

大厂学苑-大数据&人工智能 数仓

版本：V1.0



第1章 数据仓库入门

1.1 数据仓库概念

从字面上来看，数据仓库就是一个存放数据的仓库，它里面存放了各种各样的数据，而这些数据需要按照一些结构、规则来组织和存放。比如生活中，让你从货架上取货物和让你从仓库中取货物，是不是直观感受不一样，是不是从仓库中取货物会感觉比较麻烦，为什么？就是因为一般理解仓库的概念就是大，和多，咱们这里要说的数据仓库也是这个概念，数据量大，数据类型多。但是一个仓库中的东西如果是杂乱无章的话，找起来是不是就会更费劲了，对吧？所以一般的仓库都会有管理员，将不同的货物分门别类的管理起来。咱们这里讲的数据仓库也是同样道理，会将数据根据实际的环境划分不同的层次，保存不同的数据。

早期的企业环境，企业的生产与服务是一个很长周期，导致业务数据呈现一种粗粒度模式。随着互联网的快速渗透从早期的 PC 终端到当下的移动终端，对用户的需求与服务周期将逐渐的缩短，业务量级、数据类型多样化与存储的暴增，对应着技术、架构、业务呈现出迅猛发展，相应的数据沉淀与积累也成指数暴涨。如何将这些数据收集起来并加以利用，就是我们开发人员需要特别关心的了。

从“数据仓库”开始到现在的“大数据”，中间经历了太多的知识、架构模式的演进与变革，数据仓库一般指的是：在相当长的时间内堆积数据，仅仅需要处理大量数据请求中的少部分的系统。数据仓库其实是一套体系，他不是一门特定的什么技术，而是整合了很多已有的技术，来更好地组织和管理数据。数据仓库不等同于“海量数据”。恰恰相反，而是其子集。海量数据也包含：通过大量的连接提供每秒百万次服务请求的系统。大数据是海量数据 + 复杂类型数据基础上的数据分析、数据存储，数据展示等一系列的技术体系。

1.2 数据仓库 & 数据库

数据仓库和数据库从文字上来看，是比较相似的，所以一般不是很好区分，说到他们的区别，我们一般会提到 OLTP 和 OLAP。

- OLTP: On-Line Transaction Processing, 联机事务处理，主要是业务数据，需要考虑高并发、考虑事务
- OLAP: On-Line Analytical Processing, 联机分析处理，重点主要是面向分析，会产生大量的查询，一般很少涉及增删改

	OLTP System Online Transaction Processing (Operational System)	OLAP System Online Analytical Processing (Data Warehouse)
Source of data	Operational data; OLTPs are the original source of the data.	Consolidation data; OLAP data comes from the various OLTP Databases
Purpose of data	To control and run fundamental business tasks	To help with planning, problem solving, and decision support
What the data	Reveals a snapshot of ongoing business processes	Multi-dimensional views of various kinds of business activities
Inserts and Updates	Short and fast inserts and updates initiated by end users	Periodic long-running batch jobs refresh the data
Queries	Relatively standardized and simple queries Returning relatively few records	Often complex queries involving aggregations
Processing Speed	Typically very fast	Depends on the amount of data involved; batch data refreshes and complex queries may take many hours; query speed can be improved by creating indexes
Space Requirements	Can be relatively small if historical data is archived	Larger due to the existence of aggregation structures and history data; requires more indexes than OLTP
Database Design	Highly normalized with many tables	Typically de-normalized with fewer tables; use of star and/or snowflake schemas
Backup and Recovery	Backup religiously; operational data is critical to run the business, data loss is likely to entail significant monetary loss and legal liability	Instead of regular backups, some environments may consider simply reloading the OLTP data as a recovery method

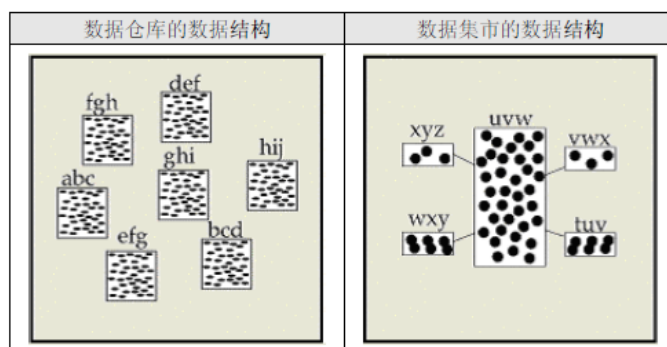
好了，举个例子，你现在有一个 3 层的抽屉，旁边有一个屋子，是资料库。那么如果别人现在给你一份文件，你会考虑将这份文件放在哪里呢？此时，就需要考虑不同的场景了，如果这份文件是平时经常用的，或者需要频繁的进行一些修改，参考的，那么一般就会放到手边的抽屉中，如果是一个需要归档的文件或以后会很少打开的文件，那么可以考虑将这份文件放到资料库中，是吗？如果这里的抽屉类比为数据库，将资料库类比为数据仓库，你懂了吗？

