2万字总结阿里大数据之路-数据模型篇（建议收藏）

大家好，我是唐三少。上期给大家带来《万字总结阿里大数据之路-数据技术篇（建议收藏）》，今天带来中篇啦，欢迎大家品读。上号上号。

第1章 大数据领域建模综述

1.1 为什么需要数据建模

- 有结构地分类组织和存储是我们面临的一个挑战。
- 数据模型强调从业务、数据存取和使用角度合理存储数据。
- 数据模型方法，以便在性能、成本、效率之间取得最佳平衡
  - 成本：良好的数据模型能极大地减少不必要的数据冗余，也能实现计算结果复用，极大地降低大数据系统中的存储和计算成本。
  - 效率：良好的数据模型能极大地改善用户使用数据的体验，提高使用数据的效率。
  - 质量：良好的数据模型能改善数据统计口径的不一致性，减少数据计算错误的可能性。

1.2 关系数据库系统和数据仓库

1.3 从 OLTP 和 OLAP 系统的区别看模型方法论的选择

- OLTP 系统通常面向的主要数据操作是随机读写，主要采用满足 3NF 的实体关系模型存储数据，从而在事务处理中解决数据的冗余和一 致性问题；
- OLAP 系统面向的主要数据操作是批量读写，事务处理中 的一致性不是OLAP 所关注的，其主要关注数据的整合，以及在一次性的复杂大数据查询和处理中的性能，因此它需要采用一些不同的数据建模方法。

1.4 典型的数据仓库建模方法论

1.4.1 ER 模型

- 采用 ER模型建设数据仓库模型的出发点是整合数据，将各个系统中的数据以整个企业角度按主题进行相似性组合和合并，并进行一致性处理，为数据分析决策服务，但是并不能直接用于分析决策。
- ER 模型在实践中最典型的代表是 Teradata 公司基于金融业务发布 的 FS-LDM（Financial Services Logical Data Model），它通过对金融业务的高度抽象和总结，将金融业务划分为10大主题，并以设计面向金融仓库模型的核心为基础，企业基于此模型做适当调整和扩展就能快速落地实施。

1.4.2 维度模型

- 维度建模从分析决策的需求出发构建模型，为分析需求服务，因此它重点关注用户如何更快速地完成需求分析，同时具有较好的大规模复杂查询的响应性能。其典型的代表是星形模型，以及在一些特殊场景下使用的雪花模型。设计步骤通常如下：
  - 选择需要进行分析决策的业务过程。业务过程可以是单个业务事件，比如交易的支付、退款等；也可以是某个事件的状态，比如当前的账户余额等；还可以是一系列相关业务事件组成的业务流程，具体需要看我们分析的是某些事件发生情况，还是当前状态，或是事件流转效率。
  - 选择粒度。在事件分析中，我们要预判所有分析需要细分的程度，从而决定选择的粒度。粒度是维度的一个组合。
  - 识别维表。选择好粒度之后，就需要基于此粒度设计维表，包括维度属性，用于分析时进行分组和筛选。
  - 选择事实。确定分析需要衡量的指标。

1.4.3 Data Vault 模型

- 它强调建立一个可审计的基础数据层，也就是强调数据的历史 性、可追溯性和原子性，而不要求对数据进行过度的一致性处理和整合；同时它基于主题概念将企业数据进行结构化组织，并引入了更进一步的范式处理来优化模型，以应对下游、系统变更的扩展性。

1.4.4 Anchor 模型

- Anchor 对 Data Vault 模型做了进一步规范化处理， Lars.Ronnback 的初衷是设计一个高度可扩展的模型，其核心思想是所有的扩展只是添加而不是修改，因此将模型规范到 6NF ，基本变成了k-v 结构化模型。

1.5 阿里巴巴数据模型实践综述

- 第一个阶段：构建在 Oracle 上，数据完全以满足报表需求为目的
- 第二个阶段：引入了当时 MPP 架构体系的 Greenplum，ODL（操作数据层）+BDL（基础数据层）+IDL（接口数据层）+ADL（应用数据层）；BDL希望引入 ER 模型，加强 数据的整合，构建一致的基础数据模型，但构建 ER 模型时遇到了比较大的困难和挑战，互联网业务的快速发展、人员的快速变化、业务知识功底的不够全面，导致 ER 模型设计迟迟不能产出。至此，我们也得到了一个经验：在不太成熟、快速变化的业务面前，构建 ER 模型的风险非常大，不太适合去构建 ER 模型。
- 第三个阶段：迎来了以Hadoop 为代表的分布式存储计算平台的快速发展，同时阿里巴 巴集团自主研发的分布式计算平台 MaxCompute 也在紧锣密鼓地进行着。以 Kimball 的维度建模为核心理念的模型方法论，构建了阿里巴巴集团的公共层模型数据架构体系。
  - 数据公共层建设的目的是着力解决数据存储和计算的共享问题。数据每年以近 2.5 倍的速度在增长，数据的增长远远超过业务的增长。
  - 统一化的集团数据整合及管理的方法体系“OneData”：一致性的指标定义体系、模型设计方法体系以及配套工具。

第2章 阿里巴巴数据整合及管理体系

- 面对爆炸式增长的数据，如何建设高效的数据模型和体系，对这些 数据进行有

序和有结构地分类组织和存储，避免重复建设和数据不一致性，保证数据的规范性，一直是大数据系统建设不断追求的方向。

2.1 概述

核心：从业务架构设计（如何快速上手工作）到模型设计，从数据研发到数据服务，做到数据可管理、可追溯、可规避重复建设。

2.1.1 定位及价值

建设统一的、规范化的数据接入层（ ODS ）和数据中间层（ DWD 和 DWS ），通过数据服务和数据产品，完成服务于阿里巴巴的大数据系 统建设，即数据公共层建设。

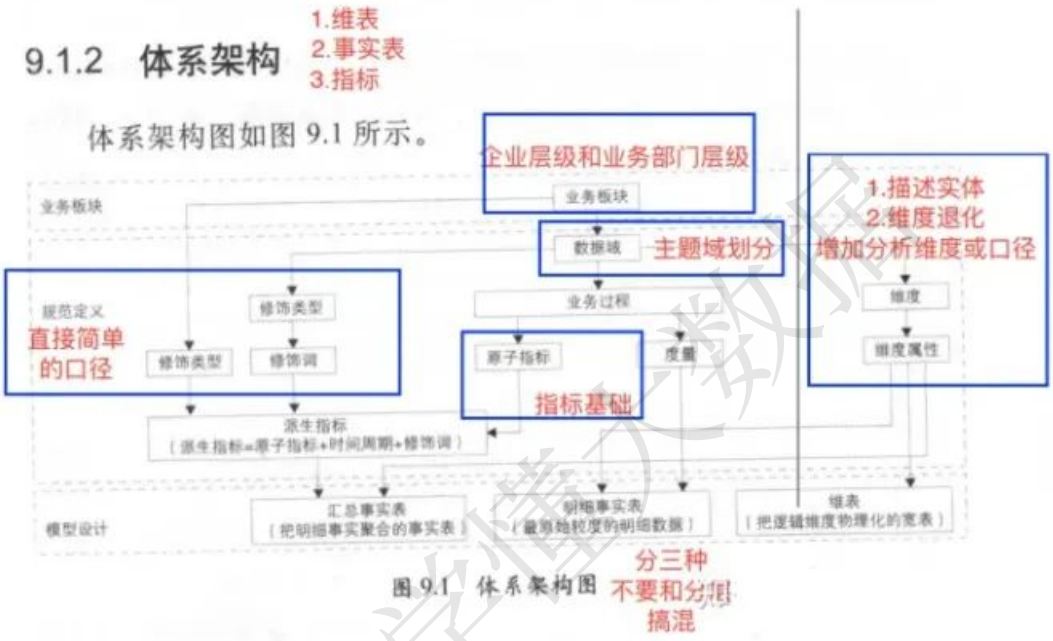


图 9.1 体系架构图

业务板块：根据业务属性划分板块，板块之间的指标或业务重叠性较小。

规范定义：一套数据规范命名体系，用在模型设计中

模型设计：以维度建模理论为基础，基于维度建模总线架构，构建一致性的维度和事实(进行规范定义)。

2.2 规范定义

规范定义指以维度建模作为理论基础，构建总线矩阵，划分和定义 数据域、业务过程、维度、度量 / 原子指标、修饰类型、修饰词、时间周期、派生指标。

2.2.1 名词术语

数据域（主题域）

- 面向业务分析，将业务过程或者维度进行抽象的集合。业务过程可以概括为一个个不可拆分的行为事件，在业务过程之下，可以定义指标；维度是指度量的环境，如买家下单事件，买家是维度 。 为保障整个体系的生命力， 数据域是需要抽象提炼，并且长期维护和更新的， 但不轻易变动。
- 常见主题域：用户、渠道、营销、流量、交易、财务、商品

