# 无埋点：Mixpanel

# 第四章 数据仓库数据采集与同步

## 4.1系统埋点设计

### 4.1.1 数据分类

在工厂环境中，我们将数据仓库获取的数据划分为业务数据和用户行为数据。

业务数据：业务流程中产生的交易、状态流转、用户等相关的数据，通常存储在DB 中， 包括 rdbms、nosql 等，这部分数据是业务相关的，具体哪些数据需要保留一般由业务侧设计，不需要过度关注，按实际需要采集即可。

用户行为数据：用户在使用产品过程中，与 C 端产品交互过程中产生的数据，比如页面浏览、点击、停留等，这部分数据由于不影响业务流程本身，所以通常不受关注，但对运营、产品优化至关重要，所以通常也是数据建设的一部分，需要单独设计数据埋点、采集策略。

### 4.1.2 用户行为数据埋点设计

用户行为数据埋点的设计有三个关键点：

1）用户标识体系建立；

2）多屏用户标识打通（多屏：手机、PC、Pad）；

3）埋点方案设计；

1）用户标识体系建立：

在电商场景中经常会遇到用户在不登录/注册的情况下访问网站的情况，对于电商平台来说，希望能够收集到用户所有的访问行为数据，包括用户在未登录/注册之前的行为数据， 这就涉及到了用户标识的问题，如何在用户不登录/注册的情况下对用户进行标识，并且将不登录/注册情况下的用户行为数据追加到用户完整的用户行为数据中，这是用户标识体系需要完成的工作。

同样的，对于运营、产品和研发部门，需要根据用户在注册前后的行为明确是什么吸引了用户最终完成注册，因此需要将用户注册之前的行为与注册之后的行为关联起来，也就是将注册前后的信息打通。

2）用户标识建立方案：

考虑到移动端和 PC 端的区别，我们在建立用户标识时划分两个场景，分别是 PC 端和APP 端：

PC 端

PC 端的电商平台在浏览器中进行展示，也就是以 H5 的形式进行展示，采用 cid （客户端id）和 uid（用户id）相结合的方式对用户进行标识。

cid：类似于 cookieId，但不是 cookieId，cid 随着不同的用户切换在改变，我们的目的是用 cid 来标识用户，同一个用户在不同的屏幕上是不同的 cid。

uid：是用户注册后分配给用户的，可以唯一标识一个用户的序号。

APP端

APP 端同样采用 cid 和 uid 相结合的方式对用户进行标识。

cid：使用设备 imei/IDfa 根据某种规则生成唯一设备标识，作为 cid。

uid：是用户注册后分配给用户的，可以唯一标识一个用户的序号。

以上的两种方案都引入了 cid 和 uid 的概念，只不过不同的平台 cid 的定义不同。

在用户注册/登录前，使用 cid 作为设备标识，跟踪用户行为数据；用户注册/登录后， 使用 uid 标识用户，跟踪用户行为数据。

cid 和 uid 相结合的多种方式：

① 一个新设备，从来没有访问过电商平台，第一次无登录访问时，电商平台给他分配一个 cid，然后，用户登录了，由于不是切换用户，那么 cid 不变，登录后的数据有一个 uid， 根据登录后的 cid 和登录前的 cid 进行数据的匹配。

② 一个新设备，从来没有访问过电商平台，第一次无登录访问时，电商平台给他分配一个 cid，然后，用户注册了，由于是新注册，cid 不变，用户登录后电商平台分配给他一个新的 uid，根据登录后的 cid 和登录前的 cid 进行数据的匹配。

③ 一个老设备，一个人登录后，cid=xyz，uid=123，退出登录后，cid 不变，uid 不变， 之后所有的未登录访问行为的 cid 和 uid 都是上一次登录的人的 cid 和 uid，因此全部会被划分到上一个登录的人的行为里。

④ 一个老设备，一个人登录后，cid=xyz，uid=123，退出登录后，cid 不变，uid 不变， 之后有人再次登录，如果 cookie 中的 uid 和之前的不一致，那么判定为用户登录切换，是不同用户，cid 改变，uid 改变，如果 cookie 中的 uid 和之前的一致，那么 cid 和 uid 都不变化。

