数据仓库与数据挖掘

Data warehouse and data mining

丁钰

yuding@njau.edu.cn

南京农业大学人工智能学院

第四章: 数据仓库与OLAP

第四章：数据仓库和联机分析处理

- 数据仓库基本概念
- 数据仓库建模-数据立方体和OLAP
- 数据仓库的设计和使用
- 数据仓库实现
- 总结

什么是数据仓库?

- 有许多不同的定义，但没有严格的定义
  - 一个决策支持数据库，与组织的业务数据库分开维护
  - 提供一个坚实的综合历史数据平台进行分析，支持信息处理。
- “数据仓库是一个面向主题的、集成的、时变的、非易失性的用于支持管理者决策过程的数据集合"。

数据仓库--面向主题

- 围绕主要主题组织，如客户、产品、销售
- 专注于为决策者建立模型和分析数据，而不是日常运作或交易处理
- 通过排除在决策支持过程中无用的数据，围绕特定的主题问题提供一个简单且简洁的视图

数据仓库-集成的

- 基于集成多个、异构的数据源进行构建
  - 关系数据库、一般文件、联机事务处理记录
- 应用数据清理及数据集成技术
  - 确保不同数据源中的命名约定、编码结构、属性度量等方面的一致性

    例如，宾馆价格：货币种类、税额、是否含早餐等等
- 当数据被移入数据仓库时将会被转换

数据仓库-时变的

- 数据仓库涵盖的时间范围要显著长于业务操作系统数据
  - 业务操作数据库数据：实时数据
  - 数据仓库数据：从历史角度提供信息（例如，过去的5-10年）
- 数据仓库中的每个关键结构
  - 隐式或显式地包括时间元素
  - 但是业务数据库中的关键结构既可包括也可以不包括“时间元素”

数据仓库-非易失的

- 独立性
  - 数据仓库将业务环境中的数据转换并在物理上分离存储
- 静态 数据仓库环境中不发生数据的操作更新
  - 不需要事务处理、恢复和并发控制机制
  - 在数据访问中只需要两个操作。
    - *数据的初始加载*和*数据的访问*

OLTP与OLAP

- OLTP：联机事务处理
  - DBMS操作
  - 查询和事务性处理
- OLAP：联机分析处理
  - 数据仓库操作
  - 钻孔、切片、切块，等等。

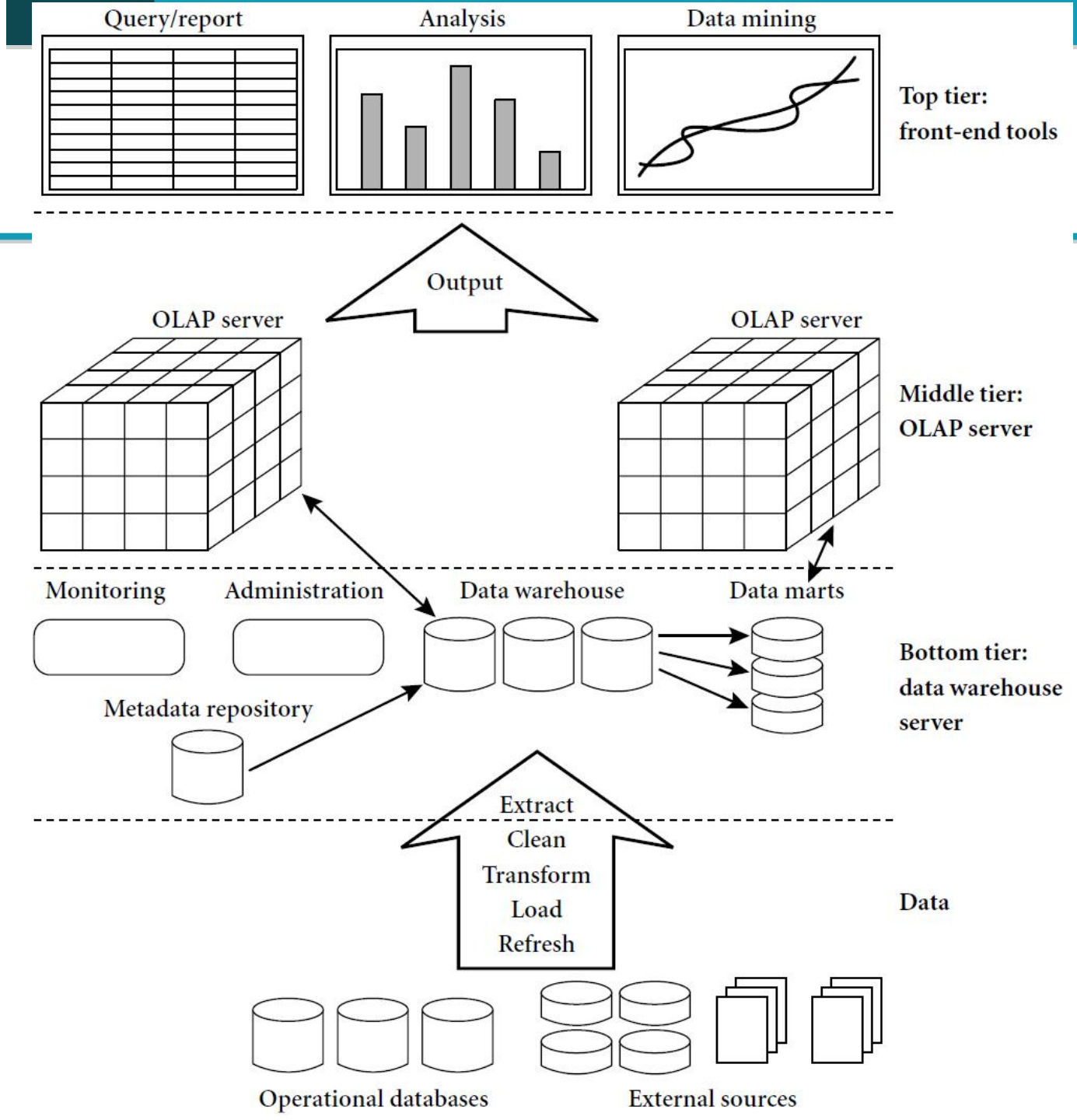
	联机事物处理(OLTP)	联机数据分析(OLAP)
用户	办事员、IT专业员工	知识工人（分析人员）
功能	日常操作	决策支持
数据库设计	面向应用的	面向主题的
数据	当前的、最新的 详细的、关系型的 独立的	历史的， 总结的，多维度的 集成的，整理过的
用法	重复的	专门的
访问方式	读/写 主键的索引/哈希	大量的浏览
工作单元	短的，简单事务处理	复杂查询
访问记录数量	数十	数百万
用户数量	数千	数百
数据库规模	100MB-GB	100GB-TB
度量	事务吞吐量	查询吞吐量、响应时间

为什么要建立一个独立的数据仓库?

- 为了两个系统都有很高的性能
  - DBMS-目的是OLTP：存取方法、索引、并发控制、恢复
  - 数据仓库-目的是OLAP：复杂的OLAP查询、多维视图、合并统一
- 不同的功能和不同的数据。
  - 缺少数据：决策支持需要历史数据，而业务数据库通常不维护这些数据。
  - 数据整合：决策支持需要将来自异种数据源的数据统一（聚合、汇总）。
  - 数据质量：不同的来源通常使用不一致的数据表示、代码和格式，必须加以协调

数据仓库：一个多层次的架构

- **顶层：前端客户层**
- 包括数据挖掘工具（如趋势分析、预测等）、数据分析工具和查询与报告工具使用OLAP相关模型将多维数据上的操作映射为标准的关系操作，或者直接实现多维数据操作。
- 用于知识工人（如经理、主管、分析人员等）直接操作获取知识
- **中间层：OLAP服务器**
- 联机分析处理（Online Analytical Processing, OLAP）是数据仓库系统前端分析服务的分析工具，能快速汇总大量数据并进行高效查询分析，为分析人员提供决策支持。
  - 使用OLAP相关模型将多维数据上的操作映射为标准的关系操作，或者直接实现多维数据操作
  - OLAP操作可以与关联、分类、预测、聚类等数据挖掘功能结合，以加强多维数据挖掘
- **底层：数据仓库服务器**
- 使用一些后端工具和实用程序，对其他外部数据源的数据进行提取、清理、变换、装入和刷新，将高质量的数据更新到数据仓库。
- 数据集市，也叫数据市场，是一个从操作的数据和其他的为某个特殊的专业人员团体服务的数据源中收集数据的仓库，是数据仓库的子集。
- **数据**



三种数据仓库模式

- **企业仓库**
  - 收集横跨整个组织的所有主体信息
- **数据集市**
  - 对特定用户群有价值的全组织数据的一个子集
  - 其范围局限于特定的、选定的群体，如营销数据集市。
    - 独立与依赖（直接来自仓库）的数据集市
- **虚拟仓库**
  - 一组关于业务数据库的视图
  - 只有一些可能的摘要视图可能会被具体化

提取、转换和加载（ETL）

- **数据抽取**（extract）
  - 从多个、异构的和外部来源获得数据
- **数据清理**
  - 检测数据中的错误，并在可能的情况下纠正它们
- **数据转换**（transform）
  - 将数据从遗留格式或主机格式转换为仓库格式
- **加载**（load）
  - 排序、汇总、合并、计算视图、检查完整性，以及建立索引和分区
- **刷新**
  - 将更新从数据源传播到仓库。

