dmGQL：一种新的数据立方梯度查询语言*)

刘玉葆　冯玉才
（华中科技大学计算机科学与技术学院　武汉430074）

摘　要　现有数据立方梯度查询语言CubegradeQL主要是针对非实例化数据立方的，实际上，为了提高OLAP查询效率，数据仓库中往往保存了大量实例化的数据立方。本文我们改进了CubegradeQL语言，给出了一个新的查询语言dmGQL，dmGQL能够支持实例化/非实例化数据立方中的梯度查询，最后，我们讨论了dmGQL的查询处理。

关键词　数据立方，数据立方梯度，数据挖掘查询语言

dmGQL: A New Query Language of Cube Gradient

LIU Yu-Bao　FENG Yu-Cai
(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　The existing data cube gradient query language CubegradeQL mainly performs on the non-materialized cube. In fact, in order to the request time of OLAP query, data cube is always computed (i.e., materialized)in data warehouse. In this paper, we present an improved query language dmGQL that can support the query on materialized/non- materialized cube. Finally, the query processing of dmGQL is discussed.

Keywords　Data cube, Data cube gradient, Data mining query language

1　引言

自从Dr. T. Imielinski在文[1]中提出了第二代数据挖掘系统的概念并且指出与数据库管理系统的紧密集成是第二代数据挖掘系统的典型特征以后，数据挖掘查询语言研究一直是数据挖掘领域中一个重要的研究方向。

数据立方梯度挖掘问题[3]是关联规则[2]在数据立方[4]上的推广和一般化。关联规则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，其度量值(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。在文[3]中，Dr. T. Imielinski等人提出了数据立方梯度类SQL的查询语言CubegradeQL。

本文我们首先分析了CubegradeQL语言的不足并且提出了一种新的查询语言dm GradientQL，简称为dmGQL，dm指我们自主版权的达梦(dm)数据库管理系统，然后讨论了dmGQL的查询处理，最后对全文进行了总结。

2　CubegradeQL语言

数据立方算子CUBE BY[4]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COUNT聚集计算，即(date, product, customer)、(date, product)、(date,

*)本文受"十五"科技部攻关项目资助(No. 2001BA110B01)。刘玉葆　博士研究生，研究方向：数据挖掘、数据仓库；冯玉才　博士生导师，研究方向：数据库理论及应用。

customer)、(product, customer)、(date)、(product)、(customer)和 ALL(即 GROUP BY 属性为空的那个聚集)。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。CUBE BY 子句中各属性又被称作维;而被聚集计算的属性则被称作度量,如 SUM (amount)中的 amount 即度量 amount,一个数据立方元组由维和度量两部分组成;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。

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 之间满足具体化或一般化或突变关系,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 中度值的变化。具体化、一般化、突变关系如图1所示,其中 * 表示 ALL。

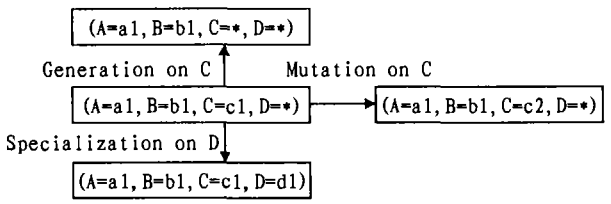

图1 一般化、具体化、突变关系

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[3]首先定义了数据立方查询语言 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 用于从给定的数据库中查询出所有满足 WHERE 条件的数据立方元组。通常 conditions 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)$A_i \in$ 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。一个数据立方查询的例子是:GET CUBES FROM customer WHERE CUBE MATCHES (areaType = 'urban' or age = '*' or income = *, or salesBread = * or salesMilk = * or salesCookies = * or salesSoda = *) 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 用于从给定数据库中查询出所有满足 WHERE 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 = *) 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。