根据③可知，如果一个人登录后再推出登录，在下次登录之前，即使是多个不同的人访问了电商平台，也会被记为上次登录的人的访问行为。虽然 cid 并不是非常精确，但是他能够让数据找到归属。

多屏用户打通：

一个用户可能同时拥有手机、平板、电脑等设备，我们将多种不同的电子设备称为多屏。我们希望能够将同一个用户在多屏上的数据进行打通。

同一用户在不同屏幕上的 cid 是不同的，但是 uid 是相同的，但是如果只使用 uid，那么注册/登录之前的数据就会丢失，因此必须将 cid 和 uid 结合使用。

综上所述，通过 cid 和 uid 相互结合，实现未注册/登录数据与注册/登录数据的整合，然后通过 uid 实现多屏数据的打通。

埋点设计：

1. 埋点对象分类：

在生产环境下，我们主要的埋点对象有两类：Native APP、Web H5。

PC 端的用户主要访问的是 H5 页面，移动端用户主要是使用 APP 进行访问，当前所接触到的 APP 有多种类型，如 Native APP、Web APP、Hybird APP，在详细讨论埋点设计之前，先了解一下几种不同的 APP：

（1）Native App：

Native App 是一种基于智能手机本地操作系统如 iOS、Android、WP 并使用原生程式编写运行的第三方应用程序，也叫本地 app。一般使用的开发语言为 JAVA、C++、Objective-C。Native App 因为位于平台层上方，向下访问和兼容的能力会比较好一些，可以支持在线或离线，消息推送或本地资源访问，摄像拨号功能的调取。但是由于设备碎片化，App 的开发成本要高很多，维持多个版本的更新升级比较麻烦，用户的安装门槛也比较高。

（2）Web APP：

WebApp 就是运行于网络和标准浏览器上，基于网页技术开发实现特定功能的应用。WebApp 是指基于 Web 的系统和应用，其作用是向广大的最终用户发布一组复杂的内容和功能。当用户登录一个网站（如百度、淘宝），大家很容易理解这是在访问一个 Web App。但是对那些仅仅提供基础服务（如电话查询或是信息查询）的网站，区分用户是否在访问 WebApp 就变得相当困难了。其实这些服务大多都是 Web App。我常常这样问自己“这个程序是否完成了某个任务？”。即便它只完成了某个非常小的任务，那么它也是一个 Web App。Google 的搜索引擎就是一个 Web App，它本质上和电话查询服务没有什么区别。

（3）Hybrid APP：

Hybrid App（混合模式移动应用）是指介于 web-app、native-app 这两者之间的 app，兼具“Native App 良好用户交互体验的优势”和“Web App 跨平台开发的优势”。

Hybrid App 是指介于 web-app、native-app 这两者之间的 app，它虽然看上去是一个 NativeApp，但只有一个 UI WebView，里面访问的是一个 Web App，比如街旁网最开始的应用就是包了个客户端的壳，其实里面是 HTML5 的网页，后来才推出真正的原生应用。再彻底一点的，如掌上百度和淘宝客户端 Android 版，走的也是 Hybrid App 的路线，不过掌上百度里面封装的不是 WebView，而是自己的浏览内核，所以体验上更像客户端，更高效。

Native APP 无法获取 URL 相关信息，Web APP 无法获取设备相关的信息，因此，市面上现有的 APP 大部分是 Hybird APP，它兼具 Native APP 和 Web APP 的特点，能够同时获得设备信息和 URL 信息。

通常设计埋点时针对 Native App、Web H5 采用不同的规范，但不利于数据整合和使用，推荐采用事件模型进行埋点设计，即把用户的一切行为动作都看作事件，包括页面浏览、按钮悬停、点击等，这样 Native 和 H5 统一数据标准，实际数据仅参数上的区别。

2）埋点分类：

根据埋点位置，可分为客户端埋点、服务端埋点。

两种埋点方式各有利弊，比如服务端埋点对无后台请求的用户行为无法捕获（比如用户在页面上的悬停，这种事件是有意义的，比如用户会悬停在感兴趣的品类、商品或者广告上）。而客户端埋点可能会由于用户的环境问题存在数据包丢失，客户端可能无法获取全部的数据（有些数据存储在服务端，因为页面不会存储过多不会经常使用的信息，客户端需要一些调用才能够获取到，对性能会有影响）等，所以无特殊情况下，建议采用服务端埋点方案。

