Dwarf 尺寸的进一步缩减*)

向隆刚　龚健雅

(武汉大学 测绘遥感信息工程国家重点实验室　武汉 430079)

摘　要　Dwarf 不仅降低了数据立方的存储开销，而且具有结构简单、易于实现、查询和维护等优点，是一种比较理想的数据立方组织方法。为了进一步缩减 Dwarf 的存储尺寸，本文通过研究 Dwarf 结构，分别提出了浓缩 Dwarf 和冰山 Dwarf：前者从 Dwarf 结构中删除了对于查询来说冗余的内容，而后者从 Dwarf 结构中去掉了对于用户来说琐碎的内容。实验和分析表明，浓缩 Dwarf 有效地减小了 Dwarf 的存储尺寸，而冰山 Dwarf 适合于忽略细节的应用场合，极大地降低了 Dwarf 的存储开销。

关键词　数据立方，Dwarf，浓缩 Dwarf，冰山 Dwarf

Further Compression of Dwarf Size

XIANG Long-Gang　GONG Jian-Ya

(State Key Laboratory of Information Engineer in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079)

Abstract　Dwarf is an appropriate way for data cube store because it not only reduces the storage size, but also has a simple structure and is easy to be queried and maintained. For further compression of Dwarf, we proposes Condensed Dwarf and Iceberg Dwarf respectively, the former deletes from Dwarf structure redundant store, while the latter deletes from Dwarf structure trivial store. Our experiments and analysis show that Condensed Dwarf reduces the storage size of Dwarf effectively, while Iceberg Dwarf works well in detail-overlooked situation, and it can reduce the storage size of Dwarf significantly in such cases.

Keywords　Data cube, Dwarf, Condensed dwarf, Iceberg dwarf

1　引言

Jim Gray 博士等人在 ICDE 大会上首次提出了数据立方的计算问题[1]，并以 Cube by 操作表示计算数据立方的算子。目前，Cube by 操作已经成为 SQL99 标准的一部分。Cube by 操作是传统的分组操作(Group by)的多维扩展，用于计算 Cube by 子句中各属性的所有组合对应的分组。假设基表 R 有 3 个属性 A、B 和 C，那么，Cube by A，B，C 将计算出 8 个分组：ABC、AB、AC、BC、A、B、C 和 ALL(即属性为空的分组)。在联机分析处理领域中，基表被称作事实表(Fact Table)，属性被称作维(Dimension)，分组被称作小方(Cuboid)。

经过计算之后，一个数据立方包含的元组数目将急剧增加。例如在气象数据集[2]中，当维数目取 9 时，事实表只有 1,013,567 条元组，而计算结果包含的元组多达 216,847,087 条，增长了近 214 倍！数据立方的巨大尺寸不仅导致长时间的计算，而且要求巨大的存储空间。

为了解决数据立方的尺寸问题，学者们提出了多种有效的立方技术，如 Condensed Cube[3]、Wavelet Cube[4]、Dwarf[5]、Quotient Cube[6] 和 QC-tree[7] 等。在这些技术之中，Dwarf 结构简单，易于实现、查询和维护，同时具有极高的浓缩率，因而成为数据立方领域的研究热点之一。在气象数据集中，当维数目取 9 时，相应的 Dwarf 占用的存储空间约 280M，不到原始存储方式的 1/16！

尽管 Dwarf 大大减小了数据立方的存储尺寸，但是，它仍有被进一步缩减的空间。本文从两个方面来考虑这一问题：(1)消除冗余的存储；(2)避免琐碎的存储。为此，我们分别提出了浓缩 Dwarf 和冰山 Dwarf，前者从 Dwarf 结构中删除了冗余的“*”项，而后者从 Dwarf 结构中删除了琐碎的 SubDwarfs。实验和分析表明，浓缩 Dwarf 进一步减小了 Dwarf 的存储尺寸，而冰山 Dwarf 在应用支持时，可以显著地降低 Dwarf 的存储开销。

