维普资讯 http://www.cqvip.com

数据立方梯度挖掘的研究

刘玉葆 冯玉才 王元珍 冯剑琳
(华中科技大学计算机科学与技术学院 武汉430074)

The Research of the Cube Gradient Mining

LIU Yu-Bao FENG Yu-Cai WANG Yuan-Zhen FENG Jian-Lin
(College of Computer of Science and Technology, Huazhong University of Science and Technology, WuHan, China, 430074)

Abstract With the rapid development of data warehouse and OLAP techniques, the researchers begin to pay attention to the data mining in the data cube. Recently, Dr. T. Imielinski etc. firstly presented the problem of the cube gradient mining that is a generalization of association rule in data cube. In this paper, we firstly introduce the related concepts of data cube and condensed cube with an emphasis. Then we introduce some interesting problems related to the cube gradient mining including: constrained cube gradient mining and the query language of cube gradient. Finally, we introduce several issues on the combination of cube gradient and the condensed cube, that is, the cube gradient mining in the materialized data cube and the integration of cube gradient mining and cube browse.

Keywords Data cube, Condensed cube, Cube gradient

1 前言

随着人们生成、收集和存储数字化数据能力的极大提高，当今世界面临着各种原始数据的爆炸性增长。数据库技术的巨大进步创建了对大量数据的有效存储，成千上万的大型数据库被广泛地应用在商业、政府和科研等等部门。大量数据资源的积累为人们从历史数据中发现有用信息提供了基础，人们期望数据库能够提供智能化或者至少是半自动化的数据分析处理的能力。于是，数据仓库技术(Data Warehouse)、联机分析处理技术(On Line Analysis Processing)以及数据挖掘技术(Data Mining)应运而生。

典型地，数据仓库利用多维数据模型来有效地表示丰富的数据资源[3]。数据立方(Data Cube)为数据提供了一种多维的视图，是表示数据仓库中多维数据的一种标准逻辑模型。数据立方中包括维属性和度属性，度属性多是数字型的数据类型，描述了数据的聚集值，而维属性则既可以是数字型的也可以是非数字型的类型并且维属性一般是具有层次性的，比如日→月→年，可以看作时间维的一个层次。按照这种多维模型组织起来的数据大大提高了数据分析能力，一般地，我们可以利用 OLAP 技术进行数据分析处理，具体地，OLAP 提供了三种数据分析操作：选择(selection)、下钻(drilldown)、上钻(rollup)。按照多维模型实现的方式，OLAP 服务器可以分为两种类型：多维(Multidimensioanl-)和关系型(Relational-)OLAP 服务器，前者一般是用多维数据库来实现，而后者则是用关系型数据库来实现的。

随着数据仓库技术和 OLAP 技术的长足发展[17]，基于数据仓库或多维数据库的数据挖掘开始引起研究者的注意[4,5]，原因主要有：(1)数据仓库中存储的数据一般都是来自不同数据源的数据，这些数据往往是一些历史的、长期的数据，数据量往往也是很大的通常是以 G 或 T 计，如此丰富的数据资源为数据挖掘技术提供了一个很好的数据环境。(2)数据仓库中使用的数据往往是经过了预处理，去掉了数据中重复的、冗余的、脏乱的信息，因此，它们具有很好的结构性，很容易被数据挖掘算法处理。(3)由于数据仓库中的数据量一般都是以 G 或 T 计的，因此，基于 OLAP 技术的手工分析方法，实际上是不可能的，研究智能化或者至少是能半自动化的数据分析处理方法开始变得越来越迫切，数据挖掘技术作为一种实际有效的数据分析方法自然而然地走进了研究者的视线。在文[4]中，Dr. Han 详细地讨论了现有的一些数据挖掘方法如关联挖掘、聚类挖掘、分类挖掘等并且首次将这些 DM(Data Mining)技术与 OLAP(On Line Analysis Processing)技术进行了集成而提出了 OLAM(On Line Analysis Mining)概念。数据立方作为 OLAP 和数据仓库技术的基石自然引起了研究者的关注，数据立方中的数据挖掘开始成为数据挖掘研究领域的一个新课题，而数据立方梯度挖掘又是其中的重要任务之一。

2 数据立方

在文[10]中，Dr. Gray 提出了数据立方计算问题并且给出了 CUBE BY 算子。CUBE BY 算子是传统关系型数据库中 GROUP BY 算子的多维扩展，用于计算 CUBE BY 子句中各属性的所有可能组合所对应的 GROUP BY。例如，考虑一个关系基表 SALES(date, product, customer, amount)的数据立方查询：

```
SELECT date, product, customer, SUM(amount)
FROM SALES
CUBE BY date, product, customer
```

