_integrations
- sessions

19.0

Features

- [DB-14527] - Adaptive Water Mark Feature - Indexer Node dynamically increase and reduce batch size without manual tuning
- [DB-14656] - Added a rest endpoint to expose a node’s configuration parameters (:9090/v1/configparams)
- [DB-15123] - Expose cluster total storage space and storage usage through virtual tables
- [DB-15515] - Add support for expr::dtype cast notation
- [DB-16289] - Remove the web ui and YAML service role configuration
- [DB-16904] - Allow any predicate type to be used in conjunction with the values in arrays
- [DB-17889] - Improve ability to continue data loading when a foundation node is down

- [DB-18393] - Leverage hyperthreading in query execution
- In v19 the service role configuration previously set through the web UI has been replaced by the ALTER … ALTER ROLE/CONFIG … DDL command. The web UI is still available in v19, but will be removed in a subsequent release. The ALTER … ALTER ROLE/CONFIG … command should be used to change system configuration, rather than the web UI. Please reference the Upgrade Ocient Software section of the user documentation for details.
- [DB-12747] - Add support for lateral joins.
- [DB-13924] - Add support for multi-column subqueries.
- [DB-14990] - Add support for native right joins.
- [DB-15996] - Add support for array_to_string function.
- [DB-16231] - Improve GIS function performance and introduce expanded support for GIS functions. Please refer to the User Documentation for details.
- [DB-17037] - Add new scalar functions and operators added for GIS types (POINT, LINESTRING, and POLYGON). Please refer to the User Documentation for details.
- [DB-17892] - Add support for right lateral joins.
- [DB-16061] - Secondary indexes can now be created on VARCHAR and VARCHAR[] columns. Please refer to the User Documentation for details.

18.0

Features

- [DB-17635] - Remove query log properties timestamp_optimizationcomplete and time_optimizationcomplete and add new properties timestamp_optimizationstart and time_optimizationstart
- [DB-16567] - Make error messages more clear for queries with GroupBy missing
- [DB-17316] - Change array_length(empty array) to return 0
- [DB-16417] - Allow for integral types for integer field is GIS functions
- [DB-16200] - Make Explains more convenient for the user.
- [DB-16092] - Distributed Result Set Caching
- [DB-15623] - Add support for rebuilding individual nodes via DDL

- [DB-15375] - ALTER CLUSTER ADD PARTICIPANTS DDL
- [DB-14720] - Provide a way to kill long running optimizations
- [DB-14017] - Support for CLI command history across sessions

17.0

Features

- [DB-12888] - Add support for array values larger than 128 KB. The new maximum value of an array is 512 MB.

16.0

Features

- [DB-10329] - Add support for full disk encryption of Opal drives. Disk encryption will be automatically enabled when Opal support is detected.

14.0

Features

- [DB-14159] - Default hex values for binary or varbinary columns must contain a leading 0x

13.0

Features

- [DB-14330] - Remove last dependencies on PostgreSQL from the database

12.0

Features

- [DB-13334] - Add support for zip unnest, which unnests multiple arrays in parallel

11.0

Features

- [DB-12887] - Add support for the array of tuples. Users can create array columns containing tuple SQL types. Please refer to the User Documentation for the latest information on supported data types.
- [DB-12885] - Add support for unnest(), which expands array elements from input array columns out to individual output rows

10.0

Features

- [DB-12394] - Added support for running on CentOS 8.

9.0

Features

- [DB-13162] - Added support for Tasks to the System Catalog
- [DB-10332] - Implemented access controls on system and database-level objects. Improved users, groups, and added new roles within Ocient.
- [DB-12829] - Optionally enforce encrypted connections for JDBC and ODBC.

8.0

Features

- [DB-10330] - External Network Security. SSL/TLS support in ODBC and JDBC, SSL support for the web interface.

© 2026 Ocient, Inc.

7.0

Features

- [DB-10921] - Adds support for multi-dimensional arrays and the ability to do joins, windows, sorts and aggregations that involve arrays.
- [DB-10927] - Adds support for global dictionary compression (GDC) on VARCHAR array columns and the ability to do replacement joins.
- [DB-10282] - Adds support for DROP COLUMN DDL to remove columns from a table.
- [DB-11472] - Adds support for skipping failed rows for CSV loading up to some specified threshold.

6.0

Features

- [DB-9707] - Scriptable Bulk Load Essentials: Allows users to create translations and launch bulk load tasks via DDL
- [DB-10479] - Adds support for Tableau through Ocient’s JDBC Custom Connector. Users can find Ocient’s connector and the installation instructions on Tableau’s extension gallery. Please refer to Tableau for more inforamation.

5.0

Features

- [DB-9477] - Adds support for the array data type. Users can create single-dimensional array columns from any other supported data type. Please refer to the User Documentation for the latest information on supported data types.
- [DB-9656] - Add column support. The engine now supports the add column DDL statement with the ability to add columns to an existing table. Existing data that was loaded without the new column uses the configured default values when queried. Please refer to User Documentation for information on the DDL syntax and default values.

4.0

Features

- [DB-6386] - Availability of the storage engine, allowing queries to run with a node or drive failure
- [DB-6588] - Bulk loading of CSV files from HDFS or an S3 endpoint
- [DB-7221] - Delta compression in the TKT engine for timestamp columns
- [DB-7623] - Virtual tables to retrieve information from the storage cluster state
- [DB-7247] - OS Upgrade functionality
- [DB-6125] - AWS initial support
- [DB-6383] - Data Definition Language (DDL) operations
- [DB-6940] - All system configuration in the System Catalog
- [DB-6362] - Stats Virtual Tables
- [DB-7098] - External Window Operator support
- [DB-7497] - List Running Tasks Page
- [DB-7097] - Segment Group Deletion
- [DB-7139] - Cancel Query and Cancel Task support
- Numerous stability and performance improvements

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

JDBC Release Notes

All Ocient® JDBC drivers are located the **Ocient Maven repository**.

4.2.1 (2026-06-26)

Internal updates only.

4.2.0 (2026-06-16)

- Added transactional support. You can now group multiple statements into a single transaction by disabling autocommit and ending the transaction with the `COMMIT` or `ROLLBACK` command. For details, see **Commands Supported by the Ocient JDBC CLI Program**.

4.1.0 (2026-04-01)

- Updated the input parameter for the `CANCEL` and `KILL` commands to be a quoted string for the Universally Unique IDentifier (UUID), such as `'c87b506f-66d6-4987-8bf7-1fd4a06b64b1'`.
- Added these new connection properties: `bulkLoadSshHostKeyVerification`, `bulkLoadSshKnownHostsPath`, and `maxRowsPerInsertBatch`. For details, see the **JDBC Connection Properties**.
- You can now configure JDBC CLI configuration options for the `PERFORMANCE` and `TIMING` commands. For details, see the **JDBC Manual**.
- Added these new extract options to the Data Extract Tool. For details, see **Data Extract Tool**:
  - FILE_PREFIX_EXISTS
  - FILE_TYPE
  - NUM_FETCH_QUERIES
  - PARTITION_MODE

© 2026 Ocient, Inc.

- PARTITION_COLUMNS
- SUCCESS_MARKER
- TARGET_FILE_SIZE_MB

4.0.0 (2026-02-05)

- Added the Apache® Spark™ Connector that enables the Ocient System for Spark workloads. The connector allows Spark to read from and write to Ocient tables using Spark APIs and SQL statements. For details, see **JDBC Spark Connector**.
- Enabled SSO-authenticated users to access multiple databases using a single identity provider configuration.
- Added graph functionality using the `OCGraph` Java® class. For details, see **OCGraph Java Library**.

3.7.0 (2025-11-21)

- Added quotes around `VARCHAR` data types in tuples and changed delimiter characters to parentheses. Now, the tuple output is `("foo", 5)` instead of `<<foo, 5>>`.

3.6.7 (2025-10-11)

- CLI is no longer replacing or removing any whitespace characters inside quoted fields.

3.6.6 (2025-10-01)

- Enabled the `source` command to handle files containing the `connect to` command.

3.6.5 (2025-09-25)

Internal updates only.

3.6.4 (2025-09-10)

- Added the `printUuid` setting to the `.ocient-cli-configuration` file that determines whether the CLI automatically prints the Universally Unique IDentifier (UUID) of the query after the submission of the query by the user. The values are `on` and `off`. If you do not specify this setting, the default value is `off`.
- Re-enabled asynchronous updates. Now, the system executes updates asynchronously by default.

3.5.1 (2025-06-16)

- Disabled asynchronous updates.

3.5.0 (2025-06-07)

- Added support for asynchronous updates, enabling you to retrieve the query identifier immediately after executing the SQL statement instead of when it finishes on the server. Now, you can use Ctrl-C to cancel the update at the command line.
- Added support for graph algorithms using vertex and edge table representations. These algorithms cover the same capabilities as GraphX from Apache® Spark™.
- Added the `DISTRIBUTE_WORK` connection parameter, which distributes queries or commands between all endpoints in a round-robin way, instead of relying on server-side load balancing.

3.4.1 (2025-05-14)

- New Features
  - Changed the default value for the `path-style-access` extract option to `true`.

3.4.0 (2025-04-30)

- New Features

- Removed quotes added during the extract around `VARCHAR` data types inside arrays by default. Added the `QUOTE_ARRAY_VARCHARS` option for the extract that, when you set the value to `true`, adds those quotes back.
- Removed the leading forward slash from IP addresses extracted from the database.
- Added periodic polling of the `.ocient-cli-configuration` file, so the system applies changes to the settings in the file at runtime. The default interval is 5 minutes. You can change the default value by setting the `com.ocient.cli.config-poll-interval` system property to the chosen number of seconds.
- Fixed the CLI result set output from formatted to a raw string.

3.3.1 (2025-03-30)

Internal updates only.

3.3.0 (2025-03-06)

- New Features
  - Added the specification of the identity provider in the connection string when connecting to the database.
  - Added support for the bzip2 and xz compression.
  - Added support for multi-line Apache® Groovy™ definitions in SQL statements.
  - Changed the default SSO callback address to `http://localhost:7050/ocient/oauth2/v1/callback`.
  - Added the `ssoNumericAddress` parameter to force the client to use the IP address `127.0.0.1` instead of `localhost` when you set this parameter to `true`.
  - Enabled switching the metrics target directory without having to restart the CLI session.
  - Added support for line and block comments in multi-line SQL statements.

3.2.4 (2025-01-03)

Internal updates only.

3.2.3 (2024-12-13)

Internal updates only.

3.2.2 (2024-11-08)

Internal updates only.

3.2.1 (2024-09-30)

- New Features
  - Changed the default escape character for quotes in VARCHAR arrays from `\` to `"` for extracts.
  - Added connection property to redact usernames and SQL text in JDBC logs.
  - Increased security for passwords in the command-line interface and history file.

3.1.7 (2024-06-03)

Internal updates only.

3.1.6 (2024-06-03)

Internal updates only.

3.1.5 (2024-06-03)

Bugs

- [DB-28722]: Fixed the issue where the client closed the result set before sending the KILL_QUERY request.

- [DB-28325]: Fixed an underlying SLF4JLogger error that prevented the driver from importing.
- [DB-24984]: Fixed HTTP client configuration to accept standard cookie specifications.

3.1.1 (2024-03-08)

- *New Features*
  - Enabled support of fetching cached queries using the JDBC 3.1.1 driver with version 22.1 and older versions of the OcientAIQ™ Unified Data Platform.
  - Fixed connection failures by adding sequence numbers to metadata responses on the client side for legacy databases without sequence number support.

3.1.0 (2024-01-31)

- *New Features*
  - Improved maximum throughput between client and server, specifically for transferring large and sorted result sets.
  - Deprecated `com.ocient.jdbc.XGConnection#redirect`. The redirect method in the `com.ocient.jdbc.XGConnection` class is now deprecated and is slated for removal in future releases. For applications that require a connection to a specific SQL Node, Ocient recommends to use the `force=true` connection property.

3.0.0 (2023-10-24)

- *New Features*
  - Improved scalability for high-volume concurrent database connections for applications such as Metabase℠ and Superset.
  - Improved latency for bulk data transfers.

2.106 (2023-06-29)

- *New Features*
  - Fixed requirement to restart the Java® application after cluster upgrade.

2.105 (2023-03-22)

- *New Features*
  - Improved performance by pooling database connections used for sideband channels.
  - Added `MAX_THREADS_PER_RESULTSET` connection property.

2.100 (2023-03-10)

- *New Features*
  - Added ability to periodically log metrics to a file.
  - Added `METRICS ON <alphanumeric>` CLI command.

2.99 (2023-01-24)

- *New Features*
  - Added `PERFORMANCE NETWORK` CLI command.
  - Added `PERFORMANCE DATABASE` CLI command.
  - Deprecate `PERFORMANCE ON` CLI command.

