10分钟了解Flink SQL使用

Flink 是一个流处理和批处理统一的大数据框架，专门为高吞吐量和低延迟而设计。开发者可以使用SQL进行流批统一处理，大大简化了数据处理的复杂性。本文将介绍Flink SQL的基本原理、使用方法、流批统一，并通过几个例子进行实践。

1、Flink SQL基本原理

Flink SQL建立在Apache Flink之上，利用Flink的强大处理能力，使得用户可以使用SQL语句进行流数据和批数据处理。Flink SQL既支持实时的流数据处理，也支持有界的批数据处理。

Flink SQL用SQL作为处理数据的接口语言，将SQL语句转换成数据流图（Dataflow Graph），再由Flink引擎执行。

2、Flink SQL固定编码套路

使用Flink SQL时，我们通常会遵循如下编码套路，这些套路和使用Flink API的套路是一样的：

* **环境准备**：初始化一个`TableEnvironment`对象，它是执行Flink SQL语句的核心。这个环境可以是流数据环境，也可以是批数据环境。
* **数据源定义**：通过`CREATE TABLE`语句定义输入数据源（source），可以是Kafka、CSV文件等。
* **数据处理**：编写SQL语句对数据进行处理，如查询、过滤、聚合等。
* **数据输出**：通过`CREATE TABLE`定义输出数据源（sink），并将处理结果输出。

3、Flink SQL代码示例

以下是一个从CSV文件读取数据，通过SQL查询，再将数据输出到CSV的完整例子。

* 先准备`input.csv文件`内容，如下：

```
1,product_A,10.5
2,product_B,20.3
3,product_C,15.8
1,product_D,12.2
2,product_A,18.7
```

* 编写demo代码

> 编写代码之前先在`pom.xml`中添加依赖：

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-csv</artifactId>
    <version>${flink.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-json</artifactId>
    <version>${flink.version}</version>
</dependency>
```

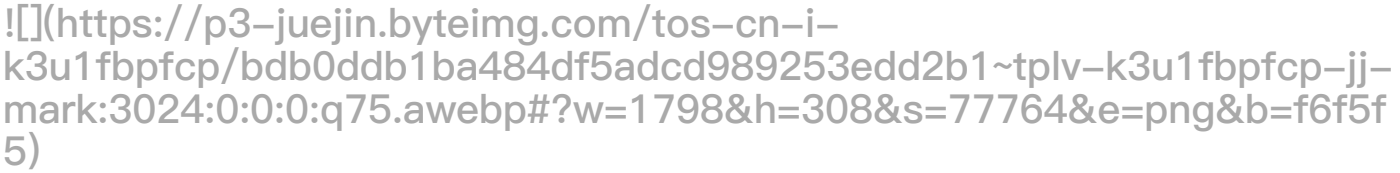
> 示例代码如下：

```
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironment
;
import org.apache.flink.table.api.EnvironmentSettings;
import org.apache.flink.table.api.bridge.java.StreamTableEnvironment;
```

```
public class FlinkSqlDemo {
    public static void main(String[] args) throws Exception {
        // 设置环境
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        env.setParallelism(1); //为了方便测试看效果，这里并行度设置为1
        // 使用EnvironmentSettings创建StreamTableEnvironment，明确设置
为批处理模式
        EnvironmentSettings settings = EnvironmentSettings
                .newInstance()
                .inBatchMode() // 设置为批处理模式，这样后续才能一次性的输
出到csv中
                .build();
        StreamTableEnvironment tableEnv =
StreamTableEnvironment.create(env, settings);

        // 定义输入数据源
        String createSourceTableDdl = "CREATE TABLE csv_source (" +
                " user_id INT," +
                " product STRING," +
                " order_amount DOUBLE" +
                ") WITH (" +
                " 'connector' = 'filesystem'," +
                " 'path' = 'file:///path/input.csv'," +
                " 'format' = 'csv'" +
                ")";
        tableEnv.executeSql(createSourceTableDdl);

//        // 编写 SQL 查询
//        String query = "SELECT user_id, SUM(order_amount) AS
total_amount FROM csv_source GROUP BY user_id";
//        // 执行查询并打印
//        tableEnv.executeSql(query).print();
//        env.execute("Flink SQL Demo");

