维普资讯 http://www.cqvip.com

计算机科学1995 Vol. 22 №. 3

关系模型中不确定信息与可能信息的引入与处理

严 丽 马宗民 薛劲松
（辽宁工学院） （中国科学院沈阳自动化所）

摘 要 In this paper, the traditional relational model is extended in order to express indefinite and maybe information. On the basis of the extended relational model, the foundamental operations in relational algebra are defined again, and the policy and algorithm for updating relational database are given.

关键词 Incomplete relational database, Indefinite information, Maybe information, Relational algebra, Update.

一、引 言

传统关系模型下的关系数据库所表示的信息都是确定信息，即元组所描述的实体确确实实被包含在关系数据库所定义的现实世界中，即使是含有空值及模糊值的不完全关系数据库，元组与相应的数据库之间也存在着这种包含关系。实际上不完全关系数据库除了在属性值上有不确定性外（如空值及模糊值），还有另外一种不确定性，这就是元组级上的不完全信息：不确定信息（indefinite information）和可能信息（maybe information）。这两种元组级上的不完全信息所描述的实体或实体集合与相应关系数据库限定的现实世界不存在完全确定的包含关系。

二、基本定义及扩展关系模型

1. 基本定义

定义1 两个或两个以上元组构成的集合称作扩展元组。

定义2 若扩展元组中至少有一个元组（但不是全部）确切属于关系，且当前无法确定是哪一个元组属于关系，则称该扩展元组属于关系。

定义3 设有关系 r 及扩展元组 ω，若有 ω 属于 r，则称 ω 是 r 的不确定信息。

定义4 设有关系 r 及元组 t，若有 t 是否属于 r 当前未知，则称 t 是 r 的可能信息。

传统关系模型只能记录确定信息，若要把这两类不完全信息引入关系模型，则必须对原有关系模型进行扩展，并重新定义信息处理规则。

2. 扩展关系模型

当在关系模型中引入不确定信息和可能信息之后，关系数据库中含有三类信息：不确定信息，可能信息及确定信息，为此需要引入三个表分别记录它们。记录确定信息的表叫 D 表，记录不确定信息的表叫 I 表，记录可能信息的表叫 M 表。D 表及 M 表的内容是元组的集合，I 表中的内容是扩展元组的集合。

3. 信息冗余及处理

扩展关系模型下关系数据库有两类信息冗余：各表内的信息冗余及各表之间的信息冗余。D 表及 M 表内的信息冗余是指表内有相同的元组存在，消除冗余的方法是去除副本信息；而 I 表中两扩展元组 ω_1 和 ω_2 若存在 $\omega_1 \subseteq \omega_2$ 的关系，则 ω_1 和 ω_2 之间有冗余，消除冗余的方法是从 I 中去除 ω_2，并把 $\omega_2 - \omega_1$ 并入 M 表。

各表之间的信息冗余及处理策略如下：

a）元组 $t \in R_D$，扩展元组 $\omega \in R_I$ 且 $t \in \omega$。从 R_I 中去除 ω，并把 $\omega - (t)$ 并入 R_M。

b）元组 $t \in R_D$ 且 $t \in R_M$。从 R_M 中去除 t。

c）扩展元组 $\omega \in R_I$，元组 $t \in R_M$ 且 $t \in \omega$。从 R_M 中去除 t。

由于篇幅的限制，去除扩展关系中冗余信息的算法文中不具体给出，处理算法名是 reduce(R_x)，它的时间复杂度是 O(mn)，其中的 m 和 n 分别是扩展关系三表中最大元组数和最大扩展元组数两数中的

严 丽 讲师，主要从事数据库理论的研究与教学工作。马宗民 助理研究员 主要研究兴趣是数据库理论与应用，人工智能。

维普资讯 http://www.cqvip.com

最大值和次大值。

三、基本关系代数运算

1. 集合并

设扩展关系模型下的两个关系 R_x 及 S_x，令 $T_x=R_x\cup S_x$，则有 $T_D=R_D\cup S_D$，$T_I=R_I\cup S_I$，$T_M=R_M\cup S_M$，$T_X=$reduce(T_X)。

2. 选择

选择运算是通过同时在三个表上分别进行运算完成的。对 D 表及 M 表的选择与传统二维表上的选择完全相同，选中的元组依据元组的性质进入结果关系中的 D 表或 M 表。I 表上选择处理的规则是：I 表中的扩展元组 ω，若 ω 的全部构成元组均满足选择条件，则 ω 进入结果关系中的 I 表，若 ω 中只有部分元组满足选择条件，则这部分元组进入结果关系中的 M 表。

算法1 Select(R_X,F)

