SPT—快速计算 FreeCube 的方法

翁 伟[1] 薛永生[2] 文 娟[2] 王劲波[3]
[1](厦门理工学院电子系,福建厦门 361005)
[2](厦门大学计算机科学系,福建厦门 361005)
[3](厦门大学计划统计系,福建厦门 361005)
E-mail: weng_wei5205@sina.com

摘 要 文章首先分析了目前国内外数据立方体计算的研究现状,指出其优缺点。接着在 free-set 的概念上,给出了一系列相关定义,挖掘了 free-set 的性质,建立了 FreeCube 的概念结构。就 FreeCube 的计算而言,充分考虑到 free-set 的性质,结合 BUC 算法的特点,提出了高效的算法 SPT(Selecting-Partition and Trimming Computation of FreeCube),并从多个角度进行了实验,与相关工作做了对比,说明该算法的优越性。

关键词 数据立方体 free-set FreeCube

文章编号 1002-8331(2006)28-0064-04 **文献标识码** A **中图分类号** TP311

SPT—Method of Fast Calculation FreeCube

WENG Wei[1] XUE Yong-sheng[2] WEN Juan[2] WANG Jin-bo[3]
[1](Department of Electronic Engineering, Xiamen University of Technology, Xiamen, Fujian 361005)
[2](Department of Computer Science, Xiamen University, Xiamen, Fujian 361005)
[3](Department of Planning Stat., Xiamen University, Xiamen, Fujian 361005)

Abatract: First of all, the current domestic and international research situation of Data Cube calculation is analyzed. Then the thesis points out its merits and demerits.And it gives out a series of related definitions on the free-set conception, excavates the free-set property and establishes the concept construction of FreeCube.With regard to FreeCube calculation, fully considering the free-set characteristics while combining the characteristics of BUC's calculation, the thesis puts forward an efficient calculation way SPT(Selecting-Partition and Trimming Computation of FreeCube), and while comparing with related work, it engages in experiments from many aspects, which illustrates the superiority of the algorithm.FreeCube maintenance problem is also discussed theoretically.

Keywords: Data Cube, free-set, FreeCube

1 引言

J.Gray1996 年提出 Cube By[1]的概念,定义它为指定属性维集合上全部可能子集合的分组求和。Cube By 操作的结果就形成了数据立方体。目前关于数据立方体计算的研究主要有计算完整数据立方体、部分计算数据立方体和减少数据立方体体积等方面。

(1) 计算完整的数据立方体

目前已经提出了许多用于计算完整数据立方体的算法,主要有 PipeSort[2]、PipeHash[2]、OverLap[2]、ArrayCube[3]和 BUC[4]等。其中,前面 4 种算法或者产生大量的中间结果而要求大容量的主存,或者 I/O 频繁,不适合于计算稀疏的数据立方体。BUC 算法是一种自底而上的计算方法,它的性能优于大多数已经提出的自顶而下的算法,并且特别适合稀疏数据立方体的计算及解决冰山数据立方体问题的时候所需要的修剪。

(2) 部分计算数据立方体

完整的数据立方体的体积是相当大的,有时候比基础表大出数百倍,因此,有必要考虑部分计算数据立方体的问题。

由于 Cube By 计算出来的数据立方体可以看成是许多 Group By 计算出来的结果,我们把每个 Group By 形成的结果称为方体, 部分计算数据立方体就转化为计算哪些方体的问题。论文[5]提出了一种部分计算数据立方体的办法。

(3) 减少数据立方体的体积

最近,研究者发现通过利用共享元组的方法可以有效减少数据立方体的体积[6~9]。Condensed Cube[6]通过发掘单一元组之间的等价关系去除冗余, 减少 Data Cube 体积。Condensed Cube 和 Quotient Cube[7]仍然使用关系表作为存储结构,但是需要附加的信息说明元组之间的联系。

