第5章　关联规则模型及应用

首先解释关联规则的定义，阐述关联规则在知识管理过程中的重要作用；详细描述Apriori关联规则算法的相关概念和计算流程；提出一种改进的Apriori关联规则方法；最后给出Apriori关联规则方法的实例。

5.1　关联规则的基础理论

5.1.1　关联规则的定义与解释

关联规则(Association Rules)是指在大型的数据库系统中，迅速找出各事物之间潜在的、有价值的关联，用规则表示出来，经过推理、积累形成知识后，得出重要的相关联的结论，从而为当前市场经济提供准确的决策手段。

关联规则的应用已经比较广泛，如条形码的应用已使大型零售商品的组织问题成为现实，从决策领域到通信报警系统的应用，以及诊断和预测等相关领域。

(1) 在主题数据库(销售分析库)中，可以对所有销售的物品、价格、数量、时间(季节)、地区等相关因素进行分析，利用关联规则来发现它们之间的联系，决定如何进货以及物品在货架的摆放形式等。

(2) 如果当前的主题是贷款客户的相关信息，那么利用关联规则算法，可以找出贷款与产品、贷款与库存、贷款与利润、贷款与收入、贷款与贷款人等之间的关联，从而分析会存在风险的贷款情况的关联因素，根据这些规则决定是否给客户发放贷款。

如果当前主题库是某类疾病的患者数据库，那么可以根据每个患者发病史、病状、饮食习惯、居住区、工作、脾气、性格及环境等因素，找出它们之间共同的、潜在的联系，可以做出预防某种疾病的措施。

关联规则的研究和应用是数据挖掘中最活跃和比较深入的分支，目前，已经提出了许多关联规则挖掘的理论和算法。最为著名的是R. Agrawal等提出的Apriori及其改进算法。为了发现有意义的关联规则，需要给定两个阈值：最小支持度(Minimum Support)和最小可信度(Minimum Confidence)。挖掘出的关联规则必须满足用户规定的最小支持度，它表示了一组项目关联在一起需要满足的最低联系程度。挖掘出的关联规则也必须满足用户规定的最小可信度，它反映了一个关联规则的最低可靠度。在这个意义上，数据挖掘系统的目的就是从数据库中挖掘出满足最小支持度和最小可信度的关联规则。

5.1.2　关联规则在知识管理过程中的作用

知识管理是一个过程，通过这一过程可以学习新知识和获得新经验，并将这些新知识和新经验反映出来，进行共享，以用来促进、增强个人的知识和机构组织的价值。如果我们将数据管理中的数据提取作为数据仓库的低层管理过程，那么数据库知识发现(Knowledge

Discovery in Databases，KDD)的过程则可作为数据仓库的高层管理的过程，而关联规则又作为数据仓库的主要内容出台，所以关联规则作为知识管理过程的重要内容，具体的知识管理过程如图 5.1 所示。

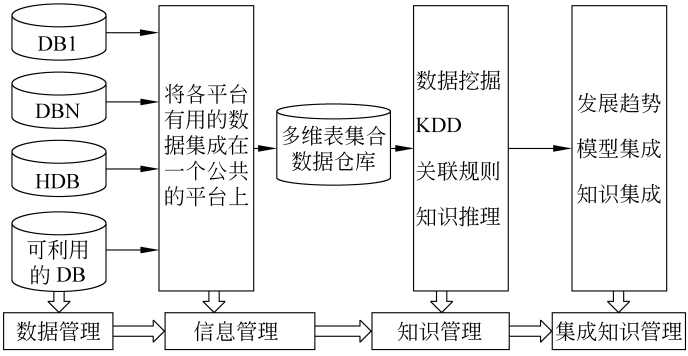

图 5.1　知识管理过程及其发展

知识管理的基础内容包括从系统继承过来的模型与知识。在未来的发展趋势中，也可以根据模型的集成进行模型的推理、知识的推理等推导过程来产生规则和获得知识。这个阶段产生的规则应该是信息集成后的规则。如果在 KDD 过程中的一条通用的规则是：IF 表达式(前件)THEN 动作(后件){其中：前件作为表达式的概念，后件作为满足表达式的逻辑状态所产生的动作}，那么在这个阶段的规则形式可以写成：IF 有用的模型 THEN 集成与提炼新的模型；或者 IF 有用的知识 THEN 集成与提炼新的知识。当然，知识集成是按照新的、更高更复杂的问题需求而集成的。这个阶段的基础应该是 KDD 处理后的结果，也就是说 KDD 作为知识集成阶段的基础。因此，知识管理是以 DBS 为基础，应用多种知识发现和决策支持理论与技术方法。而模型的挖掘与知识的挖掘、模型的集成与知识的集成阶段将是知识管理的未来发展趋势。

