Data Lake Insight

Delta SQL Syntax Reference

Issue 01

Date 2025-01-09

HUAWEI TECHNOLOGIES CO., LTD.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 DLI Delta Table Overview

Delta tables use the Delta Lake technology to offer a robust data storage solution. By extending Parquet data files with file-based transaction logs, they support ACID transactions and scalable metadata. Delta Lake seamlessly integrates with Apache Spark APIs and is designed to work closely with structured streaming, allowing for the use of a single data backup for both batch and streaming operations, and enabling large-scale incremental processing.

Constraints on Delta in DLI

- Only Spark 3.3.1 (3.0.0) or later supports Delta.
- Only Delta 2.3.0 is supported by DLI.
- Certain SQL statements in Spark 3.3.1 (3.0.0) do not support the open-source syntax related to Delta tables. For details, see **Table 1-1**.

Table 1-1 Open source syntax related to Delta tables not supported by Spark 3.3.1-3.0.0 SQL

Unsupported Statement	Example
ALTER TABLE REPLACE COLUMNS: replaces columns	alter table table0 replace columns(id1 int,name1 string);
SHOW CREATE TABLE: shows table creation statements	show create table table1;
INSERT INTO/OVERWRITE: inserts data into a table in a specified static partition	insert into table1 partition(part='part1') select * from table2;
ALTER TABLE ADD/DROP PARTITION: manages partitions	alter table test_delta_parts1 add partition('2024-10-28');
CONVERT TO DELTA: does not support tables in **parquet.'tablePath'** format	convert to delta parquet.`obs://bucket0/db0/table0`;

2 Using Delta to Develop Jobs in DLI

2.1 DLI Delta Metadata

- For how to submit a Spark SQL job in DLI using the Delta SQL syntax, see **Delta SQL Syntax Reference**.
- For how to submit a Spark Jar job in DLI using Delta, see **Using Delta to Submit a Spark Jar Job in DLI**.

DLI Delta Metadata Description

When creating a Delta table, relevant metadata information about the table will be created in the metadata warehouse.

Delta supports integration with DLI metadata and LakeFormation metadata (integration with LakeFormation metadata is only supported in Spark 3.3.1 or later), using the same integration method as Spark.

- DLI metadata can be viewed on the **Data Management** > **Databases and Tables** page of the DLI management console.
- LakeFormation metadata can be viewed on the LakeFormation management console.

Related Operations

- To connect a DLI SQL queue to DLI metadata, follow these steps:
  a. On the **SQL Editor** page of the DLI management console, click **dli** in the **Catalog** list.
  b. In the **Databases** drop-down list, select the database in the DLI metadata to connect.
- To connect a DLI general-purpose queue to DLI metadata, follow these steps:
  See **Using Spark Jobs to Access DLI Metadata**.
- To connect a DLI SQL queue to LakeFormation metadata, follow these steps:
  See **Connecting DLI to LakeFormation**.
- To connect a DLI general-purpose queue to LakeFormation metadata, follow these steps:

See **Connecting DLI to LakeFormation**.

- DLI metadata permission management

  You can manage the permissions for DLI metadata through DLI SQL permissions management or IAM authentication.

  – DLI SQL permission management:

    i. Log in to the DLI console. In the navigation pane on the left, choose **Data Management** > **Databases and Tables**. On the displayed page, search for the database you want to authorize or click the name of a database and search for the table you want to authorize.

    ii. Locate the database or table you want to authorize and click **Permissions** in the **Operation** column to view or add permissions.

       See **Managing Database Resources on the DLI Console**.

  – IAM authentication:

    See "Application Scenarios of IAM Authentication" in **Permissions Management**.

- LakeFormation metadata permissions management

  See **Connecting DLI to LakeFormation**.

2.2 Using Delta to Submit a Spark Jar Job in DLI

1. Add the following dependencies:

```
<dependency>
    <groupId>io.delta</groupId>
    <artifactId>delta-core_2.12</artifactId>
    <version>2.3.0</version>
</dependency>
```

2. Add the following parameters to SparkSession:

```
.config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
.config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")
```

3. Write code (through SQL or API).

(1) SQL development example is as follows. For specific SQL syntax, refer to "Delta SQL Syntax Reference".

```
public static void main( String[] args )
{
    SparkSession spark = SparkSession
            .builder()
            .enableHiveSupport()
            .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")
            .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
            .appName("DeltaDemo")
            .getOrCreate();
    String sql_create = "CREATE TABLE if not exists dligms.deltaTest1012 (\n" +
            "  id int,\n" +
            "  name string,\n" +
            "  start_year int,\n" +
            "  class string\n" +
            ") USING DELTA\n" +
            " partitioned by(start_year, class)\n" +
            " location 'obs://bucket_name/path/deltaTest1012'";
    spark.sql(sql_create);
    String sql_insert = "insert into dligms.deltaTest1012 values\n" +
            "(1, 'zhangsan', 2024, 'whlg0905')," +
            "(2, 'lisi', 2024, 'whlg0905')," +
            "(3, 'wangwu', 2024, 'whlg0905')," +
            "(4, 'zhaoliu', 2024, 'whlg0905')";
    spark.sql(sql_insert);
    String sql_select = "select * from dligms.deltaTest1012";
    spark.sql(sql_select).show();
    spark.stop();
}
```