Dwarf[8]对 Data Cube 进行前缀和后缀压缩,减少其体积。QC-tree[9]是在 Quotient Cube 的基础上利用前缀压缩技术,进一步减少 Data Cube 体积。Dwarf 和 QC-tree 都使用了复杂的链表作为存储结构,无法直接用于关系系统。论文[10]为利用 free-set[11]的概念,发掘基本关系表中维值之间的蕴涵规则,提

基金项目:福建省自然基金资助项目(编号:A0310008);福建省高新技术研究开放计划重点资助项目(编号:2003H043)

作者简介:翁伟(1979-),男,硕士,主要研究方向为分布式数据库、数据仓库、数据挖掘等。薛永生(1946-),男,厦门大学计算机科学系教授,主要研究方向为数据库理论与应用、数据仓库、数据挖掘等。文娟(1982-),女,厦门大学计算机科学系硕士研究生,主要研究方向为数据仓库、数据挖掘等。王劲波(1978-),男,博士研究生,主要研究方向为数据库技术,信息管理系统。

出 FreeCube 概念，有效减少了 Data Cube 的体积。

2 相关概念

用 free-set 来保存数据立方体不失为一种好的方法，论文[10]提出了基于 free-set 的数据立方体-FreeCube。在本章中，深入挖掘 free-set 的性质，提出了一种快速计算 FreeCube 的方法 SPT(Selecting-Partition and Trimming Computation of FreeCube)。该方法基于 BUC 算法，但是在对维划分的选择和 free-set 判断上利用了 free-set 的性质去掉了不必要的划分和判断，实验证明该方法大大提高了计算速度。

为了叙述方便，下文假设关系表的模式是 $R(D_1, D_2, \cdots, D_n, M_1, M_2, \cdots, M_m)$，$M_1, M_2, \cdots, M_m$ 为度量属性，并且 d_i 是某一条元组在维属性 D_i 上的分量值，其中 $D_1, D_2, \cdots, D_n$ 为维属性，$n\geq 1, m\geq 1, 1\leq i\leq n$。

定义 1（蕴涵规则）在一个有 n 维属性的基本关系表 R 中，$d_{k1}, d_{k2}, \cdots, d_{ki}, d_{km}$ $(1\leq ki\leq n, 1\leq km\leq n)$ 是某条元组在 $D_{k1}, D_{k2}, \cdots, D_{ki}, D_{km}$ 维属性上对应的分量值，如果关系表中在 $D_{k1}, D_{k2}, \cdots, D_{ki}$ 维属性上对应的分量值为 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 的所有元组在维属性 D_{km} 的分量值都为 d_{km}，则称 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 蕴涵 d_{km}，记为 $(d_{k1}, d_{k2}, \cdots, d_{ki})_R \Rightarrow (d_{km})_R$。不引起矛盾的时候，为了方便可以把 R 去掉。不妨称该蕴涵规则的前一部分 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 称为前件。

例如，在表 1 中，出现 T1 的元组就出现了 S1，所以根据定义有规则 $(T1)\Rightarrow(S1)$；但是却没有规则 $(T1)\Rightarrow(P1)$，因为出现 T1 的元组对应 P 那一维的分量值有些是 P1，有些却是 P2。

表 1 一个简单的事实表

T	S	P	M
T1	S1	P1	10
T1	S1	P2	20
T2	S1	P1	40

定理 1 若 $(d_{k1}, d_{k2}, \cdots, d_{ki})_R \Rightarrow (d_{km})_R$，且 $d'_{k1}, d'_{k2}, \cdots, d'_{kj}$ 是 R 上与 $d_{k1}, d_{k2}, \cdots, d_{ki}, d_{km}$ 不同维属性上的分量值，若某条元组对应维属性上的分量值为 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d'_{k1}, d'_{k2}, \cdots, d'_{kj})$ 则必定有 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d'_{k1}, d'_{k2}, \cdots, d'_{kj})_R \Rightarrow (d_{km})_R$。

