FLIP-302: Support TRUNCATE TABLE statement in batch mode

Status

Current state: Accepted

Discussion thread: https://lists.apache.org/thread/m4r3wrd7p96wdst3nz2ncqzog6klf51cf

Vote thread: https://lists.apache.org/thread/fosvz0zcyfn6bp6vz2oxl45vq9qhkn2v

JIRA: FLINK-31874 - Support truncate table statement in batch mode

Released: 1.18.0

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

Motivation

The TRUNCATE TABLE statement is a SQL command that allows users to quickly and efficiently delete all rows from a table without dropping the table itself. This statement is commonly used in data warehouse, where large data sets are frequently loaded and unloaded from tables. It can improve performance by reducing the amount of time and resources required to delete large amounts of data.

Also, the TRUNCATE TABLE statement can help simplify and streamlining data management tasks by eliminating the need for complex queries or data manipulation operations.

Considering the TRUNCATE TABLE statement is useful and widely-used, this FLIP is aimed to support TRUNCATE TABLE statement in Flink. This FLIP will bring Flink the TRUNCATE TABLE syntax and an interface with which the corresponding connectors can implement their own logic for truncating table.

Note: Considering the use cases of truncate table are mainly for batch scenario and the semantic in stream scenario should be discussed separately, this FLIP to support truncate table in batch only.

Public Interfaces

Syntax

We propose add the following syntax for TRUNCATE TABLE statement.

```
TRUNCATE TABLE table_name
```

It supports truncate permanent or temporary table, if it's a view table, it should throw TableException with message Truncate a view table is not support.

Public interfaces Changes

We propose add a interface for TRUNCATE TABLE statement. The connectors can implement this interface to custom their logic for truncating table.
/**
 * Enables to delete all existing data in a {link DynamicTableSink} table using {code TRUNCATE
 * TABLE} statement.
 * For {code TRUNCATE TABLE} statement, if the corresponding {link DynamicTableSink} have
 * implemented this interface, then the method {link truncateTable()} will be invoked in execution
 * phase. Otherwise, Flink will throw an exception directly.
 */
@PublicEvolving
public interface SupportsTruncate {
    /**
     * Execute truncating table.
     * Note: please remember to throw exception if the truncation hasn't been executed successfully,
     * otherwise it'll be still considered to haven been executed successfully by Flink.
     */
    void executeTruncation();
}

Proposed Changes

1: Add a parse rule in parserImpls.ftl to match TRUNCATE TABLE statement and convert it to SqlTruncateTable

2: Create a SqlTruncateTableConverter to TruncateTableOperation, and register SqlTruncateTableConverter to SqlNodeConverters.

3: TruncateTableOperation should implement ExecutableOperation, and in method #execute(Context ctx), it will get the DynamicTableSink of the table which should implement SupportsTruncate and then call method SupportsTruncate#executeTruncation directly to truncate the table. If the DynamicTableSink doesn't implemente this interface, it should throw TableException('The truncate statement for the table is not supported as it hasn't implemented the interface SupportsTruncate'). For some sinks that doesn't support deleting data, it can also implements this interface but throw more concrete exception like "xxx doesn't support to truncate a table as delete is impossible for xxx".

Compatibility, Deprecation, and Migration Plan

No any compatibility issue.

Test Plan

It'll be covered by 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.