基于近似函数依赖的关系数据属性权重评估方法

张霄雁[1] 孟祥福[1] 马宗民[2] 张文博[2] 张霄鹏[3]
(辽宁工程技术大学电子与信息工程学院 葫芦岛 125105)[1]
(东北大学信息科学与工程学院 沈阳 110819)[2] (山东建筑大学信息与电气工程学院 济南 250101)[3]

摘 要 在现实应用中,一些关系数据的规范化程度不高,往往存在数据冗余和不一致现象。为了有效评估此类数据中的属性重要程度,提出了一种基于近似函数依赖的属性权重评估方法。该方法基于一致集的概念导出最大集,生成最小非平凡函数依赖集,从而找出属性之间的近似函数依赖关系,进而求出近似候选码和近似关键字,在此基础上根据属性支持度计算属性权重。实验结果和分析表明,提出的属性权重评估方法能够合理地获取关系数据中的属性重要程度,算法具有较好的稳定性和较高的执行效率。

关键词 关系数据,近似函数依赖,属性权重,最小平凡函数依赖
中图法分类号 TP391 **文献标识码** A

Attribute Weight Evaluation Approach Based on Approximate Functional Dependencies

ZHANG Xiao-yan[1] MENG Xiang-fu[1] MA Zong-min[2] ZHANG Wen-bo[2] ZHANG Xiao-peng[3]
(College of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China)[1]
(College of Information Science and Engineering, Northeastern University, Liaoning 110819, China)[2]
(School of Information and Electrical Engineering, Shandong Jianzhu University, Jinan 250101, China)[3]

Abstract In real applications, the normalization of some relational data is unreasonable and thus leads to the problems of data redundancy and inconsistency. In order to automatically evaluate the attribute importance of this kind of relational data, this paper proposed an attribute weight evaluation approach based on approximate functional dependencies. Based on the concept of the agree set, the maximum set is exported, and the minimum nontrivial functional dependence sets are generated consequently in order to find the approximate dependence relations, thus the approximate key and approximate keywords can be found. After this, this approach computes the weight of each attribute according to the supported degree of attribute. The experimental results and analysis demonstrate that the attribute weight evaluation approach presented in this paper can reasonably obtain the importance of the attribute in a relation, and the algorithm is stable and has high performance as well.

Keywords Relational data, Approximate functional dependence, Attribute weight, Minimal trivial functional dependence

1 引言

关系数据中属性权重的确定对于数据分析、知识发现、查询优化和结果排序等方面至关重要。在关系数据中,一个属性重要是指元组对应该属性上的值一旦发生变化,该元组其他属性上所对应的值发生变化的程度很大;反之,如果元组对应该属性上的值发生变化时,对于该元组其他属性所对应值的变化影响很小,则该属性不重要。属性之间的函数依赖能够有效体现关系数据之间的这种关联关系,目前已经有一些工作研究了规范化关系数据之间的函数依赖关系挖掘方法,其中代表性的工作如文献[1-4]。然而应该指出的是,在现实应用中,一些关系数据的规范化程度仍然不高,往往还存在数据冗余和不一致等现象,对于此类数据,现有方法很难从中挖掘出严格的函数依赖关系。因此,本文从整体数据内容之间的关联关系角度出发,通过挖掘关系数据中属性列之间的近似函数依赖关系确定属性权重,以便容忍一定限度内的数据冗余和不一致现象存在,从而使属性权重评估方法更适用于客观实际。

本文第2节介绍基本概念;第3节描述近似函数依赖关系挖掘方法;第4节给出求近似候选码的方法;第5节给出属性重要程度排序和权重评估方法;第6节是性能实验分析;最后是总结和展望。

2 基本概念

定义1(关系模式) 关系模式是对属性关系的描述,它是一个五元组:$R(U, D, DOM, F)$。由于在属性关系模式中,

到稿日期:2012-04-15 返修日期:2012-06-28 本文受国家青年科学基金项目(61003162),中国煤炭工业协会科学技术研究指导性计划项目(MTKJ2009-242,MTKJ2010-337,MTKJ2011-335),辽宁省科技厅计划项目(201104090)资助。
张霄雁(1983—),女,硕士,助教,主要研究方向为Web数据库与XML查询优化技术;孟祥福(1981—),男,副教授,CCF会员,主要研究方向为Web数据库、XML柔性查询技术,E-mail:marxi@126.com(通信作者);马宗民(1965—),男,博士,教授,博士生导师,主要研究方向为智能数据与知识工程;张文博(1984—),女,硕士,工程师,主要研究方向为XML不精确查询技术。

