OLAP中基于FP-增长的关联规则挖掘*)

阎星娥 鞠时光 蔡 涛 辛 燕
(江苏大学 镇江212013)

摘 要 关联规则挖掘是一种发现属性间关系的方法,主要用于在商务事务记录中挖掘事务间关系。本文将已经广泛使用的FP-增长(frequent-pattern growth,频繁模式增长)算法进行改进,实现了OLAP中的关联规则挖掘。改进算法分别针对单维、多维、混合维三种关联规则,将多维立方体转化成不同的关系表,通过关系表产生关联规则,并利用立方体中的事实值作为进一步约束,生成了更有价值的规则。

关键词 数据挖掘,OLAP,关联规则挖掘,FP-增长

Mining of Association Rules on the Basis of FP-Growth in OLAP

YAN Xing-E JU Shi-Guang CAI Tao XIN Yan
(Jiangsu University, Zhengjiang 212013)

Abstract Mining of association rules is a method to find the relation among the attributes. It is mainly used to find the relations of transactions in the business transaction records. This paper realizes the mining of association rules in OLAP by improving the FP-Growth algorithm which is widely used. The improved algorithm converses the cube into the different relation tables according to the types of association rules, intra dimensional rules, inter dimensional rules and hybrid dimensional rules. And it generates association rules from the relation tables. This paper also introduces the method of generating more interesting rules constrained by the factual value in the cube.

Keywords Data mining, OLAP, Mining of association rules, FP-Growth

基于数据仓库的OLAP已经被证明是一种快捷、有效的多维分析工具[1],其中集成的、连贯的、清洁的数据以及勘探式的分析方法也必将使其成为数据挖掘的主要平台。

目前已有不少学者研究如何在OLAP中进行数据挖掘。Jiawei Han[2,3]介绍了通用的系统框架,在多维立方体的基础上同时创建OLAP引擎和数据挖掘引擎;Xiaohua Hu[4]提出用于Web挖掘和商务智能系统中的OLAM框架,通过用户识别、会话识别获得点击流数据,在此基础上根据一定的主题创建点击流数据集市,然后运用关联分析、分类预测以及聚类等挖掘方法识别出有意义的模式;Carem C. Fabris[5]将偏差检测这种数据挖掘方法融合到OLAP范例中以提高OLAP和数据挖掘结合效果;Zhu Hua[6]则主要研究在OLAP中进行数据挖掘的方法,将原先基于关系数据库或者事务数据库的算法进行改进,使其适用于OLAP环境。

关联规则挖掘寻找给定数据集中的频繁项,通过频繁项生成强关联规则:R. Agrawal等[7]开发的Apriori算法,它通过逐层搜索迭代的方法,先产生候选集,在候选集中寻找频繁项集。Maurice Houtsma等[8]对Apriori算法进行改进,通过基于散列的技术来压缩候选集。尽管有所改进,但候选项集的依然庞大,并且重复扫描数据库的开销也难以承受,因此文[9]提出了挖掘全部频繁项而不产生候选的方法。以上三种方法均属于事务数据库中的应用,目前也有研究将Apriori算法进行改进[6],实现了OLAP中的数据挖掘,但由于Apriori算法本身效率不高,并且在改进过程中,没有结合OLAP自身的特点,决定了最终的效果并不理想。

针对以上原因,本文在算法研究过程中,极力弥补现有研究的不足,从原型方法选择、改进过程中结合OLAP自身特点等方面加以改进,还考虑到OLAP是基于一定主题设计的,因此在规则生成过程中借助主题中的事实值,介绍基于一定约束的关联规则算法。

1 相关知识

FP-增长算法用于在商务事务中发现各个事务项之间的关联。因此,在OLAP中进行关联规则挖掘首要解决的问题是了解数据环境的不同,针对环境特点对算法进行适当的改进。

OLAP由一系列数据立方体组成,对于一个n维的立方体,其中每个维包含a_n+1行(a_n为维n中所包含的值的个数),前面a_n行存储每个维度值对应的事实值,最后一行存储前面行的聚合值。因此,一个n维数据立方体可以用一个n+1个字段的表表示,立方体中的每个单元同表的一个元组对应。例如:一个3维、每个维有2个值的立方体就可以同一个4列27行的表对应。在本系统中,立方体均由对应的立方体表表示,在立方体表上进行挖掘。

