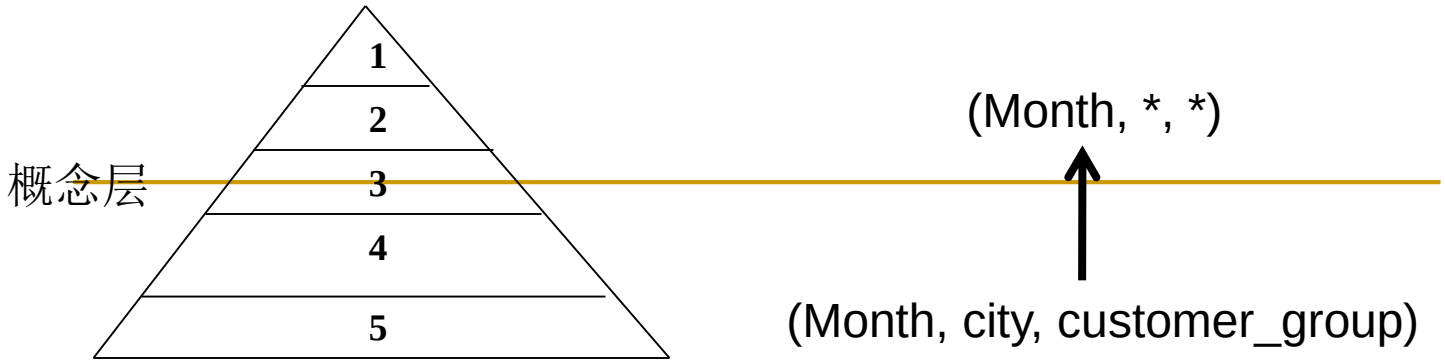
数据立方体计算与数据泛化

数据泛化

- 数据泛化
  - 数据库中的数据和对象通常包含原始概念层的细节信息，数据泛化就是将数据库中的跟任务相关的大型数据集从相对较低的概念层抽象到较高的概念层的过程。



- 主要方法：
  - 数据立方体（OLAP使用的方法）
  - 面向属性的归纳方法

两种不同类别的数据挖掘

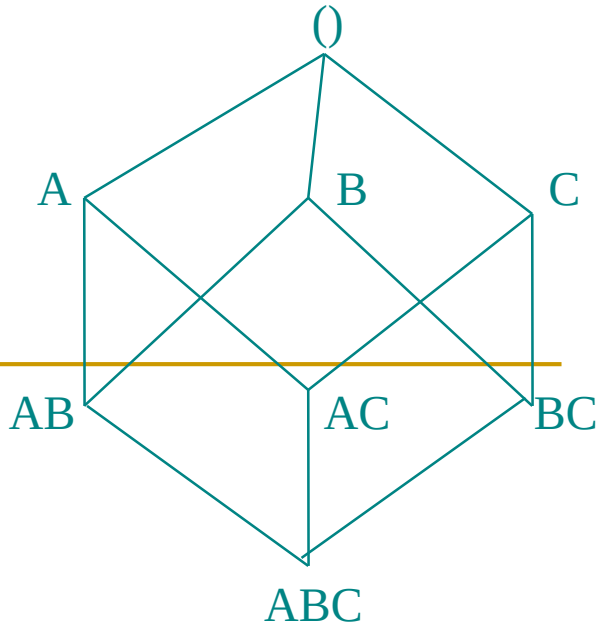
- 从数据分析的角度看，数据挖掘可以分为描述性挖掘和预测性挖掘
  - 描述性挖掘：以简洁概要的方式描述数据，并提供数据的有趣的一般性质。
    - E.g. 数据泛化就是一种描述性数据挖掘
  - 预测性数据挖掘：通过分析数据建立一个或一组模型，并试图预测新数据集的行为。
    - E.g 分类、回归分析等

数据立方体的物化

- 数据立方体有利于多维数据的联机分析处理
  - 数据立方体使得从不同的角度对数据进行观察成为可能
- 方体计算（物化）的挑战：海量数据，有限的内存和时间
  - 海量数据运算对大量计算时间和存储空间的要求

数据立方体---基本概念(1)

- 数据立方体可以被看成是一个方体的格，每个方体用一个group-by表示
- 最底层的方体ABC是基本方体，包含所有3个维
- 最顶端的方体（顶点）只包含一个单元的值，泛化程度最高
- 上卷和下钻操作与数据立方体的对应



P102 图4-1

数据立方体---基本概念(2)

- 基本方体的单元是**基本单元**，非基本方体的单元是**聚集单元**
  - 聚集单元在一个或多个维聚集，每个聚集维用"*"表示
  - E.g. (city, *, year, measure)
  - m维方体：$(a_1,a_2,\ldots,a_n)$中有m个不是"*"
- 祖先和子孙单元
  - i-D单元a=$(a_1,a_2,\ldots,a_n, measures_a)$是j-D单元b=$(b_1,b_2,\ldots,b_n, measure_b)$的祖先，当且仅当
    - (1)$i<j$，并且
    - (2)对于$1 \le m \le n$，只要$a_m \neq$ "*"就有$a_m=b_m$

冰山立方体 (1)

