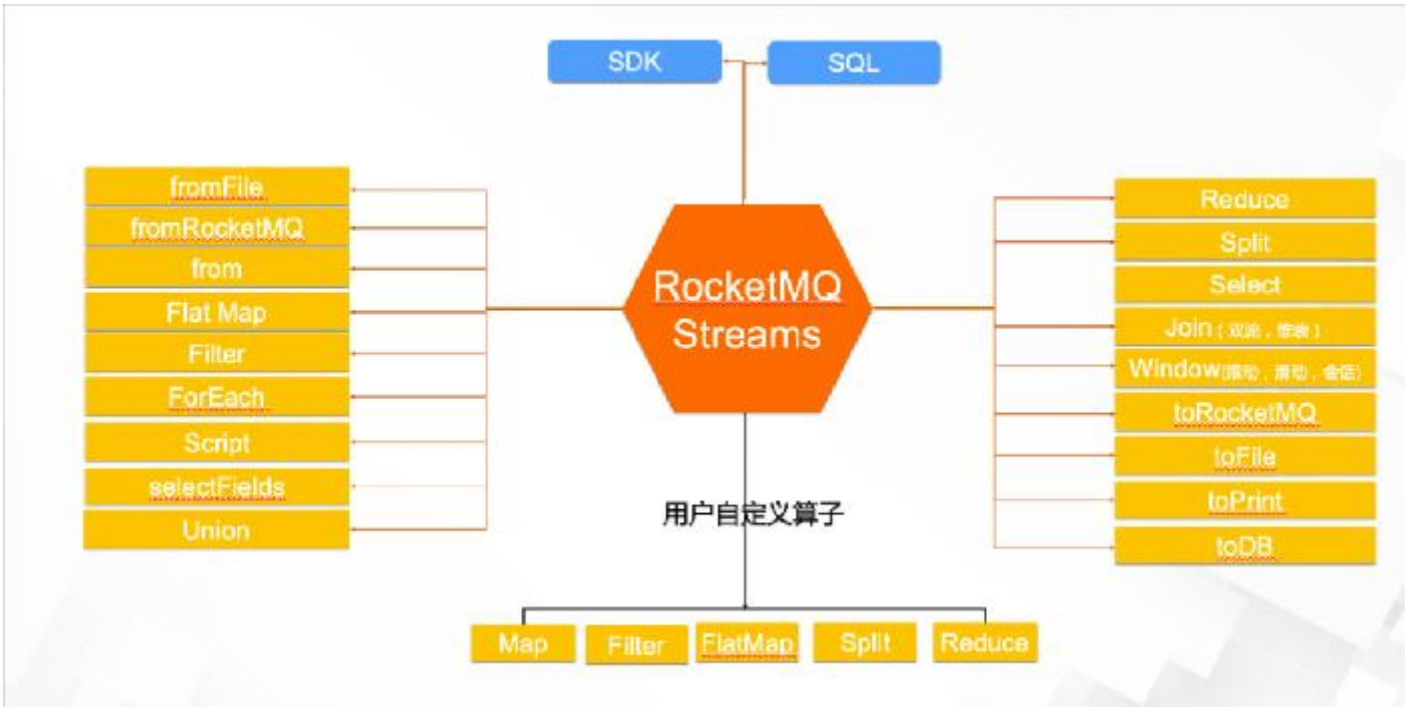
题目2：RocketMQ-streams Demo 开发

在新版本的RocketMQ（即RocketMQ5.0版本）中进行了架构重塑，新增或者修改了超过 60% 的代码，但是对 4.0 的所有功能以及整体架构进行了无缝兼容，且没有引入任何外部依赖。而且其中非常重要的一点是，RocketMQ 兼容了开源 Flink 生态。RocketMQ 直接实现了 Flink 的基础功能或者算子，并首创性地兼容了 Flink/Blink SQL 标准以及 UDF/UDAF/UDTF。

功能上 Flink 的基础功能或者算子如 Source、Reduce、Sink、map、flatmap、split、select、window、union rocketmq-streams 都做了实现，当然也包括一些基础的核心特点：Exactly-once，灵活窗口（滚动，滑动，会话），双流 Join，高吞吐、低延迟、高性能等。



rocketmq-streams 的设计之初主要是为了解决用户在使用 RocketMQ 过程中面临的一些轻量级计算的问题，在实时计算领域，社区一直以来的目标都是积极的与 Flink、Spark 等实时计算引擎通过生态合作的方式满足用户诉求。因此，rocketmq-streams 并非要做一个和 Flink 同质的大数据计算引擎，这个引擎场景很明确，主要是满足大数据量 ->高过滤 ->轻窗口计算的场景，重点要打造轻量化、高性能和低成本等优势。

