PowerData分享活动

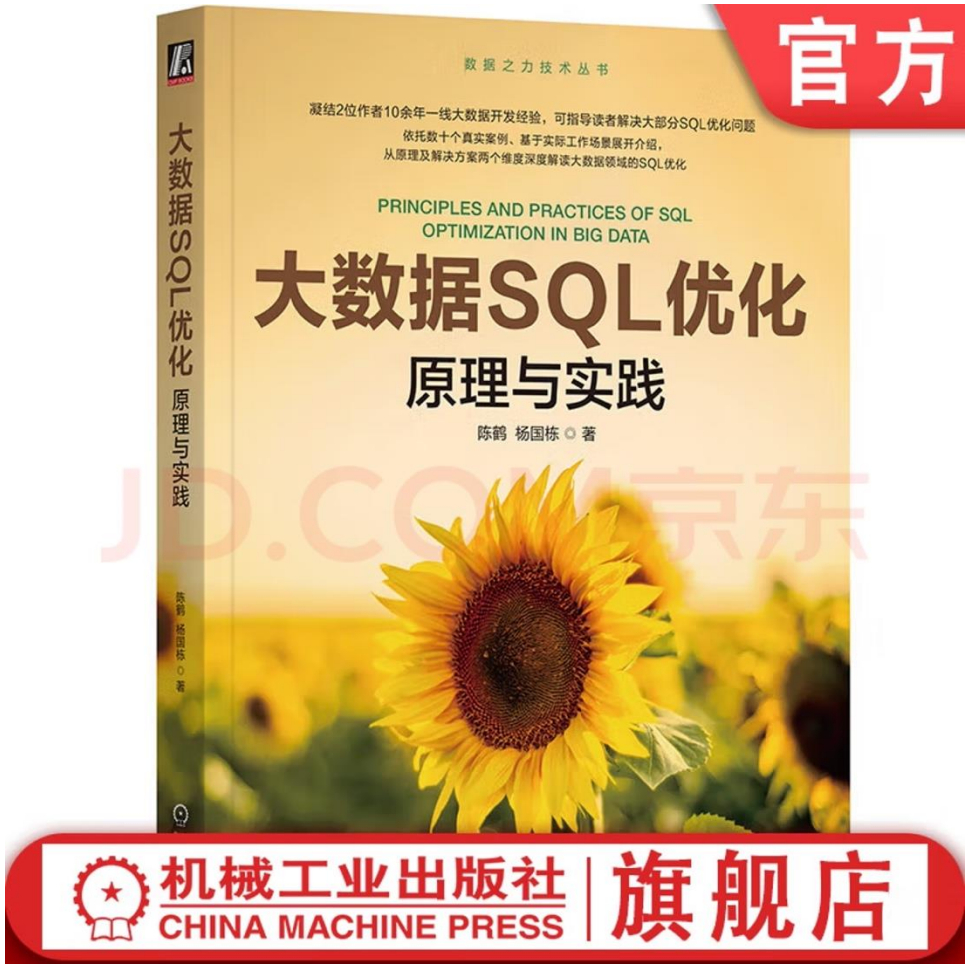
构建用户标签平台

陈鹤 Shopee大数据专家

PowerData | 凝聚国内数据力量

构建用户标签平台

构建用户标签平台


官方
数据之力技术丛书
凝结2位作者10余年一线大数据开发经验，可指导读者解决大部分SQL优化问题
依托数十个真实案例、基于实际工作场景展开介绍，
从原理及解决方案两个维度深度解读大数据领域的SQL优化
PRINCIPLES AND PRACTICES OF SQL OPTIMIZATION IN BIG DATA
大数据SQL优化
原理与实践
陈鹤 杨国栋 著
大数据SQL优化 原理与实践
机械工业出版社
CHINA MACHINE PRESS
旗舰店

