文章编号: 1001-0920(2008)09-0994-05

挖掘事务间频繁闭项集的高效率算法

董 杰, 韩 敏
(大连理工大学 电子与信息工程学院, 大连 116023)

摘 要: 事务间频繁项集将传统的单维事务内关联规则扩展到多维跨事务关联规则,但事务间频繁项集的数量随滑动时间窗口的增大而迅速增加. 利用频繁闭项集的特点,提出事务间频繁闭项集的概念及其挖掘算法(FCITA). 该算法采用分割和条件数据库技术,避免生成庞大的扩展数据库;利用扩展二进制形式压缩事务,从而提高支持度的计算效率. 此外,动态排序和哈希表极大地减少了频繁闭项集的测试次数. 仿真比较表明,FCITA 算法具有较高的挖掘效率.

关键词: 数据挖掘; 关联规则; 事务间频繁闭项集
中图分类号: TP311 **文献标识码**: A

Efficient algorithm of mining frequent closed inter-transaction itemsets

DONG Jie, HAN Min
(Department of Electronic and Information Engineering, Dalian University of Technology, Dalian 116023, China. Correspondent: HAN Min, E-mail: minhan@dlut.edu.cn)

Abstract: Frequent inter-transaction itemsets extend the scope of mining association rules from traditional single-dimensional, intra-transaction associations to multidimensional, inter-transaction associations. With the increasing of the MaxSpan, the amount of the frequent inter-transaction itemsets, however, becomes very large. Considering the characterictics of the frequent closed itemsets, we introduce the notion of frequent closed inter-transaction itemsets and develop an efficient algorithm, frequent closed inter-transaction itemsets algorithm (FCITA). FCITA adopts division and condition database to avoid the generation of huge extended database, and uses the extended Bit Table to compress the transaction and improve the counting efficiency of the support. By using dynamic ordering and hash table, the testing times of the candidate closed inter-transacation itemsets is decreased. Simulations results show that FCITA is fast and efficient.

Key words: Data mining; Association rules; Frequent closed inter-transacation itemsets

1 引 言

频繁项集挖掘是数据挖掘中的一项重要任务. 在过去的几十年里,尽管已有许多高效率的频繁项集挖掘算法[1,2],但主要集中于事务内频繁项集的获取,这类算法揭示了多个事物在相同时间点的相互关联性,但却无法获取其在不同时刻的关联性.

Lu 等[3]提出事务间频繁项集的数据挖掘问题,并给出了事务间频繁项集挖掘的有效方法: EH-Apriori 算法和 FITI 算法[3]. Feng 等[4]提出了基于模版的多维事务间频繁项集挖掘算法,并将事务间频繁项集挖掘应用于气象研究[5].

上述算法的结果都是针对完全频繁项集,但相同最小支持度下事务间频繁项集的数量要远大于事务内频繁项集,并且随着时间间隔的增大,事务间频繁项集的数量增长迅速. Zaki 等[6]提出了频繁闭项集的概念,它能唯一地确定频繁项集,并且极大地减少结果集的冗余度. 因此,研究如何快速地挖掘事务间频繁闭项集具有重要的意义. 虽然可将已有的事务内频繁闭项集算法[7~9]进行扩展,得到事务间频繁闭项集,但此类扩展的事务间频繁闭项集的挖掘算法效率低下,占用资源较多.

本文根据事务间频繁闭项集的性质和特点,提

收稿日期: 2007-06-14; **修回日期**: 2007-10-23.
基金项目: 国家自然科学基金项目(60674073); 国家科技支撑计划项目(2006BAB14B05); 国家 973 计划项目(2006CB403405).
作者简介: 董杰(1981—),男,山东青岛人,博士生,从事数据挖掘、智能算法的研究; 韩敏(1959—),女,吉林延吉人,教授,博士生导师,从事神经网络、混沌序列分析等研究.

出一种高效率的挖掘事务间频繁闭项集算法——FCITA.

