Apache Flink：Table API和SQL发展现状概述

Flink Table API

Apache Flink对SQL的支持可以追溯到一年前发布的0.9.0-milestone1版本。此版本通过引入Table API来提供类似于SQL查询的功能，此功能可以操作分布式的数据集，并且可以自由地和Flink其他API进行组合。Tables在发布之初就支持静态的以及流式数据(也就是提供了DataSet和DataStream相关APIs)。我们可以将DataSet或DataStream转成Table；同时也可以将Table转换成DataSet或DataStream，正如下面的例子所展示的：

```
/**
 * User: 过往记忆
 * Date: 2016年6月16日
 * Time: 下午23:16
 * bolg:
 * 本文地址：/archives/1691
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */
val execEnv = ExecutionEnvironment.getExecutionEnvironment
val tableEnv = TableEnvironment.getTableEnvironment(execEnv)

// obtain a DataSet from somewhere
val tempData: DataSet[(String, Long, Double)] =

// convert the DataSet to a Table
val tempTable: Table = tempData.toTable(tableEnv, 'location, 'time, 'tempF)
// compute your result
val avgTempCTable: Table = tempTable
 .where('location.like("room%"))
 .select(
   ('time / (3600 * 24)) as 'day,
   'Location as 'room,
   (('tempF - 32) * 0.556) as 'tempC
  )
 .groupBy('day, 'room)
 .select('day, 'room, 'tempC.avg as 'avgTempC)
// convert result Table back into a DataSet and print it
avgTempCTable.toDataSet[Row].print()
```

上面的例子是通过Scala语言展示的，不过我们也可以在Java中使用Table API。

本文由 过往记忆
大数据技术博客(
)编写整理，转载必须注明出处。欢迎
关注微信公共帐号：iteblog_hadoop了解最新的大数据资讯。

为什么需要引入SQL

大家可以看出，虽然Table API提供了类似于SQL的功能来操作分布式的数据，但是其有以下几点问题：

1、Table API不能独立使用。Table API必须包含在DataSet或DataStream的程序中。
2、对于批处理表(batch Tables)的查询并不支持外连接(outer joins)、排序以及其他很多在SQL查询中经常会使用到的标量函数(scalar functions)；
3、在流数据表(streaming tables)上的查询只支持fiter、union以及projections操作，并不支持aggregations或者joins操作。
4、Table查询语句的翻译(应该是翻译成逻辑计划)并没用使用到查询优化技术，只有在物理计划的时候会有相应的优化，因为物理优化会应用到所有的DataSet程序。
5、在程序中使用Table API远没有直接使用SQL来的方面。

针对以上的各种缺陷，在Apache Flink中引入SQL的支持势在必行。Flink 0.9引入的Table API、关系表达式的代码生成以及运行操作符(runtime operators)等技术都为SQL的引入奠定了基础。那为什么最初Flink社区没有选择先开发出一套新的SQL-on-Hadoop解决方案呢？那是因为社区认为在整个Hadoop生态环境下已经存在了大量的SQL-on-Hadoop解决方案；比如Apache Hive, Apache Drill, Apache Impala, Apache Tajo等等，所以集中精力提升Flink的其他方面的性能更重要。

但是随着流处理系统在业界的广泛使用以及Flink受到的关注越来越多，Flink社区最终决定很有必要为用户提供一个简单的SQL接口来操作分布式数据。于是在半年前，Flink开始着手于扩展Table API使得用户可以直接在流数据（当然静态的数据更可以使用）上使用SQL。Flink开始使用Apache Calcite来为用户提供SQL功能，并基于Apache Calcite重新设计Table API的架构。Apache Calcite是一个流行的SQL解析和优化框架，并且被许多项目使用，包括Apache Hive, Apache Drill, Cascading、Apache Kylin、Apache Storm等等；而且Apache Calcite社区将SQL on streams作为他们的未来规划，尤其适合Flink的SQL接口。

SQL的支持将在Flink 1.1.0版本正式发布。因为刚刚开始引入，所以最初版本SQL只支持在流数据上进行select、filter、union等操作；和 Flink 1.0.0相比，Table API将支持更多的scalar functions，支持从外部数据源读取数据并且支持写入到外部数据源。

在Flink 1.2.0版本，SQL的将会支持更多的特性。比如支持各种类型的window aggregates以及Streaming join。当然，这些开发是和Apache Calcite社区共同合作的结果。

如何在程序中使用SQL

在Flink程序中使用SQL查询只需要使用TableEnvironment的sql()方法。这个方法将会返回SQL查询的结果，结果的类型是Table，我们可以将它转换成DataSet或者DataStream、或者使用Table API操作它、或者将他写入到TableSink中。SQL和Table的查询API可以无缝地进行整合。

任何的Table, DataSet, DataStream或者TableSource都需要通过TableEnvironment进行注册，使得可以在其之上使用SQL查询。

SQL on Batch Tables

下面是介绍如何在Batch Tables上使用SQL的例子：

```
/**
 * User: 过往记忆
 * Date: 2016年6月16日
 * Time: 下午23:16
 * bolg:
 * 本文地址：/archives/1691
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */
val env = ExecutionEnvironment.getExecutionEnvironment
val tableEnv = TableEnvironment.getTableEnvironment(env)

// read a DataSet from an external source
val ds: DataSet[(Long, String, Integer)] = env.readCsvFile(...)
// register the DataSet under the name "Orders"
tableEnv.registerDataSet("Orders", ds, 'user, 'product, 'amount)
// run a SQL query on the Table and retrieve the result as a new Table
val result = tableEnv.sql(
  "SELECT SUM(amount) FROM Orders WHERE product LIKE '%Rubber%'")
```

目前Batch Tables只支持select、filter、projection, inner equi-joins, grouping, non-distinct aggregates以及sorting。

下面特性目前在Batch Tables不支持：

1、时间类型(DATE, TIME, TIMESTAMP, INTERVAL)以及DECIMAL类型
2、Distinct aggregates (比如：COUNT(DISTINCT name))
3、非等值Join以及笛卡儿乘积；
4、通过order里面的位置选择结果 (ORDER BY OFFSET FETCH)
5、Grouping sets
6、INTERSECT以及EXCEPT集合操作。

SQL on Streaming Tables

SQL查询可以扩展到 Streaming Tables上，我们只需要将SELECT关键字用SELECT STREAM替换即可。下面是使用示例：

```
/**
 * User: 过往记忆
 * Date: 2016年6月16日
 * Time: 下午23:16
 * bolg:
 * 本文地址：/archives/1691
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */
val env = StreamExecutionEnvironment.getExecutionEnvironment
val tEnv = TableEnvironment.getTableEnvironment(env)

// read a DataStream from an external source
val ds: DataStream[(Long, String, Integer)] = env.addSource(...)
// register the DataStream under the name "Orders"
tableEnv.registerDataStream("Orders", ds, 'user, 'product, 'amount)
// run a SQL query on the Table and retrieve the result as a new Table
val result = tableEnv.sql(
  "SELECT STREAM product, amount FROM Orders WHERE product LIKE '%Rubber%'")
```

目前在Streaming Tables上执行SQL只支持SELECT, FROM, WHERE以及UNION；Aggregations和joins暂时不支持。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接: 【】（）