快速更新全局频繁项目集*

杨 明[1,2+], 孙志挥[1], 宋余庆[1,3]

[1](东南大学 计算机科学与工程系,江苏 南京 210096)

[2](安徽工程科技学院 计算机科学与工程系,安徽 芜湖 241000)

[3](江苏大学 计算机科学与通信工程学院,江苏 镇江 212023)

Fast Updating of Globally Frequent Itemsets

YANG Ming[1,2+], SUN Zhi-Hui[1], SONG Yu-Qing[1,3]

[1](Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

[2](Department of Computer Science and Engineering, Anhui University of Technology and Science, Wuhu 241000, China)

[3](College of Computer Science and Communications Engineering, Jiangsu University, Zhenjiang 212023, China)

+ Corresponding author: Phn: +86-25-3795451, E-mail: yangm_163@163.com, http://www.seu.edu.cn

Received 2003-03-14; Accepted 2003-11-11

Yang M, Sun ZH, Song YQ. Fast updating of globally frequent itemsets. *Journal of Software*, 2004,15(8): 1189~1197.

http://www.jos.org.cn/1000-9825/15/1189.htm

Abstract: The incremental updating research of frequent itemsets is an important data mining problem in data mining fields. Many sequential algorithms have been proposed for incremental updating of frequent itemsets. However, very little work has been done in updating frequent itemsets in distributed environment. In this paper, the algorithm FUAGFI (fast updating algorithm for globally frequent itemsets) is introduced in the case of inserting, which efficiently utilizes the created locally frequent pattern trees and the mined globally frequent itemsets. Therefore, FUAGFI uses far less communication overhead and obviously improves updating efficiency of globally frequent itemsets. Experimental results show the feasibility and effectiveness of the algorithm.

Key words: data mining; distributed database; globally frequent itemset; frequent pattern tree (FP-tree); update

摘 要: 数据挖掘中的频繁项目集更新算法研究是重要的研究课题之一.目前已有的频繁项目集更新算法主要针对单机环境,有关分布式环境下的全局频繁项目集的更新算法的研究尚不多见.为此,提出了快速更新全局频繁项目集算法(fast updating algorithm for globally frequent itemsets,简称 FUAGFI).该算法主要考虑数据库记录增加时全局频繁项目集的更新情况.FUAGFI 利用已建立的各局部频繁模式树(frequent pattern tree,简称 FP-tree)及已挖掘的全局频繁项目集,可有效地降低网络通信量,提高全局频繁项目集的更新效率.实验结果表明,所提出的更新算法是行

* Supported by the National Natural Science Foundation of China under Grant No.79970092 (国家自然科学基金); the Natural Science Foundation of Anhui Province of China under Grant No.03042205 (安徽省自然科学基金)

作者简介: 杨明(1964－),男,安徽宁国人,博士,教授,主要研究领域为知识发现与数据挖掘,粗集理论及应用;孙志挥(1941－),男,教授,博士生导师,主要研究领域为复杂系统集成,数据库,知识发现,数据挖掘;宋余庆(1959－),男,博士生,教授,主要研究领域为数据挖掘,医学图像处理.

之有效的.

关键词: 数据挖掘;分布式数据库;全局频繁项目集;频繁模式树(FP-tree);更新

中图法分类号: TP311 **文献标识码**: A

自从 Agrawal 于 1993 年首次提出布尔型关联规则问题及相应的 Apriori 算法以来[1,2],数据挖掘领域的研究者在关联规则挖掘方面做了大量的工作[3~7],其中快速更新频繁项目集是数据挖掘研究的重点之一.已有的算法大多局限于单机环境,针对分布式数据库的全局频繁项目集更新算法尚不多见.

目前可用的全局频繁项目集挖掘算法有 PDM[8],CD[9],FDM[10],FPM[11],FMAGF[12]等,这些算法的目标主要是降低网络通信开销.PDM 和 CD 在 Share-Nothing 环境下扩展 Apriori 算法,每一次迭代,它们的通信开销分别为 $O(|C|n^2)$和 $O(|C|n)$,其中,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.它们存在的主要不足是,候选项目集大且须保持每一次迭代的同步.FDM 通过优化通信策略以及减少候选项目集数目,使得每次迭代的通信开销降为 $O(Pr_p|C|n)$(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),克服了 PDM 和 CD 的不足.FPM 采用依据数据分布划分和全局修剪策略对 FDM 进行了改进,可以有效地减少网络通信量并改进算法的性能.FDM 和 FPM 算法通过优化通信策略,并在自底向上(Apriori-like 算法的思想)的搜索过程中,对局部候选项目集进行有效地修剪,在一定程度上提高了算法的效率.但其存在的主要缺陷是:① 对多项目的海量数据库不可避免地会产生大量的局部候选项目集;② 采用 Apriori-like 算法,对于一个长度为 L 的全局频繁项目集,在最坏的情况下,将在网上传送 2^L-1 个候选项目集;当 L 很大时,将使网络通信代价呈指数增长.FMAGF 采用 FP-tree(frequent pattern tree)存储结构,通过传送条件频繁模式树或条件模式基来挖掘全局频繁项目集,可以有效地降低网络通信开销,提高全局频繁项目集的挖掘效率.