2　主要概念与定理

2.1　主要概念和定义

令 $\Sigma = \{e_1, e_2, \cdots, e_n\}$,其中 $e_i (i = 1, 2, \cdots, n)$ 是不同的项. D 是一个属性值,$\mathrm{Dom}(D)$ 是 D 的值域. 事务 t 的形式是 (d, E),其中:$d \in \mathrm{Dom}(D)$,$E \subseteq \Sigma$,事务数据库 T 是事务 t 的集合. $I = \{i_1, i_2, \cdots, i_n\} \mid i_k \in \Sigma$ 称为项集. 如果 $i_k \in t$, $\forall i_k \in I$,则称事务 t 包含项集 I. 项集 I 的支持度是数据库 T 中包含 I 的事务的个数,记为 $\sup(I) \mid T$. 所有包含项集 I 的事务组成的集合记为 $\sigma(I)$. 如果项集 I 的支持度大于用户指定的最小支持度 MinSup,则称 I 为频繁项集.

滑动窗口 W 是沿属性 D 的长度为 w 的连续间隔,W 从 d_0 开始,每个间隔 d_j 包含的事务记为 $W[j]$,其中 $j = d_j - d_0$. 用户指定的最大时间间隔记为 MaxSpan.

$M = \{e_i(j) \mid e_i \in W[j], 1 \leq i \leq n, 0 \leq j \leq w - 1\}$ 称为扩展事务. 所有可能的扩展事务的集合称为扩展事务数据库,记为 T'. 扩展事务数据库中项的集合记为

$$\Sigma' = \{e_1(0), \cdots, e_1(w-1), e_2(0), \cdots, e_2(w-1), \cdots, e_n(0), \cdots, e_n(w-1)\}.$$

定义 1　满足下列条件的项集 p 称为事务间频繁闭项集:

1) $p \subseteq \Sigma'$;

2) $\exists e_i(0) \in p$, $1 \leq i \leq n$;

3) $\sup(p) \mid T' \geq \mathrm{MinSup}$;

4) $\sup(q) < \sup(p)$, $\forall q \subseteq \Sigma', q \supset p$.

将 Σ 划分为两部分:$E_1 = (e_1, e_2, \cdots, e_k)$,$E_2 = (e_{k+1}, e_{k+2}, \cdots, e_n)$,则数据库 T 按此划分可分为 T_1 和 T_2. $C_1 = (c_{11}, c_{12}, \cdots, c_{1i}, \cdots, c_{1m})$ 是数据库 T_1 的频繁闭项集,c_{1i} 的条件数据库 $T(c_{1i})$ 是数据库 T 中所有包含 c_{1i} 的事务在 E_2 上投影的集合. 条件数据库 $T(c_{1i})$ 的频繁闭项集记为 $C_{1i} = (c_{1i1}, c_{1i2}, \cdots, c_{1ik}, \cdots, c_{1iz})$. 由条件数据库的定义可知

$$\sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{1i} + c_{ik}) \mid T.$$

2.2　相关定理及证明

定理 1　对于数据库 T 上任意一个频繁闭项集 c_i,按 E_1 和 E_2 划分为 c_{1i} 和 c_{1ik},则 c_{1i} 是数据库 T_1 的频繁闭项集,c_{1ik} 是 c_{1i} 条件数据库 $T(c_{1i})$ 的频繁闭项集.

证明　1) 由于 $c_{1i} \subseteq c_i$,$c_{1ik} \subseteq c_i$,由支持度定义可知

$$\sup(c_{1i}) \mid T_1 = \sup(c_{1i}) \mid T \geq \sup(c_i) \mid T,$$

$$\sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{1i} + c_{1ik}) \mid T = \sup(c_i) \mid T.$$

因为 $\sup(c_i) \mid T \geq \mathrm{MinSup}$,所以 c_{1i} 和 c_{ik} 都是频繁项集.

