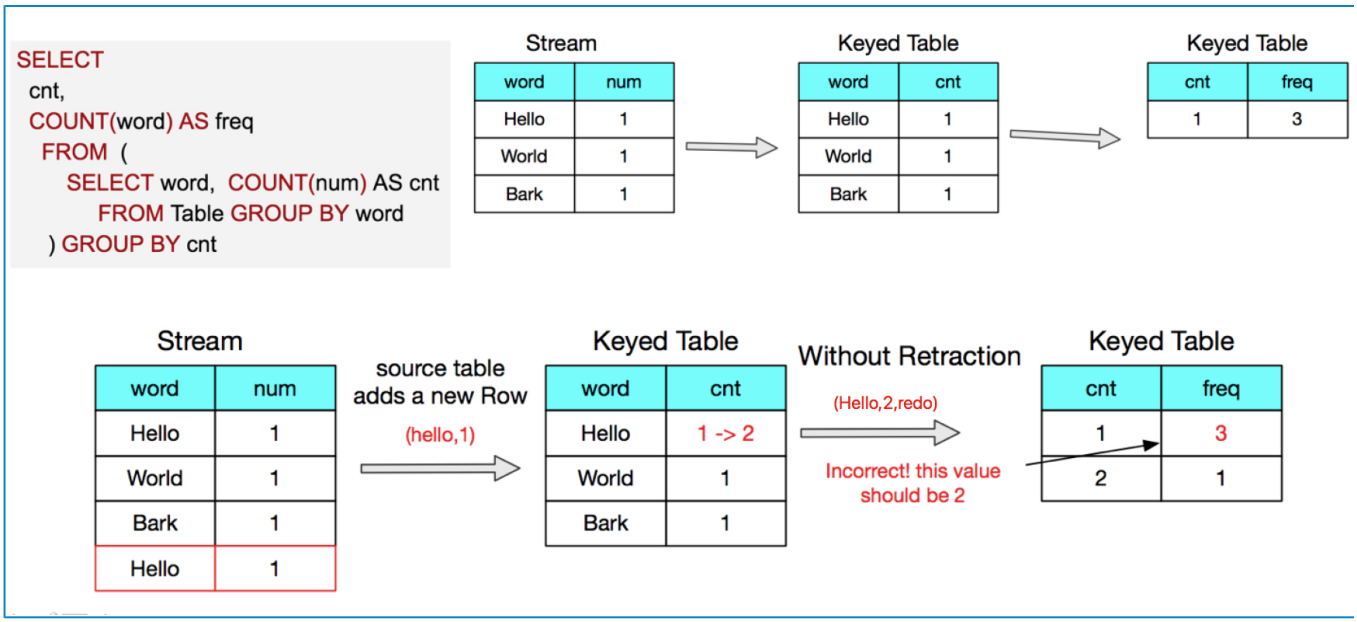
Flink SQL 功能解密系列 —— 流计算“撤回(Retraction)”案例分析

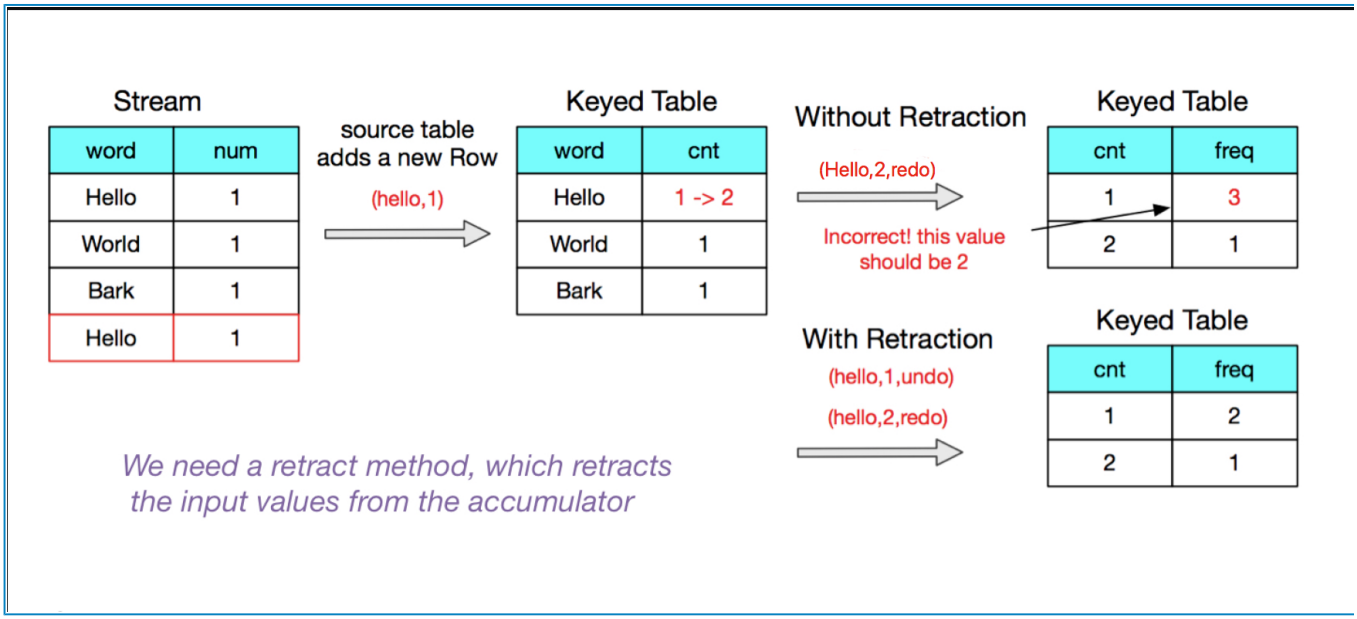
摘要： 通俗讲retract就是传统数据里面的更新操作，也就是说retract是流式计算场景下对数据更新的处理方式。

