> 关注公众号：`大数据技术派`，回复:`资料`，领取`1024G`资料。

分层建设理论

简单点儿，直接ODS+DM就可以了，将所有数据同步过来，然后直接开发些应用层的报表，这是最简单的了；当DM层的内容多了以后，想要重用，就会再拆分一个公共层出来，变成3层架构,这个过程有点类似代码重构，就是在实践中不断的进行抽象、总结。

数仓的建模或者分层，其实都是为了更好的去组织、管理、维护数据,所以当你站在更高的维度去看的话，所有的划分都是为了更好的管理。小到JVM 内存区域的划分，JVM 中堆空间的划分(年轻代、老年代、方法区等)，大到国家的省市区的划分，无一例外的都是为了更好的组织管理。

所以数仓分层是数据仓库设计中十分重要的一个环节，**优秀的分层设计能够让整个数据体系更容易理解和使用。**

这一节，我们主要是从整体上出发进行分析和介绍，就和上一节数仓建模方法论一样，进度对比分析，更多细节的东西我们后面会单独拆分出来，用案例进行演示，例如维度建模，维度表的设计，事实表的设计、以及如何设计标签、如何管理标签等等。

分层的意义

清晰数据结构体系

每一个数据分层都有它的作用域，这样在使用表的时候能更方便的定位和理解。

数据血缘追踪

由于最终给业务呈现的是一个能直接使用的业务表，但是表的数据来源有很多，如果有一张来源表出问题了，我们希望能够**快速准确的定位到问题，并清楚它的影响范围，从而及时给到业务方反馈，从而将损失降到最低。**

减少重复开发和资源浪费

- 规范数据分层，开发一些通用的中间层数据，能够减少极大的重复计算；
- 清晰明了的结构使得开发、维护的成本降低；
- 减少重复计算和存储的资源浪费；

复杂问题简单化

将一个复杂的任务分解成多个步骤来完成，每一层只处理单一的步骤，比较简单和容易理解。而且便于维护数据的准确性，当数据出现问题之后，可以不用修复所有的数据，只需要从有问题的步骤开始修复。

> 在实际的建设过程中，由于业务使用数据非常紧急以及统一数仓层建设跟不上业务的需要，所以DIM和ADS层可能直接使用ODS层进行快速的业务响应，但是这种不规范的操作可能导致数据口径不一致，**所以待数仓建设完毕，要切换到统一数仓层和DIM层。**

统一数据口径

过数据分层提供统一的数据出口，统一对外输出的数据口径，这往往就是我们说的数据应用层。

关于分层的一点思考

前面我们说到分层其实是为了更好更快更准的组织管理，但是这个是从宏观上来说的，接下来我们从微观上也来看一下分层。

越靠上的层次，对应用越友好,比如ADS层，基本是完全为应用设计,从数据聚合程度来讲，越上层的聚合程度越高，当然聚合程度越高可理解程度就越低。

数仓层内部的划分不是为了分层而分层，**分层是为了解决 ETL 任务及工作流的组织、数据的流向、读写权限的控制、不同需求的满足等各类问题**，当然我们常说的分层也是面向行业而言的，也是我们常用分层方法，但是你需要注意的是分层仅仅是手段而已。

数仓的分层

ods 操作数据层

ODS 全称是 OperationalDataStore，**操作数据层存储的是面向业务系统的数据**，也是最接近数据源中数据的一层，数据源中的数据，经过抽取、洗净、传输，也就说传说中的 ETL 之后，装入本层。

> 其实这里说ETL 有点不合适了，其实更准确的是ELT,你可以细细品品

本层的数据，总体上大多是**按照源头业务系统的分类方式而分类的**，前面我们说到为什么在数仓主要用维度建模的情况下，我们依然要学习范式建模呢，因为我们的数据源是范式建模的，所以学习范式建模可以帮助我们更好的理解业务系统，理解业务数据，所以你可以认为我们的ODS 层其实就是用的实范式建模。

> 但是，这一层面的数据却不等同于原始数据。在源数据装入这一层时，要进行诸如**去噪**(例如有一条数据中人的年龄是300岁，这种属于异常数据，就需要提前做一些处理)、去重(例如在个人资料表中，同一ID却有两条重复数据，在接入的时候需要做一步去重)、字段命名规范等一系列操作。

