文章编号:1001 - 9081(2010)11 - 2952 - 04

关联规则挖掘中 Apriori 算法的研究与改进

崔贯勋,李　梁,王柯柯,苟光磊,邹　航
(重庆理工大学 计算机科学与工程学院,重庆 400054)
(cgxy@vip.qq.com)

摘　要:经典的产生频繁项目集的 Apriori 算法存在多次扫描数据库可能产生大量候选及反复对候选项集和事务进行模式匹配的缺陷,导致了算法的效率较低。为此,对 Apriori 算法进行以下 3 方面的改进:改进由 k 阶频繁项集生成 $k+1$ 阶候选频繁项集时的连接和剪枝策略;改进对事务的处理方式,减少 Apriori 算法中的模式匹配所需的时间开销;改进首次对数据库的处理方法,使得整个算法只扫描一次数据库,并由此提出了改进算法。实验结果表明,改进算法在性能上得到了明显提高。

关键词:数据挖掘;关联规则;Apriori 算法;频繁项集;候选项集
中图分类号: TP311.13　　**文献标志码**:A

Research and improvement on Apriori algorithm of association rule mining

CUI Guan-xun, LI Liang, WANG Ke-ke, GOU Guang-lei, ZOU Hang
(*School of Computer Science and Engineering, Chongqing University of Technology, Chongqing* 400054, *China*)

Abstract: The classic Apriori algorithm for discovering frequent itemsets scans the database many times and the pattern matching between candidate itemsets and transactions is used repeatedly, so a large number of candidate itemsets were produced, which results in low efficiency of the algorithm. The improved Apriori algorithm improved it from three aspects: firstly, the strategy of the join step and the prune step was improved when candidate frequent $(k+1)$-itemsets were generated from frequent k-itemsets; secondly, the method of dealing with transaction was improved to reduce the time of pattern matching to be used in the Apriori algorithm; in the end, the method of dealing with database was improved, which lead to only once scanning of the database during the whole course of the algorithm. According to these improvements, an improved algorithm was introduced. The efficiency of Apriori algorithm got improvement both in time and in space. The experimental results of the improved algorithm show that the improved algorithm is more efficient than the original.

Key words: data mining; association rule; Apriori algorithm; frequent itemsets; candidate item set

0　引言

关联规则挖掘作为数据挖掘的重要研究内容之一,主要研究事务数据库、关系数据库和其他信息存储中的大量数据项之间隐藏的、有趣的规律。1993 年,美国著名学者 R. Agrawal 等人首次提出了挖掘布尔关联规则,之后提出了著名的基于频繁项集的 Apriori[1] 算法。关联规则挖掘最初仅限于事务数据库的布尔型关联规则,近年来广泛应用于关系数据库,因此,积极开展在关系数据库中挖掘关联规则[2] 的相关研究具有重要的意义。近年来,已经有很多基于 Apriori 算法的改进和优化[3]:

文献[4]提出杂凑表技术,它根据 C_k 确定 C'_{k+1},并用规模适当的 hash 存放 C'_{k+1}。在第 k 遍扫描数据库时,同时统计 C_k 和 hash 表中的 C'_{k+1} 项目。在求出 L_k 的同时,hash 表中 C'_{k+1} 的计数可用于进一步剪裁 C_{k+1}。文献[5 - 6] 均提出减少扫描数据库事务记录的方法,如果一个事务记录中不包含长度为 k 的频繁模式,则不可能包含长度为 $k+1$ 的频繁模式,因此可在以后的扫描中剔除。文献[7]提出 Partition 算法,它将数据库分割为若干个可调入内存的子库,分别求出各个子库的局部频繁模式,所有局部频繁模式的并集为全局频繁模式的候选集。最后一遍扫描数据库可最终求出全局频繁模式集。Partition 考虑的候选集比 Apriori 还要多,有可能加剧组合爆炸的问题。文献[8]提出抽样法,它从数据库中随机抽取一个可调入内存的子集,采用一个略低的支持率阈值,求出该子集中的局部频繁模式。第二遍扫描数据库,求出局部频繁模式的全局支持率。该文还提出了确保全局频繁模式不被遗漏的机制。文献[9]提出动态模式计数法 DIC,它在同一遍数据库描过程中分段增加候选频繁模式集。DIC 在确定一个模式的所有子集都是频繁模式集后就开始其支持率的统计,而不是等到下一轮数据库扫描。文献[10]提出了基于压缩事务矩阵相乘的改进算法,该算法扫描数据库中的数据并将数据信息映射到项目事务二进制矩阵,由项目事务二进制矩阵与相应的辅助矩阵相乘得到频繁 1-项目集,依此类推得到其他频繁项目集。该方法由于矩阵相乘花费了较多的时间,因此算法的效率只是得到了一定程度的提高。文献[11]提出了基于十字链表的改进算法,该算法将事务数据库中的信息用十字链表表示,把对数据库的扫描转变为对内存中十字链表的扫描,利用十字链表缩短需要匹配的事务长度,但该算法仍