如果只利用简单的统计与分析方法寻找事物间的关联，可能只看到外部事物间的关联，而无法找到事物内部间的关联。关联规则在大型的数据库系统中为我们提供了各属性(项)之间的潜在的、有价值的联系，使用关联规则也能找出其他主题的大型数据库中的各属性之间的潜在的间接的关联。这对于分析各类事物将要导致其他的潜在的发展趋势是十分重要的。在 KDD 中利用关联规则算法解决这一类问题是目前挖掘潜在的相关联的各事物间关系较好的方法。关联规则在知识管理中起着一种桥梁的作用，如图 5.2 所示，在数据仓库系统中属于数据挖掘和 DSS 的技术。换句话说，数据挖掘的结果会产生许多有价值的模型，在数据挖掘过程中能根据不同的主题发现不同的模式，而这些模式可以是一个表达式、一个过程、一个规则、一条有意义的信息、继承过来的知识等。

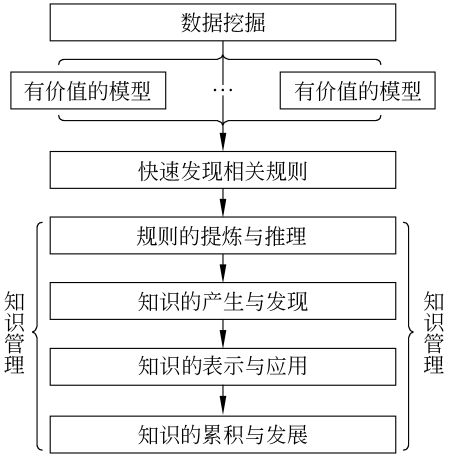

图 5.2　关联规则在知识管理中的桥梁作用

5.2 Apriori 关联规则算法

5.2.1 关联规则算法的相关概念

下面简要介绍关联规则的相关概念。

1. 项集或候选项集

项集 Item=$\{Item_1, Item_2, \cdots, Item_m\}$；TR 是事物的集合；TR⊂Item，并且 TR 是一个{0,1}属性的集合。集合 k_Item=$\{Item_1, Item_2, \cdots, Item_k\}$称为 k 项集或者 k 项候选项集。假设 DB 包含 m 个属性$(A, B, \cdots, M)$；1 项集 1_Item=$\{\{A\}, \{B\}, \cdots, \{M\}\}$，共有 m 个候选项集；2 项集 2_Item=$\{\{A, B\}, \{A, C\}, \cdots, \{A, M\}, \{B, C\}, \cdots, \{B, M\}, \{C, D\}, \cdots, \{L, M\}\}$，共有$[m\times(m-1)/2]$个项集；3 项集 3_Item=$\{\{A, B, C\}, \{A, B, D\}, \cdots, \{A, B, M\}, \{A, C, D\}, \{A, C, E\}, \cdots, \{B, C, D\}, \{B, C, E\}, \cdots, \{B, C, M\}, \cdots, \{K, L, M\}\}$；依次类推，$m$ 项集 m_Item=$\{A, B, C, \cdots, M\}$，有一个项集。

2. 支持度

支持度 support 简写为 sup，指的是某条规则的前件或后件对应的支持数与记录总数的百分比。假设 A 的支持度是 sup(A)，

$$\text{sup}(A)=|\{\text{TR}|\text{TR}\supseteq A\}|/|\text{n}|;$$

$A\Rightarrow B$ 的支持度

$$\text{sup}(A\Rightarrow B)=\text{sup}(A\cup B)=|\{\text{TR}|\text{TR}\supseteq A\cup B\}|/|n|,$$