FPM 和 FMAGF 均未讨论全局频繁项目集的更新问题,而现实数据库是动态变化的,随着数据库记录的增加,一些新的频繁模式会产生,一些旧的频繁模式会被淘汰,故须对已得到的全局频繁项目集进行高效更新.现有的可用的频繁项目集更新算法 FUP[7],IUA[8]和 IUABPGL[9]均局限于单机环境,针对分布式数据库的全局频繁项目集更新算法尚不多见.虽然 UAGFI 算法[13]讨论了最小支持度发生变化时全局频繁项目集的更新,但这仅是一种较为简单的情况.为此,本文提出了一种快速更新全局频繁项目集算法——FUAGF(fast updating algorithm for globally frequent itemsets).该算法主要考虑数据库记录增加时全局频繁项目集的更新情况.FUAGFI 利用 FMAGF 建立的各局部频繁模式树(FP-tree)及已挖掘的全局频繁项目集,可有效地降低网络通信量,提高全局频繁项目集的更新效率.实验结果表明,本文提出的更新算法是可行且有效的.

1 相关概念和结论

1.1 全局频繁项目集

沿用文献[12]的记号,在本文的模型中,分布式数据库是水平划分的,且各部分的数据库模式逻辑同构.设有 n 个站点 $S^1,S^2,\ldots,S^n$,相应的成员数据库分别为 $DB^1,DB^2,\ldots,DB^n$,$DB=\bigcup_{i=1}^{n} DB^i$ .D 和 D^i 分别表示 DB 和 DB^i 中的交易数.对某一项目集 X,$X.count$ 和 $X.count^i$ 分别表示在 DB 和 DB^i 中包含 X 的交易数,若 X 仅包含一个项目,则 $X.count$ 和 $X.count^i$ 分别为对应项目的全局频度和局部频度.

定义 1. 若 $X.sup(X.count/D)$和 $X.sup^i(X.count^i/D^i)$分别表示 X 在 DB 和 DB^i 中的支持度,则称 $X.sup$ 为 X 的全局支持度,$X.sup^i$ 为 X 在站点 S^i 的局部支持度.

定义 2. 若 $X.sup\geq minsup$(minimum support threshold,最小支持度阈值),则 X 为全局频繁项目集;而若 $X.sup^i\geq minsup$,则 X 为站点 S^i 的局部频繁项目集.

为方便描述,称长度为 k 的全局频繁项目集为全局频繁 k-项目集,称全局频繁 1-项目集所包含的项目为全局频繁项目.称长度为 k 的局部频繁项目集为局部频繁 k-项目集,称局部频繁 1-项目集所包含的项目为局部频繁项目.用 GFI^{DB} 表示从 DB 挖掘得到的全局频繁项目集所组成的集合.

引理 1[12]. 若 X 为站点 S^i 上的局部频繁项目集,则 X 的所有非空子集均为站点 S^i 上的局部频繁项目集.

推论 1. 若项目集 X 不是局部频繁项目集,则 X 的超集一定不是局部频繁项目集.

类似地,若 X 为全局频繁项目集,则 X 的所有非空子集均为全局频繁项目集.

定理 1[12]. 若 X 为全局频繁项目集,则存在一个站点 $S^i(1\le i\le n)$,使得 X 和 X 的所有非空子集在站点 S^i 上为局部频繁项目集.

可见,全局频繁集是从局部频繁项目集中得到的,因而减小由传送局部频繁项目集引起的网络通信代价是提高效率的关键.

1.2 频繁模式树

一棵频繁模式树[3]为满足以下 3 个条件的树型结构:① 它由一个标为"null"的根结点(用 root 表示)作为根结点的孩子的项目前缀子树集合,由频繁项目头表组成;② 项目前缀子树中的每一结点包含 4 个域:item-name,count,node-parent,node-link,其中,item-name 记录项目名;count 记录能到达该结点的路径所表示的交易的数目;node-link 为指向 FP-tree 中具有相同 item-name 值的下一结点,当下一个结点不存在时,node-link 为 null;node-parent 为指向父结点的指针;③ 频繁项目头表的每一表项包含 3 个域:item-name,item-count,head of node-link,其中 item-count 为 item-name 对应项目的频度,head of node-link 为指向 FP-tree 中具有相同的 item-name 值的首结点的指针.

对于给定的某一交易数据库 *db* 及最小支持度阈值 *minsup*,建立 FP-tree 的主要步骤:① 扫描 *db* 一遍得到各项目的频度,根据最小支持度阈值 *minsup* 得到频繁项目;将项目按其频度由大到小排列(次序记为 $\Re$),形成头表;② 再次扫描 *db*,对每一条交易中的所有频繁项目,按次序 $\Re$ 组成相应的项目集并插入到 FP-tree 中.对每一个频繁项目对应的条件 FP-tree(通过所有能到达该项目对应的各结点的路径生成)进行挖掘,从而获得所有满足最小支持度阈值 *minsup* 约束的频繁项目集.

以上简单介绍了 FP-tree 的存储结构、建立 FP-tree 的主要步骤及频繁项目集的挖掘思路,更详细的内容请参见文献[3].

