一种新的基于投影的频繁模式树构造算法*)

李陶深[1] 李新仕[1,2]

(广西大学计算机与电子信息学院 南宁 530004)[1](广西财经学院计算机与信息管理系 南宁 530003)[2]

摘 要 本文分析FP-growth算法存在的主要问题,提出了一种新的基于投影的频繁模式树构造算法。该算法充分利用大型数据库的投影运算能力,按层来构造频繁模式树(FP-tree),有效地解决了传统的FP-tree构造中存在的问题。实验结果表明,本文的算法与传统的频繁模式树的构造算法相比,具有比较好的时间和空间的可伸缩性。

关键词 数据挖掘,关联规则,频繁模式树,投影后插式频繁模式树

1 引言

数据挖掘指的是从大量的、不完全的、有噪声的、模糊的、随机的数据中提取默认在其中的、事先不知道的、潜在具有价值的有用信息和知识的过程[1]。关联规则的挖掘问题于1993年由Rabesh Agrawal等人首先提出[2],它作为数据挖掘中的一个重要领域,用于从大量数据中发现项集之间的有趣关联或相关联系[1]。目前,关联规则挖掘方法的研究大多集中在基于频繁模式树的关联规则挖掘算法的研究中,主要的算法是Apriori算法[3]和FP-growth算法[4],以及它们的改进算法。

FP-growth算法是一个本质上不同于Apriori算法的挖掘频繁模式的有效方法,它不生成候选频繁项集且只需要两次扫描数据库,有效克服了Apriori算法的缺点。在FP-growth算法研究领域中,基于FP-tree的频繁闭包项目集挖掘算法主要是Jian Pei等提出的Closet算法[5]和Closet算法[6],之后的闭包项目集挖掘算法主要都是在此基础上进行改进的,例如文[7,8]中提出的算法。除了挖掘最大频繁项目集和挖掘频繁闭包项目集之外,许多研究人员对频繁模式树的构造和挖掘过程也进行了大量的研究。文[9]基于共同事务频繁项目树提出的OFI-tree Mining算法,通过有效的剪枝节约大量的内存空间;文[10]提出了一种基于被约束子树挖掘频繁项集的有效算法,通过引入了被约束子树,在挖掘频繁模式时不生成条件FP树,从而提高频繁模式挖掘的时空效率;文[11]中的CFP-tree是对FP-tree的进一步压缩存储,并对交易数据库进行投影分割,然后分别进行关联规则的挖掘。

虽然FP-growth算法克服了Apriori系列算法的缺陷,取得了很好的效果,但它本身仍然存在着不足。该算法是通过逐步生成条件模式基和条件频繁模式树来挖掘频繁项目集的,在频繁项目集和事务量比较大的情况下,需要动态地产生和释放成千上万的条件频繁模式树,因此占用大量的时间和空间。另外,条件模式基的生成需要不断地搜索FP-tree,原因在于:FP-tree是自顶而下构造的,而条件模式基的生成是自底而上的顺序来产生的。如果FP-tree采用双向指针结构,则需要占用了更多的存储空间。本文提出了一种新的基于投影的频繁模式树后插式构造方法,支持自底而上的指针链,以满足产生条件模式基的需要,可有效地解决了上述问题。

2 基于投影的频繁模式树后插式构造方法

本文对频繁模式树的构造方法进行改进,以减少搜索父结点的时间,同时又能克服上述问题。我们把改进后的构造方法生成的频繁模式树称为投影后插式频繁模式树(Projection Rear-inserting Frequent Pattern Tree ,PRIFP-tree)。

2.1 构造PRIFP-tree的基本思想

由构造FP-tree的过程中可知,传统的FP-tree的构造是一个严格的串行计算的过程,当事务数据库中的数据量很大时,会造成性能的瓶颈。本文提出的基于投影的频繁模式树后插式构造方法的基本思想主要有两个方面:

(1)充分利用了大型数据库的投影运算能力,解决传统FP-tree的构造算法中随着数据库中记录的增加,算法性能急剧下降的瓶颈问题。利用数据库的投影运算,按层次来构造频繁模式树,不会造成随着数据库记录增加算法性能急剧下降的缺点,算法的时间复杂度主要与投影的次数有关系。首先创建一个临时数据库用于存放所有排序后的频繁项,这个临时数据库我们称之为投影数据库(PDB),可以被用来运用投影技术来构造树的结点,而不像传统的FP-tree那样逐个地计算每个频繁项。然后对