- 为了确保快速的联机分析，有时希望预计算整个立方体（所有方体的所有单元）
  - n维数据立方体包含2^n个方体
  - 如果考虑概念分层 $$T = \prod_{i=1}^{n} (L_i + 1)$$
- 部分物化是存储空间和响应时间的折中方案
  - 事实上，很多高维方体都是稀疏的（包含很多度量值为0的单元）

冰山立方体 (2)

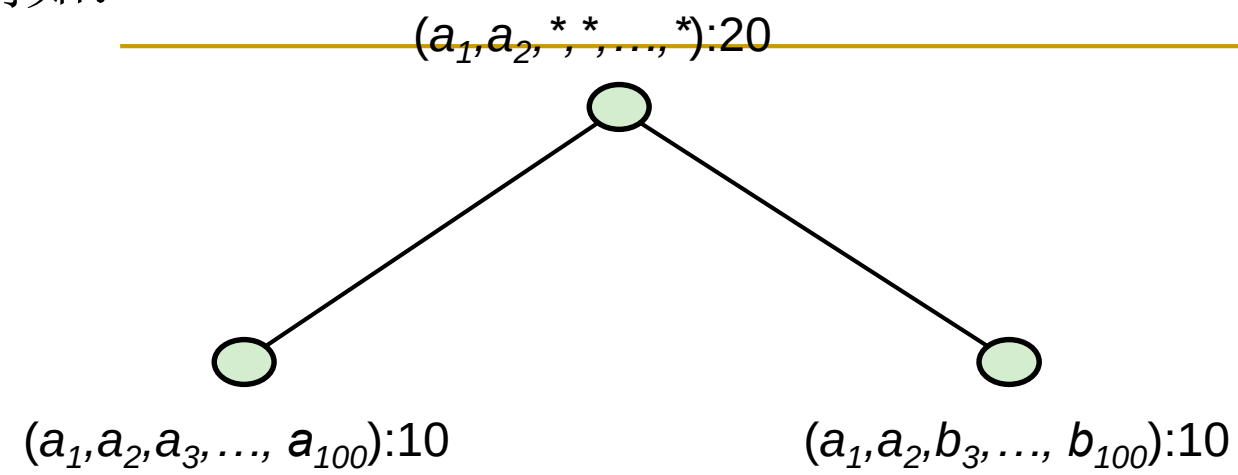
- 对于稀疏的数据立方体，我们往往通过指定一个最小支持度阈值（也称冰山条件），来进行部分物化，这种部分物化的方体称之为冰山方体。比如：
- COMPUTE CUBE Sales_Iceberg AS
- SELECT month, city, cust_grp, COUNT(*)
- FROM Sales_Info
- CUBE BY month, city, cust_grp
- HAVING COUNT(*) >= min_sup

闭立方体 (1)

- 冰山方体的计算通过冰山条件（例：HAVING COUNT(*) >= min_sup）来减轻计算数据立方体中不重要的聚集单元的负担，然而仍有大量不感兴趣的单元需要计算
  - 比如：最小支持度为10，假定100维的数据立方体有两个基本方体：{$(a_1,a_2,a_3,\ldots,a_{100})$:10, $(a_1,a_2,b_3,\ldots,b_{100})$:10}，假设冰山条件为最小支持度10
    - 则需计算和存储的单元仍是海量：2^{101}-6个
    - 如：$(a_1,a_2,a_3,\ldots,a_{99},*)$:10, $(a_1,*,a_3,\ldots,a_{100})$:10

闭立方体 (2)

- 闭单元
  - 一个单元c是闭单元，如果单元c不存在一个跟c有着相同度量值的后代d
  - 例如：上述例子中，任何一个$(a_1,a_2,a_3,*,*,\ldots,*)$:10,都和他的后代有相同度量值
- 闭立方体：一个仅有闭单元组成的数据立方体
  - 例如：



立方体外壳

- 部分物化的另外一种策略：仅预计算涉及少数维的方体（比如3到5维），这些立方体形成对应数据立方体的外壳
  - 利用外壳对其他的维组合查询进行快速计算
  - 仍将导致大量方体（n很大时），类似的我们可以利用方体的兴趣度，选择只预计算立方体外壳的部分

立方体计算的一般策略 (1)

- 一般，有两种基本结构用于存储方体
  - 关系OLAP（ROLAP）
    - 底层使用关系模型存储数据
  - 多维OLAP（MOLAP）
    - 底层使用多维数组存储数据
- 无论使用哪种存储方法，都可以使用以下立方体计算的一般优化技术
  - 优化技术1：排序、散列和分组
    - 将排序、散列(hashing)和分组操作应用于维的属性，以便对相关元组重新排序和聚类

立方体计算的一般策略 (2)

