实时热门商品

数据结构

列名称	说明
用户ID	整数类型，加密后的用户ID
商品ID	整数类型，加密后的商品ID
商品类目ID	整数类型，加密后的商品所属类目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，单位秒

DataStream

Itemid:1 ts:10:01

Itemid:1 ts:10:06

Itemid:1 ts:10:11

Itemid:1 ts:10:16

Itemid:2 ts:10:01

Keyby:itemid

KeyedStream

Itemid:1 ts:10:01 | Itemid:1 ts:10:06 | Itemid:1 ts:10:11 | Itemid:1 ts:10:16

Itemid:2 ts:10:01

SlidingWindow [10:00, 11:00)

SlidingWindow [10:05, 11:05)

Aggregate

Itemid:1 windowend:11:00 count:4

Itemid:1 windowend:11:05 count:3

Itemid:1 windowend:11:10 count:2

Itemid:1 windowend:11:15 count:1

Itemid:2 windowend:11:00 count:1

Keyby:windowend

Itemid:1 windowend:11:00 count:4

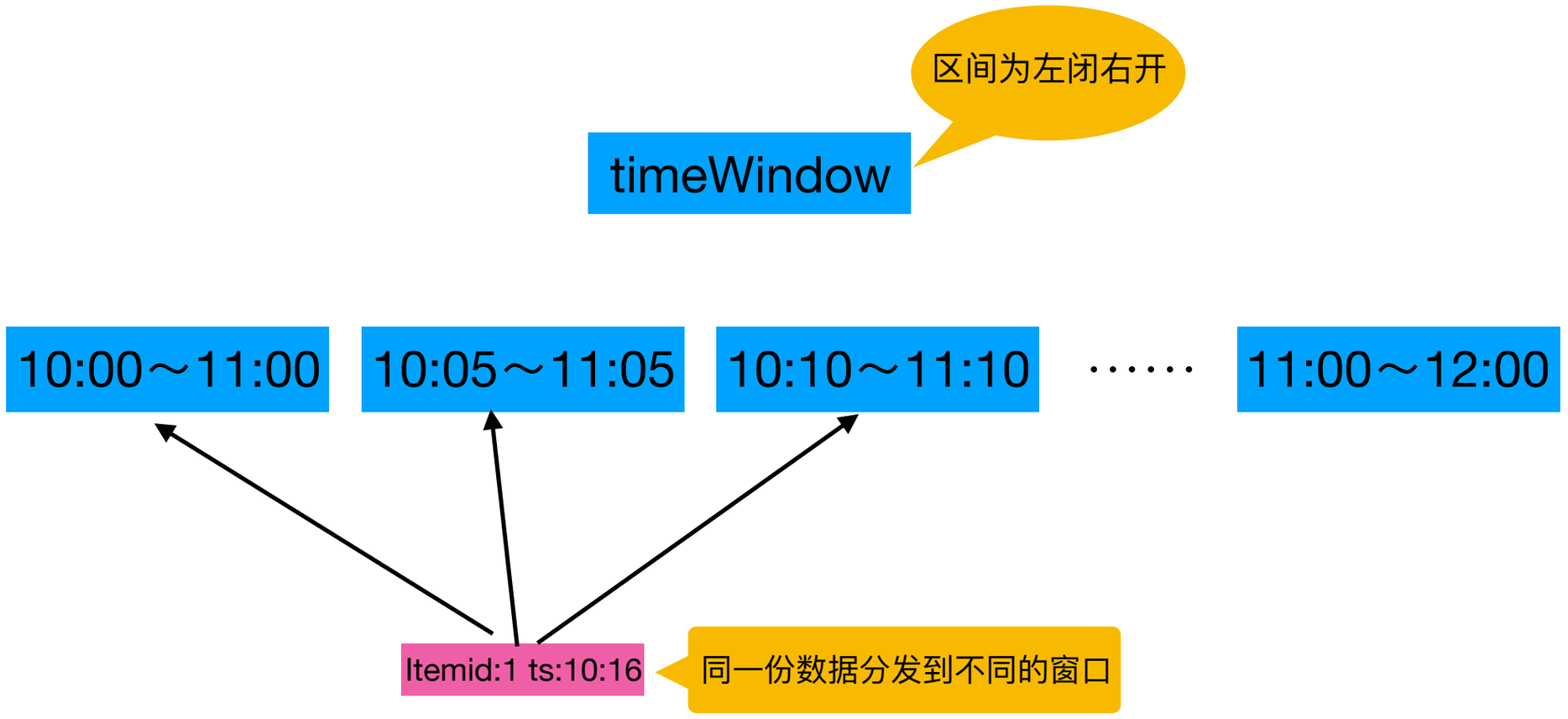
Itemid:1 windowend:11:05 count:3

Itemid:1 windowend:11:10 count:2

Itemid:1 windowend:11:15 count:1

Itemid:2 windowend:11:00 count:1

ProcessFunction
窗口内排序

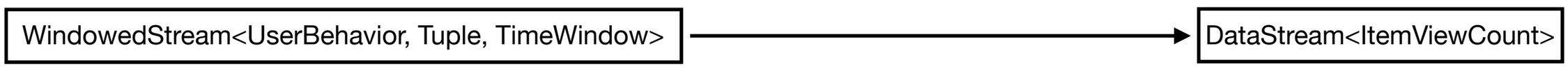

区间为左闭右开
timeWindow
10:00～11:00
10:05～11:05
10:10～11:10
……
11:00～12:00
Itemid:1 ts:10:16
同一份数据分发到不同的窗口

