# 埋点体系

## 什么是埋点？

埋点分析，是[网站分析](https://baike.so.com/doc/5698575-5911282.html" \t "https://baike.so.com/doc/_blank)的一种常用的数据采集方法。数据埋点分为初级、中级、高级三种方式。数据埋点是一种良好的私有化部署数据采集方式。

## 埋点的方式：

**数据埋点分为初级、中级、高级三种方式，分别为:**

初级:在产品、服务转化关键点植入统计代码，据其独立ID确保数据采集不重复(如购买按钮点击率);

中级:植入多段代码，追踪用户在平台每个界面上的系列行为，事件之间相互独立(如打开商品详情页--选择商品型号--加入购物车--下订单--购买完成);

高级:联合公司工程、[ETL](https://baike.so.com/doc/2126217-2249603.html" \t "https://baike.so.com/doc/_blank)采集分析用户全量行为，建立用户画像，还原用户行为模型，作为产品分析、优化的基础。

## 为什么埋点：

埋点埋什么取决于我们想从用户身上获取什么信息，

用户的基本属性信息和行为信息

（**我们的埋点，是为了监测系统优化的点，以及用户使用情况的分析**）

用户的基本属性信息主要包括：城市、地址、年龄、性别、经纬度、账号类型、运营商、网络、设备等等

行为信息即用户的点击行为和浏览行为，在什么时间，哪个用户点击了哪个按钮，浏览了哪个页面，浏览时长等等的数据。



### 报文(message)

是网络中交换与传输的数据单元，即站点一次性要发送的数据块。

## 埋点设计：

客户端、m站等平台中，全部的页面及按钮按照规律清晰的罗列出来，并为每个页面，每个按钮配置唯一的id，以此来区分每一个页面及按钮。

埋点表还需要有**上报时机**，**存到数仓中**

上报时机一般分为**点击即上报** 还有**回调上报**

## 埋点系统：

埋点系统主要功能：

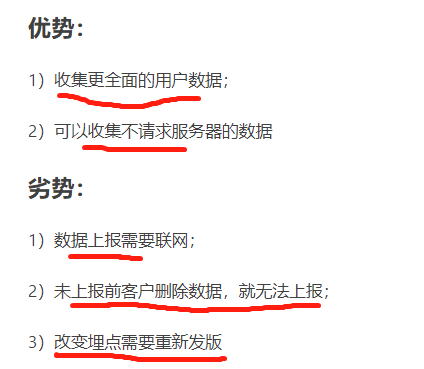
1. 维护埋点事件及埋点事件ID：
2. 数据异常报警

## 埋点的种类：

### （1）前端：客户端埋点

客户端数据很全，记录了用户绝大多数行为，其他非关键业务量或者不需要请求服务器的行为适应后端数据埋点。

音乐播放时间、用户操作序列、点击等



### （2）后端：服务端埋点

实时性高，很准确，最好用户需求请求服务器的关键业务均使用服务器后端埋点

**在线播放，中奖统计、游戏安装等**



## 埋点的方式：

### （1）代码埋点：

由APP研发工程师手工在程序中写代码实现，通过触发某个动作后程序自动发送数据



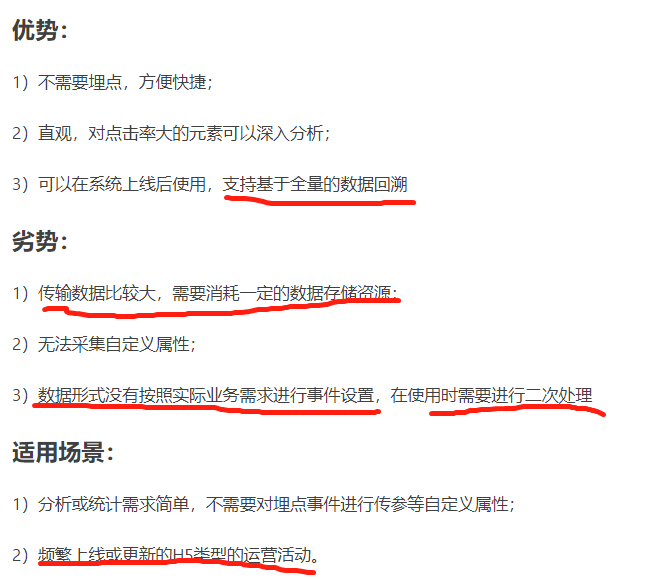
### 可视化埋点：

以前端可视化埋点的方式记录前端设置页面元素与其操作的关系，然后以后端截屏的方式统计数据。



### 无埋点：

无埋点绑定各个页面的控件，当事件触发时就会调用相关接口上报数据，可理解为通过后端或者签订可视化圈等的方式来完成关键事件的定义和捕获。他要求必须在页面嵌入数据采集基础SDK，就可以采集页面的所有点击行为，并可以绘制出用户点击的热力图。