其中，A∪B 表示 A 和 B 同时出现在一条记录中，n 是 DB 中的总的记录数目。

3. 可信度

可信度 confidence 简写为 conf，规则 $A\Rightarrow B$ 具有可信度 conf($A\Rightarrow B$)表示 DB 中包含 A 的事物同时也包含 B 的百分比，是 $A\cup B$ 的支持度 sup($A\cup B$)与前件 A 的支持度 sup(A)的百分比：

$$\text{conf}(A\Rightarrow B)=\text{sup}(A\cup B)/\text{sup}(A)$$

4. 强项集和非频繁项集

如果某 k 项候选项集的支持度大于等于所设定的最小支持度阈值，则称该 k 项候选项集为 k 项强项集(Large k-itemset)或者 k 项频繁项集(Frequent k-itemset)。同时，对于支持度小于最小支持度的 k 项候选项集称为 k 项非频繁项集。

定理(频繁项集的反单调性)：设 A、B 是数据集 DB 中的项集，若 A 包含于 B，则 A 的支持度大于 B 的支持度；若 A 包含于 B，且 A 是非频繁项集，则 B 也是非频繁项集；若 A 包含于 B，且 B 是频繁项集，则 A 也是频繁项集。

5. 产生关联规则

若 A，B 为项集，A⊂Item，B⊂Item 并且 $A\cap B=\varnothing$，一个关联规则是形如 $A\Rightarrow B$ 的蕴涵式。当前关联规则算法普遍基于 Support-Confidence 模型。支持度是项集中包含 A 和 B 的记录数与所有记录数之比，描述了 A 和 B 这两个物品集的并集 C 在所有的事务中出现的概率有多大，能够说明规则的有用性。规则 $A\Rightarrow B$ 在项集中的可信度，是指在出现了物品

集 A 的事务 T 中，物品集 B 也同时出现的概率有多大，能够说明规则的确定性。产生关联规则，即从强项集中产生关联规则。在最小可信度的条件下，若强项集的可信度满足最小可信度，称此 k 项强项集为关联规则。例如：若{A,B}为 2 项强项集，同时 conf($A\Rightarrow B$)大于等于最小可信度，即 sup($A\cup B$)≥min_sup 且 conf($A\Rightarrow B$)≥min_conf，则称 $A\Rightarrow B$ 为关联规则。

5.2.2 关联规则算法的流程

R. Agrawal 等人在 1993 年设计了一个 Apriori 算法，这是一种最有影响力的挖掘布尔关联规则频繁项集的算法。其核心是基于两阶段的频集思想的递推算法。该关联规则在分类上属于单维、单层、布尔关联规则。该算法将关联规则挖掘分解为两个子问题：

(1) 找出存在于事务数据库中所有的频繁项目集。即那些支持度大于用户给定支持度阈值的项目集。

(2) 在找出的频繁项目集的基础上产生强关联规则。即产生那些支持度和可信度分别大于或等于用户给定的支持度和可信度阈值的关联规则。

在上述子问题中，(2)相对容易些，因为它只需要在已经找出的频繁项目集的基础上列出所有可能的关联规则，同时，满足支持度和可信度阈值要求的规则被认为是有趣的关联规则。但由于所有的关联规则都是在频繁项目集的基础上产生的，已经满足了支持度阈值的要求，只需要考虑可信度阈值的要求，只有那些大于用户给定的最小可信度的规则才被留下来。第一个步骤是挖掘关联规则的关键步骤，挖掘关联规则的总体性能由第一个步骤决定，因此，所有挖掘关联规则的算法都是着重于研究第一个步骤。