业务过程

- 指企业的业务活动事件，如下单、支付、退款都是业务过程。 请注意，业务过程 是一个不可拆分的行为事件， 通俗地讲，业务过程就是企业活动中的事件

时间周期

- 用来明确数据统计的时间范用或者时间点，如最近 30天、自然周、截至当日等

修饰类型

- 是对修饰词的一种抽象划分 。 修饰类型从属于某个业务域，如日志域的访问终端类型涵盖无线端、 PC 端等修饰词

修饰词

- 指除了统计维度以外指标的业务场景限定抽象 。 修饰词隶属于一种修饰类型，如 在日志域的访问终端类型下， 有修饰词 PC 端、无线端等

度量/原子指标

- 原子指标和度量含义相同，基于某一业务事件行为下的度量，是业务定义中不可 再拆分的指标，具有明确业务含义的名词，如支付金额

维度

- 维度是度量的环境，用来反映业务的一类属性，这类属性的集合构成一个维度，也可以称为实体对象。维度属于一个数据域，如地理维度（其中包括国家、地区、 省以及城市等级别的内容)、时间维度(其中包括年、季、月、周、日等级别的内容）

维度属性

- 维度属性隶属于一个维度，如地理维度里面的国家名称、国家 ID、省份名称等都属于维度属性

派生指标

- 派生指标=一个原子指标+多个修饰词(可选)+时间周期+粒度。 可以理解为对原子指标业务统计范围的圈定。 如原子指标：支付金额，最近 1 天海外买家支付金额则为派生指标(最近1天为时间周期， 海外为修饰词， 买家作为维度，而不作为修饰词)

2.2.2 指标体系

一、基本原则

1. 组成体系之间的关系

- 派生指标由原子指标、时间周期修饰词、若干其他修饰词组合得到

图 9.3 派生指标

- 原子指标、修饰类型及修饰词，直接归属在业务过程下，其中修饰词继承修饰类型的数据域
- 派生指标可以选择多个修饰词，修饰词之间的关系为"或"或者"且"，由派生指标具体语义决定
- 派生指标唯一归属一个原子指标，继承原子指标的数据域，与修饰词的数据域无关
- 原子指标有确定的英文字段名、数据类型和算法说明；派生指标要继承原子指标的英文名、数据类型和算法要求

2. 命名约定

- 命名所用术语。指标命名尽量使用英文简写，其次是英文。太长也可以考虑汉语拼音首字母
- 业务过程。英文名：用英文或英文的缩写或者中文拼音简写
- 原子指标。英文名：动作+度量
- 修饰词。只有时间周期才会有英文名
- 派生指标。英文名：原子指标英文名+时间周期修饰词(3位,例如_1d)+序号(4位，例如_001)

表 9.2 阿里巴巴常用的时间周期修饰词

中 文 名	英 文 名	中 文 名	英 文 名
最近 1 天	1d	自然月	cm
最近 3 天	3d	自然季度	cq
最近 7 天	1w	截至当日	td
最近 14 天	2w	年初截至当日	sd
最近 30 天	1m	零点截至当前	tt
最近 60 天	2m	财年	fy
最近 90 天	3m	最近 1 小时	1h
最近 180 天	6m	准实时	ts
180 天以前	bh	未来 7 天	f1w
自然周	cw	未来 4 周	f4w

3. 算法

- 算法概述——算法对应的用户容易理解的阐述。
- 举例——通过具体例子帮助理解指标算法。
- SQL 算法说明——对于派生指标给出SQL的写法或者伪代码。

二、操作细则

派生指标可以分为三类：事务型指标、存量型指标和复合型指标。

- 事务型指标：是指对业务活动进行衡量的指标。例如新发商品数、 重发商品数、新增注册会员数、订单支付金额，这类指标需维护 原子指标及修饰词，在此基础上创建派生指标。
- 存量型指标：是指对实体对象(如商品、会员)某些状态的统计。 例如商品总数、注册会员总数，这类指标需维护原子指标及修饰词，在此基础上创建派生指标，对应的时间周期 一般为“历史截至当前某个时间”。
- 复合型指标：是在事务型指标和存量型指标的基础上复合而成的。例如浏览UV-下单买家数转化率， 有些需要 创 建新原子指标， 有些则可以在事务型或存量型原子指标的基础上增加修饰词得到派生指标。

2.3 模型设计

2.3.1 指导理论

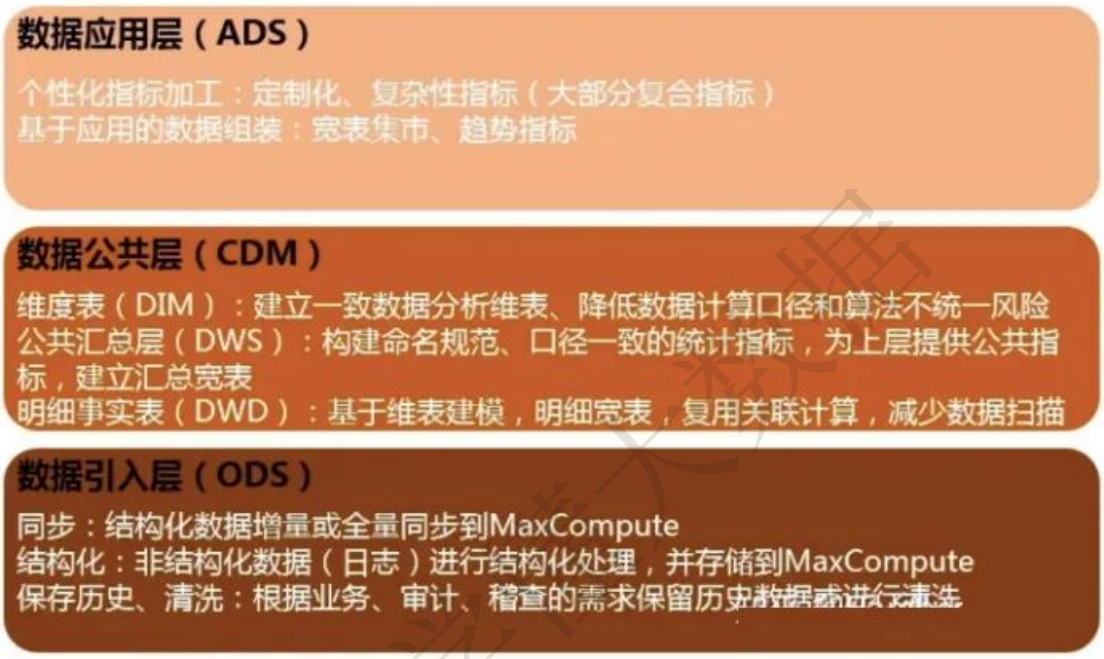
数据模型的维度设计主要以维度建模理论为基础，基于维度数据模型总线架构，构建一致性的维度和事实。

2.3.2 模型层次

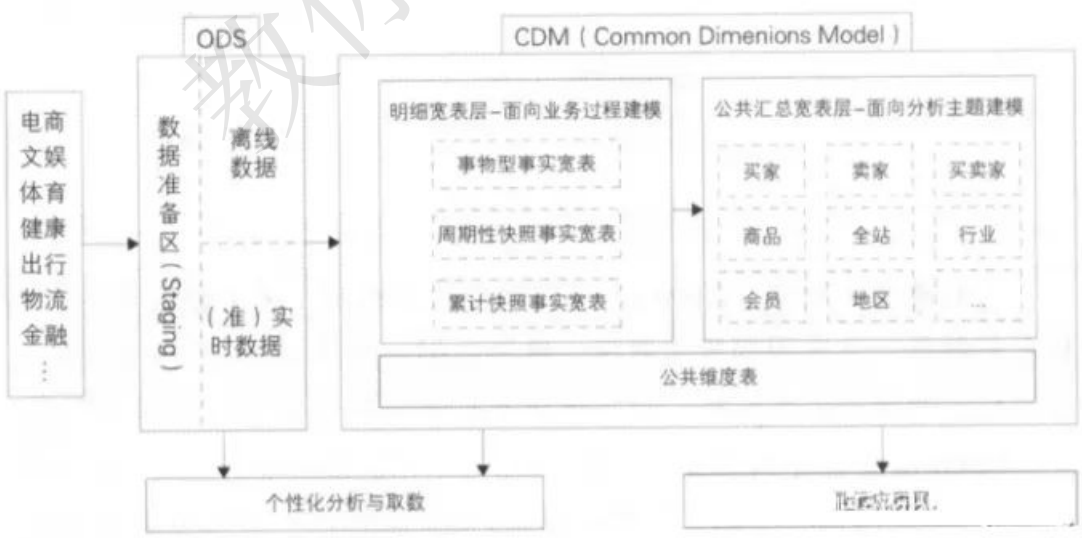
- 操作数据层（ODS）：把操作系统数据几乎无处理地存放在数据仓库系统中。
- 公共维度模型层（CDM）：存放明细事实数据、维表数据及公共指标汇总数据，其中明细事实数据、维表数据一般根据 ODS 层数据加工生成；公共指标汇总数据一般根据维表数据和明细事实数据加工生成。
- CDM 层又细分为 DWD 层和 DWS 层，分别是明细数据层和汇总数据层，采用维度模型方法作为理论基础，更多地采用一些维度退化手法， 将维度退化至事实表中，减少事实表和维表的关联，提高明细数据表的易用性；同时在汇总数据层， 加强指标的维度退化， 采取更多的宽表化手段构建公共指标数据层，提升公共指标的复用性，减少重复加工。其主要功能如下。

- 组合相关和相似数据：采用明细宽表，复用关联计算，减少数据扫描。
- 公共指标统一加工：基于 OneData体系构建命名规范、口径一致 和算法统一的统计指标，为上层数据产品、应用和服务提供公共指标建立逻辑汇总宽表。
- 建立一致性维度:建立一致的数据分析维表，降低数据计算口径、算法不统一的风险。