2.98 (2023-01-17)

- New Features
  - Removed OpenJump system dependency

2.97 (2022-12-27)

- New Features

- Added support for `DELETE FROM`

2.96 (2022-12-21)

- New Features
  - Fixed `IndexOutOfBoundsException` in server redirect

2.95 (2022-12-07)

- New Features
  - Log stack trace of `XGStatement#sendAndReceive()` read errors

2.94 (2022-12-07)

- New Features
  - Allow extract to `/dev/null`

2.93 (2022-11-28)

- New Features
  - Strip trailing statement delimiter from queries

2.92 (2022-11-22)

- New Features
  - Include `jdbc.sql.Driver service in META-INF`

2.91 (2022-11-22)

- New Features

- Add support for `REFRESH MLMODEL` command, which allows updating of a Machine Learning model without changing the ML model properties

2.90 (2022-11-22)

- [DB-22501] Never resubmit queries that have been sent to the server

2.89 (2022-11-18)

- Add support for EXPORT MLMODEL command

2.88 (2022-11-08)

- Load JDBC version from build properties file at runtime

2.87 (2022-11-07)

- Set Service Class

2.86 (2022-10-24)

- Allow connection property override

2.85 (2022-10-24)

- Address CVE-2022-33980 and CVE-2022-24823

2.84 (2022-10-24)

- Print hex instead of address of byte array in XGTuple#toString

2.83 (2022-10-18)

- Re-enabled ability to override a connection property
- Fix query timeouts which were not being honored
- Handle non-resolvable database DNS names received from the database

2.82 (2022-10-14)

- Made JDBC Connection properties case-insensitive using a new class CaselessProperties
- Converted the rest of the connection properties to the ConnectionProperty enum

2.81 (2022-10-03)

- Implement XGPreparedStatement#setObject(Index, Timestamp, Calendar)
- Add “statementPooling” driver property. Set to “OFF” to disable statement caching

2.80 (2022-10-03)

- Add `GET JDBC COMMIT INFO` command
- Add `/version`, `/commit`, `/info`, and `/pid` metrics endpoints

2.79 (2022-10-03)

- Ensure backwards compatibility with JRE8

2.78 (2022-09-19)

- Add CLI support for `EXPORT PIPELINE’

2.77 (2022-09-19)

- Add `START PIPELINE` and `STOP PIPELINE` for updates

© 2026 Ocient, Inc.

2.76 (2022-09-02)

- Made `SHOW` command work as a SELECT outside of just the CLI

2.75 (2022-09-02)

- Made `SHOW` command a select command rather than an update command

2.74 (2022-08-29)

- `BENCHMARK` command consumes result set for consistency w/ manual input

2.73 (2022-08-26)

- New Features
  - Add support for `SHOW` command, which allows easy querying of the information schema.

2.72 (2022-08-23)

- New Features
  - [DB-20912] Add support for one-time password (OTP) based SSO flows
  - [DB-20912] Add OAuth2 Authorization Code w/ PKCE Single Sign-On Flow
  - [DB-20912] Add OAuth2 Device Authorization Grant Single Sign-On Flow for input constrained environments

2.71 (2022-08-23)

- New Features
  - Add `BENCHMARK NSAMPLES COMMAND` which will run `COMMAND` `NSAMPLES` times and produce a timing report in the CLI output.
  - Add `PRINTUUID [ON | OFF]` which will print the query UUID to the CLI output.

© 2026 Ocient, Inc.

2.70 (2022-08-11)

- New Features
  - [DB-21078] Add `s3_upload_part_size` and `s3_upload_part_parallelism` extract configuration properties.

2.69 (2022-08-11)

- New Features
  - [DB-21078] Add REST endpoint to expose query and data extract performance metrics
  - [DB-21078] Add ms precision to JDBC log entries

2.68 (2022-08-09)

- New Features
  - [DB-21251] Allow extract property override via “JDBC_EXTRACT_*” prefixed environment variable

2.67 (2022-06-30)

- New Features
  - Allow aliases on nested queries in extract SQL statements (DB-21075)

2.66 (2022-06-27)

- New Features
  - Escape escape characters in quoted field for non-comma delimited extracts

2.65

- New Features

- Add support for using `CANCEL TASK` to cancel a task instead of treating it like a `CANCEL` command

2.64 (2022-06-21)

- New Features
  - Resolve CVE-2022-24823; update netty-common version (4.1.77.Final)
  - Resolve CVE-2021-43797; update netty-codec-http version (4.1.71.Final)

2.63 (2022-06-01)

- New Features
  - Replace `skip_header` with `header_mode` in CSV extract options.
  - Add CLI configuration file, `~/.ocient-cli-configuration`, and idle timeout option `cliIdleTimeoutMinutes`

2.62 (2022-06-01)

- New Features
  - Add `escape_unquoted_values`, `input_escaped`, and `quote_all_fields` in CSV extract options.
  - Add a new custom TSV writer that handles all non-comma delimiter cases.

2.61 (2022-05-31)

- New Features
  - Add `compression_block_size`, `compression_level`, and `num_compression_threads` Data extract properties.

2.60 (2022-05-24)

- New Features

- Allow non-query commands to print results when performance mode is turned on for easier debugging.

2.59 (2022-05-17)

- New Features
  - Add UTF-8 character translation to Data extract tool.

2.58 (2022-4-20)

- New Features
  - Prevent shell expansion of `!` when reading user input from CLI

2.57 (2022-3-17)

- New Features
  - Bumped protocol version for quiesce
  - Modified length checks to check if higher order bit is set, signalling a message from the server

2.56 (2022-3-15)

- New Features
  - Added SQL states for EXECUTE_PLAN_AUTH_FAILURE and EXECUTE_INLINE_PLAN_AUTH_FAILURE

2.55 (2022-3-11)

- New Features
  - Reconnect and rerun if writing into socket fails when sending request
  - Reconnect and rerun if when sending request, shows -1 length

- Added checks for -1 length on response for server signalling a quiescing connection close
- Copy secondary interfaces and secondary index when returning a cached connection

2.54 (2022-3-9)

- New Features
  - Added CLI support for SET ADJUSTFACTOR, ADJUSTTIME
  - Added new columns in system queries: sys.queries: initial_priority, initial_effective_priority, effective_priority, priority_adjust_factor, priority_adjust_time sys.completed_queries: initial_priority, initial_effective_priority, final_effective_priority, priority_adjust_factor, priority_adjust_time

2.53 (2022-2-8)

- New Features
  - Added redirect support for execute export, explain pipeline, check data

2.52 (2022-1-18)

- New Features
  - Secondary result set threads will start fetching immediately after the first result set thread gets a ping.

2.51 (2022-1-14)

- New Features
  - Make force external consistent with other connection parameters

2.50 (2022-1-13)

- New Features
  - Fix invalid argument message to not refer to ODBC client

2.49 (2022-1-6)

- New Features
  - Add more logging to multithreaded result set fetch.

2.48 (2022-1-3)

- New Features
  - Insert statements supported

2.47 (2021-12-16)

- New Features
  - Data extract tool released.
  - Deactive connection caching for calls to createConnection

2.46 (2021-12-8)

- New Features
  - Fix connection pooling handling with regards to setting schema incorrectly.

2.45 (2021-12-2)

- New Features
  - Swallow runtime error as well when trying to set socket options.

2.44 (2021-12-1)

- New Features
  - Wrap invalid socket options in try catch block and swallow exceptions. For compatability with Java versions that are older.

2.43 (2021-11-22)

- New Features
  - Change driver to not rely on the pom/MANIFEST for the driver version. Move the version into JDBCDriver

2.42 (2021-11-18)

- New Features
  - Changed connections to utilize keep alive probes in order to detect dead connections.

2.41 (2021-11-07)

- New Features
  - Upgrade Jline version to 3.21

2.40 (2021-11-03)

- New Features
  - Improve result set fetching performance

2.39 (2021-10-29)

- New Features
  - Fix handling of expired token being used to start a session

2.38 (2021-10-27)

- New Features
  - Updated auth related SQLState error codes
  - Fix handling of result set close during result set caching

2.37 (2021-10-25)

- New Features
  - Force redirect support for testing purposes.
  - Removed duplicate code from CLI

2.36 (2021-10-14)

- New Features
  - Implemented refresh sessions support.
  - Implement get server session ID.

2.35 (2021-10-11)

- New Features
  - ExecutePlan and ExecuteInlinePlan bugfixes

2.34 (2021-9-9)

- New Features
  - Add token signature and issuer fingerprint fields to sso token handshake.
  - Add explicit SSO flags to handshake GCM.
  - Pass networkTimeout to new connection upon copy.

2.33 (2021-9-7)

- New Features
  - Add support for explicit SSO handshakes
  - Revert executeQuery redirection change

2.32 (2021-9-1)

- New Features
  - CLI source command stops on error
  - Support “output next query append”
  - Add statement command “get jdbc version”

2.31 (2021-8-30)

- New Features
  - Add sso handshake support
  - Refactor duplicated code in handshakes for saving secondary interfaces

2.30 (2021-8-26)

- New Features
  - Add validation on the driver side for set parameter settings.
  - Pass up invalid set parameter command errors.

2.29 (2021-8-20)

- New Features
  - Fix the way connection resets parameters when reset is true.
  - Fix an issue where redirect was not properly using mapped secondary interfaces

© 2026 Ocient, Inc.

2.28 (2021-8-16)

- New Features
  - Fix sending parameters on reset. maxrow, maxtempdisk, maxtime, parallelism, priority now gets reset properly.
  - Map list all queries to select * from sys.queries
  - Improve logging
  - Statements with closed connections will not get returned to the cache

2.27 (2021-8-10)

- New Features
  - Fixed poorly structured if statement in CLI source command.
  - Remove more stack trace printing and add more logs.

2.26 (2021-7-30)

- New Features
  - Move the duplicated regex code in CLI.java and XGStatement.java into a new file.
  - Implement a generic regex for syntax checking the set family of sql commands.
  - Fix a bug in resetting commands using lower cases. “set maxrow reset;” does not work. Needs to be capitalized.

2.25 (2021-7-27)

- New Features
  - Default clientVersion to 0.00 if the driver provides a null clientVersion to XGConnection.

2.24 (2021-7-12)

- New Features
  - Have setSchema handle unquoted caps sql. “SET SCHEMA MADISON” will set schema to “madison”.

2.23 (2021-7-6)

- New Features
  - Change SET CONCURRENCY command to SET PARALLELISM
  - Change “concurrency” driver property to “parallelism”

2.22 (2021-7-1)

- New Features
  - Add openJump extensions to jdbc jar.
  - Fix source command for plan execute inline.

2.21 (2021-6-30)

- New Features
  - Fix new hashcode for null elements.

2.20 (2021-6-28)

- New Features
  - Update list all queries metadata to match that of select * from sys.queries
  - Add more connection level settings to hashCode.

2.19 (2021-6-26)

- New Features
  - Improve source command capabilities and performance

© 2026 Ocient, Inc.

2.18 (2021-6-26)

- New Features
  - Save command history across sessions

2.17 (2021-6-25)

- New Features
  - Include the currently set schema to the connection hashcode.

2.16 (2021-6-23)

- New Features
  - Fix another issue with timeoutMillis. The timeout will now work correctly with zero ping buffers.

2.15 (2021-6-21)

- New Features
  - For dBeaver compatability, not setting a user in driver properties will cause the driver to default to empty string.

2.14 (2021-6-17)

- New Features
  - Fix timeoutMillis by correctly inheriting timeoutMillis from properties.
  - Clear warnings before running executeQuery and executeUpdate.

2.13 (2021-6-09)

- New Features

- Added command to SET PSO SEED for the random number generator used in PSO
- Remove chatty number of rows log.
- Add remoteIp and service class to list all queries.
- Add sleep_in_optimizer command for testing.

2.12 (2021-5-26)

- New Features
  - Fix the parsing for execute() to properly route to executeStatement()

2.11 (2021-5-20)

- New Features
  - Improve cache and statement pooling support for redirection.
  - Fix timestamp and time with negative nanos.

2.10 (2021-5-19)

- New Features
  - Added 10 different colors for GIS types in KML
  - Added non-GIS types to description of each GIS object in KML
  - Moved KML feature from CLI to driver

2.09 (2021-5-18)

- New Features
  - Use daemon threads for background caching tasks. This allows the program to exit without finishing these tasks.

2.08 (2021-5-10)

- New Features
  - Add versions packaging to pom.xml.
  - Updated dependencies to latest version using versions plugin.
  - Change TLS to unverified by default.

2.07 (2021-5-05)

- New Features
  - Add OBJECT_NOT_FOUND_WARN warning.
  - Add OBJECT_ALREADY_EXISTS_WARN warning.
  - Fix some misleading log messages.

2.06 (2021-4-29)

- New Features
  - Add a log for if reconnect() fails to close a socket.
  - Driver sends over unique session ID to server.
  - Fix a misleading log in client handshake.

2.05 (2021-4-28)

- New Features
  - Fix spotbugs and narrow spotbugs filter.

2.04 (2021-4-22)

- New Features
  - Added getters to GIS types

2.03 (2021-4-22)

- New Features
  - Output next query also prints all gis types into a kml file

2.01 (2021-4-21)

- New Features
  - Fix empty point for STPoint class

2.01 (2021-4-20)

- New Features
  - Better build integration with xgsrc
  - Upgraded protobuf to version 3.14

2.00 (2021-4-17)

- New Features
  - Handling for cache limit warning.

1.99 (2021-4-13)

- New Features
  - Add support for EXPLAIN DEBUG, and make EXPLAIN format JSON by default

1.98 (2021-4-05)

- New Features
  - Add support for QUARANTINE
  - Fix CLI command force external on

© 2026 Ocient, Inc.

1.97 (2021-4-02)

- New Features
  - Queries that exceed the row limit set by XGConnection::setMaxRows now silently omit excess rows. The previous behavior would result in query failure.

1.96 (2021-4-01)

- New Features
  - Add support for POINT EMPTY
  - Fix CLI performance on/off affecting timing setting.

1.95 (2021-3-22)

- New Features
  - Add support for CHECK DATA to CLI
  - Fix bug where defaultSchema was not working.

1.94 (2021-3-18)

- New Features
  - Add major and minor version to client handshake.
  - Add logging to set param functions.

1.93 (2021-3-11)

- New Features
  - Improve performance of writing query results to a file.

1.92 (2021-3-5)

- New Features

- Add driver method to cancel all cache return threads.
- Add CLI functionality to limit max history size.

1.91 (2021-3-4)

- New Features
  - Add support for st_linestring and st_polygon

1.90 (2021-3-2)

- New Features
  - Enable spotbugs analysis
  - Increase tracing to millisecond granularity

1.89 (2021-2-24)

- New Features
  - Switch to using AES/GCM/NoPadding encryption

1.88 (2021-2-24)

- New Features
  - Fix some broken custom xgMetadata calls.