构建用户标签平台

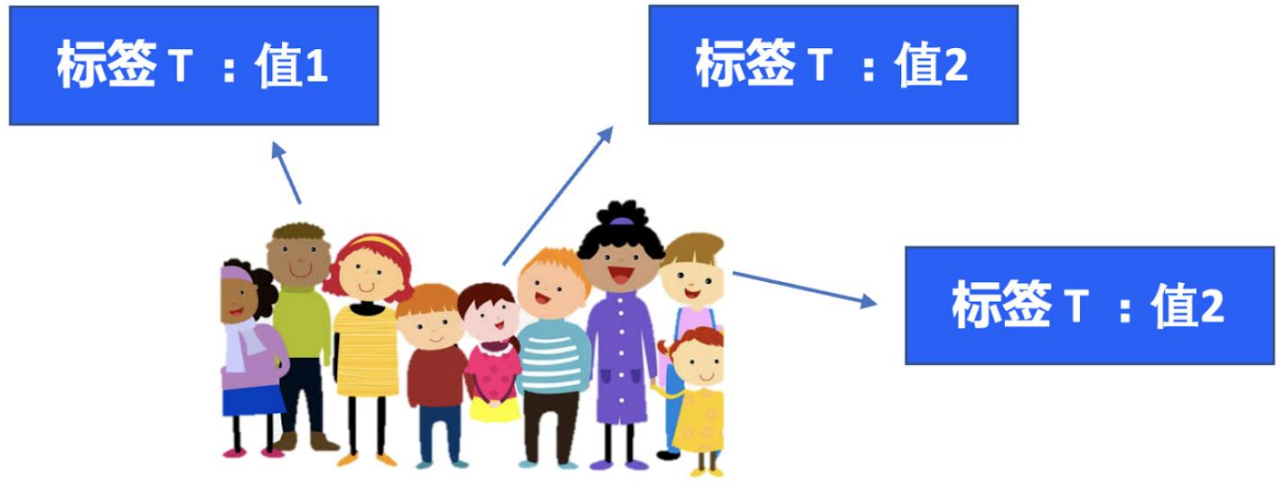
用户标签和用户群

用户标签

- 描述用户时的某个特征
- 也可以看作某个角度的用户分层

用户群

- 根据一定条件规则圈选出的部分用户



构建用户标签平台

数据分析

风控策略

广告投放

精准营销

构建用户标签平台

能够输出用户明细

支持在线交、并、补圈选用户群

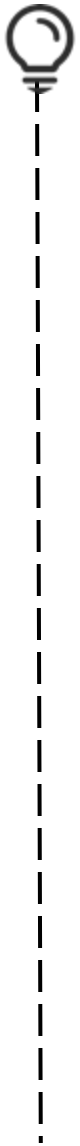
便于数据回溯

成本开销

构建用户标签平台

存储明细数据

- 行列存储
- 每一个标签扩展一列
- 查询极为容易



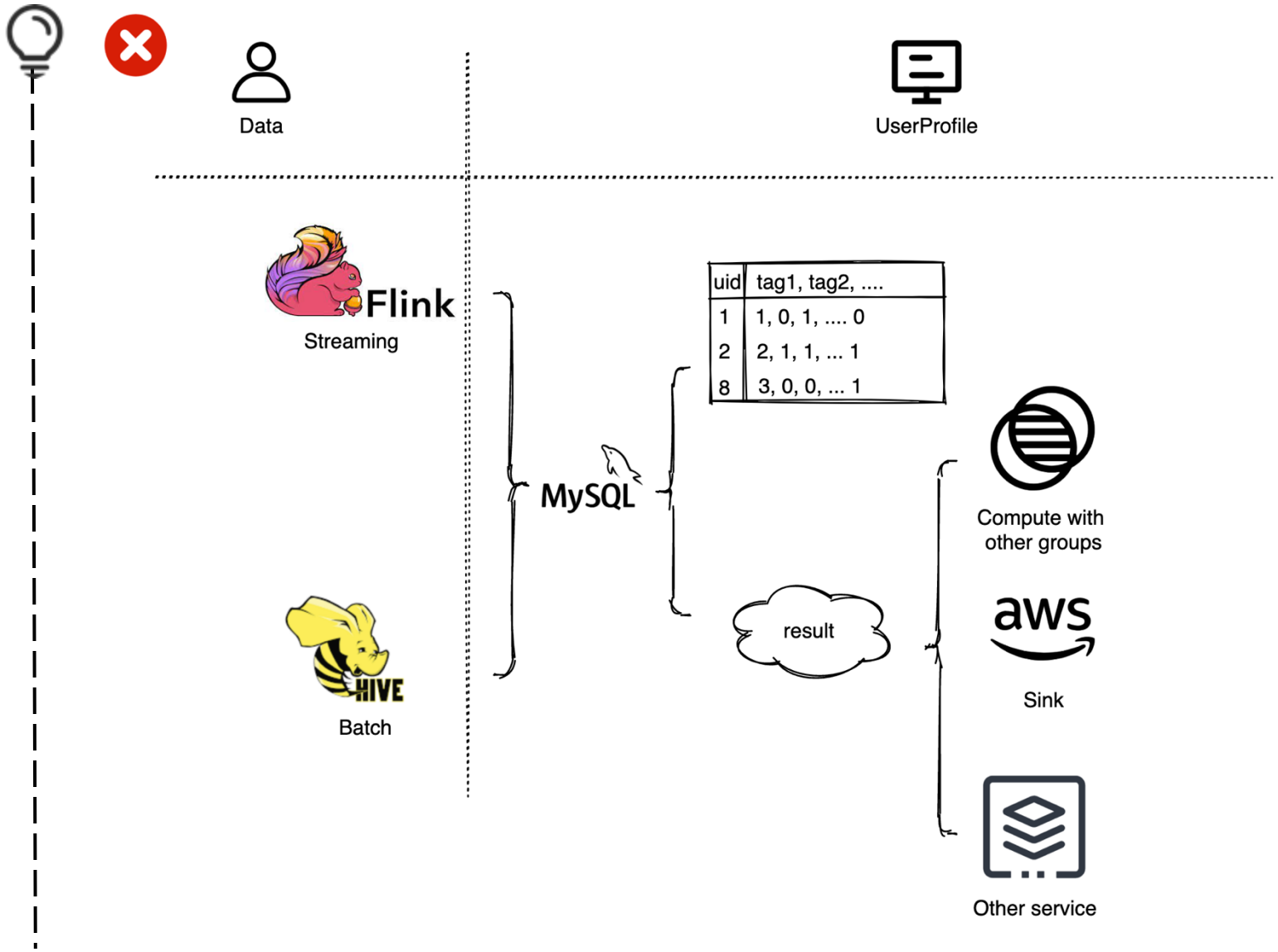
user id	gender	married	is_new_user
1	0	1	0
4	1	0	0
7	1	1	0
10	0	1	1