- 应用数据层（ADS）：存放数据产品个性化的统计指标数据，根据 CDM 层与 ODS 层加工生成 。
  - 个性化指标加工：不公用性、复杂性(指数型、比值型、排名型指标)。
  - 基于应用的数据组装 : 大宽表集市、横表转纵表、趋势指标串。



阿里巴巴通过构建全域的公共层数据，极大地控制了数据规模的增长趋势



模型架构图

- 数据调用服务优先使用公共维度模型层(CDM)数据，当公共层没有数据时，需评估是否需要创建公共层数据，当不需要建设公用的公共层时，方可直接使用操作数据层(ODS)数据。应用数据层(ADS) 作为产品特有的个性化数据一般不对外提供数据服务，但是 ADS 作为被服务方也需要遵守这个约定。

2.3.3 基本原则

- 高内聚和低耦合

一个逻辑或者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、粒度相同的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的核心业务，扩展模型包括的字段支持个性化或少量应用的需要，不能让扩展模型的字段过度侵入核心模型，以免破坏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

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。

- 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结果确定不变。

- 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。

- 命名清晰、可理解

表命名需清晰、一致，表名需易于消费者理解和使用。

2.4 模型实施

2.4.1 业界常用模型实施过程

构建维度模型一般要经历四个阶段：

- 第一个阶段是高层模型设计时期，定义业务过程维度模型的范围，提供每种星形模式的技术和功能描述；直接产出目标是创建高层维度模型图，它是对业务过程中的维表和事实表的图形描述。确定维表创建初始属性列表，为每个事实表创建提议度量；
- 第二个阶段是详细模型设计时期，对每个星形模型添加属性和度量信息；确定每个维表的属性和每个事实表的度量，并确定信息来源的位置、定义，确定属性和度量如何填入模型的初步业务规则。
- 第三个阶段是进行模型的审查、再设计和验证，本阶段主要召集相关人员进行模型的审查和验证，根据审查结果对详细维度进行再设计。
- 第四个阶段是产生详细设计文档，提交 ETL 设计和开发，最后，完成模型详细设计文档，提交 ETL 开发人员，进入 ETL 设计和开发阶段，由 ETL 人员完成物理模型的设计和开发。

2.4.2 OneData实施过程

1. 指导方针

- 首先，在建设大数据数据仓库时，要进行充分的业务调研和需求分析。这是数据仓库建设的基石，业务调研和需求分析做得是否充分直接决定了数据仓库建设是否成功。
- 其次，进行数据总体架构设计，主要是根据数据域对数据进行划分；按照维度建模理论，构建总线矩阵、抽象出业务过程和维度。
- 再次，对报表需求进行抽象整理出相关指标体系， 使用 OneData 工具完成指标规范定义和模型设计。
- 最后，就是代码研发和运维。

2. 实施工作流

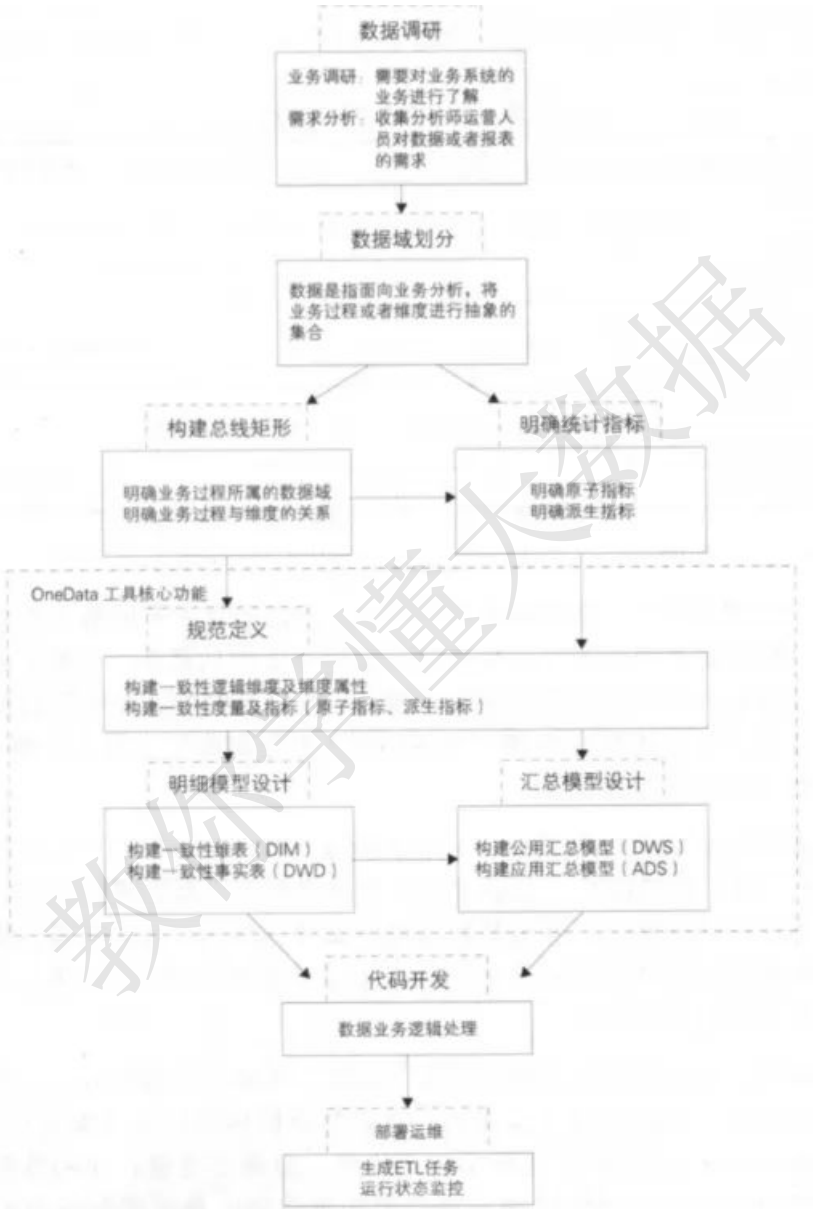


图 9.11　实施工作流

（1）数据调研

1. 业务调研：需要了解各个业务领域、业务线的业务有什么共同点和不同点，以及各个业务线可以细分为哪几个业务模块，每个业务模块具体的业务流程又是怎样的。业务调研是否充分，将会直接决定数据仓库 建设是否成功
2. 需求调研：需求调研的途径有两种：一是根据与分析师、业务运营人员的沟通（邮件、 IM ）获知需求；二是对报表系统中现有的报表进行研究分析；

（2）架构设计

1. 数据域划分

数据域是指面向业务分析，将业务过程或者维度进行抽象的集合。 业务过程可以概括为 一 个个不可拆分的行为事件，如下单、支付、退款。数据域需要抽象提炼，并且长期维护和更新，但不轻易变动。

表 9.5 数据域

数 据 域	业务过程举例
会员和店铺域	注册、登录、装修、开店、关店等
商品域	发布、上架、下架、重发、SKU 存量等
日志域	曝光、浏览、点击等
交易域	加购、下单、支付、退款、确认收货等
客服和销售域	拜访、培训、leads 管理等
工具和服务域	商品收藏、淘金币领用、优惠券领用、服务市场订购等
互动域	发帖、回帖、评论等
信用风控域	评价、申诉、投诉、纠纷、买家保障、认证等
采购分销域	商品采购（供应链管理）

2. 构建总线矩阵

在进行充分的业务调研和需求调研后，就要构建总线矩阵了。需要 做两件事情：明确每个数据域下有哪些业务过程；业务过程与哪些维度相关，并定义每个数据域下的业务过程和维度。

表 9.6 供应链管理业务过程示例

数据域	业务过程	一致性维度							
		供应商	业务类型	地区	仓库	类目	采购单	发货单	入库单
采购分销域	采购	√	√	√	√	√	√		
	发货	√	√	√	√	√		√	
	入库	√	√	√	√	√			

3. 规范定义

规范定义主要定义指标体系，包括原子指标、修饰词、时间周期和 派生指标。

4. 模型设计

模型设计主要包括维度及属性的规范定义，维表、明细事实表和汇 总事实表的模型设计。

5. 总结

OneData 的实施过程是一个高度迭代和动态的过程， 一般采用螺旋式实施方法。在总体架构设计完成之后，开始根据数据域进行迭代式模型设计和评审。在架构设计、规范定义和模型设计等模型实施过程中， 都会引入评审机制，以确保模型实施过程的正确性。

第3章 维度设计

3.1 维度设计基础

3.1.1 维度的基本概念

- 维度建模中，将度量称为“事实”，将环境描述为“维度”，维度是用于分析事实所需要的多样环境。例如，在分析交易过程时，可以通过买家、卖家、商品和时间等维度描述交易发生的环境。

- 维度所包含的表示维度的列，称为维度属性。维度属性是查询约束条件、分组和报表标签生成的基本来源，是数据易用性的关键。
- 维度使用主键标识其唯一性，主键也是确保与之相连的任何事实表 之间存在引用完整性的基础。

3.1.2 维度的基本设计方法

1. 选择维度或新建维度。须保证维度的唯一性。
2. 确定主维表。 一般是ODS表，直接与业务系统同步。
3. 确定相关维表。确定哪些表和主维表存在关联关系，并选择其中的某些表用于生成维度属性。
4. 确定维度属性。第一阶段从主维表中选择维度属性或生成新的维度属性；第二阶段是从相关维表中选择维度属性或生成新的维度属性。

