第41卷 第4期 西 安 交 通 大 学 学 报 Vol.41 №4
2007年4月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Apr. 2007

分布式数据库关联规则更新算法

宋宝莉[1,2]，覃征[1]
（1. 西安交通大学计算机科学与技术系，710049，西安；2. 深圳市劳动保障局，518029，深圳）

摘要：提出了一种分布式关联规则增量更新算法(IUAAR)，它可对数据库发生变化的情况进行归类.该算法主要采用改进了的FP树结构，通过传送被约束子树来挖掘全局频繁项目集，并充分利用快速分布式挖掘算法建立的各局部FP树，只对新增加了的全局频繁项目修改相应的改进FP树，挖掘其对应的被约束子树，同时利用已挖掘的全局频繁项目集对原全局频繁项目对应的被约束子树进行有效修剪.实验结果表明，该算法的运算速度比快速分布式挖掘算法提高了1倍，在最坏的情况下，对各局部数据库也仅需要扫描一遍，从而可提高数据库的维护效率.

关键词：分布式数据库；全局频繁项目集；约束子树；增量更新
中图分类号：TP301 **文献标识码**：A **文章编号**：0253-987X(2007)04-0416-05

Updating Mining Algorithm for Distributed Association Rules

Song Baoli[1,2], Qin Zheng[1]
(1. Department of Computer Science and Technology, Xi′an Jiaotong University, Xi′an 710049, China;
2. Shenzhen Labor and Social Security Bureau, Shenzhen 518029, China)

Abstract: A new algorithm IUAAR (incremental updating algorithm for association rules) is introduced, by which the change of database records can be classified. The improved FP-tree structure is adopted and the global frequent itemsets are mined by transmitting constrained sub-tree. Utilizing the local FP-tree created by FDMA (fast distributed mining algorithm) only the FP-tree of the added global frequent items is modified. Moreover, using the mined results, the constrained sub-trees of the incremental global frequent itemsets that are transmitted in network are mined. The constrained sub-trees of the original global frequent itemsets can be pruned without transmitting them. Experiments show that in the worst case, IUAAR only scans every local transaction database once, thus the communication cost is dramatically decreased and the maintenance efficiency of global frequent itemsets is improved, and the mining speed of IUAAR algorithm is increased by at least two times in comparison with FDMA.

Keywords: distributed database; global frequent itemset; constrained sub-tree; incremental updating

挖掘关联规则是数据挖掘研究的一个重要方面，而频繁模式的挖掘是关联规则挖掘任务中的主要步骤[1]，涉及这方面的研究主要是针对集中式频繁项目集的挖掘[1]与更新[2-3]，著名算法有计数分步算法(CD)[4]等，而在分布式环境下有分布式双决策挖掘器算法(DDDM)[5]及快速分布式算法(FDMA)[6]等.它们的目标是降低网络通讯开销，并在一定程度上提高挖掘效率，但对多项目的海量数据库仍不可避免地产生大量的局部候选项目集，使得通讯开销问题没有得到真正的解决.FDMA采用改进的FP树存储结构，通过传输被约束子树来挖掘全局频繁项目集，有效地降低了网络通讯开销，提高

收稿日期：2006-08-10. 作者简介：宋宝莉(1964～)，女，博士生，高级工程师；覃征(联系人)，男，教授，博士生导师.
基金项目：国家自然科学基金资助项目(60542004).

了全局频繁项目集的挖掘效率，但如何对全局频繁项目集进行高效挖掘、更新却无详细报道. 目前，常用的频繁项目维护算法，如频繁项集增量更新(FUP)算法等[2-3,7]，仅适应于单机环境，而全局频繁项目集更新算法(UAGFI)[8]适用于分布式环境，但由于其会生成大量的条件频繁模式树，所以网络代价比较高.

本文针对上述问题，提出了一种分布式关联规则增量更新算法(IUAAR)，它考虑了数据库记录发生变化时的全局频繁项目集的更新情况，在最坏的情况下，仅须对局部数据库扫描一遍，并能充分利用已挖掘的结果，从而避免了传送原全局频繁项目对应的被约束子树，降低了网络通讯代价.

