学掌门
atstudy.com

DESIGNED BY IBOTU

数据仓库理论基础与企业应用场景

数据仓库与维度建模

学习掌门人 职场引路人

学掌门
atstudy.com

Contents
//目录

01 认识数据仓库

02 数据仓库理论基础

03 实体关系（ER）建模理论及应用场景案例

04 数据仓库与维度建模

05 实战案例-偏业务型行业数据仓库设计

学习掌门人 职场引路人

数据仓库与维度建模

PART 01
维度建模概述

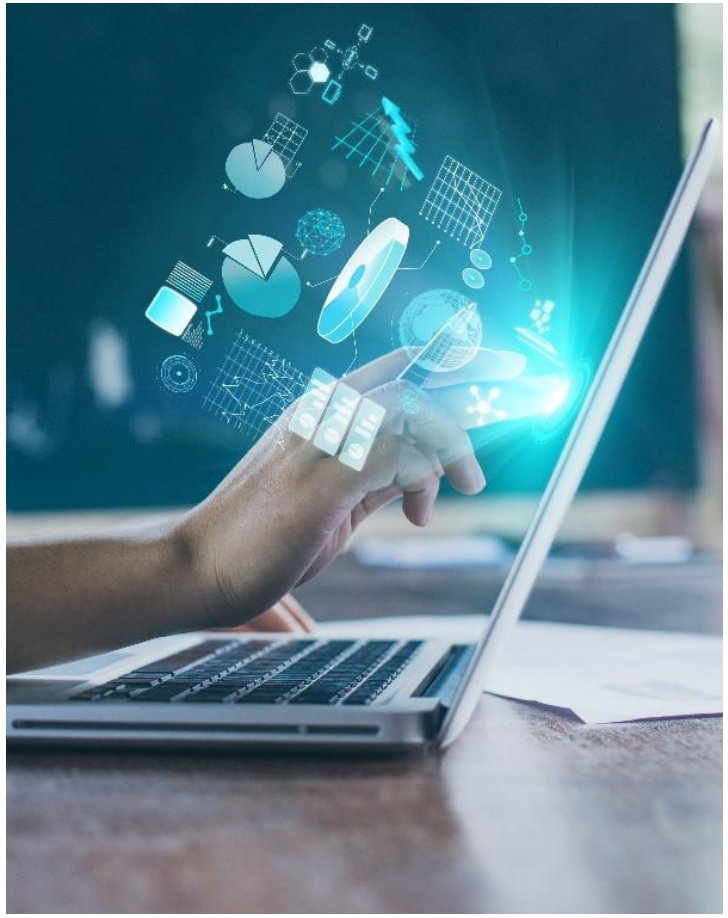
PART 02
维度建模：事实表

PART 03
维度建模：维度表

PART 04
维度建模的三种模式

PART 05
Data Value建模与Anchor建模

维度建模：是数据仓库建设中的一种数据建模方法，Kimball 最先提出这一概念。其最简单的描述就是，按照事实表，维表来构建数据仓库，数据集市，这种方法最被人广泛知晓的名字就是星型模式。

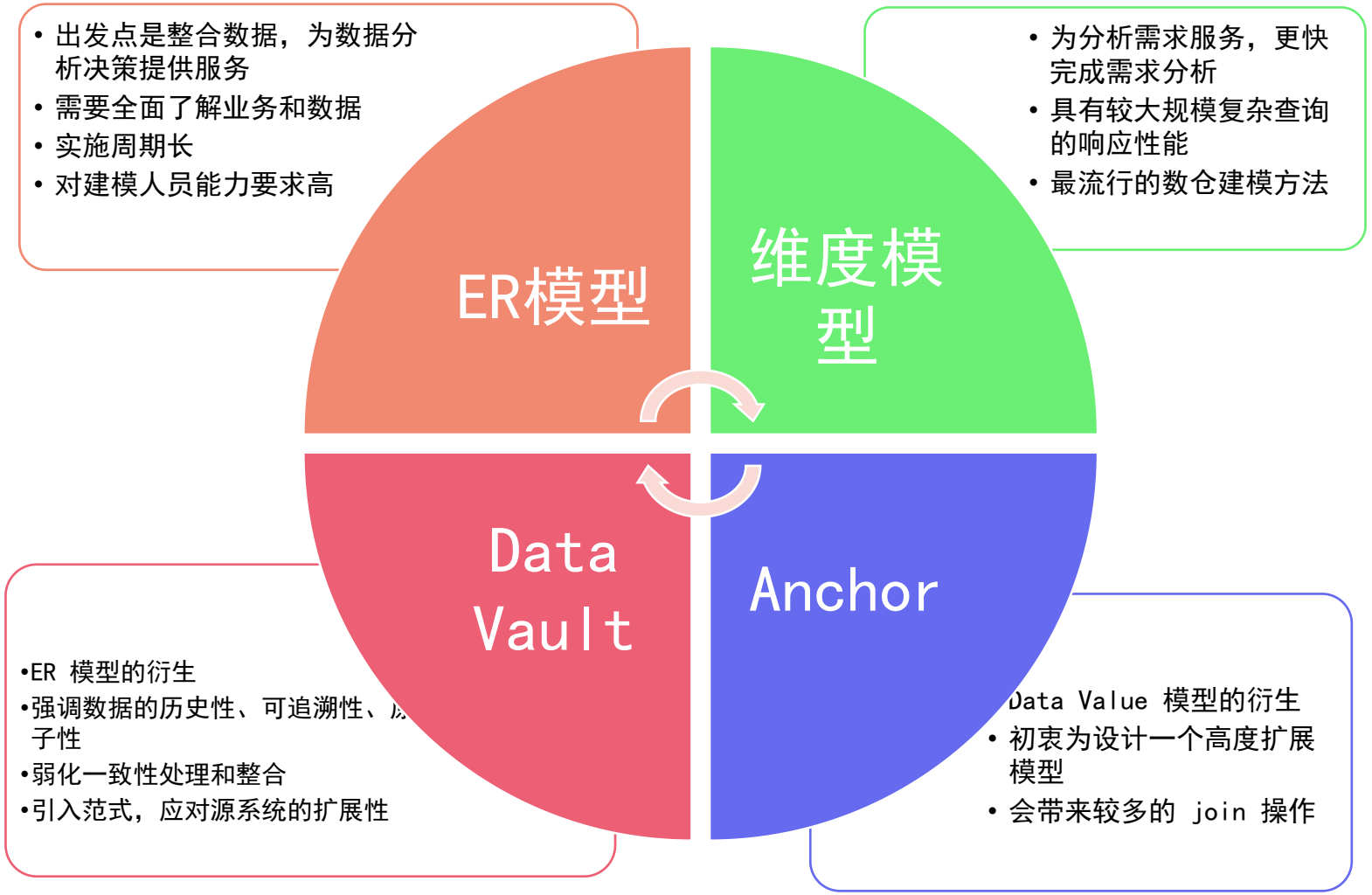


Part-01：维度建模概述

- 什么是维度建模？
- 维度建模中表的二种类型
- 维度建模中的三种模式

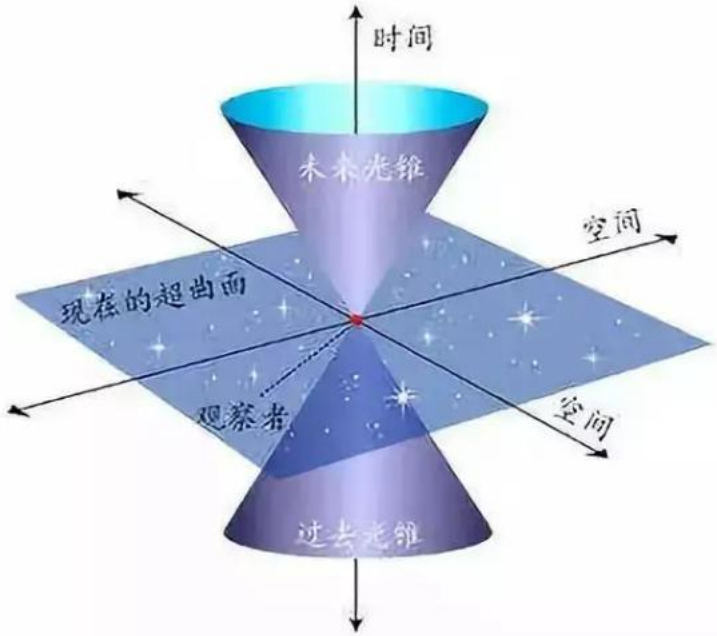
数仓建模方法

- 数仓建模方法：
  - 数据仓库建模有如下四种：ER 模型、维度模型、Data Vault、Anchor。
  - 重点掌握的是 ER模型与 **维度模型**。



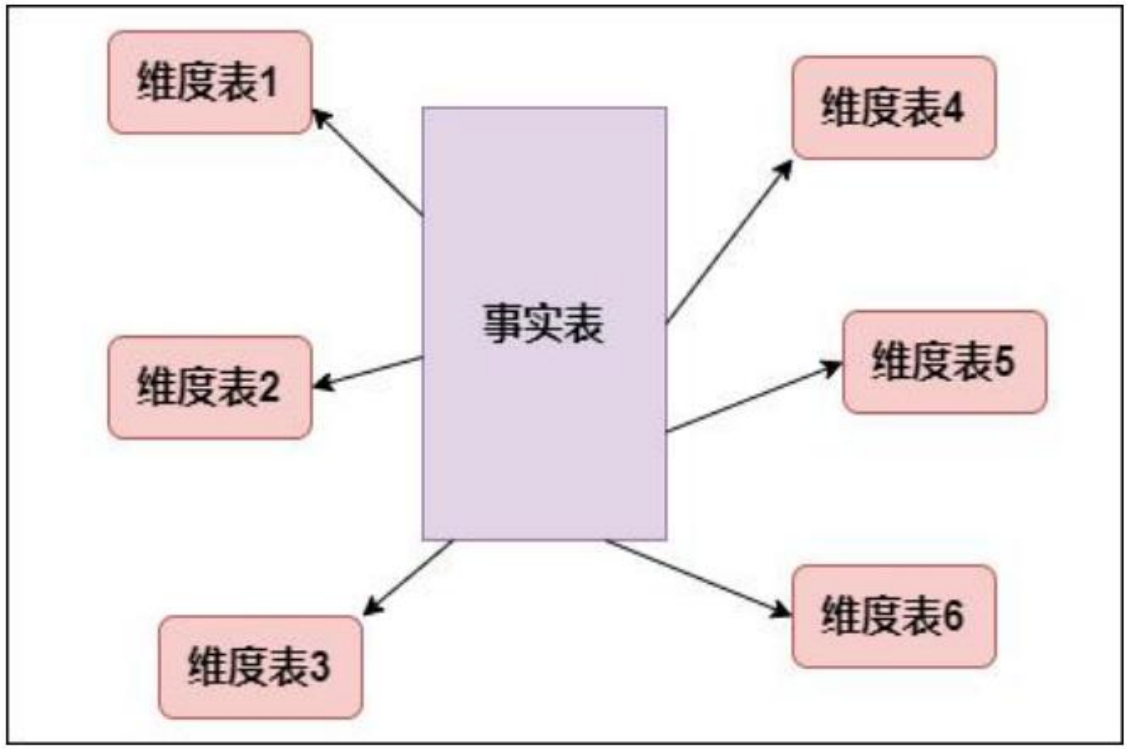
维度建模概述

- 什么是维度建模
  - 维度模型是数据仓库领域另一位大师 **Ralph Kimall** 所倡导，他的《数据仓库工具箱》是数据仓库工程领域最流行的数仓建模经典。维度建模以 **分析决策的需求** 出发构建模型，构建的数据模型为分析需求服务，因此它重点解决**用户如何更快速完成分析需求**，同时还有较好的**大规模复杂查询**的**响应性能**。



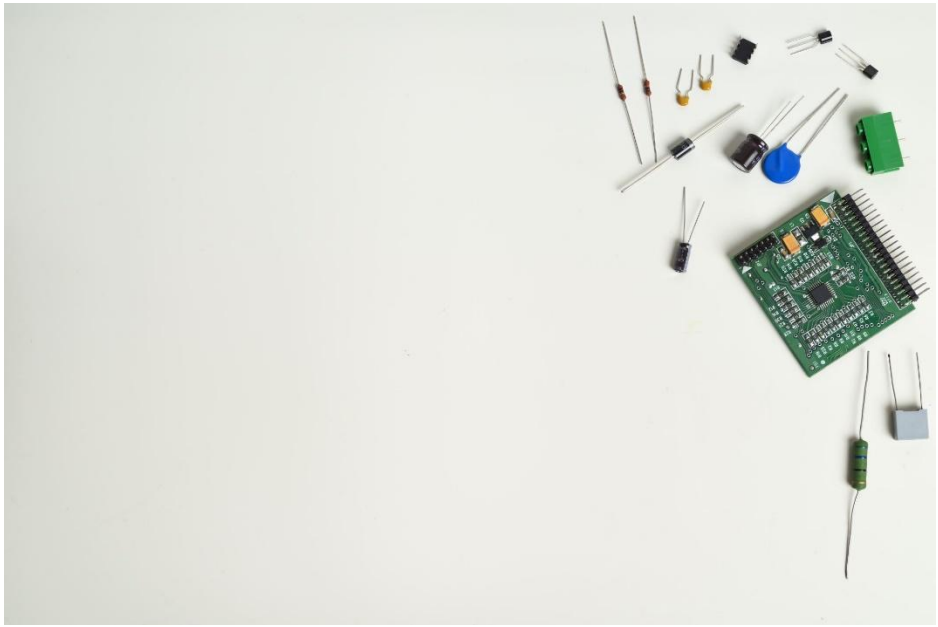
维度建模概述

- 什么是维度建模
  - 典型的代表是我们比较熟知的 **星形模型**（Star-schema），以及在一些特殊场景 下适用的 **雪花模型**（Snow-schema）。
  - 维度建模中比较重要的概念就是 **事实表**（Fact table）和 **维度表**（Dimension table）。其最简单的描述就是，按照事实表、维度表来构建数据仓库、数据集市。



