尚硅谷大数据之电商实时数仓

之

数仓概述以及建模理论

（作者：尚硅谷研究院）

版本：V3.0

第 1 章　数据仓库概述

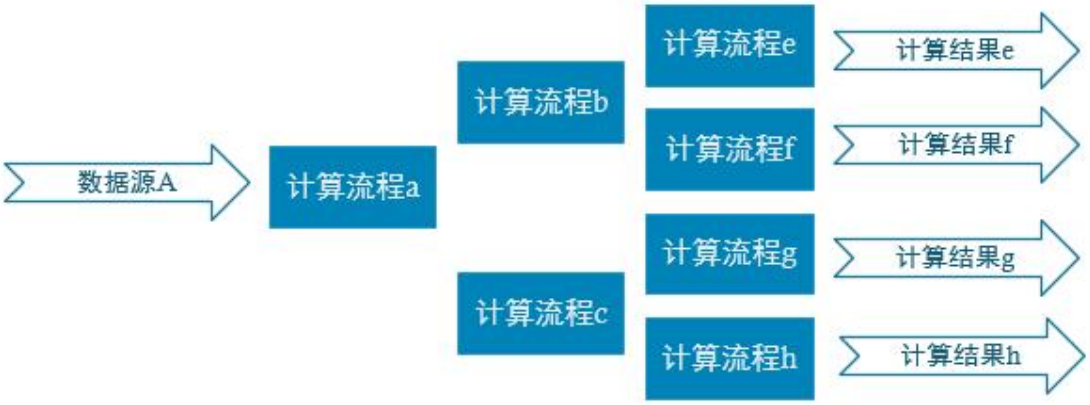
1.1 数据仓库概念

数据仓库是一个为数据分析而设计的企业级数据管理系统。数据仓库可集中、整合多个信息源的大量数据，借助数据仓库的分析能力，企业可从数据中获得宝贵的信息进而改进决策。同时，随着时间的推移，数据仓库中积累的大量历史数据对于数据科学家和业务分析师也是十分宝贵的。

1.2 普通实时计算与实时数仓比较

普通的实时计算优先考虑时效性，所以从数据源采集经过实时计算直接得到结果。如此做时效性更好，但是弊端是由于计算过程中的中间结果没有沉淀下来，所以当面对大量实时需求的时候，计算的复用性较差，开发成本随着需求增加直线上升。

实时数仓基于一定的数据仓库理念，对数据处理流程进行规划、分层，目的是提高数据的复用性。



1.3 离线计算与实时计算的比较

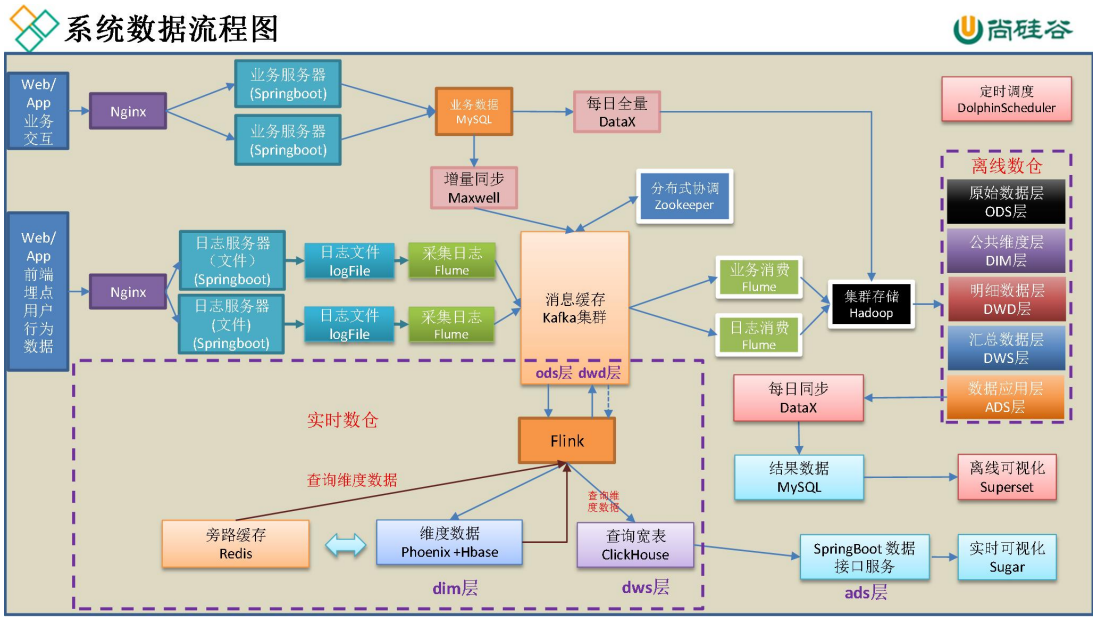
离线计算：就是在计算开始前已知所有输入数据，输入数据不会产生变化，一般计算量级较大，计算时间也较长。例如今天早上一点，把昨天累积的日志，计算出所需结果。最经典的就是 Hadoop 的 MapReduce 方式；