性别男，已婚，新用户 ⇨ gender=0 & married=1 & is_new_user=1 ⇨ user id 10

构建用户标签平台

存储明细数据

- 不利于维护
- 可扩展性一般
- 读写I/O瓶颈
- 没有解决成本开销

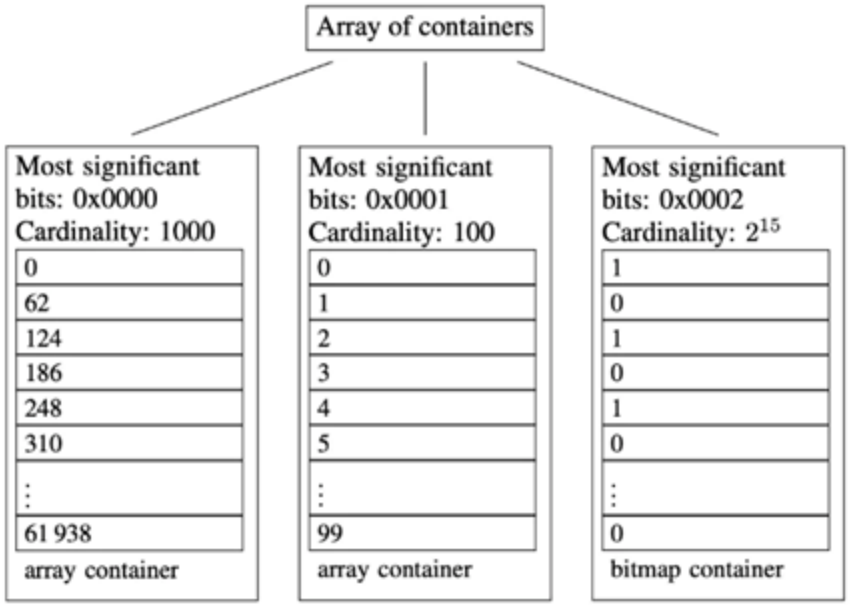


构建用户标签平台

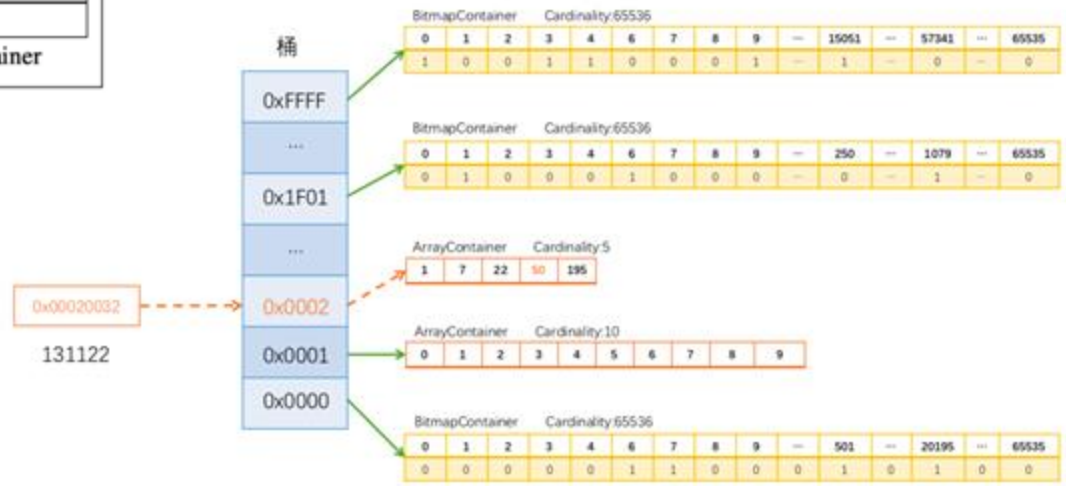
RoaringBitmap (RBM)

- 更高效的压缩（基于bitmap或基于数组）
- 存储开销更低，2700万uid大小约4.4MB，而Bit array则需要1.28GB

设计思想



写入示例



构建用户标签平台

基于DB的RBM

- 行列存储
- 每一个标签扩展一行
- 查询较为容易
- 内置bitmap function

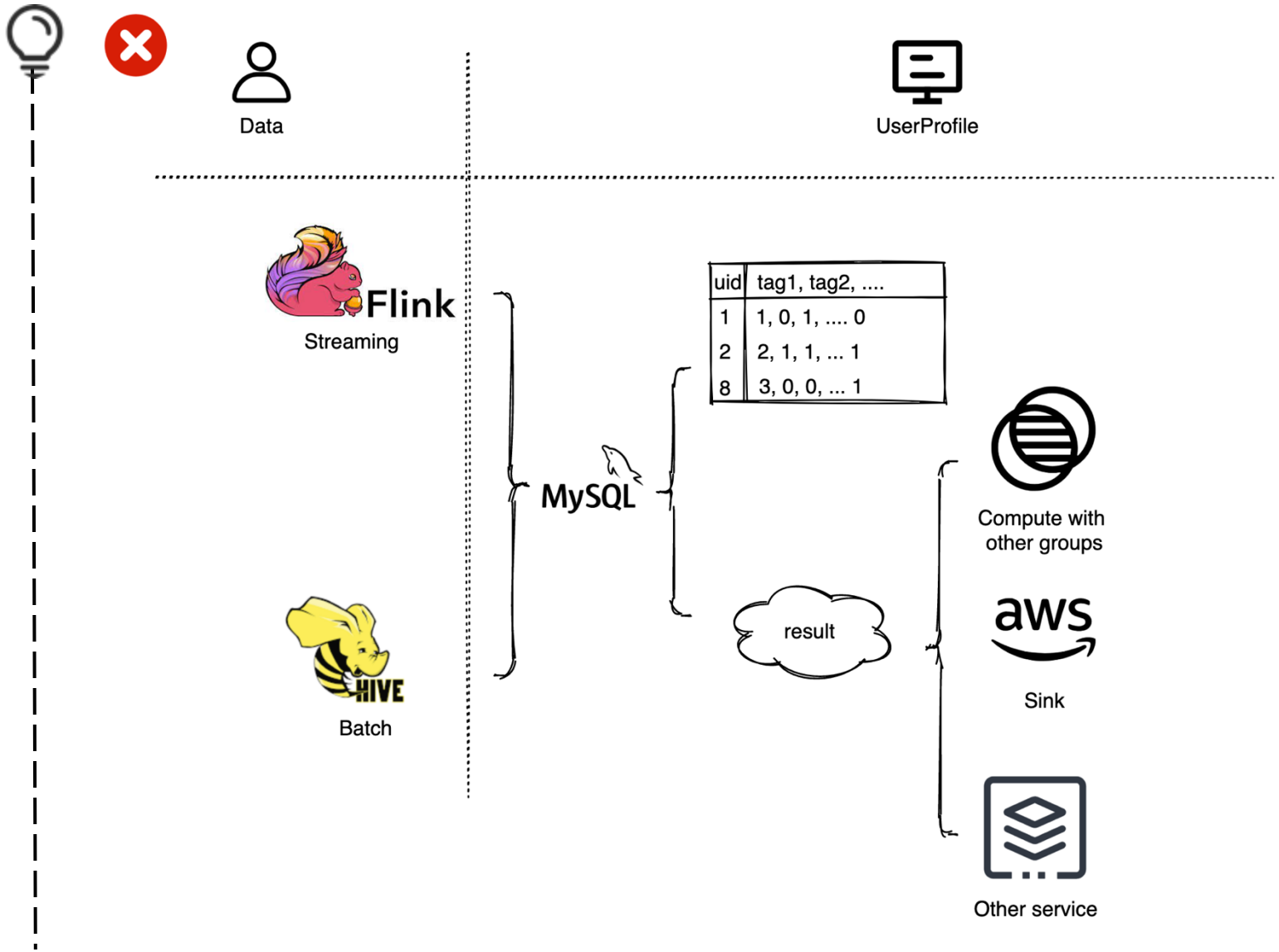
tag name	user id bitmap
男	[1, 10]
已婚	[1, 7, 10]
新用户	[10]
老用户	[1, 4, 7]

