南京航空航天大学

论文集

（二〇〇一年）

第9册

四院

南京航空航天大学科技部编

二〇〇二年六月

四　　院

○四二系

目 录

序号	姓 名	职 称	单位	论 文 题 目	刊物、会议名称	年、卷、期	类别
1	徐 敏	中级	042	基于抽样技术的序列模式的维护	计算机应用研究	011803	J
2	徐 敏 秦小麟	中级 正高	042 042	应用聚类分析对关联规则进行分组	南京航空航天大学学报	013302	J
3	刑 刚 周 良	硕士 中级	042 042	MSMQ 与 ASP 在分布式环境下的应用研究	计算机应用研究	0118 增刊	J
4	陈 兵 王立松	中级	042 042	Socks 代理服务器的研究与实现	网络安全技术与应用	010009	
5	陈 兵 王立松	中级	042 042	HTTP 代理缓存的实现	信息工程大学学报	0102 增刊	
6	范 涛 顾其威	硕士	042 042	分布对象技术及其在数据库处理中的应用	南京邮电学院学报	012101	J
7	钱红燕	中级	042	WAP 手机漫游的研究与实现	小型微型计算机系统	012210	H
8	周冬平 毛宇光	硕士 副高	042 042	Soft Base 查询优化器的设计与实现	南京航空航天大学学报	013304	J
9	毛宇光 周 勇	副高	042 042	中介逻辑谓词演算系统 MF^M	计算机科学	011809 增刊	H
10	毛宇光 韩 波	副高	042 042	中介逻辑命题演算系统 MP^M 的范式	计算机科学	012809 增刊	H
11	毛宇光	副高	042	Cbase 查询优化器的实现	计算机科学	012808 增刊	H
12	毛宇光	副高	042	用于不完全信息数据库的中介逻辑演算系统 MP^M	计算机科学	012808 增刊	H
13	臧 洌 李 俊	中级 正高	042 042	基于 B/S 模式的高校财务电算化系统的设计与实现	航空计算技术	013101	
14	臧 洌 杜庆伟	中级	042 042	高校财务电算化系统的设计与实现	计算机应用研究	2001	H*
15	臧 洌	中级	042	人工神经网络在混沌时间序列数据处理中的应用	数据采集与处理	011604	H
16	臧 洌 王珊珊 吕 兵	中级 副高	042 042 042	计算机基础课程的分级教学	计算机基础教育研究与探索	2000 年论文集	
17	臧 洌	中级	042	使用现代化教学手段管见	南京大学学报哲学社会科学版(专辑)	000300	H*
18	李 俊	正高 正高	042	联邦式异构数据库应用系统的集成框架和实现技术的研究	计算机应用研究	011804	J
19	张卓莹	中级	042	程序正确性证明方法	计算机应用研究	2001	J
20	张卓莹 是锦春	中级	042 042	异种计算机互连通信软件系统	计算机工程与设计	012202	J
21	万麟瑞 骆洪青	硕士 副高	042 042	敏捷制造框架体系结构研究	计算机工程与设计	012202	J

序号	姓　名	职 称	单位	论 文 题 目	刊物、会议名称	年、卷、期	类别
22	李志飞 万麟瑞 胡　宏	硕士 副高 硕士	042 042 042	WAP 应用中的 PUSH/PULL 集成机制研究	小型微型计算机系统	012210	H
23	李志飞 万麟瑞 潘曙光	硕士 副高 硕士	042 042 042	WAP 在 TCP/IP 网络上的协议模拟方法	小型微型计算机系统	012206	H
24	朱朝晖 李　斌	副高 副高	042 042	有限语言下的赋值结构	南京航空航天大学学报	013303	J
25	朱朝晖 李　斌	副高 副高	042 042	单可表示类的特征	计算机学报	0122406	H
26	朱朝晖 李　斌	副高 副高	042 042	A note on conditional implication	南京航空航天大学学报(英文版)	011801	J
27	朱朝晖 张东摩	副高 正高	042 042	Some contributions to nonmonotonic consequence	Journal of Computer Sci & Tech	011604	H
28	夏万军 陈松灿	硕士 正高	042 042	多重加权双向联想记忆模型及决策性能研究	南京航空航天大学学报	013206	J
29	陈松灿 蔡　骏	正高	042 042	多重加权多值指数 BAM 及其表块性能	计算机学报	012402	H
30	彭宏京 陈松灿	博士 正高	042 042	基于稀疏 RAM 的 N–tuple 神经网络模型	计算机科学	012811	H
31	段永柱 陈松灿	硕士 正高	042 042	改进的 SDM 模型及其在函数逼近中的应用	模式识别与人工智能	011402	H
32	张本柱 陈松灿	硕士 正高	042 042	改进的 CBP 网络与 VQ 的等价	数据采集与处理	011603	H
33	皮德常 秦小麟 王宁生	中级 正高 正高	042 042 053	多层分组叠加有限制特征编码	小型微型计算机系统	012205	H
34	皮德常	中级	042	C 和 Java 课程分析	南京航空航天大学学报(社科)	0103 增刊	
35	皮德常 王宁生	中级 正高	042 042	CIMS 中数据挖掘与遗传算法的自组织研究	信息与控制	013007	H
36	李臻峰 黄志球	硕士 副高	042 042	通用静态格式报表打印设计	计算机工程	012704	J
37	左银龙 黄志球 高　鹏	硕士 副高	042 042 042	分布式多层应用系统的设计与实现	计算机工程	012703	J
38	郑晓妹 徐　涛	硕士 副高	042 042	一种 Vague 数据模型及属性的归纳依赖关系	南京航空航天大学学报	013304	J
39	吴晓汶 周　良 谢　强	硕士 中级	042 042 042	基于 CB 与 B/S 混合模式的人力资源管理系统	计算机应用研究	2001 增刊	J
40	周　良 姜大志 姜正良	中级	042 042 042	组合式空调器计算绘图一体化软件系统设计	南京航空航天大学学报	013303	J
41	周　良 谢　强 郑洪源	中级	042 042	质量管理系统中故障模式的设计与实现	小型微型计算机系统	012212	H
42	陈　菲 秦小麟	硕士 正高	042 042	空间索引的研究	计算机科学	012812	H

基于抽样技术的序列模式的维护*

徐　敏[1]，金远平[2]
(1.南京航空航天大学，江苏 南京 210016；2.东南大学，江苏 南京 210018)

摘　要：维护已发现的序列模式的方法主要有两种：一种是简单地利用已有的挖掘序列模式算法对更新后的整个数据库进行操作，这种方法涉及的数据库中的数据不仅有改变的部分而且有未改变的部分，而未改变的数据数量很大，当更新频率高时，代价是非常大的；另一种方法是根据库中记录数目改变的多少来决定何时对整个数据库进行操作，但是记录数目变化大并不能代表序列模式变化亦大，因此利用样品抽样的方法来评估序列模式改变的程度，并根据改变的程度决定何时对整个数据库进行操作来更新序列模式，从而较好地解决序列模式维护的问题，能高效地、准确地发现序列模式。

关键词：数据挖掘；序列模式；抽样；SMSP(using Sampling to Maintain Sequence Pattern)算法
中图分类号：TP311.13　　文献标识码：A　　文章编号：1001-3695(2001)03-0065-03

The Maintenance of Sequential Patterns Based on Sampling Techniques

XU Min[1]，　JIN Yuan-ping[2]
(1.Dept. of Computer, Nanjing University of Aeronautics and Astronautics, Nanjing Jiangsu 210016; 2.Dept. of Computer, Southeast University, Nanjing Jiangsu 210018, China)

Abstract: The methods to solve the problem of maintaining discovered sequential patterns have mainly two kinds. One is simply applying algorithms of mining sequential patterns to the updated database, but it scans not only changed data but also unchanged data in the original database which is very large. If the database is updated frequently, it takes much time. Another is according to the number of records changed in the database to decide when to operate the whole database, but the number of sequential patterns changed is not in proportion to the number of records changed. So we use sampling techniques to estimate the degree of sequential patterns changed to determine whether we should update mined sequential patterns by operating the whole database or not. This can solve better the problem and find sequential patterns efficiently and exactly.

