面向对象数据库的推理查询实现*

张成洪　施伯乐　胡运发

（复旦大学计算机科学系　上海　200433）

摘要　上文介绍了面向对象数据库系统FOOD的推理查询语言O－Datalog，本文继续讨论对O－Datalog程序的几种变换，并证明这些变换是语义等价的，从而证明了对于一个O－Datalog程序，可以为它构造一个相应的Datalog程序，并能利用该Datalog程序对原程序进行计值．最后本文还给出了对O－Datalog程序计值的算法．

关键词　面向对象数据库系统，推理查询，Datalog．

目前关于面向对象数据模型与逻辑语言集成的研究很多[1]，但成功的系统并不多见．这些研究一类是从知识库系统出发，扩充面向对象的能力，但它们一般对面向对象数据的存储和操纵的管理不足[2,3]；另一类是从研究面向对象数据库的逻辑描述出发，其规则语言一般是基于非Horn逻辑的，其语义一般都难于利用bottom－up计值方法实现，很难使用知识库中许多成熟的算法，其效率不理想（例如F－logic[4]）．

我们的研究是从面向对象数据库系统FOOD[5]出发，利用FOOD系统对面向对象数据进行管理，并增加对这些数据进行推理的功能，从而提出一种推理查询语言O－Datalog．O－Datalog语言在形式上是一种Datalog的扩充，对于任意一个O－Datalog程序，系统能为它构造一个相应的Datalog程序，该Datalog程序事实上是用规则表达了O－Datalog规则中隐含的许多面向对象概念和系统的有关存储信息，因此它们在语义上是等价的，可以利用该Datalog程序对原来的O－Datalog程序进行计值．由于人们已经对Datalog进行过深入的研究[6]，因此有许多成熟的优化策略和计值算法可以借鉴．

1　有关数据的存储方式

O－Datalog语言是对FOOD系统中的面向对象数据进行推理查询，因此O－Datalog的实现策略与FOOD系统的数据存储方式密切相关．

FOOD系统中每个对象都有一个对应的对象标识，通常还有对应值．一个对象的对象标识在系统范围内是唯一的，并独立于该对象的内容和存储位置．一个对象和其它对象之间

---

* 本文研究得到国家863高科技项目和国家自然科学基金资助．作者张成洪，1968年生，博士，主要研究领域为面向对象数据库，知识库．施伯乐，1935年生，教授，博士导师，主要研究领域为数据库，知识库，面向对象数据库．胡运发，1940年生，教授，博士导师，主要研究领域为知识库，知识工程，多媒体技术．

本文通讯联系人：张成洪，上海 200433，复旦大学计算机科学系

本文1995－07－10收到修改稿

的联系是表示为对象之间的引用,引用是把被引用对象的对象标识作为另一对象的属性值.

FOOD 系统的存储底层是基于关系数据库 Sybase 的.FOOD 系统中有一个对象管理模块负责对象的内外存格式转换,在用户新定义一个类后,系统分析该类的定义结构,将该类的定义对应为一个或若干个关系模式,其中有些模式在关系库中已存在,有些模式尚未定义过,系统在关系库中建立未定义过的模式.当用户新创建对象时,系统为每个对象产生一个对象标识,并且把对象标识和对象状态值存放在 Sybase 中,对应于一个或多个元组.

由于 FOOD 中把所有类组织成一个类层次,子类可以从超类中继承属性和方法,一个类可以有多个子类.对象数据存储有 2 种方式,一种是向上存储,即对象的一个属性,如果是从超类中继承的,其属性值存放在超类对应的关系中,如果是该类新定义的属性,则存放在该类对应的关系中;另一种是向下存储,即对象的所有属性值都存放在它所属类对应的关系中,超类中没有子类实例的数据,只有其直接实例的数据.FOOD 系统采用的是向下存储,主要因为在存取一个对象时,向上存储需要频繁的关系连接操作,向下存储不会因为属性继承而进行连接操作.当然,向下存储在对类查询时可能要涉及到整个类层次.