1.3 数据仓库 & 数据集市

- **数据仓库**：是一个集成的面向主题的数据集合，设计的目的是支持 DSS（决策支持系统）的功能，在数据仓库里，每个数据单元都和特定的时间相关。数据仓库包括原子级别的数据和轻度汇总的数据。是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化的（Time Variant）数据集合，用以支持经营管理中的决策制定过程。数据仓库是重建企业数据流和信息流的过程，在这个过程中，构造企业的决策支持环境，以区别原来的业务系统所构建的操作型环境。数据仓库的价值并不是你在仓库中所存储的数据量的多少，而关键在于从仓库中能够获得的信息和分析结果的质量。
- **数据集市**：是一个小型的部门或工作组级别的数据仓库。有两种类型的数据集市——独立型和从属型。独立型数据集市直接从操作型环境获取数据。从属型数据集市从企业级数据仓库获取数据。从长远的角度看，从属型数据集市在体系结构上比独立型数据集市更稳定。独立的建立多个数据集市，企业只会又增加了一些信息孤岛，仍然不能以整个企业的视图分析数据，数据集市为各个部门或工作组所用，各个集市之间又会存在不一致性。当然，独立型数据集市是一种既成事实，为满足特定用户的需求而建立的一种分析型环境，但是，从长远的观点看，是一种权宜之计，必然会被企业级的数据仓库所

取代。

数据仓库和数据集市之间的区别:



数据仓库中数据结构采用的规范化模式（关系数据库设计理论），数据集市的数据结构采用的星型模式（多维数据库设计理论）。数据仓库中数据的粒度比数据集市的细

	数据仓库	数据集市
数据来源	遗留系统、OLTP 系统、外部数据	数据仓库
范围	企业级	部门级或工作组级
主题	企业主题	部门或特殊的分析主题
数据粒度	最细的粒度	较粗的粒度
数据结构	规范化结构（第 3 范式）	星型模式、雪片模式、或两者混合
历史数据	大量的历史数据	适度的历史数据
优化	处理海量数据 数据探索	便于访问和分析 快速查询
索引	高度索引	高度索引

1.4 数据仓库 & 数据湖

➤ **数据仓库(Data Warehouse)**：是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化的（Time Variant）数据集合，用于支持管理决策和信息的全局共享。其主要功能是将组织透过资讯系统之联机事务处理(OLTP)经年累月所累积的大量资料，透过数据仓库理论所特有的资料储存架构，作一有系统的分析整理，以利各种分析方法如联机分析处理(OLAP)、数据挖掘(Data Mining)之进行，并进而支持如决策支持系统(DSS)、主管资讯系统(EIS)之创建，帮助决策者能快速有效的自大量资料中，分析出有价值的资讯，以利决策拟定及快速回应外在环境变动，帮助建构商业智能(BI)。

- **所谓主题**：是指用户使用数据仓库进行决策时所关心的重点方面，如：收入、客户、销售渠道等；所谓面向主题，是指数据仓库内的信息是按主题进行组织的，而不是

像业务支撑系统那样是按照业务功能进行组织的。

- **所谓集成：**是指数据仓库中的信息不是从各个业务系统中简单抽取出来的，而是经过一系列加工、整理和汇总的过程，因此数据仓库中的信息是关于整个企业的一致性的全局信息。
- **所谓随时间变化：**是指数据仓库内的信息并不只是反映企业当前的状态，而是记录了从过去某一时点到当前各个阶段的信息。通过这些信息，可以对企业的发展历程和未来趋势做出定量分析和预测。

➤ **数据湖 (Data Lake)：**是一个存储企业的各种各样原始数据的大型仓库，其中的数据可供存取、处理、分析及传输。数据湖是以其自然格式存储的数据的系统或存储库，通常是对象 blob 或文件。数据湖通常是企业所有数据的单一存储，包括源系统数据的原始副本，以及用于报告、可视化、分析和机器学习等任务的转换数据。数据湖可以包括来自关系数据库（行和列）的结构化数据，半结构化数据（CSV，日志，XML，JSON），非结构化数据（电子邮件，文档，PDF）和二进制数据（图像，音频，视频）。