Key words: Data mining; Sequential patterns; Sampling; SMSP(using Sampling to Maintain Sequence Pattern)

1　介绍

从大量数据中发现序列模式是知识发现与数据挖掘的一个重要问题。序列模式已在很多领域得到应用，如货篮分析(Basket analysis)、通讯网络管理和WWW等方面。在超级市场的购物事务中发现序列模式可以帮助预测用户将来的行为。在通讯网络管理中发现报警序列模式可以应用到在线预测、分析和矫正网络错误方面。在Web上，通过对Server上的Web日志数据库中用户访问模式的发现，可以帮助改进系统设计(如改变超链结构)和更好地商业决策(如重要广告摆放位置)等。挖掘序列模式的有效算法[1,2]和约束条件下的序列模式[4]都已有研究，然而，它们为了发现精确的序列模式必须扫描整个数据库，同时，由于数据库中的数据每时每刻都发生变化(如Web日志数据库)，维护已发现的序列模式代价是相当大的。本文提出利用抽样技术估计当数据库中数据发生更新后序列模式改变最大数量并设定若干标准，当改变数量小于标准时，可认为原来发现的序列模式依然很正确；当该数量变大时，再利用文献[1，2]中的有关算法来刷新序列模式。由于样本相对于整个数据库数据量来说很小，可全部放置内存中处理，故速度快，且由于采用统计方法，保证了正确性。

收稿日期：2000-07-31
基金项目：江苏省自然科学基金资助项目(BK97002)

2　问题定义

定义：　设$I=\{i_1, i_2, \ldots, i_m\}$是一个字符集，称为项目的集合，每个字符称为项目。由非零个项目组成的集合称为项目集。

一个序列是项目集的一个有序集。一个序列s用$<s_1, s_2, \ldots, s_m>$来表示，这里s_j表示一个项目集，s^k表示s序列中含有k个项目集。序列中的一个元素s_j用$<x_1, x_2, \ldots, x_m>$表示，这里x_j表示一个项目。

有两个序列$a<a_1, a_2, \ldots, a_n>$和$b<b_1, b_2, \ldots, b_m>$，若存在$i_1 < i_2 <, \ldots, < i_n$使得$a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}$，则称$a$为$b$的子序列。这里再加上时间约束CT：规定一个序列中相邻的两个元素之间发生的时间间隔小于或等于用户给定的最大的时间间隔G_{max}且大于给定的最

小时间间隔G_{min}。同时规定序列中的一个元素中的事件发生的最大时间间隔小于或等于用户定义的时间窗T_w。即若上述a为一个序列还必须满足下面条件(注：$T(x_i)$为x_i发生的时间)：存在整数$l_1 \le u_1 < l_2 \le u_2 < \cdots\cdots < l_n \le u_n$使得 (1) a_i是在$[l_i, u_i]$间隔中，$1 \le i \le n$；

(2) $T(a_{u_i}) - T(a_{l_i}) \le T_w$，$1 \le i \le n$；

(3) $T(a_{l_i}) - T(a_{u_{i-1}}) > G_{min}$，$2 \le i \le n$；

(4) $T(a_{u_i}) - T(a_{l_{i-1}}) \le G_{max}$，$2 \le i \le n$。

输入是一个序列数据库D_s，每一个序列是项目集的一个有序集。这里的项目集可以是由单个项目(如WWW服务器上的页面标识等)或多个项目(如顾客一次购物的商品的集合)构成。设序列数据库D_s中的序列满足时间约束CT，$|D_s|$表示库中所含元组的总数，$|s|$表示D_s含有序列s的元组数目。序列数据库D_s存在序列模式s是指$|s|$大于用户给定的阈值$|D_s|$Supmin%。经过一些更新活动，老的序列被删除，新的序列被添加。对于被修改的序列可看作先删除再添加，所以不失一般性。假设有更新活动，所有的修改都可看作删除和添加活动。用D_s^-表示被删除的序列，D_s^+表示被添加的序列，被更新的数据库用D'_s表示，$D'_s = D_s - D_s^- \cup D_s^+$。

3 SMSP算法

用文献[3]中的GSP算法求D_s的序列模式是很有效的，但是若用GSP算法来维护已发现的序列模式则必须在数据库每次更新后对整个数据库数据进行操作，若更新频繁，代价是非常大的。若根据库中数据改变多少来决定何时对数据库进行扫描，但数据改变多少并不能代表序列模式改变很多。因此这里引进抽样方法，根据样本(可整个放入内存中)和改变的数据来评估序列模式变化情况，当由库中元组变化引起序列模式变化不显著时可忽略，可以认为序列模式没有变化；当变化显著时再用GSP算法求新库中的序列模式，这样可及时地和高效地挖掘序列模式。

3.1 随机样品的抽取

对于数据库D_s和一个序列s，若有n_s个元组包含s，根据古典概率可知从D_s中随机抽取元组含s的概率为$p_s = n_s / |D_s|$。在D_s中按独立同分布原则抽取m个元组作为样本S代替D_s。在S中每个元组含有序列s的概率为p_s。设T_s为样本S中含有序列s的元组数目，T_s是一个服从二项分布的随机变量(记为：$T_s \sim B(m, p_s) m > 30$)，$T_s$的数学期望为$mp_s$，方差为$mp_s(1-p_s)$。$n_s$的值可用$\hat{n}_s = \frac{T_s}{m}|D_s|$表示，因为$\hat{n}_s$是$n_s$的无偏估计。$n_s$期望是$mp_s(\frac{|D_s|}{m})$，方差是$mp_s(1-p_s)(\frac{|D_s|}{m})^2$。所以可以得到在显著水平$\alpha$下，$n_s$的置信区间为$[a_s, b_s]$，

$$a_s = \hat{n}_s - z_{\alpha/2}\sqrt{\hat{n}_s(|D_s| - \hat{n}_s)/m} \quad (1)$$

$$b_s = \hat{n}_s + z_{\alpha/2}\sqrt{\hat{n}_s(|D_s| - \hat{n}_s)/m} \quad (2)$$

$z_{\alpha/2}$是标准正态分布的上分位点，α一般取0.1($z_{\alpha/2}$ = 1.645)或0.05($z_{\alpha/2}$ =1.96)。

下面利用置信区间来求应该取得样本S的数目m。

由(1)(2)两式可得置信区间的间隔为$2z_{\alpha/2}\sqrt{\hat{n}_s(|D_s| - \hat{n}_s)/m}$。因为只有当$s$在$D_s$中不是序列模式时，SMSP算法才用置信区间来估计$n_s$ (理由见3.2部分)，所以$n_s < |D_s|$ Supmin%。这就决定了n_s的上限。一般Supmin<50，故$n_s < |Ds|/2$。现在看函数$\beta(\hat{n}_s) = \hat{n}_s(|D_s| - \hat{n}_s)$，它在$(-\infty, |D_s|/2)$区间递增，在$(|D_s|/2, \infty)$区间递减。对于给定的Supmin，$\hat{n}_s \approx n_s \in [0, |D_s|\text{Supmin\%}] \subseteq (-\infty, |D_s|/2)$，函数$\beta(\hat{n}_s)$在这区间中是递增的，所以$(\beta|D_s|\text{Supmin\%})$的上限为$|D_s|^2$Supmin%(1-Supmin%)。置信区间的宽度不会超过：

$$2z_{\alpha/2}\sqrt{|D_s|^2\ \text{Supmin\%}(1-\text{Supmin\%})/m} = 2z_{\alpha/2}|D_s|\sqrt{\text{Supmin\%}(1-\text{Supmin\%})/m}$$

由这个限制可知，样本数量m越大，置信区间越小，精确度越高。可以通过样本数目来控制置信区间的宽度，反之通过规定置信区间的宽度来决定样本的数量。假设置信区间的宽度不超过$|D_s|$Supmin%/5，可得到不等式：

$$2z_{\alpha/2}|D_s|\sqrt{\text{Supmin\%}(1-\text{Supmin\%})/m} \le |D_s|\text{Supmin\%}/5$$

从而解决最小样本数目问题：

$$m = \text{Supmin\%}\ (1-\text{Supmin\%})(10z_{\alpha/2}/\text{Supmin\%}) \quad (3)$$

例如：Supmin=3，α=0.05($z_{\alpha/2}$=1.96)，可得$m \ge$ 12414.836。同时，从式子(3)可知m的取值与数据库D_s的大小无关，于是当D_s中有上亿的元组时，只要取12 500个元组即可达到期望的精确度。

3.2 SMSP 算法中的标准

设在未更新的数据库D_s中，L表示存放所有的序列模式的集合，用L_k表示存放s^k的序列模式集合，用Δ表示在D_s数据库中含s^k的元组数目，这些可由GSP算法得出。L'_k表示在更新的数据库D'_s中存放s^k的序列模式集合。

SMSP算法是一个类似Apriori的循环算法。用C_k表示存放s^k的候选序列模式集合，$\hat{L}_k$表示存放s^k的序列模式集合，注意$\hat{L}_k$是L'_k的估计。

在第一次循环时，C_1是存放s^1的候选序列模式集合，即为D_s^-和D_s^+数据库中的项目集。在第k次循环时，C_k是存放s^k的候选序列模式集合，是由L_{k-1}得到的，具体算法见后面。扫描D_s^-和D_s^+数据库，可得到s^k在D_s^-和D_s^+数据库中含s^k的元组数目，用Δ^+和Δ^-分别表示。这里分两种情况：(1)若$s^k \in L_k$，那么在D_s中含s^k的元组数目可从先前挖掘的结果中得到为Δ，这时$|s^k| = \Delta^+ - \Delta^- + \Delta$其为实际值，当$|s^k| \ge |D_s|$Supmin%时，$s^k$在更新的数据库$D'_s$中为序列模式。将其加入$L^1_k$中，用