2 快速更新全局频繁项目集算法

2.1 FMAGF算法思路

在分布式数据库环境下,若记 DB 的频繁模式树为 $FP\text{-}tree(DB)$,DB^i 的频繁模式树为 $FP\text{-}tree(DB^i)$,所有的全局频繁项目组成的集合为 F,则对于给定的 *minsup*,FMAGF 将全局频繁项目集的挖掘分为两部分任务:① 建立各局部 $FP\text{-}tree(DB^i)$;② 挖掘各 $FP\text{-}tree(DB^i)$获得全局频繁项目集.文献[12]通过引入下面的定理 2 和定理 3,将分布式数据库的全局频繁项目集挖掘转化为各条件频繁模式树 $FP\text{-}tree|a(a\in F)$的全局频繁项目集挖掘.

定理 2[12]. 若 $L(FP\text{-}tree(DB)|a)$为由条件频繁模式树 $FP\text{-}tree(DB)|a(a\in F)$生成的全局频繁项目集,则 $GFI^{DB}=\bigcup_{a\in F}L(FP\text{-}tree(DB)|a)$.

定理 3[12]. 若 $L^i(FP\text{-}tree(DB^i)|a)(a\in F)$为条件频繁模式树 $FP\text{-}tree(DB^i)|a$ 生成的局部频繁项目集,则 $L(FP\text{-}tree(DB)|a)\subseteq\bigcup_{i=1}^{n}L^i(FP\text{-}tree(DB^i)|a)$.

FMAGF 采用传送条件频繁模式树或条件模式基方式来挖掘全局频繁项目集,可以有效地降低网络通信代价,提高全局频繁项目集的挖掘效率.

实例 1. 设 3 个站点 S^1,S^2 和 S^3 上的交易数据库分别为 DB^1,DB^2 和 DB^3,见表 1.对 *minsup*=0.35,依据 FMAGF(与文献[12]不同的是,在 FP-tree 的头表中增加了全局频度域,并记录所有项目的全局频度)建立的 $FP\text{-}tree(DB^1)$,$FP\text{-}tree(DB^2)$和 $FP\text{-}tree(DB^3)$分别如图 1~图 3 所示.

本文通过求 $FP\text{-}tree(DB)|m$ 的所有全局频繁项目集来说明 FMAGF 的思路.由图 1~图 3 可知,项目 m 在 S^1,S^2 和 S^3 上的条件模式基分别为{(f:1,c:1,a:1),(f:1,c:1,a:1, b:1)},∅和{(f:1,c:1,a:1)};$FP\text{-}tree(DB^1)|m$,$FP\text{-}tree(DB^2)|m$

和 *FP-tree*(DB^3)|*m* 分别为{(*f*:1,*c*:1,*a*:1),(*f*:1,*c*:1,*a*:1,*b*:1)}|*m*,∅和{(*f*:1,*c*:1,*a*:1)}|*m*.按照 FMAGF,将 *m* 的条件频繁模式树或条件模式基传送到某一个站点,不妨设为 S^1,则仅需将{(*f*:1,*c*:1,*a*:1)}|*m* 传送到 S^1,并在 S^1 挖掘可得全局频繁项目集{*f*,*c*,*a*,*m*}及其所有全局频繁项目子集.但通过 FDM 和 FPM 挖掘全局频繁集,需将生成的局部频繁项目集传送到某一个站点,不妨也设为 S^1,则为了得到全局频繁项目集{*f*,*c*,*a*,*m*}及其全局频繁项目子集,需将 *FP-tree*(DB^3)|*m* 生成的局部频繁项目集{*f*,*m*},{*c*,*m*},{*a*,*m*},{*f*,*c*,*m*},{*f*,*a*,*m*},{*c*,*a*,*m*},{*f*,*c*,*a*,*m*}传送到 S^1.显然,网络通信开销比传送条件频繁模式树{(*f*:1,*c*:1,*a*:1)}|*m* 要大.

Table 1 The transaction database at site S^i is DB^i, *i*=1,2,3

表 1 站点 S^1,S^2 及 S^3 上的各交易数据库

Database	TID	Transaction
DB^1	10	*f*,*a*,*c*,*d*,*g*,*i*,*m*,*p*
	20	*a*,*b*,*c*,*f*,*l*,*m*,*o*
DB^2	30	*b*,*f*,*h*,*j*,*o*
	40	*b*,*c*,*k*,*s*,*p*
DB^3	50	*a*,*f*,*c*,*e*,*l*,*p*,*m*,*n*
	60	*k*,*s*
db^3	70	*c*,*b*,*p*
	80	*c*,*a*,*m*,*p*

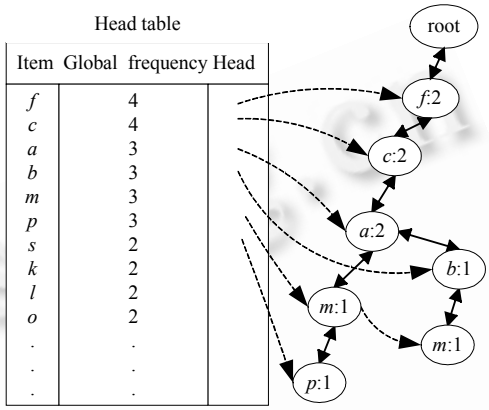

Fig.1 The frequent pattern tree on DB^1 (*FP-tree*(DB^1))

