文章编号:1001 - 9081(2005)12 - 2858 - 04

分布式数据库多层关联规则挖掘算法研究

曹洪其[1],姜志峰[2],孙志挥[2]
(1. 南通职业大学 电子工程系,江苏 南通 226007; 2. 东南大学 计算机科学与工程系,江苏 南京 210096)
(chq@mail.ntvc.edu.cn)

摘 要:对分布式数据库多层关联规则挖掘的理论和方法进行了研究,提出了一种基于频繁模式树 FP-tree(Freguent Pattern tree)的快速挖掘算法 DMAML_FPT(Distributed Mining Algorithm of Multiple Level based on FP-tree)。与类 Apriori 算法相比较,该算法最多只需扫描数据库三遍,不需产生和传输大量的候选项集,减少了数据通信量,从而提高了数据挖掘的效率。实验结果表明算法 DMAML_FPT 是可行和有效的。

关键词:数据挖掘;分布式数据库;多层关联规则;频繁模式树
中图分类号: TP311.13 **文献标识码**:A

Research of multi-level association rules mining in distributed database

CAO Hong-qi[1], JIANG Zhi-feng[2], SUN Zhi-hui[2]
(1. *Department of Electronic Engineering, Nantong Vocational College, Nantong Jiangsu* 226007, *China*;
2. *Department of Computer Science and Engineering, Southeast University, Nanjing Jiangsu* 210096, *China*)

Abstract: A fast mining algorithm named DMAML_FPT(Distributed Mining Algorithm of Multiple Level based on Frequent Pattern Tree) was presented after researching multi-level association rules mining in distributed database. Comparing with Apriori-like algorithms, DMAML_FPT only need to scan the database for three times, eliminated the need for generating the candidate items, reduced the communication cost, and improved efficiency of data mining. Experiment results show that the algorithm is feasible and efficient.

Key words: data mining; distributed database; multi-level association rules; frequent pattern tree(FP-tree)

0 引言

目前可用的分布式挖掘算法主要有用于单一概念层的关联规则挖掘算法,如:PDM[1]、CD[2]、FDM[3]、FMAGF[4]等,而对于分布式多层关联规则挖掘算法的研究报导尚不多见。MLFDM_LP[5]算法针对分布式环境,在每一层上运用类 Apriori[6]算法实现了关联规则数据挖掘。Apriori 算法是挖掘关联规则频繁项集的基本算法,该算法利用一个层次顺序搜索的循环和对候选项集的剪枝方法来完成频繁项集的挖掘。然而,由于 Apriori 算法是依赖于候选项集产生频繁项集的理论所开发的算法,它主要存在两个不足:1)它可能需要产生大量候选项集;2)它需要通过多次扫描数据库来计算频繁项目集,大量的时间消耗在内存与数据库中的数据交换上。尤其是将该算法应用于分布式多层关联规则数据挖掘时,扫描次数多,通过传送局部频繁项集来求全局频繁项集时,站点间的数据通信量大,影响了挖掘性能的提高。针对 Apriori 算法的不足,由文献[7]提出了一种在单机环境下不产生候选项集的算法 FP-Growth,该算法将提供频繁项集的数据库压缩到一棵频繁模式树 FP-tree,避免了高代价的候选项集的产生,获得了更好的效率。

本文将 FP-tree 结构应用于分布式环境,提出了分布式数据库的多层关联规则挖掘算法 DMAML_FPT。该算法突破了类 Apriori 算法的框架,不但减少了数据库扫描的次数,而且不需产生和传输大量的候选项集,减少了数据通信量,提高了数据挖掘的效率。

1 定义与定理

分布式数据库多层关联规则的挖掘问题实质上就是:在

收稿日期:2005 - 06 - 02;**修订日期**:2005 - 08 - 29 **基金项目**:国家自然科学资助基金(70371015)
作者简介:曹洪其(1957 -),男,江苏江阴人,副教授,主要研究方向:数据库、数据挖掘; 姜志峰(1979 -),女,江苏苏州人,硕士研究生,主要研究方向:数据库、数据挖掘; 孙志挥(1941 -),男,江苏南通人,教授,博士生导师,主要研究方向:复杂系统信息集成、数据库系统及应用.