Y存放变化的序列模式即$Y=(L_k-L^1{}_k)\cup(L^1{}_k-L_k)$，变化的数目用$Num1_k=|Y|$表示。(2)若$s^k\notin L_k$，那么在$D_s$中含$s^k$的元组数目不可以从先前挖掘的结果中得到，但可确定$\Delta$一定小于$|D_s|Supmin\%$。这时若$\Delta^+-\Delta^-<(|D_s^+|-|D_s^-|)Supmin\%$，则$s^k$一定不是序列模式，因为$|s^k|<|D_s|Supmin\%+(|D_s^+|-|D_s^-|)Supmin\%=(|D_s|-|D_s^-|+|D_s^+|)Supmin\%=|D'_s|Supmin\%$，即不满足支持度，可从$C_k$中删除。反之，通过扫描样本$S$来发现显著水平为$\alpha$的置信区间$[a_s, b_s]$，对应于$|s^k|$的置信区间$[a_s+(|D_s^+|-|D_s^-|)Supmin\%, b_s+(|D_s^+|-|D_s^-|)Supmin\%]$，记作$[a'_s, b'_s]$。根据$|D'_s|Supmin\%$和$|s^k|$的置信区间的比较可得到三种可能：

1)若$|D'_s|Supmin\%<a'_s$，可以认为s^k为序列模式的可能性为$(1-\alpha)\%$，将其加入$L^2{}_k$中。

2)若$a'_s<|D'_s|Supmin\%<b'_s$，这时不能确认s^k是否为序列模式，但是其接近阈值，因此可以认为其非常可能为序列模式，将其加入$L^3{}_k$中。

3)若$b'_s<|D'_s|Supmin\%$，可以认为s^k不为序列模式的可能性为$(1-\alpha)\%$，将其从C_k中删除。

设$Num2_k=|L^2{}_k|+|L^3{}_k|$。在第$k$次循环最后，用$\hat{L}_k=L^1{}_k\cup L^2{}_k\cup L^3{}_k$来作为$L'_k$的估计。完成第$k$次循环后，进行评估，判断是否需要进行第$k+1$次循环。评估依据以下三个标准：

1)由于$L^3{}_k$是否为序列模式的确定因素不大，因此为了控制其影响范围，用$u_k=|L^3{}_k|/|\hat{L}_k|$对比于用户给定的$\bar{u}$，若$u_k\geq\bar{u}$则认为可靠性差，停止SMSP算法，转向GSP算法。

2)由于$Num1_k$的数目是精确的，而$Num2_k$是估计的，为了保证估计的准确度，用$d_k=\sum_{j=1}^{K}(Num1_k+Num2_k)/|L|$对比于用户给定的$\bar{d}$，若$d_k\geq\bar{d}$则认为可靠性差，停止SMSP算法，转向GSP算法。

3)采用与Apriori算法相同的结束条件。若由$\hat{L}_k$得到的C_{k+1}不为空，则进行下一个循环；否则算法终止。假如算法依据第三标准终止，则可认为L表示存放D'_s中所有的序列模式的集合

3.3　具体的算法

$C_1=\{s^1\mid s\in D_s^-$和D_s^+数据库中的项目集$\}$
for ($k=1$; $C_k\neq\Phi$; k++)
{　foreach $s^k\in C_k$ do
　{ 扫描D_s^-和D_s^+数据库，得到$s^k\Delta^-$和$s^k\Delta^+$　}
foreach $s^k\in L_k$ do
　if $s^k\Delta^+-s^k\Delta^-+s^k\Delta<|D'_s|Supmin\%$　then　$C_k=C_k-s^k$
　　else将s^k移入到$L^1{}_k$;
　　$Num1_k=|(L_k-L^1{}_k)\cup(L^1{}_k-L_k)|$;
　　foreach $s^k\in L_k$ do
　　{ if　$s^k\Delta^+-s^k\Delta^-<(|D_s^+|-|D_s^-|)Supmin\%$ then $C_k=C_k-s^k$
　　else { 扫描样本S来发现显著水平为α的置信区间$[a_s, b_s]$;
　　if $|D'_s|Supmin\%<a_s+(|D_s^+|-|D_s^-|)Supmin\%$ then
　　　将s^k移入到$L^2{}_k$;
　　if $a_s+(|D_s^+|-|D_s^-|)Supmin\%<|D'_s|Supmin\%<b_s+(|D_s^+|-|D_s^-|))Supmin\%$ then 将s^k移入到$L^3{}_k$;
　　if $|D'_s|Supmin\%>a_s+(|D_s^+|-|D_s^-|)Supmin\%$ then $C_k=C_k-s^k$;}
　　$Num2_k=|L^2{}_k|+|L^3{}_k|$;
　　$\hat{L}_k=L^1_k\cup L^2_k\cup L^3_k$;
　　if $u_k=|L^3{}_k|/|\hat{L}_k|\geq\bar{u}$ then 停止SMSP算法，转向GSP算法;
　　if $d_k=\sum_{j=1}^{K}(Num1_k+Num2_k)/|L|\geq\bar{d}$ then 停止SMSP算法，转向GSP算法;
　　按照C_{k+1}算法$\hat{L}_k$得到C_{k+1};}

C_{k+1}算法有两个步骤：

第一步，合并步骤，是$\hat{L}_k$和$\hat{L}_k$合并。具体如下所示：

forall $p, s\in\hat{L}_k$
　if ($s_1=q_1, s_2=q_2, \ldots, s_{k-1}=q_{k-1}$ 且满足CT条件)
　then合并p, s 为$s=\{s_1, s_2, \ldots, s_k, q_k\}$加入$C_{k+1}$中;

第二步，裁减步骤，删除所有$s_{k+1}\in C_{k+1}$且s_{k+1}的s_k子集s不全在$\hat{L}_k$中，即：

forall $s_{k+1}\in C_{k+1}$ do
　forall s_{k+1}的s_k子集 s　do
　　if ($s\notin\hat{L}_k$) then 从C_{k+1}中删除s_{k+1}。

4　小结

用南京航空航天大学WWW服务器上两年的日志作为实验数据构成序列数据库，用IP地址和日期来识别来访的用户，一个序列记录一个用户访问WWW服务器页面的顺序。这里用1.5年的序列数据作为D_s，$|D_s|$=231 234，另外0.5年作为更新数据，取Supmin=3，α=0.05($z_{\alpha/2}$=1.96)，m=15 000，为了保证算法的精确度设$\bar{u}$=0.02，$\bar{d}$=0.05。用SMSP算法进行判断是否要用GSP算法，结果发现平均每六个星期需要用一次GSP算法，最短时间间隔为两个星期，最长时间间隔为九个星期。这样可不必每次更新数据库都用GSP算法求序列模式，与根据更新数据库元组数目来决定是否用GSP算法相比，更具有实时性和高效性，从而为用户提供及时、有效的信息来进行预测和决策。

参考文献：

[1] R Agrawal, R SriKant. Ming Sequential Patterns[C]. in Proc. of 11th Intl Conf. on Data Engineering, 3,1995.
[2] R SriKant, R Agrawal. Ming Sequential Patterns: Generalizations and Performance Improvements[C]. in Proc. of 5th Intl Conf. on Extending Database Technology, 3, 1996.
[3] S D Lee, David W Cheung, Ben Kao. Is Sampling Useful in Data Mining? A Case in the Maintenance of Discovery Association Rules[Z]. 1999.
[4] Minos N Garoflakis, Rajeev Rastogi, Kyuseok Shim. SPIRIT: Sequential Pattern Mining with Regular Expression Constains[C]. in Proc. of 25th Intl Conf. on VLDB, 1999.