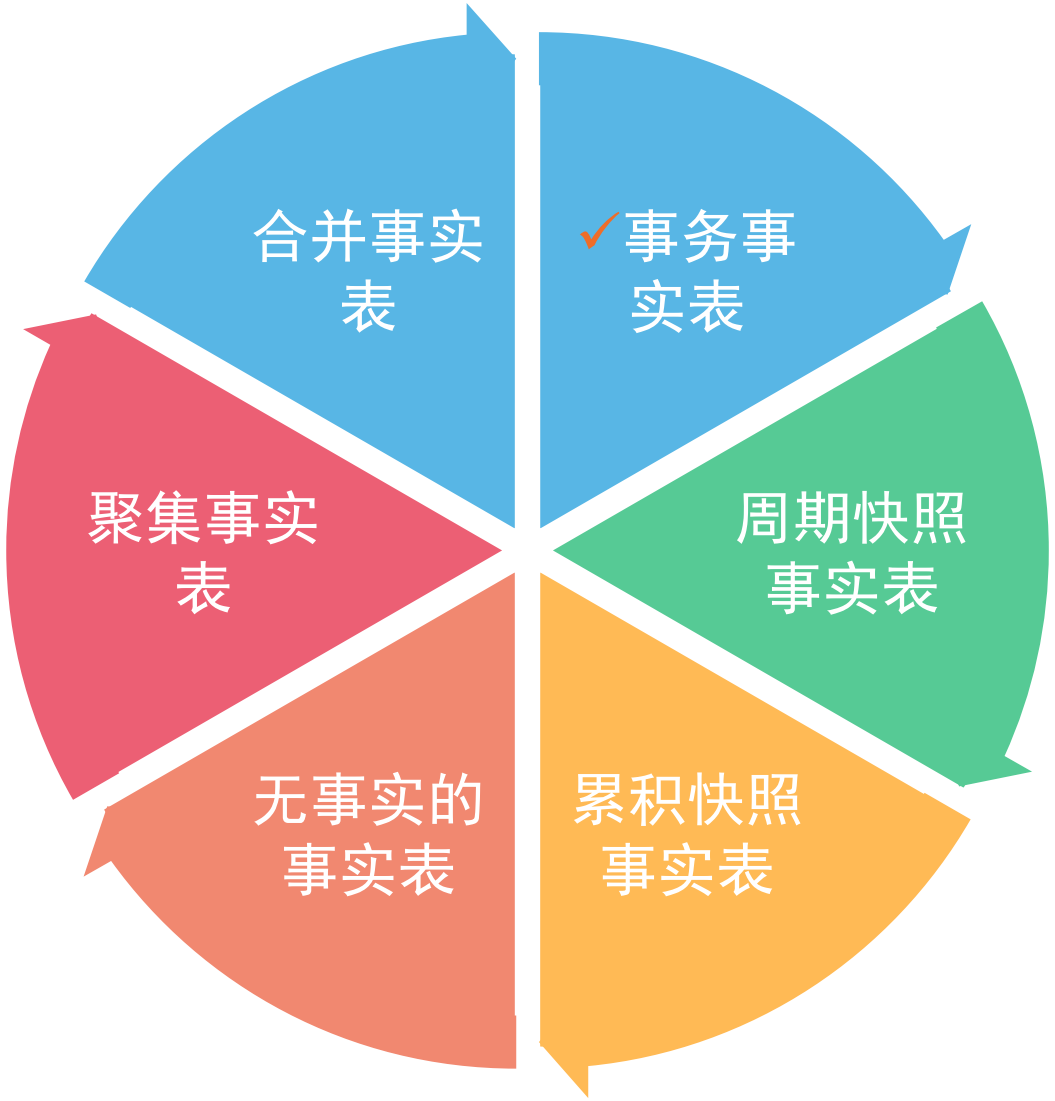
维度建模概述

- 维度建模中表的类型
  - **事实表：是一堆主键的集合，每个主键对应维度表中的一条记录，客观存在** 的，根据主题确定出需要使用的数据。必然存在的一些数据，像采集的**日志文件，订单表**，都可以作为 事实表 。
  - **维度表：维度就是所分析的数据的一个量**，维度表就是以合适的角度来创 建的表，分析问题的一个角度：**时间、地域、终端、用户**等角度。



维度建模概述

- 事实表
  - 事实表分为以下 6 类：
    - ✓ 1. 事务事实表
    - ✓ 2. 周期快照事实表
    - ✓ 3. 累积快照事实表
    - ✓ 4. 无事实的事实表
    - ✓ 5. 聚集事实表
    - ✓ 6. 合并事实表



维度建模概述

- 维度建模的模式
  - 维度建模一般会分为如下三种模式：
    - ✓ 星型模式
    - ✓ 雪花模式
    - ✓ 星座模式

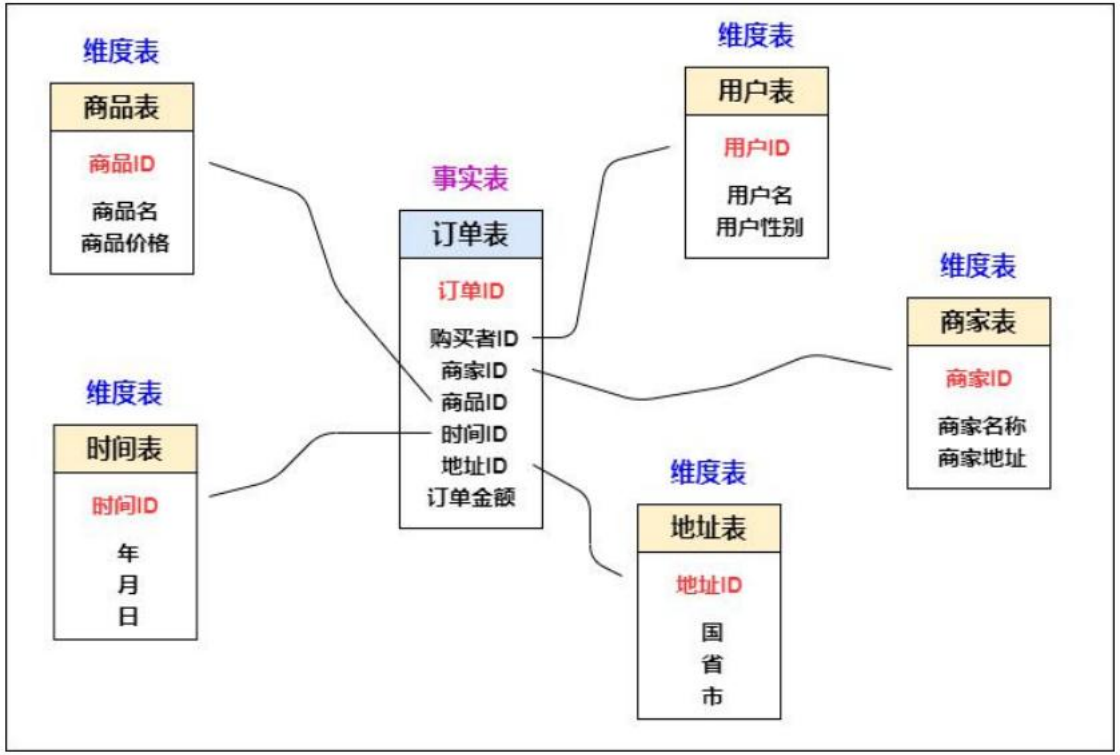
星型模式	雪花模式	星座模式
• 星形模式(Star Schema)是最常用的维度建模方式。 星型模式是以事实表为中心	• 雪花模式(Snowflake Schema)是对星形模式的扩展。雪花模式的维度表可以拥有其他维度表	• 星座模式是星型模式延伸而来，星型模式是基于一张事实表的，而 星座模式是基于多张事实表的 ，而且共享维度信息 。

Part-02：维度建模 —— 事实表

- 事实表的定义
- 事务事实表
- 周期快照事实表
- 累积快照事实表
- 无事实的事实表
- 聚集事实表
- 合并事实表

维度建模 —— 事实表

- 事实表
  - **发生在现实世界中的操作型事件，其所产生的可度量数值，存储在事实表中**。从 最低的粒度级别来看，**事实表行对应一个度量事件**，反之亦然。 事实表表示对分析主题的度量。**比如一次购买行为**我们就可以理解为是一个事实。



事实与维度

维度建模 —— 事实表

- 事实表
  - 明 细 表 （ 宽 表 ）
    - ✓ 事实表的数据中，有些属性共同组成了一个字段（糅合在一起），比如年月日时 分秒构成了时间，当需要根据某一属性进行分组统计的时候，需要截取拼接之类 的操作，效率极低。如：

local_time
2021-03-18 06:31:42

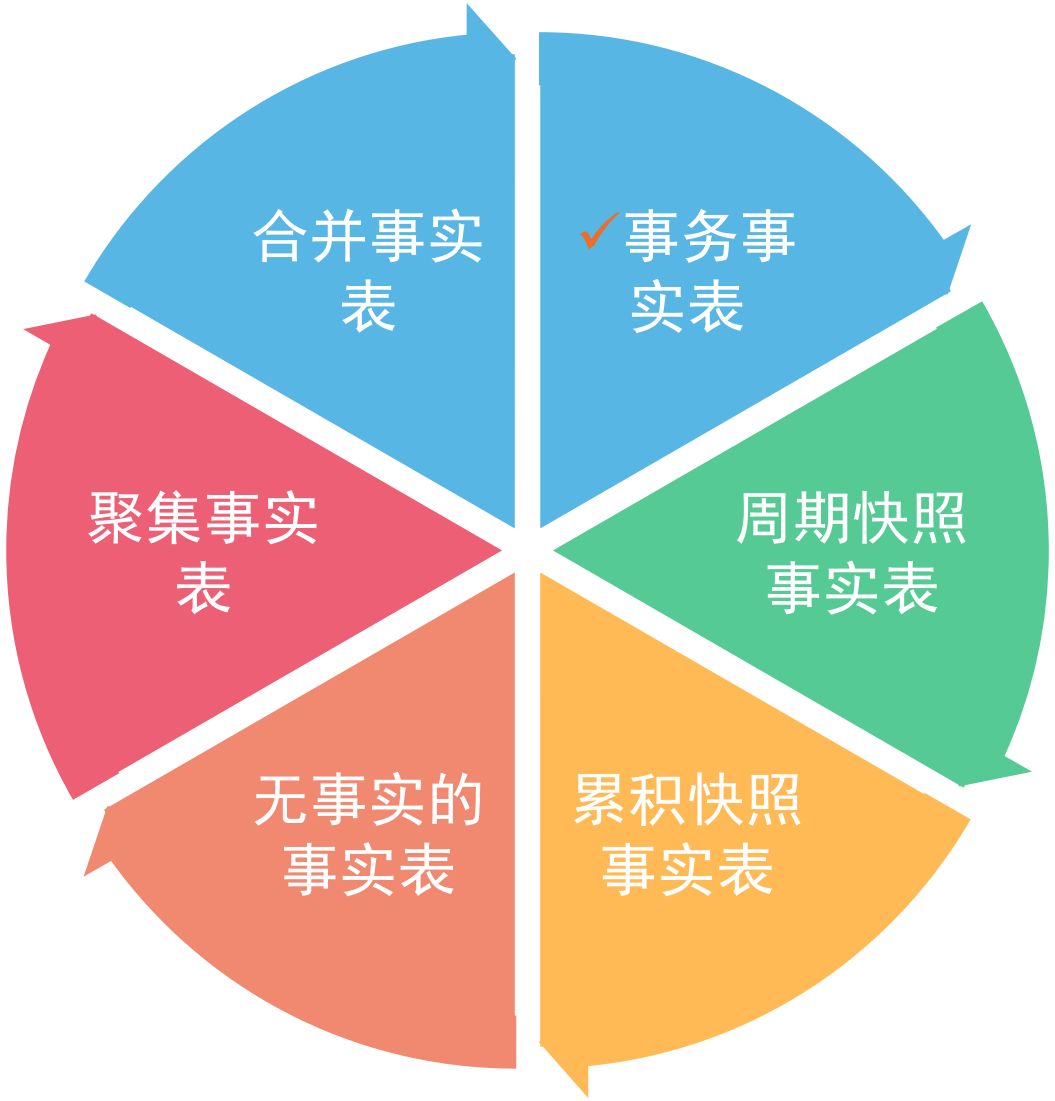
    - ✓ 为了分析方便，可以事实表中的一个字段切割提取多个属性出来构成新的字段，因 为字段变多了，所以称为宽表 段扩展为如下 6 个字段：

year	month	day	hour	m	s
2021	03	18	06	31	42

又因为宽表的信息更加清晰明细，所以也可以称之为明细表 。

维度建模 —— 事实表

- 事实表
  - 事实表分为以下 6 类：
    - ✓ 1. 事务事实表
    - ✓ 2. 周期快照事实表
    - ✓ 3. 累积快照事实表
    - ✓ 4. 无事实的事实表
    - ✓ 5. 聚集事实表
    - ✓ 6. 合并事实表



维度建模 —— 事实表

- 事实表
  - 一、事务事实表：
    - ✓ 表中的一行对应空间或时间上某点的度量事件。就是一行数据中必须有度量字段。
    - ✓ 什么是度量，就是指标，比如说销售金额，销售数量等这些可加的或者半可加就是度量值。
    - ✓ 事务事实表都包含一个与维度表关联的外键。并且度量值必须和事务粒度保持一致。

业务日期	订单ID	父订单ID	支付时间	买家ID	卖家ID	商品ID	支付度量
20160101	order1	mord1	2016-01-01 08:00:00	111	222	aa	...
20160102	order3	mord2	2016-01-01 09:00:00	333	444	bb	...
20160102	order4	mord2	2016-01-01 09:00:00	333	444	cc	...

