一个面向 OLAP 的多维层次聚簇存储模式*)

袁 霖[1] 邹恒明[1] 李战怀[2]

(上海交通大学计算机科学与工程系 上海 200240)[1]

(西北工业大学计算机学院计算机软件与理论系 西安 710072)[2]

摘 要 文献[2]针对 ROLAP 提出的多维层次聚簇存储模式(MHC),极大地提高了查询效率。然而与 ROLAP 相比,MOLAP 往往具有更高的存储效率和查询效率。这让人自然地联想到:如果能构造一个集二者优点为一身的混合型 OLAP 系统,以实现 MHC,也许能进一步提高系统性能。作为这一设想的探索性研究,本文利用 ORDBMS 的可扩展性实现了这一原形系统:多维数据按维层次分块聚簇,其中每个分块以数组 ADT 存储,分块间以 B^+ 树索引聚簇。实验表明,本文提出的 MHC 实现能有效减少存储空间,进一步提高查询性能。

关键词 联机分析处理,层次,聚簇,对象关系数据库

An Efficient Multidimensional Hierarchical Clustering Storage Schema for OLAP

YUAN Lin[1] ZOU Heng-Ming[1] LI Zhan-Huai[2]

(Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200030)[1]

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072)[2]

Abstract ROLAP system can be accelerated by MHC (multidimensional hierarchical clustering) physical layout schema introduced by literature [2], while MOLAP system has advantages over ROLAP by high compression rate and efficiency. It is natural to ask if adapting the MHC idea for a mixed system that is a combination of MOLAP and ROLAP system can further improve the performance. As an initial step toward answering this question, we have implemented this kind of OLAP system that is based on ORDBMS. In our system, the data cube is chunked and clustered in terms of dimensional hierarchies. These chunks are stored as array-ADT and clustered with a B^+-Tree index. The experiments show that our MHC schema can save more storage space and further improve the performance.

Keywords OLAP, Hierarchy, Cluster, ORDBMS

1 引言

随着数据仓库技术的发展,OLAP[1](On-line Analytical Processing)已成为决策支持和商业智能的有力工具之一。与传统的 OLTP 不同,OLAP 要求以多维方式对数据进行分析。作为本文示例,图 1 是某超市有关商品销售数据的二维数据立方。

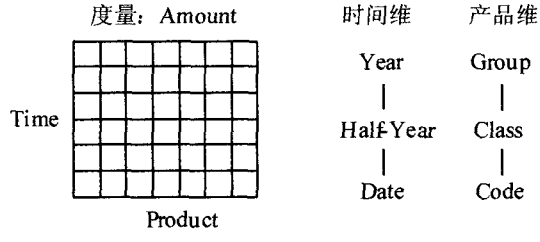


图 1 二维销售数据立方及其维层次

与编程语言中数组的线性维不同,OLAP 中的维具有层次结构,用于表示对数据的不同分类。基于维层次的多维选择操作是 OLAP 中最为重要的一类操作。对于本文示例,典型的多维选择查询如下:

Q:查询家用电器类(Group)中每种商品在 2006 年的日销售量。

聚簇是提高查询效率的有效手段之一,通过将查询所需存取的数据集中存储,能有效降低 I/O 代价。文[2]在 ROLAP 环境中,提出了一个将多维数据按维层次聚簇的存储模式(MHC):首先将多维数据按层次线性化,然后用 B^+ 树索引对其聚簇,以此在 RDBMS 中实现了一个基于一维索引的多维存取方法,能有效提高多维存取效率。

然而与 MOLAP 相比,ROLAP 的存储和查询效率往往不高。如果能构造一个具有 ROLAP 与 MOLAP 各自优点的混合 OLAP 系统,并利用这些优点实现 MHC 存储,也许能进一步提高效率。本文的研究目标就是利用 ORDBMS 所特有的可扩展性,实现这一系统。为此,我们对文[2]提出的 MHC 进行了改造:首先,将多维数据按层次分块,"分块"以 Chunk-Array-ADT 存储,这一方式类似于 MOLAP;其次,分块间按层次线性化,并用 B^+ 树索引聚簇,这正是文[2]提出的基于 ROLAP 的实现方式,但与之不同的是:索引项不再是空间中的点,而是"分块"。初步的实验表明,基于混合型 OLAP 的 MHC,比单纯基于 ROLAP 的实现相比,不仅具有更高的压缩效率,而且具有更高的查询效率。