将对 CUBE BY 的属性 date, product 和 customer 的所有组合所对应的8个 GROUP BY 上进行求和聚集计算，即(date, product, customer)、(date, product)、(date, customer)、(product, customer)、(date)、(product)、(cus-

刘玉葆 博士研究生，主要研究方向：数据挖掘。冯玉才 教授，博士生导师，主要研究数据库理论及应用。王元珍 教授，博士生导师，主要研究数据库理论及应用。冯剑琳 博士，主要研究方向：数据挖掘及数据仓库。

维普资讯 http://www.cqvip.com

tomer)和 ALL(即 GROUP BY 属性为空的那个聚集)。总之，对于含 n 个 CUBE BY 属性的 CUBE 操作，将要计算2^n个不同的 GROUP BY。在 OLAP 术语中，CUBE BY 子句中各属性又被称作维(dimension)；而被聚集计算的属性则被称作度(measure，或度量)，如 SUM(amount)中的 amount 即度 amount；一个 GROUP BY 也被称作一个数据小方(cuboid)，包含 n 个维的数据小方称为 n-数据小方，数据立方中的数据小方(cuboid)满足格的关系。图1中给出了一个有三个维 A，B，C 的数据立方中的数据小方，一般地，k-数据小方能够从 k-1-数据小方中计算出来(图1中的箭头表示了这种关系)。

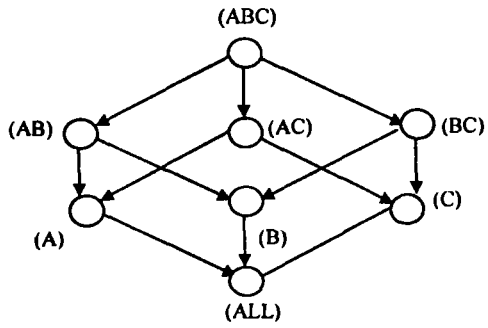

图1 一个有三个维的数据立方中的数据小方

数据立方算子一经定义，就引起了学术界和工业界的广泛关注。学术界在数据立方的计算、存储、索引、查询以及维护等等方面进行了大量的研究工作，其中文[2]的 BUC 算法已成为数据立方计算的经典算法之一。

显而易见，CUBE BY 是代价极其昂贵的算子。在 CUBE BY 子句中属性众多，基表较大而且很稀疏的情况下，它的计算结果极其庞大。虽然数据立方的尺寸问题吸引了大量的研究，然而却很少有文章从根本上，即减小数据立方的尺寸本身来加以解决。Dr. Wang 和 Dr. Feng 等提出了一种新的数据组织方法即浓缩数据立方技术[8]，利用浓缩数据立方可以减小数据立方的尺寸，从而减少计算时间与存储开销。下面我们以一个简单例子简要说明浓缩数据立方的基本思想：

假定我们要在基表 R 上计算数据立方 CUBE BY(A，B)：

基表 R(A，B，M)

TID	A	B	M
1	0	1	20
2	1	1	30

CUBE BY(A，B)

TID	Cuboid	A	B	M
1	ALL	*	*	50
2	A	0	*	20
3	AB	0	1	20
4	B	*	1	50
5	A	1	*	30
6	AB	1	1	30

图2 基表 R 及其数据立方 CUBE BY

在上述例子中，基表 R 有两个维属性：A 和 B，以及一个度属性 M，计算数据立方的聚集函数为求和 SUM。基表 R 包含两个元组(tuples)。完整的数据立方 CUBE BY(A，B)有6个元组，分别属于四个数据小方：ALL，A，B 和 AB，其中的 * 表示特殊的维值 ALL。假如我们将基表 R 在维 A 上划分，亦即计算数据小方 A。我们得到两个划分，即 A 取值为0的划分和取值为1的划分。我们看到 A 取值为0的划分仅仅由 TID 等于1的基表元组构成，对应的数据立方元组为(0，*，20)；如果将该划分进一步在维 B 上划分，即计算数据小方 AB；所得到的划分仍然只由 TID 等于1的基表元组构成，得到的数据立方元组为(0，*，20)的祖先(0，1，20)。我们将 TID 等于1这样的基表元组称为单元组(single tuple)。