数据仓库与数据湖差异：

- 在储存方面上，数据湖中数据为非结构化的，所有数据都保持原始形式。存储所有数据，并且仅在分析时再进行转换。数据仓库就是数据通常从事务系统中提取。
- 在将数据加载到数据仓库之前，会对数据进行清理与转换。在数据抓取中数据湖就是捕获半结构化和非结构化数据。而数据仓库则是捕获结构化数据并将其按模式组织。
- 数据湖的目的就是数据湖非常适合深入分析的非结构化数据。数据科学家可能会用具有预测建模和统计分析等功能的高级分析工具。而数据仓库就是数据仓库非常适用于月度报告等操作用途，因为它具有高度结构化。
- 在架构中数据湖通常，在存储数据之后定义架构。使用较少的初始工作并提供更大的灵活性。在数据仓库中存储数据之前定义架构。

数据仓库	数据湖
主要处理历史的、结构化的数据，而且这些数据必须与数据仓库事先定义模型吻合。	能处理所有类型的数据，如结构化数据，非结构化数据，半结构化数据等，数据的类型依赖于数据源系统的原始数据格式。
处理结构化数据，将它们或者转化为多维数据，或者转换为报表，以满足后续的高级报表及数据分析需求。	拥有足够强的计算能力用于处理和分析所有类型的数据，分析后的数据会被存储起来供用户使用。
数据仓库通常用于存储和维护长期数据，因此数据可以按需访问。	数据湖通常包含更多的相关的信息，这些信息有很高概率会被访问，并且能够为企业挖掘新的运营需求。



第2章 数据仓库理论

2.1 数仓分层

2.1.1 为什么要分层

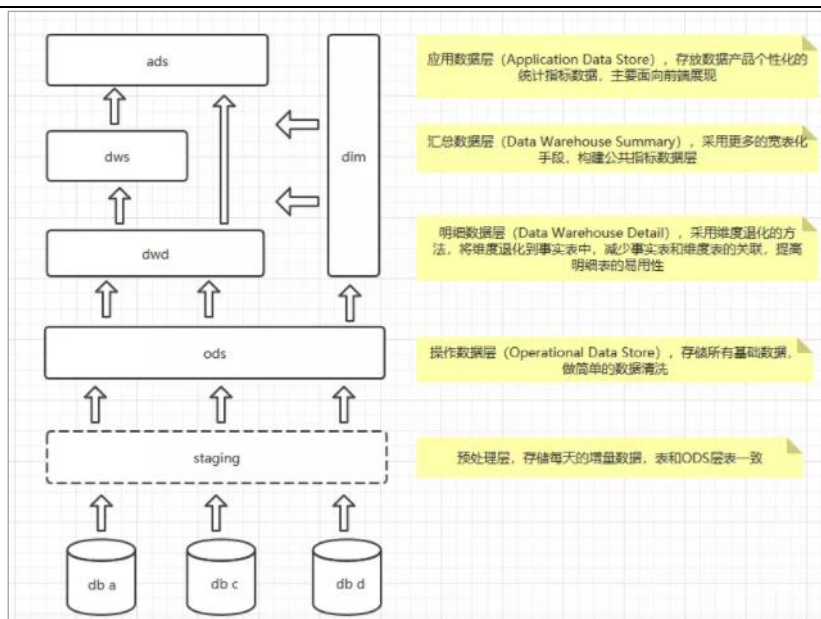
数据仓库为什么要分层？

1. **把复杂问题简单化**：将复杂的任务分解成多层来完成，每一层只处理简单的任务，方便定位问题。
2. **减少重复开发**：规范数据分层，通过中间层数据，能够减少极大的重复计算，增加一次性计算结果的复用性。
3. **隔离原始数据**：不论是数据的异常还是数据的敏感性，使真实数据和统计数据解耦开。

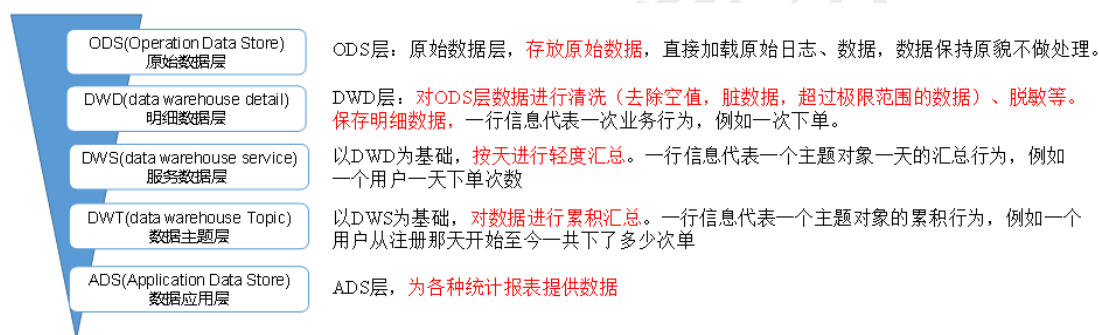
数仓的分层或者建模，其实都是为了更好的去组织、管理、维护数据。说到数仓建模，就得提下经典的 2 套理论：