(2) API development example is as follows (Java).

```
public static void main( String[] args )
{
    SparkSession spark = SparkSession
            .builder()
            .enableHiveSupport()
            .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")
            .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
            .appName("DeltaDemo")
            .getOrCreate();
    DeltaTable.createIfNotExists(spark)
            .tableName("deltaJava1011")
            .addColumn("id", "INT")
            .addColumn("name", "STRING")
            .addColumn("start_year", "INT")
            .addColumn("class", "STRING")
            .partitionedBy("start_year", "class")
            .location("obs://bucket_name/path/deltaTest1011")
            .execute();
    Dataset<Row> data = spark.read().format("csv")
            .option("header", "true")
            .option("inferSchema", "true")
            .load("obs://bucket_name/path/export/test1011/");
    data.write().insertInto("deltaJava1011");
    spark.stop();
}
```

4. Compile and package the code, and then run the job.

3 Delta Time Travel

3.1 Viewing History Operation Records of a Delta Table

Syntax

DESCRIBE HISTORY *[database_name.]table_name|DELTA.`obs_path`* [LIMIT n]

Example

```
DESCRIBE HISTORY delta_table0;
DESCRIBE HISTORY delta.`obs://tablePath` LIMIT 1;
```

System Response

Returns the table's historical operations, with the meaning of the result indicators shown in the following table.

Table 3-1 Result metrics

Metric	Description
version	Version number of the table operation
timestamp	Timestamp of the current version operation
userId	User ID of the current version operation
userName	Username of the current version operation
operation	Operation name (e.g., WRITE, CREATE TABLE, UPDATE, DELETE, MERGE, RESTORE)
operationParameters	Operation parameters
job	Detailed information about the job running the operation
notebook	Detailed information about the notebook running the operation

Metric	Description
clusterId	Cluster ID
readVersion	Table version read for the write operation
isolationLevel	Isolation level
isBlindAppend	Whether data is appended
operationMetrics	Metrics of the operation (e.g., number of files modified, rows modified, bytes modified)
engineInfo	Information on Spark and Delta versions

3.2 Querying History Version Data of a Delta Table

Syntax

Query the state of a Delta table at a specific point in time:

SELECT * FROM *[database_name.]table_name*

TIMESTAMP AS OF timestamp_expression

Query the state of a Delta table at a specific historical version:

SELECT * FROM *[database_name.]table_name* VERSION AS OF version_code

Parameter Description

Table 3-2 Parameters for querying historical versions of a Delta table

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
timestamp_expression	Timestamp, which cannot be later than the current time, formatted as **yyyy-MM-ddTHH:mm:ss.SSS**
version_code	Version number in the query result in **Viewing History Operation Records of a Delta Table**

Example

```
SELECT * FROM delta_table0 TIMESTAMP AS OF '2020-10-18T22:15:12.013Z';
SELECT * FROM delta_table0 VERSION AS OF 2 where part_col='part_value';
```

3.3 Restoring a Delta Table to an Earlier State

Syntax

Restore a Delta table to the state at a specific point in time:

RESTORE [TABLE] [database_name.]table_name|DELTA.`obs_path`

[TO] TIMESTAMP AS OF timestamp_expression

Restore a Delta table to the state at a specific historical version:

RESTORE [TABLE] [database_name.]table_name|DELTA.`obs_path`

[TO] VERSION AS OF version_code

Parameter Description

Table 3-3 Parameters for restoring a Delta table

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
obs_path	OBS path, indicating the storage location of the Delta table
timestamp_expression	Timestamp, which cannot be later than the current time, formatted as **yyyy-MM-ddTHH:mm:ss.SSS**
version_code	Version number in the query result in **Viewing History Operation Records of a Delta Table**

Example

```
RESTORE delta_table0 TO TIMESTAMP AS OF '2020-10-18T22:15:12.013Z';
RESTORE delta.`obs://bucket_name/db0/delta_table0` VERSION AS OF 2;
```

4 Delta Cleansing and Optimization

Cleansing a Delta Table

You can run the **VACUUM** command on a Delta table to remove data files that are no longer referenced and were created before the retention threshold.

```
VACUUM delta_table0;
VACUUM delta_table0 RETAIN 168 HOURS;--The unit can only be HOURS.
```

Optimizing a Delta Table

To improve query speed, Delta Lake supports optimizing the data layout in storage, which will compress many smaller files into larger ones.

```
optimize delta_table0;
optimize delta_table0 where date >= '2020-01-01';
```

Z-Ordering

Z-ordering is another technique to speed up queries. Sorting data with Z-order can reorganize the data in storage. When your data is appropriately sorted, you can skip more files and read less data, thus running faster. To sort Z-Order data, specify the columns to sort by in **ZORDER BY**.

```
OPTIMIZE delta_table0 ZORDER BY (price);
```

5 Delta SQL Syntax Reference

5.1 Delta DDL Syntax

5.1.1 CREATE TABLE

Function

This command creates a Delta table by specifying a list of fields with table attributes.

Precautions

- In this command, **IF EXISTS** and **db_name** are optional.
- In DLI, Delta only supports OBS foreign tables. Creating a table using a table name without specifying the location will fail.