影响模式设计的主要是 U 和 F，因此通常把它简记为一个三元组 $R\langle U,F\rangle$，其中 R 代表关系名，U 代表组成该关系的属性集合，D 代表属性组 U 中的属性来自的域，DOM 代表属性向域的映像的集合，F 代表属性间函数依赖关系的集合。

设 U 为属性集总体，F 是 U 上的一组函数依赖关系，于是有关系模式 $R\langle U,F\rangle$。对任何一个关系模式 R 的实例 r，若函数依赖 $X\rightarrow Y$ 都成立(即关系 r 中任意两个元组 t 和 s，若 $t[X]=s[X]$，则 $t[Y]=s[Y]$)，则称 F 逻辑蕴含 $X\rightarrow Y$。

定义 2(近似函数依赖) 对于在 R 上的函数依赖 $X\rightarrow A$，如果它对于 r 中绝大多数元组成立，而仅在一个给定阈值 $T_{er}\in(0,1)$ 内的元组集合上不成立，则称 $X\rightarrow A$ 是 R 上的近似函数依赖。误差阈值 $T_{er}\in(0,1)$ 衡量了不满足依赖关系 $X\rightarrow A$ 的元组数在元组总数中所占的比率，当且仅当 $error(X\rightarrow A)\leqslant T_{er}$，$X\rightarrow A$ 是一个 R 上的近似函数依赖。

若在 T'_{er} 内，$X\rightarrow A$ 成立，但是对于任何 $X'\subset X$，$X'\rightarrow A$ 不成立，则称 $X\rightarrow A$ 是 R 上的一个最小近似函数依赖。若在 T'_{er} 内，$A\in X$，则称 $X\rightarrow A$ 是平凡的函数依赖。当 $A\notin X$ 时，$X\rightarrow A$ 是非平凡函数依赖，本文讨论的都是非平凡函数依赖。

定义 3(最小函数依赖集) 如果函数依赖集 F 满足下列条件，则称 F 为一个极小函数依赖集，也称其为最小函数依赖集或最小覆盖。

(1) F 中任一函数依赖的右部仅含有一个属性；

(2) F 中不存在这样的函数依赖 $X\rightarrow A$，使得 F 与 $F-\{X\rightarrow A\}$ 等价；

(3) F 中不存在这样的函数依赖 $X\rightarrow A$，X 有真子集 Z，使得 $F-\{X\rightarrow A\}\cup\{Z\rightarrow A\}$ 与 F 等价。

定义 4(近似关键字) 对于属性集 $X\subset R$，如果在 r 上不存在两个不同的元组在 X 集合上一致，那么 X 是一个在 r 上的关键字。如果 X 的唯一性不包括 r 中阈值内的元组集合，那么 X 就是一个近似关键字，形式化表示为 $error(X\rightarrow A)\leqslant T_{er}$，$T_{er}\in(0,1)$，$error(X\rightarrow A)$ 衡量了要使 X 成为一个关键字需要从 r 中移除的最少元组数占 r 中元组总数的比率。近似关键字的提取是在最小非平凡函数依赖关系基础上进行的[5]。

本文使用了文献[6]提出的方法提取近似函数依赖关系，该方法基于一致集(agree set，简称 ag)的概念[7]，根据一致集导出最大集(maximal set)，然后生成最小非平凡函数依赖集。下面，令 $dep(r)$ 表示 r 中成立的所有函数依赖集，即 $dep(r)=\{X\rightarrow A\mid X\cup A\subseteq R,r\models X\rightarrow A\}$。在此基础上，定义一致集、最大集及其补集。

定义 5(一致集) 设 $t,t'\in r$，$X\subseteq R$，若 $t[X]=t'[X]$，则称 t 和 t' 在 X 上一致，t 和 t' 的一致集定义为 $ag(t,t')=\{A\in R\mid t[A]=t'[A]\}$。

