



Course 3

数据仓库与OLAP技术概述

Data Warehouse and OLAP Technology

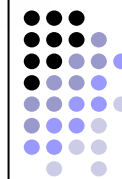
Data Mining

数据挖掘

大纲

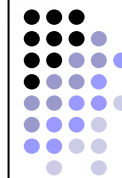


- 什么是数据仓库？
- 多维数据模型
- 数据仓库的系统结构
- 数据仓库实现
- 从数据仓库到数据挖掘



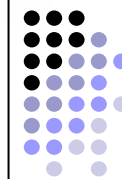
■ 什么是数据仓库？

- 数据仓库已被多种方式定义但没有一种严格的定义。
 - 一个与组织机构的操作数据库分别维护的决策支持数据库。
 - 为统一的历史数据分析提供坚实的平台，对信息处理提供支持。
- “数据仓库是一个面向主题的、集成的、时变的、非易失的数据集合，支持管理过程的决策过程” —W. H. Inmon
- 建立数据仓库
 - 构造和使用数据仓库的过程



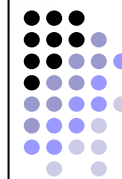
■ 数据仓库——面向主题的

- 围绕一些主题如**顾客、供应商、产品和服务组织**。
- 关注于决策者的数据建模和分析，而不是集中于组织机构的日常操作和事务处理。
- 数据仓库**排除与对于决策无用的数据**，提供特定主题的简明视图。



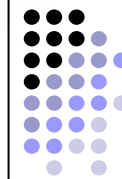
■ 数据仓库——集成的

- 通过集成多个异种数据源而构成。
 - 关系数据库、一般文件和联机事务处理记录。
- 使用数据清理和数据集成技术。
 - 在不同的数据源中，确保命名约定、编码结构、属性度量等的一致性。
 - 例如，旅馆价格：由住宿费、税收、附带的早餐费等构成。
 - 数据被移到数据仓库时就进行了数据转换。



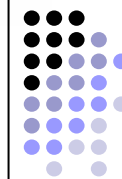
■ 数据仓库——时变的

- 数据仓库的时间范围明显长于操作系统。
 - 操作数据库：当前的有用信息。
 - 数据仓库数据：从历史的角度提供信息（例如：过去的5-10年）
- 数据仓库的每一个关键结构
 - 隐式或显示的包含时间元素
 - 但操作数据的关键结构可以包含也可以不包含“时间元素”



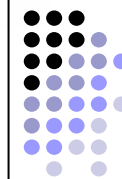
■ 数据挖掘——非易失的

- 数据仓库总是物理地分离存放数据，这些数据源于操作环境下的应用数据
- 操作性的数据更新不会发生在数据仓库的环境下。
 - 数据仓库不需要事务处理、恢复和并发控制机制
 - 它只需要两种数据访问：数据的初始装入和数据访问



数据仓库 vs. 异构DBMS

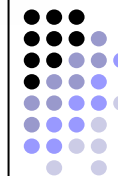
- 传统的异构数据库的集成: **查询驱动方法**
 - 在异种数据库的顶部建立一个 **包装程序** 和 **集成程序**
 - 当一个查询提交客户站点，首先使用元数据字典对查询进行转换，将它转换成相应异种站点上的查询，然后，不同站点返回的结果被集成为全局回答
 - 查询驱动方法需要复杂的信息过滤，并且与局部数据源上的处理竞争资源
- 数据仓库：使用 **更新驱动** 的方法，为集成的异种数据库系统带来了高性能
 - 将来自多个异种源的信息预先集成，并存储与数据仓库中，供直接查询和分析



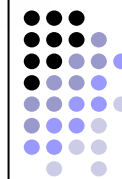
数据仓库 vs. 操作数据库系统

- 联机事务处理OLTP (on-line transaction processing)
 - 传统的关系DBMS的主要任务
 - 他们涵盖了一个组织的大部分日常操作：购买、库存、制造、银行、工资、注册、记账等。
- 联机分析处理OLAP (on-line analytical processing)
 - 数据仓库系统的主要任务
 - 数据分析和决策
- OLTP和OLAP的区别
 - 用户和系统的面向性:OLTP面向顾客，而OLAP面向市场
 - 数据内容：OLTP系统管理当前数据，而OLAP管理历史的数据。
 - 数据库设计：OLTP系统采用实体-联系 (ER)模型和面向应用的数据库设计，而OLAP系统通常采用星形和雪花模型
 - 视图：OLTP系统主要关注一个企业或部门内部的当前数据，而OLAP系统主要关注汇总的统一的数据。
 - 访问模式：OLTP访问主要有短的原子事务组成，而OLAP系统的访问大部分是只读操作，尽管许多可能是复杂的查询

OLTP vs. OLAP



特征	OLTP	OLAP
任务特点	操作处理	信息处理
面向	事务	分析
用户	办事员、DBA、数据库专业人员	经理、主管、数据分析员
功能	日常操作	长期信息分析、决策支持
DB设计	基于E-R，面向应用	星型/雪花，面向主题
数据	最新的、详细的	历史的、汇总的
视图	详细的、二维关系型	汇总的、多维的
任务单位	简短的事务	复杂的查询
访问数据量	数十个	数百万个
用户数	数千个	数百个
DB规模	100M-数GB	100GB-数TB
优先性	高性能、高可用性	高灵活性、端点用户自治
度量	事务吞吐量	查询吞吐量、响应时间



■ 为什么需要一个分离的数据仓库？

■ 提高两个系统的性能

- 数据库管理系统— OLTP: 存取方法，索引，同步控制，恢复
- 数据仓库— OLAP: 复杂的OLAP查询，多维视图，合并

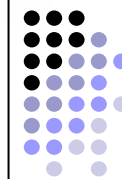
■ 不同的功能和不同的数据：

- 数据维护：决策支持需要历史数据，而操作数据库一般不维护历史数据
- 数据统一：决策支持需要将来自异种源的数据统一（如聚集和汇总）
- 数据质量：不同的数据源通常使用不一致的数据表达，代码和形式，这些都需要协调

大纲



- 什么是数据仓库？
- 多维数据模型
- 数据仓库的系统结构
- 数据仓库实现
- 从数据仓库到数据挖掘



由表和电子数据表到数据立方体

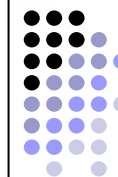
- 一个数据仓库建立在**多维数据模型**上，多维数据模型把数据看成数据立方体的形式
- 一个数据立方体，像sales,允许以多维对数据建模和观察
 - 维表，例如item的维表包含属性(item_name, brand, type), time的维表包含属性(day, week, month, quarter, year)
 - 事实表包含度量 (例如dollars_sold) 和每个相关维表的关键字
- 数据仓库语义中，一个 n-D 底层方体称为**基本方体**. 最高层的0-D方体,存放最高层的汇总,称为**顶点方体**. 所有的方体格组成了**数据立方体**

由表到数据立方体

■ 2-D表：

<i>location</i> = “Vancouver”				
<i>time</i> (季 度)	<i>item</i> (类型)			
	家 庭 娱 乐	计 算 机	电 话	安 全
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

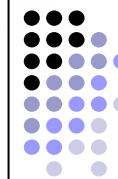
表3-2 AllElectronics的销售数据time和item维德2-D视图。其中销售是取自坐落在城市温哥华的所有分店，所显示的度量是dollars_sold（单位：千美元）



■ 3-D表

	<i>location= "Chicago"</i> <i>item</i>				<i>location= "New York"</i> <i>item</i>				<i>location= "Toronto"</i> <i>item</i>				<i>location= "Vancouver"</i> <i>item</i>			
<i>time</i>	家庭娱乐	计算机	电话	安全	家庭娱乐	计算机	电话	安全	家庭娱乐	计算机	电话	安全	家庭娱乐	计算机	电话	安全
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

表3-3 AllElectronics销售数据的time、item和location维德3-D视图，所显示的度量是dollars_sold（单位：千美元）



■ 3-D数据立方体

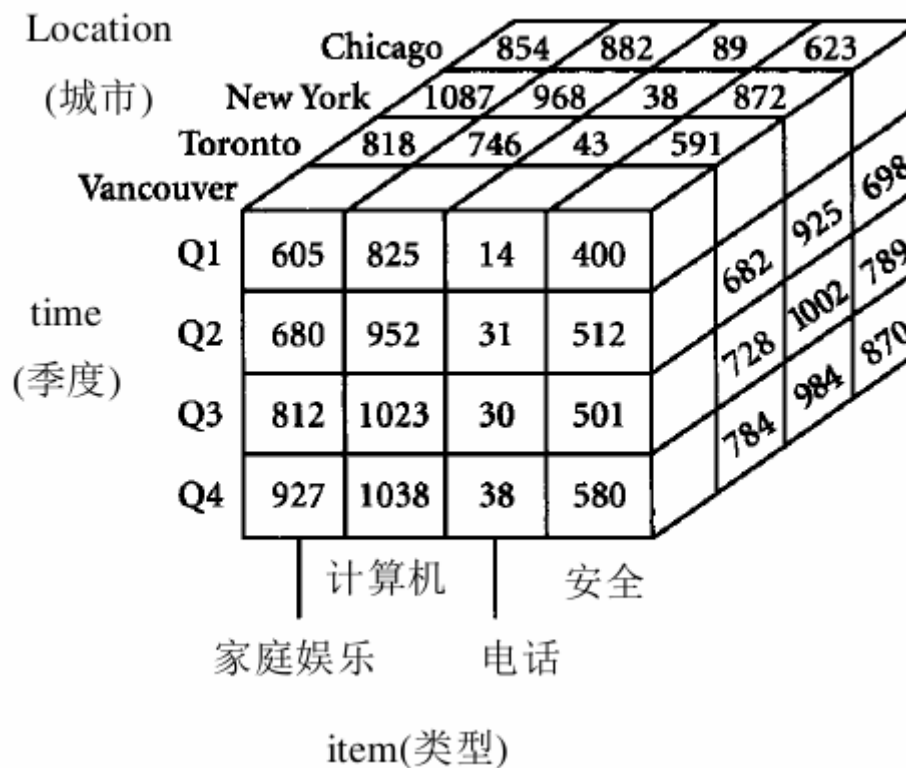
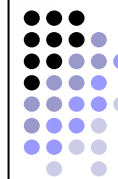