确认维度属性的几点提示：

1. 尽可能生成丰富的维度属性
2. 尽可能多地给出包括一些富有意义的文字性描述
3. 区分数值型属性和事实

- 如果通常用于查询约束条件或分组统计，则是作为维度属性；如果通常 用于参与度量的计算，则是作为事实。比如商品价格，可以用于查询约 束条件或统计价格区间的商品数量，此时是作为维度属性使用的；也可 以用于统计某类目下商品的平均价格，此时是作为事实使用的。另外， 如果数值型字段是离散值，则作为维度属性存在的可能性较大；如果数 值型字段是连续值，则作为度量存在的可能性较大，但并不绝对，需要 同时参考字段的具体用途。

4. 尽量沉淀出通用的维度属性

3.1.3 一致性维度和交叉探查

- 共享维表。比如在阿里巴巴的数据仓库中，商品、卖家、买家、 类目等维度有且只有一个。所以基于这些公共维度进行的交叉探查不会存在任何问题。
- 一致性上卷。其中一个维度的维度属性是另一个维度的维度属性 的子集，且两个维度的公共维度属性结构和内容相同。比如在阿 里巴巴的商品体系中，有商品维度和类目维度，其中类目维度的 维度属性是商品维度的维度属性的子集，且有相同的维度属性和 维度属性值。这样基于类目维度进行不同业务过程的交叉探查也 不会存在任何问题。
- 交叉属性。两个维度具有部分相同的维度属性。比如在商品维度中具有类目属性，在卖家维度中具有主营类目属性，两个维度具有相同的类目属性，则可以在相同的类目属性上进行不同业务过程的交叉探查。

3.2 维度设计高级主题

3.2.1 维度整合

1. 应用间差异：

- 应用在编码、命名习惯、度量单位等方面会存在很大的差异。
- 应用出于性能和扩展性的考虑，或者随技术架构的演变，以及业务的发展，采用不同的物理实现。

2. 集成类型（同维度整合）：

- 命名规范的统一。表名、字段名等统一。
- 字段类型的统一。相同和相似字段的字段类型统一。
- 公共代码及代码值的统一。公共代码及标志性字段的数据类型、 命名方式等统一。
- 业务含义相同的表的统一。主要依据高内聚、低耦合的理念，在物理实现中，将业务关系大、源系统影响差异小的表进行整合： 将业务关系小、源系统影响差异大的表进行分而置之。通常有如下几种集成方式：
  - 采用主从表的设计方式，将两个表或多个表都有的字段放在主表中（主要基本信息），从属信息分别放在各自的从表中。 对于主表中的主键，要么采用复合主键、源主键和系统或表 区别标志：要么采用唯一主键、“源主键和系统或表区别标志”生成新的主键。通常建议采用复合主键的方式。
  - 直接合并，共有信息和个性信息都放在同一个表中。如果表字段的重合度较低，则会出现大量空值，对于存储和易用性会有影响，需谨慎选择。
  - 不合并，因为源表的表结构及主键等差异很大，无法合并，使用数据仓库里的多个表存放各自的数据。

3. 表整合：

- 垂直整合：不同的来源表包含相同的数据集，只是存储 的信息不同，比如主表与扩展表的整合，丰富其维度属性。
- 水平整合：不同的来源表包含不同的数据集，不同子集之间无交叉，也可以存在部分交叉。
  - 存在交叉，则需要去重
  - 不存在交叉，则需要考虑不同子集的自然键是否存在冲突
  - 如果不冲突， 则可以考虑将各子集的自然键作为整合后的表的自然键
  - 设置超自然键，将来源表各子集的自然键加工成一个字段作为超自然键（即联合主键，阿里采用该方法，并将来源字段作为分区字段）

3.2.2 水平拆分

如何设计维度：

- 模型设计重点考虑的三个原则：
  - 扩展性：当源系统、业务逻辑变化时，能通过较少的成本快速扩 展模型，保持核心模型的相对稳定性。软件工程中的高内聚、低藕合的思想是重要的指导方针之一。
  - 效能：在性能和成本方面取得平衡。通过牺牲一定的存储成本， 达到性能和逻辑的优化。
  - 易用性：模型可理解性高、访问复杂度低。用户能够方便地从模型中找到对应的数据表，并能够方便地查询和分析。
- 模型设计重点考虑的两个依据：
  - 维度的不同分类的属性差异情况。当维度属性随类型变化较大时，采用方案 1 。
  - 业务的关联程度。两个相关性较低的业务，稠合在一起弊大于利，对模型的稳定性和易用性影响较大，采用方案 2。

方案参考：

- 方案 1 是将维度的不同分类实例化为不同的维度，同时在主维度中保存公共属

性，适合于当维度属性随类型变化较大的情形

  - 构建商品维度、航旅商品维度：不同分类的商品，其维度属性可能相同，也可能不同。比如航旅的商品和普 通的淘系商品，都属于商品，都有商品价格、标题、类型、上架时间、 类目等维度属性，但是航旅的商品除了有这些公共属性外，还有酒店、 景点、门票、旅行等自己独特的维度属性。

- 方案 2 是维护单一维度，包含所有可能的属性
  - 对淘系商品和 1688 商品构建两个维度，业务分析人员一般只针对本数据集市进行统计分析。1688 业务变更，此维度需要变更， 淘宝业务变更亦然，稳定性很差。

3.2.3 垂直拆分

- 出于扩展性、产出时间、易用性等方面的考虑，设计主从维度。
- 主维表存放稳定、产出时间早、热度高的属性；
- 从维表存放变化较快、产 出时间晚、热度低的属性。
  - 设计了商品主维度和商品扩展维度。其中商品主维度在每日的 1：30 左右产出，而商 品扩展维度由于有冗余的产出时间较晚的商品品牌和标签信息，在每日 的 3：00 左右产出。
  - 由于商品扩展维度有冗余的库存等变化较快 的数据，对于主维度进行缓慢变化的处理较为重要。通过存储的冗余和计算成本的增加，实现了商品主模型的稳定和产出时间的提前。

3.2.4 历史归档

- 归档策略 1：同前台归档策略，在数据仓库中实现前台归档算法，定期对历史数据进行归档。但存在一些问题，一是前台归档策略复杂，实现成本较高；二是前台归档策略可能会经常变化，导致数据仓库归档算法也要随之变化，维护和沟通成本较高。此方式适用于前台归档策略 逻辑较为简单，且变更不频繁的情况。
- 归档策略 2 ：同前台归档策略，但采用数据库变更日志的方式。对 于如此庞大的数据量，阿里巴巴采用的数据抽取策略一般是通过数据库 binlog 日志解析获取每日增量，通过增量merge 全量的方式获取最新的 全量数据。可以使用增量日志的删除标志，作为前台数据归档的标志。 通过此标志对数据仓库的数据进行归档。此方式不需要关注前台归档策 略，简单易行。但对前台应用的要求是数据库的物理删除只有在归档时 才执行，应用中的删除只是逻辑删除。
- 归档策略 3 ：数据仓库自定义归档策略。可以将归档算法用简单、直接的方式实现，但原则是尽量比前台应用晚归档、少归档。避免出现数据仓库中已经归档的数据再次更新的情况。
- 如果技术条件允许，能够解析数据库 binlog 日志，建议使用归档策略 2 ，规避前台归档算法。具体可以根据自身数据仓库的实际情况进行选择。

3.3 维度变化

3.3.1 缓慢变化维

在 Kimball 的理论中，有三种处理缓慢变化维的方式，可以根据业务需求来进行选择：

1. 重写维度值。不保留历史数据， 始终取最新数据（假设业务需求方不关心历史数据，则可以采用方案1）
2. 插入新的维度行。保留历史数据，维度值变化前的事实和过去的维度值关联，

维度值变化后的事实和当前的维度值关联。

3. 添加维度列。采用第二种处理方式不能将变化前后记录的事实归一为变化前的维度或者归一为变化后的维度（不同业务部门需要统计各自的业绩，则需 要保留历史数据）

3.3.2 快照维表

- 在 Kimball 的维度建模中，必须使用代理键（不具有业务含义的键，区别于自然键）作为每个维表的主键，用于处理缓慢变化维。
- 阿里不使用代理键的原因：数据量大、ETL复杂化；不直接使用拉链表的原因：解释成本高、随着时间的推移，分区数量会极度膨胀
- 阿里通过快照方式，每天保留一份全量快照数据，简单而有效，方便好理解，但造成存储浪费，因此配合极限存储。

3.3.3 极限存储

1. 透明化

- 底层的数据还是历史拉链存储，但是上层做一个视图操作或者在 Hive 里做一个 hook ，通过分析语句的语法树，把对极限存储前的表的 查询转换成对极限存储表的查询。

```
Select * from A where ds =20160101 ;
```

等价于

```
Select * from A_EXST where start_dt <=20160101 and end_dt
>20160101;
```

2. 分月做历史拉链表

- 每个月月初重新开始做历史拉链表

局限性：首先，其产出效率很低，大部分极限存储通常需要 t-2 ； 其次，对于变化频率高的数据并不能达到节约成本的效果。

- 在做极限存储前有一个全量存储表，全量存储表仅保留最近一段 时间的全量分区数据，历史数据通过映射的方式关联到极限存储表。即用户只访问全量存储表，所以对用户来说极限存储是不可见的。
- 对于部分变化频率频繁的字段需要过滤。例如，用户表中存在用户积分字段，这种字段的值每天都在发生变化，如果不过滤的话，极限存储就相当于每个分区存储一份全量数据，起不到节约存储成本的效果。

