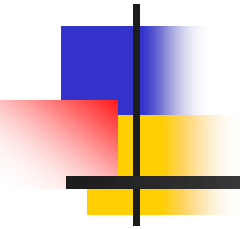
第3章
数据挖掘的数据仓库与OLAP技术



第3章: 数据挖掘的数据仓库与OLAP技术

- 什么是数据仓库?
- 多维数据模型
- 数据仓库结构
- 数据仓库实现
- 数据立方体的进一步发展
- 从数据仓库到数据挖掘

什么是数据仓库?

- 有不同的方法定义, 但不是严格的.
  - 是一个决策支持数据库, 它与组织机构的操作数据库分别维护
  - 数据仓库系统允许将各种应用系统集成在一起, 为统一的历史数据分析提供坚实的平台, 支持信息处理.
- **W. H. Inmon的定义:** 数据仓库是 面向主题的**(subject-oriented),** 集成的**(integrated),** 时变的**(time-variant),** 和非易失的**(nonvolatile)** 数据集合, 支持管理决策过程
- 建立数据仓库**(Data warehousing):**
  - 构造和使用数据仓库的过程

数据仓库—面向主题的

- 围绕重要的主题(如顾客、产品、销售等) 组织.
- 关注决策制定者的数据建模与分析, 而不是日常的操作和事务处理.
- 数据仓库排除对于决策过程无用的数据, 提供特定主题的简明视图.

数据仓库— 集成的

- 通过将多个异种的数据源集成在一起, 而构造
  - 比如，关系数据库, 一般文件, 联机事务记录

- 使用数据清理和数据集成技术.
  - 确保命名约定, 编码结构, 属性度量等的一致性
    - 例如, 饭店价格: 货币种类, 税, 是否含早餐, 等.
  - 当数据装入数据仓库时, 数据将被转换.

数据仓库— 时变的

- 数据仓库的时间跨度显著地比操作数据库长.
  - 操作数据库数据: 当前值数据.
  - 数据仓库数据: 从历史的角度提供数据 (例如, 过去 5-10 年)
- 数据仓库中的每个键结构
  - 显式或隐式地包含时间元素,
  - 但是, 操作数据的键可能包含, 也可能不包含“时间元素”.

数据仓库— 非易失的

- 从操作环境转换过来的数据物理地分离存放.
- 数据的更新不在数据仓库环境中出现.
  - 不需要事务处理, 恢复, 和并发控制机制
  - 只需要两种数据存取操作:
    - *数据的初始化装入* 和 *数据访问*.

数据仓库和异种DBMS

- 传统的异种数据库集成:
  - 在异种数据库上建立一个包装程序(wrappers)或中介程序(/mediators)
  - 查询驱动的方法
    - 当查询提交给一个站点时, 使用元数据词典将查询转换成所涉及的异构站点上的相应查询, 查询的结果被集成为一个全局回答的集合
    - 需要：复杂的信息过滤, 对资源的竞争
- 数据仓库: 更新驱动的, 高性能
  - 来自异种信息源的数据被预先集成并存储在数据仓库中, 直接用于查询和分析

数据仓库VS.操作数据库

- **OLTP (on-line transaction processing, 联机事务处理)**
  - 传统关系 DBMS的主要任务
  - 涵盖日常操作: 购买, 库存, 银行, 制造, 工资单, 注册, 记帐, 等.
- **OLAP (on-line analytical processing, 联机分析处理)**
  - 数据仓库系统的主要任务
  - 数据分析和决策制定上提供服务
- 不同的特点 (OLTP vs. OLAP):
  - 用户和系统的面向性: 顾客 vs. 市场
  - 数据内容: 当前的, 细节的 vs. 历史的, 合并的
  - 数据库设计: ER + 应用 vs. 星型 + 主题
  - 视图: 当前的, 局部的 vs. 进化的, 集成的
  - 访问模式: 更新 vs. 只读的, 但是复杂的查询

OLTP vs. OLAP

	OLTP	OLAP
用户	办事员, IT 从业人员	知识工人
功能	日常操作	决策支持
DB 设计	面向应用	面向主题
数据	当前的, 最新的, 细节的, 展平的关系的, 孤立的	历史的, 汇总的, 多维的, 集成的, 加固的
用法	重复	特殊的
访问	读/写 在主键上索引/散列	大量扫描
工作单位	短的, 简单的事务	复杂的查询
访问的记录量	数以十计	数百万
用户数	数千	数百
数据库大小	100MB-GB	100GB-TB
度量	事务吞吐量	查询吞吐量, 响应时间

为什么建立分离的数据仓库?

- 为了两个系统的高性能
  - **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, 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, 通常操作数据库并不维护这些数据
  - 数据统一: 决策支持需要将来自异种数据源的数据统一 (聚集, 汇总)
  - 数据质量: 不同的数据源通常使用不同的数据表示, 编码, 和应当遵循的格式

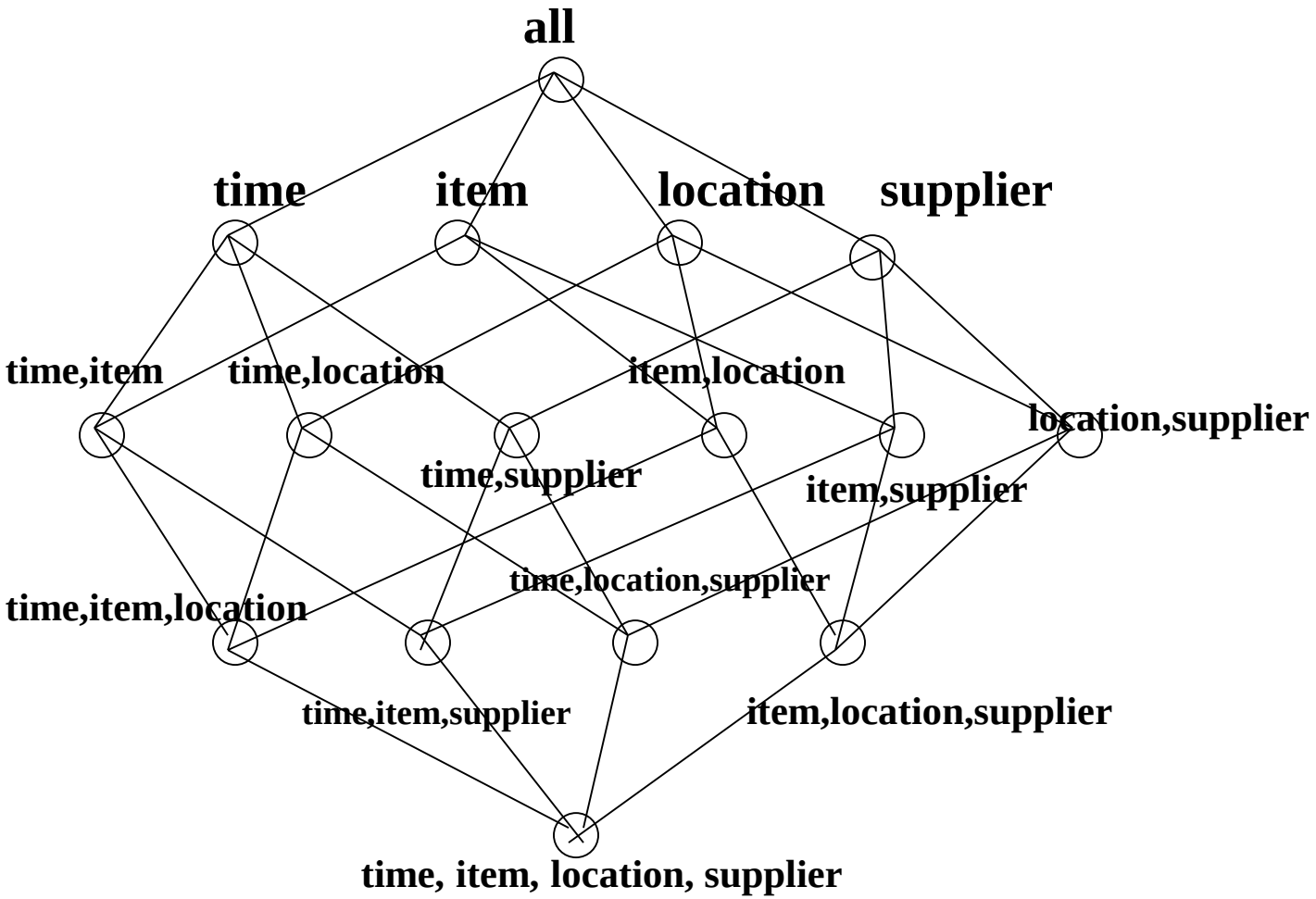
第2章: 数据挖掘的数据仓库与OLAP技术

- 什么是数据仓库?
- 多维数据模型
- 数据仓库结构
- 数据仓库实现
- 从数据仓库到数据挖掘
- 数据立方体的进一步发展

由表和电子数据表到数据方

- 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sales) 可以将数据建模, 并允许由多个维进行观察
  - 维表, 如 item (item_name, brand, type), 或 time(day, week, month, quarter, year)
  - 事实表包含度量 (如 dollars_sold) 和每个相关维表的键
- 在数据仓库的文献中, 一个 n-D 基本立方体 称作基本方体 (base cuboid). 最顶部的 0-D方体存放最高层的汇总，称作顶点方体(apex cuboid). 方体的格形成数据方.