Syntax

- Create a Delta table by table name.

  CREATE[OR REPLACE] TABLE *[IF NOT EXISTS] [database_name.]table_name*

  [(columnTypeList)]

  USING DELTA

  [COMMENT table_comment]

  [PARTITIONED BY (partColumnList)]

  LOCATION location_path

- Create a Delta table by **delta.'Obs path'**.

  CREATE[OR REPLACE] TABLE [IF NOT EXISTS] DELTA.`obs://*bucket_name/tbl_path*`

  [(columnTypeList)]

  USING DELTA

  [COMMENT table_comment]

[PARTITIONED BY (partColumnList)]

NOTE

Creating a table by table name allows the table to be found with **show tables**. In the current version, the location must be specified and can only be set to an OBS path. Creating a table by **delta.'Obs path'** means the table cannot be found with **show tables**.

Parameter Description

Table 5-1 Parameter descriptions of CREATE TABLE

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	Name of the OBS bucket
tbl_path	Storage location of the Delta table in the OBS bucket
columnTypeList	Comma-separated list with data types. Column names consist of letters, numbers, and underscores (_).
using	Parameter **delta**, defines and creates the Delta table
table_comment	Description of the table
partColumnList	List of partition fields, with column names from **columnTypeList**
location_path	Storage location of the Delta table. In the current version, it must be specified and can only support OBS paths. Specifying this path creates the Delta table as a foreign table.

Required Permissions

- SQL permissions

Table 5-2 Permissions required for executing CREATE TABLE

Permission Description
CREATE_TABLE permission on the database

- Fine-grained permission: **dli:database:createTable**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

- **Create a non-partitioned table by table name.**
```
create table if not exists delta_table0 (
id int,
name string,
addr struct<priv:string,city:string>,
price double
) using delta
location 'obs://bucket_name0/db0/delta_table0';
```
- **Create a partitioned table by table name.**
```
create table if not exists delta_table0 (
id bigint,
name string,
ts bigint,
dt string,
hh string
) using delta
partitioned by (dt, hh)
location 'obs://bucket_name0/db0/delta_table0';
```
- **Create a non-partitioned table by delta path.**
```
create table if not exists delta.`obs://bucket_name0/db0/delta_table0`(
id int,
name string,
price double
) using delta;
```
- **Create a partitioned table by delta path.**
```
create table if not exists delta.`obs://bucket_name0/db0/delta_table0` (
id bigint,
name string,
ts bigint,
dt string,
hh string
) using delta
partitioned by (dt, hh);
```
- **Create a table using create table like.**
```
create table delta_table2 like delta_table1
using delta
location 'obs://bucket_name0/db0/delta_table2';
```

System Response

Displays the successful creation of the table.

5.1.2 DROP TABLE

Function

This command deletes an existing table.

Syntax

DROP TABLE *[IF EXISTS] [db_name.]table_name;*

Parameter Description

Table 5-3 Parameter descriptions of DROP TABLE

Parameter	Description
db_name	Database name. If not specified, the current database is selected.
table_name	Name of the table to be deleted.

Required Permissions

- SQL permissions

  Table 5-4 Permissions required for executing DROP TABLE

Permission Description
DROP_TABLE permission on the database where the table is

- Fine-grained permission: **dli:table:dropTable**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

DROP TABLE IF EXISTS db0.delta_table0;

System Response

The table is successfully deleted from the metadata and cannot be found using **show** and **describe**.

5.1.3 DESCRIBE

Function

This command shows detailed information or statistics about a table.

Syntax

Show statistics about a table:

DESCRIBE *[EXTENDED|FORMATTED] [database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`;*

Show detailed information about a table:

DESCRIBE *DETAIL [database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`;*

Required Permissions

- SQL permissions

Table 5-5 Permissions required for executing DESCRIBE

Permission Description
DESCRIBE_TABLE permission on a table

- Fine-grained permission: **dli:table:describeTable**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

```
DESCRIBE FORMATTED delta_table0;

DESCRIBE FORMATTED delta.`obs://bucket_name0/db0/delta_table0`;

DESCRIBE DETAIL delta_table0;
```

System Response

Returns details or statistics about a table.

Table 5-6 Result parameters

Parameter	Description
format	Format of the table, which is delta in this example
id	Unique ID of the table
name	Table name defined in MetaServer
description	Description about the table
location	Storage path of the table
createdAt	Timestamp when the table was created
lastModified	Timestamp when the table was last modified
partitionColumns	Partition column
numFiles	Number of files in the latest version of the table
sizeInBytes	Size of the latest snapshot of the table, in bytes
properties	All the attributes set for this table
minReaderVersion	Earliest reader version that can read the table

Parameter	Description
minWriterVersion	Earliest writer version that can write the table

5.1.4 ADD CONSTRAINT

Function

This command adds a CHECK constraint. Before such a constraint is added to a table, the system verifies whether all existing rows meet the constraint.

Precautions

Before adding a constraint to a table, the system verifies whether all existing rows meet the constraint. If any row fails to meet the constraint, the system will not add the constraint. Therefore, you need to delete the rows that do not meet the constraint before adding it.

