CDP 实时人群圈选方案

- CDP 实时人群圈选方案
  - 修订内容
    - 0.0.3
    - 0.0.2
    - 0.0.1
  - 1. 背景
  - 2. 方案设计
    - 处理流程
      - 数据接入
      - 规则计算
      - 数据分发
  - 3. 实时人群管理平台
    - 功能模块
      - 数据视图
      - 标签计算
      - 人群圈选
  - 4. 场景应用
    - 早餐二次卷
      - 创建流程示例
      - 处理流程拆解

修订日期	修订人
2021-09-29	Konka
2021-09-24	Konka
2021-10-26	Konka

修订内容

0.0.3

```
#### 2021-10-29
1. 去除 HAC 离线人群导入的流程
```

0.0.2

```
#### 2021-10-26
1. 流程设计
2. 增加场景案例
3. 增加管理平台--标签和人群功能模块页面示例与处理流程
```

0.0.1

```
#### 2021-09-24
1. 完成架构设计
2. 设计 MGPT 基础功能模块
```

1. 背景

目前系统发券会从 `MPS` 取数，来判断一个会员是否符合发券规则。

而 `MPS` 的数据是从上游 `Databank` 获取的，上游的数据延迟本身是 `T+3`，到 `MPS` 后再计算会达到 `T+4`，最后在实时场景上产生了 `4` 天的时差。

在判断发券规则时需要将这部分数据差异弥补进去，目前有这部分差异数据和计算能力的系统是 `Dataswift`，以下所述方案会围绕这个场景结合 `CDP` 和 `DataSwift` 两个系统来做一个整合。

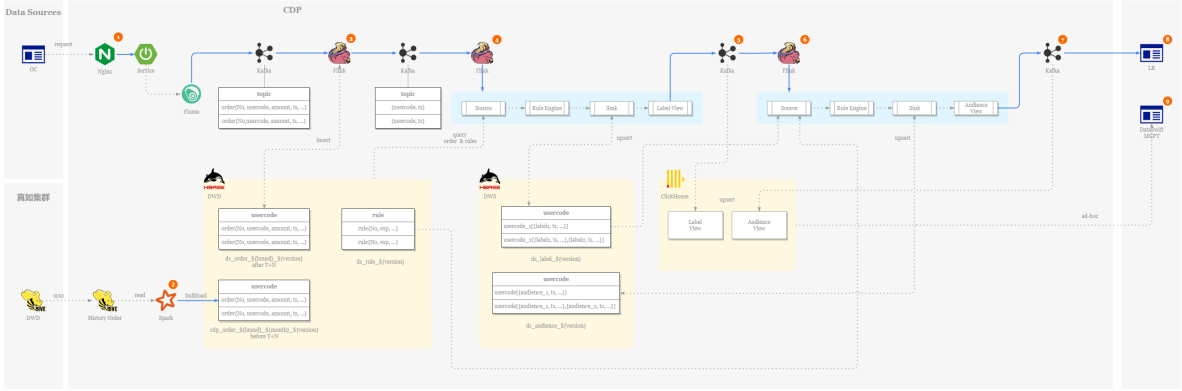
=

场景案例：

- 最近1年未购买的人送一元吃鸡的券，在人进来的时刻判断这个人是否满足。
- 早餐二次卷，在客户购买早餐后判断其是否二次购买，如是则推送二次卷。

早餐二次券的场景需要从 `CDP` 中引入历史订单，作为初始状态，在引入后发生的订单和行为都从 `Dataswift` 原有的数仓内获取，从而达到离线用户行为与实时用户行为的合并，提供更长时间范围上的任务计算支撑。

2. 方案设计



处理流程

数据接入

- 【1】订单和任务接入
  - `Nginx` 代理请求到 `Service`
  - `Service` 记录请求日志
  - `Flume` 推送日志进 `Kafka`
- 【2】历史订单数据接入
  - 从真如集群同步历史订单数据到 `Dataswift` 环境的 `Hive` 数仓中
  - 之后按天增量同步订单数据，分区为 `${brand} / ${yyyyMMdd}`，如数据有误，则重建这个分区