收稿日期:2010 - 05 - 17;**修回日期**:2010 - 07 - 17。　**基金项目**:教育部科学研究项目(09yjc870032);重庆市科技攻关计划项目(CSTC2008AC2126;CSTC2009AC2034);重庆市自然科学基金资助项目(CSTC2008BB2065);重庆理工大学科研青年基金资助项目(2010ZQ22)。
作者简介:崔贯勋(1978 -),男,河南鄢陵人,实验师,硕士,主要研究方向:数据库;　李梁(1964 -),男,重庆人,副教授,主要研究方向:软件工程;　王柯柯(1977 -),女,四川南充人,讲师,硕士,主要研究方向:软件工程;　苟光磊(1980 -),男,重庆人,实验师,硕士,主要研究方向:人工智能;　邹航(1979 -),男,重庆人,实验师,硕士,主要研究方向:数据挖掘。

然需要进行模式匹配,也使得算法的提高程度受到限制。

尽管 Apriori 算法有如上诸多改进方法,但时间效率还不尽理想,为了更进一步提高算法的效率,提出了基于集合的改进 Apriori 算法,目的就是进一步提高算法的性能。

1　Apriori 算法的基本原理

Apriori 是最有影响的挖掘布尔关联规则频繁项目集的经典算法。

在 Apriori 算法中,通过遍历数据库得到大一项集 L_1。如果 L_1 非空,由 L_1 产生长度为 2 的候选项集合 C_2,然后对事务数据库中的每一个事务 t,求出 t 在 C_2 中的全部子集 C_t,对于 C_t 中的每一个长度为 2 的候选项集 c,令 c 的计数加 1。当扫描事务数据库一遍后,筛选出候选项集合 C_2 中所有计数满足最小支持度的项集组成了长度为 2 的频繁项集合。用以上步骤重复处理新得到的频繁项集合,直到没有频繁项集合产生。其中候选项集产生的过程被分为连接与剪技两个部分。采用这种方式,使得所有的频繁项集既不会遗漏又不会重复。为提高频繁项集逐层产生的效率,Apriori 算法利用了两个重要的性质用于压缩搜索空间。

性质 1　k 维数据项目集 X 是频繁项目集的必要条件是它的所有 $k-1$ 维子集均是频繁项目集。

性质 2　若 k 维项目集 X 中有一 $(k-1)$ 维子集不是频繁项集,则 X 不是频繁项集。

2　Apriori 算法的改进

Apriori 算法的优点是结构简单,易于理解,没有复杂的推导。另外,算法应用性质 1 和性质 2 而设计的候选产生——检查方法在许多情况下大大缩小了需要检查的候选规模,使算法效率大幅度提高。但 Apriori 算法依然存在 3 个主要的问题:

1)多次扫描数据库。Apriori 算法需要在每进行一次迭代的时候扫描一次数据库,当挖掘出的最大频繁项集的长度为 N 时,需要扫描 N 次数据库,而在实际应用中经常需要挖掘很长的模式,多次扫描数据库带来的开销非常大。