Syntax

ALTER TABLE *[database_name.]table_name*|*DELTA.`obs://bucket_name/tbl_path`*

ADD CONSTRAINT constraint_name

CHECK(*boolExpression*)

Parameter Description

Table 5-7 Parameter descriptions of ADD CONSTRAINT

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
constraint_name	Constraint name
boolExpression	Constraint expression

Required Permissions

- SQL permissions

Table 5-8 Permissions required for executing ADD CONSTRAINT

Permission Description
ALTER permission on a table

- Fine-grained permission: **dli:table:alter**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
alter table delta_table0 add constraint const_price check(price>0);
alter table delta.`obs://bucket1/dbgms/h0` add constraint const_id check(id>0);
```

System Response

Displays whether the task is successfully executed in the execution history or job list.

5.1.5 DROP CONSTRAINT

Function

This command deletes a CHECK constraint.

Syntax

ALTER TABLE *[database_name.]table_name*|*DELTA.`obs://bucket_name/tbl_path`*

DROP CONSTRAINT constraint_name

Parameter Description

Table 5-9 Parameter descriptions of DROP CONSTRAINT

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
constraint_name	Constraint name

Required Permissions

- SQL permissions

Table 5-10 Permissions required for executing DROP CONSTRAINT

Permission Description
ALTER permission on a table

- Fine-grained permission: **dli:table:alter**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
alter table delta_table0 drop constraint const_price;

alter table delta.`obs://bucket1/dbgms/h0` drop constraint const_id;
```

System Response

Displays whether the task is successfully executed in the execution history or job list.

5.1.6 CONVERT TO DELTA

Function

This command converts an existing Parquet table to a Delta table in-place. This command lists all the files in the directory, creates a Delta Lake transaction log to track these files, and automatically infers the data schema by reading the footer of all Parquet files. The conversion process collects statistics to improve the query performance of the resulting Delta table. If a table name is provided, the metastore will also be updated to reflect that the table is now a Delta table.

Precautions

To convert partitioned tables, set **spark.sql.forcePartitionPredicatesOnPartitionedTable.enabled** to **false**.

Syntax

CONVERT TO DELTA *[database_name.]table_name* [NO STATISTICS]

Parameter Description

Table 5-11 Parameter descriptions of CONVERT TO DELTA

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
NO STATISTICS	The collection of statistics is bypassed during the conversion process to achieve faster conversion.

Required Permissions

- SQL permissions

Table 5-12 Permissions required for executing CONVERT TO DELTA

Permission Description
CREATE_TABLE permission on the database where the table is
ALTER permission on a table
INSERT_INTO_TABLE permission on a table
DROP_TABLE permission on a table

- Fine-grained permissions: **dli:database:createTable**, **dli:table:alter**, **dli:table:insertIntoTable**, and **dli:table:dropTable**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
create table if not exists parquet_table0 (id int,name string,price double) using parquet location 'obs://bucket_name0/db0/parquet_table0';
convert to delta parquet_table0;
```

System Response

Displays successful execution.

5.1.7 SHALLOW CLONE

Function

This command creates a shallow copy of an existing Delta table in a specific version. The cloned information includes the schema, partition information, and data file paths.

Any changes made to the cloned table will only affect the clone itself and not the source table, as long as they do not touch the source data. Note that the cloned table may still point to the data files of the source table, which may cause issues if the source table undergoes a vacuum operation and the files are not found for the clone.

Syntax

CREATE TABLE *[target_db.]target_table*

SHALLOW CLONE *[source_db.]source_table|DELTA.`obs://bucket_name/tbl_path`*

location 'obs://bucket_name/tbl_path'

Parameter Description

Table 5-13 Parameter descriptions of SHALLOW CLONE

Parameter	Description
target_db	Name of the target database, consisting of letters, numbers, and underscores (_)
target_table	Name of the target table, consisting of letters, numbers, and underscores (_)
source_db	Name of the source database, consisting of letters, numbers, and underscores (_)
source_table	Name of the source table, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
constraint_name	Constraint name
boolExpression	Constraint expression

Required Permissions

- SQL permissions

Table 5-14 Permissions required for executing SHALLOW CLONE

Permission Description
CREATE_TABLE permission on the target database
SELECT permission on the source table

- Fine-grained permissions: **dli:database:createTable** and **dli:table:select**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
CREATE OR REPLACE TABLE delta_table1 SHALLOW CLONE delta_table0 LOCATION 'obs://bucket0/db0/
delta_table1';
CREATE TABLE delta.`obs://bucket0/db0/delta_table1` SHALLOW CLONE delta_table0 VERSION AS OF 10;
```

System Response

Displays whether the task is successfully executed in the execution history or job list.

5.2 Delta DML Syntax

5.2.1 INSERT

Function

This command inserts **SELECT** query results to a Delta table.