元数据存储库

- 元数据是定义数据仓库对象的数据
- 元数据包括以下内容
  - 数据仓库结构的描述

    模式、视图、维、分层结构、导出数据的定义、数据集市的位置及内容
  - 操作数据源

    数据血统（迁移数据的历史和它使用的变换序列），数据流通（主动的、档案的或者净化的）和管理信息（仓库使用的统计量、错误报告和审计跟踪）
  - 用于汇总的算法
  - 由操作环境到数据仓库的映射
  - 关于系统性能的数据

    数据仓库模式、视图和导出数据的定义
  - 商务数据

    商务术语和定义、数据拥有者信息、收费策略

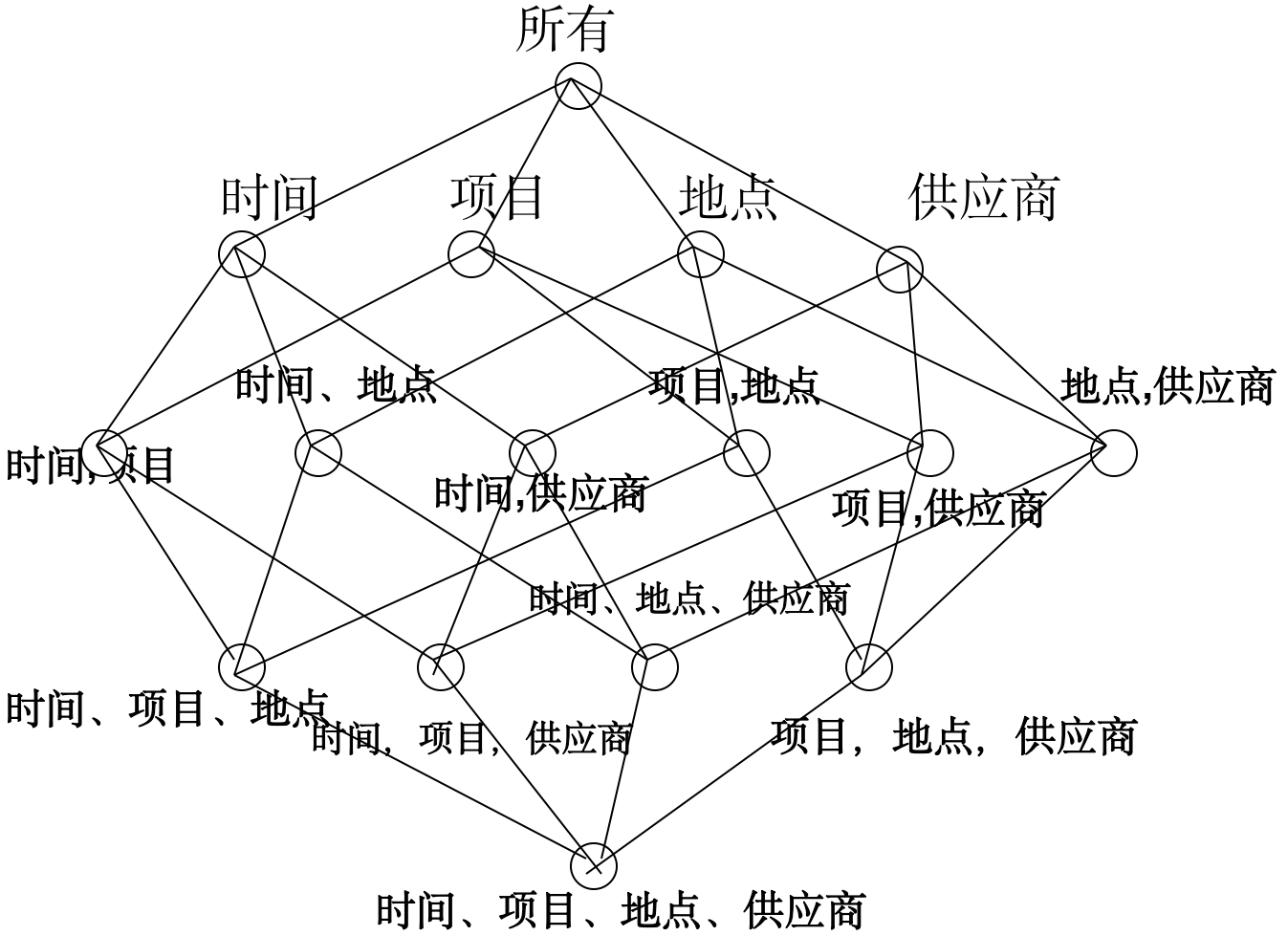
第四章：数据仓库和联机分析处理

- 数据仓库基本概念
- 数据仓库建模-数据立方体和OLAP
- 数据仓库的设计和使用
- 数据仓库实现
- 总结

从表格和电子表格到数据立方体

- **数据仓库**基于多维数据模型，多维数据模型将数据视为数据方(data cube)形式
- 数据方，如销售数据，允许在多个维度上对数据进行建模和查看
  - **维度表**，维度表用于描述维度;它们包含维度键、值和属性 如项目（项目名称、品牌、类型），或时间（日、周、月、季、年）。
  - **事实表**，事实表是包含感兴趣度量的表（如销售金额）和每个相关维度表的键。
- **数据方**。一个立方体的网格
  - 在数据仓库文献中，一个n-D基立方体被称为**基本方体**
  - 最上面的0-D立方体，拥有最高级别的总结，被称为**顶点方体。**
  - 方体的格构成了数据方。