输入：扩展模型下的关系 R_X 及选择条件 F；

输出：结果关系 S_X；

```
S_D:=σ_F(R_D);S_M:=σ_F(R_M);S_I:=∅;
for i:=1 to k do {k 是 R_I 中扩展元组数}
    θ:=σ_F(ω_i);if θ=ω_i then S_I:=S_I+{θ}
        else if θ≠∅ then S_M:=S_M+θ;
next i. {ω_i∈R_I}
```

说明：$\sigma_F(R)$是传统二维表上的选择运算。算法1的时间复杂度是 O(m)，其中的 m 是扩展元组数与扩展元组含最大元组数乘积、D 表及 M 表中元组数三数中的最大值。

3. 投影

投影是通过在三个表的指定属性(集合)上分别进行投影而完成的。投影运算可能产生信息冗余，要予以消除。

算法2 Project(R_X,Y)

输入：扩展关系模型下的关系 R_X 及属性(集)Y；

输出：结果关系 T_X；

```
T_D:=Π_Y(R_D);T_M:=Π_Y(R_M);T_I:=∅;
for i:=1 to k do {k 是 R_I 中扩展元组数}
   θ:=Π_Y(ω_i);{ω_i∈R_I}
  if(θ 中只有一个元组)then T_D:=T_D+θ else T_I
       :=T_I+{θ};
next i;reduce(T_X).
```

说明：算法中的 $\Pi_Y(R)$是传统二维表上的投影运算。该算法的时间复杂度与算法1相同。

4. 集合差

两关系之间的差是通过相同性质表之间分别进行差完成的。D 表及 M 表之间的差运算与经典关系中的差运算完全相同；I 表间的差运算规则是，任两个分属两个 I 表的扩展元组 ω_i 和 θ_j，若 $\omega_i-\theta_j$ 与 ω_i 不相等，则 $\omega_i-\theta_j$ 进入结果 M 表，而若 $\omega_i-\theta_j$ 等于 ω_i，则 $\omega_i-\theta_j$ 进入结果 I 表。

算法3 Difference(R_X,S_X)

输入：扩展关系模型下的两个关系 R_X 和 S_X；

输出：R_X 和 S_X 差运算后的结果关系 T_X；

```
T_D:=R_D-S_D;T_M:=R_M-S_M;T_I:=∅;
for i:=1 to m do {m 是 R_I 中的扩展元组数}
  for j:=1 to n do {n 是 S_I 中的扩展元组数}
    ω:=ω_i-θ_j;{ω_i∈R_I,θ_j∈S_I}
    if ω=ω_i then T_I:=T_I+{ω}else T_M:=T_M+ω;
  next j;
next i.
```

该算法的时间复杂度是 O(k*l*mn)，其中的 k 和 *l* 是两关系中的扩展元组数，m 和 n 分别是扩展元组所含元组的最大个数。

5. 笛卡尔积

设有两扩展关系 R_X 和 S_X，令 $T_X=R_X\times S_X$，则有：$T_D=R_D\times S_D$；$T_I=R_D\times S_I+R_I\times S_D$；$T_M=R_D\times S_M+R_M\times S_D+R_M\times S_M+R_M\times S_I+R_I\times S_M+R_I\times S_I$。

上述各表达式中需要说明的是 D 表或 M 表与 I 表及 I 表与 I 表之间笛卡尔积的运算规则。

算法4 Cartesion(R,S)

输入：R 是 D 表或 M 表、I 表，S 是 I 表；

输出：R 与 S 笛卡尔积 T；

```
T:=∅;
if(R 是 M 表)then
  for i:=1 to k do {k 是 S 中扩展元组数}
  X:=R×ω_i;T:=T+X {ω_i∈S}
  next i;
if(R 是 D 表)then
  for i:=1 to k do
    for j:=1 to l do {l 是 R 中的元组数}
    θ:={t_j}×ω_i;T:=T+{θ};{t_j∈R,ω_i∈S}
    next j
  next i;
if(R 是 I 表)then
  for i:=1 to k do {k 是 R 中扩展元组数}
     for j:=1 to m do {m 是 S 中扩展元组数}
     Y:=ω_i×θ_j;T:=T+Y;{ω_i∈R,θ_j∈S}
     next j
  next i.
```

笛卡尔积的时间复杂度由算法4的时间复杂度决定，与算法3的时间复杂度相同。

基本关系代数中的5种运算在关系代数中是最小完备的，其它的运算均可由它们推导得出，所以文中只给出基本关系代数的算法描述。

四、扩展关系模型下关系数据库的更新

1. 插入操作

由于扩展关系模型下关系中含有三类不同性质的信息，因此插入操作就有三类信息的插入，插入信

维普资讯 http://www.cqvip.com

息根据其性质进入对应的表中，但不论进行哪种性质信息的插入，插入操作的策略相同，即先检查插入信息的合法性，避免插入操作破坏关系的一致性，之后是消除插入操作可能引起的信息冗余。