## 埋点事件：

### 点击事件

### 曝光事件

### 页面停留时间 time on page

## 埋点指标：

### 用户变化

**新用户：**下载后第一次启动就是新用户。以后统计还需要去重。（按照设备特征码）

**活跃用户：**在某个时间段内，对APP有任何操作行为的用户（去重）

**活跃账户：**在某个时间段内，对启动过APP的账户（按账户去重）

**日活跃用户：**日新增用户（DNU）+日老活跃用户（DOU）+ 回流用户。

**周活用户：**最近7天内（包含今天）启动APP的用户数

**月活用户：最近30天内（包含今天）启动**APP的用户数（去重）。月活减少趋势明显，说明用户流失现象严重。

**回流用户：**超过N天内启动APP的用户在今天启动APP（去重），N一般=7

**累计用户：**截止到当前，启动过APP的用户（去重）

**活跃占比：**当日活跃用户数/指定时间段的活跃总数

**日活/周活：**当日活跃用户数/7天内（含当天）的活跃总数

**日活/月活：**当日活跃用户数/30天内（含当天）的活跃总数；比值越高，依赖性越高，粘性越强

**沉默用户：**激活APP后N天没有启动APP的用户数，N一般=90；

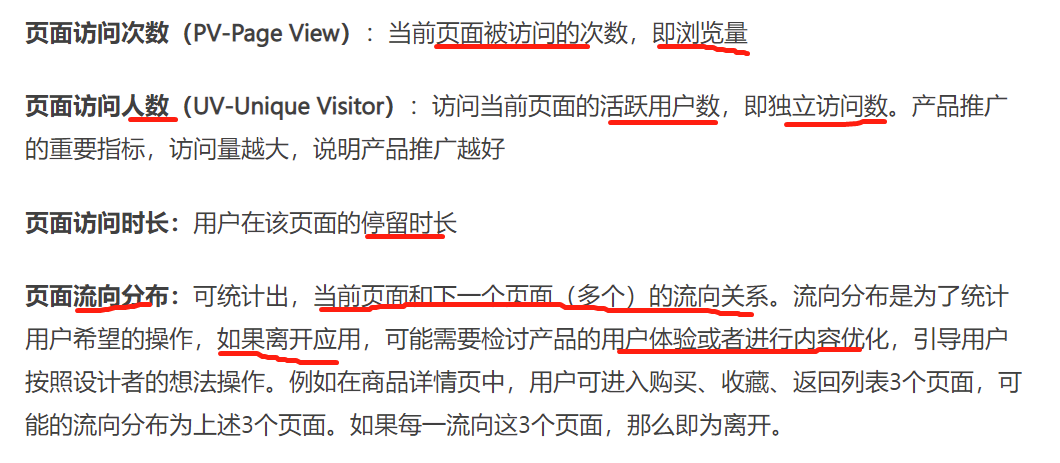
**月激活用户数：**过去30天内没有启动APP的用户数（去重），突然有大量的用户被激活，可能是最近活动或者版本更新产生的效果。

**激活行为：**一般定义用户**安装日以及安装次日**的多次启动都被视为激活行为

**新用户留存：**指定时间内新增用户在经过一段时间后，还有启动APP的用户

**新用户留存率**=留存用户/当时的新增用户，在一定程度上反映用户的忠诚度，值越高，用户忠诚度就越高。

### 页面统计：



### 终端特征

机型

分辨率

操作系统

### 版本占比

APP版本分布，了解某个时间段内，市面各个版本的用户分布

升级用户：从其他版本升级到当前版本的用户

### 交易概况

会员充值额度，同比、环比、走势

订单、交易额度、退款额、同比、环比、走势

### 渠道投放效果

了解某个时间段，APP发布渠道的用户概况，可用于后续的投放决策，天猫、京东等。

## 埋点文档如何设计：

1. 梳理出**产品的功能结构及业务流程**，将**核心流程**梳理出来，确定**关键指标**，并细化各流程的影响因素，同时想清楚上下游的**接入口**是什么，避免**埋点的重复**，**提高埋点复用性**。
2. 规划出**数据分析的框架**，基于产品功能的**路径转化**和**重要指标链路**，设计出可供记录的埋点框架，使埋点契合分析框架的逻辑，避免冗余。
3. 埋点文档需要提供给**前、后端研发**进行埋点开发，所以文档中的信息尽量描述清楚，并且与**开发拉会议**，要求**对埋点的理解**对齐一致。