单元组最重要的特性就是一旦确定某个数据立方元组源自一个单元组，那么是它祖先的任意数据立方元组就被唯一确定：只需简单地将后续各维的取值取 * 或者该单元组的对应维值，而度的取值均相等。因此假如我们将某个单元组以及使之成为单元组的维集合(即在其上做划分的维集合)记下来，我们就可以表示源自该单元组的所有数据立方元组。在上述例子中，基表元组(0，1，20)加上维集合{A}就可以表示两个数据立方元组(0，*，20)和(0，1，20)。更一般地，假定有从1到 N 个维，又设某个基表元组是相对维集合{1，2，…，i}的单元组，这里 i⇐N，那么该基表元组可以表示2^{N-i}个数据立方元组。例如，i＝3，N＝10，则该个单元组可以表示2^7＝128个数据立方元组，换句话说，128个数据立方元组好比浓缩到一个单元组中去了。与相关工作相比，浓缩数据立方具有以下独特的特征：

·浓缩数据立方不是压缩的。因此当浓缩数据立方用来响应查询的时候不需要通常开销很大的解压缩。

·浓缩数据立方是经完整计算的。它不同于那些仅仅选择部分数据小方进行计算来减小数据立方尺寸的方法。因此当浓缩数据立方用来响应查询的时候不需要进一步应用聚集函数。

·浓缩数据立方提供精确的聚集值。它不同于那些通过提供近似聚集值来减小数据立方尺寸的方法。这些近似方法包括小波(wavelet)，多变量多项式(multivariate ploynomials)，柱状图(histogram)，抽样(sampling)等等。

·浓缩数据立方支持全面的 OLAP 应用。它不同于那些通过裁减数据立方、只响应某些查询类型来减小数据立方尺寸的方法。

文[8]提出了类 BUC 算法和 MinCube 算法来计算维顺序给定情况下的浓缩数据立方，类 BUC 算法沿维1到维 N 单向搜索基表单元组，而 MinCube 算法则同时从左右两个方向搜索所有可能的基表单元组，能够生成一个最小的浓缩数据立方，但是由于搜索空间太大而导致 MinCube 算法的计算开销很大，并且在展开 MinCube 所生成的浓缩数据立方的时候会重复生成某些数据立方元组，因此 MinCube 主要具有理论意义，它通过算法构造出唯一最小数据立方的存在性证明，而类 BUC 算法所生成的浓缩数据立方在展开的时候，不会重复生成任何数据立方元组。浓缩数据立方因其高效的数据存储和组织的特点必将在 OLAP 领域中拥有广泛的应用。

3 数据立方梯度挖掘

在文[4]中，Dr. Han 首次详细地讨论了数据挖掘与 OLAP 的结合问题并且介绍了数据挖掘与 OLAP 技术结合的产物 DBMiner 的应用。在 DBMiner 中集成了很多有效的数据挖掘算法，如特征挖掘(characterization)、分类挖掘(classification)、聚类挖掘(clustering)和关联挖掘(association rule)等，然而，人们并不满足于从数据立方中挖掘出在原始数据中(不象数据立方那样组织的数据)也能挖掘的模式或知识如：关联、分类、聚类等，人们希望能从数据立方中挖掘出一些新

维普资讯 http://www.cqvip.com

的、数据立方所独有的一些模式或知识。

近来，Dr. T. Imielinski 等首次提出了数据立方梯度挖掘技术[7]。数据立方梯度挖掘是关联规则在数据立方上的推广和一般化。关联规则挖掘[13]一直以来都是数据挖掘领域中的重要任务之一，自从它在1993年由 Dr. R. Agrawal 与 Dr. T. Imielinski 和 Dr. A. Swami 提出以后，一直受到研究者广泛的关注[16]，关联规则最先是针对交易数据库提出来的，其直观的意义就是客户在购买某些东西的时候有多大的倾向也会购买另外一些东西。因此通过关联规则可以挖掘客户的购买模式，从而提高商业决策(例如进什么货来销售，货物如何摆设才能获得最大的利润等等)的质量。一个关联规则的例子就是"90%的客户在购买面包和黄油的同时也会购买牛奶"(这里90%就称为该关联规则的可信度(Confidence))，而同时购买了面包和黄油的交易占全部交易的0.6%(这里0.6%就称为该关联规则的支持度(Support))。上面的陈述通常表达为概率规则的形式。这条规则的前件(antecedent)由面包和黄油构成，后件(consequent)则由牛奶单独构成。事实上，0.6%就是面包和黄油同时被购买的概率，90%则是在已经购买了面包和黄油的前提下，再购买牛奶的概率，也就是牛奶的条件概率。如果我们换一个角度，上述关联规则也可以被看成有关下述变化的陈述：表示规则前件的数据立方元组(面包，黄油，支持度)，当在加上规则后件而具体化(specialization，也就是 OLAP 术语中的下钻(drilldown)操作)成为数据立方元组(面包，黄油，牛奶，支持度)时，它的度量值(即其支持度)的变化。这里数据立方元组(面包，黄油)以及(面包，黄油，牛奶)的支持度实际上都是通过聚集函数 COUNT 计算而来，两者支持度的差别就表现在可信度上。如果把聚集函数 COUNT 换成其它的聚集函数，例如 MAX，MIN，SUM 和 AVG 等等，那么我们就可以推而广之，获得类似表达因数据立方元组发生变化而引起的各种度量变化。为了规则易于解释以及避免搜索空间过大，目前数据立方梯度只考虑下述引发数据立方元组发生变化的操作：具体化(即下钻)，一般化(generalization，也就是 OLAP 术语中的上钻(rollup)操作))和突变(mutation，即仅仅改变数据立方元组在某一维的值)。这就是数据立方梯度的基本思想，它反映了数据立方中不同元组之间度值变化的规律，显然，数据立方梯度要比关联规则更富于表达性，能够支持更复杂、更广泛的假设分析(what-if)以及趋势分析等。然而，数据立方梯度挖掘存在着下几个不足之处：(1)挖掘算法的效率比较低。(2)挖掘结果很大，分析者不易理解。