2　Dwarf 结构

Dwarf 是一种高度浓缩的类似于树的数据结构，它基于这样一种事实：立方元组之间存在前缀冗余(Prefix redundancy)和后缀冗余(Suffix redundancy)，在稠密的数据立方中，频繁出现的是前缀冗余，而在稀疏的数据立方中，更普遍的是后缀冗余。通过发现并消除立方元组之间的前后缀冗余，Dwarf 有效地减小了数据立方的存储尺寸。在 Dwarf 结构中，每一条立方元组被表示为一条路径，相同的前缀被共享，仅仅存储一次，相同的后缀被收拢，仅仅存储一次，这使得 Dwarf 看起来像一个两头小中间大的纺锤。

对于表 1 所示的事实表，当聚集函数是 SUM()时(除非特别指出，本文使用 SUM()作为聚集函数)，相应的 Dwarf 见图 1 所示，其中，节点旁数字表示节点的后序编号，“*”表示同维中任意值匹配的一个特殊值，虚线箭头表示后缀收拢。从图 1 所示的 Dwarf 中可以看出，Dwarf 结构有以下三个特点。

*)基金项目：国家九七三重点基础研究发展计划(2006CB701300)资助。向隆刚　博士后，研究方向为数据库技术与空间数据处理；龚健雅　教授，博士生导师，研究方向为地理信息系统与摄影测量技术。

1. Dwarf 是一个非循环有向图(Directed Acyclic Graph, DAG),它有一个唯一的根节点,其它节点可由根节点到达。

2. 一个 Dwarf 有 n 层,其中,n 是相应数据立方的维数目。自顶向下,第一层对应第一个维,第二层对应第二个维,依此类推。

3. 第 n 层节点(叶子节点)的项有如下形式:(key, $aggval$),其中,key 是第 n 个维的值,包括"*",$aggval$ 是相应立方元组的度量值。其它层节点(内部节点)的项有如下形式:(key, $pointer$),其中,key 是相应维的一个值,包括"*",$pointer$ 是指向下一层节点的指针。

表 1 样例事实表

A	B	C	D	M
0	0	0	0	5
1	0	0	1	3
1	1	1	1	4

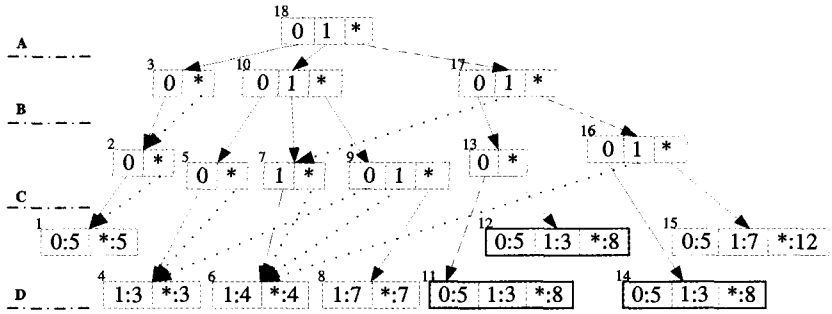


图 1 表 1 所示事实表的 Dwarf

响应查询时,Dwarf 总是从根节点开始,搜索被查询覆盖的所有叶子节点:在第 l 层,检查当前节点包含的项和查询的第 l 维覆盖的值,对于每一个被匹配项,沿着其儿子指针下降到第(l+1)层。例如在图 1 所示 Dwarf 中,回答点查询(*,*,1,*)的 Dwarf 路径是 18(*)→17(*)→16(1)→6(*),因而查询结果为"4"。在任何情况下,一条点查询总是访问 n 个节点,其中,n 是被查询 Dwarf 的层次高度。

3 进一步缩减 Dwarf 的尺寸

Dwarf 是一种相当紧凑的数据立方存储技术,大大降低了数据立方的存储开销,但是,我们研究发现:Dwarf 结构中有些内容仍然是冗余的,从 Dwarf 结构中删除它们,不会影响到 Dwarf 的查询响应能力。另外在许多情况下,应用对细枝末节并不感兴趣,于是,我们可以从 Dwarf 结构中将其删除。