*)基金项目:广西"新世纪十百千人才工程"专项基金项目(桂人字2001213号)和广西自然科学基金项目(桂科自0229008)联合资助。

PDB进行投影计算，一次投影两列，j-1列和j列，第j列被称为当前结点列，用来统计所有不同频繁项的出现次数，第j-1列被称之为父结点列，用以判断当前结点的父结点。这样，一次就可以计算出树的一层结点并把它们链接到对应的父结点上，不必要逐个地计算每个频繁项。

⑵在构造频繁模式树时，采用后插式的方法，以便解决随着频繁模式树深度增加、搜索插入点(父结点)的时间不断增加的问题。在重构FP-tree时，将FP-tree形成一个链表，在链表中的结点按支持度的降序排列。数据库中的频繁事务项是严格按照支持度从高到低排列的，新一次投影得到的分枝结点在一般情况下会比前一次投影得到的分枝结点的支持度低，所以当按层构造FP-tree时，新一层结点一般都插入到链表的尾部，而其父结点为上一层，搜索其父结点也不会花费很大的代价。

2.2 PRIFP-tree的构造算法

假设算法使用的树结构中的每个结点有四个域，分别为：item_name(存放项目值)、count(支持度计数)、parent(指向父结点)、node_link(指向前一个结点)。下面给出PRIFP-tree的构造算法。

输入：事务数据库DB，最小支持度阈值ξ

输出：PRIFP-tree

Step1：扫描事务数据库DB一次，收集频繁项的集合F和它们的支持度。对F按支持度降序排序，结果为频繁项表L。

Step2：⑵根据L，对DB的每个事务中的频繁项按L中的顺序进行排序，排序后的结果存放在临时投影数据库PDB中。

Step3：创建根结点。假设j为列号，对PDB中的每一列顺序执行以下操作，直到投影完PDB中所有的列：

if j=1 then {

① 对第1列作投影，并计算该列中每个频繁项的出现次数，结果形为[root,q:n](其中q为频繁项名，n为累计出现次数，root代表该结点的父结点是根结点)；

② 将这些频繁项[q:n]作为root的子结点加入到PRIFP-tree中；

}

else {

①对第j－1列和第j列两列进行投影，统计所有父结点和当前列结点相同的二元频繁项组的出现次数，结果形为二元频繁项组[p,q:n](n为累计出现的次数)，并对二元频繁项组按q的支持度升序排序；

②对上面的二元频繁项组进行比较，若存在二元频繁项组中的q相同，则将它们进行区别，即在q的后面加上后缀(顺序号，从1开始编号)，并修改PDB相关的项目；

③对二元频繁项目组[p,q:n]的每一个二元组进行以下操作：从表尾(rear)开始向前搜索直到某个结点的值与q相等(或者某一个结点的支持度大于q)为止，产生新结点，将新结点插入到该结点之后，将item_name域设置为q，将count域设置为n，然后继续往前搜索直到某一个结点值为p为止，该结点即为新结点的父结点，将新结点的parent域指向父结点。

}

j=j+1;

2.3 PRIFP-tree的构造过程

下面通过一个实例，说明PRIFP-tree的构造过程。所用的事务交易数据库如表1所示。

表1 事务交易数据库

TID	事务项	排序后的频繁事务项(PDB)	投影修改后的PDB
100	f,a,c,d,g,l,m,p	f,c,a,m,p	f,c,a,m,p
200	a,b,c,f,l,o	f,c,a,b,o	f,c,a,b,o
300	b,f,h,j,m,p	f,b,m,p	f,b1,m1,p
400	b,c,k,m,o,s	c,b,m,o	c,b2,m2,o1
500	a,f,c,e,l,n,o,p	f,c,a,o,p	f,c,a,o2,p

由上述算法，可以看出PRIFP-tree的构造思路是：

(1) 第一遍扫描数据库，得到各项目的出现频度(支持度)如下：a:3,b:3,c:4,d:1,e:1,f:4,g:1,h:1,i:1,j:1,k:1,l:2,m:3,n:1,o:3,p:3,s:1 (":"后的数字表示支持度)