性别男，已婚，新用户  tag_name=男 & tag_name=已婚 & tag_name=新用户  user id 10

构建用户标签平台

基于DB的RBM

- 读写I/O瓶颈
- 还原用户明细存在瓶颈
- 倾斜情况需要考虑分段

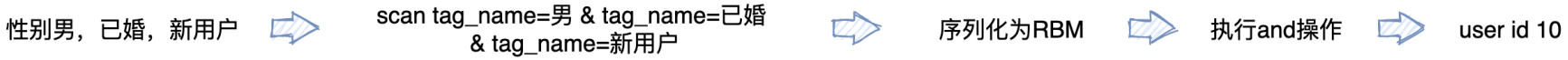


构建用户标签平台

基于HBase的RBM

- 基于HBase存储反序列化后的RBM
- 每一个标签扩展一个Rowkey
- 非明文，查询需要转换

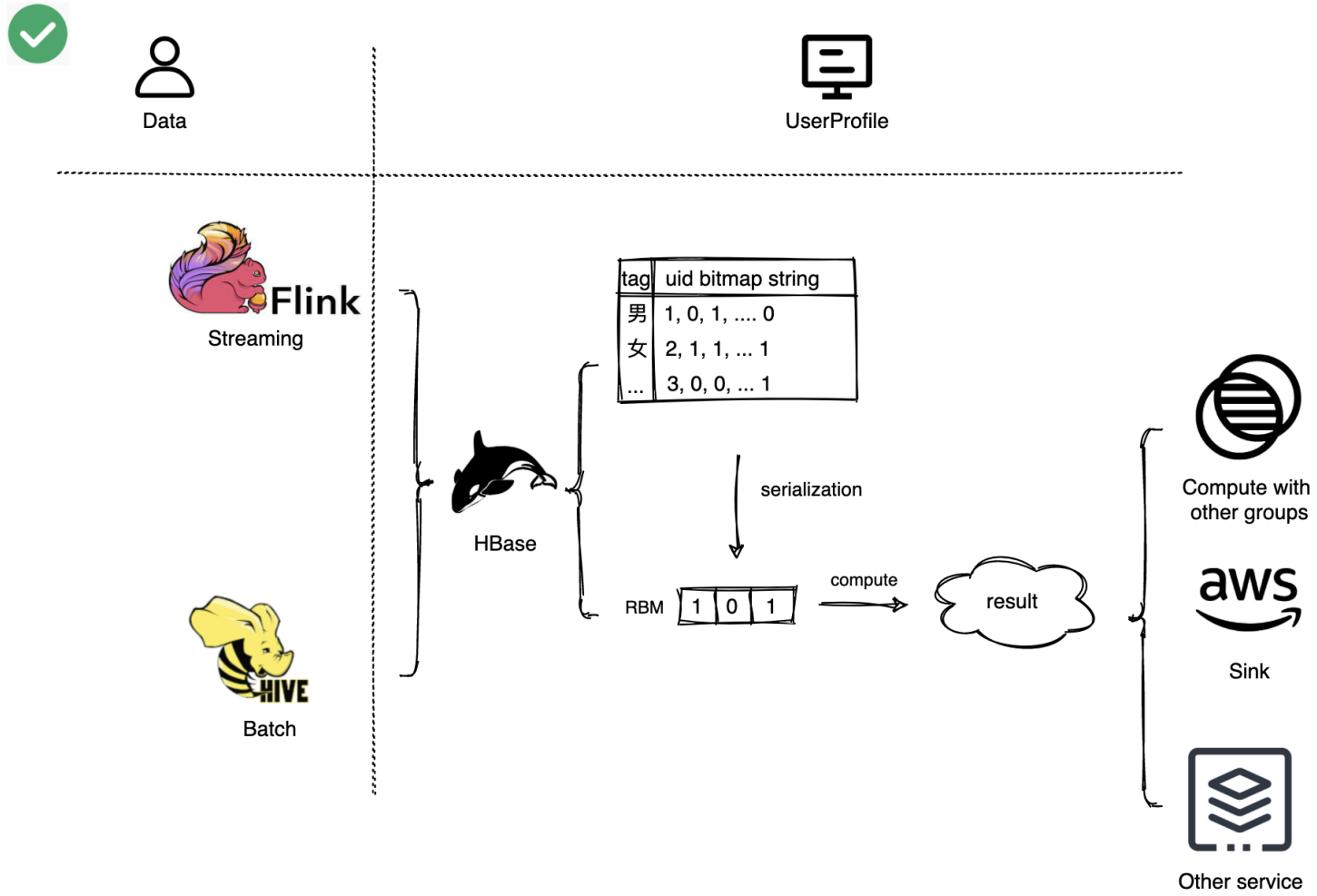
tag name rowkey	user id bitmap deserialization
男	[1, 10]
已婚	[1, 7, 10]
新用户	[10]
老用户	[1, 4, 7]