定义 6(最大集) 设 $A\in R$，属性集 A 的最大集是 r 中不能决定 A 的最大属性集 X 的集合，即 $\max(dep(r),A)=\{X\subseteq R\mid r\not\models X\rightarrow A$ and $\forall Y\subseteq R,X\subset Y,r\models Y\rightarrow A\}$。

定义 7(最大集的补集) $cmax(dep(r),A)=\{R-X\mid X\in\max(dep(r),A)\}$，其中 $A\in R$。

定义 8(函数依赖集左部) 设属性集 $A\in R$，r 中函数依赖集($dep(r)$)的左部集定义为 $lhs(dep(r),A)=\{A\subseteq R\mid r\models X\rightarrow A$ and $\forall X'\subset Y,r\not\models X'\rightarrow A\}$。

根据文献[2]中关于 $dep(r)$ 的定义可知，$\{X\rightarrow A\mid X\in lhs(dep(r),A),A\in R\}$ 与 $dep(r)$ 等价。

3 挖掘近似函数依赖关系

近似函数依赖关系提取的基本思想是，先计算一致集 $ag(r)$，根据 $ag(r)$ 计算最大集 $\max(dep(r),A)$ 和它的补集 $cmax(dep(r),A)$，由 cmax 集算出 $lhs(dep(r),A)$ 进而提取出近似函数依赖关系，其中只有计算 $ag(r)$ 与元组集合 r 有关。因为 $ag(r)=\{ag(t,t')\mid t,t'\in r,t\neq t'\}$，若 r 中有 m 个元组，则不同元组的配对数达到 $O(m^2)$，计算每对元组 t,t' 的 $ag(t,t')$ 需要 $O(n)$，n 为 R 中的属性个数，这样朴素计算 $ag(r)$ 的时间复杂度达到 $O(nm^2)$(没有考虑数据内外存交换的时间)。所以，当 m 足够大时，计算 $O(m^2)$ 就变得不实际了。本文利用了一种计算 $O(m^2)$ 的新方法[8]，并给出了对其结论的证明。在不影响计算结果的前提下，其减少了候选的数据单元对，把对计算 $O(m^2)$ 无贡献的元组对排除掉，这种方法使用带状划分数据库(stripped partition database)[5,8] 的概念实现这一目的。

3.1 带状划分数据集

设 $X\subseteq R$，$t\subseteq r$，$[t]_x=\{t'\in r\mid\forall A\in X\}$，则 $[t]_x$ 是 X 的一个基本等价类。X 在 r 中形成的基本等价类的集合构成 r 的一个划分(partition)，表示为 $X^*=\{[t]_x\mid t'\in r\}$。若划分 X^* 中的一个等价类只有一个元组，则 r 中没有与它在 X 上一致的其它元组。所以，定义属性集 X 的带状划分(stripped partition) $\overline{X}^*=\{c\in X^*\mid\mid c\mid>1\}$。同样，当划分在两类以上时，才去考虑匹配，因为一类的时候，就相当于把列表中所有的元组全部重新匹配了一遍，在其它划分中已经做了这个工作，不需要再重复进行。

3.2 计算一致集

先定义一个最大等价类的集合 $MC=\mathrm{Max}\subseteq\{c\in\overline{A}^*\mid\overline{A}^*\in\overline{r}\}$，其中算子“Max$\subseteq$”是求集合中按“$\subseteq$”最大的成员。

定理 1 $ag(r)=\bigcup_{c\in MC}ag(c)$，其中，$ag(c)=\{ag(t,t')\mid t,t'\in c,t\neq t'\}$，即所有记录一致集的计算可以用下面的方法求出：

(1) 求出每两条记录 t 和 t' 的一致集；

(2) 求出(1)的结果的并。

下面给出定理 1 的证明。

证明：设 $t,t'\in r,t\neq t'$，若 $ag(t,t')\neq\emptyset$，设 $A\in ag(t,t')$。

$\because t[A]=t'[A]$，$\therefore\exists c'\in\overline{A}^*$，$t,t'\in c'$。

根据 MC 的定义，$\exists c\in MC,c'\subseteq c$，$\therefore t,t'\in c$。

$\therefore ag(r)=\bigcup_{c\in MC}ag(c)$。

根据定理 1 知，当计算 $ag(r)$ 时，只需考虑 MC 等价类中配对的元组。当 r 中属性的取值大不相同时，通过带状划分数据集 $\overline{r}$ 去除掉很多单个元组组成的等价类，使其不再与其它元组配成对，这样计算 $ag(r)$ 的时间将大大减少，这是与 Tane[5] 算法提取函数依赖关系的不同之处。