作者简介：

徐敏，男，教师，硕士，主要研究方向为数据挖掘、数据库系统；金远平，男，教授。

第 33 卷第 2 期
2001 年 4 月

南 京 航 空 航 天 大 学 学 报
Journal of Nanjing University of Aeronautics & Astronautics

Vol. 33 No. 2
Apr. 2001

文章编号:1005-2615(2001)02-0197-03

应用聚类分析对关联规则进行分组

徐 敏 秦小麟

(南京航空航天大学信息科学与技术学院 南京,210016)

摘要 关联规则是要从大量的数据中找到数据之间的规律,但有时所产生的规律十分繁多,从而形成新的知识管理问题。针对该问题本文提出了一个新的算法,该算法利用系统聚类分析方法对规则进行分组,从而可更好地帮助用户理解所发现的规律,该方法的距离(RatioD)是基于关联规则本身,因此,可对规则进行高效地分组。实验结果表明,该算法是有效的。

关键词:数据挖掘;关联规则;分组;系统聚类法;RatioD 距离

中图分类号:TP311.13 **文献标识码**:A

近年来随着数据库和计算机网络的广泛应用,全球范围内数据库中存储的数据量急剧增大,人们需要能够对数据进行较高层次处理的技术,从中找出规律和模式,以帮助人们更好地利用数据进行决策和研究,数据挖掘因此成为研究热点。关联规则[1~6]是数据挖掘的主要内容之一。但在挖掘的结果中所产生的规则十分繁多,必须对发现的关联规则采取一定的方法进行处理,以更好地理解所发现的规律。文[5]利用 OLAP 的"preprocess-once-query-many"框架,在挖掘关联规则过程中,消除一定范围内的规则冗余,减少生成的规则数目,便于人们理解。但其方法应用范围有局限性。文[4]利用关联规则的独立性对规则进行修剪和分组,但必须扫描数据库中的数据,所需时间和空间大。本文利用聚类分析方法和规则本身结构特点对规则进行高效地分组。

1 问题分析

给定一组项目 $I=\{i_1,i_2,\cdots,i_m\}$和事务数据库 D,D 中每个事务 T 是 I 的子集,且由其 TID 唯一标识。如果项目组 $X\subseteq T$,则称 T 包含 X。关联规则是形如 $X\rightarrow Y$ 的逻辑蕴涵式,其中 $X\subset I$,$Y\subset I$ 且 $X\cap Y=\Phi$。若 D 有 $s\%$的事务包含 $X\cup Y$,则称 $X\rightarrow Y$ 的支持度为 s;若 D 中包含 X 的事务中有 $c\%$的事务同时也包含 Y,则称 $X\rightarrow Y$ 的可信度为 c。若 $X\rightarrow Y$ 的支持度和可信度分别大于用户设定的最小支持度 minsup 和最小可信度 mincon,则称关联规则 XY 成立(记为 $X\rightarrow Y(s,c)$其中 s 表示支持度,c 表示可信度)。

在寻找关联规则的结果中,经常会发现关联规则之间有某些相似关系,利用这些关系可对规则进行分类,以致得到更简明有效的结果。

例如,在学生注册课程数据库进行关联规则查找,发现关联规则

C 语言程序设计,对象数据库→数据通信 (0.02,0.9)

C 语言程序设计,数据通信→分布数据库 (0.01,0.9)

机械原理,画法几何及机械制图,微机原理及应用→机械设计基础(0.01,0.8)

它们分别由若干门课程组成。很显然,上面两个规则所涉及的是计算机专业的学生,应该分在一组。因此,可利用系统聚类分析方法[7]根据规则之间的相似程度来对规则进行分组。

基金项目:江苏省自然科学基金(编号:BK97002)、南京航空航天大学青年科研基金(编号:S0030-802)资助项目。

收稿日期:2000-05-15;修订日期:2000-10-24

作者简介:徐 敏,男,助教,1971 年 4 月生;秦小麟,男,教授,博士导师,1953 年 6 月生。

2 规则分组的算法

2.1 距离定义

采用聚类方法进行分组首先要定义距离，而传统的欧氏距离已不适用这种情况。

例如：考虑一组规则 $X_i \rightarrow Y_i (i=1,\cdots,4)$，用 g_i $(i=1,\cdots,4)$表示每个规则所含的属性集合，其中 $X \cup Y$ 的属性有 6 个，$g_1=\{1,2,3,5\}$，$g_2=\{2,3,4,5\}$，$g_3=\{1,4\}$，$g_4=\{6\}$。它们可用向量表示，那么 $g_1=111010$，$g_2=011110$，$g_3=100100$，$g_4=000001$。根据系统聚类法合并原则，开始每个样品为一个类，则有 4 个类 G_1,G_2,G_3,G_4，类之间的距离即为样品之间的距离。用欧氏距离计算它们之间的距离分别为 $D_{12}=d_{12}=\sqrt{2}$，$D_{13}=d_{13}=2$，$D_{14}=d_{14}=\sqrt{5}$，$D_{23}=d_{23}=2$，$D_{24}=d_{24}=\sqrt{5}$ 和 $D_{34}=d_{34}=\sqrt{3}$。根据系统聚类法合并原则，开始每个样品为一个类，d_{12} 距离最小，因此它们首先合并，合并的新类重心为(0.5,1,1,0.5,1,0)。这时，删除 G_1,G_2 两个类，新类标记为 G_1，它与其他类的距离分别为 $D_{13}=\sqrt{3.5}$，$D_{14}=\sqrt{4.5}$。因此 $D_{34}=\sqrt{3}$ 的距离最短，它们合并。

然而，G_3 为 $X_3=\{1,4\}$ 和 G_4 为 $X_4=\{6\}$，它们没有相同的属性，但根据传统的欧氏距离的重心法却合并了。所以按照这样的方法会造成属于不同组的规则分在一组。一旦不属于一类的规则分在一类，那么在聚类过程中会变得越来越坏。这使两个规则只具有少数属性不同与两个规则都具有少数属性的情况难以区分。

为了克服上述缺陷，现在采用新的距离定义，先计算两个类中的规则所含属性的交集的数目与它们规则所含属性的并集的数目的比值 k，再用 $1-k$ 的值作为它们的距离，即

$$D_{ij}=1-\frac{|X_i \cap X_j|}{|X_i \cup X_j|} \qquad (i,j=1,2,\cdots,n) \tag{1}$$

合并后的新类为两个规则集合的并集(称该距离为 RatioD 距离)。

例如　就上面的例子，第一次合并时类之间的距离分别为：$D_{12}=d_{12}=\frac{2}{5}$，$D_{13}=d_{13}=\frac{4}{5}$，$D_{14}=d_{14}=1$，$D_{23}=d_{23}=\frac{4}{5}$，$D_{24}=d_{24}=1$ 和 $D_{34}=d_{34}=1$，根据系统聚类法合并原则，开始每个样品为一个类，d_{12} 距离最小，因此它们首先合并，这与上例一致。合并的新类用属性向量表示为(1,1,1,1,1,0)。这时，删除 G_1,G_2 两个类，新类标记为 G_1，它与其他类的距离分别为 $D_{13}=\frac{3}{5}$，$D_{14}=1$。因此 $D_{13}=\frac{3}{5}$ 的距离最短，它们合并，而不是 G_3 和 G_4 合并，从而解决了该矛盾。

2.2 分组算法

基于 RatioD 距离的分组算法的主要思路是：

先将所求得的所有的关联规则各自看作一类，然后按照式(1)求规则之间的距离和类与类之间的距离。开始，每个规则自成一类，因此类与类之间的距离和规则与规则之间的距离是相等的，选择距离最小的一对并成一个新类，计算新类和其他类的距离，再将距离最近的两类合并，这样每次减少一类，直到所有的规则都成一类为止。新类为被合并的两个类集合的并集。

具体算法如下：

(1)按照式(1)计算规则两两之间的距离，开始每个规则自成一类，这时显然有 $D_p=d_{pq}$。

(2)选择最小 D_p，即将 G_p 和 G_q 和并成一个新类，记为 G_r，$G_r=\{G_p,G_q\}$。

(3)计算新类与其他类的距离

$$D_{rk}=\min_{i\in G_r,j\in G_k} d_{ij}=\min\{\min_{i\in G_p,j\in G_k} d_{ij},\min_{i\in G_q,j\in G_k} d_{ij}\}=\min\{D_{pk},D_{qk}\}$$

将 G_p 和 G_q 两个类删除。

(4)对剩下的类重复上述(2,3)，直到所有的规则成为一类为止。

如果某一步最小元素不止一个，则对应这些最小元素的类可以同时合并。

3 实验结果

本算法在 PC 586/MMX166 (32M 内存)机上用 Visual C^{++} 6.0 实现，并用学生注册课程数据库作为测试数据进行测试。

库中有 1 050 个学生在课程注册数据库中注册，每个学生至少注册两门课。平均每行有 5 门课。总共有 110 门课。在这数据库中取学生在计算机系注册的数据，用文[2]中的 Apriori 算法进行查找满足最小支持度 0.02，最小可信度为 0.8 的关联规则有 216 个，共有 15 个不同的属性。对这些规则，用本文实现的算法进行聚类分析，得出的聚类图，结果分成三类，发现第一类包含的课程有计算机网络原理、协议工程和分布式操作系统，很明显

这部分学生的专业应该是研究网络的；第二类包含的课程有数据库原理，面向对象数据库和计算机图形学，与这些课程相匹配的学生的专业应该是信息管理学；第三类包含的课程有计算机原理，数据结构，C 语言程序设计，与这些课程相匹配的学生的专业则应比较分散。对分析的结果与实际这三类规则所涉及的学生相比一致率达到 98%。因此，从试验结果中可表明发现规则集合的分类符合实际情况，它能使用户得到更清晰的信息。