这里的数据处理，并不涉及业务逻辑，仅仅是针对数据完整性以及重复值和空值的处理，其实就是做的是数据规约，数据清洗，但是为了考虑后续可能追溯数据源问题，因此**对这一层不建议做过多的数据清洗工作**，原封不动接入源数据即可，至于数据的去噪，去重，异常值处理等过程可以放在后面的DW层

> 其实关于这一层，很多人的理解不太一样，那就是是否要进行数据清洗，其实还是取决于公司的使用习惯，其实有很多公司在这一层之前也会形成一个层，名字千奇百怪，但是它的目的是数据缓冲，然后进行清洗，清洗之后的数据存入ODS ,而这个时候缓冲层数据存放一般为一周左右，几乎不会超过一个月；而ODS则永久存放。

设计原则

表名的设计 `ODS_业务系统_表名_标记`，这样的设计可以保持与业务表名一致，又可以有清晰的层次，还可以区分来源。标记一般指的是其他数仓特有的属性，例如表是天级的还是小时的，是全量的还是增量的。

- ods 层**不做字段名归一和字段类型统一的操作，如果需要则使用兼容的数据类型**
- 对于增量表，需要设计增量表(ODS*业务系统*表名_delta)和全量表,然后将增量表合并成全量表数据；
- 对于半结构化数据需要设计解析；
- 由于业务数据库（OLTP）基本按照维度模型建模，因此ODS层中的建模方式也是维度模型；

`ods` 的设计可以保证所有的数据按照统一的规范进行存储。

DW 统一数仓层

DW是数据仓库的核心，从ODS层中获得的数据按照主题建立各种数据模型。DW又细分数据明细层 DWD 和轻度汇总层DWS

这一层和维度建模会有比较深的联系，业务数据是按照**业务流程方便操作的角度**来组织数据的，而统一数仓层是**按照业务易理解的角度或者是业务分析的角度**进行数据组织的，定义了一致的指标、维度，各业务板块、数据域都是按照统一的规范来建设，从而形成统一规范的**标准业务数据体系**，它们通常都是基于Kimball的维度建模理论来构建的，**并通过一致性维度和数据总线来保证各个子主题的维度一致性。**

> 如果 ods 层的数据就非常规整，基本能满足我们绝大部分的需求，这当然是好的，这时候dwd层其实就简单了很多，但是现实中接触的情况是 ods 层的数据很难保证质量，毕竟数据的来源多种多样，推送方也会有自己的推送逻辑，在这种情况下，我们就需要通过额外的一层 dwd 来屏蔽一些底层的差异。有没有很像JVM。

设计原则

一致性维度规范

公共层的维度表中相同维度属性在不同物理表中的字段名称、数据类型、数据内容必须保持一致，因为这样可以降低我们在使用过程中犯错误的概率，例如使用了不正确的字段，或者因为数据类型的原因导致了一些奇怪的错误

维度的组合与拆分

将维度所描述业务相关性强的字段在一个物理维表实现。相关性强是指经常需要一起查询或进行报表展现、两个维度属性间是否存在天然的关系等。例如，商品基本属性和所属品牌。

DWD 明细数据层

公告明细数据层，可以说是我们数仓建设的核心了。

DWD层要做的就是将**数据清理、整合、规范化、脏数据、垃圾数据、规范不一致的、状态定义不一致的、命名不规范的数据都会被处理。**然后加工成面向数仓的基础明细表，这个时候可以加工一些面向分析的大宽表。

DWD层应该是覆盖所有系统的、完整的、干净的、具有一致性的数据层。在DWD层会根据维度模型，设计事实表和维度表，也就是说DWD层是一个非常规范的、高质量的、可信的数据明细层。

DWS 轻度汇总层

DWS层为**公共汇总层**，这一层会进行轻度汇总，粒度比明细数据稍粗，**基于DWD层上的基础数据，整合汇总成分析某一个主题域的服务数据**，一般是也是面向分析宽表或者是面向某个注意的汇总表。DWS层应覆盖80%的应用场景，这样我们才能快速响应数据需求，否则的话，如果很多需求都要从ods开始做的话，那说明我们的数仓建设是不完善的。

例如按照业务划分，例如流量，订单，用户等，生成字段比较多的宽表，用于后续的业务查询，OLAP分析，数据分析等。

一般采用维度模型方法作为理论基础，更多的采用一些维度退化手法，将维度退化至事实表中，减少维度表与事实表的关联，提高明细数据表的易用性；同时在汇总数据层要加强指标的维度退化，采用更多的宽表化手段构建公共指标数据层，提升公共指标的复用性，减少重复加工。

