

# Flink SQL Client综合实战

在《[Flink SQL Client初探](#)》一文中，我们体验了Flink SQL Client的基本功能，今天来通过实战更深入学习和体验Flink SQL；

## 实战内容

本次实战主要是通过Flink SQL Client消费kafka的实时消息，再用各种SQL操作对数据进行查询统计，内容汇总如下：

1. DDL创建Kafka表
2. 窗口统计；
3. 数据写入ElasticSearch
4. 联表操作

## 版本信息

1. Flink: 1.10.0
2. Flink所在操作系统: CentOS Linux release 7.7.1908
3. JDK: 1.8.0\_211
4. Kafka: 2.4.0 (scala: 2.12)
5. Mysql: 5.7.29

## 数据源准备

1. 本次实战用的数据，来源是阿里云天池公开数据集的一份淘宝用户行为数据集，获取方式请参考《[准备数据集用于flink学习](#)》
2. 获取到数据集文件后转成kafka消息发出，这样我们使用Flink SQL时就按照实时消费kafka消息的方式来操作，具体的操作方式请参考《[将CSV的数据发送到kafka](#)》
3. 上述操作完成后，一百零四条淘宝用户行为数据就会通过kafka消息顺序发出，咱们的实战就有不间断实时数据可用了，消息内容如下：

```
{"user_id":1004080,"item_id":2258662,"category_id":79451,"behavior":"pv","ts":"2017-11-2  
{"user_id":100814,"item_id":5071478,"category_id":1107469,"behavior":"pv","ts":"2017-11-  
{"user_id":114321,"item_id":4306269,"category_id":4756105,"behavior":"pv","ts":"2017-11-
```

1. 上述消息中每个字段的含义如下表：

列名称	说明
用户ID	整数类型，序列化后的用户ID
商品ID	整数类型，序列化后的商品ID
商品类目ID	整数类型，序列化后的商品所属类目ID
行为类型	字符串，枚举类型，包括('pv', 'buy', 'cart', 'fav')
时间戳	行为发生的时间戳
时间字符串	根据时间戳字段生成的时间字符串

## jar准备

实战过程中要用到下面这五个jar文件：

1. flink-jdbc\_2.11-1.10.0.jar
2. flink-json-1.10.0.jar
3. flink-sql-connector-elasticsearch6\_2.11-1.10.0.jar
4. flink-sql-connector-kafka\_2.11-1.10.0.jar
5. mysql-connector-java-5.1.48.jar

我已将这些文件打包上传到GitHub，下载地址：

[https://raw.githubusercontent.com/zq2599/blog\\_demos/master/files/sql\\_lib.zip](https://raw.githubusercontent.com/zq2599/blog_demos/master/files/sql_lib.zip)

请在flink安装目录下新建文件夹sql\_lib，然后将这五个jar文件放进去；

## Elasticsearch准备

如果您装了docker和docker-compose，那么下面的命令可以快速部署elasticsearch和head工具：

```
wget https://raw.githubusercontent.com/zq2599/blog_demos/master/elasticsearch_docker_com
docker-compose up -d
```

准备完毕，开始操作吧；

## DDL创建Kafka表

1. 进入flink目录，启动flink： bin/start-cluster.sh
2. 启动Flink SQL Client： bin/sql-client.sh embedded -l sql\_lib

3. 启动成功显示如下：

```
[root@maven flink-1.10.0]# bin/sql-client.sh embedded -l sql_lib
No default environment specified.
Searching for '/root/flink-1.10.0/conf/sql-client-defaults.yaml'...found.
Reading default environment from: file:/root/flink-1.10.0/conf/sql-client-defaults.yaml
No session environment specified.
```



BETA

# Flink SQL Client

Welcome! Enter 'HELP;' to list all available commands. 'QUIT;' to exit.