4 结　论

本文提出了一种距离(RatioD)是基于关联规则本身的系统聚类的方法进行规则分组，它与以前的方法(如文[4])相比，由于分组仅从关联规则本身出发，不必为了分组再对数据库进行扫描而浪费时间，也不必为了节省时间对每个关联规则加若干项说明进行分组而占用更大的空间，所以可快速地、节省存贮空间地给规则进行分组，并且适用性更广。实验结果表明，该方法对规则进行分组能使用户得到简洁、清晰、明确的信息，便于分析。

参 考 文 献

1 Agrawal R, Imieinski T, Swmi S. Mining association rules between sets of items in large databases[C]. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, USA: Washington DC. May 1993. 207～216

2 Agrawal R, Srikant R. Fast alogrithms for mining association rules in large databases[C]. In: Proceedings of the 20th international Conference on Very Large Databases, September 1994. 452～461

3 Jong Soo Park, Chen Mingsyan, PhiliP S Yu. An effective hash-based algorithm for mining association rules[C]. In: ACM, 1995. 175～186

4 Toivonen H, Klemettinen M, Ronkainen P. Pruning and grouping discovered association rules[C]. In: On Statistics, Machine Learning and Discovery in Databases, Heraklion, Greece, 1995. 47～52

5 Aggarwal C C, Philip S Yu. Online generation of association Rules[C]. In: Proceedings of the Fourteenth International Conference on Data Engineering, USA: Orlando, Florida, IEEE Computer Society, 0-8186-8289-2/98, 1998. 402～411

6 Lin Junlin, Dunham M H. Mining association rules: anti-skew algorithms [C]. In: Proceedings of the Fourteenth International Conference on Data Engineering, Orlando, Florida, USA: IEEE Computer Society, 0-8186-8289-2/98, 1998. 486～493

7 张尧庭，方开泰. 多元统计分析引论[M]. 北京：科学出版社，1982. 393～426

Grouping Association Rules with Cluster Analysis

Xu Min　　*Qin Xiaolin*

College of Information Science and Technology,
Nanjing University of Aeronautics & Astronautics　Nanjing 210016, P. R. China

Abstract　Association rules show the regularities in a large amount of data, but sometimes they are too many for us to be understood. So a new problem of knowledge management is created. To solve the problem a novel algorithm is presented which utilizes a hierarchical clustering method. The distance definition is based on the attributes of association rules so that these rules can be effectively grouped and be better understood. The experimental results show that the algorithm is effective.

Key words: data mining; association rules; grouping; hierarchical clustering methods; RatioD distance

MSMQ 与 ASP 在分布式环境下的应用研究

邢 刚,周 良,丁秋林
(南京航空航天大学 计算机科学与工程系,江苏 南京 210016)

摘 要:介绍了 MSMQ 和 ASP 的基本概念,并结合实例介绍了 MSMQ 和 ASP 的实现方法及两者结合在分布式环境下的应用。

关键词:MSMQ;ASP;分布式应用

1 引言

随着 Internet 的不断发展,越来越多的信息和数据通过万维网(WWW)来进行传递,越来越多的人开始关注万维网,对 Web 的要求也越来越高,人们已经不再满足于 HTML 产生的静态页面,而是要看到数据实时变化的动态页面。于是,微软适时推出的 ASP(Active Server Pages)以其简单易学、对多种 ODBC 数据库的支持、与浏览器无关性等一系列的优点成为了一种很好的解决方案。而 MSMQ(MicroSoft Message Queue)也是微软的产品,主要用于分布式应用环境中应用程序之间的通信。本文利用 ASP 和 MSMQ 来实施企业远程办事处向总部上报信息的管理系统的设计与开发,主要介绍了 ASP 和 MSMQ 在分布式环境下的应用。

2 MSMQ 及 ASP 技术的基本要点

2.1 MSMQ 的基本原理

消息队列是分布式应用程序之间的通信路径。MSMQ 技术与命名管道,Socke,DCOM 和其它目的较简单的协议的不同之处在于,它提供无连接的消息服务,也就是说应用程序不必保持会话,消息队列允许当前并未连接的计算机通过它的存储转发能力进行通信。消息队列可以传递各种数据,并不仅仅限于文本数据。简言之,消息队列是位于两个或多个应用程序之间可靠而持久的传送媒介。MSMQ 的消息传递过程如图 1 所示。

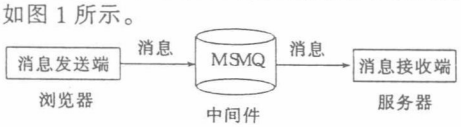

图 1 MSMQ 的通信机制

由图 1 可知,浏览器创建了消息并把它们发到了消息队列中,而服务器端则从队列中接收消息。即使两个应用程序从未同时连接到网络中,也可使用消息队列进行通信。MSMQ 将一直存储消息直到把它们可以被运送到目的地为止,这是消息队列特有的性质。

MSMQ 支持两种传输模式:快速传递(Express)和可恢复传递(Recoverable)。使用哪种传输模式与事务的性质和错误恢复所用的资源有关。一般来说,快速消息占用较少的资源,并且比可恢复性消息有更快的吞吐速度。但是,如果计算机保存的内存映射消息文件被破坏或是丢失,则快递消息不可恢复。

假设一个基于 MSMQ 的应用正在通过一个 MSMQ 路由服务器发送快递消息到另一个队列中,在这时 MSMQ 路由服务器和目的计算机之间的网络连接中断,则该消息将继续保存在 MSMQ 路由服务器的内存中;如果在网络连接被恢复之前 MSMQ 路由服务器被关闭,则该快递消息将丢失。然而,要是使用的是可恢复性消息,则消息不会丢失,在 MSMQ 服务器和网络连接恢复以后,该消息可继续被传递。

由于网络的迅速发展,越来越多的分布式应用被开发出来,使以前那些集中的企业信息管理发展成了分布式系统。在许多情况下,连接也许不是永久性的,而当你要与其它系统交流信息时,你只需将机器联到网上,通过消息队列把消息发出,同时能够接收发给你的消息并自动处理这些消息。能够支持这种分布式应用,使消息队列在创建明天的分布式企业信息应用中占有重要的一席之地。

要正确地使用消息队列,首先一定要了解消息队列的类型和对象属性,下面就做一简要介绍。

(1)消息队列的类型

消息队列可以划分为两个基本类别:

·应用程序队列 应用程序用它来传递消息,它是由使用 MSMQ 的进程创建的。

·系统队列 该队列由操作系统来创建和维护。

应用程序队列包括下面几种类型:

·消息队列 用于一个应用向另一个应用发送消息。

·管理队列 发送端程序用来接收由 MSMQ 传回的关于消息状态的反馈。

·响应队列 接收接收程序向发送程序返回的响应信息。

·报告队列 用于跟踪消息队列中消息的进度,一般是由 MSMQ 产生。

MSMQ 还创建了一些系统队列,用来支持消息队列的运行,用户不需要自己创建,但用户对它们有存取的权限。系统队列包括下面几种类型:

·日志队列 用于跟踪已经成功地被从消息队列中删除的消息。

·死信队列 用于保存无法投递的消息。

(2)MSMQ 对象属性

在使用 MSMQ 的时候,用户可能会用到三种不同的对象,分别是:机器、队列和消息。与这三种类型对象相对应的有三类属性:

·机器属性 它与特定机器上的队列管理器相关联。所有的机器属性都是只读的。

·队列属性 它与特定的消息队列相关联。每一个消息队列都具有可决定此队列行为的属性集合。

·消息属性 它与通过队列发送的单个消息相关联。通过消息队列发送的每一条消息都包含有大量属性。一些属性值是由创建该消息的应用程序确定的,而其它属性是当该消息通过消息队列进行发送时由 MSMQ 设置的。

以上只是简单介绍了各类型属性的意义,本文后面将有一些属性的应用。如要了解有关类型属性信息的详细列表,请参阅 Platform SDK 的有关文档。

2.2 Active Server Pages 概述

ASP 实际上是一种开放、不需编译的应用开发环境。它完全摆脱了 CGI 和 IDC 技术的局限性,并将 HTML 页面、Script 语言和动态服务器组件结合在一起,用户可以更方便地控制和管理数据库中的数据。程序员可以通过使用 ASP,生成动态、可交互性强、高效的 Web 应用程序。

ASP 程序可以内嵌在 HTML 文本中,并在页面被请求时由 Web 服务器解释执行后,生成 HTML 文本发送到浏览器端,由浏览器显示。而由此带来的好处是:由于 ASP 程序是由 Web 服务器解释执行后才发送到浏览器,所以浏览器看不到 ASP 源程序,这样既保证了网站的安全,也维护了作者的知识产权,并且由于整个 HTML 页面的生成过程对浏览器是透明的,所以 ASP 实现了与具体浏览器的无关性。