交易订单支付事务事实表

维度建模 —— 事实表

- 事实表
  - 二、周期快照事实表：
    - ✓ 周期事实表就是每行都带有时间值字段，代表周期。
    - ✓ 通常时间值都是标准周期，如某一天，某周，某月等。
    - ✓ 粒度是周期，而不是个体的事务。
    - ✓ 一个周期快照事实表中数据可以是多个事实，但是它们都属于某个周期内。

维度建模 —— 事实表

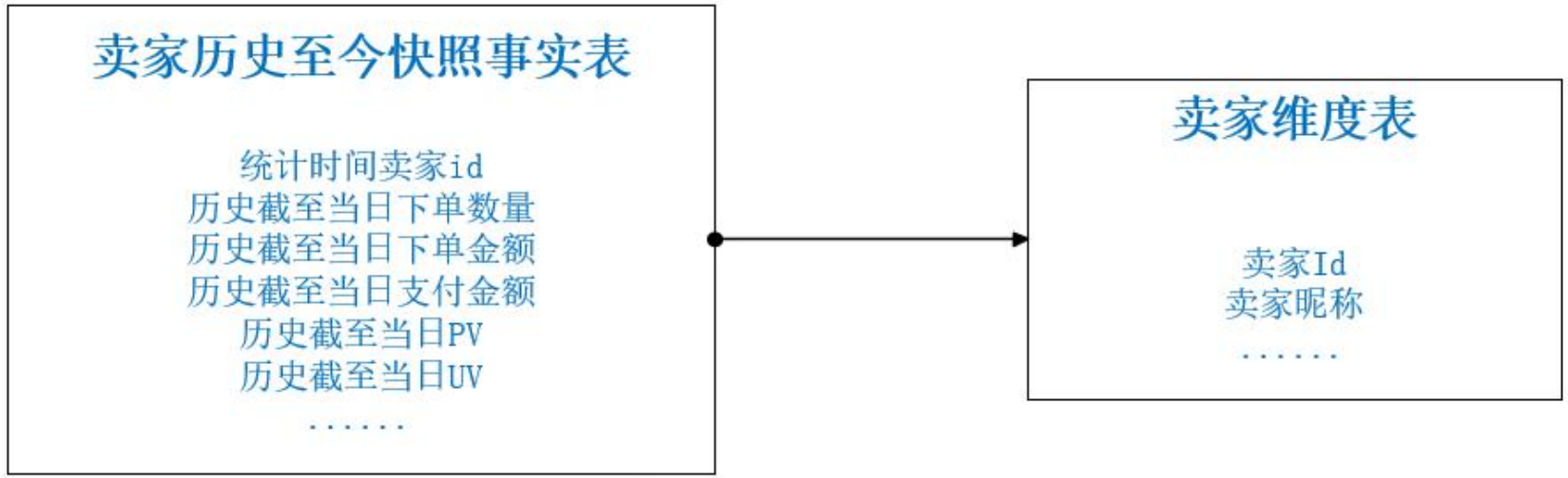
- 事实表
  - 周期快照事实表（特征）：
    - ✓ 1）统计的是间隔周期内的度量统计，如历史至今、自然年至今、季度至今等等
    - ✓ 2）周期快照表没有粒度的概念，取而代之的是周期+状态度量的组合，如历史至今的订单总数
    - ✓ 3）事实事务表是稀疏表，周期快照表是稠密表
      - 稀疏表：当天只有发生了操作才会有记录
      - 稠密表：当天没有操作也会有记录，便于下游使用

维度建模 —— 事实表

- 事实表
  - 周期快照事实表的分类：
    - ✓ 1）单维度的周期快照事实表
    - ✓ 2）混合维度的周期快照事实表
    - ✓ 3）全量快照事实表

维度建模 —— 事实表

- 事实表
  - 1）**单维度的周期快照事实表实例：**



卖家每日快照事实表

维度建模 —— 事实表

- 事实表
  - **2）混合维度的周期快照事实表实例：**反映不同买家对于不同买家的统计信息

买家每日快照事实表

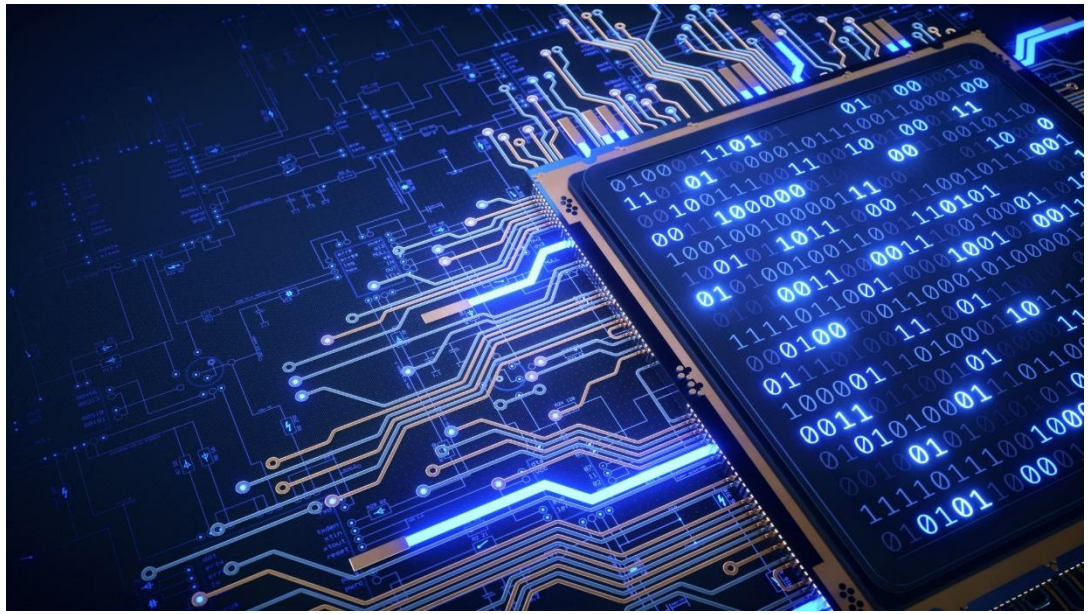
维度建模 —— 事实表

- 事实表
  - **3）全量快照事实表实例**：对于状态一直变化的数据，用全量快照表统计至今最新的状态，如订单评价，好中差评会每天变化，事实表的粒度确定为每一条评价，加之冗余常用维表属性。

全量快照事实表

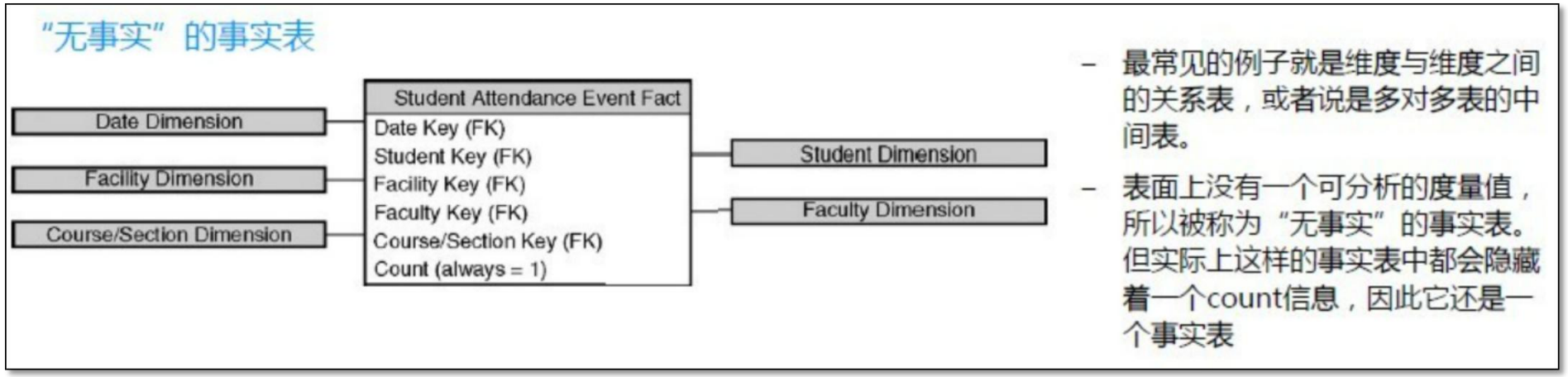
维度建模 —— 事实表

- 事实表
  - 三、累积快照事实表
    - ✓ 周期快照事实表是单个周期内数据，而累计快照事实表是由多个周期数据组成，
    - ✓ 每行汇总了过程开始到结束之间的度量。每行数据相当于管道或工作流，有事件的起点，过程，终点，并且每个关键步骤都包含日期字段。
    - ✓ 如订单数据，累计快照事实表的一行就是一个订单，当订单产生时插入一行，当订单发生变化时，这行就被修改。



维度建模 —— 事实表

- 事实表
  - 四、无事实的事实表
    - ✓ 实际应用中绝大多数都是数字化的度量，但是也可能会有少量的没有数字化的值但是还很有价值的字段，无事实的事实表就是为这种数据准备的，利用这种事实表可以分析发生了什么。

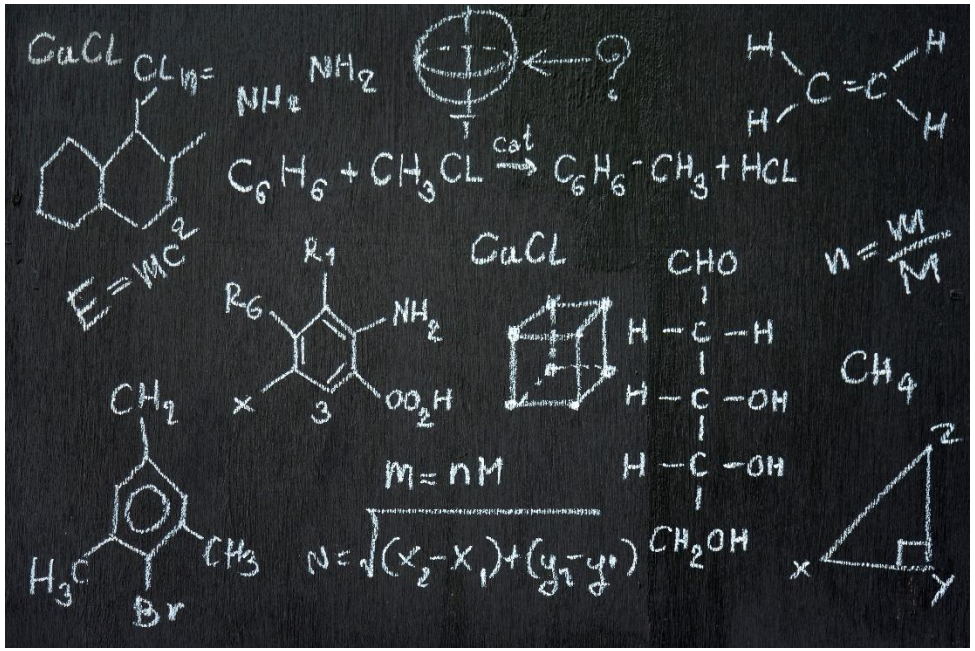


维度建模 —— 事实表

- 事实表
  - 四、无事实的事实表
    - ✓ 在应用中，任何设计方法都是有缘由的，“无事实”的事实表由KIMBALL提出，原文是Factless FACT，在很多项目中也叫Dimensional Fact table。
    - ✓ 说直白点，就是讲述不同维度之间的对应关系，帮助业务模型落地到数据模型过程中，更好地梳理维度之间的对应关系，并且能更快获得关系数据。
    - ✓ 另外它也不仅仅是桥梁作用，还是更快捷获得关联信息的工具，如果你只想获得关联信息，就不需要去事实表中找了。事实表中，各个维表的组成，结果就产生一个粒度，而这个事实表，他的组成不是为了说明某个粒度的事实，它仅仅是为了让你更快捷获得维度之间的关系和快捷获得关联信息。

维度建模 —— 事实表

- 事实表
  - 五、聚集事实表
    - ✓ 聚集，就是对原子粒度的数据进行简单的聚合操作，**目的就是为了提高查询性能**。
    - ✓ 如：我们需求是查询全国所有门店的总销售额，我们原子粒度的事实表中每行是每个分店每个商品的销售额，聚集事实表就可以先聚合每个分店的总销售额，这样汇总所有门店的销售额时计算的数据量就会小很多。



维度建模 —— 事实表

- 事实表
  - 六、合并事实表
    - ✓ （1）来自多个过程，以相同粒度表示的事实合并为一个单一的合并事实表。
    - ✓ （2）合并事实表会增加ETL处理过程的负担，但是降低了BI应用的分析代价。
    - ✓ （3）合并事实表适合经常需要共同分析的多过程度量