[https://blog.csdn.net/boling\\_cavalry](https://blog.csdn.net/boling_cavalry)

Flink SQL> █

4. 执行以下命令即可创建kafka表，请按照自己的信息调整参数：

```
CREATE TABLE user_behavior (
  user_id BIGINT,
  item_id BIGINT,
  category_id BIGINT,
  behavior STRING,
  ts TIMESTAMP(3),
  proctime as PROCTIME(), -- 处理时间列
  WATERMARK FOR ts as ts - INTERVAL '5' SECOND -- 在ts上定义watermark, ts成为事件时间列
) WITH (
  'connector.type' = 'kafka', -- kafka connector
  'connector.version' = 'universal', -- universal 支持 0.11 以上的版本
  'connector.topic' = 'user_behavior', -- kafka topic
  'connector.startup-mode' = 'earliest-offset', -- 从起始 offset 开始读取
  'connector.properties.zookeeper.connect' = '192.168.50.43:2181', -- zk 地址
```

```
'connector.properties.bootstrap.servers' = '192.168.50.43:9092', -- broker 地址
'format.type' = 'json' -- 数据源格式为 json
);
```

1. 执行SELECT \* FROM user\_behavior;看看原始数据, 如果消息正常应该和下图类似:

user_id	item_id	category_id	behavior	ts	proctime
126195	3123300	811638	pv	2017-11-23T15:35:19	2020-05-03T14:30:42.118
126195	3863755	3139583	pv	2017-11-23T15:36:39	2020-05-03T14:30:42.618
126195	4300792	4014446	pv	2017-11-23T15:36:45	2020-05-03T14:30:43.119
126195	3430706	605882	pv	2017-11-23T15:38:11	2020-05-03T14:30:43.619
126195	4300792	4014446	pv	2017-11-23T15:38:53	2020-05-03T14:30:44.120
126195	4879883	1264908	pv	2017-11-23T15:47:49	2020-05-03T14:30:44.620
126195	4950636	3860845	pv	2017-11-23T15:49:42	2020-05-03T14:30:45.121
126195	1263242	605882	pv	2017-11-23T15:50:48	2020-05-03T14:30:45.621
126195	2332151	605882	pv	2017-11-23T15:51:29	2020-05-03T14:30:46.122
121536	2183824	223690	pv	2017-11-23T17:04:03	2020-05-03T14:30:46.624
1014480	3529703	2520377	pv	2017-11-23T20:11:28	2020-05-03T14:30:47.124
1003613	3089876	2355072	pv	2017-11-23T22:18:14	2020-05-03T14:30:47.625
118691	3251448	4462359	pv	2017-11-23T23:10:34	2020-05-03T14:30:48.126
118691	4973304	927764	pv	2017-11-23T23:11:06	2020-05-03T14:30:48.627

[https://blog.csdn.net/boling\\_cavalry](https://blog.csdn.net/boling_cavalry)

## 窗口统计

1. 下面的SQL是以每十分钟为窗口, 统计每个窗口内的总浏览数, TUMBLE\_START返回的数据格式是timestamp, 这里再调用DATE\_FORMAT函数将其格式化成了字符串:

```
SELECT DATE_FORMAT(TUMBLE_START(ts, INTERVAL '10' MINUTE), 'yyyy-MM-dd hh:mm:ss'),
DATE_FORMAT(TUMBLE_END(ts, INTERVAL '10' MINUTE), 'yyyy-MM-dd hh:mm:ss'),
COUNT(*)
FROM user_behavior
WHERE behavior = 'pv'
GROUP BY TUMBLE(ts, INTERVAL '10' MINUTE);
```

1. 得到数据如下所示:

EXPR\$0	EXPR\$1	EXPR\$2
2017-11-24 04:10:00	2017-11-24 04:20:00	5
2017-11-24 04:20:00	2017-11-24 04:30:00	1
2017-11-24 04:30:00	2017-11-24 04:40:00	2
2017-11-24 04:40:00	2017-11-24 04:50:00	1
2017-11-24 04:50:00	2017-11-24 05:00:00	1
2017-11-24 05:10:00	2017-11-24 05:20:00	1
2017-11-24 05:20:00	2017-11-24 05:30:00	1
2017-11-24 05:40:00	2017-11-24 05:50:00	3
2017-11-24 05:50:00	2017-11-24 06:00:00	2
2017-11-24 06:00:00	2017-11-24 06:10:00	2
2017-11-24 06:10:00	2017-11-24 06:20:00	2
2017-11-24 06:20:00	2017-11-24 06:30:00	
2017-11-24 06:40:00	2017-11-24 06:50:00	3

[https://blog.csdn.net/boling\\_cavalry](https://blog.csdn.net/boling_cavalry)