获得更高压缩率的原因在于如下压缩方法的使用:

(1)块压缩。在每一个"分块"内部可使用 bitmap 或

*)本文研究得到国家自然科学基金(60073055)资助。**袁 霖** 博士后,主要研究领域为数据仓库与 OLAP 技术;**邹恒明** 博士,副教授,研究领域为软件工程;**李战怀** 博士,教授,主要研究领域为数据库理论与技术。

chunk-offset[7]方法压缩，以解决多维数据的稀疏性问题；每一个 chunk 在 ORDBMS 中以大对象存储，其内部实现采用 LZW 压缩算法，进一步对数据进行压缩。

(2)索引压缩：由于索引项是“块”，而不是单独的“点”，使得聚簇索引的规模大大减小。

获得更高查询效率的原因在于：

(1)由于是按层次分块，因此对基于维层次的选择操作而言，可避免对无用数据的读取；

(2)更高的压缩率(包括索引压缩和块压缩)，能有效减少磁盘 I/O 的代价。

2 聚簇模型

就本文而言，建立聚簇模型就是要寻求一个算法，将多维数据线性化，使得多维数据在线性化序列中按维层次分块聚集。我们将结合本文示例，对这一算法进行描述。

2.1 多维数据的线性化方法

在数据仓库之中，多维数据以星型模式存储，例如

时间维表：Time(Month，Half-Year，Year)，

产品维表：Product(Code，Class，Group)，

事实表：Sales(Month，Code，Amount)。

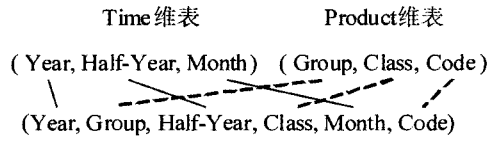
首先，将事实表与维表连接，得到一个“逆规范化的事实表”。其次，将各个维层次中的不同等级按如下规则构造一个复合排序码，并进行排序。

[规则 1]属于同一个维的等级，高等级在左，低等级在右。

[规则 2] 所有的最低等级集中在一起，中间不能有更高的等级出现。

其中，第一条规则确保多维数据是按层次聚簇的，第二条规则保证了多维数据是按层次聚簇的。

对本文示例，其线性化序列如图 2 所示。复合排序码为(Year，Group，Half-Year，Class，Month，Code)，相当于度量(Amount)的线性化标识。如果将其分成前后两部分(Year，Group，Half-Year，Class)、(Month，Code)，(Year，Group，Half-Year，Class)相当于块标识，而(Month，Code)相当于块内标识。