- **范式（关系）建模**：Inmon 提出的集线器的自上而下（EDW-DM）的数据仓库架构。
- **维度建模**：Kimball 提出的总线式的自下而上（DM-DW）的数据仓库架构。维度建模，一般都会提到星型模型、雪花模型，星型模型做 OLAP 分析很方便

2.1.2 数据仓库分层



每个企业对数仓的分层没有固定的标准，有的分为四层，有的分为五层。



2.1.2 数仓命名规范

2.1.2.1 表命名

- ODS 层命名为 ods_表名
- DWD 层命名为 dwd_dim/fact_表名
- DWS 层命名为 dws_表名
- DWT 层命名为 dwt_表名
- ADS 层命名为 ads_表名
- 临时表命名为 xxx_tmp
- 用户行为表，以 log 为后缀。

2.1.2.2 表字段类型

- 数量类型为 bigint

- 金额类型为 decimal(16, 2)，表示：16 位有效数字，其中小数部分 2 位
- 字符串(名字，描述信息等)类型为 string
- 主键外键类型为 string
- 时间戳类型为 bigint

2.2 范式理论

2.2.1 定义

范式可以理解为设计一张数据表的表结构，符合的标准级别、规范和要求。目前业界范式有：**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)**、**第五范式(5NF)**。

优点：

采用范式，可以**降低数据的冗余性**。

为什么要降低数据**冗余性**？

- (1) 十几年前，磁盘很贵，为了减少磁盘存储。
- (2) 以前没有分布式系统，都是单机，只能增加磁盘，磁盘个数也是有限的
- (3) 一次修改，需要修改多个表，很难保证**数据一致性**