图3-1 表3-3数据的3-D数据立方体表示，维是time、item和location。
所显示的度量为dollars_sold（单位：千美元）



■ 4-D数据立方体

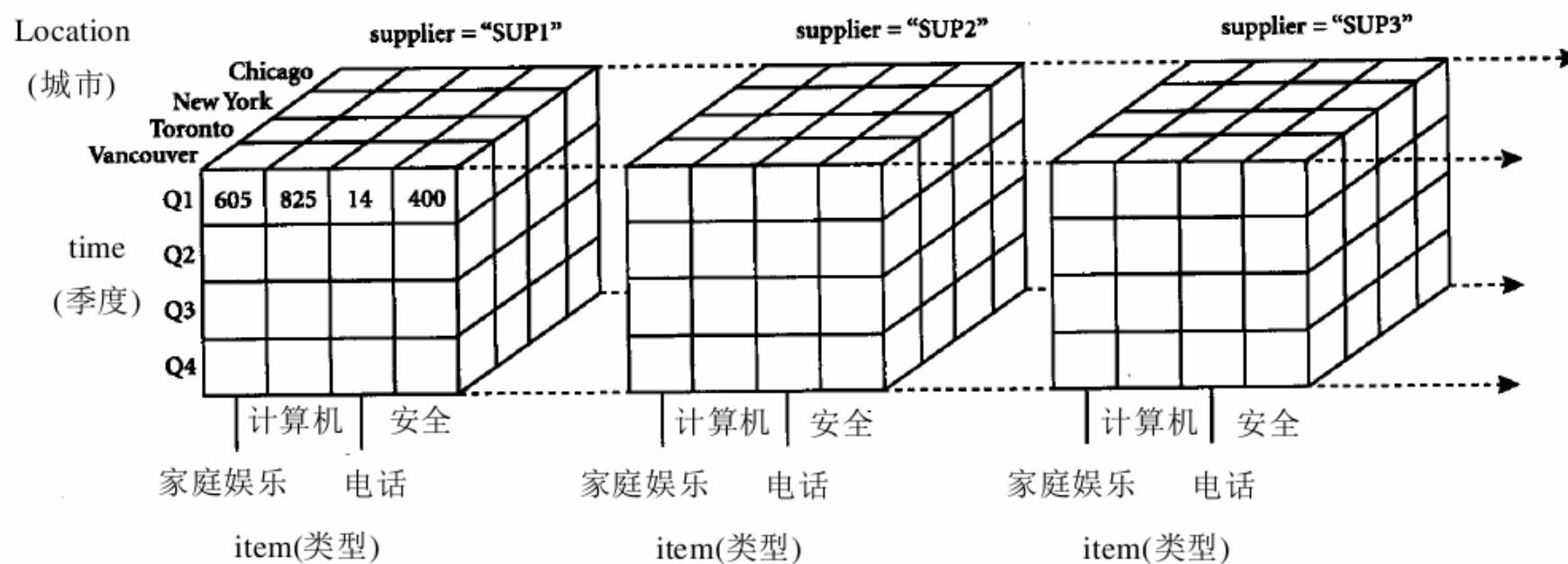
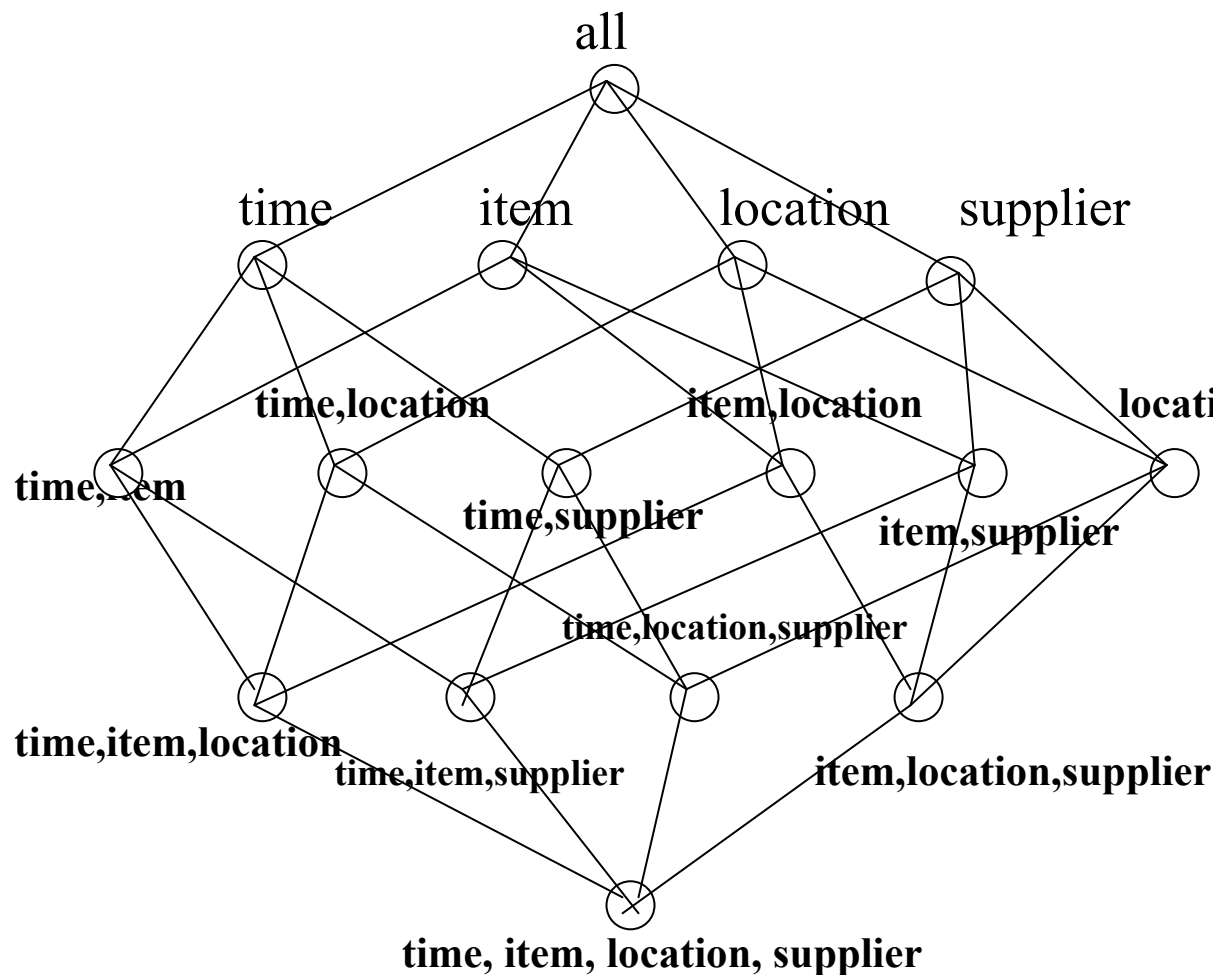
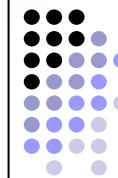


图3-2 销售数据的4-D数据立方体表示，维是time、item、location和supplier。所显示的度量为dollars_sold（单位：千美元）。

立方体：一个方体格



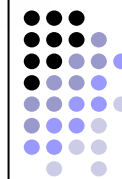
0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D cuboids

4-D(base) cuboid

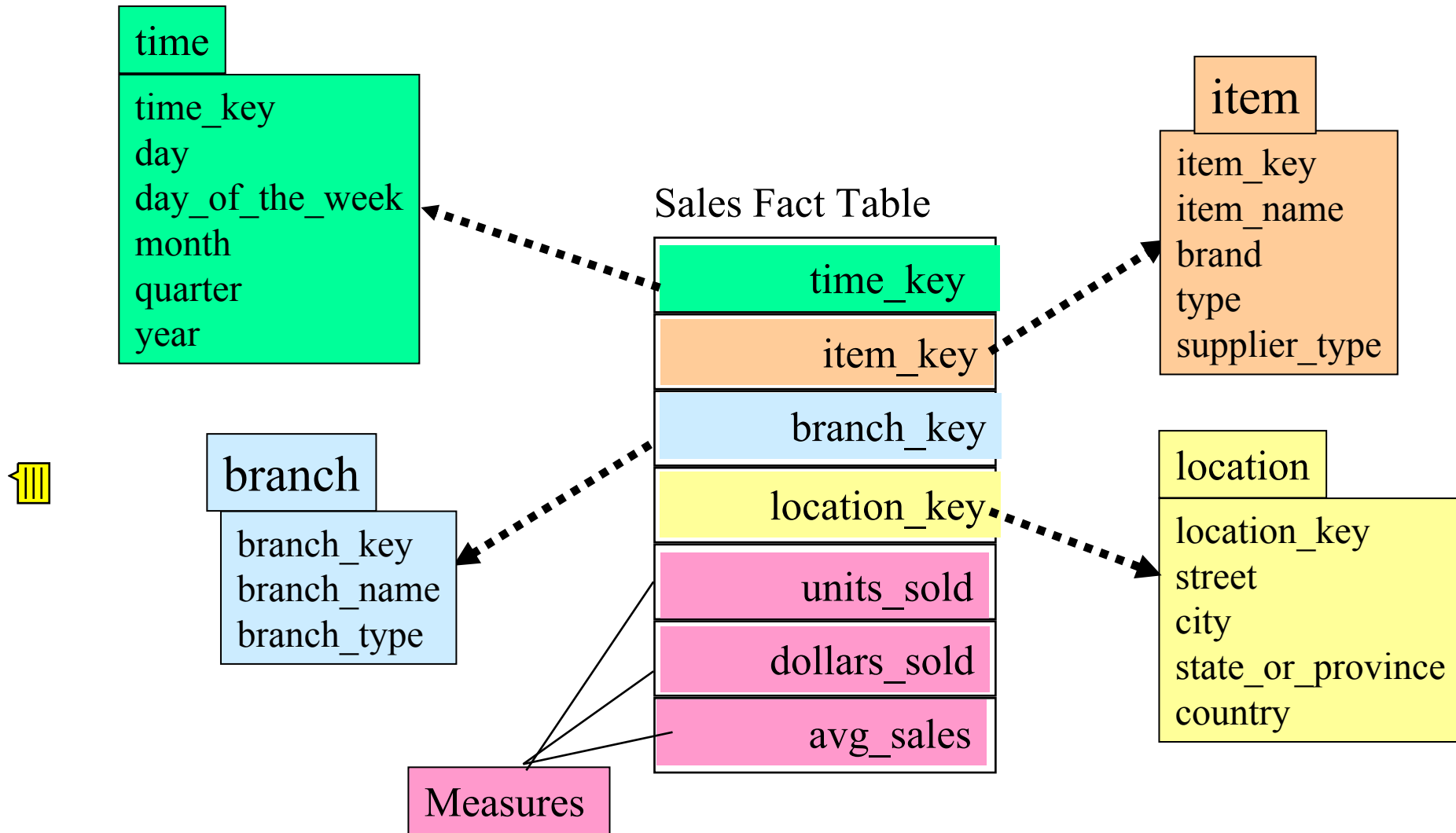


■ 数据仓库的概念性模型

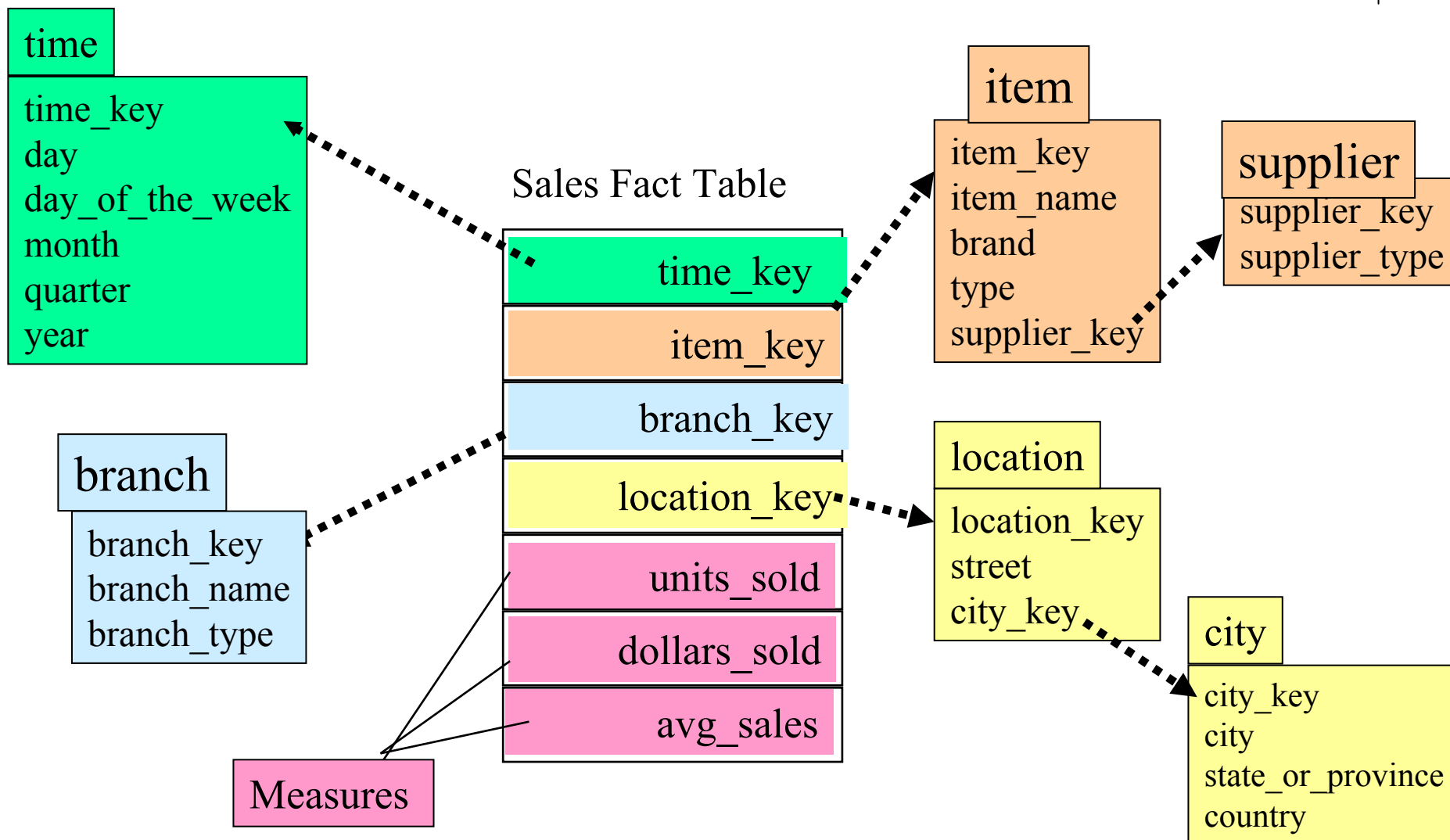
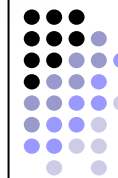
■ 建立数据仓库模型：维与度量

- **星型模型**：中间是事实表，连接一组维表
- **雪花模式**：雪花模式是星型模式的变种，其中某些维表示规范化的，而数据进一步分解到附加的维表中，它的图形类似于雪花的形状
- **事实星座表**：多个事实表共享维表，这种模式可以看作星型模式及，因此称为星系模式或事实星座