ASP 提供了丰富的组件,使 Web 开发人员可以更方便、更高效地编写出向用户提供动态服务的程序。特别是其中的 ADO(Active Data Object)组件,它提供了与任何 ODBC 兼容的数据库或 OLE DB 数据源的高性能连接,并通过其内部

收稿日期:2000-11-27

的对象方便地对数据库进行各种操作。

ASP还提供了对象用于多个用户共享数据和单个用户信息管理。我们知道HTTP协议的工作方式是:用户发出请求,服务器作出响应,这种联系是离散的,非连续的。在HTTP协议中没有什么能够允许服务器端跟踪用户的信息,从网站的观点看,每一个新的HTTP请求都是孤立存在的,所以HTTP协议被认为是"无状态"(Stateless)协议。Session对象的出现弥补了这个缺陷,利用Session,当一个用户在多个页面之间转换时,也能保存他的信息。而Application对象是个应用程序级的对象,用于在所有用户间共享数据,在Web应用程序运行期间一直保持数据,如不加限制,所有的客户都可以访问该对象。

3 在分布式环境下ASP与MSMQ技术的实现

在实际应用中,就像其它的MS BackOffice服务一样,MSMQ向开发者提供了API,可以利用这些接口来编制ASP程序。在MSMQ提供的API中最常用的有三个类,分别为:MSMQQueueInfo,MSMQQueue,MSMQMessage。下面将重点介绍这些类在企业远程办事处向总部上报信息的管理系统中的实现方法。

3.1 消息队列的基本管理

MSMQQueueInfo提供了队列管理。它允许你创建一个队列,打开一个存在的的队列,并可改变队列的属性和删除队列。

打开消息队列一般采用如下形式:

```
<%
Dim Obj _ QueueInfo
Dim Obj _ Queue
Set Obj _ QueueInfo = Server. CreateObject("MSMQ. MSMQQueueInfo")
Obj _ QueueInfo. PathName = ". \ Com1Queue"      '设置路径
Set Obj _ Queue = Obj _ QueueInfo. Open(MQ _ PEEK _ ACCESS, MQ _ DE-
NY _ NONE)
%>
```

以上代码打开了一个名为Com1Queue的本地服务器队列,如要打开的是异地服务器上的队列,代码可以这样写:Obj _ QueueInfo. PathName = "ServerName \ QueueName"。可以看到,QueueInfo的Open方法有两个参数,即:Access和ShareMode。Access参数是指将要对队列执行的操作,一般有以下三种操作:(1)MQ _ PEEK _ ACCESS;(2)MQ _ RECEIVE _ ACCESS;(3)MQ _ SEND _ ACCESS。MQ _ PEEK _ ACCESS在指定的队列中查找消息,但不对消息进行任何操作;MQ _ RECEIVE _ ACCESS在指定队列读取消息后删除它;MQ _ SEND _ ACCESS是向队列发送信息,但不接收信息。ShareMode参数是指定别的用户对队列的访问权限,它有以下三种选择:(1)MQ _ DENY _ NONE;(2)MQ _DENY _ RECEIVE _ SHARE;(3)MQ _ ERROR _ SHARING _ VIOLATION。MQ _ DENY _ NONE指定队列能被任何人访问,当Access为MQ _ PEEK _ ACCESS或MQ _ SEND _ ACCESS时,必须设置为这个值;MQ _ DENY _ RECEIVE _ SHARE限制那些能从队列中接收消息到这个进程的用户的使用,假如队列已经被其它进程打开并接收消息,则这个调用失败;MQ _ ERROR _ SHARING _ VIOLATION仅当Access被设置为MQ _ RECEIVE _ ACCESS时才能使用。下面是比较常见的新建和删除队列的例程:

```
<%
Dim Obj _ Queue
Set Obj _ Queue = Server. CreateObject("MSMQ. MSMQQueueInfo")
Obj _ Queue. PathName = ". \ Com1Queue"
Obj _ Queue. Create '在指定路径创建队列
%>
<%
Dim Obj _ Queue
Set Obj _ Queue = Server. CreateObject("MSMQ. MSMQQueueInfo")
Obj _ Queue. PathName = ". \ Com2Queue"
Obj _ Queue. Delete '删除指定路径的队列
%>
```

应当注意到,用MSMQQueueInfo打开一个队列后就返回了MSMQQueue对象。MSMQQueue用来描述一个在MSMQ服务中打开的队列。它提供了类似指针的方式来遍历一个消息队列,在任何时刻它都指向队列中某个特定的位置。当一应用同步读取队列中的消息时,所有的事件调用都将阻塞,直到消息可获取或超时。如果超时则MSMQ返回一个空消息。

3.2 消息的发送和接收

企业的远程办事处向总部领导发送消息,不可避免地要用到MSMQMessage。MSMQMessage提供了大量属性和方法来定义和操纵消息队列。

下列代码用于企业总部领导查看远程某办事处传到服务器队列中的消息:

```
Sub FindMessage()
Dim Obj _ QueueInfo As New MSMQQueueInfo
Dim Obj _ QueueReceive As MSMQQueue
Dim Obj _ QueueMessage As MSMQMessage
Obj _ QueueInfo. PathName = ". \ Com1Queue"
Set Obj _ QueueReceive = Obj _ QueueInfo. Open(MQ _ RECEIVE _ ACCESS,
MQ _ DENY _ NONE)
Do While True
  Set Obj _ QueueMessage = Obj _ Queue. Peek(,,,1000)
  IF Not Obj _ QueueMessage Is Nothing THEN
    '显示消息的标题和内容
    MsgBox Obj _ QueueMessage. Label&"_"&Obj _ QueueMessage. body
    ELSE
      MsgBox"Nothing in the queue"
      Exit Do
    END IF
Loop
MsgBox "The End of Message"
Obj _ QueueReceive. Close '关闭对象
Set Obj _ QueueInfo = Nothing
Set Obj _ QueueReceive = Nothing
Set Obj _ QueueMessage = Nothing
End Sub
```

这段程序用于查看一个队列中的消息,使用Peek方法,主要是设置一个1 000微秒的延时。如果想在读完消息后从消息队列中删除该消息,则需要使用Receive方法。

下面是远程办事处发送消息到总部服务器指定队列的代码:

```
<%
Dim Obj _ QueueInfo
Dim Obj _ QueueSend
Dim Obj _ QueueMessage
Set Obj _ QueueInfo = Server. CreateObject("MSMQ. MSMQQueueInfo")
Qbj _ QueueInfo. PathName = ". \ Com1Queue"
Set Obj _ QueueSend = Obj _ QueueInfo. Open(MQ _ SEND _ ACCESS, MQ _ DE-
NY _ NONE)
Set Obj _ QueueMessage = Server. CreateObject("MSMQ. MSMQMessage")
Obj _ QueueMessage. Label = "Message Label"      '定义消息标题
Obj _ QueueMessage. Body = "Message Body"        '定义消息内容
Obj _ QueueMessage. Send Obj _ QueueSend
Obj _ QueueSend. Close
Set Obj _ QueueInfo = Nothing
Set Obj _ QueueSend = Nothing
Set Obj _ QueueSMessage = Nothing
%>
```

以上介绍了在ASP中对MSMQ的一些重要属性的基本应用,要实现复杂的分布式应用则需要将这些属性和方法有机地结合起来使用。

4 结束语

本文介绍了通过ASP与MSMQ的结合使用,实现了现代大型企业的远程办事处与总部的消息通信。这种实现方法一方面保持了ASP在Web数据库应用方面的强大优势,以及与浏览器无关、源代码保密、高效等特点;另一方面通过MSMQ,可以对消息队列进行各种操作,能够支持分布式应用之间的消息传递,这是传统方法所不能比拟的。无疑,这种方法能够很好地适应当前分布式应用和计算的潮流,是Web开发者可以优先考虑的一种解决方案。

参考文献:

[1] Mickey Williams. Windows 2000 编程技术内幕. 北京:机械工业出版社,1999.

[2] 李刚,王茜. 基于Web访问数据库的实现方案. 计算机工程与应用,2000,(2).

[3] 相关文章. http://topcgi.isme.net.