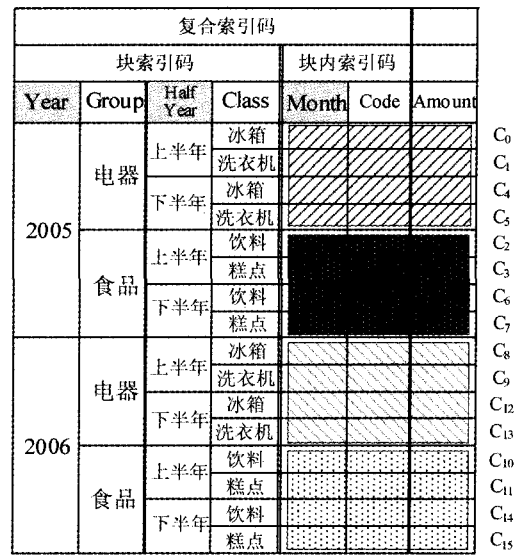


图 2 多维数据的线性化

2.2 聚簇特性分析

上述“多维数据线性化方法”的聚簇特性，如图 3 所示。具有如下特点：

(1)“块”是按层次线划分的，块内是按行线性化的；

(2)多维空间按维层次递归划分，并按同样的方式在线性化序列中聚簇。

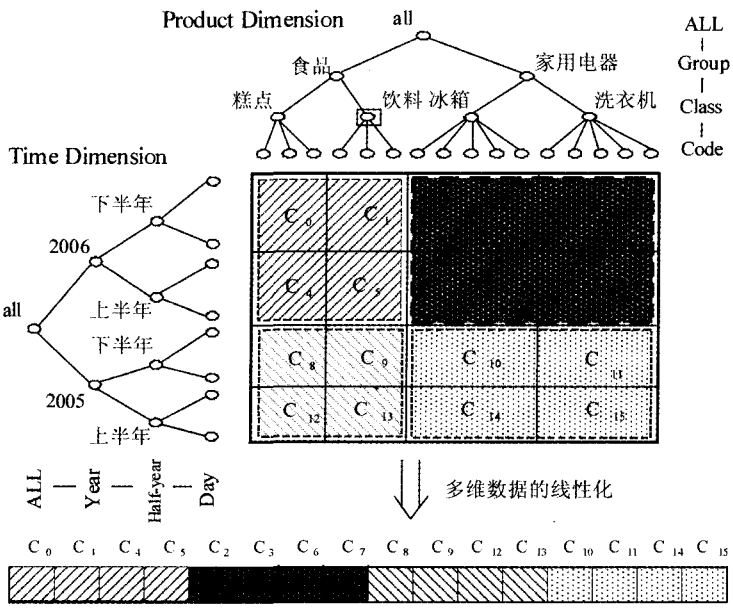


图 3 多维数据的层次分块线性化

图 3 中的多维数据首先按“Year”和“Group”划分为 4 个区域。其中的每一个区域，进一步按照下一等级“Half-Year”和“Class”划分为“块”，同一个区域中的“块”聚簇在线性化序列中聚簇在一起。对于查询 Q 所需读取的数据，在线性化序

列中，集中在 C_2、C_3、C_6、C_7 连续的四个分块之中。

与面向 ROLAP 实现的“多维数据线性化方法”[2] 相比，本文的方法更适合于混合型 OLAP。文[2]使用 Z-order 将多维数据线性化，不能保证对多维数据按层次分块，因此无法实现基于分块的多种压缩方法(详见下一节)。然而，准确地说，这一方法并不是我们的独创，而应该是文[2]方法的一个特例。

需要特别说明的是，我们的线性化方法并不要求所有维等级都参与多维数据的聚簇方法。也就是说，多维数据并不要求按所有的维层次以及维等级进行聚簇，而应该根据具体的工作载荷分布进行优化设计。这一问题将是我们未来的研究方向。

3 实现

3.1 多维数据的压缩

3.1.1 维层次的路径编码压缩

直接以复合排序码作为多维数据的线性化标识，将过于庞大，致使存储效率低下，因此必须对其进行压缩。我们使用与文[2]相同的方法，对维层次进行编码。时间维的编码如图 4 所示。

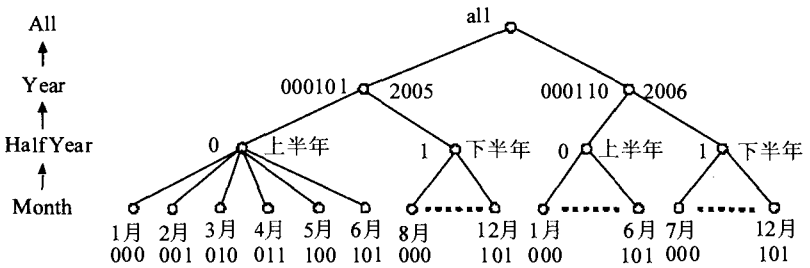


图 4 时间维层次的路径编码

通过编码方式，“复合排序码”可由一个二进制编码 m_idx 取代，作为多维数据的线性化标识，用(m_idx, m)表示。同样，m_idx 也可分为两部分 m_idx = chunk_idx | local_idx，分别为块标识和块内标识。

3.1.2 块压缩

我们没有直接使用 ORDBMS 固有的 Array 类型[4] 表示“分块”，是因为它无法解决多维数据的稀疏性问题。为此，我们重新定义了一个抽象数据类型 Chunk-Array-ADT，用于表示“分块”，其内部使用 Bitmap 和 offset-data[5] 两种方法对“分块”压缩，并根据二者的存储代价动态选择压缩算法，这是面向 OLAP 的语义压缩。压缩后的分块以大对象方式存储。

