维普资讯 http://www.cqvip.com

基于敏捷分桶的频繁项目集生成新算法

周启海[1] 陈勇明[2]

(西南财经大学经济信息工程学院[1] 西南财经大学统计学院[2] 成都 610074)

摘 要 指出用于数据挖掘的频繁项目集生成的常规 Hash 算法存在两个主要缺点:1)难挑选合适的 Hash 函数,2)易导致 Hash 冲突。为了克服了这些缺点,提出了一种能动态适应频繁项目集生成实际需要的敏捷分桶新算法,该算法对任何项目集均有按需反应能力,且无需寻找任何 Hash 函数,更不会导致任何 Hash 冲突。同时给出了进一步改进和提高新算法效率的研究方向。

关键词 数据挖掘,频繁项目集,Hash 函数,Hash 冲突,敏捷分桶

A New Algorithm Based on Agile Separating into Buckets for Finding Frequent Item Sets

ZHOU Qi-Hai[1] CHEN Yong-Ming[2]

(School of Economic Information Engineering[1], School of Statistics[2], Southwestern University of Finance and Economics, Chengdu 610074)

Abstract In this paper, tow main shortcomings of a conventional hash algorithm used for finding frequent item sets in data mining are pointed out: 1) it is difficult to choose a suitable hash function; 2) it is easy to lead to a hash conflict. In order to overcome the above shortcomings, a new algorithm which can separate into buckets and can suit dynamically the practical need of finding frequent item sets is advanced, which has a reaction ability to any item sets according to need and does not look for any hash function and lead to any hash conflict. At the same time, the directions for further improving and razing the efficiency of this new algorithm are given.

Keywords Data mining, Frequent item sets, Hash function, Hash conflict, Separate agilely into buckets

1 现行频繁项目集生成算法概述

生成频繁项目集,是数据挖掘的基本前提条件。频繁项目集的生成算法是数据挖掘的重要基础算法。故长期来,寻求可进一步提高频繁项目集生成效率的新算法,一直是数据挖掘的研究热点之一。

20 世纪 90 年代,人们发现 2-频繁项目集的生成是频繁项目集寻找处理的重心与瓶颈之一。因为实验表明"频繁项目集的寻找处理工作重心,主要集中于 2-频繁项目集的计算"。1995 年,Park 等顺应此重心特性要求,提出了一个基于散列(Hash)技术的生成频繁项目集的算法[1,2]。这种基于 Hash 函数的 2-频繁项目集生成的 Hash 算法,其主要特点是:它把所扫描到的各项目对,逐一分别投放入对应的确定但不同的 Hash 桶中,而每对项目必须、只能且最多可存放在一个特定的 Hash 桶中。这样,生成 2-频繁项目集时就可只对所得各桶中的项目对进行测试,从而减少了候选项目集生成的代价。与经典的 Apriori 算法[3]需要对事务数据库进行多次扫描相比,该 Hash 算法显然只需要扫描一次事务数据库即可,故它在克服 Apriori 算法需要较大 I/O 负载的处理开销瓶颈方面有所改进。然而,这种 Hash 算法也存在一些缺点,尤其是存在下列两个主要问题:

1)难挑选合适的 Hash 函数问题。Hash 函数的一般形式为 $H(x,y)=F(x,y) \bmod p$,而如何根据数据挖掘需要来敏捷地确定比较适宜的 Hash 函数的 $F(x,y)$ 函数形式和模 p 都比较困难。

2)易导致 Hash 冲突问题(或称 Hash 碰撞)[4]。由于不同的候选项目集可能通过 Hash 函数投射到同一桶中,故此时会造成"桶地址与桶内容不一致"的非唯一性冲突。这种传统 Hash 冲突的存在,需要对每个桶的内容(即候选项目集)进行一致性检测与处理,故会不可避免地限制和降低了算法效率。