立方体: 方体的格



0-D(顶点) 方体

1-D 方体

2-D方体

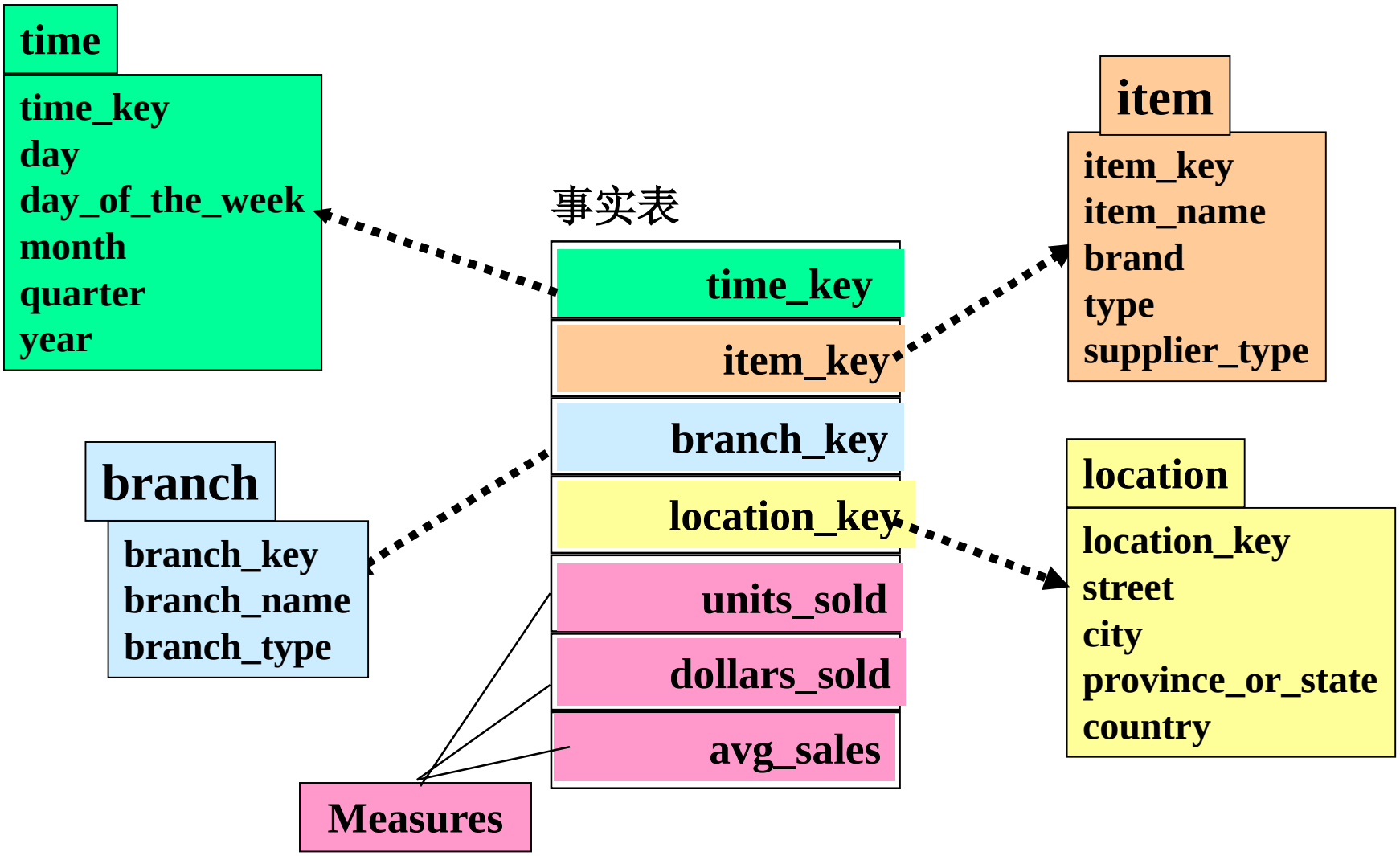
3-D方体

4-D(基本)方体

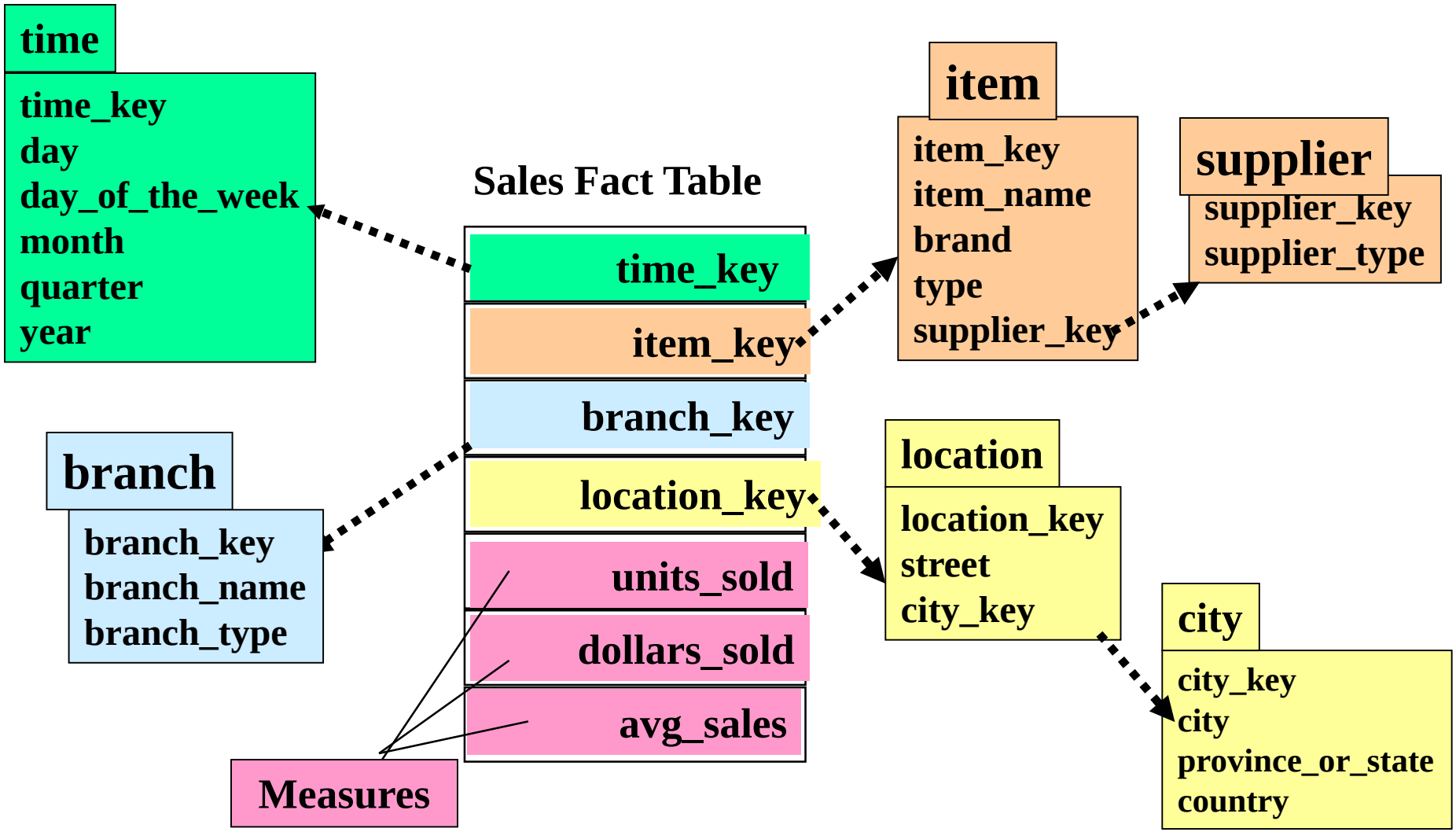
数据仓库的概念建模

- 数据仓库建模: 多维模型，涉及维和度量
  - 星型模式: 事实表在中央, 连接一组维表
  - 雪花模式：星型模式的精炼, 其中一些维分层结构被规范化成一组较小的维表, 形成类似于雪花的形状，减少冗余
  - 事实星座：多个事实表共享维表, 可以看作星星的集合, 因此称作星系模式, 或事实星座

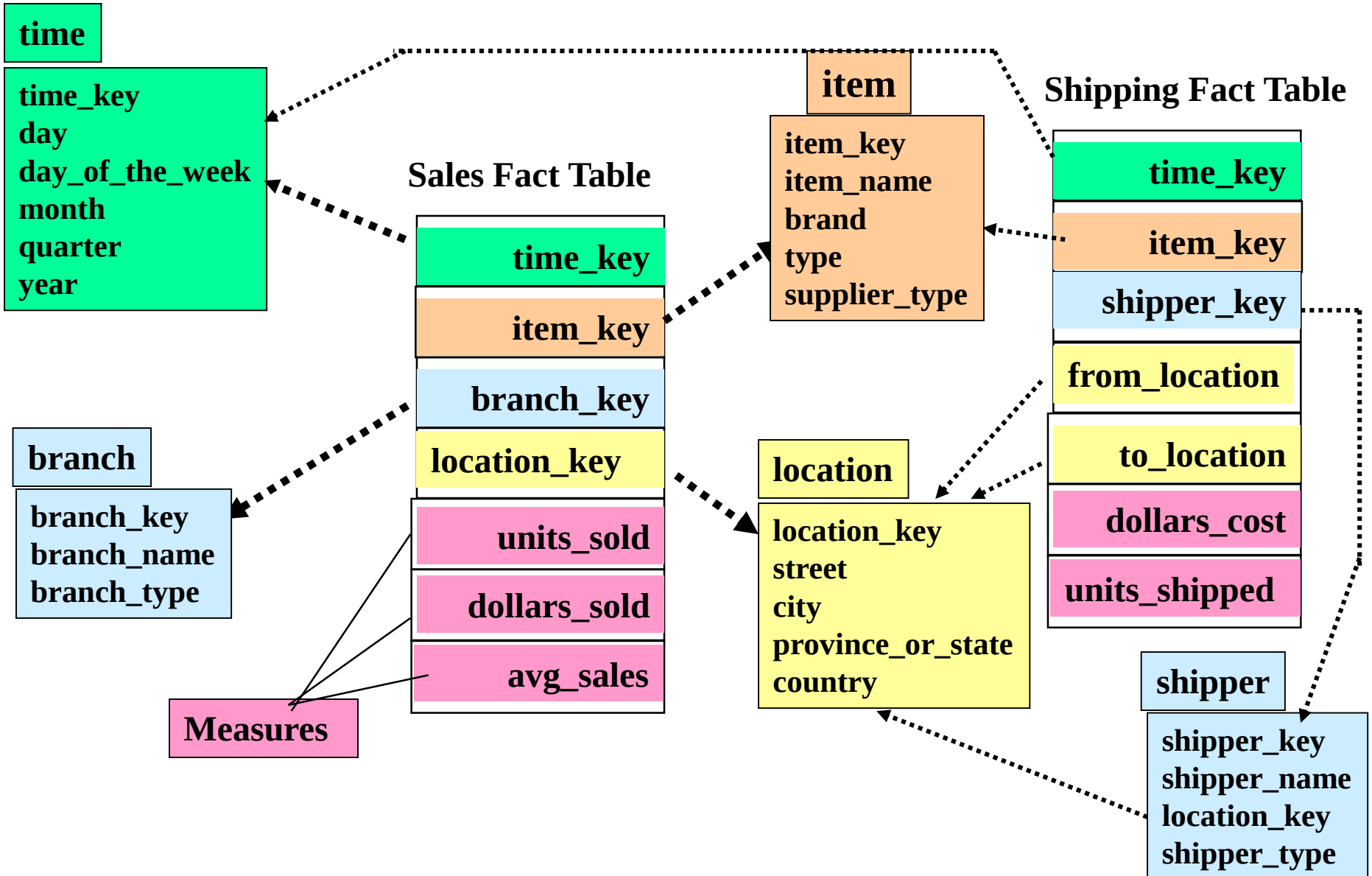
星型模式的例子


time
time_key
day
day_of_the_week
month
quarter
year
事实表
time_key
item_key
branch_key
location_key
units_sold
dollars_sold
avg_sales
item
item_key
item_name
brand
type
supplier_type
branch
branch_key
branch_name
branch_type
location
location_key
street
city
province_or_state
country
Measures

雪花模式的例子


time
time_key
day
day_of_the_week
month
quarter
year
Sales Fact Table
time_key
item_key
branch_key
location_key
units_sold
dollars_sold
avg_sales
item
item_key
item_name
brand
type
supplier_key
supplier
supplier_key
supplier_type
branch
branch_key
branch_name
branch_type
location
location_key
street
city_key
city
city_key
city
province_or_state
country
Measures

事实星座的例子


time
time_key
day
day_of_the_week
month
quarter
year
item
item_key
item_name
brand
type
supplier_type
Shipping Fact Table
time_key
item_key
shipper_key
from_location
to_location
dollars_cost
units_shipped
Sales Fact Table
time_key
item_key
branch_key
location_key
units_sold
dollars_sold
avg_sales
branch
branch_key
branch_name
branch_type
location
location_key
street
city
province_or_state
country
shipper
shipper_key
shipper_name
location_key
shipper_type
Measures

数据挖掘查询语言 DMQL: 语言原语

- 立方体定义 (事实表)

  define cube <cube_name> [<dimension_list>]: <measure_list>

- 维定义 (维表)

  define dimension <dimension_name> **as**
  (<attribute_or_subdimension_list>)

