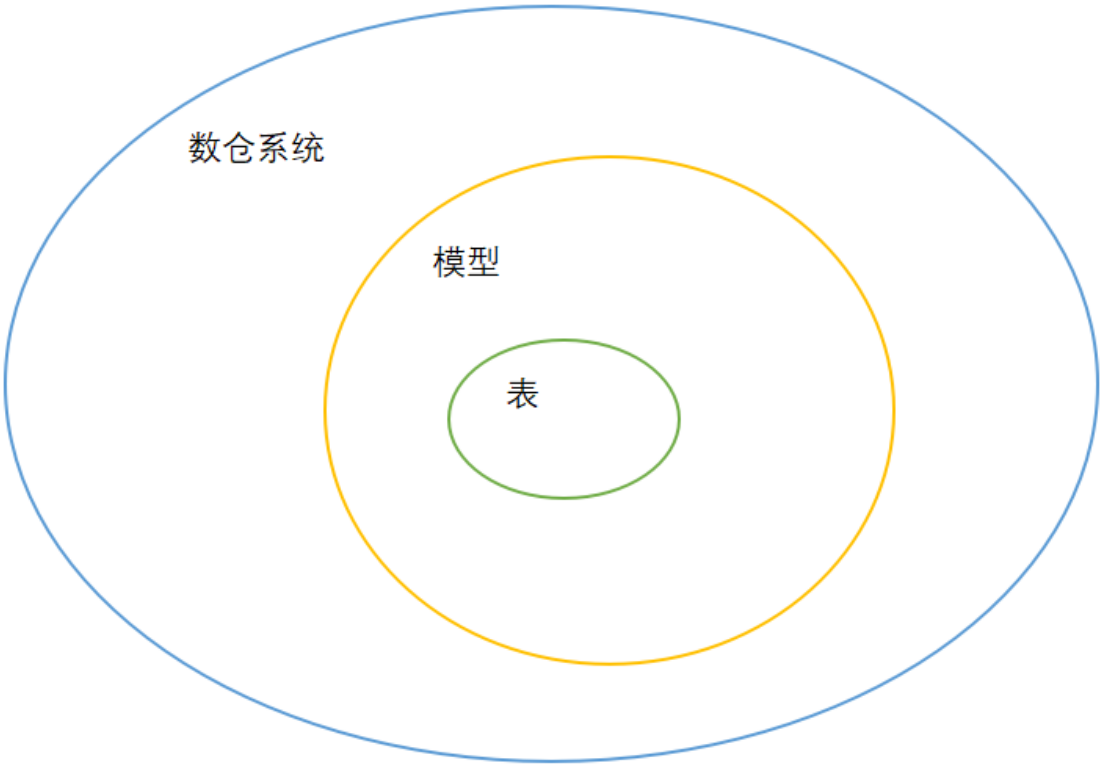
一、数仓为什么要建模（坑）

1. 一个需求一个模型，要膨胀了……
2. 原来的模型跑的时间越来越长了？
3. 有些kpi指标要早点出来，但是受模型里个别指标硬生生的拖慢了速度？
4. 发现一个bug，想快速补数据，发现只能串行一天天的跑？
5. 怎么好多模型的指标有重复咧？
6. 这个指标长的一样，咋定义不一样呢？
7. 怎么同样口径的指标，从不同的表计算，数值不一样呢？
8. 这咋每天都在延迟报警诶？每天都要运维，心塞塞~~
9. 业务要这个数据，哪个表计算的来着？
10. 常用指标和僵尸指标都在一个模型躺着，删怕影响业务，用户难受，不删我难受，浪费资源 ……

淌过这些坑后，如醍醐灌顶，这其实就是一个系统嘛。它不是一张表，也不是一个模型，而是一个有着完整体系架构和规范，需要同时考虑功能性需求和非功能性需求的系统。功能需求很好理解，就是满足业务需求，产出数据和指标。难点在于非功能性需求，比如：指标质量高不高、数据安不安全、数据产出快不快、模型是否稳定可扩展、数据服务稳不稳定、代码是否健壮、是否浪费了计算资源和存储资源……