1.87 (2021-2-24)

- New Features
  - Added support for TUPLE columns as a SQL Struct type

1.86 (2021-2-19)

- New Features
  - Fix more OWASP bugs

1.85 (2021-2-18)

- New Features
  - Added CLI support for EXPORT VIEW

1.84 (2021-2-16)

- New Features
  - Update the request type for EXPLAIN PIPELINE

1.83 (2021-2-15)

- New Features
  - Fixed a bunch of OWASP bugs.

1.82 (2021-2-12)

- New Features
  - Add CLI support for SET MAXROWS, PRIORITY, CONCURRENCY, MAXTIME, and MAXTEMPDISK

1.81 (2021-2-12)

- New Features
  - Fix a compiler error with CLUSTER_NOT_FOUND error

1.80 (2021-2-12)

- New Features
  - Add CLUSTER_NOT_FOUND error

1.79 (2021-2-11)

- New Features
  - Add spotbugs
  - Add support for EXPLAIN PIPELINE to CLI

1.78 (2021-2-5)

- New Features
  - Add SqlStates for LUP decomposition
  - Add additional cache logging.
  - Fix incorrect schema generated when first statements are created.

1.77 (2021-2-5)

- New Features
  - Remove unecessary calls to fetchServerVersion
  - Fix schema for pooled connections and statement.

1.76 (2021-2-2)

- New Features
  - Pooling improvements

1.75 (2021-2-1)

- New Features

- Allow changing session variable defaults via connection properties and make session variable overrides local to statement objects.

1.74 (2021-1-31)

- New Features
  - Performance optimizations and restructuring

1.73 (2021-1-30)

- New Features
  - The JDBC driver now does connection and statement pooling that is automatic and transparent to the caller.

1.72 (2021-1-27)

- New Features
  - Add support for clearBatch()

1.71 (2021-1-14)

- New Features
  - Add support for session overrides of service class limits
  - SET MAXROWS {N}
  - SET MAXTIME {N}
  - SET PRIORITY {N}
  - SET CONCURRENCY {N}

1.70 (2021-1-12)

- New Features

- Fix race condition between adding fetch threads to array and iterating. Also removed some duplicated code.

1.69 (2021-1-07)

- New Features
  - Remove deprecated plan proto

1.68 (2020-12-24)

- New Features
  - Fix describe view truncate in CLI.

1.67 (2020-12-14)

- New Features
  - Fixed cols2Types mapping in RS returned by LIST ALL COMPLETED QUERIES.

1.66 (2020-12-10)

- New Features
  - Add spaces to syntax parsing.

1.65 (2020-12-9)

- New Features
  - Completed queries update.

1.64 (2020-12-7)

- New Features

© 2026 Ocient, Inc.

- Properly handle nullability for the LIST ALL COMPLETED QUERIES command.

1.63 (2020-12-1)

- New Features
  - Don’t print stack trace when select queries complete exceptionally.

1.62 (2020-11-30)

- New Features
  - Added support for LIST ALL COMPLETED QUERIES command.

1.61 (2020-11-15)

- New Features
  - Fix reconnect and resending logic.

1.60 (2020-11-10)

- New Features
  - DESCRIBE TABLE now consistently uses SMALLINT and BIGINT to describe inner types of arrays

1.59 (2020-11-4)

- New Features
  - remove another unnecessary driver version fetch.
  - add jar version to log

1.58 (2020-11-4)

- New Features

- fix fetch version redirect loop.

1.57 (2020-11-4)

- New Features
  - make SQLException codes unique.

1.56 (2020-11-2)

- New Features
  - kill and cancel now throws syntax errors.

1.55 (2020-10-28)

- New Features
  - CLI now recognizes a DDL statement for INVALIDATE STATS.

1.54 (2020-10-21)

- New Features
  - fix another spot where all the slq nodes can be brought down.

1.53 (2020-10-20)

- New Features
  - exportTranslation will close the result set.

1.52 (2020-10-19)

- New Features
  - executeQuery will not rerun query after reconnect.

© 2026 Ocient, Inc.

1.51 (2020-10-11)

- New Features
  - Add token fields and username to localQueries protobuf

1.50 (2020-10-11)

- New Features
  - Multiple result set threads support is complete

1.49 (2020-10-06)

- New Features
  - Experimental support for multiple result set threads

1.48 (2020-09-28)

- New Features
  - Minor fix of SQLStates
  - setParms() does no work when there are no parameters

1.47 (2020-09-26)

- New Features
  - Improve performance for large SQL statements

1.46 (2020-09-21)

- New Features
  - Bug fix for driver not reconnecting when a previous result set is still open.

© 2026 Ocient, Inc.

1.45 (2020-09-16)

- New Features
  - Move export translation into result set.

1.44 (2020-09-15)

- New Features
  - Support granting and revoking both privileges and role membership.

1.43 (2020-09-14)

- New Features
  - Explain, listTables, listViews, getSchema, describeTable, describeView, explainPlan, listAllQueries, and exportTable.

1.42 (2020-09-2)

- New Features
  - Move some custom functionalities into the driver including: get/set schema, list (system) tables, list views, describe table/views, list indexes, execute/explain plan, cancel/kill query, list all queries, export table, set max rows, set pso.

1.41 (2020-08-20)

- New Features
  - Ability to set max output rows from the CLI

1.40 (2020-08-14)

- New Features
  - SSL support.

© 2026 Ocient, Inc.

1.37 (2020-07-26)

- New Features
  - Timestamp and Time now have nanosecond precision

1.36 (2020-07-25)

- New Features
  - JDBC driver can now handle hostnames sent back in interface list

1.35 (2020-07-23)

- New Features
  - Capture initial connection IP and use as last resort for reconnect()

1.34 (2020-07-14)

- New Features
  - Implement new time/date/timestamp subclasses to circumvent bugs in the standard versions

1.33 (2020-07-04)

- New Features
  - Add support for load balancing with secondary SQL interfaces

1.32 (2020-06-26)

- New Features
  - Add support for Export Translation to CLI

© 2026 Ocient, Inc.

1.31 (2020-06-23)

- New Features
  - Fix misbehavior with ‘plan execute inline’ using the CLI

1.30 (2020-06-22)

- New Features
  - Deprecated the HDFS connection table, which was consolidated into the external connection table

1.29 (2020-06-18)

- New Features
  - Creating an MLModel on no data is now an error, not a warning

1.28 (2020-06-11)

- New Features
  - Allow one connection to be shared by multiple threads, as long as each thread uses its own Statement object

1.27 (2020-06-11)

- New Features
  - Improve trace log formatting, including adding thread id

1.26 (2020-06-09)

- New Features
  - Add support for weeks interval type

1.25 (2020-06-05)

- New Features
  - Support for getTableTypes()

1.24 (2020-06-05)

- New Features
  - Protobuf message changes for Add Column feature

1.23 (2020-06-02)

- New Features
  - Add new error code: Value too large, which indicates that a column value is larger than internal limits

1.22 (2020-05-31)

- New Features
  - Add calendar support
- Bug Fixes
  - Properly handle case-insensitivity of column names on result sets

1.21 (2020-05-26)

- New Features
  - Add tracing support

1.20 (2020-05-19)

- Bug Fixes

- DB-11119 - JDBC multi-host support was not implemented correctly

1.19 (2020-05-13)

1.18 (2020-05-11)

1.16 (2020-04-15)

- Bug Fixes
  - DB-10687 - Fix for Kill/Cancel query

1.15 (2020-04-14)

- Bug Fixes
  - DB-9928 - Fixes for SQL Array type

1.14 (2020-04-11)

- New Features
  - Protocol support for the NULLS FIRST in the ORDER BY clause

1.13 (2020-03-20)

- New Features
  - Binary support for the serialization of the following data types: ST_POINT, UUID, IP and IPV4
  - Implementation of the Array SQL type

1.12 (2020-03-01)

- Bug Fixes
  - DB-10155 - Broadcast kill query and cancel query requests to all sql nodes.

- New Features
  - DB-10137 - Implement JDBC cancel query. Now it is possible to kill query in dbeaver.
  - DB-10120 - JDBC driver support list of sql nodes to connect to
  - DB-10119 - Support for multiple IPs under the same DNS address

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

pyocient Release Notes

3.8.0

- Added the `memory_tracing` flag for memory-based debugging.

3.7.0

- Added transactional support.
- Added asynchronous update support.
- Added the display of the full complex type information.
- Enabled the printing of the query Universally Unique IDentifier (UUID) when the system fails or experiences a Keyboard Interrupt.
- Added generic server-side `cancel` and `kill` commands. This addition is not compatible with an Ocient® System that has a version less than 25.0.

3.6.0

- Added support for Dataflow SQL statements. Dataflow is a procedural superset of SQL that acts as an ad-hoc, anonymous, and imperative script. This new syntax parses, compiles, and executes logic in a single request using flow control, variables, scoped temporary tables, and exception handling. This new functionality requires version 27 or later of the Ocient® System.
- Added support for cross-database SSO authentication.
- Added support for Google® as an OIDC provider with server-side token exchange and polling.
- Added support for specifying identity providers and SSO OAuth flow options.
- Added support for SSL and hostname in SSO callback URLs.
- Added support for token lifetimes in the client wire protocol.
- Added the commit SHA to the pyocient package for version tracking.
- Added support for URL-encoded special characters in DSN passwords.

© 2026 Ocient, Inc.

- Fixed the driver to return NULL elements in multidimensional arrays as `None` instead of empty arrays.
- Fixed precision and scale handling during the cast of arrays and tuples that contain DECIMAL types.
- Removed an unnecessary debug print statement.
- Fixed the parsing of CLI argument keys to be lowercase.
- Improved CLI parsing of Ocient SQL statements.
- Reduced the verbosity of debug messages during normal operation.
- Added graph functionality using the `ocient_graph` Python® package.

3.5.0

- Added support for executable SQL files using shebang lines in the command-line interface (CLI).
- Added `expect error` and `expect rowcount` commands to the CLI.
- Added enhanced help functionality to the CLI.
- Added nanosecond precision support for timestamp data types.
- Added server-side `DESCRIBE TABLE` and `DESCRIBE VIEW` command support with fallback to the client-side `DESCRIBE` command for older server versions.
- Added connection timeout support to prevent hanging connections.
- Added configurable redirect URIs per OIDC integration for SSO functionality.
- Fixed an infinite loop issue when the fetch size is set to 0.
- Improved parameter expansion handling and error messaging.
- Improved Unicode character formatting in JSON output and CLI display.
- Updated protobuf dependency to version 5.29.2 or higher.
- Reduced verbosity for connection-type logging messages.
- Added support for preserving schema and service class settings after reconnection.
- Improved error handling for invalid DSNs in the CLI.

3.4.0

- Added support for the `PREVIEW` DDL SQL statement.
- Support cryptography versions greater than 42.0.0.
- Fixed an issue that returned `-1` rows modified.
- Fixed the handling of an invalid DSN on the pyocient CLI.
- Formatted Unicode characters in JSON output.
- Fixed an issue where not all warnings were reported.
- Added the messages attribute to the cursor.
- Improved parameter expansion.
- Supported loglevel and logfile parameters on DSNs.
- Preserved the Connection object for reconnects.

3.3.2

- Fixed ending SSO sessions when you close the connection.

3.3.0

- Pinned the cryptography package to a version earlier than 42.0.0 for stability purposes.
- Added OIDC single sign on (SSO).
- Added the `performance mode` command.
- Enabled the display of nulls as NULL when using the table format.
- Set the command-line default fetch size to 30000.
- Fixed the implementation of the `set serviceclass` command.
- Changed `pyocient.Warning` to be a subclass of the Python® `Warning` class.
- Added the `set printuuid on` command.
- Added the `--version` option at the command line.
- Set the TCP `keepalive` options on connections.
- Added a valid error message when the system rejects a connection without SSL encryption.

3.0.3

- The `TypeCodes` class has been changed to a Python `IntEnum`.
- GIS types _STPoint, _STLinestring, and _STPolygon have been renamed to STPoint, STLinestring, and STPolygon, respectively.
- The `user` and `password` fields have been removed from the `Connection` object. This object no longer stores these fields.
- You can pass the `security_token` from another connection to the `connect` function.
- The `Session` object has been removed.
- The `custom_type_to_json` function has been moved into the pyocient API to allow the consistent conversion of the result set to the JSON format.
- The `patch` portion of the pyocient version is sent to and logged in the database.
- The `--nooutput` command-line parameter does not throw a Python exception.
- The `local variable 'return_code' referenced before assignment` error has been fixed.
- Added the `--rows` argument to the command-line program to load the result set in batches.

2.0.0

- Packaging has been significantly refactored to match Python® packaging standards.
- Python context manager support has been added to connections and cursors.
- Python type annotations have been added to the pyocient API.
- Fixed the command-line interface (CLI) to return a non-zero code for statements that fail.
- CLI output has been standardized to either go to the output file or standard output (stdout).
- The option to echo statements has been added as a command-line option.
- The `pyocient.SyntaxError` exception, which conflicted with the built-in Python `SyntaxException`, has been removed.

© 2026 Ocient, Inc.

- `pyocient.version` has been removed in favor of the more standard `pyocient.__version__`.

1.0.15

- The dependency on `protobuf` has been upgraded from `>=3.12.0`, `<=3.19.1` to `>=3.20.0`, `<=4.22.0`.
- Fixed an error that occurs when you do not specify the host parameter in the `connect()` API.

1.0.14

- Updated pyocient dependencies to eliminate some unused dependencies and tighten the dependencies on `protobuf` and `dsnparse`.