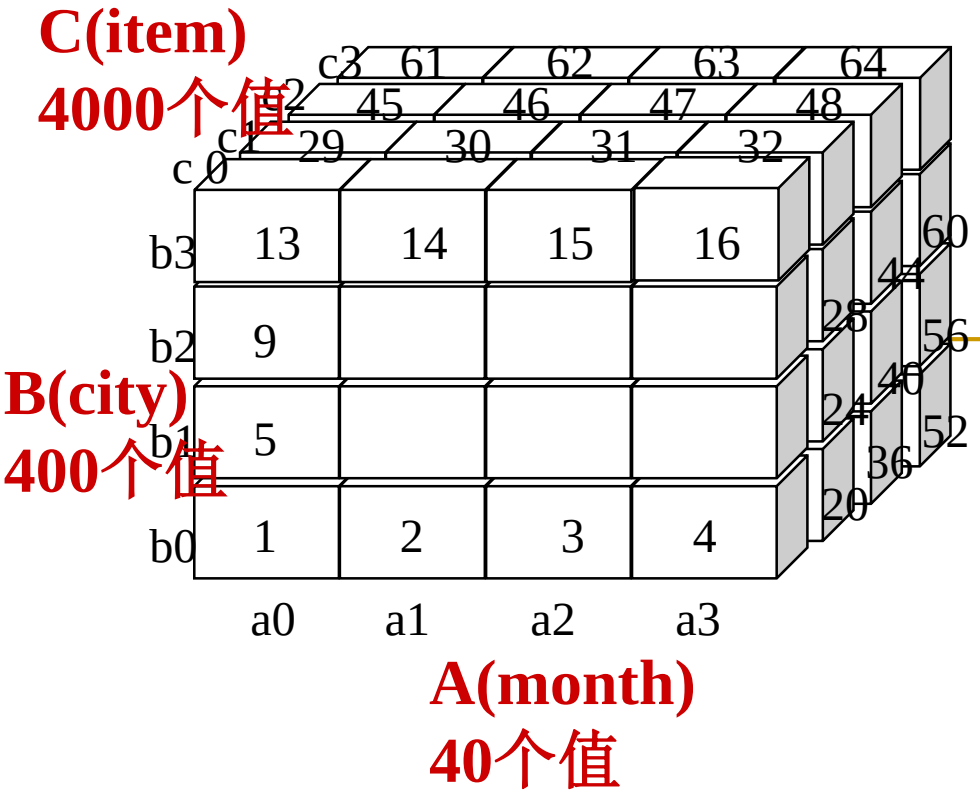
- 优化技术2：同时聚集和缓存中间结果
  - 由先前计算的较低层聚集来计算较高层聚集，而非从基本方体开始计算，减少I/O
- 优化方法3：当存在多个子女时，由最小的子女聚集
  - 例如，计算C_{branch}，可以利用$C_{(branch, year)}$或者$C_{(branch, item)}$，显然利用前者更有效
- 优化技术4：可以使用*Apriori*剪枝方法有效的计算冰山方体
  - 如果给定的单元不能满足最小支持度，则该单元的后代也都不满足最小支持度

完全立方体计算的多路数组聚集方法(1)

- 使用多维数组作为基本数据结构，计算完全数据立方体
  - 一种使用数组直接寻址的典型MOLAP方法
- 计算步骤
  - （1）将数组分成块（chunk,一个可以装入内存的小子方）
    - 块还可以进一步被压缩，以避免空数组单元导致的空间浪费（处理稀疏立方体）
  - （2）通过访问立方体单元，计算聚集。
    - 可以优化访问单元组的次序，使得每个单元被访问的次数最小化，从而减少内存访问和磁盘I/O的开销。

完全立方体计算的多路数组聚集方法(2)