缺点：

范式的缺点是获取数据时，需要通过 Join 拼接出最后的数据。

2.2.2 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。

2.2.3 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

2.2.4 第三范式

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

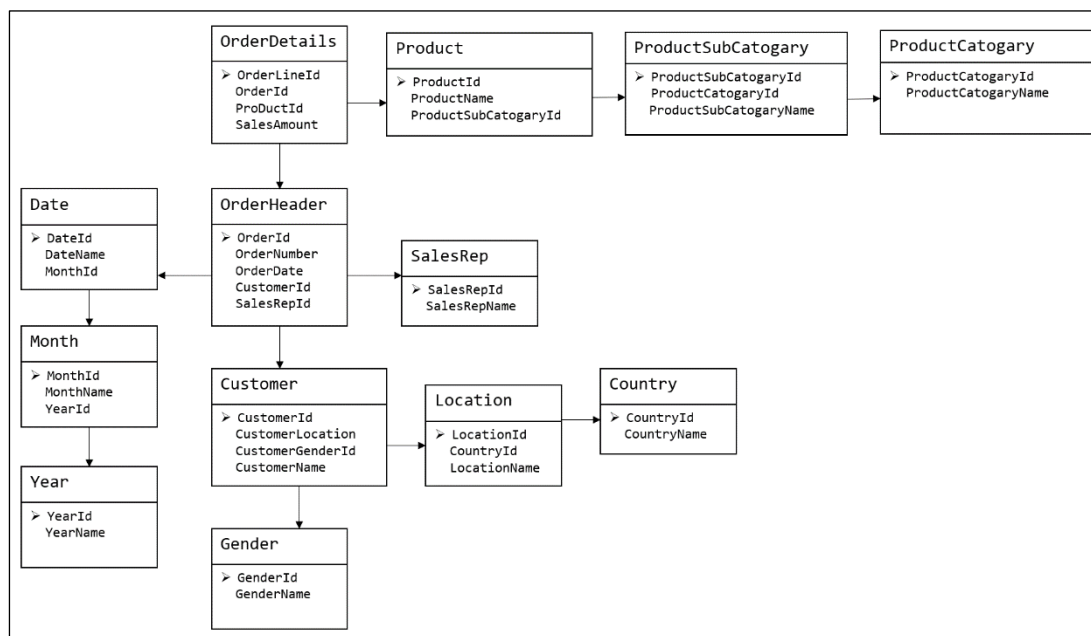
上面表需要再次拆解：

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

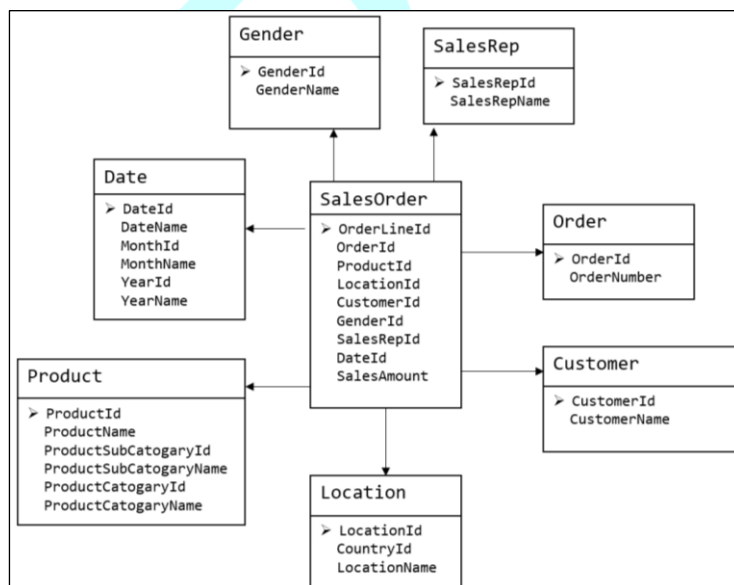
系名	系主任
经济系	王强
法律系	刘玲

2.3 数仓建模

2.3.1 关系建模



关系模型如图所示，严格遵循第三范式（3NF），从图中可以看出，较为松散、零碎，物理表数量多，而数据冗余程度低。由于数据分布于众多的表中，这些数据可以更为灵活地被应用，功能性较强。关系模型主要应用与 OLTP 系统中，为了保证数据的一致性以及避免冗余，所以大部分业务系统的表都是遵循第三范式的。



维度模型如图所示，主要应用于 OLAP 系统中，通常以某一个事实表为中心进行表的

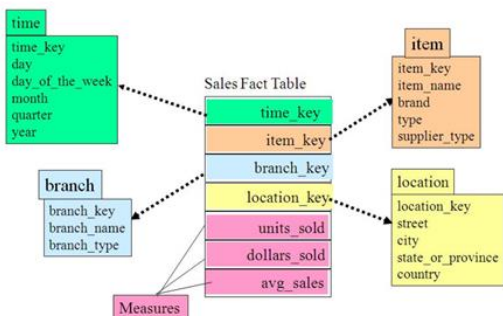
组织，主要面向业务，特征是可能存在数据的冗余，但是能方便的得到数据。

关系模型虽然冗余少，但是在大规模数据，跨表分析统计查询过程中，会造成多表关联，这会大大降低执行效率。所以通常我们采用维度模型建模，把相关各种表整理成两种：事实表和维度表两种。

2.3.2 维度建模

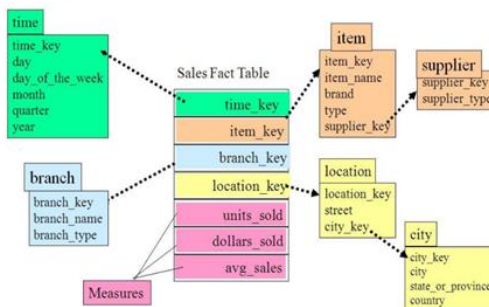
在维度建模的基础上又分为三种模型：星型模型、雪花模型、星座模型。

1、星型模型



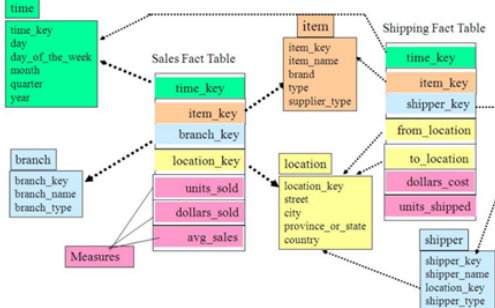
雪花模型与星型模型的区别主要在于维度的层级，标准的星型模型维度只有一层，而雪花模型可能会涉及多级。

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。

3、星座模型



星座模型与前两种情况的区别是事实表的数量，星座模型是基于多个事实表。

基本上是很多数据仓库的常态，因为很多数据仓库都是多个事实表的。所以星座不星座只反映是否有多个事实表，他们之间是否共享一些维度表。

所以星座模型并不和前两个模型冲突。

4、模型的选择

首先就是星座不星座这个只跟数据和需求有关系，跟设计没关系，不用选择。

星型还是雪花，取决于性能优先，还是灵活更优先。

目前实际企业开发中，不会绝对选择一种，根据情况灵活组合，甚至并存（一层维度和多层维度都保存）。但是整体来看，更倾向于维度更少的星型模型。尤其是Hadoop体系，减少Join就是减少Shuffle，性能差距很大。（关系型数据可以依靠强大的主键索引）