Apriori 算法在寻找频繁项集时，利用了频繁项集的向下封闭性(反单调性)，即频繁项集的子集必须是频繁项集，采用逐层搜索的迭代方法，由候选项集生成频繁项集，最终由频繁项集得到关联规则，这些操作主要是由连接和剪枝来完成。下面为 Apriori 算法的基本流程。

```
L1={Large 1-itemsets}          //扫描所有事务，计算每项出现次数，产生频繁 1-项集集合 L1
for(k=2; Lk-1≠∅; k++) do       //进行迭代循环，根据前一次的 Lk-1 得到频繁 k-项集集合 Lk
  begin
   Ck'=join(Lkm,Lkn)
                    //join 对每两个有 k-1 个共同项目的长度为 k 的模式 Lkm 和 Lkn 进行连接
   Ck=prune(Ck')               //prune 根据频繁项集的反单调性，对 Ck'进行减枝，得到 Ck
   Ck=apriori-gen(Lk-1)        //产生 k 项候选项集 Ck
   for all transactions t∈D do      //扫描数据库一遍
    begin
      Ct=subset(Ck,t)               //确定每个事务 t 所含 k-候选项集的 subset(Ck,t)
      for all candidates c∈ Ct do
        c.count++                   //对候选项集的计数存放在 hash 表中
        end
      Lk={c∈Ct|c.count≥min_sup} //删除候选项集中小于最小支持度的，得到 k-频繁项集 Lk
    end
    for all subset s⊆Lk             //对于每个频繁项集 Lk，产生 Lk 的所有非空子集 s
    If conf(s⇒Lk-s)>=min_conf       //可信度大于最小可信度的强项集为关联规则
    Then Output(s⇒Lk-s)             //由频繁项集产生关联规则
  end
end                                 //得到所有的关联规则
```

Apriori 算法最大的问题是产生大量的候选项集，可能需要频繁重复扫描数据库，因此为候选项集合理分配内存，实现对大型数据库系统快速扫描的技术和方法是提高管理规则效率的重要途径，面向大型数据库，从海量数据中高效提取关联规则是非常重要的。

5.2.3 基于 Apriori 算法的关联规则算例

根据某超市的五条客户购物清单记录(见表 5.1)，设最小支持度为 40%，最小置信度为 60%，计算基于 Apriori 算法的频繁项集和关联规则(见表 5.2)。

表 5.1 五条客户购物清单记录

记录号	购物清单	记录号	购物清单
401	咖啡、果酱、面包、香皂、香肠	404	咖啡、洗衣粉、香肠
402	果酱、香肠	405	咖啡、面包、牛奶
403	咖啡、牛奶、香肠		

表 5.2 频繁项集计算过程

候选项集	频繁项集
1 项候选项集 C1 咖啡 4 香肠 4 果酱 2 面包 2 香皂 1 牛奶 2 洗衣粉 1	1 项频繁项集 L1 咖啡 4 香肠 4 果酱 2 面包 2 牛奶 2
2 项候选项集 C2 咖啡,香肠 3 咖啡,果酱 1 咖啡,面包 2 咖啡,牛奶 2 香肠,果酱 2 香肠,面包 1 香肠,牛奶 1 果酱,面包 1 果酱,牛奶 0 面包,牛奶 1	2 项频繁项集 L2 咖啡,香肠 3 咖啡,面包 2 咖啡,牛奶 2 香肠,果酱 2

续表

候选项集	频繁项集
3 项候选项集 C3 咖啡,香肠,面包 1 咖啡,香肠,牛奶 1 咖啡,面包,牛奶 1 咖啡,香肠,果酱 1	3 项频繁项集 L3 空集

根据表 5.2,通过 L2 进行链接,形成三项候选项集,但因为该 C3 集合中的每个项集都有不频繁子集,所以该三项集的集合应被剪掉,L3 为空;最大频繁项集为 L2。

由 L2 形成的可能关联规则如下:

(1) 咖啡⇒香肠,confidence=3/4=75% (2) 香肠⇒咖啡,confidence=3/4=75%

(3) 咖啡⇒面包,confidence=2/4=50% (4) 面包⇒咖啡,confidence=2/2=100%

(5) 咖啡⇒牛奶,confidence=2/4=50% (6) 牛奶⇒咖啡,confidence=2/2=100%

(7) 香肠⇒果酱,confidence=2/4=50% (8) 果酱⇒香肠,confidence=2/2=100%

因为最小置信度为 60%,关联规则为(1)、(2)、(4)、(6)、(8)。

5.3 改进的 Apriori 关联规则方法

本节介绍一种改进 Apriori 关联规则算法,实现对大型数据库系统扫描时关联规则的快速提取,采用了合理分配内存的方法,给出计算长度 k 的强项集存储分配公式,提出了由候选集快速产生强项集的算法,提高了大型数据库产生强项集的效率,该算法与现存算法相比,降低了时间复杂度,同时对于动态存储空间的分配有更强的准确性,在各种条件下效率方面优于 Apriori 算法。