文[4]提出了一种生成频繁项目集的二维 Hash 算法。它采用两个 Hash 函数,并用一个二维向量作为其 Hash 桶的二维桶地址。它虽可使因用一维桶地址而产生 Hash 冲突的可能性较前大大减少,但并不会也没有完全避免 Hash 冲突。为此,文[5]进一步对文[4]做了改进,特意设计了两个新的 Hash 函数,并据此得出可避免 Hash 冲突的结论[5]。但应指出:文[5]同样也没有从理论上完全解决 Hash 冲突问题。事实上,文[5]所给出的下列两个 Hash 函数

$$h_1(\{i_x,i_y\})=[S(m-1)+n-1-m(m+1)/2] \bmod p_1$$

$$h_2(\{i_x,i_y\})=[S(m-1)+n-1-m(m+1)/2] \bmod p_2$$

其中:S 为事务数据库中事务总数;m 和 n 分别为项目 i_x 和 i_y 所对应的自然数,且 $m<n$;$p_1 \neq p_2$,$p_1 \times p_2$ 大于事务数据库两项组合的项目对个数,通常 p_1 与 p_2 互质;文[5]设 $F(m,n)=S(m-1)+n-1-m(m+1)/2$,并证明了:若 $F(m_1,n_1)=F(m_2,n_2)$,则必有 $m_1=m_2$,$n_1=n_2$。但仅据此还不能说明 *Hash* 冲突就已经解决,因为 Hash 桶地址是由 $F(m,n)$ 取模 p_1、p_2 所得余数来决定,而不同的数对 (m,n) 所决定的 $F(m,n)$ 在对 p_1、p_2 取模后,是完全可能得到相同的余数,从而将有可能使不同项目集被存放在同一 Hash 桶中。

周启海 教授,博(硕)士生导师,主要研究方向:计算几何、算法研究与应用、财经计算、同构化信息处理等;**陈勇明** 博士研究生,主要研究方向:统计分析、应用数学。

维普资讯 http://www.cqvip.com

为了提高频繁项目集生成效率并从根本上消除上述Hash算法的两个主要问题，本文创用了多维向量表征的Hash桶地址描述法(即不失一般性，本文提出的敏捷分桶算法所用n-频繁项目集L_n的Hash桶地址，必须且只需用n维向量来描述即可)，提出了一种能动态适应频繁项目集生成实际需要而逐个增加所需桶地址的敏捷分桶新算法。本算法不仅对任何项目集均有按需反应能力，而且既无需寻找任何Hash函数，更不会导致任何Hash冲突。

2 敏捷分桶算法基本思想

项目集通常是有限集，故可设其项目个数为m，从而项目集和数集$\{1,2,\cdots,m\}$可以建立一一对应关系。一般地，设$I=\{i_1,i_2,\cdots,i_m\}$是一个项目集合，事务数据库$D=\{t_1,t_2,\cdots,t_n\}$是由一系列具有唯一标识TID($\in\{1,2,\cdots,n\}$)的事务组成，而事务t_i都对应于项目集合I的一个子集I_i。

为了节省篇幅，本文仅以2-频繁项目集L_2生成算法为例，简述生成频繁项目集的敏捷分桶新算法。于是，描述2-频繁项目集的Hash桶地址，显然只需用2维向量(s,t)来描述即可，其中s,t分别为事务s与事务t的序号下标。据此，2-频繁项目集的敏捷分桶新算法的基本思想可简述如下：

第1步，初始化处理。记A为已有桶地址的桶地址集合，并置A为初始值空集。对于2-项目集$\{i_s,i_t\}$以二维向量(s,t)作为桶地址，以计数器n_{st}标记桶地址为(s,t)的桶中2-项目集$\{i_s,i_t\}$的个数。

第2步，事务数据库扫描处理。当事务数据库还有未处理事务时，顺序取出事务数据库D中的一个当前未处理事务t_s，并从未处理事务t_s的2-事务集中扫描出事务t_s与事务t_i，并由此得到2-项目集$\{i_s,i_t\}$。与此同时，检查桶地址集合A，即若当前桶地址$(s,t)\notin$A，则表明此时桶地址集合A中尚无此新桶地址(s,t)，故应将此新桶地址(s,t)添加到A中，并置此新桶的2-项目集$\{i_s,i_t\}$的个数计数器n_{st}为1；否则(即桶地址$(s,t)\in$A)，表明此时桶地址为(s,t)的该桶内又被新投入了一个相同的2-项目集$\{i_s,i_t\}$，故置此旧桶的2-项目集$\{i_s,i_t\}$的个数计数器n_{st}为$n_{st}+1$。