DIM 维度层

维表层，所以其实维度层就是大量维表构成的，为了统一管理这些维度表，所以我们就建设维度层，维度表本身也有很多类型，例如稳定维度维表，渐变维度维表。

维度指的是观察事物的角度，提供某一业务过程事件涉及用什么过滤和分类的描述属性，"谁、什么时候、什么地点、为什么、如何"干了什么，维度表示维度建模的基础和灵魂。

> 比如，"小王早上在小卖部花费5元钱购买了包子"，时间维度——早上，地点维度——小卖部，商品维度——包子 那么事实表呢？

所以可以看出，维度表包含了业务过程记录的业务过程度量的上下文和环境。维度表都包含单一的主键列，**维度表设计的核心是确定维度字段，维度字段是查询约束条件(where)、分组条件(group)、排序(order)，与报表标签的基本来源。**

维度表一般为**单一主键**，在ER模型中，实体为客观存在的事务，会带有自己的描述性属性，属性一般为文本性、描述性的，这些描述被称为维度。维度建模的核心是**数据可以抽象为事实和维度**，维度即观察事物的角度，事实某一粒度下的度量词，**维度一定是针对实体而言的。**

每个维度表都**包含单一的主键列。**维度表的主键可以作为与之关联的任何事实表的外键，当然，维度表行的描述环境应与事实表行完全对应。 维度表通常比较宽，是扁平型非规范表，包含大量的低粒度的文本属性。例如customer（客户表）、goods(商品表)、d_time(时间表)这些都属于维度表，这些表都有一个唯一的主键，然后在表中存放了详细的数据信息。

设计原则

维度表通常比较宽**，包含多个属性、是扁平的规范表**，实际应用中包含几十个或者上百个属性的维度并不少见，所以**维度表应该包括一些有意义的描述，方便下游使用。**

维度表的维度属性，应该尽可能的丰富，所以维度表中，经常出现一些反范式的设计，把其他维度属性并到主维度属性中，**达到易用少关联的效果。**

维度表的设计包括维度选择，主维表的确定，梳理关联维度，定义维度属性的过程。

维度的选择一般从报表需求和从业务人员的交谈中发现，主要用于过滤、分组、排序，主维度表一般从业务库直接同步，比如用户表，但是数仓的本身也会有自己的维度，这是因为数仓是面向分析的，所以会有很多从分析的角度出发的维度。

关联维度主要是不同业务系统或者同一业务系统的表之间存在关联性(范式建模)，根据对业务表的梳理，确定哪些表和主维度表之间存在关联关系，并选择其中的某些表用于生成维度属性。

TDM 标签数据层

随着互联网的普及，获客成本越来越高，这也使得公司对用户运营提出了更高的要求，不仅需要精细化更需要个性化。解决这一问题的办法之一就是建立相对完备的标签系统，而数仓的标签层对于标签系统而言就像数据仓库对于数据系统一样，有着举足轻重的地位，这样的标签系统需要与业务进行紧密结合，**从业务中获取养分—用户标签，同时也要服务于业务—给用户提供更加精准和个性的服务。**

底层的标签系统就像一个索引，层层展示大千世界，而用户就从这大千世界中不断选择一些东西表明自己的身份和喜好，也不断反哺，使得这个大千世界更加丰富多彩。其实到最后用户就是一些标签的集合。

对跨业务板块、跨数据域的特定对象进行数据整合，通过统一的ID-Mapping 把各个业务板块，各个业务过程中**同一对象的数据打通**，形成对象的全域数据标签体系，方便深度分析、挖掘、应用。ID-Mapping 可以认为是通过对象的标识对不同数据体系下相同对象进行关联和识别。 对象的标识可以标识一个对象，一般是对象的ID,比如手机号，身份证，登录账号

> 一个自然人他有身份证号码进行唯一标识，但是在医保的时候他使用的实医保账号，缴纳水电费的时候又是不同的账号，使用手机的时候又是设备账号，上网的时候是网商账号。在确认对象后，由于同一对象在不同的业务体系中的对象标识是不一样的，因此需要将同一对象上的不同ID 标识打通，以便所有的业务数据都能够在该对象上打通。这就是ID-Mapping。