        // 定义输出数据源
        String createSinkTableDdl = "CREATE TABLE csv_sink (" +
                " user_id INT," +
                " total_amount DOUBLE" +
                ") WITH (" +
                " 'connector' = 'filesystem'," +
                " 'path' = 'file:///path/output.csv'," +
                " 'format' = 'csv'" +
                ")";
        tableEnv.executeSql(createSinkTableDdl);

        // 执行查询并将结果输出到csv_sink
```

```
        String query = "INSERT INTO csv_sink " +
                "SELECT user_id, SUM(order_amount) as total_amount " +
                "FROM csv_source " +
                "GROUP BY user_id";
        tableEnv.executeSql(query);
//        env.execute("Flink SQL Job");
    }
}
```

* 执行结果如下：



4、Flink SQL做流批统一

什么是流批统一？

`流批统一`是大数据处理领域的一个概念，它指的是使用一套代码来同时处理流数据（Streaming）和批数据（Batching）。

流处理和批处理的区别如下：

1. 批处理（Batch Processing）：

* 批处理是指在某一时间点处理大量数据的手段。
* 它通常涉及到对大量静止的（不再变化的）数据集进行一次性的处理。
* 批处理作业通常在数据集完整可用后开始执行，并且经常是在数据仓库中进行。
* 例如，一个电商平台可能在一天结束时运行一个批处理作业来处理当天所有的交易记录。

2. 流处理（Stream Processing）：

* 流处理是指对数据实时进行处理，通常是数据生成或接收的同时立即进行。
* 流处理适用于连续的数据输入，这些数据一直在变化，需要立即响应。

* 例如，社交媒体平台在接收到新的帖子时，可能会实时分析这些帖子的内容和流行趋势。

在早期，流处理和批处理通常需要不同的系统来执行。对于批处理，可能使用如Hadoop这样的框架；而对于流处理，可能使用如Apache Storm这样的框架。这就导致开发者要同时学习多种框架才能处理不同类型的数据作业。

`流批统一`的概念，就是将这两种数据处理方式合并到一个平台中，这样一个系统既可以处理静止的大批量数据集，也可以处理实时的数据流。这样做的优点是显而易见的：

* 统一的API：开发人员只需要学习和使用一套工具和API，可以共享更多的代码和逻辑。
* 维护简便：只需维护一个系统，可以减少学习成本，减轻运维压力，减少故障点。
* 灵活的数据处理：可以根据不同的业务需求灵活选择数据处理方式。

Flink SQL流批一体的实现原理

Flink很好的实现了流批统一，可以让开发人员用相同的方式来编写批处理和流处理程序。不论是对有界（批处理）还是无界（流处理）的数据源，Flink都可以使用相同的API和处理逻辑来处理数据。

Flink 通过内置的表抽象来实现流批一体，这里的”表”可以是动态变化的（例如，来自实时数据流的表）或是静态的（例如，存储在文件或数据库中的批量数据表）。***Flink SQL引擎会根据数据的实际来源自动优化执行计划。***

Flink SQL的流批统一核心在于三点：

* **统一的API和SQL语义**：Flink SQL提供一致的查询构建块（如窗口、时间处理函数），这些在流处理和批处理中语义一致，确保不同模式下行为的统一性。
* **透明的状态处理**：无论是流处理还是批处理，Flink都能够保持和恢复状态，为开发者提供一致的高容错性体验。
* **多模态存储和处理能力**：Flink SQL能够访问不同存储介质的数据，这意味着相同的SQL语句可以无缝在流数据和存储的批量数据上执行。

Flink SQL流批统一的代码示例

以下是一个完整的代码示例，用Flink来实现流批统一处理。Flink同时从Kafka和 CSV读取数据，然后合并查询再输出结果：

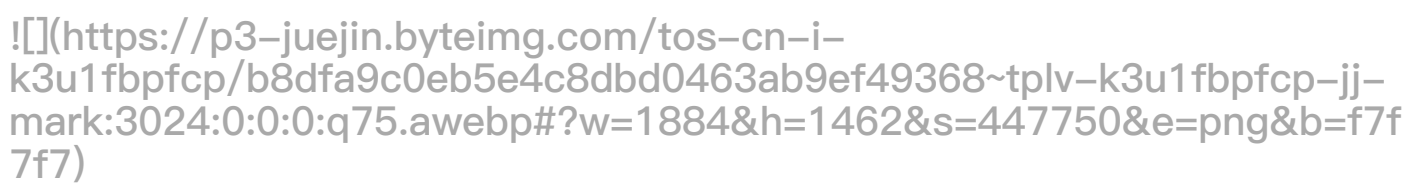
* 代码示例

> 代码中，先配置了Flink的流处理环境和表环境，然后用DDL语句在Flink中注册了Kafka和文件系统数据源。接着执行了一个SQL查询来合并来自这两种数据源的数据，并计算总金额。最后，打印出查询结果并开始执行Flink作业。