- `HBase` 表命名规则： 按品牌与月或季度来建立分区，版本号格式为 `yyyyMMdd_HHmm`，如需重刷数据，则更换版本号

  ```
  #talbename:
  cdp_order_${brand_[month,quarterly]}_${version}
  ```

  执行版本号更换流程后需要切换 hbase 表，并清理掉旧版本

- 以 `bulkload` 方式导入数据到 HBase 表中

规则计算

- 【3】实时处理订单数据
  - 通过 `Flink` 将 `Kafka` 中的 `OC` 数据写入 `HBase`
  - `HBase` 表命名规则： 按品牌来建立分区，版本号格式为 `yyyyMMdd_HHmm`，如需重刷数据，则更换版本号

    ```
    #talbename:
    ds_order_${brand}_${version}
    ```

  执行版本号更换流程后需要切换 `HBase` 表，并清理掉旧版本
- 【4】实时计算标签
  - 从 `DWD` 中取用户对应的订单数据与规则
  - 进行规则计算
  - 将 `label` 计算结果与 `DWS` 中的历史结果做合并，发生变更的用户推送到 `Kafka` 中
- 【6】实时计算人群
  - 从 `DWD` 中取用户对应的规则
  - 从 `DWS` 中取用户的 `label`
  - 进行规则计算
  - 将 `audience` 计算结果与 `DWS` 中的历史结果做合并，发生变更的用户推送到 `Kafka` 中

数据分发

- 【5】标签数据分发
  - 将 `Kafka` 中 `label` 变更的用户记录存入 `ClickHouse`，方便管理平台上查询历史日志
- 【7】人群数据分发
  - 将 `Kafka` 中 `audience` 变更的用户记录存入 `ClickHouse`，方便管理平台上查询历史日志
- 【8】下游系统对接
  - 下游 `LR` 系统直接对接 `Kafka`，实时获取人群计算结果
- 【9】统计与分析
  - 通过 `ClickHouse` 中存储的 `label` 和 `audience` 数据，在管理平台上快速做统计分析

3. 实时人群管理平台

`DataSwift` 目前没有管理平台，所有规则配置都是基于线下配置；这次通过整合 `CDP` 的过程来建立一个统一的实时人群管理平台，在此平台上来创建并管理标签与人群规则； 规则引擎以及交互方式与 `HAC` 对齐。

功能上主要面对两类场景：

- 实时和离线皆为明细数据
- 实时为明细数据，离线为导入的人群包或聚合结果

针对这两类场景做了如下的功能模块区分，从设计和流程上划分开来。

功能模块

数据视图

- 从 `HBase` 中选择一个实时表作为数据源。
- 从 `DWD` 层或 `DWS` 层选择表作为数据源。
- 从上述数据源中选择相同的字段做 `union`，从逻辑上把实时和离线的数据做联合形成视图，输出为一个新的数据源。

标签计算

- 选择数据源：
  - 从 `HBase` 中选择一个实时表
  - 从 `DWD` 层或 `DWS` 层选择表
  - 从联合视图里选择一个表
- 创建标签计算规则
- 输出计算结果