图像的人脸检测、识别、跟踪,以及基于对象的检索等相关研究中。

参考文献:

[1] 陈泽宇,戚飞虎,陈刚.利用颜色信息的人脸检测方法[J].上海交通大学学报, 2000,34(6):793 - 795.
[2] 黄福珍,苏剑波. 基于 Level Set 方法的人脸轮廓提取与跟踪[J]. 计算机学报,2003,26(4):491 - 496.
[3] 梁路宏,艾海舟,何克忠. 基于多模板匹配的单个人脸检测[J]. 中国图象图形学报,1999,4(A)(10):825 - 829.
[4] PHILLIPS PJ. Matching pursuit filters applied to face identification [J]. IEEE Transactions Image processing, 1998, 8(7): 1150 - 1164.
[5] 康学雷,邵凌,张立明.一种基于肤色和模板的人脸检测方法[J].红外与毫米波学报, 2000,19(3).
[6] TERRILLON JC, SHIRAZI MN, FUKAMACHI H, *et al.* Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images[A]. Proceedings of Conference on Automatic face and gesture Recognition[C]. Grenoble, France, 2000. 54 - 61.
[7] PHILIPS TY, ROSENFELD A, *et al.* O(logn) bimodality analysis [J]. Pattern Recognition, 1994, 13(2): 210 - 216.

给定各层最小支持度的情况下,在多个站点上挖掘出各层全局频繁项集。分布式数据库的数据挖掘技术涉及到数据库系统的设计、概念层次的表示等。

① 本文所讨论的分布式数据库是水平划分的,分布在 n 个不同的站点 $S^1,S^2,\cdots,S^n$,且各部分的数据库模式逻辑同构,它们之间除了通过网络传递信息外,其他资源全部独立。站点只处理自身的私有数据库 $DB^i(i=1,2,\cdots,n)$,各站点之间仅通过网络传递有限的统计信息,最终在整个事务数据库 DB 中挖掘出多层关联规则。

② 概念层次结构通常使用概念树表示,树的结点表示概念,树枝表示偏树,树中的每一层都使用一个数字来表示层数,根节点的层数为 1,根节点以下节点的层数定义为其父节点层数加 1。很明显,较高层的层数较低,相反较低层的层数较高。本文所讨论的概念树最底层层数设为 m。

设 $I=\{i_1,i_2,\cdots,i_m\}$ 是 m 个不同项目的一个集合,其中每一个交易 T_i 是 I 中的一组项目的集合,每一个交易都与一个唯一的标识符 TID 相联,$DB^i(i=1,2,\cdots,n)$ 是存储在 $S^i(i=1,2,\cdots,n)$ 站点上的同构分布事务数据库,总的事务数据库 $DB=\sum DB^i$。分布事务数据库 DB^i 中的事务数为 $D^i(i=1,2,\cdots,n)$,则总的事务数 $D=\sum D^i$。并且各站点都采用相同的编码方式。

定义 1　j 层 k-项集 $A[j,k]$ 在 DB 中的支持数称为 $A[j,k]$ 的全局支持数,记为 $A[j,k].Sup$;在 $DB^i(i=1,2,\cdots,n)$ 中的支持数,称为 $A[j,k]$ 的局部支持数,记为 $A[j,k].Sup^i$。

定义 2　设 $minsup.j$ 为 j 层的最小支持度。若 $A[j,k].Sup\geqslant minsup.j\times D$,则 $A[j,k]$ 是全局 j 层的频繁 k-项集,记为 $L[j,k]$;若 $A[j,k].Sup^i\geqslant minsup.j\times D^i(i=1,2,\cdots,n)$,则 $A[j,k]$ 在 S^i 上是局部 j 层的频繁 k-项集,记为 $L^i[j,k]$。

定义 3　若 $A[j,k]$ 在 $S^i(i=1,2,\cdots,n)$ 上是局部 j 层频繁 k-项集,它还是全局 j 层频繁 k-项集,则称 $A[j,k]$ 在 $S^i(i=1,2,\cdots,n)$ 上是 j 层强频繁 k-项集。

定理 1　若 $A[j,k]$ 是全局 j 层频繁 k-项集,则必存在一个站点 $S^i(i=1,2,\cdots,n)$,使得 $A[j,k]$ 在 $S^i(i=1,2,\cdots,n)$ 上是 j 层强频繁 k-项集。