该性质可以由定义 1 直接得到。

例如表 1，因为有规则 $(T1)\Rightarrow(S1)$，故也有 $(T1, P1)\Rightarrow(S1)$。

定理 2 若 $(d_{k1}, d_{k2}, \cdots, d_{ki})_R \Rightarrow (d_{km})_R$，给定聚集函数（例如求和函数 SUM）f，则 $f(d_{k1}, d_{k2}, \cdots, d_{ki})_R = f(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{km})_R$，其中 $f(d_{k1}, d_{k2}, \cdots, d_{ki})_R$ 表示在关系表 R 上对含有 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 的所有元组在度量属性上聚集，$f(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{km})_R$ 表示在关系表 R 上含有 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{km})$ 的所有元组的聚集。

例如，在表 1 中，因为有 $(T1)\Rightarrow(S1)$，因此在维属性 T 上值为 T1 的所有元组在度量属性 M 上的聚集值（聚集函数设为 SUM）是 30，而在维属性 T 上的值为 T1 并且在维属性 S 上的值为 S1 的所有元组的聚集值也是 30，也就是有 SUM(T1)=SUM(T1, S1)。

从这个性质我们可以得到启发，因为在数据立方体中有许多分组的聚集值是相同的，因此我们只需要保存其中一个分组的聚集值就可以了，从而达到了节省存储空间的目的。

定义 2（free-set）在一个有 n 个维属性的基本关系表 R 中，$d_{k1}, d_{k2}, \cdots, d_{ki}$ 是某条元组在对应维属性上的分量值，如果对任何含 $d_{k1}, d_{k2}, \cdots, d_{ki}$ 的元组，在不同于 $d_{k1}, d_{k2}, \cdots, d_{ki}$ 所在维的任何一个维属性上的分量值 d_{km}，都不能得到 $(d_{k1}, d_{k2}, \cdots, d_{ki})_R \Rightarrow (d_{km})_R$，那么称 $d_{k1}, d_{k2}, \cdots, d_{ki}$ 是一个 free-set。如果 i=n，则显然 $d_{k1}, d_{k2}, \cdots, d_{ki}$ 是一个 free-set。若一个 free-set 的任何子集都不是 free-set，则称该 free-set 为最小 free-set。

例如，表 1 中 (S1) 是 free-set，(T1, S1) 也是 free-set。

定理 3 若 $d_{k1}, d_{k2}, \cdots, d_{ki}$ 不是一个 free-set，则必是某一蕴涵规则的前件。

证明：用反证法易证。

定理 4 若 $d_{k1}, d_{k2}, \cdots, d_{ki}$ 不是一个 free-set，则有某一个不同于 $d_{k1}, d_{k2}, \cdots, d_{ki}$ 所在维属性的一个维分量值 d_{km}，有规则 $(d_1, d_2, \cdots, d_i)_R \Rightarrow (d_m)_R$，那么对于 $d'_{k1}, d'_{k2}, \cdots, d'_{kj}$，若 $d'_{k1}, d'_{k2}, \cdots, d'_{kj}$ 所对应的维属性与 $d_{k1}, d_{k2}, \cdots, d_{ki}, d_{km}$ 所对应的维属性都不同，则有 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d'_{k1}, d'_{k2}, \cdots, d'_{kj})$ 也不是 free-set。该性质可由定理 1 和定义 2 得到。

定理 5 若有 $(d_{k1}, d_{k2}, \cdots, d_{ki})_R \Rightarrow (d_{km})_R$，则任何包含 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 的 free-set 必然也包含 d_{km}。

这个定理由定理 4 可以直接得到。

定理 6 若 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 不是一个 free-set，则不存在这样的 d_{kp}, d_{kq}，使得 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kp})$ 与 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kq})$ 同时是 free-set。其中，$d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kp}, d_{kq}$ 均是不同维属性上的分量值。

证明：用反证法证明。反设 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kp})$ 与 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kq})$ 同时是 free-set。