3.2 多维数据的存储

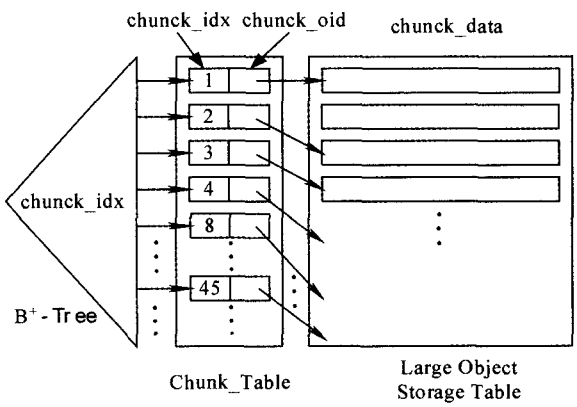


图 5 Chunk_Table 在 ORDBMS 中的物理存储

在 ORDBMS 中，由“块”构成的线性化序列按如下方式存储：

Chunk_Table(chunk_idx Bitstring, chunk Chunk-Array-ADT)

内部实现如图 5 所示。通过在 chunk_idx 上建立 B^+ 树主索引，对“块”序列进行聚簇。

在 ORDBMS 内部，chunk 作为大对象[3] 被进一步分成若干个更小的块，用特定的算法(如 LZW)压缩后，存在另外一个数据文件中，仅将对象标识保留在 Chunk_table 中。

4 查询处理

文[2]提出了一个 Tetris 算法，用于高效地实现基于一维 B^+ 树聚簇索引的多维存取操作。虽然这一算法是基于 ROLAP 的，但对于以“分块”的索引项的聚簇方式而言，仍然是有效的。所不同的是查询处理需要分成两个部分：

(1)对定义在 chunk_idx 的 B^+ 树索引，使用 Tetris 算法实现“块”查找；

(2)根据块内的局部坐标 local_idx，进行块内查找。这是由定义在 Chunk-Array-ADT 之中的操作函数实现的。

5 实验

实验目的是比较以下三种存储模式的存储效率和查询效率：

(1)星型模式 StarSchema；

(2)基于 ROLAP 的非分块聚簇模式 MidxTable；

(3)基于混合型 OLAP 的分块聚簇模式 ChunkedTable。

实验平台为对象关系型的 PostgreSQL[6]，操作系统为 Linux Redhat9. 0，内存 512M，其中排序内存 8M，赛扬 900CPU(奔腾 III 系列)，硬盘为 Seagate120G、7200rpm、2M cache、ATA66 接口。

实验使用 OLAP 委员会发布的 APB-1R2 Benchmark 数据集，是关于某连锁超市的销售数据，包含四个维(Product, Custom, Channel, Time)和两个度量单元(Units Sold, Dollar Sales)，以星型模式表示如下。

事实表：(code, store, base, month, Units Sold, Dollar Sales)

各个维等级及其编码长度如下：

Product: ALL ← division ← line ← family ← group ← class ← code

编码长度: 2 2 3 2 2 4

Custom: ALL ← retailer ← store

编码长度: 7 4

Channel: ALL ← Base

编码长度： 4

Time:ALL ← year ← Qtr ← month

编码长度： 1 2 2

按层次线性化之后的多维数据表示如下：

division, retailer/7, year, line, quarter, family, group, class, code, store/4, base, *month*, unitssold, dollarsales