构建用户标签平台

基于HBase的RBM

- 基于HBase存储反序列化后的RBM
- 每一个标签扩展一个Rowkey
- 非明文，查询需要转换

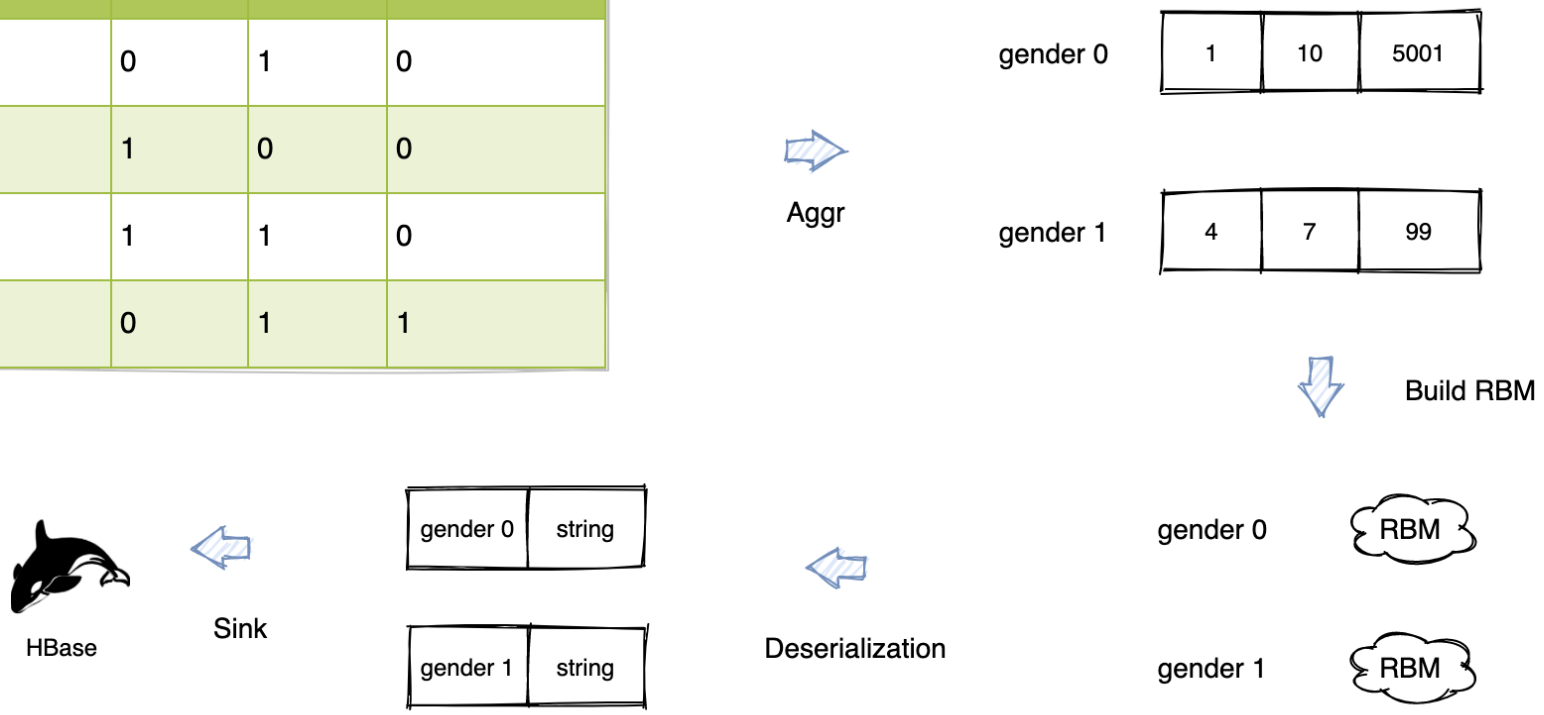


构建用户标签平台

入HBase表过程

- Hive宽表，提取指定标签字段进行聚合
- 聚合后结果构建RBM
- 将反序列化结果写入HBase

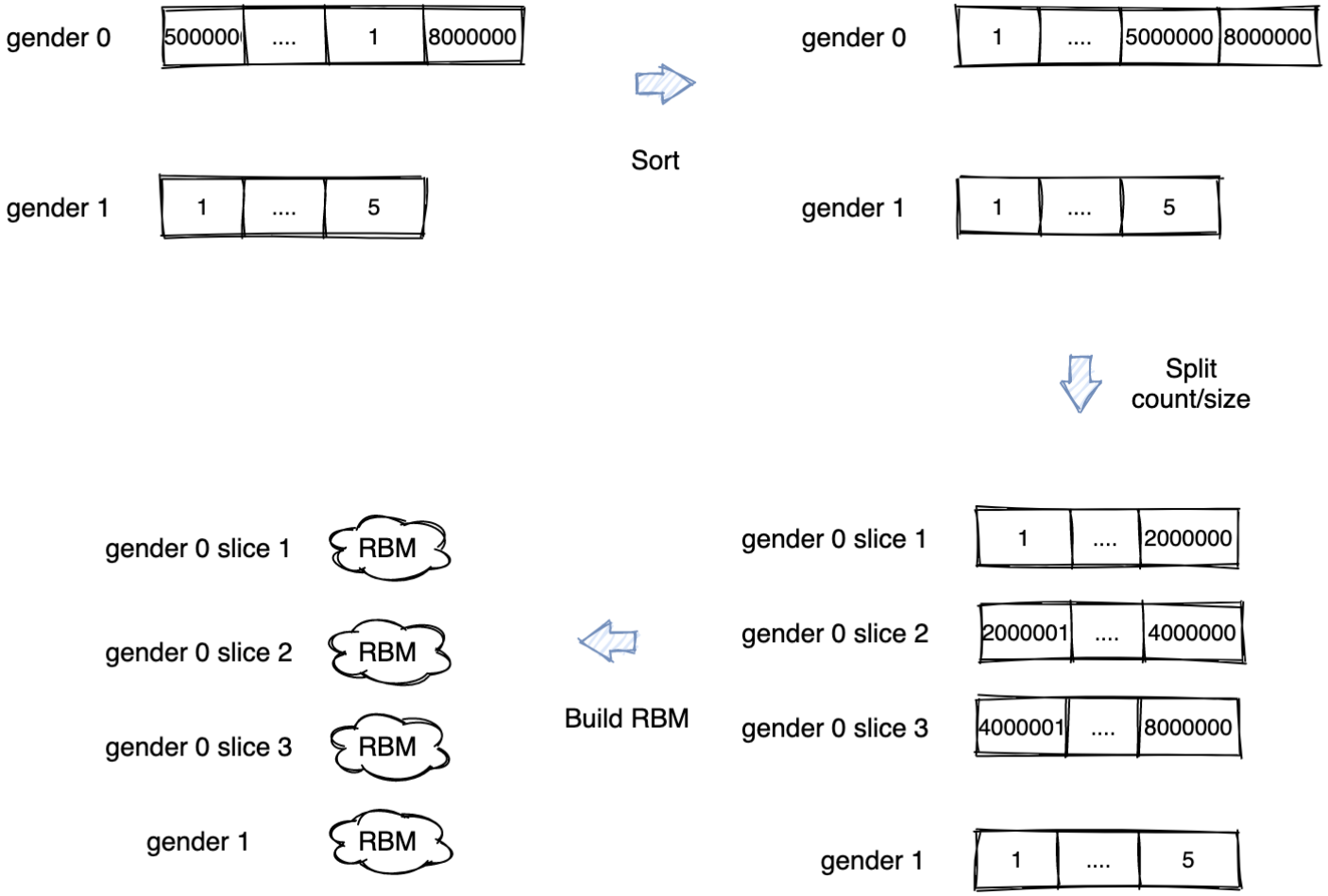
user id	gender	married	is_new_user
1	0	1	0
4	1	0	0
7	1	1	0
10	0	1	1



