

# StreamCQL在携程的使用

数据平台 - 实时平台  
郑志锋

I. StreamCQL介绍

II. 用例演示

III. 使用与改造

# StreamCQL介绍

StreamCQL是华为开源的一个可持续数据流查询语言，与传统的SQL执行一次返回一次结果不同，提交之后的作业可以基于时间/长度划分数据流为多个窗口持续输出结果, 或者基于消息事件直接输出结果。语法与SQL类似，底层的计算基于Storm，数据完全缓存于内存中。对于想使用流式计算的人来说，能够简单的上手实现复杂的逻辑，相比于直接编写storm程序也更加效率。

主要特性:

- 可持续性查询

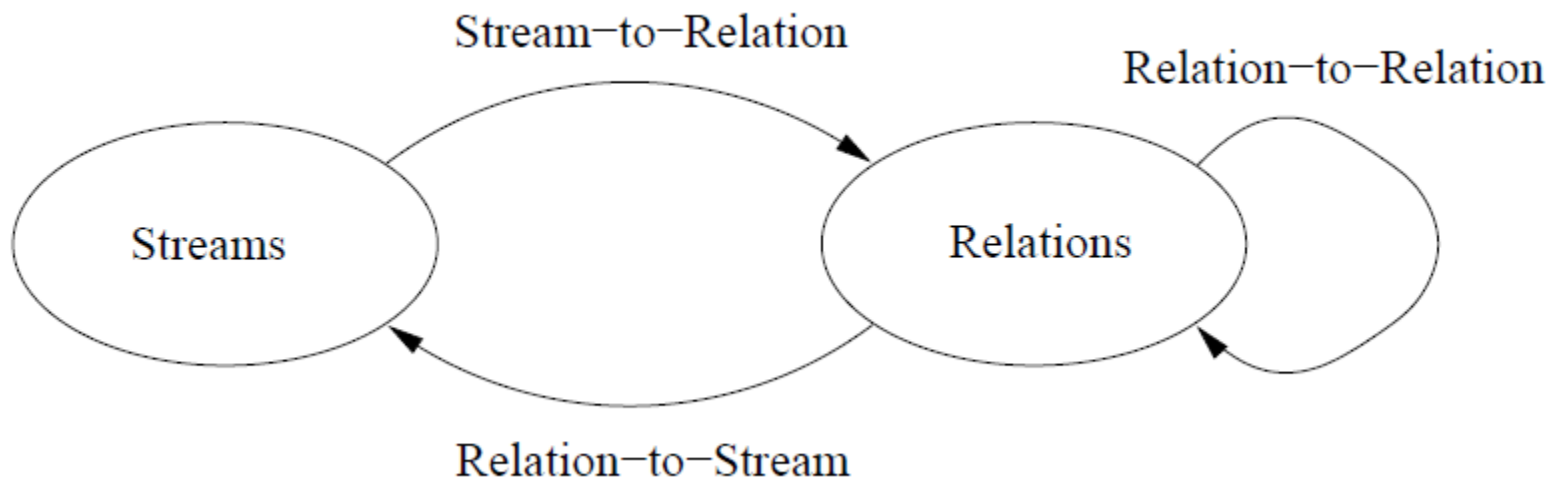
- 时间/长度窗口

- 流计算的抽象

# StreamCQL介绍

## 可持续性查询

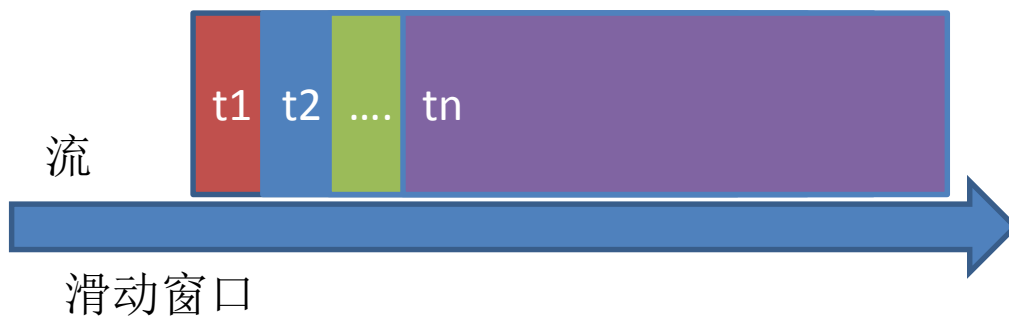
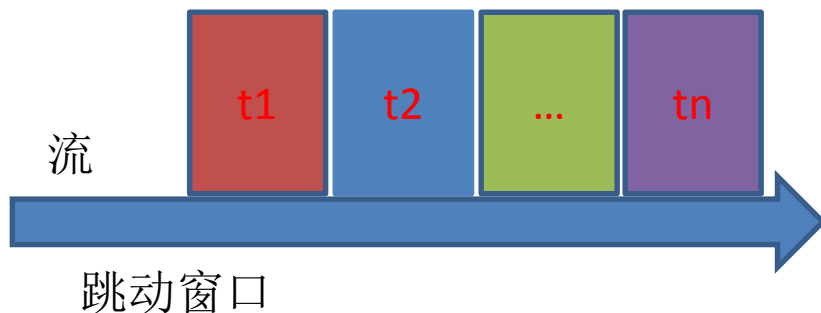
- 流: 一个流就是一组(无穷)元素的集合,  $\langle s, t \rangle$  表示元组  $s$  在时间  $t$  到达流.
- 关系: 一个关系就是一个随时间变化的数据集