3.3.4 微型维度

- 微型维度的创建是通过将一部分不稳定的属性从主维度中移出，并 将它们放置到拥有自己代理键的新表中来实现的。这些属性相互之间没有直接关联，不存在自然键。通过为每个组合创建新行的一次性过程来加载数据。
- 比如淘宝用户维度，用户的注册日期、年龄、性别、身份信息等基本不会发生变化，但用户 VIP 等级、用户信用评价等级会随着用户的行为不断发生变化。
- 其中 VIP 等级共有 8 个值，即 -1 ～6 ；用户信用评价等级共有 18 个值。假设基于 VIP 等级和用户信用评价等级构建微型维度，则在此微型维度中共有 8 x 18 个组合，即 144 条记录，代理键可能是 1～ 144

阿里在实践中并未使用此技术：

- 微型维度的局限性：必须是枚举值，且考虑所有可能组合
- ETL 逻辑复杂
- 破坏了维度的可浏览性

3.4 特殊维度

3.4.1 递归层次

- 维度的递归层次，按照层级是否固定分为均衡层次结构（如一级类目、二级类目等）和非均衡层次结构（如公司之间的公司，数量级别不固定）
- 递归 SQL 成本较高，且很多工具不支持递归SQL，因此在维度模型中对层次结构进行处理

1. 层次结构扁平化

表 10.13　数据存储示例 2

类目 ID	类目级别	是否叶子	一级类目	二级类目	三级类目
21	1	N	21	**21**	**21**
50026576	2	N	21	50026576	**50026576**
50026579	3	Y	21	50026576	**50026579**
121456022	2	Y	21	121456022	**121456022**
…	…	…	…		

扁平化仅包含固定数量的级别，对于非平衡层次结构，可以通过预留级别的方式来解决，但扩展性较差（图为阿里巴巴中文站的类目体系，粗体部分为回填内容）

2. 层次桥接表

- 解决了层次结构扁平化带来的一些问题，加工逻辑复杂，使用逻辑复杂，实际工作很少应用

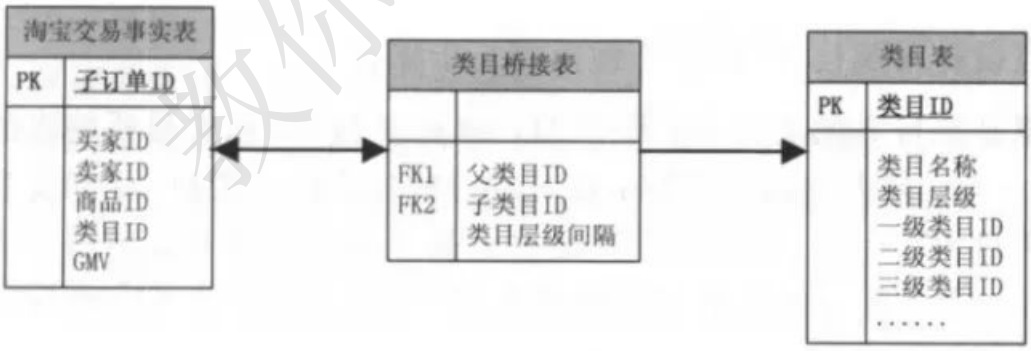


图 10.5　类目

3.4.2 行为维度

理解为事实衍生的维度，按照加工方式划分：

- 另一个维度的过去行为，如买家最近一次访问淘宝的时间、买家最近一次发生淘宝交易的时间等。
- 快照事实行为维度，如买家从年初截至当前的淘宝交易金额、买 家信用分值、卖家信用分值等。
- 分组事实行为维度，将数值型事实转换为枚举值。如买家从年初截至当前的淘宝交易金额按照金额划分的等级、买家信用分值按 照分数划分得到的信用等级

等。

- 复杂逻辑事实行为维度，通过复杂算法加工或多个事实综合加工得到。如前面提到的卖家主营类目，商品热度根据访问、收藏、 加入购物车、交易等情况综合计算得到。

对于行为维度，有两种处理方式，其中一种是将其冗余至现有的维表中，如将卖家信用等级冗余至卖家维表中另一种是加工成单独的行为维表，如卖家主营类目。具体采用哪种方式主要参考如下两个原则：

- 第一，避免维度过快增长。比如对商品表进行了极限存储，如果将 商品热度加入现有的商品维表中，则可能会使每日商品变更占比过高， 从而导致极限存储效果较差。
- 第二，避免耦合度过高。比如卖家主营类目，加工逻辑异常复杂， 如果融合进现有的卖家维表中，那么过多的业务稠合会导致卖家维表刷新逻辑复杂、维护性差、产出延迟等。

3.4.3 多值维度

- e.g. 交易父订单事实表 与 商品表

多值维度的处理方式

- 降低事实表的粒度（子订单建立事实）
- 采用多字段（售楼合同，多个买受方，已是最细粒度；由于个数不会太多，预留字段：买受方1，买受方2，买受方3）
- 桥接表：通过桥接表，则会产生多条重复记录，业务上注意区分重复计算是否符合业务逻辑

3.4.4 多值属性

e.g. 商品和 SKU、属性、标签都是多对多的关系

多值属性的处理方式：

- 保持维度主键不变，将多值属性放在维度的一个属性字段中（通过 k-v 对的形式放在 property 字段中，数据示例如下：10281239:156426871; 137396765:29229; 137400766:3226633）
- 保持维度主键不变，但将多值属性放在维度的多个属性字段中（卖家主营类目，只取TOP 3）
- 维度主键发生变化，一个维度值存放多条记录，扩展性好，使用方便（比如商品 SKU 维表，对于每个商品，有多少 SKU，就有多少记录，主键是商品的 ID 和 SKU 的 ID）

3.4.5 杂项维度

- 杂项维度是由操作型系统中的指示符或者标志字段组合而成的，一般不在一致性维度之列。
- 将这些字段建立到一个维表中，在事实表中只需保存一个外键即可。多个字段的不同取值组成 一条记录，生成代理键，存入维表中，并将该代理键保存到相应的事实表字段下。建议不要直接使用所有的组合生成完整的杂项维表，在抽取遇到新的组合时生成相应的记录即可。
- 阿里：存在非枚举字段，如交易留言、交易属性、交易标签等；通过子订单维度实现，且作为逻辑模型，不进行物理化。

第4章 事实表设计

4.1 事实表基础

4.1.1 事实表特性

- 事实表作为数据仓库维度建模的核心，紧紧围绕着业务过程来设计，通过获取描述业务过程的度量来表达业务过程，包含了引用的维度和与业务过程有关的度量。
- 事实表中一条记录所表达的业务细节程度被称为粒度。通常粒度可以通过两种方式来表述：一种是维度属性组合所表示的细节程度：一种是所表示的具体业务含义。
- 作为度量业务过程的事实，一般为整型或浮点型的十进制数值，有可加性、半可加性和不可加性三种类型。
  - 可加性事实是指可以按照与事实表关联的任意维度进行汇总。
  - 半可加性事实只能按照特定维度汇总， 不能对所有维度汇总，比如库存可以按照地点和商品进行汇总，而按时间维度把一年中每个月的库存累加起来则毫无意义。
  - 完全不具备可加性，比如比率型事实。对于不可加性事实可分解为可加的组件来实现聚集。
- 维度属性也可以存储到事实表中，这种存储到事实表中的维度列被称为“退化维度”。与其他存储在维表中的维度一样 ，退化维度也可以 用来进行事实表的过滤查询、实 现聚合操作等。
- 事实表有三种类型 : 事务事实表、周期快照事实表和累积快照事实表。
  - 事务事实表用来描述业务过程，跟踪空间或时间上某点的度量事件，保存的是最原子的数据，也称为“原子事实表“。
  - 周期快照事实表以具有规律性的、可预见的时间间隔记录事实 ，时间间隔如每天、每月、每年等。
  - 累积快照事实表用来表述过程开始和结束之间的关键步骤事件，覆盖过程的整个生命周期，通常具有多个日期字段来记录关键时间点，当过程随着生命周期不断变化时，记录也会随着过程的变化而被修改。

4.1.2 事实表设计原则

原则 1：尽可能包含所有与业务过程相关的事实

- 事实表设计的目的是为了度量业务过程，所以分析哪些事实与业务过程有关是设计中非常重要的关注点。在事实表中应该尽量包含所有与业务过程相关的事实，即使存在冗余，但是因为事实通常为数字型，带来的存储开销也不会很大。

原则 2：只选择与业务过程相关的事实

- 在选择事实时，应该注意只选择与业务过程有关的事实。比如在订单的下单这个业务过程的事实表设计中 ，不应该存在支付金额这个表示支付业务过程的事实。

原则 3：分解不可加性事实为可加的组件

对于不具备可加性条件的事实，需要分解为可加的组件。比如订单的优惠率，应该分解为订单原价金额与订单优惠金额两个事实存储在事实表中。

原则 4：在选择维度和事实之前必须先声明粒度

- 粒度的声明是事实表设计中不可忽视的重要一步，粒度用于确定事实表中一行所表示业务的细节层次，决定了维度模型的扩展性，在选择维度和事实之前必须先声明粒度，且每个维度和事实必须与所定义的粒度保持一致。在设计事实表的过程中，粒度定义得越细越好，建议从最低级别的原子粒度开始，因为原子粒度提供了最大限度的灵活性，可以 支持无法预期的各种细节层次的用户需求。在事实表中，通常通过业务描述来表述粒度，但对于聚集性事实表的粒度描述，可采用维度或维度属性组合的方式。

原则 5：在同一个事实表中不能有多种不同粒度的事实

事实表中的所有事实需要与表定义的粒度保持一致，在同一个事实表中不能有多种不同粒度的事实。

表 11.1　事务事实表

机票 ID	订单 ID	票支付金额	票折扣金额	订单支付金额	订单票数
23459801	100901	1000.00	100.00	3700.00	3
23459802	100901	1200.00	120.00	3700.00	3
23459803	100901	1500.00	150.00	3700.00	3
23459804	100902	1600.00	160.00	2600.00	2
23459805	100902	1000.00	100.00	2600.00	2
23459806	100903	1500.00	150.00	1500.00	1
.....					