一般是根据前一日的数据生成报表，虽然统计指标、报表繁多，但是对时效性不敏感。从技术操作的角度，这部分属于批处理的操作。即根据确定范围的数据一次性计算。

实时计算：输入数据是可以以序列化的方式一个个输入并进行处理的，也就是说在开始的时候并不需要知道所有的输入数据。与离线计算相比，运行时间短，计算量级相对较小。强调计算过程的时间要短，即所查当下给出结果。

主要侧重于对当日数据的实时监控，通常业务逻辑相对离线需求简单一下，统计指标也少一些，但是更注重数据的时效性，以及用户的交互性。从技术操作的角度，这部分属于流处理的操作。根据数据源源不断地到达进行实时的运算。

1.4 离线和实时数仓架构



第 2 章　数据仓库建模概述

2.1 数据仓库建模的意义

如果把数据看作图书馆里的书，我们希望看到它们在书架上分门别类地放置；如果把数据看作城市的建筑，我们希望城市规划布局合理；如果把数据看作电脑文件和文件夹，我们希望按照自己的习惯有很好的文件夹组织方式，而不是糟糕混乱的桌面，经常为找一个文件而不知所措。

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。只有将数据有序的组织和存储起来之后，数据才能得到高性能、低成本、高效率、高质量的使用。

高性能：良好的数据模型能够帮助我们快速查询所需要的数据。

低成本：良好的数据模型能减少重复计算，实现计算结果的复用，降低计算成本。

高效率：良好的数据模型能极大的改善用户使用数据的体验，提高使用数据的效率。

高质量：良好的数据模型能改善数据统计口径的混乱，减少计算错误的可能性。

2.2 数据仓库建模方法论

2.2.1 ER 模型

数据仓库之父 Bill Inmon 提出的建模方法是从全企业的高度，用实体关系（Entity Relationship，ER）模型来描述企业业务，并用规范化的方式表示出来，在范式理论上符合 3NF。

(1) 实体关系模型

实体关系模型将复杂的数据抽象为两个概念——实体和关系。实体表示一个对象，例如学生、班级，关系是指两个实体之间的关系，例如学生和班级之间的从属关系。

(2) 数据库规范化

数据库规范化是使用一系列范式设计数据库（通常是关系型数据库）的过程，其目的是减少数据冗余，增强数据的一致性。

这一系列范式就是指在设计关系型数据库时，需要遵从的不同的规范。关系型数据库的范式一共有六种，分别是第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF）、第四范式(4NF）和第五范式（5NF）。遵循的范式级别越高，数据冗余性就越低。

(3) 三范式

- 第一范式

第一范式

1、第一范式1NF核心原则就是：属性不可切割

表 不符合一范式的表格设计

ID	商品	商家ID	用户ID
001	5 台电脑	XXX旗舰店	00001

很明显上图所示的表格设计是不符合第一范式的，商品列中的数据不是原子数据项，是可以进行分割的，因此对表格进行修改，让表格符合第一范式的要求，修改结果如下图所示：

表 符合一范式的表格设计

ID	商品	数量	商家ID	用户ID
001	电脑	5	XXX旗舰店	00001

实际上，1NF是所有关系型数据库的最基本要求，你在关系型数据库管理系统（RDBMS），例如SQL Server，Oracle，MySQL中创建数据表的时候，如果数据表的设计不符合这个最基本的要求，那么操作一定是不能成功的。也就是说，只要在RDBMS中已经存在的数据表，一定是符合1NF的。