针对上述两个主要不足，Dr. Gong 和 Dr. Han 等人提出了约束性数据立方梯度挖掘技术[11]并且将该技术集成到了在线数据立方浏览工具 CubeExplorer[14]。跟一般数据立方梯度挖掘相比，约束性数据立方梯度挖掘的主要不同是引入了不同的"约束"条件，利用"约束"不仅可以限制问题的搜索空间而且还可以挖掘出用户感兴趣的数据立方梯度，而不是象一般数据立方梯度挖掘一样挖掘出数据立方中所有隐藏的数据立方梯度。直观地来看，约束性数据立方梯度挖掘可以看作是一般数据立方梯度挖掘的简体版本。

下面我们将具体地介绍一下约束性数据立方梯度挖掘和数据立方梯度的查询。

3.1 约束性数据立方梯度

定义1 一般地，假定关系表 R 上的数据立方 D 中有 n 个维属性$(a_1, a_2, \cdots, a_n)$，数据立方元组 c 在维 a_i 上的值记为 $c(a_i)$，对 D 中的不同元组 c_1、c_2，我们说 c_1 是 c_2 的祖先(c_2 是 c_1 的后代)，如果 $c_1(a_i)=c_2(a_i)\neq *$，$c_1(a_j)=*$，$c_2(a_j)\neq *$ $(1\leqslant i,j\leqslant n, i\neq j)$；我们说 c_1 是 c_2 的兄弟(c_2 是 c_1 的兄弟)，如果存在唯一的 j 使得 $c_1(a_j)\neq c_2(a_j)$，$c_1(a_i)=c_2(a_i)$ $(1\leqslant i,j\leqslant n, i\neq j)$。为描述方便，如果元组 c_1、c_2 满足祖先或后代或兄弟关系，则称 c_1 与 c_2 相似。

定义2 探测约束(probe constrains)C_{prb}，重要约束(significant constraints)C_{sig}，梯度约束(gradient constrains)C_{grad}。我们说数据立方中的元组 c 是重要的如果 $C_{sig}(c)=$true，如果 c 是重要的并且 $C_{prb}(c)=$true，则说 c 是探测元组，记为 c_p，探测元组集记为 P，如果元组 c 是重要的并且与 P 中的探测元组满足梯度约束，则 c 称作 P 的梯度元组，记为 c_g。重要约束 C_{sig} 定义了数据立方元组度属性上的约束条件。这些约束可能是反单调的(anti-monotonic)，也可能是非反单调的。反单调性是指如果元组 c 不满足 C_{sig}，则 c 的后代也不会满足 C_{sig}，例如 $C_{sig}\equiv(M>10)$，反单调性通常可以用于剪枝，不是反单调的约束通常可以转化为弱反单调约束(weaker anti-monotonic)[6]。探测约束 C_{prb} 定义了数据立方元组维属性上的约束条件，用于选择用户感兴趣的数据立方元组，类似 SQL 语言中的 SELECT 语句。梯度约束 C_{grad} 定义不同数据立方元组间度属性上取值变化的约束。一般表示为：$C_{grad}(c_g, c_p)\equiv(g(c_g, c_p)\theta v)$，其中 $\theta\in\{\leqslant,\geqslant,<,>\}$，v 是一个常数，$g(c_g, c_p)$ 是梯度函数。我们说梯度-探测元组对(c_g, c_p)是有趣的如果 c_g 与 c_p 相似并且 $C_{grad}(c_g, c_p)=$true。