第3步，使事务数据库记录指针指向事务数据库中的下一未处理事务记录(不失一般性，可仍记之为t_s)，并转到第2步。

第4步，2-频繁项目集L_2生成处理。输入用户提供的2-频繁项目集判据(即最小支持度Minsupport，或者最小频繁项目集个数)；以判据Minsupport为尺度，对桶地址(s,t)所对应的2-项目集$\{i_s,i_t\}$的个数n_{st}进行2-频繁项目集判别式(即$n_{st}/N\geq$Minsupport，其中N为各2-项目集总个数)处理；收集并整理得2-频繁项目集Frequent_Itemsets；如果需要，还可从2-频繁项目集Frequent_Itemsets中筛选出最大2-频繁项目集Max_Frequent_Itemsets。

由此可见，频繁项目集生成的敏捷分桶算法实际上只需要扫描一次事务数据库D，就可随其扫描处理的完成而完整得出所求各桶(顺便指出：此时的桶个数恰为所需无歧义Hash桶的最小桶数，故本算法也可称频繁项目集生成的最小桶数算法)。每个桶内容(即各桶地址所对应的2-项目子集)总是唯一的，故已不必再一一检验每一个桶内是否出现Hash冲突。每个桶地址为(s,t)的桶中2-项目集i_s,i_t的个数自然也随即得出。

至此，已不难将2-频繁项目集L_2生成敏捷分桶算法同理推广为n-频繁项目集L_n生成敏捷分桶算法(略)。

3 算例实证分析与对比

为了更有说服力，本文特以文[2]所提供样本数据为例(如表1所示)，简要说明敏捷分桶算法的具体过程。

表1 事务数据库示例表

TID	Items	TID	Items
1	I1, I2, I5	6	I2, I3
2	I2, I4	7	I1, I3
3	I2, I3	8	I1, I2, I3, I5
4	I1, I2, I4	9	I1, I2, I3
5	I1, I3		

第1步，初始化处理。记A为已有桶地址的桶地址集合，并置A为初始值空集。对于2-项目集$\{i_s,i_t\}$，以二维向量(s,t)作为桶地址，以n_{st}标记桶地址为(s,t)的桶中2-项目集$\{i_s,i_t\}$的个数。

第2步～第3步，事务数据库扫描处理。当事务数据库还有未处理事务时，分别对表1所示各事件作顺次扫描处理，并可得其主要过程与相应结果如下：

对事件1，得到2-项目集$\{I1,I2\}$，$\{I1,I5\}$，$\{I2,I5\}$。$(1,2)\notin A$，此时让$A=\{(1,2)\}$，$n_{12}=1$；$(1,5)\notin A=\{(1,2)\}$，此时让$A=\{(1,2),(1,5)\}$，$n_{15}=1$；$(2,5)\notin A=\{(1,2),(1,5)\}$，此时让$A=\{(1,2),(1,5),(2,5)\}$，$n_{25}=1$。

对事件2，得到2-项目集$\{I2,I4\}$。$(2,4)\notin A=\{(1,2),(1,5),(2,5)\}$，此时让$A=\{(1,2),(1,5),(2,5),(2,4)\}$，$n_{24}=1$。

对事件3，得到2-项目集$\{I2,I3\}$。$(2,3)\notin A=\{(1,2),(1,5),(2,5),(2,4)\}$，此时让$A=\{(1,2),(1,5),(2,5),(2,4),(2,3)\}$，$n_{23}=1$。

对事件4，得到2-项目集$\{I1,I2\}$，$\{I1,I4\}$，$\{I2,I4\}$。$(1,2)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3)\}$，此时让$n_{12}=1+1=2$；$(1,4)\notin A=\{(1,2),(1,5),(2,5),(2,4),(2,3)\}$，此时让$A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4)\}$，$n_{14}=1$；$(2,4)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4)\}$，此时让$n_{24}=1+1=2$。