3 dmGQL 语言

从 CubegradeQL 的语法中可以看出 CubegradeQL 主要是从给定数据库中挖掘数据立方梯度,也就是说从数据库中计算出数据立方是 CubegradeQL 的一个主要任务,文[3]给出了基于宽度搜索的查询评价算法。实际上,为了支持快速的 OLAP 响应,数据仓库存放了很多事先已经计算好的实例化的数据立方。为了支持实例化数据立方中的梯度查询,我们对 CubegradeQL 进行了改进,提出了一种新的查询语言

dmGQL。

dmGQL 语言的语法与 CubegradeQL 的语法类似，不同之处是 dmGQL 中的 FROM 字句中可以是实例化的数据立方名，具体语法格式如下：

GET[SPECIALIZE(X)|GENERALIZE(X)|MUTATE(X)] CUBEGRADES FROM DB|CUBE WHERE (condition)。

一个 dmGQL 的例子为：GET SEPECIALIZATION CUBEGRADES FROM customer_cube 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 = *) AND TARGET-CUBE SPECIALIZE SOURCE-CUBE AND MEASURES = {COUNT (*), AVG (salesCereal)} AND COUNT(*)>1000 AND AVG(salesCereal)<20 DeltaAVG (salesCereal) > 1.5 and DeltaCOUNT (*) > 0.5，其中 customer_cube 是一个实例化的销售立方。

因为 dmGQL 的 FROM 字句中既可以是关系表名也可以是数据立方名，因此，dmGQL 支持实例化和非实例化数据立方中数据立方梯度的查询。

4 查询处理

4.1 评价算法

虽然 dmGQL 的语法格式与 CubegradeQL 的很相似，但是 dmGQL 查询处理过程中的查询评价算法与 CubegradeQL 有着本质的不同。实例化数据立方的大小往往是很大的，为了能够减少实例化数据立方的大小，我们采用了最近提出的浓缩数据立方数据结构[5]。浓缩数据立方利用单元组概念来达到减少数据立方计算时间与数据立方的存储开销。下面我们以一个简单例子(表1)简要说明浓缩数据立方的基本思想。

表1 基表 R 及其数据立方

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

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

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。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(basic 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“导出”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。

在文[6]中，我们给出了基于浓缩数据立方的数据立方梯度挖掘算法 eLiveSet 算法，eLiveSet 算法针对的是非实例化数据立方，从给定关系表中计算出浓缩数据立方是 eLiveSet 算法的一个重要部分。eLiveSet 算法中采用了自底向上的数据立方计算算法，是一种深度搜索算法，通常自底向上的数据立方计算算法要比文[3]中宽度搜索算法采用的自顶向下的数据立方计算算法效率要高，尤其是对稀疏数据。

数据立方实例化以后，数据立方梯度查询本质上变成了一个从数据立方中搜索出满足一般化、具体化、突变关系以及 dmGQL 查询中约束条件的数据立方元组对(SourceCube，TargetCube)，通过元组对的度量值可以计算出梯度元组的 Delta Values，搜索算法比较直观，其算法框架如图2所示，由于浓缩数据立方大大减少了数据立方的大小，算法的搜索空间将会大大减少。

算法1 GSCC (Gradient Search based on Condensed Cube)

Input: (1)the condensed cube D, (2)the dmGQL
Output: the gradient in D
Method:
1. for every tuple p in D
2. for every tuple p in D
3. if p,q statisfy the constraints in dmGQL output a gradient between p, q.

图2 GSCC 算法描述

从 GSCC 算法的描述中我们可以看出，数据立方梯度查询代价 $C_{gradient}$ 可以通过满足查询语句中约束条件的 SourceCube 和 TargetCube 的元组个数来评价：$C_{gradient} = N_{SourceCube} \times N_{TargetCube}$，其中 $N_{SourceCube}$，$N_{TargetCube}$ 表示满足查询条件的 SourceCube 和 TargetCube 元组个数，值得指出的是 $N_{SourceCube}$，$N_{TargetCube}$ 是指“导出”后的一般性的数据立方元组个数，也就是说如果 GSCC 算法中的 p，q 是单元组的话，SourceCube 和 TargetCube 是被 p 或 q 所浓缩的数据立方元组。