图 1 DB^1 的频繁模式树(*FP-tree*(DB^1))

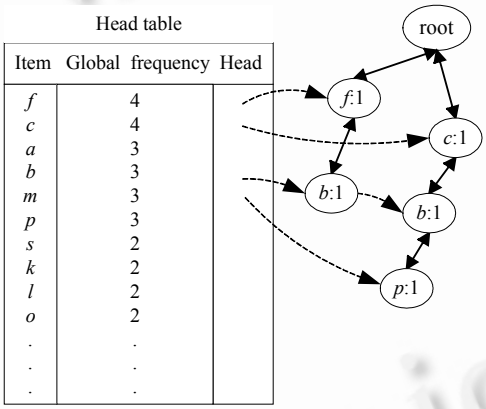

Fig.2 The frequent pattern tree on DB^2 (*FP-tree*(DB^2))

图 2 DB^2 的频繁模式树(*FP-tree*(DB^2))

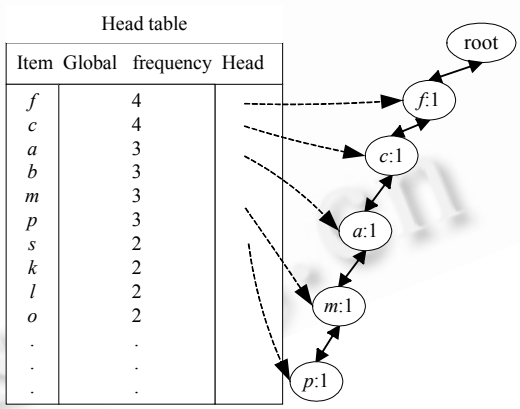

Fig.3 The frequent pattern tree on DB^3 (*FP-tree*(DB^3))

图 3 DB^3 的频繁模式树(*FP-tree*(DB^3))

FMAGF 未讨论全局频繁项目集的更新问题.虽然可以将 FMAGF 再运行一遍来重新得到全局频繁项目集,但该方法更新效率低.为此,在 *minsup* 保持不变而一些站点数据库增量改变的情况下,设计全局频繁项目集的高效更新算法是本文的主要研究目标.

2.2 全局频繁项目集的快速更新算法——FUAGFI

在实际应用中,随着交易数据库记录动态增加,会产生一些新的频繁模式,同时也会淘汰一些旧的频繁模式,因而对全局频繁项目集进行更新具有十分重要的意义.虽然可以采用重新运行 FPM 或 FMAGF 的方法得到新的全局频繁项目集,但这难以满足用户对较快响应时间的需求,因而要求设计高效的全局频繁项目集更新算法.

为了对全局频繁项目集进行高效更新,减少候选项目集数目,降低网络通信代价是关键.为此,更新算法必须充分利用已建立的局部频繁模式树和已挖掘的结果.为了便于讨论问题,不妨假设站点 S^1,S^2,…,S^p 的交易数据

库发生了增量改变,增量数据库分别是 $db^1,db^2,\ldots,db^p$.若把 $DB=\bigcup_{i=1}^{n}DB^i$ 看成一个逻辑站点上的交易数据库,而把 $db^1,db^2,\ldots,db^p$ 看成其他 p 个逻辑站点上的交易数据库,X 是 $TDB=DB\cup db^1\cup db^2\cup\ldots\cup db^p$ 上的全局频繁项目集,则由定理 1 可知,X 必是 $DB,db^1,db^2,\ldots,db^p$ 中某一局部数据库的局部频繁项目集.若将任意交易数据库 db 的所有局部频繁 k-项目集及局部频繁项目集分别记为 $LF_k(db)$和 $LF(db)$,则 $GFI^{TDB}\subseteq LF(DB)\cup LF(db^1)\cup LF(db^2)\cup\ldots\cup LF(db^p)$,其中 $LF(DB)=GFI^{DB}$.可见,TDB 的全局频繁项目集可由 $DB,db^1,db^2,\ldots,db^p$ 的局部频繁项目集得到.

性质 1. 设有 n 个局部交易数据库 $db_1,db_2,\ldots,db_n$,对给定最小支持度阈值 $minsup$,若 $X\in LF(db_1)\cap LF(db_2)\cap\ldots\cap LF(db_n)$,则 X 是全局频繁项目集.

为了更有效地修剪局部频繁项目集,减少网络通信量,我们引入了强频繁项目集的概念.

定义 3. 若 X 是全局频繁项目集,且在局部数据库 DB^i 上又是局部频繁项目集,则称 X 是 DB^i 的强频繁项目集.并将 DB^i 上的所有强频繁 k-项目集及所有强频繁项目集集合分别记为 $HF_k(DB^i)$和 $HF(DB^i)$.

性质 2. 若 X 不是局部交易数据库 db 的强频繁项目集,则 X 的超集一定不是 db 的强频繁项目集.