立方体：方体的格



0-D顶点方体

1-D方体

2-D方体

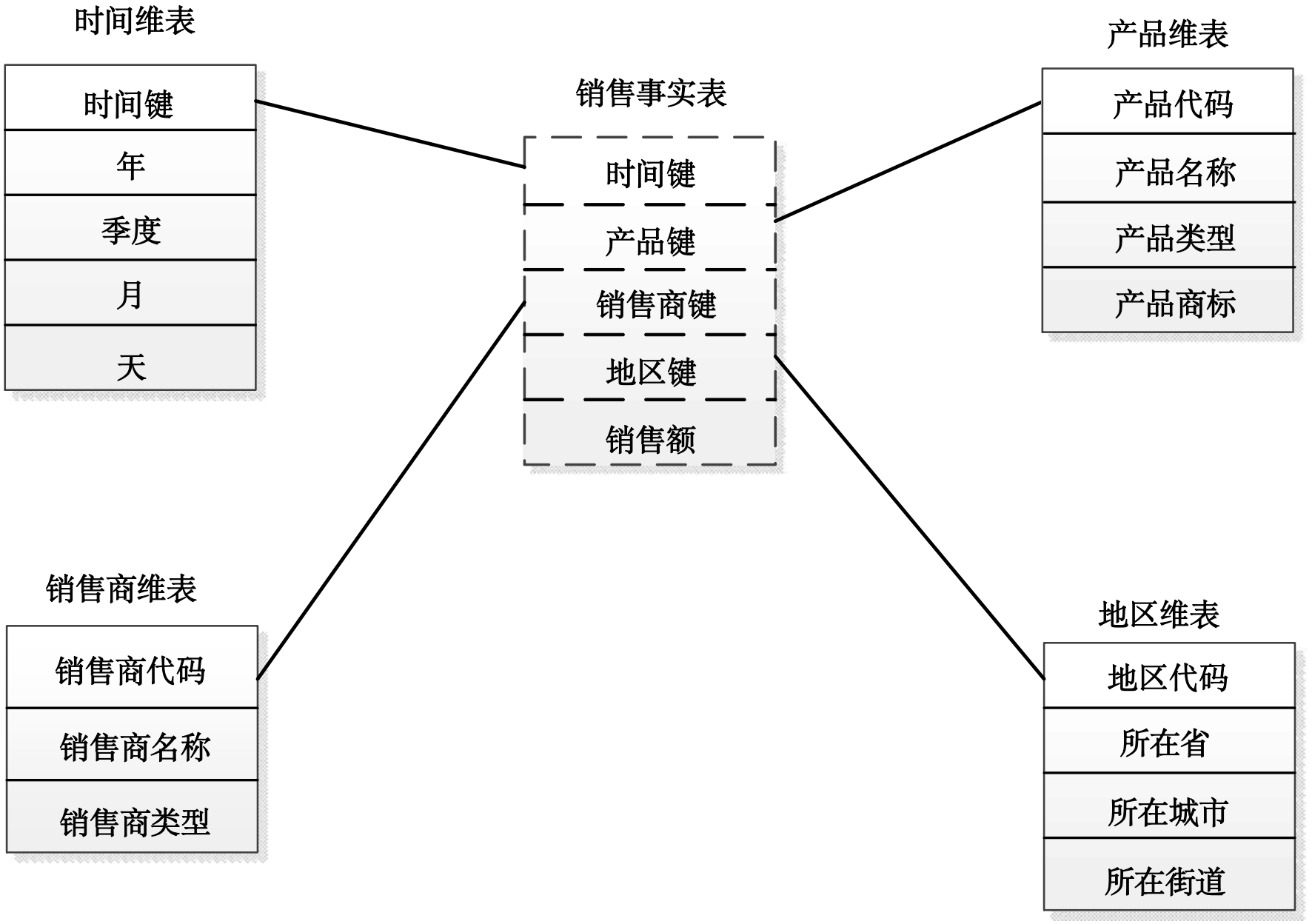
3-D方体

4-D（*基本*）方体

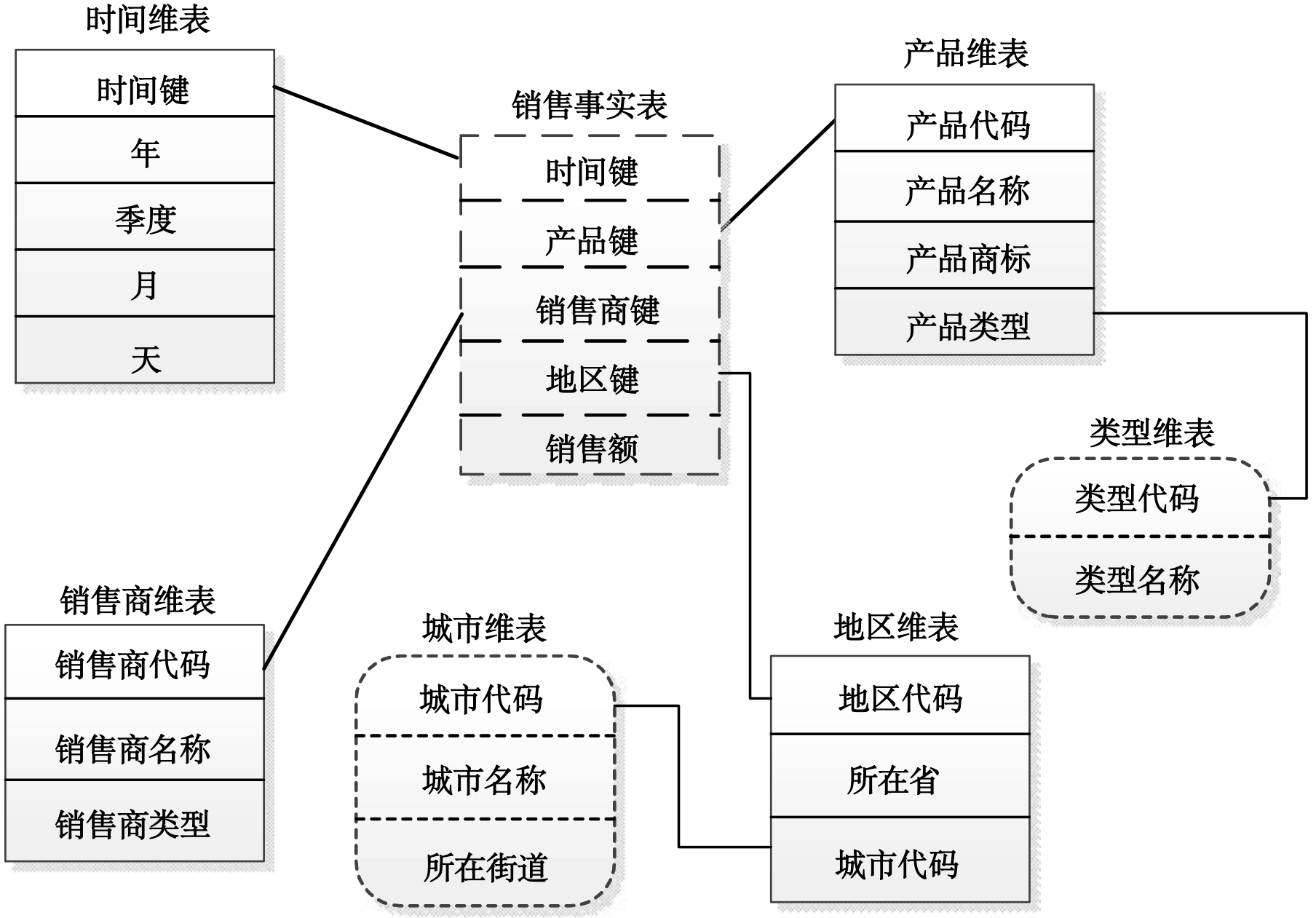
数据仓库的概念性建模

- 数据仓库的建模：维度和度量值
  - 星形模式。事实表在中间，与一组维度表相连
  - 雪花模式。 星型模式的细化，其中一些维度层次被规范化为一组较小的维度表，形成类似于雪花的形状，减少冗余。
  - 事实星座。多个事实表共享维度表，被看作是星星的集合，因此称为星系模式或事实星座

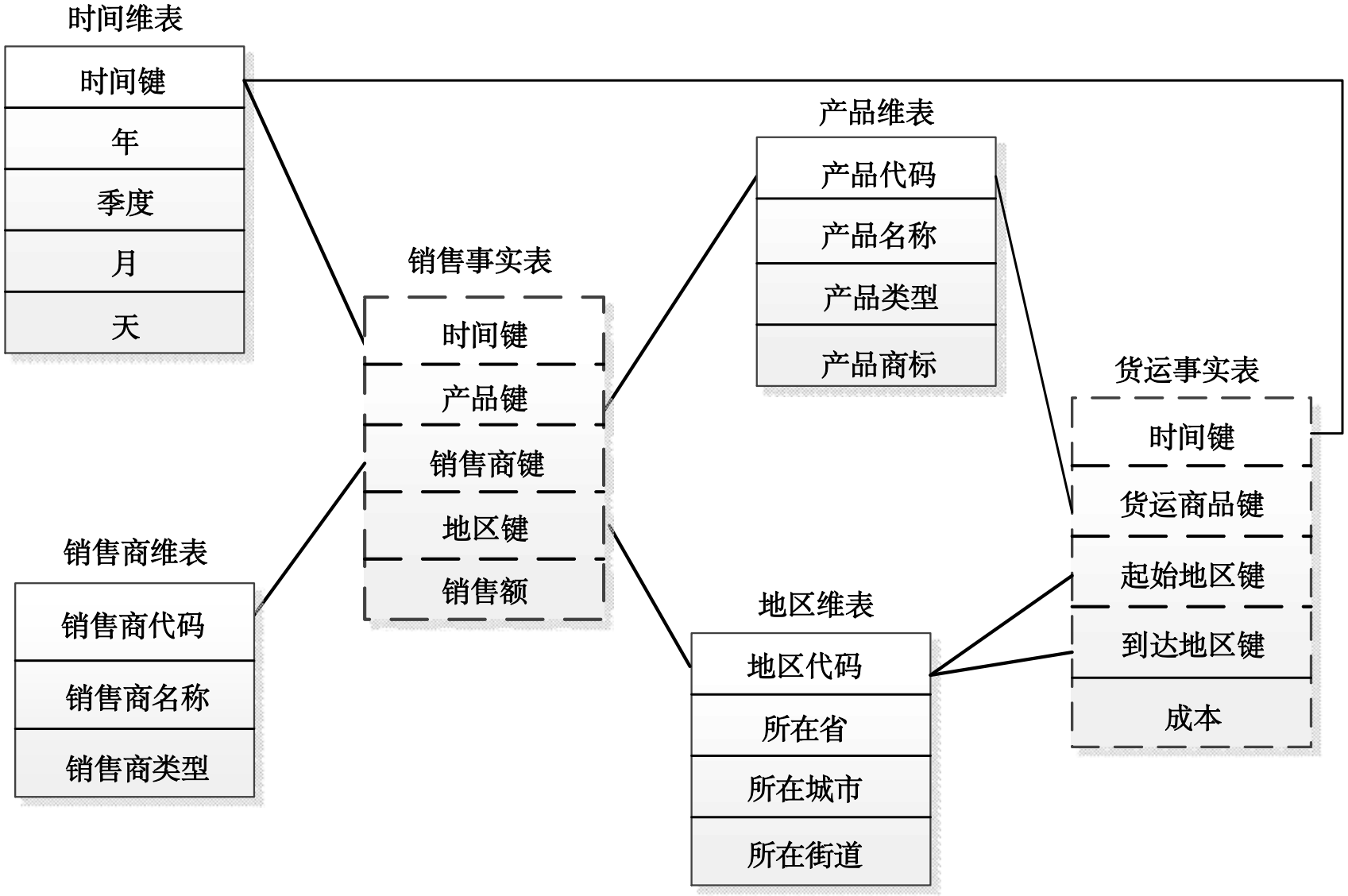
星形模式的例子


时间维表
时间键
年
季度
月
天
销售事实表
时间键
产品键
销售商键
地区键
销售额
产品维表
产品代码
产品名称
产品类型
产品商标
销售商维表
销售商代码
销售商名称
销售商类型
地区维表
地区代码
所在省
所在城市
所在街道