一个包含A,B,C的3-D数组，假定维A,B,C的基数分别是40、400和4000

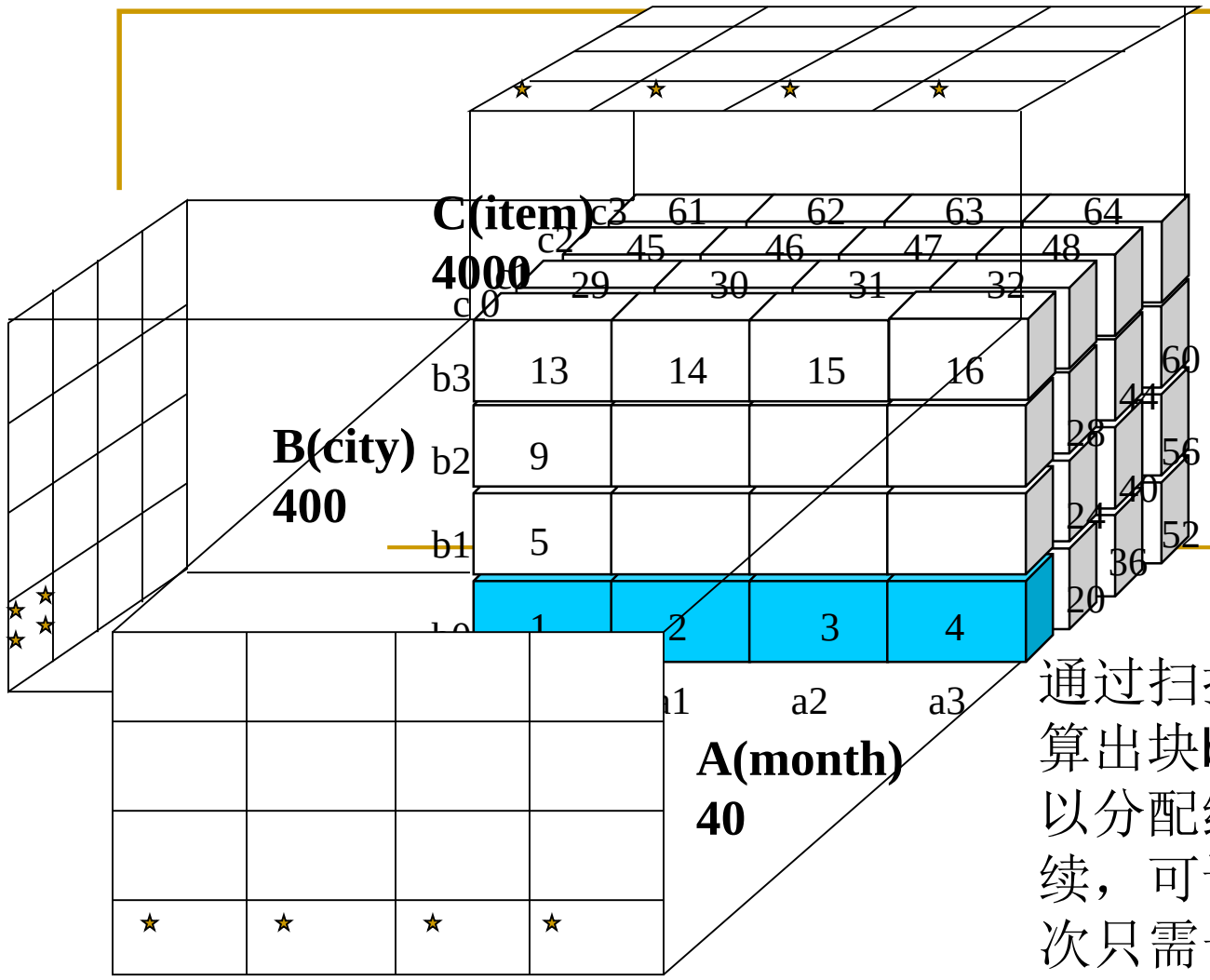


将要物化的立方体：

- 基本方体ABC，已计算，对应于给定的3-D数组
- 2D方体AB，AC和BC
- 1D方体A,B,C
- 0D顶点方体，记作all

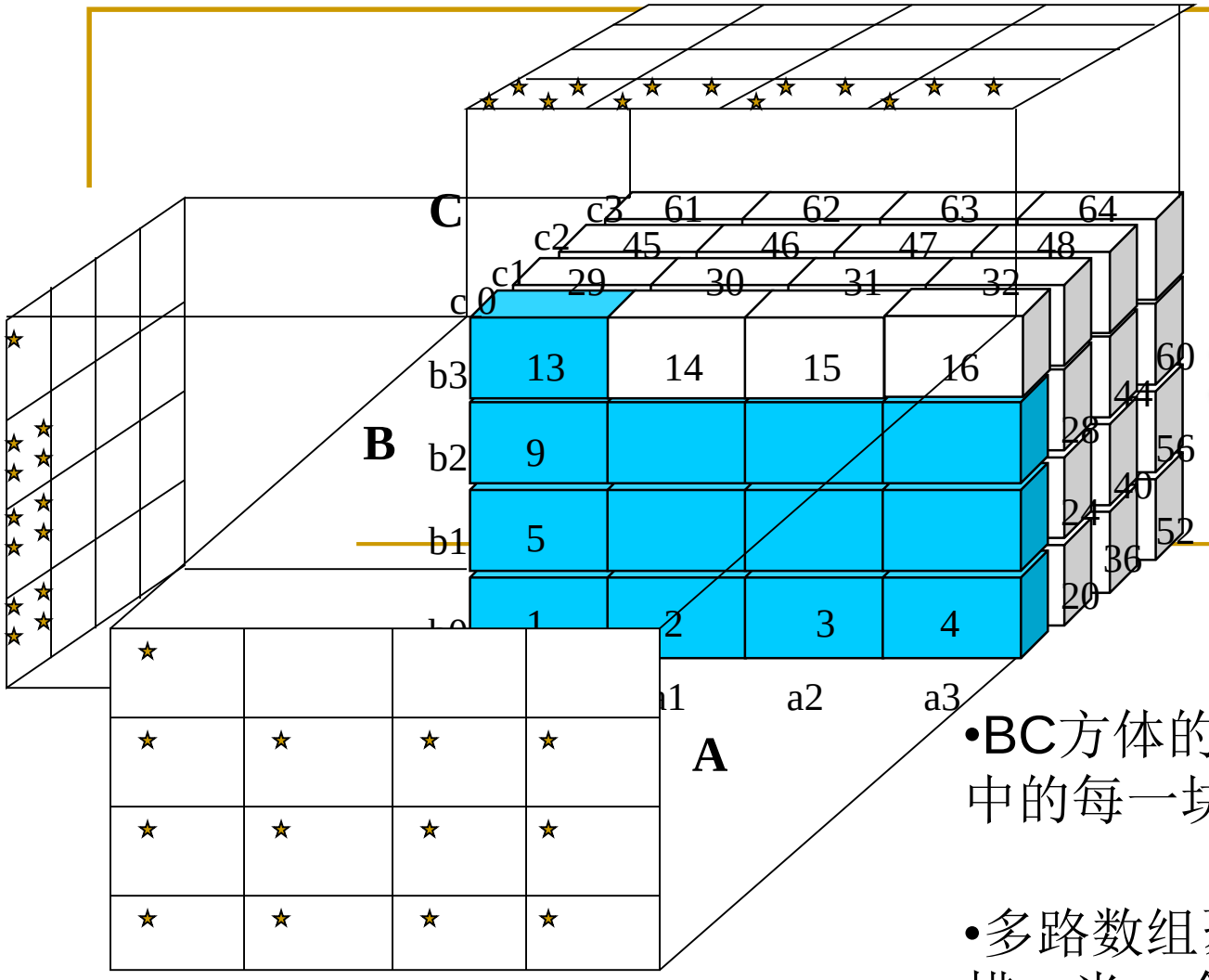
哪个是多路数组聚集的最佳遍历次序？

完全立方体计算的多路数组聚集方法(3)



通过扫描ABC的1～4块，计算出块b0c0，然后块内存可以分配给下一刻b1c0,如此继续，可计算整个BC方体（一次只需一个BC块在内存）

完全立方体计算的多路数组聚集方法(4)