接下来，将要部分算子进行简单的分析。

1. union算子：在做数据同步的时候，多个相同结构的源，通过这个union变成一个流，然后执行process及sink。

示例代码如下，从RocketMQ两个topic接收canal解析的binlog数据，然后简单转成String后，合并成一个流。

```
public class UnionDemo {
    private static Logger logger = Logger.getLogger(UnionDemo.class);

    public static void main(String[] args) {
        try {
            // 1.初始化两个数据源
            StreamExecutionEnvironment env1 = StreamExecutionEnvironment.getExecutionEnvironment();

            Properties consumerProps1 = new Properties();
            consumerProps1.setProperty(RocketMQConfig.NAME_SERVER_ADDR, Constant.SOURCE_NAME_SERVER_ADDR);
            consumerProps1.setProperty(RocketMQConfig.CONSUMER_GROUP, "flink_demo");
            consumerProps1.setProperty(RocketMQConfig.CONSUMER_TOPIC, "BinLogFromCanal");
            Properties consumerProps2 = new Properties();
            consumerProps2.setProperty(RocketMQConfig.NAME_SERVER_ADDR, Constant.SOURCE_NAME_SERVER_ADDR);
            consumerProps2.setProperty(RocketMQConfig.CONSUMER_GROUP, "flink_demo1");
            consumerProps2.setProperty(RocketMQConfig.CONSUMER_TOPIC, "MsgFromOds");

            // 2.初始化数据源,对数据源进行映射，过滤，根据表名分成多个侧数据流
            DataStream<String> dataStream1 = env1
                .addSource(new RocketMQSource(
                    new SimpleKeyValueDeserializationSchema(Constant.MQ_CONSTANT_ID, Constant.MQ_CONSTANT_ADDRESS),
                    consumerProps1))
                .name("source1").setParallelism(1)
                .map(new MapFunction<Map<String, String>, String>() {
                    @Override
                    public String map(Map<String, String> value) throws Exception {
                        StringBuffer str = new StringBuffer("source1:");
                        BinLogMsgEntity msgEntity = JSON.parseObject(value.get(Constant.MQ_CONSTANT_ADDRESS),
                            new TypeReference<BinLogMsgEntity>() {});
                        str.append(msgEntity.getDatabase());
                        return str.toString();
                    }
                });
            DataStream<String> dataStream2 = env1
                .addSource(new RocketMQSource(

```

```
                    new SimpleKeyValueDeserializationSchema(Constant.MQ_CONSTANT_ID, Constant.MQ_CONSTANT_ADDRESS),
                    consumerProps2))
                .name("source2").setParallelism(1)
                .map(new MapFunction<Map<String, String>, String>() {
                    @Override
                    public String map(Map<String, String> value) throws Exception {
                        StringBuffer str = new StringBuffer("source2:");
                        BinLogMsgEntity msgEntity = JSON.parseObject(value.get(Constant.MQ_CONSTANT_ADDRESS),
                                new TypeReference<BinLogMsgEntity>() {});
                        str.append(msgEntity.getDatabase());
                        return str.toString();
                    }
                });

            dataStream2.print();
            dataStream1.print();
            //3.将两个数据流合并
            DataStream<String> unionStream  = dataStream2.union(dataStream1).map(new MapFunction<String, String>() {

                @Override
                public String map(String value) throws Exception {
                    String str = value.replaceAll("source1", "unionSource").replaceAll("source2", "unionSource");
                    return str;
                }});

            //4.打印
            unionStream.print();

            //执行数据流
            env1.execute("geekplus_dws_etl_job1");
        } catch (Exception e) {
            e.printStackTrace();
            logger.error("error:" + e.getMessage());
        }
    }
}
```