维度建模 —— 事实表

- 事实表
  - 六、合并事实表
    - ✓ 合并事实表与聚合事实表的区别与联系
      - 聚集事实表和合并事实表的主要差别是合并事实表一般是从多个事实表合并而来。
      - 但是它们的差别不是绝对的，一个事实表既是聚集事实表又是合并事实表是很有可能的。
      - 因为一般合并事实表需要按相同的维度合并，所以很可能在做合并的同时需要进行聚集，即粒度变粗。

小结

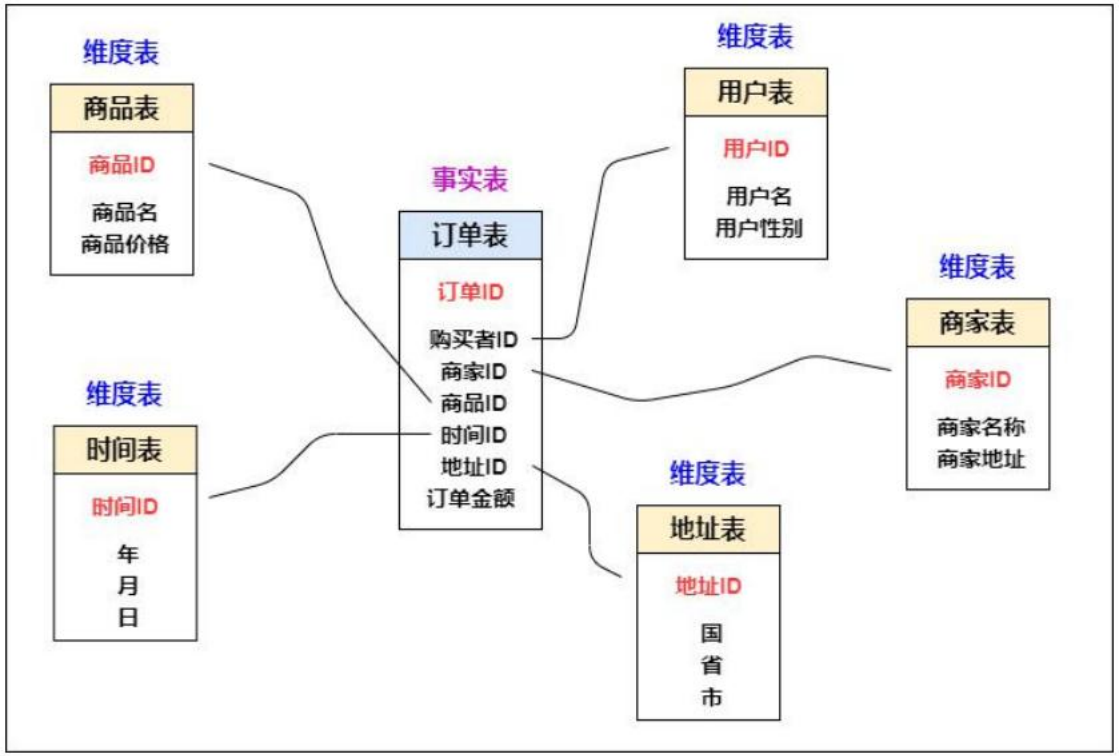
- **事务事实表：**发生在某个时间点上的一个事件。
- **周期快照事实表：**如果需要对某一天或者某个月的数据进行分析，那么可以使用周期快照事实表。
- **累计快照事实表：**累计快照表，具有确定的开始和结束事件，并且记录关键事件或者过程的里程碑。
- **无事实的事实表：**就是讲述不同维度之间的对应关系。
- **聚集事实表：**是原子事实表上的汇总数据，也称为汇总事实表。
- **合并事实表：**是指将位于不同事实表中处于相同粒度的事实进行组合建模而成的一种事实表。

Part-03：维度建模 —— 维度表

- 维度表的定义
- 维度表的原则
- 维度表的设计
- 维度表的术语

维度建模 —— 维度表

- 维度表的特点
  - 维度表谨记一条原则，包含单一主键列，但有时因业务复杂，也可能出现联合主键，请尽量避免。如果无法避免，也要确保必须是单一的，这很重要。
  - 如果维表主键不是单一，和事实表关联时会出现数据发散，导致最后结果可能出现错误。
  - 维度表的主键可以作为与之关联的任何事实表的外键。
  - 维度表通常比较宽，包含大量的低粒度的文本属性。



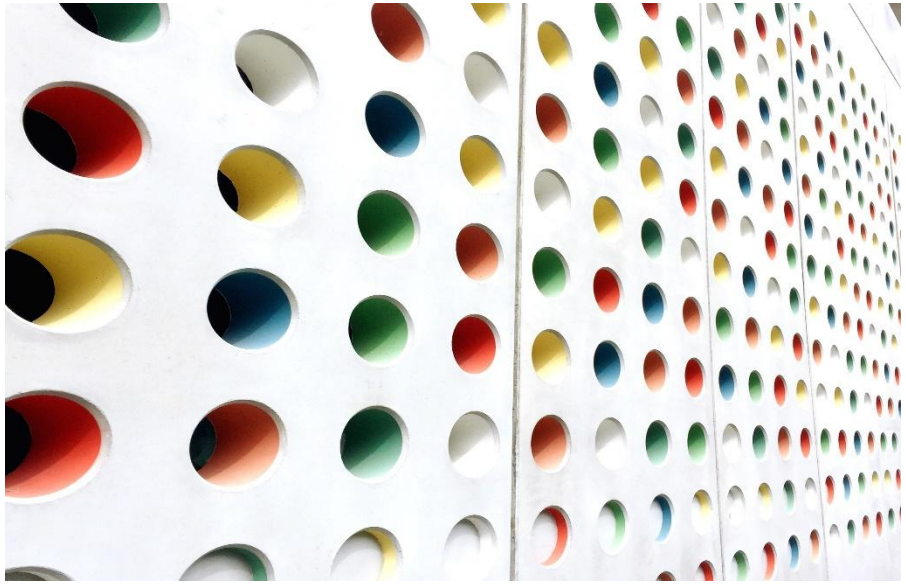
事实与维度

维度建模 —— 维度表

- 维度表的设计原则
  - 在数据仓库中不需要严格遵守规范化设计原则。因为数据仓库的主导功能就是面向分析，以查询为主，不涉及数据更新操作。
  - 事实表的设计是以能够正确记录历史信息为准则。
  - 维度表的设计是以能够以合适的角度来聚合主题内容为准则 。

维度建模 —— 维度表

- 维度表相关术语：跨表钻取
  - 跨表钻取意思是当每个查询的行头都包含相同的一致性属性时，使不同的查询能够针对两个或更多的事实表进行查询。钻取可以改变维的层次，变换分析的粒度。它包括上钻/下钻：
    - ✓ **上钻（roll-up）**：上卷是沿着维的层次向上聚集汇总数据。例如，对产品销售数据，沿着时间维上卷，可以求出所有产品在所有地区每月（或季度或年或全部）的销售额。
    - ✓ **下钻（drill-down）**：下钻是上钻的逆操作，它是沿着维的层次向下，查看更详细的数据。



维度建模 —— 维度表

- 维度表相关术语：退化维度
  - 退化维度就是将维度退回到事实表中。因为有时维度除了主键没有其他内容，虽然也是合法维度键，但是一般都会退回到事实表中，减少关联次数，提高查询性能。

维度建模 —— 维度表

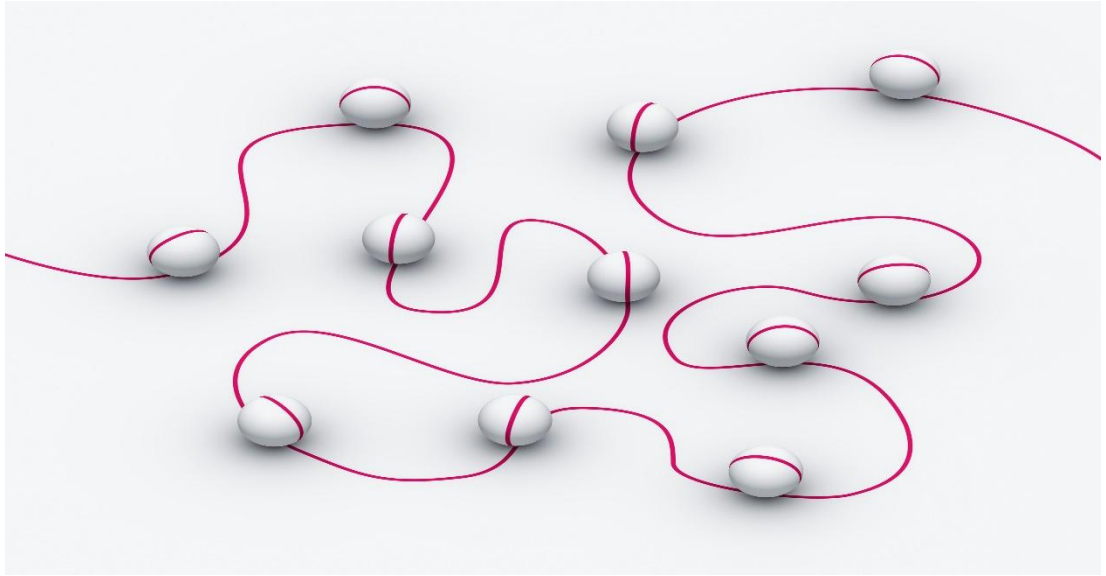
- 维度表相关术语：多层次维度
  - 多数维度包含不止一个自然层次，如日期维度可以从天的层次到周到月到年的层次。所以在有些情况下，在同一维度中存在不同的层次。

维度建模 —— 维度表

- 维度表相关术语：维度表空值属性
  - 当给定维度行没有被全部填充时，或者当存在属性没有被应用到所有维度行时，将产生空值维度属性。上述两种情况，推荐采用描述性字符串代替空值，如使用unknown 或 not applicable 替换空值。

维度建模 —— 维度表

- 维度表相关术语：日历日期维度
  - 在日期维度表中，主键的设置不要使用顺序生成的 id 来表示，可以使用更有意义的数据表示，比如将年月日合并起来表示，即 YYYYMMDD，或者更加详细的精度。



小结

- 事实表的设计是以能够正确记录历史信息为准则 ， 而维度表的设计是以能够以合适的角度来聚合主题内容为准则。
- 维度表谨记一条原则，包含单一主键列。
- 维度表通常比较宽，包含大量的低粒度的文本属性。

- 维度建模三种模式
  - 星型建模
  - 雪花建模
  - 星座建模
- 维度设计
- 维度建模四步走

Part-04：维度建模的三种模式

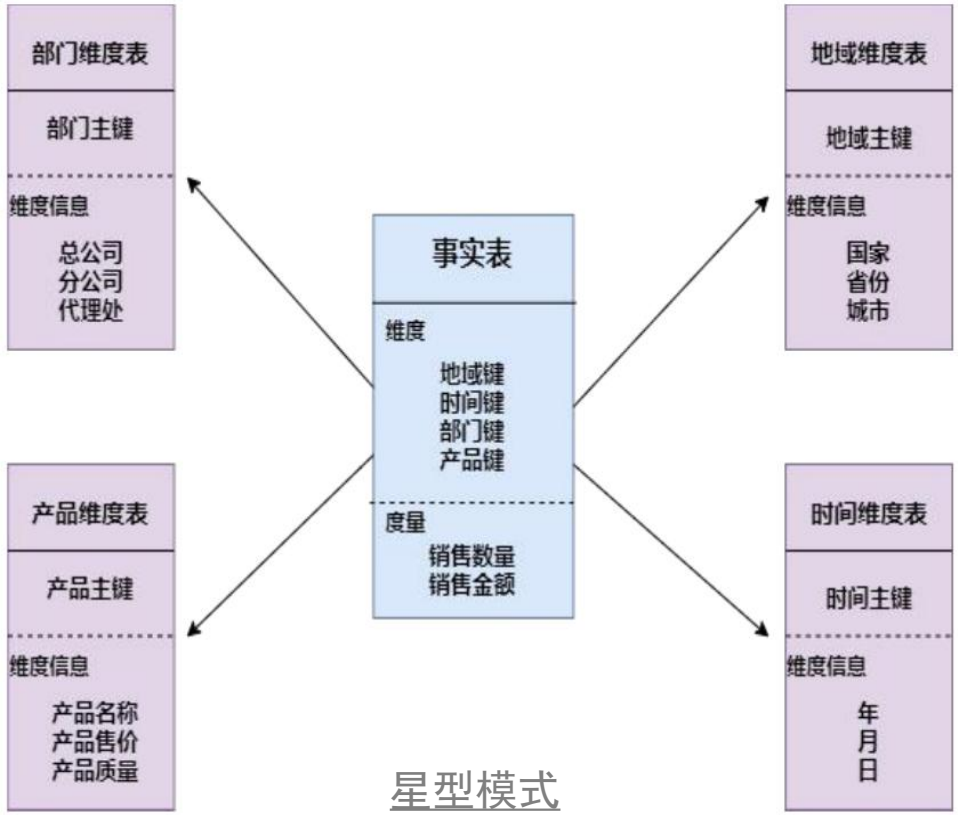
维度建模的三种模式

- 维度建模的模式
  - 维度建模一般有如下三种模式：
    - ✓ 星型模式
    - ✓ 雪花模式
    - ✓ 星座模式

星型模式	雪花模式	星座模式
• 星形模式(Star Schema)是最常用的维度建模方式。 星型模式是以事实表为中心	• 雪花模式(Snowflake Schema)是对星形模式的扩展。雪花模式的维度表可以拥有其他维度表	• 星座模式是星型模式延伸而来，星型模式是基于一张事实表的，而 星座模式是基于多张事实表的 ，而且共享维度信息 。