- 第二范式

第二范式

2、第二范式2NF核心原则：不能存在“部分函数依赖”

学号	姓名	系名	系主任	课名	分数
1022211101	李小明	经济系	王强	高等数学	95
1022211101	李小明	经济系	王强	大学英语	87
1022211101	李小明	经济系	王强	普通化学	76
1022211102	张莉莉	经济系	王强	高等数学	72
1022211102	张莉莉	经济系	王强	大学英语	98
1022211102	张莉莉	经济系	王强	计算机基础	88
1022511101	高芳芳	法律系	刘玲	高等数学	82
1022511101	高芳芳	法律系	刘玲	法学基础	82

以上表格明显存在，部分依赖。比如，这张表的主键是（学号，课名），分数确实完全依赖于(学号，课名），但是姓名并不完全依赖于(学号，课名）

学号	课名	分数
1022211101	高等数学	95
1022211101	大学英语	87
1022211101	普通化学	76
1022211102	高等数学	72
1022211102	大学英语	98
1022211102	计算机基础	88
1022511101	高等数学	82
1022511101	法学基础	82

学号	姓名	系名	系主任
1022211101	李小明	经济系	王强
1022211102	张莉莉	经济系	王强
1022511101	高芳芳	法律系	刘玲

以上符合第二范式，去掉部分函数依赖依赖

- 第三范式

第三范式

尚硅谷

3、第三范式3NF核心原则：不能存在传递函数依赖

在下面这张表中，存在传递函数依赖：学号->系名->系主任，但是系主任推不出学号。

学号	姓名	系名	系主任
1022211101	李小明	经济系	王强
1022211102	张莉莉	经济系	王强
1022511101	高芳芳	法律系	刘玲

上面表需要再次拆解：

学号	姓名	系名
1022211101	李小明	经济系
1022211102	张莉莉	经济系
1022511101	高芳芳	法律系

系名	系主任
经济系	王强
法律系	刘玲

- 函数依赖

函数依赖

尚硅谷

学号	姓名	系名	系主任	课名	分数
1022211101	李小明	经济系	王强	高等数学	95
1022211101	李小明	经济系	王强	大学英语	87
1022211101	李小明	经济系	王强	普通化学	76
1022211102	张莉莉	经济系	王强	高等数学	72
1022211102	张莉莉	经济系	王强	大学英语	98
1022211102	张莉莉	经济系	王强	计算机基础	88
1022511101	高芳芳	法律系	刘玲	高等数学	82
1022511101	高芳芳	法律系	刘玲	法学基础	82

1、完全函数依赖：

设X，Y是关系R的两个属性集合，X'是X的真子集，存在X→Y，但对每一个X'都有X'!→Y，则称Y完全函数依赖于X。记做：$X \xrightarrow{F} Y$

人类语言：

比如通过，(学号，课程) 推出分数 ，但是单独用学号推断不出来分数，那么就可以说：分数 完全依赖于(学号，课程) 。

即：通过AB能得出C，但是AB单独得不出C，那么说C完全依赖于AB。

2、部分函数依赖

假如Y函数依赖于X，但同时Y并不完全函数依赖于X，那么我们就称Y部分函数依赖于X，记做：$X \xrightarrow{P} Y$

人类语言：

比如通过，(学号，课程) 推出姓名，因为其实直接可以通过，学号推出姓名，所以：姓名 部分依赖于 (学号，课程)