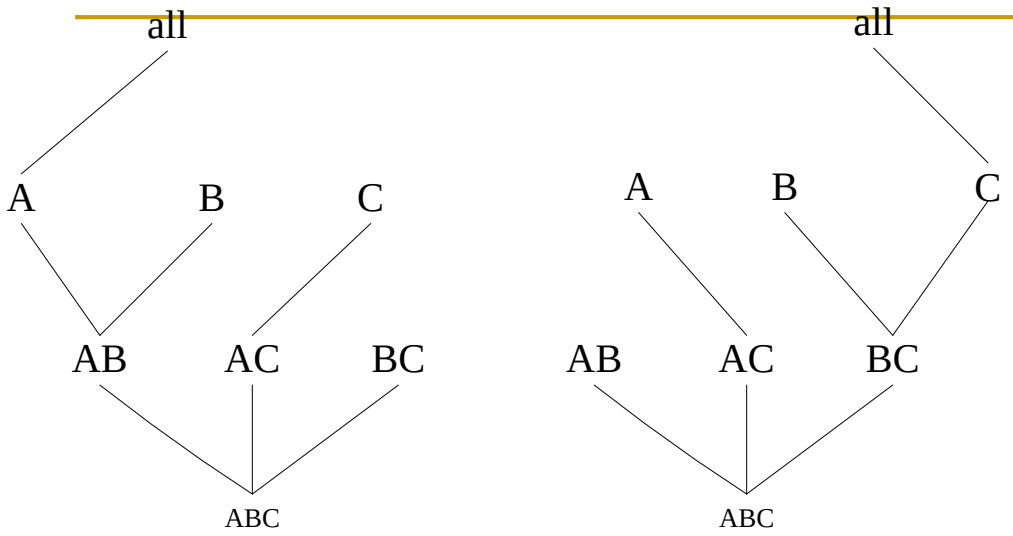
思考：计算时需要多少内存？

- BC方体的计算，必须扫描64块中的每一块；计算其他块亦然

- 多路数组聚集方法避免重复扫描：当一个3D块在内存时，向每一个平面同时聚集

完全立方体计算的多路数组聚集方法(5)

- 方法：各平面要按他们大小的升序排列进行排序和计算
  - 详见书P108例4-4
  - 思想：将最小的平面放在内存中，对最大的平面每次只是取并计算一块



内存空间需求最小的块计算次序　　内存空间需求最大的块计算次序

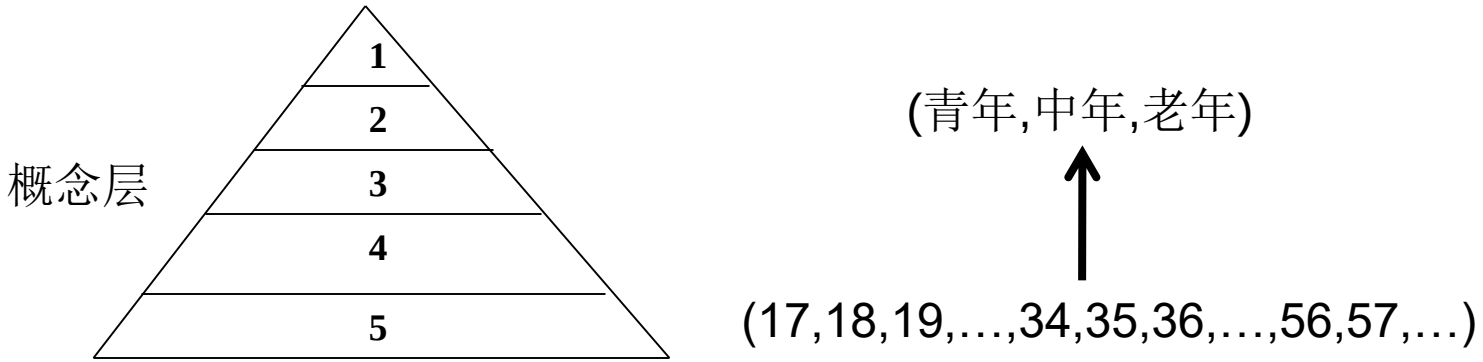
完全立方体计算的多路数组聚集方法(6)