FOOD 系统的一些存储策略如下:①一个类直接对应于一个关系模式,在此模式中,除了类的所有属性之外,还增加了一个属性 lid,它作为主关键字,即 lid 的值能唯一确定一个元组.②当一个对象的属性值又是另外一个对象时(即对象引用的情况),把被引用对象的对象标识作为属性值存放.③如果一个类的一个属性的值域为数组型,那么这个属性名对应于一个关系模式中的多个属性,即每个数组元素对应于这个关系模式中的一个属性,因此一个对象的数组属性值是把每个数组元素存放到对应的列中.④如果一个类的一个属性的值域为集合型 Set(t),Sybase 中有一个对应的关系 t_set(set,element),存放集合名和相应的元素.系统把每个集合看成一个新的对象,生成一个对象标识,如果一个对象的某个属性值是一个集合,那么在该属性值对应的位置中存放这个集合的标识,而这个集合标识与集合元素的对应关系存放在另一个关系 t_set 中.⑤对于类层次,FOOD 中采用的是向下存储方式.⑥Sybase 中不存放方法,方法是一段类似 C++的程序代码,系统在方法调用的时候动态执行.

2　O−Datalog 程序的变换

2.1　从一般 O−Datalog 程序到简单 O−Datalog 程序

在一般 O−Datalog 程序中:①谓词的参数标签可以是方法调用.②谓词中的参数可以是另一个对象的属性值、方法调用或者嵌套的属性值、方法调用.③谓词的参数可以是另外一个谓词,即允许谓词嵌套.④谓词的参数标签可以是嵌套路径.⑤谓词的参数类型可以是数组型.⑥谓词的参数类型可以是集合型.⑦谓词允许带"*"号,表示关心该谓词对应的类的所有逻辑实例.⑧谓词中的参数标签可能重复,也可能只出现它所对应类的部分属性.

我们把没有上述 8 种情况的 O−Datalog 程序称为简单 O−Datalog 程序.下面讨论在保持语义的情况下,用其它方式表达前面 8 种情况,从而把 O−Datalog 程序变换成简单 O−Datalog程序.

我们首先引进一类特殊的"方法原子",形式为 *send*(*object*,*method*,*result*).当对象*object*

的方法调用 *method* 的返回值为 *result* 时，该原子为真，即对于解释 I 和变量赋值 v，$I\models_v send(object, method, result)$ *iff*

$$Iv(object).Iv(method)=Iv(result).$$

其次在以下的讨论中，假设 *variable* 代表一个不属于变量集合 V 和数组变量集合 *ArrayV* 的变量.

变换 1. 对于原子公式 $b(a_1:t_1, a_2:t_2, \cdots, a_n:t_n)$，记作 L. 若其中 $a_j(1\leqslant j\leqslant n)$ 为一个方法调用，如果把 $b(oid:variable, a_1:t_1, \cdots, a_{j-1}:t_{j-1}, a_{j+1}:t_{j+1}, \cdots, a_n:t_n)$ 记作 L'，$send(variable, a_j, t_j)$ 记作 A，则 L 可转换成 L'，A.

引理 1. 变换 1 是语义等价的.

证明：如果在变量集合 V 中新增加一些元素，变成 V'，变量赋值 $v(V)$ 和 $v'(V')$ 对于任何一个变量 $var\in V\cap V'$，$v(var)$ 和 $v'(var)$ 都相同，因此除了新增加的变量外，变量赋值 v 和 v' 完全一样，我们称 v' 与 v 相容. 引理 1 即：对于解释 I，变量赋值 v，存在一个与 v 相容的变量赋值 v'，$I\models_v L$ *iff* $I\models_v L'$，A. 下面从 2 个方向分别证明：

①若 $I\models_v L$，则存在 $oid\in I_C(b)$，使每个 $a_i:t_i$ 和 oid 满足文献[1]中的条件(Ⅰ，Ⅱ). 特别地，$oid.Iv(a_j)$ 有定义，且 $Iv(t_j)=oid.Iv(a_j)$. 令 $v'(variable)=oid$，由于 *lid* 为恒等变换，$oid.Iv(lid)=oid=v'(variable)$，这样显然 $I\models_{v'} L'$；而根据 $v'(variable)=oid$ 和 $oid.Iv(a_j)=Iv(t_j)$，$send(variable, a_j, t_j)$ 显然成立，即 $I\models_{v'} A$，所以 $I\models_v L'$，A.