即：通过AB能得出C，通过A也能得出C，或者通过B也能得出C，那么说C部分依赖于AB。

3、传递函数依赖

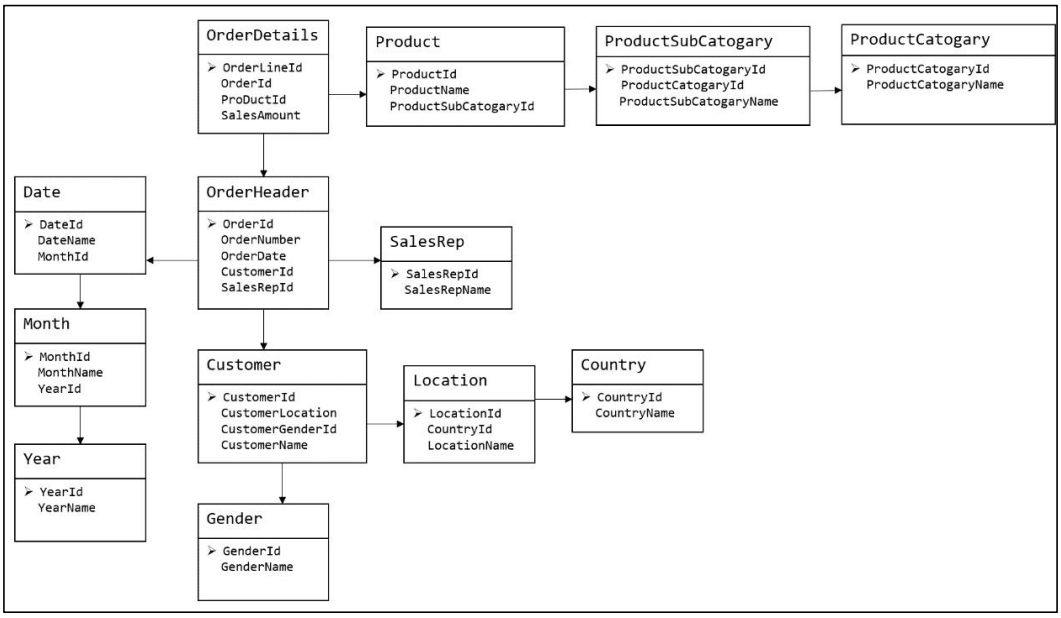
传递函数依赖：设X，Y，Z是关系R中互不相同的属性集合，存在X→Y(Y !→X),Y→Z，则称Z传递函数依赖于X。记做：$X \xrightarrow{T} Z$

人类语言：

比如：学号 推出 系名 ， 系名 推出 系主任， 但是，系主任推不出学号，系主任主要依赖于系名。这种情况可以说：系主任 传递依赖于 学号

通过A得到B，通过B得到C，但是C得不到A，那么说C传递依赖于A。

下图为一个采用 Bill Inmon 倡导的建模方法构建的模型，从图中可以看出，较为松散、零碎，物理表数量多。



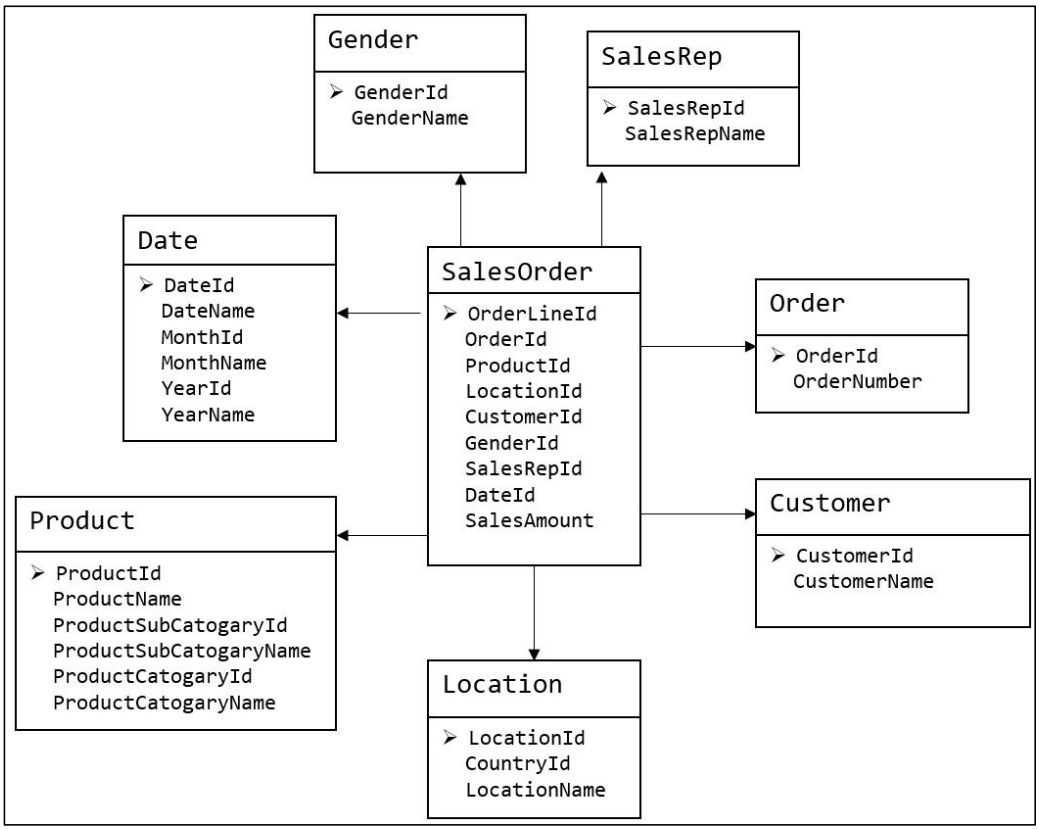
这种建模方法的出发点是整合数据，其目的是将整个企业的数据进行组合和合并，并进行规范处理，减少数据冗余性，保证数据的一致性。这种模型并不适合直接用于分析统计。

