**商务智能**

1. 企业的新问题

企业决策应当源于企业和行业的真实状况，真实状况来源于真实完整的数据

如何有效将企业收集的数据转化为信息，并最终为企业决策所使用

在数据爆炸和信息贫乏的前提下，需要解决信息的

可访问性

及时性

表达格式

完整性

1. BI的目标
   1. 决策支持（Decision-making Supporting）
   2. 改善信息访问（Information access-improving）
2. 商务智能定义：

访问、钻研、分析和挖掘数据获得启发和了解，从而提供更完善和更全面考虑的决策支持。 ——Gartner Group

商务智能是企业利用现代信息技术收集、管理和分析结构化和非结构化的商务数据和信息，创造和累计商务知识和见解，改善商务决策水平，采取有效的商务行动，完善各种商务流程，提升各方面商务绩效，增强综合竞争力的智慧和能力。 ——《三位一体的商务智能：管理、技术与应用》

1. 数据抽取、转换和装载（ETL）的过程
   1. 辨识与主题相关的原始数据
      1. 有用的数据
   2. 开发数据抽取策略
      1. 正确和完整的数据
   3. 将原始数据转换为目标规格
   4. 将原始数据加载到预定目标区域
2. 数据仓库 - 数据管理

存储，索引和备份

不仅管理关系数据库中的数据

而且管理数据立方体（ Data Cube ）中的多维数据（multi-dimensional data）

1. 数据仓库 - 数据访问

* 一种或多种访问数据仓库中的数据的方式
* 面向多种商业用户
  + 行政管理人员
  + 商务分析人员
  + 操作经理
  + 临时终端用户
  + 其他
* 需要支持多种访问方式以及分析和展现工具
  + 查询和报告工具
  + 桌面OLAP
  + 关系OLAP
  + 多维OLAP
  + 数据挖掘
  + 基于仪表盘和基于代理技术等方面的客户决策支持界面

1. 数据统计

数据统计是将数据中含有的信息概括为统计值

例如最大值、最小值、平均值等

典型的数据统计方法

关系分析

要素分析

回归分析

1. 数据挖掘

数据挖掘就是对数据库（数据仓库）中蕴涵的、未知的、非平凡的、有潜在应用价值的模式（规则）的提取

用于发掘数据中隐藏的模式

借鉴各种相关领域的理论和方法

用于发现隐藏模式的算法既可以是自动进行，也可以在人工指导下完成

构造数据仓库

* 明确需求
  + 用户的主观分析需求
* 选择工具
  + 数据抽取（data extract）工具
  + 数据清洗（data cleaning）工具
  + 元数据(metadata)管理工具
  + 数据分析工具 ( look for patterns )
  + 数据挖掘工具 ( look for hidden patterns )
  + 数据展现工具
* 构建数据仓库

操作型处理

* 也叫事务处理，是指对数据库的日常联机访问操作，通常是对一个或一组记录的查询和修改，主要是为企业特定的应用服务的，所以也叫联机事务处理。
* On-Line Transaction Processing（OLTP）
  + 通常仅仅是对一个或一组记录的查询或修改；
  + 查询简单，但执行频率高；
  + 人们关心的是处理的响应时间、数据的安全性和完整性等指标。

分析型处理

* 也叫做信息型处理，主要用于企业管理人员的决策分析，为制订企业的未来经营管理计划提供辅助决策信息。
  + 需要对大量的事务型数据进行统计、归纳和分析；
  + 需要访问大量的历史数据；
  + 执行频率和对响应时间的要求都不高。

-典型的的分析型处理

* + 决策支持系统 ( DSS --Decision Support System )

事务处理环境不适宜DSS应用的原因

* 在传统的以数据库为核心的事务处理环境中不适宜建立DSS等分析型应用，其原因主要有以下六条：
  + 事务处理和分析处理的性能特性不同：

用户每次操作处理的时间短，存取数据量小，但操作频率高，并发程度大。

每次分析可能需要连续运行很长的时间，存取数据量大，但很少做这样的分析处理，也没有并发执行的要求。

* + 数据集成问题：

事务处理一般只需要与本部门业务有关的当前细节数据

分析处理的数据可能来自多种不同的数据源

对于需要集成数据的DSS应用来说，在应用程序中对事务处理环境中的这些纷繁复杂的数据进行集成,将带来下述问题：

大大加重程序员的负担

重复计算

极低的分析处理效率

* + 数据的动态集成问题：

集成数据必须以一定的周期进行刷新（即采用动态集成策略），但传统的事务处理环境并不具备动态集成的能力。

* + 历史数据问题

事务处理一般只需要当前数据

分析处理更看重历史数据

* + 数据的综合问题

事务处理需要的是当前的细节性操作数据，而分析处理需要的往往是大量的总结性分析型数据，而非数据库中的细节性操作型数据。

在分析前往往需要对细节数据进行不同程度的综合，传统的事务处理系统不具备这种综合能力

* + 数据的访问问题

事务处理对于需要修改的数据必须实时‘更新’数据库

分析处理不需要实时的‘更新’操作，但需要定时‘刷新’， 数据的访问操作以“读”操作为主

操作型数据和分析型数据

|  |  |  |
| --- | --- | --- |
| 特 性 | 操 作 型 数 据（DB） | 分 析 型 数 据（DW） |
| 定位 | 面向应用的事务处理 | 面向主题的数据分析 |
| DB设计 | E-R模型 | 星型/雪花模型，数据立方体 |
| 数据 | 当前的、最新的 | 历史的，具有时间跨度 |
| 汇总 | 原始的，细节的 | 集成的，一致的 |
| 视图 | 详细的，关系的 | 总体的，多维的 |
| 操作类型 | 读/写（可变的） | 读（稳定的） |
| 存取请求 | 可预知的 | 事先未知的 |
| 访问记录 | 一次操作少量记录 | 一次操作大量记录 |
| DB规模 | 100MB ~ GB | TB |
| 工作单位 | 短的，简单事务 | 复杂查询 |
| 性能要求 | 对性能要求高 | 对性能要求较宽松 |

将数据仓库与操作型数据库分离开来的原因

* 提高两个系统的性能
* 提高操作型数据库的事务吞吐量
* 两个系统中数据的结构、内容和用法的不同

**建立数据仓库的目的**：为了适应因市场商业经营行为的改变和市场竞争程度的加剧而进行的DSS的需要。

**数据仓库定义：**

数据仓库就是一个面向主题的、集成的、不可更新的（稳定的）、随时间不断变化的数据集合，用于支持经营管理过程中的决策制定。

**面向应用的数据组织特点**

* 表达数据流程
* 和业务中的单据或文档对应
* 逻辑、数据不完全分离
* 和部门、组织相关

面向应用的数据组织基本上是按照企业内部的业务活动及其需要的相关数据来组织数据的存储的，虽然能够方便高效的支持OLTP，但没有实现真正的数据与应用分离，其抽象程度也不够高。

**数据仓库的四个特征：**

* + **面向主题**

**主题的定义：** 主题是较高层次上将企业信息系统中的数据综合、归类并进行分析利用的抽象。在逻辑意义上，是对应企业中某一宏观分析领域涉及的分析对象。

面向主题是指数据仓库内的信息是按主题进行组织的，为按主题进行决策的过程提供信息。

数据仓库中的主题有时会因用户主观要求的变化而变化。

如果按照面向主题的方式进行数据组织，首先应该抽取主题，即按照管理人员的分析要求来确定主题，而与每个主题相关的数据又与有关的事务处理所需的数据不尽相同。

在每个主题中，都包含了有关该主题的所有信息，同时又抛弃了与分析处理无关或不需要的数据，从而将原本分散在各个操作性处理系统中的有关信息集中在一个主题中，形成有关该主题的一个完整一致的描述

不同的主题之间也有重叠的内容，但这种重叠的特点是：

* + - 是逻辑上的，而不是物理存储上的重叠；
    - 是部分细节的重叠，而不是统计信息的重叠；
    - 可以反映不同主题之间的直接和间接的联系。
  + **集成**

数据仓库中的数据是为分析服务的，而分析需要多种广泛的不同数据源以便进行比较、鉴别，因此数据仓库中的数据必须从多个数据源中获取，这些数据源包括多种类型数据库、文件系统以及Internet网上数据等，它们通过数据集成而形成数据仓库中的数据。