基于OLAP的关联规则挖掘就是从数据立方体结构中挖掘关联规则。根据规则的种类可以将挖掘任务分为3种:单维、多维、混合维挖掘,对每种规则OLAP对其的支持也不同,因此改进的FP-增长算法将分别针对这三种关联规则介绍不同的挖掘方法。

2 FP-增长在OLAP中单维规则上的应用

单维关联是指一个维度内的关联,通常我们称这个维为

*)本课题得到国家863计划重点课题(2002AA412020)及江苏省自然科学基金(No.BK200204)的资助。

项维，区分项维的维度称为事务维，也就是说在单维关联中涉及到两个维，因此在OLAP中，生成2维立方体进行挖掘。

2.1 基于关系表结构的改进

我们借助在关系表中进行单维关联规则挖掘的想法，将OLAP中与任务相关的2维立方体表示成关系表的格式，见表1。采用关系表结构，将待挖掘维中的所有项作为表中的一个字段存在，采用这种表结构，除频繁1-项集的计算有所不同外，算法基本不变。原算法计算频繁1-项集时，对每个项，若项维中存在此项，则此项的支持度加1。如今每个项作为一个字段存在，因此项的支持度计算有所不同，若记录中字段对应的值为0，则说明字段对应的项未出现在记录对应的事务中，若大于0，则说明项出现在事务中，项的支持度加1。

获得频繁1-项集算法如图1，判断条件变为表的单元值是否大于0：

```
GetFrequentItem(decTable,minSupport)
1. for(i=0;i<decTable. Rows. Count;i++)//计算每个项的支持度
2.    for(j=1;jexdecTable. Columns. Count;j++)
3.       if(decTable. Rows[i][j]>0)//如果事务中对应项的值大于0
4.          L[j]. FrequenTimes++;//项集中对应项的支持度加1
5. for(i=0;i<L. Count-1;i++)//将项集中的项进行排序，按支持度递减
6.    if(L[i]. FrequenTimes>minSupport)//大力支持度的项进行排序
7.       for(j=i+1;j<L. Count;j++)
8.          if(L[j]. FrequenTime<minSupport)//若小于支持度，除去
9.             L. RemoveAt(j);
11.         else 调节顺序
12.   else . RemoveAt(i);
```

图1 改进的单维关联规则的生成频繁1-项集算法

表1 一个用于挖掘单维规则的立方体表

id	开关柜	元器件	灯具	母线槽	桥架	Any
1	20	10	20		20	70
2		100	60		25	185
3	20	110		45		175
4			40		30	70
Any	40	220	120	45	75	500

以表1为例：id作为事务维，各个产品作为项维，得到的项支持度为：{开关柜：2，元器件：3，灯具：3，母线槽：1，桥架：3}。如果最小支持度计数为2，则最后得到的频繁项列表L为{元器件：3，灯具：3，桥架：3，开关柜：2}，且L按支持度递减顺序排列。

构造FP树和FP树挖掘同原FP-增长算法一样。

采用表1这种结构很容易地解决了OLAP中单维关联挖掘的问题，但是存在一个问题：假如产品很多，但实际拥有订单的却很少，那么表中势必会出现大量的空值，这会产生大量无用开销，因此，我们提出另外一种基于存储效率的改进。

2.2 基于存储效率的改进

表2 只保存拥有订单的产品

id	Product
1	开关柜
1	元器件
1	灯具
1	桥架
2	元器件
…	…

为了节省对大量空值进行的存储和运算，我们采用如表2格式的结构存储数据，只对拥有订单的产品进行显示，与表1比较，它只保存值不为空的单元。因此，在表比较稀疏的情况下，采用这种结构会大大节省存储空间和运算消耗。

算法略有改变：初始化一个空的项集列表，遍历表，若产品没有出现过，添加此项到项集列表，项支持度为1，若产品项曾经出现过，则项支持度加1，最后，项集列表中大于最小支持度的项构成频繁1-项集。构造FP树时，将表中拥有相同id号码的项作为一次事务处理，项按支持度递减顺序插入到树中。

