1. 数据层次的划分

- ODS：Operational Data Store，操作数据层，在结构上其与源系统的增量或者全量数据基本保持一致。

  它相当于一个数据准备区，同时又承担着基础数据的记录以及历史变化。其主要作用是把基础数据引入到MaxCompute。

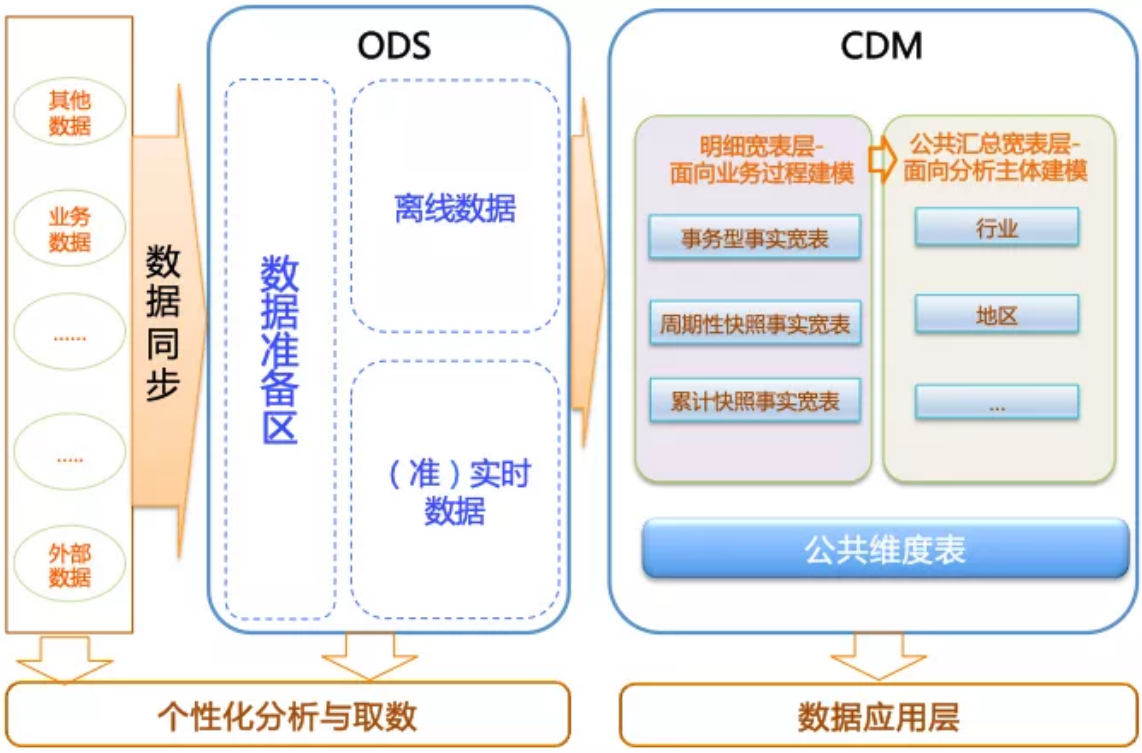
- CDM：Common Data Model，公共维度模型层，又细分为DWD和DWS。

  它的主要作用是完成数据加工与整合、建立一致性的维度、构建可复用的面向分析和统计的明细事实表以及汇总公共粒度的指标。

-
  - DWD：Data Warehouse Detail，明细数据层。
  - DWS：Data Warehouse Summary，汇总数据层。
- ADS：Application Data Service，应用数据层。