3.1 浓缩 Dwarf

如果 Dwarf 的一个节点仅仅包含两个项,那么,其中一个是普通项,另外一个是"*"项,这一特征是由 Dwarf 结构的定义决定的。在这种情况下,我们完全可以从该节点中删除其中的"*"项,其原因是这两个项指向同一棵 sub-Dwarf。浓缩 Dwarf(Condensed Dwarf)即是从 Dwarf 中删除那些不必要的"*"项之后得到的一种数据结构。对于表 1 所示的事实表,相应的浓缩 Dwarf 见图 2 所示。将一棵 Dwarf 转换成相应的浓缩 Dwarf 是简单直观的:遍历 Dwarf,并按上述方法处理每一个节点。我们以图 1 所示的 Dwarf 为例,节点 3 包括两个项:"0"和"*",因此,在相应的浓缩 Dwarf(图 2)中,项"*"被删除了,其它节点的处理与此类似。

需要指出的是,浓缩 Dwarf 仍然是一种完全且精确的数据立方存储。虽然在浓缩 Dwarf 中,某些节点缺少"*"项,但是,它并没有因此损失响应能力。浓缩 Dwarf 的查询响应和 Dwarf 的查询响应稍有不同,即当一个节点仅仅包含一个项时,该项除了匹配自身之外,还可以匹配"*"值。例如在图 2 所示的 Dwarf 中,点查询(*,0,*,1)是可回答的,对应的路径是 16(*)→15(0)→12(0)→11(1),而查询(*,0,1,1)是不可回答的,其原因是节点 12 只有一个项,可匹配的值只能是"0"和"*"。

显然,浓缩 Dwarf 进一步浓缩了 Dwarf 的存储尺寸,而这种浓缩效果在实际数据集中表现得更加明显,这是因为实际数据集往往是稀疏的,并且各个维之间存在相关性,导致 Dwarf 中出现大量仅含两项的节点,其中的"*"项不会出现在相应的浓缩 Dwarf 中。

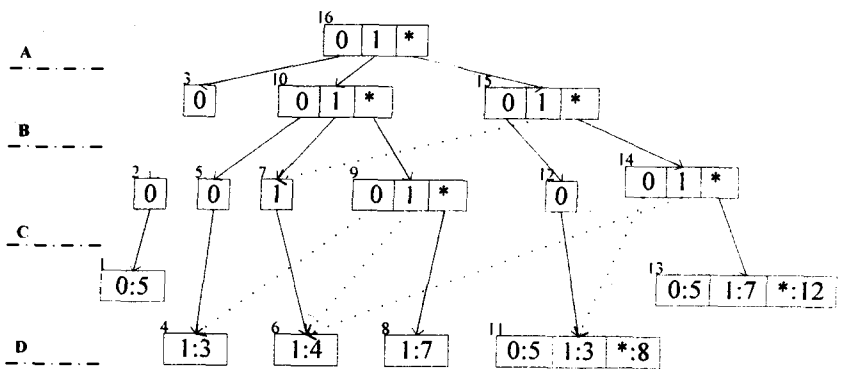


图 2 表 1 所示事实表的浓缩 Dwarf

3.2 冰山 Dwarf

用户很可能只对那些超过某一阈值的立方元组感兴趣,而很少关心其它的立方元组,这正是冰山立方(Iceberg Cube)[8]的核心思想。冰山立方仅仅保留用户感兴趣的立方元组,由于消除了琐碎的立方元组,冰山立方需要存储的立方元组数目大大减少了。显然,冰山立方损失了部分的响应能