5.1 存储效率的比较

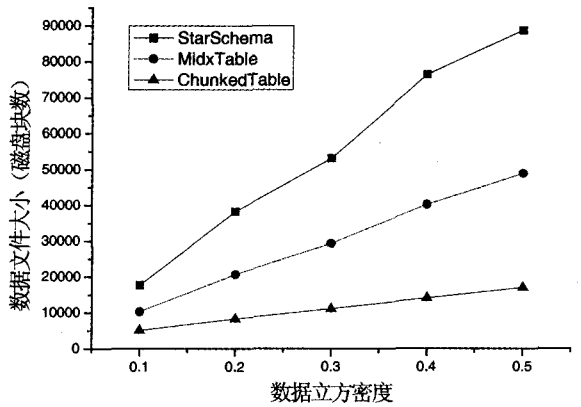


图 6 多维数据不同存储模式的数据文件大小的比较

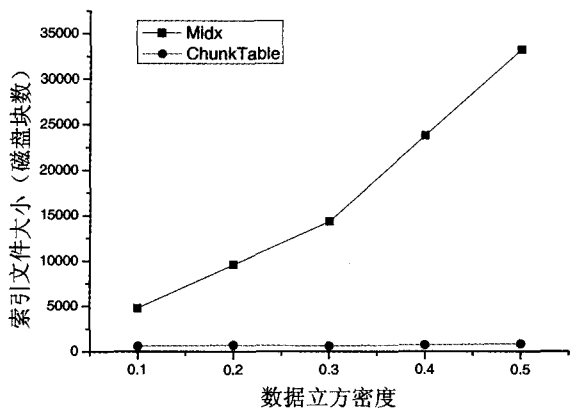


图 7 不同存储模式的索引文件大小的比较

从图 6 可以看出，ChunkedTable 的数据文件大小不到 MidxTable 的 1/2；从图 7 可以看出，ChunkTable 的索引文件大小至少是 MidxTable 的 1/8，而数据密度为 0.5 时，为 1/40。可见我们提出的聚簇策略的存储效率是相当高的。

5.2 多维存取效率的比较

对于星型模式而言，基于维层次的多维选择查询，可用如下 SQL 表示：

```
Select f. * FROM actvars as f, custlevel as c, prodlevel
    as p, timelevel as t
WHERE c. store = f. customer AND p. code = f. prod-
    uct AND t. month = f. time
AND predicts_on_ hierarchies;
```

其中 predicts_on_ hierarchies 是作用在维层次上的谓词表达式。

对于 MidxTable 存储模式，这一查询经过重写，转换成基于线性化标识 M_idx 上的索引扫描；对于 ChunkTable 存储模式，则转换成基于 chunk_idx 的索引扫描和基于 local_idx 的块内查找。

由于选择谓词 *predicts_on_ hierarchies* 所引用的维等级不同，导致其在复合排序码中所占据的不同位置也有所不同，使查询分成如下三种类型：

(1)前缀查询 Q1

predicts_on_ hierarchies: c. retailer = ? AND t. year = ? AND P. line = ?;

division, retailer, year, line, quarter, family, group, class, code, store, base, month, unitssold, dollarsales

(2)部分前缀查询 Q2

predicts_on_ hierarchies: t. Year = ? AND P. line = ?;

division, retailer, year, line, quarter, family, group, class, code, store, base, month, unitssold, dollarsales

(3)非前缀查询 Q3

predicts_on_ hierarchies: c. Retailer = ? AND t. Year = ?;

division, retailer, year, line, quarter, family, group, class, code, store, base, month, unitssold, dollarsales

在以上谓词表达式中，符号“?”代表随机选取的维成员。对每一个具体的存储模式，每个查询运行 30 次。由于聚簇索引所固有的特性，以上三类查询的效率是不同的。在现实应用中，这三类查询的载荷分布是不同的。根据载荷分布，优化设计聚簇模式，是一个非常值得研究的问题，但不是本文论述的重点。本实验的目的只是针对每类查询比较它们对于不同存储模式的查询效率，实验结果见表 1、2、3。我们根据选择率对测试结果进行了分组。

表 1 前缀查询 Q1

选择率	0		(0—0.3)	
统计参数	Avg	StdDev	Avg	StdDev
Star Schema	80178.5	32609.4	76366.0	9808.4
MidxTable	32.2	52.8	313.2	130.7
ChunkedTable	29.4	43.0	150.3	36.6

表 2 部分前缀查询 Q2

选择率	0		(0—0.1)		(0.1—0.2)	
统计参数	Avg	StdDev	Avg	StdDev	Avg	StdDev
Star Schema	93182.5	29935.1	145125.0	387777.9	182483.2	46095.6
MidxTable	177.4	257.9	11058	4113.8	23328.1	8235.0
ChunkedTable	156.0	62.6	3096.2	1253.4	6692.5	2013.2