维度建模的三种模式

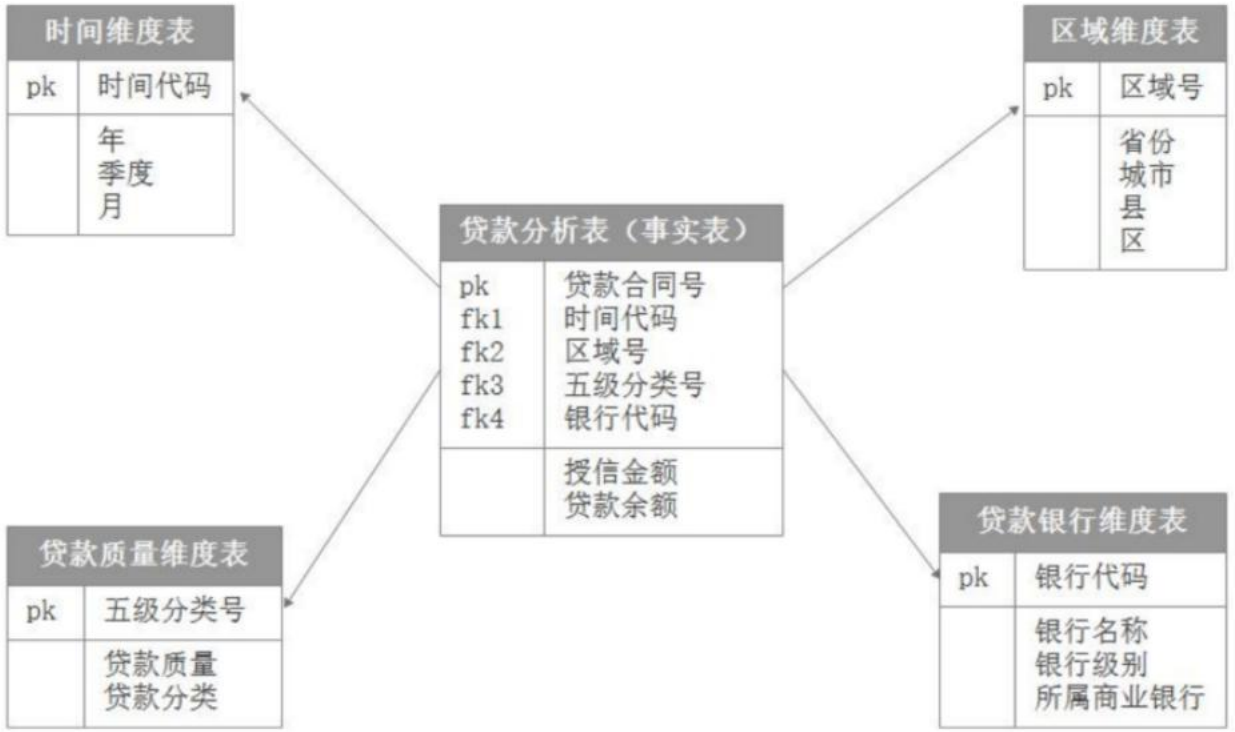
- 维度建模的模式
  - 星型模式
    - ✓ 星形模式(Star Schema)是最常用的维度建模方式。 星型模式是以事实表为中心 ，所有的维度表直接连接在事实表上 ， 像星星一样 。星形模式的维度建模由一个事实表和一组维表成，且具有以下特点：
    - ✓ a. 维表只和事实表关联，维表之间没有关联；
    - ✓ b. 每个维表主键为单列，且该主键放置在事实表中，作为两边连接的外键；
    - ✓ c. 以事实表为核心，维表围绕核心呈星形分布；



星型模式

维度建模的三种模式

- 维度建模的模式
  - 星型模式（示例）



星型模式示例

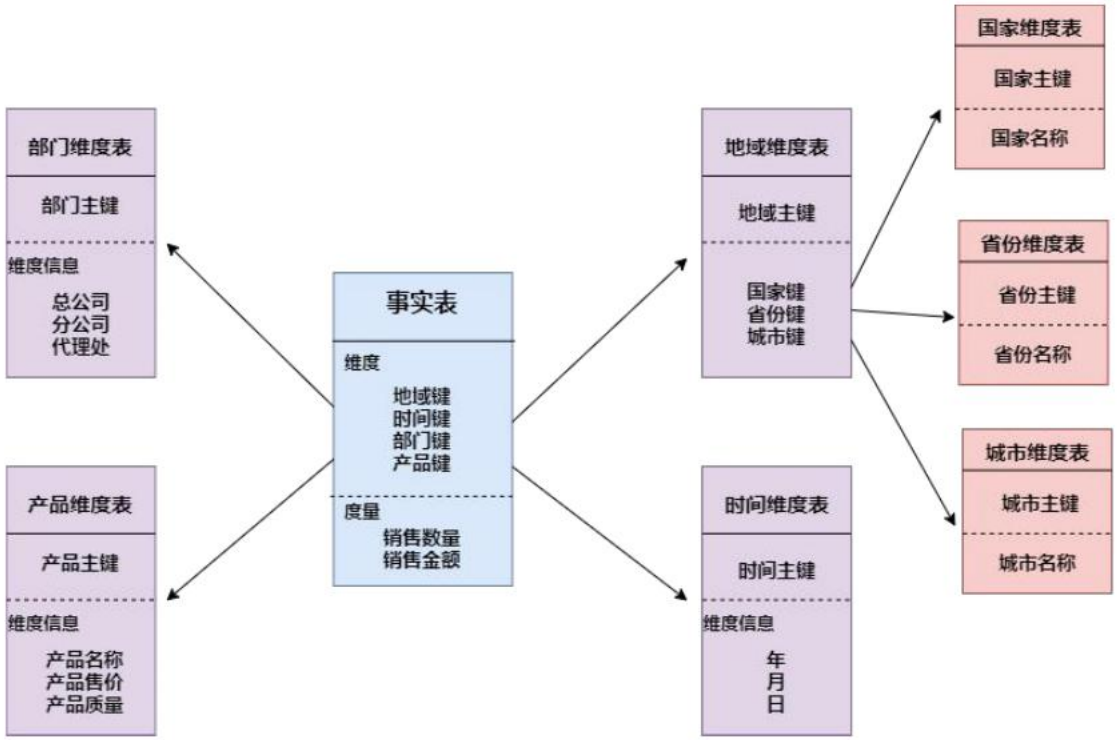
维度建模的三种模式

- 维度建模的模式
  - 星型模式（优缺点）
    - ✓ 优点：
      - 1）星型模型因为数据的冗余所以很多统计查询不需要做外部的连接，因此一般情况下效率要比其他模式要高一些。
      - 2）星型模型不用考虑很多正规化的因素，设计与实现都比较简单。
    - ✓ 缺点：
      - 1）产生冗余，不符合3NF（数据库设计三大范式）。
      - 2）不够标准正规，可维护性差。

星型模式示例

维度建模的三种模式

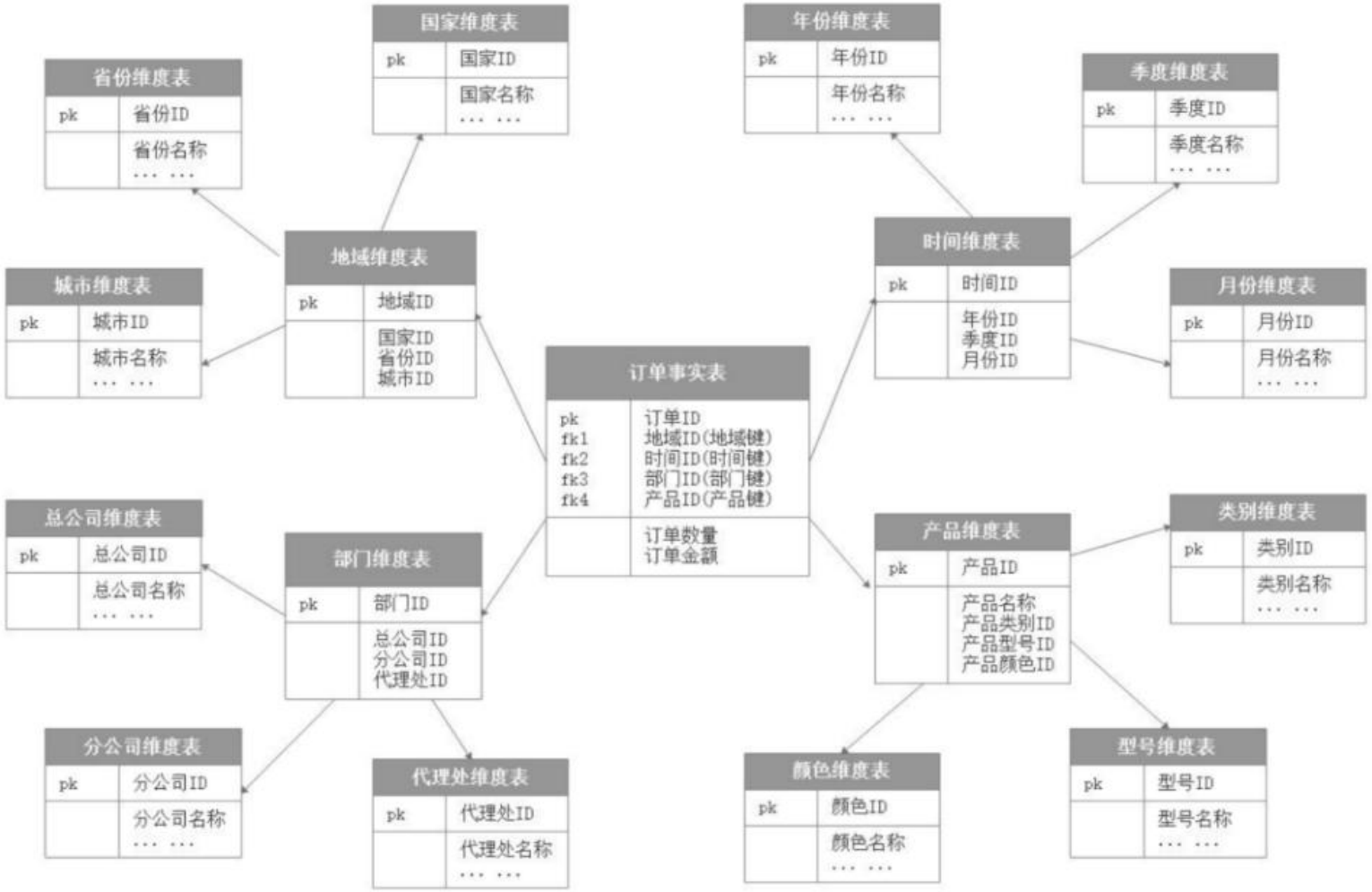
- 维度建模的模式
  - 雪花模式
    - ✓ 雪花模式(Snowflake Schema)是对星形模式的扩展。 雪花模式的维度表可以拥有其他维度表的。
    - ✓ 虽然这种模型相比星型更规范一些，但是由于这种模型不太容易理解，维护成本比较高，而且性能方面需要关联多层维表，性能也比星型模型要低。所以一般不是很常用。



雪花模式

维度建模的三种模式

- 维度建模的模式
  - 雪花模式（示例）



雪花模式示例

维度建模的三种模式

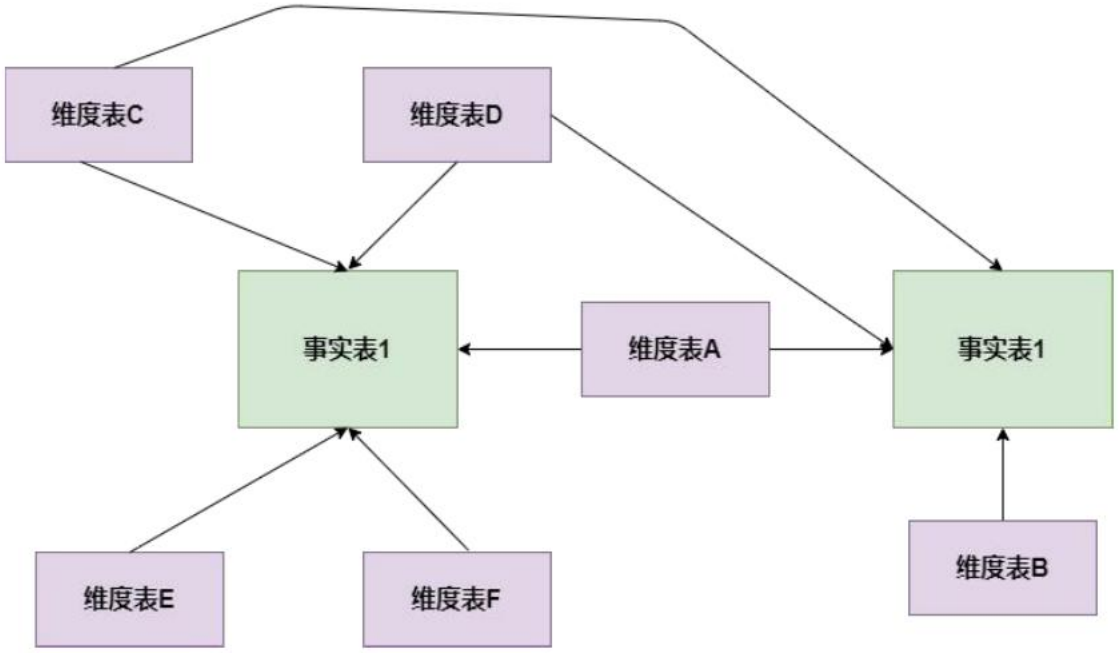
学掌门 atstudy.com

- 维度建模的模式
  - 雪花模式（优缺点）
    - ✓ 优点：
      - 1）通过最大限度地减少数据存储量。
      - 2）雪花型结构去除了数据冗余。
      - 3）符合3NF，相对标准正规，可维护性好
    - ✓ 缺点：
      - 1）消除冗余，导致更多的多表关联查询，性能上会有所降低。
      - 2）数据库表结构的设计会更加复杂