力，即不能回答对于琐碎立方元组发出的查询。

冰山 Dwarf 是从 Dwarf 结构中删除那些琐碎路径之后得到的一种数据结构，它是冰山立方思想和 Dwarf 结构相结合的产物。在现实世界中，我们往往仅对立方元组的某一子集感兴趣，如聚集值超过均值的那些立方元组。在这种情况下，我们可以选择冰山 Dwarf 技术，从而极大地减小 Dwarf 的存储尺寸。例如在一个合成的均匀数据集(9 个维，每个维的基数是 1000，共 1,000,000 条元组)中，所得到的 Dwarf 的存储尺寸约 800M，当最小支持度(Minimum Support)取均值的 3 倍时，所得到的冰山 Dwarf 的存储尺寸只有约 60M，不到相应的 Dwarf 的存储尺寸的 1/13!

对于表 1 所示的事实表，当最小支持度取 5 时，相应的冰山 Dwarf 见图 3 所示。从 Dwarf 结构计算出冰山 Dwarf 结构的方法如下：后序遍历 Dwarf 结构，当一个叶子节点项不满足最小支持度时，删除该节点项，当一个节点为空时，删除该叶子节点，并从其父节点(可能有多个)中删除指向该节点的节点项。我们以图 1 所示的 Dwarf 为例(最小支持度取 5)，节点 4 的两个节点项“1”和项“*”均小于最小支持度，于是在相应的冰山 Dwarf(见图 3 所示)中，节点 4 和节点 5 被删除，节点 9 中的项“0”和节点 10 中的项“0”被删除。

冰山 Dwarf 的查询处理同 Dwarf 完全一样，但是，冰山 Dwarf 并不是一种完全的数据立方存储，因此，当某条查询的查询结果为空时，有两种可能性：或者不存在，或者不满足最小支持度。例如在图 3 所示的冰山 Dwarf 中，查询(1，*，*，1)是可回答的，对应的路径是 12(1)→6(*)→5(*)→4(1)，而查询(1,0,0,1)和查询(1,0,1,1)的结果均为空：前者不满足最小支持度，而后者根本就不存在。需要说明的是，冰山 Dwarf 本身并不能指出查询结果为空源于哪一种可能性。

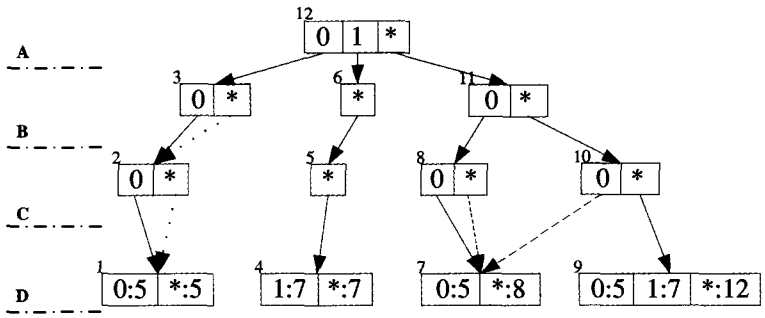


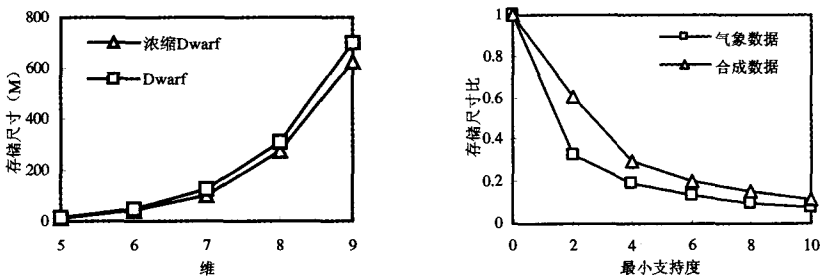
图 3　表 1 所示事实表的冰山 Dwarf(最小支持度取 5)

4　实验与分析

为了验证本文关于 Dwarf 的研究成果，我们分别比较了浓缩 Dwarf 和 Dwarf，冰山 Dwarf 和 Dwarf 在空间开销方面的表现。使用的数据集包括气象数据和合成数据[9]，实验平台是一台 PIV 1.8G 处理器、512MB DDR 主存和 80GB IDE 硬盘的 PC 机，运行 Windows 2000 Advanced Server 操作系统。