2) 假设 c_{1i} 不是 T_1 的频繁闭项集,则 $\exists c_j \in C_1, c_{1i} \subset c_{1j}$,有 $\sup(c_{1j}) \mid T_1 = \sup(c_{1i}) \mid T_1$. 因为

$$\sup(c_i) \mid T = \sup(c_{1i} + c_{1ik}) \mid T = \sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{1ik}) \mid T(c_{1j}) = \sup(c_{1ik} + c_{1j}) \mid T,$$

且 $c_i \subset c_{1ik} + c_{1j}$,与 c_i 是频繁闭项集矛盾,所以不存在这样的 c_{1j}.

3) 假设 c_{1ik} 不是 $T(c_{1i})$ 的频繁闭项集,则 $\exists c_{1il} \in C_{1i}, c_{1ik} \subset c_{1il}$,有 $\sup(c_{1il}) \mid T(c_{1i}) = \sup(c_{1ik}) \mid T(c_{1i})$. 因为

$$\sup(c_i) \mid T = \sup(c_{1i} + c_{1ik}) \mid T = \sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{1il}) \mid T(c_{1i}) = \sup(c_{1il} + c_{1i}) \mid T,$$

且 $c_i \subset c_{1il} + c_{1i}$,与 c_i 是频繁闭项集矛盾,所以不存在这样的 c_{1il}.

由 1) 可知 c_{1i} 和 c_{1ik} 是频繁项集,2) 和 3) 则证明了其封闭性,因此定理得证. □

定理 2　c_{1i} 是数据库 T_1 的频繁闭项集,c_{1ik} 是条件数据库 $T(c_{1i})$ 上的频繁闭项集,如果不存在满足以下条件的 c_{1j}:

1) $c_{1j} \in C_1, c_{1i} \subset c_{1j}$;

2) c_{1ik} 也是条件数据库 $T(c_{1j})$ 上的频繁闭项集;

3) $\sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{1ik}) \mid T(c_{1j})$.

则 $c_{1i} + c_{1ik}$ 是数据库 T 的一个频繁闭项集.

证明　1) 因为 $c_{1ik} \in C_{1i}$,$\sup(c_{1ik}) \mid T(c_{1i}) \geq \mathrm{MinSup}$,$T(c_{1i})$ 是 T 的一部分,则 $\sup(c_{1ik}) \mid T \geq \mathrm{MinSup}$,且 $\sigma(c_{1ik}) \subseteq \sigma(c_{1i})$,所以 $\sigma(c_{1ik}) = \sigma(c_{1i} + c_{1ik})$. 可知 $\sup(c_{1i} + c_{1ik}) \mid T \geq \mathrm{MinSup}$,因此 $c_{1i} + c_{ik}$ 是频繁的.

2) 如果 $c_{1i} + c_{1ik}$ 不是封闭的,假设 $\exists I \supset c_{1i} + c_{ik}$,其中 I 是数据库 T 的频繁闭项集,且 $\sup(I) \mid T = \sup(c_{1i} + c_{1ik}) \mid T$,则 I 共有 3 种情况:

① I 表示为 $c_{1i} + c_{1l} + c_{ik}$,其中 $c_{1l} \subset E_1$.

因为 I 是数据库 T 的频繁闭项集,由定理 1 可知 $c_{1i} + c_{1l} \in C_1, c_{1ik} \in C_{1i+1l}$,有

$$\sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{1ik} + c_{1i}) \mid T = \sup(I) \mid T = \sup(c_{1i} + c_{1l} + c_{1ik}) \mid T = \sup(c_{1ik}) \mid T(c_{1i} + c_{1l}).$$

$c_{1i} + c_{1l}$ 符合定理中的 3 个条件,此种情况在定理中被排除,因此不存在这样的 I.

② I 表示为 $c_{1i} + c_{1ik} + c_{1il}$,其中 $c_{1il} \subset E_2$.