让天下没有难学的技术

2.4 表分类

2.4.1 维度表

维度表：一般是对事实的描述信息。每一张维表对应现实世界中的一个对象或者概念。

例如：用户、商品、日期、地区等。

维表的特征:

- 维表的范围很宽（具有多个属性、列比较多）
- 跟事实表相比，行数相对较小：通常< 10 万条
- 内容相对固定：编码表

时间维度表:

日期 ID	day of week	day of year	季度	节假日
2020-01-01	2	1	1	元旦
2020-01-02	3	2	1	无
2020-01-03	4	3	1	无
2020-01-04	5	4	1	无
2020-01-05	6	5	1	无

2.4.2 事实表

事实表中的**每行数据代表一个业务事件（下单、支付、退款、评价等）**。“事实”这个术语表示的是业务事件的**度量值（可统计次数、个数、金额等）**，例如，2020 年 5 月 21 日，张三在淘宝上买了一双 360 块钱的乔丹篮球鞋。维度表：时间、用户、商品、商家。事实表：360 块钱、一双

每一个事实表的行包括：具有可加性的数值型的度量值、与维表相连接的外键，通常具有两个和两个以上的外键。

事实表的特征:

- 非常的大
- 内容相对的窄：列数较少（主要是外键 id 和度量值）
- 经常发生变化，每天会新增加很多。

1) 事务型事实表

以**每个事务或事件为单位**，例如一个销售订单记录，一笔支付记录等，作为事实表里的一行数据。一旦事务被提交，事实表数据被插入，数据就不再进行更改，其更新方式为增量更新。

2) 周期型快照事实表

周期型快照事实表中**不会保留所有数据，只保留固定时间间隔的数据**，例如每天或者每月的销售额，或每月的账户余额等。

例如购物车，有加减商品，随时都有可能变化，但是我们更关心每天结束时这里面有多

少商品，方便我们后期统计分析。

3) 累积型快照事实表

累计快照事实表用于跟踪业务事实的变化。例如，数据仓库中可能需要累积或者存储订单从下订单开始，到订单商品被打包、运输、和签收的各个业务阶段的时间点数据来跟踪订单声明周期的进展情况。当这个业务过程进行时，事实表的记录也要不断更新。