# StreamCQL介绍

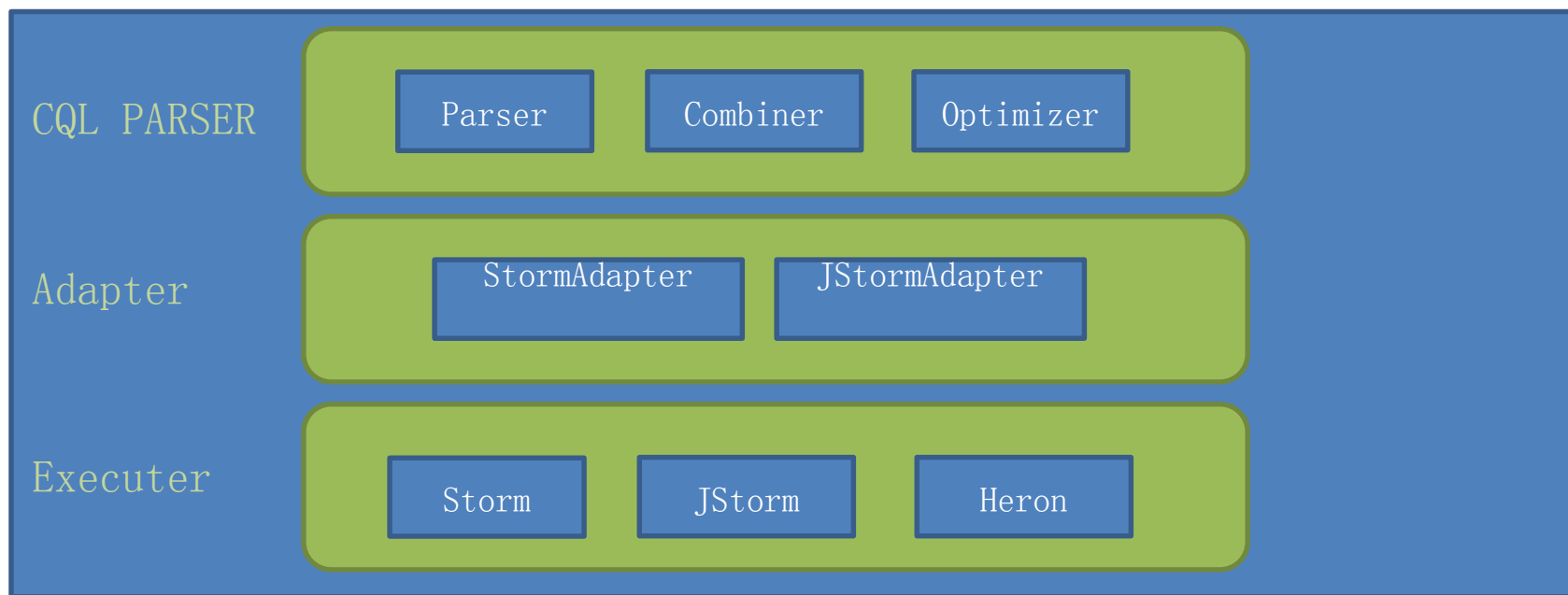
## 时间/长度窗口

- 窗口是有限范围内，某个时间点的数据集合，可以理解为流的边界



# StreamCQL介绍

## 流计算的抽象

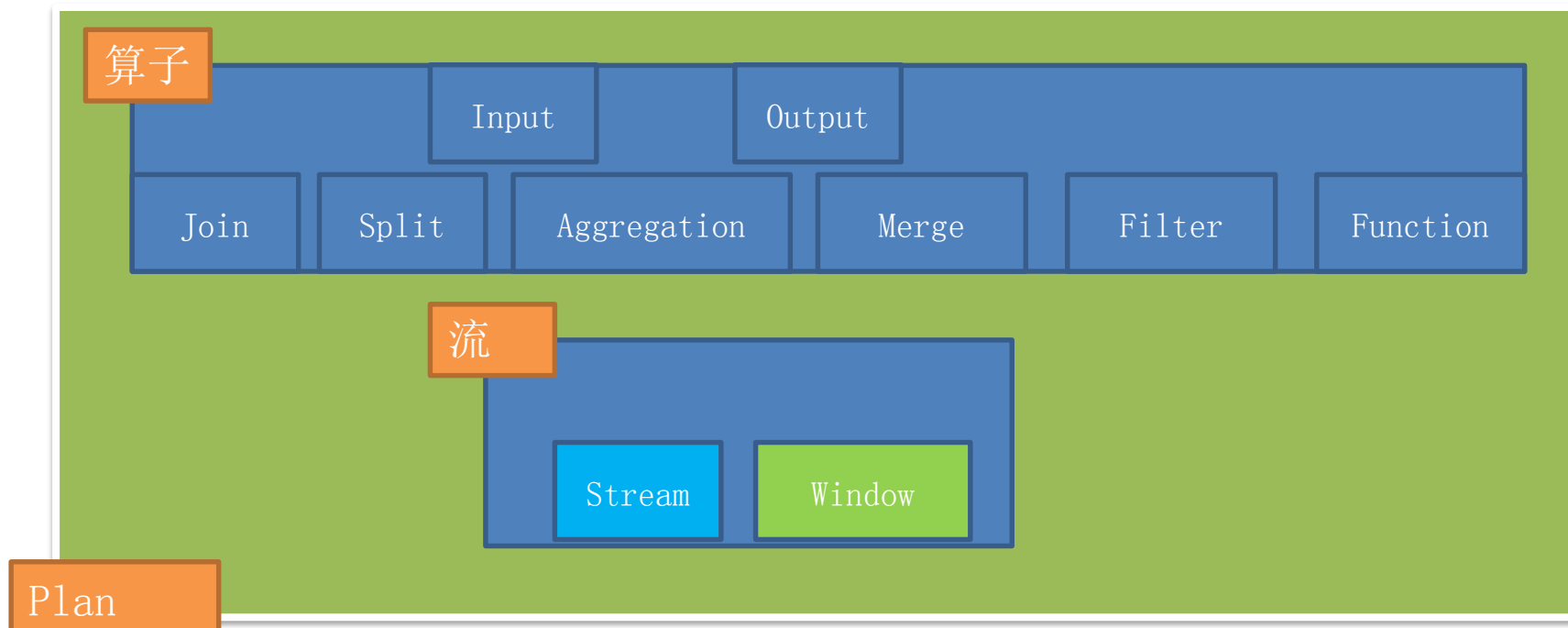


Cql Parser: 基于ANTLR4定义的一类SQL语法解析器, 用于将输入的cql语句作业一个执行计划

Adapter: 将框架产生的算子适配为底层运行的计算平台组件

Executer: 实际执行计算的处理平台, 比如storm, jstorm, heron等

# 用例演示



算子:StreamCQL内部的抽象处理单元，经过Storm适配器的转换后，除了Input算子对应Spout，其他每个算子会被放入一个bolt中。算子内部的处理过程是分层的，每层的具体计算由表达式来完成。

流: 对应storm中的shuffle group 或者field group。

Plan:经过适配器转换会变成topology。

# 用例演示

- 在用户浏览的信息中，筛选出旅游产品的PV, 查找出产品所途径的城市，更新放入Hbase。以下是cql的实现步骤
  1. 创建一个输入流, 提供用户的浏览信息
  2. 筛选出浏览的页面为旅游产品
  3. 创建hive数据源，用于查询产品信息
  4. 查找产品途径目的地城市
  5. 按UID+日期组合行键，用于从hbase中查找历史数据
  6. 创建hbase数据源
  7. 查找历史数据
  8. 创建hbase输出流
  9. 按渠道分流进入online, 更新列表值



# 用例演示

- 创建一个输入流, 把消息队列里的avro格式的数据提取出用户的浏览信息

```
create input stream trancelog (context__vid String, custom__key
String, custom__data__data String, user__id String, context__clientId String)
serde HermesSerDe source HermesSourceOp
properties(avroclass="hermes.ubt.custom.CustomEvent",
topic="ubt.custom", groupid="va_detail_destcityshbase_reader")
```