2.2.2 维度模型

数据仓库领域的令一位大师——Ralph Kimball 倡导的建模方法为维度建模。维度模型将复杂的业务通过事实和维度两个概念进行呈现。事实通常对应业务过程，而维度通常对应业务过程发生时所处的环境。

注：业务过程可以概括为一个个不可拆分的行为事件，例如电商交易中的下单，取消订单，付款，退单等，都是业务过程。

下图为一个典型的维度模型，其中位于中心的 SalesOrder 为事实表，其中保存的是下单这个业务过程的所有记录。位于周围每张表都是维度表，包括 Date（日期），Customer（顾客），Product（产品），Location（地区）等，这些维度表就组成了每个订单发生时所处的环境，即何人、何时、在何地下单了何种产品。从图中可以看出，模型相对清晰、简洁。



维度建模以数据分析作为出发点，为数据分析服务，因此它关注的重点的用户如何更快的完成需求分析以及如何实现较好的大规模复杂查询的响应性能。

第 3 章　维度建模理论之事实表

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。其包含与该业务过程有关的维度引用（维度表外键）以及该业务过程的度量（通常是可累加的数字类型字段）。

事实表通常比较“细长”，即列较少，但行较多，且行的增速快。

事实表有三种类型：分别是事务事实表、周期快照事实表和累积快照事实表，每种事实表都具有不同的特点和适用场景，下面逐个介绍。

3.1 事务型事实表

3.1.1 概述

事务事实表用来记录各业务过程，它保存的是各业务过程的原子操作事件，即最细粒度的操作事件。粒度是指事实表中一行数据所表达的业务细节程度。

事务型事实表可用于分析与各业务过程相关的各项统计指标，由于其保存了最细粒度的记录，可以提供最大限度的灵活性，可以支持无法预期的各种细节层次的统计需求。

3.1.2 设计流程

设计事务事实表时一般可遵循以下四个步骤：

选择业务过程→声明粒度→确认维度→确认事实

(1) 选择业务过程

在业务系统中，挑选我们感兴趣的业务过程，业务过程可以概括为一个个不可拆分的行为事件，例如电商交易中的下单，取消订单，付款，退单等，都是业务过程。通常情况下，一个业务过程对应一张事务型事实表。