**集成的方法：**

* 统一

消除不同数据源之间的数据不一致的现象，如字段的同名异义、异名同义、单位不统一、字长不一致……

* 综合

对原有数据进行综合和计算，如统计、抽样……

* + **不可更新**

数据仓库的数据与操作性数据环境隔离；

数据仓库中的数据是经过抽取而形成的分析型数据，不具有原始性，主要供企业决策分析之用，执行的主要是‘查询’操作，一般情况下不执行‘更新’操作。同时，一个稳定的数据环境也有利于数据分析操作和决策的制订；

但这也不等于数据仓库中的数据不需要“更新”操作，如：

在需要进行新的分析决策时，可能需要进行新的数据抽取（“插入”操作）和“更新”操作

数据仓库中的一些过时的数据，也可以通过“删除”操作丢弃掉

* + **随时间不断变化**

数据仓库内的信息并不只是关于企业当时或某一时点的信息，而是系统记录了企业从过去某一时点到目前的各个阶段的信息，通过这些信息可以对企业的发展历程和未来趋势作出定量分析和预测。

因此数据仓库中的数据通常都带有时间属性，同时必须以一定时间段为单位进行统一更新。如：

不断增加新的数据内容

不断删去旧的数据内容

更新与时间有关的综合数据

**数据仓库的关键技术**

* 数据的ETL（抽取、转换、装载）
* 存储和管理
* 数据的访问和表现
* 元数据

**数据抽取**

* + 数据仓库中的数据来源于数据源，将数据源中数据通过网络进行抽取，并经加工、转换、综合后形成数据仓库中的数据，这就是数据仓库的数据抽取。

**数据刷新**

* + 经过抽取进入数据仓库的数据，在经过一段时间后要重新修正，修改那些过时的数据，保存那些不变的数据，此种动作称为数据仓库的数据刷新。
  + 数据刷新的过程与抽取类似，但刷新的数据量往往小于抽取的数据量。由于仅需要对修改过的数据进行刷新，因而其实现难度与复杂性要大于数据抽取。

**一般数据刷新的方法包括：**

* + 时间戳

适用情况

若数据库中的记录有时间属性，则可根据OLTP数据库中的数据有无更新，以及在执行更新操作时数据的修改时间标志来实现数据仓库中数据的动态刷新。

缺点:大多数数据库系统中的数据并不含有时间属性。

* + DELTA文件

适用情况

有些OLTP数据库的应用程序在工作过程中会形成一些DELTA文件以记录该应用所作的数据修改操作，可根据该DELTA文件进行数据刷新。

优点:采用此方法可避免对整个数据库的对比扫描，具有较高的刷新效率。

缺点:这样的应用程序并不普遍，修改现有的应用程序的工作量又太大。

* + 建立映象文件

实现方法

在上一次数据刷新后对数据库作一次快照

在本次刷新之前再对数据库作一次快照

比较两个快照的不同，从而确定数据仓库的数据刷新操作。

缺点:需要占用大量的系统资源,可能较大地影响原有数据库系统的性能

* + 日志文件

实现方法

一般OLTP数据库都有日志文件，可根据OLTP数据库的日志信息来实现数据仓库的数据刷新。

优点:日志是OLTP数据库的固有机制,不会影响原有OLTP数据库的性能,具有比DELTA文件和建立映象文件更高的刷新效率

**数据周期定义：**

指从操作型环境中的数据发生变化起，到这个变化反映到数据仓库中所用到的时间。

通常，数据周期应该不低于24个小时，因为：

操作型环境与数据仓库环境结合得越紧密，那么所需技术也就越昂贵越复杂。

一个更有说服力的原因是：时间间隔给环境附加了一个特殊的限制。间隔24小时，使得不必要在数据仓库环境中做操作性处理；也不必在操作型环境做数据仓库处理，间隔如果太短了可能会达不到这种效果。时间间隔的一个好处是能够保证在转入到数据仓库之前，数据可以达到稳定。数据在进入数据仓库之前进行调整十分简单。而如果数据被送到数据仓库中之后，一旦发现必须对这些数据进行调整，就必须在操作型环境和数据仓库中同时调整。

**原子层：**数据仓库的基础, 上层分析型应用的数据来源，所有战略分析型数据的基础。

**具有以下特点**

原子层保持历史集成性

原子层拥有数据仓库的最低细节（粒度）数据

原子层的构建是迭代的

原子层的数据结构是面向企业的

原子层可以是集成的

原子层是静态的

**粒度：**

对数据仓库中的数据的综合程度的一个度量

既影响数据仓库中数据量的多少

也影响数据仓库能够回答询问的种类

**应对不同级别的粒度要求：**

* 大粒度数据

快速存储设备

提高性能

* 小粒度数据

低速存储设备

满足细节查询

**样本数据库：**从数据仓库中取得的真实档案数据或轻度综合数据的一个子集

特点：

* 仓库中的一部分数据子集
* 周期性刷新
* 不能用于一般的分析目的

张三是不是我的顾客？

* 只能用于分析统计

在顾客中，多少是具有大学学历的未婚男性？

**数据仓库是多维度多层次的**

* + 维度是观察数据对象的角度
  + 层次是数据对象的综合程度

**数据仓库的数据组织形式**

* 简单堆积文件
* 轮转综合文件
* 简化直接文件
* 连续文件

**快照：**

数据仓库内部以一种称之为“快照”的数据结构为中心来组织。数据仓库中的数据记录是某一时刻生成的快照，包含多种数据类型，通常包括：

* 关键字，标志快照的关键字
* 时间，标志事件发生的时间单元
* 非关键字的主要数据，与关键字相关连的主要非关键字数据
* 二级数据。在形成快照时偶然捕获并被置入快照中的数据

当数据量不是太大，数据稳定，并且需要详细记录历史时，通过存储已发生的每次活动的详细情况，数据仓库可以跟踪每一件业务事件。否则，需要存放集成数据

快照的生成由一些离散活动的发生而触发，或由规律性的时间推移而触发

引发快照的业务事件可能是一个重要活动的发生

例如，填写支票、打电话、收到货物等

在离散活动的情况下，一般是出现了一些业务活动需要记录下来

离散活动是随机发生的

一种快照触发器是时间

例如一天的结束、一周的结束、一个月的结束

与时间相关的快照的建立是有规律的并且是可以预知的

**快照由四个基本的组成部分：**

* 关键字可以唯一也可以是不唯一的，通常是复合关键字，用来识别记录和主要数据
* 时间单元通常是指快照所描述事情发生的时刻，有时，时间单元指的是捕获数据的时刻。（在有些情况下，会对事情发生的时刻和捕获时间信息的时刻加以区别，而在有些情况下则不对它们进行区别。）在由时间推移触发事件的情况下，时间元素可以暗含于而不是直接附于快照中
* 主要数据是与记录的关键字直接相关的非关键字数据。例如，假设关键字标识产品的销售，时间元素描述的是销售活动终结的时刻，主要数据描述的是销售什么产品以及销售的价格、条件、地点和代理等
* 作为快照一部分而被捕获的，但与主要数据和关键字都无直接关系的二级数据（可选）。二级数据表示快照记录创建时捕获的外来信息。如与销售相关的二级数据是关于被售产品的一些附带信息。将来可能会在DSS处理过程中使用到的任何附带信息都可以加入到数据仓库记录中去

**元数据**

元数据是关于数据的数据，是如何管理数据仓库的重要数据。

描述了数据的结构、内容、编码、索引等内容

表名，系统名，索引，数据结构，编码，ID……

种类

关于数据源的元数据

关于数据模型的元数据

关于数据仓库映射的元数据

关于数据仓库使用的元数据

元数据是数据仓库的一个重要组成部分，处于数据仓库的上层，并且记录数据仓库中对象的位置。一般，元数据存储记录了以下内容：

* 数据仓库程序员所知道的数据结构。
* DSS分析员所知道的数据结构。
* 数据仓库的源数据。
* 数据进入数据仓库时发生的转换。
* 数据模型。
* 数据模型和数据仓库的关系。
* 抽取数据的历史记录。