星型模式示例

维度建模的三种模式

- 维度建模的模式
  - 星座模式
    - ✓ 星座模式是星型模式延伸而来，星型模式是基于一张事实表的，而 星座模式是基于多张事实表的 ， 而且共享维度信息 。
    - ✓ 前面介绍的两种维度建模方法都是多维表对应单事实表，但在很多时候维度空间内的事实表不止一个，而一个维表也可能被多个事实表用到。
    - ✓ 在业务发展后期，绝大部分维度建模都采用的是星座模式。



星座模式

维度设计

- 维度设计
  - 代理键
  - 稳定维度
  - 缓慢渐变维

维度设计

- 维度设计
  - 代理键
    - ✓ 维度表中必须有一个能够唯一标识一行记录的列，通过该列维护维度表与事实表之间的关系，一般在维度表中业务主键符合条件可以当作维度主键。
    - ✓ 问题：
      - 当整合多个数据源的维度时，不同数据源的业务主键重复怎么办?
      - 涉及维度拉链表时，同一主体对调记录，业务键重复怎么办?

id	name	note
1	da	finc
2	jiang	finc

S1. goods

id	name	note
1	jiangtai	risk
2	tai	risk

S2. goods

维度建模

- 维度设计
  - 代理键

S1. goods

id	name	note
1	da	finc
2	jiang	finc

S2. goods

id	name	note
1	jiangtai	risk
2	tai	risk

gid	id	name	note	source
1	1	da	finc	s1
2	2	jiang	fin	s1
3	1	jiangtai	risk	s2
4	2	tai	ris	s2

维度设计

- 维度设计
  - 代理键
    - ✓ 是由数据仓库处理过程中产生的、与业务本身无关的、唯一标识维度表中一条记录并充当维度表主键的列。
    - ✓ 也是描述维度表与事实表关系的纽带，所以在设计有代理键的维度表中，事实表中的关联键是代理键而不是原有的业务主键。
    - ✓ 既业务关系是靠代理键维护，这样有效避免源系统变化对数仓数据对影响。

在实际业务中，代理键通常是数值型、自增的值问题，传统数据库有自增id默认功能。

维度设计

- 维度设计
  - 缓慢渐变维度
    - ✓ 维度数据会随着时间发生变化，变化速度比较缓慢，这种维度数据通常称作缓慢渐变维；
    - ✓ 由于数据仓库需要追溯历史变化，尤其是一些重要的数据，所以历史状态也需要采取一定的措施进行保存。
      - 每天保存当前数据的全量快照数据，该方案适合数据量较小的维度，使用简单的方式保存历史状态
      - 在维表中添加关键属性值的历史字段，仅保留上一个的状态值

id	name	dept	last_dept	.....
1	jiangtai	dp1	dp3	

- 维度设计
  - 缓慢渐变维度
    - ✓ 拉链表
      - 当维度数据发生变化时，将旧数据置为失效，将更改后的数据当作新的记录插入到维度表中，并开始生效，这样能够记录数据在某种粒度上的变化历史。

id	name	dept	Start_date	End_date
1	jiangtai	dp3	2018-05-01	2018-05-09
1	jiangtai	dp1	2018-05-10	9999-12-31

问题：拉链表怎么和事实表关联？

问题：拉链表怎么和事实表关联？

- 维度设计
  - 缓慢渐变维度
    - ✓ 拉链表
      - **答案：添加代理键**

- Fact_order

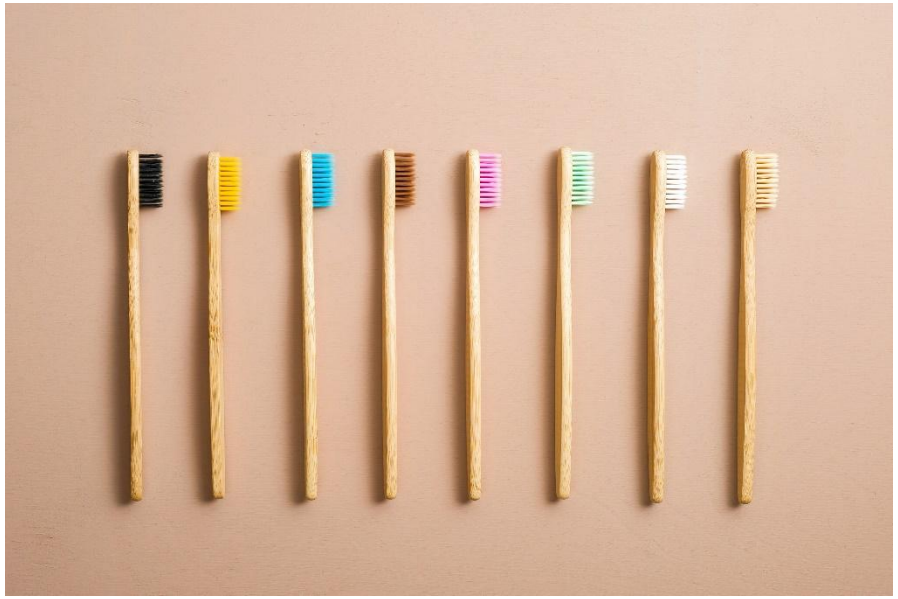
oid	uid	tm_id
1	1	9
2	2	10

- Dim_user

uid	id	name	dept	Start_date	End_date
1	1	jiangtai	dp3	2018-05-01	2018-05-09
2	1	jiangtai	dp1	2018-05-10	9999-12-31

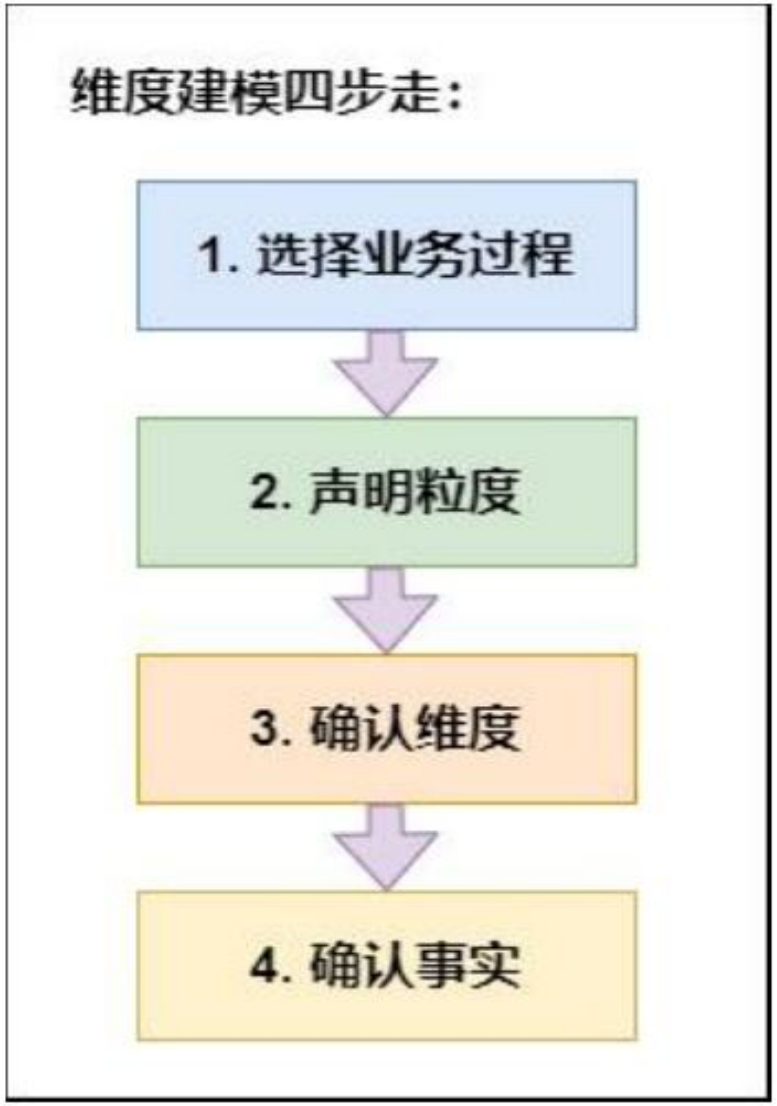
小结

- 维度设计的三种形式：代理键、稳定维度、缓慢渐变维。
- 代理键是维度建模中极力推荐的方式，它的应用能有效的隔离源端变化带来的数仓结构不稳定问题，同时也能够提高数据检索性能。
- 但同时代理键维护代价也非常高，尤其是数据装载过程中，对事实表带来了较大的影响。在基于hive的数据仓库建设影响更加严重，比如代理键的生成、事实表中关联键的装载、不支持非等值关联等问题，带来ETL过程更加复杂。
- 在大数据体系下，谨慎使用代理键。
- 同时对于缓慢渐变维场景，可以考虑用空间换取时间，每天保留维表全量快照;但这样会带来存储成本，要根据实际情况衡量



维度建模步骤

- 维度建模的流程
  - 实际业务中，给了我们一堆数据，我们怎么拿这些数据进行数仓建设呢？
  - 维度建模四步走 ：



维度建模步骤

- 维度建模四步走
  - step-01、选择业务过程
    - ✓ 维度建模是紧贴业务的，所以必须以业务为根基进行建模.
    - ✓ 选择业务过程，就是在整个业务流程中选取我们需要建模的业务，根据运营提供的需求及日后的易扩展性等进行选择业务。
    - ✓ 比如商城：
      - 整个商城流程分为商家端，用户端，平台端
      - 运营需求是总订单量，订单人数，及用户的购买情况等
      - 我们选择业务过程就选择用户端的数据，商家及平台端暂不考虑。
    - ✓ 业务选择非常重要，因为后面所有的步骤都是基于此业务数据展开的。

维度建模步骤

- 维度建模四步走
  - step-02、声明粒度
    - ✓ 维度建模中要求我们，在同一事实表中，必须具有相同的粒度 。
    - ✓ 存在一对一的关系就是相同粒度。
    - ✓ 同一事实表中不要混用多种不同的粒度，不同的粒度数据建立不同的事实表。
    - ✓ 从给定的业务过程获取数据时强烈建议从关注原子粒度开始设计，也就是从最细粒度开始

维度建模步骤

- 维度建模四步走
  - step-03、确认维度
    - ✓ 维度表是作为业务分析的入口和描述性标识，所以也被称为数据仓库的“灵魂”。
    - ✓ 在一堆的数据中怎么确认哪些是维度属性呢？
      - ✓ 如果该列是对具体值的描述，是一个文本或常量，某一约束和行标识的参与者，此时该属性往往是维度属性，如：部门、产品、城市、省份等，
    - ✓ 数仓工具箱中告诉我们 牢牢掌握事实表的粒度，就能将所有可能存在的维度区分开，并且要 确保维度表中不能出现重复数据，应使维度主键唯一。

维度建模步骤

- 维度建模四步走
  - step-04、确认事实
    - ✓ 事实表是用来度量的，基本上都以数量值表示，事实表中的每行对应一个度量，
    - ✓ 每行中的数据是一个特定级别的细节数据，称为粒度。
    - ✓ 维度建模的核心原则之一是同一事实表中的所有度量必须具有相同的粒度 。这样能确保不会出现重复计算度量的问题。有时候往往不能确定该列数据是事实属性还是维度属性。
    - ✓ 最实用的事实就是数值类型和可加类事实 。
    - ✓ 所以可以通过分析该列是否是一种包含多个值并作为计算的参与者的度量，这种情况下该列往往是事实。

小结-1

- 星型模型与雪花模型对比：
  - 1、**星型模型和雪花模型主要区别就是对维度表的拆分**，对于雪花模型，维度表的设计更加 规范，一般符合三范式设计;而星型模型，一般采用降维的操作，维度表设计不符合三范式 设计，反规范化，**利用冗余牺牲空间来避免模型过于复杂，提高易用性和分析效率**。
  - 2、**星型模型**因为**数据的冗余**所以很多统计查询不需要做外部的连接，因此一般情况下**效率 比雪花型模型要高**。**星型结构**不用考虑很多正规化的因素，**设计与实现都比较简单**。
  - 3、**雪花型模型**由于去除了冗余，有些统计就需要通过表的联接才能产生，所以**效率不一定 有星型模型高**。正规化也是一种比较复杂的过程，相应的数据库结构设计、数据的 ETL、以 及**后期的维护都要复杂一些**。因此在冗余可以接受的前提下，**数仓构建实际运用中星型模型 使用更多，也更有效率**。
  - 4、此外，在数据仓库中**星座模型**也使用比较多，当**多个事实表共用多张维度表**时，就构成 了星座模型。

小结-2

- 星型模型与雪花模型对比：

属性	星型模型	雪花模型
数据总量	多	少
可读性	容易	差
表个数	少	多
查询速度	快	慢
冗余度	高	低
对实时表的情况	增加宽度	字段比较少，冗余低
扩展性	差	好