1.0.13

- Fixed bug that left `Cursor.rowcount` unset after Data Manipulation Language (DML) query execution.
- The `cryptography` package requirement of version 36 or lower is no longer required.

1.0.12

- Added `--nohistory` command-line option. The history file does not store commands from the session. Added `ignorespace` option to command-line history. The history file omits lines that begin with a white space character.

1.0.11

- Added a link to the release notes to the package description.

1.0.10

- Fixed error in the parameter substitution docstring.
- Added support for `SERVICECLASS`, `ADJUSTFACTOR`, `ADJUSTTIME`, and `PSO` query settings.
- Added support for the `SHOW` command.
- Added support for the `CANCEL TASK` command.
- Removed support for `PRIORITY_ADJUSTMENT_FACTOR`, `PRIORITY_ADJUSTMENT_TIME`, `MIN_PRIORITY`, and `MAX_PRIORITY` query settings.
- Deprecate support for the `CHECK DATA` command.
- Fixed output of binary types at the command line.

1.0.9

- Added CSV output support
- Updated cryptography package dependency from <3.4 to <=36.0
- Updated cffi and google-auth dependencies
- Improved decoding performance by caching the decoders
- Added formal STPoint, STLinestring, and STPolygon classes rather than using tuples and lists
- Added support for PRIORITY_ADJUSTMENT_FACTOR, PRIORITY_ADJUSTMENT_TIME, MIN_PRIORITY, and MAX_PRIORITY query settings.
- Enabled the handling of invalid DNS names received from the database

1.0.7

- Fixed crash on invalid UTF-8 characters received from the database
- Fixed a bug related to specifying the user through the connect API
- Standardized Python(R) code formatting

1.0.6

- Updated handling of the force option on connections for accuracy

1.0.5

- Added additional debug logging to the redirection of connection

1.0.4

- Updated the redirect logic for accuracy

1.0.3

- Added multiple host support to the DSN
- Removed —tables, —systables, and —views command-line options
- Added support for the source, connect to, and set commands
- Added support for starting without a DSN

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Glossary

Use this table to reference frequently used acronyms and terms with the Ocient® System or the OcientAIQ™ Unified Data Platform.

Name	Definition
AA	Auto Acquire
AdTech	Advertising technology
ANTLR	ANother Tool for Language Recognition
API	Application Programming Interface
ASIO	Async I/O library
AST	Abstract Syntax Tree
CFL	Continuous File Loading
CmdComp	Command Compiler
CTAS	CREATE TABLE AS SELECT SQL statement
CTE	Common Table Expression
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DR	Disaster Recovery
ELT	Extract, Load, and Transform
ETL	Extract, Transform, and Load
FinTech	Financial technology
GDC	Global Dictionary Compression
GDS	Global Data Storage
GIS	Geospatial Information System
HA	High Availability
IAS	INSERT INTO SELECT SQL statement

© 2026 Ocient, Inc.

Name	Definition
IDA	Information Dispersal Algorithm
LCK	Local Cluster Key
LTS	Foundation Node
NUMA	Non-Uniform Memory Architecture
NVMe	Non-Volatile Memory Express
OLAP	Online Analytical Processing
OSC	Ocient Stream Client
OSN	Ownership Sequence Number
TKT	TimeKey® Time
TTQ	Time to Queryability
UDT	User-defined Transform
UIO	User mode I/O
UUID	Universally Unique Identifier
VM	Virtual Machine
WLM	Workload Management

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

Open Source Acknowledgments

The Ocient® System uses some third-party software to perform certain operations. Here, you can find acknowledgments for this software.

asn1bean

Text

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object

form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use,

reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only

```
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

flat_hash_map (skarupke)

Text

```
//          Copyright Malte Skarupke 2017.
// Distributed under the Boost Software License, Version 1.0.
//    (See http://www.boost.org/LICENSE_1_0.txt)
```

mpmcQueue

Text

```
Copyright (c) 2020 Erik Rigtorp <erik@rigtorp.se>
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

antlr

Text

```
Copyright (c) 2012-2022 The ANTLR Project. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

3. Neither name of copyright holders nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

aws-sdk-cpp

Text

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the

© 2026 Ocient, Inc.

editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with

the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf

```
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "{}"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

libbacktrace

Text

```
# Copyright (C) 2012-2016 Free Software Foundation, Inc.

# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are
# met:

#     (1) Redistributions of source code must retain the above copyright
#     notice, this list of conditions and the following disclaimer.

#     (2) Redistributions in binary form must reproduce the above copyright
#     notice, this list of conditions and the following disclaimer in
#     the documentation and/or other materials provided with the
#     distribution.

#     (3) The name of the author may not be used to
#     endorse or promote products derived from this software without
#     specific prior written permission.

# THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
# IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
# INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
# STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
# IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
# POSSIBILITY OF SUCH DAMAGE.
```

boost

Text

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

cityhash

Text

```
// Copyright (c) 2011 Google, Inc.
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in
// all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
// THE SOFTWARE.
```

cjose

Text

```
The MIT License (MIT)

Copyright (c) 2014-2016 Cisco Systems, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

abseil-cpp

Text

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object

form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use,

reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only

```
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

protobuf

Text

```
Copyright 2008 Google Inc.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
    * Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner
of the input file used when generating it.  This code is not
standalone and requires a support library to be linked with it.  This
support library is itself covered by the above license.
```

cpp_httplib

Text

```
The MIT License (MIT)

Copyright (c) 2017 yhirose

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

cpr

Text

```
This license applies to everything except the contents of the "test"
directory and its subdirectories.

MIT License

Copyright (c) 2017-2021 Huu Nguyen
Copyright (c) 2022 libcpr and many other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

curl

Text

```
COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1996 - 2025, Daniel Stenberg, <daniel@haxx.se>, and many
contributors, see the THANKS file.

All rights reserved.

Permission to use, copy, modify, and distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright
notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN
NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not
be used in advertising or otherwise to promote the sale, use or other dealings
in this Software without prior written authorization of the copyright holder.
```

date

Text

```
// The MIT License (MIT)
//
// Copyright (c) 2015, 2016, 2017 Howard Hinnant
// Copyright (c) 2016 Adrian Colomitchi
// Copyright (c) 2017 Florian Dang
// Copyright (c) 2017 Paul Thompson
// Copyright (c) 2018, 2019 Tomasz Kamiński
// Copyright (c) 2019 Jiangang Zhuang
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in all
// copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
// SOFTWARE.
//
// Our apologies.  When the previous paragraph was written, lowercase had not yet
```

dpdk

Text

```
Redistribution and use in source and binary forms, with or without modification,

1. Redistributions of source code must retain the above copyright notice, this li

2. Redistributions in binary form must reproduce the above copyright notice, this

3. Neither the name of the copyright holder nor the names of its contributors may

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND A
```

icu4c

Text

```
UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE

See Terms of Use <https://www.unicode.org/copyright.html>
for definitions of Unicode Inc.’s Data Files and Software.

NOTICE TO USER: Carefully read the following legal agreement.
BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S
DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"),
YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE
TERMS AND CONDITIONS OF THIS AGREEMENT.
IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE
THE DATA FILES OR SOFTWARE.

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2022 Unicode, Inc. All rights reserved.
Distributed under the Terms of Use in https://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining
a copy of the Unicode data files and any associated documentation
(the "Data Files") or Unicode software and any associated documentation
(the "Software") to deal in the Data Files or Software
without restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, and/or sell copies of
the Data Files or Software, and to permit persons to whom the Data Files
or Software are furnished to do so, provided that either
(a) this copyright and permission notice appear with all copies
of the Data Files or Software, or
(b) this copyright and permission notice appear in associated
Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT OF THIRD PARTY RIGHTS.
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS
NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL
DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,
DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THE DATA FILES OR SOFTWARE.
```

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

Third-Party Software Licenses

This section contains third-party software notices and/or additional terms for licensed third-party software components included within ICU libraries.

ICU License - ICU 1.8.1 to ICU 57.1

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2016 International Business Machines Corporation and others
All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

```
Except as contained in this notice, the name of a copyright holder
shall not be used in advertising or otherwise to promote the sale, use
or other dealings in this Software without prior written authorization
of the copyright holder.

All trademarks and registered trademarks mentioned herein are the
property of their respective owners.

--------------------------------------------------------------------

Chinese/Japanese Word Break Dictionary Data (cjdict.txt)

 #     The Google Chrome software developed by Google is licensed under
 # the BSD license. Other software included in this distribution is
 # provided under other licenses, as set forth below.
 #
 #  The BSD License
 #  http://opensource.org/licenses/bsd-license.php
 #  Copyright (C) 2006-2008, Google Inc.
 #
 #  All rights reserved.
 #
 #  Redistribution and use in source and binary forms, with or without
 # modification, are permitted provided that the following conditions are met:
 #
 #  Redistributions of source code must retain the above copyright notice,
 # this list of conditions and the following disclaimer.
 #  Redistributions in binary form must reproduce the above
 # copyright notice, this list of conditions and the following
 # disclaimer in the documentation and/or other materials provided with
 # the distribution.
 #  Neither the name of  Google Inc. nor the names of its
 # contributors may be used to endorse or promote products derived from
 # this software without specific prior written permission.
 #
 #
 #  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
 # CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
 # INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 # MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 # DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
 # LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 # CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
```

```
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
# BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
# LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
# SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
#  The word list in cjdict.txt are generated by combining three word lists
# listed below with further processing for compound word breaking. The
# frequency is generated with an iterative training against Google web
# corpora.
#
#  * Libtabe (Chinese)
#    - https://sourceforge.net/project/?group_id=1519
#    - Its license terms and conditions are shown below.
#
#  * IPADIC (Japanese)
#    - http://chasen.aist-nara.ac.jp/chasen/distribution.html
#    - Its license terms and conditions are shown below.
#
#  ---------COPYING.libtabe ---- BEGIN--------------------
#
#  /*
#   * Copyright (c) 1999 TaBE Project.
#   * Copyright (c) 1999 Pai-Hsiang Hsiao.
#   * All rights reserved.
#   *
#   * Redistribution and use in source and binary forms, with or without
#   * modification, are permitted provided that the following conditions
#   * are met:
#   *
#   * . Redistributions of source code must retain the above copyright
#   *   notice, this list of conditions and the following disclaimer.
#   * . Redistributions in binary form must reproduce the above copyright
#   *   notice, this list of conditions and the following disclaimer in
#   *   the documentation and/or other materials provided with the
#   *   distribution.
#   * . Neither the name of the TaBE Project nor the names of its
#   *   contributors may be used to endorse or promote products derived
#   *   from this software without specific prior written permission.
#   *
#   * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#   * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
```

```
#   * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
#   * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#   * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#   * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#   * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#   * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
#   * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#   * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
#   * OF THE POSSIBILITY OF SUCH DAMAGE.
#   */
#
#  /*
#   * Copyright (c) 1999 Computer Systems and Communication Lab,
#   *                    Institute of Information Science, Academia
#       *                    Sinica. All rights reserved.
#   *
#   * Redistribution and use in source and binary forms, with or without
#   * modification, are permitted provided that the following conditions
#   * are met:
#   *
#   * . Redistributions of source code must retain the above copyright
#   *   notice, this list of conditions and the following disclaimer.
#   * . Redistributions in binary form must reproduce the above copyright
#   *   notice, this list of conditions and the following disclaimer in
#   *   the documentation and/or other materials provided with the
#   *   distribution.
#   * . Neither the name of the Computer Systems and Communication Lab
#   *   nor the names of its contributors may be used to endorse or
#   *   promote products derived from this software without specific
#   *   prior written permission.
#   *
#   * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#   * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#   * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
#   * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#   * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#   * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#   * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#   * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
#   * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#   * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
```

```
#   * OF THE POSSIBILITY OF SUCH DAMAGE.
#   */
#
#  Copyright 1996 Chih-Hao Tsai @ Beckman Institute,
#      University of Illinois
#  c-tsai4@uiuc.edu  http://casper.beckman.uiuc.edu/~c-tsai4
#
#  ---------------COPYING.libtabe-----END--------------------------------
#
#
#  ---------------COPYING.ipadic-----BEGIN-------------------------------
#
#  Copyright 2000, 2001, 2002, 2003 Nara Institute of Science
#  and Technology.  All Rights Reserved.
#
#  Use, reproduction, and distribution of this software is permitted.
#  Any copy of this software, whether in its original form or modified,
#  must include both the above copyright notice and the following
#  paragraphs.
#
#  Nara Institute of Science and Technology (NAIST),
#  the copyright holders, disclaims all warranties with regard to this
#  software, including all implied warranties of merchantability and
#  fitness, in no event shall NAIST be liable for
#  any special, indirect or consequential damages or any damages
#  whatsoever resulting from loss of use, data or profits, whether in an
#  action of contract, negligence or other tortuous action, arising out
#  of or in connection with the use or performance of this software.
#
#  A large portion of the dictionary entries
#  originate from ICOT Free Software.  The following conditions for ICOT
#  Free Software applies to the current dictionary as well.
#
#  Each User may also freely distribute the Program, whether in its
#  original form or modified, to any third party or parties, PROVIDED
#  that the provisions of Section 3 ("NO WARRANTY") will ALWAYS appear
#  on, or be attached to, the Program, which is distributed substantially
#  in the same form as set out herein and that such intended
#  distribution, if actually made, will neither violate or otherwise
#  contravene any of the laws and regulations of the countries having
#  jurisdiction over the User or the intended distribution itself.
#
#  NO WARRANTY
```

```
#
#  The program was produced on an experimental basis in the course of the
#  research and development conducted during the project and is provided
#  to users as so produced on an experimental basis.  Accordingly, the
#  program is provided without any warranty whatsoever, whether express,
#  implied, statutory or otherwise.  The term "warranty" used herein
#  includes, but is not limited to, any warranty of the quality,
#  performance, merchantability and fitness for a particular purpose of
#  the program and the nonexistence of any infringement or violation of
#  any right of any third party.
#
#  Each user of the program will agree and understand, and be deemed to
#  have agreed and understood, that there is no warranty whatsoever for
#  the program and, accordingly, the entire risk arising from or
#  otherwise connected with the program is assumed by the user.
#
#  Therefore, neither ICOT, the copyright holder, or any other
#  organization that participated in or was otherwise related to the
#  development of the program and their respective officials, directors,
#  officers and other employees shall be held liable for any and all
#  damages, including, without limitation, general, special, incidental
#  and consequential damages, arising out of or otherwise in connection
#  with the use or inability to use the program or any product, material
#  or result produced or otherwise obtained by using the program,
#  regardless of whether they have been advised of, or otherwise had
#  knowledge of, the possibility of such damages at any time during the
#  project or thereafter.  Each user will be deemed to have agreed to the
#  foregoing by his or her commencement of use of the program.  The term
#  "use" as used herein includes, but is not limited to, the use,
#  modification, copying and distribution of the program and the
#  production of secondary products from the program.
#
#  In the case where the program, whether in its original form or
#  modified, was distributed or delivered to or received by a user from
#  any person, organization or entity other than ICOT, unless it makes or
#  grants independently of ICOT any specific warranty to the user in
#  writing, such person, organization or entity, will also be exempted
#  from and not be held liable to the user for any such damages as noted
#  above as far as the program is concerned.
#
#  ---------------COPYING.ipadic-----END---------------------------------

-------------------------------------------------------------------------
```