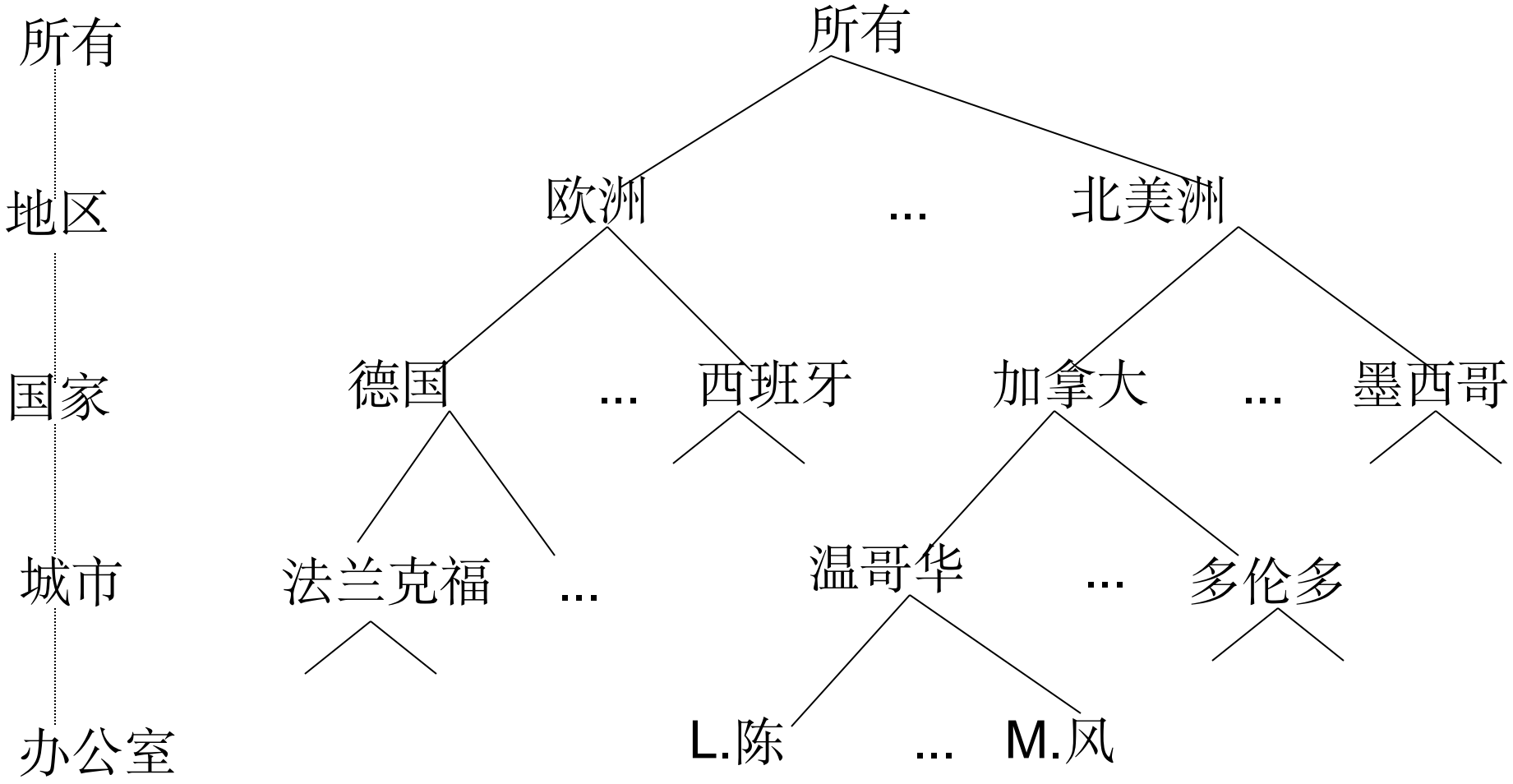
雪花模式的例子


时间维表
时间键
年
季度
月
天
销售事实表
时间键
产品键
销售商键
地区键
销售额
产品维表
产品代码
产品名称
产品商标
产品类型
类型维表
类型代码
类型名称
销售商维表
销售商代码
销售商名称
销售商类型
城市维表
城市代码
城市名称
所在街道
地区维表
地区代码
所在省
城市代码

事实星座的例子


时间维表
时间键
年
季度
月
天
销售事实表
时间键
产品键
销售商键
地区键
销售额
产品维表
产品代码
产品名称
产品类型
产品商标
货运事实表
时间键
货运商品键
起始地区键
到达地区键
成本
销售商维表
销售商代码
销售商名称
销售商类型
地区维表
地区代码
所在省
所在城市
所在街道

一个概念分层: 维Location



三类度量（数值函数）

- 分布式：将数据划分为n个集合，函数在每一部分上的计算得到一个聚集值。如果将函数应用于*n*个聚集值得出的结果与将函数应用于所有数据得出的结果相同，则该函数可以用分布式计算
  - 例如，count(), sum(), min(), max()。
- 代数的：如果它可以由一个具有*M*个参数（其中*M*是一个有界的整数）的代数函数来计算，其中每个参数都可以通过应用一个分布式聚集函数求得
  - avg(x) = sum(x) / count(x)
  - min_N()是一个代数量吗？ standard_deviation()呢？
- 整体性：如果描述它的子聚集所需的存储没有一个常数界
  - 例如，median()、mode()、rank()。