Trancelog : 输入流的名字, 由括号内的字段构成 , 类似于sql语句中的表。

Source HermesSourceOp: 获取输入数据的组件, 之后将数据交给serde, 将输出的字段放入流中, properties提供必要

Serde HermeSerDe: 使用HermeSerDe组件反序列化输入的数据, 得到声明所需要的字段



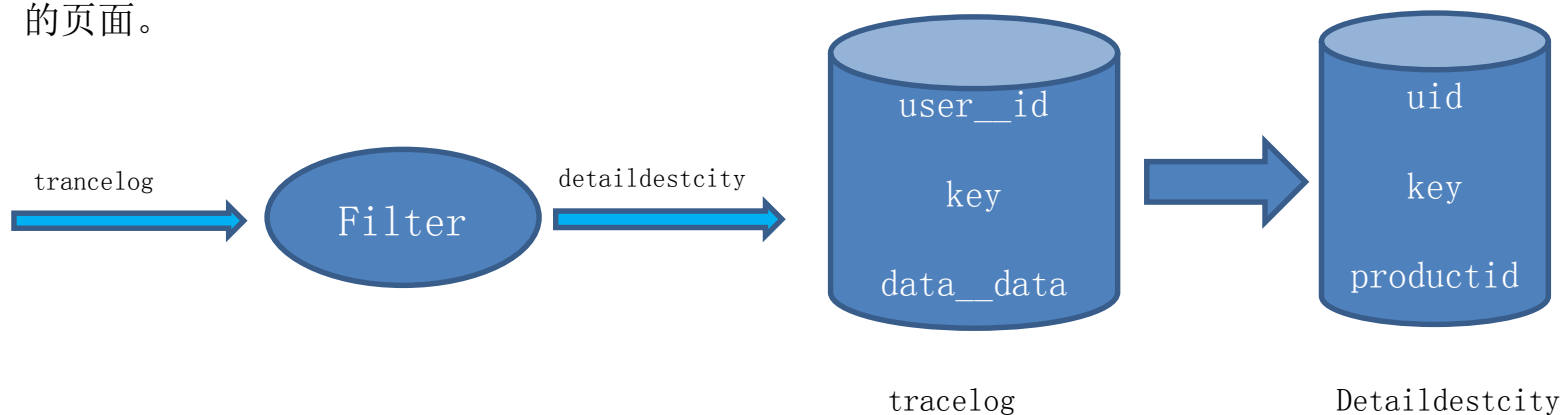
# 用例演示

- 筛选出浏览的页面为旅游产品, 从json字符串中提取productid。

```
insert into stream detaildestcitys select user__id as uid, custom__key as
key, jsonobj(custom__data__data, 'productid') as productid from trancelog
where user__id != "" and custom__key in
('pkg_detail_view_online_basic', 'pkg_detail_view_h5_Basic', 'pkg_detail_view_app_Basic',
'pkg_detail_view_h5_basic', 'pkg_detail_view_app_basic') and
jsonobj(custom__data__data, 'productid') != "" parallel 10;
```

Where条件: 对流中的数据过滤, 筛选出旅游产品的页面。

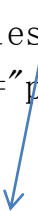
执行的并发度



# 用例演示

- 创建hive数据源，用于查询产品信息

```
create datasource productdb source FileDataSource  
properties(tablename="vadmdb.ddestcitys",  
rowkeys="productid",user="vadm",columns="productid,destcitys");
```



DataSource productDB: 需要用到外部数据源，创建一个DataSource流. 这是一个被动的流，有输入的情况下才会有输出。



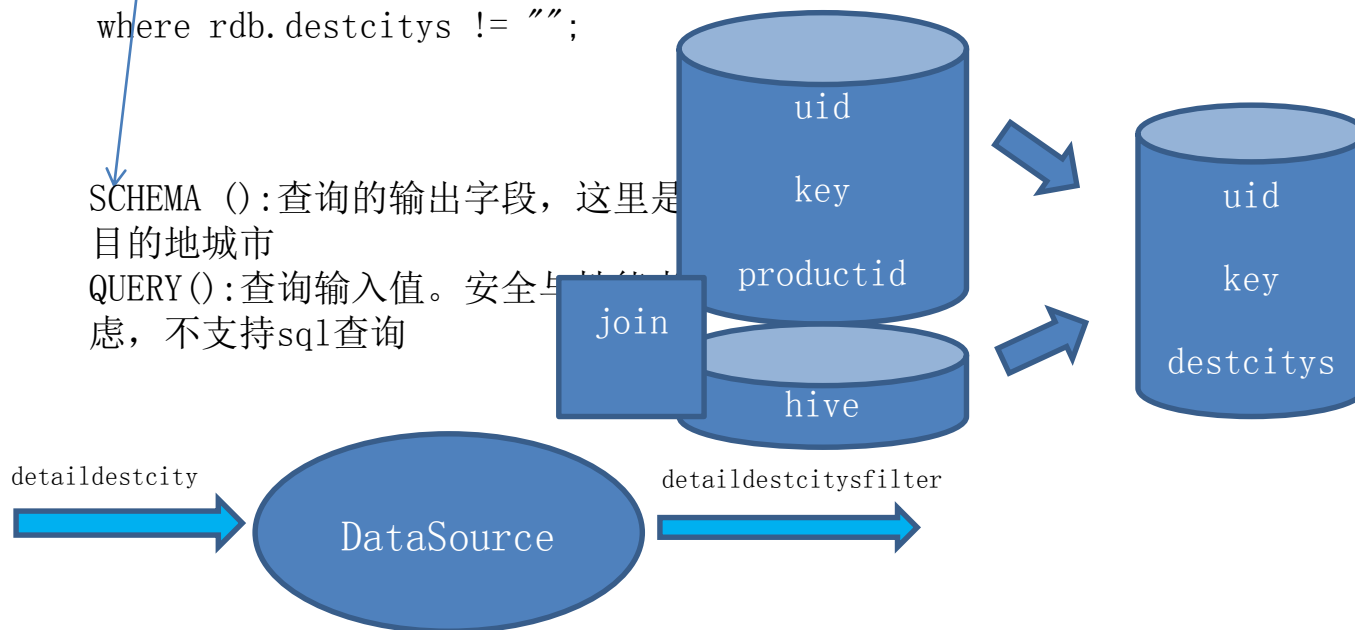
DataSource

# 用例演示

- 查找产品途径目的地城市, 根据productid与上一步生成的hive算子做join, 查找目的城市。

```
insert into stream detaildestcitysfiler select uid,key,rdb.destcitys as
destcitys from detaildestcitys, DATASOURCE productdb
[ SCHEMA (destcitys String), QUERY("rowkeys=?",productid)] rdb
where rdb.destcitys != "";
```