```
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironment
;
import org.apache.flink.table.api.EnvironmentSettings;
import org.apache.flink.table.api.Table;
import org.apache.flink.table.api.bridge.java.StreamTableEnvironment;
import org.apache.flink.types.Row;

public class StreamBatchUnifiedDemo {
    public static void main(String[] args) throws Exception {
        // 设置流处理的环境
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        EnvironmentSettings settings = EnvironmentSettings.newInstance()
                .inStreamingMode()
                .build();
        StreamTableEnvironment tableEnv =
StreamTableEnvironment.create(env, settings);

        // Kafka 流处理表
        String createKafkaSourceDDL = "CREATE TABLE
kafka_stream_orders (" +
                "order_id STRING," +
                "amount DOUBLE)" +
                "WITH (" +
                "'connector' = 'kafka'," +
                "'topic' = 'topic_test'," +
                "'properties.bootstrap.servers' = '10.20.1.26:9092'," +
                "'format' = 'json'," +
                "'scan.startup.mode' = 'latest-offset'" +
                ")";
        tableEnv.executeSql(createKafkaSourceDDL);
```

```
        // 文件系统批处理表
        String createFilesystemSourceDDL = "CREATE TABLE file_batch_orders (" +
                "order_id STRING," +
                "amount DOUBLE)" +
                "WITH (" +
                "'connector' = 'filesystem'," +
                "'path' = 'file:///Users/yclxiao/Project/bigdata/flink-blog/doc/input_order.csv'," +
                "'format' = 'csv'" +
                ")";
        tableEnv.executeSql(createFilesystemSourceDDL);

        // 执行统一查询，计算总金额
        Table resultTable = tableEnv.sqlQuery("SELECT SUM(amount) FROM (" +
                "SELECT amount FROM kafka_stream_orders " +
                "UNION ALL " +
                "SELECT amount FROM file_batch_orders)");

        // 打印结果
        tableEnv.toRetractStream(resultTable, Row.class).print();

        // 开始执行程序
        env.execute("Stream-Batch Unified Job");
    }
}
```

* 执行效果



通过以上示例代码，可以看出Flink SQL的流批一体设计：相同的SQL语句可以用在流处理和批处理中，而不需要做任何修改。Flink背后的执行引擎会自动根据数据的特性（流或者批）来进行相应的优化执行。

这就是Flink SQL非常强大的地方，它减少了开发者需要写不同代码逻辑的需求，简化了复杂的数据处理流程。

5、总结

Flink SQL是一个非常强大的数据处理工具，可以应对多种复杂的数据处理场景。

本文主要介绍了Flink SQL的基本原理、编码套路、流批统一，再结合正确的代码示例进行实践。希望对你有帮助。

完整代码地址：[github.com/yclxiao/fli...](http://cxyroad.com/”https://github.com/yclxiao/flink-blog”)

======>>>>>> [关于我](http://cxyroad.com/”https://mp.weixin.qq.com/s/xHu3SS2fKqw7dvzNlGBLOQ”)
<<<<<<<======

本篇完结！欢迎点赞 收藏！！！

原文链接： [mp.weixin.qq.com/s/WqyCjilMK...](http://cxyroad.com/”https://mp.weixin.qq.com/s/WqyCjilMK49T6eDl_Upc1A”)

原文链接: https://juejin.cn/post/7367626204205858843