SOCKS代理服务器的研究与实现

南京航空航天大学计算机科学和工程系　陈 兵 王立松

摘　要　代理服务器是企业Intranet用户访问Internet的桥梁，按照层次可以分为应用层代理、传输层代理和SOCKS代理。在这三种代理中，SOCKS代理提供了更好的安全性和简便性，SOCKS V5支持对客户端进行授权和认证，支持第三方的加密，有效地解决了Intranet和Internet连接时产生的安全控制问题。本文首先介绍了SOCKS的基本原理和特点，然后讨论了SOCKS的关键实现技术，最后给出SOCKS代理的实现流程。

关键词　防火墙，代理服务器，SOCKS，身份认证

代理服务器最基本的功能是连接，此外还包括安全性、缓存、内容过滤、访问控制管理等功能。在代理服务器的众多功能中，安全性是一个突出且敏感的问题。绝大多数企业、部门在使用代理服务器的时候，都会考虑这个问题，把它作为选购代理服务器产品的重要依据。目前市场上流行的代理服务器，如Microsoft Proxy Server、WinGate等，都是国外的产品，虽然功能和性能等方面都不错，但所有技术和版权都是国外的，从保证系统安全性的角度出发，很有必要开发一个拥有自主版权和内核技术的代理服务器。因此研究并实现代理服务器不仅有助于我们深入了解各种Internet协议的实现细节，掌握代理的技术，而且可实现软件的国产化，满足国内用户的安全性需求，对于社会和个人都有积极意义。

根据代理服务器工作的层次，一般可分为应用层代理、传输层代理和SOCKS代理。应用层代理工作在TCP/IP的应用层之上，对于不同的应用层协议需要有不同的代理，如HTTP、FTP、SMTP等；传输层代理工作在传输层，代理服务器能够接收内部网的TCP和UDP包并将其发送到外部网；SOCKS代理是可用的最强大、最灵活的代理标准协议，它允许代理服务器内部的客户端完全地连接到代理服务器外部的服务器，而且它对客户端提供授权和认证，因此SOCKS代理是一种安全性较高的代理。

1 SOCKS简介

SOCKS包括两部分：SOCKS服务器和SOCKS客户端。参照OSI的七层参考协议，SOCKS服务器在OSI的应用层实现，SOCKS客户端在OSI的应用层和传输层之间实现，如图1所示。SOCKS是一种非常强大的电路级网关防火墙，使用SOCKS代理，应用层不需要作任何改变，但是客户端需要专用的程序，即如果一个

作者简介：陈兵，男，1970年出生，南京航空航天大学计算机科学和工程系讲师，研究方向为计算机网络和信息安全。

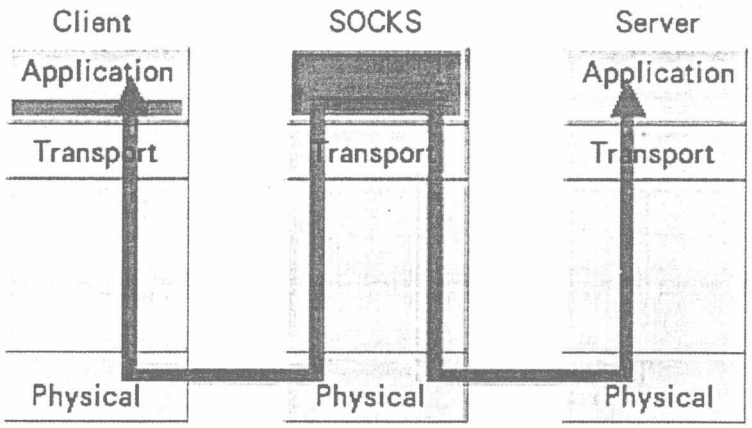

图1 SOCKS工作模型

基于TCP的应用需要通过SOCKS代理进行中继，首先必须将客户端程序SOCKS化(SOCKSified)。

当一个主机需要连接应用程序服务器时，它先通过SOCKS客户端连接到SOCKS代理服务器。这个代理服务器将代表该主机连接应用程序服务器，并在主机和应用程序服务器之间中继数据。对于应用程序服务器，SOCKS代理服务器相当于客户端。

目前SOCKS有两个版本，SOCKS V4和SOCKS V5。SOCKS V4为基于TCP的客户机/服务器应用程序提供了一种不安全的穿越防火墙的机制，包括TELNET，FTP和当前最流行的信息查询协议如HTTP，WAIS和GOPHER。SOCKS V5协议为了包括对UDP的支持而扩展了SOCKS V4，SOCKS V5为了包括对一般环境下更强的认证机制的支持而扩展了协议架构，为了包括对域名和IPv6地址的支持而扩展了地址集。

2 SOCKS的特性

由于SOCKS的简单性和可伸缩性，SOCKS已经广泛地作为标准代理技术应用于内部网络对外部网络的访问控制。SOCKS的主要特性有：

(1) 简便的用户认证和建立通信信道

SOCKS协议在建立每一个TCP或UDP通信信道时，都把用户信息从SOCKS客户端传输到SOCKS服务器进行用户认证，从而保证了TCP或UDP信道的完整性和安全性。

(2) SOCKS与具体应用无关

作为代理软件，SOCKS协议建立通信信道，并为任何应用管理和控制信道。当新的应用出现时，SOCKS不需要任何扩展就可进行代理。

(3) 灵活的访问控制策略

IP路由器在IP层通过IP包的路由来控制网络访问，SOCKS在TCP或UDP层控制TCP或UDP连接。它可以与IP路由器防火墙一起工作，也可以独立工作。SOCKS的访问控制策略可基于用户、应用、时间、源地址和目的地址，加强了控制的灵活性，能更好地控制网络访问。

(4) 支持双向代理

大多数的代理机制(如网络地址解析NAT[1])只支持单向代理，而SOCKS通过域名来确定通信目的地，克服了使用私有IP地址的限制。SOCKS能够使用域名在不同的局域网间建立通信信道。

3 SOCKS V4和SOCKS V5的比较

SOCKS有两种版本：SOCKS V4和SOCKS V5[2]。SOCKS V4是一个较为简单的代理协议，在许多方面还不完善。主要体现在：认证机制不强，不支持UDP代理，客户端需要有一个域名解析过程。SOCKS V5则弥补了上述不足，通过采用以下关键技术增强了SOCKS V4的功能。

(1) 强壮的认证机制技术

SOCKS V5协议有两种认证方法：RFC1929描述的用户名/口令认证[3]和RFC1961定义的普通安全服务应用编程接口[4] (Generic Security Service Application Programming Interface，GSS-API)认证。

(2) 认证方法协商技术

SOCKS V5增加了两个消息，第一个消息由客户端发给SOCKS V5服务器，表明它支持的认证方法。第二个消息是SOCKS V5服务器发给客户端的响应，表明客户端应使用的认证方法。根据配置的安全策略，SOCKS V5服务器决定具体的认证方法。如果客户端支持的认证方法不符合其安全策略，则SOCKS V5服务器将断开本次连接。