代码如图2所示：

```
GetFrequentItem(decTable,minSupport)
1. foreach row in decTable
2.    if L contain row[1]//如果频繁1-项集列表包含行的第二列
3.       L[row[1]]. FrequenTimes++;//项支持度加1
4.    else L. Add(row[1]);//如果是新项，添加此项到L中
5. 将项集中的项进行顺序，按支持度递减
ConstructTree(decTable,L)
1. while not eof(decTable)
2.    pattern=每个相同id生成的列表://只包含频繁项，按支持度递减顺序
3.    fpTree=null;//创建树的头节点
4.    if(pattern. Count>0)
5.       InSertTree(pattern,fpTree);//将对应事务的项列表插入到树中
```

图2 只记录拥有订单产品的表结构下的频繁1-项集算法

我们对两种算法进行一下比较。设有表1形式的n行m列的一个表，在图2所示算法中：计算频繁1-项集时读取单元数为n*m，比较次数为n*m。设其中有k个非空值，则在图3所示算法中：计算频繁1-项集时读取单元数为k个，比较次数为k。因此，当k相对于n*m非常小时，我们建议采用表2的结构，存储结构和计算开销都非常小，当k接近于n*m时，采用表1这种压缩的存储结构要比表2效果好。抽象级别很高的情况下，表比较稀疏，可采用表1结构；抽象级别比较低的情况下，表比较稠密，可采用表2结构。

3 FP-增长在OLAP中多维规则上的应用

多维关联是指多个维度间的关联。单维关联中通过计算项的出现次数决定是否频繁，在多维关联中可以通过立方体中的聚合值直接获得支持度，这一改进直接依赖于OLAP的数据结构。多维关联中与任务相关的立方体表如表3所示。在获得立方体表时，如果不加任何限制，则可能出现大量的空值，比如：如果产品维还有值“开关柜”，则会出现行：{凯帆 开关柜 NULL NULL}。为了节省资源并且显示直观，在筛选行时，加“非空”条件。从表3可以很容易地获得分公司维的项“凯帆”的支持度为70。

表3 一个用于挖掘多维规则的立方体表

分公司	产品	利润	Count
凯帆	灯具	好	30
凯帆	灯具	非常好	10
凯帆	灯具	Any	40
凯帆	桥架	好	30
凯帆	桥架	Any	30
凯帆	Any	Any	70
…	…	…	…
Any	Any	Any	480

原FP-增长算法是适用于单维关联规则的挖掘算法，只需计算项维中每个项的支持度，在多维规则中，则需要计算所有维中所有项的支持度。在最后获得的频繁1-项集列表中，所有项不管来自哪个维度，均按支持度递减顺序排列。算法如图3。

```
GetFrequentItem(decTable,minSupport)
1. foreach column in decTable//在立方体表中,一个 column 代表一个维度
2.    foreach item in column
3.      item.frequency=the value in cell(item)//在立方体表中,
        //这些值在类似于(item,any,any)形式的行中,这行的事实值即为所需要的 frequency
4.    if(item.frequency>minSupport)
5.        L.Add(item);
6. for(i=0;i<L.Count-1;i++)//将项集中的项进行排序,按支持度递减
7.    if(L[i].FrequenTimes>minSupport)//大于支持度的项进行排序
8.       for(j=i+1;j<L.Count;j++)
9.          if(L[j].FrequenTimes<minSupport)//若小于支持度,除去
10.            L.RemoveAt(j);
12.         else 调节顺序
13.   else L.RemoveAt(i);
```

图3　改进的多维关联规则的生成频繁1-项集算法

在多维规则挖掘中,我们很容易确定 FP 树的深度,不计算树的头节点 null,深度最大为维数,即模式长度可以确定,但是模式个数可能相当多,比如,分析表2,设支持度为80,获得的频繁1-项集为{元器件:220,好:200,阳光:185,银河:175,灯具:125,差:125}。构造 FP 树时,对于表中第二行,只有项"灯具"属于频繁1-项,因此在 FP 树中,它要么是一个分支的头节点,要么是一个独立节点。但事实上,这种形式的节点对于产生频繁模式没有任何用处。因此,在构造 FP 树时,我们只对包含两个以上频繁1-项的行进行操作,可以大大减少这种节点的出现次数。