4.1　浓缩 Dwarf

本小节比较了浓缩 Dwarf 和 Dwarf 在不同维数目下的存储尺寸。合成数据集的基本参数是：Zipf=1.5，维数目从 5 到 9，每个维的基数是 100，元组数目是 1,000,000。实验结果见图 4 所示。我们可以从中看出，浓缩 Dwarf 进一步浓缩了 Dwarf 的存储尺寸，并且，这种浓缩效果随着维数目的增大而变得更加显著，其原因是高维的数据集将生成更多数量的两项节点。此外，在实际数据集中，浓缩 Dwarf 的存储空间减少量是相当可观的，其原因是在实际的数据集中，维之间往往有较大的相关性。



(a)合成数据　　(b)气象数据

图 4　浓缩 Dwarf 的存储性能

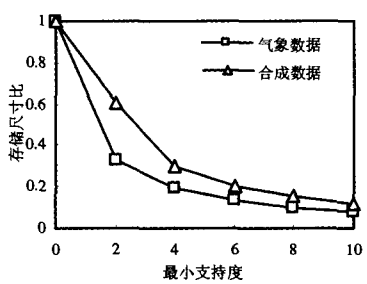


图 5　冰山 Dwarf 的存储性能

4.2　冰山 Dwarf

本小节比较了冰山 Dwarf 和 Dwarf 在不同最小支持度下关于存储尺寸的比率。合成数据集的基本参数是：Zipf=1.5，维数目是 6，每个维的基数是 100，元组数目是 1,000,000。实验结果见图 5 所示，其中，X 轴表示最小支持度同均值(即所有立方元组的度量值的平均值)的倍数。我们可以从中看出，即使设置一个较小的最小支持度，冰山 Dwarf 也能够显著地减小 Dwarf 的存储尺寸。例如在图 5 中，当最小支持度取均值的 2 倍时，对于合成数据和气象数据来说，冰山 Dwarf 的压缩率分别是 39%和 67%。

(下转第 170 页)

$(D_j)-\underline{A}_\beta(D_j)$。

步骤 3. 求 $BN_A^\beta(x)=\{D_j: x\in BN_A^\beta(D_j)(x\in U,j=1,\cdots,r)$。

步骤 4. 求 $D^\beta=(D^\beta(C_i,C_j),i,j\leqslant t)$。其中，

$$D^\beta(C_i,C_j)=\begin{cases}\{a_k\in A: f_k(C_i)\neq f_k(C_j)\}, & (C_i,C_j)\in D^{*\beta},\\ A, & (C_i,C_j)\notin D^{*\beta}.\end{cases}$$

和 $D^{*\beta}=\{([x]_A,[y]_A): BN_A^\beta(x)\neq BN_A^\beta(y)\}$。

步骤 5. 求 β 边界辨识公式 M^β，$M^\beta=\wedge\{\vee\{a_k: a_k\in D^\beta(C_i,C_j)\}: i,j\leqslant t\}=\wedge\{\vee\{a_k: a_k\in D^\beta(C_i,C_j)\}: (C_i,C_j)\in D^{*\beta}\}$。

步骤 6. 化简 M^β，得到 M^β 的极小析取范式形式 $M^\beta=\bigvee_{k=1}^{p}(\bigwedge_{s=1}^{q_k}a_s)$，记 $B_k=\{a_s: s=1,2,\cdots,q_k\}$，则 $B=\{B_k: k=1,2,\cdots,r\}$ 是 β 边界约简形成的集合。

4 B^β-约简应用实例

文[6]中给出了一个目标信息系统，其中 $U=\{x_1,\cdots,x_6\}$ 为论域，$A=\{a_1,\cdots,a_4\}$ 为条件属性集，$D=\{d\}$ 为目标属性集。用它来求 B^β-约简，其中 $\beta=0.3$。