5.3.1 动态存储空间的构建

为了充分利用空间,在程序设计中采用了合理分配内存的方法,给出了计算长度 k 的强项集存储分配公式:$b_k = \sum_{i=1}^{n_k} C_{p_{ki}}^2$,其中 C_k 表示 k 项候选项集。

这个公式为动态运行机制开辟了准确的存储空间。以下部分为分配空间的具体解释:

设共有 M 个属性$\{a_1, a_2, \cdots, a_M\}$

$k=1$ 时,1-项强项集共有 m_1 个属性,即$\{a_{11}, a_{12}, \cdots, a_{1m_1}\}$。

$k=2$ 时,2-项候选集为 1-项强项集中属性的两两组合,所以 2-项候选集中所占空间为 $b_2 = C_{m_1}^2$;扫描数据库,求 2-项强项集。2-项强项集共有 m_2 个属性即$\{a_{21}, a_{22}, \cdots, a_{2m_2}\}$。

$k=3$ 时,3-项候选集为 2-项强项集中每次取出首位相同的两个项集做连接操作,其中,将首位相同的这些属性的集合用 S_{3i} 表示$\{s_{31}, s_{32}, \cdots, s_{3n_3}\}$。相对应在 2-项强项集中,包含这些属性的项出现的次数分别合计为$\{p_{31}, p_{32}, \cdots, p_{3n_3}\}$,3-项候选项集所占空间为 $b_3 =$

$\sum_{i=1}^{n_3} C_{p_{3i}}^2 p_{3i} \geqslant 2, i = 1,2,\cdots,n_3$。扫描数据库，求 3- 项强项集。

同理，依次求 $k-1$ 项强项集，$k-1$ 项强项集共有 m_{k-1} 个属性$\{a_{(k-1)1}, a_{(k-1)2}, \cdots, a_{(k-1)m_{k-1}}\}$。

当求 k 项强项集时，将($k-1$)项强项集中各个项集前($k-2$)个属性相同的这些属性的集合用 S_{ki} 表示$\{s_{k1}, s_{k2}, \cdots, s_{kn_k}\}$，相对应在($k-1$)项强项集中，包含这些属性的项出现的次数分别合计为$\{p_{k1}, p_{k2}, \cdots, p_{kn_k}\}$，$k$ 项候选项集所占空间为 $b_k = \sum_{i=1}^{n_k} C_{p_{ki}}^2 p_{ki} \geqslant 2, i = 1,2,\cdots,n_k$。

5.3.2 快速产生强项集的算法流程

快速产生强关联属性(L_k)的方法描述如图 5.3 所示。

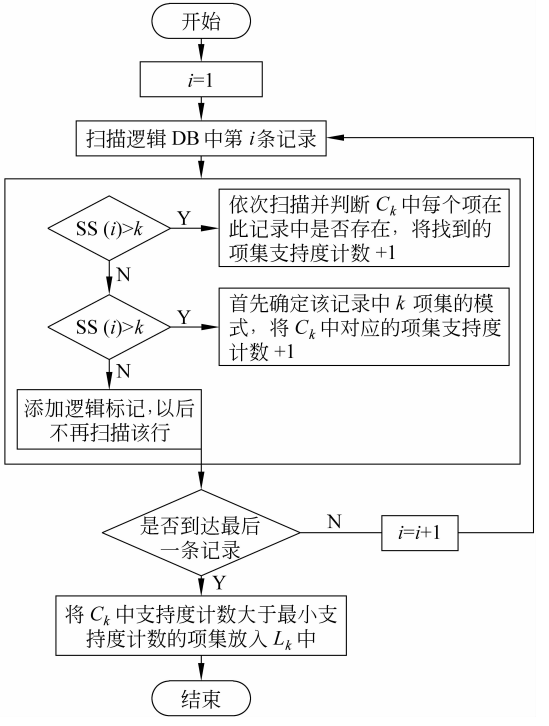

图 5.3 快速产生强关联属性(L_k)的方法

(图中 SS(i)代表第 i 条记录中含有 1 的个数)

(1) 扫描事务数据库中的每个事务，产生候选 1-项集的集合 C_1；

(2) 根据最小支持度 min_sup，由候选 1-项集的集合 C_1，产生强 1-项集合 L_1，对于在事务数据库中出现次数比最小支持度 min_sup 计数少的属性列进行逻辑标记，在以后的各次扫描中跳过这些属性；