Lao Word Break Dictionary Data (laodict.txt)

```
 # Copyright (C) 2016 and later: Unicode, Inc. and others.
 # License & terms of use: http://www.unicode.org/copyright.html
 # Copyright (c) 2015 International Business Machines Corporation
 # and others. All Rights Reserved.
 #
 # Project: https://github.com/rober42539/lao-dictionary
 # Dictionary: https://github.com/rober42539/lao-dictionary/laodict.txt
 # License: https://github.com/rober42539/lao-dictionary/LICENSE.txt
 #          (copied below)
 #
 #  This file is derived from the above dictionary version of Nov 22, 2020
 #  --------------------------------------------------------------------
 #  Copyright (C) 2013 Brian Eugene Wilson, Robert Martin Campbell.
 #  All rights reserved.
 #
 #  Redistribution and use in source and binary forms, with or without
 #  modification, are permitted provided that the following conditions are met:
 #
 #  Redistributions of source code must retain the above copyright notice, this
 #  list of conditions and the following disclaimer. Redistributions in binary
 #  form must reproduce the above copyright notice, this list of conditions and
 #  the following disclaimer in the documentation and/or other materials
 #  provided with the distribution.
 #
 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 # "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 # LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 # FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 # COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 # INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 # (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 # SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 # HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 # STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 # ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 # OF THE POSSIBILITY OF SUCH DAMAGE.
 #  --------------------------------------------------------------------

-----------------------------------------------------------------------
```

```
Burmese Word Break Dictionary Data (burmesedict.txt)

 #  Copyright (c) 2014 International Business Machines Corporation
 #  and others. All Rights Reserved.
 #
 #  This list is part of a project hosted at:
 #    github.com/kanyawtech/myanmar-karen-word-lists
 #
 #  -------------------------------------------------------------------------
 #  Copyright (c) 2013, LeRoy Benjamin Sharon
 #  All rights reserved.
 #
 #  Redistribution and use in source and binary forms, with or without
 #  modification, are permitted provided that the following conditions
 #  are met: Redistributions of source code must retain the above
 #  copyright notice, this list of conditions and the following
 #  disclaimer.  Redistributions in binary form must reproduce the
 #  above copyright notice, this list of conditions and the following
 #  disclaimer in the documentation and/or other materials provided
 #  with the distribution.
 #
 #    Neither the name Myanmar Karen Word Lists, nor the names of its
 #    contributors may be used to endorse or promote products derived
 #    from this software without specific prior written permission.
 #
 #  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
 #  CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
 #  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 #  MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 #  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
 #  BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 #  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
 #  TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 #  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
 #  ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
 #  TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
 #  THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 #  SUCH DAMAGE.
 #  -------------------------------------------------------------------------

--------------------------------------------------------------------

Time Zone Database
```

ICU uses the public domain data and code derived from Time Zone Database for its time zone support. The ownership of the TZ database is explained in BCP 175: Procedure for Maintaining the Time Zone Database section 7.

```
 # 7.  Database Ownership
 #
 #    The TZ database itself is not an IETF Contribution or an IETF
 #    document.  Rather it is a pre-existing and regularly updated work
 #    that is in the public domain, and is intended to remain in the
 #    public domain.  Therefore, BCPs 78 [RFC5378] and 79 [RFC3979] do
 #    not apply to the TZ Database or contributions that individuals make
 #    to it.  Should any claims be made and substantiated against the TZ
 #    Database, the organization that is providing the IANA
 #    Considerations defined in this RFC, under the memorandum of
 #    understanding with the IETF, currently ICANN, may act in accordance
 #    with all competent court orders.  No ownership claims will be made
 #    by ICANN or the IETF Trust on the database or the code.  Any person
 #    making a contribution to the database or code waives all rights to
 #    future claims in that contribution or in the TZ Database.
```

Google double-conversion

Copyright 2006-2011, the V8 project authors. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

File: aclocal.m4 (only for ICU4C)
Section: pkg.m4 - Macros to locate and utilise pkg-config.

Copyright © 2004 Scott James Remnant <scott@netsplit.com>.
Copyright © 2012-2015 Dan Nicholson <dbn.lists@gmail.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

As a special exception to the GNU General Public License, if you distribute this file as part of a program that contains a configuration script generated by Autoconf, you may include it under the same distribution terms that you use for the rest of that program.

(The condition for the exception is fulfilled because ICU4C includes a configuration script generated by Autoconf,

namely the `configure` script.)

File: config.guess (only for ICU4C)

This file is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, see <https://www.gnu.org/licenses/>.

As a special exception to the GNU General Public License, if you
distribute this file as part of a program that contains a
configuration script generated by Autoconf, you may include it under
the same distribution terms that you use for the rest of that
program. This Exception is an additional permission under section 7
of the GNU General Public License, version 3 ("GPLv3").

(The condition for the exception is fulfilled because
ICU4C includes a configuration script generated by Autoconf,
namely the `configure` script.)

File: install-sh (only for ICU4C)

Copyright 1991 by the Massachusetts Institute of Technology

Permission to use, copy, modify, distribute, and sell this software and its
documentation for any purpose is hereby granted without fee, provided that
the above copyright notice appear in all copies and that both that
copyright notice and this permission notice appear in supporting

```
documentation, and that the name of M.I.T. not be used in advertising or
publicity pertaining to distribution of the software without specific,
written prior permission.  M.I.T. makes no representations about the
suitability of this software for any purpose.  It is provided "as is"
without express or implied warranty.
```

interval-tree

Text

Creative Commons Legal Code

CC0 1.0 Universal

    CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS DOCUMENT DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED HEREUNDER, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED HEREUNDER.

Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer exclusive Copyright and Related Rights (defined below) upon the creator and subsequent owner(s) (each and all, an "owner") of an original work of authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for the purpose of contributing to a commons of creative, cultural and scientific works ("Commons") that the public can reliably and without fear of later claims of infringement build upon, modify, incorporate in other works, reuse and redistribute as freely as possible in any form whatsoever and for any purposes, including without limitation commercial purposes. These owners may contribute to the Commons to promote the ideal of a free culture and the further production of creative, cultural and scientific works, or to gain reputation or greater distribution for their Work in part through the use and efforts of others.

For these and/or other purposes and motivations, and without any expectation of additional consideration or compensation, the person associating CC0 with a Work (the "Affirmer"), to the extent that he or she is an owner of Copyright and Related Rights in the Work, voluntarily elects to apply CC0 to the Work and publicly distribute the Work under its terms, with knowledge of his or her Copyright and Related Rights in the Work and the meaning and intended legal effect of CC0 on those rights.

1. Copyright and Related Rights. A Work made available under CC0 may be protected by copyright and related or neighboring rights ("Copyright and

Related Rights"). Copyright and Related Rights include, but are not limited to, the following:

i. the right to reproduce, adapt, distribute, perform, display, communicate, and translate a Work;
ii. moral rights retained by the original author(s) and/or performer(s);
iii. publicity and privacy rights pertaining to a person's image or likeness depicted in a Work;
iv. rights protecting against unfair competition in regards to a Work, subject to the limitations in paragraph 4(a), below;
v. rights protecting the extraction, dissemination, use and reuse of data in a Work;
vi. database rights (such as those arising under Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, and under any national implementation thereof, including any amended or successor version of such directive); and
vii. other similar, equivalent or corresponding rights throughout the world based on applicable law or treaty, and any national implementations thereof.

2. Waiver. To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

3. Public License Fallback. Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected

person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "License"). The License shall be deemed effective as of the date CCO was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's express Statement of Purpose.

4. Limitations and Disclaimers.

a. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.
b. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.
c. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.
d. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CCO or use of the Work.

isa-l

Text

```
Copyright(c) 2011-2024 Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
  * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in
    the documentation and/or other materials provided with the
    distribution.
  * Neither the name of Intel Corporation nor the names of its
    contributors may be used to endorse or promote products derived
    from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SPDX-License-Identifier: BSD-3-Clause
```

isa-l_crypto

Text

```
Copyright(c) 2011-2024 Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
  * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in
    the documentation and/or other materials provided with the
    distribution.
  * Neither the name of Intel Corporation nor the names of its
    contributors may be used to endorse or promote products derived
    from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SPDX-License-Identifier: BSD-3-Clause
```

jansson

Text

```
Copyright (c) 2009-2020 Petri Lehtinen <petri@digip.org>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

jemalloc

Text

```
Unless otherwise specified, files in the jemalloc source distribution are
subject to the following license:
--------------------------------------------------------------------------------
Copyright (C) 2002-present Jason Evans <jasone@canonware.com>.
All rights reserved.
Copyright (C) 2007-2012 Mozilla Foundation.  All rights reserved.
Copyright (C) 2009-present Facebook, Inc.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice(s),
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice(s),
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER(S) ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
EVENT SHALL THE COPYRIGHT HOLDER(S) BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
--------------------------------------------------------------------------------
```

jsoncpp

Text

```
The JsonCpp library's source code, including accompanying documentation,
tests and demonstration applications, are licensed under the following
conditions...

Baptiste Lepilleur and The JsonCpp Authors explicitly disclaim copyright in all
jurisdictions which recognize such a disclaimer. In such jurisdictions,
this software is released into the Public Domain.

In jurisdictions which do not recognize Public Domain property (e.g. Germany as c
2010), this software is Copyright (c) 2007-2010 by Baptiste Lepilleur and
The JsonCpp Authors, and is released under the terms of the MIT License (see belc

In jurisdictions which recognize Public Domain property, the user of this
software may choose to accept it either as 1) Public Domain, 2) under the
conditions of the MIT License (see below), or 3) under the terms of dual
Public Domain/MIT License conditions described here, as they choose.

The MIT License is about as close to Public Domain as a license can get, and is
described in clear, concise terms at:

   http://en.wikipedia.org/wiki/MIT_License

The full text of the MIT License follows:

========================================================================
Copyright (c) 2007-2010 Baptiste Lepilleur and The JsonCpp Authors

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
```

```
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
==========================================================================
(END LICENSE TEXT)

The MIT license is compatible with both the GPL and commercial
software, affording one all of the rights of Public Domain with the
minor nuisance of being required to keep the above copyright notice
and license text in the source code. Note also that by accepting the
Public Domain "license" you can re-license your copy using whatever
license you like.
```

krb5

Text

```
Copyright (C) 1985-2024 by the Massachusetts Institute of Technology.

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in the
  documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Downloading of this software may constitute an export of cryptographic
software from the United States of America that is subject to the
United States Export Administration Regulations (EAR), 15 CFR 730-774.
Additional laws or regulations may apply.  It is the responsibility of
the person or entity contemplating export to comply with all
applicable export laws and regulations, including obtaining any
required license from the U.S. government.

The U.S. government prohibits export of encryption source code to
certain countries and individuals, including, but not limited to, the
countries of Cuba, Iran, North Korea, Sudan, Syria, and residents and
nationals of those countries.

