在 Delta Lake 中启用 Spark SQL DDL 和 DML

Delta Lake 0.7.0 是随着 Apache Spark 3.0 版本发布之后发布的，这个版本比较重要的特性就是支持使用 SQL 来操作 Delta 表，包括 DDL 和 DML 操作。本文将详细介绍如何使用 SQL 来操作 Delta Lake 表，关于 Delta Lake 0.7.0 版本的详细 Release Note 可以参见这里。

使用 SQL 在 Hive Metastore 中创建表

Delta Lake 0.7.0 支持在 Hive Metastore 中定义 Delta 表，而且这些操作支持使用 SQL 进行，包括创建表和修改表，如下：

```
-- Create table in the metastore
CREATE TABLE events (
   date DATE,
   eventId STRING,
   eventType STRING,
   data STRING)
USING DELTA
PARTITIONED BY (date)
LOCATION '/delta/events'

-- If a table with the same name already exists, the table is replaced with
the new configuration, else it is created
CREATE OR REPLACE TABLE events (
   date DATE,
   eventId STRING,
   eventType STRING,
   data STRING)
USING DELTA
PARTITIONED BY (date)
```

修改表：

```
-- Alter table and schema
ALTER TABLE table_name ADD COLUMNS (
   col_name data_type
      [COMMENT col_comment]
      [FIRST|AFTER colA_name],
   ...)
```

当然，我们也可以使用 Scala/Java/Python APIs:

- DataFrame.saveAsTable(tableName) 和 DataFrameWriterV2 APIs (#307).
- 使用 DeltaTable.forName(tableName) API 来创建 io.delta.tables.DeltaTable 实例，这个实例在 Scala/Java/Python 中运行 Update/Delete/Merge 操作是非常有用的。

支持使用 SQL 进行 Insert, Delete, Update 和 Merge 操作

在 Delta Lake 0.7.0 之前，被问的最多的问题就是支不支持使用 SQL 对 Delta Lake 表进行 delete, update 和 merge 操作。下面就是这些操作的例子：

```
-- Using append mode, you can atomically add new data to an existing Delta table
INSERT INTO events SELECT * FROM newEvents

-- To atomically replace all of the data in a table, you can use overwrite mode
INSERT OVERWRITE events SELECT * FROM newEvents


-- Delete events
DELETE FROM events WHERE date < '2017-01-01'


-- Update events
UPDATE events SET eventType = 'click' WHERE eventType = 'click'

-- Upsert data to a target Delta
-- table using merge
MERGE INTO events
USING updates
ON events.eventId = updates.eventId
WHEN MATCHED THEN UPDATE
SET events.data = updates.data
WHEN NOT MATCHED THEN INSERT
(date, eventId, data)
VALUES (date, eventId, data)
```

值得注意的是，Delta Lake 中的 MERGE 操作比标准 ANSI SQL 语法支持更高级的语法。例如，Delta Lake 的 merge 支持：

- Delete 操作 - 当数据与输入行匹配时删除当前行的数据。比如 "... WHEN MATCHED THEN DELETE ..."
- 具有子条件的多个匹配操作 - 目标行和源行匹配时可以进行更多灵活的操作。 例如：

  ```
  ...
  WHEN MATCHED AND events.shouldDelete THEN DELETE
  WHEN MATCHED THEN UPDATE SET events.data = updates.data
  ```

- 星形语法 - 将数据匹配的源列值设置目标列值的简写。例如：

  ```
  WHEN MATCHED THEN SET *
  WHEN NOT MATCHED THEN INSERT *
  -- equivalent to updating/inserting with event.date = updates.date,
     events.eventId = updates.eventId, event.data = updates.data
  ```

自动或增量的形式生成 Presto/Athena manifest 文件

Delta Lake 0.5.0 新增通过 manifest 文件的方式支持在 Presto/Athena 中查询 Delta Lake 的表，manifest 文件里面包含了生成 manifest 文件时刻的最新版本文件。为了支持在 Presto/Athena 中读取 Delta Lake 表需要做以下的操作：

- 生成 Delta Lake Manifest 文件；
- 配置 Presto 或 Athena 以支持读取生成的 manifests 文件；
- 手动再生成 Manifest 文件。

现在在 Delta Lake 0.7.0 里面支持使用以下命令来自动生成 manifest 文件：

```
ALTER TABLE delta.`pathToDeltaTable`
SET TBLPROPERTIES(
   delta.compatibility.symlinkFormatManifest.enabled=true
)
```

通过表属性来对表进行配置

通过使用 ALTER TABLE SET TBLPROPERTIES 来为表设置表属性。我们可以启用、禁用或配置 Delta 的许多特性，比如自动清单生成。可以使用表属性 delta.appendOnly=true 来阻止 Delta 表中的删除和更新。

我们还可以通过以下属性轻松控制 Delta Lake 表保留的历史:

- delta.logRetentionDuration: 控制表的历史记录（比如交易记录）保留多长时间。默认情况下，会保留三十天的历史记录，但是我们可能需要根据自己的要求（例如 GDPR 历史背景）更改此值；
- delta.deletedFileRetentionDuration：控制在成为VACUUM候选者之前必须删除文件的时间。默认情况下，超过7天的数据文件将被删除。

在 Delta Lake 0.7.0 版本, 我们可以使用 ALTER TABLE SET TBLPROPERTIES 来配置这些属性

```
ALTER TABLE delta.`pathToDeltaTable`
SET TBLPROPERTIES(
   delta.logRetentionDuration = "interval "
   delta.deletedFileRetentionDuration = "interval "
)
```

支持在 Delta 表 commit 文件中添加用户定义的元数据

我们可以使用 DataFrameWriter 选项 userMetadata 或在 SparkSession 中配置spark.databricks.delta.commitInfo.userMetadata 来支持在 Delta 表的 commit 文件中保存用户定义的字符串作为提交中的元数据。

在以下示例中，我们将根据每个用户请求从数据湖中删除一个用户（1xsdf1）。为确保我们将用户的请求与删除相关联，我们还将 DELETE 请求 ID 添加到userMetadata 中。

```
SET spark.databricks.delta.commitInfo.userMetadata={
   "GDPR":"DELETE Request 1x891jb23"
};
DELETE FROM user_table WHERE user_id = '1xsdf1'
```

当检查用户表(user_table)的历史操作时，我们可以轻松地在事务日志中识别相关的删除请求。

```
1 DESCRIBE HISTORY user_table
```

▸ (1) Spark Jobs

	d	operationMetrics	userMetadata
1		▾ object numRemovedFiles: "1" numDeletedRows: "1" numAddedFiles: "1" numCopiedRows: "1"	{ "GDPR":"DELETE request 1x891jb23" }
2		▸ {"numFiles": "8", "numOutputBytes": "3880", "numOutputRows": "10"}	{}

Showing all 2 rows.

如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：iteblog_hadoop

本文参考：https://databricks.com/blog/2020/08/27/enabling-spark-sql-ddl-and-dml-in-delta-lake-on-apache-spark-3-0.html