什么是retraction（撤回）

通俗讲retract就是传统数据里面的更新操作，也就是说retract是流式计算场景下对数据更新的处理
方式。
首先来看下流场景下的一个词频统计列子。



没有retract会导致最终结果不正确↑：



retract发挥的作用

下面再分享两个双十一期间retract保证数据正确性的业务case:

case1： 菜鸟物流订单统计

同一个订单的商品在运输过程中，因为各种原因，物流公司是有可能从A变成B的。为了统计物流公司承担的订单数目，菜鸟团队使用blink计算的retraction机制进行变key汇总操作。

```
-- TT source_table 数据如下:
order_id      tms_company
0001           中通
0002           中通
0003           圆通

-- SQL代码
create view dwd_table as
select
    order_id,
    StringLast(tms_company)
from source_table
group by order_id;

create view dws_table as
select
    tms_company,
    count(distinct order_id) as order_cnt
from dwd_table
group by tms_company


此时结果为:
```

```
tms_company  order_cnt
中通          2
圆通          1

----------------------
之后又来了一条新数据 0001的订单 配送公司改成 圆通了。这时，第一层group by的会先向下游发送

order_id      tms_company
0001           中通
0002           中通
0003           圆通
0001           圆通

写入ADS结果会是（满足需求）
tms_company  order_cnt
中通          1
圆通          2
```

case2： 天猫双十一购物车加购统计:

双11爆款清单与知名综艺IP“火星情报局”跨界合作，汪涵、撒贝宁、陶晶莹等大咖主持加盟，杭州、长沙两地联播，成功打造为“双11子IP”与“双11购物风向标”，树立电商内容综艺化、娱乐化创新典范，为长线模式探索打下基础。

首次深度联动线下场景，在银泰门店落地爆款清单超级大屏，商场人流截停率28%，用户互动时间占营业时间的40%。

选品模式创新，打造最全维度爆款清单：TOP2000性价比爆款+TOP100小黑盒推荐（新品清单）+TOP200买手天团推荐（人群/场景/地域 清单）

核心业务指标

- 加购金额
- 加购件数
- 加购UV

业务计算逻辑

- 来自TT的数据要进行去重；
- 以投放场景和购物车维度进行分组，获取每个分组的最后一条（最新）数据；
- 以投放场景和小时为维度进行分组，统计 加购金额，加购件数和加购UV 业务指标；