2)可能产生大量候选。Apriori 算法在迭代过程中要在内存中产生、处理和保存候选频繁项集,这个数量有时候是非常巨大的,导致算法在广度和深度上的适应性很差。

3)在扫描数据库时需要对候选项集和事务进行模式匹配,花费大量的时间。

总之,Apriori 算法有多次扫描数据库、可能产生大量的候选频繁项集及进行大量的模式匹配计算的严重缺陷,使得算法花费在 I/O 上的时间很多,从而导致算法的效率非常低。因此,为了提高 Apriori 算法的效率,需要对算法的上述缺陷进行改进。

2.1　相关定义和定理[12]20-21,[13]190,[14]342

定义 1　设 $x \subseteq t, t \in D$,则称事务 t 支持 x。

定义 2　支持 x 的事务的 ID 号的集合称为 T_x,若 $x \in L_k$,则称支持 x 的事务的 ID 号的集合称为 T_x^l,即 $T_x^l = \{t_{id} \mid t \in D \wedge x \subseteq t \wedge x \in L_k\}$,集合 T_x^l 中事务的数量记为 $|T_x^l|$。

定义 3　设 $x \in L_k, y \in L_k$,若 $x_1 = y_1 \wedge x_2 = y_2 \wedge \cdots \wedge x_{k-1} = y_{k-1} \wedge x_k < y_k$,则称 $x_1x_2\cdots x_{k-1}x_ky_k$ 为 x 与 y 的连接,记做 $x \infty y$。

定义 4　最小支持度与事务数据库中事务的数量的乘积称为最小支持事务数。

定理 1　设 $x \in L_k, y \in L_k$,若 $x_1 = y_1 \wedge x_2 = y_2 \wedge \cdots \wedge x_{k-1} = y_{k-1} \wedge x_k < y_k$,则 $T_{x\infty y}^l = T_x^l \cap T_y^l$。

证明　对 $\forall t \in T_{x\infty y}^l$,由定义 1 可知,$x \infty y \subseteq t$,由定义 3 可知,$x \subseteq t$ 且 $y \subseteq t$,由定义 2 可知,$t \in T_x^l$ 且 $t \in T_y^l$,即 $t \in T_x^l \cap T_y^l$。

反之,对 $\forall t \in T_x^l \cap T_y^l$,则 $t \in T_x^l$ 且 $t \in T_y^l$,由定义 2 可知,$x \subseteq t$ 且 $y \subseteq t$,由定义 3 可知,$x \infty y \subseteq t$ 由定义 1 可知,$\forall t \in T_{x\infty y}^l$。

所以 $T_{x\infty y}^l = T_x^l \cap T_y^l$。　证毕。

定理 2　若 k 维数据项目集 $X = \{x_1, x_2, \cdots, x_k\}$,$X$ 中存在一个项目 $p \in X$,使得 $|L_{k-1}(p)| < k-1$,则 X 不是频繁项目集。其中,$|L_{k-1}(p)|$ 表示 $(k-1)$ 维频繁项目集的集合 L_{k-1} 中包含 p 的频繁项目集的个数。

证明　用反证法证明。假设 X 是 k 维频繁项目集,由性质 1 可知,它的 k 个 $k-1$ 维子集均在 L_{k-1} 中。而由 X 生成的 k 个 $k-1$ 维子集中,每一个项目 $p \in X$ 均出现 $k-1$ 次,故 $\forall p \in X$,均有 $|L_{k-1}(p)| \geqslant k-1$,这与条件矛盾,故 x 不是频繁项目集。　证毕。

推论 1　若 L_{k-1} 中有元素 e 包含项目 p,使得 $|L_{k-1}(p)| < k-1$,则 L_{k-1} 中所有不同于 e 的元素与 e 连接而成的候选 k 维项目集均不可能是频繁项目集。

证明　假设由 e 生成的一个候选 k 维数据项目集 x 是频繁项目集,则 e 显然是 x 的一个 $k-1$ 维子集,由于 $p \in e$,因此 $p \in X$,由定理 2 可知 $|L_{k-1}(p)| \geqslant k-1$,与条件矛盾。证毕。