## 埋点验收：

### 埋点文档评审

埋点文档设计完成后，数据组内需要进行评审，对埋点和参数逐一检验，包含：

**合理性：**是否符合用户行为路径；

**完整性：**是否覆盖产品的所有场景，可以支撑后续的数据应用；

**正确性：** 埋点文档中，除功能的特性埋点，还有一定的公共埋点、公共参数，查看是否与BI报表开发时的规范一致，如果不一致，BI报表不会产生数据。

### 埋点开发阶段

埋点开发阶段，与研发团队保持密切沟通，确保和研发的理解保持一致，使其了解每个点的意义以及后续的应用计划。针对重要埋点，重要的参数，研发需提供对应的源码，确保每个枚举值都录入代码中。

例如：用户关于货币获得，会有多重路径，如果研发将其中两种路径漏写，后续分析中，会造成数据结果的缺失。

### 埋点验收阶段

埋点完成，安装包提交之后，数据同学会配合QA同学，一起做埋点验收，需注意以下几个方面：

**转化漏斗是否正常：**例如广告链路中，从广告展现-曝光-点击-关闭，这条链路的pv是呈漏斗逐渐减少，如果不是，那么需要定位埋点问题；

**上报顺序是否正常：**新手引导中，id顺序为1-2-3-4，可追踪单一用户id，按时间戳查看上报顺序是否符合规范；

**埋点上报**是否对应规范中的触发场景。

实际业务中的埋点验收，是一个复杂繁琐的工作，每个项目对应不同的规范，建议**建立一个《埋点验收清单list》**，记录需要验收的部分，指派到责任人，逐一签名查验，防止错漏。

## 数据存储：

数据从上报开始——Nginx——Filebeat——Kafka——Spark——Kudu

### （1）Nginx

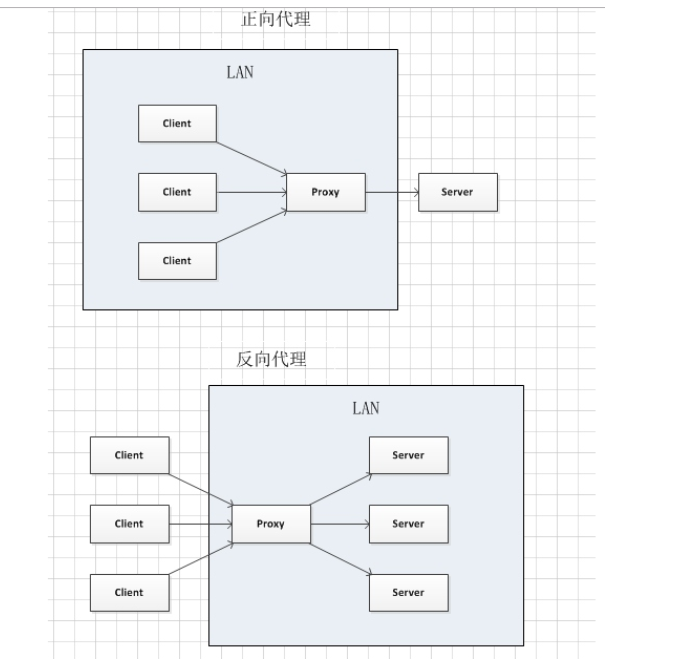
Http代理，反向代理：作为web服务器最常用的功能之一，尤其是反向代理。

Nginx同Apache一样都是一种WEB服务器，基于REST架构风格，以统一资源描述符(Uniform Resources Identifier)URI或者统一资源定位符(Uniform Resources Locator)URL作为沟通依据，通过HTTP协议提供各种网络服务。