原则 6：事实的单位要保持一致

对于同一个事实表中事实的单位，应该保持一致。比如原订单金额、 订单优惠金额、订单运费金额这三个事实，应该采用一致的计量单位， 统 一 为元或分，以方便使用。

原则 7：对事实的 null 值要处理

对于事实表中事实度量为 null 值的处理，因为在数据 库中 null 值 对常用数字型字段的 SQL 过滤条件都不生效，比如大于、小于、等于、 大于或等于、小于或等于，建议用零值填充。

原则 8：使用退化维度提高事实表的易用性

- 在 Kimball 的维度建模中，通常按照星形模型的方式来设计，对于 维度的获取采用的是通过事实表的外键关联专门的维表的方式，谨慎使 用退化维度。而在大数据领域的事实表设计中，则大量采用退化维度的 方式，在事实表中存储各种类型的常用维度信息。这样设计的目的主要 是为了减少下游用户使用时关联多个表的操作，直接通过退化维度实现 对事实表的过滤查询、控制聚合层次、排序数据以及定义主从关系等。 通过增加冗余存储的方式减少计算开销，提高使用效率。

4.1.3 事实表设计方法

对于维度模型设计采用四步设计方法：选择业务过程、声明粒度、确定维度、确定事实。

1. 选择业务过程及确定事实表类型

在明确了业务需求以后，接下来需要进行详细的需求分析，对业务 的整个生命周期进行分析，明确关键的业务步骤，从而选择与需求有关的业务过程。

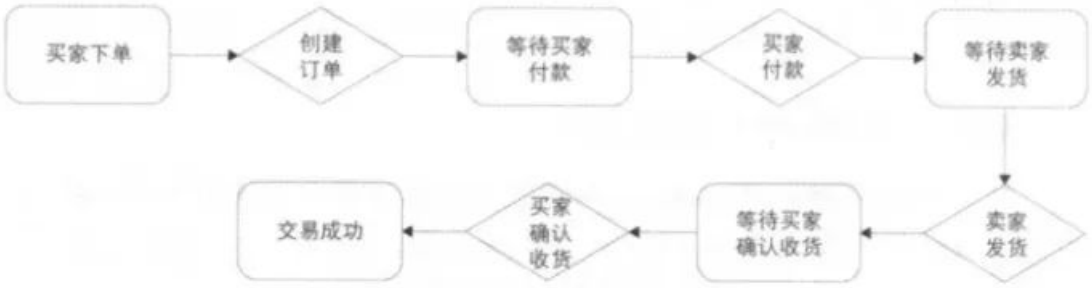


图 11.1 淘宝交易订单的流转过程

业务过程通常使用行为动词表示业务执行的活动。比如图 4.1中的淘宝订单流转的业务过程有四个：创建订单、买家付款、卖家发货、买家确认收货。在明确了流程所包含的业务过程后，需要根据具体的业务需求来选择与维度建模有关的业务过程。比如是选择买家付款这个业务过程，还是选择创建订单和买家付款这两个业务过程，具体根据业务情 况来确定。

在选择了业务过程以后，相应的事实表类型也随之确定了。比如选择买家付款这个业务过程，那么事实表应为只包含买家付款这一个业务过程的单事务事实表；如果选择的是所有四个业务过程，并且需要分析各个业务过程之间的时间间隔，那么所建立的事实表应为包含了所有四个业务过程的累积快照事实表。

2. 声明粒度

粒度的声明是事实表建模非常重要的一步，意味着精确定义事实表 的每一行所表示的业务含义，粒度传递的是与事实表度量有关的细节层次。明确的粒度能确保对事实表中行的意思的理解不会产生混淆，保证所有的事实按照同样的细节层次记录。

应该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。同时对于订单过程而言，粒度可以被定义为最细的订单级别。 比如在淘宝订单中有父子订单的概念，即一个子订单对应一种商品，如 果拍下了多种商品，则每种商品对应一个子订单:这些子订单一同结算 的话，则会生成一个父订单。那么在这个例子中，事实表的粒度应该选择为子订单级别。

3. 确定维度

完成粒度声明以后，也就意味着确定了主键，对应的维度组合以及相关的维度字段就可以确定了，应该选择能够描述清楚业务过程所处的环境的维度信息。比如在淘宝订单付款事务事实表中，粒度为子订单，相关的维度有买家、卖家、商品、收货人信息 、业务类型、订单时间等维度。

4. 确定事实

事实可以通过回答“过程的度量是什么”来确定。应该选择与业务 过程有关的所有事实，且事实的粒度要与所声明的事实表的粒度一致。 事实有可加性、半可加性、非可加性三种类型 ， 需要将不可加性事实分解为可加的组件。

比如在淘宝订单付款事务事实表中，同粒度的事实有子订单分摊的支付金额、邮费、优惠金额等。

5. 冗余维度

在传统的维度建模的星形模型中，对维度的处理是需要单独存放在专门的维表中的，通过事实表的外键获取维度。这样做的目的是为了减少事实表的维度冗余，从而减少存储消耗。而在大数据的事实表模型设计中，考虑更多的是提高下游用户的使用效率，降低数据获取的复杂性，减少关联的表数量。所以通常事实表中会冗余方便下游用户使用的常用维度，以实现对事实表的过滤查询、控制聚合层次、排序数据以及定义主从关系等操作。

比如在淘宝订单付款事务事实表中，通常会冗余大量的常用维度字段，以及商品类目、卖家店铺等维度信息。

4.2 事务事实表

订单作为交易行为的核心载体，直观反映了交易的状况。订单的流转会产生很多业务过程，而下单、支付和成功完结三个

业务过程是整个 订单的关键节点。获取这三个业务过程的笔数、金额以及转化率是日常 数据统计分析的重点，事务事实表设计可以很好地满足这个需求。本节 将介绍三种不同事务事实表的设计方式，以及在淘宝交易订单中关于邮 费和折扣分摊到子订单的算法。

4.2.1 设计过程

任何类型的事件都可以被理解为一种事务。比如交易过程中的创建 订单、买家付款，物流过程中的揽货、发货、签收，退款中的申请退 款、 申请小二介入等，都可以被理解为一种事务。事务事实表， 即针对这些过程构建的一类事实表，用以跟踪定义业务过程的个体行为，提供丰富的分析能力，作为数据仓库原子的明细数据。下面以淘宝交易事务事实表为例，阐述事务事实表的一般设计过程。

1. 选择业务过程

图 4.1 给出了淘宝交易订单的流转过程，其中介绍了四个重要过程：创建订单、买家付款、 卖家发货、买家 确认收货，即下单、支付 、 发货和成功完结四个业务过程。这四个业务过程不仅是交易过程中的重要时间节点 ，而且也是下游统计分析的重点，因此淘宝交易事务事实表 设计着重从这四个业务过程进行展开 。

Kimball 维度建模理论认为，为了便于进行独立的分析研究，应该为每个业务过程建立一个事实表。 对于是否将不同业务过程放到同一个事实表 中，将在下一节中详细介绍。

2. 确定粒度

业务过程选定以后，就要针对每个业务过程确定一个粒度，即确定事务事实表每一行所表达的细节层次。下面先介绍淘宝订单的产生过程。

对于每一种商品产生的订单就称为子订单，子订单记录了父订单的订单号，并且有子订单标志。 如果在同一个店铺只购买了一种商品，则会将父子订单进行合并，只保留一条订单记录。如图 4.2 和图 4.3 所示示例。

订单ID	父订单ID	创建时间	买家ID	卖家ID	商品ID	金额	数量	是否子订单	是否父订单
1	1	2016-01-01 00:00:00	111	222	11111	100.0	1	Y	Y
9	9	2016-01-03 00:00:00	444	555	66666	500.0	2	Y	Y

图 11.2 父子订单合并成一条记录

订单ID	父订单ID	创建时间	买家ID	卖家ID	商品ID	金额	数量	是否子订单	是否父订单
2	0	2016-01-01 00:00:00	111	222	0	0.0	3	N	Y
4	2	2016-01-01 00:00:00	111	222	11111	100.0	1	Y	N
5	2	2016-01-01 00:00:00	111	222	22222	50.0	2	Y	N
3	0	2016-01-02 00:00:00	333	777	0	0.0	6	N	Y
6	3	2016-01-02 00:00:00	333	777	33333	70.0	1	Y	N
7	3	2016-01-02 00:00:00	333	777	44444	80.0	2	Y	N
8	3	2016-01-02 00:00:00	333	777	55555	90.0	3	Y	N