性质 3. 局部数据库 db 的强频繁项目集一定是 db 的局部频繁项目集,但 db 的局部频繁项目集未必是 db 的强频繁项目集.

定理 4. 设有 n 个局部交易数据库 $db_1,db_2,\ldots,db_n$,对给定最小支持度阈值 $minsup$,若 X 是全局频繁项目集,则存在局部交易数据库 db_i 使得 X 在 db_i 上是强频繁项目集.

证明:由定理 1 知,X 在某一局部交易数据库 db_i 上是局部频繁项目集,因而 X 在 db_i 上是强频繁项目集. □

由定理 4 可知,对 $TDB=DB\cup db^1\cup db^2\cup\ldots\cup db^p$ 有 $GFI^{TDB}=HF(DB)\cup HF(db^1)\cup HF(db^2)\cup\ldots\cup HF(db^p)$,其中,对任意 $db\in\{DB,db^1,db^2,\ldots,db^p\}$有 $HF(db)=\cup HF_k(db)$.因而 TDB 的全局频繁项目集可由 $DB,db^1,db^2,\ldots,db^p$ 的强频繁项目集得到.可见,通过性质 1~性质 3 和定理 4 可以有效地修剪候选项目集数目.因而 FUAGFI 算法的主要思路是:① 扫描各 $db^i(i=1,\ldots,p)$一遍,得到 db^i 中各项目的局部频度,扫描原某一局部数据库的频繁模式树头表中的各项目的全局频度域可以得到 TDB 的全局频度;由 $minsup$ 可以得到 TDB 的全局频繁项目,对项目按其频度由大到小排列(次序记为 $\Re$),形成 db^i 头表;② 再次扫描 db^i,对每一条交易中的所有全局频繁项目,按次序 $\Re$ 组成相应的项目集并插入到 $FP\text{-}tree(db^i)$中;③ 调整各 $FP\text{-}tree(DB^i)(i=1,2,\ldots,n)$;④ 依据性质 1~性质 3 和定理 4,自底向上挖掘 TDB 的全局频繁集;⑤ 将 $FP\text{-}tree(db^i)$合并到 $FP\text{-}tree(DB^i)$.其中,对 $FP\text{-}tree(DB^i)$进行调整可类似于文献[6],分为以下 3 种情况进行讨论:

(1) 全局频繁项目不变且各全局频繁项目保持原有排序;

(2) 全局频繁项目不变但全局频繁项目按全局频度的排列次序发生改变;

(3) 全局频繁项目发生改变.

对于第(1)种情况,无须调整.对于第(2)种情况,不妨设 $x_1,x_2,\ldots,x_s$ 是原 $FP\text{-}tree(DB^i)$的全局频繁项目,排列次序为 $x_1x_2\ldots x_s$.若当局部交易数据库增加时,仅有项目 x_i 与项目 x_j 之间的位置发生了改变($i<j$),则只须对到达 x_j 的所有路径进行调整,依据 FP-tree 结构,通过项目 x_j 找到到达 x_j 的所有路径.对于到达 x_j 的某条路径,设项目 x_j 对应的结点的 count 域值为 q,若路径中有 x_i 与 x_{j-1} 之间的项目存在,且全局频度最大的项目为 x_k,则将项目 x_j 对应的结点及其父结点的 count 域值均减少 q,并将 count 域值为 0 的结点删除;然后将该路径中的项目构成一个项目集,并将项目 x_k 和项目 x_j 的位置互换后重新插入到 $FP\text{-}tree(DB^i)$中,相应结点的 count 域值增加 q.对于有多个项目的位置发生改变情况,可类似地处理.对于第(3)种情况,可类似于第(2)种情况进行处理(先删除,后插入).

对于实例 1,当站点 S^3 上数据发生增量变化(即新增数据库为 db^3)时,项目 f,c,a,b,m,p 的全局频度由 4,4,3,3,3,3 变为 4,6,4,4,4,5,即有多个项目位置发生变化.图 2 的调整方法如下:若选择的后缀项目为 p,则沿着父指针得到到达 p 的路径 $f\rightarrow c\rightarrow a\rightarrow m\rightarrow p$,得到项目集$\{f,c,a,m,p\}$,在删除 p,m 对应的结点且使 f,c,a 对应的结点的 count 值减 1 后,将重新排序后的项目集$\{c,p,f,a,m\}$重新插入到 $FP\text{-}tree(DB^2)$中,但在插入时要注意区分原有路径和新增路径.类似地,得到项目集$\{f,c,a,b,m\}$,删除 f,c,a,b,m 对应的各结点,并将$\{c,f,a,b,m\}$重新插入.

类似于 FDM 算法,采用计数站点(polling site)来优化网络通信,使得在含有 n 个站点的分布式环境中,信息的传送代价为 $O(n)$.FUAGFI 算法描述如下:

FUAGFI 算法.