## 数据写入ElasticSearch

1. 确保elasticsearch已部署好;
2. 执行以下语句即可创建es表, 请按照您自己的es信息调整下面的参数:

```
CREATE TABLE pv_per_minute (
start_time STRING,
```

```

end_time STRING,
pv_cnt BIGINT
) WITH (
  'connector.type' = 'elasticsearch', -- 类型
  'connector.version' = '6', -- elasticsearch版本
  'connector.hosts' = 'http://192.168.133.173:9200', -- elasticsearch地址
  'connector.index' = 'pv_per_minute', -- 索引名, 相当于数据库表名
  'connector.document-type' = 'user_behavior', -- type, 相当于数据库库名
  'connector.bulk-flush.max-actions' = '1', -- 每条数据都刷新
  'format.type' = 'json', -- 输出数据格式json
  'update-mode' = 'append'
);

```

1. 执行以下语句, 就会将每分钟的pv总数写入es的pv\_per\_minute索引:

```

INSERT INTO pv_per_minute
SELECT DATE_FORMAT(TUMBLE_START(ts, INTERVAL '1' MINUTE), 'yyyy-MM-dd hh:mm:ss') AS start_time,
DATE_FORMAT(TUMBLE_END(ts, INTERVAL '1' MINUTE), 'yyyy-MM-dd hh:mm:ss') AS end_time,
COUNT(*) AS pv_cnt
FROM user_behavior
WHERE behavior = 'pv'
GROUP BY TUMBLE(ts, INTERVAL '1' MINUTE);

```

1. 用es-head查看, 发现数据已成功写入:

Elasticsearch

http://192.168.133.173:9200/

连接

docker-cluster

集群健康值: green (10 of 10)

概览

索引

数据浏览

基本查询 [+]

复合查询 [+]

数据浏览

所有索引

索引

类型

user\_behavior

字段

▶ end\_time

▶ pv\_cnt

▶ start\_time

查询 5 个分片中用的 5 个. 374 命中. 耗时 0.003 秒

index	type	id	score	start_time	end_time	pv_cnt
pv_per_minute	user_behavior	P4YI23EBxTDXNcKGUPAF	1	2017-11-24 08:10:00	2017-11-24 08:11:00	1
pv_per_minute	user_behavior	P4YI23EBxTDXNcKGUPAZ	1	2017-11-24 08:18:00	2017-11-24 08:19:00	1
pv_per_minute	user_behavior	QYYI23EBxTDXNcKGUPAo	1	2017-11-24 08:21:00	2017-11-24 08:22:00	2
pv_per_minute	user_behavior	RYYI23EBxTDXNcKGUPBT	1	2017-11-24 08:39:00	2017-11-24 08:40:00	2
pv_per_minute	user_behavior	ToYI23EBxTDXNcKGUPCU	1	2017-11-24 09:09:00	2017-11-24 09:10:00	1
pv_per_minute	user_behavior	UIYI23EBxTDXNcKGUPCg	1	2017-11-24 09:16:00	2017-11-24 09:17:00	1
pv_per_minute	user_behavior	U4YI23EBxTDXNcKGUPCw	1	2017-11-24 09:23:00	2017-11-24 09:24:00	1
pv_per_minute	user_behavior	VYYI23EBxTDXNcKGUPC7	1	2017-11-24 09:30:00	2017-11-24 09:31:00	1
pv_per_minute	user_behavior	W4YI23EBxTDXNcKGUPDq	1	2017-11-24 10:03:00	2017-11-24 10:04:00	1
pv_per_minute	user_behavior	ZoYI23EBxTDXNcKGufBA	1	2017-11-24 10:53:00	2017-11-24 10:54:00	2
pv_per_minute	user_behavior	aoYI23EBxTDXNcKGufBY	1	2017-11-24 11:10:00	2017-11-24 11:11:00	2

cooling cavity

## 联表操作

1. 当前user\_behavior表的category\_id表示商品类目, 例如11120表示计算机书籍, 61626表示牛仔裤, 本次实战的数据集中, 这样的类目共有五千多种;
2. 如果我们将这五千多种类目分成6个大类, 例如11120属于教育类, 61626属于服装类, 那么应该有个大类和类目的关系表;
3. 这个大类和类目的关系表在MySQL创建, 表名叫category\_info, 建表语句如下:

```

CREATE TABLE `category_info`(
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `parent_id` bigint ,
  `category_id` bigint ,