定义3 一般地，给定关系表 R 以及重要约束 C_{sig}，探测约束 C_{prb}，梯度约束 C_{grad}，约束立方梯度挖掘问题就是找出 R 的数据立方中所有感兴趣的梯度-探测元组对(c_g, c_p)。

例：假设用户指定的 $C_{prb}\equiv(A=4, B=*, C=*)$，$C_{sig}\equiv(M>50)$，$C_{grad}(c_1, c_2)\equiv(c_1(M)-c_2(M))>0$，则重要元组 $c_g=(4,*,*,150)$与探测元组 $c_{p1}=(4,5,1,70)$，$c_{p2}=(4,5,2,80)$构成梯度-探测元组，需要说明的是因为 C_{prb} 中 $A=4$，所以$(*,5,1,70)$不是探测元组。

定义4 假设探测元组 c_p，梯度元组 c_g 有 m 个维属性，将两个元组中满足条件：$c_p(a_i)=*$，$c_g(a_i)\neq *$(a_i 是数据立方元组的维，$1\leqslant i\leqslant m$)的维称为 * 不匹配；将满足条件：$c_p(a_i)\neq c_g(a_i)$，$c_p(a_i)\neq *$，$c_g(a_i)\neq *$的维称为定值不匹配。我们说 c_p 与 c_g 是匹配的，如果 c_p 与 c_g 的维中没有定值不匹配或者只有一个定值不匹配但是没有 * 不匹配。

例：探测元组 $cp=(1,1,1)$与梯度元组 $c_g=(1,*,*)$是匹配的，因为它们没有定值不匹配，而 $c_p=(4,*,*)$与 $c_g=(3,5,*)$，因为它们有一个定值不匹配和一个 * 不匹配。

定义5 梯度元组 c_g 的活动集(live set)是指有可能跟 c_g 的后代构成梯度-探测元组对的探测元组的集合，我们用 $LiveSet(c_g)$表示梯度元组 c_g 的活动集。

一般地，我们可以通过维匹配方法来生成梯度元组的活动集，具体地，我们可以删除 $LiveSet(c_g)$中与 c'_g 不匹配的探测元组得到 $LiveSet(c'_g)$，这里梯度元组 c'_g 是 c_g 的后代。在生成感兴趣的梯度-探测元组对时，我们只要将梯度元组 c_g 与 c_g 相关的探测元组即 $LiveSet(c_g)$中的探测元组进行比较即可。

文[11]中给出了一个高效的挖掘算法 LiveSet-driven (LiveSet 算法)，LiveSet 算法的主要步骤如下：

(1)利用数据立方计算算法从给定关系表 R 中计算出满足重要约束 C_{sig}，探测约束 C_{prb} 的探测元组集 P。

(2)初始化 $c_g=(*,*,\cdots *)$以及 $LiveSet(c_g)=P$。

(3)如果 c_g 与 $LiveSet(c_g)$中的探测元组满足梯度-探测元组对条件，则输出该元组对。

(4)从关系表 R 中计算出满足重要约束 C_{sig} 的新的梯度元组 c_g，从 c_g 祖先的活动集中，利用维匹配方法生成 $LiveSet(c_g)$。

维普资讯 http://www.cqvip.com

下面，我们通过一个具体例子来说明 LiveSet 算法过程。

例：假设给定的关系表 R 如图2所示，探测约束 C_{prb}≡(A＝*，B＝1)，重要约束 C_{sig}≡(M＞15，梯度约束 C_{grad}≡m(c_g)－m(c_p)＞0，数据立方计算算法，我们选用 BUC 算法，在图2中给出了关系表 R 对应的数据立方。

(1)利用 BUC 算法和 C_{prb}、C_{sig}，计算 P＝{(0，1，20)，(＊，1，50)，(1，＊，30)}。

(2)初始化 c_g＝(＊，＊，50)以及 LiveSet(c_g)＝P＝{(0，1，20)，(＊，1，50)，(1，＊，30)}。

(3)比较 c_g 与 LiveSet(c_g)中的探测元组，可以得到梯度-探测元组对：{(＊，＊，50)，(0，1，20)}，{(＊，＊，50)，(1，＊，30)}

(4)按照 BUC 算法继续生成新的梯度元组，假设 c_g＝(0，＊，20)，利用维匹配法从 c_g 的祖先(c_g＝(＊，＊，50))的活动集中生成新的梯度元组 c_g＝(0，＊，20)的活动集，LiveSet(c_g＝(0，＊，20))＝{(0，1，20)，(＊，1，50)，(1，＊，30)}