(2) 取最小支持度阈值ξ=3，得到频繁项目集L并按支持度由高到低排列如下(支持度相同按先后顺序)：L=[f:4,c:4,a:3,b:3,m:3,o:3,p:3]

(3) 根据L，对DB的每一个事务中的频繁项按L中的顺序进行排序，排序后的结果存放在临时的投影数据库PDB中。

(4) 最后利用PDB进行投影，最后得到如图1所示的PRIFP-tree。

3 性能分析

3.1 PRIFP-tree构造算法的性质

从以上PRIFP-tree的构造过程，可以得出PRIFP-tree的构造算法具有以下性质：

引理1 给定一个事务数据库DB和它的最小支持度阈值ξ，它对应的PRIFP-tree包含了数据库中用于挖掘所有的频繁项的信息。

证明：在FP-tree的构造过程中，事务数据库中的每一个事务都对应着FP-tree的一条路径，而每一个事务中频繁模式项的信息完全保存在FP-tree中。另外，在FP-tree中的每个路径中频繁项集有可能被多个事务所共享，因为所有的事务路径都必

须从 root 结点开始，它们有可能共享前缀路径。因此 FP-tree 包含了数据库中用于挖掘所有频繁项的信息。而 PRIFP-tree 仅仅对 FP-tree 的指针方向进行改变，因此与 FP-tree 一样，数据库中的所有频繁项模式信息被完整地保存到了对应的 PRIFP-tree 中。证毕。

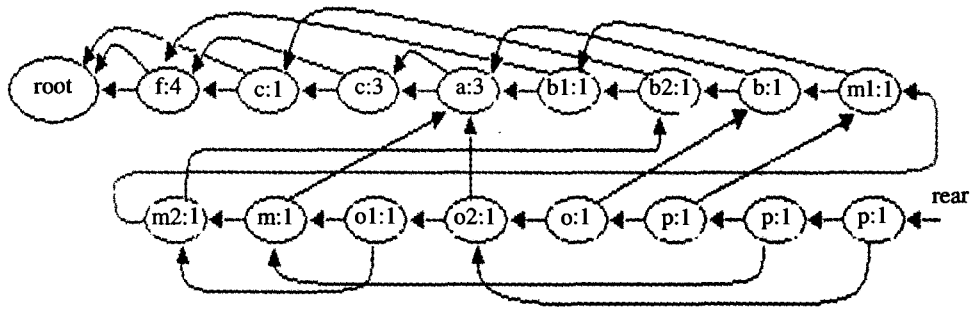


图 3　PRIFP-tree

引理 2　对第 j－1 和第 j 列进行投影得到的序列如果按照第 j 列的支持度由小到大进行排列，然后按顺序插入到 FP-tree 中，不会产生异常。

证明：由于频繁项目集中的每个事务中的项目是按支持度从大到小排列的，因此对第 j－1 列和第 j 列进行投影时得到的分枝集合中的任一个分枝（设为[x,y]），x 的支持度比 y 的支持度大。若分枝集合按第 j 列的支持度从小到大排列，则对于任一个分枝[x,y]，x 是 y 的父结点，而 x 作为孩子的分枝（设为[z,x]），其必排列在分枝[x,y]之后，因此分枝[x,y]优先插入，即插入时往前搜索到的第一个 x 结点不可能是同层结点而是上一层结点，因此不会产生异常。证毕。

3.2　实验分析

本文的测试主要针对频繁模式树的构造及频繁模式基的产生这个阶段。实验环境为：P4 1.4GHZ CPU，256MB 内存，Windows XP 操作系统，采用 Delphi 7.0 进行编程。为了分析不同数据库系统对 PRIFP-tree 构造性能的影响，数据库系统分别采用 SQL server 2000 和 Visual Foxpro 6.0 进行测试。为了测试算法的运行时间和占用存储空间随着支持度减小而变化的情况，采用的模拟数据库共有 3000 个事务，200 个项目，事务平均长度为 10，运行时间随支持度变化情况如图 2 示。