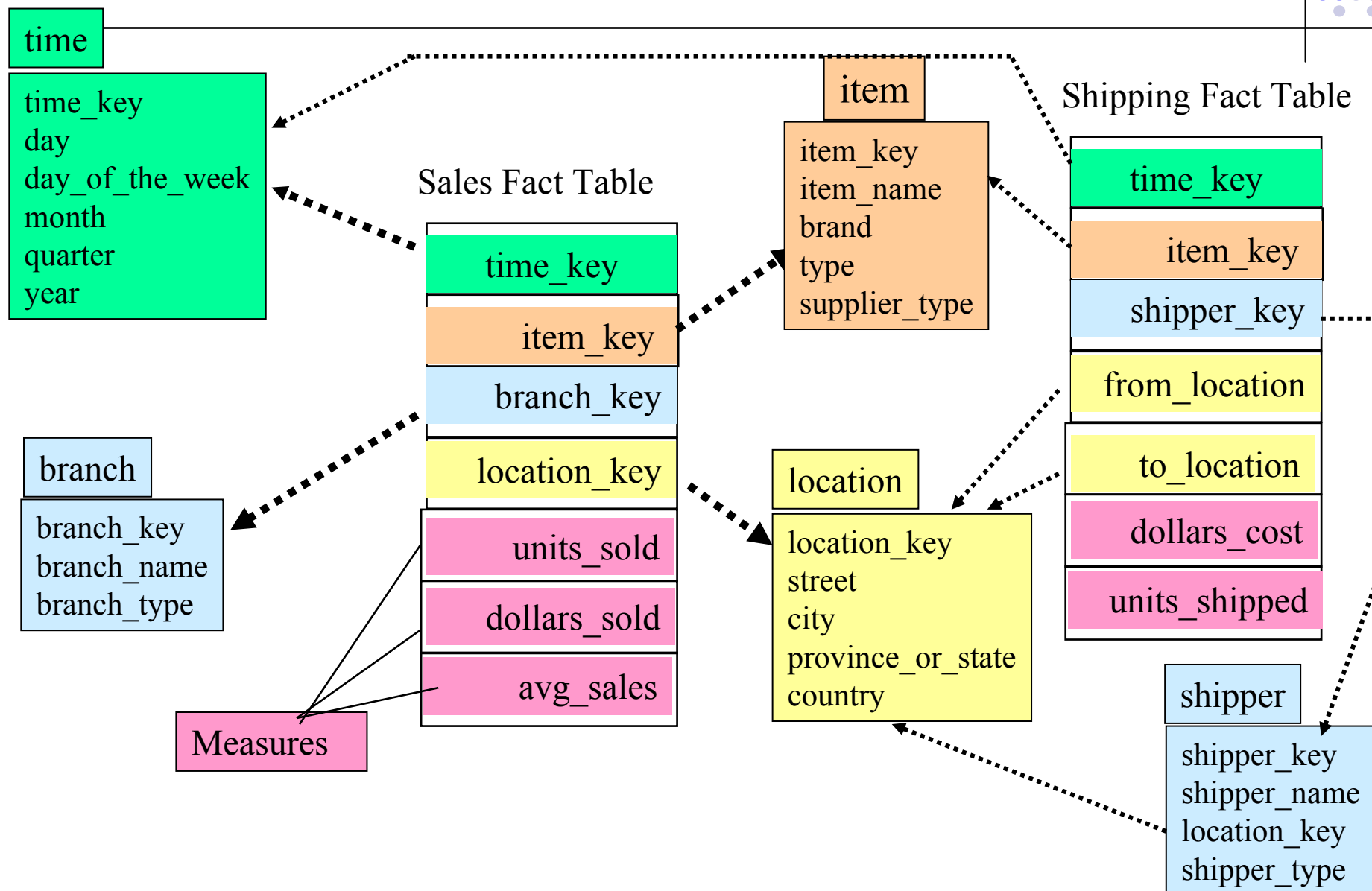
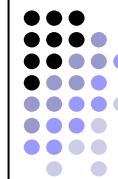
星型模式的例子

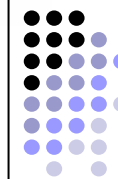


雪花模式的例子



事实星座模式的例子





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>

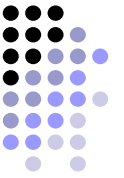
■ 维定义（维表）

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

■ 特殊情况（共享维表）

□ 首先进行“立方体定义”

□ **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)

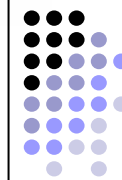


用DMQL定义雪花模式

- **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定义星系模式

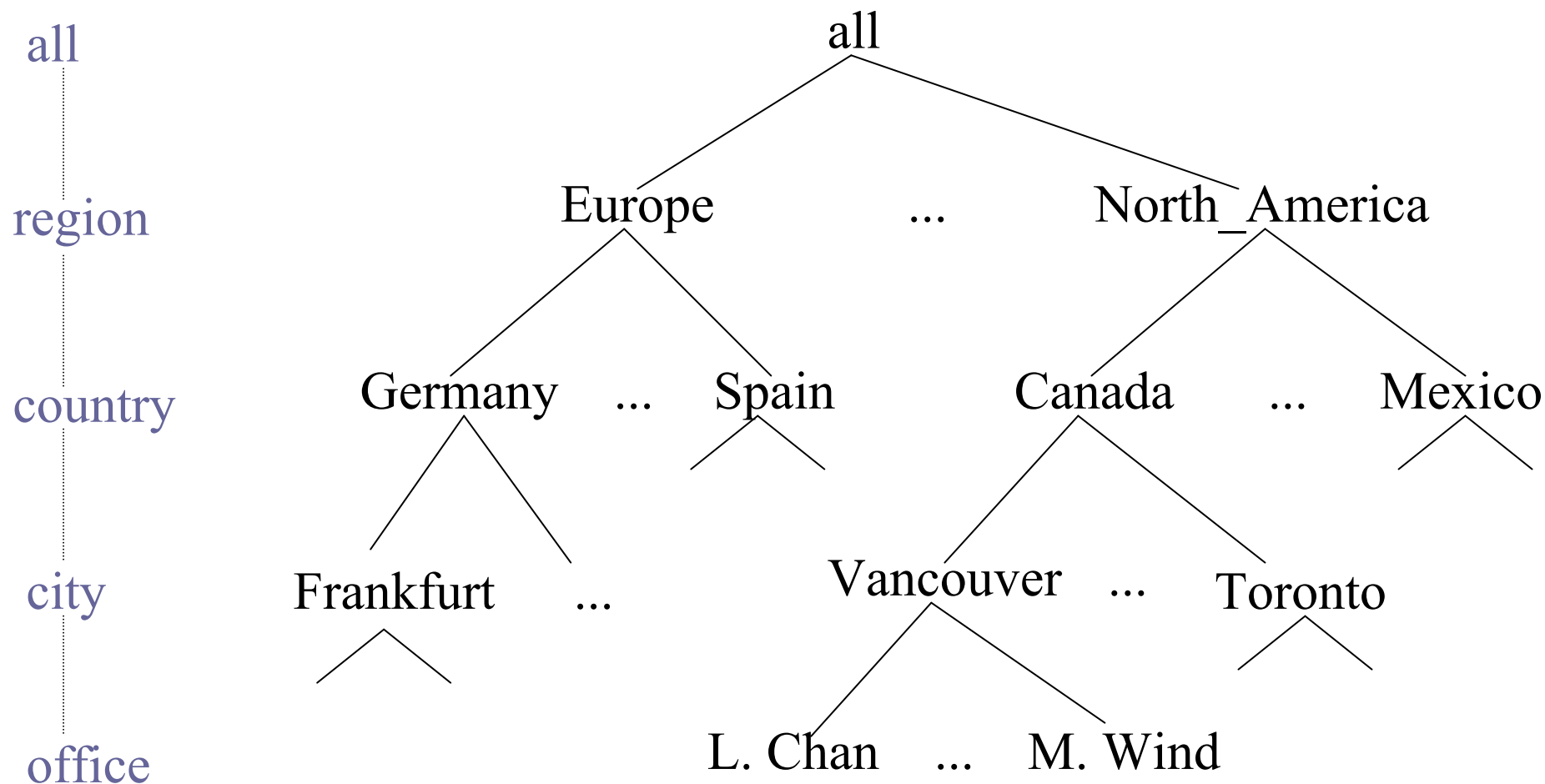
- **define cube** sales [time, item, branch, location]: **dollars_sold** = **sum(sales_in_dollars)**, **avg_sales** = **avg(sales_in_dollars)**, **units_sold** = **count(*)**
- **define dimension** time **as** (time_key, day, day_of_week, month, quarter, year)
- **define dimension** item **as** (item_key, 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



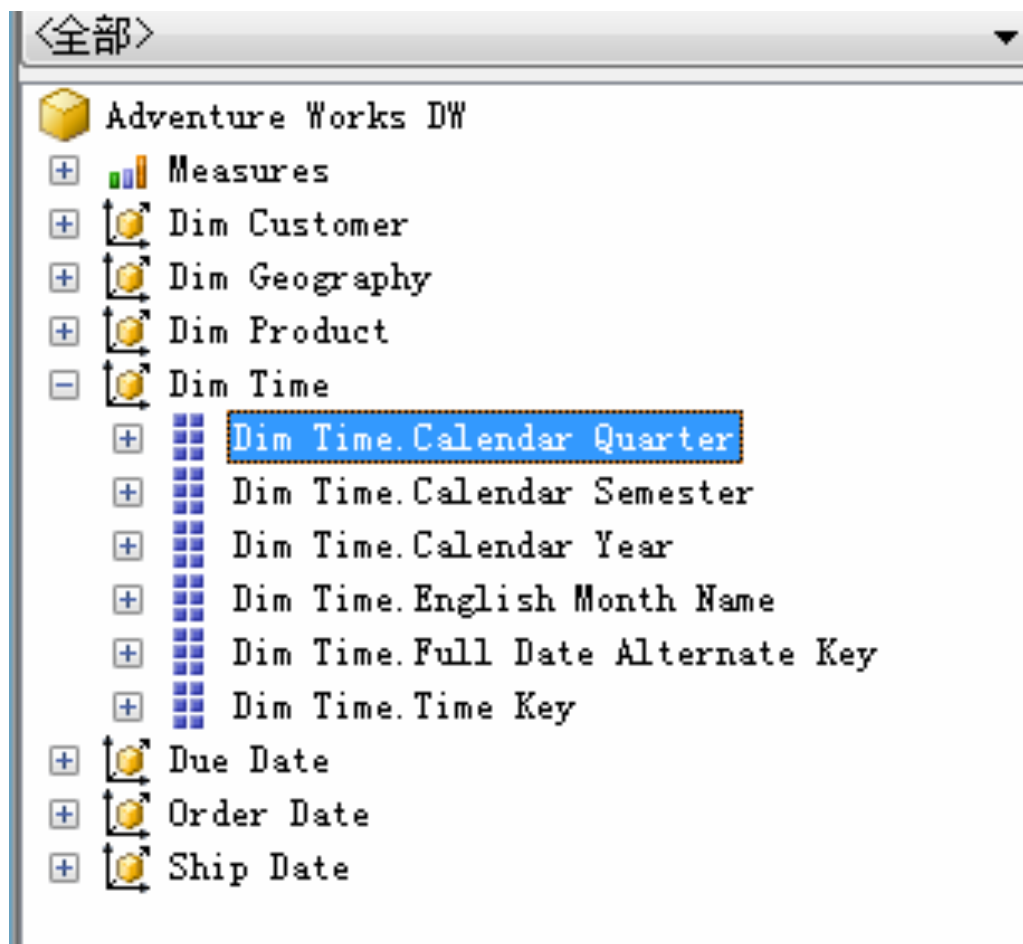
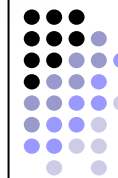
度量的三种分类

- 分布的 (distributive) : 如果将函数用于n个聚集值得到的结果, 与将函数用于所有的数据得到的结果一样
 - 例如, `count()`, `sum()`, `min()`, `max()`.
- 代数的 (algebraic) : 如果它能够有一个具有M参数的代数函数计算 (其中M是一个有界整数) 而每个参数都可以用一个分布聚集函数求得。
 - 例如, `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。
 - 例如, `median()`, `mode()`, `rank()`.