输入:(1) DB^i 为站点 S^i 上的交易数据库,它含有 D^i 条交易,i=1,2,…,n;

(2) $minsup$ 为最小支持度阈值,且各站点 S^i 的最小支持度阈值均为 $minsup$,i=1,2,…,n;

(3) 由 FMAGF 算法得到的局部频繁模式树为 $FP\text{-}tree(DB^i)$,i=1,2,…,n,全局频繁集为 GFI^{DB};

(4) 假设站点 S^1,S^2,…,S^p 的交易数据库发生了增量改变,增量数据库分别是 db^1,db^2,…,db^p.

输出:全局频繁集 GFI^{TDB}. /* $TDB=DB\cup db^1\cup db^2\cup\ldots\cup db^p$ */

方法:按照以下步骤挖掘全局频繁集.

步骤 1. /* 在站点 S^i 建立 $FP\text{-}tree(db^i)$,i=1,2,…,p */

(1) Scan the transaction database db^i once;

Collect the set of local items F^i and their frequencies;

Get the global frequency of every item from all F^i and the head table of $FP\text{-}tree(DB^i)$;

Get the set of all globally frequent items, denoted as F;

Sort all items in frequency descending order as $\Re$, the list of all items.

Broadcast F. /* 将 F 传送到各站点 */

(2) create($FP\text{-}tree(db^i)$); /* 建立 $FP\text{-}tree(db^i)$ */

(3) generate($LF(db^i)$); /* 生成 db^i 的所有局部频繁项目集 $LF(db^i)$*/

步骤 2. /* 对 $FP\text{-}tree(DB^i)$进行调整 */

(4) adjust($FP\text{-}tree(DB^i)$);

步骤 3. /* 由各 $LF(db^i)$和 GFI^{DB} 挖掘 TDB 的全局频繁项目集 */

(5) let $db^{p+1}=DB$; $LF(db^{p+1})=GFI^{DB}$; /* 把 DB 看成是逻辑站点 S^{p+1} 上的 db^{p+1} */

/* 在站点 S^{p+1},对 GFI^{DB} 中的每一项目集,仅须向站点 S^i 的 $FP\text{-}tree(db^i)$)发出请求 */

(6) k=2; $GFI^{TDB}=commset=LF(db^1)\cap LF(db^2)\cap\ldots\cap LF(db^{p+1})$; /* 由性质 1 对 GFI^{TDB} 进行初始化 */

步骤 4. 重复步骤 4 直到 CF_k 为空为止.

(7) $GFI^{TDB}=GFI^{TDB}\cup F$; $HF_1(db^i)=LF_1(db^i)$;

$$CF_k=\bigcup_{i=1}^{p+1}CF_k(db^i)=\bigcup_{i=1}^{p+1}Apriori\_gen(HF_{k-1}(db^i));$$ /* $HF_k(db^i)\subseteq CF_k(db^i)$ */

(8) for all $X\in CF_k(db^i)-commset$ do

if $X\in LF_k(db^i)$ then

for j=1 to p+1 do

if $polling\_site(X)=S^j$ then

add $\langle X, X.count^i\rangle$ into $CF_k^{i,j}$; /* $CF_k^{i,j}$ 用来存放 S^i 中所有需传送到 S^j 的项目集 */

(9) for j=1,…,p+1 do

send $CF_k^{i,j}$ to S^j; /* 发送候选项目集到计数站点 S^j */

(10) for j=1 to p+1 do { receive $CF_k^{j,i}$; /* 计数站点接受候选项目集 */

for all $X\in CF_k^{j,i}$ do

store X in LP_k^i and update $X.larger\text{-}sites$ in LP_k^i; /* LP_k^i 记录已发送 X 到站点 S^i 的站点地址 */

(11) /* 对 LP_p^i 中的项目集 X,向没有发送 X 到站点 S^i 的其他站点发送请求 */

broadcast polling requests for X to the sites S^j, where $S^j\notin X.larger\text{-}sites$;

/* $X.count^j$ 可方便地从 $FP\text{-}tree(db^j)$的各路径中得到 */

receive $X.count^j$ from the $FP\text{-}tree(db^j)$ of sites S^j, where $S^j\in X.larger\text{-}sites$;

(12) if $i\neq p$+1 then

if $X\notin LF(db^{p+1})$ then /* 对 $LF(db^i)$(i=1,…,p)中的项目集还需向 $FP\text{-}tree(DB^j)$(j=1,…,n)发出请求 */

{ broadcast polling requests for X to all sites S^j(j=1,…,n);

/* $X.count^j$ 可方便地从 $FP\text{-}tree(DB^j)$的各路径中得到 */

receive $X.count^j$ from the $FP\text{-}tree(DB^j)$ of sites S^j;

}

else get $X.count^i$ from $LF(db^{p+1})$;

(13) for all $X \in LP_P^i$ do

{ $X.count=\sum_{i=1}^{n} x.count^i$; /* 计算全局支持度 */

if $X.sup=X.count/D \geq$minsup then insert X into $HF_k(db^i)$;

}

(14) broadcast $HF_k(db^i)$;

receive $HF_k(db^j)$ from all other sites $S^j(j \neq i)$;

$HF_k(TDB)=\bigcup_{i=1}^{p+1} HF_k(db^i)$;

$GFI^{TDB}=GFI^{TDB} \cup HF_k(TDB)$;

步骤 5. /* 将 $FP\text{-}tree(db^i)$合并到 $FP\text{-}tree(DB^i)$ */

(15) $merge(FP\text{-}tree(db^i), FP\text{-}tree(DB^i))$.