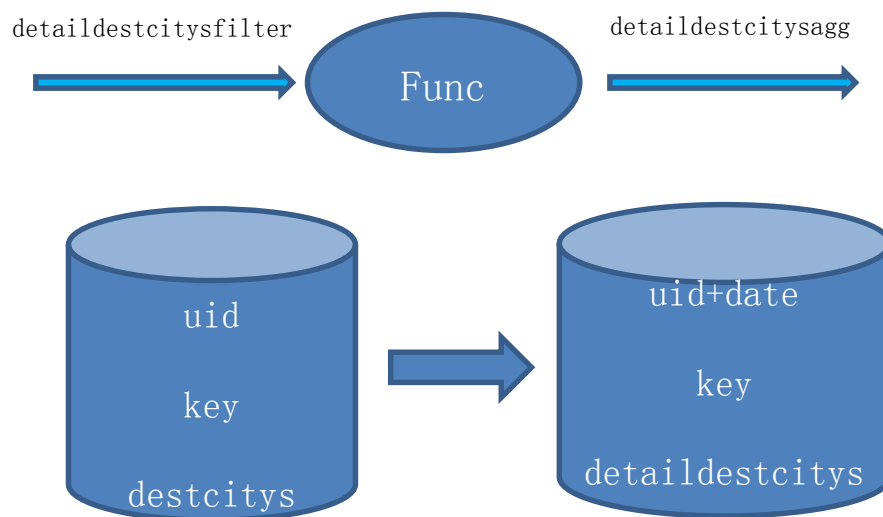
SCHEMA (): 查询的输出字段, 这里是目的地城市  
 QUERY(): 查询输入值。安全与稳定性考虑, 不支持sql查询



# 用例演示

- 按UID+日期组合行键，用于从hbase中查找历史数据

```
insert into stream detaildestcitysagg  
select concat(uid,'#',currentdate('yyyyMMdd'))  
as userid,key,destcitys as detaildestcitys from detaildestcitysfilter;
```



- 创建hbase数据源

```
create datasource hbasedb source HBaseDataSource  
properties(hbase.rowkey.field="userid",  
hbase.kerberos="false",  
hbase.table.name="baseprofile",  
hbase.family.name="f");
```

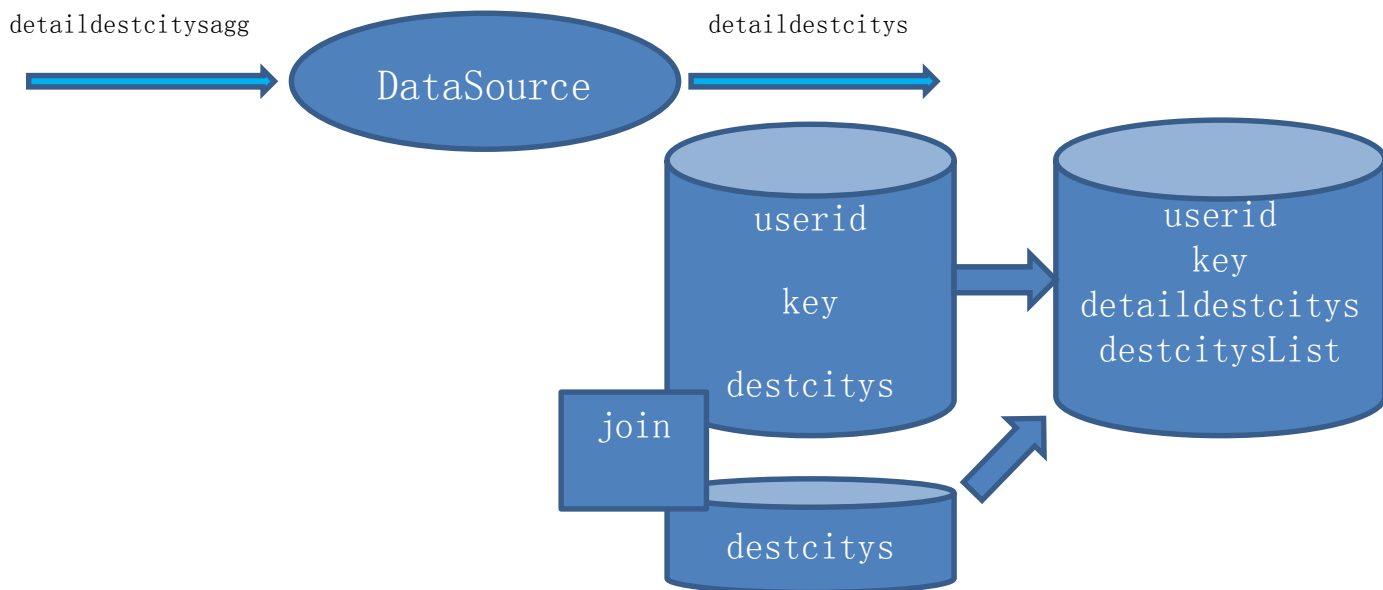


DataSource

# 用例演示

- 查找历史数据, 根据uid+date组合成的rowkey查找用户当天的浏览记录, 返回城市列表。

```
insert into stream detaildestcityslp select  
userid, key, detaildestcitys, hdb.va_detail_destcitys_ol  
from detaildestcitysagg, DATASOURCE hbasedb  
[ SCHEMA (va_detail_destcitys_ol String), QUERY("rowkeys=?",userid) ] hdb;
```

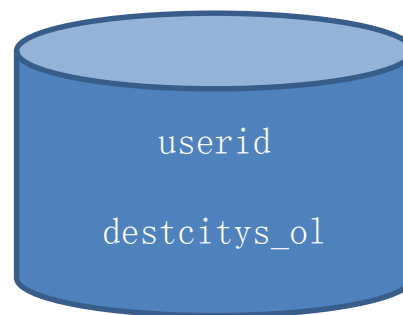


# 用例演示

- 创建hbase输出流, 用于输出访问列表

```
create output stream hbase_output (userid String, va_detail_destcitys_ol)
sink HBaseFunctionOp properties(hbase.rowkey.field="userid",
hbase.kerberos="false",
hbase.table.name="va_userpf",
hbase.family.name="f") parallel 6;
```

