Flink1.11中的CDC Connectors操作实践

Flink1.11引入了CDC的connector，通过这种方式可以很方便地捕获变化的数据，大大简化了数据处理的流程。Flink1.11的CDC connector主要包括：`MySQL CDC`和`Postgres CDC`,同时对Kafka的**Connector**支持`canal-json`和`debezium-json`以及`changelog-json`的format。本文主要分享以下内容：

- CDC简介
- Flink提供的 table format
- 使用过程中的注意点
- mysql-cdc的操作实践
- canal-json的操作实践
- changelog-json的操作实践

简介

Flink CDC Connector 是ApacheFlink的一组数据源连接器，使用变化数据捕获change data capture (CDC)从不同的数据库中提取变更数据。Flink CDC连接器将Debezium集成为引擎来捕获数据变更。因此，它可以充分利用Debezium的功能。

特点

- 支持读取数据库快照，并且能够持续读取数据库的变更日志，即使发生故障，也支持**exactly-once** 的处理语义
- 对于DataStream API的CDC connector，用户无需部署Debezium和Kafka，即可在单个作业中使用多个数据库和表上的变更数据。
- 对于Table/SQL API 的CDC connector，用户可以使用SQL DDL创建CDC数据源，来监视单个表上的数据变更。

使用场景

- 数据库之间的增量数据同步
- 审计日志
- 数据库之上的实时物化视图
- 基于CDC的维表join

- ...

Flink提供的 table format

Flink提供了一系列可以用于table connector的table format，具体如下：

Formats	Supported Connectors
CSV	Apache Kafka, Filesystem
JSON	Apache Kafka, Filesystem, Elasticsearch
Apache Avro	Apache Kafka, Filesystem
Debezium CDC	Apache Kafka
Canal CDC	Apache Kafka
Apache Parquet	Filesystem
Apache ORC	Filesystem

使用过程中的注意点

使用MySQL CDC的注意点

如果要使用MySQL CDC connector，对于程序而言，需要添加如下依赖:

```
<dependency>
  <groupId>com.alibaba.ververica</groupId>
  <artifactId>flink-connector-mysql-cdc</artifactId>
  <version>1.0.0</version>
</dependency>
```

如果要使用Flink SQL Client，需要添加如下jar包：**flink-sql-connector-mysql-cdc-1.0.0.jar**，将该jar包放在Flink安装目录的lib文件夹下即可。

使用canal-json的注意点

如果要使用Kafka的canal-json，对于程序而言，需要添加如下依赖:

```
<!-- universal -->
<dependency>
    <groupId>org.apache.flink</groupId>
```

```
    <artifactId>flink-connector-kafka_2.11</artifactId>
    <version>1.11.0</version>
</dependency>
```

如果要使用Flink SQL Client，需要添加如下jar包：**flink-sql-connector-kafka_2.11-1.11.0.jar**，将该jar包放在Flink安装目录的lib文件夹下即可。由于Flink1.11的安装包 的lib目录下并没有提供该jar包，所以必须要手动添加依赖包，否则会报如下错误：

```
[ERROR] Could not execute SQL statement. Reason:
org.apache.flink.table.api.ValidationException: Could not find any factory for identifier 'kafk

Available factory identifiers are:

datagen
mysql-cdc
```

使用changelog-json的注意点

如果要使用Kafka的changelog-json Format，对于程序而言，需要添加如下依赖：

```
<dependency>
  <groupId>com.alibaba.ververica</groupId>
  <artifactId>flink-format-changelog-json</artifactId>
  <version>1.0.0</version>
</dependency>
```