概念分层：location维的一个概念分层



仓库和分层视图



分层的详述

- 模式分层

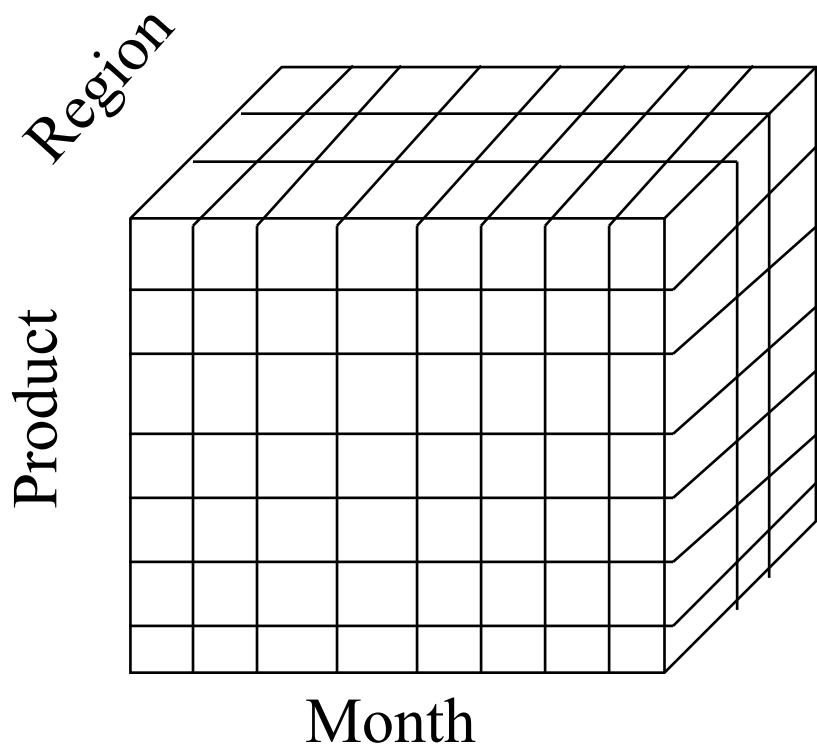
day < {month <
quarter; week} < year

- 集合分组分层

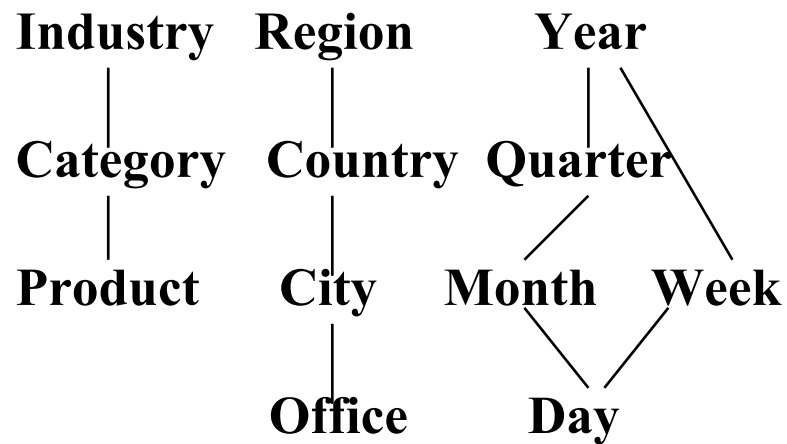
{1..10} < inexpensive

多维数据

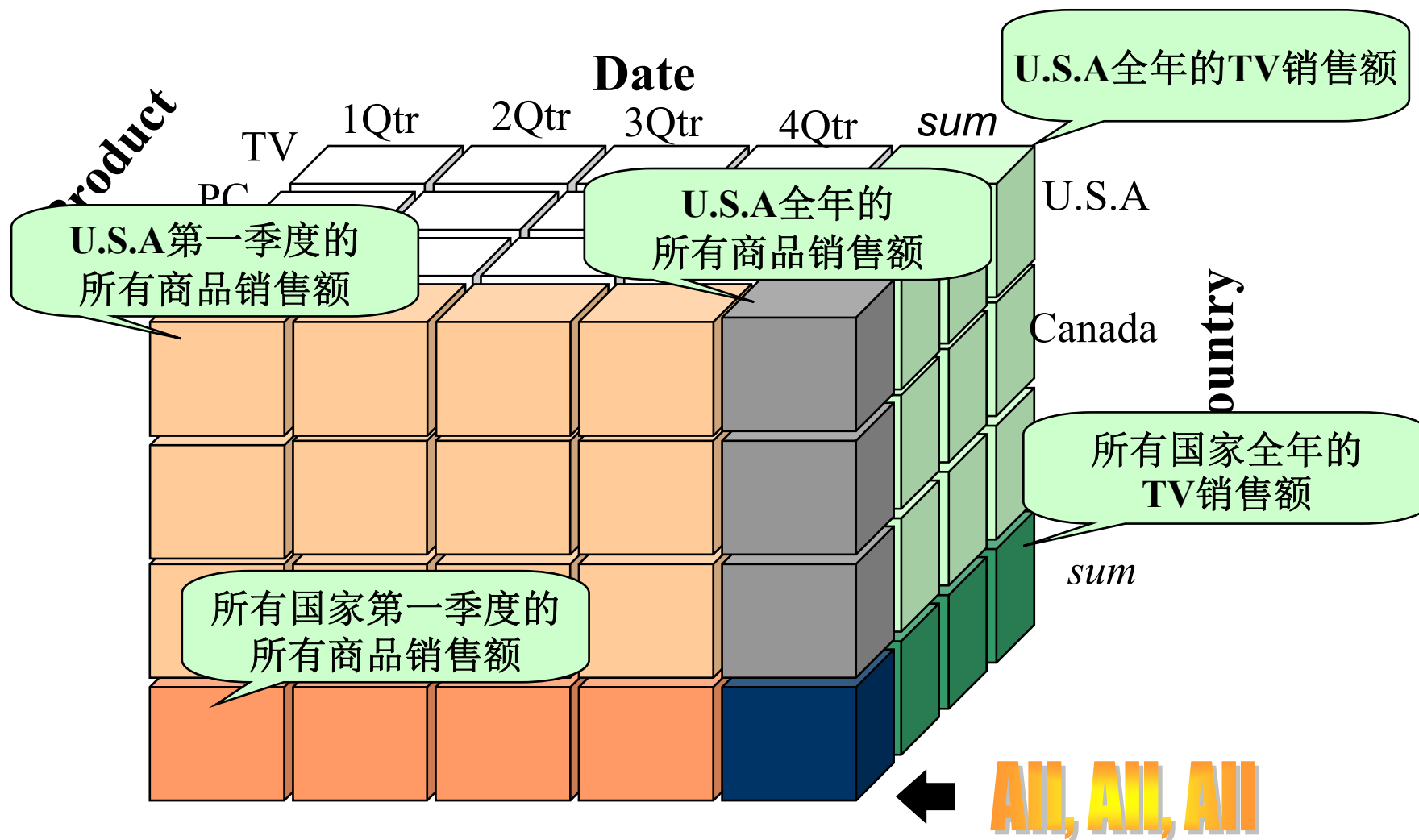
- 具有 product, month, and region三个维的销售立方体

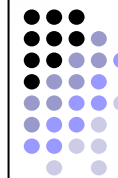


维: Product, region, month
分层的汇总路径

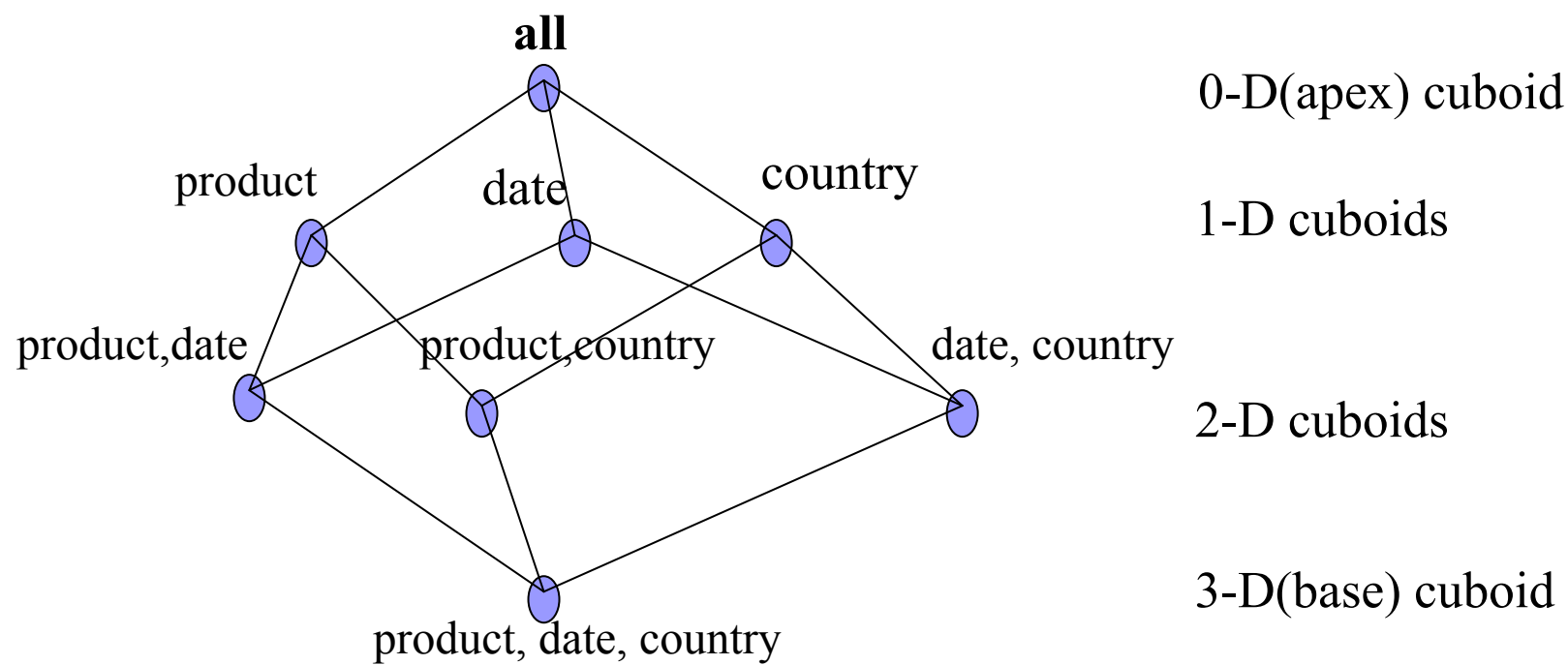


一个数据立方体的例子





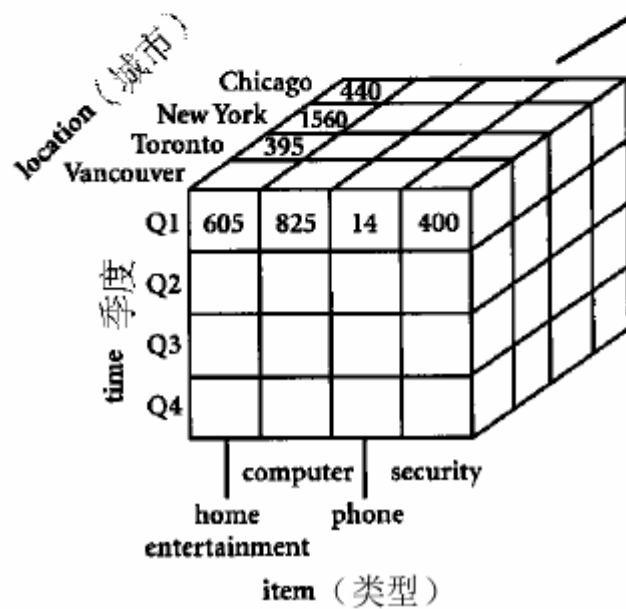
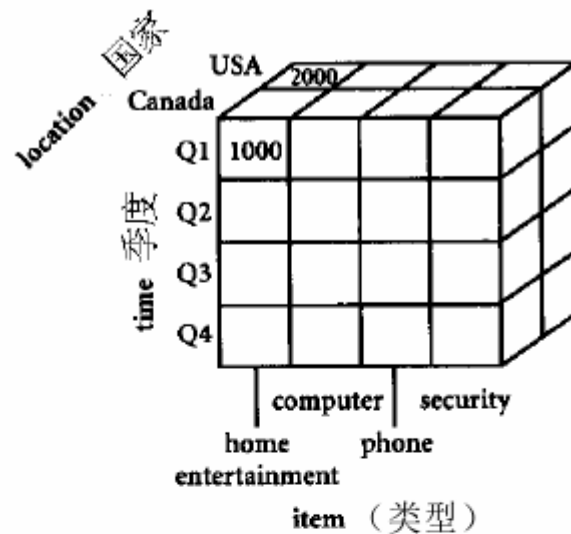
对应于立方体的方体格



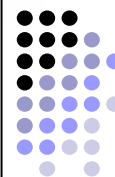
■ 典型的OLAP操作

- 上卷 (Roll up) : 汇总数据
 - 通过维的概念分层向上攀升或者通过维归约来实现
- 下钻 (roll down) : 上卷的逆操作
 - 从高层的汇总到低层汇总或详细数据, 或者引入新的维来实现
- 切片 (Slice) 和切块 (dice) :
 - 映射和选择
- 转轴 (Pivot) :
 - 是一种目视操作, 它转动数据的视角, 提供数据的替代表示。
如: 将一个3-D立方体转换成2-D平面序列。
- 其他的操作 :
 - 钻过 (drill across): 涉及多个事实表的查询
 - 钻透 (drill through) : 钻到数据立方体的底层, 到后端关系表 (使用SQL)