小结-3

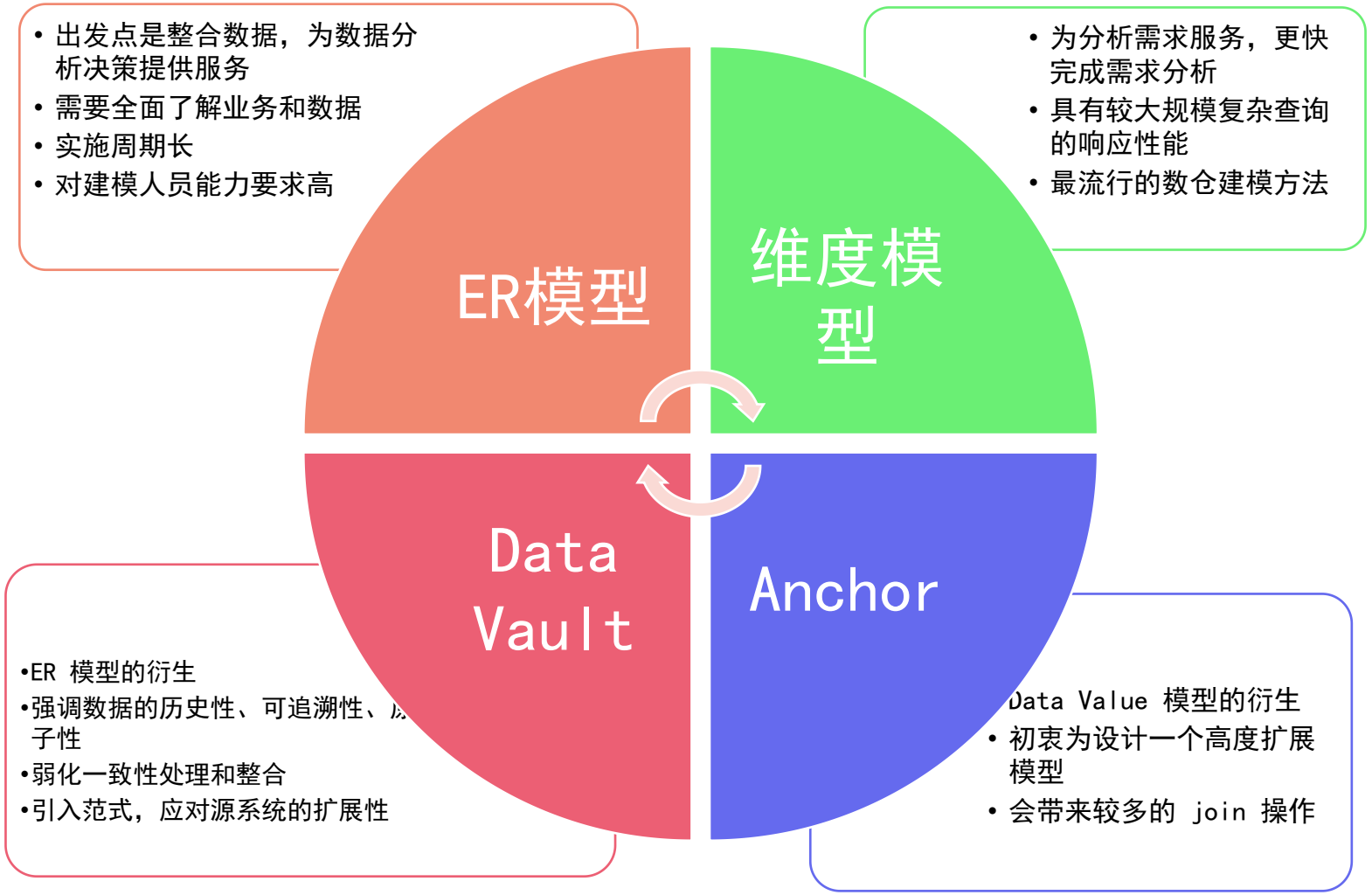
- 维度建模四大步
  - 选择业务过程
  - 声明粒度
  - 确认维度
  - 确认事实

- DataVault 模型
- Anchor 模型
- 数仓建模理论知识总结

Part-05：DataVault 建模与 Anchor 建模

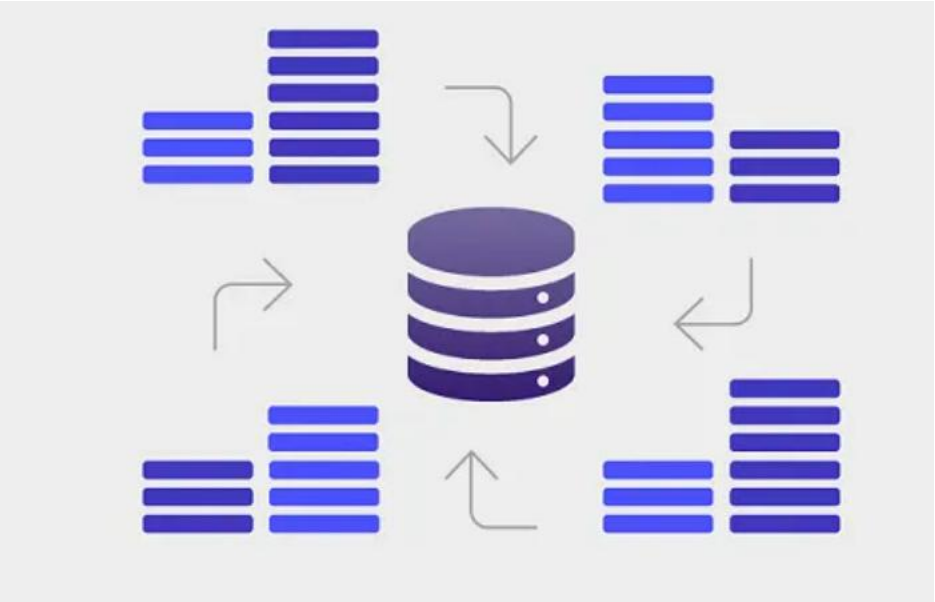
数仓建模方法

- 数仓建模方法：
  - 数据仓库建模有如下四种：ER 模型、维度模型、Data Vault、Anchor。
  - 重点掌握的是 ER模型与 **维度模型**。



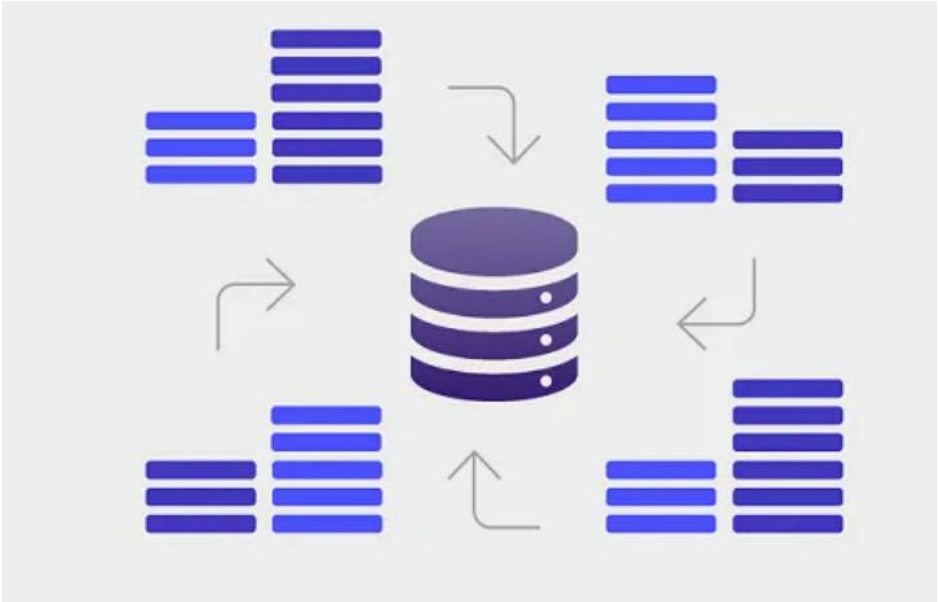
DataVault 建模

- DataVault 模型简介
  - Data Vault是在ER模型的基础上衍生而来。
  - 模型设计的初衷是有效的组织基础数据层，使之易扩展、灵活的应对业务的变化。
  - 同时强调历史性、可追溯性和原子性，不要求对数据进行过度的一致性处理。
  - 并非针对分析场景所设计。



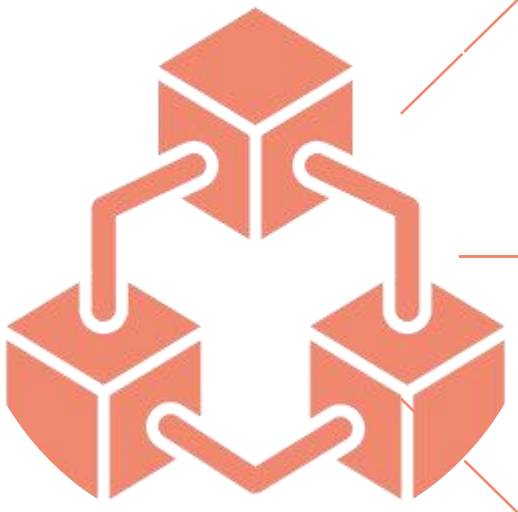
DataVault 建模

- DataVault 模型简介
  - Data Vault 模型是一种中心辐射式模型，其设计重点围绕着业务键的集成模式。
  - 这些业务键是存储在多个系统中的、针对各种信息的键，用于定位和唯一标识记录或数据。



DataVault 建模

- DataVault 模型基本结构
  - 包含三种基本结构
    - ✓ 中心表-Hub
    - ✓ 链接表-Link
    - ✓ 卫星表- Satellite



中心表-Hub

- 唯一业务键的列表，唯一标识企业实际业务，企业的业务主体集合

链接表-Link

- 表示中心表之间的关系，通过链接表串联整个企业的业务关联关系

卫星表-Satellite

- 历史的描述性数据，数据仓库中数据的真正载体

DataVault 建模

- 中心表-Hub
  - 只包含业务主键信息以及数据装载的描述，不包含非键值以外的业务数据属性本身；
  - 比如中心表商品，在DataVault下的涉及：

Hub_商品	
PK	Hub_商品ID
	商品ID load_time

注：商品属性以及描述信息，都属于卫星表的范畴！

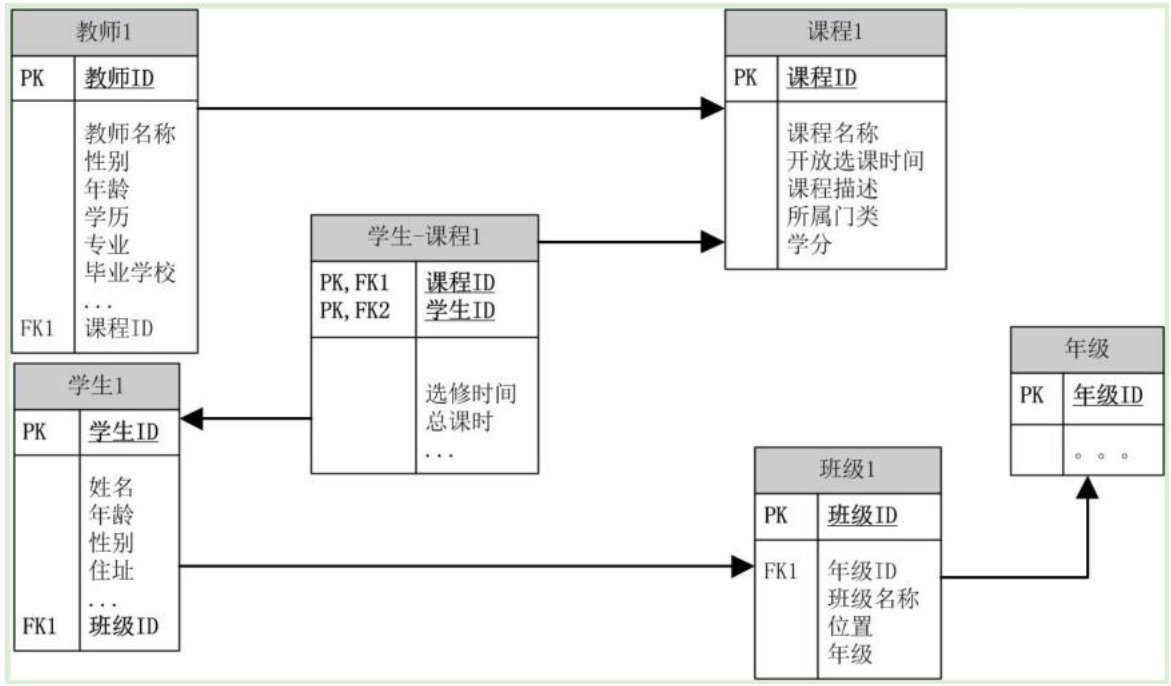
- 链接表-Link
  - 链接表用来描述中心表间的关联关系，其不包含业务键值以及数据装载描述以外的任何非键值数据。
  - 比如学生选课链接表，其设计如下：