此时，因为 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kp})$ 是 free-set，故任意与 $d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kp}$ 所在维属性不同的某一维属性上的值 d_r，蕴涵规则 $(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{kp})\Rightarrow(d_r)$ 不成立，由定理 1 的逆否命题我们可以知道，蕴涵规则 $(d_{k1}, d_{k2}, \cdots, d_{ki})\Rightarrow(d_r)$ 也不成立。但因为 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 不是一个 free-set，由定理 3 可知，$(d_{k1}, d_{k2}, \cdots, d_{ki})\Rightarrow(d_{kp})$ 成立。同理，也有 $(d_{k1}, d_{k2}, \cdots, d_{ki})\Rightarrow(d_{kq})$ 成立。

但是由定理 5，我们知道，包含 $(d_{k1}, d_{k2}, \cdots, d_{ki})$ 的 free-set 应包含 (d_{kp}, d_{kq})，这与反设矛盾。所以原命题成立。

定义 3（Free Cube）利用基本关系表得到所有 free-set，在这些 free-set 上分组聚集产生的元组构成了一个 FreeCube。

例如关系表 1，与之相应的 FreeCube 是表 2，从与表 3 的对照上可以看出，元组数目已经大大减少了。

表 2 FreeCube 示例

T	S	P	SUM(M)
T1	S1	P1	10
T1	S1	P2	20
T2	S1	P1	40
T1	S1	ALL	30
ALL	S1	P1	50
ALL	S1	ALL	70

表 3　关系存储的数据立方体

T	S	P	SUM(M)
T1	ALL	ALL	30
T2	ALL	ALL	40
T1	S1	ALL	30
T2	S1	ALL	40
T1	S1	P1	10
T1	S1	P2	20
T2	S1	P1	40
T1	ALL	P1	10
T1	ALL	P2	20
T2	ALL	P1	40
ALL	S1	ALL	70
ALL	S1	P1	50
ALL	S1	P2	20
ALL	ALL	P1	50
ALL	ALL	P2	20
ALL	ALL	ALL	70

值得注意的是，在论文[10]中有指出，FreeCube 的完整性是指它不是部分数据立方体，而是完整的数据立方体；FreeCube 也不是压缩数据立方体，在回答用户的聚集查询的时候不存在解压缩的过程。

3　SPT—FreeCube 的快速计算算法

计算 FreeCube 的关键是判断 free-set。当然，计算 FreeCube 可以利用现有的算法，例如论文[10]就直接利用 BUC 算法来计算。但是这种方法没有充分利用 free-set 的性质，导致在许多不必要的划分上进行分组、排序和判断。在这一节里，将详细讨论一种快速的优化算法—SPT(Selecting-Partition and Trimming Computation of FreeCube)。其关键有两点：一是选择分组进行判断；二是选择分组进行剪枝。

3.1　划分的概念

先解释一下划分的概念。所谓划分就是对一组元组在一个或者多个维属性上进行归类形成分组。如图 1 所示，设某关系表有四个维属性为 A、B、C 和 D。对该关系表中的元组按 A 维属性划分，把整个关系中的元组分成四个分组：A 维属性上的分量值为 A1 的所有元组为第一个分组；分量值为 A2 的所有元组为第二个分组；同理，A3、A4 也形成一个分组。对第(A1)分组，若接着在第二个维属性上划分，又把第一个分组划分成四个分组(A1B1)、(A1B2)、(A1B3)和(A1B4)。如此下去，划分到最后一个维属性上去的时候，每一条元组都是一个分组。

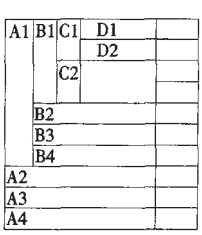

图 1　划分的示意图

在一个分组中，我们把被划分的维属性称为划分维，其它的维属性称为未划分维。如(A1)分组中的划分维是<A>，非划分维是<B, C, D>。

3.2　分组中的 free-set

若把元组集合进行了划分，那么判断 free-set 就只需要在分组中进行。若某个分组$(d_{k1}, d_{k2}, \cdots, d_{ki})$的划分维是$\langle D_{k1}, D_{k2}, \cdots, D_{ki}\rangle$，则该分组中所有元组在该划分维上的分量值都相同，并且根据划分的概念，其它分组的任何元组在维属性$\langle D_{k1}, D_{k2}, \cdots, D_{ki}\rangle$上的分量值不可能等于$(d_{k1}, d_{k2}, \cdots, d_{ki})$，所以我们判断$(d_{k1}, d_{k2}, \cdots, d_{ki})$是否为 free-set，只需要根据 free-set 的定义，在该分组上在对所有的非划分维 D_{km}以及该维上的分量值 d_{km}，若都不存在$(d_{k1}, d_{k2}, \cdots, d_{ki}) \Rightarrow (d_{km})$，则$(d_{k1}, d_{k2}, \cdots, d_{ki})$是 free-set，否则不是。