Sink: 用于自动写入外部数据源时所用到的组件, 这里用的是hbase, 不需要序列化, 所以没有指定serde, 使用系统默认的序列化组建



hbase\_output

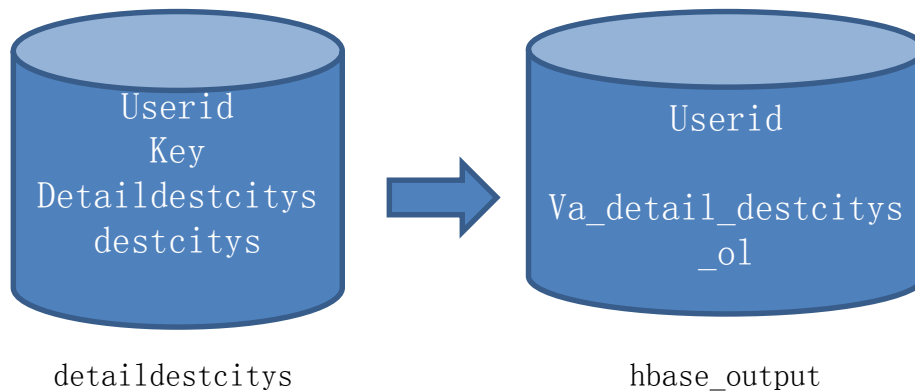
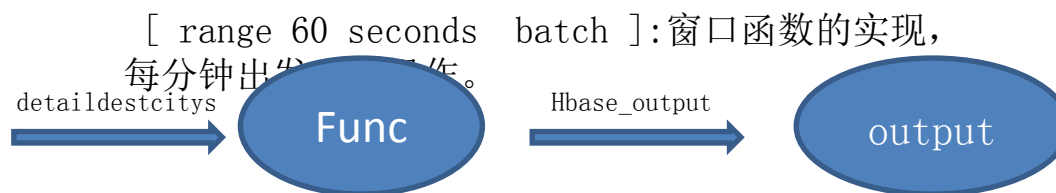


# 用例演示

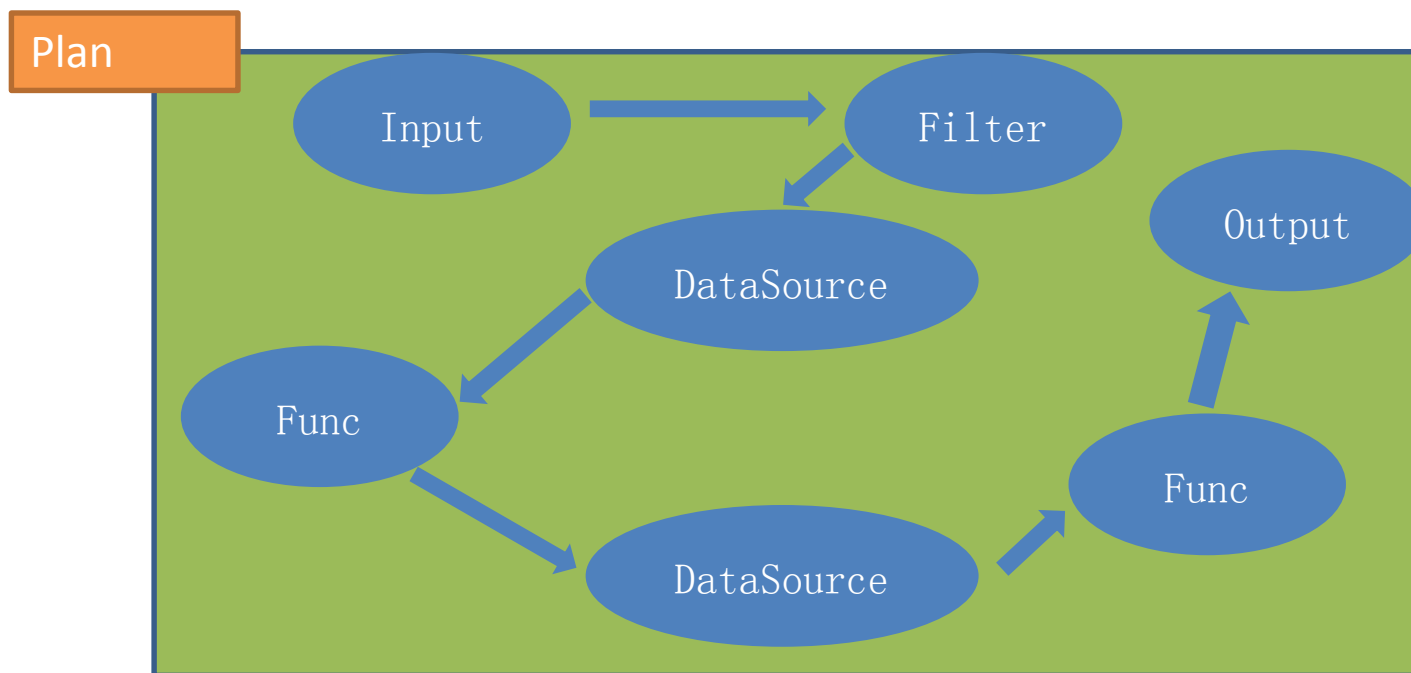
- 按渠道分流进入online, 更新列表值

```
insert into stream hbase_output select
userid, concat(detaildestcitys, "|", va_detail_destcitys_ol)
```

```
From detaildestcitys where key='pkg_detail_view_online_basic' and
va_detail_destcitys_ol is not null
```