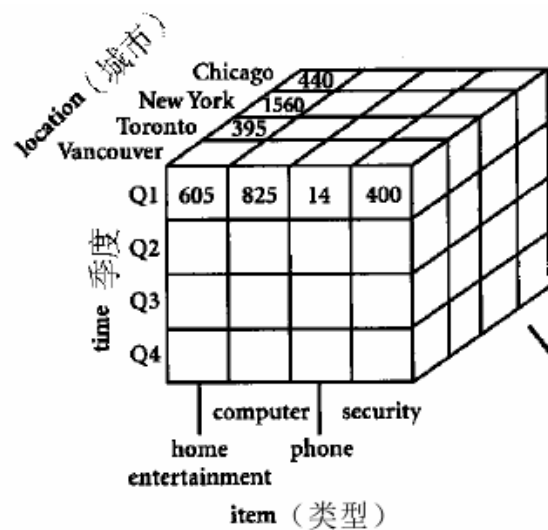
上卷



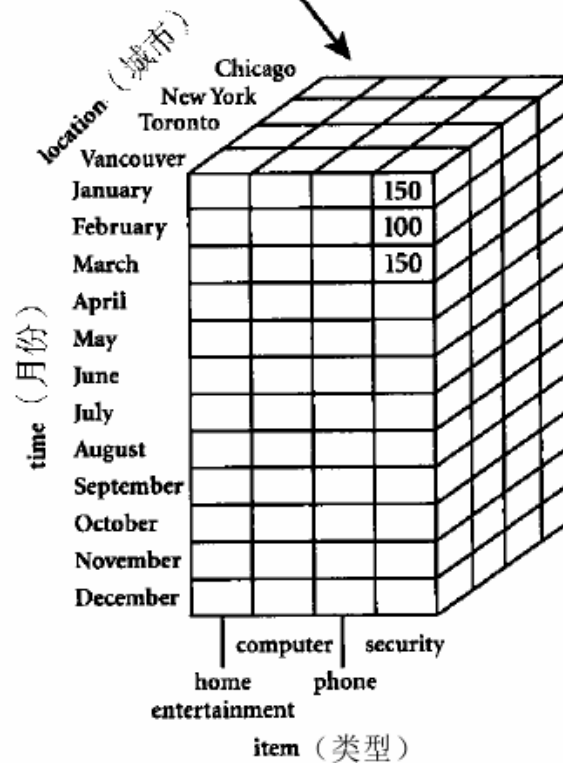
按location
上卷 (从城市到国家)



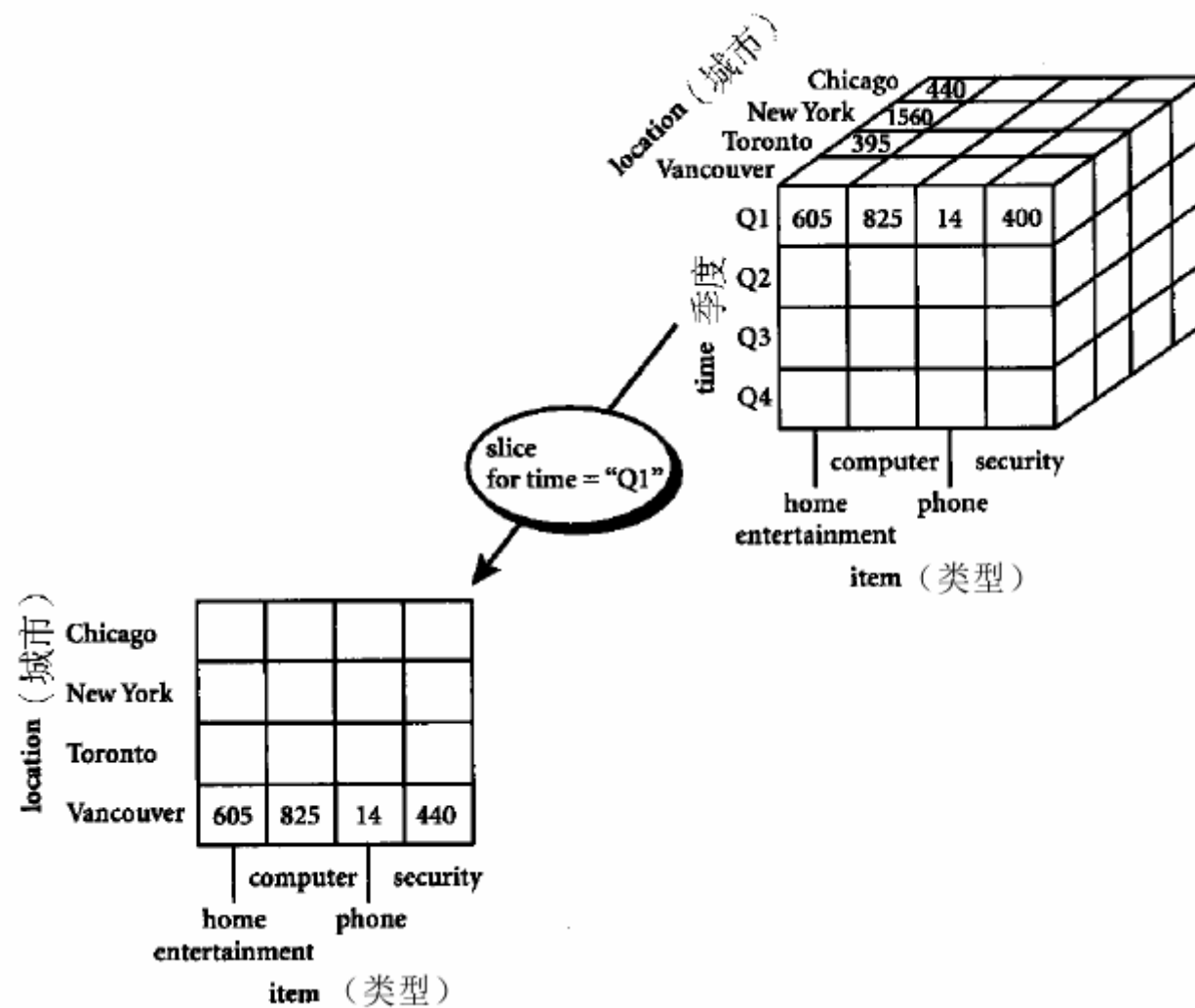
下钻



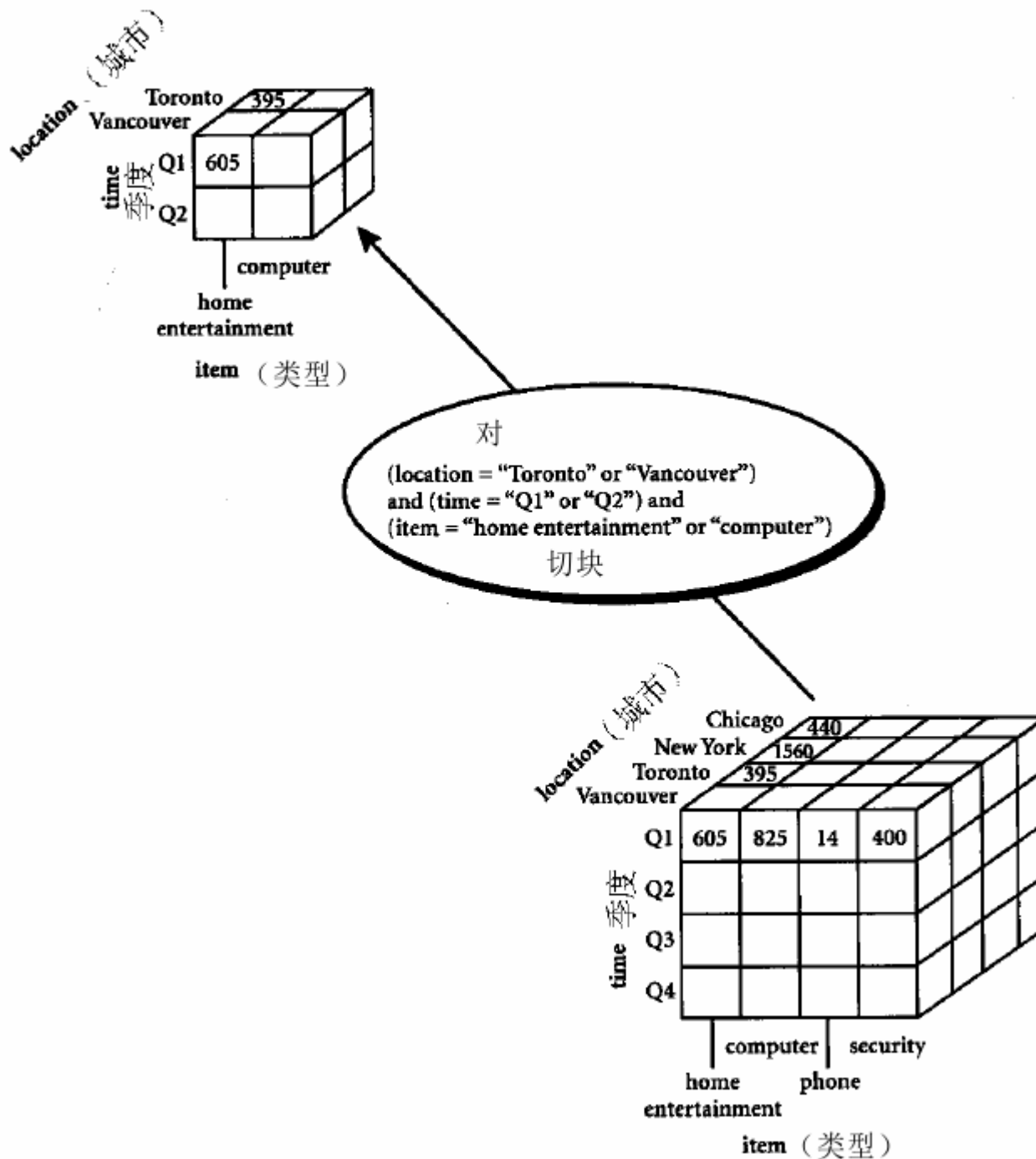
按time
下钻 (从季度
到月份)



切片

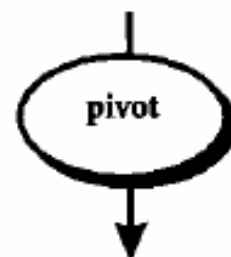


切块



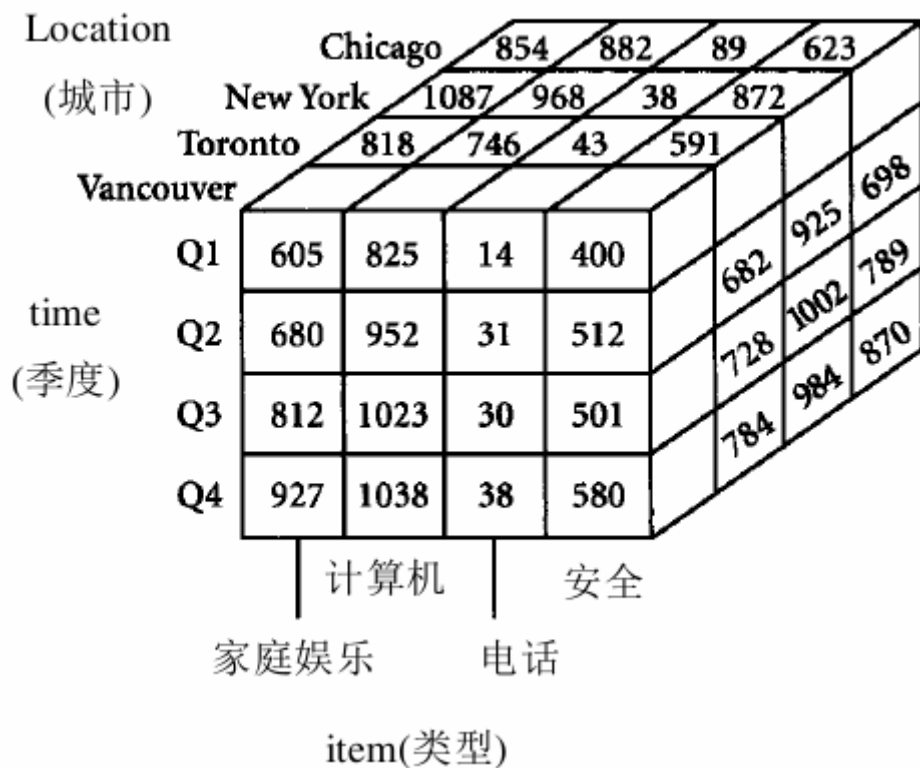
转轴

location (城市)	Chicago				
	New York				
	Toronto				
	Vancouver	605	825	14	440
		computer		security	
		home		phone	
		entertainment			
		item (类型)			