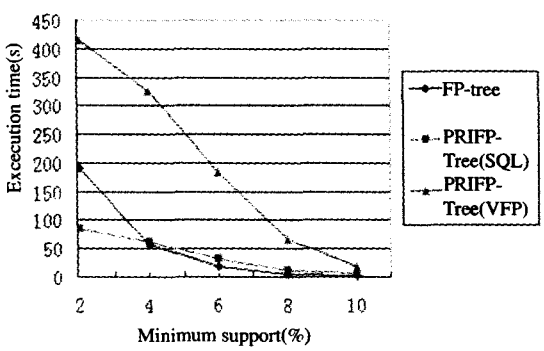
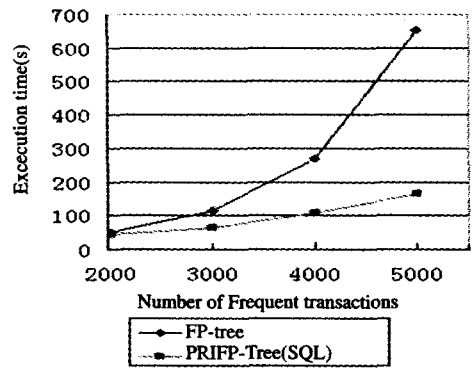


图 2　运行时间与支持度的关系

为了测试算法随着事务数增加而变化的情况，采用的模拟数据库共 200 个频繁项目，频繁事务的平均长度为 6，实验结果如图 3 所示。



分析以上实验结果可得出以下结论：随着支持度的减少，无论是频繁项目集和频繁事务长度的增加，还是随着事务数的增加，FP-tree 的结点数都增长比较快，因此搜索 FP-tree 产生条件模式基的时间也相应地增长比较快。而基于 PRIFP-tree 的算法在产生条件模式基时不需要搜索频繁模式树，因此花费的时间主要集中在构造频繁模式树时需要不断地对数据库进行更新，但所花费的时间的增长速度却比较平缓。

结束语　本文提出一种基于投影的频繁模式树后插式构造方法（PRIFP-tree 构造方法），通过对传统的频繁模式树进行重构，解决了频繁模式树的自顶而下构造与频繁模式自底而上挖掘的矛盾。新构造方法主要运用大型数据库的投影技术，充分利用计算机的内存和计算能力，按层来生成频繁模式树，克服了传统频繁模式树构造中的缺点，即传统的频繁模式树的构造是一个严格的串行计算过程。

实验结果表明：基于投影的频繁模式树后插式构造方法与传统的频繁模式树的构造方法相比较，具有比较好的时间和空间的可伸缩性。

参 考 文 献

1　Han J, Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufman, San Francisco, CA, 2001

（下转第 177 页）

根据指定的接收管理站，打开一个 udp 端口（默认值 162），然后进入一个无限循环，等待接收 trap 信息，系统出现差错或异常时，接收邮箱发送的 trap 信息，包括系统时间、 trap 类型以及产生 trap 的 OID 和值。然后将 trap 信息绑定到 trap pdu 变量绑定列表中，生成报文，并用将报文发送到管理站。最后释放资源，返回循环头部，等待接收下一条 trap 信息。

5 结果分析

笔者使用 AdventNet 公司的 MibBrowser，模拟 SNMP 管理站。通过测试，该设计能够及时响应管理站发出的请求，并在系统异常时，向管理站发送告警信息。结果如图 4 所示。

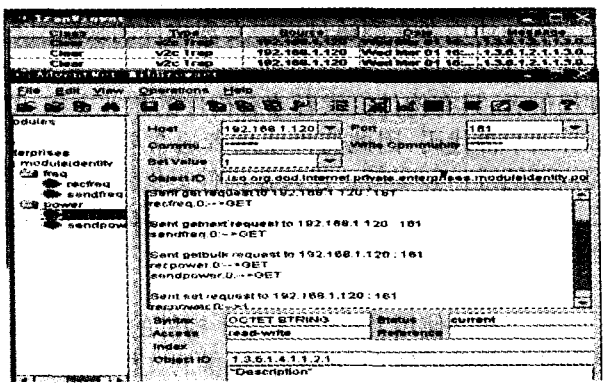


图 4 测试结果

结束语 该设计不但在计算机上得到仿真验证，而且在具体工程实践中得到应用。μC/OS-Ⅱ只是一个实时操作系统的内核，没有网络、文件系统等额外模块。对网络功能要求比较高的 SNMP 代理系统而言，μC/OS-Ⅱ确实不如嵌入式 Linux、VxWorks 等具有强大网络功能的操作系统方便，虽然可以移植 TCP/IP 协议栈，但高层开发依然十分繁琐，特别是 SNMPv3 代理。笔者实现了 SNMPv2 代理，若要实现 SNMPv3 代理，以下方法可供参考：