证明:用反证法。

假设,项 $A[j,k]$ 是全局 j 层的频繁 k-项集,但在每个站点上,$A[j,k]$ 都不是 j 层强频繁 1-项集。则由定义 2 和定义 3 得 $A[j,k].Sup^i<minsup.j\times D^i(i=1,2,\cdots,n)$,从而推导出 $\sum_{i=1}^{n}A[j,k].Sup^i<minsup.j\times D$,即 $A[j,k].Sup<minsup.j\times D$,说明项 $A[j,k]$ 不是全局 j 层频繁项集,这与假设相矛盾,故定理 1 成立。

定理 1 说明全局频繁项集可从局部频繁项集中产生。因此,减小局部频繁项集引起的通信代价是提高效率的关键之一。

在基于 FP-tree 分布式数据库多层关联规则挖掘环境中,将 DB 对应的 j 层频繁模式树称为 j 层全局频繁模式树,记为 $Fptree[j]$;将 DB^i 对应的 j 层频繁模式树称为 j 层局部频繁模式树,记为 $Fptree^i[j](i=1,2,\cdots,n)$。

定理 2　若 $CPB[j]$ 为 j 层全局频繁模式树 $Fptree[j]$ 生成的 j 层全局条件模式基,$CPB^i[j]$ 为 j 层局部频繁模式树 $Fptree^i[j]$ 生成的 j 层局部条件模式基,则 $CPB[j]=\sum_{i=1}^{n}CPB^i[j]$。

证明:根据全局频繁模式树是由各局部频繁模式树组成,就可得到定理 2。

定义 4　$\lambda(p)$ 是一个函数,其功能为返回项 p 上一层的父节点项 q。如果 j 层的项 p 在 DB 中为非全局 j 层频繁项,但 q 在 DB 中为全局 $j-1$ 层频繁项,则 p 称为 $j-1$ 层项 q 的修补项。

由概念分层和定义 4 可得到:$j-1$ 层项 q 的支持度为 j 层中所有父节点为 q 的全局 j 层频繁项的支持度与 q 的所有修补项的支持度之和。

2　轮询计数

本文中,在形成各层局部频繁模式树及求全局条件模式基过程中,使用了轮询技术。轮询技术是通过使用一个分配函数为每一个局部频繁项或各局部条件模式基指定一个轮询站点,这个分配函数可以是一个哈希函数或者其他自定义的函数,并且每个站点上都存在这个函数,这样在不同站点上的相同的局部频繁项或局部条件模式基都将被送到同一个轮询站点。轮询站点的任务是判断送来的局部频繁项是否为全局频繁项或计算全局条件模式基,其所需的通信次数为 $O(n)$。

3　全局频繁项目集的挖掘

假设在各不同站点上项目的分类层次和采用的编码方式均相同,并采用文献[8]中的编码方式,将事务数据库中的每个项目用一个编码串表示后的编码交易表,用于数据挖掘。在下面的讨论中,以表 1 为例介绍。该交易表表示的数据分布在 3 个站点上,概念树共 3 层,在较低层使用递减的最小支持度(支持数),1 层的最小支持数 $SP1=5(minsup.1=20\%)$,2 层的最小支持数 $SP2=3(minsup.2=12\%)$,3 层的最小支持数 $SP3=2(minsup.3=8\%)$。

表 1　编码交易表

站点	TID	交易集
S^1	T1	{111,323,211,221}
	T2	{111,211,222,421}
S^2	T3	{112,121,221,411}
	T4	{111,721}
S^3	T5	{111,122,211,221}
	T6	{211,323,524}
	T7	{323,411,524,813}

3.1　DMAML_FPT 算法思路

DMAML_FPT 算法采用层交叉单项过滤搜索策略,该算法的关键在于建立起各层局部条件模式树 Fptree,形成各层全局条件模式基,从而获得各层全局频繁项目集。主要由以下步骤组成:

(1) 扫描各站点数据库 DB^i,统计出 1 层 1-项集 $A[1,1]$ 各自的局部支持度,产生 1 层局部频繁 1-项集 $L^i[1,1]$。然后采用轮询计数的策略,通过一个哈希函数($polling\_site$)为每一个局部频繁 1-项集 $L^i[1,1]$ 指定一个轮询站点,并将这些局部频繁 1-项集作为全局频繁项集的候选集送往各自的轮询站点。各轮询站点计算出候选集各自的全局支持数,产生全局频繁 1-项集 $L[1,1]$。各站点广播 1 层频繁项。例如:表 1 中 {1 **} 在 S^1、S^2、S^3 站点上的局部支持数分别为 2、3、2,将它们通过 hash 函数发送到某一轮询站点(设为 S^1),得到 {1 **} 的全局支持数为 7,是频繁项。同理可得 {2 **} 的全局支持数为 8,是频繁项。而 {3 **}、{4 **}、{5 **}、{7