(2) 声明粒度

业务过程确定后，需要为每个业务过程声明粒度。即精确定义每张事务型事实表的每行数据表示什么，应该尽可能选择最细粒度，以此来应各种细节程度的需求。

典型的粒度声明如下：

订单事实表中一行数据表示的是一个订单中的一个商品项。

(3) 确定维度

确定维度具体是指，确定与每张事务型事实表相关的维度有哪些。

确定维度时应尽量多的选择与业务过程相关的环境信息。因为维度的丰富程度就决定了维度模型能够支持的指标丰富程度。

(4) 确定事实

此处的“事实”一词，指的是每个业务过程的度量值（通常是可累加的数字类型的值，例如：次数、个数、件数、金额等）。

经过上述四个步骤，事务型事实表就基本设计完成了。第一步选择业务过程可以确定有哪些事务型事实表，第二步可以确定每张事务型事实表的每行数据是什么，第三步可以确定每张事务型事实表的维度外键，第四步可以确定每张事务型事实表的度量值字段。

3.1.3 不足

事务型事实表可以保存所有业务过程的最细粒度的操作事件，故理论上其可以支撑与各业务过程相关的各种统计粒度的需求。但对于某些特定类型的需求，其逻辑可能会比较复杂，或者效率会比较低下。例如：

(1) 存量型指标

例如商品库存，账户余额等。此处以电商中的虚拟货币为例，虚拟货币业务包含的业务过程主要包括获取货币和使用货币，两个业务过程各自对应一张事务型事实表，一张存储所有的获取货币的原子操作事件，另一张存储所有使用货币的原子操作事件。

假定现有一个需求，要求统计截至当日的各用户虚拟货币余额。由于获取货币和使用货币均会影响到余额，故需要对两张事务型事实表进行聚合，且需要区分两者对余额的影响（加或减），另外需要对两张表的全表数据聚合才能得到统计结果。

可以看到，不论是从逻辑上还是效率上考虑，这都不是一个好的方案。

(2) 多事务关联统计

例如，现需要统计最近 30 天，用户下单到支付的时间间隔的平均值。统计思路应该是找到下单事务事实表和支付事务事实表，过滤出最近 30 天的记录，然后按照订单 id 对两张事实表进行关联，之后用支付时间减去下单时间，然后再求平均值。

逻辑上虽然并不复杂，但是其效率较低，因为下单事务事实表和支付事务事实表均为大表，大表 join 大表的操作应尽量避免。

可以看到，在上述两种场景下事务型事实表的表现并不理想。下面要介绍的另外两种类型的事实表就是为了弥补事务型事实表的不足的。

3.2 周期型快照事实表

3.2.1 概述

周期快照事实表以具有规律性的、可预见的时间间隔来记录事实，主要用于分析一些存量型（例如商品库存，账户余额）或者状态型（空气温度，行驶速度）指标。

对于商品库存、账户余额这些存量型指标，业务系统中通常就会计算并保存最新结果，所以定期同步一份全量数据到数据仓库，构建周期型快照事实表，就能轻松应对此类统计需求，而无需再对事务型事实表中大量的历史记录进行聚合了。

对于空气温度、行驶速度这些状态型指标，由于它们的值往往是连续的，我们无法捕获其变动的原子事务操作，所以无法使用事务型事实表统计此类需求。而只能定期对其进行采样，构建周期型快照事实表。

3.2.2 设计流程

(1) 确定粒度

周期型快照事实表的粒度可由采样周期和维度描述，故确定采样周期和维度后即可确定

粒度。

采样周期通常选择每日。

维度可根据统计指标决定，例如指标为统计每个仓库中每种商品的库存，则可确定维度为仓库和商品。

确定完采样周期和维度后，即可确定该表粒度为每日-仓库-商品。

(2) 确认事实

事实也可根据统计指标决定，例如指标为统计每个仓库中每种商品的库存，则事实为商品库存。

3.2.3 事实类型

此处的事实类型是指度量值的类型，而非事实表的类型。事实（度量值）共分为三类，分别是可加事实，半可加事实和不可加事实。

(1) 可加事实

可加事实是指可以按照与事实表相关的所有维度进行累加，例如事务型事实表中的事实。

(2) 半可加事实

半可加事实是指只能按照与事实表相关的一部分维度进行累加，例如周期型快照事实表中的事实。以上述各仓库中各商品的库存每天快照事实表为例，这张表中的库存事实可以按照仓库或者商品维度进行累加，但是不能按照时间维度进行累加，因为将每天的库存累加起来是没有任何意义的。