具体仓库的分层情况需要结合业务场景、数据场景、系统场景进行综合考虑。

1.1. 数据分类架构



该数据分类架构在ODS层分为三部分：数据准备区、离线数据和准实时数据区。在进入到CDM层后，由以下几部分组成：

- 公共维度层：

  基于维度建模理念思想，建立整个企业的一致性维度。

- 明细粒度事实层：

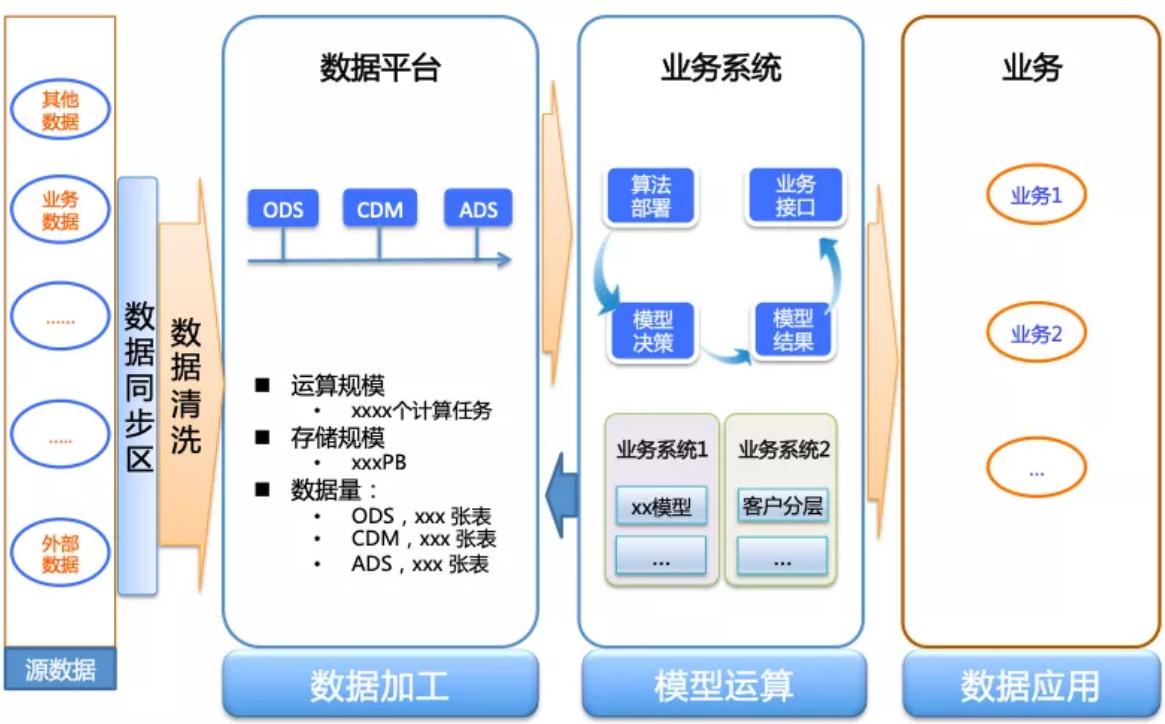
  以业务过程为建模驱动，基于每个具体业务过程的特点，构建最细粒度的明细层事实表。

  您可以结合企业的数据使用特点，将明细事实表的某些重要维度属性字段做适当的冗余，即宽表化处理。

- 公共汇总粒度事实层：

  以分析的主题对象为建模驱动，基于上层的应用和产品的指标需求，构建公共粒度的汇总指标事实表，以宽表化手段来物理化模型。

1.2. 数据处理流程架构



1.3. 数据划分及命名空间约定

请根据业务划分数据并约定命名，建议针对业务名称结合数据层次约定相关命名的英文缩写，这样可以给后续数据开发过程中，对项目空间、表、字段等命名做为重要参照。

- 按业务划分：

  命名时按主要的业务划分，以指导物理模型的划分原则、命名原则及使用的ODS project。

  例如，按业务定义英文缩写，阿里的“淘宝”英文缩写可以定义为“tb”。

- 按数据域划分：

  命名时按照CDM层的数据进行数据域划分，以便有效地对数据进行管理，以及指导数据表的命名。

  例如，“交易”数据的英文缩写可定义为“trd”。

- 按业务过程划分：

  当一个数据域由多个业务过程组成时，命名时可以按业务流程划分。

  业务过程是从数据分析角度看客观存在的或者抽象的业务行为动作。

  例如，交易数据域中的“退款”这个业务过程的英文缩写可约定命名为“rfd_ent”。

1.4. 数据模型

模型是对现实事物的反映和抽象，能帮助我们更好地了解客观世界。数据模型定义了数据之间关系和结构，使得我们可以有规律地获取想要的数据。例如，在一个超市里，商品的布局都有特定的规范，商品摆放的位置是按照消费者的购买习惯以及人流走向进行摆放的。

- 数据模型的作用

  数据模型是在业务需求分析之后，数据仓库工作开始时的第一步。良好的数据模型可以帮助我们更好地存储数据，更有效率地获取数据，保证数据间的一致性。

- 模型设计的基本原则

- 高内聚和低耦合

  一个逻辑和物理模型由哪些记录和字段组成，应该遵循最基本的软件设计方法论中的高内聚和低耦合原则。主要从数据业务特性和访问特性两个角度来考虑：将业务相近或者相关的数据、粒度相同数据设计为一个逻辑或者物理模型；将高概率同时访问的数据放一起，将低概率同时访问的数据分开存储。