3.3 计算最大集及其补集

为了计算最大集 $\max(dep(r),A)$ 和它的补集 $cmax(dep(r),A)$，给出定理 2。

定理 2 设 $A\in R$，$\max(dep(r),A)=\text{Max}\subseteq\{X\in ag(r)\mid A\notin X\}$。

证明：设 $X\in\text{Max}\subseteq\{X\in ag(r)\mid A\notin X\}$。

$\because\exists t,t'\in r, ag(t,t')=X,\therefore t[X]=t'[X]$。

又$\because A\notin X,\therefore A\notin ag(t,t')$。

$\therefore r\mid\neq X\rightarrow A$。

对 X 的任意一个超集 $X'\subseteq R$，若 $A\in X'$，$r\mid\neq X'\rightarrow A$；若 $A\notin X'$，则根据 X 的性质，$X'\notin ag(r)$，所以 $\forall t,t'\in r, t\neq t', t[X']\neq t'[X'],\therefore r\mid\neq X'\rightarrow A$。所以，$X\in\max(dep(r),A)$。另一方面，设 $X\in\max(dep(r),A)=\{X\mid r\mid\neq X\rightarrow A, X$ 的任意一个超集 $X'\subseteq R, r\mid\neq X'\rightarrow A\}$。

$\because r\mid\neq X\rightarrow A$。

$\therefore A\notin X$，并且 $\exists t,t'\in r, X\subseteq ag(t,t')$。

若 $\exists B\in ag(t,t'), B\notin X$，则 $r\mid\neq BX\rightarrow A$，这与 X 的性质矛盾，所以 $X=ag(t,t')$。同时，由 X 的最大集性质知，不存在 X 的超集 $X'\subseteq R, X'\in ag(r)$ 且 $A\notin X'$，所以 $X\in\text{Max}\subseteq\{X\in ag(r)\mid A\notin X\}$。

3.4 求近似依赖函数左部

为了计算函数依赖集左部 $lhs(dep(r),A)$，需引进超图(hypergraph)概念。

定义 9(超图) 关系模式 R 的属性集构成一个简单的超图 H，H 中的属性集称为超图的边，每个属性称为超图的顶点；H 的一个横截(transversal) T 是 R 的一个属性集，它与 H 的每一条边都相交；H 的一个最小横截 T 满足：T 是 H 的一个横截，$\forall T'\subset T$，T' 不是 H 的横截。H 的所有最小横截集记为 $T_r(H)$。把 $cmax(dep(r),A)$ 看成一个超图，则有结论：$(cmax(dep(r),A))=lhs(dep(r),A)$。

基于超图，给出计算 $lhs(dep(r),A)$ 的算法描述。

算法 1 求解近似函数依赖左部算法

输入：cmax(dep(r))

输出：最小函数依赖集的左部 lhs(dep(r))

```
1. for ∀A∈R do
2.   begin i=1;
3.     L_i={{B}|B∈X,X∈cmax(dep(r),A)};
4.       while L_i≠∅ do
5.         begin
6.           LHS_i[A]={l∈L_i|l∩X≠∅, ∀X∈cmax(dep(r),A)};
7.         L_i=L_i-lhs_i[A];
8.         L_{i+1}={l' | |l'|=i+1 and ∀l⊂l', |l|=i⇒l∈L_i};
9.         i=i+1;
10.       end
11.    lhs (dep(r),A)=∪lhs_i[A];
12. end
13. return lhs (dep(r))
```

该算法对于 R 中所有的属性 A，开始时令 $i=1$，让 L_i 取所有的 B 属性，B 属性在 $cmax(dep(r),A)$ 中(step1－3)；开始循环条件：L_i 不为空的时候开始(step4)；step5－6，$lhs_i[A]$ 取所有的符合下面条件的 l：其中，l 是 L_i 中的元素，并且对于每一个在 $cmax(dep(r),A)$ 中的属性集 X，l 与 X 的交集都不为空；step7，L_i 变为 $L_i-lhs_i[A]$；step8－9，L_{i+1} 集合的元素是 L_i 中的 l_i 增加一个属性得到的 l_{i+1}，同时满足 l_{i+1} 分成的每一个 l_i 必须都在 L_i 中，然后 $i+1$；step10，当 L_i 为空时，算法结束；step11－12，返回 $lhs(dep(r),A)$，它是所有 $lhs_i[A]$ 的并集。