(3) 不可加事实

不可加事实是指完全不具备可加性，例如比率型事实。不可加事实通常需要转化为可加事实，例如比率可转化为分子和分母。

3.3 累积型快照事实表

3.3.1 概述

累积快照事实表是基于一个业务流程中的多个关键业务过程联合处理而构建的事实表，如交易流程中的下单、支付、发货、确认收货业务过程。

累积型快照事实表通常具有多个日期字段，每个日期对应业务流程中的一个关键业务过程（里程碑）。

订单 id	用户 id	下单日期	支付日期	发货日期	确认收货日期	订单金额	支付金额
1001	1234	2020-06-14	2020-06-15	2020-06-16	2020-06-17	1000	1000

累积型快照事实表主要用于分析业务过程（里程碑）之间的时间间隔等需求。例如前文提到的用户下单到支付的平均时间间隔，使用累积型快照事实表进行统计，就能避免两个事务事实表的关联操作，从而变得十分简单高效。

3.3.2 设计流程

累积型快照事实表的设计流程同事务型事实表类似，也可采用以下四个步骤，下面重点描述与事务型事实表的不同之处。

选择业务过程→声明粒度→确认维度→确认事实。

(1) 选择业务过程

选择一个业务流程中需要关联分析的多个关键业务过程，多个业务过程对应一张累积型快照事实表。

(2) 声明粒度

精确定义每行数据表示的是什么，尽量选择最小粒度。

(3) 确认维度

选择与各业务过程相关的维度，需要注意的是，每各业务过程均需要一个日期维度。

(4) 确认事实

选择各业务过程的度量值。

第 4 章　维度建模理论之维度表

4.1　维度表概述

维度表是维度建模的基础和灵魂。前文提到，事实表紧紧围绕业务过程进行设计，而维度表则围绕业务过程所处的环境进行设计。维度表主要包含一个主键和各种维度字段，维度字段称为维度属性。

4.2　表设计步骤

(1) 确定维度（表）

在设计事实表时，已经确定了与每个事实表相关的维度，理论上每个相关维度均需对应一张维度表。需要注意到，可能存在多个事实表与同一个维度都相关的情况，这种情况需保证维度的唯一性，即只创建一张维度表。另外，如果某些维度表的维度属性很少，例如只有一个名称，则可不创建该维度表，而把该表的维度属性直接增加到与之相关的事实表中，这个操作称为**维度退化**。

(2) 确定主维表和相关维表

此处的主维表和相关维表均指**业务系统**中与某维度相关的表。例如业务系统中与商品相关的表有 sku_info，spu_info，base_trademark，base_category3，base_category2，base_category1 等，其中 sku_info 就称为商品维度的主维表，其余表称为商品维度的相