多个客户端给服务器发送的请求，Nginx服务器接收到之后，按照一定的规则分发给了后端的业务处理服务器进行处理了。

正向代理，"它代理的是客户端，代客户端发出请求"，是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。

多个客户端给服务器发送的请求，Nginx服务器接收到之后，按照一定的规则分发给了后端的业务处理服务器进行处理了。此时~请求的来源也就是客户端是明确的，但是请求具体由哪台服务器处理的并不明确了，Nginx扮演的就是一个反向代理角色。



在正向代理中，Proxy和Client同属于一个LAN（图中方框内），隐藏了客户端信息；

在反向代理中，Proxy和Server同属于一个LAN（图中方框内），隐藏了服务端信息；

### （2）Filebeat

Filebeat是用于转发和集中日志数据的轻量级传送工具。Filebeat监视您指定的日志文件或位置，收集日志事件，并将它们转发到Elasticsearch或 Logstash进行索引。

<https://www.cnblogs.com/zsql/p/13137833.html>

### （3）Kafka

Kafka是最初由Linkedin公司开发，是一个分布式、分区的、多副本的、多订阅者，基于zookeeper协调的**分布式日志系统**（也可以当做**MQ系统**），常见可以用于web/nginx日志、访问日志，消息服务等等。

主要应用场景是：**日志收集系统**和**消息系统**。

布式消息传递基于可靠的消息队列，在客户端应用和消息系统之间异步传递消息。有两种主要的消息传递模式：点对点传递模式、发布-订阅模式。大部分的消息系统选用发布-订阅模式。Kafka就是一种发布-订阅模式。

与点对点消息系统不同的是，消费者可以订阅一个或多个topic，消费者可以消费该topic中所有的数据，同一条数据可以被多个消费者消费，数据被消费后不会立马删除。在发布-订阅消息系统中，消息的生产者称为发布者，消费者称为订阅者。

### （4）Spark

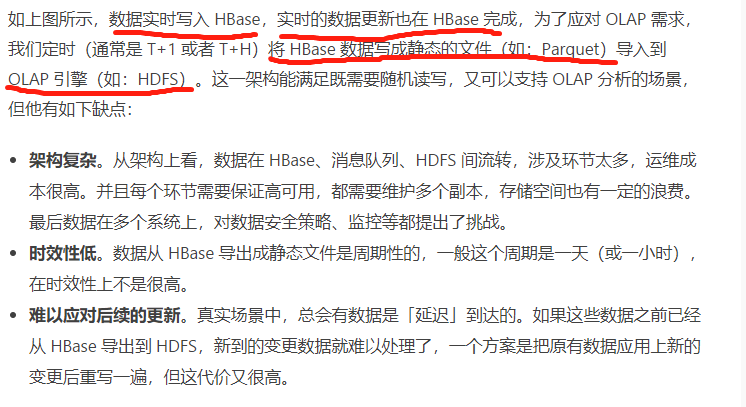
Apache Spark 是专为大规模[数据处理](https://baike.so.com/doc/6144874-6358051.html)而设计的快速通用的计算引擎。是类Hadoop MapReduce的通用并行框架，Spark，拥有Hadoop MapReduce所具有的优点;但不同于MapReduce的是--Job中间输出结果可以保存在内存中，从而不再需要读写HDFS，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的MapReduce的算法。

### Kudu

在 KUDU 之前，大数据主要以两种方式存储：

**静态数据：**以 HDFS 引擎作为存储引擎，适用于高吞吐量的离线大数据分析场景。这类存储的局限性是数据无法进行随机的读写。

**动态数据：**以 HBase、Cassandra 作为存储引擎，适用于大数据随机读写场景。这类存储的局限性是批量读取吞吐量远不如 HDFS，不适用于批量数据分析的场景。



为了解决上述架构的这些问题，KUDU 应运而生。KUDU 的定位是 「Fast Analytics on Fast Data」，是一个既支持随机读写、又支持 OLAP 分析的大数据存储引擎。