表 1　目标信息系统

U	a_1	a_2	a_3	a_4	d
x_1	1	0	0	0	1
x_2	0	1	1	1	2
x_3	0	1	0	0	2
x_4	0	1	1	0	2
x_5	0	1	0	0	1
x_1	0	1	0	0	1

按照 3 中的 B^β-约简算法得：

步骤 1　$U/R_A=\{C_1,C_2,\cdots,C_4\}$，其中 $C_1=\{x_1\}$，$C_2\{x_2\}$，$C_3=\{x_3,x_5,x_6\}$，$C_4=\{x_4\}$；$U/R_D=\{D_1,D_2\}$，$D_1=\{x_1,x_5,x_6\}$，$D_2=\{x_2,x_3,x_4\}$。

步骤 2　$\mu_{D_1}(x_1|A)=1$，$\mu_{D_1}(x_2|A)=0$，$\mu_{D_1}(x_3|A)=2/3$，$\mu_{D_1}(x_4|A)=0$，$\mu_{D_1}(x_5|A)=2/3$，$\mu_{D_1}(x_6|A)=2/3$，$\mu_{D_2}(x_1|A)=0$，$\mu_{D_2}(x_2|A)=1$，$\mu_{D_2}(x_3|A)=1/3$，$\mu_{D_2}(x_4|A)=1$，$\mu_{D_2}(x_5|A)=1/3$，$\mu_{D_2}(x_6|A)=1/3$。

当 $\beta=0.3$ 时，$\overline{A}_{0.3}(D_1)=\{x_1,x_3,x_5,x_6\}$，$\underline{A}_{0.3}(D_1)=\{x_1\}$，$BN_A^{0.3}(D_1)=\{x_3,x_5,x_6\}$；$\overline{A}_{0.3}(D_2)=\{x_2,x_3,x_4,x_5,x_6\}$，$\underline{A}_{0.3}(D_2)=\{x_2,x_4\}$，$BN_A^{0.3}(D_2)=\{x_3,x_5,x_6\}$。

步骤 3　$BN_A^{0.3}(x_1)=\varnothing$，$BN_A^{0.3}(x_2)=\varnothing$，$BN_A^{0.3}(x_3)=\{D_1,D_2\}$，$BN_A^{0.3}(x_4)=\varnothing$，$BN_A^{0.3}(x_5)=\{D_1,D_2\}$，$BN_A^{0.3}(x_6)=\{D_1,D_2\}$。

步骤 4　$D^{*0.3}=\{(C_1,C_2),(C_2,C_3),(C_3,C_4)\}$，$D^{0.3}(C_1,C_3)=\{a_1,a_2\}$，$D^{0.3}(C_2,C_3)=\{a_3,a_4\}$，$D^{0.3}(C_3,C_4)=\{a_3\}$。目标信息系统的 $\beta=0.3$ 边界可辨识属性矩阵的其它元素全为 A。

步骤 5　$\beta=0.3$ 边界辨识公式 $M^{0.3}=(a_1\vee a_2)\wedge(a_3\vee a_4)\wedge a_3$。

步骤 6　化简得 $M^{0.3}=(a_1\vee a_2)\wedge a_3=(a_1\wedge a_3)\vee(a_2\wedge a_3)$，从而得到 $B=\{\{a_1,a_3\},\{a_2,a_3\}\}$ 是 $\beta=0.3$ 边界约简形成的集合。

结论　在变精度粗糙集模型中，目前已有的属性约简方法并不多，该种保持边界的属性约简方法可以作为已有方法的补充。该方法条理清晰，从理论上证明了它的正确性，从实例上说明了它的可操作性，很适合用计算机编程实现。但是，能否实现复杂的大型数据库的约简还有待进一步的研究。