该推论应用于修剪频繁项集方法中将大大减少寻找候选频繁项集的时间开销,示例如下:

例 1　假设 $L_3 = \{\{a,b,c\},\{a,b,d\},\{a,b,e\},\{a,b,f\},\{a,b,g\},\{b,c,d\},\{b,c,e\},\{b,c,f\},\{b,c,g\}\}$,求 C'_4。

由定理 2 可知:$|L(a)| = 5, |L(b)| = 9, |L(c)| = 5, |L(d)| = 2, |L(e)| = 2, |L(f)| = 2, |L(g)| = 2$,由于 $|L(d)| = |L(e)| = |L(f)| = |L(g)| = 2 < 3$,由推论 1 可知,与项目 d、e、f、g 连接生成的 $k(k \geqslant 4)$ 维项目集均不是频繁项目集,故在连接之前就去掉 L_3 中包含项目 d、e、f、g 的频繁项目集 $L'_3 = \{\{a,b,c\}\}$。从而 C'_4 为空集,显然 L_4 也是空集。

如果用 Apriori 算法计算,首先要生成候选集 $C_4 = \{\{a,b,c,d\},\{a,b,c,e\},\{a,b,c,f\},\{a,b,c,g\},\{a,b,d,e\},\{a,b,d,f\},\{a,b,d,g\},\{a,b,e,f\},\{a,b,e,g\},\{a,b,f,g\},\{b,c,d,e\},\{b,c,d,f\},\{b,c,d,g\},\{b,c,e,f\},\{b,c,e,g\},\{b,c,f,g\}\}$,共计 16 个候选频繁项集,然后取出每个候选频繁项集的子集,对每个子集查找其是否在 L_3 中,最后用性质 2 过滤掉非频繁候选项目集,最终还是得到 C'_4 为空集。

2.2　改进的 Apriori 算法

在 Apriori 算法中对数据库的处理是将数据库看做是水平结构,如图 1 所示,本文借鉴文献[15-16]的思想采用新的数据结构,将其看做是项目事务垂直对应关系数据结构,如图 2 所示。

Tid	Items
101	*ABC*
102	*ABD*
103	*ABCE*
104	*CE*

图 1　原事务数据库

A	*B*	*C*	*D*	*E*
101	101	101	102	103
102	102	103		104
103	103	104		

图 2　项目事务库

经过这样变换很容易计算支持 k- 项目的事务数:已知支

持项目 A 和 B 的事务的集合为 T_{AB}、支持项目 C 的事务的集合为 T_C，则由定理1可知，同时支持项目 A、B、C 的事务为两个集合的交集，即：

$$T_{ABC} = T_{AB} \cap T_C$$

算法改进的思路如下：

1）首先扫描源数据库，扫描过程中记录支持每个项的事务代码，然后，统计支持每个项的事务数，删除支持事务数小于最小支持事务数的项，进而得出频繁1-项集。

2）对频繁1-项集中的项两两连接得出候选2-项集，然后通过计算支持候选2-项集中各项集的两个项的事务的交集，得出支持各项集的事务集，删除支持事务数小于最小支持事务数的项集，得出频繁2-项集。

3）以此类推，对频繁 $(k-1)$-项集利用定理2和推论1去掉不可能为频繁 k-项集的项集，再利用定义3两两连接得出候选 k-项集，然后通过计算支持候选 k-项集中各 $(k-1)$-频繁项集的事务的交集（设 $x \in L_{k-1}, y \in L_{k-1}$，则 $x \infty y \in C_k$，即计算 $T_{x\infty y} = T_x \cap Y_y$），得出支持各 k-项集的事务集，删除支持事务数小于最小支持事务数的项集，得出频繁 k-项集。