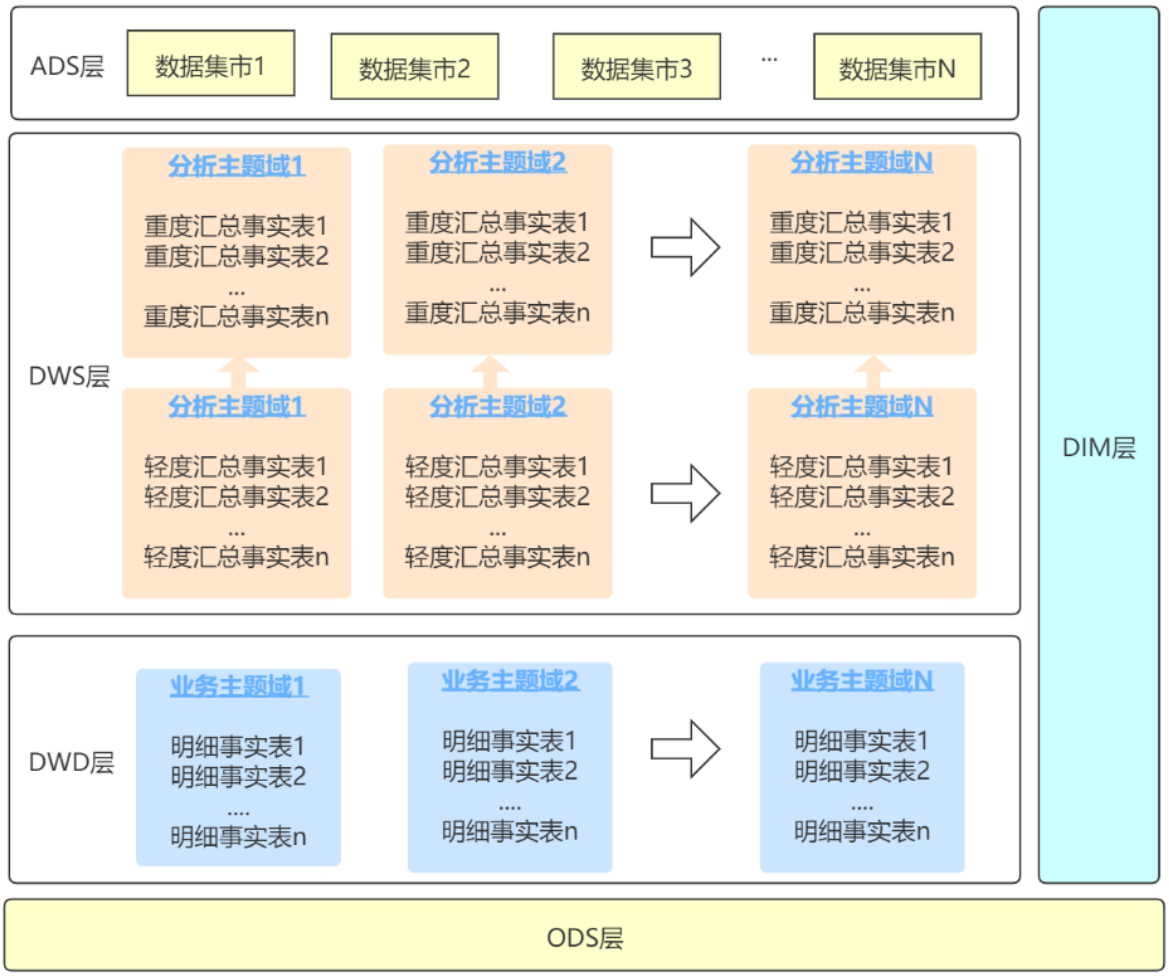


总之，数据模型就是数据组织和存储方法，它强调从业务、数据存取和使用角度合理存储数据。它的价值体现在：数据质量、健壮水平、服务响应速度、资源消耗。转化成对企业数仓的要求就是：强稳定、高质量、高效率、低成本。