- 特殊情况 (共享维表)
  - 第一次, 如 “cube definition”
  - **define dimension** <dimension_name> **as**
    <dimension_name_first_time> **in cube** <cube_name_first_time>

用DMQL定义星型模式

define cube sales_star [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier_type)

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city, province_or_state, country)

用DMQL定义雪花模式

define cube sales_snowflake [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales =
avg(sales_in_dollars), units_sold = count(*)

define dimension time **as** (time_key, day, day_of_week, month, quarter, year)

define dimension item **as** (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))

define dimension branch **as** (branch_key, branch_name, branch_type)

define dimension location **as** (location_key, street, city(city_key, province_or_state, country))

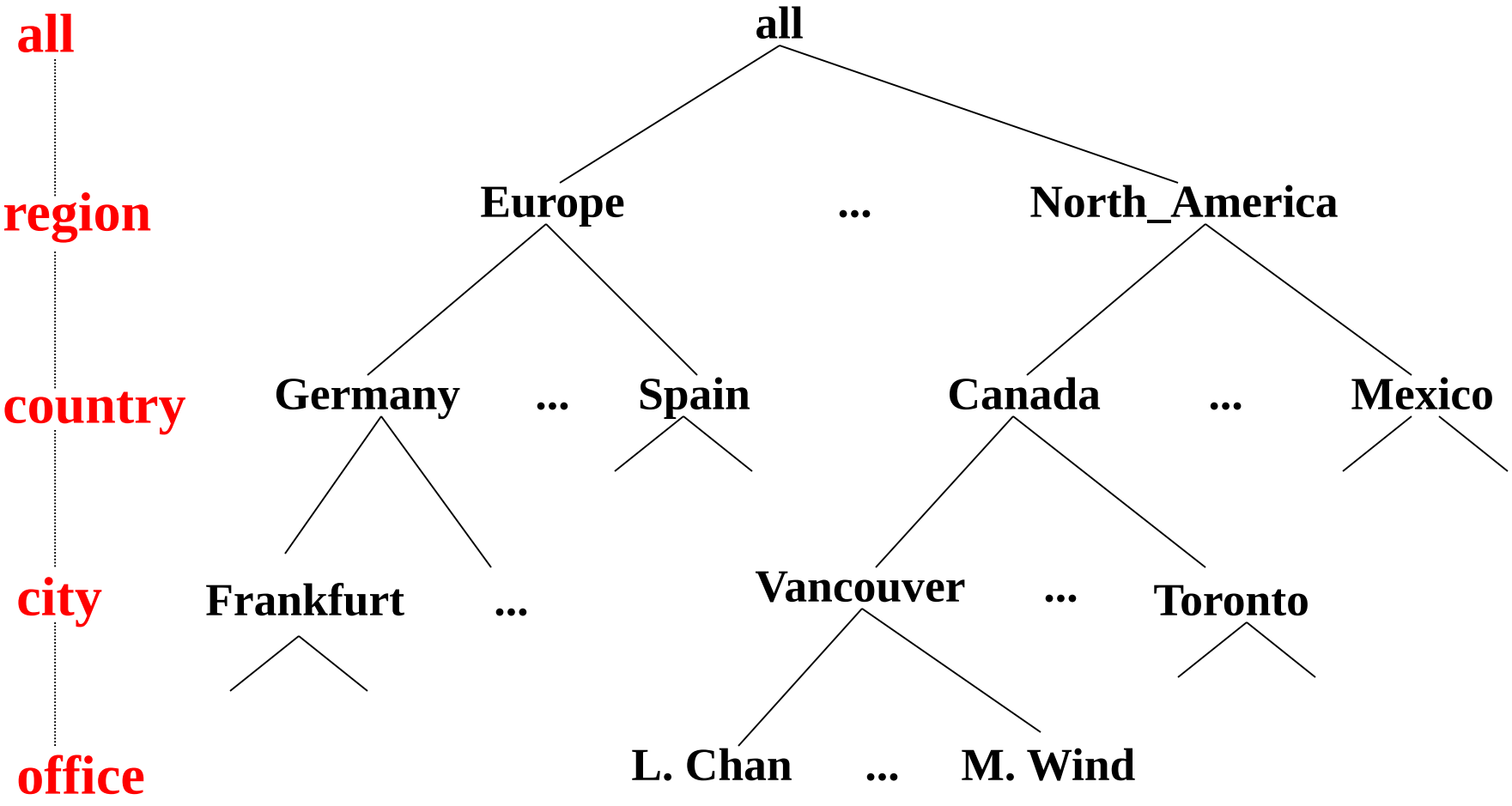
用DMQL定义事实星座

```
define cube sales [time, item, branch, location]:
        dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars),
          units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, province_or_state, country)
define cube shipping [time, item, shipper, from_location, to_location]:
        dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper_key, shipper_name, location as location in
    cube sales, shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales
```

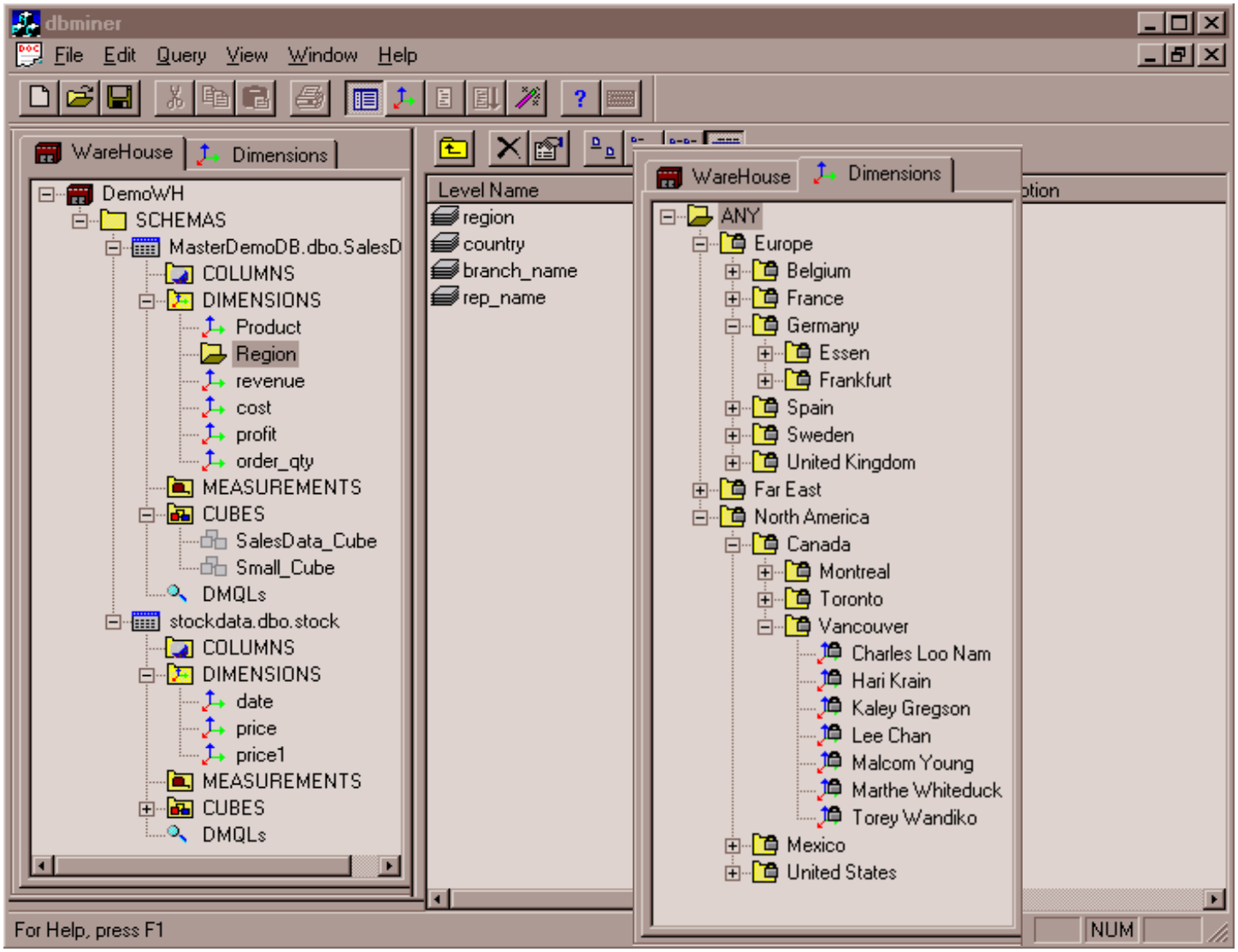
三类度量（数值函数）

- 分布的(distributive)：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. 如果将函数用于*n*个聚集值得到的结果, 与将函数用于所有数据得到的结果一样, 则该函数可以用分布方式计算.
  - 例, count(), sum(), min(), max().
- 代数的(algebraic):如果它能够由一个具有*M*(其中, *M*是一个整数界)个参数的代数函数计算, 而每个参数都可以用一个分布聚集函数求得 .
  - 例， avg(), min_N(), standard_deviation().
- 整体的(holistic):如果描述它的子聚集所需的存储没有一个常数界 .
  - 例, median(), mode(), rank().