1　相关概念和结论

1.1　全局频繁项目集

本文沿用了文献[6]的记号，而且模型中的分布数据库是水平划分的，各部分的数据库模式逻辑同构. 有 n 个站点 $S^1,S^2,\cdots,S^n$，相应的成员数据库分别为 $DB^1,DB^2,\cdots,DB^n$，$DB=\bigcup_{i=1}^{n} DB^i$，而 D 及 D^i 分别表示 DB 及 DB^i 中的交易数. 对某一项目集 X，$X.\mathrm{count}$ 及 $X.\mathrm{count}^i$ 分别表示在 DB 及 DB^i 中含 X 的交易数.

定义 1　若 $X.\sup(X.\mathrm{count}/D)$ 及 $X.\sup^i$ $(X.\mathrm{count}^i/D^i)$ 分别表示 X 在 DB 及 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 $X.\sup\geqslant\sup_{\min}$(最小支持度阈值)，则 X 为全局频繁项目集；若 $X.\sup^i\geqslant\sup_{\min}$，则 X 为站点 S^i 的局部频繁项目集.

引理 1　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[8].

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.

性质 1　若 X 为全局频繁项目集，则存在一站点 S^i $(1\leqslant i\leqslant n)$ 使得 X 及 X 的所有非空子集在站点 S^i 上为局部频繁项目集.

证明　若将 n 个站点看成是 n 个鸽巢，项目集 X 在 DB 中出现的次数对应于鸽子的个数，则 $X.\mathrm{count}\geqslant(D^1+D^2+\cdots+D^n)\sup_{\min}$. 由鸽巢原理知，至少存在一个站点 S^i 使得 $X.\mathrm{count}^i\geqslant D^i\sup_{\min}$，即 X 在站点 S^i 是局部频繁项目集. 由引理 1 得，X 的所有非空子集在站点 S^i 上为局部频繁项目集，证毕.

可见，全局频繁项目集是从局部频繁项目集中获得的，而减少局部频繁项目集引起的通讯代价是提高效率的关键.

1.2　FP′-Tree 及被约束子树

文献[6]中 FP′树是改进的 FP 树，与传统 FP 树不同之处为：①FP 树是双向的，而 FP′树是单向的，包含较少的指针，可节省 1/4 的空间；②FP 树的结点采用项标记，而 FP′树的结点采用项的序号标记，且包含 4 个字段(项的序号 order、支持度计数 count、指向最左子女结点或父结点的指针 ahead 和指向右兄弟结点或结点链中下一结点的指针 next).

FP′树中所有被 $\{k_i,\cdots,k_2,k_1\}$ 约束的子路径组成的子树称为被 $\{k_i,\cdots,k_2,k_1\}$ 约束的子树，记作 $\mathrm{ST}(k_1,k_2,\cdots,k_i)$，它是 FP′树的一条包含根的特殊子树. 该子树可用指针数组 branch[]表示且指向一条被约束子路径的端点，用整型数组 base-count[]记录每个被约束子路径的基本频度计数，用一个整型数组 $\mathrm{ST}(k_1,k_2,\cdots,k_i).\mathrm{count}[]$ 记录 $\mathrm{ST}(k_1,k_2,\cdots,k_i)$ 中具有相同序号的结点频度计数和.

1.3　关联规则的更新模式

通常把关联规则的更新挖掘分为以下 5 种情形来处理[4].

情形 1　新支持度阈值 $\sup_{new}$ 小于原支持度阈值 $\sup_{old}$，这种情形可能会产生新频繁项目，也可能出现原频繁项目计数增加.

情形 2　$\sup_{new}$ 大于 $\sup_{old}$，这种情形可能会删除新频繁项目，也可能出现原频繁项目计数减小.

情形 3　向交易数据集中增加新交易数据，在支持度保持不变的条件下可能产生新频繁项目，也可能出现原频繁项目计数增加，或转化为情形 1.

情形 4　从交易数据集中删除旧交易数据. 由于新交易数据集 DB′比原交易数据集 DB 的交易数据要少，因此在支持度保持不变的条件下可能会删除原频繁项目，也可能出现频繁项目计数减小，这类似于情形 2.

情形 5　根据已发现的规则集 R_1，利用知识推理手段，通过扫描数据集 DB 形成新规则集 R_2.

前 4 种情形实际可归为 2 种情形，即情形 1 和情形 2.

对于情形 5，知识库的知识可用规则表示，它的

知识一部分是原有的，另一部分是推理机在扫描了数据集 DB 之后，利用现有的知识(R_1)生成新的知识(R_2)并追加进知识库，由此关联规则得以更新. 这里不再赘述. 下面只讨论情形 1 和情形 2.