②若 $I\models_v L'$，A，即 $I\models_v L'$ 和 $I\models_v A$. 根据 $I\models_v A$，则 $v'(variable).Iv'(a_j)=Iv'(t_j)$，而根据 $I\models_{v'} L'$，存在 $oid\in I_C(b)$，使每个 $a_i:t_i$(其中 $0\leqslant i\leqslant n$)满足条件(Ⅰ，Ⅱ)，特别地，$oid.Iv'(lid)=v'(variable)$，即 $oid=v'(variable)=Iv'(variable)$，因此 $oid.Iv'(a_j)=Iv'(t_j)$，即有 $I\models_{v'} L$，因为 L 中没有出现 *variable*，所以 $I\models_v L$. □

变换 2. 对于公式 L，若其中的参数用到形式为 *object.property* 的简单表达式，可在 L 中用 *variable* 替代该表达式，成为 L'. 如果 *object* 所属类为 c'，则把 $c'(oid:object, property:variable)$ 记为 A，可把 L 变换成 L'，A.

引理 2. 变换 2 是语义等价的.(限于篇幅，引理 2 及以下引理的证明从略).

变换 3. 对于原子公式 $b(a_1:t_1, a_2:t_2, \cdots, a_n:t_n)$，记作 L. 若其中 $t_j(1\leqslant j\leqslant n)$ 为另一个类名原子公式 $c'(a_1':t_1', \cdots, a_m':t_m')$，如果把 $b(a_1:t_1, \cdots, a_{j-1}:t_{j-1}, a_j:variable, a_{j+1}:t_{j+1}, \cdots, a_n:t_n)$ 记作 $L1$，$c'(oid:variable, a_1':t_1', \cdots, a_m':t_m')$ 记作 $L2$，则 L 可转换成 $L1$，$L2$.

引理 3. 变换 3 是语义等价的.

变换 4. 对于原子公式 $b(a_1:t_1, a_2:t_2, \cdots, a_n:t_n)$，记作 L. 若其中 $a_j(1\leqslant j\leqslant n)$ 为一个路径性质 $basic\_prop.path\_prop$，如果把 $b(a_1:t_1, \cdots, a_{j-1}:t_{j-1}, basic\_prop:variable, a_{j+1}:t_{j+1}, \cdots, a_n:t_n)$ 记作 L'，令 b 中 *basic_prop* 的值域为 $Ref(c')$，则 $c'(oid:variable, path\_prop:t_j)$ 记作 A，可把 L 变成 L'，A.

引理 4. 变换 4 是语义等价的.

变换 5. 对于原子公式 $b(a_1:t_1, a_2:t_2, \cdots, a_n:t_n)$，记作 L. 若其中 $a_j(1\leqslant j\leqslant n)$ 的类型为数组型，它对应的参数 t_j 应是形式为 $[expr_1, expr_2, \cdots, expr_m]$ 的数组表达式，如果把 $b(a_1:t_1, a_2:t_2, \cdots, a_{j-1}:t_{j-1}, a_j[1]:expr_1, a_j[2]:expr_2, \cdots, a_j[m]:expr_m, a_{j+1}:t_{j+1}, \cdots, a_n:t_n)$ 记作 L'，则 L 可变成 L'.

引理 5. 变换 5 是语义等价的.