4）重复操作3），直到不再有频繁项集产生。

算法对文献[12]的改进之处在于在生成候选 k-项集之前先利用定理2和推论1过滤掉 L_{k-1} 中不可能生成频繁 k-项集中的项目，这样做比 Apriori 算法直接用性质2来过滤在时间上的开销要小得多，也就大大减少了生成频繁 k-项集的时间。对文献[13-14]的改进之处在于生成频繁 k-项集时记录下来每一频繁项集由哪些事务支持（只记录事务的 ID 号，而不是整个事务都记录下来）。假设 $x \in L_k, y \in L_k, x = x_1 \infty x_2 \infty \cdots \infty x_{k-1} \infty x_k, y = y_1 \infty y_2 \infty \cdots \infty y_{k-1} \infty y_k$，若 $x_1 = y_1 \wedge x_2 = y_2 \wedge \cdots \wedge x_{k-1} = y_{k-1} \wedge x_k < y_k$，在判断 $x \infty y$ 是否为频繁 $(k+1)$-项集时就不需要重新扫描数据库，只需找出 T_x 和 T_y 的交集，由定理1可知，T_x 和 T_y 的交集中的事务支持 $x \infty y$，如果交集中事务的数量大于或等于最小支持事务数，则它就是频繁项集，否则，它就不是频繁项集。这样做可以避免 Apriori 算法中的模式匹配造成时间开销非常大的问题。

改进后的 IApriori 算法描述如下：

输入　事务数据库 D；最小支持度阈值 min_sup。

输出　D 中的频繁项集 L。

```
L1 = find_L1(D, min_sup);
for(k = 2; L_{k-1} ≠ ∅; k++){
L'_{k-1} = L_{k-1};
// 统计每个项目在 L_{k-1}，中出现的次数，并记录支持该项
// 目的频繁项集
For each item x ∈ L_{k-1}
  For each field f ∈ x
  { f.count++;
    I_f[f.count] = x;}
// 根据定理2及推论1，删除 L'_{k-1} 中不可能生成频繁 k 项
// 集的项
Delete(L'_{k-1}, I_f | f.count < k - 1);
For each item x ∈ L'_{k-1}
For each item y ∈ L'_{k-1}
// 找出可以连接的频繁项
if (x1 = y1 ∧ x2 = y2 ∧ ⋯ ∧ x_{k-2} = y_{k-2} ∧ x_{k-1} < y_{k-1})
{ n = 0;
// 事务有先后顺序的特点便于快速找出两个集合中相同
// 的事务的数量
For (i = 1, j = 1; i ≤ | T_x |, j ≤ | T_y |;)
if (T_x[i] < T_y[j])
  i++;
else if (T_x[i] > T_y[j])
      j++;
    else
    { n++;
      T_{x∞y} = T_{x∞y} ∪ {T_x[i]}
      i++;
      j++;
    }
if(n ≥ N × min_sup)                    // 找到一个频繁项集
{ T^l_{x∞y} = T_{x∞y};
  L_k = L_k ∪ {x∞y}
}
return L = ⋃_k L_k;
// 生成1-项频繁项集
Procedure find_L1(D, min_sup);
C1 = get_item(D)                       // 取出事务数据库的项目
For each transaction t ∈ D
{ N++;                                 // 统计数据库中事务的数量
  For each item x ∈ t
  { x.count++;                         // 事务在数据库中有顺序
    T_x[x.count] = t_id;               // 集合中也按顺序排列
  }
}
L1 = {x | x ∈ C1 ∧ x.count ≥ N × min_sup}
Return L1
```

3 算法分析与测试

与经典的 Apriori 算法相比，改进后的算法有以下优点：

1）整个算法只对数据库扫描一次，在后续过程中也避免了对不必要的事务和项目的处理，并且随着算法的运行，需要处理的事务的规模也在不断减小，这些优点都是原始的 Apriori 算法所不具备的。

2）算法使用的数据结构比较简单，主要使用了集合，并且对集合的操作也比较节省时间，避免了其他改进算法中构造复杂的数据结构花费较多的时间[11]38，也避免了某些算法对相应的数据结构操作起来也比较费时，如文献[10]对矩阵进行相乘运算就比较费时。