(3) 求 k 项集，令 $k=1$；

(4) 由 L_k 产生候选($k+1$)-项集的集合 C_{k+1}；

(5) 根据最小支持度 min_sup，由候选($k+1$)-项集的集合 C_{k+1}，产生($k+1$)-强项集的

集合 L_{k+1}，方法是扫描数据库，当执行到第 i 行时：

① 若该行的项集长度小于$(k+1)$，则对该行作出逻辑标记，在以后的各次扫描中，都可以跳过该行，不再扫描；

② 若该行的项集长度等于$(k+1)$，确定该行项集的模式，与候选项集中的模式进行匹配，匹配成功则该项集的支持度计数器+1，对候选项集中的其他模式，在本行中不再扫描；匹配不成功则跳过本行；

③ 若该行的长度大于$(k+1)$，将此行中与候选 $k+1$ 项集模式相匹配的项集支持度计数器+1。将候选集 C_{k+1} 中所有项集的支持度与 min_sup 进行比较，产生 L_{k+1}。

(6) 若 $L_{k+1}\neq\varnothing$，则 $k=k+1$，跳往步骤(4)，否则，跳往步骤(7)；

(7) 根据最小置信度 min_conf，由强项集产生关联规则，结束。

5.3.3 改进算法的时间复杂性分析

Apriori 算法的时间复杂性为 $\lg\begin{bmatrix}m\\k\end{bmatrix}\approx k\lg(m/k)$。一般来说，$k\ll p$，而 p 作为被删除的列，k 作为强项集的长度。对改进后的关联规则算法的时间复杂度的分析如下：

(1) 在最坏的情况下，当 $p=k$ 时，有 $\lg\begin{bmatrix}m-p\\k\end{bmatrix}\approx k\lg((m-k)/k)\approx k\lg(m/k)$；

(2) 当 $k<p$ 或者 $k\ll p$(属于一般的情况)时，满足 $\lg\begin{bmatrix}m-p\\k\end{bmatrix}\approx k\lg((m-p)/k)$。

因此，共节省时间是 $k\lg(p/k)$｛一般地说，$k\ll p$｝，这对于一个大型的数据库提高系统的使用效率来说是非常重要的。

在解决以上三个主要研究问题后，总结改进的 Apriori 方法的计算步骤，快速产生强关联属性的关联规则方法总体流程为：

(1) 将 DBS 问题转换成抽象的 DBS：将数据库中的数量相关的问题转换成逻辑相关的问题。按照决策问题要求，将数据库中的各个属性转换成多维逻辑属性。

(2) 求强项集：该问题可以分解为两个子问题：

① 求出 D 中满足最小支持度 min_sup 的所有强项集；

② 利用强项集生成满足最小可信度 min_conf 的所有关联规则。

本方法对子问题①的求解是知识发现的关键部分。具体方案描述如下：由候选 1-项集的集合 C_1，产生强 1-项集合 L_1，对于在数据库中出现次数比 min_sup 计数少的属性列进行逻辑标记，在以后的各次扫描中跳过这些属性；求 k 项集，令 $k=1$；由 L_k 产生候选$(k+1)$-项集的集合 C_{k+1}；根据 min_sup，由候选$(k+1)$-项集的集合 C_{k+1} 产生$(k+1)$-强项集的集合 L_{k+1}，当执行到第 i 行，若该行的项集长度小于$(k+1)$，则对该行做出逻辑标记，在以后的各次扫描中，都可以跳过该行，不再扫描；若该行的项集长度等于$(k+1)$，确定该行项集的模式，与候选项集中的模式进行匹配，匹配成功则该项集的支持度计数器+1，对候选项集中的其他模式，在本行中不再扫描；匹配不成功则跳过本行；若该行的长度大于$(k+1)$，将此行中与候选 $k+1$ 项集模式相匹配的项集支持度计数器+1，将候选集 C_{k+1} 中所有项集的支持度与 min-sup 进行比较，产生 L_{k+1}。