（1）使用 FPGA&μCLinux 的方式。μCLinux 适用于不含 MMU 微处理器的运行环境，如 Nios Ⅱ软核。它具有稳定、强大的网络功能，较低的资源占用率。用它实现 SNMPv3 代理将会事半功倍。

（2）使用 ARM&Linux 的方式。ARM9 以上的处理器都有 MMU，支持完整的 Linux。在 Linux 的环境下使用日益成熟的 ARM 处理器实现 SNMPv3 代理也会十分的方便。

参 考 文 献

1 王兆峰，彭晓燕. 三层以太网交换机 SNMP 代理软件的设计与实现[J]. 微计算机信息，2004，20(5)：60～61

2 Stalling W. SNMP 网络管理[M]. 胡成松，江凯译. 北京：中国电力出版社，2003

3 罗雪松，罗蕾，许子辛. 嵌入式 SNMP Agent 的设计与实现[J]. 计算机应用研究，2004，21(10)：220～221

4 白璘，孙肖子，白玉霞. 基于 NIOS Ⅱ的嵌入式 Web 服务器[J]. 现代电子技术，2005，28(12)：30～31

5 周博，邱卫东. 挑战 SOC-基于 Nios 的 SOPC 设计与实践[M]. 北京：清华大学出版社，2004

6 刘红，白栋，孔令山，等. 嵌入式 SNMP 软件的设计与实现[J]. 计算机工程与应用，2001，37(21)：52～54

7 顾华玺，邱智亮，刘增基. 嵌入式代理在以太网交换机中的实现[J]. 电信科学，2002，18(6)：24～28

（上接第 138 页）

2 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proceedings of 1993 ACM-SIGMOD International Conference on Management of Data, Washington, D. C., May 1993. 207～216

3 Agrawal R, Srikant R. Fast algorithms for mining association rules in large databases. In: Bocca JB, Jarke M, Zaniolo C, eds. Proc. of the 20th Int'l Conf. on Very Large Data Bases. Santiago: Morgan Kaufmann, 1994. 478～499

4 Han J, Pei J, Yin Y. Mining Frequent Patterns without Candidate Generation. In: Proceedings of the 2000 ACM-SIGMOD International Conference on Management of Data, Dallas, TX: ACM Press, 2000. 1～12

5 Pei Jian, Han Jiawei, Mao Runying. Closet: An efficient algorithm for mining frequent closed itemsets. In: SIGMOD International Workshop on Data Mining and Knowledge Discovery, May 2000

6 Pei Jian, Han Jiawei, Wang Jianyong. Closet+: Searching for the best strategies for mining frequent closed itemsets. In: SIGKDD'03, August 2003

7 Lucchese C, Orlando S, Perego R. Fast and Memory Efficient Mining of Frequent Closed Itemsets: [Technical Report CS-2004-9]. Nov 2004

8 杨红菊，梁吉业. 一种挖掘频繁项集和频繁闭包项集的算法. 计算机工程与应用，2004，40(13)：176～178

9 Mohammad El-Hajj, Zaiane O R, COFI-tree Mining: A New Approach to Pattern Growth with Reduced Candidacy Generation. In: Workshop on Frequent Itemset Mining Implementations (FIMI'03) in conjunction with IEEE-ICDM 2003, Melbourne, Florida, USA, November 2003

10 范明，李川. 在 FP-树中挖掘频繁模式而不生成条件 FP-树. 计算机研究与发展，2003，40(8)：1216～1222

11 Gopalan R, Sucahyo YG. High Performance Frequent Patterns Extraction sing Compressed FP-tree. In: Proceedings of the SIAM International Workshop on High Performance and Distributed Mining, Orlando, USA, April 2004