Documentation components of this software distribution are licensed
```

under a Creative Commons Attribution-ShareAlike 3.0 Unported License.
(https://creativecommons.org/licenses/by-sa/3.0/)

Individual source code files are copyright MIT, Cygnus Support,
Novell, OpenVision Technologies, Oracle, Red Hat, Sun Microsystems,
FundsXpress, and others.

Project Athena, Athena, Athena MUSE, Discuss, Hesiod, Kerberos, Moira,
and Zephyr are trademarks of the Massachusetts Institute of Technology
(MIT). No commercial use of these trademarks may be made without
prior written permission of MIT.

"Commercial use" means use of a name in a product or other for-profit
manner. It does NOT prevent a commercial firm from referring to the
MIT trademarks in order to convey information (although in doing so,
recognition of their trademark status should be given).

==

The following copyright and permission notice applies to the
OpenVision Kerberos Administration system located in "kadmin/create",
"kadmin/dbutil", "kadmin/passwd", "kadmin/server", "lib/kadm5", and
portions of "lib/rpc":

   Copyright, OpenVision Technologies, Inc., 1993-1996, All Rights
   Reserved

   WARNING: Retrieving the OpenVision Kerberos Administration system
   source code, as described below, indicates your acceptance of the
   following terms. If you do not agree to the following terms, do
   not retrieve the OpenVision Kerberos administration system.

   You may freely use and distribute the Source Code and Object Code
   compiled from it, with or without modification, but this Source
   Code is provided to you "AS IS" EXCLUSIVE OF ANY WARRANTY,
   INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR
   FITNESS FOR A PARTICULAR PURPOSE, OR ANY OTHER WARRANTY, WHETHER
   EXPRESS OR IMPLIED. IN NO EVENT WILL OPENVISION HAVE ANY LIABILITY
   FOR ANY LOST PROFITS, LOSS OF DATA OR COSTS OF PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR
   CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, INCLUDING,
   WITHOUT LIMITATION, THOSE RESULTING FROM THE USE OF THE SOURCE
   CODE, OR THE FAILURE OF THE SOURCE CODE TO PERFORM, OR FOR ANY

OTHER REASON.

OpenVision retains all copyrights in the donated Source Code. OpenVision also retains copyright to derivative works of the Source Code, whether created by OpenVision or by a third party. The OpenVision copyright notice must be preserved if derivative works are made based on the donated Source Code.

OpenVision Technologies, Inc. has donated this Kerberos Administration system to MIT for inclusion in the standard Kerberos 5 distribution. This donation underscores our commitment to continuing Kerberos technology development and our gratitude for the valuable work which has been performed by MIT and the Kerberos community.

==========

Portions contributed by Matt Crawford "crawdad@fnal.gov" were work performed at Fermi National Accelerator Laboratory, which is operated by Universities Research Association, Inc., under contract DE-AC02-76CH03000 with the U.S. Department of Energy.

==========

Portions of "src/lib/crypto" have the following copyright:

Copyright (C) 1998 by the FundsXpress, INC.

All rights reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations

about the suitability of this software for any purpose. It is
provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

==========

The implementation of the AES encryption algorithm in
"src/lib/crypto/builtin/aes" has the following copyright:

    Copyright (C) 1998-2013, Brian Gladman, Worcester, UK. All
    rights reserved.

The redistribution and use of this software (with or without
changes) is allowed without the payment of fees or royalties
provided that:

    source code distributions include the above copyright notice,
    this list of conditions and the following disclaimer;

    binary distributions include the above copyright notice, this
    list of conditions and the following disclaimer in their
    documentation.

This software is provided 'as is' with no explicit or implied
warranties in respect of its operation, including, but not limited
to, correctness and fitness for purpose.

==========

Portions contributed by Red Hat, including the pre-authentication
plug-in framework and the NSS crypto implementation, contain the
following copyright:

    Copyright (C) 2006 Red Hat, Inc.
    Portions copyright (C) 2006 Massachusetts Institute of Technology
    All Rights Reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above
  copyright notice, this list of conditions and the following
  disclaimer in the documentation and/or other materials provided
  with the distribution.

* Neither the name of Red Hat, Inc., nor the names of its
  contributors may be used to endorse or promote products derived
  from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

==========

The bundled verto source code is subject to the following license:

Copyright 2011 Red Hat, Inc.

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

==========

The MS-KKDCP client implementation has the following copyright:

Copyright 2013,2014 Red Hat, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

```
The implementations of GSSAPI mechglue in GSSAPI-SPNEGO in
"src/lib/gssapi", including the following files:

   lib/gssapi/generic/gssapi_err_generic.et
   lib/gssapi/mechglue/g_accept_sec_context.c
   lib/gssapi/mechglue/g_acquire_cred.c
   lib/gssapi/mechglue/g_canon_name.c
   lib/gssapi/mechglue/g_compare_name.c
   lib/gssapi/mechglue/g_context_time.c
   lib/gssapi/mechglue/g_delete_sec_context.c
   lib/gssapi/mechglue/g_dsp_name.c
   lib/gssapi/mechglue/g_dsp_status.c
   lib/gssapi/mechglue/g_dup_name.c
   lib/gssapi/mechglue/g_exp_sec_context.c
   lib/gssapi/mechglue/g_export_name.c
   lib/gssapi/mechglue/g_glue.c
   lib/gssapi/mechglue/g_imp_name.c
   lib/gssapi/mechglue/g_imp_sec_context.c
   lib/gssapi/mechglue/g_init_sec_context.c
   lib/gssapi/mechglue/g_initialize.c
   lib/gssapi/mechglue/g_inquire_context.c
   lib/gssapi/mechglue/g_inquire_cred.c
   lib/gssapi/mechglue/g_inquire_names.c
   lib/gssapi/mechglue/g_process_context.c
   lib/gssapi/mechglue/g_rel_buffer.c
   lib/gssapi/mechglue/g_rel_cred.c
   lib/gssapi/mechglue/g_rel_name.c
   lib/gssapi/mechglue/g_rel_oid_set.c
   lib/gssapi/mechglue/g_seal.c
   lib/gssapi/mechglue/g_sign.c
   lib/gssapi/mechglue/g_store_cred.c
   lib/gssapi/mechglue/g_unseal.c
   lib/gssapi/mechglue/g_userok.c
   lib/gssapi/mechglue/g_utils.c
   lib/gssapi/mechglue/g_verify.c
   lib/gssapi/mechglue/gssd_pname_to_uid.c
   lib/gssapi/mechglue/mglueP.h
   lib/gssapi/mechglue/oid_ops.c
   lib/gssapi/spnego/gssapiP_spnego.h
   lib/gssapi/spnego/spnego_mech.c

and the initial implementation of incremental propagation, including
the following new or changed files:
```

```
   include/iprop_hdr.h
   kadmin/server/ipropd_svc.c
   lib/kdb/iprop.x
   lib/kdb/kdb_convert.c
   lib/kdb/kdb_log.c
   lib/kdb/kdb_log.h
   lib/krb5/error_tables/kdb5_err.et
   kprop/kpropd_rpc.c
   kprop/kproplog.c

are subject to the following license:

   Copyright (C) 2004 Sun Microsystems, Inc.

   Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation
   files (the "Software"), to deal in the Software without
   restriction, including without limitation the rights to use, copy,
   modify, merge, publish, distribute, sublicense, and/or sell copies
   of the Software, and to permit persons to whom the Software is
   furnished to do so, subject to the following conditions:

   The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.

   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
   BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
   ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
   SOFTWARE.

========================================================================

Kerberos V5 includes documentation and software developed at the
University of California at Berkeley, which includes this copyright
notice:

      Copyright (C) 1983 Regents of the University of California.
      All rights reserved.
```

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==

Portions contributed by Novell, Inc., including the LDAP database backend, are subject to the following license:

Copyright (C) 2004-2005, Novell, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

```
   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.

   * Redistributions in binary form must reproduce the above
     copyright notice, this list of conditions and the following
     disclaimer in the documentation and/or other materials provided
     with the distribution.

   * The copyright holder's name is not used to endorse or promote
     products derived from this software without specific prior
     written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
   COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
   INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
   (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
   SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
   HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
   STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
   OF THE POSSIBILITY OF SUCH DAMAGE.

========================================================================

Portions funded by Sandia National Laboratory and developed by the
University of Michigan's Center for Information Technology
Integration, including the PKINIT implementation, are subject to the
following license:

      COPYRIGHT (C) 2006-2007
      THE REGENTS OF THE UNIVERSITY OF MICHIGAN
      ALL RIGHTS RESERVED

   Permission is granted to use, copy, create derivative works and
   redistribute this software and such derivative works for any
   purpose, so long as the name of The University of Michigan is not
   used in any advertising or publicity pertaining to the use of
   distribution of this software without specific, written prior
   authorization.  If the above copyright notice or any other
   identification of the University of Michigan is included in any
```

copy of any portion of this software, then the disclaimer below must also be included.

THIS SOFTWARE IS PROVIDED AS IS, WITHOUT REPRESENTATION FROM THE UNIVERSITY OF MICHIGAN AS TO ITS FITNESS FOR ANY PURPOSE, AND WITHOUT WARRANTY BY THE UNIVERSITY OF MICHIGAN OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE REGENTS OF THE UNIVERSITY OF MICHIGAN SHALL NOT BE LIABLE FOR ANY DAMAGES, INCLUDING SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WITH RESPECT TO ANY CLAIM ARISING OUT OF OR IN CONNECTION WITH THE USE OF THE SOFTWARE, EVEN IF IT HAS BEEN OR IS HEREAFTER ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

==========

The pkcs11.h file included in the PKINIT code has the following license:

Copyright 2006 g10 Code GmbH
Copyright 2006 Andreas Jellinghaus

This file is free software; as a special exception the author gives unlimited permission to copy and/or distribute it, with or without modifications, as long as this notice is preserved.

This file is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, to the extent permitted by law; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

==========

Portions contributed by Apple Inc. are subject to the following license:

Copyright 2004-2008 Apple Inc. All Rights Reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Apple Inc. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Apple Inc. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

==========

The implementations of UTF-8 string handling in src/util/support and src/lib/krb5/unicode are subject to the following copyright and permission notice:

The OpenLDAP Public License
Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices,

2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and

3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distribute verbatim copies of this document is granted.

==========

Marked test programs in src/lib/krb5/krb have the following copyright:

Copyright (C) 2006 Kungliga Tekniska Högskola
(Royal Institute of Technology, Stockholm, Sweden).
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials provided
   with the distribution.

3. Neither the name of KTH nor the names of its contributors may
   be used to endorse or promote products derived from this
   software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY KTH AND ITS CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KTH OR ITS
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

==

The KCM Mach RPC definition file used on macOS has the following
copyright:

    Copyright (C) 2009 Kungliga Tekniska Högskola
    (Royal Institute of Technology, Stockholm, Sweden).
    All rights reserved.

Portions Copyright (C) 2009 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above
   copyright notice, this list of conditions and the following
   disclaimer.

2. Redistributions in binary form must reproduce the above

copyright notice, this list of conditions and the following
disclaimer in the documentation and/or other materials provided
with the distribution.

3. Neither the name of the Institute nor the names of its
contributors may be used to endorse or promote products derived
from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

==========

Portions of the RPC implementation in src/lib/rpc and
src/include/gssrpc have the following copyright and permission notice:

Copyright (C) 2010, Oracle America, Inc.

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above
copyright notice, this list of conditions and the following
disclaimer.

2. Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following
disclaimer in the documentation and/or other materials provided
with the distribution.

3. Neither the name of the "Oracle America, Inc." nor the names
   of its contributors may be used to endorse or promote products
   derived from this software without specific prior written
   permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

==========

Copyright (C) 2006,2007,2009 NTT (Nippon Telegraph and Telephone
Corporation). All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above
   copyright notice, this list of conditions and the following
   disclaimer as the first lines of this file unmodified.

2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials provided
   with the distribution.

THIS SOFTWARE IS PROVIDED BY NTT "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL NTT BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

Copyright 2000 by Carnegie Mellon University

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Carnegie Mellon University not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

CARNEGIE MELLON UNIVERSITY DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL CARNEGIE MELLON UNIVERSITY BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

==========

Copyright (C) 2002 Naval Research Laboratory (NRL/CCS)

Permission to use, copy, modify and distribute this software and its documentation is hereby granted, provided that both the copyright notice and this permission notice appear in all copies of the software, derivative works or modified versions, and any portions thereof.

NRL ALLOWS FREE USE OF THIS SOFTWARE IN ITS "AS IS" CONDITION AND DISCLAIMS ANY LIABILITY OF ANY KIND FOR ANY DAMAGES WHATSOEVER RESULTING FROM THE USE OF THIS SOFTWARE.

```
================================================================================

   Copyright (C) 2022 United States Government as represented by the
   Secretary of the Navy.  All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.

   * Redistributions in binary form must reproduce the above
     copyright notice, this list of conditions and the following
     disclaimer in the documentation and/or other materials provided
     with the distribution.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
   COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
   INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
   (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
   SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
   HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
   STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
   OF THE POSSIBILITY OF SUCH DAMAGE.

================================================================================

   Copyright (C) 1991, 1992, 1994 by Cygnus Support.

   Permission to use, copy, modify, and distribute this software and
   its documentation for any purpose and without fee is hereby
   granted, provided that the above copyright notice appear in all
   copies and that both that copyright notice and this permission
   notice appear in supporting documentation. Cygnus Support makes no
   representations about the suitability of this software for any
   purpose.  It is provided "as is" without express or implied
   warranty.
```

```
================================================================================

   Copyright (C) 2006 Secure Endpoints Inc.

   Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation
   files (the "Software"), to deal in the Software without
   restriction, including without limitation the rights to use, copy,
   modify, merge, publish, distribute, sublicense, and/or sell copies
   of the Software, and to permit persons to whom the Software is
   furnished to do so, subject to the following conditions:

   The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.

   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
   BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
   ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
   SOFTWARE.

================================================================================

Portions of the implementation of the Fortuna-like PRNG are subject to
the following notice:

      Copyright (C) 2005 Marko Kreen
      All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   1. Redistributions of source code must retain the above
      copyright notice, this list of conditions and the following
      disclaimer.

