消费者基于自定义属性实现SQL过滤

在发送消息时，发送方可以自定义消息的用户属性，消费者可以利用SQL92的WHERE子句语法实现消息过滤。

相比Tag过滤，消息过滤使用更加灵活，也更容易被程序猿接受，但相较Tag过滤执行效率较低。

下面咱们来看案例：

消息生产者

com.itlaoqi.rocketmq.sqlfilter.SfProducer

消息发送方和标准发送有两点变化：

- 可以不设置消息的Tag与Key，转而使用用户自定义属性，这里实现了source与id两个自定义属性的赋值
- 利用message.putUserProperty为用户赋予自定义属性

```
@Slf4j
public class SfProducer {
    public static void main(String[] args) {
        //DefaultMQProducer用于发送非事务消息
        DefaultMQProducer producer = new DefaultMQProducer("sf-producer-group");
        //注册NameServer地址
        producer.setNamesrvAddr("192.168.31.103:9876");
        /*//异步发送失败后Producer自动重试2次
        producer.setRetryTimesWhenSendAsyncFailed(2);*/
        try {
            //启动生产者实例
            producer.start();
            for(Integer i = 0 ; i < 10 ; i++) {
                Thread.sleep(1000);
                Integer rnd = new Random().nextInt(10);
                //用户自定义属性
                String source = "";
                switch (rnd % 3){
                    case 0:
                        source = "jd";
                        break;
                    case 1:
                        source = "tmall";
                        break;
                    case 2:
                        source = "taobao";
                        break;
                }
                //消息数据
                String data = "第" + i + "条消息数据";
                //消息主题，使用用户自定义属性时可以不设置tag与key
                Message message = new Message("sf-sample-data", data.getBytes());
                message.putUserProperty("id" , i.toString());
                message.putUserProperty("source", source);
                //发送结果
                SendResult result = producer.send(message);
                log.info("id:{},source:{},data:{}" ,i.toString(), source,data);
            }
        }catch (Exception e){
            e.printStackTrace();
        }finally {
            try {
                //关闭连接
                producer.shutdown();
                log.info("连接已关闭");
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }
}
```

运行结果

```
09:34:52.771 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:0,source:jd,data:第0条消息数据
09:34:53.788 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:1,source:jd,data:第1条消息数据
09:34:54.801 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:2,source:tmall,data:第2条消息数据
09:34:55.818 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:3,source:tmall,data:第3条消息数据
09:34:56.836 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:4,source:jd,data:第4条消息数据
09:34:57.850 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:5,source:taobao,data:第5条消息数据
09:34:58.865 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:6,source:taobao,data:第6条消息数据
09:34:59.880 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:7,source:jd,data:第7条消息数据
09:35:00.896 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:8,source:tmall,data:第8条消息数据
09:35:01.911 [main] INFO com.itlaoqi.rocketmq.sqlfilter.SfProducer -
id:9,source:taobao,data:第9条消息数据
```

消息消费者

默认RocketMQ并未开启自定义属性SQL过滤的选项，需要在配置文件中额外开启，如下所示：

master.conf:Master节点配置文件追加下面选型：

```
#开启自定义属性SQL过滤
enablePropertyFilter=true
```

完整如下：

```
brokerClusterName=DefaultCluster
brokerName=broker-a
brokerId=0
deleteWhen=04
fileReservedTime=48
brokerRole=SYNC_MASTER
flushDiskType=SYNC_FLUSH
namesrvAddr=192.168.31.103:9876
autoCreateTopicEnable=true
#开启自定义属性SQL过滤
enablePropertyFilter=true
```

slave.conf：Slave节点也要追加该配置项，别忘记

京东消费者

com.itlaoqi.rocketmq.sqlfilter.SfJDConsumer

京东消费者负责消费source='jd'的数据，和标准消费者最大的不同便是在subscribe方法第二个参数不再是Tag，而改为MessageSelector.bySql方法，利用WHERE子句写法对自定义属性实现过滤，源码如下

```
@Slf4j
public class SfJDConsumer {
    public static void main(String[] args) throws Exception {
        DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("sf-jd-
consumer-group");
        consumer.setNamesrvAddr("192.168.31.103:9876");
        consumer.setMessageModel(MessageModel.CLUSTERING);

        //利用SQL WHERE子句写法对自定义属性进行过滤
        consumer.subscribe("sf-sample-data", MessageSelector.bySql("source='jd'"));

       consumer.registerMessageListener(new MessageListenerConcurrently() {
            @Override
            public ConsumeConcurrentlyStatus consumeMessage(List<MessageExt> list,
ConsumeConcurrentlyContext consumeConcurrentlyContext) {
                list.forEach(msg->{
                    log.info("id:{},source:{},data:{}"
,msg.getUserProperty("id"),msg.getUserProperty("source") , new
String(msg.getBody()));
```

```
                });
                return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
            }
        });
        consumer.start();
        log.info("集群消费者启动成功，正在监听新消息");
    }
}
```

运行结果

```
09:34:52.770 [ConsumeMessageThread_2] INFO
com.itlaoqi.rocketmq.sqlfilter.SfJDConsumer - id:0,source:jd,data:第0条消息数据
09:34:53.788 [ConsumeMessageThread_3] INFO
com.itlaoqi.rocketmq.sqlfilter.SfJDConsumer - id:1,source:jd,data:第1条消息数据
09:34:56.836 [ConsumeMessageThread_4] INFO
com.itlaoqi.rocketmq.sqlfilter.SfJDConsumer - id:4,source:jd,data:第4条消息数据
09:34:59.880 [ConsumeMessageThread_5] INFO
com.itlaoqi.rocketmq.sqlfilter.SfJDConsumer - id:7,source:jd,data:第7条消息数据
```

阿里消费者

`com.itlaoqi.rocketmq.sqlfilter.SfAliConsumer`

京东消费者负责消费天与猫淘宝的数据，与京东消费者最明显的区别是：

- 因为业务范围不同，消费者组不一样；
- bySQL的要获取多个数值，可用下面语法
  - source in ('tmall','taobao')
  - source = 'tmall' or source = 'taobao'

```
...
DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("sf-ali-
consumer-group");
...
consumer.subscribe("sf-sample-data", MessageSelector.bySql("source in ('tmall'
,'taobao')"));
```

运行结果

```
09:34:54.803 [ConsumeMessageThread_14] INFO
com.itlaoqi.rocketmq.sqlfilter.SfAliConsumer - id:2,source:tmall,data:第2条消息数据
09:34:55.819 [ConsumeMessageThread_15] INFO
com.itlaoqi.rocketmq.sqlfilter.SfAliConsumer - id:3,source:tmall,data:第3条消息数据
09:34:57.850 [ConsumeMessageThread_16] INFO
com.itlaoqi.rocketmq.sqlfilter.SfAliConsumer - id:5,source:taobao,data:第5条消息数据
09:34:58.865 [ConsumeMessageThread_17] INFO
com.itlaoqi.rocketmq.sqlfilter.SfAliConsumer - id:6,source:taobao,data:第6条消息数据
09:35:00.896 [ConsumeMessageThread_18] INFO
com.itlaoqi.rocketmq.sqlfilter.SfAliConsumer - id:8,source:tmall,data:第8条消息数据
```

```
09:35:01.911 [ConsumeMessageThread_19] INFO
com.itlaoqi.rocketmq.sqlfilter.SfAliConsumer - id:9,source:taobao,data:第9条消息数据
```