4.2 查询过程

dmGQL 的查询处理过程如图3所示，主要包括语法分析、语义检查、查询计划、查询执行几个主要步骤。

1. 语法分析，检查 dmGQL 查询语句语法格式是否正确，与一般 SQL 语言不同，在 dmGQL 中我们考虑没有多个关系表或多个数据立方的情况。

2. 语义检查，主要检查查询语言的语义是否合法，包括检查条件中的属性是否属于某个关系或数据立方，属性的数据类型是否合法等，通过数据字典或元数据来检查 FROM 字句的关系表或数据立方是否存在。

(下转第98页)

$F(S)$中有以下形式的公式为公理：

(L1)$A\rightarrow(B\rightarrow A)$

(L2)$(A\rightarrow(B\rightarrow C))\rightarrow((A\rightarrow B)\rightarrow(A\rightarrow C))$

(L3)$(\neg A\rightarrow\neg B)\rightarrow(B\rightarrow A)$

2°. 推理规则：MP 规则，由 A 和 $A\rightarrow B$ 推得 B。

由公式集 $F(S)$，公理$(L1)$，$(L2)$，$(L3)$以及 MP 规则组成的系统，称为经典逻辑的形式演绎系统 $\mathscr{L}$。

注. 经典逻辑形式系统 $\mathscr{L}$ 中的$(L2)$不是形式系统 $\mathscr{D}$ 中的定理。例如，$v(A)=0.9$，$v(B)=0.5$，$v(C)=0.3$，$v((A\rightarrow(B\rightarrow C))\rightarrow((A\rightarrow B)\rightarrow(A\rightarrow C)))=0.9\neq1$。由系统 $\mathscr{D}$ 的可靠性可知，$(L2)$不是 $\mathscr{D}$ 的重言式，从而不是 $\mathscr{D}$ 的定理。

由定义1.1的 H-赋值可知，当赋值区间[0,1]退化为{0,1}时，H-赋值退化为经典逻辑的真值表。且由命题1.2可知，理想状态下的系统 $\mathscr{D}$ 中命题联结词之间的关系与经典逻辑中命题联结词之间的关系完全相同，从而形式系统 $\mathscr{D}$ 中的定理都是形式系统 $\mathscr{L}$ 的定理，所以系统 $\mathscr{D}$ 包含经典逻辑形式系统 $\mathscr{L}$ 作为其特例。

参 考 文 献

1 Zadeh L A. Fuzzy Sets. Inform Contr, 1965, 8: 338～353

2 Zadeh L A. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans SMC, 1973, 1: 28～44

3 王国俊. Fuzzy 命题演算的一种形式演绎系统. 科学通报, 1997, 42(10): 1041～1045

4 王国俊. 修正的 Kleene 系统中的 Σ－(α－重言式)理论. 中国科学, E 辑, 1998, 28(2): 146～152

5 王国俊. 模糊推理的全蕴含三 I 算法. 中国科学, E 辑, 1999, 29(1): 43～53

6 Wang G J. On the logic foundation of fuzzy reasoning. Information Science, 1999, 177: 47～88

7 Pei D W, Wang G J. The completeness and applications of the formal system L*. Science in China (Series F), 2002, 45(1): 40～50

8 王国俊. 非经典数理逻辑与近似推理. 北京: 科学出版社, 2000

9 何华灿, 刘永怀, 何大庆. 经验性思维中的泛逻辑. 中国科学, E 辑, 1996, 26: 72～78

10 何华灿. 泛逻辑学原理. 北京: 科学出版社, 2001

11 张小红, 何华灿, 李伟华. 泛逻辑的基本形式演绎系统 UL 及其可靠性. 计算机科学, 待发表

(上接第88页)