# 用例演示



在经过以上的几个步骤之后，CQLParser再经过部分优化和删减冗余的算子之后得到上图这样的执行计划，之后调用Adpater将执行计划转换为实际计算平台上运行的Topology

# 使用与改造

## ● 使用

### — 交互

我们提供了一个portal页面替代了命令行式的交互。

### — 资源控制

对提交作业的worker数量进行审核。

### — 使用场景

面向BI等对SQL较熟悉又想使用流式处理的人员，应用于用户画像，实时推荐等项目的数据清洗，特征抽取。

- 交互
  1. worker数量
  2. 是否开启ack
  3. Topic name
  4. 语法检查
  5. 本地化运行

# 使用与改造

名称

hotel

vacation\_cql

是否做容灾 ☐

描述

用户标签

Topology配置

workerNum

3



ackable

false

topicName

ubt.pageview

sql

```
create input stream pageview (context__page String,context__url String,user__id String)
  comment "这里使用kafka作为输入组建,上线后替换为hermes"
  serde HermesSerDe
  source HermesSourceOp properties(avroclass="hermes.ubt.pageview.Pageview",topic="ubt.pageview",groupid="pageView_cql_reader");
```

;创建hbase输出流,用于输出访问次数

```
create output stream hbase_output (type1_count Int , type2_count Int, type3_count Int,userid String) sink HBaseFunctionOp properties(
  hbase.rowkey.field="userid",
  hbase.kerberos="false",
  hbase.table.name="va_userpf_baseprofile",
  .. .. .)
```

详情

下载jar

删除jar

✓ 更新

↺ 取消

# 使用与改造

## • 本地化运行

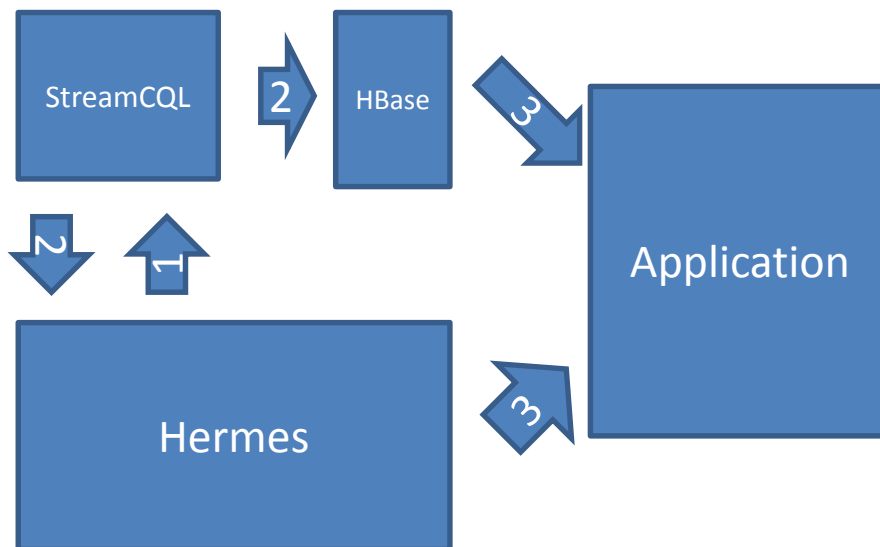
我们开发了控制台输入算子，文件输入算子，用户可以使用这两个算子来模拟消息队列的数据发送，结合控制台输出算子用于排错，或者直接将用控制台输出算子替换掉Hbase等需要落地的算子检查作业的功能是否正常，之后，在portal上提交时再替换成Hbase算子。

```
→ cql-local-1.0 git:(ackmode) X sh run.sh ~/test_cql
开始执行cql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zzf/workspace/StreamCQL/cql-local/target/cql-local-1.0/cql_repo/log4j-slf4j-impl-2.1.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zzf/workspace/StreamCQL/cql-local/target/cql-local-1.0/cql_repo/logback-classic-1.0.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
create input stream kafka_avro
(context__page String,context__type String,action__category String ,action__type String)
  serde hermesSerde
    source HermesSourceOp
      properties(avroclass="hermes.ubt.action.UserAction",
        topic="ubt.action",
        groupid="json_hermes")
编译完成
使用环境文件:streaming-local.xml
工作进程数量:1
是否启用ack: false
是否开启debug:false
8218 [main] INFO  c.h.s.c.s.p.ApplicationParser - start to parse cql : create input stream kafka_avro
(context__page String,context__type String,action__category String ,action__type String)
  serde hermesSerde
    source HermesSourceOp
      properties(avroclass="hermes.ubt.action.UserAction",
        topic="ubt.action",
        groupid="json_hermes")
8303 [main] INFO  c.h.s.c.s.p.ApplicationParser - Parse Completed
8318 [main] INFO  c.h.s.c.t.LazyTask - start to execute CREATE INPUT STREAM kafka_avro (context__page STRING, context__type STRING, action__category STRING, action__type STRI
```