4 求近似候选码

为了求得给定关系模式的码，从一组函数依赖求得蕴含的函数依赖，例如已知函数依赖 F，要判断 $X\rightarrow Y$ 是否为 F 所蕴含，就需要一套推理规则，这组推理规则于 1974 年由 Armstrong[9] 首先提出。它包括自反律、增广律和传递律。由自反律得到的函数依赖均是平凡的函数依赖，自反律的使用并不依赖于 F。

定义 10(闭包) 设 X 为关系模式 $R\langle U,F\rangle$ 中的任一属性(组)，记 $X^+=\{A\mid X\rightarrow A$ 能由 Armstrong 系统从 F 中导出$\}$，则称 X^+ 为属性(组)X 关于函数依赖集 F 的闭包。属性(组)X 的闭包 X^+ 可通过算法来求解。

定义 11(候选码) 设 K 为关系模式 $R\langle U,F\rangle$ 中的属性(组)，$K^+=U$，且对任一属性 $A\in K$，有 $(K-A)^+\neq U$，则称 K 为 R 的一个候选码。

一个关系模式的候选码 K 有以下两个特性。

唯一性：在任一给定时间，关系 r 的任意两个不同元组，其属性集 K 的值是不相同的。设有关系模式 $R\langle U,F\rangle$，其中 $U=k\bigcup WU, X\in W$ 为关系模式 R 的一个非主属性；并且在关系 r 中存在这样两个不同元组，其属性集 K 的值是相同的，但在属性 X 上的值不同。由函数依赖的定义可知，属性集 K 不能通过函数决定属性 X，从而 K 不是关系模式 R 的候选码。

最小性：属性集 U 中的任一属性都不能从集合 K 中删掉。若能从属性集 K 中去掉某一属性 X，且有 $(K-X)\xrightarrow{f}U$，就有 $K\xrightarrow{p}U$，从而 K 不是关系模式 R 的候选码。

例如，设有关系模式 $R\langle U,F\rangle$，式中 $U=\{A,B,C,D\}$；$F=\{A\rightarrow B, C\rightarrow D\}$。

可以看出，R 的唯一候选码是 (AC)。下面在关系模式 R 上的函数依赖集 F 为最小的函数依赖集的基础上，求出候选码。对于关系模式 R 中的属性集 U，对其做出如下划分：U_L 表示仅在函数依赖集中各依赖关系式左边出现的属性集合，U_R 表示仅在函数依赖集中各依赖关系式右边出现的属性集合。令 $U_B=U-U_L-U_R$，当 $U_B=\emptyset$ 时，表示函数依赖关系式左右边都出现的属性集合。例如，从上面给出的函数依赖关系可以得出，$U_L=\{AC\}$，$U_R=\{BD\}$，$U_B=\emptyset$。根据文献[10]，可得出如下结论：

(1) 设有关系模式 $R\langle U,F\rangle$，$X\in U$，若 $X^+=U$，则 X 中必定包含关系模式 R 的一个候选码；

(2) 设有关系模式 $R\langle U,F\rangle$，若 U_L 非空，则 U_L 中任一属性必包含在关系模式 R 的候选码中；

(3) 设有关系模式 $R\langle U,F\rangle$，若 U_R 非空，则 U_R 中的任一属性必定不包含在关系模式 R 的任一候选码中；

(4) 设有关系模式 $R\langle U,F\rangle$，若 U_L 非空，且 $U_L^+=U$，则 U_L 为关系模式 R 的唯一候选码。

根据上面的结论，可得出如下求解候选码的具体步骤，算法描述如下。

算法 2 求解候选码算法

1. 求关系模式 R〈U,F〉的最小函数依赖集 F;
2. 按照上面的结论,分别计算出 U_L,U_R,U_B;
3. 若 $U_L^+ = U$,则 U_L 为 R 的唯一候选码,算法结束;若 $U_L^+ \neq U$,转到 4 步;若 $U_L = \emptyset$,转到 5 步;
4. 将 U_L 依次与 U_B 中的 1 个、2 个直到所有的属性组合成新的属性集合,利用上述结论判断该属性组合是否是候选码;找出所有的候选码后,算法结束;
5. 对 U_B 中的属性及属性组合利用上述结论依次进行判断,找出所有的候选码后,算法结束。