- 根据1到64的扫描次序，在块内存中保存所有相关的2-D平面所需的最小存储为：
  - 40×400（用于整个AB平面）＋40×1000（用于AC平面一行）＋100×1000（用于BC平面一块)＝156，000
- 这种方法的限制：只有在维数比较小的情况下，效果才比较理想(要计算的立方体随维数指数增长)
  - 如果维的数目比较多，可以考虑使用“自底向上的计算”或者时“冰山方体” 计算

数据立方体计算与数据泛化（2）

数据泛化

- 数据泛化
  - 通过将相对层次较低的值（如属性age的数值）用较高层次的概念（如青年、中年、老年）置换来汇总数据



- 主要方法：
  - 数据立方体（OLAP使用的方法）
  - 面向属性的归纳方法

什么是概念描述？

- 概念描述是一种数据泛化的形式。
  - 概念通常指数据的汇集
    - 如frequent buyers，graduate students
- 概念描述产生数据的特征化和比较描述，当所描述的概念所指的是对象类时，也称为**类描述**
  - **特征化**：提供给定数据汇集的简洁汇总
  - **比较**：提供两个或多个数据集的比较描述

概念描述 VS. OLAP

- 相似处：
  - 数据泛化
  - 对数据的汇总在不同的抽象级别上进行呈现
- 区别：
  - 复杂的数据类型和聚集
    - OLAP中维和度量的数据类型都非常有限（非数值型的维和数值型的数据），表现为一种简单的数据分析模型
    - 概念描述可以处理复杂数据类型的属性及其聚集
  - 用户控制与自动处理
    - OLAP是一个由用户控制的过程
    - 概念描述则表现为一个更加自动化的过程

数据特征化的面向属性的归纳

- 一种面向**关系数据**查询的、基于**汇总**的**在线**数据分析技术。
  - 受数据类型和度量类型的约束比较少
- 面向属性归纳的基本思想：
  - 使用关系数据库查询收集任务相关的数据
  - 通过考察任务相关数据中每个属性的不同值的个数进行泛化，方法是属性删除或者是属性泛化
  - 通过合并相等的，泛化的广义元组，并累计他们对应的计数值进行聚集操作
  - 通过与用户交互，将广义关系以图表或规则等形式，提交给用户

数据聚焦（1）

- 目的是获得跟任务相关的数据集，包括属性或维，在DMQL中他们由in relevance to子句表示。
- 示例：
  - DMQL: 描述Big-University数据库中*研究生*的一般特征

```
use Big_University_DB
mine characteristics as "Science_Students"
in relevance to name, gender, major,
   birth_place, birth_date, residence, phone#,
   gpa
from student
where  status in "graduate"
```

数据聚焦（2）

- 上述DMQL查询转换为如下SQL查询，收集任务相关数据集

```
Select name, gender, major, birth_place, birth_date, residence,
    phone#, gpa
from student
where  status in {"Msc", "M.A.", "MBA", "PhD"}
```

- 初始工作关系

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…

数据泛化

- 数据泛化的两种常用方法：属性删除和属性泛化
  - 属性删除的适用规则：对初始工作关系中具有大量不同值的属性，符合以下情况，应使用属性删除：
    - 在此属性上没有泛化操作符（比如该属性没有定义相关的概念分层）
    - 该属性的较高层概念用其他属性表示
  - 属性泛化的使用规则：如果初始工作关系中的某个属性具有大量不同值，且该属性上存在泛化操作符，则使用该泛化操作符对该属性进行数据泛化操作

属性泛化控制

- 确定什么是“具有大量的不同值”，控制将属性泛化到多高的抽象层。
- 属性泛化控制的两种常用方法：
  - 属性泛化阈值控制
    - 对所有属性设置一个泛化阈值或者是对每个属性都设置一个阈值（一般为2到8）
  - 泛化关系阈值控制
    - 为泛化关系设置一个阈值，确定泛化关系中，不同元组的个数的最大值。（通常为10到30，允许在实际应用中进行调整）
  - 两种技术的顺序使用：使用属性泛化阈值控制来泛化每个属性，然后使用关系阈值控制进一步压缩泛化的关系

归纳过程中的聚集值计算

- 在归纳过程中，需要在不同的抽象层得到数据的量化信息或统计信息
- 聚集值计算过程
  - 聚集函数count与每个数据库元组相关联，
    - 初始工作关系的每个元组的值初始化为1
  - 通过属性删除和属性泛化，初始工作关系中的元组可能被泛化，导致相等的元组分组
    - 新的"相等的元组分组"的计数值设为初始工作关系中相应元组的计数和
      - e.g. 52个初始工作关系中的元组泛化为一个新的元组T，则T的计数设置为52
    - 还可以应用其他聚集函数，包括sum，avg等