描述了数据仓库中有什么数据以及数据之间的关系。它们是用户使用、管理数据仓库的基础

可以支持用户从数据仓库中获取数据

**元数据管理：**

元数据类似于数据库管理系统中的数据字典，主要用于数据的抽取与刷新操作，是数据抽取与刷新的基本依据。

元数据一般存放于数据仓库中并受元数据管理系统的管理，这被称为元数据管理。

一个完整的数据仓库/数据集市体系结构一般由三个层次组成,它们是：

数据源

数据仓库

数据集市（Data Mart）

三者之间通过数据仓库管理软件联系起来构成一个完整的数据体系。

数据仓库与数据集市的关系类似于传统关系数据库系统中的 基表 与 视图 的关系。数据集市的数据来自数据仓库，它是数据仓库中数据的一个部分与局部，是一个数据的再抽取与组织的过程。

**数据集市具有以下特点**

* 面向主题（Subject-oriented)
* 存储了预聚集数据
* 特定分析需求或用户群快速获取信息
* 体现终端用户的观点，面向数据仓库的界面
* 多层次、多维度

数据仓库与数据集市的关系

* 自顶向下的结构
* 自底向上的结构
* 总线结构的数据集市
* 企业级数据集市结构

自顶向下的结构

* 构建企业数据仓库
* 基于企业数据仓库构建数据集市
* 优点
  + 建立数据集市能够减轻DW访问负载
  + 各部门可以任意处理数据
  + 数据转换和整合在DW阶段统一完成
  + 数据缓冲功能
* 缺点
  + 成本高、见效慢、数据集市间不共享资源

自底向上的结构

* 构建数据集市
* 基于数据集市构建企业数据仓库
* 优点
  + 见效快、启动资金少
* 缺点
  + 各个部门都要进行数据清理整合
  + 可能造成“蜘蛛网”、数据不一致等问题
  + 并且总体上没有节约资金

总线结构的数据集市

* + 不建立数据仓库而直接建立数据集市
  + 各个数据集市不是孤立的，相互之间通过一种共享维表和事实表的“总线结构”紧密联系在一起。
* 优点
  + 共享维表和事实表，解决了建立数据集市的许多问题
* 缺点
  + 这种结构基于多维模型，应用限制于OLAP
  + 多个数据源直接影响多个集市造成结构不十分稳定

**数据资源的利用有三种方式**

* 数据资源的查询服务
* 数据资源的演绎

演绎数据库，知识库

统计分析软件（SAS，SPSS）

* 数据资源的归纳

数据挖掘

数据仓库数据的间接访问

参加ppt 4 26

|  |  |  |
| --- | --- | --- |
| 比较项目 | OLAP | OLTP |
| 应用基础 | 数据仓库 | DBMS |
| 用户 | 决策者（高级管理人员） | 一般操作员（管理人员） |
| 目的 | 为决策和管理提供支持 | 为日常工作服务 |
| 数据特征 | 导出数据 | 原始数据 |
| 数据细节 | 综合性数据，细节程度低 | 细节程度高 |
| 时间特征 | 历史数据，横跨一个时段 | 当前数据 |
| 更新方法 | 周期性刷新 | 可实时更新 |
| 数据量需求 | 一次处理需大量数据 | 一次处理需少量数据 |

OLAP中的几个基本概念

对象（Object）

* 在分析型处理中，我们所关心和分析的对象
* 观察对象又被称为 “度量值”
  + 在多维数据集中，度量值是一组值，而且通常为数字值。
  + 度量值是所分析的多维数据集的中心值，即：度量值是最终用户浏览多维数据集时重点查看的数值型数据。
  + 度量值的选择取决于最终用户所请求的信息类型。

维（Dimension）

* 观察分析对象的角度
* 例如：可以从三个‘维’角度观察‘销售金额’这个对象
  + 时间维
    - 可从时间角度统计（所有）商品在不同时间段内的销售（总）金额，以便于分析其与时间之间的关系
  + 商品维
    - 根据商品的分类情况统计每一类商品的销售金额，以便于分析其与商品类型之间的关系
  + 地域维
    - 可根据每个连锁店所在的地域统计其销售（总）金额，以便于分析其与地域之间的关系

层（Layer）

* 在分析型应用中，对分析对象可以在不同的深度层面上进行分析与观察，并可能得到不同的分析结果。因此，‘层’ 反映了对分析对象的观察深度。

维成员

* 维的一个取值称为该维的一个‘维成员’
  + 如果一个维是多层次的，则该维的‘维成员’可以是
    - 在不同维层次上的取值的组合
      * 例如：对具有日，月，年三个层次的‘时间维’来说，‘某年某月某日’、‘某年某月’、‘某月某日’、‘某年’都是其维成员，如：

1998年，1月，1998年1月，1998年1月1日，1月1日

* + - 在某个维层次上的取值
      * 例如：‘地域’维中的‘江苏’，‘南京’，......
      * 例如：‘商品’维中的‘电视机’，‘服装’，......
  + 对一个度量值（分析‘对象’）来说，维成员是该数据项在某维中位置的描述。

数据单元（单元格）

* 当多维数组的每一维都选中一个维成员，这些维成员的组合就唯一确定了一个观察对象的值，即：

（维成员1，维成员2，……，维成员n，对象值）

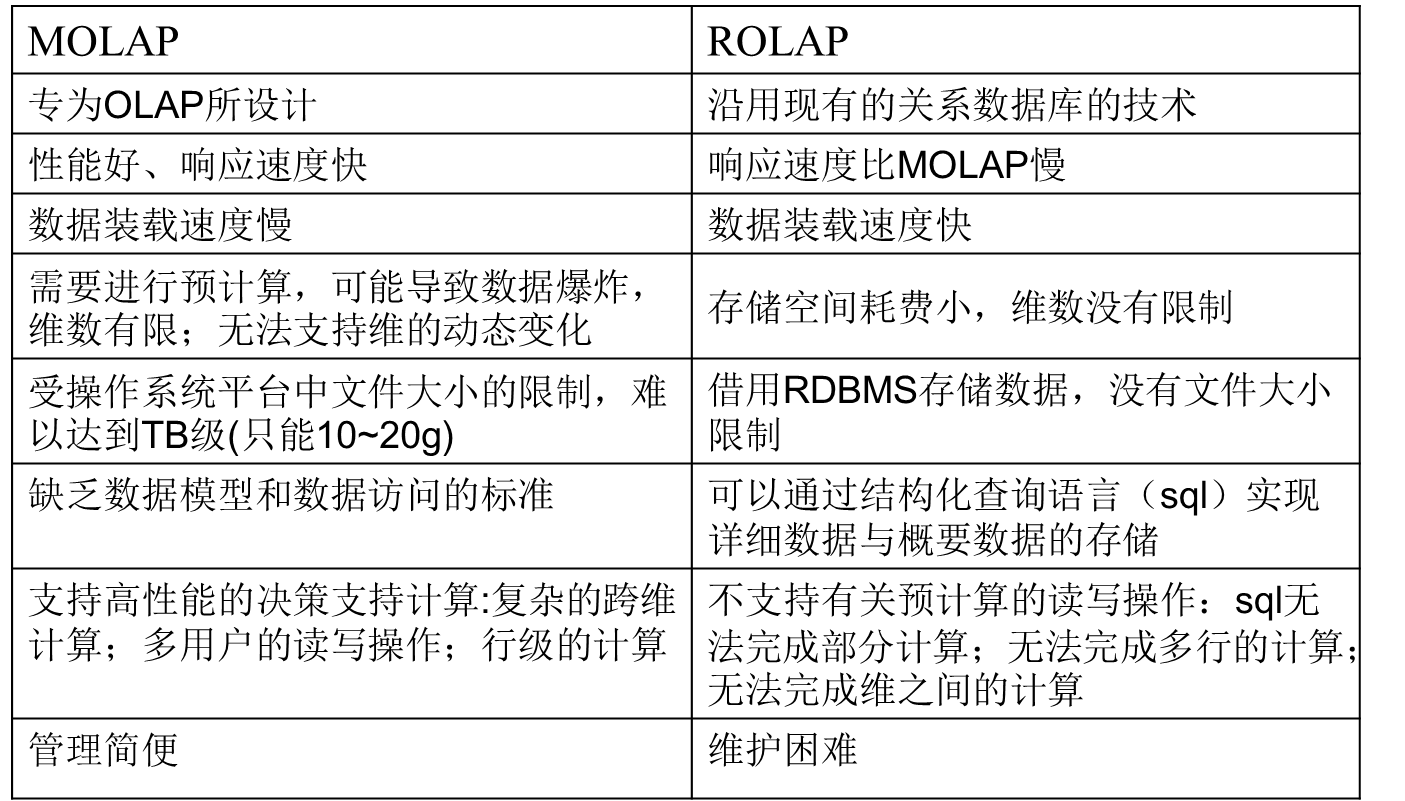
* 这样一个值或存放该值的地方我们称其为一个‘数据单元’