2 IUAAR 算法

2.1 FDMA 算法思路

为了叙述方便，将 DB 对应的 FP′树记为 F_{tree}，DB^i 对应的 FP′树记为 $F^i_{tree}(1\leqslant i\leqslant n)$，所有的全局频繁项目组成的集合记为 F，从 DB 挖掘得到的全局频繁项目集所组成的集合为 L. 对于给定的 sup_{min}，FDMA 算法挖掘全局频繁集的任务可分为 2 部分：①建立 F^i_{tree}及相应的被约束子树$ST^i(j)$；②挖掘各 F^i_{tree}的被约束子树 $ST^i(j)(F^i_{tree}|ST^i(j))$，获得全局频繁集. 文献[6]通过引入下面的定理 1 和定理 2，并将分布式数据库的全局频繁项目集的挖掘转化为基于各约束子树的挖掘，即对 $F_{tree}|ST(a)$进行全局频繁项目集的挖掘，而 $F_{tree}|ST(a)$的全局频繁项目集的挖掘可从 $F^i_{tree}|ST^i(a)$的局部频繁项目集中得到.

定理 1 若 $L(F_{tree}|ST(a))(a\in F)$为对 FP 树的被约束子树 $ST(a)$进行挖掘而生成的全局频繁项目集，则 $L=\bigcup_{a\in F}L(F_{tree}|ST(a))$.

定理 2 若 $L^i(F^i_{tree}|ST^i(a))(a\in F)$ 为对 F^i_{tree}的被约束子树 $ST^i(a)$进行挖掘而生成的局部频繁项目集，则 $L(F_{tree}|ST(a))\subseteq\bigcup_{i=1}^{n}L^i(F^i_{tree}|ST^i(a))$.

证明 若 $X\in L(F_{tree}|ST(a))$，即 X 是全局频繁项目集，则由性质 1 知，存在一站点 $S^i(1\leqslant i\leqslant n)$使得 X 为局部频繁项目集. 由在 S^i 上对 F^i_{tree}的 $ST^i(a)$挖掘的生成过程知，$X\in L^i(F^i_{tree}|ST^i(a))$，证毕.

由于 FDMA 算法采用传送被约束子树(可用 3 个很小的数组来表示)的方式来挖掘全局频繁项目集，避免了传送大量候选项集或频繁模式树，从而显著降低了网络通讯代价，提高了全局频繁项目集的挖掘效率.

例 1 设 3 个站点 S^1、S^2 及 S^3 上的交易数据库分别为 DB^1、DB^2 及 DB^3，如表 1 所示.

对 $sup_{min}=0.5$，由算法 FDMA 建立的 F^1_{tree}和 F^3_{tree}如图 1、图 2 所示. 为了便于维护全局频繁项目集，在各局部频繁模式树上的头表中保留所有项目的全局频度. 下面通过求 $F_{tree}|ST(6)$得到全局频繁项目集来说明 FDMA 算法. F^3_{tree}中的 $ST^3(6)$如图 3 所示.