二、搭建模型架构

这一环节，我给它取了个名字，叫"定调"。这里主要根据各自部门的业务形态和数据服务特点来确定整体数仓建设的基调。

首先，明确数仓各层次的要做的事情，下面是云音乐模型层次的框架图：

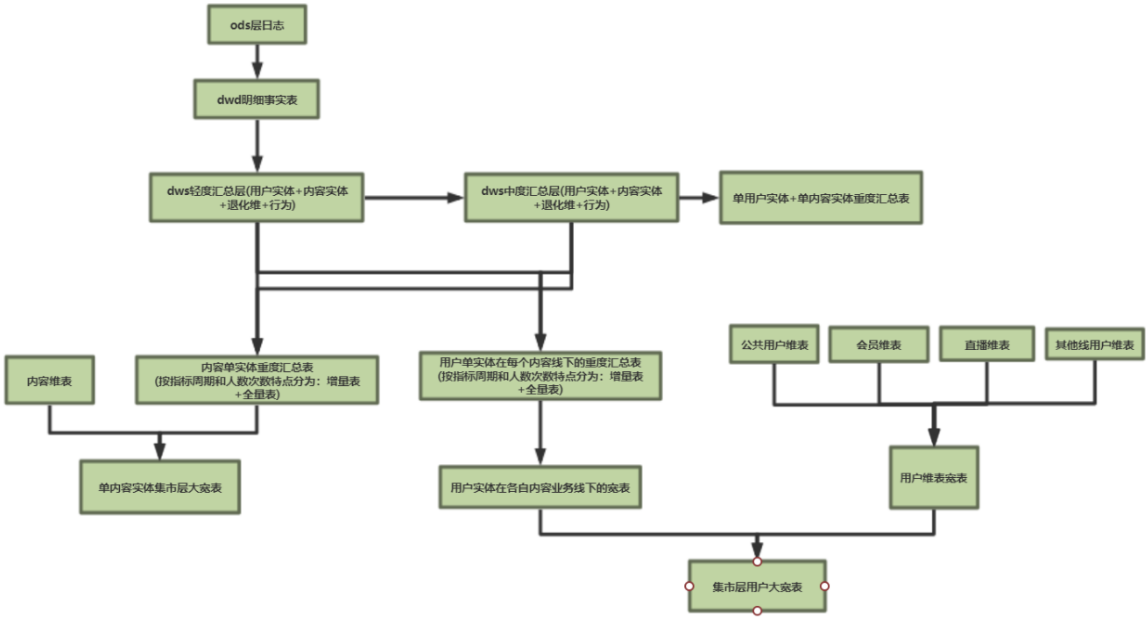


- DWD：存放面向业务过程的明细事实数据，做一些数据清洗和规范化等。
- DWS：面向分析主题建模，这里要说明的是，云音乐dws层的基架是轻度汇总事实表，这里做了一些常用的退化维，基本要求是：大部分指标都可以从轻度汇总层上计算得出。中度汇总层按需建设。重度汇总层基本上是单实体指标。
- DIM：所有实体的维度，云音乐含有不同身份的用户和多种类的内容实体，所以会有大量的维度模型
- ADS：数据集市层。基于dws和dim表，云音乐建设了大量面向应用的数据集市。

确定横向模型层的任务后，接下来要确定下纵向数据域下各业务线的建设模式。

根据经验，数仓的建设难度与业务深度和复杂度是成正比的，dws层的建设尤为复杂。dws层的设计要考虑三个方向：

1. 数据域的划分特点
2. 业务的复杂度，各业务线、业务过程之间的区别和联系，用户在各业务线的特点和关系，业务数据体量，解耦因素等等。
3. 指标的特点，指标的构成无非是退化维+时间周期+原子指标构成，所以分析下指标的退化维一般有哪些、时间周期有哪些、原子指标种类有哪些。