**Codd关于OLAP的评价准则**

1. OLAP必须提供多维概念视图
   * 从多个维度考察对象
2. 透明性准则
   * OLAP在体系结构中的位置对用户是透明的
   * OLAP的数据源对用户也是透明的
3. 存取能力准则
   * 将OLAP的概念视图映射到异质的数据存储上，能访问数据并执行所需转换，从而提供单一、完整、连续的用户视图。
4. 稳定的报表功能
   * 数据的维数和数据的综合层次增加时，提供给最终分析人员的报表能力和响应速度不应该有明显的降低和减慢
5. 客户/服务器体系结构
   * 服务器保证透明性和建立统一的公共概念模式、逻辑模式和物理模式。
   * 客户端负责应用逻辑和界面
6. 维的同等性原则
   * 每一数据维在数据结构和操作能力上都是等同的
7. 动态的稀疏矩阵处理准则
   * OLAP工具必须使得模型的物理模式充分适应指定的维数，尤其是特定模型的数据分布
8. 多用户支持能力准则
   * OLAP工具必须提供并发访问、数据完整性及安全性机制
9. 非受限的跨维操作
   * 对于多维数据之间存在的固有的层次关系，OLAP工具应自己推导而不是由用户明确定义相关计算
   * 对于无法从固有关系中得出得计算，提供计算完备的语言来定义各种计算公式
10. 直观的数据操作
11. 灵活的报表生成
12. 不受限维与聚集层次
    * 维数不应小于15
    * 任意聚集层次

另一个评价标准：FASMI （Fast Analysis of Shared Multidimensional Information）

MOLAP vs ROLAP



**混合联机分析处理（HOLAP）:**

* 试图将 MOLAP 和ROLAP进行融合
* 从而在大量数据上获得高效率。
* 方式
  + 同时提供多维数据库（MDDB）和关系数据库（RDB）。
  + 将 RDB的查询结果存储到 MDDB。
  + 使用MDDB存储高层次数据，RDB存储细节数据。

**提高ROLAP处理效率的方法：**

* 采用物化视图方式

将视图这种“虚表”形式转换成实际存在的二维表，以达到快速取得综合数据的目的。

* 采用特殊的索引与集簇方式，以加速星型模式内表的联接速度
* 尽量采用并行操作方式以提高处理速度
* 采用OLAP中的查询优化技术，如共享排序技术等。
* 采用增量技术，在OLAP数据更新时保留不变的数据，仅更改变动的数据以加快数据更新速度

**星型模式**

* 星型模式是一种多维表结构，它一般由两种不同性质的二维表组成：
  + 事实表（fact table）
    - 它存放多维表中的主要事实，我们称其为量（Measure）
  + 维表（Dimension Table）
    - 用以存放多维表中的维成员的取值
* 一般一个n维的多维表往往有n个维表和一个事实表，它们构成了一个星形结构，因而称其为‘星型模式’。
  + 在星型模式中主体是事实表，而有关维的细节则构作于维表内以达到简化事实表的目的，事实表与维表间有公共属性相连以使它们构成一个整体。

**雪花模式**

* 雪花模型是对星型模型的扩展
  + 雪花模型对星型模型的维表进一步层次化，原来的各维表可能被扩展为小的事实表，形成一些局部的“层次”区域。
* 优点
  + 最大限度地减少数据存储量，使维表尽可能地规范化。
* 缺点
  + 执行查询需要更多的连接操作，可能会影响查询性能。

**物化视图**

为了提高对统计信息的查询速度，我们可以预先计算好数据视图中的统计信息并保存在数据仓库中，这称为“物化视图”，即将虚的视图转变成实际的视图

**数据立方体**

存放物化视图的三维数据模型叫“数据立方体”

**切片，切块，上钻，下钻**

* + 切片（Slice）
    - 根据某一维上的某个维成员值选择统计数据进行分析
  + 切块（Dice）
    - 根据某一维上的某个维成员取值的区间选择统计数据进行分析
    - 根据多个维度上的维成员取值的区间选择统计数据进行分析
  + 数据概括（roll\_up/上钻）
    - 将多维下标的取值提升到较高的概念层次上，从而形成新的统计查询结果，并进行分析。
  + 数据细化（drill\_down/下钻）
    - 将多维下标的取值降低到较低的概念层次上，从而形成更细致的统计查询结果，并进行分析。

**数据仓库vs.操作型数据库**

* 面向的处理类型不同
  + 面向分析vs.面向应用
* 面向的需求不同
  + 不确定的分析需求vs.确定的应用需求
* 系统设计目标不同
  + 全局一致的数据环境vs.事务处理性能
* 数据来源或系统的输入不同
  + 多种多样vs.事务相关数据
* 系统设计的方法和步骤不同
  + CLDS vs. SDLC（系统生命周期）

**数据仓库设计的原则**

* + 面向主题原则
  + 数据驱动原则
  + 原型法设计原则

**面向主题原则**

* 建立数据仓库的目的
  + 构建数据仓库的目的是面向企业的管理人员，为经营管理提供决策支持信息。因此数据仓库的组织设计必须以用户决策的需要来确定，即从用户决策的主观需求（主题）开始。
* 数据仓库中数据的组织方法
  + 为了进行数据分析首先要有分析的主题，以主题为起始点，进行相关数据的设计，最终建立起一个面向主题的分析型环境。
  + 在数据库设计中则是以客体（Object）为起始点，即以客观操作需求为设计依据。
* 例如：‘商品销售’主题
  + 建立目的
    - 管理人员能够在适当的时候，订购适当的商品，并把它们分发到适当的商店中去销售，以提高商品的销售总金额。
  + 需要执行的分析操作
    - 分析什么样的商品，在什么样的时间和商店内畅销
      * 即分析商品的销售额与商品类型、销售时间及商店位置之间的变化关系
    - 管理人员将据此决定他们的经营策略

**数据驱动原则**

* + 在数据仓库设计中，由于其所有数据均应建立在已有的数据库基础上，即是从已经存在于操作型环境中的数据出发进行数据仓库的设计，这种设计方法被称为 “数据驱动” 方法

**原型法设计原则**

* + 数据仓库系统的原始需求不明确，且不断变化与增加，开发者最初并不能确切了解到用户的明确而详细的需求，用户所能提供的无非是需求的大方向或部分需求，更不能较准确地预见到以后的需求。
  + 因此，采用原型法来进行数据仓库的开发是比较合适的，即从构建系统的基本框架着手，不断丰富与完善整个系统。

**操作型环境到数据仓库有三种装载工作要做：**

* 装载档案数据
* 装载在操作型系统中目前已有的数据
* 将自数据库上次刷新以来在操作型环境中不断发生的变化（更新）从操作型环境中装载到数据仓库中
* 时间戳
* DELTA文件
* 建立映象文件
* 日志文件

**数据仓库设计的三级数据模型**

* 概念模型
  + 为一定目标设计系统、收集信息而服务的概念型工具，是客观世界到机器世界的一个中间层次
  + E-R法
* 逻辑模型
  + 描述了数据仓库的主题的逻辑实现
  + 关系模型
* 物理模型
  + 逻辑模型在数据仓库中的实现

或理解：

* 高级模型
  + 对数据的抽象程度最大
  + E-R图
* 中级模型
  + 数据项（dis-data item set）
  + dis是E-R图的细分，大体上E-R图中的每个实体对应一个dis
* 低级模型
  + 物理数据模型