数据仓库和分层结构图

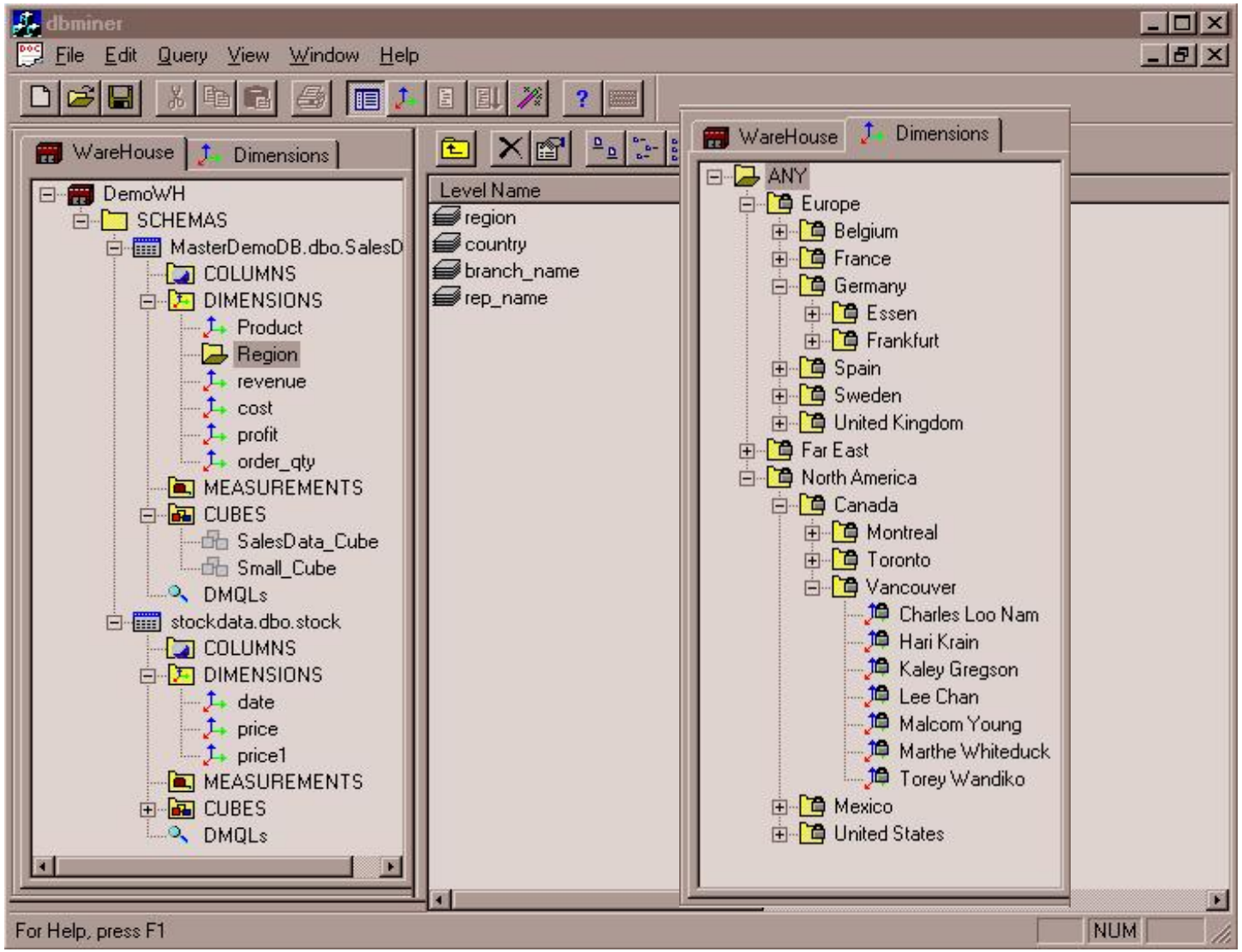
层次结构的规范

- 模式的层次结构

  日<{月<季；周}<年

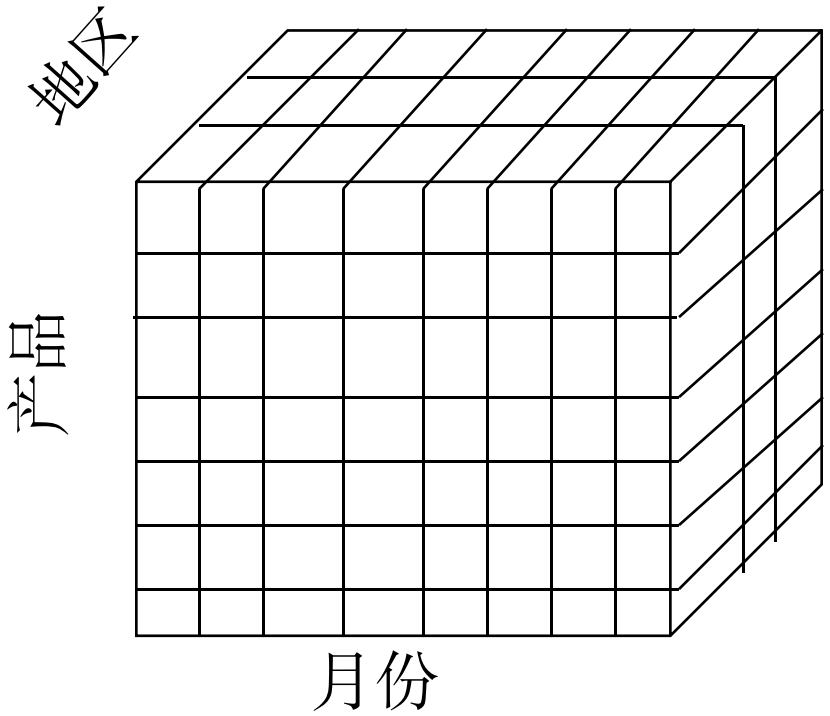
- Set_grouping 层次结构

  {1...10} < 廉价的

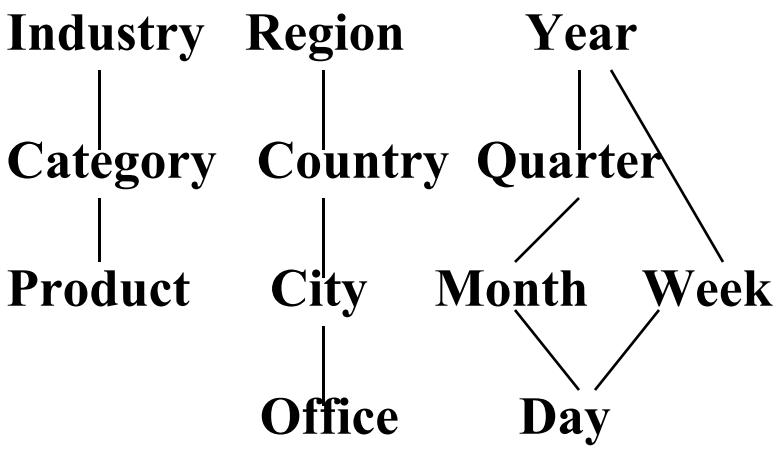


多维数据

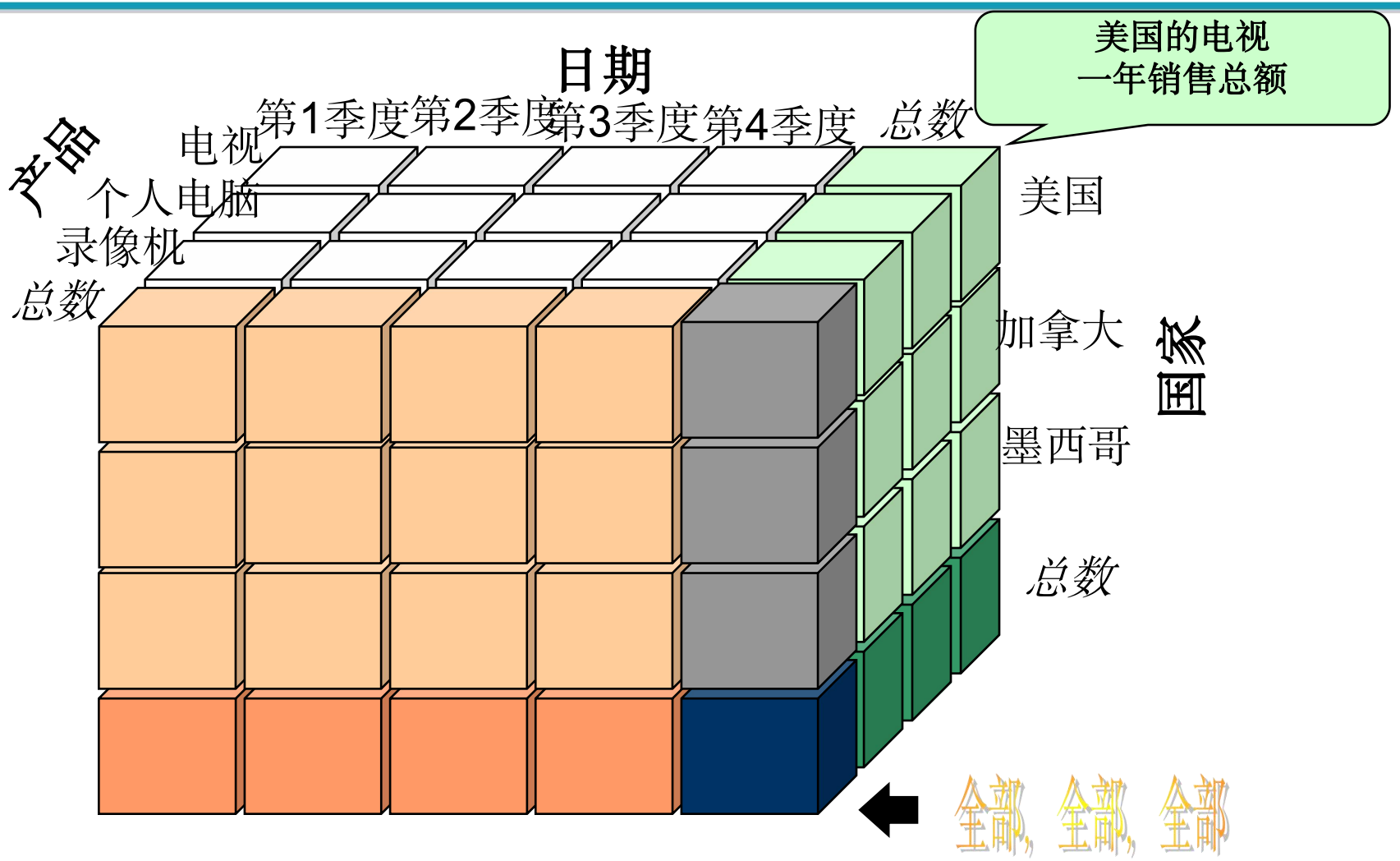
- 销售量与产品、月份和地区的关系



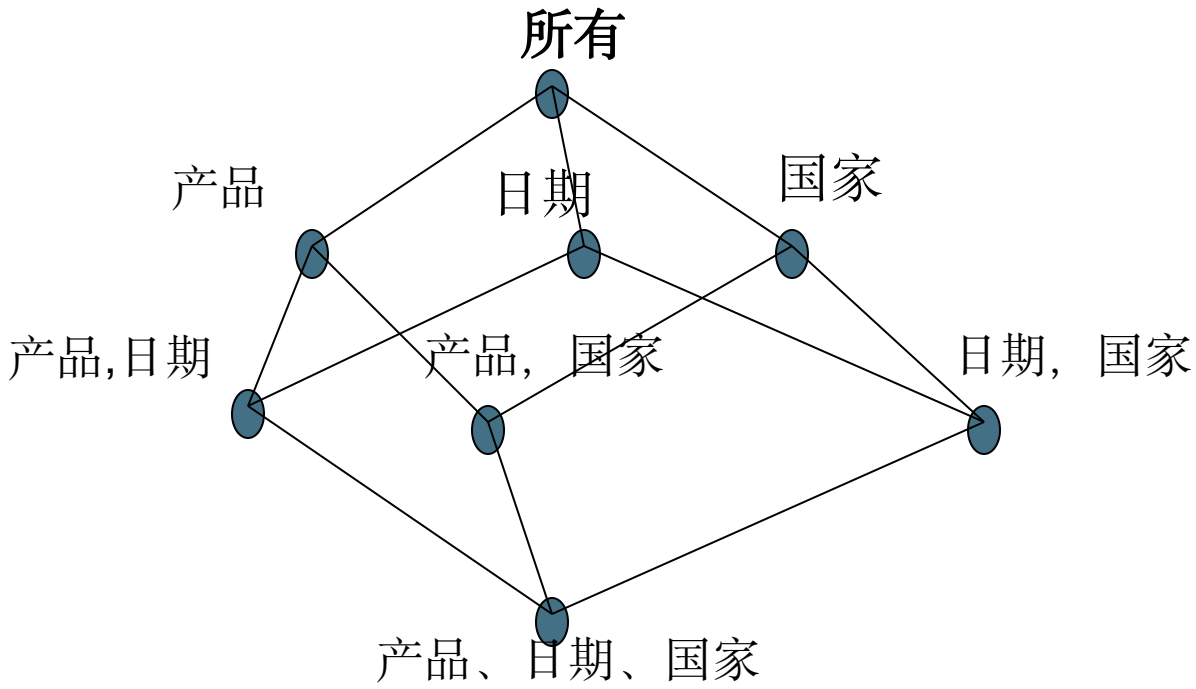
维度：产品、地点、时间
的分层结构



一个数据立方体样本


日期
第1季度
第2季度
第3季度
第4季度
总数
产品
电视
个人电脑
录像机
总数
美国的电视
一年销售总额
美国
加拿大
墨西哥
总数
国家
全部
全部
全部