APP 的更新是周期性的，你没有办法强制用户去更新 APP，如果在客户端埋点的话就会由于版本更新的问题没有办法改变埋点，但是如果在服务器埋点的话就没有这个问题，可以在服务器端按照需求设置埋点，不关心 APP 的版本。

如果有一些类似于悬停这种事件，服务端无法捕获到，而这些事件又是非常重要的，那么可以在页面上加入一些代码，当发生这些事件的时候执行一个请求服务器的动作，让服务端进行埋点，负责这部分数据的落地。

### 4.1.3 业务数据埋点设计

业务数据一般存储于 RDBMS中，有业务RD根据业务流程设计，具体记录信息无需过度关注，但在可能的情况下，尽可能提出相应的规范。

每张业务表必须有自增 id 能够唯一标识一行记录必须包含该记录的写入时间(created\_time)、数据更新时间(updated\_time)，部分表最好能够在 updated\_time 列构建索引（方便后续的数据采集，根据时间进行数据的获取）

独立的业务数据埋点：由业务服务端根据具体场景的需要，独立记录和业务强相关数据日志（不是用户行为数据），通常落地于相应业务应用的服务器磁盘，需要单独采集，方案可以参考行为数据的采集。

### 4.1.4 数据埋点案例分析

1）背景：

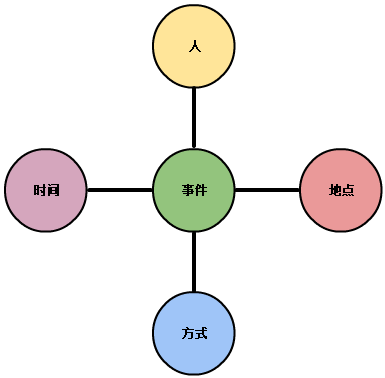
某电商平台，产品覆盖 app、pc 端，其中 app 为 hybrid 架构，业务数存储 mysql 库中；

2）需求：

需要分析用户访问路径、各步转化、流失等进行页面优化、运营效果评估、商品推荐；

3）设计埋点规范：

* 一切用户操作行为都看作事件
* 应覆盖事件的核心要素



（一）数据格式：

为确保灵活、可扩展性，上报数据采用 json 格式，不要太深的嵌套（99%的埋点都是一层），将数据分为两部分：公有参数、自定义私有参数。

（1）公有参数：

公有参数需要覆盖事件核心要素，实际上报数据时不可缺失。

{

timestamp cid

uid sessionId ipAddress page ctype type event useragent content{}

version

**时间**

**C 端标识用户 id**

**session**

**ip 地址页面**

**C 端类别(native\h5)**

**类型(pageview\click\start…) 事件**

**手机平台（Android/IOS）等信息私有自定义信息**

**日志版本（日志规范可能会升级，此处标识日志遵循的规范版本）**

}

（2）私有参数：

私有参数即根据事件发生时的场景、需求，定义需要抓取的数据内容。

某种意义上该部分数据也分公共参数与私有参数，比如事件发生在 app 中，设备信息都会伴随当时的设备信息；事件发生在 h5 中，一般会伴随 url。

对于一般的推广 H5，需要抓取基本数据：

currurl（当前页面的 URL）、refferurl（上一级页面的独立 URL）、refferpage（上一级页面的独立名称）（适配 App 页面）、sourceid（跟踪访问的源头，是微信、微博还是其他的推广平台，用以判断哪边的推广效果更好）、自定义参数(比如商品 ID、活动 ID、商家 ID、posid 等)

对于 App，一般需要附加设备信息：

refferpage（上一级页面的独立名称）、自定义参数(商品 id、商家 id 等等)、设备信息(imei、idfa、os、gps、wifi 等信息)。

对于 hybrid APP，其中包括 native 和 h5，虽然是 h5，但由于在 app 中，依然能够拿到相关设备信息，因此按照 app 的参数设计进行埋点，同时也要包括 h5 自身的特有数据，比如 url、refferurl 等。