数据仓库的设计步骤

* 数据仓库设计大致有如下几个步骤：
  1. 系统规划
     + 明确主题

在数据仓库设计的开始，首先要做的事是有关分析人员需要确定具体领域的分析对象，这个对象就是主题。

主题是一种较高层次的抽象，对它的认识与表示是一个逐步完善的过程。因此，在开始时不妨先确定一个初步的主题概念以利于设计工作的开始，此后随着设计工作的进一步开展，再逐步扩充与完善。（原型设计法）

* + - 技术准备

准备具体的技术要求和物理实现环境，包括：

* 技术评估

其内容包括数据仓库的性能指标，如：

* 数据存取能力（包括管理大数据量数据的能力）
* 模型重组能力
* 数据装载能力
* 技术环境准备

在评估基础上提出数据仓库的软硬件平台要求，包括计算机、网络结构、操作系统、数据库及数据仓库软件的选购要求等

* 1. 概念设计
     + 确定系统边界
     + 确定主要的主题及其内容

确定主要的主题即明确数据仓库的分析对象，然后对每个主题的内容进行较详细的描述，包括：

* 确定主题及其属性信息
* 描述每个属性的取值情况：固定不变的/半固定的/经常变化的
* 确定主题的公共码键
* 主题之间的关系：主题间联系及其属性

在确定上述内容后，就可以用传统的实体联系模型（E-R模型）来表示数据仓库的概念数据模型

* + - OLAP等分析应用的设计
  1. 逻辑模型设计

将E－R图转换成关系数据库的二维表

定义数据源和数据抽取规则

在逻辑模型的设计过程中，需要考虑以下一些问题：

* 适当的粒度划分：粒度的划分将直接影响到数据仓库中的数据量以及所适合的查询类型，粒度划分是否适当是影响数据仓库性能的一个重要方面

一般将数据划分为：详细数据、轻度总结、高度总结三种粒度，或者采用更多级的粒度划分方法

* 合理的数据分割策略：粒度的划分将直接影响到数据仓库中的数据量以及所适合的查询类型，粒度划分是否适当是影响数据仓库性能的一个重要方面。

通常采用“时间”属性作为数据分割的依据

数据分割技术类似于数据库中的数据分片技术，其目的是为了提高数据仓库的性能选择数据分割的因素有：

数据量的大小

数据分析处理的对象（主题）

简单易行的数据分割标准

数据粒度的划分策略

* 定义合适的数据来源
  1. 物理模型设计
  2. 数据仓库生成
     + 建立数据模式
       - 根据逻辑设计与物理设计的设计结果建立数据仓库的数据模式
     + 编制数据抽取程序
       - 根据数据仓库元数据中的定义信息，编制抽取程序，将数据源中的数据作加工以形成数据仓库中的数据
     + 数据加载
       - 将数据源中的数据，通过数据抽取程序加载到数据仓库的数据模式中去
  3. 数据仓库的运行与维护
     + 在应用系统的使用过程中不断加深理解，改进主题，依照原型法的思想使系统更趋完善
     + 在系统的运行过程中，随着数据源中数据的不断变化，需要通过数据刷新操作来维护数据仓库中数据的一致性

物理模型设计

* 在逻辑模型设计基础上确定数据的存储结构、确定索引策略、确定存储分配及数据存放位置等与物理有关的内容，物理模型设计的具体方法与数据库设计中的大致相似。其目的是为了提高数据仓库系统的访问性能。常用的一些技术有：
  + 合并表
  + 建立数据序列
  + 引入冗余
  + 表的物理分割
  + 生成导出数据
  + 建立广义索引
* 规范化/反规范化

**物理模型设计 – 合并表**

* 在常见的一些分析处理操作中，可能需要执行多表连接操作。为了节省I/O开销，可以把这些表中的记录混合存放在一起，以减低表的连接操作的代价。这样的技术我们称为 合并表 。

**物理模型设计 – 建立数据序列**

考虑创建一个数据数组，这样如果数据存放在一行中，那么一次I/O就足以检索到了。通常当数列中值的数量稳定、数据是按顺序访问的、数据的创建与修改在统计上是以非常有规律的方式进行等条件都满足时，创建一个数组才是有意义的。

**物理模型设计 – 引入冗余**

在面向某个主题的分析过程中，通常需要访问不同表中的多个属性，而每个属性又可能参与多个不同主题的分析过程。因此可以通过修改关系模式把某些属性复制到多个不同的主题表中去，从而减少一次分析过程需要访问的表的数量。

**物理模型设计 – 表的物理分割**

对于访问频率较高的属性，可以单独考虑其物理存储组织，以便选择合适的索引策略和特定的物理组织方式。

对于需要频繁更新的属性，也可以单独组织其物理存储，以免因数据更新而带来的空间重组、重构等工作。

**物理模型设计 – 生成导出数据**

在原始、细节数据的基础上进行一些统计和计算，生成导出数据，并保存在数据仓库中。

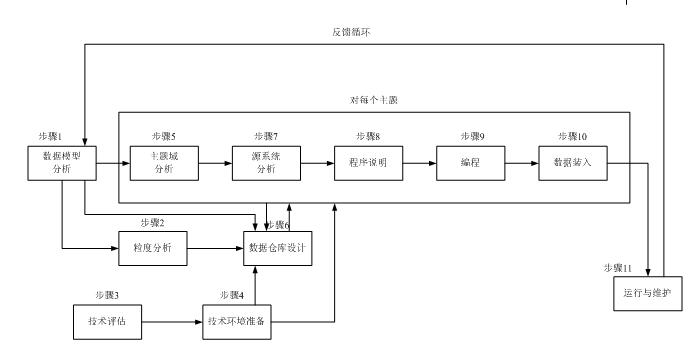
* 优点
  + 避免在分析过程中执行过多的统计或计算操作，减少输入/出的次数，提高分析操作的性能
  + 避免了不同用户进行重复统计操作可能产生的偏差

**物理模型设计 – 建立广义索引**

定义：用于记录数据仓库中数据与 “最” 有关的统计结果的索引被称为 “广义索引”。

这样的广义索引的数据量是非常小的，可以在每次进行数据仓库数据加载工作时生成或刷新这样的广义索引。用户可以从已经建立的广义索引里直接获取这些统计信息，而不必对整个数据仓库进行扫描。

**数据仓库的设计步骤：**



事实表

* 是维度建模的核心和基本表
* 每一事实表都对应着一个或若干个‘度量值’
  + 度量值是事实表的核心，也是趋势分析的对象
  + 通过事实表来记录维度值与度量值之间的关系
* 事实表中的一行对应一个度量值
  + 事实表中的所有度量值必须具有相同的粒度
  + 粒度划分：事务，周期快照，累积快照
* 最常用的度量值：数值类型
* 三种类型的度量值
  + 可做加法运算
  + 可沿着某些维度做加法运算
  + 不能做加法运算
    - 计数统计
    - 计算平均值
    - 取样统计
* 很少采用文本形式的度量值
* 度量值通常是一个可以连续取值的量
* 每个事实表都有两个或两个以上的外关键字(Foreign Key)
  + 通过外关键字建立事实表与维表之间的联系，从而可以通过维度表来存取事实表中的度量值
  + 可以由外关键字的组合构成事实表的主关键字(Primary Key)

**维度表：**

* 维度表是事实表的入口，为用户提供了使用数据仓库的接口。
* 维度表中的维度属性通常用于定义事实表上的查询条件，也可作为定义报表和统计查询的 “列”。
* 维度表的定义通常包括
  + 尽可能多的列
  + 尽可能少的行（相对于事实表）
* 通常是文本数据，或者是离散数据
* 尽量减少使用编码属性
* 维度属性与度量值（属性）的区别
* 度量值属性：有许多取值可能并可以参与统计运算的属性
* 维度属性：离散的或取值可能不多的属性；取值不变或很少产生变化的属性；从不参与统计计算但经常用作查询条件的属性

**维度建模的设计过程**

* 选取要建模的业务处理过程
  + 分析需要：数据的入口（数据驱动），管理决策需要（面向主题）
* 定义业务处理的粒度
  + 事实表中每一行的度量值的取值粒度
* 选择事实表中的维度
* 选择事实表中的度量值
  + 分析对象
  + 可以有多个度量值