- 核心模型与扩展模型分离

  建立核心模型与扩展模型体系，核心模型包括的字段支持常用核心的业务，扩展模型包括的字段支持个性化或是少量应用的需要。在必须让核心模型与扩展模型做关联时，不能让扩展字段过度侵入核心模型，以免破坏了核心模型的架构简洁性与可维护性。

- 公共处理逻辑下沉及单一

  底层公用的处理逻辑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，不要让公共逻辑在多处同时存在。

- 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。

- 数据可回滚

  处理逻辑不变，在不同时间多次运行数据的结果需确定不变。

- 一致性

  相同的字段在不同表中的字段名必须相同。

- 命名清晰可理解

  表命名规范需清晰、一致，表命名需易于下游的理解和使用。

2. 补充说明

- 一个模型无法满足所有的需求。
- 需合理选择数据模型的建模方式。
- 通常，设计顺序依次为：概念模型->逻辑模型->物理模型。

3. 层级调用

应用层应优先调用公共层数据，必须存在中间层数据，不允许应用层跨过中间层从ODS层重复加工数据。一方面，中间层团队应该积极了解应用层数据的建设需求，将公用的数据沉淀到公共层，为其他团队提供数据服务；另一方面，应用层团队也应积极配合中间层团队进行持续的数据公共建设的改造。必须避免出现过度的引用ODS层、不合理的数据复制以及子集合冗余。

- ODS层数据不能被应用层任务引用，中间层不能有沉淀的ODS层数据，必须通过CDM层的视图访问。

  CDM层视图必须使用调度程序进行封装，保持视图的可维护性与可管理性。

- CDM层任务的深度不宜过大（建议不超过10层）。
- 原则上一个计算刷新任务只允许一个输出表。
- 如果多个任务刷新输出一个表（不同任务插入不同的分区），DataWorks上需要建立一个依赖多个刷新任务的虚拟任务，通常下游应该依赖此虚拟任务。
- CDM汇总层应优先调用CDM明细层。

在调用可累加类指标计算时，CDM汇总层尽量优先调用已经产出的粗粒度汇总层，以避免大量汇总直接从海量的明细数据层计算。

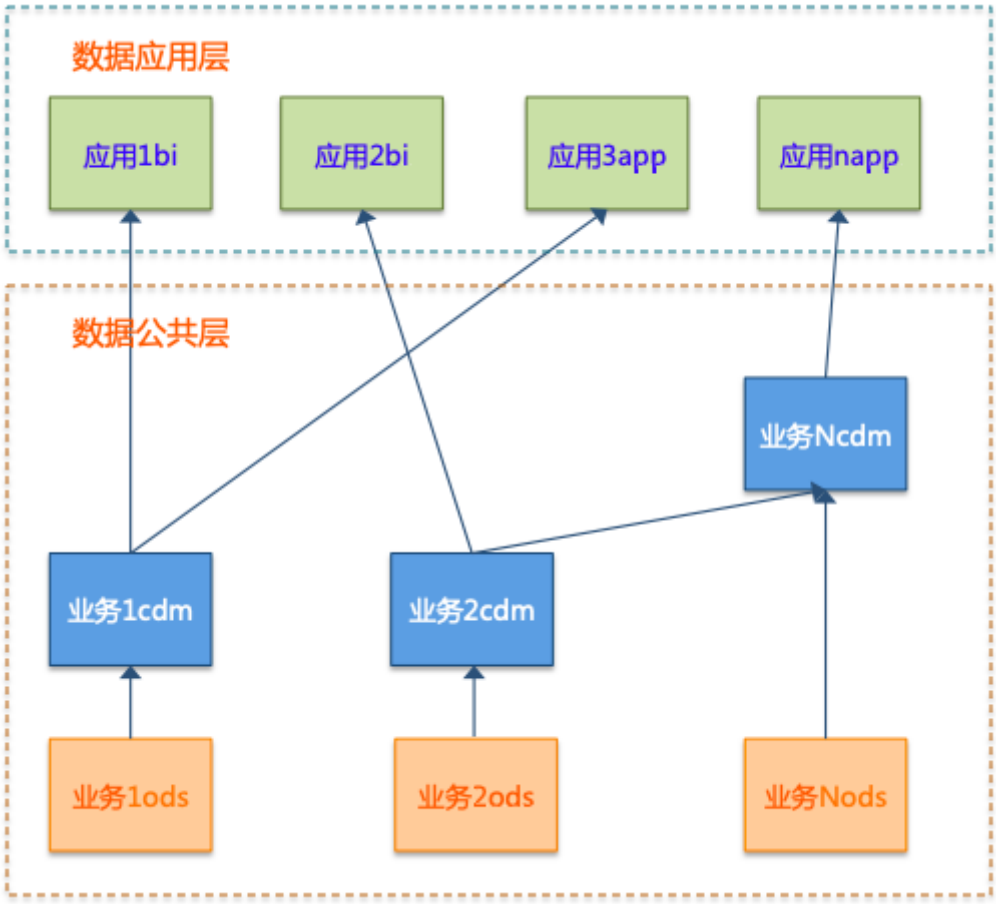
- CDM明细层累计快照事实表优先调用CDM事务型事实表，以保持数据的一致性产出。
- 避免应用层过度引用和依赖CDM层明细数据，需要针对性地建设好CDM公共汇总层。