```



```
PRIMARY KEY ( `id` )
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

1. 表category\_info所有数据来自对原始数据中category\_id字段的提取，并且随机将它们划分为6个大类，该表的数据请在我的GitHub下载：

[https://raw.githubusercontent.com/zq2599/blog\\_demos/master/files/category\\_info.sql](https://raw.githubusercontent.com/zq2599/blog_demos/master/files/category_info.sql)

2. 请在MySQL上建表category\_info，并将上述数据全部写进去；
3. 在Flink SQL Client执行以下语句创建这个维表，mysql信息请按您自己配置调整：

```
CREATE TABLE category_info (
    parent_id BIGINT, -- 商品大类
    category_id BIGINT -- 商品详细类目
) WITH (
    'connector.type' = 'jdbc',
    'connector.url' = 'jdbc:mysql://192.168.50.43:3306/flinkdemo',
    'connector.table' = 'category_info',
    'connector.driver' = 'com.mysql.jdbc.Driver',
    'connector.username' = 'root',
    'connector.password' = '123456',
    'connector.lookup.cache.max-rows' = '5000',
    'connector.lookup.cache.ttl' = '10min'
);
```

1. 尝试联表查询：

```
SELECT U.user_id, U.item_id, U.behavior, C.parent_id, C.category_id
FROM user_behavior AS U LEFT JOIN category_info FOR SYSTEM_TIME AS OF U.proctime AS C
ON U.category_id = C.category_id;
```

1. 如下图，联表查询成功，每条记录都能对应大类：

Refresh: 1 s

SQL Query Result (Table)  
Page: Last of 74

user_id	item_id	behavior	parent_id	category_id
120774	3154594	pv	3	2465336
118238	4647234	cart	5	4339722
104570	822453	buy	1	384755
11096	2292075	pv	4	3415144
1006015	3154594	pv	3	2465336
120774	3071316	pv	3	2465336
1003509	3095445	pv	6	4801426
115411	441950	pv	6	4672807
1006015	5144418	pv	6	4357323
1006015	4557399	pv	6	4357323
121330	3178710	buy	5	3607361
107916	135088	pv	1	211810
121526	211625	pv	2	1226203

<https://blog.csdn.net/boingboingboy>

2. 再试试联表统计，每个大类的总浏览量：

```
SELECT C.parent_id, COUNT(*) AS pv_count
FROM user_behavior AS U LEFT JOIN category_info FOR SYSTEM_TIME AS OF U.proctime AS C
ON U.category_id = C.category_id
WHERE behavior = 'pv'
GROUP BY C.parent_id;
```

1. 如下图，数据是动态更新的：

Refresh: 1 s

SQL Query Result (Table)  
Page: Last of 1

parent_id	pv_count
5	1137
4	674
6	816
1	725
2	1058
3	1120

[https://blog.csdn.net/boling\\_cavalry](https://blog.csdn.net/boling_cavalry)

2. 执行以下语句，可以在统计时将大类ID转成中文名：

```
SELECT CASE C.parent_id
  WHEN 1 THEN '服饰鞋包'
  WHEN 2 THEN '家装家饰'
  WHEN 3 THEN '家电'
  WHEN 4 THEN '美妆'
  WHEN 5 THEN '母婴'
  WHEN 6 THEN '3C数码'
  ELSE '其他'
END AS category_name,
COUNT(*) AS pv_count
FROM user_behavior AS U LEFT JOIN category_info FOR SYSTEM_TIME AS OF U.proctime AS C
ON U.category_id = C.category_id
WHERE behavior = 'pv'
GROUP BY C.parent_id;
```

1. 效果如下图：

Refresh: 1 s

category_name	pv_count
家装家饰	1104
母婴	1217
家电	1225
服饰鞋包	793
3C数码	882
美妆	749

[https://blog.csdn.net/boling\\_cavalry](https://blog.csdn.net/boling_cavalry)

至此，我们借助Flink SQL Client体验了Flink SQL丰富的功能，如果您也在学习Flink SQL，希望本文能给您一些参考；