由定理 1 知 $c_{1ik} + c_{1il} \in C_{1i}$,有

$$\sup(I) \mid T = \sup(c_{1i} + c_{1ik} + c_{1il}) \mid T = \sup(c_{1ik} + c_{1il}) \mid T(c_{1i}).$$

因为 $c_{1ik} + c_{1il}$ 是 $T(c_{1i})$ 的频繁闭项集,则

$$\sup(c_{1ik} + c_{1il}) \mid T(c_{1i}) < \sup(c_{1ik}) \mid T(c_{1i}),$$

所以

$$\sup(I) \mid T < \sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{1ik} + c_{1i}) \mid T.$$

与假设矛盾,因此不存在这样的 I.

③ I 表示为 $c_{1i} + c_{1l} + c_{1ik} + c_{1il}$,其中:$c_{1l} \subset E_1$, $c_{1il} \subset E_2$.

由定理 1 知 $c_{1i} + c_{1l} \in C_1$, $c_{1ik} + c_{1il} \in C_{1i+1l}$,则有

$$\sup(I) \mid T = \sup(c_{1i} + c_{1l} + c_{1ik} + c_{1il}) \mid T = \sup(c_{1ik} + c_{1il}) \mid T(c_{1i} + c_{1l}) \leq \sup(c_{1ik}) \mid T(c_{1i} + c_{1l}) \leq \sup(c_{1ik}) \mid T(c_{1i}).$$

由 ①的证明过程知,不存在 c_{1l} 使得

$$\sup(c_{1ik}) \mid T(c_{1i} + c_{1l}) = \sup(c_{1ik}) \mid T(c_{1i}),$$

则

$$\sup(c_{1ik}) \mid T(c_{1i} + c_{1l}) < \sup(c_{1ik}) \mid T(c_{1i}),$$

因此

$$\sup(I) \mid T < \sup(c_{1ik}) \mid T(c_{1i}) = \sup(c_{ik} + c_{1i}) \mid T.$$

与假设矛盾,因此不存在这样的 I.

综上,$c_{1i} + c_{1ik}$ 是数据库 T 的一个频繁闭项集.

由 1) 可知 $c_{1i} + c_{1ik}$ 是频繁项集,2) 和 3) 则证明了其封闭性,因此定理得证. □

推论 1 符合定理 2 的频繁闭项集的集合即为数据库 T 的频繁闭项集的集合.

定理 3 条件数据库 $T(c_{1i})$ 的频繁闭项集 $C_{1i} = (c_{1i1}, c_{1i2}, \cdots, c_{1ik}, \cdots, c_{1iz})$ 均为数据库 T_2 的频繁闭项集.

证明 假设 c_{1ik} 不是数据库 T_2 的频繁闭项集,则存在 $c_{1il} \subset E_2$, $c_{1ik} + c_{1il}$ 是数据库 T_2 的频繁闭项集,且 $\sup(c_{1ik} + c_{1il}) \mid T_2 = \sup(c_{1ik}) \mid T_2$. 因为 C_{1ik} 是数据 $T(c_{1i})$ 的频繁闭项集,则

$$\sup(c_{1ik} + c_{1il}) \mid T(c_{1i}) < \sup(c_{1ik}) \mid T(c_{1i}).$$

由支持度定义可知

$$\sup(c_{1ik} + c_{1il}) \mid T_2 - T(c_{1i}) < \sup(c_{1ik}) \mid T_2 - T(c_{1i}),$$

所以 $\sup(c_{1ik} + c_{1il}) \mid T_2 < \sup(c_{1ik}) \mid T_2$. 与假设矛盾,因此不存在这样的 c_{1il}. □

推论 2 定理 1 ~ 定理 3 可推广到划分为 n 的数据库中.

3 FCITA 算法

3.1 算法基本思想

将扩展事务数据库 T 按时间窗口划分为 w 部分:$T_1', \cdots, T_k', \cdots, T_w'$,则每一部分 T_k' 可看作是事务数据库 T 沿属性 D 移动 k 之后的变换;同样,数据库 T 的频繁闭项集沿属性 D 移动后,得到的项集便是数据库 T_k' 的频繁闭项集. 根据定理 3,可以快速生成条件数据库的频繁闭项集. 根据定理 2,扩展事物数据库 T 的频繁闭项集可由各部分条件数据库的频繁闭项集组合得到.