构建用户标签平台

HBase分段存储

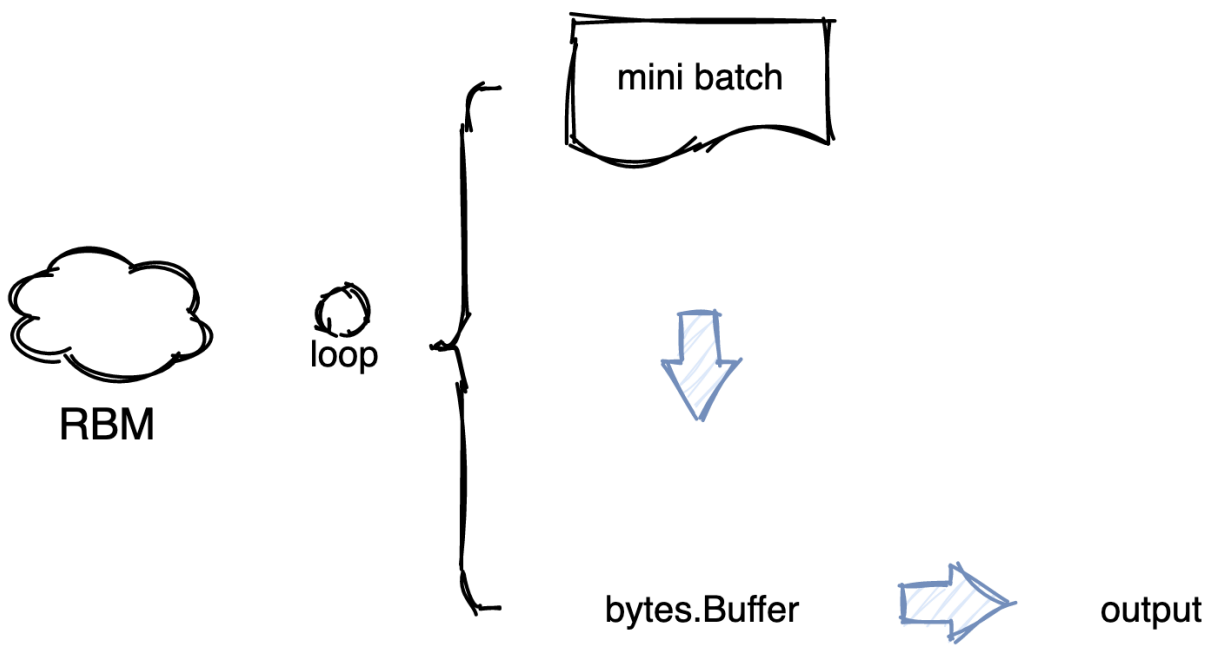
- Hive表入HBase表时转换
- 分段存储缓解倾斜
- 依托RBM，基于内存的交、并、补、计数



构建用户标签平台

还原用户明细

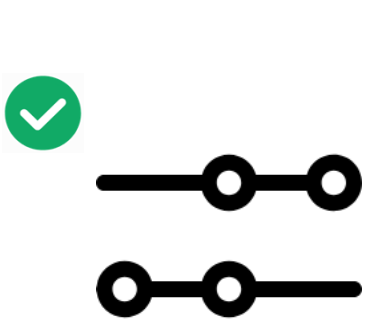
- 依托RBM还原用户明细
- 分段查询和输出
- 检测节点内存占用过高时sleep



构建用户标签平台

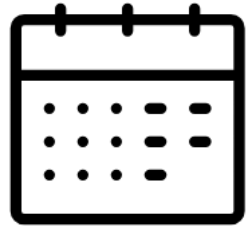
RBM局限性

- 只能作用于属性值
- 对于统计值只能分段处理
- 无法处理OLAP式诉求



区间范围

数值类型



日期类型

订单金额

访问次数

点击次数

构建用户标签平台

BSI

- 以二进制后切片的值为键
- 保留属性值

user id	order cnt
1	3
4	6
7	10
10	7

(个位 - 0)

(十位 - 0)

order cnt	user id 1	user id 4	user id 7	user id 10
0	0	0	1	0
1	0	0	0	0
2	0	0	0	0
3	1	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	1	0	0
7	0	0	0	1
8	0	0	0	0
9	0	0	0	0
0	1	1	0	1
1	0	0	1	0

order cnt=7 ⇨ 十位0 = 1 & 个位7 = 1 ⇨ [1, 4, 10] & [10] ⇨ user id 10

order cnt < 7 ⇨ 十位0 = 1 & 个位0~6 = 1 ⇨ [1] ∪ [4] ⇨ user id 1, 4

构建用户标签平台

BSI

- 以二进制后切片的值为键
- 保留属性值

user id	order cnt
1	3
4	6
7	10
10	7

user id	order cnt（十进制）	order cnt（二进制）
1	3	0011
4	6	0110
7	10	1010
10	7	0111

order cnt slice 第几位bit = 1	user id bitmap
0	[1, 10]
1	[1, 4, 7, 10]
2	[4, 10]
3	[7]

order cnt = 7 → 10进制7 = 2进制0111 → slice 2 = 1 &
slice 1 = 1 &
slice 0 = 1 → [4, 10] & [1, 4, 7, 10]
& [1, 10] → user id 10

order cnt < 7 → 10进制7 = 2进制0111 → 最高位 = 0
& 二进制值 != 0111 → [1, 4, 10] - [10] → user id 1, 4