表 1 站点 S^1、S^2 及 S^3 上的交易数据库数据

数据库	标识	交易
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

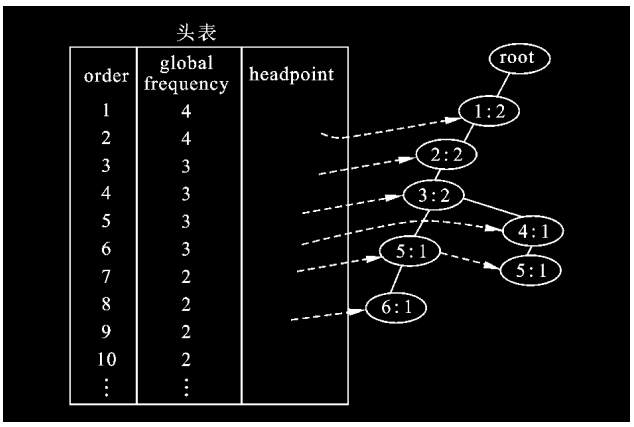

图 1 F^1_{tree}

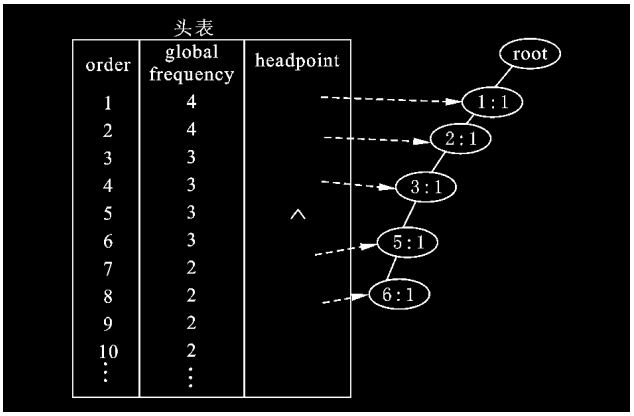

图 2 F^3_{tree}

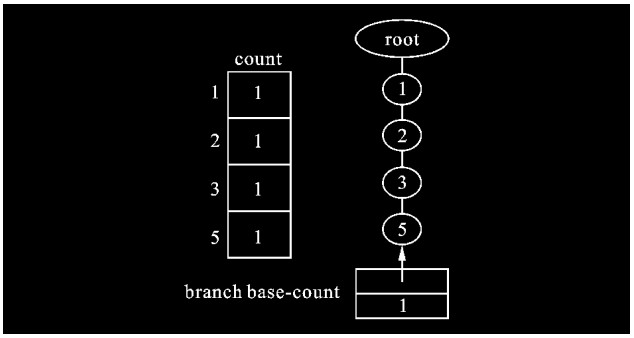

图 3 F^3_{tree}的被束子树 $ST^3(6)$

按照 FDMA 算法的通讯策略，将 $ST^3(6)$传送到站点 S^1，并在 S^1 上进行挖掘便可得到全局频繁项目集$\{f,c,a,m,p\}$及全局频繁项目子集. 通过 DDDM 算法挖掘全局频繁集，需将 $F^3_{tree}|p$ ({(f:1,

$c:1,a:1,m:1)\}|p)$生成的局部频繁项目集 $\{f,p\}$、$\{c,p\}$、$\{a,p\}$、$\{m,p\}$、$\{f,c,p\}$、$\{f,a,p\}$、$\{f,m,p\}$、$\{a,c,p\}$、$\{c,m,p\}$、$\{a,m,p\}$、$\{f,c,a,p\}$、$\{f,c,m,p\}$、$\{f,a,m,p\}$、$\{f,c,a,m,p\}$传送到 S^1，得到全局频繁项目集$\{f, c, a, m, p\}$及其全局频繁项目子集. 显然，FDMA 网络开销比 DDDM 算法要小.

FDMA 算法的思路详见文献[6]，而如何对由 FDMA 挖掘出的全局频繁项目集进行高效维护是本文的主要目标.

2.2　全局频繁项目集的维护算法

为了发现事先未知的全局频繁集，用户必然要通过 sup_{min}的不断调整来逐步聚焦到那些真正令人感兴趣的全局频繁项目集，显然这是一个动态的交互过程. 因此，本文提出了全局频繁项目集的维护算法——IUAAR.

设 sup_{new}为新的最小支持度阈值，sup_{old}为原最小支持度阈值.

当 $sup_{new} > sup_{old}$时，仅需对原全局频繁集 L 进行筛选，从 L 中删除全局支持度小于 S'的项目集，剩下便为挖掘的全局频繁集. 因而，IUAAR 算法主要是在 $sup_{new} < sup_{old}$下进行的.

当 $sup_{new} < sup_{old}$时，F 中的项目仍为全局频繁项目，而 F'-F 中的有些项目将成为全局频繁项目，不妨将这些新的全局频繁项目组成的集合记为 F''.

把全局频繁集的维护工作分为 3 步：①扫描 $DB^i(1\leqslant i\leqslant n)$，将每一交易中属于 F 或 F''的项目按频度由大到小组成项目序号集，并将其对应的项目序号插入到 F^i_{tree}中，在插入时，属于 F 的项目对应结点的 count 值不变，F''的项目的处理类似 FDMA 算法；②对属于新的全局频繁项目对应的约束子树进行全局频繁集挖掘；③经对 F 中的项目所对应的约束子树筛选后再进行全局频繁项目的挖掘.

IUAAR 算法步骤描述如下.