3.3　计算顺序的考虑

在这里借鉴 BUC 算法的思想，BUC 采用自下而上递归划分的方法，不仅可以获得数据立方体格中所有结点中的分组，而且在划分时实现了共享划分的目的。更重要的是，在这个过程中我们实现了两个关键的优化：选择分组进行判断和剪枝。选择判断是指并非当前的分组都需要判断才知道是否 free-set，可以利用 free-set 的特性略去某些判断；剪枝是指当判断出数据立方体格中某些结点中的某些分组不再能产生新的 free-set 的时候，就不再对其进行进一步的划分，可以略过不必要的分组判断与聚集，相当于对 BUC 处理树进行了剪枝。

3.4　选择分组进行判断

首先考虑一个这样的问题，当判断出了$(d_{k1}, d_{k2}, \cdots, d_{ki})$不是一个 free-set 时，比如有$(d_{k1}, d_{k2}, \cdots, d_{ki}) \Rightarrow (d_{km})$，由定理 3，我们知道$(d_{k1}, d_{k2}, \cdots, d_{ki})$的某个超集一定是 free-set，并且，由定理 5，我们知道该超集必然包括了 d_{km}。因此，对分组$(d_{k1}, d_{k2}, \cdots, d_{ki})$进行进一步的划分所形成的分组中，只有这些分组的划分维包含 d_{km}所对应的维属性 D_{km}时，才需要判断其是否含有 free-set。要注意的是，对分组$(d_{k1}, d_{k2}, \cdots, d_{ki})$在不同与 D_{km}的维属性上进行划分所形成的分组是肯定找不出 free-set 的，这就是我们选择分组进行判断的思想，可以略去很多不必要的判断。

对应表 1，比如在分组(T1)中我们判断出了$(T1) \Rightarrow (S1)$，那么若对分组(T1)在 P 维属性上进行划分形成分组(T1, P1)，判断(T1, P1)是否含有 free-set 是没有比要的，因为根据定理 5 就可以判断(T1, P1)不是 free-set，无须在(T1, P1)分组中去判断。

3.5　对分组进行剪枝

选择划分与 BUC 的计算顺序和 free-set 的性质有关。设一有 n 维属性 $D_1, D_2, \cdots, D_n$的关系表，进行 BUC 划分时先对 D_1进行划分，再对 D_2进行划分，依此类推。那么当判断出了某分组$(d_{k1}, d_{k2}, \cdots, d_{ki})$不含 free-set，不妨设$(d_{k1}, d_{k2}, \cdots, d_{ki}) \Rightarrow (d_{km})$，若 $d_{km} < d_{ki}$，则根据 BUC 划分的顺序，有 $k1 < k2 < \cdots < ki$。由于对$(d_{k1}, d_{k2}, \cdots, d_{ki})$继续进行划分下去不会划分到 d_{km}对应的维属性上去，根据定理 4，产生不了新的 free-set，并且包含$(d_{k1}, d_{k2}, \cdots, d_{ki}, d_{km})$的 free-set 已经产生过，因此没必要对分组$(d_{k1}, d_{k2}, \cdots, d_{ki})$继续划分下去，直接剪枝。

3.6　SPT 算法描述

本算法的输入 inputs 表示一组记录的集合；dim 表示将对记录 inputs 在第 dim 维属性上划分；unpartitionDimSet 表示当前分组的未划分维；postfixDim 表示生成该分组的分组中蕴涵规则的后件。算法的输出是 FreeCube；writeFreeSet 是用于将当