Syntax

Append mode:

INSERT INTO *[database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`*

select query;

Overwrite mode:

INSERT OVERWRITE *[database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`*

select query;

Parameter Description

Table 5-15 Parameter descriptions of INSERT INTO

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
select query	Query statement

Required Permissions

- SQL permissions

Table 5-16 Permissions required for executing INSERT INTO

Permission Description
INSERT_INTO_TABLE permission on a table

- Fine-grained permission: **dli:table:insertIntoTable**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
insert into delta_table0 values(1, 'a1', 20);

insert into delta_table0 select 1, 'a1', 20;

insert into test_delta_parts1 PARTITION(dt) select id,name,dt from test_delta1;

-- insert overwrite table
insert overwrite table delta_table0 select 1, 'a1', 20;
```

System Response

Displays whether the task is successfully executed in the execution history or job list.

5.2.2 CREATE TABLE AS SELECT

Function

This command creates a Hudi table by specifying a list of fields with table attributes.

Syntax

CREATE[OR REPLACE] TABLE *[IF NOT EXISTS] [database_name.]table_name| DELTA.`obs://bucket_name/tbl_path`*

USING DELTA

[COMMENT table_comment]

[PARTITIONED BY (partColumnList)]

[LOCATION location_path]

[AS query_statement]

Parameter Description

Table 5-17 Parameter descriptions of CREATE TABLE AS SELECT

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
using	Parameter **delta**, defines and creates the Delta table
table_comment	Description of the table
location_path	Storage location of the Delta table. In the current version, it must be specified when a Delta table is created based on the table name and can only support OBS paths. Specifying this path creates the Delta table as a foreign table.
query_statement	SELECT statement

Required Permissions

- SQL permissions

Table 5-18 Permissions required for executing CREATE TABLE AS SELECT

Permission Description
CREATE_TABLE permission on the database
SELECT permission to query a table

- Fine-grained permissions: **dli:database:createTable** and **dli:table:select**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

- Create a partitioned table.
```
create table if not exists delta_table0
using delta
partitioned by (dt)
location 'obs://bucket_name0/db0/delta_table0'
as
select 1 as id, 'a1' as name, 10 as price, 1000 as dt;
```
- Create a non-partitioned table.
```
create table if not exists delta_table0
using delta
location 'obs://bucket_name0/db0/delta_table0'
as
select 1 as id, 'a1' as name, 10 as price;

create table delta.`obs://bucket_name0/db0/delta_table0`
using delta
partitioned by (part_col1, part_col2)
as select id,name,year,class_name from table1 where part_col1=2024;
```

5.2.3 MERGE INTO

Function

This command is used to query another table based on the join condition of a table or subquery. If **UPDATE** or **DELETE** is executed for the table matching the join condition, and **INSERT** is executed if the join condition is not met. This command completes the synchronization requiring only one full table scan, delivering higher efficiency than **INSERT** plus **UPDATE**.

Precautions

To merge partitioned tables, set **spark.sql.forcePartitionPredicatesOnPartitionedTable.enabled** to **false**.

Syntax

MERGE INTO *[database_name.]table_name AS target_alias*

USING *(sub_query | tableIdentifier) AS source_alias*

ON *<merge_condition>*

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN MATCHED [AND <condition>] THEN <matched_action>]

[WHEN NOT MATCHED [AND <condition>] THEN <not_matched_action>]

<merge_condition> =A equal bool condition

<matched_action> =

DELETE |

*UPDATE SET * |*

UPDATE SET column1 = expression1 [, column2 = expression2 ...]

<not_matched_action> =

*INSERT * |*

INSERT (column1 [, column2 ...]) VALUES (value1 [, value2 ...])

Parameter Description

Table 5-19 Parameter descriptions of MERGE INTO

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
target_alias	Alias of the target table
sub_query	Subquery.
source_alias	Alias of the source table or source expression
merge_condition	Condition for associating the source table or expression with the target table.
condition	Filtering condition. This parameter is optional.
matched_action	DELETE or UPDATE operation to be performed when conditions are met.
not_matched_action	INSERT operation to be performed when conditions are not met.

Required Permissions

- SQL permissions

Table 5-20 Permissions required for executing MERGE INTO

Permission Description
UPDATE permission on a table
ALTER permission on a table
DELETE permission on a table

- Fine-grained permissions: **dli:table:update**, **dli:table:insertIntoTable**, and **dli:table:delete**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

- Update some fields.

```
create table h0(id int, comb int, name string, price int) using delta location 'obs://bucket_name0/db0/
h0';
create table s0(id int, comb int, name string, price int) using delta location 'obs://bucket_name0/db0/
s0';
insert into h0 values(1, 1, 'h1', 1);
insert into s0 values(1, 1, 's1', 1),(2, 2, 's2', 2);
// Method 1
merge into h0 using s0
on h0.id = s0.id
when matched then update set h0.id = s0.id, h0.comb = s0.comb, price = s0.price * 2;
// Method 2
merge into h0 using s0
on h0.id = s0.id
when matched then update set id = s0.id,
name = s0.name,
comb = s0.comb + h0.comb,
price = s0.price + h0.price;
```

- Update and insert default fields.

```
create table h0(id int, comb int, name string, price int, flag boolean) using delta location 'obs://
bucket_name0/db0/h0';
create table s0(id int, comb int, name string, price int, flag boolean) using delta location 'obs://
bucket_name0/db0/s0';
insert into h0 values(1, 1, 'h1', 1, false);
insert into s0 values(1, 2, 's1', 1, true);
insert into s0 values(2, 2, 's2', 2, false);

merge into h0 as target
using (
select id, comb, name, price, flag from s0
) source
on target.id = source.id
when matched then update set *
when not matched then insert *;
```

- Update and delete data based on multiple conditions.