一个概念分层: 维Location

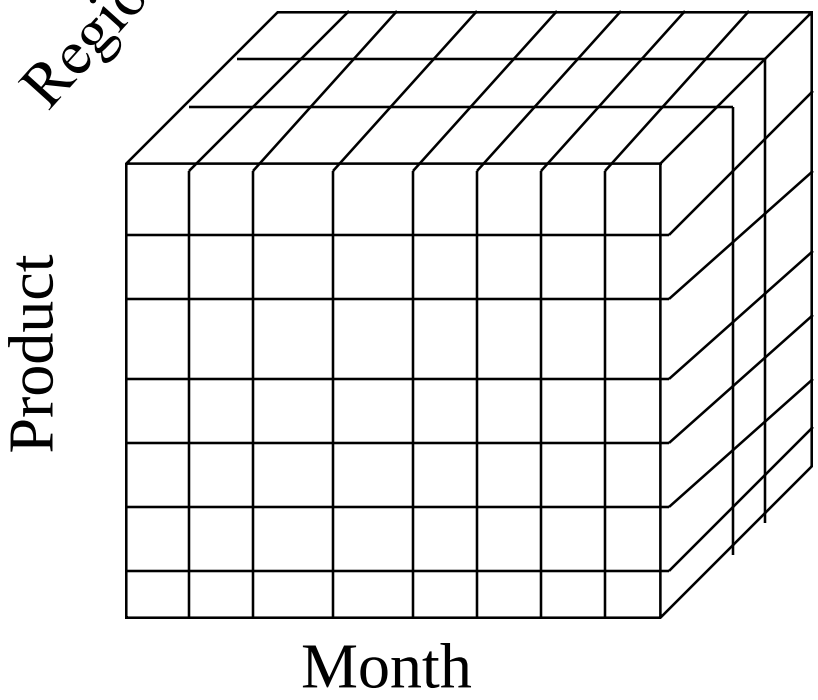


数据仓库和分层结构视图

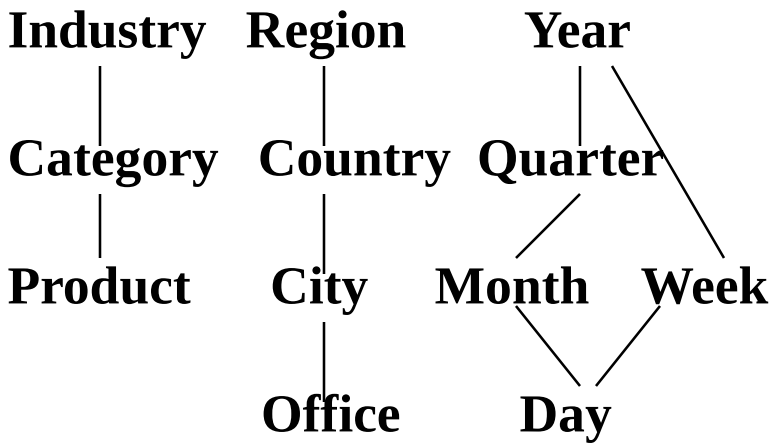


多维数据

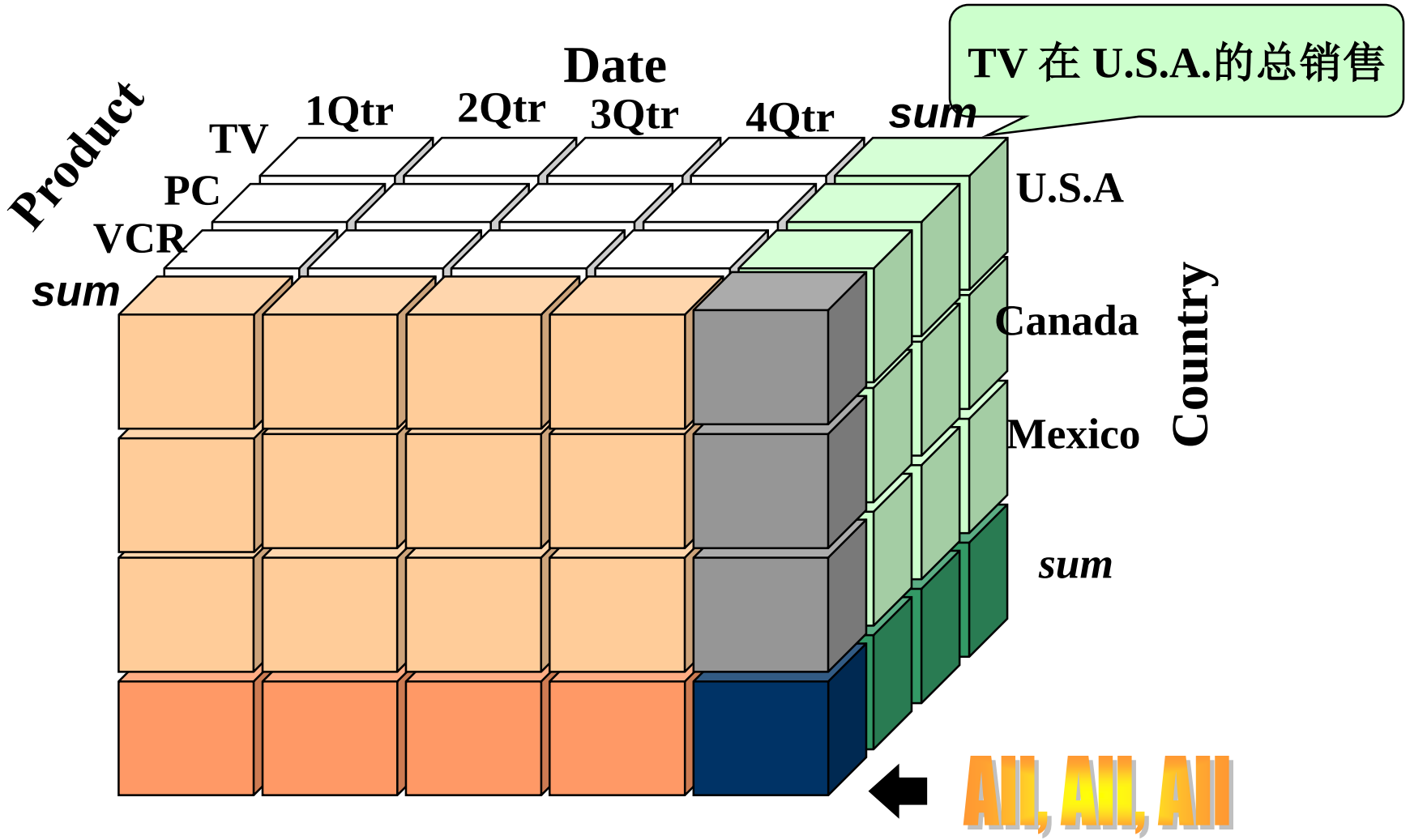
- 多维模型中，数据组织成多维，每维包含由概念分层定义的多个抽象层
- 销售量作为 **product, month, 和 region**的函数



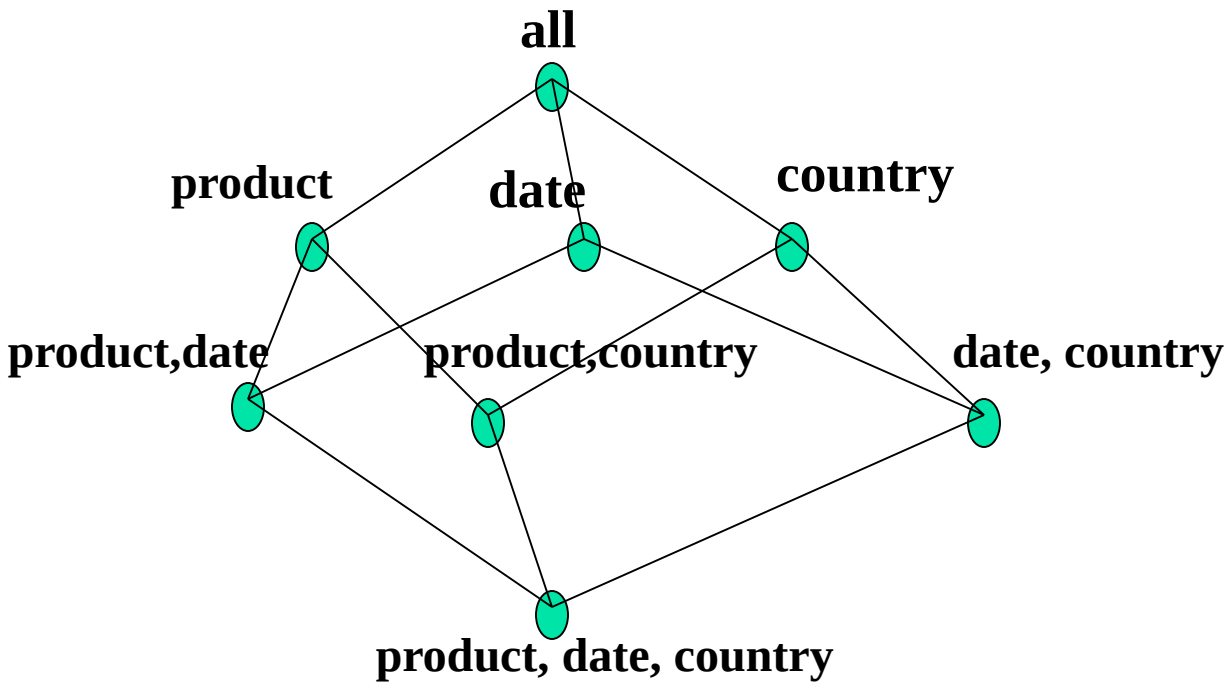
维: Product, Location, Time 的分层结构



一个数据方的样本


TV 在 U.S.A.的总销售
Date
1Qtr
2Qtr
3Qtr
4Qtr
sum
Product
TV
PC
VCR
sum
U.S.A
Canada
Mexico
sum
Country
All, All, All