在实际求解过程中,算法中步骤 4 和 5 最多重复 2^n-1 次,其中 n 为 U_B 中的属性个数。若为 U_B,则为最坏的情况,采用以下 3 个原则进行判断:

(1) 在关系模式 R 中,若 A 为码,且 $A \longleftrightarrow B$,则 B 必为码;

(2) 在关系模式 R 中,若 A 不为码,且 $A \longleftrightarrow B$,则 B 必不为码;

(3) 在关系模式 $R\langle U,F\rangle$ 中,若 K 为码,$W \subseteq U$,且 W 中不包含 K 的任一属性,则 KW 必不为码,而是超码。

在候选码中,计算所有候选码的支持度(根据式(1))及其误差(根据式(2)),得到最后的近似关键字。

$$\text{support}(X \rightarrow Y) = \frac{|T(X \cup Y)|}{|T|} \tag{1}$$

$$T_{err}(X \rightarrow A) = 1 - \text{support}(X \rightarrow A)/|r| \tag{2}$$

5 属性重要程度排序和权重评估

根据依赖集和决定集,给出属性重要程度排序算法——AIR(Attribute Importance Ranking)算法。

算法 3 属性重要程度排序

输入:关系模式 R,R 的实例 r,误差阈值 T'_{err}

输出:属性松弛顺序排序序列

1. $S_{AFD} = \{x \mid x \in \text{GetAFDs}(R,r), T_{err}(x) < T'_{err}\}$;
2. $S_{AK} = \{x \mid x \in \text{GetAKeys}(R,r), T_{err}(x) < T'_{err}\}$;
3. $AK = \{k \mid k \in S_{AK}, \forall k' \in S_{AK}, \text{support}(k) \geqslant \text{support}(k')\}$;
4. $\overline{AK} = \{k \mid k \in R - AK\}$;
5. $\forall k \in AK$
6. $W_{t_{decides}}(k) = \sum \frac{\text{support}(\hat{A} \rightarrow k')}{\text{size}(\hat{A})}$, where $k \in \hat{A} \subset R, k' \in R - \hat{A}$;
7. $W_{t_{AK}} = W_{t_{AK}} \cup [k, W_{t_{decides}}(k)]$;
8. $\forall j \in \overline{AK}$
9. $W_{t_{depends}}(j) = \sum \frac{\text{support}(\hat{A} \rightarrow j)}{\text{size}(\hat{A})}$;
10. $W_{t_{\overline{AK}}} = W_{t_{\overline{AK}}} \cup [j, W_{t_{depends}}(j)]$;
11. return $[\text{sort}(W_{t_{AK}}), \text{sort}(W_{t_{\overline{AK}}})]$。

算法 3 首先提取出所有可能的近似函数依赖和近似候选码(step1—2),其中 S_{AFD} 是 R 中所有的近似函数依赖关系集合,S_{AK} 是所有的近似候选码集合;然后,从 S_{AK} 中选择具有最大支持度的候选码作为关键字,关键字将 R 中的属性集划分成决定集(即 AK)和依赖集(即 $\overline{AK}$)两部分(step3—4);接着,求出决定集 AK 中每个属性 k 的重要程度(step5—7),其计算方法是:对于每个包含了 k 的候选码 $\hat{A}$,统计出 $\hat{A}$ 对所有在 $R-\hat{A}$ 中的属性 k' 的支持度,求出支持度与 $\hat{A}$ 中属性个数的比值(这里,size($\hat{A}$)为 k 所在的近似函数依赖关系左部 $\hat{A}$ 中的属性个数,$\hat{A}$ 取值为所有包含了属性 k 的近似函数依赖关系的左部);最后把所有的比值求和。同理,对于依赖集 $\overline{AK}$ 中的所有属性,计算其中每个属性对于决定属性的依赖程度:对于所有候选码 $\hat{A}$,统计出它对第 j 个属性的支持度与 $\hat{A}$ 中的属性个数的比值再求和(step8—10);最后,分别按照支持度和依赖度,降序返回依赖集和决定集中的属性序列,决定集中的属性序列在依赖集之前(step11)。误差阈值 T'_{err} 可由领域专家确定。