关维表。维度表的粒度通常与主维表相同。

(3) 确定维度属性

确定维度属性即确定维度表字段。维度属性主要来自于业务系统中与该维度对应的主维表和相关维表。维度属性可直接从主维表或相关维表中选择，也可通过进一步加工得到。

确定维度属性时，需要遵循以下要求：

- **尽可能生成丰富的维度属性**

维度属性是后续做分析统计时的查询约束条件、分组字段的基本来源，是数据易用性的关键。维度属性的丰富程度直接影响到数据模型能够支持的指标的丰富程度。

- **尽量不使用编码，而使用明确的文字说明，一般可以编码和文字共存。**
- **尽量沉淀出通用的维度属性**

有些维度属性的获取需要进行比较复杂的逻辑处理，例如需要通过多个字段拼接得到。为避免后续每次使用时的重复处理，可将这些维度属性沉淀到维度表中。

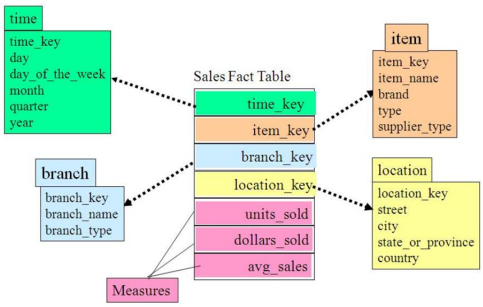
4.3 维度设计要点

4.3.1 规范化与反规范化

规范化是指使用一系列范式设计数据库的过程，其目的是减少数据冗余，增强数据的一致性。通常情况下，规范化之后，一张表的字段会拆分到多张表。

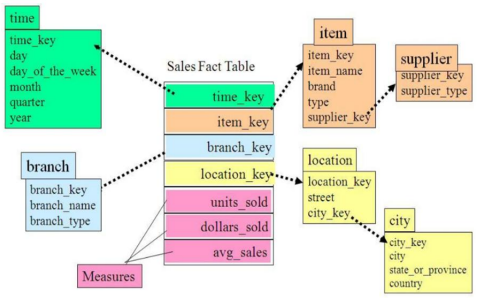
反规范化是指将多张表的数据冗余到一张表，其目的是减少 join 操作，提高查询性能。在设计维度表时，如果对其进行规范化，得到的维度模型称为雪花模型，如果对其进行反规范化，得到的模型称为星型模型。

1、星型模型



雪花模型与星型模型的区别主要在于维度表是否进行规范化。

2、雪花模型



雪花模型，比较靠近3NF，但是无法完全遵守，因为遵循3NF的性能成本太高。

数据仓库系统的主要目的是用于数据分析和统计，所以是否方便用户进行统计分析决定了模型的优劣。采用雪花模型，用户在统计分析的过程中需要大量的关联操作，使用复杂度高，同时查询性能很差，而采用星型模型，则方便、易用且性能好。所以出于易用性和性能的考虑，离线维度表一般是很不规范化的-星型模型。

4.4 实时数仓维度设计

在电商离线数仓中，普通维度表是通过主维表和相关维表做关联查询生成的。与之对应的业务表数据是通过每日一次全量同步导入到 HDFS 的，只须每日做一次全量数据的关联查询即可。而实时数仓中，系统上线后我们采集的是所有表的变化数据，这样就会导致一旦主维表或相关维表中的某张表数据发生了变化，就需要和其它表的历史数据做关联。

此时我们会面临一个问题：如何获取历史数据？

对于这个问题，我们可以考虑当某张维度表发生变化后，执行一次 maxwell-bootstrap 命令，将相关维度表的数据导入 Kafka。但是这样做又会面临三个问题：

(1) Kafka 中存储冗余数据

(2) maxwell-bootstrap 命令交给谁去执行？必然会面临谁去调度的问题；

(3) 实时数仓中的数据是以流的形式存在的，如果不同流中数据进入程序的机器时间差异过大就会出现 join 不上的情况。如何保证导入的历史数据和变化数据可以关联上？势必要尽可能及时地执行历史数据导入命令且在 Flink 程序中设置足够的延迟。而前者难以保证，后者又会影响整个实时数仓的时效性。

基于上述分析，维度退化的方式并不适用于实时数仓。

因此，在实时数仓中，我们不再对业务数据库中的维度表进行退化，仅对一些不需要的字段进行过滤，然后将维度数据写入 HBase 的维度表中，业务数据库的维度表和 HBase 的维度表是一一对应的。

写入维度数据使用 HBase 的 Phoenix 客户端提供的 upsert 语法，实现幂等写入。当维度数据发生变化时，程序会用变化后的新数据覆盖 Phoenix 维表中相同主键的旧数据。从而保证 Phoenix 表中保存的是一份全量最新的维度数据。

这样做会产生一个问题：实时数仓没有保存历史维度数据，与数仓特征（保存历史数据）相悖。那么，维度表可以按照上述思路设计吗？

首先，我们要明确：离线数仓之所以要保存历史数据，是为了运用历史数据做一些相关指标的计算，而实时数仓本就是对最新的业务数据做分析计算，不涉及历史数据，因此无须保存历史数据。

此外，生产环境中实时数仓的上线通常不会早于离线数仓，如果有涉及到历史数据的指标，在离线数仓中计算即可。因此，实时数仓中只需要保留一份最新的维度数据，上述方案是切实可行的。

特别地，对于字典表，数据一般不会变化，而且我们至多只会用到 dic_code，dic_name 和 parent_code 三个字段，建立单独的维度表意义不大，选择将维度字段退化到事实表中。