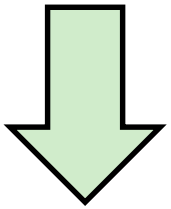
面向属性的归纳——示例

- 挖掘BigUniversity数据库中研究生的一般特征
  - name：删除属性（大量不同值，无泛化操作符）
  - gender：保留该属性，不泛化
  - major：根据概念分层向上攀升{文，理，工…}
  - birth_place：根据概念分层location向上攀升
  - birth_date：泛化为age，再泛化为age_range
  - residence：根据概念分层location向上攀升
  - phone#：删除属性
  - gpa：根据GPA的分级作为概念分层

面向属性的归纳——示例

初始工作关系

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…



主泛化关系

Gender	Major	Birth_region	Age_range	Residence	GPA	Count
M	Science	Canada	20-25	Richmond	Very-good	16
F	Science	Foreign	25-30	Burnaby	Excellent	22
…	…	…	…	…	…	…

面向属性的归纳算法

- 输入
  - 1. DB; 2. 数据挖掘查询DMQuery; 3. 属性列表; 4. 属性的概念分层; 5. 属性的泛化阈值；
- 输出
  - 主泛化关系P
- 算法描述：

1. W←get_task_relevant_data(DMQuery, DB)
2. prepare_for_generalization(W)
   1. 扫描W，收集每个属性a的不同值
   2. 对每个属性a，根据阈值确定是否删除，如果不删除，则计算其最小期望层次L，并确定映射对(v,v`)
3. P←generalization(W)
   - 通过使用v`代替W中每个v，累计计数并计算所有聚集值，导出P
   1. 每个泛化元组的插入或累积计数
   2. 用数组表示P

导出泛化的表示 (1)

- 泛化关系
  - 一部分或者所有属性得到泛化的关系，包含计数或其他度量值的聚集
- 交叉表
  - 二维交叉表使用每行显示一个属性，使用每列显示另外一个属性将结果集映射到表中
- 可视化工具：
  - 条形图、饼图、曲线和数据立方体浏览工具（用单元的大小代表计数，用单元亮度代表另外的度量）

导出泛化的表示 (2)

- 量化规则
  - 使用t_weight表示主泛化关系中每个元组的典型性

$$t\_weight = count(q_a) / \sum_{i=1}^{n} count(q_i)$$

- 量化特征规则
  - 将泛化的结果映射到相应的量化特征规则中，比如：

$$\forall X, target\_class(X) \Rightarrow condition_l(X)[t: w_l] \vee ... \vee condition_m(X)[t: w_m]$$

量化特征规则中每个析取代表一个条件，一般，这些条件的析取形成目标类的必要条件，因为该条件是根据目标类的所有情况导出的。也就是说，目标类的所有元组必须满足该条件。然而，该规则可能不是目标类的充分条件，因为满足同一条件的元组可能属于其他类。

E.g.

$$\forall X, item(X) = "computer" \Rightarrow (location(X) = "Asia")[t: 25\%] \vee (location(X) = "Europe")[t: 30\%] \vee (location(X) = "North\ American")[t: 45\%]$$

挖掘类比较：区分不同的类

- 类比较挖掘的目标是得到将目标类与对比类相区分的描述。
  - 目标类和对比类间必须具有可比性，即两者间要有相似的属性或维。
    - 本科生 VS. 研究生；student VS. address
- 很多应用于类特征化的技巧（处理单个类的多层数据的汇总和特征化）可以应用于类比较，比如属性泛化
  - 属性泛化必须在所有比较类上同步进行，将属性泛化到同一抽象层后进行比较。
    - E.g. City VS country

类比较的过程

- 数据收集
  - 通过查询处理收集数据库中相关的数据，并将其划分为一个目标类和一个或多个对比类
- 维相关分析
  - 如果存在较多的维，则应当对这些类进行维相关分析，仅选择高度相关的维进行进一步分析。（可以使用基于熵的度量）
- 同步泛化
  - 同步的在目标类和对比类上进行泛化，泛化到维阈值控制的层，得到*主目标类 关系l方体* 和 *主对比类 关系l方体*
- 导出比较的表示
  - 用可视化技术表达类比较描述，通常会包含“对比”度量，反映目标类与对比类间的比较 (e.g count%)

类比较挖掘——示例（1）

- 任务
  - 挖掘描述BigUniversity本科生和研究生的类比较
- 任务的DMQL描述

use Big_University_DB
mine comparison as “grad_vs_undergrad_students”
in relevance to *name, gender, major, birth_place, birth_date, residence, phone#, gpa*
for “graduate_students”
where status in “graduate”
versus “undergraduate_students”
where status in “undergraduate”
analyze count%
from student

类比较挖掘——示例（2）

- 进行类比较挖掘的输入：
  - 给定的属性：*name, gender, major, birth_place, birth_date, residence, phone# and gpa*
  - 在属性a_i上定义的概念分层 $Gen(a_i)$
  - 在属性a_i上定义的属性分析阈值 U_i
  - 在属性a_i上定义的属性泛化阈值T_i
  - 属性相关性阈值R