(5)比较 c_g＝(0，＊，20)及其 LiveSet(c_g)，因为 c_g 的度值与 LiveSet(c_g)中的探测元组的度值不满足梯度约束，因此 c_g 及其 LiveSet(c_g)不能生成梯度-探测元组对。

(6)重新计算新的梯度元组，其他步骤类似前面过程。

3.2 数据立方梯度查询语言

数据立方梯度挖掘产生的结果集往往是很大的，而用户感兴趣的可能只是其中的一小部分，如何让用户从结果集中查询自己感兴趣的立方梯度就显得很重要了，文[15]还给出了 SQL99标准中数据挖掘查询语言定义的标准，在文[7]中，Dr. T. Imielinski 等人设计了数据立方梯度的类 SQL 的查询语言 CubegradeQL。

一般地，一个数据立方梯度可用一个五元组表示为：

SourceCube → TargetCube [Measures，Values，Delta Values]

其中 SourceCube(源立方)，TargetCube(目的立方)表示数据立方中的不同元组，SourceCube 和 TargetCube 之间满足具体化(specification)或一般化(generalization)或突变(mutation)关系，Measures 表示 SourceCube、TargetCube 中不同的度属性集合，Values 表示 SourceCube 中度的取值，而 Delta Values 表示 SourceCube、TargetCube 中度的取值的变化。根据 SourceCube，TargetCube 之间的关系，数据立方梯度分为三种：具体化立方梯度、一般化立方梯度、突变立方梯度。

例：假设在一个销售数据立方中存在如下数据立方梯度：

(salesMilk＝[＄20，＄30])→(salesMilk＝[＄20，＄30]，salesCereal＞＄0)[AVG(Age)，AVG(Age)＝23，DeltaAVG(Age)＝90%]。这个数据立方梯度表示的含义是：那些每月花20-30美元购买牛奶的购买者与那些每月花20-30美元购买牛奶和谷类食品的购买者相比，购买者的平均年龄下降了90%。

文[7]首先定义了数据立方的查询语言 CubeQL，然后在 CubeQL 的基础上给出了数据立方梯度查询语言 CubegradeQL。

CubeQL 语言的语法表示如下：

GET CUBES
FROM DB
WHERE ⟨conditions⟩，condtitions 表示为数据立方元组要满足的约束条件。CubeQL 用于给定的数据库的数据立方中查询出所有满足 WHERE 子句中的条件的数据立方元组。通常 condtitions 由如下条件连接通过逻辑连接词 and，or 连接而成：

·描述条件，表示为 CUBE MATCHES ⟨conjunct pattern⟩。联结模式(conjunct pattern)由维属性上的属性-值表达式和逻辑连接词 and，or 连接而成，一般地，属性-值表达式有三种情况：(1)A_i＝v，(2)A_i∈Interval，(3)A_i＝＊，A_i 是数据立方的维属性。一个描述条件的例子是：CUBE MATCHES(areaType＝'urban' or age＝'＊' or income＝＊，or salesBread＝＊ or salesMilk＝＊ or salesCookies＝＊ or salesSoda＝＊)。

·度条件，指定数据立方中需要计算的度属性列表，表示为 MEASURES＝⟨measure list⟩，如：MEASURES＝AVG(salesMilk)。

·值条件，指定作用在度值上的布尔表达式，比如 AVG(salesMilk)＞50。

·长度条件，指定数据立方元组中取值为非＊的维属性的个数，用关键字 LENGTH 表示。

下面，给出一个数据立方查询的例子：

```
GET CUBES
FROM customer
WHERE
CUBE MATCHES(areaType='urban' or age='*' or income=
*, or salesBread= * or salesMilk= * or salesCookies= * or sa-
lesSoda= * )
AND MEASURES={COUNT( * ),AVG(salesCereal)}
AND COUNT( * )>1000
AND AVG(salesCereal<20)
```

CubegradeQL 语言的语法：

```
GET [SPECIALIZE (X) | GENERALIZE (X) | MUTATE (X)]
CUBEGRADES
FROM DB
WHERE ⟨condition⟩
```

CubegradeQL 用于从给定的数据库的数据立方中查询出所有满足 WHERE 子句中的条件的数据立方梯度，WHERE 条件由如下条件连接通过逻辑连接词 and，or 连接而成：

·描述条件，CubegradeQL 中有源立方描述条件和目的立方描述条件两种，分别用 SOURCE_CUBE MATCHES ⟨conjunct pattern⟩和 TARGET-CUBE MATCHES⟨conjunct pattern⟩，这里的联结模式(conjunct pattern)与 CubeQL 的联结模式是一样的。