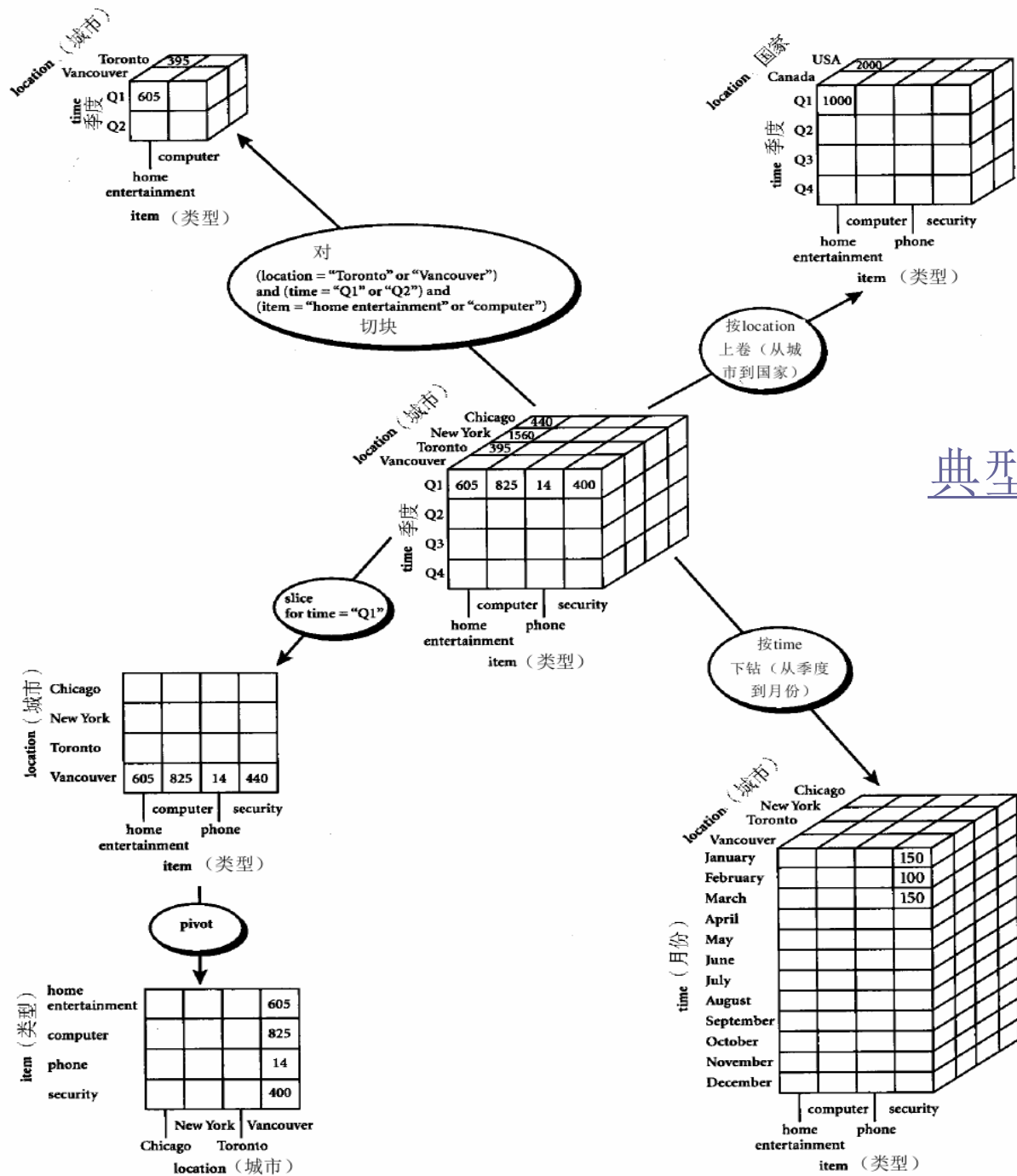


item (类型)	home				605
	entertainment				
	computer				825
	phone				14
	security				400
		New York		Vancouver	
		Chicago		Toronto	
		location (城市)			

转轴

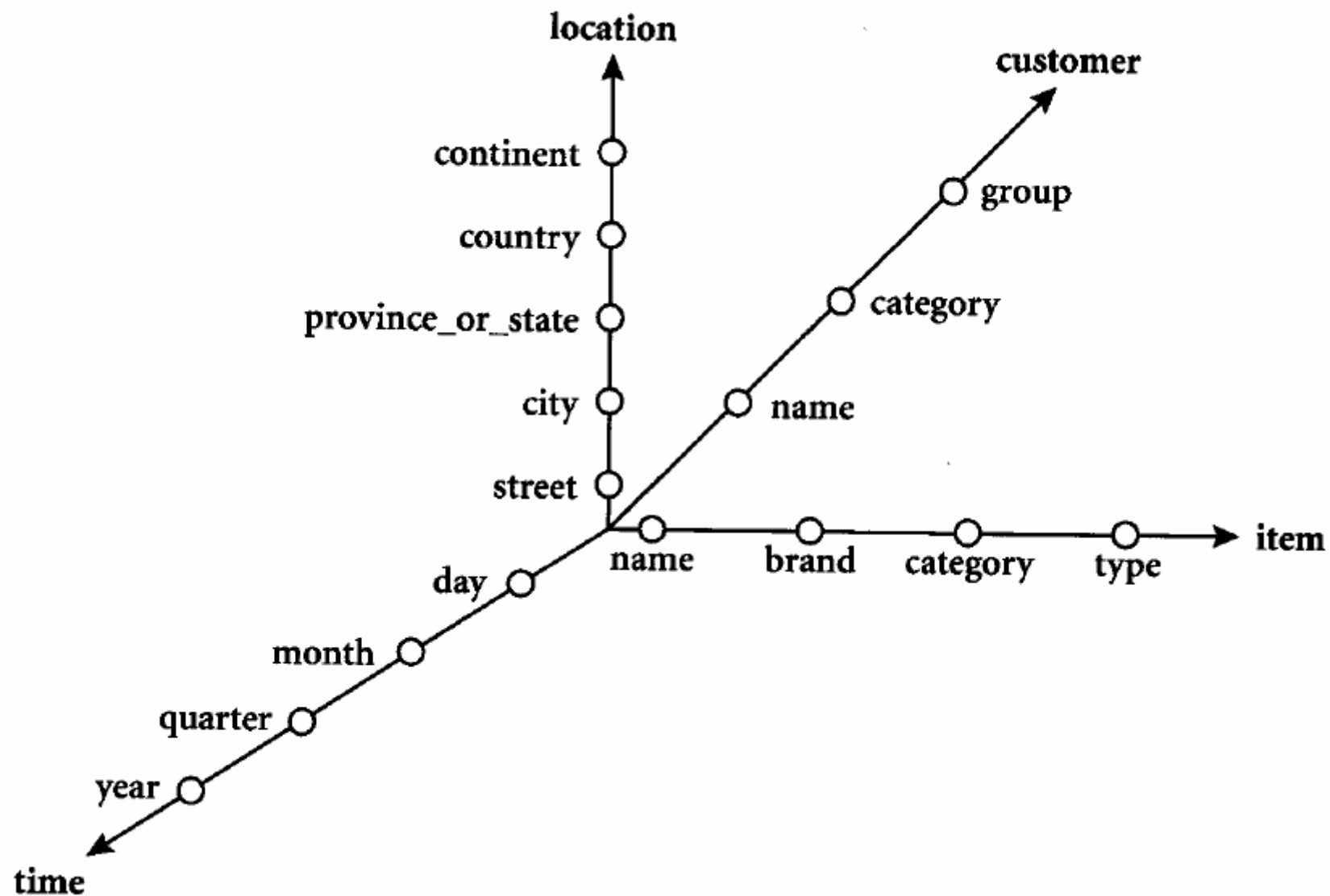


	<i>location= "Chicago"</i> <i>item</i>				<i>location= "New York"</i> <i>item</i>				<i>location= "Toronto"</i> <i>item</i>				<i>location= "Vancouver"</i> <i>item</i>			
<i>time</i>	家庭娱乐	计算机	电话	安全	家庭娱乐	计算机	电话	安全	家庭娱乐	计算机	电话	安全	家庭娱乐	计算机	电话	安全
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