前的 free-set 写入 FreeCube 的函数；函数 isFreeset 用于判断当前的分组是否有 free-set，是的话返回 0，否则返回蕴涵规则的后件标号；函数 Cardinality 用于收集划分信息，这些信息保存在数组 dataCount 中；函数 Partition 根据 dataCount 中的的划分信息对当前分组继续划分。

算法 1 SPT 算法：

```
Procedure SPT (inputs[lower, upper], dim, unpartitionDimSet, postfixDim)
(1) If upper-lower+1= =1 then //单元组划分
(2)    writeFreeSet(inputs[lower, upper], FreeCube);
(3)    return;
(4) end if
(5) if postfixDim is not in unpartitionDimset then
(6)    k=isFreeset(inputs[lower, upper], unpartitionDimSet);
(7) else
(8)    k=-1;
(9) end if
(10) if k=0 then
(11) writeFreeSet(inputs[lower, upper]);
(12) end if
(13) if k=0 or k>=dim or k=-1 then //k 为其它情况直接剪枝
(14) for d=dim to numDims do //逐维划分//下去
(15)     Cardinality(inputs[lower, upper], d, dataCount);
(16)     Partition(inputs[lower, upper], d, dataCount); //划分
(17)     c=lower;
(18)     for i=1 to ubound(dataCount)//
(19)       if(k>=dim) then
(20)       SPT(inputs[c, c+dataCount(i).ds-1], d+1, unpartition
             Dimset-{d}, k);
(21)       else
(22)       SPT(inputs[c, c+dataCount(i).ds-1], d+1, unpartition
             Dimset-{d}, -1);
(23)       End if
(24)       c=c+dataCount(i).ds;
(25)     End for
(26)   End for
(27) End if
```

4 实验及其性能分析

为了验证 FreeCube 的存储效率和本章提出的算法的时间效率，我们做了三组实验。FreeCube 的存储效率是用其记录条数与 Data Cube[1]的记录条数相比较得出的，其时间效率是与 FreeCube[10]算法比较得出的。实验条件是奔腾 CPU 4 1.6GHz，内存 518M，操作系统是 Windows 2000 Advance server，所用数据是随机生成的记录，记录数（数据量）精确到万，时间单位是 s。

表 4 表明了数据维属性数目对存储空间和算法时间的影响。随着维属性数目的增加，Data Cube 记录数急剧增加，FreeCube 记录数也会增加，不过趋势较缓。维属性数目增加，会导致数据立方体格中的增多，并且是呈指数增加，所以导致了 Data Cube 和 FreeCube 的记录数目大量增多。从 Data Cube 记录数和 FreeCube 记录数的对照来看，FreeCube 确实能大大减少数据立方体的存储空间。从时间上来看，SPT 算法总比 FreeCube 算法速度快，并且随着维属性的增多，这种速度上的差异越来越明显。

表 4 维属性数变动的情况（数据量 10 万条，维定义域）

维属性个数	DataCube 记录数/万	FreeCube 记录数/万	FreeCube 算法时间/s	SPT 算法时间/s
2	6	1	7	7
4	86	8	31	29
8	2 191	53	540	460

表 5 说明了数据量变化时对数据立方体存储空间和算法速度的影响。当数据量增多的时候，使得基本表的数据密度增大，也会使得 Data Cube 的记录数目增加。FreeCube 增长缓慢，并且大大少于 Data Cube 记录数。在这种情况下，SPT 算法的速度仍然比 FreeCube 算法快，并且差异明显，不过这种差异不一定随着基本表数量的增多而增大，这是因为这两种算法的速度还受到数据分布的影响。

表 5 数据量变动的情况（维属性数为 8，维定义域 100）

数量/万	DataCube 记录数/万	FreeCube 记录数/万	FreeCube 算法时间/s	SPT 算法时间/s
5	1 116	33	69	68
10	2 191	53	540	460
15	3 254	82	940	900