参 考 文 献

1 Pawlak Z. Rough sets[J]. International journal of Information & Computer Science, 1982, 11(5): 341～356
2 刘清. Rough集及Rough推理[M]. 北京：科学出版社，2003. 40～80
3 Ziarko W. Variable precision rough set model. Journal of computer system science, 1993, 46(1): 39～59
4 陶志，许宝栋，汪定伟，李冉. 基于变精度粗糙集理论的粗糙规则挖掘算法. 信息与控制，2004，33(1)：18～22
5 Beynon M. Reducts within the variable precision rough sets model: A further investigation[J]. European journal of operational research, 2001, 134: 592～605
6 Zhang Wenxiu, Liang Yi, Wu Weizhi. Information System and Knowledge Discoery[M]. Beijing: Science Press, 2003. 56～67
张文修，梁怡，吴伟志，等. 信息系统与知识发现[M]. 北京：科学出版社，2003. 56～67
7 Inuiguchi M. Seeral approaches to attribute reduction in ariable precision rough set mode[A]. In: Proceeding of Modeling Decisions for Artificial Intelligence [C]. MDAI2005, Springer-Verlag, 2005. 523～528
8 Qin Keyun, Pei Zheng, Du Weifeng. The relationship among several knowledge reduction approaches[A]. In: Proceedings of the 2nd international conference on fuzzy systems and knowledge discovery[C]. FSKD2005, ChangSha, 2005.8, Berlin: Spinger, 2005, 1: 1232～1241

(上接第 105 页)

结束语　Dwarf 是一种类树结构，通过共享前缀和收拢后缀，有效地降低了数据立方的存储开销。我们对 Dwarf 结构进行了进一步的研究，指出 Dwarf 结构中仍然存在冗余，为此提出了浓缩 Dwarf，进一步减小了 Dwarf 的存储尺寸。另外，通过将冰山立方思想融合到 Dwarf 技术之中，我们提出了冰山 Dwarf。冰山 Dwarf 适合于用户不太关心细节的应用场合，在这种情况下，冰山 Dwarf 能够极大地减小 Dwarf 的存储开销。

参 考 文 献

1 Gray J, Bosworth A, Layman A, et al. Datacube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, Sub-Totals. In: Proceedings of 12th International Conference on Data Engineering (ICDE'96). Louisiana, USA, 1996. 152～159
2 Hahn C, Warren S, London J. Edited synoptic cloud reports from ships and land stations over the globe, 1982-1991. http://cidiac. est. ornl. gov/ftp/ndp026b/SEP85L. z, 1994
3 Wang W, Feng J, Lu H, et al. Condensed Cube: An Effective Approach to Reducing Data Cube Size. In: Proceedings of 18th International Conference on Data Engineering (ICDE'02). CA, USA, 2002. 155～165
4 Vitter J S, Wang M, Iyer B. Data Cube Approximation and Histograms via Wavelets. In: Proceedings of 7th International Conference on Information and Knowledge Management (CIKM'98). Bethesda, Maryland, USA, 1998. 96～104
5 Sismanis Y, Deligiannakis A, Roussopoulos N, et al. Dwarf: Shrinking the PetaCube. In: Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'02). Madison, Wisconsin, USA, 2002. 464～475
6 Lakshmanan L V S, Pei J, Han J. Quotient Cube: How to Summarize the Semantics of a Data Cube. In: Proceedings of 24th International Conference on Very Large Data Bases (VLDB'02). Hongkong, China, 2002. 766～777
7 Lakshmanan L V S, Pei J, Zhao Y. QC-Trees: An effective Summary structure for Semantic OLAP. In :SIGMOD'03, 2003
8 Fang M, Shivkumar N, Garcia-Molina H, et al. Computing iceberg queries efficiently. In: A. Guppta, O. Shmueli, J. Widom, eds. Proceedings of 24th International Conference on Very Large Data Bases (VLDB'98). New York, USA. Morgan Kaufmann, 1998. 299～310
9 Gray J, Sundaresan P, Englert S. Quickly Generating Billion-Record Synthetic Databases. In: R. T. Snodgrass, M. Winslet, eds. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD'94). Minnesota, USA, 1994. 243～252