对事件5，得到2-项目集$\{I1,I3\}$。$(1,3)\notin A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4)\}$，此时让$A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，$n_{13}=1$。

对事件6，得到2-项目集$\{I2,I3\}$。$(2,3)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，此时让$n_{23}=1+1=2$。

对事件7，得到2-项目集$\{I1,I3\}$。$(1,3)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，此时让$n_{13}=1+1=2$。

对事件8，得到2-项目集$\{I1,I2\}$，$\{I1,I3\}$，$\{I1,I5\}$，$\{I2,I3\}$，$\{I2,I5\}$，$\{I3,I5\}$。$(1,2)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，此时让$n_{12}=2+1=3$；$(1,3)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，此时让$n_{13}=2+1=3$。$(1,5)A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，此时让$n_{15}=1+1=2$；$(2,3)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，此时让$n_{23}=2+1=3$；$(2,5)\in A=\{(1,2),(1,5),(2,5),(2,4),(2,3),(1,4),(1,3)\}$，此时让$n_{25}=1$

维普资讯 http://www.cqvip.com

+1=2；(3,5)∉A={(1,2)，(1,5)，(2,5)，(2,4)，(2,3)，(1,4)，(1,3)，(3,5)}，此时让n_{35}=1。

对事件9，得到2-项目集{ $I1$，$I2$}，{ $I1$，$I3$}，{ $I2$，$I3$}。(1，2)∈A={(1,2)，(1,5)，(2,5)，(2,4)，(2,3)，(1,4)，(1,3)，(3,5)}，此时让n_{12}=3+1=4；(1，2)A={(1，2)，(1,5)，(2,5)，(2,4)，(2,3)，(1,4)，(1,3)，(3,5)}，此时让n_{12}=3+1=4；(2，3)∈A={(1,2)，(1,5)，(2,5)，(2,4)，(2,3)，(1,4)，(1,3)，(3,5)}，此时让n_{23}=3+1=4。

至此，通过敏捷分桶算法处理已可得到如表2所示的桶分配及其内容示例表。

表2 2-项目集敏捷分桶的桶分配及其内容示例表

桶地址	(1,2)	(1,5)	(2,5)	(2,4)	(2,3)	(1,4)	(1,3)	(3,5)
桶计数	4	2	2	2	4	1	4	1
桶内容	{ I1，I2}	{ I1，I5}	{ I2，I5}	{ I2，I4}	{ I2，I3}	{ I1，I4}	{ I1，I3}	{ I3，I5}
	{ I1，I2}	{ I1，I5}	{ I2，I5}	{ I2，I4}	{ I2，I3}		{ I1，I3}	
	{ I1，I2}				{ I2，I3}		{ I1，I3}	
	{ I1，I2}				{ I2，I3}		{ I1，I3}	

第4步，2-频繁项目集L_2生成处理。显然，只要用户提供2-频繁项目集判据的最小支持度Minsupport值，就可据此求得相应的2-频繁项目集L_2。例如，若取最小支持度Minsupport为0.2(即4/(4+2+2+2+4+1+ 4+1)之值)，就可得所求2-频繁项目集L_2为{{ $I1$,$I2$}，{ $I2$,$I3$}，{ $I1$，$I3$}}。

应当指出：表2说明，敏捷分桶算法所得桶分配及其内容，地址含义明确，数据简明无歧(义)，处理敏捷高效。而与此形成鲜明对比是：尽管同样针对表1所示事务数据库的同一数据源，但文[2]通过Hash函数$(10x+y)$ mod 7所得如表3所示的桶分配及其内容，却“地址含义不明，数据可能有歧，处理繁琐低效”(不难看出，表3中，桶地址为0的桶发生了Hash冲突，因为此桶内容有两个不同2-项目集{$I1$,$I4$}和{$I3$,$I5$}，而采用敏捷分桶的算法，可有效地将这两个不同的项目集各放在两个不同桶中)。