**｝、｛8 **｝是非频繁项。因此，经本步骤可得$L[1,1]$为｛1 **｝7、｛2 **｝8。

（2）第二遍各站点扫描数据库DB^i，根据$L[1,1]$对DB^i进行层间裁减，统计出最底层即m层1-项集$A[m,1]$的局部支持数及建立1层局部Fptree。然后把项集$A[m,1]$发往各自的轮询站点，各轮询站点建立hash树，由hash树统计各层的全局频繁1-项集及修补项。例如：由表1及相应的$L[1,1]$经本步骤可得图1。

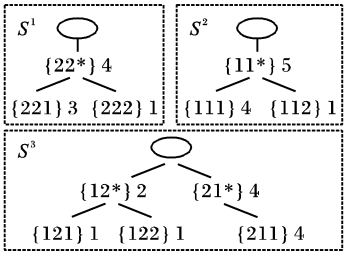

图1 产生各层（除1层）全局频繁1-项集及修补项集

图1中S^1中2层｛22 *｝支持数为4是频繁项，3层中｛221｝支持数为3是频繁项，而｛222｝支持数为1是非频繁项，但由于｛22 *｝是频繁项，则｛222｝为｛22 *｝的修补项，该修补项用编码｛22#｝表示（22#由222在第3位变为#而得到），｛222｝的支持数记入修补项的支持数。同理，S^2中频繁项为｛11 *｝、｛111｝，｛112｝是非频繁项，产生修补项为｛11#｝；S^3中频繁项为｛21 *｝、｛211｝，而｛12 *｝、｛121｝、｛122｝都为非频繁项。各站点广播频繁项、频繁项支持数及修补项、修补项支持数。

（3）建立各层局部FP-tree：①建立头表。将接受到的各层全局频繁项及修补项分别按降序排列，各层的频繁项后接上修补项作为相应层FP-tree的项集；②第三次扫描数据库，进行层间裁减，减去不属于$L[1,1]$的项，建立m层各站点局部FP-tree树；③遍历树产生包含修补项和不包含修补项的条件模式基，用路径产生上一层的FP-tree树（注意：1层FP-tree树已生成，不由2层FP-tree树产生）。局部条件模式基发往轮询站点。例如：图2表示了S^1站点上形成的各层局部FP-tree树。

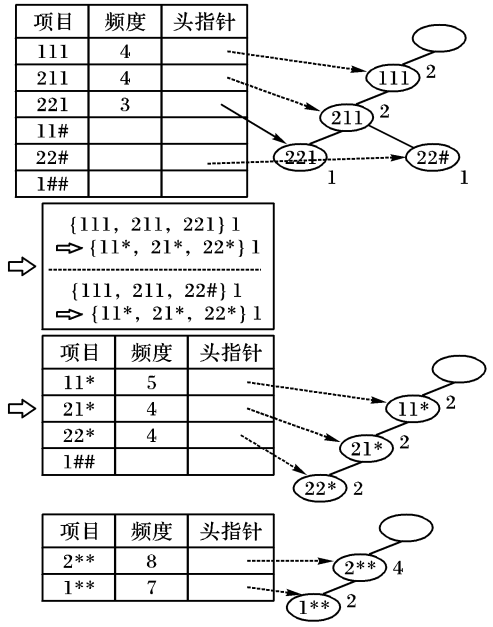

图2 建立各层局部FF-Tree

（4）产生全局频繁项目集：按散列方法来指定各局部条件模式基的轮询地址，各轮询站点统计局部条件模式基，产生全局条件模式基，并建立全局条件模式树，得出全局频繁项目集。如项221在S^1、S^2、S^3站点上的局部条件模式基分别为：｛111，211｝1、∅、｛111，211｝1，将这些局部条件模式基通过hash函数发送到某一轮询站点（设为S^1），得到项221的全局条件模式基：｛111，211｝2，由此可得全局频繁项目集｛221，211｝2，｛221，111｝2，｛221，111，211｝2。