对应于数据方的方体



0-D（*顶点*）长方体

一维立方体

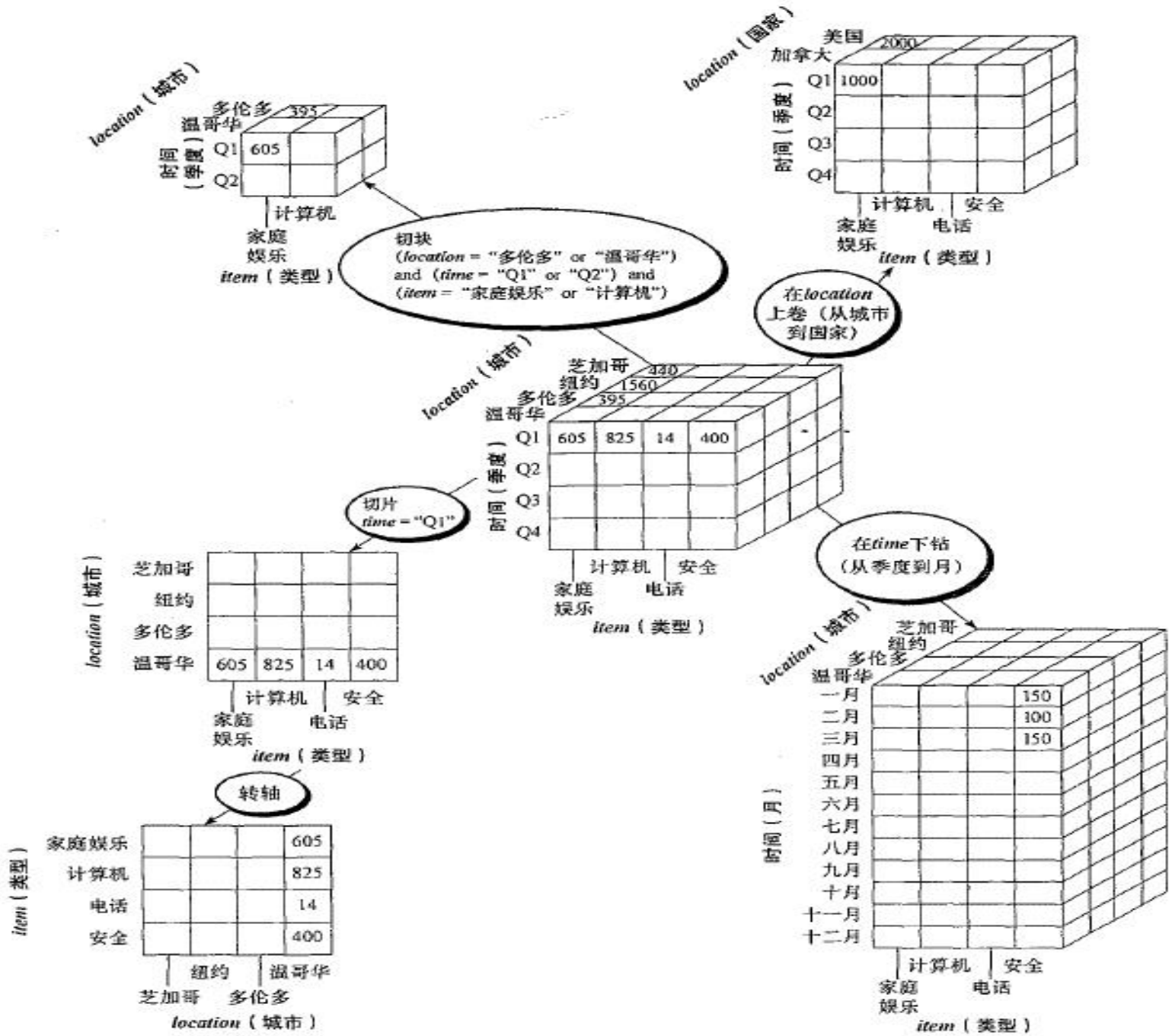
2-D立方体

3-D（*基*）长方体

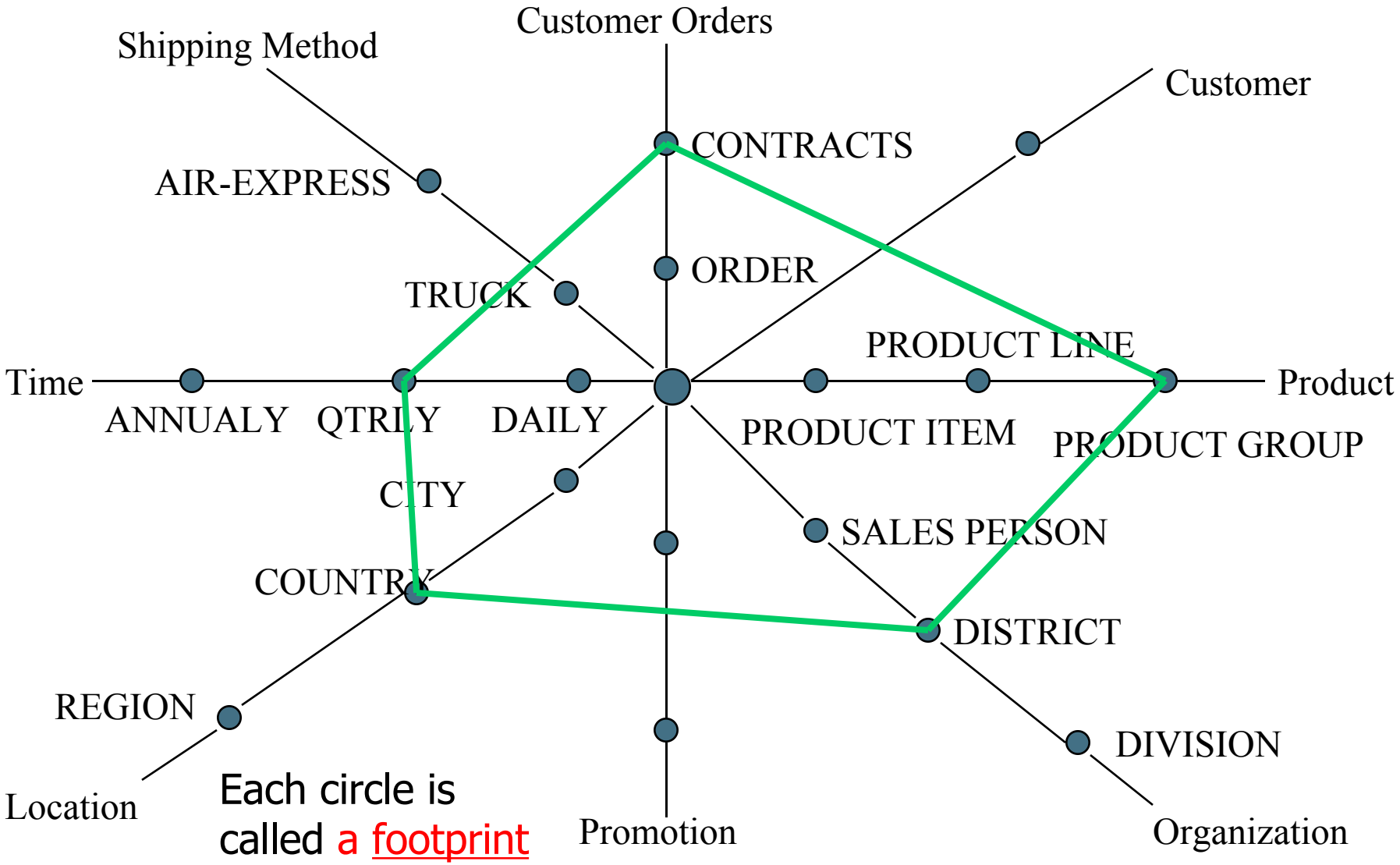
典型的OLAP操作

- 上卷（上钻）：汇总数据
  - *通过概念分层向上攀登或减少维度来实现。*
- 下钻（滚落）：与上卷相反。
  - *从高层次的摘要到低层次的摘要或详细数据，或引入新的维度*
- 切片和切块：*投影和选择*
- 转轴（旋转）。
  - *调整数据方、可视化、3D到一系列的2D平面*
- 其他业务
  - *钻过:跨越式钻探：涉及（跨越）一个以上的事实表*
  - *钻透：通过立方体的底层到其后端关系表（使用SQL）。*