```
create table h0(id int, comb int, name string, price int, flag boolean) using delta location 'obs://
bucket_name0/db0/h0';
create table s0(id int, comb int, name string, price int, flag boolean) using delta location 'obs://
bucket_name0/db0/s0';
insert into h0 values(1, 1, 'h1', 1, false);
insert into h0 values(2, 2, 'h2', 1, false);
insert into s0 values(1, 1, 's1', 1, true);
insert into s0 values(2, 2, 's2', 2, false);
insert into s0 values(3, 3, 's3', 3, false);

merge into h0
using (
select id, comb, name, price, flag from s0
) source
on h0.id = source.id
when matched and h0.flag = false then update set id = source.id, comb = h0.comb + source.comb,
price = source.price * 2
when matched and h0.flag = true then delete
when not matched then insert *;
```

System Response

You can view the result in driver logs or on the client.

5.2.4 UPDATE

Function

This command updates a Delta table based on column expressions and optional filter conditions.

Syntax

UPDATE *[database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`*

SET column = EXPRESSION(,column = EXPRESSION)

[WHERE boolExpression]

Parameter Description

Table 5-21 Parameter descriptions of UPDATE

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
column	Target column to be updated
EXPRESSION	Expression of the source table column to be updated in the target table
boolExpression	Filter condition expression

Required Permissions

- SQL permissions

Table 5-22 Permissions required for executing UPDATE

Permission Description
UPDATE permission on a table

- Fine-grained permission: **dli:table:update**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

```
update delta_table0 set price = price + 20 where id = 1;

update delta.`obs://bucket0/db0/delta_table1` set price = price *2, name = 'a2' where part0='xx' and id = 2;
```

System Response

You can view the result in driver logs or on the client.

5.2.5 DELETE

Function

This command deletes records from a Delta table.

Syntax

DELETE from *[database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`*

[WHERE boolExpression]

Parameter Description

Table 5-23 Parameter descriptions of DELETE

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket

Parameter	Description
boolExpression	Filter conditions for deleting records

Required Permissions

- SQL permissions

Table 5-24 Permissions required for executing DELETE

Permission Description
DELETE permission on a table

- Fine-grained permission: **dli:table:delete**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

```
delete from delta_table0 where column1 = 'value1';

delete from delta_table0 where column1 IN ('value1', 'value2');

delete from delta.`obs://bucket_name0/db0/delta_table0` where column1 = 'value1';
```

System Response

You can view the result in driver logs or on the client.

5.2.6 VACUUM

Function

This command removes all files that are not managed by Delta from the table directory and to delete data files that are no longer in the latest state of the table transaction log and exceed the retention threshold. The default threshold is 7 days.

Precautions

RETAIN num HOURS indicates the retention threshold, which is recommended to be at least 7 days.

If you run **VACUUM** on a Delta table, you will no longer be able to view versions created before the specified data retention period.

Delta Lake has a safety check to prevent running dangerous **VACUUM** commands, which will report an error when the specified retention threshold is less than 168 hours. If you determine the retention threshold for the vacuum operation, you can disable this safety check by setting the Spark configuration property **spark.databricks.delta.retentionDurationCheck.enabled** to **false**.

Syntax

VACUUM *[database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`* [RETAIN num HOURS];

You can simulate the execution of the vacuum operation using the **DRY RUN** parameter, which returns a list of files that the vacuum will delete:

VACUUM *[database_name.]table_name|DELTA.`obs://bucket_name/tbl_path`* [RETAIN num HOURS] [DRY RUN];

Parameter Description

Table 5-25 Parameter descriptions of VACUUM

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
bucket_name	OBS bucket name
tbl_path	Storage location of the Delta table in the OBS bucket
num	Retention period

Required Permissions

- SQL permissions

Table 5-26 Permissions required for executing VACUUM

Permission Description
UPDATE permission on a table

- Fine-grained permission: **dli:table:update**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

```
VACUUM delta_table0 RETAIN 168 HOURS;

VACUUM delta_table0 RETAIN 48 HOURS DRY RUN;
VACUUM delta.`obs://bucket_name0/db0/delta_table0` RETAIN 168 HOURS;
```

System Response

You can view the result in driver logs or on the client.

5.2.7 RESTORE

Function

This command restores a Delta table to an earlier state, supporting restoration to a previous version number or timestamp.

Syntax

Restore a Delta table to the state at a specific point in time:

RESTORE [TABLE] [database_name.]table_name|DELTA.`obs_path`

[TO] TIMESTAMP AS OF timestamp_expression

Restore a Delta table to the state at a specific historical version:

RESTORE [TABLE] [database_name.]table_name|DELTA.`obs_path`

[TO] VERSION AS OF version_code

Parameter Description

Table 5-27 Parameters

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
obs_path	OBS path, indicating the storage location of the Delta table
timestamp_expression	Timestamp, which cannot be later than the current time, formatted as **yyyy-MM-ddTHH:mm:ss.SSS**
version_code	Version number in the query result in version 1.3.1

Required Permissions

- SQL permissions

Table 5-28 Permissions required for executing RESTORE

Permission Description
UPDATE permission on a table

- Fine-grained permission: **dli:table:update**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
RESTORE delta_table0 TO TIMESTAMP AS OF '2020-10-18 22:15:12.013';
RESTORE delta.`obs://bucket_name/db0/delta_table0` VERSION AS OF 2;
```

System Response

You can view command execution results in the driver log or on the client.