3.2 算法

输入：存储在各站点S^i中的数据库$DB^i(i=1,2,\cdots,n)$；各层的最小支持度$minsup.j(j=1,2,\cdots,n)$。

输出：各层的全局频繁项集。

方法：

```
(1) L[1,1] = {1层全局频繁1-项集}
//第一遍扫描数据库，产生 L[1,1]
(2) For (i=1;i≤n;i ++)
For each Transaction t in DB^i do
//第二遍扫描数据库
{ t′ = Filter Transaction(t);
inset the t′ to the Fptree^i[1];
//产生局部一层 FP-tree
For each item in t′ do
//统计 m 层 1-项集各自的局部支持数，
//并通过 hash 函数指派相应的轮询站点
{ k = polling_site(item);
if (item not in TransItem^i[k])
inset(item,1) into TransItem^i[k];
else TransItem^i[k].item.sup ++;
}
}
(3) For (i=1;i≤n;i ++)
For (k=1;k≤n;k ++)
if TransItem^i[k] ≠ ∅
Send TransItem^i[k] to S^k;
//将 A[m,1]发送到各自的站点
(4) Receive the TransItem from the other sites;
(5) Get _each_level_items(TransItem,minsup);
//处理 m 层 1-项集，建立 hash 树，
//获各层全局频繁 1-项集及修补项（除 1 层）
(6) For(j=2;j≤m;j ++)
Broadcast ( L[j,1],Lre[j,1]);
(7) Receive L[j,1] and Lre[j,1]
from the other sites;
(8) For(j=2;j≤m;j ++)
{sort L[j,1] in desending order;
sort Lre[j,1] in desending order;
}
(9) For (i=1;i≤n;i ++)
{For each Transaction t in DB^i do
//第三遍扫描数据库
{t′ = Filter Transaction(t);
inset the t′ to the Fptree^i[m]
//产生局部 m 层 FP-tree
}
For(j=m-1;j>1;j --)
Fptree^i[j] = Get_FP-tree(Fptree^i[j+1],minsup.j);
//遍历 Fptree^i[j+1]，产生局部条件模式基，
//形成 Fptree^i[j]
}
(10) For (i=1;i≤n;i ++)
For (j=1;j≤m;j ++)
```

For each Fptree^i[j] | item do
//Fptree^i[j] | item 表示
//项 item 的局部条件模式基
{k = polling_site(Fptree^i[j] | item);
Send Fptree^i[j] | item to S^k;
//条件模式基发送到函数各自轮询站点
}
(11) Receive Condition Pattern Base from other sites;
(12) Get_each_level_ global _CPB( );
//用 ftpgrowth 算法建立全局条件模式基,
//得出频繁项集
(13) Broadcast Frequent Itemsets;
(14) Receive Frequent Itemsets;

4　算法分析和实验及性能分析

4.1　算法分析

由上可知,DMAML_FPT 算法采用了一系列优化技术来保证算法的性能。主要包括:

1) 引入修补项的概念,以便从低到高逐层建立 FP-tree,有效地减少了扫描数据库的次数,最多只需扫描三次,避免了类 Apriori 算法需要多次扫描数据库的缺点。

2) 在网络通信量方面:①在形成各层局部频繁模式树及求全局条件模式基过程中,采用了按散列方法来指定各局部频繁项目或各局部条件模式基的轮询地址,所需的通信次数仅为 $O(n)$;②由于在各站点采用了频繁模式树的存储结构,通过采用传送条件模式基来挖掘各层(1 层除外)全局频繁项集,以此减小网络通信量,提高全局挖掘效率。

可以将本算法与 FDM 和 MLFDM-LP 算法进行性能比较。后两者是通过类 Apriori 算法来挖掘局部频繁项目集,各站点通过传送局部频繁项,再由局部频繁项获得全局频繁项。FDM 算法每次迭代的通信开销为 $O(Pr_p \mid C \mid n)$,其中,Pr_p 为某一候选项目集能成为全局频繁项目集潜在的概率,C 为一次迭代的候选项目集的集合,n 为分布式环境下站点的数目。MLFDM-LP 算法在单一概念层次上和 FDM 采用的局部和全局修剪策略基本相同,在每个概念层的每次迭代的通信开销为 $O(Pr_m \mid C \mid n)$,Pr_m 略小于 Pr_p。

DMAML_FPT 算法在单一概念层次的网络通信量为 $O(\mid J \mid n)$,其中,J 表示单一概念层中条件模式基的集合。由于条件模式基其实是局部频繁项信息的压缩,所以,$\mid J \mid < \mid C \mid$。并且对于多项目的海量数据库,在 *minsup* 取值较小的情况下,局部频繁项目集迅速增长,Pr_p 的可扩展性差,随着 n 的增大,Pr_p 很快达到 1。因此,算法 DMAML_FPT 避免了在网上传送大量的项目集,有效的减少了网络通讯量。