输入：①DB^i 为站点 S^i 上的交易数据库，其含有 D^i 条交易，$i=1,2,\cdots,n$；②sup_{old}为原最小支持度阈值，sup_{new}为新的最小支持度阀值. 各站点 S^i 的最小支持度阈值均为 sup_{new}，且 $sup_{new} < sup_{old}$，$i=1,2,\cdots,n$；③L 及 F^i_{tree}分别为 sup_{old}下的全局频繁项集和 FP′-Tree，$i=1,2,\cdots,n$；④F 及 F' 分别为 sup_{new}和 sup_{old}下的全局频繁项目组成的集合.

输出：全局频繁集 L'.

步骤 1：修改各站点 S^i 的 F^i_{tree}，即：

从 F'-F 收集全局频繁项集 F''；

对 $F\cup F''$ 按支持度降序 $\mathcal{R}$ 排序；

for $i=1$ to n do；

Generate F^i_{tree} // 在此过程中，要生成“项目-序号，序号-项目”转换表，树的结点用项的序号表示，并反复将项目集序号插入到 F^i_{tree}中求得转换表（该过程类似 FDMA 算法生成 F^i_{tree}的过程，不同的是属于 F 的项目所对应的结点的 count 值不变，而为属于 F''的项目新增结点或 count 值加 1).

步骤 2：快速维护全局频繁集，即：

For each $a\in F''$ do；// 对 F''中的项目所对应的被约束子树进行全局频繁集挖掘

$L_a=$mining($F_{tree}|ST(a)$)；// 类似 FDMA 算法的步骤 2

Broadcast L_a；// L_a到其他站点

Receive L_b from other sites S^j，$(j\neq i)$；// 在新的最小支持度阈值 S'下，一些非全局频繁项目集 X $(X\subseteq F)$变成了全局频繁项目集

for each $a\in F$ do；

$L_a=$mining($F_{tree}|ST^i(a)$)$-L$；// 类似 FDMA 算法的步骤 2，但将过程 FST_generate 调用改为 NFST_generate 调用

Broadcast L_a；// L_a 到其他站点

Receive L_b from other sites $S^j(j\leqslant\geqslant i)$；

Return $L=\bigcup\limits_{a\in F\cup F''}L_a\cup L$.

NFST_generate 描述如下.

Procedure NFST_generate(Tree，α，a)

{

collect combination(denoted as β) of all the nodes in the path P；

if itemset $X=\alpha\cup\beta$ is not in L then；// 对被约束子树修剪，减小网络通讯量

for $t=$max-order down to 1 do {；// max-order 是最大序号

build $ST^i(t)$ based on F^i_{tree}；

insert $ST^i(t)$ into tree β；

add $\langle\alpha,\beta,count^i\rangle$ into $F^{i,j}_{tree}|ST^i(t)$；

}}// 用 $\beta|\alpha$ 表示 α 的被约束子树

对于例 1，当 sup_{old}由 0.5 减小到 0.3，项目 l，o，k，s 变为全局频繁项目，即 $F''=\{l,o,k,s\}$，扫描各 DB^i 一遍，修改相应的 $F^i_{tree}(i=1,2,3)$. 按 IUAAR 算法中的步骤 2 对全局频繁项目集进行维护，结果使得全局频繁项目的挖掘效率明显提高.

3　实验结果

为了进一步测试算法的性能，采用文献[1]的数

据合成方法合成一个交易数据库集 DB，并将该数据库集均分到 3 个站点. 其中，总的交易数为 $D=390\ 000$，项目数为 $N=1\ 000$，潜在的全局频繁项目数 $|L|=1\ 500$. 在操作系统为 Windows 2000 Server 的局域网中，由 3 台 CPU 为 PⅣ350，内存为 128 MB 的 PC 机构成分布式站点，且各站点的 DB^i 记录数相等. 用 VC++6.0 对 DDDM、FDMA 和 IUAAR 算法进行测试，结果如图 4～图 6 所示. 由图 4、图 5 知，由于 IUAAR 有效地利用了最小支持度阈值为 2%下挖掘的全局频繁集，因而有效地提高了全局频繁项目集的维护效率.