表 6 说明了维属性定义域变化的情况。当维属性定义域增加的时候，会导致数据越来越稀疏，所以 Data Cube 记录数和 FreeCube 记录数都会增加，同时 FreeCube 记录数比 Data Cube 记录数少得多。在时间上来说，SPT 算法比 FreeCube 明显快，不过这种差别也不一定随着维属性定义域的增加而增大，同样是因为与数据分布有关。

表 6 维定义域变动的情况（数据量为 10 万，维属性数为 8）

维定义数	DataCube 记录数/万	FreeCube 记录数/万	FreeCube 算法时间/s	SPT 算法时间/s
100	2 191	53	540	460
200	2 290	620	960	880
300	2 359	544	860	840

5 结语

本文重点描述了 FreeCube 概念和理论，提出了高效的算法—SPT，用实验说明了其时空效率。对于 FreeCube 的概念，本文首先严格描述 free-set 的定义，证明了其一系列的性质。然后利用 BUC 算法的思想，实施选择分组判断和剪枝这两个关键的优化，使得 FreeCube 的计算效率大大提高。

关于 FreeCube 还有许多地方值得研究，下一步的工作主要集中在以下几个方面：(1) 如何通过 FreeCube 重新恢复历史数据，比如在分布式环境中，个别场地上需要历史数据，而从中央数据库传输大量数据代价高，或是历史数据丢失，就有这种需求；(2) 设定支持度和致信度，通过研究 FreeCube 如何识别出其中的关联规则；(3) 如何定义一种高效的 FreeCube 存储结构以提高 FreeCube 响应用户查询的速度；(4) 若考虑维的层次，如何高效地计算 FreeCube。（收稿日期：2005 年 12 月）

参考文献

1.Graefe G.Query evaluation techniques for large databases[J].ACM Computing Surveys,1993;25(2):73~130
2.Jiang XD,Zhou LZ.A new aggregation algorithm for OLAP query evaluation[J].Journal of software,2002;13(1):65~70
3.Y Zhao,P M Deshpaned,Jeffrey F Naughton.An Array-Based Algo-

（下转 190 页）

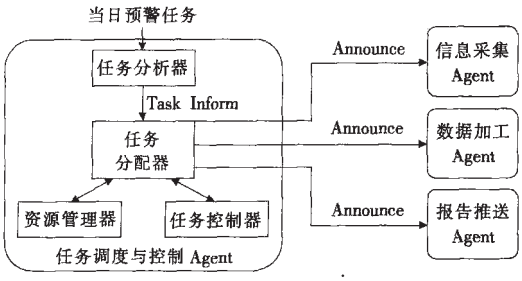

图 4 排程本体在 TBT 预警中的应用

务”，向任务分配器发送任务通知和消息，从“当日预警任务”的语义标注中读取数据并写入其消息内容：Task Inform Message (Task Number, Item, Plan, Time, Condition)；其中：

Task Number: 任务编号；

Item: 任务内容；

Plan: 任务执行计划；

Time: 任务执行时间；

Condition: 约束条件。

(2) 任务分配器接到任务通知后，首先与任务控制器和资源管理器交互，如果这时没有任务在执行无须排队，并且资源都够用，则任务分配器根据任务要求向各个 Agent 发出执行指令：Announce Message(Serial, Content, Start, Resource)；其中：

Serial: 工序号；

Content: 执行的内容；

Start: 工序开始时间；

Resource: 工序需要调用的资源。

工序号是系统自动生成，其他三项内容从计划排程本体中读取。

我们以预警监测美国半导体行业为例，其流程算法的部分伪代码如下：

```
当日预警任务到来；
When 任务调度与控制 Agent 的任务分析器空闲
{
分析“当日预警任务”，得知需要：采集数据、生成一份预警报告并发给推送系统；
Task Inform Message→任务分配器；
  If 任务分配器不空闲 Then
{waiting;}
  Else {
分析 Task Inform Message，得知要先调度信息采集 Agent 采集数据，再生成一份预警报告，最后发给数据推送 Agent；
向任务管理器和资源管理器发出请求；
If 当前没有任务队列阻塞 and 资源够分配 Then
{
Announce Message→信息采集 Agent;
Announce Message→数据加工 Agent;
Announce Message→数据推送 Agent;
}
信息采集 Agent 调度网络爬行器，到“采集对象”的语义标注中指定的 URL 地址采集所需的“采集数据项”，并存入数据库；
数据加工 Agent 从数据库中取出采集的数据，调度规则分析器生成预警报告，推送报告；
数据推送 Agent 收到推送通知后，等待从数据加工 Agent 送来的预警报告，并根据个性化原则向不同的专家用户推送报告；
  }
}
```