# 2、指标系统构建：

数仓的底层表格结构，将指标的组成部分解耦拆分数来，并在逻辑表中进行结构化处理，后续可以进行一定规则的自由拼装，实现自定义指标的功能。

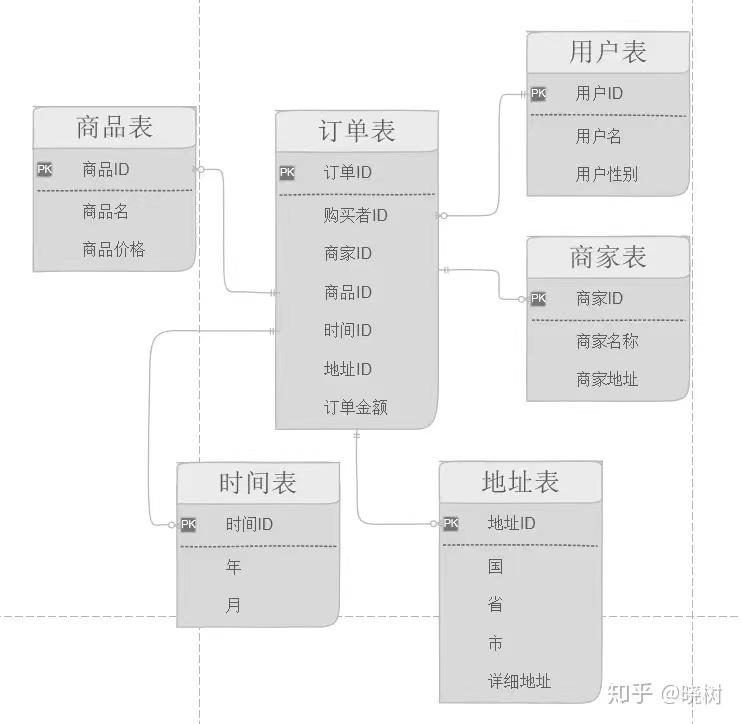
如何构建？盘点业务+数据种类  
**第一：**梳理产品中的实体，实体表。实体和关系。用户、银行、活动、会员

所以会有**用户表**（用户自己的属性）、**商品表**等。用户与商品之间通过**事件关联**

**第二：**实体间的行为（行为关系、归属关系），事实表和维度表（维度+度量值）的设计

维度表：维度属性的集合，【商品表、用户表、商家表、时间表、地址表】

事实表：事件发生的时候的度量值 比如某区的销售额【订单表】



**第三：**梳理产品+业务事件=====通过外键与维度表进行关联【星型模型】

**第四：**多维事件模型，为了查询和分析，就建立了多维事件模型OLAP

以上完成了 梳理实体模型，初步定义实体表。即**领域建模概念建模。**

**第五：**从数据分析的角度出发，拆分维度和事实。进行逻辑建模，即数仓模型

**第六：**定义指标的生产逻辑。

**原子指标+派生指标**

原子：不可拆分，是事件行为下的度量。

订单支付金额、支付单数、PV/UV 但不能取数。

派生指标：可以直接取数的。店铺近一天的订单支付金额【谁+时间+什么事件】

但是也不能直接从数仓中统一中间层里取数，需要一个桥梁连接数仓中间层与业务方的指标需求，故而产生了派生指标

**派生指标=维度+原子指标+修饰词**

**第七：指标管理平台对指标进行规范化生产**

**所以需要有功能模块**

1. **数据源同步：DW库，为保证指标的正确唯一性，指标库是元数据。维度信息+原子指标**
2. **维度管理：数仓设置维度主表和关联的事实表 group by**
3. **原子指标的管理：**
4. **修饰词的管理：where 条件**

# 银行指标：

**业务层核心指标：**交易量、交易失败量、交易变化率、活跃用户变化率。

银行业务的核心信息系统，将IT指标分为四层：应用层、服务层、中间件层、主机层

# 指标体系

### 什么是指标体系？

指标体系是从不同维度梳理业务，把指标有系统地组织起来。简而言之，指标体系=指标+体系，所以一个指标不能叫指标体系，几个毫无关系的指标也不能叫指标体系。

### 指标体系有什么用?

对于一家公司的业务是否正常（健康），可以通过指标体系对业务进行监控。

指标体系是数据分析、数据产品、数据运营和数据营销的一项基本技能。借助指标体系把业务逻辑化，可以快速测量用户对产品的使用情况，精准的定位问题分析问题，也可以有助于洞见商业机会。