窗口聚合策略

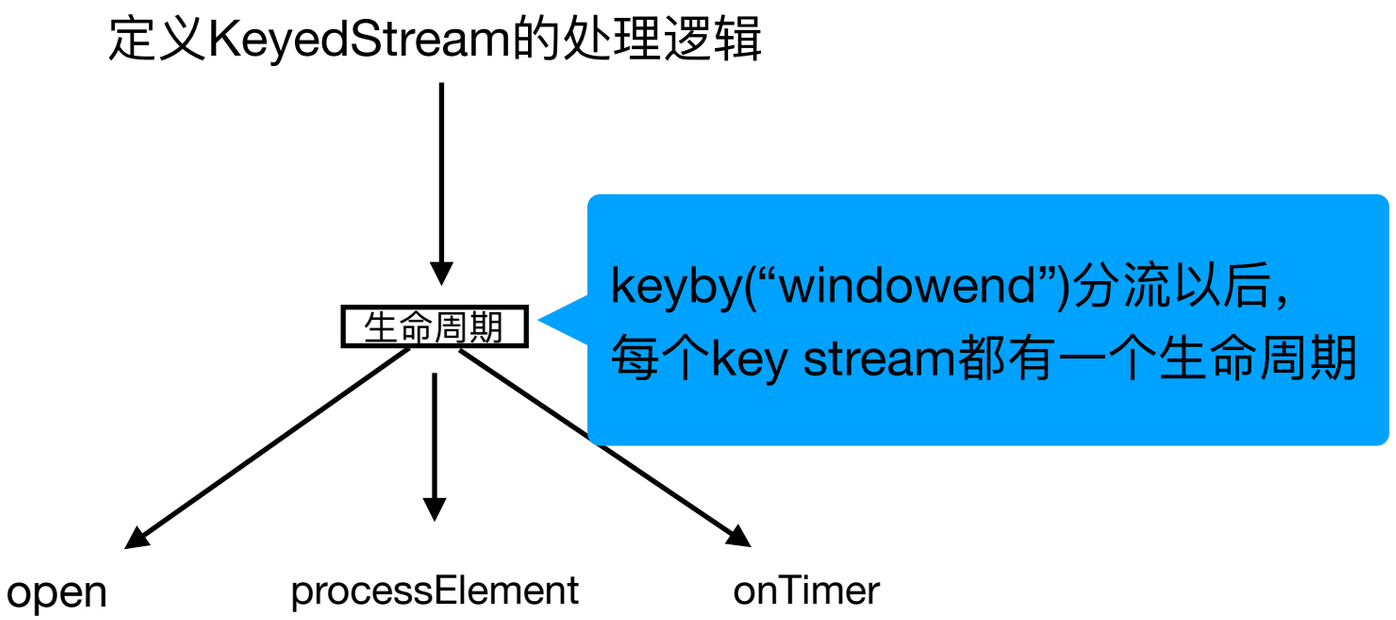
```
.aggregate(new CountAgg(), new WindowResultFunction())
```

定义窗口聚合规则：每出现一条记录就加一

定义输出数据结构



KeyedProcessFunction



open：生命周期初始化


```
ListStateDescriptor<ItemViewCount> itemsStateDesc = new ListStateDescriptor<>(
  "itemState-state",
  ItemViewCount.class);
itemState = getRuntimeContext().getListState(itemsStateDesc);
```

DataStream通过keyby分流以后，每个流都有一个生命周期，
生命周期初始化时，创建一个状态：ListState，用来存储数据

processElement：处理流中的每一个元素时调用

```
// 每条数据都保存到状态中
itemState.add(input);
// 注册 windowEnd+1 的 EventTime Timer, 当触发时, 说明收齐了属于windowEnd窗口的所有商品数据
context.timerService().registerEventTimeTimer(input.windowEnd + 1);
```

将每一个元素都添加到itemState中去

一旦碰到Watermark大于等于input.windowEnd+1的Watermark，说明以input.windowEnd结尾的窗口的数据都已经收集完成，调用回调函数onTimer

onTimer:定时器

```
// 获取收到的所有商品点击量
List<ItemViewCount> allItems = new ArrayList<>();
for (ItemViewCount item : itemState.get()) {
  allItems.add(item);
}
// 提前清除状态中的数据，释放空间
itemState.clear();
// 按照点击量从大到小排序
allItems.sort(new Comparator<ItemViewCount>() {
  @Override
  public int compare(ItemViewCount o1, ItemViewCount o2) {
    return (int) (o2.viewCount - o1.viewCount);
  }
});
```

当窗口收集到所有数据，触发onTimer定时器