# 使用与改造

## • 使用场景

APP上的旅游目的地推荐项目，用于特征抽取，计算用户的实时行为。



# 使用与改造

我们在使用StreamCQL的过程中，也对其做了部分改造以适用携程内部的使用环境，包括一些功能的补足，易用性的增强，同时对碰到的bug做修复反馈社区。

## ● 改造

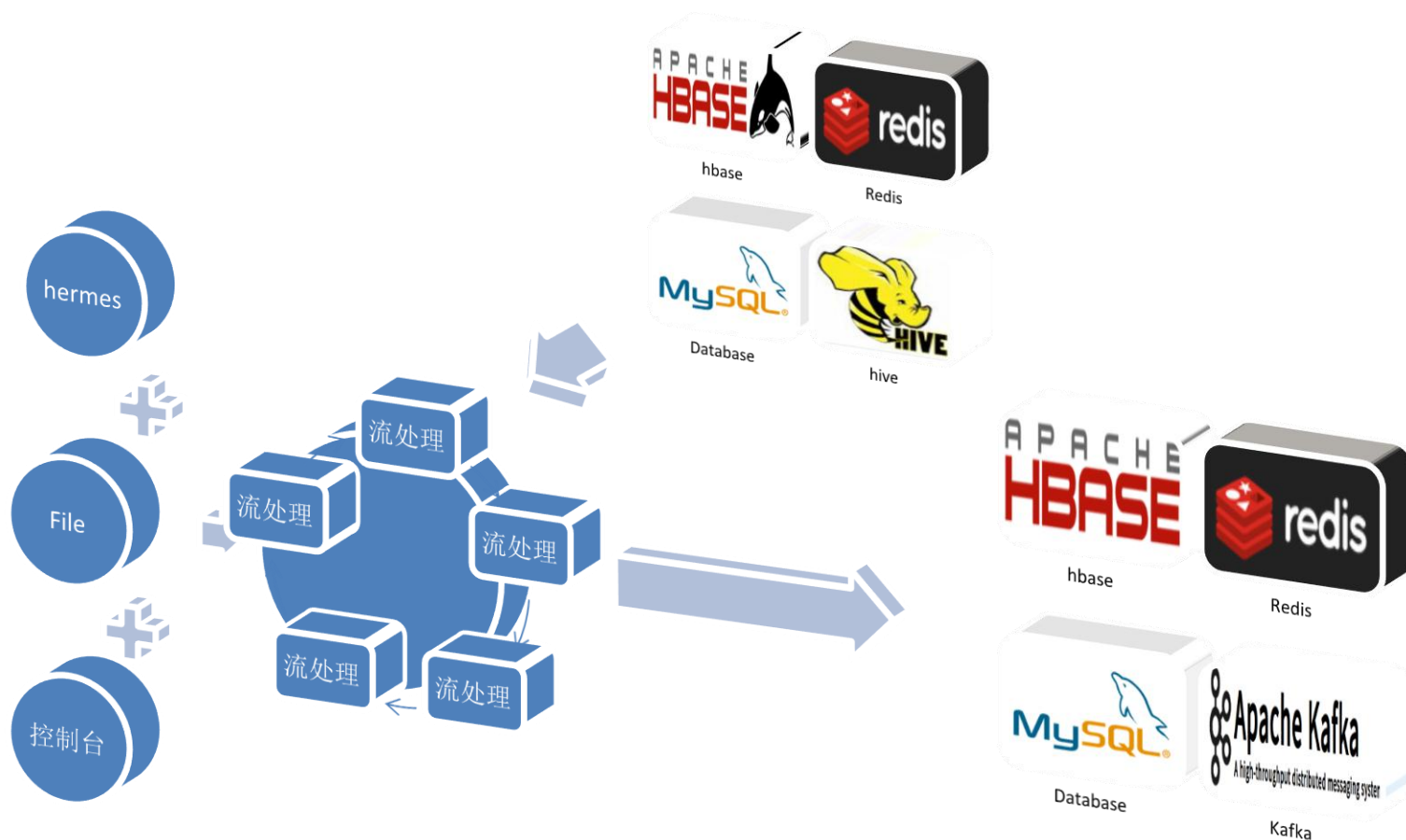
- 语法功能补全，增加In/like语法
- 增加易用性
  1. portal界面
  2. 本地化运行
  3. avro/json结构化数据的支持
  4. 添加一些常用的函数，比如日期，大小写转换，默认值返回等。



# 使用与改造

- 适配携程内的各个数据源
  - 1. Hermes的输入输出算子
  - 2. Hbase的DataSource与输出算子
  - 3. Hive的DataSource算子
  - 4. Mysql的DataSource与输出算子
  - 5. Redis的DataSource与输出算子
  - 6. 用于本地运行的控制台输入输出算子
  - 7. 用于本地运行的文件输入算子

# 使用与改造



# 使用与改造

- Bug修正以及反馈社区
  1. Multi inert语句功能错误
  2. 创建不使用的冗余算子会导致storm作业工作不正常
  3. Self join只支持Input Stream
- 底层对Ack功能的支持
- Jstorm平台的适配
  1. Thirft生成接口名字不一致的问题
  2. 序列化配置信息错误
  3. Jstorm-core与storm-core两个jar包无法共存



Thank you