#### 监控业务情况；

#### 通过拆解指标寻找当前业务问题

#### 评估业务可改进的地方，找出下一步工作的方向。

### 如何构建指标体系？

数据时代**，用户**在某种**场景（业务）**下活动产生大量**数据。**

**用户维度、业务维度**



#### 明确部门KPI，找到合适的一级指标

一级指标是用来评价公司或部门运营情况最核心的指标。

#### 了解业务运营情况，找到二级指标

有了一级指标以后，可以进一步将一级指标拆解为二级指标。具体如何拆解，要看业务是如何运营的。

#### 梳理业务流程，找到三级指标

一级指标往往是业务流程最终的结果，例如积分抵扣金额，是业务流程（会员-›购买旅游产品-›使用积分抵扣-›支付金额） 最后的一个结果。

在业务流程中不同会员等级可以抵扣的金额不一样。不同旅游产品线可以抵扣的金额比例也不一样。所以，需要把二级指标按照业务流程拆解为更细的三级指标。

#### （4）从指标确定统计口径

**指标含义：**这个指标在业务上表示什么？

**指标定义：**这个指标是怎么定义的？

**数据来源：**从什么地方收集的原始数据？数据统计的时间范围是什么？

#### （5）通过报表监控指标，不断更新指标体系

可以把这些指标制作到报表中，通过报表监控指标，不断更新指标体系。

### 指标体系的分类：

把**顶层指标**分为**独立指标体系**和**复合指标体系**，复合指标体系可以拆解成一个公式。

### 指标体系的要素：

母指标、子指标和指标之间的关系，这种关系通常体现为四则运算。

### 指标体系的指导思想：

构建指标体系的**指导思想就是拆解**，即按一定规则，把母指标拆解成多个子指标，构成一个指标体系。

从二维空间视角看，拆解就两个方向：水平方向（横向）、垂直方向（纵向）。

总的拆解思路是：横向**扩展拆解**，纵向**细分拆解，**建立层次**指标体系。**