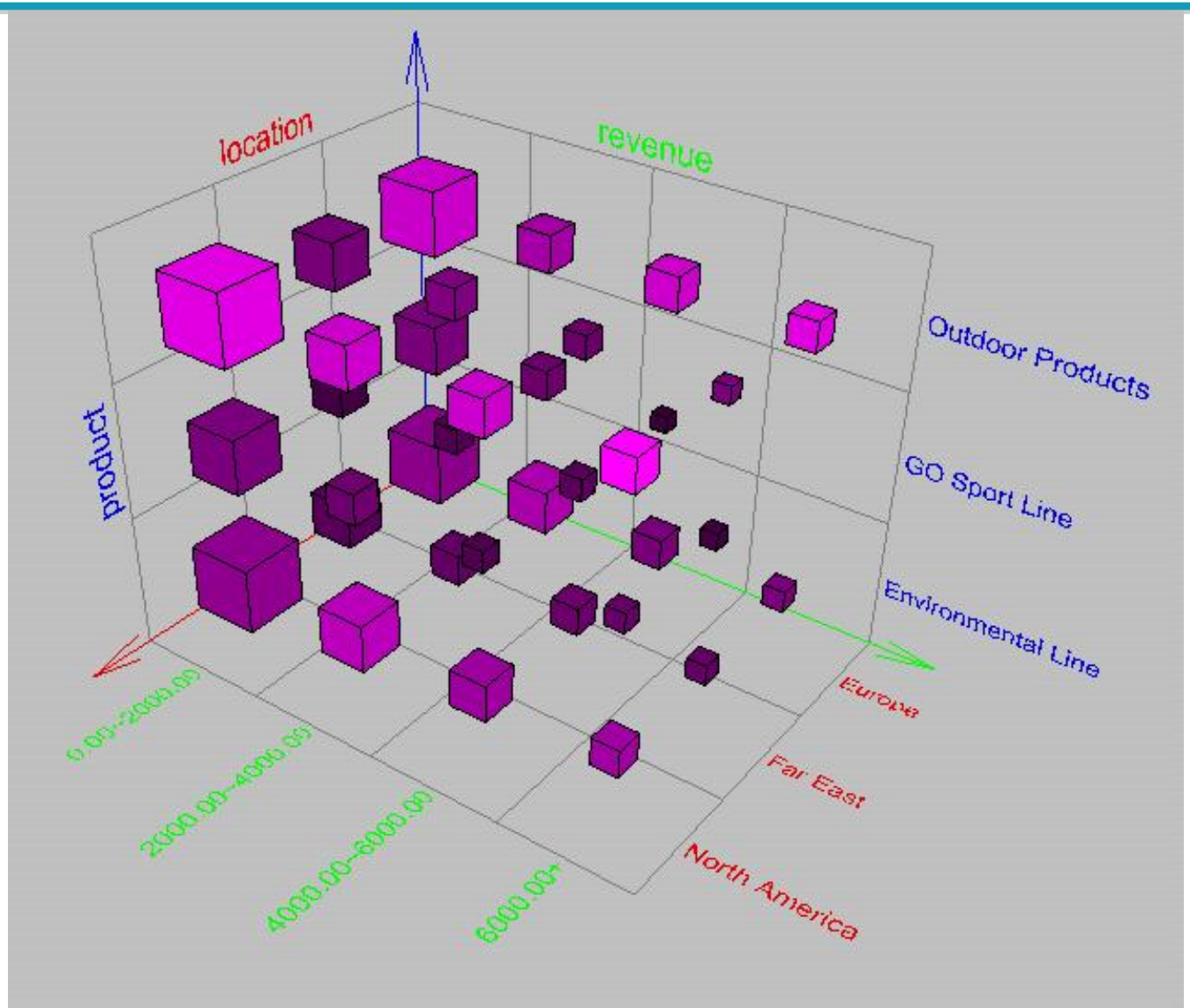
典型的OLAP操作


location（城市）
location（国家）
time（季度）
item（类型）
time（月）
多伦多
温哥华
芝加哥
纽约
美国
加拿大
Q1
Q2
Q3
Q4
605
825
14
400
1000
2000
395
1560
440
计算机
安全
家庭娱乐
电话
切块
(location = "多伦多" or "温哥华")
and (time = "Q1" or "Q2") and
(item = "家庭娱乐" or "计算机")
在location上卷（从城市到国家）
切片
time = "Q1"
在time下钻（从季度到月）
转轴
一月
二月
三月
四月
五月
六月
七月
八月
九月
十月
十一月
十二月
150
100
150

星网查询模型


Customer Orders
Shipping Method
Customer
CONTRACTS
AIR-EXPRESS
ORDER
TRUCK
PRODUCT LINE
Time
Product
ANNUALY
QTRLY
DAILY
PRODUCT ITEM
PRODUCT GROUP
CITY
SALES PERSON
COUNTRY
DISTRICT
REGION
DIVISION
Location
Promotion
Organization
Each circle is
called a footprint

浏览一个数据立方体



- 可视化
- OLAP能力
- 互动式操作

第四章：数据仓库和联机分析处理

- 数据仓库基本概念
- 数据仓库建模-数据立方体和OLAP
- 数据仓库的设计和使用
- 数据仓库实现
- 总结

数据仓库的设计:一个商务分析框架

- 在设计数据仓库的时候需要考虑不同的视图，关于数据仓库设计的四种视图
  - 自顶向下视图
    - 选择数据仓库所需的相关信息
  - 数据源视图
    - 揭示（操作）数据库系统捕获、存储、和管理的信息
  - 数据仓库视图
    - 由事实表和维度表组成
  - 业务查询视图
    - 从最终用户的角度看仓库中的数据。

数据仓库设计过程

- **自顶向下、自底向上的方法或**两者的**结合**
  - **自顶向下** 从整体设计和规划开始（成熟）。
  - **自底向上** 从实验和原型开始（快速）。
- **从软件工程的角度来看**
  - 瀑布式：在进入下一步之前，在每一步都进行结构化和系统化的分析
  - 螺旋式：快速生成功能越来越强的系统，周转时间短，周转快
- **典型的数据仓库设计过程**
  - 选择一个业务流程进行建模，例如，订单、发票等。
  - 选择业务流程的*粒度*（*数据的原子级别*）。例如，单个事务、一天的快照等
  - 选择将适用于每个事实表记录的维度，如,时间、商品、顾客、供应商、仓库、事务类型和状态
  - 选取将安放在事实表中的度量. 典型的度量是可加的数值量, 如dollars_sold和units_sold

数据仓库的使用

- 三种类型的数据仓库应用
  - 信息处理
    - 支持查询、基本统计分析，以及使用交叉表、表格、图表和图形进行报告
  - 分析处理
    - 数据仓库数据的多维分析
    - 支持基本的OLAP操作，切片-切块，钻取，透视
  - 数据挖掘
    - 从隐藏的模式中发现知识
    - 支持关联，构建分析模型，进行分类和预测，并使用可视化工具展示挖掘结果

从联机分析处理(OLAP) 到联机分析挖掘(OLAM)

- 为什么要进行**联机分析挖掘**?
  - 数据仓库中的数据的高质量
    - 数据仓库包含集成的、一致的、经过清理的数据
  - 围绕数据仓库的有价值的信息处理基础设施
    - ODBC、OLEDB、Web访问、服务机制、报告和OLAP工具
  - 基于OLAP的探索性数据分析
    - 用上下钻、切片、转轴等方式进行挖掘。
  - 数据挖掘功能的联机选择
    - 多个数据挖掘功能、算法和任务的整合和互换

第四章：数据仓库和联机分析处理

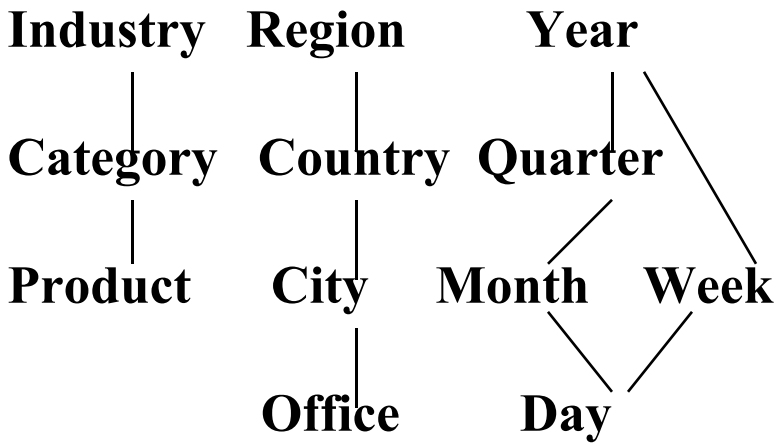
- 数据仓库基本概念
- 数据仓库建模-数据立方体和OLAP
- 数据仓库的设计和使用
- 数据仓库实现
- 总结

高效的数据立方体计算：概述

- 数据立方体可以被看作是立方体的格
  - 最底层的方体是基本方体
  - 最上面的立方体（顶点）只包含一个单元
  - 一个有L层的n维立方体中有多少个立方体？
  - 其中Li是与维i相关联的层数
- 数据立方体的物化
  - **全物化**。物化每一个方体
  - **不物化**。不物化任何方体
  - **局部物化**。将某些立方体物化
    - 哪些立方体要实体化？
      - 根据大小、共享、访问频率等进行选择。

$$T = \prod_{i=1}^{n} (L_i + 1)$$

为什么是这个公式？



数据立方体计算

- DMQL中的立方体定义和计算

  define cube sales [item, city, year]: sum (sales_in_dollars)

  compute cube sales

- 将其转化为类似SQL的语言（有一个新的运算符cube by，由Gray等人在96年提出）。

  SELECT item, city, year, SUM (amount)