3）该算法只在开始扫描数据库时需要处理每个事务的项目，在后面的过程中只需要对事务的事务代码进行匹配，而不需要对事务的每个项目进行匹配。设已知支持项目 A 和 B 的事务的集合为 T_{AB}、支持项目 C 的事务的集合为 T_C，则同时支持项目 A、B、C 的事务为两个集合交集（交集由两个集合中相同的事务代码组成，只对事务代码进行匹配，不再对事务中的项目进行匹配）。另外，事务有一个特点就是有时间先后顺序，这里以事务代码的大小为依据，这就使得寻找两个集合的交集时无需循环扫描集合，而只顺序扫描一次两个集合即可找出两个集合的交集，详细过程见算法的描述。正是避免了对事务的项目进行大量的模式匹配计算和循环扫描集合，使得算法的时间效率大大提高。

4）利用定理2、推论1和性质2过滤 k-维候选频繁项集时的计算量为：

$$|N_{k-1}| \times |L_{k-1}| + |C'_k| \times |L'_{k-1}| \times (k-1)$$

而 Apriori 算法直接用性质2的计算量为：

$$|C_k| \times |L_{k-1}| \times (k-1)$$

一般情况下,前者花费的时间比后者少得多。其中:L'_{k-1} 和 C'_k 分别表示 L_{k-1} 经过定理2和推论1过滤后的$(k-1)$-维频繁项集和由 L'_{k-1} 生成的 k-项候选项集,N_{k-1} 为应用定理2和推论1过滤掉的项目。

为了测试改进后的算法的效率,用VC++6.0分别实现了 Apriori、PM-Apriori[10]58、Apriori'[11]38、GE-Apriori[13]192 和 IApriori 共5个算法,并在内存为1 GB,CPU 主频为2930 MHz,操作系统为 Windows XP SP3 的计算机上进行了实验,数据库则是文献[17]中提供的蘑菇数据(mushroom database),该数据库共有8 124条记录、22个属性,实验结果如图3所示,从图中可以看出,经过优化的算法在速度上有了较大的提高。

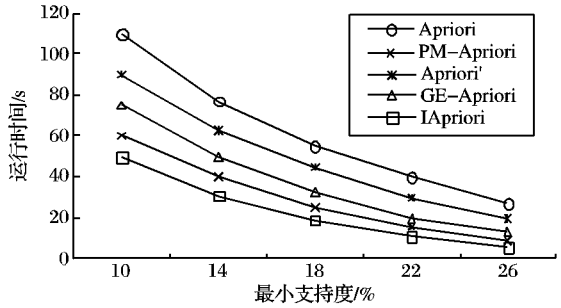

图3　不同算法在最小支持度下的运行时间

4　结语

本文通过分析 Apriori 算法及其优化算法的优劣,提出了一个改进的 IApriori 算法,在关联规则挖掘的发现频繁项集的步骤中进行了优化,并给出了优化算法的完整描述,通过利用蘑菇数据库进行实验分析评价,证明了改进后算法的有效性。

参考文献:

[1] AGRWAL R, SRIKAN R. Fast algorithms for mining association rules in large databases [C]// Proceedings of the 20th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1994: 487 – 499.

[2] 何军,刘红岩,杜小勇.多关系关联规则挖掘研究综述[J].软件学报,2007,18(11):2752 – 2765.

[3] 刘君强,孙晓莹,潘云鹤.关联规则挖掘技术研究的新进展[J].计算机科学,2004,31(1):110 – 113.

[4] PARK J S, CHEN M S, YU P S. An effective Hash based algorithm for mining association rules [C]// Proceedings of International Conference on the Special Interest Group on Management of Data. New York: ACM, 1995: 175 – 186.

[5] 尤磊,兰洋,熊炎.一种基于关系代数的 Apriori 优化方法[J].信阳师范学院学报:自然科学版,2010,23(1):156 – 160.

[6] HAN J, FU Y. Discovery of multiple-level association rules from large databases [C]// Proceedings of the 20th International Conference on Very Large Database. Zuich, Switzerland: [s. n.], 1995: 420 – 431.