   2. Redistributions in binary form must reproduce the above
      copyright notice, this list of conditions and the following
      disclaimer in the documentation and/or other materials provided
```

with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 1994 by the University of Southern California

EXPORT OF THIS SOFTWARE from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to copy, modify, and distribute this software and its documentation in source and binary forms is hereby granted, provided that any documentation or other materials related to such distribution or use acknowledge that the software was developed by the University of Southern California.

DISCLAIMER OF WARRANTY. THIS SOFTWARE IS PROVIDED "AS IS". The University of Southern California MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. By way of example, but not limitation, the University of Southern California MAKES NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. The University of Southern California shall not be held liable for any liability nor for any direct, indirect, or consequential damages with respect to any claim by the user or distributor of the ksu software.

==========

Copyright (C) 1995
The President and Fellows of Harvard University

This code is derived from software contributed to Harvard by Jeremy Rassen.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

Copyright (C) 2008 by the Massachusetts Institute of Technology.
Copyright 1995 by Richard P. Basch. All Rights Reserved.

Copyright 1995 by Lehman Brothers, Inc. All Rights Reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Richard P. Basch, Lehman Brothers and M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Richard P. Basch, Lehman Brothers and M.I.T. make no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

==========

The following notice applies to "src/lib/krb5/krb/strptime.c" and "src/include/k5-queue.h".

Copyright (C) 1997, 1998 The NetBSD Foundation, Inc.
All rights reserved.

This code was contributed to The NetBSD Foundation by Klaus Klein.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this

software must display the following acknowledgement:

This product includes software developed by the NetBSD Foundation, Inc. and its contributors.

4. Neither the name of The NetBSD Foundation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

The following notice applies to Unicode library files in "src/lib/krb5/unicode":

Copyright 1997, 1998, 1999 Computing Research Labs, New Mexico State University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE COMPUTING RESEARCH LAB OR NEW MEXICO STATE UNIVERSITY BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

==========

The following notice applies to "src/util/support/strlcpy.c":

Copyright (C) 1998 Todd C. Miller "Todd.Miller@courtesan.com"

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

==========

The following notice applies to "src/util/profile/argv_parse.c" and "src/util/profile/argv_parse.h":

Copyright 1999 by Theodore Ts'o.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THE SOFTWARE IS PROVIDED "AS IS" AND THEODORE TS'O (THE AUTHOR) DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,

INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. (Isn't it sick that the U.S. culture of lawsuit-happy lawyers requires this kind of disclaimer?)

==========

The following notice applies to SWIG-generated code in "src/util/profile/profile_tcl.c":

Copyright (C) 1999-2000, The University of Chicago

This file may be freely redistributed without license or fee provided this copyright message remains intact.

==========

The following notice applies to portiions of "src/lib/rpc" and "src/include/gssrpc":

Copyright (C) 2000 The Regents of the University of Michigan. All rights reserved.

Copyright (C) 2000 Dug Song "dugsong@UMICH.EDU". All rights reserved, all wrongs reversed.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived

from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==========

Implementations of the MD4 algorithm are subject to the following notice:

Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD4 Message Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD4 Message Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

==========

Implementations of the MD5 algorithm are subject to the following

notice:

   Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.

   License to copy and use this software is granted provided that it
   is identified as the "RSA Data Security, Inc. MD5 Message- Digest
   Algorithm" in all material mentioning or referencing this software
   or this function.

   License is also granted to make and use derivative works provided
   that such works are identified as "derived from the RSA Data
   Security, Inc. MD5 Message-Digest Algorithm" in all material
   mentioning or referencing the derived work.

   RSA Data Security, Inc. makes no representations concerning either
   the merchantability of this software or the suitability of this
   software for any particular purpose. It is provided "as is"
   without express or implied warranty of any kind.

   These notices must be retained in any copies of any part of this
   documentation and/or software.

==

The following notice applies to
"src/lib/crypto/crypto_tests/t_mddriver.c":

   Copyright (C) 1990-2, RSA Data Security, Inc. Created 1990. All
   rights reserved.

   RSA Data Security, Inc. makes no representations concerning either
   the merchantability of this software or the suitability of this
   software for any particular purpose. It is provided "as is" without
   express or implied warranty of any kind.

   These notices must be retained in any copies of any part of this
   documentation and/or software.

==

Portions of "src/lib/krb5" are subject to the following notice:

      Copyright (C) 1994 CyberSAFE Corporation.

Copyright 1990,1991,2007,2008 by the Massachusetts
Institute of Technology.
All Rights Reserved.

Export of this software from the United States of America may
require a specific license from the United States Government. It
is the responsibility of any person or organization
contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and
distribute this software and its documentation for any purpose and
without fee is hereby granted, provided that the above copyright
notice appear in all copies and that both that copyright notice and
this permission notice appear in supporting documentation, and that
the name of M.I.T. not be used in advertising or publicity
pertaining to distribution of the software without specific,
written prior permission. Furthermore if you modify this software
you must label your software as modified software and not
distribute it in such a fashion that it might be confused with the
original M.I.T. software. Neither M.I.T., the Open Computing
Security Group, nor CyberSAFE Corporation make any representations
about the suitability of this software for any purpose. It is
provided "as is" without express or implied warranty.

==========

Portions contributed by PADL Software are subject to the following
license:

Copyright (c) 2011, PADL Software Pty Ltd. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above
   copyright notice, this list of conditions and the following
   disclaimer.

2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials provided
   with the distribution.

3. Neither the name of PADL Software nor the names of its
   contributors may be used to endorse or promote products derived
   from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY PADL SOFTWARE AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL PADL SOFTWARE
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

==========

The bundled libev source code is subject to the following license:

All files in libev are Copyright (C)2007,2008,2009 Marc Alexander
Lehmann.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above
  copyright notice, this list of conditions and the following
  disclaimer in the documentation and/or other materials provided
  with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Alternatively, the contents of this package may be used under the terms of the GNU General Public License ("GPL") version 2 or any later version, in which case the provisions of the GPL are applicable instead of the above. If you wish to allow the use of your version of this package only under the terms of the GPL and not to allow others to use your version of this file under the BSD license, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL in this and the other files of this package. If you do not delete the provisions above, a recipient may use your version of this file under either the BSD or the GPL.

==========

Files copied from the Intel AESNI Sample Library are subject to the following license:

Copyright (C) 2010, Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

==========

The following notice applies to
"src/ccapi/common/win/OldCC/autolock.hxx":

Copyright (C) 1998 by Danilo Almeida. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above
  copyright notice, this list of conditions and the following
  disclaimer in the documentation and/or other materials provided
  with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

==========

The following notice applies to portions of
"src/plugins/preauth/spake/edwards25519.c" and
"src/plugins/preauth/spake/edwards25519_tables.h":

The MIT License (MIT)

Copyright (c) 2015-2016 the fiat-crypto authors (see the AUTHORS
file).

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

==========

The following notice applies to portions of
"src/plugins/preauth/spake/edwards25519.c":

Copyright (c) 2015-2016, Google Inc.

Permission to use, copy, modify, and/or distribute this software for
any purpose with or without fee is hereby granted, provided that the
above copyright notice and this permission notice appear in all

```
copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR
PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

libfuse

Text

```
The following files may be used under the terms of the GNU Lesser
General Public License, version 2.1 ("LGPL"):

- All files in the include/ directory.
- All files in the lib/ directory.
- meson.build

The full terms of the LGPL can be found in the LGPL2.txt file.


All other files may be used only under the terms of the GNU General
Public License, version 2 ("GPL"). The full text of this license can
be found in the GPL2.txt file.
```

librdkafka

Text

```
LICENSE
--------------------------------------------------------------
librdkafka - Apache Kafka C driver library

Copyright (c) 2012-2022, Magnus Edenhill
              2023, Confluent Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.


LICENSE.cjson
--------------------------------------------------------------
For cJSON.c and cJSON.h:

Copyright (c) 2009-2017 Dave Gamble and cJSON contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
```

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

LICENSE.crc32c

For src/crc32c.c copied (with modifications) from
http://stackoverflow.com/a/17646775/1821055

/* crc32c.c -- compute CRC-32C using the Intel crc32 instruction
 * Copyright (C) 2013 Mark Adler
 * Version 1.1 1 Aug 2013 Mark Adler
 */

/*
  This software is provided 'as-is', without any express or implied
  warranty. In no event will the author be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Mark Adler

```
  madler@alumni.caltech.edu
 */


LICENSE.fnv1a
--------------------------------------------------------------
parts of src/rdfnv1a.c: http://www.isthe.com/chongo/src/fnv/hash_32a.c


Please do not copyright this code.  This code is in the public domain.

LANDON CURT NOLL DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO
EVENT SHALL LANDON CURT NOLL BE LIABLE FOR ANY SPECIAL, INDIRECT OR
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF
USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.

By:
    chongo <Landon Curt Noll> /\oo/\
    http://www.isthe.com/chongo/

Share and Enjoy!    :-)


LICENSE.hdrhistogram
--------------------------------------------------------------
This license covers src/rdhdrhistogram.c which is a C port of
Coda Hale's Golang HdrHistogram https://github.com/codahale/hdrhistogram
at revision 3a0bb77429bd3a61596f5e8a3172445844342120

------------------------------------------------------------------------------

The MIT License (MIT)

Copyright (c) 2014 Coda Hale

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
```

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE

LICENSE.lz4
----
src/rdxxhash.[ch] src/lz4*.[ch]: git@github.com:lz4/lz4.git 5ff839680134437dbf467

LZ4 Library
Copyright (c) 2011-2020, Yann Collet
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this
  list of conditions and the following disclaimer in the documentation and/or
  other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LICENSE.murmur2
---
parts of src/rdmurmur2.c: git@github.com:abrandoned/murmur2.git

MurMurHash2 Library
//---
// MurmurHash2 was written by Austin Appleby, and is placed in the public
// domain. The author hereby disclaims copyright to this source code.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

LICENSE.nanopb

For files in src/nanopb : https://github.com/nanopb/nanopb/blob/8ef41e0ebd45daaf1

Copyright (c) 2011 Petteri Aimonen <jpa at nanopb.mail.kapsi.fi>

This software is provided 'as-is', without any express or
implied warranty. In no event will the authors be held liable
for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any
purpose, including commercial applications, and to alter it and
redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you
   must not claim that you wrote the original software. If you use
   this software in a product, an acknowledgment in the product
   documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and
   must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source
   distribution.

LICENSE.opentelemetry

For files in src/opentelemetry: https://github.com/open-telemetry/opentelemetry-p

                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,

including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of,

publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or

documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly

negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,

```
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.

LICENSE.pycrc
--------------------------------------------------------------
The following license applies to the files rdcrc32.c and rdcrc32.h which
have been generated by the pycrc tool.
==============================================================================

Copyright (c) 2006-2012, Thomas Pircher <tehpeh@gmx.net>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.


LICENSE.queue
--------------------------------------------------------------
For sys/queue.h:

 * Copyright (c) 1991, 1993
 *  The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
```

```
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *  @(#)queue.h 8.5 (Berkeley) 8/20/94
 * $FreeBSD$

LICENSE.regexp
--------------------------------------------------------------
regexp.c and regexp.h from https://github.com/ccxvii/minilibs sha 875c33568b5a4aa

"
These libraries are in the public domain (or the equivalent where that is not pos
"


LICENSE.snappy
--------------------------------------------------------------
######################################################################
# LICENSE.snappy covers files: snappy.c, snappy.h, snappy_compat.h   #
# originally retrieved from http://github.com/andikleen/snappy-c     #
# git revision 8015f2d28739b9a6076ebaa6c53fe27bc238d219              #
######################################################################

The snappy-c code is under the same license as the original snappy source

Copyright 2011 Intel Corporation All Rights Reserved.
```

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
    * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LICENSE.tinycthread

From https://github.com/tinycthread/tinycthread/README.txt c57166cd510ffb5022dd5f

License

Copyright (c) 2012 Marcus Geelnard
              2013-2014 Evan Nemerson

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,

```
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

    1. The origin of this software must not be misrepresented; you must not
    claim that you wrote the original software. If you use this software
    in a product, an acknowledgment in the product documentation would be
    appreciated but is not required.

    2. Altered source versions must be plainly marked as such, and must not be
    misrepresented as being the original software.

    3. This notice may not be removed or altered from any source
    distribution.


LICENSE.wingetopt
--------------------------------------------------------------
For the files wingetopt.c wingetopt.h downloaded from https://github.com/alex85k/

/*
 * Copyright (c) 2002 Todd C. Miller <Todd.Miller@courtesan.com>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 *
 * Sponsored in part by the Defense Advanced Research Projects
 * Agency (DARPA) and Air Force Research Laboratory, Air Force
 * Materiel Command, USAF, under agreement number F39502-99-1-0512.
 */
/*-
 * Copyright (c) 2000 The NetBSD Foundation, Inc.
 * All rights reserved.
 *
 * This code is derived from software contributed to The NetBSD Foundation
```

```
* by Dieter Baron and Thomas Klausner.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in the
*    documentation and/or other materials provided with the distribution.
*
* THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS
* ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
* TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS
* BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*/
```

libuuid

Text

```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, and the entire permission notice in its entirety,
   including the disclaimer of warranties.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ALL OF
WHICH ARE HEREBY DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS SOFTWARE, EVEN IF NOT ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

libuv

Text

```
Copyright (c) 2015-present libuv project contributors.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to
deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE SOFTWARE.
```

llvm

Text

```
==============================================================================
The LLVM Project is under the Apache License v2.0 with LLVM Exceptions:
==============================================================================

                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

    TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

    1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
```

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s)

with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and

may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer,

and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

---- LLVM Exceptions to the Apache 2.0 License ----

As an exception, if, as a result of your compiling your source code, portions of this Software are embedded into an Object form of such source code, you may redistribute such embedded portions in such Object form without complying with the conditions of Sections 4(a), 4(b) and 4(d) of the License.

In addition, if you combine or link compiled forms of this Software with software that is licensed under the GPLv2 ("Combined Software") and if a court of competent jurisdiction determines that the patent provision (Section 3), the indemnity provision (Section 9) or other Section of the License conflicts with the conditions of the GPLv2, you may retroactively and prospectively choose to deem waived or otherwise exclude such Section(s) of the License, but only in their entirety and only with respect to the Combined Software.

==
Software from third parties included in the LLVM Project:
==
The LLVM Project contains third party software which is under different license terms. All such code will be identified clearly using at least one of two mechanisms:
1) It will be in a separate directory tree with its own `LICENSE.txt` or `LICENSE` file at the top containing the specific license and restrictions which apply to that software, or
2) It will contain specific license and restriction terms at the top of every file.

==
Legacy LLVM License (https://llvm.org/docs/DeveloperPolicy.html#legacy):
==
University of Illinois/NCSA
Open Source License

Copyright (c) 2003-2019 University of Illinois at Urbana-Champaign.
All rights reserved.

Developed by:

    LLVM Team

    University of Illinois at Urbana-Champaign

    http://llvm.org

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal with the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies

```
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimers.