表3 2-项目集Hash函数的桶分配示例

桶地址	0	1	2	3	4	5	6
桶计数	2	2	4	2	2	4	4
桶内容	{ I1，I4}	{ I1，I5}	{ I2，I3}	{ I2，I4}	{ I2，I5}	{ I1，I2}	{ I1，I3}
	{ I3，I5}	{ I1，I5}	{ I2，I3}	{ I2，I4}	{ I2，I5}	{ I1，I2}	{ I1，I3}
			{ I2，I3}			{ I1，I2}	{ I1，I3}
			{ I2，I3}			{ I1，I2}	{ I1，I3}

结论 本文提出的生成频繁项目集的敏捷分桶算法，可有效克服基于Hash函数的频繁项目集生成传统算法的上述两个主要缺点。同时，本敏捷分桶算法还可进一步予以改进。例如(我们拟另文阐述)，1)可利用其桶地址与桶内容的一一映射关系，基于所设计的用其桶地址间接而唯一地表征桶内容的有效描述方法，就可通过只对桶地址进行相应等价处理来直接找出所求频繁项目集，从而可再次提高本算法效率；2)对n-频繁项目集，还可利用适当的树结构(譬如n叉树)来描述和存放其各桶地址，就可采用基于树结构的桶地址快速查找技术，来更加提高本算法效率。

参 考 文 献

1 Park J. Efficient parallel data mining for association rules. In: Proc. of the 4th Int Conf on Information and Knowledge Management. Baltimore, USA, 1995. 31～36
2 毛国君，段立娟，王实，等. 数据挖掘原理与算法[M]. 北京：清华大学出版社，2005
3 Agrawal R, Srikant R. Fast algorithms for mining association rules [C]. In: Proceeding of the 20th International Conference on Very Large Databases, 1994. 487～499
4 何小卫. Apriori算法强项集产生的二维哈希算法[J]. 计算机与现代化，2003,4:10～12
5 张江，傅鹤岗. 基于关联规则的二维哈希算法的改进[J]. 计算机工程与设计，2225，26(8)：2178～2179

(上接第138页)

签名文档内容的一致性，最后将病历存储到系统数据库中。

ISignatureProvider接口 该接口用来对任意的XML文档的内容进行数字签名。clsEMRSinger类实现了这个接口来对电子病历文档数字签名。

IVerificationProvider接口 该接口用来验证任意XML文档的数字签名。clsEMRValidator类实现了这个接口来验证电子病历文档签名的正确性。

总结 为了保证对信息的不可抵赖性，本文一开始就引入了XML数字签名的概念。我们用一系列XML Schema图来说明XML签名如何保证对病历文档的不可抵赖性。然后给出了目前系统实现的步骤与方法。文章最后给出了系统的API接口，实现了电子病历文档的签名与验证。

参 考 文 献

1 W3C. XML Signature Syntax and Processing [R]. http: //www.w3.org/TR/XMLDSIG,2003
2 W3C. XML Schema [EB/OL] http://www.w3.org/XML/Schema
3 Microsoft. netframework [R]. http: //msdn.microsoft/netframework/security,2002
4 BoyerJ. Canonical XML Version 1.0 [S]. RFC 3076,March 2001
5 李凤银. 基于XML的多人数字签名技术研究与实现[J]. 计算机科学，2004,1(7):109～114
6 李双庆，游莲，古平，等. 一种基于XML的数据交换中间件技术[J]. 计算机科学，2003
7 马永恒，熊前兴，杨金娥. W3C XML Schema模式的设计方法研究[J]. 计算机应用研究，2006(5)
8 肖林. 浅谈电子病历[J]. 档案管理，2000,4(1)
9 严维良，于津. XML在电子病历中的应用[J]. 汕头大学学报，2003,15(3)
10 张勇，冯玉才. XML数字签名技术及其在java中的具体实现[J]. 计算机应用，2003,23(9)
11 李浩，孙统风，孟现飞，等. 基于面向对象思想构建XML Schema[J]. 微机发展，2003,13(6):59～64