·连接条件，表示源立方和目的立方之间的关系，即：一般化、具体化、突变，分别表示为：TARGET-CUBE SPECIALIZES SOURCE-CUBE(目的立方是源立方的具体化)，TARGET-CUBE GENERALIZES SOURCE-CUBE(目的立方是源立方的一般化)，TARGET-CUBE UNION-COMPATIBLE SOURCE-CUBE(目的立方是源立方的突变)。上述三种表示还有基于谓词变体表示，如 SPECIALIZES(X)，X 表示目的立方具体化了源立方的维，类似地，可得其他的表示法 GENERALIZES(X)，UNION-COMPATIBLE(X)。

·度条件，类似 CubeQL 中的度条件，指定数据立方中需要计算的度属性列表。

·值条件和变化值条件，指定源立方和目的立方中需要计算的度值和变化值。

·长度条件，指定源立方和目的立方中取值为非＊值的维的个数，分别用 SOURCE-LENGTH 和 TARGET-LENGTH 表示。

一个数据立方梯度查询的例子是：

```
GET SEPECIALIZATION CUBEGRADES
FROM customer
WHERE SOURCE-CUBE MATCHES(areaType= * or
age= * or income= * or salesBread= * or salesMilk= *
or salesCookies= * or salesSoda= * )
AND TRAGET-CUBE MATCHES (areaType = * or
age= * or income= * or salesBread= * or salesMilk= *
or salesCookies= * or salesSoda= * )
```

维普资讯 http://www.cqvip.com

AND TARGET-CUBE SPECIALIZE SOURCE-CUBE
AND MEASURES={COUNT(*),AVG(salesCereal)}
AND COUNT(*)>1000 and AVG(salesCereal)<20
AND DeltaAVG(salesCereal)>1.5 and DeltaCOUNT(*)>0.5

4 数据立方梯度与浓缩数据立方

这一节中,我们介绍了将数据立方梯度挖掘与浓缩数据立方技术结合起来进行研究的几个主要问题:实化数据立方中的梯度挖掘、梯度挖掘与数据立方浏览的集成。

实化数据立方中梯度的挖掘:现有的数据立方梯度的研究都是作用在非实化数据立方之上的,也就是说从数据库中计算出数据立方是数据立方梯度挖掘的一个重要的任务。实际中,为了支持快速的OLAP查询,一般都会预先计算出数据立方,而数据立方梯度就隐含在数据立方之中,尽管由于空间和时间约束的限制,有可能只能实化部分数据小方。因此如何从实化数据立方中挖掘数据立方梯度是一件有趣而有意义的问题。不过实化数据立方的大小将会给我们的挖掘任务提出严峻的挑战,因为实化数据立方往往是很大的,通常是以G或T计的。一个可行的办法就是采用浓缩数据立方来存储实化数据立方,利用浓缩数据立方可以大大减少数据立方的存储空间,尤其在稀疏的数据立方中,理想情况下可以减少70%-80%。另一方面,在浓缩数据立方中,单元组及其所浓缩的数据立方元组都具有相同的度值,也就是说这些元组之间是没有梯度的,从而减少了需要进行度值比较的数据立方元组的个数。类似地,浓缩数据立方的这种性质对现有数据立方梯度的挖掘也同样适合。例如在约束性数据立方梯度的挖掘中,当使用浓缩数据立方来存储数据时,探测元组集P的大小将会减少,计算出来的梯度元组个数也将会减少,因此产生梯度-探测元组对时,元组的搜索空间也将减少。当然,采用浓缩数据立方存储数据以后,面对的一个新的问题就是如何从单元组中提取出被其所浓缩的数据立方元组。这个问题可以通过单元组的扩展操作(Expand)来解决。利用扩展操作可以还原出被单元组浓缩的所有元组,并且该操作是不需要额外开销的。

数据立方梯度与数据立方浏览:假设有一个商品销售数据立方,维属性有:销售时间、商品名称、销售部门,度属性是销售数量。假设用户想要从中找出食品部门在不同时期内销售数据变化比较大的商品。传统的做法是用户首先选定"销售部门"是食品部的维,然后,利用OLAP中的下钻操作(drill-down)从"商品名称"或"销售时间"维的最高层到底层进行浏览,如果沿着某条路径不能发现自己满意的商品销售记录时,用户又得利用上钻(drillup)操作或者是选择(selection)操作回到某一维的某一层重新开始浏览。由于数据立方往往是比较大的,典型的一个数据立方通常有5—8维,每个维有2—8层次,每个层次有数百个数据成员。从如此大的数据立方中用人工的办法找出隐含在其中的极少的一部分异常的数据是一件费力的事情。另外,由于数据量太大,用户通过这种人工浏览方式获得的结果集可能存在着不完整、不准确。实际上,我们可以将数据立方梯度挖掘与数据立方的浏览结合起来,利用挖掘出来的数据立方梯度来指导数据立方的浏览。