图 11.3 父子订单拆开成多条记录

卖家发货这个业务过程可以选择子订单粒度，即将每个子订 单作为卖家发货事实表的一个细节。然而，在实际操作中发现，卖家发货更多的是物流单粒度而非子订单粒度，同 一个子订单可以拆开成多个物流单进行发货。在事务事实表设 计过程中，秉承确定为最细粒度的原则，因此对于卖家发货确定为物流单粒度，和其他三个业务过程不同，这样可以更好地给下游统计分析带来灵活性。

3. 确定维度

选定好业务过程并且确定粒度后，就可以确定维度信息了。在淘宝交易事务事实表设计过程中，按照经常用于统计分析的场景，确定维度包含:买家、卖家、商品、商品类目、发货地区、收货地区、父订单维度以及杂 项维度。由于订单的属性较多，比如订单的业务类型、是否无线交易、订单的 attributes 属性等，对于这些使用较多却又无法归属到上述买卖家或商品维度中的属性，则新建一个杂项维度进行存放，如图 4.4所示。

图 11.4 淘宝交易事务事实表（确定粒度

4. 确定事实

作为过程度量的核心，事实表应该包含与其描述过程有关的所有事实。以淘宝交易事务事实表为例，选定三个业务过程一一下单、支付和 成功完结，不同的业务过程拥有不同的事实。比如在下单业务过程中， 需要包含下单金额、下单数量、下单分摊金额；在支付业务过程中，包含支付金额、分摊邮费、折扣金额、红包金额、积分金额；在完结业务过程中包含确认收货金额等。由于粒度是子订单，所以对于一些父订单上的金额需要分摊到子订单上，比如父订单邮费、父订单折扣等。

5. 冗余维度

在确定维度时，包含了买卖家维度、商品维度、类目维度 、收发货维度等， Kimball维度建模理论建议在事实表中只保存这些维表的外键， 而淘宝交易事务事实表在 Kimball 维度建模基础之上做了进 一 步的优 化，将买卖家星级、标签、店铺名称、商品类型、商品特征、商品属性、 类目层级等维度属性都冗余到事实表中，提高对事实表进行过滤查询、统计聚合的效率，如图 4.5 所示 。

图 11.5 冗余维度的淘宝交易事务事实表

4.2.2 单事务事实表

单事务事实表，顾名思义，即针对每个业务过程设计一个事实表。这样设计的优点不言而喻，可以方便地对每个业务过程进行独立的分析研究。1688 交易流程则采用这 种模式构建事务事实表。

1688 交易和淘宝交易相 似，主要流程也是下单、支付、发货和完结，而在这四个关键流程中 1688 交易选择下单和支付两个业务过程设 计事务事实表，分别是1688交易订单下单事务事实表和1688交易订单支付事务事实表。

选定业务过程后，将对每个业务过程确定粒度、维度和事实。对于 1688 交易订单下单事务事实表，确定子订单粒度，选择买家、卖家、商品、父订单、收货地区维度，事实包含下单分摊金额和折扣金额，如 图4.6 所示；而对于 1688 交易订单支付事务事实表 ，粒度和维度与交易订单下单事务事实表相同，所表达的事实则不 一样 ，包含支付金额、支付调整金额和支付优惠等，如图4.7 所示；

图 11.6 1688 交易订单下单事务事实表 图 11.7 1688交易订单支付事务事实表

1688 交易针对下单和支付分别建立单事务事实表后，每天的下单记录则进入当天的下单事务事实表中，每天的支付记录进入当天的支付事务事实表中，由于事实表具有稀疏性质，因此只有当天数据才会进入当天的事实表中。下面以具体交易订单为例，展示单事务事实表的设计实例。

订单ID	父订单ID	下单时间	支付时间	完结时间	买家ID	卖家ID	商品ID	下单度量	支付度量	是否子订单	是否父订单
order1	mord1	2016-01-01 08:00:00	2016-01-01 10:00:00	2016-01-02 10:00:00	111	222	aa	...	...	Y	Y
order2	mord2	2016-01-01 09:00:00	2016-01-02 09:00:00	2016-01-04 09:00:00	333	444	0	...	...	N	Y
order3	mord2	2016-01-01 09:00:00	2016-01-02 09:00:00	2016-01-04 09:00:00	333	444	bb	...	...	Y	N
order4	mord2	2016-01-01 09:00:00	2016-01-02 09:00:00	2016-01-04 09:00:00	333	444	cc	...	...	Y	N

图 11.8　1688 交易订单详情实例

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

图 11.9　1688 交易订单下单事务事实表数据实例

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	...

图 11.10　1688 交易订单支付事务事实表数据实例

4.2.3 多事务事实表

多事务事实表，将不同的事实放到同一个事实表中，即同一个事实表包含不同的业务过程。多事务事实表在设计时有两种方法进行事实的处理：

- ①不同业务过程的事实使用不同的事实字段进行存放；
- ②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。

如何选择：

- 当不同业务过程的度量比较相似、差异不大时，可以采用第 二种 多事务事实表的设计方式，使用同 一个字段来表示度量数据 。但 这种方式存在一个问题一一在同一个周期内会存在多条记录（如淘宝收藏商品事务事实表，增加【收藏删除类型】，collect/delete）
- 当不同业务过程的度量差异较大时，可以选择第一种多事务事实 表的设计方式，将不同业务过程的度量使用不同 字段冗余到 表 中，非当前业务过程则置零表示。这种方式所存在的问题是度量字段零值较多（如淘宝交易事务事实表，针对不同业务过程如下单，则打一个是否当天下单的标签）

4.2.4 两种事实表对比

1. 业务过程

对于单事务事实表，一个业务过程建立一个事实表，只反映一个业务过程的事实；对于多事务事实表，在同 一个事实表中反映多个业务过程。多个业务过程是否放到同一 个事实表中，首先需要分析不同业务过 程之间的相 似性和业务源系统。比如淘宝交易的下单、支付和成功完结 这三个业务过程是存在相似性的，都属于订单处理中的 一环，并且都来自于交易系统，因此适合放到同 一个事务事实表中。

2. 粒度和维度

在考虑是采用单事务事实表还是多事务事实表时，另一个关键点就是粒度和维度，在确定好业务过程后，需要基于不同的业务过程确定粒度和维度，当不同业务过程的粒度相同，同时拥有相似的维度时，此时就可以考虑采用多事务事实表。 如果粒度不同，则必定是不同的事实表。 比如交易中支付和发货有不同的粒度，则无法将发货业务过程放到淘宝交易事务事实表中。

3. 事实

对于不同的业务过程，事实往往是不同的，单事务事实表在处理事实上比较方便和灵活，仅仅体现同一个业务过程的事实即可， 而多事务事实表由于有多个业务过程， 所以有更多的事实需要处理。如果单一业务过程的事实较多，同时不同业务过程的事实又不相同，则可以考虑使 用单事务事实表，处理更加清晰 ; 若使用多事务事实表， 则会导致事实表零值或空值字段较多。

4. 下游业务使用

单事务事实表对于下游用户而言更容易理解 ， 关注哪个业务过程就使用相应的事务事实表；而多事务事实表包含多个业务过程，用户使用时往往较为困惑。1688 和淘宝交易分别采用了这两种方式，从日常使用来看，对于淘宝交易事务事实表下游用户确实有 一定的学习成本 。

5. 计算存储成本

针对多个业务过程设计事务事实表，是采用单事务事实表还是多事务事实表，对于数据仓库的计算存储成本也是参考点之一 ，当业务过程 数据来源于同 一个业务系统，具有相同的粒度和维度，且维度较多而事实相对不多时，此时可以考虑使用多事务事实表，不仅其加工计算成本 较低，同时在存储上也相对节省，是一种较优的处理方式。

表 11.2　单事务事实表和多事务事实表的比较

	单事务事实表	多事务事实表
业务过程	一个	多个
粒度	相互间不相关	相同粒度
维度	相互间不相关	一致
事实	只取当前业务过程中的事实	保留多个业务过程中的事实，非当前业务过程中的事实需要置零处理
冗余维度	多个业务过程，则需要冗余多次	不同的业务过程只需要冗余一次
理解程度	易于理解，不会混淆	难以理解，需要通过标签来限定
计算存储成本	较多，每个业务过程都需要计算存储一次	较少，不同业务过程融合到一起，降低了存储计算量，但是非当前业务过程的度量存在大量零值

4.2.5 父子事实的处理方式

- e.g. 子订单分摊的有效下单金额和支付金额

4.2.6 事实的设计准则

1. 事实完整性：尽可能多地获取所有的度量
2. 事实一致性：事实表中统一计算可以保证度量的一致性（比如金额由数量*单价先在事实表算出来）
3. 事实可加性：事务事实表中关注 更多的是可加性事实，下游用户在聚合统计时更加方便