算法 FUAGFI 中的(12)~(13)表示 $X \in CF_k(db^i)$,其中 i=1,…,p,因而要得到 X 的全局频度,则需得到所有不同于站点 S^i 的新增和原交易数据库中 X 的频度,同时还需得到站点 S^i 的原交易数据库中 X 的频度.

对于实例 1,当 S^3 的局部数据库交易增量改变(增量交易数据库为 db^3)时,依据算法 FUAGFI,无须传送项目集{c,p},{c,a},{c,m},{a,m},{c,a,m},{c,b,p};同时,为了得到项目集{f,c,a},{f,c,m},{f,a,m},{f,c,a}的频度,仅需请求 $FP\text{-}tree(db^3)$即可,因而有效地减少了网络通信量,提高了全局频繁项目集的更新效率.

可以看出,FUAGFI 在最坏的情况下仅需扫描原局部数据库一遍,扫描新增数据库两遍;利用原局部数据库对应的局部频繁模式树和新建的增量数据库的局部频繁模式树,FUAGFI 无须扫描各局部数据库,便可快速地从各局部频繁模式树的相关路径中获得任意项目集的全局频度.FUAGFI 将原 DB 看成是某一逻辑站点,把由 FMAGF 挖掘出的全局频繁项目集看成是该局部站点对应的局部频繁项目集,通过该局部频繁项目集和各新增局部数据库对应的局部频繁项目集来挖掘全局频繁集.同时,在引入强频繁项目集的概念后,依据性质 1~性质 3 和定理 4,可以有效地减少局部频繁项目集数目和局部频繁项目集的传送量,降低网络通信代价,提高全局频繁项目集的更新效率.

3 实验结果

为了测试算法的性能,我们在 10Mb 的局域网中,用 3 台 PC 机构成分布式站点,各台 PC 机配置均为:操作系统为 Windows2000 server, CPU 为 PII350,内存为 128M.实验数据采用文献[2]的合成数据方法,并将合成的交易数据库均分到 3 个站点,使各站点的 DB^i 记录数相等(i=1,2,3).为了方便叙述,用 DB 表示合成的原交易数据库;D 表示总的交易数;N 表示交易项目的个数;$|T|$表示交易数据记录的平均长度;$|I|$表示最大的潜在频繁项目集的平均长度;$|L|$表示最大的潜在全局频繁项目集数目;$minsup$ 表示最小支持度阈值,而用 $Tx.Iy.DBmk$ 表示原测试数据库实例,其中,$|T|=x$,$|I|=y$,$|DB|=D=m$k.用 VC++6.0 对 FUAGFI,FMAGF 和 FPM 进行了以下 4 组实验(各组实验中,站点 S^1 的增量交易数据库 db^1 的合成参数与该组实验中 DB 的合成参数相同,且 5 个不同的增量交易数据库 db^1 交易数分别为 5k,10k,15k,20k,30k):① 对 D=300K,N=1000,$|L|$=1500,$|T|$=10,$|I|$=4,$minsup$=1%,测试结果如图 4 和图 5 所示;② 对 D=300k,N=1000,$|L|$=1500,$|T|$=12,$|I|$=8,$minsup$=1%,测试结果如图 6 所示;③ 对 D=360k,N=1000,$|L|$=1500,$|T|$=12,$|I|$=8,$minsup$=1%,测试结果如图 7 所示;④ 对 D=360k,N=1000,$|L|$=1500,$|T|$=12,$|I|$=8,$minsup$=2%,测试结果如图 8 所示.由图 4~图 8 可以看出,FUAGFI 的执行效率是 FPM 和 FMAGF 的 3~5 倍,有效地提高了全局频繁项目集的更新效率.另外,我们用某药品销售公司的药品销售数据库作为实验数据对本文提出的算法进

行了验证.验证结果表明,FUAGFI 的执行效率是 FPM 和 FMAGF 的 2~4 倍.限于篇幅,本文对此不作详细叙述.

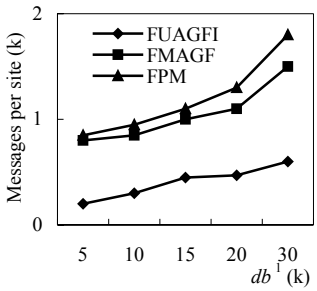

Fig.4 Messages size transmitted (*minsup*=1%)

图 4 网络通信量(*minsup*=1%)

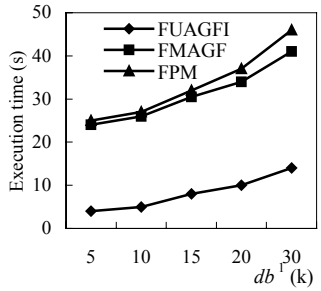

Fig.5 Execution time (*minsup*=1%)

图 5 执行时间(*minsup*=1%)