对应于数据方的方体



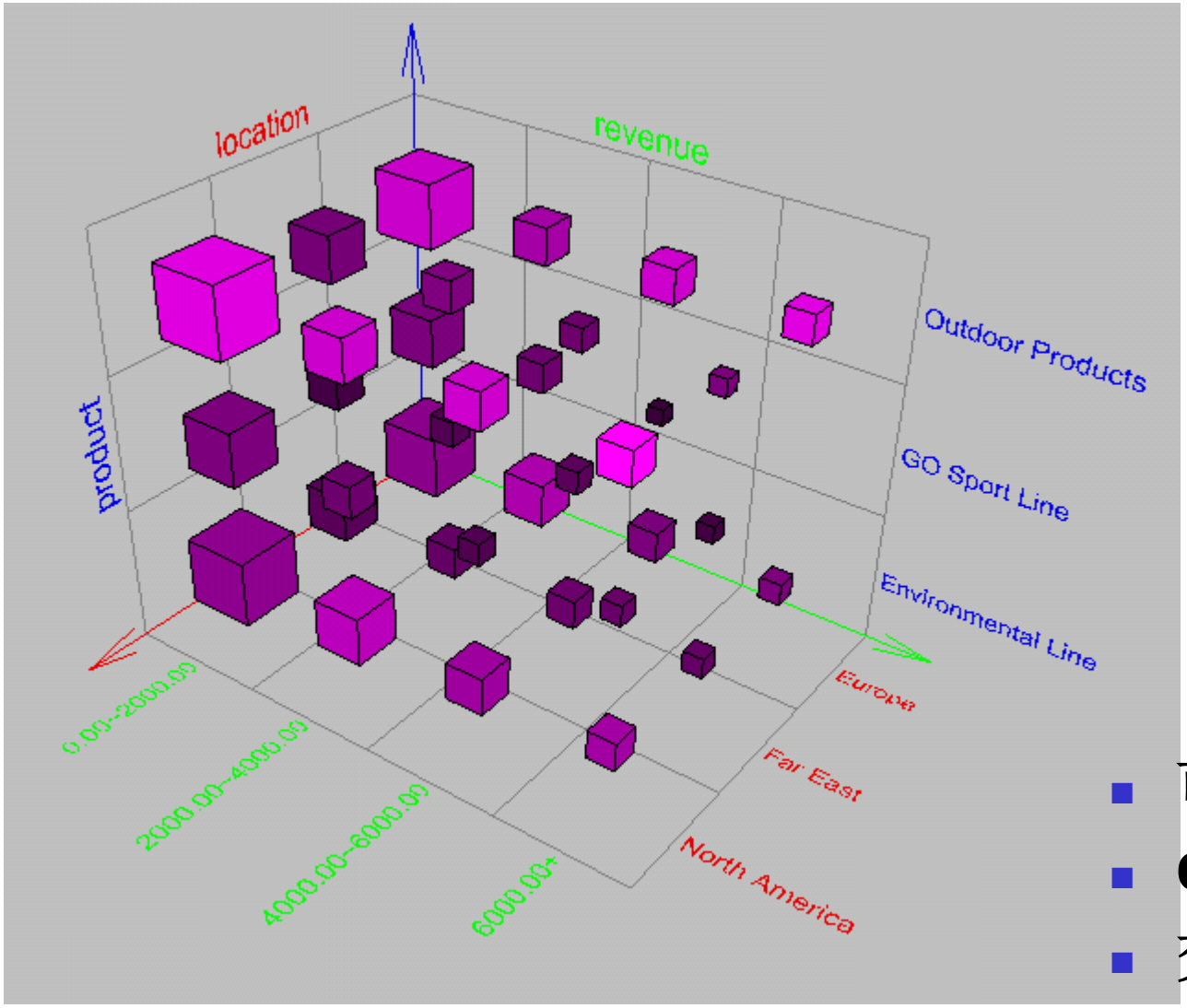
0-D(顶点) 方体

1-D方体

2-D方体

3-D(基本)方体

浏览数据方



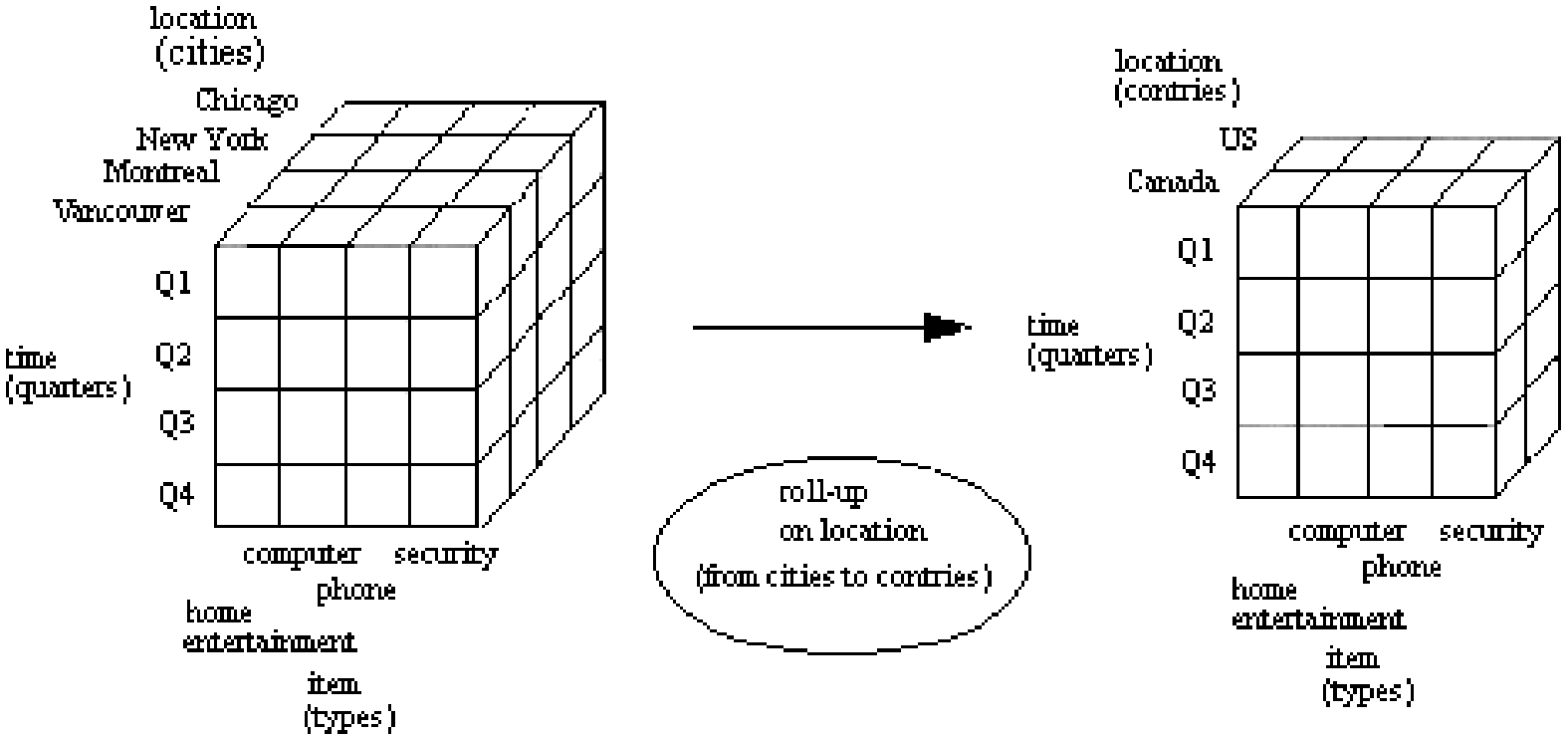
- 可视化
- **OLAP** 的能力
- 交互式操作

典型的OLAP操作

- 上卷(Roll up)/上钻 (drill-up): 汇总数据
- 下钻(Drill down)/下卷 (roll down): 上卷的逆操作
- 切片(Slice)和切块 :
  - 投影和选择
- 转轴(Pivot)/旋转 (rotate):
  - 调整数据方, 目视操作, 3D 到 2D 平面.
- 其它操作
  - 钻过(drill across): 涉及多个事实表
  - 钻透(drill through): 通过数据方的最底层, 到它背后的关系表 (使用 SQL)

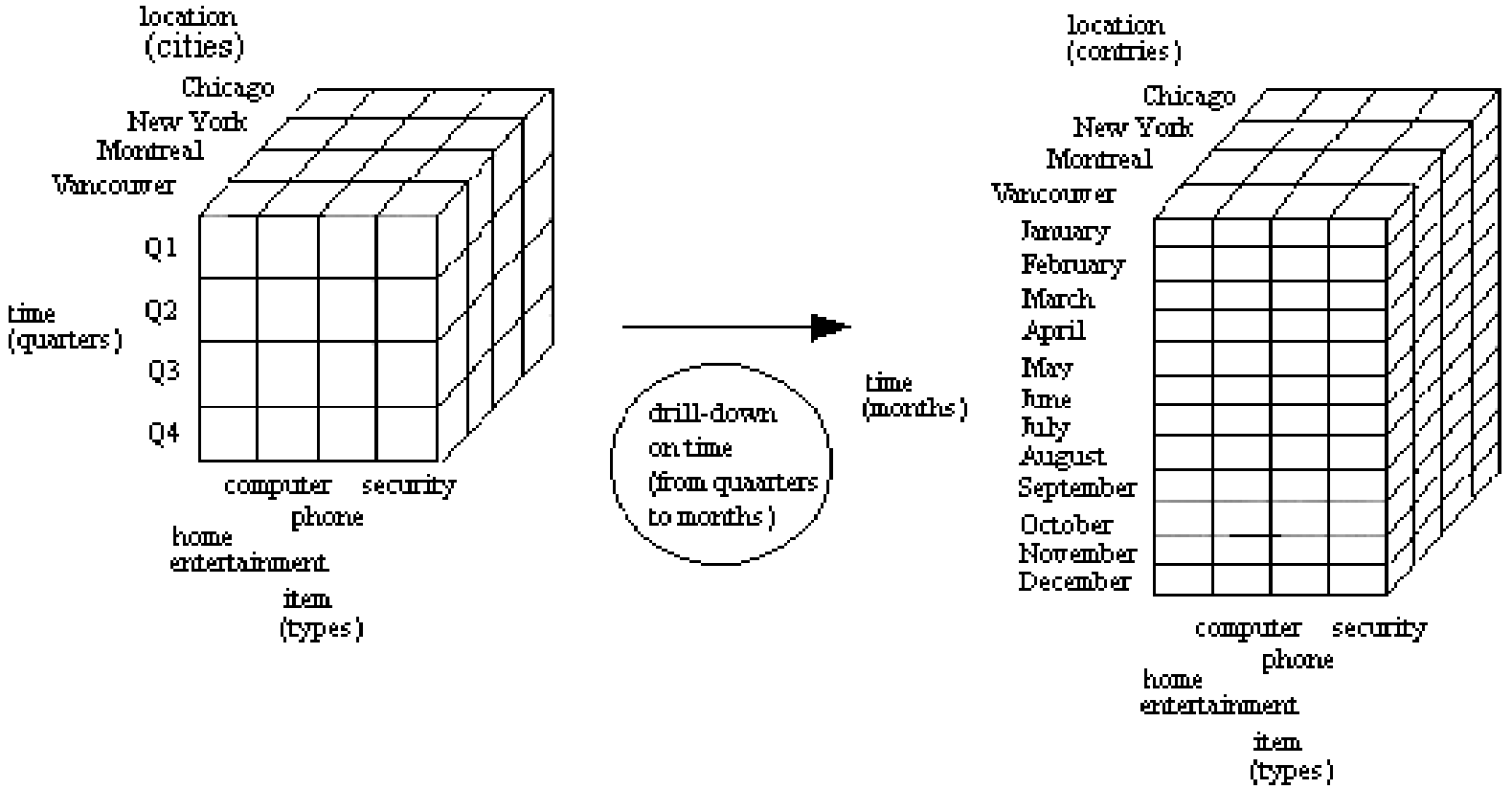
OLAP 操作: 上卷

- 上卷(Roll up)/上钻 (drill-up): 汇总数据
  - 通过沿概念分层攀升或通过维归约
- 在 location上卷(由 cities 到 countries)



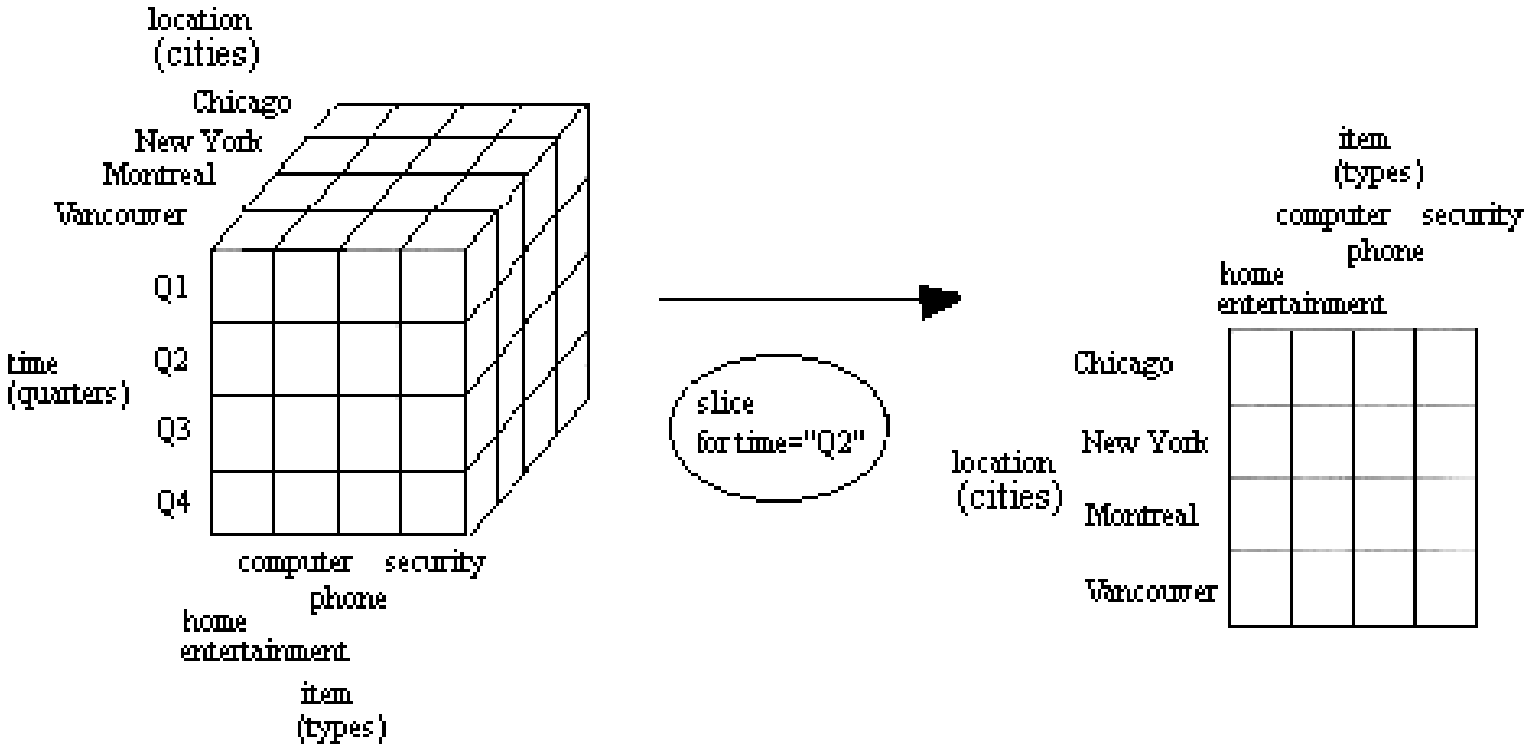
OLAP 操作: 下钻

- 下钻(Drill down)/下卷 (roll down): 上卷的逆操作
  - 由较高层的汇总到较低层的汇总或详细数据, 或者引进新的维
- 在 time下钻 (由 quarters 到 months)