完成对象的ID 打通需要给对象设置一个超级ID,需要根据对象当前业务体系的ID和获取得到或者计算得到超级ID,进而完成所有业务标识的ID打通一般来说ID打通是建设标签体系的前提，如果没有ID打通就无法收集到一个对象的全面信息，也就无法对这个对象进行全面的标签刻画。

传统的计算方法要有 ID-ID之间的两两关系，例如邮箱和手机号可以打通，手机号和身份证号可以打通，那么邮箱就和身份证号可以打通，但是当数据量非常大，且业务板块非常多的时候，例如有上一个对象，每个对象有数十种ID,这个时候打通就需要非常漫长的计算

那么什么是标签呢，利用原始数据，通过一定的逻辑加工产出直接能被业务所直接使用的、可阅读的，有价值的数据。标签类目，是标签的分类组织方式，是标签信息的一种结构化描述，目的是管理、查找，一般采用多级类目，一般当一个对象的标签个数超过50个的时候，业务人员查找标签就会变得非常麻烦，这个时候我们往往会通过标签类目进行组织管理

标签的分类

标签按照产生和计算方式的不同可分为属性标签，统计标签，算法标签，关联标签。

属性标签

对象本身的性质就是属性标签，例如用户画像的时候打到用户身上的标签。

统计标签

对象在业务过程中产生的原子指标，通过不同的计算方法可以生成统计标签。

算法标签

对象在多个业务过程中的特征规律通过一定的算法产出的标签。

关联标签

对象在特定的业务过程会和其他对象关联，关联对象的标签也可以打在主对象上。

设计原则

我们的标签一定是针对用户的，而不是一些虚假、高大上、无用的标签，一定要真实反映用户行为喜好的，所以我们不能只依赖人工智能算法的分析，来完成对一个用户标签的建立与定期维护，我们需要走出去和用户交互，引导用户使用，要抓住用户痛点，及时获取用户反馈，形成闭环。

如何引导使用呢？这个方式有很多我们就不再这里介绍了，后面我们会专门介绍这一层的建设细节。

ADS 层

数据应用层ApplicationDataService面向业务定制的应用数据，主要提供给数据产品和数据分析使用的数据，一般会放在ES，MYSQL，Redis等系统供线上系统使用，也可以放在Hive中供数据分析和数据挖掘使用，或者使用一下其他的大数据工具进行存储和使用。

数仓层，DIM 层，TDM 层是相对稳定的，所以无法满足灵活多变业务需求，所以这和数仓层的规范和划分相矛盾，所以我们在此基础上建立了另外一个层，这就是ADS 层，解决了规划稳定和灵活多变之间的矛盾。其实到这里你也就慢慢的看明白了，分层和分类其实没多大差别，其实就是相似的放在一起，有点代码重构的意味啊。

数据应用层，按照业务的需要，然后从统一数仓层和DIM进行取数，并面向业务的特殊需求对数据进行加工,以满足业务和性能的需求。ADS 层因为面向的实众多的需求，所以这一层没有太多的规范，只需要按照命名规范来进行就可以了。

设计原则

前面也说了，ADS 层因为面向的实众多的需求，所以这一层没有太多的规范，但是ADS 层的建设是强业务推动的，业务部门需要参与到ADS 的建设中来，至少我们得了解用户的痛点才能对症施药啊。

实现流程

理清需求，了解业务方对数据内容、使用方式(怎么交互的，报表、接口、即席查询、在线查询、指标查询、搜索)、性能的要求。

盘点现有的数仓表是否可以支持，看以前有没有类似的需求，有没有可以复用的接口、报表什么的。

代码实现，选择合适的存储引擎和查询引擎，配置线上监控然后交付。

使用场景与性能

- 针对业务方的使用场景，我们需要设计出高效，满足要求的ADS 层表
- 如果是多维分析，为了减少连接，提升性能，我们一般采用大宽表设计，使用高性能引擎支撑
- 如果是特定指标查询，一般采用KV的形式组织
- 如果是搜索场景，一般采用搜索引擎

DM 数据集市层

主要是提供数据产品和数据分析的数据，一般会存放在ES、Mysql、也可能直接存储在hive中或者druid供数据分析和数据挖掘使用。主要**解决部门用户报表和分析需求**而建立数据库，数据集市就代表数据仓库的主题域。

DM 是面向单个主题的，所以它不会从全局考虑进行建设，只专注于自己的数据、往往是某个业务线，例如流量主题、社交主题、电商主题等等。