如果要使用Flink SQL Client，需要添加如下jar包：**flink-format-changelog-json-1.0.0.jar**，将该jar包放在Flink安装目录的lib文件夹下即可。

mysql-cdc的操作实践

创建MySQL数据源表

在创建MySQL CDC表之前，需要先创建MySQL的数据表，如下：

```
-- MySQL
/*Table structure for table `order_info` */
DROP TABLE IF EXISTS `order_info`;
CREATE TABLE `order_info` (
```

```
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '编号',
  `consignee` varchar(100) DEFAULT NULL COMMENT '收货人',
  `consignee_tel` varchar(20) DEFAULT NULL COMMENT '收件人电话',
  `total_amount` decimal(10,2) DEFAULT NULL COMMENT '总金额',
  `order_status` varchar(20) DEFAULT NULL COMMENT '订单状态,1表示下单，2表示支付',
  `user_id` bigint(20) DEFAULT NULL COMMENT '用户id',
  `payment_way` varchar(20) DEFAULT NULL COMMENT '付款方式',
  `delivery_address` varchar(1000) DEFAULT NULL COMMENT '送货地址',
  `order_comment` varchar(200) DEFAULT NULL COMMENT '订单备注',
  `out_trade_no` varchar(50) DEFAULT NULL COMMENT '订单交易编号（第三方支付用)',
  `trade_body` varchar(200) DEFAULT NULL COMMENT '订单描述(第三方支付用)',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `operate_time` datetime DEFAULT NULL COMMENT '操作时间',
  `expire_time` datetime DEFAULT NULL COMMENT '失效时间',
  `tracking_no` varchar(100) DEFAULT NULL COMMENT '物流单编号',
  `parent_order_id` bigint(20) DEFAULT NULL COMMENT '父订单编号',
  `img_url` varchar(200) DEFAULT NULL COMMENT '图片路径',
  `province_id` int(20) DEFAULT NULL COMMENT '地区',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='订单表';
-- ----------------------------
-- Records of order_info
-- ----------------------------
INSERT INTO `order_info`
VALUES (476, 'lAXjcL', '13408115089', 433.00, '2', 10, '2', 'OYyAdSdLxedceqovndCD', 'ihjAYsSjr
INSERT INTO `order_info`
VALUES (477, 'QLiFDb', '13415139984', 772.00, '1', 90, '2', 'OizYrQbKuWvrvdfpkeSZ', 'wiBhhqhMn
INSERT INTO `order_info`
VALUES (478, 'iwKjQD', '13320383859', 88.00, '1', 107, '1', 'cbXLKtNHWOcWzJVBWdAs', 'njjsnknHx

/*Table structure for table `order_detail` */
CREATE TABLE `order_detail` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '编号',
  `order_id` bigint(20) DEFAULT NULL COMMENT '订单编号',
  `sku_id` bigint(20) DEFAULT NULL COMMENT 'sku_id',
  `sku_name` varchar(200) DEFAULT NULL COMMENT 'sku名称（冗余)',
  `img_url` varchar(200) DEFAULT NULL COMMENT '图片名称（冗余)',
  `order_price` decimal(10,2) DEFAULT NULL COMMENT '购买价格(下单时sku价格）',
  `sku_num` varchar(200) DEFAULT NULL COMMENT '购买个数',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='订单明细表';

-- ----------------------------
-- Records of order_detail
-- ----------------------------
INSERT INTO `order_detail`
VALUES (1329, 476, 8, 'Apple iPhone XS Max (A2104) 256GB 深空灰色 移动联通电信4G手机 双卡双待', 'htt
INSERT INTO `order_detail`
VALUES (1330, 477, 9, '荣耀10 GT游戏加速 AIS手持夜景 6GB+64GB 幻影蓝全网通 移动联通电信', 'http://ixOC
INSERT INTO `order_detail`
VALUES (1331, 478, 4, '小米Play 流光渐变AI双摄 4GB+64GB 梦幻蓝 全网通4G 双卡双待 小水滴全面屏拍照游戏智能
INSERT INTO `order_detail`
VALUES (1332, 478, 8, 'Apple iPhone XS Max (A2104) 256GB 深空灰色 移动联通电信4G手机 双卡双待', 'htt
INSERT INTO `order_detail`
VALUES (1333, 478, 8, 'Apple iPhone XS Max (A2104) 256GB 深空灰色 移动联通电信4G手机 双卡双待', 'htt
```