构建用户标签平台

BSI

- 以二进制后切片的值为键
- 保留属性值

```
private Roaring64Bitmap oNeilCompare(
    BitmapSliceIndex.Operation operation, long predicate, Roaring64Bitmap foundSet) {
  Roaring64Bitmap fixedFoundSet = foundSet == null ? this.ebM : foundSet;

  Roaring64Bitmap GT = new Roaring64Bitmap();
  Roaring64Bitmap LT = new Roaring64Bitmap();
  Roaring64Bitmap EQ = this.ebM;

  for (int i = this.bitCount() - 1; i >= 0; i--) {
    int bit = (int) ((predicate >> i) & 1);
    if (bit == 1) {
      LT = Roaring64Bitmap.or(LT, Roaring64Bitmap.andNot(EQ, this.bA[i]));
      EQ = Roaring64Bitmap.and(EQ, this.bA[i]);
    } else {
      GT = Roaring64Bitmap.or(GT, Roaring64Bitmap.and(EQ, this.bA[i]));
      EQ = Roaring64Bitmap.andNot(EQ, this.bA[i]);
    }
  }
```

	初始化	0	1	1	1
全量集合	[1, 4, 7, 10]				
GT >	[]	[] or ([1, 4, 7, 10] & [7]) = [7]	[7]	[7]	[7]
LT <	[]	[]	[] or ([1, 4, 10] - [4, 10]) = [1]	[1] or ([4, 10] - [1, 4, 7, 10]) = [1]	[1] or ([4, 10] - [1, 10]) = [1, 4]
EQ =	[1, 4, 7, 10]	[1, 4, 7, 10] - [7] = [1, 4, 10]	[1, 4, 10] & [4, 10] = [4, 10]	[4, 10] & [1, 4, 7, 10] = [4, 10]	[4, 10] & [1, 10] = [10]

order cnt < 7 ⇨ 十位0 = 1 & 个位0～6 = 1 ⇨ [1] ∪ [4] ⇨ user id 1, 4

```
  }
}
```

构建用户标签平台

BSI局限性

- 处理逻辑较为复杂
- 受极值（MAX、MIN）限制极为严重
- 对于稀疏数据压缩性能一般
- 构建BSI过程I/O瓶颈

	Range	Int	BSI	BSI Slice
大小（MB）	538.13	615.75	913.04	913.95
Rowkey个数	5 * 14	7130 * 1	1 * 115	80 * 2
单查(全范围查询[0, +)	5s	10s	24s	20s
并发5(全范围查询)	6s	20s	97s	54s
并发10(全范围查询)	9s	39s	183s	109s
单查([100,3000])	-	9s	19s	14s
并发5([100,3000])	-	10s	62s	32s
并发10([100,3000])	-	12s	118s	60s

构建用户标签平台

Hive表注意事项

- 不受宽表、DA表限制
- 多维时需要组合
- 非数值类型建议维护维表

user id	shop	is_new_user
1	A	否
4	A	否
1	B	是

shop	user id bitmap
A	[1, 4]
B	[1]

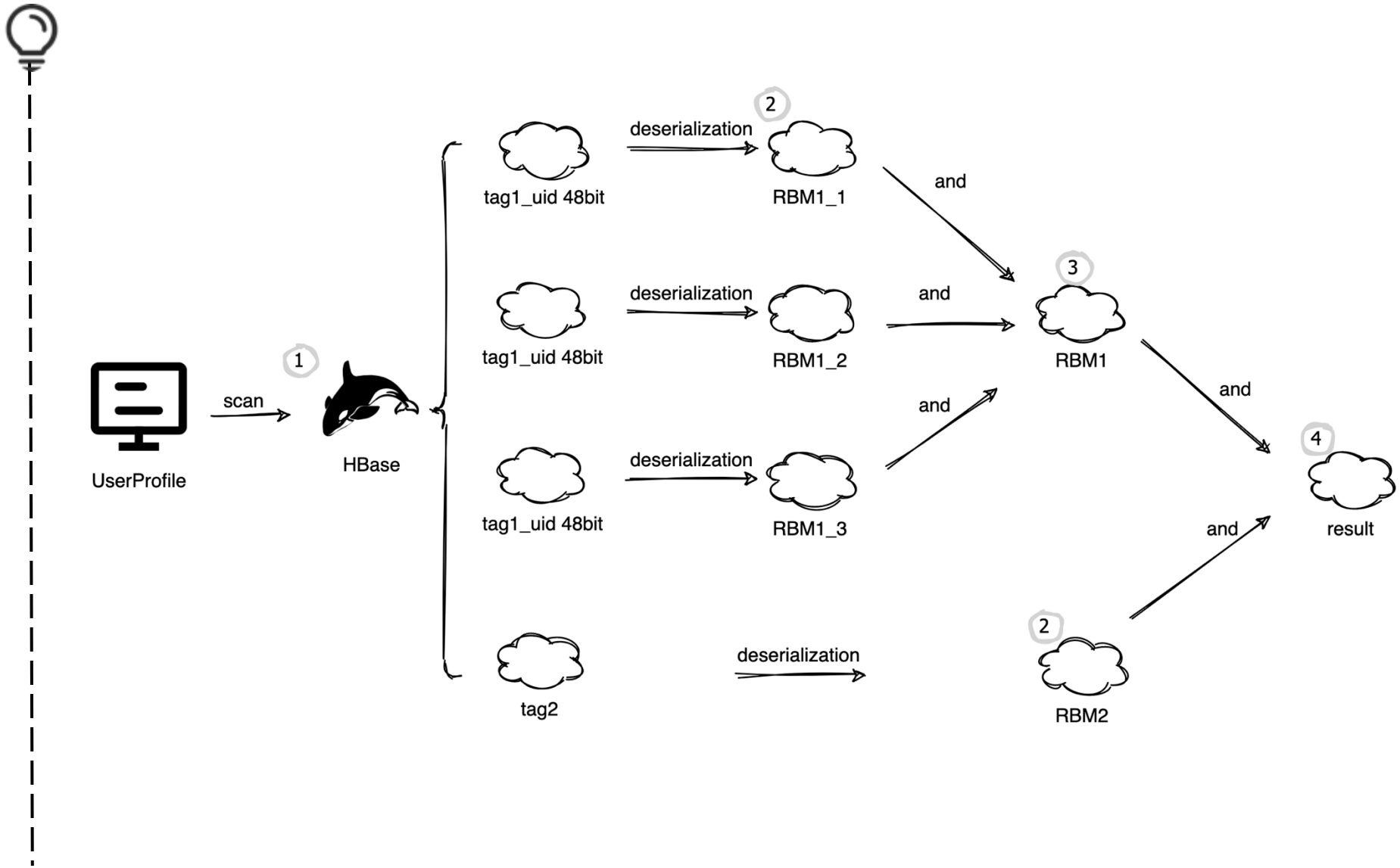
is_new_user	user id bitmap
是	[1]
否	[1, 4]