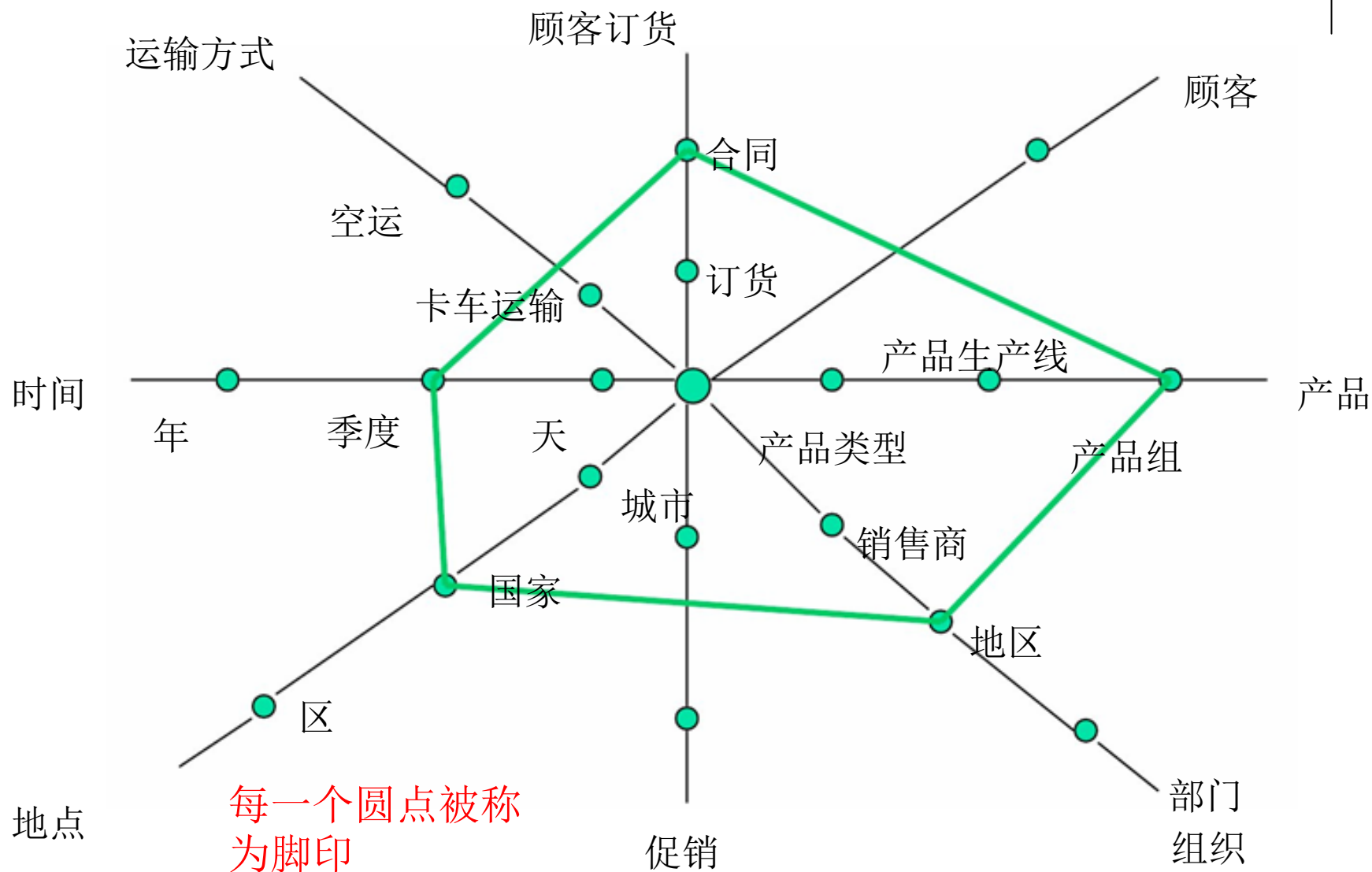


典型的OLAP操作

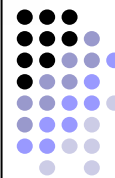
星形网模型



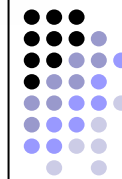
一个星型网的查询模型



大纲



- 什么是数据仓库？
- 多维数据模型
- 数据仓库的系统结构
- 数据仓库实现
- 从数据仓库到数据挖掘



■ 数据仓库的设计：一个商务分析框架

■ 关于数据仓库设计的四种视图

□ 自顶向下视图

- 允许选择数据仓库的所需的相关信息

□ 数据源视图

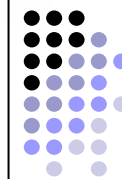
- 揭示被操作数据库系统收集、存储和管理的信息。

□ 数据仓库视图

- 由事实表和维表构成

□ 商务查询视图

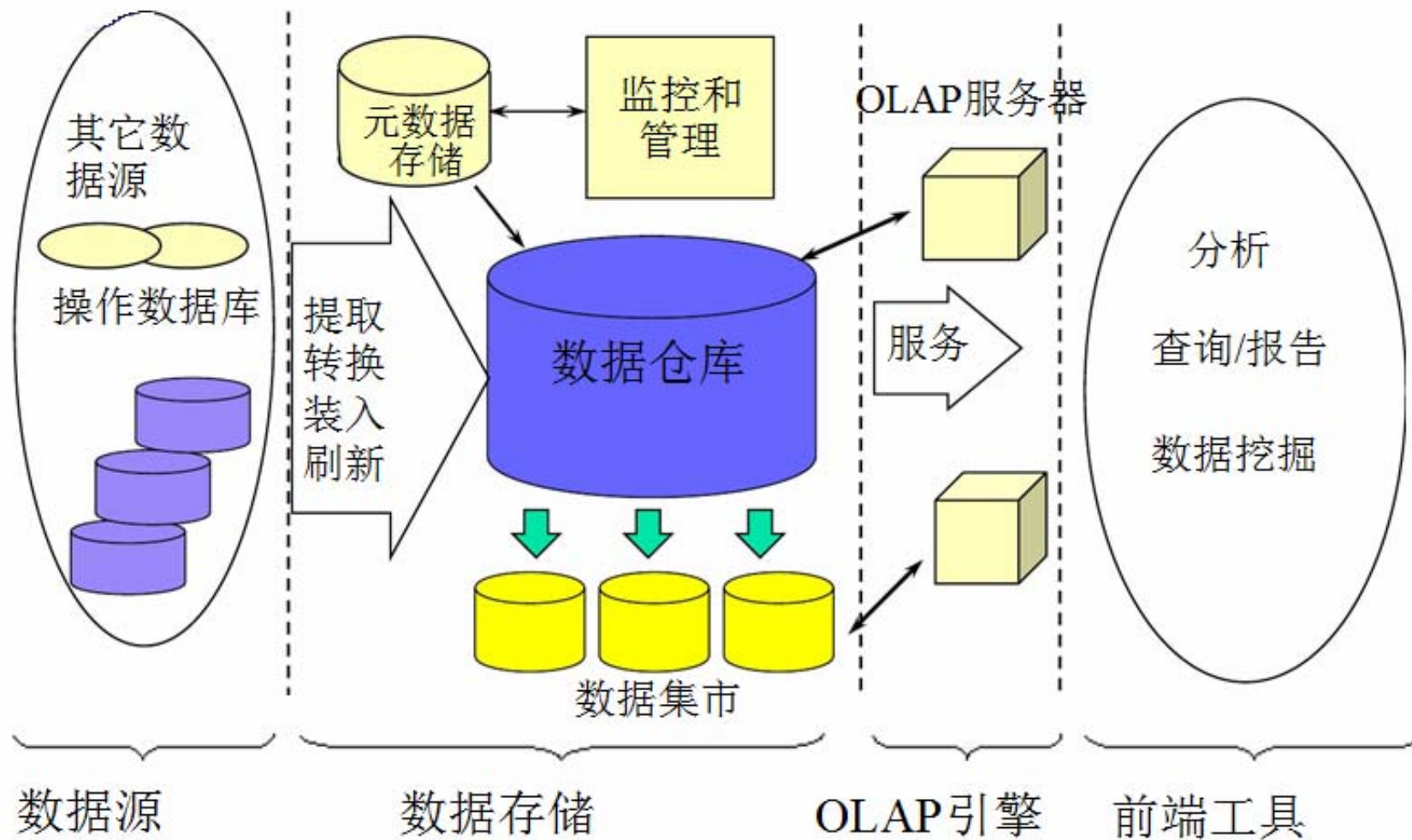
- 从最终用户的角度透视数据仓库的数据

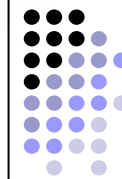


数据仓库的设计过程

- 使用自顶向下方法、自底向上方法或二者结合的混合方法设计。
 - 自顶向下：由总体设计和规划开始（当技术成熟并已掌握，这种方法是有用的）
 - 自底向上方法：以实验和原型开始（在商务建模和技术开发的早期阶段，这种方法是有用的）
- 从软件工程的观点
 - 瀑布式方法：在进行下一步前，每一步都进行结构化和系统的分析
 - 螺旋式方法：涉及功能渐增的系统的快速产生，相继版本的时间间隔很短
- 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，发票等
 - 选取商务处理的**粒度**
 - 选取用于每个事实表记录的**维**
 - 选取安放在事实表中的**度量**

多层体系结构





■ 三种数据仓库模型

■ 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

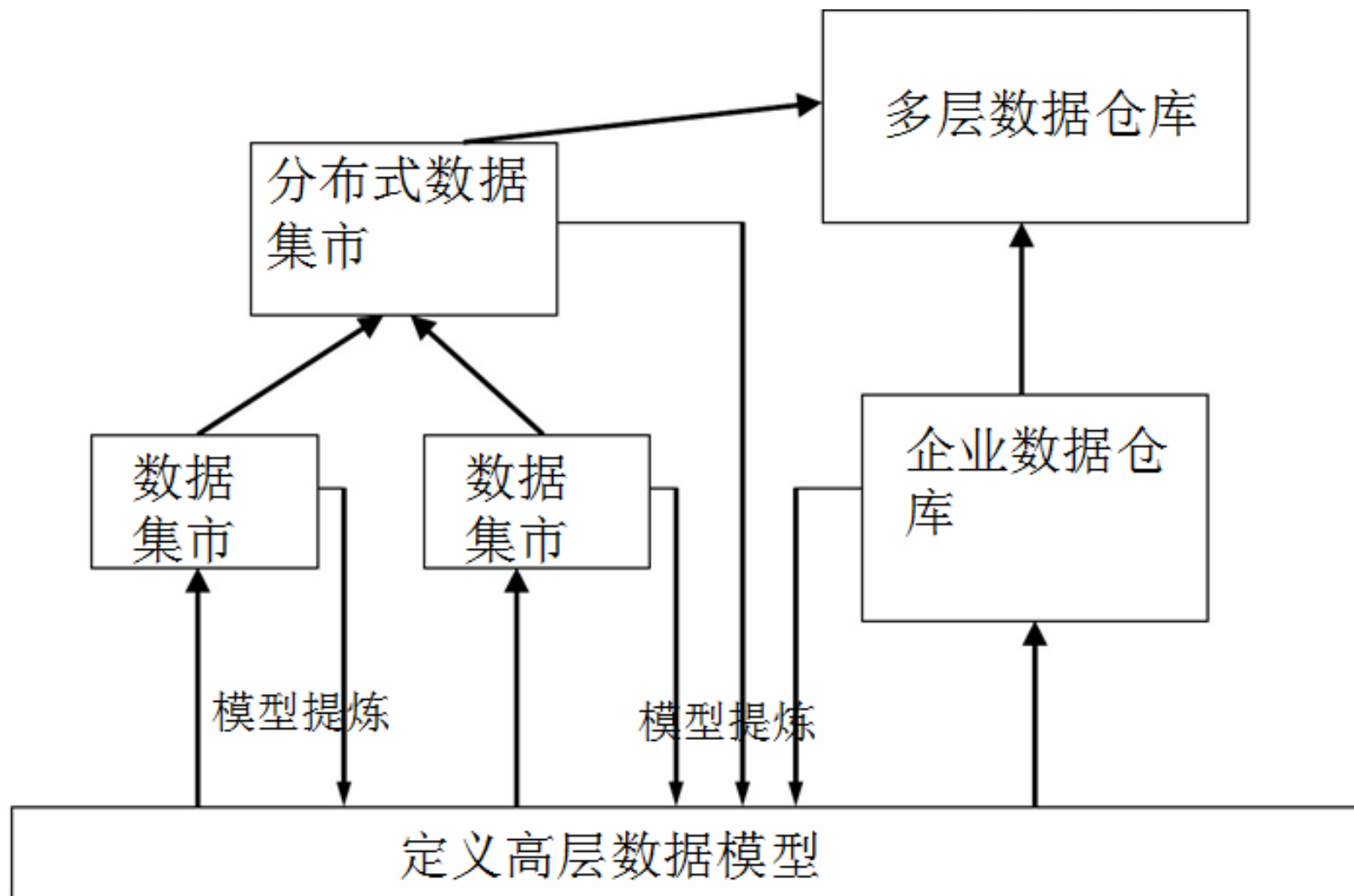
- 包含企业范围数据的一个子集，对于特定的用户是有用的，其范围限于特定的、选定的用户，如：销售数据集市可能限定其主题为顾客、商品和销售。

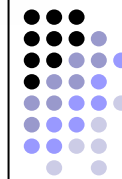
- 独立的数据集市和依赖的数据集市（直接来自数据仓库）

■ 虚拟仓库

- 操作数据库上视图的集合
- 只有一些可能的汇总视图被物化。

数据仓库开发的推荐方法





数据仓库后端工具和使用程序

■ 数据提取

- 通常由多个异构和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

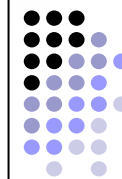
- 将数据由遗产或宿主格式转换成数据仓库格式

■ 装入

- 排序、汇总、合并、计算视图、检查完整性，并建立索引和划分

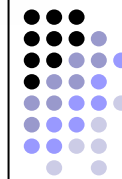
■ 刷新

- 传播由数据源到数据仓库的更新



元数据储存库

- 元数据是定义仓库对象的数据，包括：
- 数据仓库结构的描述
 - 仓库模式、视图、维、分层结构和导出数据定义，以及数据集的位置和内容
- 操作元数据
 - 数据血统（迁移数据的历史和它所使用的变换序列）、数据流通（主动的、档案的或净化的）和监控信息（仓库使用统计量、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
- 商务元数据
 - 商务术语和定义，数据拥有者信息和收费策略



OLAP服务器类型

■ 关系OLAP (ROLAP)

- 使用关系和扩充关系DBMS存放并管理数据仓库，而OLAP中间件支持其余部分。
- 包括每个DBMS后端的优化，聚集导航逻辑的实现，和附加的工具和服务
- 更大的伸缩性

■ 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的快速索引

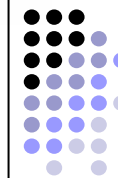
■ 混合OLAP (HOLAP)

- ROLAP的可伸缩性+MOLAP的快速计算

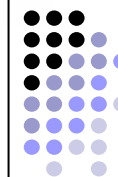
■ 特殊的SQL服务器

- 在星型和雪花模式上支持SQL查询

大纲



- 什么是数据仓库？
- 多维数据模型
- 数据仓库的系统结构
- 数据仓库实现
- 从数据仓库到数据挖掘



数据立方体的有效计算

■ 数据立方体可以被看成是方格体

- 最底层的方体是基本方体
- 最上层方体（顶点方体）只包含一个元



- 那么一个具有L层的n维立方体有多少个方体？

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

- 维灾难

■ 数据立方体的物化

- 物化每一个立方体（全物化），不物化任何“非基本”方体（不物化）或有选择的物化其中一部分（部分物化）
- 选取哪一些立方体进行物化？
 - 根据方体的大小，存取频率等

立方体操作

- DMQL中的方体定义和计算

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

compute cube sales

- 上述的compute cube子句可以转化为一个类似于SQL的语句

SELECT item, city, year, SUM (amount)
FROM SALES

CUBE BY item, city, year

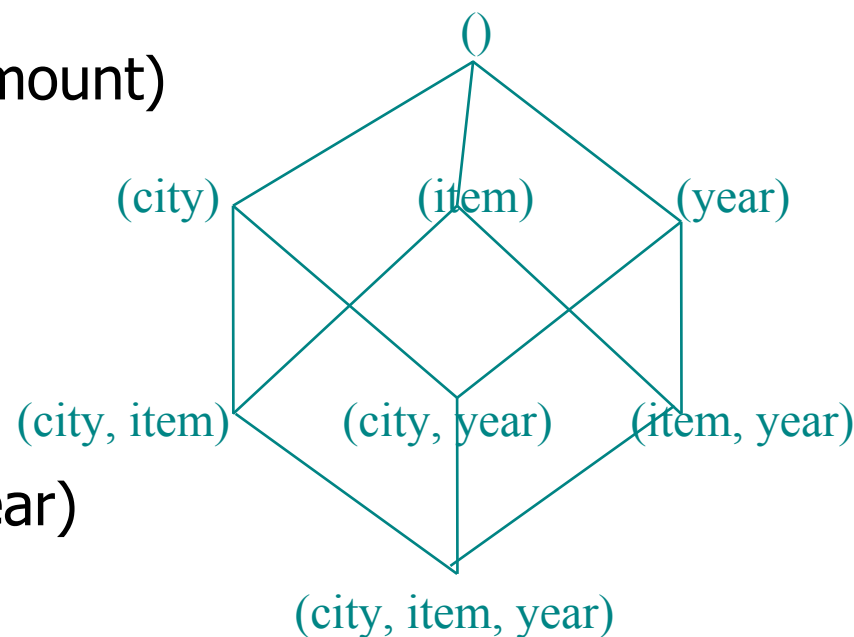
- 需要计算以下的group by子句

(item, city, year)