4. 项目分配

按实际需求分配不同的ODS和CDM项目。一个ODS层项目对应一个CDM项目。例如：

- ODS层项目，按业务部门的粒度建立。
- CDM层项目，按业务部门的粒度建立。
- ADS层项目，按应用的粒度建立。

一个项目的划分结构如下图所示。



4.1. 命名规范

- ODS层项目名称以ods为后缀，例如tbods。
- 中间层项目名称以cdm为后缀，例如tbcdm。
- 应用层项目中，数据报表、数据分析等应用名称以bi为后缀，例如tbbi；

  而数据产品等应用名称以app为后缀，例如sycmapp。

4.2. 数据类型规范

ODS层的数据类型应基于源系统数据类型转换。例如，源数据为MySQL时的转换规则如下。

MySQL数据类型	MaxCompute数据类型
TINYINT	TINYINT
SMALLINT/MEDIUMINT	SMALLINT
INTEGER	INT
BIGINT	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE
DECIMAL	DECIMAL
CHAR/VARCHAR	VARCHAR
LONGTEXT/TEXT	STRING
DATE/TIMESTAMP/TIME/YEAR	STRING
DATETIME	DATETIME

CDM数据公共层如果是引用ODS层数据，则默认使用ODS层字段的数据类型。其衍生加工数据字段按以下标准执行：

- 金额类及其它小数点数据使用DOUBLE类型。
- 字符类数据使用STRING类型。
- ID类和整形数值使用BIGINT类型。
- 时间类型数据使用STRING类型（如果有特殊的格式要求，可以选择性使用DATETIME类型）。
- 状态使用STRING类型。

4.3. 公共字段定义规范

- 数据统计日期的分区字段按以下标准：
- - 按天分区：

    ds(YYYYMMDD)。
  - 按小时分区：

    hh(00-23)。
  - 按分钟：

    mi (00-59)。
- is_{业务}：

  表示布尔型数据字段。

  以Y和N表示，不允许出现空值域。
- 原则上不需要冗余分区字段。

4.4. 数据冗余

一个表做宽表冗余维度属性时，应该遵循以下建议准则：

- 冗余字段与表中其它字段高频率（大于3个下游应用SQL）同时访问。
- 冗余字段的引入不应造成其本身的刷新完成时间产生过多后延。
- 公共层数据不允许字段重复率大于60%的相同粒度数据表冗余，可以选择在原表基础上拓宽或者在下游应用中通过JOIN方式实现。

4.5. 数据拆分

数据的水平和垂直拆分是按照访问热度分布和数据表非空数据值、零数据值在行列二维空间上分布情况进行划分的。

- 在物理上划分核心模型和扩展模型，将其字段进行垂直划分。
- 将访问相关度较高的列在一个表存储，将访问相关度较低的字段分开存储。
- 将经常用到的Where条件按记录行进行水平切分或者冗余。

  水平切分可以考虑二级分区手段，以避免多余的数据复制与冗余。
- 将出现大量空值和零值的统计汇总表，依据其空值和零值分布状况可以做适当的水平和垂直切分，以减少存储和下游的扫描数据量。

4.6. 空值处理原则

- 汇总类指标的空值：

  空值处理，填充为零，当前MaxCompute基于列存储的压缩技术不会由于填充大量空值导致存储成本上升。
- 维度属性值为空：

  在汇总到对应维度上时，对于无法对应的统计事实，记录行会填充为-99（未知），对应维表会出现一条-99（未知）的记录。