变换 6. 对于原子公式 $b(a_1:t_1,a_2:t_2,\cdots,a_n:t_n)$,记作 L. 若其中 $t_j(1\leqslant j\leqslant n)$是形式为 $\{expr_1,expr_2,\cdots,expr_m\}$的集合表达式,如果把 $b(a_1:t_1,a_2:t_2,\cdots,a_{j-1}:t_{j-1},a_j:variable,a_{j+1}:t_{j+1},\cdots,a_n:t_n)$记作 L',如果 a_j 的值域为 $Set(t)$型,则存放这类集合元素的关系模式为 $t\_set(set,element)$,把 $t_set(variable,expr_1)$记为 $A1$,$t_set(variable,expr_2)$记为 $A2$,$\cdots$,$t_set(variable,expr_m)$记为 A_m,则 L 可变成 $L',A_1,A_2,\cdots,A_m$.

引理 6. 变换 6 是语义等价的.

变换 7. 如果规则 r 的规则体中有一个谓词 c 是带 * 号的,设 c 的所有子类(包括直接子类和间接子类)为 $c_1,c_2,\cdots,c_n$. 把 r 中的 c^* 替代成 c,记为 r',把 r 中的 c^* 替代成 c_1,记为 r_1,把 r 中的 c^* 替代成 c_2,记为 r_2,$\cdots$,把 r 中的 c^* 替代成 c_n,记为 r_n,则规则 r 可替换成 $n+1$ 个规则 $r',r_1,r_2,\cdots,r_n$.

引理 7. 变换 7 是语义等价的.

变换 8. 对于原子公式 $b(a_1:t_1,a_2:t_2,\cdots,a_n:t_n)$,记作 L. 如果参数标签个数和名称与对应类的属性不一致,它可能只出现部分属性,也可能重复. 当 $attr_1,\cdots,attr_m$ 是类的属性,而不出现在 L 的参数标签中时,可把 L 转换成 $b(a_1:t_1,a_2:t_2,\cdots,a_n:t_n,attr_1:variable_1,\cdots,attr_m:variable_m)$,记作 L';如果 L'中的 a_i 和 a_j 同名,在 L'中去掉 $a_j:t_j$ 项,记作 L'',则 L 可变成 $L'',t_i=t_j$.

引理 8. 变换 8 是语义等价的.

通过变换 1～8,可以把 O－Datalog 程序变成简单 O－Datalog 程序,而且根据以上 8 个引理,它们保持相同的语义,因此:

定理 1. 对于任何一个 O－Datalog 程序 p,经过反复使用上述 8 种变换,最终能变成一个简单的 O－Datalog 程序 p',并且 p 和 p'保持相同的语义.

2.2 简单 O－Datalog 程序与 Datalog 程序

在简单 O－Datalog 程序中,对于类名原子 $c(a_1:t_1,a_2:t_2,\cdots,a_n:t_n)$,其中 $a_1,a_2,\cdots,a_n$ 是类 c 的所有属性,相应关系库中一定存在一个关系 r,其模式为 $(a_1,a_2,\cdots,a_n)$,因此该原子能对应于 Datalog 原子 $r(t_1,t_2,\cdots,t_n)$,而由于 Datalog 程序中的谓词参数不带标签,各个成分以固定的顺序出现,一个参数的意义只能根据它在谓词中的位置确定. 因此引进一个变换 9.

变换 9. 把简单 O－Datalog 程序中的各个类名原子的参数及标签按关系库中对应的属性顺序排列好,再去掉所有的参数标签. 尽管方法返回值一般没有存放在关系库中,而是动态计算的. 但在 O－Datalog 中出现方法调用时,由于它的各参量都是常量,也可以在计算这个 O－Datalog 程序之前先计算这些方法. 假设 O－Datalog 程序中出现的方法调用都预先计算好了,存放在另一个关系 r 中,这个关系的名是根据类名和方法调用来确定的,关系的第一个属性是对象标识,以后的属性是根据方法返回值类型确定的.

变换 10. 对于简单 O－Datalog 程序中的方法原子 $send(object,method,result)$,找到它相应的关系名 r,用 $r(object,result)$替代该原子.