### 指标体系拆解的原则：

#### MECE原则（相互独立，完全穷尽）、CSCE原则（完备性、系统性、可执行性、可解释性）。

#### 价值化、可操作性、独立性

### 问题指标体系构建中

#### 没有一级指标，抓不住重点

#### 指标之间没有逻辑关系

如果不按照业务流程来建立指标体系，虽然指标很多，但是指标之间没有逻辑关系。以至于出现问题的时候，找不到对应的业务节点是哪个，没办法解决问题。

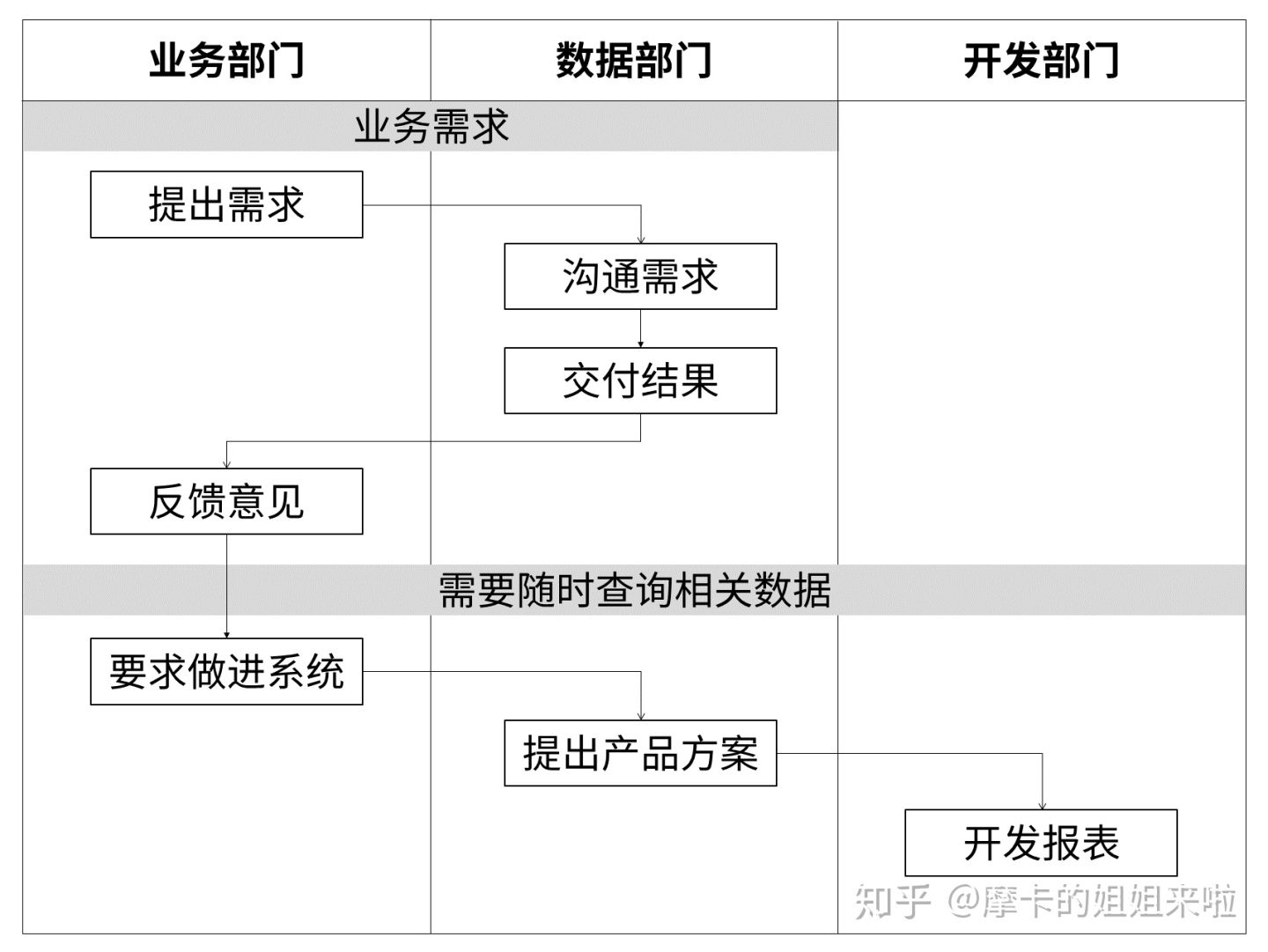
#### 拆解的指标没有业务意义

有的报表上的指标很丰富，但是却没有实际的业务意义，导致报表就是一堆“没有用”的数字。

例如：在销售部门，最关注的是销售目标有没有达成，现在达成了多少，接下来的每天应该达成多少，哪些些区域达成最高，哪些区域达成最低。如果不围绕这个业务目标拆解指标，而是随意把指标拆解为用户年龄、性别，这就与业务没有任何关系，只是为了拆解而拆解。

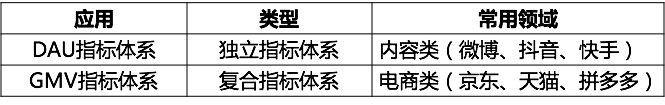
#### 一个人就完成了指标体系和报表，也不和业务沟通

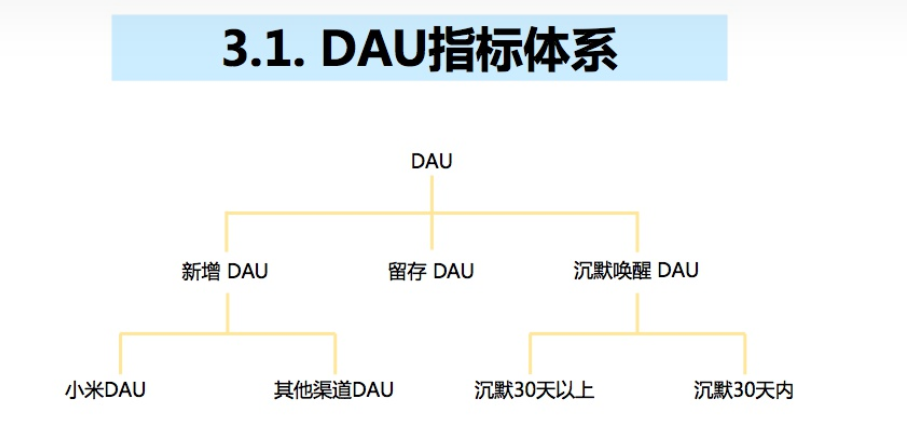
建立指标体系不是1个人能够完成的，需要业务部门（市场、运营、产品等部门统称为业务部门）、数据部门（这里把数据分析师所在的部门统称为数据部门）、开发部门相互之间进行协作。



### 指标体系应用：

独立指标体系（DAU）和复合指标体系（GMV）的应用







### 指标体系的价值：

指标体系最大的价值就是把业务逻辑化，在工作中有了目标，有了方向，有了评估迭代准则。



# 运营商主要有三类：

切分主营业务收入类的考核

用户规模、收入、业务量的展现监控考核

一线营销人员的积分计算、量化薪酬等



经营分析系统 应该具备 考核功能、监控功能、报表功能、数据采集加工上传

数据种类：收入类、用户类 业务使用类 业务受理类