专家在收到预警报告后，通过网上的专家研讨厅，研究报告的可行性，并将信息反馈给计划部门，由计划部门的专家重新研究预警方案。当日预警任务完成后，预警结果存于计划部门，每个月根据月度计划指标评价这个月的预警任务完成情况，以此类推年度计划的评估。

4 结束语

将生产企业中的敏捷制造思想引进 TBT 预警信息系统的设计模式中，在预警任务到来时，将分散的资源整合起来，形成一种柔性的、敏捷的、能随外界环境变化而迅速动态重构的信息系统。对各个子系统(Agent)模仿企业的计划排程进行科学的规划，优化系统结构和资源分配，最后建立排程的本体模型并且用本体语言表达出来，作为 Agent 之间的通信元语，真正实现智能调度。由于将敏捷制造虚拟企业的思想引进本课题，目前只是做了初步尝试，还有许多具体工作有待深入，例如，如何将敏捷制造中的约束机制、规则算法应用到本系统中，此外建立更加明晰的本体模型也是今后需要研究的方面。TBT 预警系统的建立是目前我国在贸易预警方面的当务之急，系统的智能化是趋势，系统运作的计划排程是智能化首要问题，该研究必将有良好的发展前景和意义。(收稿日期：2006 年 6 月)

参考文献

1.李轩，蒲国蓉.TBT 和中国纺织品出口贸易[J].特区经济，2005;(4)
2.张旭梅，黄河，刘飞.敏捷虚拟企业[M].科学出版社，2003-03
3.GRUBERCTR.A translation approach to portable ontologies[J].Knowledge Acquisition, 1993; 5(2): 199~220
4.郭鸣，李善平，董金祥等.基于本体论及语义 Web 的产品信息模型研究[J].浙江大学学报，2004;(1)

(上接 67 页)

rithm for Simultaneous Multidimensional aggregates[C].In: Proc ACM-SIGMOD Int Conf Management of Data(SIGMOD), 1997: 159~170
4.Bacyer K, Ramakrishnan R.Botton-up Computation of Sparse and Iceberg CUBEs[C].In: Proceedings of ACM SIGMOD International Conference on Management of Data, Philadephia, PennsyIvania, USA: ACM Press, 1999-06: 359~370
5.V Harinarayan, A Rajaraman, J D Ullman.Implementing data cubes efficiently[C].In: Proc of the 1996 ACM SIGMOD Int'l Conf on Management of Data(SIGMOD'96), New York: ACM Press, 1996: 359~370
6.Wang W, Lu H, Feng J et al.Condensed Cube: an effective approach to reducing data cube size[C].In: ICDE, 2002: 155~165
7.Lakshmanan L V S, Pei J, Han J.Quotient cube: How to summarize the Semantics of a data cube[C].In: VLDB, 2002: 778~789
8.Sismanis Y, Deligiannakis A, Roussopoulos N et al.Dwarf: Shrinking the Petacube[C].In: SIGMOD '02, 2002: 464~475
9.Lakshmanan L V S, Pei J, Zhao Y.QC-tree: An efficient summary structure for semantic OLAP[C].In: SIGMOD '03, 2003: 64~75
10.孙延凡，陈红，王珊.FreeCube: 有效减少 Data Cube 体积[J].计算机科学，2003; 30(增刊): 241~245
11.Boulicaut J-F, Bykowski A, Rigotti C.Approximation of frequency queries by means of free-sets[C].In: PKDD'00, 2000: 75~85