5.2.8 OPTIMIZE

Function

This command optimizes the data layout in storage to improve query speed.

Precautions

- Since optimization is time-consuming, you need to determine the frequency of running **OPTIMIZE** based on the trade-off between better end-user query performance and the optimization computation time.
- To optimize partitioned tables, set **spark.sql.forcePartitionPredicatesOnPartitionedTable.enabled** to **false**.

Syntax

OPTIMIZE [database_name.]table_name

[WHERE boolExpression]

Z-ordering:

OPTIMIZE [database_name.]table_name

[WHERE boolExpression]

ZORDER BY (*columnList*);

Parameter Description

Table 5-29 Parameter descriptions

Parameter	Description
database_name	Name of the database, consisting of letters, numbers, and underscores (_)
table_name	Name of the table in the database, consisting of letters, numbers, and underscores (_)
boolExpression	Filter condition expression
columnList	List of fields specified for z-ordering, and the Z-order columns should be different from the partition columns.

Required Permissions

- SQL permissions

Table 5-30 Permissions required for executing OPTIMIZE

Permission Description
UPDATE permission on a table

- Fine-grained permission: **dli:table:update**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Examples

```
OPTIMIZE delta_table0;

optimize delta_table0 where dt >= '2020-01-01';

OPTIMIZE delta_table0 WHERE dt >= current_timestamp() - INTERVAL 1 day ZORDER BY (price);
```

System Response

Display the success or failure of the command execution in the driver logs and the client.

5.3 Schema Evolution Syntax

Function

This capability supports column alterations on Hudi tables with SparkSQL. Schema evolution must be enabled before using this capability.

Scope of Schema Evolution

The scope of schema evolution includes:

- Column operations: Supports adding, deleting, modifying, and adjusting the position of columns (including nested columns).
- Partition columns: Does not support schema evolution on partition columns.
- Array type columns: Does not support adding, deleting, or altering nested columns of Array type.

5.3.1 ALTER COLUMN

Function

This syntax modifies the current column properties, including column comments and null constraints. Currently, it does not support modifying column types or column positions.

Precautions

- Modifying column types is currently not supported.
- Modifying the order of existing columns is currently not supported.
- Adding columns in a specified order is currently not supported.

Syntax

ALTER TABLE *Table name* **ALTER**

[COLUMN] *col_name*

[COMMENT] *col_comment*

Parameter Description

Table 5-31 ALTER COLUMN parameters

Parameter	Description
tableName	Table name.
col_name	Name of the column to be altered.
column_type	Type of the target column.
col_comment	Column comment.
column_name	New position to place the target column. For example, **AFTER column_name** indicates that the target column is placed after **column_name**.

Required Permissions

- SQL permissions

Table 5-32 Permissions required for executing ALTER TABLE

Permission Description
ALTER permission on a table

- Fine-grained permission: **dli:table:alter**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

- Altering other attributes
```
ALTER TABLE table1 ALTER COLUMN col_a DROP NOT NULL

ALTER TABLE table1 ALTER COLUMN col_a COMMENT 'new comment'
```

Response

You can run the **DESCRIBE** command to view the modified column.

5.3.2 ADD COLUMNS

Function

The **ADD COLUMNS** command is used to add a column to an existing table.

Syntax

ALTER TABLE *Table name* **ADD COLUMNS** *(col_spec[, col_spec ...])*

Parameter Description

Table 5-33 ADD COLUMNS parameters

Parameter	Description
tableName	Table name.

Parameter	Description
col_spec	Column specifications, consisting of four fields, **col_name**, **col_type**, **nullable**, and **comment**. ● **col_name**: name of the new column. It is mandatory. Adding new subcolumns to nested columns is currently not supported. ● **col_type**: type of the new column. It is mandatory. ● **nullable**: whether the new column can be null. The value can be left empty. ● **comment**: comment of the new column. The value can be left empty.

Required Permissions

- SQL permissions

Table 5-34 Permissions required for executing ALTER TABLE

Permission Description
ALTER permission on a table

- Fine-grained permission: **dli:table:alter**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
alter table h0 add columns(ext0 string);

alter table h0 add columns(new_col int comment 'add new column');

alter table delta.`obs://bucket_name0/db0/delta_table0` add columns(new_col string);
```

Response

You can run the **DESCRIBE** command to view the new column.

5.3.3 RENAME COLUMN

Function

The **ALTER TABLE ... RENAME COLUMN** command is used to change the column name.

Precautions

- If your table is already on the desired protocol version, you need to execute the following statement before the modification can be successful:
  ```
  ALTER TABLE table_name SET TBLPROPERTIES ('delta.columnMapping.mode' = 'name');
  ```
- If your table is not on the desired protocol version, you need to execute the following statement before the modification can be successful:
  ```
  ALTER TABLE table_name SET TBLPROPERTIES (
  'delta.columnMapping.mode' = 'name',
  'delta.minReaderVersion' = '2',
  'delta.minWriterVersion' = '5')
  ```

Syntax

ALTER TABLE *tableName* **RENAME COLUMN** *old_columnName* **TO** *new_columnName*

Parameter Description

Table 5-35 RENAME COLUMN parameters

Parameter	Description
tableName	Table name.
old_columnName	Old column name.
new_columnName	New column name.

Required Permissions

- SQL permissions

Table 5-36 Permissions required for executing ALTER TABLE

Permission Description
ALTER permission on a table

- Fine-grained permission: **dli:table:alter**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
ALTER TABLE table1 RENAME COLUMN addr to address

ALTER TABLE table1 RENAME COLUMN addr.priv to province
```