FCITA 算法共分为以下 3 步:

1) 挖掘事务内频繁闭项集;

2) 生成事务间频繁闭项候选集并计算支持度;

3) 检查事务间频繁项集的封闭性.

算法的第 1 步可采用现有的各种高效率事务内频繁闭项集挖掘算法,在 FCITA 算法中采用了 Charm 算法[6].

3.2 事务间频繁闭项候选集生成与支持度计算

每个事务间频繁闭项候选集均由 w 部分组成,其中每一部分均为数据库 T 的一个频繁闭项集或空集. 当前生成的候选集由候选集栈存储,候选集栈共有 w 层,每一层存储一个指向频繁闭项集链表的指针,代表一段时刻的项集. 其中:栈底表示 d_0 时刻的项集,栈顶表示 d_{w-1} 时刻的项集. 候选集栈将每一层的指针初始化为指向频繁闭项集链表的表头. 生成新的候选集时,对栈顶进行更新操作,将栈顶指针指向链表的下一个元素. 如果到达链表尾部,则对栈顶的下一层进行更新操作,并依次递归.

候选集支持度计算在条件数据库上完成,d_j 层的条件数据库在 d_{j-1} 层条件数据库的基础上生成. 首先取出候选集栈 d_j 层的频繁闭项集 c_i,得到所有支持 c_i 的事务集合 $\sigma(c_i)$,将 $\sigma(c_i)$ 中的所有事务沿属性 D 移动 $j = d_j - d_0$;然后与 d_{j-1} 层的条件数据库取交集,即为 d_j 层的条件数据库,自第 d_0 层向第 d_{w-1} 层依次生成条件数据库,第 d_{w-1} 层条件数据库的事务数量就是候选集的支持度.

FCITA 算法采用支持度栈进行快速支持度计算. 支持度栈共 w 层,每一层存储代表条件数据库的 Bit Table[10]. Bit Table 数据结构是利用二进制对数据库事务或集合进行压缩. 如图 1 所示,含有 4 条事务的数据库可压缩为集合{10,7,14,8,7}. 当计算项集{2,3,5} 的支持度时,直接将对应的 Bit Table 元素{7,14,7} 进行按位“与”操作,便可得到支持度.

为了提高算法效率,节省内存空间,算法对于所有支持 c_i 的事务集合 $\sigma(c_i)$,仅存储事务属性 D,

TID	Item 1	Item 2	Item 3	Item 4	Item 5
1	1	0	1	1	0
2	0	1	1	0	1
3	1	1	1	0	1
4	0	1	0	0	1
BitTable	10	7	14	8	7

item 2		item 3		item 5		support
0		1		0		0
1	&	1	&	1	→	1
1		1		1		1
1		0		1		0
7		14		7		6

图 1　BitTable 数据结构及其快速支持度算法

并对支持事务内频繁闭项集的事务属性 D 进行二进制压缩存储,置入 Bit Table 数据结构. 对支持度栈的操作与候选集栈的操作同步进行,并且仅对候选集栈当前处理的层进行操作. FCITA 对传统 Bit Table 的操作进行扩展,采用二进制移位操作进行事务沿属性 D 的移动.

3.3　事务间频繁项集封闭性检查

当事务间频繁闭项候选集的支持度大于用户指定的最小支持度 MinSup 时,根据定理 2,只需检查是否存在前半部分 c_{1i} 的超集 c_{1l},并且后半部分 c_{1ik} 在 c_{1i} 和 c_{1l} 两个条件数据库上的支持度相同. 如果不存在满足此条件的 c_{1l},则候选集就是事务间频繁闭项集.