Link_授课	
PK	Link 授课ID
FK1 FK2	hub_教师ID hub_课程ID load_time

注：与选课相关的课时数等描述信息，都属于卫星表的范畴 ！

DataVault 建模

- DataVault模型案例（1/3）
  - 学生选课ER模型：

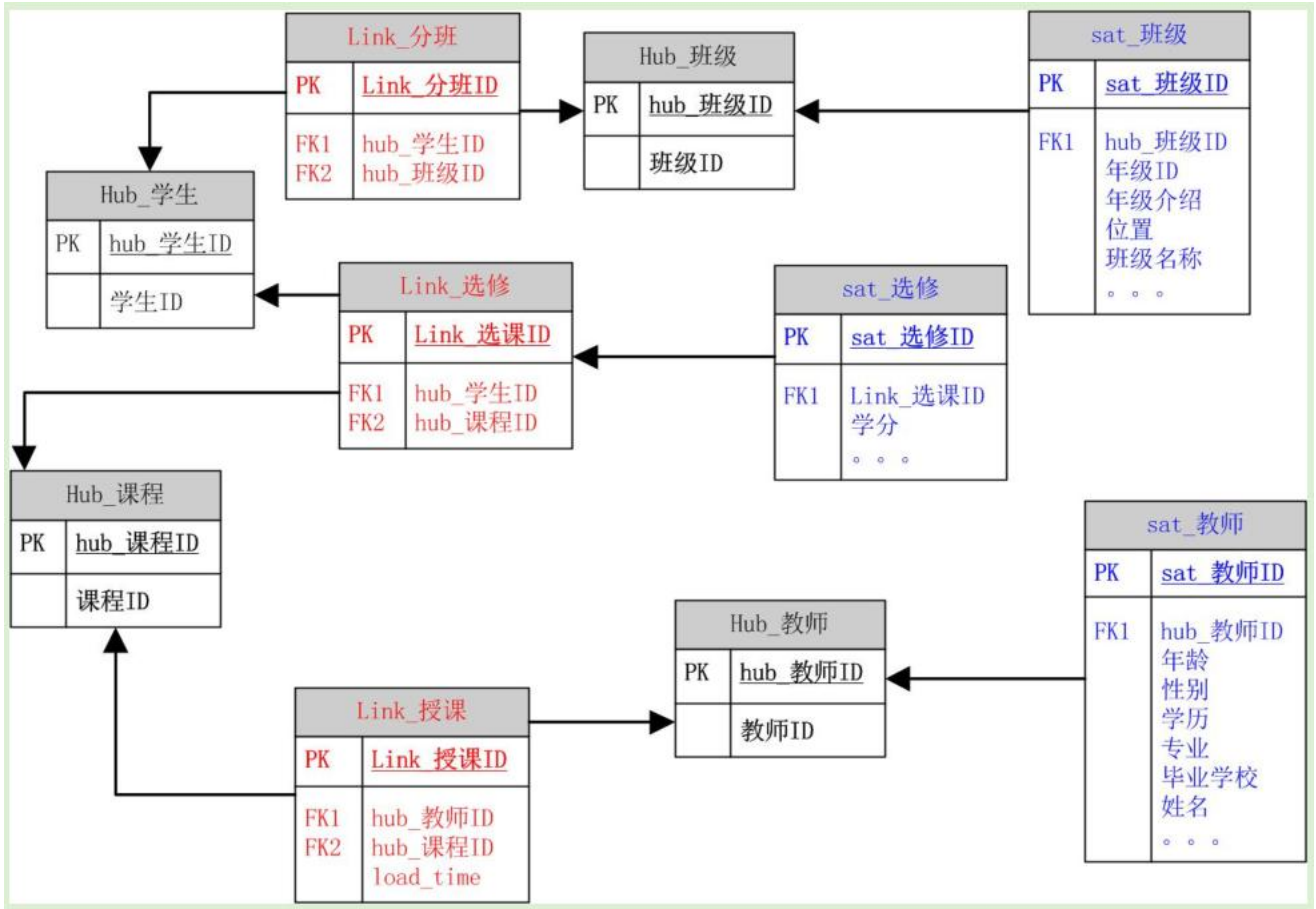


DataVault 建模

- DataVault模型案例（2/3）
  - 将ER模型重构为DataVault模型思路：
    - ✓ I. 梳理所有主要实体
    - ✓ II. 将有入边的实体定义为中心表
    - ✓ III. 将没有入边且仅有一个出边的表定义为中心表
    - ✓ IV. 源模型中没有入边且有两条或以上出边的表定义为链接表
    - ✓ V. 将外键关系定义为链接表

DataVault 建模

- DataVault模型案例（3/3）
  - 按DataVault改造完成后的大概模型（卫星表未补充）：



DataVault 建模

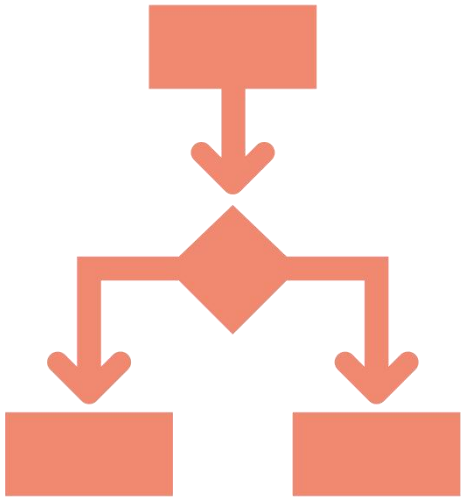
- DataVault模型的特点
  - Data Vault 模型更容易设计，ETL过程中更易配置化实现。
  - Hub想像成人体的骨架，那么Link就是连接骨架的韧带组织，而satelite就是骨架上的血肉。
  - Data Vault是对ER模型更近一步的规范化，由于对数据的拆解和更偏向于基础数据组织，在处理分析类场景时相对复杂。
  - Data Vault 模型适合数仓低层构建，目前实际应用场景较少。

Anchor 建模

- Anchor模型
  - Anchor是对Data Vault模型做了更近一步的规范会处理。
  - 初衷是为了设计高度可扩展的模型，核心思想是所有的扩张只添加而不修改，于是设计出的模型基本变成了k-v结构的模型。
  - 模型范式达到了6NF。 由于过度规范化，使用中牵涉到太多的join操作。
  - 目前没有实际案例，仅作了解。

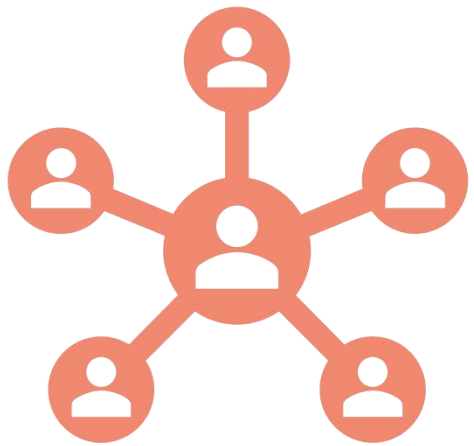
数据仓库建模理论小结

- 数据仓库建模小结：
  - 时下四种基本的建模方法，当前主流建模方法为：ER模型、维度模型。
    - ✓ **ER模型：**常用于OLTP数据库建模，应用到构建数仓时更偏重数据整合，站在企业整体考虑，将各个系统的数据按相似性一致性、合并处理，为数据分析、决策服务，但并不便于直接用来支持分析。
    - ✓ ER模型的问题：
      - 需要全面梳理企业所有的业务和数据流
      - 实施周期长
      - 对建模人员要求高



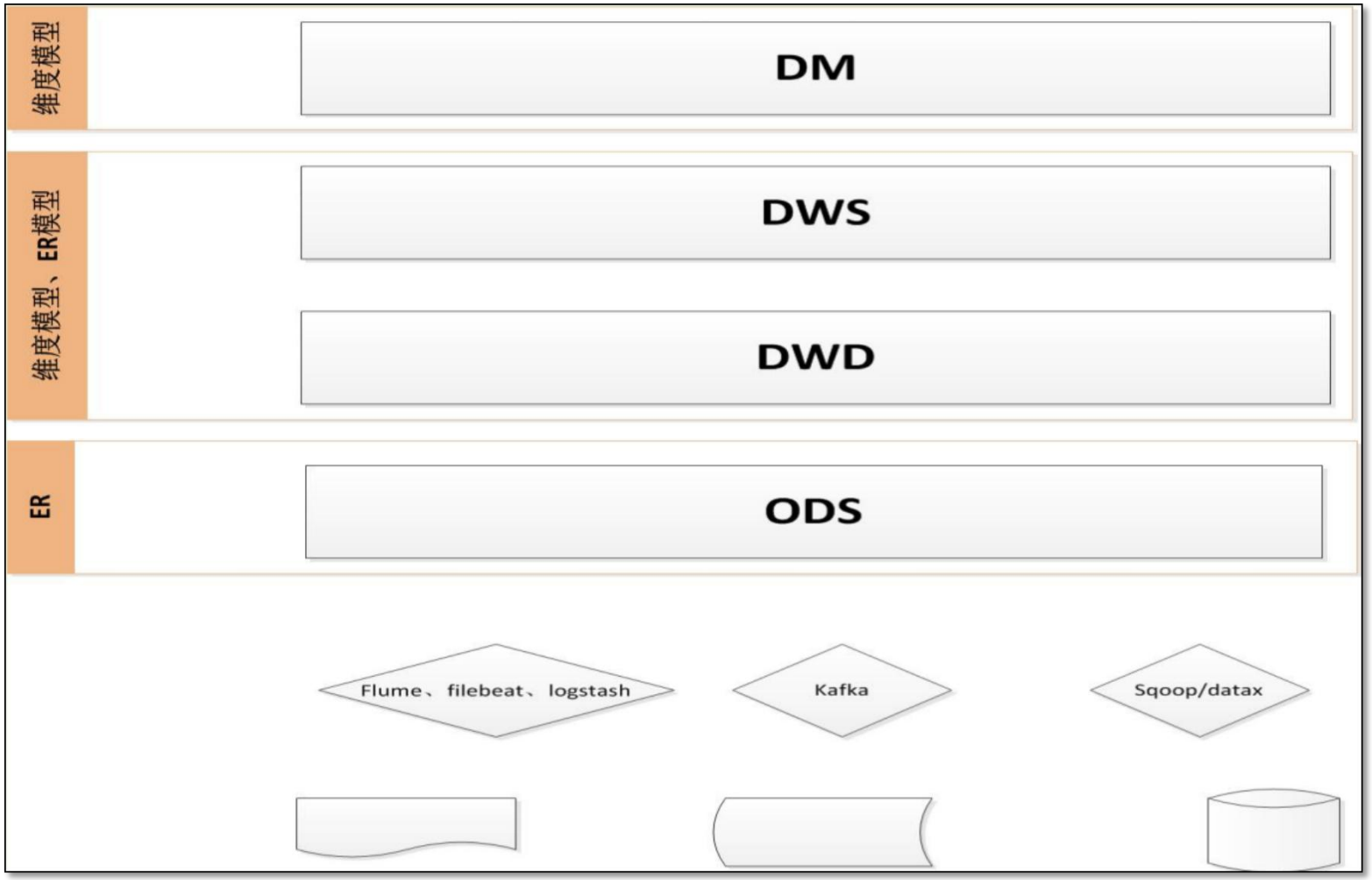
数据仓库建模理论小结

- 数据仓库建模小结：
  - 时下四种基本的建模方法，当前主流建模方法为：ER模型、维度模型。
    - ✓ **维度模型**：维度建模是面向分析场景而生，针对分析场景构建数仓模型；重点关注快速、灵活的解决分析需求，同时能够提供大规模数据的快速响应性能。针对性强，主要应用于数据仓库构建和OLAP引擎低层数据模型。
    - ✓ 维度模型的优点：
      - 不需要完整的梳理企业业务流程和数据
      - 实施周期根据主题边界而定，容易快速实现demo



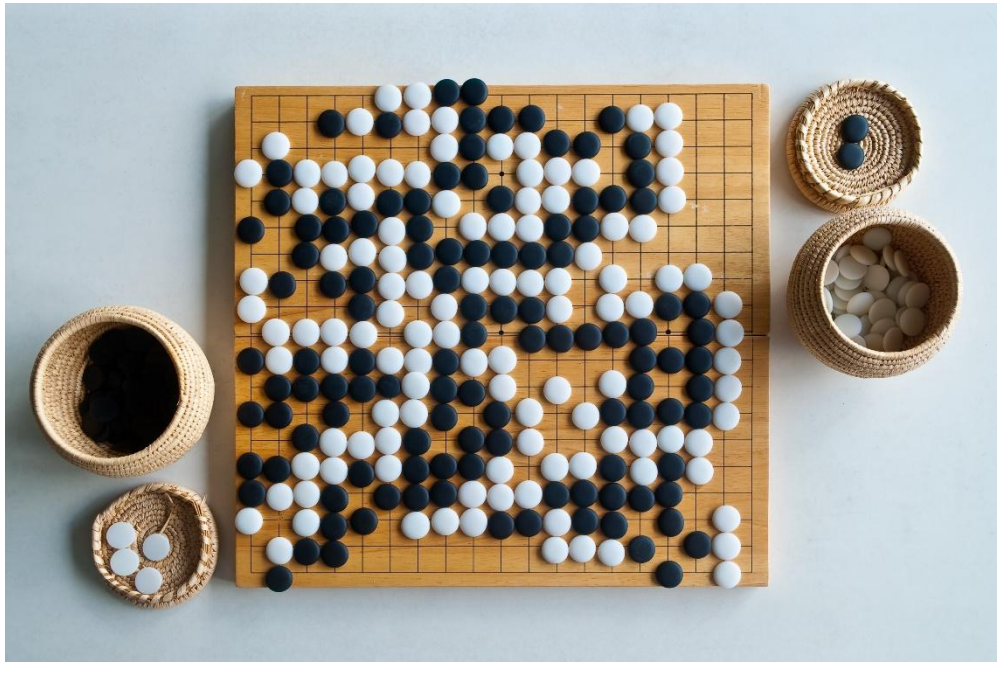
数据仓库建模理论小结

- 数据仓库建模小结：
  - CIF层次架构



数据仓库建模理论小结

- 数据仓库建模小结：
  - 数仓模型的选择是灵活的，不局限于某一种模型方法
  - 数仓模型的设计也是灵活的，以实际需求场景为导向
  - 模型设计要兼顾灵活性、可扩展，而对终端用户透明性
  - 模型设计要考虑技术可靠性和实现成本



学掌门
atstudy.com

谢谢观看
Thanks

学习掌门人 职场引路人