定理 2. 对于一个简单 O－Datalog 程序 p,通过变换 9 和变换 10 可构造一个相应的 Datalog 程序 q,并且 q 和 p 保持相同的语义.

证明:由于简单 O－Datalog 程序中的每个 *EDB* 谓词都和关系库中的一个关系对应,对于类名原子 $c(a_1:t_1,a_2:t_2,\cdots,a_n:t_n)$,记作 L,关系库中一定存在一个关系 r,其模式为 $(a_1,a_2,\cdots,a_n)$,因为关系 r 存放类 c 的所有实例的属性值,所以 r 在解释 I 下为 $I(r)=\{[oid.I(a_1),\cdots,oid.I(a_n)]|oid\in I_C(c)\}$,因此 $I\models_v L$ *iff* $[Iv(t_1),Iv(t_2),\cdots,Iv(t_n)]\in I(r)$. 如果我们考察一个 Datalog 的 *EDB* 谓词 $p(t_1,t_2,\cdots,t_n)$,根据 Datalog 的 *EDB* 谓词的意义,有 $I\models_v p(t_1,t_2,\cdots,t_n)$ *iff* $[Iv(t_1),Iv(t_2),\cdots,Iv(t_n)]\in I(r)$. 所以,$I\models_v L$ *iff* $I\models_v p(t_1,t_2,\cdots,t_n)$,即变换 9 是语义等价的.

在"类 c 的方法调用 *method* 的返回值已存储在关系库 r 中"的前提下,对于类 c 的对象 *object*,显然 $I\models_v send(object,method,result)$ *iff* $I\models_v r(object,result)$,即变换 10 是语义等价的. 因此定理 2 成立. □

3 O－Datalog 程序计值

根据定理 1 和定理 2 可知,对于一般 O－Datalog 程序,可以通过一些变换,转化成简单 O－Datalog 程序,再为它构造一个语义等价的 Datalog 程序,利用 Datalog 的语义模型和不动点理论,可以对之进行优化和计值.

在变换 1～6 之后,如果引进变换 11,可同时解决变换 7、8、9 所要解决的问题:

变换 11. 对于 O－Datalog 程序中的每一个 *EDB* 原子 $b(a_1:t_1,a_2:t_2,\cdots,a_n:t_n)$,作

①在原规则中把谓词名 b 换成 b',并把参数标签去掉,参数写成 Datalog 形式;

②新增加一条 Datalog 规则,其头谓词名为 b',其中参数是原来谓词 b 中的参数标签,规则体也只有一个谓词,而且谓词名为 b,其参数和关系库中的关系模式的各属性名一致;

③如果 b 带"＊"号,找出 b 的所有子类,为每个子类按第②步的方法写一条规则.

虽然变换 11 本身不是语义等价的,但同一个 O－Datalog 程序经过变换 11 得到的 Datalog程序和经过变换 7、8、9 得到的 Datalog 程序的最小不动点相同,即经过变换 11 后得到的 Datalog 程序和原来的 O－Datalog 程序的最小不动点相同.

另外,关于方法的计值问题,下面的算法是采用预先计算的策略,当然也可以在执行半朴素算法时动态计算.

算法 1. O－Datalog 程序计值

输入:一个 O－Datalog 程序 p.

输出:程序 p 的最小不动点.

方法:(1)找出 O－Datalog 程序中的所有是方法调用的参数标签,预先计算好谓词名所对应类的实例的方法值.

(2)把程序 p 拷贝给一个临时程序 T. 对 T 作以下变换:

①对 T 中的每一个是另一个对象的属性或方法值的参数,使用变换 2.

②对 T 中的每一个嵌套谓词,使用变换 3.

③对 T 中的所有嵌套路径标签,使用变换 4.

④对 T 中的所有为方法调用的标签,使用变换 1.

⑤反复考察 T,如果有一个参数的类型是数组型,使用变换 5. 如果有一个参数的类型

是集合型,使用变换 6.

⑥进行变换 11 和变换 10,并把结果程序放到 p' 中.