OLAP 操作:切片

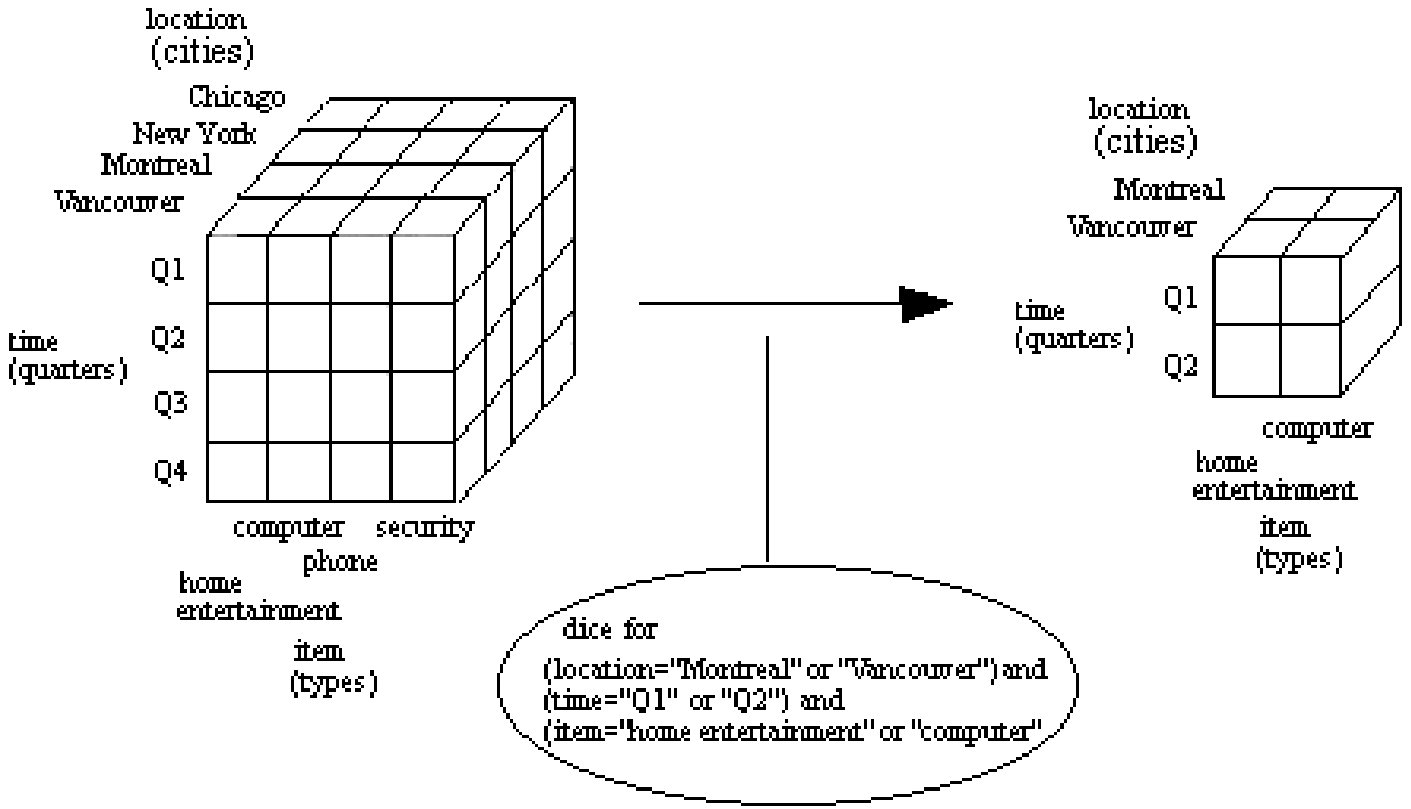
- 切片(Slice) :
  - 投影和选择，对一个维进行选择，导致子方体
- 切片条件: time="Q2"



OLAP 操作: 切块

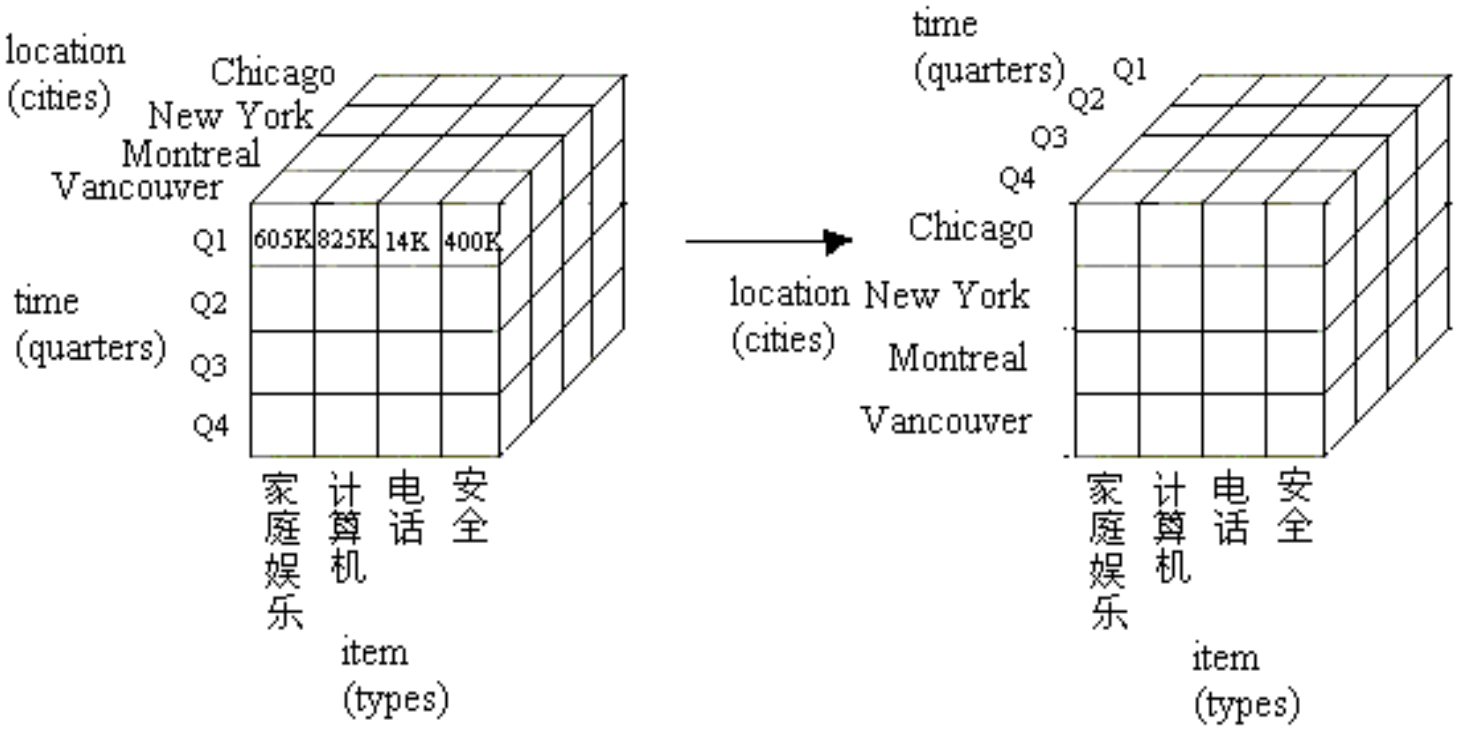
- 切块 : 对两个或多个维执行选择，导致子方体

- 切块条件: **(*location*=“*Montreal*” or “*Vancouver*”) and (*time*=“*Q1*” *or* “*Q2*”) and (*item*=“*home entertainment*” or “*computer*”)**



OLAP 操作: 转轴

- **转轴(Pivot)/旋转 (rotate):**
  - **调整数据方, 可视化操作，提供数据的替代表示.**



其他操作

- 其它操作
  - 钻过(drill across): 涉及多个事实表
  - 钻透(drill through): 通过数据方的最底层, 到它背后的关系表 (使用 SQL)
  - 统计计算
    - 比率、方差；增长率
  - 分析建模，等