(3) 将抽象的 DBS 问题转换成 DBS，表达关联规则。

总体流程图如图 5.4 所示。

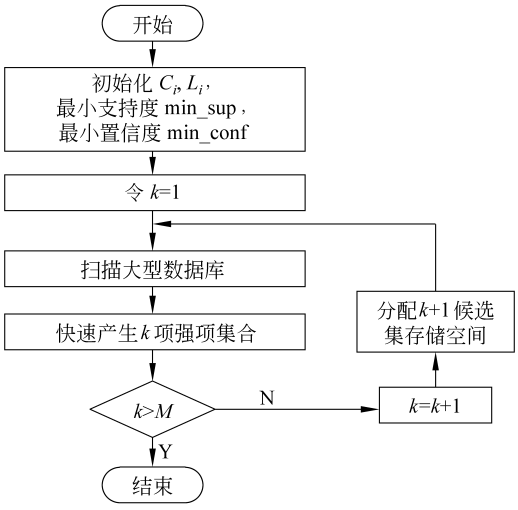

图 5.4　快速产生强关联属性的关联规则方法总体流程图

5.4　Apriori 关联规则方法的实例

通过关联规则分析受过高等教育与性别、工资收入、职业、年龄等之间的潜在关联。给出一个简单的数据库的例子，如表 5.3 所示。

表 5.3　一个简单的数据库的例子

RECID	SEX	AGE	KNOWLEDGE	OCCUPATION	WAGES
100	male	46	Doctor	Teacher	7500
200	female	32	Master	Teacher	6500
300	male	35	Bachelor	Technician	4900
400	male	40	Master	Teacher	6000
500	male	37	Doctor	Teacher	7000
600	male	25	Bachelor	Technician	4000

1. 首先将实际的 DBS 问题转换成逻辑值

对性别 SEX 二元化(1：male，2：female)；对年龄 AGE 离散化(3：old，AGE≥40；4young，AGE<40)；对是否受过研究生教育 KNOWLEDGE 离散化(博士或者硕士，5：high；本科和本科以下，6：low)；对职业 OCCUPATION 进行二元化处理(7：Teacher，高校教师；8：Technician，非高校教师)；对收入 WAGES 进行二元化处理(9：WAGES>5000，10：WAGES<5000)。通过以上的数据规约，表 5.4 给出了与表 5.3 相对应的逻辑表格。

表 5.4　数据库对应的逻辑库

RECID	SEX		AGE		KNOWLEDGE		OCCUPATION		WAGES	
	1	2	3	4	5	6	7	8	9	10
100	1	0	1	0	1	0	1	0	1	0
200	0	1	0	1	1	0	1	0	1	0
300	1	0	0	1	0	1	0	1	0	1
400	1	0	1	0	1	0	1	0	1	0
500	1	0	0	1	1	0	1	0	1	0
600	1	0	0	1	0	1	0	1	0	1

用关联规则算法找出表 5.4 中各属性之间有价值的、潜在的关联的信息即规则，希望最终可以获得高等教育与工资、性别与职业、职务与工资等属性之间的关联。经过检索逻辑库(参见表 5.4)得到每条记录中各个 Item 的取值，如表 5.5 所示。

2. 设最小支持度 min_sup＝0.5，最小置信度 min_conf＝0.7 求得关联规则

通过数据库查询(参见表 5.5)得到 k 项候选集和 k 项强项集(L_k)及关联规则。

(1) 求 1 项集和 1 项强项集，如表 5.6 所示。

表 5.5　数据库中记录的属性项取值集合

Recid	Items	Recid	Items
100	1，3，5，7，9	400	1，3，5，7，9
200	2，4，5，7，9	500	1，4，5，7，9
300	1，4，6，8，10	600	1，4，6，8，10

表 5.6　1 项集和 1 项强项集

Item	Sum	sup(I)	L_1	Item	Sum	sup(I)	L_1
{1}	5	5/6	√	{6}	2	2/6	
{2}	1	1/6		{7}	4	4/6	√
{3}	2	2/6		{8}	2	2/6	
{4}	4	4/6	√	{9}	4	4/6	√
{5}	4	4/6	√	{10}	2	2/6	

所以 1 项强项集 L_1＝{{1}，{4}，{5}，{7}，{9}}。

(2) 通过 1 项强项集得到 2 项候选集，再计算 2 项集的支持度得到 2 项强项集，如表 5.7 所示。所以 2 项强项集

$$L_2=\{\{1,4\},\{1,5\},\{1,7\},\{1,9\},\{5,7\},\{5,9\},\{7,9\}\}$$