通过对上述实验数据的比较可以发现如下的规律：

(1)选择率为 0，是由于多维数据的稀疏性导致符合条件的度量单元全部为空，这时在 ChunkedTable 上的查询响应速度快于 MidxTable，这是由于索引项是“块”，而非“点”，使得索引高度大大减小。

(2)当选择率大于 0 时，ChunkedTable 查询效率比 MidxTable 提高 3～4 倍。这得益于 ChunkedTable 高效的压

(下转第 124 页)

R_{14}、直接购买商 L_{11} 至 L_{13} 的信息；提取相关的间接供应商 R_{21} 至 R_{28}、间接购买商 L_{21} 至 L_{26} 的信息；……；直至提取到其最终产品的终端零销商信息、初始原料的源头提供商信息为止。②根据所提取到的各级供应商信息、购买商信息，进行构图。图中→代表供应渠道，箭头指向购买方，每条渠道又载有其运输成本费用、产品交易希望价格等相关信息。

结论 本文的研究表明：首先，多 Agent 技术可以应用于电子商务技术的供应链匹配系统架构设计，并提高其性能与品质。其次，给出了一种基于多 Agent 技术的供应链自动匹配管理平台系统模型，阐明了它的整体组成架构及其各单 Agent 的具体功能，阐述了平台系统模型中的两项关键技术——数据挖掘(包括基本任务与任务扩展)与供应链网络图构建，实现了供应链自动动态匹配管理；从而，可达到"消费者需求能得到积极满足，而供应链上不同环节众多经营商能实现互利共赢"的双向目标。

参 考 文 献

1 周启海，张元新，吴红玉. 一种基于多 Agent 的双向智能自动匹配系统模型[J]. 计算机应用，2006(7)
2 魏修建，严建援，王焰. 电子商务物流[M]. 北京：人民邮电出版社，2001
3 魏修建. 电子商务物流管理[M]. 重庆：重庆大学出版社，2004
4 刘业政，李亚飞，杨善林. 电子商务环境下基于移动 Agent 的 Web 数据挖掘[J]. 计算机工程，2004，20：107～109
5 高翔，林杰，张炜. 基于 Agent 的供应链仿真模型设计与实现[J]. 计算机工程与应用，2005，32：183～186

(上接第 113 页)

缩率。

表 3 非前缀查询 Q3

选择率	0		(0—0.3)	
统计参数	0	0	0.022	0.006
Star Schema	62962.8	2894.7	73453.4	2311.7
MidxTable	116.1	99.2	1960.0	596.0
ChunkedTable	31.2	40.4	506.1	143.2

以上实验结果表明，我们提出的基于混合 OLAP 的多维层次分块聚簇策略，能有效提高多维数据的压缩效率和基于维层次的多维选择效率。

6 相关研究

文[4,5]虽然对数据立方进行分块存储，但既不是按维层次分块，也不是按维层次聚簇的，因此对于基于维层次的操作，将不可避免地读取无用的数据单元。文[2]是通过将来自不同维的路径编码进行位交错，将多维数据线性化，然而它不是按层次分块的，而且这一实现方式完全基于关系技术，难以获得更高的压缩率和查询效率。文[7,8]介绍了一个面向 OLAP 的多维聚簇模式，但这一聚簇模式不是按多维"层次"聚簇的。虽然这一聚簇模式对多维数据进行了分块，但此分块是"物理"块，而不是本文介绍的按层次划分的"逻辑"分块。

结论 为提高基于维层次的多维存取效率，本文利用 ORDBMS 所特有的可扩展性，实现了一个同时基于"关系"和"多维"的混合型层次聚簇存储模式。基于 OLAP Benchmark 的初步实验表明，比纯粹基于"关系"技术的层次聚簇模式，具有更好的存储效率和查询效率。进一步取得更为全面的实验数据，以及根据具体的查询载荷设计层次聚簇模式，将是我们下一步的研究工作。