第3章: 数据挖掘的数据仓库与OLAP技术

- 什么是数据仓库?
- 多维数据模型
- 数据仓库结构
- 数据仓库实现
- 从数据仓库到数据挖掘
- 数据立方体的进一步发展

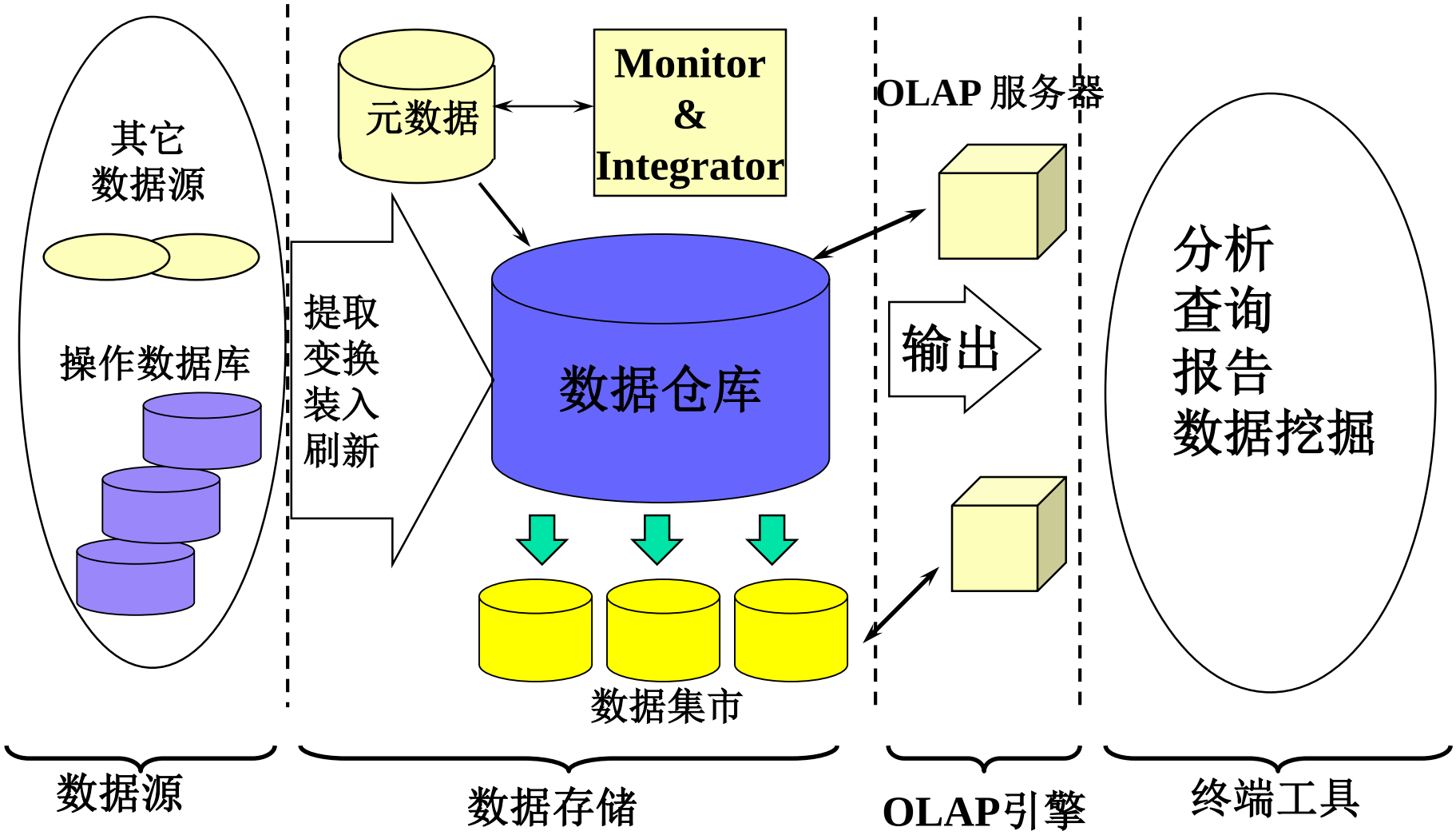
数据仓库设计

- 数据仓库设计中，必须考虑四种视图
  - 自顶向下视图
    - 选择数据仓库所需的有关信息
  - 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
    - 由事实表和维表组成
  - 商务查询视图
    - 从最终用户的角度透视数据仓库中的数据

数据仓库设计过程

- 自顶向下, 自底向上方法或二者的结合
  - 自顶向下: 由总体设计和规划开始 (成熟)
  - 自底向上: 由实验和原型开始 (快速)
- 软件工程的观点
  - 瀑布式: 在进行下一步之前, 每一步都进行结构化和系统的分析
  - 螺旋式 :功能渐增的系统的快速产生, 相继版本之间的间隔很短 , 快速转向
- 典型的数据仓库设计过程
  - 选取待建模的商务处理, 例如, 订单, 发票, 库存等.
  - 选取商务处理的粒度 (原子层数据), 例如， 单个事务、一天的快照等
  - 选取用于每个事实表记录的维, 如,时间、商品、顾客、供应商、仓库、事务类型和状态 等
  - 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***

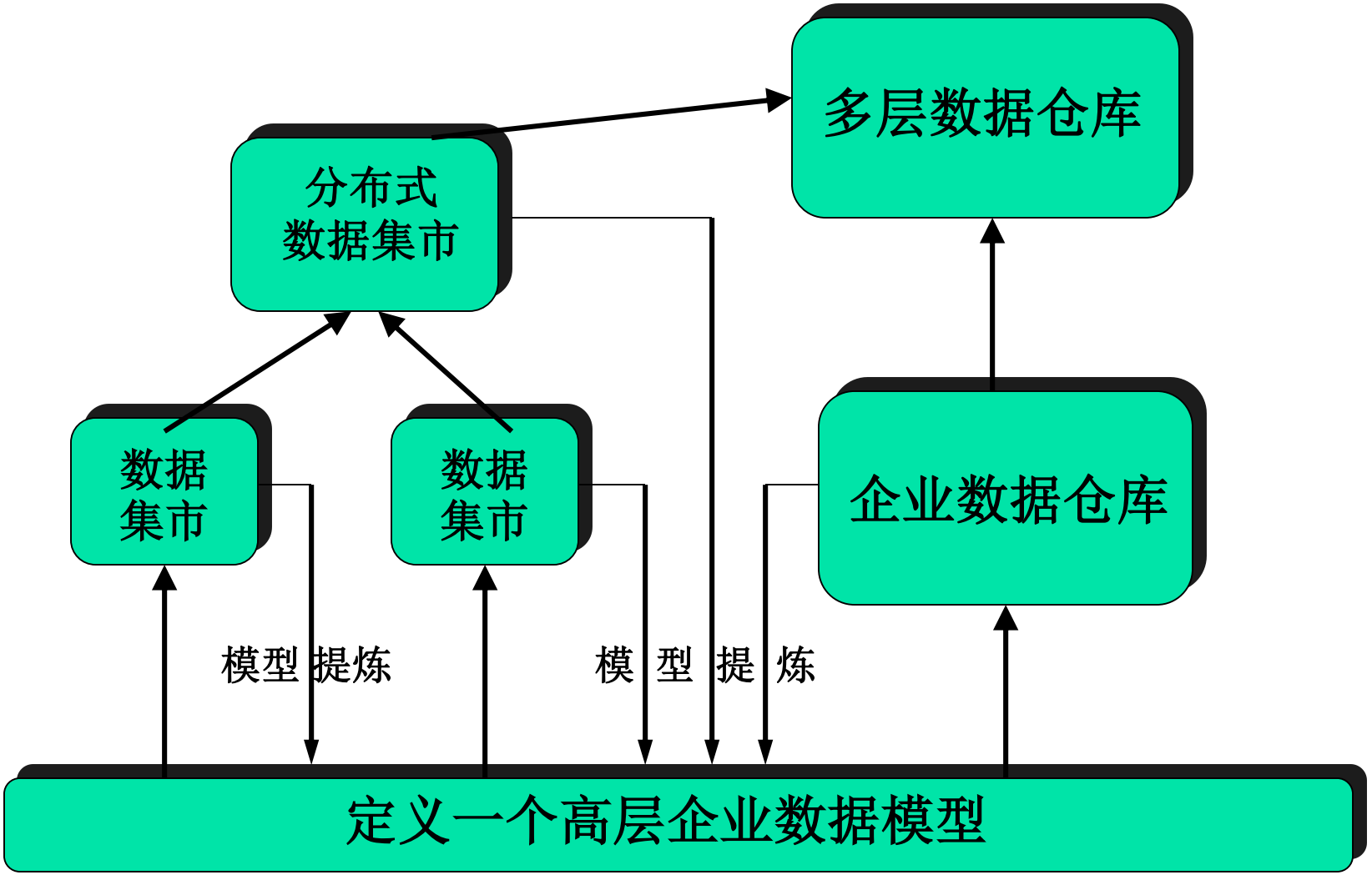
多层结构


其它
数据源
操作数据库
元数据
Monitor
&
Integrator
提取
变换
装入
刷新
数据仓库
数据集市
OLAP 服务器
输出
分析
查询
报告
数据挖掘
数据源
数据存储
OLAP引擎
终端工具

三层数据仓库模型

- 企业仓库
  - 搜集了关于主题的所有信息, 跨越整个组织
- 数据集市
  - 数据集市包含企业范围数据的一个子集, 对于特定的用户是有用的. 其范围限于选定的主题, 如销售数据
    - 独立的 vs. 依赖的 (直接来自数据仓库) 数据集市
- 虚拟仓库
  - 操作数据库上视图的集合
  - 只有部分可能的汇总视图被物化

数据仓库开发: 一种推荐的方法



OLAP 服务器结构

- 关系OLAP (ROLAP)
  - 使用关系或扩充关系的 DBMS 存放和管理仓库数据, 使用OLAP中间件支持其它部分
  - 包含一个优化的 DBMS 后端, 聚集导航逻辑的实现, 以及附加的工具和服务
  - 较大的可伸缩性
- 多维 OLAP (MOLAP)
  - 基于数组的多维存储引擎 (稀疏矩阵技术)
  - 对预计算的汇总数据快速索引
- 混合 OLAP (HOLAP)
  - 弹性, 底层: 关系的, 高层: 数组.
- 专门的 SQL 服务器
  - 对星型/雪花型模式上的SQL查询提供特殊的支持

元数据存储

- 元数据是定义数据仓库的数据. 有如下类型
  - 描述数据仓库的结构
    - 模式, 视图, 维, 分层结构, 数据源定义, 数据集市的位置和内容
  - 操作元数据
    - 数据血统 (数据变迁历史和转换路径), 数据流通 (主动, 存档, 或净化), 管理信息 (数据仓库使用统计, 错误报告, 审计跟踪)
  - 用于汇总的算法
  - 由操作环境到数据仓库的映射
  - 涉及系统性能的数据
    - 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, 数据的所有者, 收费政策

数据仓库的后端工具和实用程序

- 数据提取:
  - 由多个异种, 外部数据源收集数据
- 数据清理:
  - 检测数据中的错误, 可能时订正它们
- 数据变换:
  - 将数据由遗产或宿主格式转换成数据仓库格式
- 装载:
  - 排序, 综合, 加固, 计算视图, 检查整体性, 并建立索引和划分
- 刷新
  - 传播由数据源到数据仓库的更新

第2章: 数据挖掘的数据仓库与OLAP技术

- 什么是数据仓库?
- 多维数据模型
- 数据仓库结构
- 数据仓库实现
- 从数据仓库到数据挖掘
- 数据立方体的进一步发展

数据方的有效计算

- 数据方可以视为方体的格
  - 最下面的方体是基本方体
  - 最上面的 (顶点) 方体只包含一个单元
  - 具有L层的n-D数据方包含多少个方体?
  - 其中Li是与维i相关联的层数

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

- 数据方的物化(Materialization)
  - 物化每一个方体 (全物化), 不物化任何方体(不物化), 或物化某些方体(部分物化)
  - 物化方体的选择
    - 基于大小, 共享, 访问频率, 等.