  FROM SALES

  CUBE BY item, city, year

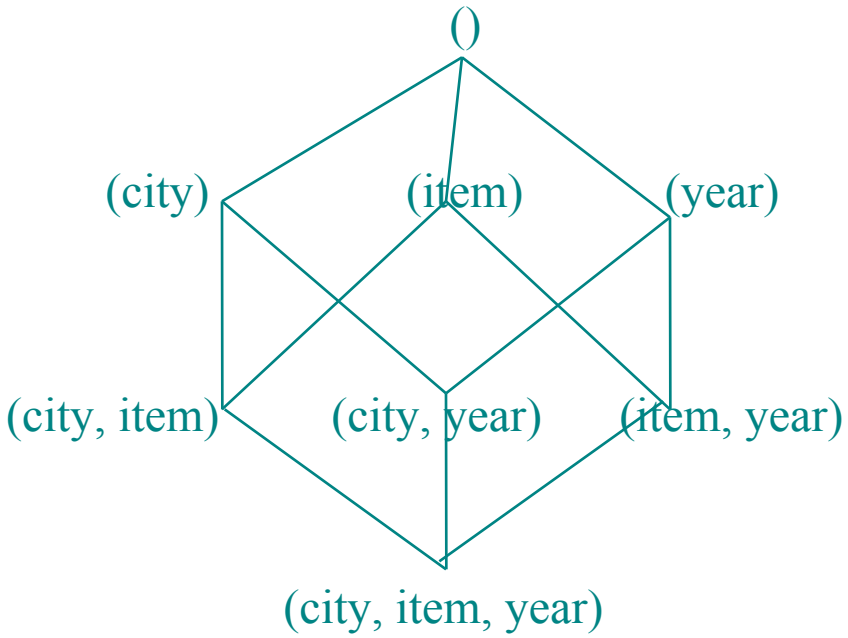
- 需要计算以下的Group-Bys

  (date, product, customer),

  (date, product),(date, customer), (product, customer),

  (date), (product), (customer)

  ()



索引OLAP数据：位图索引

- 对某一特定列的索引
  - 列中的每个值都有一个位向量：位操作是快速的
  - 位向量的长度。# 基表中的记录数
  - 如果数据表中给定行的属性值为v, 则在位图索引的对应行, 表示该值的位为1, 该行的其它位均为0
  - 不适合势(不同值个数)很高的域

基础表

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

地区索引

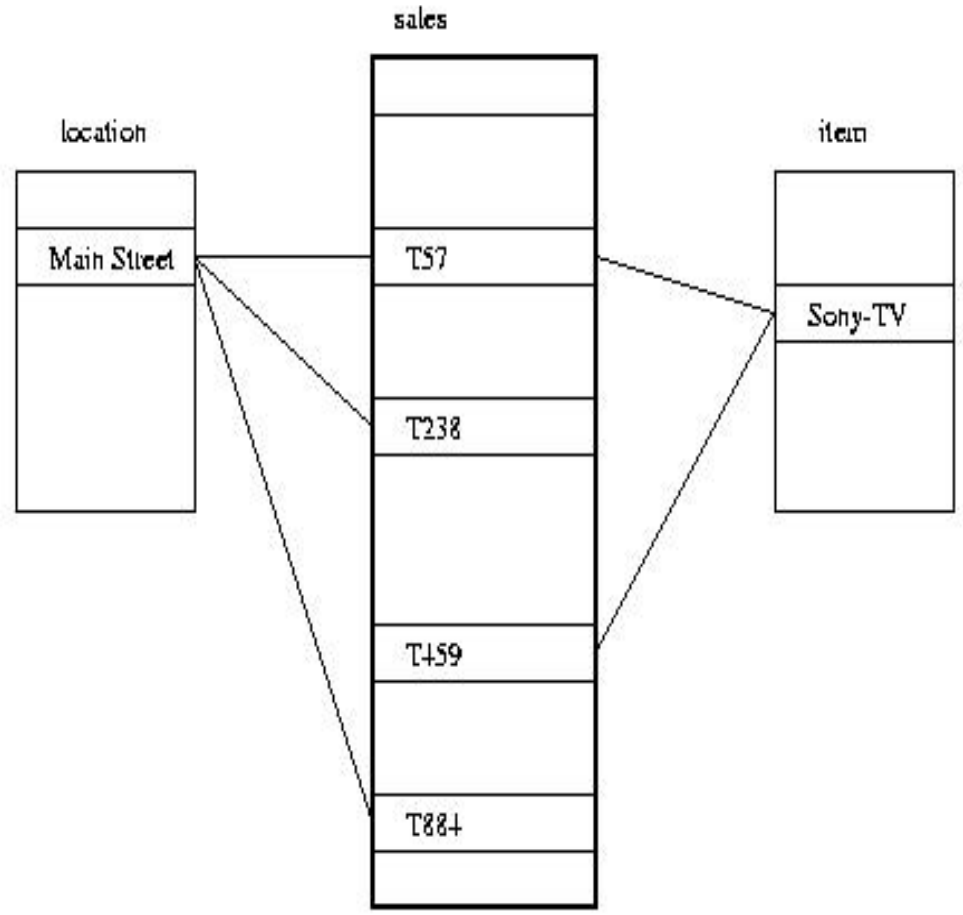
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

类型索引

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

索引|OLAP数据： 连接索引

- 连接索引。JI(R-id, S-id) 其中R (R-id, ...) ⊳⊲ S (S-id, ...)
- 传统的索引将数值映射到一个记录ID的列表中
  - 它在JI文件中实现了关系连接，加快了关系连接的速度。
- 在数据仓库中，连接索引将起始模式的维度值与事实表中的行联系起来。
  - 例如，事实表Sales和两个维度*city*和*product*
    - City上的连接索引为每个不同的城市保留了一个记录该城市销售的图元的R-ID列表。
  - 连接索引可以跨越多个维度



高效处理OLAP查询

- 确定哪些操作可以在可用的方体上进行
  - 将钻孔、滚动等转化为相应的SQL和/或OLAP操作，例如，dice = selection + projection
- **确定应当使用哪些物化立方体**。
  - 让要处理的查询在{brand, province_or_state} 上，条件是 “*year = 2004* ，并且有4个物化立方体可用。

    1) {year, item_name, city}

    2) {year, brand, country}

    3) {year, brand, province_or_state}

    4) {item_name, province_or_state} where year = 2004

    应该选择哪个来处理查询？
- 探索MOLAP中的索引结构和压缩数组与密集数组的关系

OLAP服务器架构

- **关系型OLAP (ROLAP)**
  - 使用关系型或扩展关系型DBMS来存储和管理仓库数据和OLAP中间件
  - 包括优化DBMS后端，实施聚合导航逻辑，以及额外的工具和服务
  - 更大的可扩展性
- **多维OLAP (MOLAP)**
  - 基于稀疏阵列的多维存储引擎
  - 对预先计算好的汇总数据进行快速索引
- **混合OLAP（HOLAP）**（例如，微软SQLServer）。
  - 灵活性，例如，低层次：关系型，高层次：阵列型
- 专门的SQL服务器（如Redbricks）。
  - 对星形/雪花模式的SQL查询的专门支持

第四章：数据仓库和在线分析处理

- 数据仓库。基本概念
- 数据仓库建模。数据方块和OLAP
- 数据仓库的设计和使用
- 数据仓库的实施
- 摘要

摘要

- 数据仓库。数据仓库的多维模型
  - 一个数据立方体由*维度表*和*事实表*组成
  - 星星模式、雪花模式、事实星座
  - OLAP操作：钻孔、滚动、切片、切块和透视
- 数据仓库架构、设计和使用
  - 多层次的架构
  - 业务分析设计框架
  - 信息处理、分析处理、数据挖掘、OLAM
- 实施。高效地计算数据方块
  - 局部与全部与无物化
  - 对OALP数据进行索引。位图索引和连接索引
  - OLAP查询处理
  - OLAP服务器。ROLLAP, MOLAP, HOLAP

参考文献(一)

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD'97
- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- **S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997**
- J. Gray, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
- A. Gupta and I. S. Mumick. Materialized Views: Techniques, Implementations, and Applications. MIT Press, 1999
- J. Han. Towards on-line analytical mining in large databases. *SIGMOD Record*, 1998
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD'96

参考文献(二)

- C. Imhoff, N. Galemmo, and J. G. Geiger. Mastering Data Warehouse Design: Relational and Dimensional Techniques. John Wiley, 2003
- W. H. Inmon. Building the Data Warehouse. John Wiley, 1996
- R. Kimball and M. Ross. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2ed. John Wiley, 2002
- P. O'Neil and D. Quass. Improved query performance with variant indexes. SIGMOD'97
- S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. ICDE'94
- P. Valduriez. Join indices. ACM Trans. Database Systems, 12:218-246, 1987.
- J. Widom. Research problems in data warehousing. CIKM'95.
- K. Wu, E. Otoo, and A. Shoshani, Optimal Bitmap Indices with Efficient Compression, ACM Trans. on Database Systems (TODS), 31(1), 2006, pp.