Flink SQL Cli创建CDC数据源

启动 Flink 集群，再启动 SQL CLI,执行下面命令：

```
-- 创建订单信息表
CREATE TABLE order_info(
    id BIGINT,
    user_id BIGINT,
    create_time TIMESTAMP(0),
    operate_time TIMESTAMP(0),
    province_id INT,
    order_status STRING,
    total_amount DECIMAL(10, 5)
  ) WITH (
    'connector' = 'mysql-cdc',
    'hostname' = 'kms-1',
    'port' = '3306',
    'username' = 'root',
    'password' = '123qwe',
    'database-name' = 'mydw',
    'table-name' = 'order_info'
);
```

在Flink SQL Cli中查询该表的数据：result-mode: tableau，+表示数据的insert

```
Flink SQL> select * from order_info;
+-----+----------------------+----------------------+-------------------------+-------------------------+-------------+----------------------+
| +/- |                   id |              user_id |             create_time |          operate_time | province_id |         order_status |
+-----+----------------------+----------------------+-------------------------+-------------------------+-------------+----------------------+
|   + |                  476 |                   10 |     2020-06-18T02:21:38 |                  (NULL) |           9 |                    1 |
|   + |                  477 |                   90 |     2020-06-18T09:12:25 |                  (NULL) |           3 |                    1 |
|   + |                  478 |                  107 |     2020-06-18T15:56:34 |                  (NULL) |           7 |                    1 |
```

在SQL CLI中创建订单详情表：

```
CREATE TABLE order_detail(
    id BIGINT,
    order_id BIGINT,
    sku_id BIGINT,
    sku_name STRING,
    sku_num BIGINT,
    order_price DECIMAL(10, 5),
 create_time TIMESTAMP(0)
 ) WITH (
    'connector' = 'mysql-cdc',
    'hostname' = 'kms-1',
    'port' = '3306',
    'username' = 'root',
    'password' = '123qwe',
    'database-name' = 'mydw',
    'table-name' = 'order_detail'
);
```

查询结果如下：

```
Flink SQL> select * from order_detail;
+-----+----------------------+----------------------+----------------------+--------------------------------+----------------------+-----------------+-------------------------+
| +/- |                   id |             order_id |               sku_id |                       sku_name |             sku_num |     order_price |             create_time |
+-----+----------------------+----------------------+----------------------+--------------------------------+----------------------+-----------------+-------------------------+
|   + |                 1329 |                  476 |                    8 | Apple iPhone XS M...           |                    3 | 8900.0000...    |     2020-06-18T02:21:38 |
|   + |                 1330 |                  477 |                    9 | 荣耀10 GT游戏加速...           |                    4 | 2452.0000...    |     2020-06-18T09:12:25 |
|   + |                 1331 |                  478 |                    4 | 小米Play 流光渐变...           |                    1 | 1442.0000...    |     2020-06-18T15:56:34 |
|   + |                 1332 |                  478 |                    8 | Apple iPhone XS M...           |                    3 | 8900.0000...    |     2020-06-18T15:56:34 |
|   + |                 1333 |                  478 |                    8 | Apple iPhone XS M...           |                    1 | 8900.0000...    |     2020-06-18T15:56:34 |
```

执行JOIN操作：

```
SELECT
    od.id,
    oi.id order_id,
    oi.user_id,
    oi.province_id,
    od.sku_id,
    od.sku_name,
    od.sku_num,
    od.order_price,
    oi.create_time,
    oi.operate_time
FROM
   (
```

```
    SELECT *
    FROM order_info
    WHERE
      order_status = '2'-- 已支付
  ) oi
  JOIN
 (
    SELECT *
    FROM order_detail
 ) od
 ON oi.id = od.order_id;
```

canal-json的操作实践

关于cannal的使用方式，可以参考我的另一篇文章：基于Canal与Flink实现数据实时增量同步(一)。我已经将下面的表通过canal同步到了kafka，具体格式为：

```
{
    "data":[
        {
            "id":"1",
            "region_name":"华北"
        },
        {
            "id":"2",
            "region_name":"华东"
        },
        {
            "id":"3",
            "region_name":"东北"
        },
        {
            "id":"4",
            "region_name":"华中"
        },
        {
            "id":"5",
            "region_name":"华南"
        },
        {
            "id":"6",
            "region_name":"西南"
        },
        {
            "id":"7",
            "region_name":"西北"
        }
```

```
    ],
    "database":"mydw",
    "es":1597128441000,
    "id":102,
    "isDdl":false,
    "mysqlType":{
        "id":"varchar(20)",
        "region_name":"varchar(20)"
    },
    "old":null,
    "pkNames":null,
    "sql":"",
    "sqlType":{
        "id":12,
        "region_name":12
    },
    "table":"base_region",
    "ts":1597128441424,
    "type":"INSERT"
}
```