确定事实：

通过计算而获得的可加性度量值也应该物理存储在事实表中，如：毛利润金额

不具有可加性的计算结果则应该由数据存取工具在访问过程中进行计算，如：毛利润率，单价，……

**维度的组合和分散：**

* 维度的组合：
  + 参与组合的维度高度相关，组合起来的维度就不会比分开的维度大许多
  + 组合起来的维度能够高效地进行浏览
* 维度的分散：
  + 在用户分开考虑时，分开的维度更加容易理解
  + 独立维度的管理对于组合维度来说，更加直截了当

退化维度

* 维度表为空，具体的维度值直接存放在事实表中

维度的规范化处理

|  |  |
| --- | --- |
| 规范化 | 非规范化 |
| 雪花模型 | 星型模型 |
| 复杂的表关系 | 简单的表关系 |
| 节省存储空间 | 记录之间存在数据冗余 |
| 连接的复杂，高开销 | 连接简单，低开销 |
| 低维度浏览能力 | 高维度浏览能力 |
| 不支持物理加速技术 | 支持物理加速技术 |

**代理关键字**，避免直接使用操作型数据作为维度表和事实表的主关键字和外关键字

* + 可以缓冲操作型数据的变化对数据仓库数据的影响
  + 性能优势
  + 操作型数据可能无法作为关键字
  + 日期维度的特殊要求
  + 历史一致性

**值链**

* + 由企业的关键业务组成
  + 值链确定了企业主体活动的自然逻辑流程

**三种互补的库存模型**

* 库存周期快照

目标：

确保合适的商场在合适的时间中存在合适的商品

可最大限度地减少脱销现象，并减少存货维护的总体开销

缺点：

发生过多少次产品入柜以后又在同一天的不同时间将它取出的情形？

从某厂家那里接收过多少次分开装运的货，以及是什么时候收到的？

哪些产品是由于出现多次检验不合格而导致向厂家退货的？

* 库存事务：频度测算和具体事务类型的计算需要库存事务模型的支持

记录影响库存水平的主要因素

常见的库存事务类型

* 产品接收
* 产品送检
* 对检验合格的产品进行分发
* 将检验不合格的产品退给厂商
* 产品入柜
* 产品销售审批
* 产品出柜
* 运输前的产品包装向顾客发货
* 从顾客那里回收产品
* 对回收产品进行封存
* 从库存中删除产品
* 库存累积快照

记录每件商品的分发历史，直至其离开仓库为止

特点：

* 库存累积快照事实表中存在多个取值为日期的外关键字
* 需要对事实表中的每一行进行多次的访问和修改操作
* 很少用于长期运行而需要不断进行补充的库存处理

库存周期快照事实表与销售事务事实表的区别

* + 销售事实表是稀疏的，而库存事实表则是稠密的
    - 在销售事实表中记录每天实际发生的商品销售情况
    - 而库存事实表则需要记录每天、每种商品、在每个商场的库存情况（不管是否发生了实际的销售事实）
  + 解决办法
    - 随着时间的推移可降低周期快照的频度
      * 最近60天内的以天为粒度单位的周期快照
      * 最近 3年内的以周为粒度单位的周期快照

半加型事实 (Semiadditive Facts)

* + 只在部分维度上具有可加性的度量值被称为‘半加型事实’
  + 在商品营销中，绝大部分的度量值在所有的维度范围内都具有极好的可加性。
  + 在库存快照模型中，‘库存量’可以跨‘产品’或‘商场’进行汇总（具有可加性），但不具有跨‘日期’的可加性。

几种常见的半加型事实

* + 库存数量，银行帐户余额，温度，水位，含量，……
  + 用于记录静态水平的度量值在跨日期维度以及可能的其它维度范围内都是不可加的。
    - 对于不可加的度量值，可用的聚集方法如：平均统计
    - 不能简单地利用SQL中的AVG函数来完成这样的平均统计计算工作
    - AVG\_DATE\_SUM

**数据仓库总线结构**

* 一种可以按增量开发方式分步建造企业数据仓库的方法
  + 计算机中的总线
  + 通过为数据仓库环境定义标准的总线接口，独立的数据集市就可以由不同的开发小组在不同的时间进行实现。只要遵循这个标准，独立的数据集市就可以插入到一起并有效地共享。
    - 数据集市

**数据仓库总线矩阵（2/2）**

* 矩阵的行：对应着数据集市
  + 如果数据来源不同，处理功能不同，或者矩阵行代表的内容无法在单个迭代过程中合理完成，就应当创建独立的矩阵行
* 矩阵的列：对应着共享的公共维度

**数据集市：**

* 一级数据集市 (first-level data marts)

基于单一数据源系统的数据集市

ETL的设计相对简单

* 合并数据集市 (consolidated data marts)

跨越业务处理过程的多源数据集市

ETL的设计相当复杂

**一致性维度**

* 一致性维度是进一步开发总线结构数据仓库系统的基础
* 一致性维度
  + 要么是同一的，要么是具有最佳粒度与细节性的维度在严格数学意义上的子集
  + 一致的维度具有如下特征
    - 一致的维度关键字
    - 一致的属性列名字
    - 一致的属性定义
    - 一致的属性值
* 一致的维度可能意味着是相同的维度表
  + 与它们相连的事实表具有完全相同的内容（不同的度量值）。例如：
    - 连接到销售事实表与库存事实表上的日期维度表是同一的，意味着销售事实表和库存事实表中的内容是相同的
  + 这样的维度表在物理上可能是同一张表，也可能是不同的表，但它们应该具有*相同数目的行、相同的关键字值、相同的属性标签、相同的属性定义与相同的属性值*。
* 原子型维度
  + 在最佳粒度层次上的维度定义（最小的粒度）
* 堆积维度/上钻维度 (roll-up dimensions)
  + 在较高层次上的维度定义（较大的粒度）
    - 每日快照 vs. 每周快照
  + 如果堆积维度是基本层次上原子型维度严格意义上的子集，则堆积维度与原子型维度保持一致。

**一致性事实**

* 同样的事实在不同的数据集市进行存储的一致性
  + 取值单位的一致性
  + 值的一致性
  + 自然关键字的一致性
* 一般说来，事实表数据不在多个数据集市明确的进行拷贝。
* 如果事实表存在于多个数据集市，那么支撑这些事实的定义和方程必须都是相同的
* 如果无法使事实完全保持一致，那么应该对不同的解释给与不同的名称

**事实表规范化的目的**

在对事实表进行规范后，可以连同标识事实类型维度一起得到单一的一般性事实量

**规范化的时机**

事实行的事实设置比较稀疏

不在事实之间施加运算

**事实表规范化的考虑：**

* 一般不考虑事实表的规范化。除非不同度量值的计算处于不同的粒度层次上，那么则需要将它们分解到不同的事实表中去
* 如果可以将“粗”粒度的度量值分配到“细”的粒度层次上，那么则可以在尽量细的粒度层次上通过统一粒度层次来建立一张统一的事实表
  + 事实表中的粒度层次越“细”，则可以提供的分析操作就越多

**日期维度的角色模仿**

* + 后台只维持一个单一的日期维度表
  + 为事实表中的每一个日期外关键字建立一个日期维表上的视图

优点：降低存储空间开销，方便使用

**三种类型事实表的比较**



**实时分区：**

在数据仓库环境中，对实时业务数据的访问需要

* + 在常规静态数据仓库的前面建立一个物理的实时分区
  + 对实时分区的约束要求
    - 包括静态数据仓库最后一次更新以来出现的所有行为
    - 尽可能无缝地连接到静态数据仓库事实表的粒度与内容上
    - 能够轻松地建立索引，以致于总是可以不断吸纳新来的数据

**三种不同类型的实时分区**

* + 事务粒度 – 当天的记录（并非统计结果）

实时分区具有与它的支撑静态事实表具有完全相同的维度

* + 周期快照 – 最近一个周期内的统计结果
* 静态数据仓库事实表具有一个周期粒度，实时分区可以看作是当前的热积月
* 实时分区是当前正在开发的月份的映像，随着月份的推进不断更新。半加性或全加性事实随报表不断调整
* 月份结束时累计起来的实时分区，作为最新月份加载到静态仓库
  + 累积快照 – 只记录最近被更新的项