（二）埋点数据内容设计：

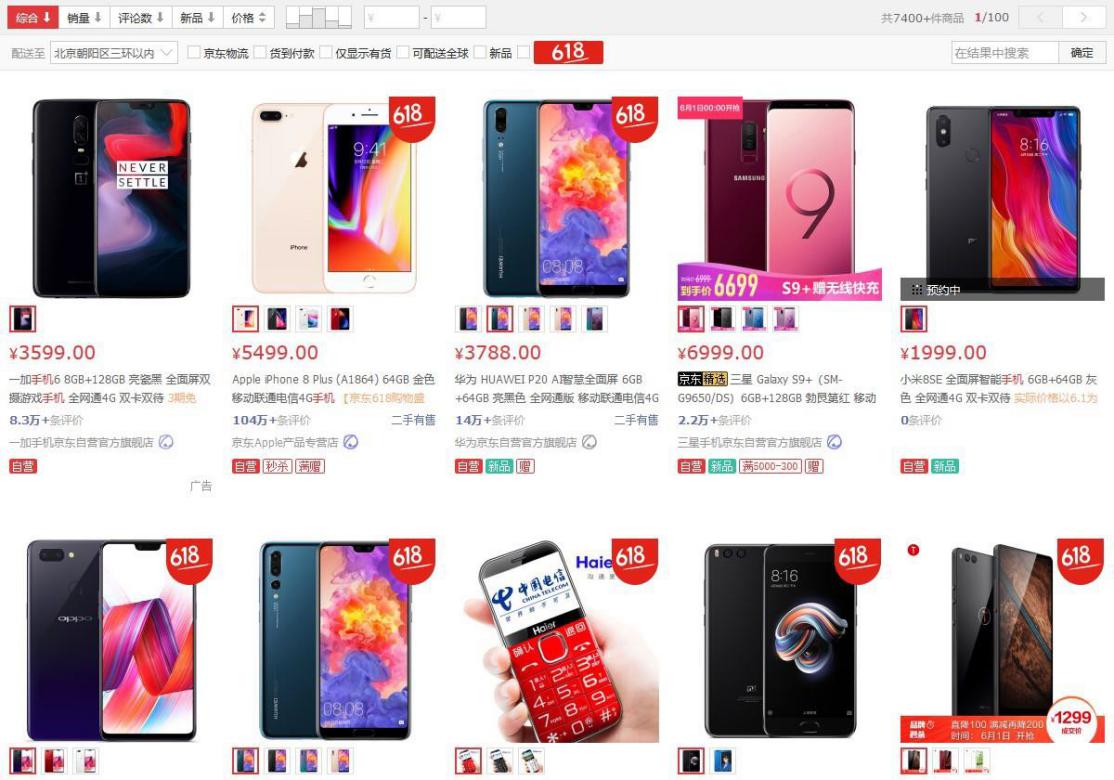
（1）APP 启动埋点：

本着抓取高价值、符合需求、不影响用户体验和性能的原则，仅在用户启动 app 时触发，在用户授权的情况下抓取用户手机中安装的 app 信息、通讯录信息，频率可以自定义。

埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识用户id*  *Uid 用户id*  *sessionId 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *Content{ 用户自定义信息*  *installapps:[{appname:, intime}, {}, ...], 用户已安装 App 列表*  *activeapps:[{appname:, starttime}, {}, ...] 用户当前启动的 App 列表*  *phones:[{name:, phone}, {}, ...] 通讯录*  *imei IMEI 号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps\_lng GPS 信息*  *gps\_lat GPS 信息*  *wifiname WIFI 名称*  *appversion App 版本*  *}*  *version 日志版本*  *}* |

（1.1）列表页：



（1.2）普通版本：

埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识*  *uid 用户id*  *sessionID 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *content{ 私有自定义信息*  *url 当前页面URL*  *refferurl 上一页URL*  *refferpage 上一页名称*  *pagenum 列表页页码（商品有多页，记录当前是第几页）*  *imei IMEI号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps GPS信息*  *wifiname WIFI名称*  *appversion APP版本*  *}*  *version 日志版本*  *}* |