    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimers in the
      documentation and/or other materials provided with the distribution.

    * Neither the names of the LLVM Team, University of Illinois at
      Urbana-Champaign, nor the names of its contributors may be used to
      endorse or promote products derived from this Software without specific
      prior written permission.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL THE
CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE
SOFTWARE.
```

ISL (part of LLVM/Polly)

Text

```
MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

imath (part of LLVM/Polly)

Text

```
IMath is Copyright © 2002-2009 Michael J. Fromberger
You may use it subject to the following Licensing Terms:

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

libyaml (part of LLVM)

Text

```
Copyright (c) 2006 Kirill Simonov

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

BLAKE3 (part of LLVM)

Text

This work is released into the public domain with CC0 1.0. Alternatively, it is licensed under the Apache License 2.0.

Creative Commons Legal Code

CC0 1.0 Universal

    CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE
    LEGAL SERVICES. DISTRIBUTION OF THIS DOCUMENT DOES NOT CREATE AN
    ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS
    INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES
    REGARDING THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS
    PROVIDED HEREUNDER, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM
    THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED
    HEREUNDER.

Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer exclusive Copyright and Related Rights (defined below) upon the creator and subsequent owner(s) (each and all, an "owner") of an original work of authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for the purpose of contributing to a commons of creative, cultural and scientific works ("Commons") that the public can reliably and without fear of later claims of infringement build upon, modify, incorporate in other works, reuse and redistribute as freely as possible in any form whatsoever and for any purposes, including without limitation commercial purposes. These owners may contribute to the Commons to promote the ideal of a free culture and the further production of creative, cultural and scientific works, or to gain reputation or greater distribution for their Work in part through the use and efforts of others.

For these and/or other purposes and motivations, and without any expectation of additional consideration or compensation, the person associating CC0 with a Work (the "Affirmer"), to the extent that he or she is an owner of Copyright and Related Rights in the Work, voluntarily elects to apply CC0 to the Work and publicly distribute the Work under its

terms, with knowledge of his or her Copyright and Related Rights in the Work and the meaning and intended legal effect of CCO on those rights.

1. Copyright and Related Rights. A Work made available under CCO may be protected by copyright and related or neighboring rights ("Copyright and Related Rights"). Copyright and Related Rights include, but are not limited to, the following:

  i. the right to reproduce, adapt, distribute, perform, display, communicate, and translate a Work;
 ii. moral rights retained by the original author(s) and/or performer(s);
iii. publicity and privacy rights pertaining to a person's image or likeness depicted in a Work;
 iv. rights protecting against unfair competition in regards to a Work, subject to the limitations in paragraph 4(a), below;
  v. rights protecting the extraction, dissemination, use and reuse of data in a Work;
 vi. database rights (such as those arising under Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, and under any national implementation thereof, including any amended or successor version of such directive); and
vii. other similar, equivalent or corresponding rights throughout the world based on applicable law or treaty, and any national implementations thereof.

2. Waiver. To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

3. Public License Fallback. Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "License"). The License shall be deemed effective as of the date CCO was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's express Statement of Purpose.

4. Limitations and Disclaimers.

a. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.
b. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.
c. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.
d. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CCO or use of the Work.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the

editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with

the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf

of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright 2019 Jack O'Connor and Samuel Neves

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

lz4

Text

```
LZ4 Library
Copyright (c) 2011-2020, Yann Collet
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this
  list of conditions and the following disclaimer in the documentation and/or
  other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

numactl

Text

```
/* Simple NUMA library.
   Copyright (C) 2003,2004,2005,2008 Andi Kleen,SuSE Labs and
   Cliff Wickman,SGI.

   libnuma is free software; you can redistribute it and/or
   modify it under the terms of the GNU Lesser General Public
   License as published by the Free Software Foundation; version
   2.1.

   libnuma is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
   Lesser General Public License for more details.

   You should find a copy of v2.1 of the GNU Lesser General Public License
   somewhere on your Linux system; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

openssl

Text

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object

form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use,

reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only

```
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS
```

pcre2

Text

```
PCRE2 LICENCE
-------------

PCRE2 is a library of functions to support regular expressions whose syntax
and semantics are as close as possible to those of the Perl 5 language.

Releases 10.00 and above of PCRE2 are distributed under the terms of the "BSD"
licence, as specified below, with one exemption for certain binary
redistributions. The documentation for PCRE2, supplied in the "doc" directory,
is distributed under the same terms as the software itself. The data in the
testdata directory is not copyrighted and is in the public domain.

The basic library functions are written in C and are freestanding. Also
included in the distribution is a just-in-time compiler that can be used to
optimize pattern matching. This is an optional feature that can be omitted when
the library is built.


THE BASIC LIBRARY FUNCTIONS
---------------------------

Written by:       Philip Hazel
Email local part: Philip.Hazel
Email domain:     gmail.com

Retired from University of Cambridge Computing Service,
Cambridge, England.

Copyright (c) 1997-2024 University of Cambridge
All rights reserved.


PCRE2 JUST-IN-TIME COMPILATION SUPPORT
--------------------------------------

Written by:       Zoltan Herczeg
Email local part: hzmester
Email domain:     freemail.hu

Copyright(c) 2010-2024 Zoltan Herczeg
All rights reserved.
```

STACK-LESS JUST-IN-TIME COMPILER

Written by: Zoltan Herczeg
Email local part: hzmester
Email domain: freemail.hu

Copyright(c) 2009-2024 Zoltan Herczeg
All rights reserved.

THE "BSD" LICENCE

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notices,
      this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above copyright
      notices, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.

    * Neither the name of the University of Cambridge nor the names of any
      contributors may be used to endorse or promote products derived from this
      software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

```
EXEMPTION FOR BINARY LIBRARY-LIKE PACKAGES
------------------------------------------

The second condition in the BSD licence (covering binary redistributions) does
not apply all the way down a chain of software. If binary package A includes
PCRE2, it must respect the condition, but if package B is software that
includes package A, the condition is not imposed on package B unless it uses
PCRE2 independently.

End
```

pstreams

Text

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

scrypt

Text

```
The included code and documentation ("scrypt") is distributed under the
following terms:

Copyright 2005-2016 Colin Percival.  All rights reserved.
Copyright 2005-2016 Tarsnap Backup Inc.  All rights reserved.
Copyright 2014 Sean Kelly.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

spdk

Text

```
The SPDK repo contains multiple git submodules each with its own
license info.

Submodule license info:
dpdk: see dpdk/license
intel-ipsec-mb: see intel-ipsec-mb/LICENSE
isa-l: see isa-l/LICENSE
libvfio-user: see libvfio-user/LICENSE
ocf: see ocf/LICENSE

The rest of the SPDK repository uses the Open Source BSD-3-Clause
license.  SPDK also uses SPDX Unique License Identifiers to eliminate
the need to copy the license text into each individual file.

Any new file contributions to SPDK shall adhere to the BSD-3-Clause
license and use SPDX identifiers. Exceptions are subject to usual
review and must be listed in this file.

Exceptions:

* include/linux/* header files are BSD-3-Clause but do not use SPDX
  identifier to keep them identical to the same header files in the
  Linux kernel source tree.

* include/spdk/tree.h and include/spdk/queue_extras are BSD-2-Clause,
  since there were primarily imported from FreeBSD.  tree.h uses an SPDX
  identifier but also the license text to reduce differences from the
  FreeBSD source tree.

* lib/util/base64_neon.c is BSD-2-Clause.

* lib/mlx5/mlx5_ifc.h is BSD-2-Clause .
```

strtk

Text

```
[COPYRIGHT NOTICE]
Free use of the String Toolkit Library is permitted under the
guidelines and in accordance with the most current version of the
MIT License.
http://www.opensource.org/licenses/MIT
```

tbb

Text

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the

editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with

the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf

```
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

tclap

Text

```
Copyright (c) 2003-2012 Michael E. Smoot
Copyright (c) 2004-2016 Daniel Aarno
Copyright (c) 2017-2021 Google LLC

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN
AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR
IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

libudev, part of systemd

Text

```
/***
  This file is part of systemd.

  Copyright 2008-2012 Kay Sievers <kay@vrfy.org>

  systemd is free software; you can redistribute it and/or modify it
  under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation; either version 2.1 of the License, or
  (at your option) any later version.

  systemd is distributed in the hope that it will be useful, but
  WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public License
  along with systemd; If not, see <http://www.gnu.org/licenses/>.
***/
```

xxhash

Text

```
xxHash Library
Copyright (c) 2012-2021 Yann Collet
All rights reserved.

BSD 2-Clause License (https://www.opensource.org/licenses/bsd-license.php)

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this
  list of conditions and the following disclaimer in the documentation and/or
  other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

yaml-cpp

Text

```
Copyright (c) 2008-2015 Jesse Beder.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

zlib

Text

```
Copyright notice:

 (C) 1995-2022 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly        Mark Adler
  jloup@gzip.org          madler@alumni.caltech.edu
```

zstd

Text

BSD License

For Zstandard software

Copyright (c) Meta Platforms, Inc. and affiliates. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name Facebook, nor Meta, nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

© 2026 Ocient All Rights Reserved Terms Of Use Privacy Statement

User Documentation

Click below to download the latest PDFs for the Ocient user documentation, or **click here** for online access.

Version 25.2

DOWNLOAD PDF

Version 26.1

DOWNLOAD PDF

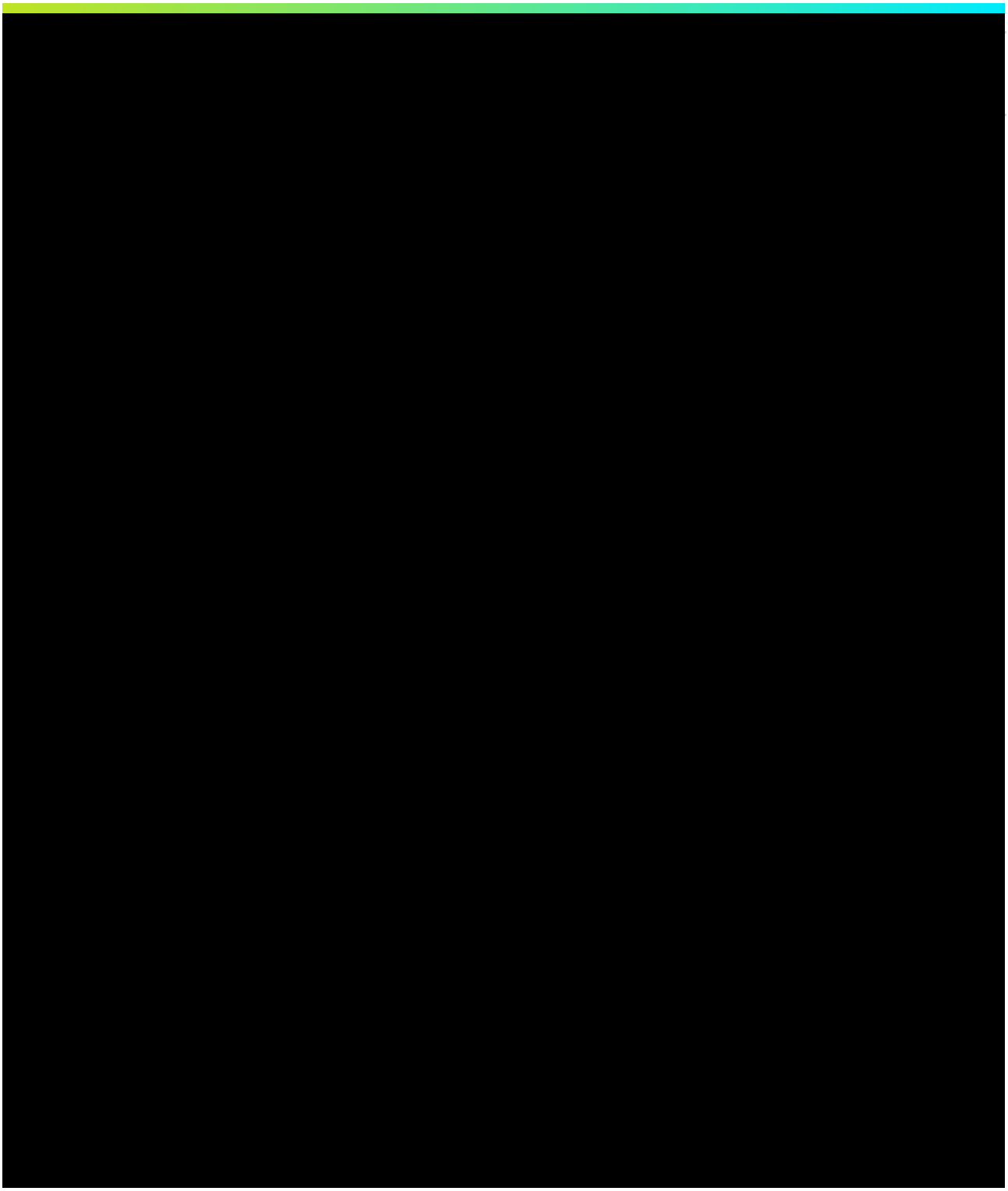
Version 27

DOWNLOAD PDF

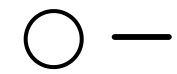
DOWNLOAD MARKDOWN

Version 27.1

DOWNLOAD MARKDOWN



© 2026 Ocient, Inc.



© 2026 Ocient All Rights Reserved

Terms Of Use **Privacy Statement** Cookie Settings