a.b.c indicates the full path of a nested column. For details about the nested column rules, see **ADD COLUMNS**.

Response

You can run the **DESCRIBE** command to view the new column name.

5.3.4 RENAME TABLE

Function

The **ALTER TABLE ... RENAME** command is used to change the table name.

Syntax

ALTER TABLE *tableName* **RENAME TO** *newTableName*

Parameter Description

Table 5-37 RENAME parameters

Parameter	Description
tableName	Table name.
newTableName	New table name.

Required Permissions

- SQL permissions

Table 5-38 Permissions required for executing ALTER TABLE

Permission Description
ALTER permission on a table

- Fine-grained permission: **dli:table:alter**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
ALTER TABLE table1 RENAME TO table2
```

Response

You can run the **SHOW TABLES** command to view the new table name.

5.3.5 DROP COLUMN

Function

The **ALTER TABLE ... DROP COLUMN** command is used to delete a column.

Syntax

ALTER TABLE *tableName* **DROP COLUMN|COLUMNS** *cols*

Parameter Description

Table 5-39 DROP COLUMN parameters

Parameter	Description
tableName	Table name.
cols	Columns to be deleted. You can specify multiple columns.

Required Permissions

- SQL permissions

Table 5-40 Permissions required for executing ALTER TABLE

Permission Description
ALTER permission on a table

- Fine-grained permission: **dli:table:alter**
- Metadata services provided by LakeFormation. Refer to the LakeFormation documentation for details on permission configuration.

Example

```
ALTER TABLE table1 DROP COLUMN a.b.c
ALTER TABLE table1 DROP COLUMNS a.b.c, x, y
```

a.b.c indicates the full path of a nested column. For details about the nested column rules, see **ADD COLUMNS**.

Response

You can run the **DESCRIBE** command to check which column is deleted.

6 Typical Delta Configurations

To set Delta parameters while submitting a DLI Spark SQL job, access the **SQL Editor** page and click **Settings** in the upper right corner. In the **Parameter Settings** area, set the parameters.

Table 6-1 Typical Delta configurations

Parameter	Description	Default Value
spark.databricks.delta.retentionDurationCheck.enabled	Whether the retention period is checked when vacuum clears files that are no longer referenced.	true
spark.databricks.delta.properties.defaults.deletedFileRetentionDuration or delta.deletedFileRetentionDuration	Retention period of files that are no longer referenced by Delta. If **spark.databricks.delta.retentionDurationCheck.enabled** is set to **true**, an exception will be thrown when files that have not yet reahced their retention period are cleared.	168 hours (1 week)
spark.databricks.delta.properties.defaults.logRetentionDuration or delta.logRetentionDuration	Expiration time of Delta log files. When a Delta log executes a checkpoint action, it scans for expired files that need deletion. If such files are found, they are removed to prevent the Delta log files from growing indefinitely.	30 days

7 DLI Delta FAQ

When Executing insert into/overwrite table_name partition(part_key='part_value') select ..., Error DLI.0005: DeltaAnalysisException: Partition column 'dt' not found in schema [id, name] Occurs

Root cause analysis: The syntax **insert into/overwrite table_name partition(part_key='part_value')** is not supported.

When Executing SQL, Error DLI.0005: There should be at least one partition pruning predicate on partitioned table '777dd'.'test_delta_parts1' Occurs

Solution: Add the parameter **spark.sql.forcePartitionPredicatesOnPartitionedTable.enabled** set to **false** in the console settings.

When Using show create table to View the Table Creation Statement, Error DLI.0005: Operation not allowed: 'SHOW CREATE TABLE' is not supported for Delta tables Occurs

Root cause analysis: This syntax is not supported. You can run **describe formatted** to view the table structure.

When Executing Vacuum, Error DLI.0001: IllegalArgumentException: requirement failed: Are you sure you would like to vacuum files with such a low retention period? Occurs

Root cause analysis: The RETAIN period is too short (less than 168 hours). You need to check whether the data before this time can be cleaned, as you will no longer be able to view versions created before the specified data retention period. To confirm the cleanup, add the parameter **spark.databricks.delta.retentionDurationCheck.enabled** set to **false** in the console settings.

When Executing rename/drop column, Error DLI.0005: DeltaAnalysisException: Column rename is not supported for your Delta table... Occurs

Solution: Execute the following statements first, then perform the rename:

ALTER TABLE delta_perms1 SET TBLPROPERTIES (

'delta.columnMapping.mode' = 'name',

'delta.minReaderVersion' = '2',

'delta.minWriterVersion' = '5');