实时分区仅仅包含当天更新的分列项。

当天结束时，实时分区包含了需要写到主要事实表上的记录的最新版本

**支架维度：**

* + 将一组低基数属性单独构成客户维度的一个维度（称为支架维度），从而使整个模型呈雪花状
  + 支架维度中的数据一般是从外部数据提供者那里获得的。
    - 如：县人口统计支架维度
* 使用维度支架的好处
  + 客户维度与支架维度具有相差悬殊的粒度
  + 具有不同的管理与加载次数
  + 可以节省客户维度表的存储空间
  + 如果用户的查询工具坚持使用星型结构，那么可以通过视图定义来隐藏维度支架

在数据仓库的维度模型中，部分维度属性是会随时间而发生变化的，若只是将这些变化的维度属性值作简单的修正，即在维度表中只保留该维度属性的当前值，这会直接影响到对事实表中该维度属性所对应的事实数据元组的访问，特别是无法根据维度属性值的变化情况来进行分析处理。

**维度表的划分**

* + 稳定维度
  + 渐变维度
  + 快变维度

**渐变维度的处理办法**

* + 类型1：改写属性值
  + 类型2：添加维度行
    - 在新的元组上记录修改后的属性值，同时系统将为该元组生成新的代理关键字
    - 其它维度属性上的值不变
    - 可以考虑在维度表中增加两个日期属性：维度的 *生效日期* 和 *截止日期*
  + 类型3：添加维度列
    - 在新增加的属性列上记录修改后的属性值

**快变维度的处理办法**

* + 微型维度
    - 将分析频率高或变化频率大的属性拆成为独立的微型维度。
    - 例如：客户维度中的年龄，性别，收入水平等属性，它们的每一种取值组合构成微型维度表中的一行。
  + 预设波段
    - 对于诸如收入与购买总额等不断变化的属性，应该被转换成呈波段分布的范围，即进行离散化处理，使其只能在数目相当小的离散值中取值，以减少维度表中的数据量。

**什么是数据挖掘？**

* + 定义一：数据挖掘就是对数据库(数据仓库)中***蕴涵的、未知的、非平凡的、有潜在应用价值的模式(规则)***的提取。
  + 定义二：数据挖掘就是从大型数据库(数据仓库)的数据中提取人们***感兴趣的知识***。这些知识是***隐含的、事先未知的潜在有用信息***。

**数据挖掘中的几个基本概念**

* 模式（pattern）
* 知识（discovered knowledge）
* 置信度（confidence）
* 兴趣度（interestingness）
* 非平凡性（nontrivial）
* 有效性（effectiveness）

**模式**

* 用高级语言表示的表达一定逻辑含义的信息，这里通常指数据库中数据与数据之间的逻辑关系。
  + 例如：在某超市的商品销售数据库中，我们可以找到以下信息：
    - 男性顾客在购买婴儿尿布时也往往同时购买啤酒
    - 在购买面包和黄油的顾客中，大部分的人同时也买了牛奶

**知识**

* 满足用户对兴趣度和置信度的要求的模式。

**置信度**

* 模式在某一数据集上成立的程度。
  + 例如：
    - 模式R1：***在购买面包和黄油的顾客中，大部分的人同时也买了牛奶***
      * 该模式的置信度为：同时购买‘面包、黄油、牛奶’的顾客人数占同时购买‘面包、黄油’的顾客人数的百分比，即：
* 通过数据挖掘所发现的模式的置信度的大小涉及到许多因素，如数据的完整性、样本数据的大小、领域知识的支持程度等。
  + 如果没有足够的置信度，模式不能成为知识。因此在数据挖掘过程中，通常要规定模式的***最小置信度***

**兴趣度**

* 在一定数据集上为真的知识被用户关注的程度。
  + 用户对知识感兴趣的程度也可以用知识的支持度（support）来表示。
    - 例如：模式R1的支持度为“***同时购买‘面包，黄油和牛奶’的顾客人数占总顾客人数的百分比***”，即：
* 只有当一个模式的‘***兴趣度***’达到一定的程度时，那么该模式才是一个有意义的模式，才能引起用户的注意，有助于用户的决策制订过程。
  + 因此，在数据挖掘过程中也要规定模式的‘***最小支持度***’，以淘汰哪些在极少情况下才会出现的模式。

**非平凡性**

* 平凡知识
  + 能够以确定的计算过程提取的模式称为平凡知识。
    - 例如：根据数据库中的薪水字段求得职员的平均薪水
  + 平凡的知识不是数据挖掘的目标。
* 在数据挖掘中，知识的发现过程都应具有某种不确定性和一定的自由度，也就是要发现*不平凡的知识*。

**有效性**

* 知识的发现过程必须能够有效地在计算机上实现，分为时间和空间有效性。

**数据挖掘的特点：**

* 数据挖掘要处理大量的数据
  + 它所处理的数据库(数据仓库)的规模十分庞大，达到GB、TB，甚至更大
* 由于用户不能形成精确的查询要求，因此要依靠数据挖掘技术为用户找寻他可能感兴趣的东西
* 在数据挖掘过程中，规则的发现基于统计规律；
  + 所发现的规则不必适用于所有数据，而是当达到一定的“门槛”时，即认为具有此规则。因此，利用数据挖掘技术可能会发现大量的规则
* 数据挖掘所发现的规则是动态的，它只反映了当前状态的数据集合具有的规则。
  + 随着不断地向数据库（数据仓库）中加入新数据，需要不断地重新进行数据挖掘以更新所发现的规则

**利用数据库系统进行数据挖掘的缺点：**

* 动态数据（Dynamic Data）
  + 大多数数据库的基本特点是内容将经常变化。在一个在线系统中，必须采用预警机制来保证数据库中的这些变化不会导致错误的数据挖掘结果
* 噪声和不确定性（Noise & Uncertainty）
  + 噪声数据
    - 数据库中的错误数据和异常现象
  + 不确定性
    - 发现的模式可能只在一部分数据上有效
* 冗余信息（Redundant Information）
  + 同一数据在数据库中的多处出现。
  + 这种冗余信息有时会误导知识的发现过程。
    - 有可能会“夸大”某个模式的置信度，从而导致发现大量的无意义的模式。



* + - 也有可能“低估”一个模式的的兴趣度，从而导致丢失一些有意义的模式。



* 不完整数据（Incomplete Data）
  + 由于不完整的数据域和数据域上值的缺少造成的不完整数据当然会影响发现的结果。
  + 数据库的最初设计并没有考虑知识发现的应用，模式的发现、评价、解释很可能需要在当前数据库中并不存在的信息。
* 稀疏数据（Sparse Data）
  + 数据库中的信息在实例空间中可能是稀疏的，这会严重影响发现的效率

**数据挖掘的步骤：**

**1数据集成**

* 在数据仓库数据的加载过程中，一般需要需要对数据作以下的预处理：
  + 数据清理
    - 填补丢失的数据
    - 清除噪声数据
    - 修正数据的不一致性
  + 数据集成
  + 数据转换
    - 我们收集到的数据有时并不一定适合数据挖掘的需要。如已有的挖掘方法可能无法处理这些数据，存在一些不规则的数据，或者数据本身不够充分等，因此需要对收集到的数据进行转换

**2数据归约：**有时用于数据挖掘的数据量是非常巨大的，通过数据归约技术可以减低数据量，提高数据挖掘操作的性能

常见的数据归约技术有：

* 挖掘范围的选择
  + 在不影响挖掘结果的前提下，尽可能地选取哪些与挖掘操作有关的属性集参与到数据挖掘中。
* 数据压缩
  + 通过数据压缩技术可以减低数据的规模，节省存储空间开销和数据通讯开销。
  + 如果采用的数据挖掘算法不需要解压就可以直接利用那些压缩数据进行数据挖掘，那么数据压缩技术将是非常有用的。
* 离散化处理
  + 如果一个属性的值域是一个连续区域，我们可以将它划分为若干个区域，然后用每个区域的标识值来代替原来的值。这样可以减低该属性上属性值的个数。
  + 我们也可以利用这种数据归约技术来自动地建立该属性的概念层次树