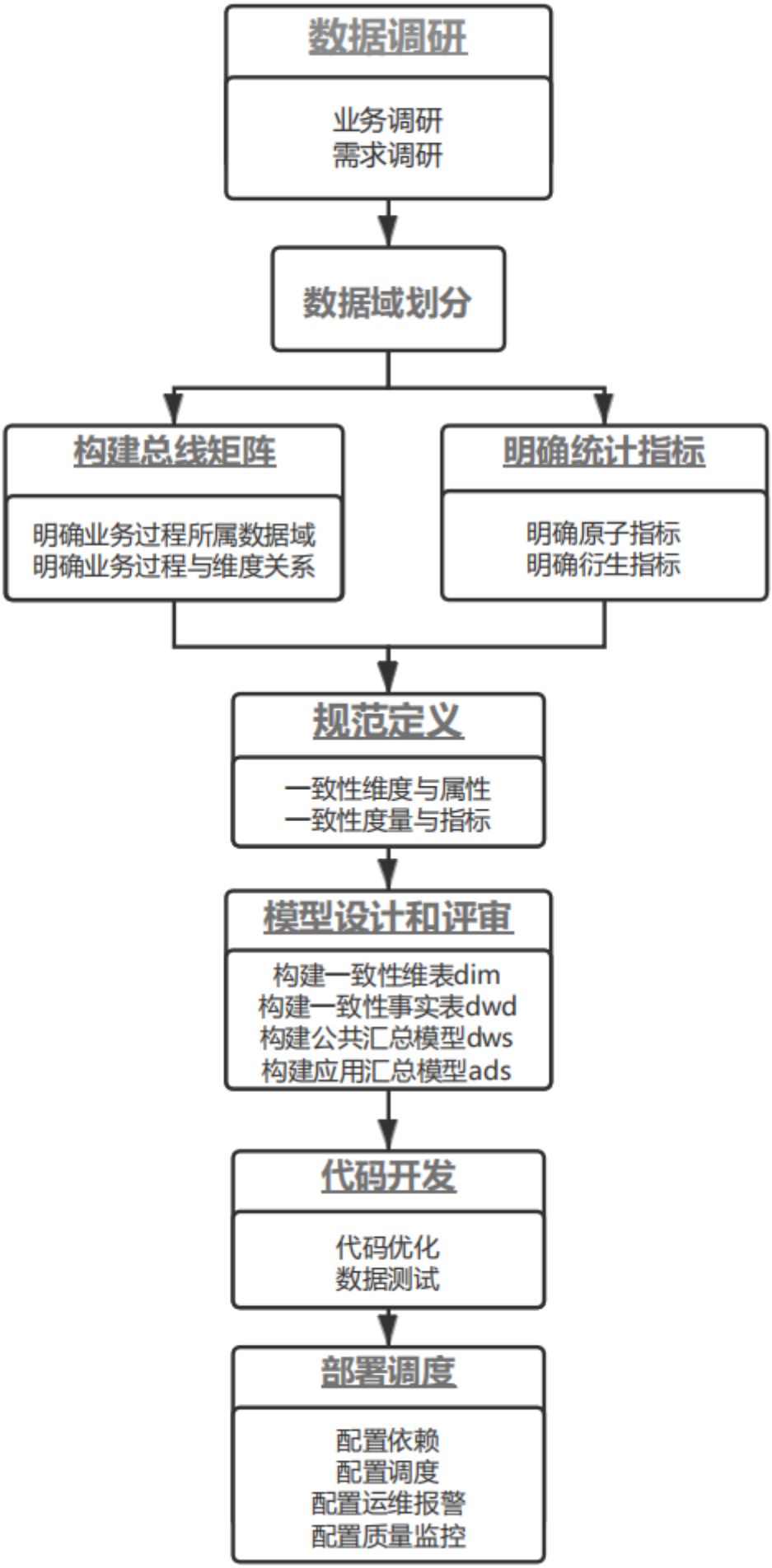


主要思路是：

1. 各内容业务线各自独立建设内容和用户类指标，按照上面架构图分数据域逐步建设
2. 轻度汇总表和中度汇总表都是用户实体+内容实体+退化维模式。轻度汇总表基本涵盖大部分需求场景
3. 重度汇总表要么是单实体，要么是用户+内容双实体组成，且不包含退化维
4. 重度汇总表分为增量表和全量表，增量包含近1/3/7/30天时间周期的指标，全量表计算历史累计指标。这么做的目的是在计算人数类指标和计算资源之间做了平衡。
5. 每个内容业务线最后都有自己内容大宽表和维表
6. 用户指标拆分到具体的内容业务线上，各自独立建设用户业务宽表，做到充分解耦，以满足各自场景需求
7. 用户维表采用先分后总模型：先建设不受业务数据干扰的公用基础维表+各业务线用户维表，强解耦，保持独立，再建设全域用户维表

三、建模流程

定下数仓的基调，即模型架构后。接下来就要开始进行具体单个模型的设计了。具体建模流程按照下面流程进行：


数据调研
业务调研
需求调研
数据域划分
构建总线矩阵
明确业务过程所属数据域
明确业务过程与维度关系
明确统计指标
明确原子指标
明确衍生指标
规范定义
一致性维度与属性
一致性度量与指标
模型设计和评审
构建一致性维表dim
构建一致性事实表dwd
构建公共汇总模型dws
构建应用汇总模型ads
代码开发
代码优化
数据测试
部署调度
配置依赖
配置调度
配置运维报警
配置质量监控

四、模型设计方法论

有了模型体系架构和建模流程，我们要实打实地设计一个个模型。我们知道表的分类主要是事实表和维表。事实表是维度建模的核心，维度是维度建模的基础和灵魂。设计模型其实就是在设计事实表和维度表。简单介绍下模型设计的基本原则和设计方法。