参 考 文 献

1 Pendsen N. What is OLAP? 2003. http://www.olapreport.com/fasmi.htm
2 Markl V, Ramsak F, Bayer R. Improving OLAP performance by multidimensional hierarchical clustering. In: Proc. Int Conf. on Database Engineering and Applications Symp(IDEAS), 1999. 165～177
3 Stonebraker M, Olson M A. Large Object Support in POSTGRES. ICDE, 1993. 355～362
4 Sarawagi S, Stonebraker M. Efficient Organization of Large Multidimensional Arrays. ICDE, 1994. 328～336
5 Zhao Y, Deshpande P M, Naughton J F. An array-based algorithm for simultaneous multidimensional aggregates. In: SIGMOD'97, Tucson, Arizona, May 1997. 159～170
6 PostgreSQL. http://www.postgresql.org
7 Padmanabhan S, Bhattacharjee B, Malkemus T, et al. Multi-Dimensional Clustering: A New Data Layout Scheme in DB2. In: SIGMOD 2003, San Diego, USA, 2003
8 Bhattacharjee B, Padmanabhan S, Malkemus T, et al. Efficient Query Processing for Multi-Dimensionally Clustered Tables in DB2. In: Proc. Intl Conf Very Large Data Bases (VLDB), 2003

(上接第 121 页)

责。基于 CSCW 平台的完整信任链，Alice 可以信任 F 已被 Bob(拥有角色 R_3)签署，而不是被其他人冒名行使了 R_3 的权利。

结束语 本文基于可信计算技术，结合 CSCW 系统访问控制的具体特性，提出形式化的访问控模型 FAC for CSCW 和体系架构，最后通过应用举例描述了完整的访控过程。该模型实现了基于角色具有时间依赖和约束特征的协同工作，并通过可信计算技术实现了 CSCW 环境中的实体身份鉴别和安全策略完整性实施，从而保障了整个系统资源的秘密性、完整性和可用性。如何基于 TC 实现 CSCW 中的委托授权传播与访问，使其更好地适应分布式协同工作环境将作为进一步的研究方向。

参 考 文 献

1 Sandhu R, Ranganathan K, Zhang X. Secure Information Sharing Enabled by Trusted Computing and PEI Model. In: Proc. of ASIACCS06, Mar, Taipei, Taiwan, 2006
2 Sandhu R, Zhang X. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In: Proc of SACMAT05, Stockholm, Sweden, June 2005
3 龚能，李玉顺，史美林. 协作环境中的关键技术研究[J]. 计算机科学，2005，32(9)：230～233
4 Zhu H. Some issues of Role-based Collaboration. In: Proceedings of IEEE CCECE2003- CCGEI2003, Montreal, May 2003
5 Zhu H. Conflict Resolution with Roles in a Collaborative System. International Journal of Intelligent Control and Systems, 2005, 10(1)：11～20
6 Lang Bo, Lu You, Zhang Xin, et al. A flexible access control mechanism supporting large scale distributed collaboration. In: Proceedings of the 8th International Conferenceon Computer Supported Cooperative Work in Design. Vol 1. May 2004. 500～504
7 Tripathi A R, Ahmed T, Kumar R. Specification of Secure Distributed Collaboration System[C]. In: IEEE Proceedings of the sixth International Symposium on Autonomous Decentralized Systems, 2003
8 李成锴，詹永照，茅兵，等. 基于角色的 CSCW 系统访问控制模型[J]. 软件学报，2000，11(7)：931～937
9 Sandhu R, Zhang X, Ranganatham K, et al. Client-side access control enforcement using trusted computing and PEI models. Journal of High Speed Network, 2006, 15：229～245
10 Sandhu R. Engineering Authority and Trust in Cyberspace: The OM-AM and RBAC Way. In: the Proc. of ACM Workshop on Role Based Access Control, Berlin, Germany, 2000
11 Balacheff B, Chen L, Pearson S, et al. Trusted Computing Platforms: TCPA Technology in Context[M]. Prentice Hall Press, Jul, 2002
12 TCG Specification Architecture Overview Revision 1.2. Trusted Computing Group, Apr, 2004. http://www.trustedcomputinggroup.org
13 Bishop M. Computer Security: Art and Science(English version)[M]. 北京：清华大学出版社，2004，5.47，103