在生成频繁闭项集链表前,先对事务内频繁闭项集按项集长度进行降序排序,可保证生成长度较长的事务间频繁闭项集. 对排序后的事务内频繁闭项集重新编号,每个事务内频繁闭项集赋予一个 ID,并生成指向所有超集的指针. 为减少计算量,避免将候选集与所有先前得到的事务间频繁闭项集进行比较,FCITA 采用哈希表存储事务间频繁闭项集,哈希方程采用所有包含项集的事务属性 D 的和. 如果候选集与其超集的支持度相等,则必被相同的事务所包含,即它们的事务属性 D 的和一定相等. 采用哈希方程极大地减少了每个哈希元中项集的数量. 如果相应的哈希元不存在候选集的超集,则候选集就是事务间频繁闭项候选集.

4　性能比较

关于事务间频繁闭项集的挖掘,目前国内外相关研究较少. 对于此类问题,可采用如下算法予以解决:1) 生成扩展数据库;2) 在扩展数据库上,利用现有的频繁闭项集挖掘算法求出频繁闭项集;3) 删除其中不符合事务间频繁闭项集定义的项集,即将事务内频繁闭项集挖掘算法扩展到事务间频繁闭项集的挖掘. 此类方法称为扩展的事务间频繁闭项集挖掘算法. 将此类扩展方法与 FCITA 进行对比分析,可得出以下结论:

1) FCITA 直接在原数据库进行计算,而此类扩展方法需要先生成扩展数据库. 当数据库较大或时间间隔较长时,需要较长的初始化时间.

2) 此类扩展方法生成的扩展数据库容量约为原数据库的 MaxSpan 倍. 如果原数据库较大或指定的时间间隔较长,会使扩展数据库变得非常庞大,导致事务内频繁闭项集挖掘算法效率降低.

上述分析表明:FCITA 算法在挖掘事务间频繁闭项集的效率上有较大的优势. 为了验证算法的效率,用 VC + + 6.0 实现了 FCITA 算法,并对 Charm[6] 算法作了修改,使之能挖掘事务间频繁闭项集,称为 ExtendCharm. 试验环境为 3.0 GHz Pentium IV CPU,512 M 内存,Windows 2000 操作系统. 选取两组常用测试数据集:1) 机器学习数据资源库中的实际数据集 connect,具有 67 557 个事务,130 个项,事务平均项数为 43,是稠密型数据集;2) T40I10D100K 是人造数据集,具有 100 K 个事务,1 K 个项,事务平均项数为 40,是稀疏型数据集.

图 2 给出了在稠密型数据集 connect 上 FCITA 和 ExtendCharm 在相同支持度下随时间间隔变化的比较情况. 图 3 是相同时间间隔下支持度变化的比较情况. 图 4 和图 5 给出了在稀疏型数据集 T40I10D100K 上的比较情况. 从比较情况可以看出,FCITA 在两类数据集上均有较高的执行效率.