人群圈选

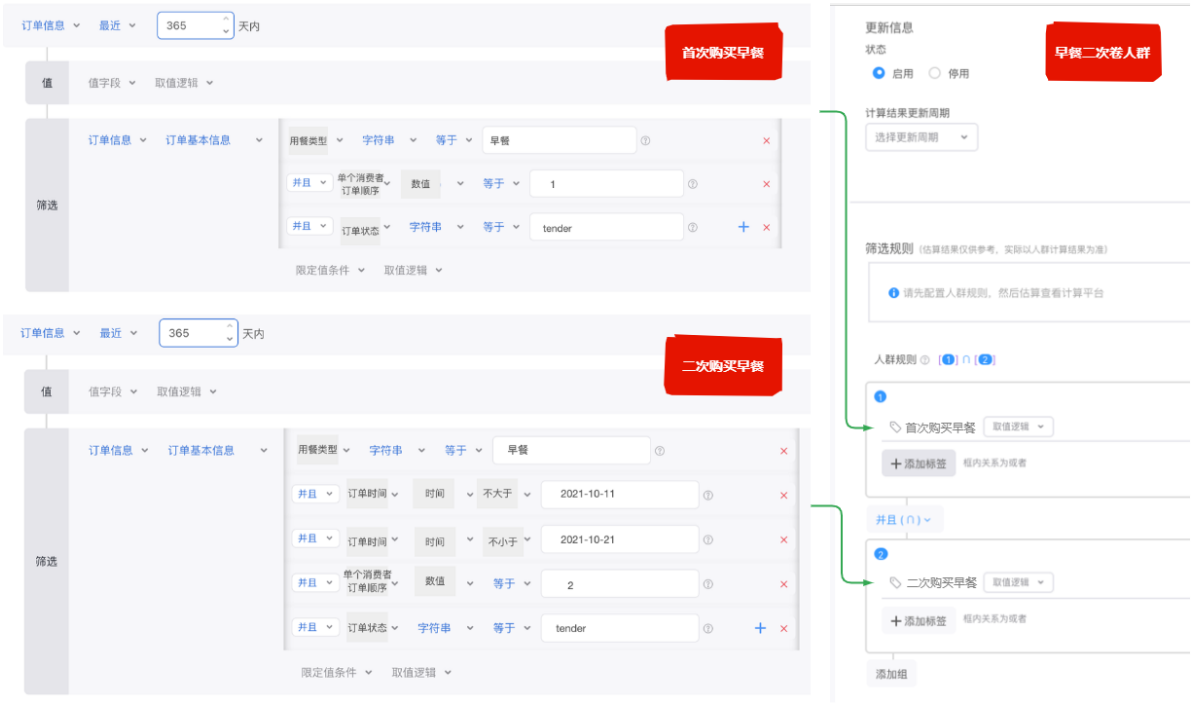
- 选择标签作为人群规则输入，设置计算规则
- 输出匹配结果

4. 场景应用

早餐二次卷

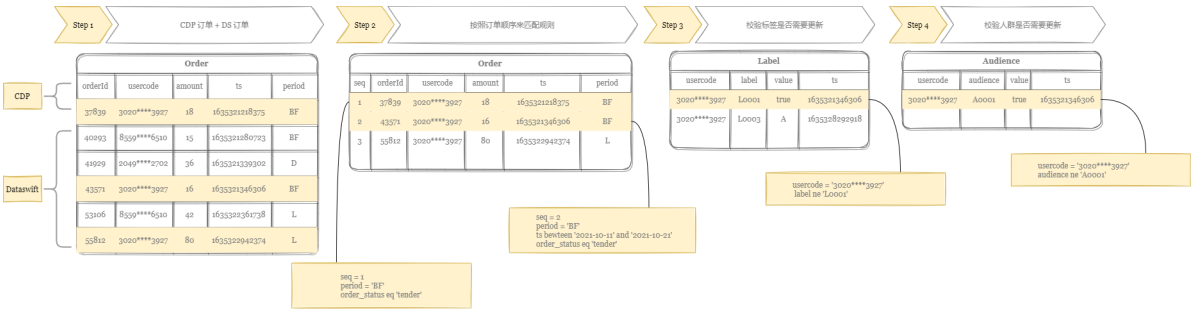
- 在客户购买早餐后判断其是否满足早餐二次卷活动规则：
  - 订单发生在过去 `365` 天内，第 `1` 单的购买类型为早餐，并完成定单
  - 订单发生在活动时间范围内，第 `2` 单的购买类型为早餐，并完成定单

创建流程示例



- 标签配置
  - 创建首次购买早餐标签
  - 创建二次购买早餐标签
- 人群配置
  - 选择首次购买早餐标签和二次购买早餐标签
  - 设置标签计算规则

处理流程拆解



- 通过合并 `CDP` 与 `DS` 的订单数据，获取到该用户的全量订单记录
- 订单可以对时间升序排列，序号字段作为系统默认字段来参与计算
- 计算得到的 `label` 结果需要与已有的记录做对比
  - 若发生变化，则推送出去做人群计算
  - 若未有变化，则认为之前已满足过匹配条件，不需要推送
- 计算得到的 `audience` 结果需要与已有的记录做对比
  - 若发生变化，则推送给下游
  - 若未有变化，则认为之前已满足过匹配条件，不需要推送