类比较挖掘——示例（3）

- 任务的处理过程
  - 数据收集
    - DMQL查询转化为关系查询，得到*初始目标类工作关系*和*初始对比类工作关系*
    - 可以看成使构造数据立方体的过程
      - 引入一个新维status来标志目标类和对比类（graduate, undergraduate）
      - 其他属性形成剩余的维
  - 在两个数据类上进行维相关分析
    - 删除不相关或者使弱相关的维：*name, gender, major, phone#*

类比较挖掘——示例（4）

- 同步泛化
  - 在目标类和对比类上同步的进行泛化，将相关的维泛化到由维阈值控制的层，形成*主目标类 关系/方体* 和*主对比类 关系/方体*
- 导出比较的表示
  - 用表、图或规则等形式表达类比较描述的挖掘结果
  - 用户应该能够在*主目标类 关系/方体* 和*主对比类 关系/方体*进行进一步的OLAP操作

类比较挖掘——示例（5）

Major	Age_range	Gpa	Count%
Science	20-25	Good	5.53%
Science	25-30	Good	2.32%
Science	Over_30	Very_good	5.86%
…	…	…	…
Business	Over_30	Excellent	4.68%

目标类的主泛化关系: 研究生

Major	Age_range	Gpa	Count%
Science	15-20	Fair	5.53%
Science	15-20	Good	4.53%
…	…	…	…
Science	25-30	Good	5.02%
…	…	…	…
Business	Over_30	Excellent	0.68%

对比类的主泛化关系: 本科生

类比较描述的量化判别规则表示（1）

- 类比较描述中的目标类和对比类的区分特性也可以用量化规则来表示，即量化判别规则
  - 量化判别规则使用d-weight作为兴趣度度量 （

$$d-weight=\frac{count(q_a\in C_j)}{\sum_{i=1}^{m}count(q_a\in C_i)}$$

    - q_a—概化元组
    - C_j—目标类
    - q_a的d-weight是初始目标类工作关系中被q_a覆盖的元组数 与初始目标类和对比类工作关系中被q_a覆盖的总元组数的比

类比较描述的量化判别规则表示（2）

- 目标类中较高的d-weight表明概化元组所代表的概念主要来自于目标类
- 较低的d-weight值则表明该概念主要来自于对比类

Status	Birth_country	Age_range	Gpa	Count
Graduate	Canada	25-30	Good	90
Undergraduate	Canada	25-30	Good	210

对给定的*status=“Graduate”, Birth_coutry=“Canada”, Age_range=“25-30”, Gpa=“Good”* 概化元组，其*d-weight*=90/(90+210)=30% (什么意思？)

类比较描述的量化判别规则表示（3）

- 使用类比较描述的量化判别规则表示可以更好的描述上述的情况，其形式为：

$$\forall X, target\_class(X) \Leftarrow condition(X) \quad [d : d - weight]$$

- 比如，刚才的挖掘结果可以使用量化判别规则表达如下：

$$\forall X,\ graduate\_student(X) \Leftarrow$$
$$birth\_country(X) = "Canada" \wedge age\_range(X) = "25-30" \wedge gpa(X) = "good" \ \ [d : 30\%]$$

- 请注意该区分规则表达的是充分条件，即X满足条件，则X为研究生的概率为30% (特征化量化规则表达的是什么条件？)

类描述：特征化和比较的表示

- 类特征化和类比较是形成类描述的两个方面，我们可以通过综合类特征化规则和类区分规则来形成类描述规则。
  - 量化特征化规则

    $$\forall X,\ \text{target\_class}(X) \Rightarrow \text{condition}(X)\ \ [t : t\_weight]$$

    - 必要条件
  - 量化判别规则

    $$\forall X,\ \text{target\_class}(X) \Leftarrow \text{condition}(X)\ \ [d : d\_weight]$$

    - 充分条件
  - 量化描述规则

    $$\forall X,\ target\_class(X) \Leftrightarrow$$

    $$condition_1(X)[t : w_1, d : w'_1] \vee \ldots \vee condition_n(X)[t : w_n, d : w'_n]$$

    - 充要条件

量化描述规则——示例 (1)

Location/item		TV			Computer			Both_items	
	Count	*t-wt*	*d-wt*	*Count*	*t-wt*	*d-wt*	*Count*	*t-wt*	*d-wt*
Europe	80	25%	40%	240	75%	30%	320	100%	32%
N_Am	120	17.65%	60%	560	82.35%	70%	680	100%	68%
Both_ regions	200	20%	100%	800	80%	100%	1000	100%	100%

- 一个给定类的概化元组的t-weight表明给定类中该元组的典型性（e.g.欧洲的销售(类)中，电视机(元组)占多少百分比?）
- 一个元组的d-weight表明，给定类的元组和对比类的元组相比，有多大区别（e.g.欧洲(类)的电视机(元组)销售和北美的电视机销售比如何？）

量化描述规则——示例 (2)

- 对于上述交叉表，可以直接用量化描述规则来表示

$$\forall X, Europe(X) \Leftrightarrow$$
$$(item(X) = "TV")[t:25\%, d:40\%] \vee (item(X) = "computer")[t:75\%, d:30\%]$$

  - 表明对99年AllElectronics公司的TV和计算机销售，如果一商品在欧洲售出，则其为TV的概率为25％…该公司40％的TV在欧洲售出…