1.模型设计基本原则有：

1.1高内聚低耦合（最重要，非功能性需求决定架构）

- 产出时间
- 粒度
- 业务相关性
- 访问频率
- 运行方式：并行、串行
- 资源消耗情况

1.2核心模型与扩展模型分离

- 核心模型：支持常用的核心业务
- 扩展模型：支持个性化或少量应用需要
- 比如：mlog的分享引入，只有在做增长时才需要，需要拆开成立专门的分享引入模型

1.3公共处理逻辑下沉及单一

- 越是底层公用的处理逻辑越应该在数据调度依赖的底层进行封装与实现，不要让公用的处理逻辑暴露给应用层实现，不要让公共逻辑多处同时存在。

1.4成本与性能平衡

- 适当的数据冗余可换取查询和刷新性能，不宜过度冗余与数据复制。

1.5数据可回滚

- 处理逻辑不变，在不同时间多次运行数据结果确定不变。

1.6一致性

- 具有相同含义的字段在不同表中的命名必须相同，必须使用规范定义中的名称。

1.7命名清晰、可理解

- 表命名需清晰、一致，表名需易于用户理解和使用。

2.事实表设计方法

2.1选择业务过程及确定事实表类型

事实表类型：

1）单事务事实表：单个业务过程

2）多事务事实表：多个业务过程放在一个事实表中

①原则：

- 业务过程是否有相似性？
- 是否来源同一业务源系统？
- 用户使用的学习成本哪种更低？
- 计算和存储成本是否会降低？

②方式

- 不同的业务过程使用不同的事实字段进行存放
- 不同业务过程的事实使用同一个事实字段进行存放，但增加一个业务过程标签

2.2声明粒度（**非常重要**）

粒度传递的是与事实表度量有关的细节层次，精确定义事实表每一行所代表的业务含义

尽量选择最细级别的原子粒度，以确保事实表的应用具有最大的灵活性。

2.3确定维度：应该选择能够描述清楚业务过程所处的环境的维度信息。

2.4确定事实：

- 应该选择与业务过程有关的所有事实
- 事实的粒度要与所声明的事实表的粒度一致

2.5冗余维度：

冗余下游用户方便使用的常用维度：常用的用于统计的维度属性

冗余原则

不冗余查询速度是不是很慢

存储资源是不是够

查询频率是不是很多

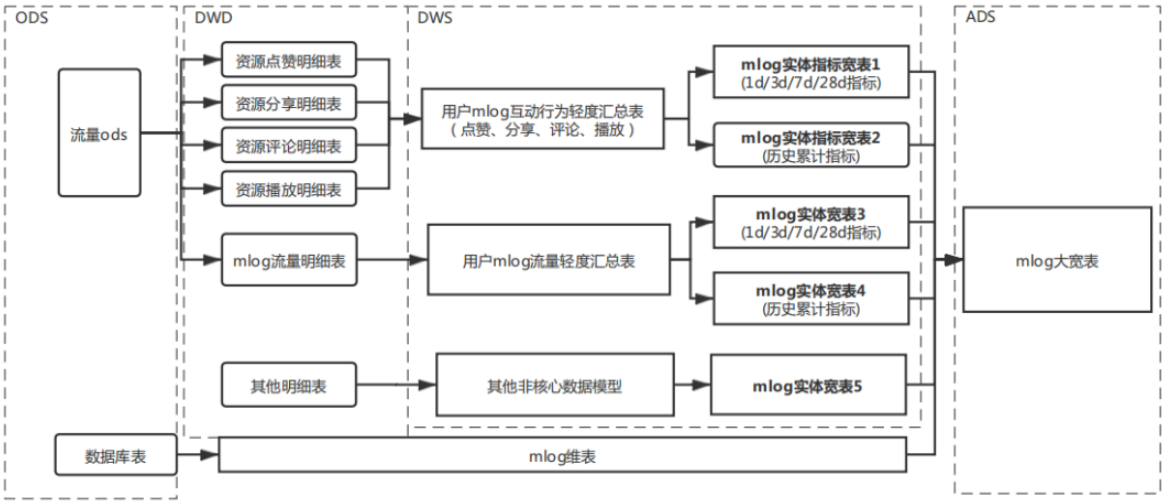
冗余在哪一模型层的事实表中：明细层、轻度汇总层?

3.维度设计的基本方法

- 选择维度和新建维度
- 确定主维表
- 确定相关维表
- 确定维度属性：从主维表和相关维表中抽取
- 尽可能生成丰富的维度属性
- 尽可能多地给出包括一些富有意义的文字性描述，例如名称等
- 区分数值型属性和事实：看用途
- 尽量沉淀出通用的维度属性
- 维度设计遵循的通用原则：反范式、扁平化

五、实例介绍

根据前面介绍的方法论，云音乐社区mlog内容线的部分模型架构如下：



- 维表：一张mlog主维表
- mlog互动行为轻度汇总表：mlog的互动行为合并，因为具有相同的环境信息
- 流量轻度汇总表：曝光点击浏览合并，环境信息相似
- 历史累计指标单独：运行方式解耦
- 1d/3d/7d/28d指标：合并，运行时间不影响，不解耦

六、总结

上述提到的模型分层、建模流程和建模方法论，在团队内部形成了较为稳定的规范，并在团队内部得到大力推广。好的规范离不开产品功能的加持，网易有数大数据平台的模型设计中心基于网易内部多个团队的数仓建设经验，将数仓建模实践经验和方法论进行了产品化，为云音乐数仓团队落地内部规范、完成数仓建模方面提供了极大的帮助。