业务BlinkSQL代码

```sql
--Blink SQL
--********************************************************************--
--Comment: 天猫双11官方爆款清单统计计算
--********************************************************************--
CREATE TABLE dwd_mkt_membercart_ri(
    cart_id      BIGINT, -- '购物车id',
    sku_id       BIGINT, -- '存放商品的skuId，无sku时，为0',
    item_id      BIGINT, -- '外部id：商品id或者skuid',
    quantity     BIGINT, -- '购买数量',
    user_id      BIGINT, -- '用户id',
    status       BIGINT, -- '状态1：正常-1：删除',
    gmt_create   VARCHAR, -- '属性创建时间',
    gmt_modified VARCHAR, -- '属性修改时间',
    biz_id VARCHAR, -- 投放场景,
    start_time VARCHAR, -- 投放开始时间
    end_time VARCHAR, -- 投放结束时间
    activity_price_time VARCHAR, -- 活动开始时间
    price VARCHAR, -- 商品价格
    dbsync_operation BIGINT -- 时间自动用于排序
)
WITH
(
    type='tt'
    -- 其他信息省略
);

--groub by 方式重复，防止TT重发
CREATE VIEW distinct_dwd_mkt_membercart_ri AS
SELECT
    cart_id,
    sku_id,
    item_id,
    quantity,
    user_id,
    status,
    gmt_create,
    gmt_modified,
    biz_id,
    start_time,
    end_time,
    activity_price_time,
    price,
    dbsync_operation
FROM
    dwd_mkt_membercart_ri
GROUP BY
    cart_id,
    sku_id,
```

```
    item_id,        quantity,
    user_id,
    status,
    gmt_create,
    gmt_modified,
    biz_id,
    start_time,
    end_time,
    activity_price_time,
    price,
    dbsync_operation;

-- 每个投放和购物车数据的最后一条
CREATE VIEW tmp_dwd_mkt_membercart_ri AS
SELECT
    biz_id as biz_id,
    LAST_VALUE(user_id,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation) as
    LAST_VALUE(item_id,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation) as
    LAST_VALUE(sku_id,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation) as s
    LAST_VALUE(start_time,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation)
    LAST_VALUE(end_time,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation) as
    LAST_VALUE(activity_price_time,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_op
    LAST_VALUE(price,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation) as pr
    LAST_VALUE(quantity,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation) as
    LAST_VALUE(status,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation) as s
    LAST_VALUE(gmt_modified,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation
    LAST_VALUE(gmt_create,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_operation)
    LAST_VALUE(dbsync_operation,UNIX_TIMESTAMP(gmt_modified)*10+dbsync_opera
FROM distinct_dwd_mkt_membercart_ri
WHERE DATE_FORMAT(gmt_create , 'yyyy-MM-dd HH:mm:ss' , 'yyyyMMdd')=DATE_FORM
GROUP BY
cart_id,biz_id;

--存储小时维度的计算结果
CREATE TABLE result_mkt_membercart_ri_eh(
    id VARCHAR,
    data_time VARCHAR,
    all_preheating_cart_cnt BIGINT, -- 预热期间的 加购件数
    all_preheating_cart_alipay BIGINT,-- 预热期间的 加购金额
    eh_all_preheating_cart_uv BIGINT,-- 预热期间的 加购UV
    all_cart_cnt BIGINT, -- 投放期间的 加购件数
    all_cart_alipay BIGINT, -- 投放期间的 加购金额
    eh_all_cart_uv BIGINT, -- 投放期间的 加购UV
    primary key(id,data_time)
) WITH (
    type = 'custom',
     -- 其他信息省略
    timeDiv='hour'
) ;
```

```
--统计小时维度的 xx xx xx 业务指标
                                    INSERT INTO result_mkt_membercart_ri_eh
SELECT
    biz_id,
    DATE_FORMAT(gmt_create , 'yyyy-MM-dd HH:mm:ss' , 'yyyyMMddHH') data_time
    `sum`(case when gmt_modified<=COALESCE(activity_price_time,end_time) the
    `sum`(case when gmt_modified<=COALESCE(activity_price_time,end_time) the
    `sum`((case when gmt_modified<=COALESCE(activity_price_time,end_time) th
    `sum`(quantity) as all_cart_cnt,
    `sum`(quantity*CAST(price AS BIGINT)) as all_cart_alipay,
    `count`(distinct user_id) eh_all_cart_uv
FROM tmp_dwd_mkt_membercart_ri
WHERE
   status>0
GROUP BY  biz_id ,DATE_FORMAT(gmt_create , 'yyyy-MM-dd HH:mm:ss' , 'yyyyMMdd
```

上面case2天猫业务场景里面的加购金额统计来说，当每个投放场景的购物车的数据发生变化时候，就意味着上面【CREATE VIEW tmp_dwd_mkt_membercart_ri 】中的LAST_VALUE发生变化，最外层的sum统计【INSERT INTO result_mkt_membercart_ri_eh 】就要将前一条的LAST_VALUE【VALUE-1】撤回，用update的新LAST_VALUE【VALUE-2】进行求和统计，这样blink就需要有一种机制将VALUE-1进行撤回，利用【VALUE-2】进行计算，这种机制我们称为retract。

retract 实现原理参考

https://docs.google.com/document/d/18XlGPcfsGbnPSApRipJDLPg5IFNGTQjnz7emkVpZlkw/edit#heading=h.cjkoun4w44l4

转载于:https://my.oschina.net/u/2935389/blog/3023086