Flink 1.10.0 SQL DDL中如何定义watermark和计算列

随着Flink1.10.0版本的发布,在SQL上一个重大的优化是支持了watermark语义的计算,在之前的Flink1.9.x版本中是不支持的,当时只能用SQL DDL进行processing time的计算,但是现在可以进行eventtime语义的计算了,那在Flink1.10.0版本中也推出了很多新的特性,这里就不在多介绍了,本篇文章主要是接上一篇文章,FlinkSQL使用DDL语句创建kafka源表,主要来介绍一下Flink1.10.0中怎么定义watermark.

SQL DDL 中的 watermark 和计算列语法

Flink 1.10 在 SQL DDL 中增加了针对流处理定义时间属性及产生 watermark 的语法扩展（FLIP-66 [22]）。这使得用户可以在用 DDL 语句创建的表上进行基于时间的操作（例如窗口）以及定义 watermark 策略[23]。

```
CREATE TABLE table_name (


WATERMARK FOR columnName AS <watermark_strategy_expression>


) WITH (
...
)
```

Flink提供了几种常用的watermark策略。

1,严格意义上递增的时间戳,发出到目前为止已观察到的最大时间戳的水印。时间戳小于最大时间戳的行不会迟到。

WATERMARK FOR rowtime_column AS rowtime_column

2,递增的时间戳,发出到目前为止已观察到的最大时间戳为负1的水印。时间戳等于或小于最大时间戳的行不会迟到。

WATERMARK FOR rowtime_column AS rowtime_column - INTERVAL '0.001' SECOND.

3,有界时间戳(乱序) 发出水印，它是观察到的最大时间戳减去指定的延迟，例如，WATERMARK FOR rowtime_column AS rowtime_column-INTERVAL'5'SECOND是5秒

的延迟水印策略。

WATERMARK FOR rowtime_column AS rowtime_column - INTERVAL 'string' timeUnit.

使用 DDL 创建 Kafka 表

```
val createTable =
            """
              |CREATE TABLE PERSON (
              |    name VARCHAR COMMENT '姓名',
              |    age VARCHAR COMMENT '年龄',
              |    city VARCHAR COMMENT '所在城市',
              |    address VARCHAR COMMENT '家庭住址',
              |    ts BIGINT COMMENT '时间戳',
              |    t as TO_TIMESTAMP(FROM_UNIXTIME(ts/1000,'yyyy-MM-dd HH:mm:s
              |    proctime as PROCTIME(),
              |    WATERMARK FOR t AS t - INTERVAL '5' SECOND
              |)               |WITH (
              |    'connector.type' = 'kafka', -- 使用 kafka connector
              |    'connector.version' = '0.11',  -- kafka 版本
              |    'connector.topic' = '',  -- kafka topic
              |    'connector.startup-mode' = 'latest-offset', -- 从起始 offset
              |    'connector.properties.zookeeper.connect' = '',  -- zk连接信息
              |    'connector.properties.bootstrap.servers' = '',  -- broker连
              |    'connector.properties.group.id' = '',
              |    'update-mode' = 'append',
              |    'format.type' = 'json',  -- 数据源格式为 json
              |    'format.derive-schema' = 'true' -- 从 DDL schema 确定 json 解
              |)               """.stripMargin
```

需要注意:

1,Flink目前不支持直接把Long类型的转成Timestamp类型的,如果你的数据源中ts是Long类型的时间戳,建表语句不能直接写成ts TIMESTAMP(3),如果想用timestamp类型的,数据源中的时间需要时UTC格式的.或者可以用我上面那种写法,利用Flink的日期函数TO_TIMESTAMP把bigint类型的转成timestamp类型的.

2,PROCTIME是内置函数产生的一个虚拟列,可以参与后续的计算.

3,WATERMARK是在ts的列上声明了一个可以容忍5秒的乱序.

4,新版本的一些connector属性和之前略微有差异,比如

connector.properties.zookeeper.connect连接zk的属性,之前是

'connector.properties.0.key' = 'zookeeper.connect', -- 连接信息

 'connector.properties.0.value' = 'xxx',

现在是合在一起了.

5,format.type目前支持的数据结构主要有三种,'csv', 'json' and 'avro'.

6,connector.version,在kafka0.11以上的版本可以写universal表示通用.

下面看完成的代码:

```
package flink.ddl


import org.apache.flink.streaming.api.TimeCharacteristic
import org.apache.flink.streaming.api.scala._
import org.apache.flink.table.api.EnvironmentSettings
import org.apache.flink.table.api.scala.StreamTableEnvironment
import org.apache.flink.types.Row
import org.apache.flink.table.api.scala._


/**
  * @program: Flink1.9.0
  * @description: Flink1.9.1使用DDL创建kafka源表和mysql sink表
  * @author: JasonLee
  * @create: 2020-01-14 19:49
  */
object FlinkKafkaDDLDemo {
    def main(args: Array[String]): Unit = {
        val env = StreamExecutionEnvironment.getExecutionEnvironment
        env.setParallelism(6)
        env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime)
        val settings = EnvironmentSettings.newInstance()
                .useBlinkPlanner()
                .inStreamingMode()
                .build()


        val tEnv = StreamTableEnvironment.create(env, settings)
        // 创建kafka源表
        val createTable =
            """
              |CREATE TABLE PERSON (
              |    name VARCHAR COMMENT '姓名',
              |    age VARCHAR COMMENT '年龄',
```

```
                |    city VARCHAR COMMENT '所在城市',
                |    address VARCHAR COMMENT '家庭住址',
                |    ts BIGINT COMMENT '时间戳',
                |    t as TO_TIMESTAMP(FROM_UNIXTIME(ts / 1000,'yyyy-MM-dd HH:mm
                |    proctime as PROCTIME(),
                |    WATERMARK FOR t AS t - INTERVAL '5' SECOND
                |)
                |WITH (
                |    'connector.type' = 'kafka', -- 使用 kafka connector
                |    'connector.version' = '0.11',  -- kafka 版本
                |    'connector.topic' = '',  -- kafka topic
                |    'connector.startup-mode' = 'latest-offset', -- 从起始 offset
                |    'connector.properties.zookeeper.connect' = '',  -- zk连接信息
                |    'connector.properties.bootstrap.servers' = '',  -- broker连
                |    'connector.properties.group.id' = '',
                |    'update-mode' = 'append',
                |    'format.type' = 'json',  -- 数据源格式为 json
                |    'format.derive-schema' = 'true' -- 从 DDL schema 确定 json 解
                |)
                """.stripMargin
        tEnv.sqlUpdate(createTable)


        val query =
            """
              | SELECT name,COUNT(age) as pv,count(distinct age) as uv,
              | TUMBLE_START(t, INTERVAL '5' second) as t_start,
              | TUMBLE_END(t, INTERVAL '5' second) as t_end
              | FROM PERSON GROUP BY name,TUMBLE(t, INTERVAL '5' second)
            """.stripMargin


        val result = tEnv.sqlQuery(query)
        result.toRetractStream[Row].print()


        tEnv.execute("Flink SQL DDL")
    }
}
```