（1.3）计费版本一：

页面采用翻页模式，一页只有一个事件，进入列表页后，将这一页的所有商品进行上报。

埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识*  *uid 用户id*  *sessionID 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *content{ 私有自定义信息*  *url 当前页面URL*  *refferurl 上一页URL*  *refferpage 上一页名称*  *pagenum 列表页页码（商品有多页，记录当前是第几页）*  *Goodlist[{},{}] 商品列表*  *imei IMEI号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps GPS信息*  *wifiname WIFI名称*  *appversion APP版本*  *}*  *version 日志版本*  *}* |

（1.4）计费版本二：

对于精确的曝光计费，特别是滚动刷新的方式（非翻页），用户不真正看到商品是不计入 goodlist 的，之后用户真正滚动到了商品的位置，才会计入 goodlist，上报服务器。

在计费方式二的情况下，会随着每次页面的翻转发送一条列表页日志数据。

埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识*  *uid 用户id*  *sessionID 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *content{ 私有自定义信息*  *url 当前页面URL*  *refferurl 上一页URL*  *refferpage 上一页名称*  *pagenum 列表页页码（商品有多页，记录当前是第几页）*  *goodlist[{goodsId, pos},{}] 商品列表*  *imei IMEI号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps GPS信息*  *wifiname WIFI名称*  *appversion APP版本*  *}*  *version 日志版本*  *}* |

（1.5）详情页：



（1.5.1）click：通过点击的方式进入商品详情页的埋点设计

埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识*  *uid 用户id*  *sessionID 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *content{ 私有自定义信息*  *url 当前页面URL*  *refferurl 上一页URL*  *refferpage 上一页名称*  *Goodsid 商品id*  *Sjid 商家id*  *pagenum 列表页页码（商品有多页，记录当前是第几页）*  *Positioon 点击位置*  *Prearvtime 预计收货时间*  *imei IMEI号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps GPS信息*  *wifiname WIFI名称*  *appversion APP版本*  *}*  *version 日志版本*  *}* |

（1.5.2）preview：页面加载日志

preview 事件的 PV 往往低于 click 事件，因为有的访问会加载失败。如果 preview 和 click 相差太多（同一个页面），那么就要审视是否是产品出了问题，为什么这么多次没有加载出来，是后台性能太差还是网络一直有问题。

埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识*  *uid 用户id*  *sessionID 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *content{ 私有自定义信息*  *url 当前页面URL*  *refferurl 上一页URL*  *refferpage 上一页名称*  *Goodsid 商品id*  *Sjid 商家id*  *Prearvtime 预计收货时间*  *imei IMEI号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps GPS信息*  *wifiname WIFI名称*  *appversion APP版本*  *}*  *version 日志版本*  *}* |

（1.6）首页

我们关注用户是通过什么途径进入主页的，这就涉及到渠道推广，因为很多电商在其他平台进行推广，而推广是需要交纳一定费用的，而其他平台向你提出的收费额度，你怎么确定是否准确，就要通过首页的埋点进行采集。



埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识*  *uid 用户id*  *sessionID 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *content{ 私有自定义信息*  *url 当前页面URL*  *refferurl 上一页URL*  *refferpage 上一页名称*  *Sourceid 从哪里进入的主页*  *imei IMEI号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps GPS信息*  *wifiname WIFI名称*  *appversion APP版本*  *}*  *version 日志版本*  *}* |

（1.7）banner（广告位）：

页面上的不同广告位的定价不同，同一广告位会有多个广告（滚动屏，自动切换），这些广告由于前后位置的不同收费不同，除了首页占据大面积的滚动屏，还有固定位置的广告。



埋点数据公有参数

|  |
| --- |
| *{*  *timestamp 时间*  *cid C端标识*  *uid 用户id*  *sessionID 会话id*  *ipAddress ip地址*  *page 页面*  *ctype C端类别（native、H5）*  *type 类型（pageview、click、start...）*  *event 事件*  *useragent 手机平台信息（Android、IOS）*  *content{ 私有自定义信息*  *url 当前页面URL*  *refferurl 上一页URL*  *refferpage 上一页名称*  *Position 点击位置*  *Gourl 跳转到的URL*  *imei IMEI号*  *idfa 设备唯一标识*  *os 操作系统类型*  *gps GPS信息*  *wifiname WIFI名称*  *appversion APP版本*  *}*  *version 日志版本*  *}* |