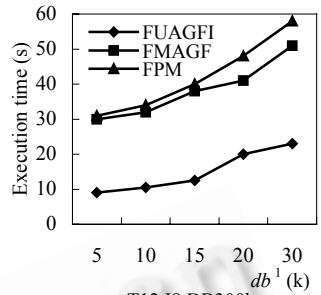

Fig.6 Execution time (*minsup*=1%)

图 6 执行时间(*minsup*=1%)

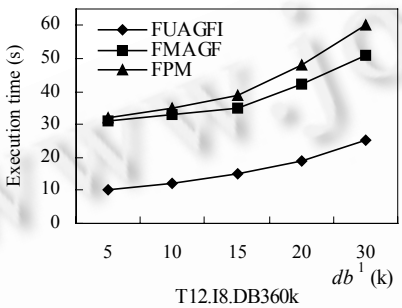

Fig.7 Execution time (*minsup*=1%)

图 7 执行时间(*minsup*=1%)

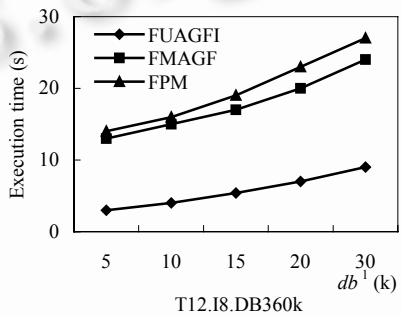

Fig.8 Execution time (*minsup*=2%)

图 8 执行时间(*minsup*=2%)

4 结 语

本文提出了快速更新全局频繁项目集算法FUAGFI.该算法主要考虑数据库记录增加时全局频繁项目集的更新情况.FUAGFI 在最坏情况下仅需扫描原局部数据库一遍,扫描局部新增数据库两遍,且利用已建立的各局部频繁模式树及已挖掘的全局频繁项目集,可以有效地降低网络通信量,提高全局频繁项目集的更新效率.实验结果表明,本文提出的更新算法是有效、可行的.下一步研究的主要目标是,在实际应用中对算法进行进一步的测试,以推广其应用.

References:

[1] Agrawal R, ImielinSki T, Swami A. Mining association rules between sets of items in large database. In: Buneman P, Jajodia S, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1993. 207~216.

[2] Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Bocca JB, Jarke M, Zaniolo C, eds. Proc. of the 20th Int'l Conf. Very Large Data Bases (VLDB'94). Morgan Kaufmann, 1994. 487~499.

[3] Han JW, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: Chen WD, Naughton JF, Bernstein PA, eds. Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data. ACM, 2000. 1~12.

[4] Cheung DW, Han J W, Ng VT, Wong V. Maintenance of discovered association rules in large databases: An incremental updating technique. In: Su SYW, ed. Proc. of the 12th Int. Conf. on Data Engineering. IEEE Computer Society, 1996. 106~114.

[5] Cheung DW, LEE SD, Kao B. A general incremental technique for maintaining discovered association rules. In: Topor RW, Tanaka K, eds. Proc. of the 5th Int'l Conf. on Database Systems for Advanced Applications. World Scientific, 1997. 185~194.

[6] Yang M, Sun ZH. An incremental updating algorithm based on prefix general list for association rules. Chinese Journal of Computers, 2003,26(10):1318~1325 (in Chinese with English abstract).

[7] Feng YC, Feng JL Incremental updating algorithms for mining association rules. Journal of Software, 1998,9(4):301~306 (in Chinese with English abstract).

[8] Park JS, Chen MS, Yu PS. Efficient parallel data mining for association rules. In: Park JS, Chen MS, Yu PS, eds. Proc. of the 4th Int'l Conf. on Information and Knowledge Management. ACM, 1995. 31~36.

[9] Agrawal R, Shafer J. Parallel mining of association rules. IEEE Trans. on Knowledge and Data Engineering, 1996,8(6):962~969.

[10] Cheung DW, Han JW, Ng VT, Fu AW, Fu YJ. A fast distributed algorithm for mining association rules. In: Proc. of IEEE the 4th Int'l Conf. Parallel and Distributed Information Systems. IEEE Computer Society, 1996. 31~44. http://www.sigmod.org/sigmod/dblp/db/conf/pdis/pdis96.html#CheungHNFF96

[11] Cheung DW, Lee SD, Xiao YQ. Effect of data skewness and workload balance in parallel data mining. IEEE Trans. on Knowledge and Data Engineering, 2002,14(3):498~514.

[12] Yang M, Sun ZH, Ji GL. Fast mining of global frequent itemsets. Computer Research and Development, 2003,40(4):620~626 (in Chinese with English abstract).

[13] Yang M, Sun ZH, Ji GL. Algorithm based on distributed database for updating global frequent itemsets. Journal of Southeast University (Natural Science Edition), 2002,32(6):879~883 (in Chinese with English abstract).

附中文参考文献:

[6] 杨明,孙志挥.一种基于前缀广义表的关联规则增量式更新算法.计算机学报,2003,26(10):1318~1325.

[7] 冯玉才,冯剑琳.关联规则的增量式更新算法.软件学报,1998,9(4):301~306.

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

[13] 杨明,孙志挥,吉根林.一种基于分布数据库的全局频繁项目集更新算法.东南大学学报(自然科学版),2002,32(6):879~883.

△△△