根据属性序列、决定集中的属性支持度和依赖集中的属性依赖度,可为关系中的每个属性统一分配一个权重。对于决定集中的每个属性 $k \in W_{t_{AK}}$,其权重根据式(3)计算:

$$W(k) = \frac{RelaxOrder(k)}{count(Attributes(R))} \times \frac{W_{t_{decides}}(k)}{\sum W_{t_{decides}}} \tag{3}$$

式中,$RelaxOrder$ 代表属性 k 在属性序列中的位置,最不重要的属性位置为 1,最重要属性的位置为关系 R 中的属性总数。同理,对于依赖集中的每个属性 $k \in W_{t_{\overline{AK}}}$,其权重的计算只需将式(3)中的 $W_{t_{decides}}$ 用 $W_{t_{depends}}$ 替换即可。

6 性能实验评价

本实验的目的是测试在数据集大小发生改变时,属性重要程度排序算法——AIR 的性能变化情况,性能测试主要包括 AIR 算法的执行时间、稳定性和合理性 3 个方面。

6.1 测试数据集

测试数据使用了来自两个不同应用领域的数据集,分别介绍如下。

(1) 二手车数据集

从 Yahoo!Autos 网站[10]随机抽取 100000 条二手车信息记录,合成关系表 CarDB(Make, Model, Year, Color, Trans, Price),其中 Make、Model、Trans 和 Color 是分类型属性,Price 和 Year 是数值型属性。

(2) 房产数据集

从 MSN 网站[13]随机抽取 25000 条房产信息记录,合成关系表 HouseDB(City, Price, SqFt, Bedrooms, Bathrooms, Schooldistrict, View, Neighborhood, Buildyear, Garage, Boatdock),其中 City、Schooldistrict、View、Garage、Boatdock 和 Neighborhood 是文本型属性,Price、SqFt、Bedrooms、Bathrooms 和 Buildyear 是数值型属性。

6.2 实验结果与分析

(1) AIR 算法的执行时间

从 HouseDB 和 CarDB 数据集中分别提取 2M 和 4M 大小的数据量,在其上测试 AIR 算法的执行时间。从表 1 可以看出,虽然 HouseDB 的数据量比较小,但由于它所包含的属性个数比 CarDB 多,因此 HouseDB 数据集的近似函数依赖关系提取时间也不短。

表 1 数据集 HouseDB 和 CarDB 的 AIR 算法执行时间

	HouseDB Dataset	CarDB Dataset
AIR algorithm	2.8 min	3.4 min

(2) AIR 算法的稳定性

对于 AIR 算法的稳定性评价，主要看算法是否受数据集大小变化的影响。评价标准是根据依赖集中的属性对于近似关键字的依赖度，即计算近似关键字对于每个依赖集属性的支持度随数据集大小的变化情况，这可以衡量近似关键字和依赖集属性两方面的稳定性。

对于 CarDB 数据集，属性序列为(1：Trans，2：Make，3：Model，4：Year，5：Color，6：Price)。在数据量大小分别取 1M、2M 和 4M 时，计算近似关键字对依赖集中各属性的支持度，计算结果变化情况如图 1 所示。

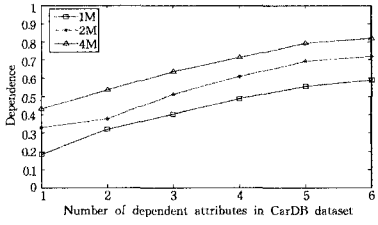

图 1　CarDB 数据集上 AIR 算法确定的属性顺序稳定性

对于 HouseDB 数据集，属性序列为(1：Garage，2：Boatdock，3：City，4：View，5：Neighborhood，6：Schooldistrict，7：Bedrooms，8：Bathrooms，9：Buildyear 10：SqFt，11：Price)。在数据量大小分别取 1M、2M 和 4M 情况下，计算近似关键字对依赖集中各属性的支持度，计算结果变化情况如图 2 所示。