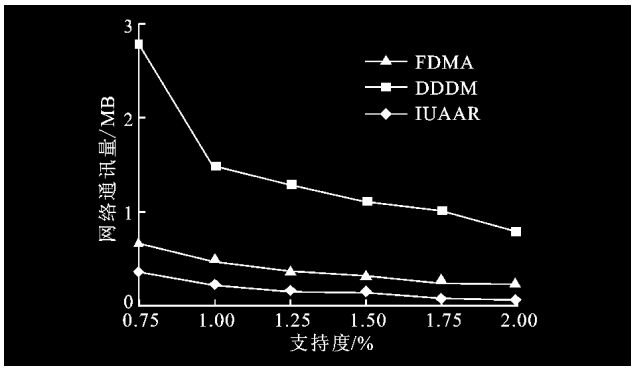

图 4 网络通讯量

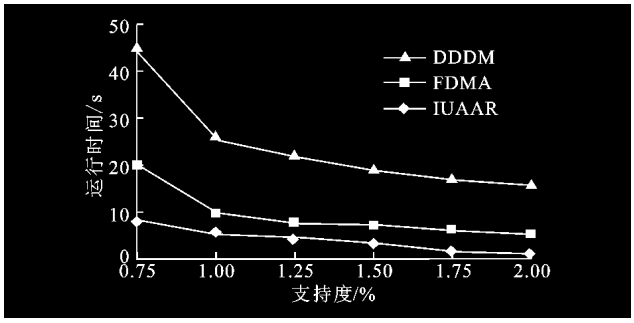

图 5 算法的执行时间

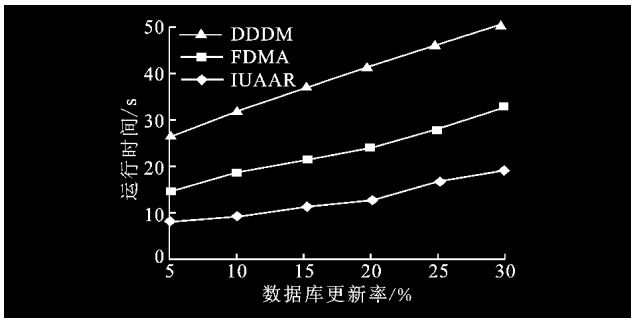

图 6 算法的执行时间(支持度为 1%)

4 结束语

本文在算法 FDMA 的基础上，提出了一种分布式数据库关联规则更新算法，它在最坏的情况下仅对各局部数据库扫描一遍，并可利用已挖掘的结果，从而避免了传送某些原全局频繁项目对应的被约束子树，降了低网络通讯代价. 实验结果表明 IUAAR 算法是有效、可行的.

参考文献：

[1] Agrawal R, Srikant R. Fast algorithms for mining association rules [EB/OL]. [2006-05-11]. http://www.rsrikant.com/papers/vldb94.pdf.

[2] Cheung D W, Han J W, Ng V T, et al. Maintenance of discovered association rules in large databases: an incremental updating technique [C]// Proceedings of the 12th International Conference on Data Engineering. Los Alamitos: IEEE Computer Society, 1996: 106-114.

[3] 易彤，徐宝文，吴方君. 一种基于 FP 树的挖掘关联规则的增量式更新算法 [J]. 计算机学报，2004，27(5)：703-710.
Yi Tong, Xu Baowen, Wu Fangjun. An FP-tree based incremental updating algorithm for mining association rules [J]. Chinese Journal of Computers, 2004, 27(5):703-710.

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

[5] Schuster A, Wolff R. Communication efficient distributed mining of association rules [C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM, 2001:473-484.

[6] 宋宝莉，覃征. 分布式全局频繁项目集的快速挖掘方法 [J]. 西安交通大学学报，2006，40(8)：923-927.
Song Baoli, Qin Zheng. Fast mining algorithm for distributed global frequent itemsets [J]. Journal of Xi'an Jiaotong University, 2006, 40(8):923-927.

[7] 冯玉才，冯剑琳. 关联规则的增量式更新算法 [J]. 软件学报，1998，9(4)：301-306.
Feng Yucai, Feng Jianlin. Incremental updating algorithms for mining association rules [J]. Journal of Software, 1998, 9(4):301-306.

[8] 杨明，孙志挥，宋余庆. 快速更新全局频繁项目集 [J]. 软件学报，2004，15(08)：1189-1197.
Yang Ming, Sun Zhihui, Song Yuqing. Fast updating of globally frequent itemsets [J]. Journal of Software, 2004, 15(8):1189-1197.

(编辑 苗凌)