多维规则的 FP 树挖掘中,尽管组成 FP 树的项不同于单维规则,但挖掘算法是一样的。

4　FP-增长在 OLAP 中混合维规则上的应用

混合维关联规则挖掘在传统的事务数据库上一直是一个比较复杂的问题。这是因为在传统事务数据库中,进行混合维关联时项维和普通维度的处理有所区别:普通维度只能进行维间连接,而项维内部的项是可以同时进行维内连接和维间连接的。因此,在生成频繁项的过程中,需要进行复杂的判断以决定实施维内还是维间的连接。而在 OLAP 下采用 FP-增长的混合维关联规则算法,完全避免了这种复杂的判断。

表4　一个用于挖掘混合维关联规则的立方体表

ID	分公司	利润	开关柜	元器件	灯具	母线槽	桥架	Count
1	凯帆	好	20	10	20		20	30
…	…	…	…	…	…	…	…	

在与任务相关的立方体表的表示上,我们综合了多维和单维中基于关系表结构的改进的特点,普通维度的表示同多维规则中一样,项维则采用关系表结构,其中每一项作为一个字段存在,以表4为例:分公司、利润是普通维度,开关柜、元器件、灯具、母线槽以及桥架这五种产品是项维中的项。

我们对这种结构进行一下理论分析:首先计算所有普通维度以及所有项的支持度,频繁项不分种类按递减顺序排列;其次构造 FP 树,由 FP-增长算法的证明[9]可以知道,表中的每一行都将作为 FP 树中的一个分支存在(分支可以合并),因此不会遗漏表中任一行,对于每一行所形成的分支,其上的节点由普通维度和项维中的项组成,对于普通维度,每个分支上至多有一个节点代表此维,对于项维,凡是属于频繁1-项集的项都将在分支上以一个节点的形式存在;最后进行 FP 树挖掘,FP 树的挖掘过程最终归结到求单个分支的所有组合,由于分支上至多有一个节点代表一个普通维度,因此最终只能出现此维度与其它普通维度或者项维的频繁模式,这样就完全避免了此维度的维内连接,而对于项维中的项来说,凡是属于频繁1-项集的项都将在分支上以独立节点的形式存在,所以绝对不会遗漏项维的维内连接。

由分析可以得出:采用这种数据结构,我们完全可以实现 OLAP 中的混合维关联规则挖掘,并且避免了维内连接和维间连接的复杂比较。

从分析过程还可以看到:只要采用 OLAP 环境下的基于 FP-增长的多维关联挖掘算法,就可以有效解决混合维关联挖掘。因此在 OLAP 中采用 FP-增长算法进行挖掘关联规则时,我们可以将单维、多维和混和维三种关联规则合并为两类:单维和多维。

5　基于 OLAP 中事实值的约束关联规则

到目前为止,我们已经通过从立方体中获得频繁模式实现了 OLAP 中的关联规则挖掘。但实际情况中,尽管有最小支持度作为限定条件,最后依然会生成大量的频繁模式,势必会阻碍使用者尽快地发现最有意义的模式。在本小节中,将给出另外一个改进:结合 OLAP 中的事实值生成更有意义的频繁模式。

在上面三种规则挖掘中,通过计算模式的出现次数(单维)或者直接依赖 OLAP 生成的 count 值(多维和混合维)决定模式的支持度,这两种情况都是单纯地依赖模式出现的频率决定模式支持度。而事实上除了模式频率,还有另外一些更有意义的值是令人关注的,这些值往往就是 OLAP 中的事实值,如利润、销量、成本等等,结合这些事实值,模式可以有更多的含义,比如:在购物栏分析中,某些模式可能出现次数很高,但是利润极低,某些模式出现次数并不是太多,但结合利润来分析,可能更有价值。因此,可以以模式出现的频率为频繁模式的主要判断依据,同时,根据给定事实值进一步限定频繁模式的条件,从而生成更有价值的规则。