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

(item), (city), (year)

()



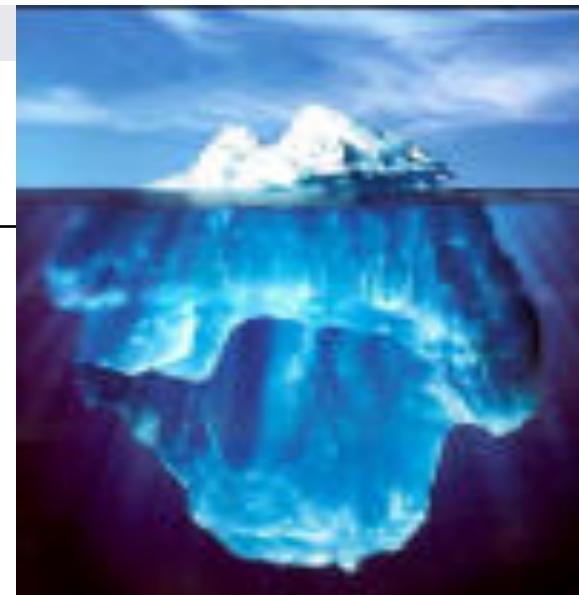
冰山方体Iceberg Cube

- 冰山立方体是一个数据立方体，只存放其聚集值大于某个最小支持度阈值的立方体单元，即满足：

$\text{HAVING COUNT}(*) \geq \text{minsup}$

- 动机

- ☐ 在稀疏方体中，可能只有一小部分的立方体单元“露出水面”
- ☐ 只需要计算那些“有趣”的立方体单元——度量值大于某个最小阈值
- ☐ 避免维灾难



索引 OLAP 数据：位图索引

- 在特定栏上的索引
- 这一栏上的每一个值都对应于一个位向量
- 位向量的长度：基本表中特定栏属性值的个数。
- 如果基本表中的给定行的属性值为 v ，则在为图索引的对应行，表示该值的位为1，该行的其它位均为0
- 对于基数较大的域不大适合

基本表

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

Index on Region

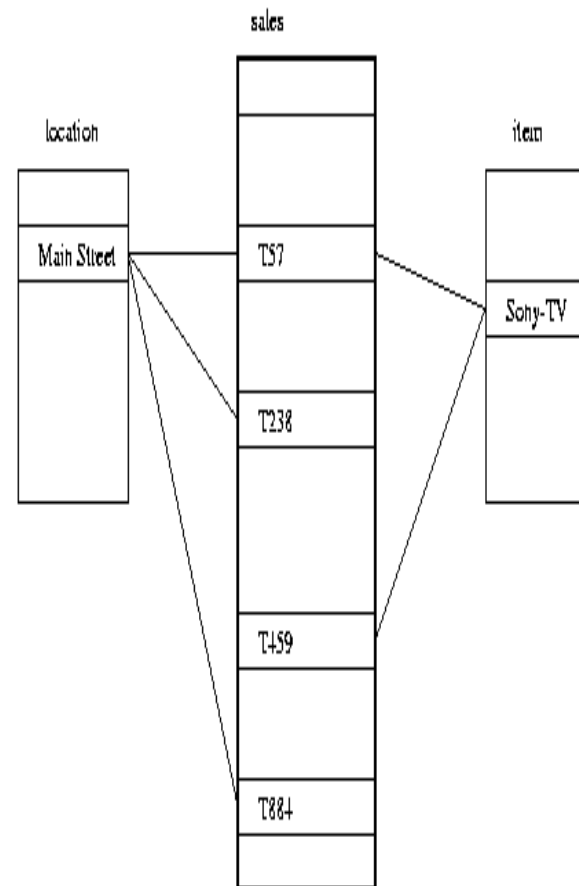
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type

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：如果两个关系 $R(RID, A)$ 和 $S(B, SID)$ 在属性 A 和 B 上连接，则连接索引记录包含 $JI(RID, SID)$ 对，其中 RID 和 SID 分别来自 R 和 S 的记录标识符。
- 传统的索引将给定列上的值映射到具有该值的列表上
 - 它物化了 JI 文件中的关系连接——这是一个相当昂贵的操作
- 在数据仓库中，连接索引把星型模式的维值连接到事实表中的行
 - 例如：对 $sales$ 事实表和 $city, product$ 两个维
 - 城市的连接索引使每一个不同的城市有一个元组的记录标识符列表记录个城市的销售
 - 连接索引可以跨越多维



索引 OLAP数据 :连接索引

location/sales

连接索引表

location	sales_key
...	...
Main Street	T57
Main Street	T238
Main Street	T884
...	...

item/sales

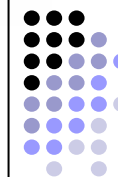
连接索引表

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


location/item/sales

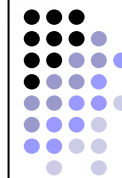
链接两个维的连接索引表

location	item	sales_key
...
Main Street	Sony-TV	T57
...



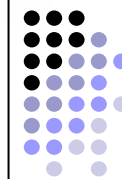
OLAP查询的有效处理

- 确定哪些
 - time: “day < month < quarter < year”
 - 将查询
 - item: “item_name < brand < type”
 - 如, 数
 - location: “street < city < province_or_state < country”
 - 和/或投影操作
- 确定相关操作应当使用哪些物化的方体
 - 假设对{brand, province_or_state}处理查询, 选择常量为“year = 2004”, 还假定有四个物化的方体可用:
 1. 方体1: {year, item_name, city}
 2. 方体2: {year, brand, country} 
 3. 方体3: {year, brand, province_or_state}
 4. 方体4: {item_name, province_or_state}, 其中year = 2004
 - 以上四个方体, 应当选择哪一个处理查询?
- 探索MOLAP中的索引结构和压缩数组结构



元数据存储

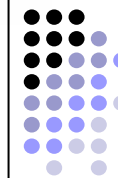
- 元数据是定义仓库对象的数据。它有以下几种类型
 - 数据仓库结构的描述
 - 包括仓库模式、视图、维、层次结构和导出的数据定义，以及数据集的位置和内容
 - 操作元数据
 - 包括数据血统（移植数据的历史和用于它的转换序列），数据流通（主动地、档案的或净化的），以及监视信息（仓库使用统计，错误报告，审计跟踪）
 - 汇总用的算法
 - 有操作环境到数据仓库的映射
 - 关于系统性能的数据
 - 数据仓库模式，视图和导出的数据定义
 - 商务元数据
 - 商务术语和定义，数据拥有者信息和收费策略



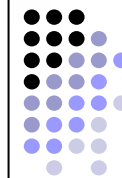
■ 数据仓库后端工具和实用程序

- 数据提取：
 - 从多个异种的外部数据源收集数据
- 数据清理：
 - 检测数据中的错误，可能时更正它们。
- 数据变换：
 - 将数据由遗产或宿主格式转换成数据仓库格式
- 装入：
 - 排序、综合、合并、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

大纲



- 什么是数据仓库？
- 多维数据模型
- 数据仓库的系统结构
- 数据仓库实现
- 从数据仓库到数据挖掘



■ 数据仓库的使用

■ 三种数据仓库应用

□ 信息处理

- 支持查询和基本的统计分析，并使用交叉表、表、图表或图进行报告

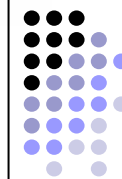
□ 分析处理

- 数据仓库数据的多维分析
- 支持基本的OLAP 操作，包括切片、下钻、上卷和转轴

□ 数据挖掘

- 隐含模式中的知识发现
- 支持关联模式，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

■ 三种应用的不同点



■ 从联机分析处理 (OLAP) 到联机分析挖掘 (OLAM)

■ 为什么进行联机分析挖掘？

□ 数据仓库中数据的高质量

- 数据仓库包含集成的、一致的和清理过的数据

□ 环绕数据仓库的有价值的信息处理基础设施

- ODBC/ OLE DB连接, Web 访问, 服务工具以及报表和OLAP分析工具

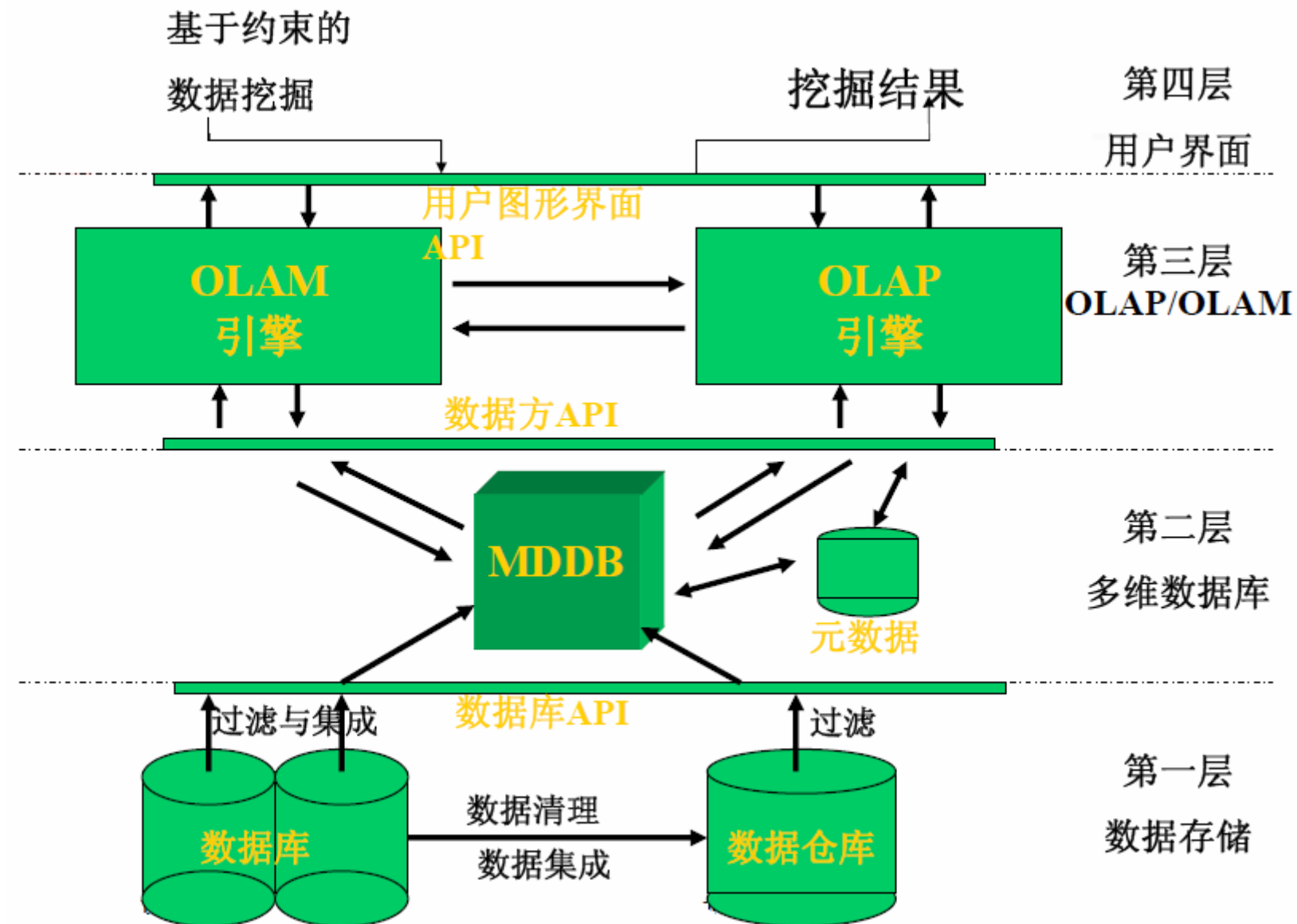
□ 基于OLAP的探测式数据分析

- 用下钻, 切片, 旋转等进行挖掘

□ 数据挖掘功能的联机分析选择

- 将多种数据挖掘功能、算法和任务聚集起来

一个OLAM结构



小结

■ 数据仓库

- 数据仓库是面向主题的、集成的、时变的和非易失的有组织的数据集合，支持管理的决策过程。

■ 数据仓库的多维数据模型

- 星型模式、雪花模式和事实星座模式
- 一个数据立方体有大量事实（或度量）和许多维组成

■ OLAP 操作：上卷、下钻、切片、切块和转轴

■ OLAP 服务器：关系OLAP(ROLAP), 多维OLAP(MOLAP), 混合OLAP(HOLAP)

■ 数据立方体的有效计算

- 部分物化、全物化和不物化
- 多路数组聚集
- 位图索引和连接索引

■ 从联机分析处理（ OLAP） 到联机分析挖掘 （OLAM）