FLIP-303: Support REPLACE TABLE AS SELECT statement

Status

Current state: Under Discussion

Discussion thread: https://lists.apache.org/thread/39mwckdsdgck48tzsdfm66hhxoj3

Vote thread: https://lists.apache.org/thread/5fzfqc6dw6wyx2xsnh2psotsxhbp26

JIRA: [FLINK-32515] - FLIP-303: Support REPLACE TABLE AS SELECT statement

Released: 1.18.0

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

In FLIP-218 & FLIP-305, Flink supports CREATE TABLE AS SELECT statement which allows users to create new tables based on existing tables or query results. It's convenient for data analysts and data scientists to manage their data. However, Flink does not currently support the REPLACE TABLE AS SELECT statement which enables users to replace an existing table with new data. With REPLACE TABLE AS SELECT, they won't need to drop the table first and use CREATE TABLE AS SELECT then. Only one single REPLACE TABLE AS SELECT statement can meet their needs.

So, this FLIP is aimed to support REPLACE TABLE AS SELECT statement in Flink.

Note: this FLIP is much similar to FLIP-218 & FLIP-305, you may need to read these two FLIP to get more context.

Public Interfaces

Syntax

We propose adding the following syntax for REPLACE TABLE AS SELECT statement:

```
REPLACE TABLE table_identifier
[ WITH ( key1=val1, key2=val2, ... ) ]
AS <table subquery>
```

Also, we would like to propose to CREATE OR REPLACE TABLE AS to wrap CREATE TABLE AS SELECT and REPLACE TABLE AS SELECT which will create a table if the table to be replaced doesn't exist.

```
CREATE OR REPLACE TABLE table_identifier
[ WITH ( key1=val1, key2=val2, ... ) ]
AS <table subquery>
```

Public Interfaces Change

To support atomic, we propose to add the following part to the interface SupportsStaging proposed in FLIP-305
/**
 * Enables different staged operations to ensure atomicity in a (@link DynamicTableSink).
 * By default, if this interface is not implemented, indicating that atomic operations are not supported, then a non-atomic implementation is used.
 */
@PublicEvolving
public interface SupportsStaging {
    //.... emit the parts proposed in FLIP-305
    enum StagingPurpose {
        CREATE_TABLE_AS,
        CREATE_TABLE_AS_IF_NOT_EXISTS,
        REPLACE_TABLE_AS,
        CREATE_OR_REPLACE_TABLE_AS
    }
}
Also, we propose to modify the name of the option "table.cats.atomicity-enabled" proposed in FLIP-305:

@PublicEvolving
public class TableConfigOptions {
    @Documentation.TableOption(execMode = Documentation.ExecMode.BATCH_STREAMING)
    public static final ConfigOption<Boolean> TABLE_RTAS_CTAS_ATOMICITY_ENABLED =
            key("table.rtas-ctas.atomicity-enabled")
                    .booleanType()
                    .defaultValue(false)
                    .withDescription("Specifies if the create table/replace table/create or replace table as select operation is executed atomically. " + "By default, the operation is non-atomic. The target table is created/replaced in Client side, and it won't rollback even though the job fails or is cancelled. " + "If set this option to true and DynamicTableSink implements the SupportsStaging interface, the create table/replace table/create or replace table as select operation is expected to be executed atomically. " + "the behavior of which depends on the actual DynamicTableSink.");
}

Proposed Changes

For REPLACE TABLE AS statement:

1: Construct the table to be created

2: Check the table exists or not. If the table doesn't exist, throw TableException(String.format("The table %s to be replaced doesn't exist. You may want to use CREATE TABLE AS statement or CREATE OR REPLACE TABLE AS statement.*",tableIdentifier)).

3:Check the atomicity is enabled, it requires both the option table.rtas-ctas.atomicity-enabled is set to true and the corresponding table sink implements SupportsStaging.

   a: if atomic is enabled, it expects the atomicity to be guaranteed by external connector implementation. The Flink will generate an insert job according to the table subquery in REPLACE TABLE AS statement, and call method StagedTable#begin before the insert job start, call method StagedTable#commit after the job finish, call method StagedTable#abort if the job fail or canceled.

   b: if not, then the atomicity can not be guaranted. Flink will do the operations for Replace Table one by one without atomicity guarantee. More exactly, it will drop the old table, create the new table, insert data into the new tables.

For CREATE OR REPLACE TABLE AS statement, when the table exists, it'll consider it as REPLACE TABLE AS statement. Otherwise, it'll consider it as CREATE TABLE AS statement.

Note: Again, the propose changes much depend on FLIP-305. For more detail, please see the proposed change part in FLIP-305.
Compatibility, Deprecation, and Migration Plan

No any compatibility problem.

Test Plan
UT & IT

Rejected Alternatives

*If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.*