2. Reduce算子：将新流入的数据，和最后一个reduce计算后的值进行计算，生成一个新值。

```
public class ReduceDemo {
    private static Logger logger = Logger.getLogger(ReduceDemo.class);

    public static void main(String[] args) {
        try {
            // 1.初始化数据源
            StreamExecutionEnvironment env1 = StreamExecutionEnvironment.getExecutionEnvironment();

            Properties consumerProps1 = new Properties();
            consumerProps1.setProperty(RocketMQConfig.NAME_SERVER_ADDR, Constant.SOURCE_NAME_SERVER_ADDR);
            consumerProps1.setProperty(RocketMQConfig.CONSUMER_GROUP, "flink_demo");
            consumerProps1.setProperty(RocketMQConfig.CONSUMER_TOPIC, "BinLogFromCanal");

            // 2.初始化数据源,对数据源进行映射，过滤，根据表名分成多个侧数据流
            DataStream<BinLogMsgEntity> dataStream1 = env1
                .addSource(new RocketMQSource(
                    new SimpleKeyValueDeserializationSchema(Constant.MQ_CONSTANT_ID, Constant.MQ_CONSTANT_ADDRESS),
                    consumerProps1))
                .name("source1").setParallelism(1)
                .map(new MapFunction<Map<String, String>, BinLogMsgEntity>() {
                    @Override
                    public BinLogMsgEntity map(Map<String, String> value) throws Exception {
                        BinLogMsgEntity msgEntity = JSON.parseObject(value.get(Constant.MQ_CONSTANT_ADDRESS),
                            new TypeReference<BinLogMsgEntity>() {});
                        return msgEntity;
                    }
                })
                .filter(new FilterFunction<BinLogMsgEntity>() {

                    @Override
                    public boolean filter(BinLogMsgEntity value) throws Exception {
                        if(null == value.getData() ) {
                            return false;
                        }else {
                            return true;
                        }
                    }})
                .keyBy("table").timeWindow(Time.seconds(5))
```

```
                .reduce(new ReduceFunction<BinLogMsgEntity>() {

                    @Override
                    public BinLogMsgEntity reduce(BinLogMsgEntity value1, BinLogMsgEntity value2) throws Exception {
                        System.out.println("-------------------redeceFunction:tablie1:{"+value1.getTable()+"},tablie2:{"+value2.getTable()+"}。"
                            + "size1:{"+value1.getData().size()+"},size2:{"+value2.getData().size()+"}。"
                            + "type1:{"+value1.getType()+"},type2:{"+value2.getType()+"}----------------------");
                        return value1.getData().size()>value2.getData().size() ? value1 : value2 ;
                    }});

            dataStream1.print();

            // 执行数据流
            env1.execute("geekplus_dws_etl_job1");
        } catch (Exception e) {
            e.printStackTrace();
            logger.error("error:" + e.getMessage());
        }
    }

}
```

3. split算子与select算子：split算子用于将一个数据流拆分成两个或多个数据流，但是，我们如何拿到切出来的那个数据流，这就需要用到select算子。

```
public class SplitDemo {
    private static Logger logger = Logger.getLogger(SplitDemo.class);


    public static void main(String[] args) {
        try {
            // 1.加载数据源参数
            StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecut
ionEnvironment();
            Properties consumerProps = new Properties();
            consumerProps.setProperty(RocketMQConfig.NAME_SERVER_ADDR, Constant.S
OURCE_NAME_SERVER_ADDR);
            consumerProps.setProperty(RocketMQConfig.CONSUMER_GROUP, "flink_demo"
);
            consumerProps.setProperty(RocketMQConfig.CONSUMER_TOPIC, "BinLogFromC
anal");

            // 2.初始化数据源,对数据源进行映射，过滤，根据表名分割成多个数据流
            SplitStream<BinLogMsgEntity> splitStream = env
                .addSource(new RocketMQSource(
                    new SimpleKeyValueDeserializationSchema(Constant.MQ_CONSTANT_
ID, Constant.MQ_CONSTANT_ADDRESS),
                    consumerProps))
                .name(Constant.FLINK_SOURCE_NAME).setParallelism(1)
                .map(new MapFunction<Map<String, String>, BinLogMsgEntity>() {
                    @Override
                    public BinLogMsgEntity map(Map<String, String> value) throws
 Exception {
                        BinLogMsgEntity msgEntity = JSON.parseObject(value.get(Co
nstant.MQ_CONSTANT_ADDRESS),
                            new TypeReference<BinLogMsgEntity>() {});
                        return msgEntity;
                    }
                }).split(new OutputSelector<BinLogMsgEntity>() {

                    @Override
                    public Iterable<String> select(BinLogMsgEntity value) {
                        List<String> output = new ArrayList<String>();
                        if(value.getTable().equals("out_order")) {
                            output.add("out_order");
                        }else if(value.getTable().equals("out_order_details")) {
                            output.add("out_order_details");
                        }else {
                            output.add("other");
                        }
                        return output;
```

```
                    }});

            DataStream<String> outOrderStream = splitStream.select("out_order").map(new MapFunction<BinLogMsgEntity, String>() {

                @Override
                public String map(BinLogMsgEntity value) throws Exception {
                    return "out_order:"+value.getEs();
                }});
            DataStream<String> outOrderDetailStream = splitStream.select("out_order_details").map(new MapFunction<BinLogMsgEntity, String>() {

                @Override
                public String map(BinLogMsgEntity value) throws Exception {
                    return "out_order_details:"+value.getEs();
                }});
            DataStream<String> otherStream = splitStream.select("other").map(new MapFunction<BinLogMsgEntity, String>() {

                @Override
                public String map(BinLogMsgEntity value) throws Exception {
                    return "other:"+value.getTable()+"——"+value.getEs();
                }});
            // 4.对分割出来的数据流进行打印
            outOrderStream.print();
            outOrderDetailStream.print();
            otherStream.print();

            env.execute("geekplus_dws_etl_job");
        } catch (Exception e) {
            e.printStackTrace();
            logger.error("error:" + e.getMessage());
        }
    }

}
```