产生与任务相关的立方体,转化成相应的立方体表。借助 OLAP 的结构,可以同时获得出现次数和事实值。生成两种频繁1-项集:基于频率和基于事实值。代码如图4。

```
GetFrequentItem(decTable,minFequency,minSupport)
1. foreach column in decTable//在立方体表中,一个 column 代表一个维度
2.    foreach item in column
3.      item.frequency=the value in cell(item);
4.      item.factsup=the value in cell(item);
        //在立方体表中,这些值在类似于(item,any,any)形式的行中,并且,在每个 cell 中,可以存储多个这种事实值
5.    if(item.frequency>minFrequency)
6.        L.Add(item);
7.    if(item.factsup>minSupport)
8.        FactL.Add(item);//FactL 存放基于事实值的频繁1-项集
9. for(i=0;i<L.Count-1;i++)//将项集中的项进行排序,按支持度递减
10.   if(L[i].FrequenTimes>minSupport)//大于支持度的项进行排序
11.      for(j=i+1;j<L.Count;j++)
12.         if(L[j].FrequenTimes<minSupport)//若小于支持度,除去
13.            L.RemoveAt(j);
14.         else 调节顺序
15.   else L.RemoveAt(i);
```

图4　基于一定约束的关联规则挖掘的频繁1-项集生成算法

构造 FP 树。以基于频率的频繁1-项集为主,同时属于基于事实值的频繁1-项集的项才是最终的频繁1-项集。以最终频繁1-项集为基础构造 FP 树。代码如图5。

```
InsertTree(pattern,fpTree)
1. if pattern[0] in FactL
2.    if fpTree.Nodes contain pattern[0]//fpTree 有子女节点同首项
                                          相同
3.       fpTree.Nodes[i].Support++;
4.       pattern.RemoveAt(0);//除去列表首项
5.       if(pattern.Count>1)
6.     InsertTree(pattern,fpTree.Nodes[i])//剩余项继续插入树中
7.    else
8.       fpTree.Add(newNode);
9.       在项头表中找到此项的头 hn,记录此项;
10.       if(pattern.Count>1)InsertTree(pattern,fpTree.Nodes[i])
```

图5　依据两种频繁1-项集构造 FP 树的算法

进行 FP 树挖掘,此过程同其它类型算法一样。

下面给出一个产生频繁模式的完整例子。Factory、Product、Class 为三个维度,Count 为每个模式的出现次数,Profit 是每个模式的利润(事实值)。首先我们分别依据 Count 和 Profit 生成两种频繁1-项集,设 minFrequency 为80,minSupport 为500,见图6。我们看到,Class 维中的"4"出现在基于 Count 生成的频繁1-项集中,但是另外一个频繁1-项集中却没有,因此在图7中所示的构造 FP 树过程中,对于"4"我们不构造任何节点。而对于 Factory 维中的"M"在基于 Profit 的频繁1-项集中出现,因为我们以基于 Count 生成的频繁1-项集为主,所以认为尽管利润可观,但不频繁,视为偶然情况,最终也不生成任何节点。图8显示了根据 FP 树生成的频繁模式。由于构造 FP 树的过程中没有对"4"构造节点,FP 树减少了3个节点,FP 树挖掘产生的频繁模式中减少了2个频繁2-项、一个频繁3-项。相对于单个条件约束的情况,多重条件下生成模式的个数大大减少,且更有意义。

Factory	Product	Class	Count	Profit
K	S	1	30	7
K	S	0	10	10
K	W	1	30	8
M	A	4	20	2
M	C	4	85	1
M	C	2	25	6
M	T	3	25	4
M	T	0	20	11
D	C	1	100	9
D	S	4	20	2
D	S	1	40	8
D	W	2	25	7
Y	C	3	10	4
Y	S	2	20	5
Y	t	3	20	3

基于出现频率的频繁1-项集:

C	1	D	M	4	S
220	200	185	175	125	120

基于事实值的频繁1-项集:

1	D	C	S	M	K
1670	1435	1175	770	595	550

图6　生成两个频繁1-项集

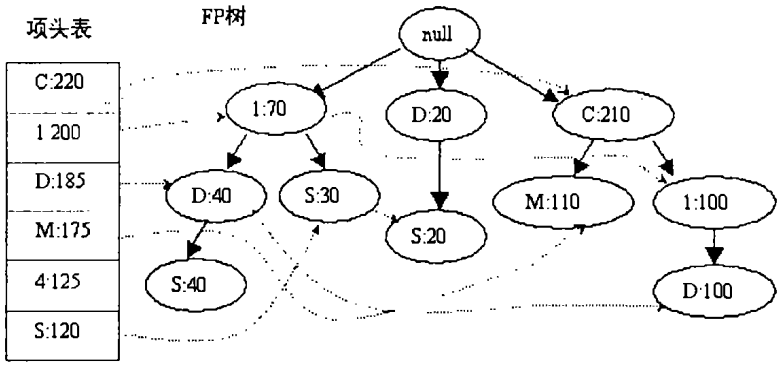

图7　构造 FP 树

频繁1-项	条件模式基	频繁模式
S	1,D:40 1:30 D:20	
M	C:110	C,M:110
D	1:40 C,1:100	C,D:100 1,D:140 C,1,D:100
1	C:100	C,1:100

图8　依据 FP 树获得的频繁模式

结论　数据仓库和 OLAP 凭借其集成的数据结构、勘探式的分析方法必将成为数据挖掘的主要平台。基于此种情况,本文提出了将 OLAP 和数据挖掘相结合的想法,将原先适用于事务数据库的 FP-增长算法进行改进,使其适用于 OLAP 环境。

在算法研究过程中,我们针对目前已有的 OLAP 中的关联规则挖掘算法的不足,从多个方面进行了改进:首先在原型方法选择上,我们比较了众多关联规则挖掘算法,特别是已经运用到 OLAP 环境中的 Apriori 算法,相对于这些算法,FP-增长算法对于挖掘长的和短的频繁模式,都是有效的和可伸缩的,并且大约比 Apriori 算法快一个数量级。其次在与 OLAP 对关联规则挖掘的支持上,我们针对单维、多维、混和维三种关联规则各自的特点采用了不同的方法。在单维上,我们通过计算待挖掘维中每个项的出现次数决定项是否频繁,并且根据数据特点给出了两种改进,基于关系表结构和存储效率,分别对应于稠密数据和稀疏数据;在多维和混合维上,我们直接借助 OLAP 中立方体的聚合值获得支持度,大大简化了计算。最后结合 OLAP 自身的特点,提出了 OLAP 中的基于一定约束的关联规则挖掘算法。考虑到 OLAP 是基于一定主题构造的,其中包含着使用者更为关注的一些事实值,因此,我们在传统的借助模式频率决定模式频繁程度的基础上,结合事实值进行判断,加强频繁1-项集的生成条件,产生更有意义的频繁模式。

(下转第122页)

表2

净收益	人口密度	人均月收入	可信度	统计值
高	?	?	0.1	50
高	<30000	?	0.05	35
中	>30000	?	0.2	28
中	<30000	<2000	0.01	33
…	…	…	…	…
低	>30000	>2000	0.192	20

对人口密度、人均月收入等属性设定其到泛化阈值，再按照上表泛化各属性，并分别修正可信度和统计值，得出规则结论。

从结果上看，本算法在运算效率上，优于 RBAO 算法大约30%～40%，特别是利用可信度因子丢弃了一些在 RBAO 算法中根本无法判别其是否病态的结论，收到了良好的效果。

结束语 本算法引入可信度因子并实行主概念属性一次性泛化来实施概念树提升，不仅解决了算法回溯的问题，而且对推理结果进行了有效验证。因而其效率和准确性都有了较大提升。周生炳等人提出的无回溯的 RBAO 算法，更多地考虑了属性间的依赖关系，但空间数据库数据之间并无明显的属性依赖关系。由规则得出的属性依赖往往是不准确的。本算法主要引入可信度因子并在泛化过程中施以变化，对于解决空间数据推理问题具有较强的适应性和效率。作者利用本算法在地理信息系统中进行概念提升获得了较好的效果。

参 考 文 献

1 Stefanovic N, Han J, Koperski K. Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes. IEEE Transactions on Knowledge and Data Engineering, 2000,12(6)

2 Han J,Cai Y D,Cercone N. Knowledge discovery in databases: An attribute-oriented Approach. In: Proc. of 18th Intel Conf. on Very Large Data Bases, Vancouver, Aug. 1992. 547～559