4.2　实验及性能分析

为了测试算法的性能,将多台 CPU 为 Pentium Ⅲ,内存为 256M,硬盘为 40G 的计算机构成局域网络。网络每台计算机配有 100M 的以太网,局域网内子网段间通过 100M 的交换机相连。软件运行环境 Windows 2000 Server 操作系统,VC++ 6.0。本文实验数据来自于某药品行业销售数据库中 100 000 个样本记录,交易记录采用随机抽样的方法将数据分割到各个站点上,*maxlevel* = 3。实验结果如图 3 和图 4 所示。

图 3 和图 4 分别表示了网络通信量与最小支持度的关系以及执行时间与最小支持度的关系(第 3 层)。实验结果说明 DMAML_FPT 算法在网络通信量上少于 MLFDM_LP 算法的网络通信量,在执行效率上优于 MLFDM_LP 算法,并且当最小支持度减小时,DMAML_FPT 算法性能更好。

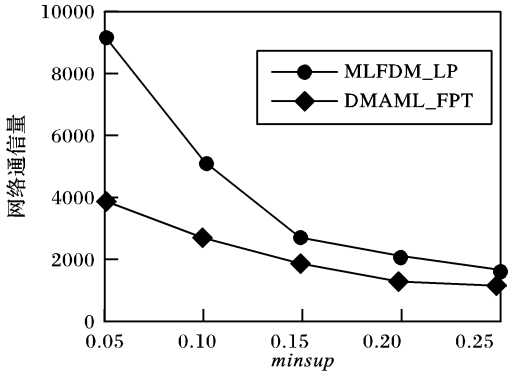

图 3　算法的网络通信量与最小支持度的关系(第 3 层)

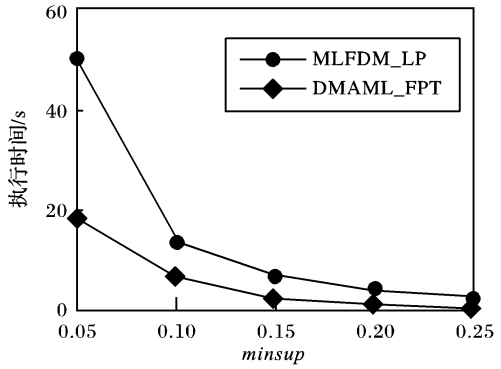

图 4　算法的执行时间与最小支持度的关系(第 3 层)

5　结语

本文提出的在分布式环境下多层全局频繁项目集挖掘算法 DMAML_FPT 已通过原型系统的验证,实验结果证明该算法和相关技术是有效、可行的。而且,通过对 DMAML_FPT 进行部分修改,使算法可根据跨层挖掘信息在特定的层间挖掘出层间关联规则。

参考文献:

[1] PARK JS, CHEN MINI-SYAN, YU PS. Efficient Parallel data mining for association rules[A]. Proceedings of the 4th Intern'l Conference on Information and Knowledge Management[C]. Baltimore, 1995. 31-36.

[2] Agrawal R, Shafer JC. Parallel mining of association rules[J]. IEEE Transaction on Knowledge and Data Engineering, 1996, 8(6): 962-969.

[3] CHEUNG DW, HAN JW, VINCENT T NG, *et al*. A fast distributed algorithm for mining association rules[A]. Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems[C]. Miami Beach, Florid, USA, 1996. 31-44.

[4] 杨明, 孙志挥, 吉根林. 快速挖掘全局频繁项目集[J]. 计算机研究与发展, 2003, 40(4): 620-626.

[5] 王春花, 黄厚宽, 李红莲. 一种快速有效的分布式开采多层关联规则的算法[J]. 计算机研究与发展, 2001, 38(4): 438-443.

[6] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules[A]. Proceedings of the 20th International Conference on VLDBP[C]. Santiago, chile, 1994. 487-499.

[7] HAN J, JIAN P, YIWEN Y. Mining frequent patterns without candidate generation[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data[C]. Dallas, 2000. 1-12.

[8] HAN JW, FU YJ. Mining multiple-level association rules in Large databases[J]. IEEE Transactions on Knowledge and Data Engineering, 1999, 11(5).