现在给出进行插入信息合法性检查的算法。

算法5 Check(R_X,p,FDS)

输入：扩展关系模型下的关系 R_X，插入信息 p，FD 集 FDS；

输出：标志变量 flag；

```
Procedure Proc1(R,s,FDS);
[flag1:=false;
  for i:=1 to k do {k 是 R 中的元组数}
    for j:=1 to l do {l 是 FDS 中 FD 的个数}
    if(t_i[X]=S[X])and(t_i[Y])=S[Y]then flag1
        :=true
    else [flag1:=false;return];{X→Y∈FDS,
                                  t_i∈R}
    next j
  next i];
Proc2(R_X,s,FDS);
  [Proc1(R_D,s,FDS);
  if(¬flag1)then[flag2:=false;return]else Proc1
    (R_M,s,FDS);
  if(¬flag1)then [flag2:=false;return]else
    for i:=to k do {k 是 R_I 中扩展元组数}
    Proc1(ω_i,s,FDS);
    if(¬flag1)then [flag2:=false;return];
    next i;
  flag2:=flag1];
if(p 是确定信息 OR p 是可能信息)then
  [Proc2(R_X,p,FDS);flag:=flag2];
if(p 是不确定信息)then
[for j:=1 to l do {l 是 p 中的元组数}
  Proc2(R_x,t_j,FDS);if(¬flag2) then[flag:=false;
      return];{t_j∈p}
next j;flag:=flag2].
```

该算法的时间复杂度是 O(kmn)，其中 k 是插入信息含元组个数，n 是 FD 集中 FD 的个数，m 与算法1时间复杂度中的 m 相同。为减少算法中的比较次数，文中的 FD 集假定已化成了最小覆盖集(以下相同)。

下面给出插入操作的实现算法。

算法6 Insert(R_X,p)

输入：扩展关系模型下的关系 R_X，插入信息 p 及 FD 集 FDS；

输出：插入操作完成后的 R_X；

```
Check(R_X,p,FDS);if(¬flag) then return;
if(p 是确定信息)then R_D:=R_D+{p}
else if(p 是可能信息)then R_M:=R_M+{p}
    else R_I:=R_I+{p};reduce(R_X).
```

Insert 算法的时间复杂度同 Check 算法。

2. 删除操作

删除操作是对关系中三类信息进行条件删除的，其中对 D 表及 M 表的删除操作与经典关系数据库下的删除操作完全相同；对 I 表的删除策略是：把扩展元组中满足删除条件的元组删除掉，此时若扩展元组非空，则剩余元组进入 M 表；若扩展元组中无元组满足删除条件，则该扩展元组保留。

算法7 Delete(R_X,F)

输入：扩展关系模型下的关系 R_X，删除条件 F；

输出：删除操作完成后的 R_X；

```
Select(R_X,F);Difference(R_x,S_X);R_X:=T_X;
for i:=1 to m do {m 是 R_I 中扩展元组数}
  if σ_F(ω_i)≠∅ then[R_M:=R_M+ω_i-σ_F(ω_i);R_I:=
    R_I-{ω_i}];{ω_i∈R_I}
next i.
```

该算法的时间复杂度同算法3的时间复杂度。

3. 修改操作

修改操作是对关系中满足条件的信息进行修改，且修改后的信息应不违背完整性约束及不出现信息冗余，在实现上是通过对三个表分别进行修改操作而完成的。

算法8 Modify(R_X,F_1,F_2)

输入：扩展关系模型下的关系 R_X，被修改信息满足的条件 F_1，修改后信息满足的条件 F_2 及 FD 集 FDS；

输出：修改后的关系 R_X；

```
Procedure modi(r,F1,F2);
[r':=σ_F1(r);
for i:=1 to n do{n 是 r 中的元组数}
  check(R_X,t_i,FDS);{t_i∈r'}
  if(¬flag)then Loop else[S:=t_i 依 F2的修改结
    果;r':=r-{t_i}+{s}];
next i];
modi(R_D,F1,F2);modi(R_M,F1,F2);
for j:=1 to k do{k 是 R_I 中扩展元组数}
  modi(ω_i,F1,F2);{ω_i∈R_I}
next j;reduce(R_X).
```

该算法的时间复杂度同 reduce(R_X)的时间复杂度。

参考文献

[1] Ken-chin Liu and R. Sunderraman, Indefinite and Maybe Information in Relational Database, ACM TODS, 5:1(1990)

[2] 马宗民，郝忠孝，空值环境下关系数据库的更新—第一部分：扩展关系模型及基本运算，计算机学报，17:7(1994)

[3] 马宗民，严丽，空值环境下含不确定及可能信息关系数据库的更新，计算机研究与发展(已录用)