订单 id	用户 id	下单时间	打包时间	发货时间	签收时间	订单金额
		3-8	3-8	3-9	3-10	

第3章 数据仓库架构

3.1 离线数仓

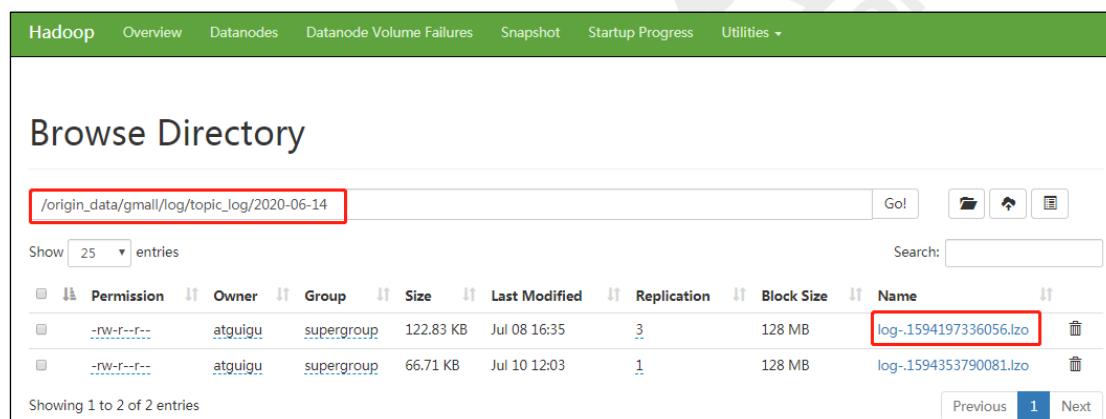
3.1.1 概念

所谓的离线数仓，其实就是指数据源为离线数据所形成的数据仓库

3.1.2 数仓分层

3.1.2.1 ODS 层

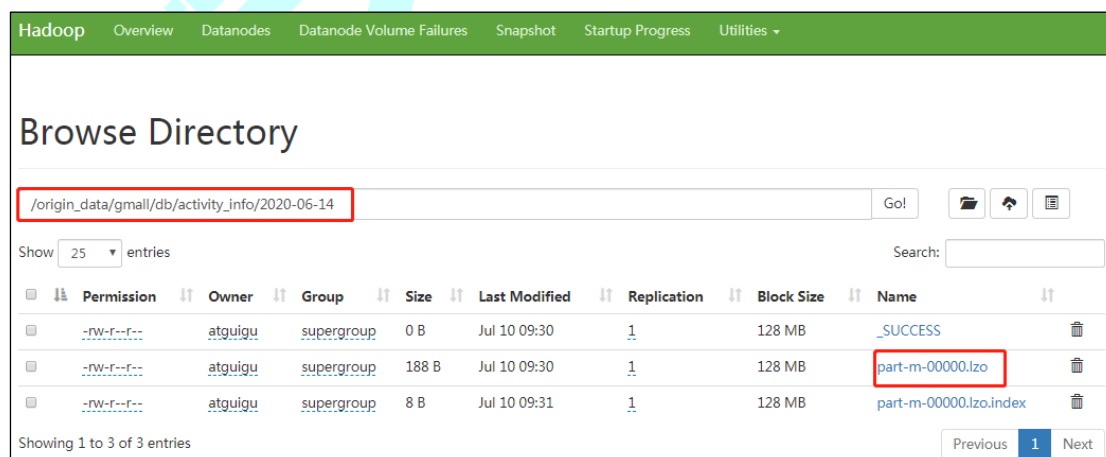
1) HDFS 用户行为数据



The screenshot shows the Hadoop web interface for browsing the directory `/origin_data/gmail/log/topic_log/2020-06-14`. The table lists two files:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	atguigu	supergroup	122.83 KB	Jul 08 16:35	3	128 MB	log-1594197336056.lzo
-rw-r--r--	atguigu	supergroup	66.71 KB	Jul 10 12:03	1	128 MB	log-1594353790081.lzo

2) HDFS 业务数据



The screenshot shows the Hadoop web interface for browsing the directory `/origin_data/gmail/db/activity_info/2020-06-14`. The table lists three files:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	atguigu	supergroup	0 B	Jul 10 09:30	1	128 MB	_SUCCESS
-rw-r--r--	atguigu	supergroup	188 B	Jul 10 09:30	1	128 MB	part-m-00000.lzo
-rw-r--r--	atguigu	supergroup	8 B	Jul 10 09:31	1	128 MB	part-m-00000.lzo.index

3) 针对 HDFS 上的用户行为数据和业务数据，我们如何规划处理？

(1) 保持数据原貌不做任何修改，起到备份数据的作用。

(2) 数据采用压缩，减少磁盘存储空间（例如：原始数据 100G，可以压缩到 10G 左右）

(3) 创建分区表，防止后续的全表扫描

3.1.2.2 DWD 层

DWD 层需构建维度模型，一般采用星型模型，呈现的状态一般为星座模型。维度建模一般按照以下四个步骤：

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线，比如下单业务，支付业务，退款业务，物流业务，一条业务线对应一张事实表。

如果是中小公司，尽量把所有业务过程都选择。

如果是大公司（1000 多张表），选择和需求相关的业务线。

(2) 声明粒度

数据粒度指数据仓库的数据中保存数据的细化程度或综合程度的级别。

声明粒度意味着精确定义事实表中的一行数据表示什么，应该尽可能选择最小粒度，以此来应各种各样的需求。

典型的粒度声明如下：

订单事实表一行数据表示的是一个订单中的一个商品项。

支付事实表一行数据表示的是一个支付记录。

(3) 确定维度

维度的主要作用是描述业务是事实，主要表示的是“谁，何处，何时”等信息。

确定维度的原则是：后续需求中是否要分析相关维度的指标。例如，需要统计，什么时间下的订单多，哪个地区下的订单多，哪个用户下的订单多。需要确定的维度就包括：时间维度、地区维度、用户维度。

(4) 确定事实

此处的“事实”一词，指的是业务中的度量值（次数、个数、件数、金额，可以进行累加），例如订单金额、下单次数等。

在 DWD 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。事实表可做适当的宽表化处理。

事实表和维度表的关联比较灵活，但是为了应对更复杂的业务需求，可以将能关联上的表尽量关联上。如何判断是否能够关联上呢？在业务表关系图中，只要两张表能通过中间表能够关联上，就说明能关联上。

	时间	用户	地区	商品	优惠券	活动	编码	度量值
订单	√	√	√			√		件数/金额
订单详情	√	√	√	√				件数/金额
支付	√	√	√					金额
加购	√	√		√				件数/金额
收藏	√	√		√				个数
评价	√	√		√				个数
退款	√	√		√				件数/金额
优惠券领用	√	√			√			个数