4.3 周期快照事实表

- 状态度量，比如 账户余额、买卖家星级、商品库存、卖家累积交易额等

- 无法聚集，比如温度等
- 简称“快照事实表”：在确定的间隔内对实体的度量进行抽样，这样可以很容易地研究实体的度量值，而 不需要聚集长期的事务历史

4.3.1 特性

1. 用快照采样状态

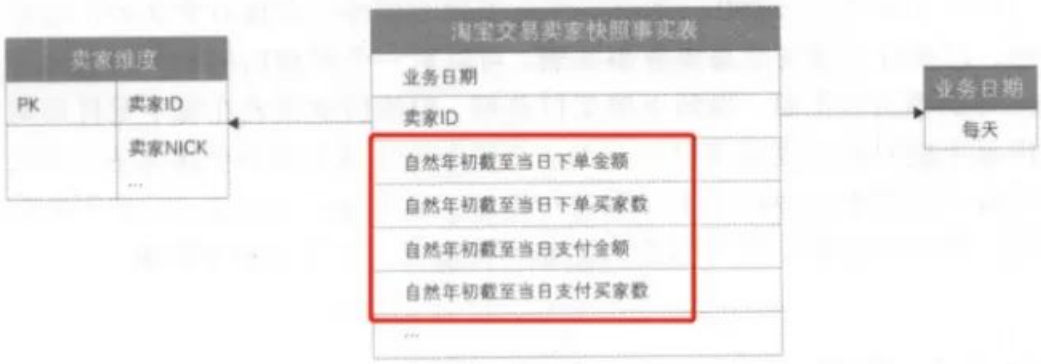


图 11.16 淘宝交易卖家快照事实

2. 快照粒度

- 快照需要采样的周期以及什么将被采样
- e.g. 淘宝交易有针对卖家加类目的每月汇总事实表，每月统计一次， 同时维度也不仅一个，包含了卖家和类目。

3. 密度与稀疏性

- e.g. 针对卖家的历史至今的下单和支付金额，无论 当天卖家是否有下单支付事实，都会给该卖家记录一行。

4. 半可加性

- 半可加性事实不能根据时间维度获得有意义的汇总结 果
- 虽然不能汇总，但可以计算一些平均值

4.3.2 实例

- 单维度的每天快照事实表
- 混合维度的每天快照事实表
- 直接使用操作型系统的数据作为周期快照事实表的数 据源进行加工，e.g. 淘宝卖家信用分 / DSR 快照事实表 / 货值拍照表
- 全量快照事实表：e.g. 淘宝好中差评快照事实表，无事实的事实表，更多关注评价的状态

4.3.3 注意事项

1. 事务与快照成对设计

- 数据仓库维度建模时，对于事务事实表和快照事实表往往都是成对 设计的，互相补充，以满足更多的下游统计分析需求，特别是在事务事实表的基础上可以加工快照事实表，如前面所述的淘宝卖家历史至今快 照事实表，就是在事务事实表的基础上加工得到的，既丰富了星形模型， 又降低了下游分析的成本。

2. 附加事实

- 快照事实表在确定状态度量时，一般都是保存采样周期结束时的状态度量。但是也有分析需求需要关注上一个采样周期结束时的状态度量，而又不愿意多次

使用快照事实表，因此一般在设计周期快照事实表 时会附加一些上一个采样周期的状态度量。

3. 周期到日期度量

- 在介绍淘宝卖家历史至今快照事实表时，指定了统计周期是卖家历史至今的一些状态度量，比如历史截至当日的下单金额、成交金额等。然 而在实际应用中，也有需要关注自然年至今、季度至今、财年至今的一些状态度量，因此在确定周期快照事实表的度量时，也要考虑类似的度量值， 以满足更多的统计分析需求。阿里巴巴数据仓库在设计周期快照事实表 时，就针对多种周期到日期的度量设计了不同的快照事实表，比如淘宝 卖家财年至今的下单金额、淘宝商品自然年至今的收藏次数等。

4.4 累积快照事实表

- 研究事件之间时间间隔的需求
- 保存全量数据，存放加工后的事实，并将各维度常用属性和订单 杂项维度退化到此表中
- e.g. 统计买家下单到支付的时长、买家支付到卖家发货的时长、买家从下单到确认收货的时长等

4.4.1 设计过程

- 累积快照事实表解决的最重要的问题 是统计不同业务过程之间的时间间隔，建议将每个过程的时间间隔作为事实放在事实表中（设计过程同4.2.1）

4.4.2 特点

1. 数据不断更新
2. 多业务过程日期

4.4.3 特殊处理

1. 非线性过程

- 业务过程的统一
- 针对业务关键里程碑构建全面的流程
- 循环流程的处理：主要问题是解决一个业务过程存在多个里程碑日期的问题。使用业务过程第一次发生的日期还是最后一次发生的日期，决定权在商业用户，而不是设计或开发人员。

2. 多源过程

- 针对多源业务建模，主要考虑事实表的粒度问题。对于退款，由于每个子订单可能 存在多次退款，此时如果要将退款相关业务过程加入模型中，则需要和 商业用户确定存在多次退款时如何取舍，确保模型粒度不变。

3. 业务过程取舍

- 对于多源业务过程，模型的精合度过高，此时需要根据商业用户需求，选取关键的里程碑。

4.4.4 物理实现

1. 全量表的形式：此全量表一般为日期分区表，每天的 分区存储昨天的全量数据和当天的增量数据合并的结果，保证每条记录 的状态最新。此种方式适用于全量数据较少的情况。

2. 全量表的变化形式：比如针对交易订单，我们以 200 天作为订单从产生到消亡的最大间隔。设计最近 200 天的交易订单累积快照事实表，每天的分 区存储最近 200 天的交易订单而 200 天之前的订单则按照 gmt_create 创 建分区存储在归档表中。
3. 业务实体的结束时间分区：每天的分区存放当天结 束的数据，设计一个时间非常大的分区，比如 3000-12-31 ，存放截至当前未结束的数据。由于每天将当天结束的数据归档至当天分区中，时间 非常大的分区数据量不会很大， ETL 性能较好；并且无存储的浪费，对于业务实体的某具体实例，在该表的全量数据中唯一。但接口等原因，存在结束标志的确认问题，有以下两个方案：

- 使用相关业务系统的业务实体的结束标志作为此业务 系统的结束标志。
- 和前端业务系统确定口径或使用前端归档策略。

4.5 三种事实表的比较

表 11.7　三种事实表的比较

	事务事实表	周期快照事实表	累积快照事实表
时期/时间	离散事务时间点	以有规律的、可预测的间隔产生快照	用于时间跨度不确定的不断变化的工作流
日期维度	事务日期	快照日期	相关业务过程涉及的多个日期
粒度	每行代表实体的一个事务	每行代表某时间周期的一个实体	每行代表一个实体的生命周期
事实	事务事实	累积事实	相关业务过程事实和时间间隔事实
事实表加载	插入	插入	插入与更新
事实表更新	不更新	不更新	业务过程变更时更新

4.6 无事实的事实表

- 事件类的，记录事件的发生。比如用户的浏览日志。
- 条件、范围或资格类的，记录维度与维度多对多之间的关系。比如客户和销售人员的分配情况、产品的促销范围等。

4.7 聚集型事实表DWS

- 公共汇总层：比如卖家最近 1 天的交易汇总表、卖家最近 N 天的交易汇总表、卖家自然年交易汇总表等。

4.7.1 聚集的基本原则

- 一致性。表必须提供与查询明细粒度数据一致的查询结果。
- 避免单一表设计。不要在同一个表中存储不同层次的聚集数据； 否则将会导致双重计算或出现更糟糕的事情。在聚集表中有些行 存放按天汇总的交易额，有些行存放按月汇总的交易额，这将会让使用者产生误用导致重复计算。行之有效的另一种方法是把按天与按月汇总的交易额用两列存放，但是需要在列名或者列注释上能分辨出来。
- 聚集粒度可不同。聚集并不需要保持与原始明细粒度数据一样的粒度，聚集只关心所需要查询的维度。

4.7.2 聚集的基本步骤

1. 确定聚集维度

在原始明细模型中会存在多个描述事实的维度，如日期、商品类别、 卖家等，这时候需要确定根据什么维度聚集，如果只关心商品的交易额 情况，那么就可以根据商品维度聚集数据。

2. 确定一致性上钻

- 这时候要关心是按月汇总还是按天汇总，是按照商品汇总还是按照 类目汇总，如果按照类目汇总，还需要关心是按照大类汇总还是小类汇 总。当然，我们要做的只是了解用户需要什么，然后按照他们想要的进行聚集。

3. 确定聚集事实

- 在原始明细模型中可能会有多个事实的度量，比如在交易中有交易 额、交易数量等，这时候要明确是按照交易额汇总还是按照成交数量汇总。

4.7.3 阿里公共汇总层

1. 基本原则

- 数据公用性
- 不跨数据域
- 区分统计周期：在表的命名上要能说明数据的统计周期，如 1d 表示最近 1 天，td 表示截至当天， nd 表示最近 N 天

2. 交易汇总表设计

- 最近1天商品粒度汇总表
- 最近N天卖家粒度汇总表
- 最近1天卖家、买家、商品粒度汇总表
- 最近1天二级类目汇总表

4.7.4 聚集补充说明

- 聚集是不跨越事实的：横向钻取是针对多个事实基于一致性维度进行的分析，很多时候采用融合事实表，预先存放横向钻取的结果，从而提高查询性能。
- 聚集带来的问题：当子类目对应的一级类目发生变更时，先前存在的、已经被汇总到聚集表中的数据需要被重新调整。这一额外工作随着业务复杂性的增加，会导致多数 ETL 人员选择简单强力的方法，删除并重新聚集数据。