(3)对 Datalog 程序 p' 进行优化.

(4)用半朴素算法对 p' 进行计值.

定理 3. 算法 1 是正确的.

证明:首先假设在算法 1 中使用变换 7、8、9,而不用变换 11,则在算法 1 的第 2 步中,①步消除了原程序中的所有是另一个对象的属性或方法值的参数,②步消除了其中的所有嵌套谓词,③步消除了其中的所有嵌套路径标签,④步消除了其中的所有为方法调用的标签,⑤步消除了所有的参数类型为数组型或集合型的情况,⑥步中的变换 7、8 最终把 p 变成简单 O—Datalog 程序,再由变换 9、10 产生 Datalog 程序 p',根据定理 1、2 可知 p 和 p' 的语义等价,因此算法 1 对 p' 的计值能正确得到 p 的最小不动点.

事实上算法 1 中使用变换 11,根据前面的讨论,变换 11 必须在其它变换进行之后才能做,这时经过变换 11 得到的 Datalog 程序和经过变换 7、8、9 得到的 Datalog 程序的最小不动点相同,算法 1 在第⑥步才使用变换 11,因此得到的 Datalog 程序和原来的 O—Datalog 程序的最小不动点相同.所以算法 1 正确求出了程序 p 的最小不动点. □

4 结　论

一般 O—Datalog 程序可以语义等价地变换成简单 O—Datalog 程序,而简单 O—Datalog 程序事实上是一种 Datalog 程序的变形,因此最终可以为一个 O—Datalog 程序构造一个相应的 Datalog 程序,从而利用 Datalog 对原程序计值,而 Datalog 已经被大量研究过,有很多经验和算法可以利用,因此对 O—Datalog 程序的计值是正确而且有效的,我们已经在 SUN 工作站上实现了 O—Datalog 语言原型.

本文还对 O—Datalog 程序的变换方法作过初步优化,例如用变换 11 替代变换 7、8、9,并保证变换 11 所得到的 Datalog 程序和原程序的最小不动点相同.但 O—Datalog 程序的优化还需要进一步研究,例如关于方法的预先计算策略,如果多个 O—Datalog 程序(或者一个程序中的多条规则)中多次用到相同的方法调用,这是一种有效的办法;而如果一个方法只被一个谓词调用,并且该谓词的参数中有些是受限的,这时预先计算方法就有浪费.因此,对 O—Datalog 程序的优化策略是我们下一步研究的目标.

参考文献

1　张成洪,施伯乐,胡运发.面向对象数据库的推理查询语言.软件学报,1996,7(增刊):30～37.

2　Greco S, Leone N, Rullo P. Complex: an object－oriented logic programming system. IEEE Transactions on Knowledge and Data Engineering, 1992, 4(4): 344～359.

3　Srivastava D *et al*. Coral＋＋: adding object－orientation to a logic database language. In: Proc. 19th VLDB. 1993. 158～170.

4　Kifer M, Kim W, Sagiv Y. Querying object－oriented databases. In: Proc. of ACM SIGMOD, 1992. 392～402.

5　施伯乐,张成洪,周傲英.FOOD:一个面向对象数据库系统.计算机应用与软件,1994,11(6):47～53.

6　Ullman J D. Principles of database and knowledge－base systems. Vol Ⅱ. Rockville: Computer Science Press, 1989.

THE IMPLEMENTATION OF THE DEDUCTIVE QUERY OF AN OBJECT ORIENTED DATABASE

Zhang Chenghong Shi Baile Hu Yunfa

(*Department of Computer Science Fudan University Shanghai* 200433)

Abstract The last paper has introduced the O—Datalog, a deductive query language of the object—oriented database system FOOD. This paper puts great emphasis on discussing some transformations of the O—Datalog programs, and proving that these transformations preserve the same semantics. So an O—Datalog program can be transformed into a corresponding Datalog program and can be evaluated using the Datalog program, then an algorithm of evaluating the O—Datalog programs is given.

Key words Object—oriented database system, deductive query, Datalog.