A店铺，新用户 ⇨ shop=A & is_new_user=是 ⇨ [1,4] & [1] ⇨ [1]

构建用户标签平台

实时标签

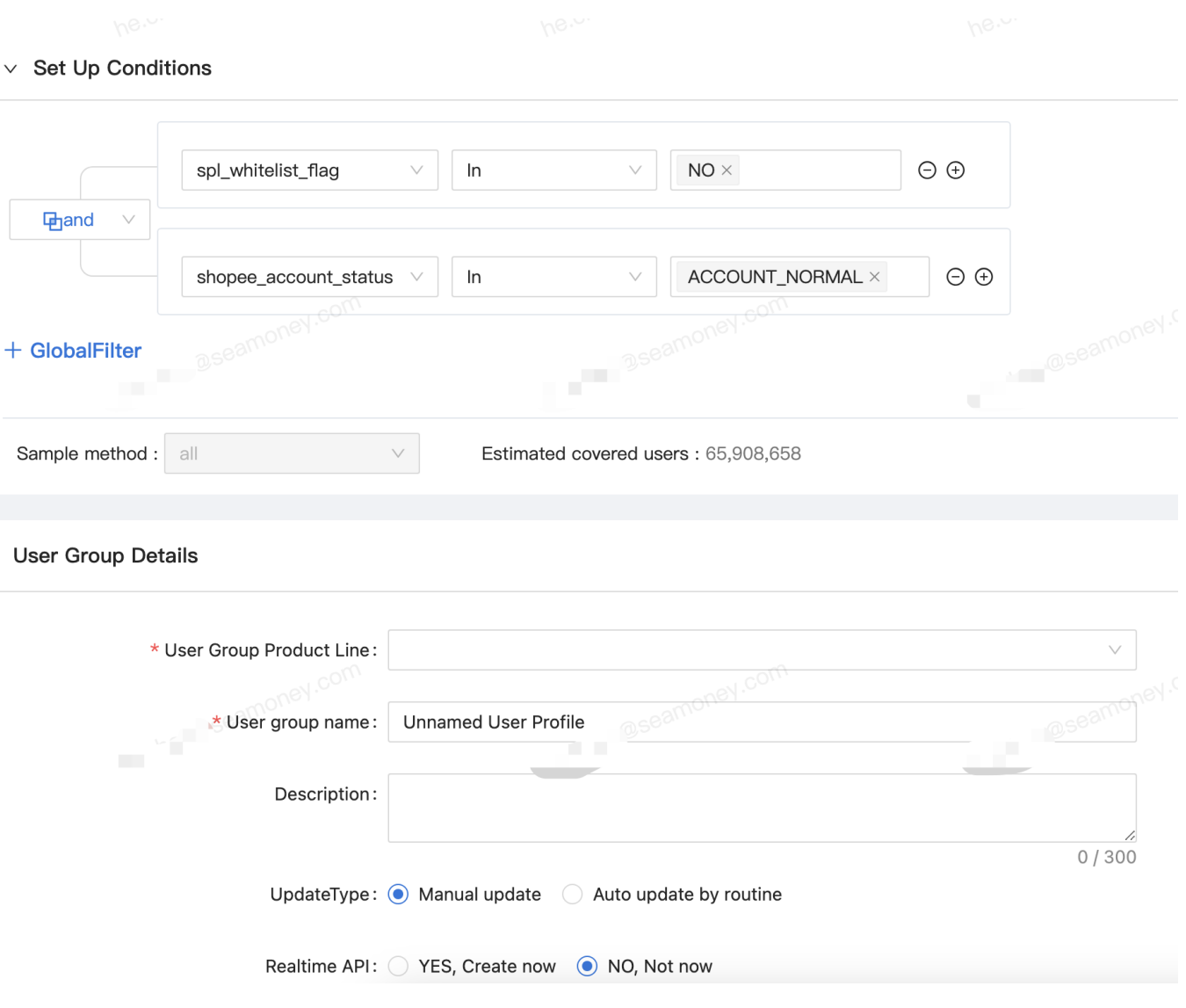
- 仿照RMB的分桶思想
- 构建多段BSI



构建用户标签平台

其他功能

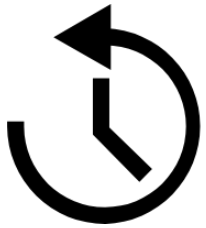
- 标签血缘
- 链路告警
- 延迟告警
- 数据回溯
- 效果回收
- 标签/用户群生命周期
- 数据包下载
- …



构建用户标签平台

2K

支持2000+标签



秒级别在线圈选

覆盖金融业务营销场景

《大数据SQL优化：原理与实践》

PowerData社区公众号
扫码即可加入社区

Thank You!