至此，数据仓库的维度建模已经完毕，DWD 层是以**业务过程为驱动**。

DWS 层、DWT 层和 ADS 层都是以**需求为驱动**，和维度建模已经没有关系了。

DWS 和 DWT 都是建宽表，按照主题去建表。主题相当于观察问题的角度。对应着维度表。

3.1.2.3 DWS 层 与 DWT 层

DWS 层和 DWT 层统称宽表层，这两层的设计思想大致相同，通过以下案例进行阐述。

1) 问题引出：两个需求，统计每个省份订单的个数、统计每个省份订单的总金额

2) 解决办法：都是将省份表和订单表进行 join，group by 省份，然后计算。同样数据被计算了两次，实际上类似的场景还会更多。

那怎么设计能避免重复计算呢？

针对上述场景，可以设计一张地区宽表，其主键为地区 ID，字段包含为：下单次数、下单金额、支付次数、支付金额等。上述所有指标都统一进行计算，并将结果保存在该宽表中，这样就能有效避免数据的重复计算。

3) 总结：

(1) 需要建哪些宽表：以维度为基准。

(2) 宽表里面的字段：是站在不同维度的角度去看事实表，重点关注事实表聚合后的度量值。

(3) DWS 和 DWT 层的区别：DWS 层存放的所有主题对象当天的汇总行为，例如每个地区当天的下单次数，下单金额等，DWT 层存放的是所有主题对象的累积行为，例如每个地

区最近 7 天（15 天、30 天、60 天）的下单次数、下单金额等。

3.1.2.4 ADS 层

对电商系统各大主题指标分别进行分析。



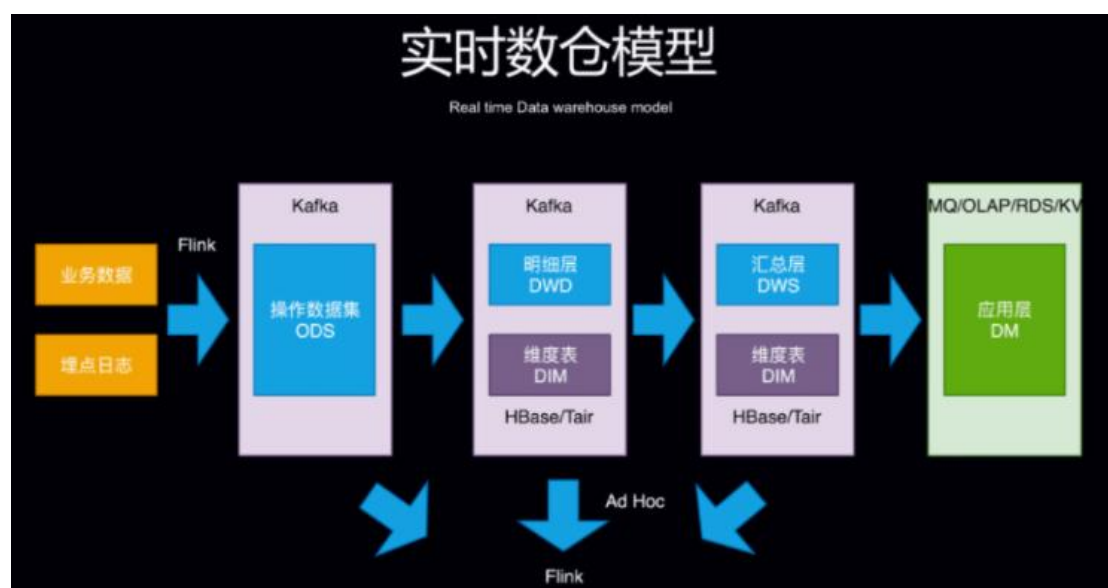
3.2 实时数仓

3.2.1 概念

所谓的实时数仓，其实就是指的数据源为实时数据所形成的数据仓库

3.2.2 数仓分层

实时数仓的分层方式一般也遵守传统数据仓库模型,也分为了 ODS 操作数据集、DWD 明细层和 DWS 汇总层以及应用层。但实时数仓模型的处理的方式却和传统数仓有所差别,如明细层和汇总层的数据一般会放在 Kafka 上,维度数据一般考虑到性能问题则会放在 HBase 或者 Tair 等 KV 存储上,即席查询则可以使用 Flink 完成。



实时数仓和传统数仓的对比主要可以从四个方面考虑:

- 第一个是分层方式, 离线数仓为了考虑到效率问题, 一般会采取空间换时间的方式, 层级划分会比较多; 则实时数仓考虑到实时性问题, 一般分层会比较少, 另外也减少了中间流程出错的可能性。
- 第二个是事实数据存储方面, 离线数仓会基于 HDFS, 实时数仓则会基于消息队列 (如 Kafka)。
- 第三个是维度数据存储, 实时数仓会将数据放在 KV 存储上面。
- 第四个是数据加工过程, 离线数仓一般以 Hive、Spark 等批处理为主, 而实时数仓则是基于实时计算引擎如 Storm、Flink 等, 以流处理为主。