3 周生炳，张钹，成栋. 基于规则面向属性的数据库归纳的无回溯算法. 软件学报，1999,10(7)

4 Xu X, Ester M, Kriegel H-P, Sander J. A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases. In: Proc. 14th Int. Conf. on Data Engineering (ICDE'98), Orlando, FL, 1998

5 Han J, Stefanovic N, Koperski K. Selective Materialization: An Efficient Method for Spatial Data Cube Construction. In: Proc. 1998 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'98), Melbourne, Australia, April 1998

6 Lu W,Han J. Information Associated Join Index for Spatial Range Search. International Journal of Geographical Information Systems, 1995,9(3):221～249

7 Ester M, Kriegel H-P, Sander J. Spatial Data Mining: A Database Approach. In: Proc. 5th Symp. on Spatial Databases, Berlin, Germany, 1997

8 Han J, Huang Y, Cercone N,et al. Intelligent query answering by knowledge discovery techniques. IEEE Transactions on Knowledge and Data Engineering, 1996,8(3):373～390

9 Carter C L, Hamilton H J. Performance improvement in the implementation of DBLEARN: [Technical Report 94-5]. Department of Computer Science, University of Regina, Sask., Canada, Jan. 1994

（上接第116页）

将基于 Apriori 算法实现的 OLAP 中的关联规则挖掘同本文中的算法从性能上比较来看：首先原型算法的性能不同。FP-增长算法只要遍历数据库2次，Apriori 算法则要根据最终生成的模式长度决定，最终生成的频繁模式越长，遍历数据库的次数越多。Apriori 算法需要频繁的连接、候选、产生频繁项的步骤，并且最小支持度越低，候选集越大，开销也越庞大，FP-增长算法则在第二次遍历数据库后，将整个数据库压缩到 FP 树上，通过 FP 树挖掘一次性地产生所有的频繁模式，省去了候选过程。其次本文提出的恰当的数据结构提高了性能。在单维中对于稀疏数据我们提出了表2的存储结构，减少了计算频繁1-项集时的读取单元数和比较次数。在混合维关联规则挖掘方面，同事辛燕采用 Apriori 算法实现了 OLAP 中的混合维关联规则挖掘，其中要判断连接项进行维内连接还是维间连接，根据维内连接还是维间连接再采用不同的判定条件决定是否可以进行连接，本文中我们在表3的基础上采用 FP-增长算法则巧妙地避免了这一系列的复杂比较，并将其与多维关联挖掘进行了合并。由此可以看到，采用本文中介绍的算法和数据结构可以大大提高性能。

除了文中介绍的几种关联规则挖掘，利用 OLAP 还可容易实现多层次关联规则挖掘：依据 OLAP 数据结构，首先获得感兴趣的任何层次的数据，然后调用规则挖掘算法。对于基于约束的关联规则挖掘，本文只是基于事实值对生成模式进行了一定的约束，实际情况中，还可以限定模式的前件、后件以及模式长度等，在进一步的研究中，我们将在此方面进行更多的研究。

参 考 文 献

1 Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 1997,26:65～74

2 Han Jiawei. Towards On-Line Analytical Mining in Large Databases. 1998

3 Han Jiawei, Chee S H S, Chiang J Y. Issues for On-Line Analytical Mining of Data Warehouses. 1998

4 Hu Xiaohua, Cercone N. An OLAM Framwork for Web Usage Mining and Business Intelligence Reporting. 2002 IEEE

5 Fabris C C, Freitas A A. Incorporating Deviation-Detection Functionality into the OLAP Paradigm. 2001

6 Zhu Hua. Online analytical mining of association rules. 1998

7 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proc. of the ACM SIGMOD Conf. on Management of data, 1993. 207～216

8 Houtsma M. A run Swam i. Set-oriented mining of association rules [A]: [Research Report RJ 9567]. IBM Almaden Research Center [C]. San Jose, California: [s. n.] 1993

9 Han Jiawei, Pei Jian, Yin Yiwen. Mining Frequent Patterns without Candidate Generation. In: Proc. of the ACM Int. Conf. on Management of Data, Dallas, TX, May 2000