以上述的浏览为例,假设用户浏览到食品维的某一层次上时还没有找出自己感兴趣的立方元组,也许这时用户自己也不知道使用何种OLAP操作继续浏览了。这时,用户可以调用数据立方梯度挖掘算法,找出食品维的兄弟、或祖先、或后代中是否存在元组与用户正在浏览的元组构成感兴趣的立方梯度。当然,我们不能简单地列出所有的立方梯度,因为数据立方梯度的结果集可能很大而不便于用户理解。因此挖掘算法输出给用户的应该是一些比较概括性的立方梯度,比如,梯度值最大的几个立方梯度。如果结果集中不存在感兴趣的立方梯度,则用户可以考虑浏览其他的"销售部门",否则,用户可以利用挖掘出来的概括性的立方梯度指导用户采用OLAP的相关操作大致地定位到感兴趣的数据立方元组的范围之中,然后再继续具体地浏览。值得说明的是这种梯度挖掘应该是交互式的,允许用户在浏览时随时调用进行挖掘。

小结 本文中,我们首先介绍了数据立方,重点是浓缩数据立方的相关概念。然后,我们介绍了数据立方梯度挖掘研究的一些研究现状,包括约束性数据立方梯度挖掘、数据立方梯度查询语言等。最后,我们介绍了将数据立方梯度挖掘与浓缩数据立方技术结合起来进行研究的几个主要问题:实化数据立方中的梯度挖掘、梯度挖掘与数据立方浏览的集成。这些也是我们目前比较感兴趣的问题。数据立方梯度挖掘是数据挖掘研究领域中一个比较新的课题,很多的工作才刚开始,要解决的问题还有很多,新的研究成果和研究论文不断涌现,是一个比较有前景的研究课题。

参考文献

1 Agarwal S, et al. On the computation of multidimensional aggregates. VLDB, 1996
2 Beyer K, Ramakrishnan R. Bottom-up computation of sparse and iceberg cubes. ACM SIGMOD, 1999
3 Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 1997, 26: 65～74
4 Han J. Towards On-Line Analytical Mining in Large Databases. ACM SIGMOD Record, Mar. 1998. 97～107
5 Palpanas T. Knolwedge Discovery in Data Warehouses. ACM SIGMOD Record, 2000
6 Han J, Pei J, Dong G, Wang K. Efficient computation of iceberg cubes with complex measures. ACM SIGMOD, 2001
7 Imielinski T, Khachiyan L, Abdulghani A. Cubegrades: Generalization Association Rules: [Tech. Rep]. Dept Computer Science, Rutgers University, Aug. 2000
8 Wang Wei, Feng Jianlin, Lu Hongjun, Yu J X. Condensed Cube: An Effective Approach to Reducing Data Cube Size. IEEE ICDE, 2002
9 Sarawagi S, Agrawal R, Megiddo N. Discovery driven exploration of OLAP and data cubes. EDBT 1998
10 Gray J, Bosworth A, Layman A, Pirahesh H. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. IEEE ICDE, 1996
11 Dong Guozhu, Han Jiawei, Lam J, Pei Jean, Wang Ke. Mining Multi-Dimensional Constrained Gradients in Data Cubes. VLDB, 2001
12 Ng R, Lakshmanan L V S, Han J, Pang A. Exploratory mining and pruning optimizations of constraned association rules. ACM SIGMOD, 1998
13 Agrawal R, Imielinski T, Swami A. Mining association ruels between sets of items in large databases. ACM SIGMOD, 1993
14 Han Jiawei, Wang Jianyong, Dong Guozhu, Pei Jian, Wang Ke. CubeExplorer: online exploration of data cube. ACM SIGMOD, 2002
15 Mattos N M, et al. SQL99, SQL/MM, and SQLJ: An Overview of the SQL Standards. IBM Database Common Technology, 1999
16 Chen M S, Han J, Yu P S. Data Mining: An overview from a database perspective. IEEE Trans. Knowledge and Data Engineering, 1996, 8: 866～883
17 Codd E F, Codd S B, Salley C T. Providing OLAP to User-Analysis: An IT Mandate. http://www.hyperion.com, 1993