在SQL CLI中创建该canal-json格式的表：

```
CREATE TABLE region (
  id BIGINT,
  region_name STRING
) WITH (
 'connector' = 'kafka',
 'topic' = 'mydw.base_region',
 'properties.bootstrap.servers' = 'kms-3:9092',
 'properties.group.id' = 'testGroup',
 'format' = 'canal-json' ,
 'scan.startup.mode' = 'earliest-offset'
);
```

查询结果如下：

```
Flink SQL> select * from region;
+-----+----------------------+----------------------+
| +/- |                   id |          region_name |
+-----+----------------------+----------------------+
|   + |                    1 |                 华北 |
|   + |                    2 |                 华东 |
|   + |                    3 |                 东北 |
|   + |                    4 |                 华中 |
|   + |                    5 |                 华南 |
|   + |                    6 |                 西南 |
|   + |                    7 |                 西北 |
```

changelog-json的操作实践

创建MySQL数据源

参见上面的**order_info**

Flink SQL Cli创建changelog-json表

```
CREATE TABLE order_gmv2kafka (
  day_str STRING,
  gmv DECIMAL(10, 5)
) WITH (
    'connector' = 'kafka',
    'topic' = 'order_gmv_kafka',
    'scan.startup.mode' = 'earliest-offset',
    'properties.bootstrap.servers' = 'kms-3:9092',
    'format' = 'changelog-json'
);

INSERT INTO order_gmv2kafka
SELECT DATE_FORMAT(create_time, 'yyyy-MM-dd') as day_str, SUM(total_amount) as gmv
FROM order_info
WHERE order_status = '2' -- 订单已支付
GROUP BY DATE_FORMAT(create_time, 'yyyy-MM-dd');
```

查询表看一下结果：

```
Flink SQL> select * from order_gmv2kafka;
+-----+----------------------+--------------+
| +/- |              day_str |          gmv |
+-----+----------------------+--------------+
|   + |           2020-06-18 | 433.00000... |
```

再查一下kafka的数据：

```
{"data":{"day_str":"2020-06-18","gmv":433},"op":"+I"}
```

当将另外两个订单的状态order_status更新为2时，`总金额=443+772+88=1293` 再观察数据：

```
Flink SQL> select * from order_gmv2kafka;
+-----+----------------------+--------------+
| +/- |              day_str |          gmv |
+-----+----------------------+--------------+
|   + |           2020-06-18 | 433.00000... |
|   - |           2020-06-18 | 433.00000... |
|   + |           2020-06-18 | 1205.0000... |
|   - |           2020-06-18 | 1205.0000... |
|   + |           2020-06-18 | 1293.0000... |
```

再看kafka中的数据：

```
[kms@kms-2 kafka_2.11-2.1.0]$ ./kafka-consumer.sh  order_gmv_kafka;
kafka consumer
{"data":{"day_str":"2020-06-18","gmv":433},"op":"+I"}
{"data":{"day_str":"2020-06-18","gmv":433},"op":"-U"}
{"data":{"day_str":"2020-06-18","gmv":1205},"op":"+U"}
{"data":{"day_str":"2020-06-18","gmv":1205},"op":"-U"}
{"data":{"day_str":"2020-06-18","gmv":1293},"op":"+U"}
```

总结

本文基于Flink1.11的SQL，对新添加的CDC connector的使用方式进行了阐述。主要包括MySQL CDC connector、canal-json及changelog-json的format，并指出了使用过程中的注意点。另外本文给出了完整的使用示例，如果你有现成的环境，那么可以直接进行测试使用。