数据方计算

- 用DMQL定义和计算数据方

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

  compute cube sales

- 将它变换成类——SQL语句 (用新的操作 **cube by扩充**, 由Gray 等’96 引进)

  SELECT item, city, year, SUM (amount)

  FROM SALES

  CUBE BY item, city, year

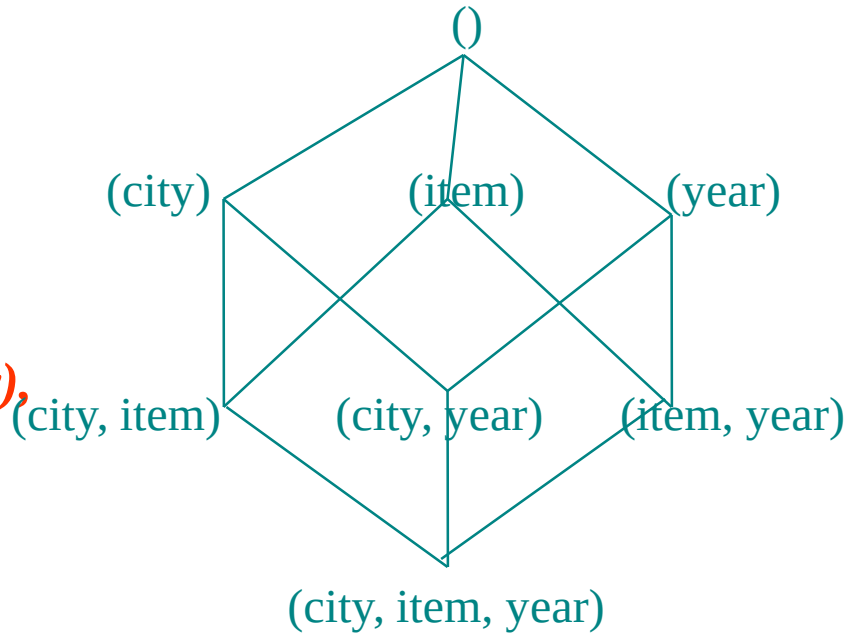
- 需要计算的分组

  (city, item, year),

  (city,item),(city, year), (item, city),

  (city), (item), (year)

  ()



数据方计算: 基于ROLAP的方法(1)

- 有效的方计算方法
  - 基于ROLAP的方计算算法 (Agarwal et al'96)
  - 基于数组的方计算算法 (Zhao et al'97)
  - 自底向上的方法 (Beyer & Ramarkrishnan'99)
  - 混合的方法 (Han, Pei, Dong & Wang:SIGMOD'01)
- 基于ROLAP的方计算算法
  - 排序, 散列,和分组操作用于维属性, 以便对相关元组重新排序和分簇
  - 在某些子聚集上分组，作为“部分分组”
  - 由以前计算的聚集计算新的聚集，而不必由基本事实表计算

数据方计算: 基于ROLAP的方法(2)

- 取自研究论文
- 基于Hash/排序 的方法 (Agarwal 等. VLDB'96)
  - 最小双亲(Smallest-parent): 由最小的, 先前计算的方体计算方体
  - 存储结果(Cache-results): 存储先前计算的方体, 由它可以计算其它方体, 以减少磁盘I/O
  - 分摊扫描(Amortize-scans): 同时计算尽可能多的方体, 以分摊磁盘的读操作开销
  - 共享排序(Share-sorts): 使用基于排序的方法时, 在多个方体之间共享排序开销
  - 共享划分(Share-partitions): 使用基于hash的方法时, 在多个方体之间共享划分开销

索引OLAP 数据

- 为了有效的访问，大部分数据仓库系统支持索引结构
- 两种常用的方法对OLAP数据进行索引
  - 位图索引 **bitmap indexing**
  - 连接索引 **join indexing**

索引OLAP 数据: 位图索引

- 在一个特定列上索引
- 列上的每个值是一个位向量 : 位操作很快
- 位向量的长度: 基本表的记录数
- 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

基本表

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

在 **Region**上索引

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

在 **Type**上索引

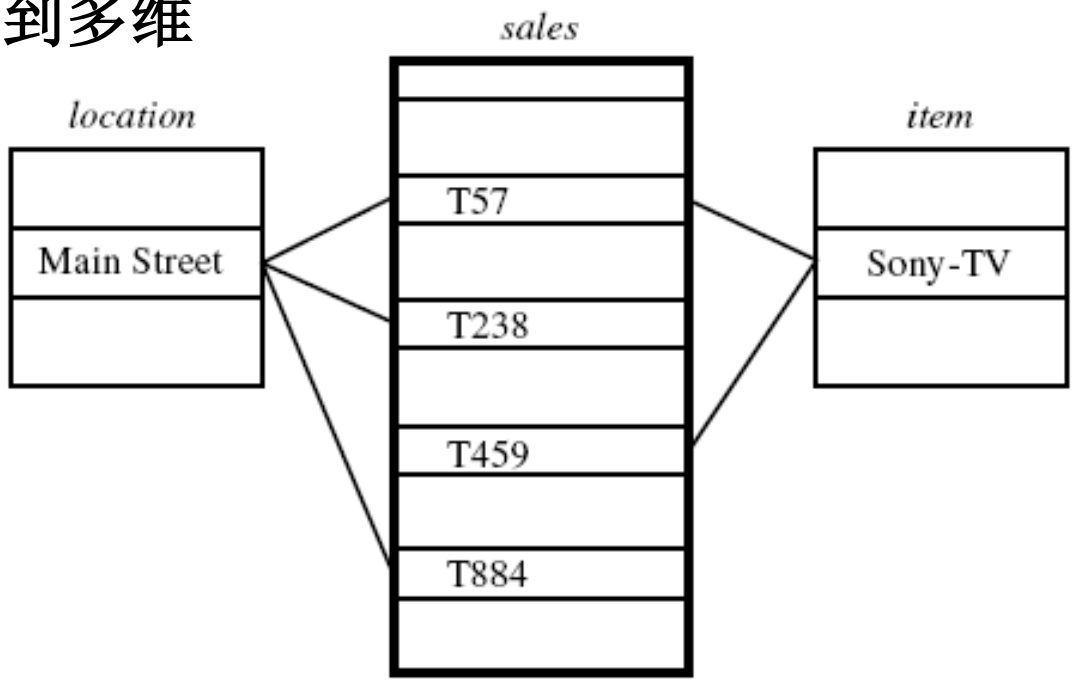
RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

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, …)**
  - 将关系的连接物化在**JI**文件中, 加快了关系连接的速度
- 数据仓库中, 连接索引将星型模式维表的值关联到事实表的行.
  - 例, 事实表***Sales*** 和两个维 ***city*** 和 ***product***
    - ***city*** 上的连接索引对每个不同的城市, 维护一张记录该城市销售的元组的**R**
  - 连接索引可以扩展到多维

Join index table for *item/sales*

item	*sales_key*
. . .	. . .
Sony-TV	T57
Sony-TV	T459
. . .	. . .



OLAP查询的有效处理

- 物化方体和构造OLAP索引结构的目的是加快数据立方体的查询处理速度。
- 查询处理按如下步骤进行：
- 确定哪些操作可以在可用的方体上进行：
  - 将下钻, 上卷等操作变换成对应的SQL和/或OLAP操作, 例如, dice = selection + projection
- 确定相关的操作应当使用哪些物化的方体.

第3章: 数据挖掘的数据仓库与OLAP技术

- 什么是数据仓库?
- 多维数据模型
- 数据仓库结构
- 数据仓库实现
- 从数据仓库到数据挖掘
- 数据立方体的进一步发展

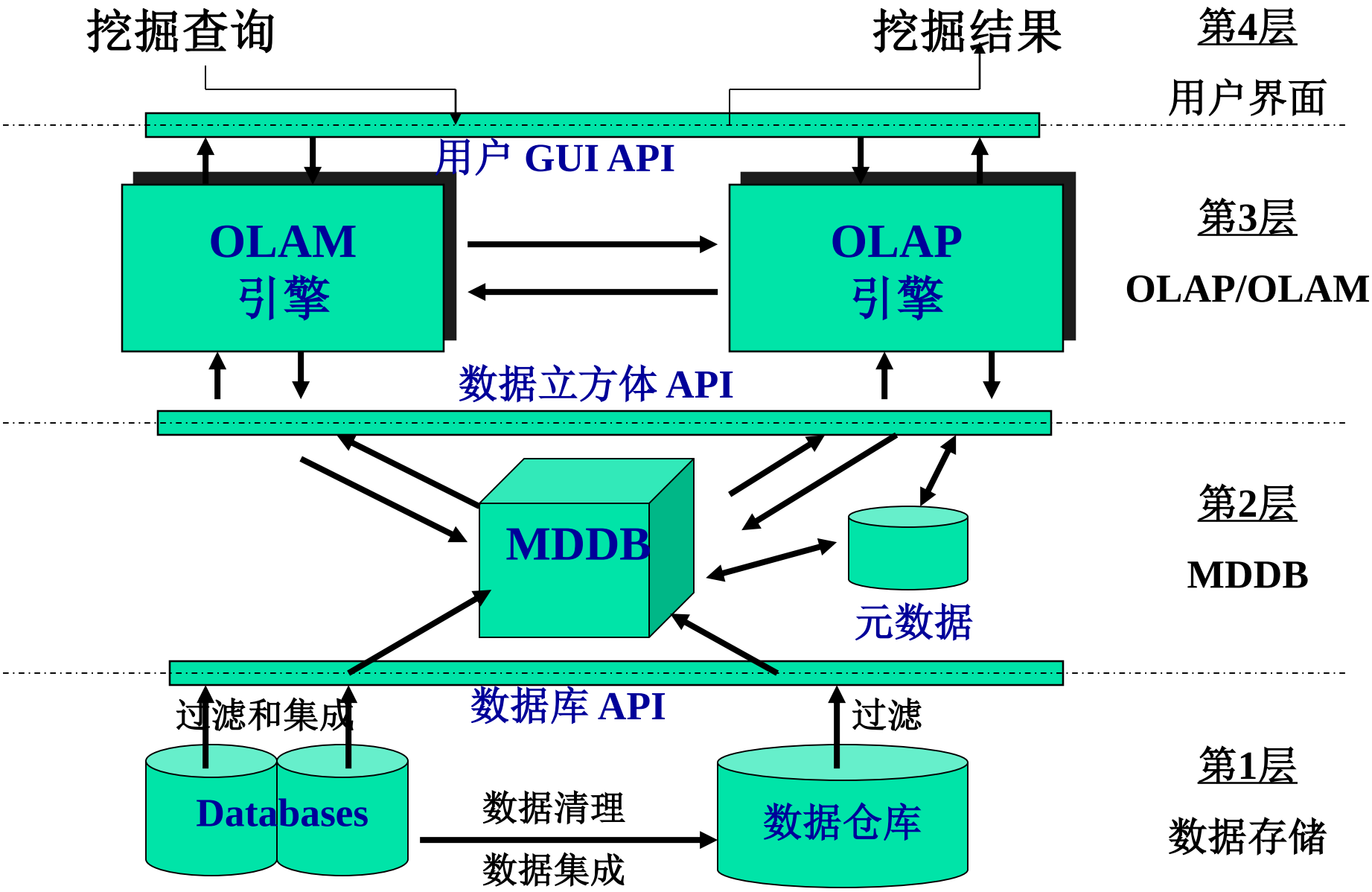
数据仓库使用

- 数据仓库应用的三种类型
  - 信息处理
    - 支持查询, 基本统计分析, 使用交叉表, 表, 图表和图进行报告
  - 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, 上下钻, 转轴
  - 数据挖掘
    - 隐藏模式的知识发现
    - 支持关联, 构造分析模型, 进行分类和预测, 并使用可视化工具提供挖掘结果.
- 三类任务的差别

从联机分析处理到联机分析挖掘

- 为什么要进行联机分析挖掘(OLAM)?
  - 数据仓库中数据的高质量
    - 数据仓库包含集成的, 一致的, 清理过的数据
  - 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的探测式数据分析
    - 使用上下钻, 切片, 切块, 转轴等进行挖掘.
  - 数据挖掘功能的联机选择
    - 集成多种挖掘功能, 算法和任务, 并进行切换.
- OLAM的结构

OLAM 的结构


挖掘查询
挖掘结果
第4层
用户界面
用户 GUI API
OLAM
引擎
OLAP
引擎
第3层
OLAP/OLAM
数据立方体 API
MDDB
元数据
第2层
MDDB
数据库 API
过滤和集成
过滤
Databases
数据清理
数据集成
数据仓库
第1层
数据存储

小结

- 数据仓库
- 数据仓库的 多维数据模型
  - 星型模式, 雪花模式, 事实星座
  - 数据方由维和度量组成
- **OLAP** 操作: 下钻, 上卷, 切片, 切块 和转轴
- **OLAP** 服务器: **ROLAP, MOLAP, HOLAP**
- 数据方的有效计算
  - 部分 **vs.** 全部 **vs.** 不物化
  - 多路数组聚集
  - 位图索引和连接索引的实现