(3) 地址解析技术

SOCKS V5内建的地址解析代理，简化了域名解析的管理，其客户端可以将域名代替IP地址

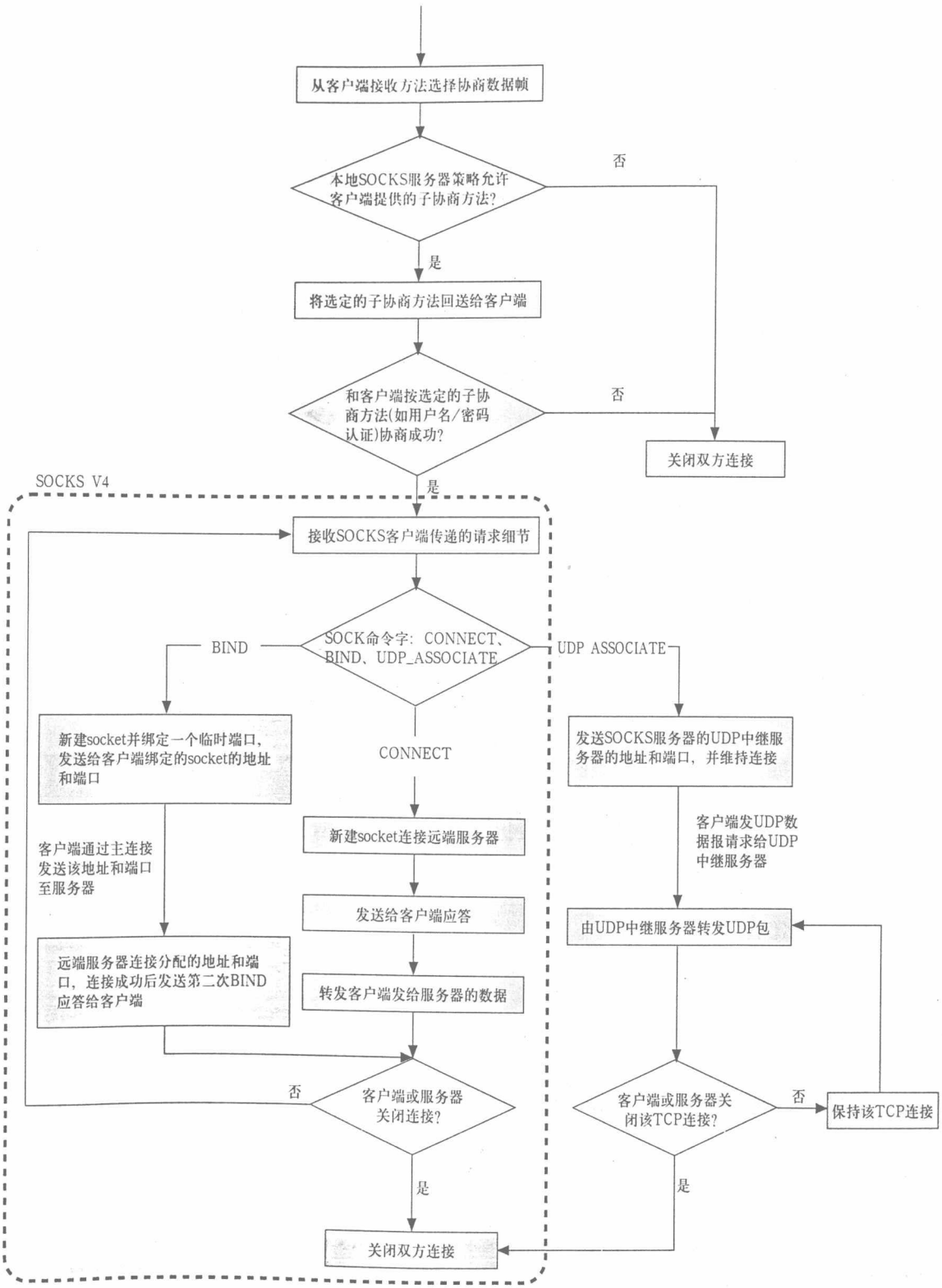

图2 SOCKS工作流程

直接发送给SOCKS V5服务器，由该服务器进行地址解析，并负责与应用程序服务器建立连接。

(4) UDP代理技术

SOCK V5允许创建UDP代理。

综上所述，采用以上关键技术的SOCKS V5，已经成为一个需要认证的防火墙协议。当SOCKS和SSL协议配合使用时，可作为建立高度安全的VPN(Virtual Private Network虚拟专用网)的基础，SOCKS协议的优势在于访问控制，因此适合用于安全性较高的VPN。所以，SOCKS V5的优点非常明显：

(1) SOCKS V5在OSI模型的会话层控制数据流，定义了非常详细的访问控制。而在网络层只能根据源和目的IP地址来允许或拒绝数据包通过，在会话层则可采用更多控制手段。

(2) SOCKS V5在客户机和主机之间建立了一条虚电路，可根据对用户的认证进行监视和访问控制

(3) SOCKS V5能提供非常复杂的方法来保证信息安全传输。用SOCKS V5的代理服务器可以隐藏网络地址结构。如果SOCKS V5同防火墙结合起来使用，数据包经一个惟一的防火墙端口(缺省端口号是1080)到代理服务器，然后代理服务器过滤发往目的计算机的数据，这样可以防止防火墙上存在的漏洞。

(4) SOCKS V5能为认证、加密和密钥管理提供“插件”模块，可让用户很自由地采用他们所需要的技术。SOCKS V5可根据规则过滤数据流，包括Java Applet和Active X控件。

但是，由于SOCKS V5通过代理增加一层安全性，因此其性能比低层协议差。SOCKS V5尽管比网络层和传输层的方案更安全，但需要比低层协议制定更为复杂的安全管理策略。

4 SOCKS代理的实现

实现SOCKS的主要模块包括：

(1) 服务监听进程，处理客户端请求模块；

(2) 认证和协商模块，处理安全问题；

(3) 数据转发模块，实际上是SOCKS V4的功能；

(4) UDP代理模块，SOCKS V5新增的功能。

图2表示了SOCKS的总体工作流程。

SOCKS代理用一个CSOCKS类实现，便于以后的功能扩充。CSOCKS继承了CasyncSocket类。整个流程完全按照SOCKS V5协议规范来设计，在具体实现时，首先实现SOCKS V4的基本功能，使系统能正常运行，然后再增加SOCKS V5的认证和UDP代理功能，完善整个SOCKS代理服务。

5 结束语

随着Internet和企业Intranet的不断发展，网络安全是目前企业连接到Internet最关心的问题。尽管防火墙是网络安全的首选措施，但是现有市场上的防火墙产品一般都是价格昂贵，功能复杂。因此，采用SOCKS V5，复杂的网络安全问题将变得比较简单，并且它支持开放的Internet标准，支持身份认证、流量控制、日志管理、协议过滤和数据库应用等功能。通过这些功能，企业完全可以建立灵活动态的访问策略，从而去除没有价值的网络资源访问，节约大量的资源使用费用。因此，研究并实现SOCKS代理服务具有重要的意义。 (收稿日期：2001.7.29)

参考文献

[1] K. Egevang, P. Francis. The IP Network Address Translator (NAT) RFC1631, 1994.5.1

[2] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones. SOCKS Protocol Version 5 (RFC1928), 1996.4

[3]M.Leech.Username/Password Authentication for SOCKS V5, 1996.3

[4] P. McMahon. GSS-API Authentication Method for SOCKS Version 5 (RFC1961), 1996.6

HTTP 代理缓存的实现

陈　兵，王立松

（南京航空航天大学计算机科学与工程系，江苏 南京 210016）

摘要：随着 Internet/Intranet 的飞速发展，越来越多的用户通过 Intranet 与 Internet 连接，局域网内的众多用户如何能够通过一条 Internet 连接进行快速的网络浏览？这涉及到 Web 代理服务器的核心部分——HTTP 缓存系统。HTTP 缓存系统将用户浏览的网页保存在代理服务器中，一旦有用户浏览相同的网页并且该网页还没有更新，则代理服务器直接将网页传递给用户，无需重新下载，因此，缓存明显加快了浏览速度。本文首先介绍了代理服务器的缓存原理，然后对缓存的具体功能进行了划分，最后给出基于哈希链表和时间链表的缓存实现方法。

关键词：代理服务器；HTTP；Cache；Hash

1　代理服务器缓存的作用

代理服务器（Proxy）[1]是接收或解释客户端连接并发起到服务器的新连接的网络节点。它是客户端/服务器关系的中间人。代理服务器主要用于将企业网（Intranet）连接到 Internet，它允许内部客户端使用常用的应用程序如 Web 浏览器和 FTP 客户端访问 Internet。而代理服务器使用单个合法 IP 地址处理所有的发出请求，因此无论客户端是否具有合法 IP 地址都允许访问 Internet。

随着 Internet 通信量的持续增加，带宽越来越珍贵。对于向用户提供 Internet 服务的组织，如果按照使用收费，网络通信就很昂贵。减少通信的方法之一是在 LAN 中配置代理服务器并设置缓存。

被动或按需缓存是代理服务器用于保存远端服务器 Web 页的页面机制，以便能直接从缓存中读取，加快响应。缓存的内容最终会过时，Web 页面一旦过期，返回给用户前必须从 Internet 上重新下载。客户端请求 Web 页面之后才读入缓存，这样有效地利用了网络带宽，但增加了延时。

主动缓存帮助减少人们在刷新 Web 页面时的等待时间。代理服务器观察缓存中的文件，如果文件到期，就会请求 Web 服务器获得最新文件。这样，缓存会接收文件的更新版本，使得用户不必等待文件更新，因为代理服务器能在非高峰时间主动更新。用户等待页面时间大大减少，但网络通信量实际增加了，因为代理服务器不等待用户请求就重新请求页面，如果页面不再使用，就浪费了时间和带宽。

有的代理服务器允许管理员指定 Web 页面按照一定的规律更新，也可能支持批处理更新。

总而言之，主动缓存只在希望减少返回 Web 页面的时间时有用。关心网络带宽的用户不会使用这项功能，但是，被动缓存将重新获取 Web 的工作放到非高峰时间，从而减少了高峰时间的通信量。

2　HTTP/1.0 常用的消息头

由于 HTTP 的 Cache 算法主要根据 HTTP 的消息头来判断 Cache 是否过期等信息，所以首先需要了解 HTTP 消息头的格式。

HTTP 消息头是 HTTP 客户端和服务器文本命令和响应的基础。这种消息格式很灵活，可以用于客户端到服务器的命令，也可以用于服务器到客户端的响应。RFC2068 定义了 HTTP 消息头的格式，这种格式可以表示为：

Message - type: Arguments[CR/LF]

HTTP 协议的消息头由几个字段组成，第一个字段 Message - type 是“消息类型”，告诉接收进程如何解释后面的信息。第二个字段 Arguments 是消息使用的数据，最后是回车换行。下面举例说明客户端和服务器的对话：

< client request > GET /HTTP/1.0

Connection: Keep - Alive

User - Agent: Mozilla/4.04[en](WinNT;U)

Host: www.company.com

Accept: image/gif/, image/x - xbitmap, image/jpeg, image/pipeg.