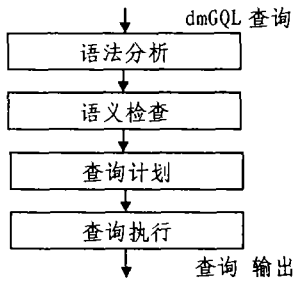

图3 dmGQL 查询处理过程

3. 查询计划，与一般 SQL 语言中的查询计划相比，dmGQL 语言中的查询计划要简单得多，没有复杂的逻辑查询计划优化过程。dmGQL 中的查询计划主要有两个任务，一个是从输入的查询语句中提出属性相关的约束等、度量值相关约束如，以第3节中的 dmGQL 语句为例，areaType＝＊等即是属性相关的约束，COUNT(＊)＞1000、AVG(salesCereal)＜20、DeltaAVG(salesCereal)＞1.5、DeltaCOUNT(＊)＞0.5即为度量值相关的约束，在 dmGQL 中我们只考虑数据库中常用的聚集函数如 MIN, MAX, AVG, SUM, COUNT 等；查询计划另一个任务就是通过检查 FROM 字句来确定查询类型，如果 FROM 字句中的是关系表名，则为非实例化查询，否则如果是数据立方名则为实例化查询。

4. 执行计划，根据查询类型调用梯度挖掘算法进行梯度挖掘，如果是非实例化查询，则调用 eLiveSet 算法，如果是实例化查询则调用 GSCC 算法。最后，输出梯度查询结果。

4.3 基于 DM 的实现策略

dmGQL 的查询处理过程跟标准 SQL 语言的查询处理过程类似，而达梦(dm)数据库管理系统中已经实现了标准 SQL 查询语言，因此我们可以在 dm 的基础上实现 dmGQL 语言。对于 dmGQL 中的语法分析、语义检查模块可以直接扩展 dm 中查询处理的相应部分实现，由于 dmGQL 中目前只考虑了单个表和单个数据立方的情况，因此其语法分析和语义检查模块的实现过程将会更加简单。dm 查询处理中的查询计划还牵涉到查询树转换和等价代数变换等比较复杂的处理逻辑，这些内容在 dmGQL 中是不需要的，因此我们要采用新的查询计划模块来处理 dmGQL 语句。虽然 dmGQL 中的查询执行采用了全新的算法，在物理实现的底层，我们还是需要调用 dm 的 API 来访问存放在 dm 中的关系表数据。为了提高数据立方梯度的查询，我们可以考虑专门对浓缩数据立方建立索引，不过，浓缩立方索引问题还是一个需要进一步深入研究的开(opening)问题。另外，在非实例化查询中，我们需要从关系表中计算浓缩数据立方，计算完一个浓缩数据立方以后，我们需要修改系统的元数据存入新立方的元数据信息。

小结 DBMS 尤其是关系型 DBMS 之所以取得极大的成功，标准 SQL 语言起到了重要作用，我们相信第二代数据挖掘系统要想取得成功，数据挖掘查询语言的作用同样不可忽视。本文我们研究了数据立方梯度查询语言，指出了现有梯度查询语言 CubegradeQL 只支持非实例化数据立方中的梯度查询的不足并且提出了一种新的梯度查询语言 dmGQL 语言，dmGQL 能够支持实例化/非实例化数据立方中的梯度查询。最后，我们讨论了 dmGQL 语言的查询处理过程。

参 考 文 献

1 Imielinski T, Mannila H. A database perspective on knowledge discovery. Communications of the ACM, 1996, 39(11): 58～64

2 Agrawal R, Imielinski T, Swami A. Mining associations rules between sets of items in large databases. In: Proc. of ACM SIGMOD Conf. on Management of Data (SIGMOD'93), Washington D. C., 1993. 207～216

3 Imielinski T, Khachiyan L, Abdulghani A. Cubegrades: Generalizing Association Rules. Data Mining and Knowledge Discovery, 2002, 6(6): 219～257

4 Gray J, Bosworth A, Layman A, Pirahesh H. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In: Proc. of Intl. Conf. on Data Engineering (ICDE'96), Washington, DC, USA. 1996. 152～159

5 Wang W, Feng J, Lu H, Jeffrey X Y. Condensed Cube: an effective approach to reducing data cube size. In: Proc. of Intl. Conf. on Data Engineering (ICDE'02). Washington, DC, USA. 2002. 155～165

6 冯玉才, 刘玉葆, 冯剑琳. 浓缩数据立方中约束立方梯度的挖掘(已录用). 软件学报, 2003