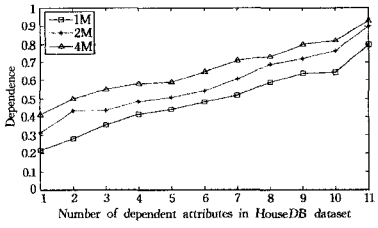

图 2　HouseDB 数据集上 AIR 算法确定的属性顺序稳定性

从图 1 和图 2 可以看出，随着数据集大小的增加，各属性的重要程度保持不变；而且，随着数据集大小的增加，依赖集中各属性对于近似关键字的依赖度是逐渐增加的。由此可见，属性重要程度排序算法具有很好的稳定性。

(3) AIR 算法的合理性

在 CarDB. xml 和 HouseDB. xml 文档上各提取 10000 条数据记录，在其上运行 AIR 算法，得到各数据集中所有属性的重要程度排序。

对于 CarDB，属性重要程度排序的结果是 Make＞Model＞Price＞Year＞Color＞Trans，即 Make 是 CarDB 中最重要的属性，当 Make 的值发生变化时，记录中其他属性上的值也很可能随之发生变化。实际上，上述属性重要程度顺序也是合理的，人们在购买二手车过程中通常首先考虑 Make、Model 和 Price。

对于 HouseDB，属性重要程度排序的结果是 SqFt＞Price＞City＞Schooldistrict＞Buildyear＞Bedrooms＞View＞Neighborhood＞Garage＞Bathrooms＞Boatdock。显然，上述排序结果与现实情况较为接近，面积(SqFt)、价格(Price)、城市(City)和学区(Schooldistrict)对于购房者来说通常是较为重要的；另外，这些重要属性上的数据一旦发生变化，将会对其他属性上的数据产生较大影响，例如面积(SqFt)发生变化，将会直接影响到价格(Price)、卧室数(Bedrooms)、卫生间数(Bathrooms)等属性上的数据。

上述实验结果和分析可以表明，本文方法对属性重要程度评估的合理性。

结束语　本文提出了一种基于近似函数依赖的关系数据属性权重评估方法。该方法基于一致集的概念导出最大集，生成最小非平凡函数依赖集，从而找出属性之间的近似函数依赖关系，进而求出近似候选码和近似关键字，并在此基础上根据属性支持度计算属性权重。实验结果和分析表明，提出的属性权重评估方法能够合理地对关系中的属性进行重要程度排序并计算属性权重，算法具有较好的稳定性和较高的执行效率。将该方法应用于半结构化数据中的近似函数依赖关系挖掘是进一步的研究工作。

参 考 文 献

[1] Zaniolo C. Analysis and design of relational schemata for database systems [D]. University of California, 1976

[2] Maier D. The theory of relational databases [M]. Rockville, Md: Computer Science Press, 1983

[3] Mannila H, Räihä K-J. Algorithms for inferring functional dependencies from relations [J]. Data and Knowledge Engineering, 1994, 12(1): 83-99

[4] Savnik I. Bottom-Up induction of functional dependencies from relations [C] // Proceedings of the AAAI'93 Workshop on Knowledge Discovery in Databases. 1993: 174-185

[5] Huhtala Y, Kärkkäinen J, Porkka P, et al. Tane: an efficient algorithm for discovering functional and approximate dependencies [J]. The Computer Journal, 1999, 42(2): 100-111

[6] Zhang S Z, Shi B L. A method for discovering functional dependencies and its application [J]. Journal of Software, 2003, 14(10): 1692-1696

[7] Beeri C, Dowd M, Fagin R, et al. On the structure of armstrong relations for functional dependencies [J]. Journal of the ACM, 1984, 31(1): 30-46

[8] Lopes S, Petit J M, Lakhal L. Efficient discovery of functional dependencies and Armstrong relations [C] // Proceedings of the International Conference on Extending Database Technology. 2000: 350-364

[9] Sa S X, Wang S. Introduction to database systems [M]. Beijing: Higher Education Press, 2000

[10] Beeri C, Fagin R, Howard J H. A complete axiomatization for functional and multivalued dependencies [C] // Proceedings of the 1977 ACM SIGMOD International Conference on Management of Data. 1977: 82-93