**3挖掘**

根据挖掘要求选择相应的方法与相应的挖掘参数，如支持度、置信度参数等，在挖掘结束后即可得到相应的规则

**4评价**

经过挖掘后所得结果可能有多种，此时可以对挖掘的结果按一定标准作出评价，并选取评价较高者作为最终结果

**5表示**

数据挖掘结果的规则可在计算机中用一定形式表示出来，它可以包括文字、图形、表格、图表等可视化形式，也可同时用内部结构形式存储于知识库中供日后进一步分析之用

**常用的数据挖掘方法**

* + - 特征规则挖掘：是一种常见的知识形式，它用于描述一类数据对象的普遍特征，是普化知识的一种
      * 面向属性归约方法
      * 数据立方方法
    - 关联规则挖掘
    - 序列模式分析
    - 分类分析
    - 聚类分析

**面向属性归约方法**

* 这是一种常用的特征规则的挖掘方法。
  + 它通过对属性值间概念的层次结构进行归约，以获得相关数据的概括性知识，通常又称为普化知识。
* 在实际情况中，许多属性都可以进行数据归类，形成概念汇聚点。
  + 这些概念依抽象程度的不同可构成描述它们层次结构的概念层次树。
  + 根据概念层次树可以对供挖掘用的数据进行预处理，以生成一个适合于进行数据挖掘工作的数据集。因此‘面向属性’的数据规约过程实际上是为进行数据挖掘工作而进行的数据预处理。

**数据挖掘前的预处理**

* + 在开始特征规则挖掘之前，需要从概括关系表中剔除那些出现频率过低的噪声数据（宏元组）。以减少数据挖掘所处理的宏元组的数量，提高挖掘的效率；同时也避免得到过多的兴趣度不满足要求的挖掘结果
  + 通常我们会定义一个噪声数据的阀值M
    - M通常指概括关系表中的count属性的值
    - 如果某个宏元组在count属性上的取值小于或等于M，则该宏元组将被看作为噪声数据，不参与后续的数据挖掘过程
      * 虽然噪声数据不参与挖掘过程，但并不能就此从概括关系表中删除噪声数据
      * 在计算规则的兴趣度(支持度)时需要包括这些噪声数据

**概括关系表：**

* 概括关系表通过基本关系表规约而来，其属性包括：
  + 目标数据集中参与数据挖掘的一个或多个属性
    - 每一个属性都通过相关的概念层次树进行了规约
  + 系统为每个概括关系表新增加的一个COUNT属性
* 基本关系表中的元组被称为“基本元组”，而概括关系表中的元组则被称为“宏元组”。
  + 一个宏元组概括了多个基本元组，其中的COUNT属性被用来记录该宏元组所概括的基本元组数
* 在概括关系表上进行数据挖掘的优点
  + 可以通过面向属性的规约方法进行数据的预处理工作，以规范化属性的取值；
    - 填补缺少的属性值，剔除噪声数据
    - 将不适宜数据挖掘工作的属性值进行转化
  + 根据概括关系表中的属性对基本关系表进行规约，可以限制每个属性可以取值的数量，从而将概括关系表中的宏元组的数量控制在一个合适的范围内，以提高数据挖掘的性能
  + 可以在不同的概念层次上进行数据挖掘
    - 在不同概念层次上进行数据挖掘，完全可能获得不同的挖掘结果

**数据立方方法：**预先做好某些经常需要用到但花费较高的统计、求和等集成计算，并将统计结果存放在多维数据库中，在构造概括关系表时，就可以直接从多维数据库中获得所需要的统计结果，从而节省数据规约的时间，提高数据挖掘的效率

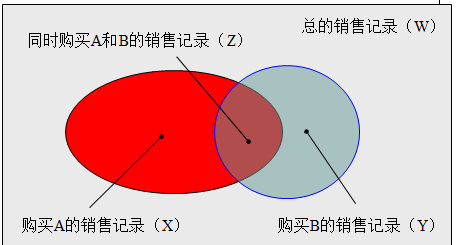
* 在数据立方方法中，常用的分析方法有：
  + 数据概括（roll\_up上翻）
    - 将属性值提升到较高的概念层次上
      * 如：从‘基本关系表’ 到 ‘概括关系表一’， 再到‘概括关系表二’的分析过程。
  + 数据细化（drill\_down下翻）
    - 将属性值减低一些层次
      * 如：从‘概括关系表二’ 到 ‘概括关系表一’， 再到‘基本关系表’的分析过程。
        + 要降低属性值的概念层次，通常需要在最初的基本关系表中重新进行统计计算。除非在多维数据库中已经保存有所需要的概念层次上的统计结果。

**特征规则挖掘 vs. OLAP**

* 特征规则挖掘是由参数主导的自动化过程，而OLAP是由分析人员主导的人工过程
* 在特征规则挖掘过程中，算法可以在阀值的指导下：
  + 自动决定排除冗余以及和当前挖掘任务无关的属性
  + 自动决定各个属性规约的层次
  + 在对比集的指导下，在挖掘结果中去除与当前挖掘任务关联不大的属性

**关联规则挖掘：**

* 关联规则（Association Rule）
  + 关联规则用于表示事务数据库中诸多属性之间的关联程度。而关联规则挖掘则是利用数据库中的大量数据通过关联算法寻找属性间的相关性
    - ‘属性’在这里也被称为‘项’（Item），若干个属性所构成的一个属性集也被称为一个‘项集’（Item Set）
  + 例：(超级市场)在购买商品A的客户中有90%的人会同时购买商品B，则可用关联规则表示为：
    - * 规则R1： A → B



* 支持度(Support)(A->B)=Z/W
* 置信度(Confidence)(A->B)=Z/X

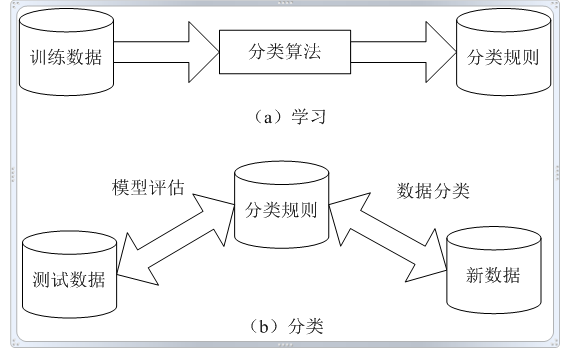
***Apriori算法!!!(P60)***

**序列模式分析**

* 序列模式分析与关联规则挖掘类似，也是为了找出数据对象之间的联系，但序列模式分析法的侧重点是为了找出数据对象之间的前因后果关系。
  + 被分析对象具有前后的时序关系
* 例如：
  + 下雨 ---- 洪涝
  + 电筒 ---- 电池

**分类分析**

* **数据分类是一个两个步骤的过程：**
  + 第1步：建立一个模型，描述给定的数据类集或概念集（简称训练集）。通过分析由属性描述的数据库元组来构造模型。每个元组属于一个预定义的类，由类标号属性确定。用于建立模型的元组集称为训练数据集，其中每个元组称为训练样本。由于给出了类标号属性，因此该步骤又称为有指导的学习。如果训练样本的类标号是未知的，则称为无指导的学习（聚类）。学习模型可用分类规则、决策树和数学公式的形式给出。
  + 第2步：使用模型对数据进行分类。包括评估模型的分类准确性以及对类标号未知的元组按模型进行分类。

****

**决策树（Decision Tree）**又称为判定树，是运用于分类的一种树结构。其中的每个内部结点（internal node）代表对某个属性的一次测试，每条边代表一个测试结果，叶结点（leaf）代表某个类（class）或者类的分布（class distribution），最上面的结点是根结点

* **使用决策树进行分类分为两步：**
  + 第1步：利用训练集建立并精化一棵决策树，建立决策树模型。这个过程实际上是一个从数据中获取知识，进行机器学习的过程。
  + 第2步：利用生成完毕的决策树对输入数据进行分类。对输入的记录，从根结点依次测试记录的属性值，直到到达某个叶结点，从而找到该记录所在的类。

**聚类分析**

* 聚类分析又称集群分析，它是研究分类问题的一种多元统计方法
* 聚类分析分为距离聚类和相似系数聚类
* **聚类分析与分类分析相反：**
  + 首先输入的是一组没有被标记的记录，系统按照一定的规则合理地划分记录集合（相当于给记录打标记，只不过分类标准不是用户指定的）
  + 然后可以采用分类分析法进行数据分析，并根据分析的结果重新对原来的记录集合（没有被标记的记录集合）进行划分，进而再一次进行分类分析，如此循环往复，直到获得满意的分析结果为止。