[7] SAVASERE A, OMIECINSKI E, NAVATHE S. An efficient algorithm for mining association rules in large databases [C]// Proceedings of the 21st International Conference on Very Large Database. New York: ACM, 1995: 432 – 443.

[8] TOLVONEN H . Sampling large databases for association rules [C]// Proceedings of the 22nd International Conference on Very Large Database. Bombay, India [s. n.], 1996: 134 – 145.

[9] BRIN S. Dynamic itemset counting and implication rules for market basket analysis [C]// Proceedings of International Conference on the Special Interest Group on Management of Data. New York ACM, 1997: 255 – 264.

[10] 杨志刚,何月顺.基于压缩事务矩阵相乘的 Apriori 改进算法[J].中国新技术新产品,2010,30(6):57 – 58.

[11] 黄建明,赵文静,王星星.基于十字链表的 Apriori 改进算法[J].计算机工程,2009,35(2):37 – 38.

[12] 王伟勤,郑海.Apriori 算法的进一步改进[J].计算机与数字工程,2009,37(4):20 – 23.

[13] 徐章艳,刘美玲,张师超,等.Apriori 算法的三种优化方法[J].计算机工程与应用,2004,40(36):190 – 192.

[14] 陈应霞,陈艳.关联规则中的 Apriori 挖掘算法改进[J].长江大学学报:自然科学版,2008,5(4):341 – 343.

[15] 曾万聪,周绪波,戴勃,等.关联规则挖掘的矩阵其法[J].计算机工程,2006,32(02):45 – 47.

[16] 李云峰,陈建文,程代杰.关联规则挖掘的研究及对 Apriori 算法的改进[J].计算机工程与科学,2002,24(6):65 – 68.

[17] SCHLIMMER J. Mushroom data set[DB/OL]. [2010 – 04 – 30]. http://archive. ics. uci. edu/ml/machine-learning-databases/mushroom/agaricus-lepiota. data.

(上接第2944页)

对于 Schema 中的 include、redefine 和 import 也没有进行讨论和实现,接下来的工作就是完善映射模型,实现与包装器的连接以及处理包装器传递过来的查询 XML 文档数据的命令。

参考文献:

[1] MANI M, LEE D, MUNTZ R R. Semantic data modeling using XML schemas[C]// Proceedings of the 20th International Conference on Conceptual Modeling, LNCS 2224. Berlin: Springer-Verlag, 2001: 149 – 163.

[2] LEE D, CHU W W. Comparative analysis of six XML schema languages [J]. ACM SIGMOD Record, 2000, 29(3): 76 – 87.

[3] LEE D, MANI M, MURATA M. Reasoning about XML schema languages using formal language theory, RJ# 10197, Log# 95071 [R]. [S.l.]: IBM Almaden Research Center, 2002.

[4] MANI M, LEE D. XML to relational conversion using theory of regular tree grammars [C]// Proceedings of the VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb on Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web-Revised Papers, LNCS 2590. Berlin: Springer-Verlag, 2002: 81 – 103.

[5] SUN HONGWEI, ZHANG SHUSHENG, ZHOU JINGTAO, *et al*. Constraints-preserving mapping algorithm from XML-schema to relational schema [C]// Proceedings of the First International Conference on Engineering and Deployment of Cooperative Information Systems, LNCS 2480. Berlin: Springer-Verlag, 2002: 193 – 207.

[6] SUN HONGWEI, ZHANG SHUSHENG, ZHOU JINGTAO, *et al*. Mapping XML schema to relational schema [C]// EurAsia-ICT 2002: Information and Communication Technology, LNCS 2510. Berlin: Springer-Verlag, 2002: 322 – 329.

[7] 苏宝程.从 XML 模式到关系模式的映射与规范化设计[J].内蒙古煤炭经济:工作研究与理论探讨,2006(9):29 – 33.

[8] 黄根平,郭绍忠,陈海勇,等.数据集成中 XML 模式和关系模式映射模型研究[J].信息工程大学学报,2009,10(4):527 – 531.

[9] WALMSLEY P. XML 模式权威教程[M].陈维安,乔安平,英宇,译.北京:清华大学出版社,2003:9.