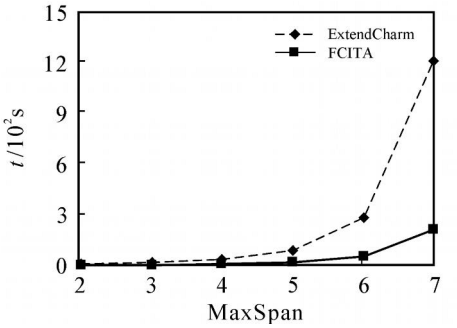

图 2　支持度 99 %时在 connect 数据集上的比较

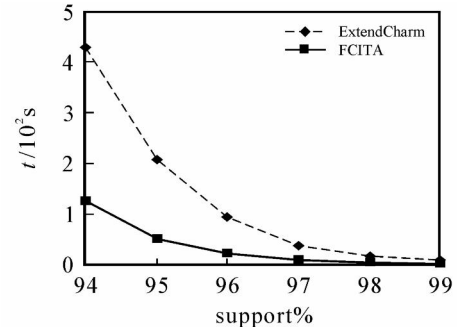

图 3　时间间隔为 2 时在 connect 数据集上的比较

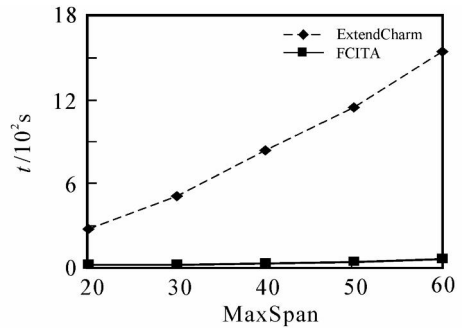

图 4 支持度 10 %时在 T40I10D100K数据集上的比较

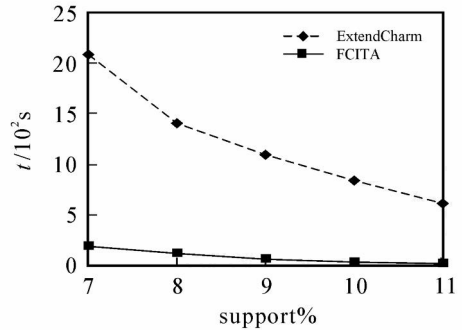

图 5 时间间隔为 40 时在 T40I10D100K数据集上的比较

5 结 论

本文根据事务间频繁闭项集的性质和特点,提出一种高效率的挖掘事务间频繁闭项集算法 FCITA.算法采用分割和条件数据库方法挖掘事务间频繁闭项集,避免生成庞大的扩展数据库,并采用栈和 Bit Table 数据结构进行快速支持度计算.实验结果表明,FCITA 对于稀疏型和稠密型数据集均有较好的执行效率.因此,本文提出的 FCITA 算法能高效地进行事务间频繁闭项集的挖掘,解决了事务间频繁闭项集挖掘的效率问题.

参考文献(References)

[1] Han J W, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]. Proc of the 2000 ACM SIGMOD Int Conf on Management of Data. ACM Press, 2000: 1-12.

[2] Holt J D, Chung S M. Mining association rules using inverted hashing and pruning [J]. Information Processing Letters, 2002, 83(4): 211-220.

[3] Lu H J, Han J W, Feng L. Stock movement prediction and *n*-dimensional inter-transaction association rules [C]. Proc of the SIGMOD '98 Workshop on Research Issues on Data Mining and Knowledge Discovery. Seattle: ACM Press, 1998: 121-127.

[4] Feng L, Lu H J, Yu J X, et al. Mining inter-transaction association rules with templates[C]. Proc of ACM CIKM Int Conf Information and Knowledge Management. Kansas City: ACM Press, 1999: 225-233.

[5] Feng L, Dillon T, Liu J. Inter-transactional association rules for multi-dimensional contexts for prediction and their application to studying meteorological data [J]. Data and Knowledge Engineering, 2001, 37(1): 85-115.

[6] Zaki M J, Hsiao C J. Efficient algorithms for mining closed itemsets and their lattice structure [J]. IEEE Trans on Knowledge and Data Engineering, 2005, 17(4): 462-478.

[7] Lucchese C, Orlando S, Perego R. Fast and memory efficient mining of frequent closed itemsets [J]. IEEE Trans on Knowledge and Data Engineering, 2006, 18(1): 21-36.

[8] 刘君强, 孙晓莹, 庄越挺, 等. 挖掘闭合模式的高性能算法[J]. 软件学报, 2004, 15(1): 94-102.
(Liu J Q, Sun X Y, Zhuang Y T, et al. Mining frequent closed patterns by adaptive pruning[J]. J of Software, 2004, 15(1): 94-102.)

[9] 陈耿, 朱玉全, 杨鹤标, 等. 关联规则挖掘中若干关键技术的研究[J]. 计算机研究与发展, 2005, 42(10): 1785-1789.
(Chen G, Zhu Y Q, Yang H B, et al. Study of some key techniques in mining association rule [J]. J of Computer Research and Development, 2005, 42(10): 1785-1789.)

[10] Dong J, Han M. BitTableFI: An efficient mining frequent itemsets algorithm [J]. Knowledge-Based Systems, 2007, 20(4): 329-335.