Flink CDC 简化数据入仓链路

一、Flink CDC 介绍

能够捕获数据变更的技术统称为 CDC（Change Data Capture）。按照实现机制，CDC 可以分为两种类型：基于查询和基于日志的 CDC。

常见 CDC 方案比较

	Flink CDC	Debezium	DataX	Canal	Sqoop	kettle	Oracle Goldengate
CDC 机制	日志	日志	查询	日志	查询	查询	日志
增量同步	✓	✓	×	✓	✓	×	✓
断点续传	✓	✓	×	✓	×	×	✓
全量同步	✓	✓	✓	×	✓	✓	✓
全量+增量	✓	✓	×	×	✓	×	✓
架构	分布式	单机	单机	单机	分布式	分布式	分布式
Transformation	☆☆☆☆☆	☆☆	☆☆	☆☆	☆☆	☆	☆
生态	☆☆☆☆☆	☆☆☆	☆☆☆	☆☆☆	☆☆	☆☆	☆☆☆

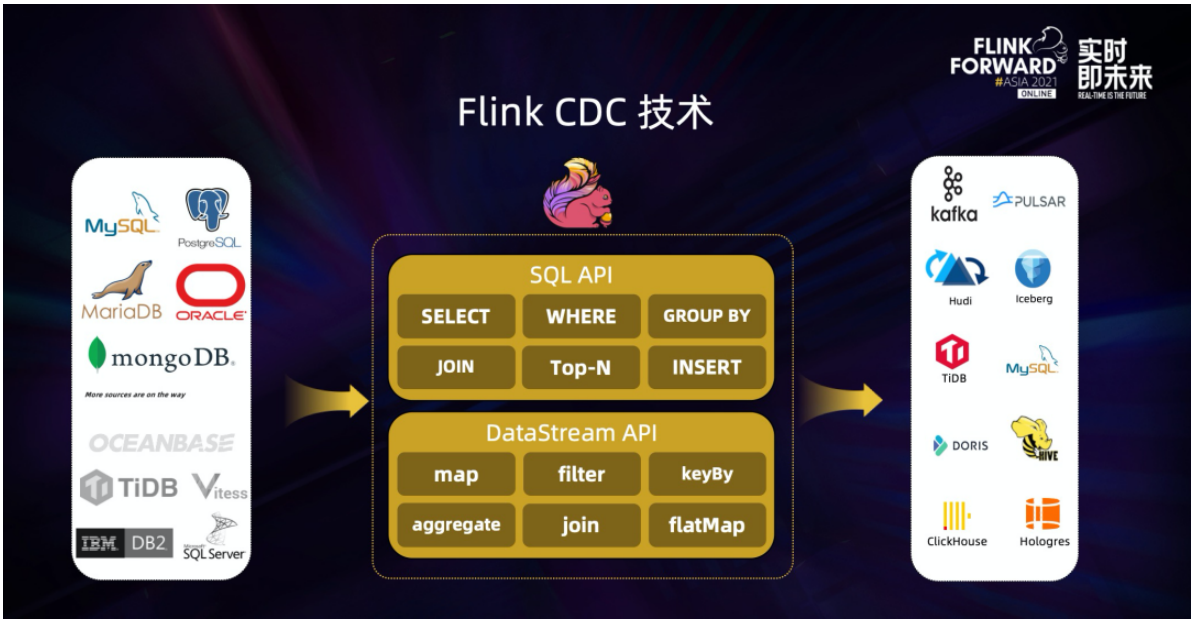
在早期的数据入仓架构中，一般会每天 SELECT 全量数据导入数仓后再做离线分析。这种架构有几个明显的不足：

1. 每天查询全量的业务表会影响业务自身稳定性。
2. 离线天级别调度的方式，天级别的产出时效性差。
3. 基于查询方式，随着数据量的不断增长，对数据库的压力也会不断增加，架构性能瓶颈明显。

后续进化到了 Lambda 架构，增加了实时同步导入增量的链路。整体来说，Lambda 架构的扩展性更好，也不再影响业务的稳定性，但仍然存在一些问题：

1. 依赖离线的定时合并，只能做到小时级产出，延时还是较大；
2. 全量和增量是割裂的两条链路；
3. 整个架构链路长，需要维护的组件比较多，该架构的全量链路需要维护 DataX 或 Sqoop 组件，增量链路要维护 Canal 和 Kafka 组件，同时还要维护全量和增量的定时合并链路。

Flink CDC 的出现为数据入仓架构提供了一些新思路。借助 Flink CDC 技术的全增量一体化实时同步能力，不会影响业务稳定性。其次，提供分钟级产出，满足近实时业务的需求。同时，全量和增量的链路完成了统一，实现了一体化同步。最后，该架构的链路更短，需要维护的组件更少。



二、Flink CDC 使用

Usage for DataStream API

```
package com.tz.flink.datastream;

import com.alibaba.ververica.cdc.debezium.StringDebeziumDeserializationSchema;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.api.functions.source.SourceFunction;
import com.alibaba.ververica.cdc.connectors.mysql.MySQLSource;
```

```
/**
 * test flink mysql cdc
 */
public class MySqlBinlogSourceExample {
    public static void main(String[] args) throws Exception {
        SourceFunction<String> sourceFunction = MySQLSource.<String>builder()
                .hostname("xx.xx.xx.xx")
                .port(3306)
                .databaseList("davincidb") // monitor all tables under inventory
database
                .username("root")
                .password("xysh1234")
                .deserializer(new StringDebeziumDeserializationSchema()) //
converts SourceRecord to String
                .tableList(new String[]{"davincidb.demo_orders"}) // 指定表需加上库
名，不然监控不到
                .build();

        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        env.addSource(sourceFunction).print().setParallelism(1); // use
parallelism 1 for sink to keep message ordering
        env.execute();
    }
}
```

Usage for Table/SQL API

```
-- creates a mysql cdc table source
Flink SQL > CREATE TABLE mysql_binlog (
 id INT NOT NULL,
 name STRING,
 description STRING,
 weight DECIMAL(10,3)
) WITH (
 'connector' = 'mysql-cdc',
 'hostname' = 'localhost',
 'port' = '3306',
 'username' = 'flinkuser',
 'password' = 'flinkpw',
 'database-name' = 'inventory',
 'table-name' = 'products'
);

-- read snapshot and binlog data from mysql, and do some transformation, and
show on the client
Flink SQL >SELECT id, UPPER(name), description, weight FROM mysql_binlog;
```

注：所有图来源于 Flink Forward Asia 2021