一个实现对象查询语言的形式化基础*

石祥滨 张斌 王国仁 于戈 郑怀远

(东北大学计算机系 沈阳 110006)

摘要 在基于CORBA(common object request broker)的面向对象多数据库系统SCOPE/CIMS中,作者选择了对象数据库管理组ODMG(object database management group)提出的对象数据库标准ODMG-93的OQL(object query language)作为全局查询语言.为此,提出了一种实现OQL的形式化基础,包括适合建模OQL的一种对象演算和一种对象代数;对象演算的的规范化规则和规范化步骤;对象演算与对象代数的等价映射规则及转换策略.

关键词 数据库,面向对象数据库,查询模型,查询语言.

中图法分类号 TP311

作为一种面向对象的查询语言,OQL(object query language)是对象数据库管理组ODMG(object database management group)提出的面向对象数据库标准ODMG-93[1]的一个重要成分,已成为事实上的工业标准.在基于CORBA(common object request broker)的面向对象多数据库系统SCOPE/CIMS中,我们选择了OQL作为全局查询语言.

为了有效地实现面向对象查询语言,需要有像关系模型的关系演算和关系代数那样的形式化基础.面向对象的查询语言首先被转换为对象演算表达式,对象演算表达式经规范化后被等价地转化为对象代数表达式,由对象代数表达式产生查询执行计划.近年来,关于对象代数和对象演算的研究很多,这些研究可以分为如下几类:①综述性研究,如文献[2~4]全面讨论了对象查询模型应具备的基本特征;②有关一种演算或代数的研究,如文献[5]提出了一种扩展的关系演算,文献[6,7]分别提出了一种对象代数;③一些特殊问题的研究,如文献[8]重点讨论了路径表达式的处理问题,文献[9]讨论了利用代数重写解决查询语言的任意嵌套问题;④利用对象演算和对象代数实现一种对象查询语言的研究,如文献[10~12].以上这些研究与本文密切相关的是第4类研究,文献[10]讨论了从查询语言直接到对象代数的转换,其对象模型和语言特征与ODMG-93类似,但缺乏相应的等价变换基础;文献[11]讨论了对象查询语言—对象演算—对象代数的转换方法及相应的等价变换机制,但其对象模型与ODMG-93的对象模型有很大差异,并且未涉及具体的语言特征;文献[12]讨论了OQL—对象演算—对象代数的转换问题,但在其代数中采用了单一的对象操作符—类群同构操作,与普遍采用多个操作符(如选择、投影、连接等)的代数有相当大的区别,因此,无法很好地利用很多已有的查询优化的研究.所以,对于适合实现OQL的形式化基础的研究仍然是今后一段时间内的一个重要的研究课题.

尽管OQL在SCOPE/CIMS中是作为全局查询语言使用的,但对象代数、对象演算及等价变换基础依然是全局查询处理的基础.因此,本文的目标就是提出一种合适的形式化基础,实现OQL—对象演算—对象代数的等价变换.由于SCOPE/CIMS系统目前不支持任意的方法调用,因此本文仅涉及无参数的方法.本文的主要内容包括:适合建模OQL的一种对象演算和一种对象代数;对象演算的规范化规则和规范化步骤;对象演算与对象代数的等价映射规则及转换策略.

1 对象演算

实现OQL的对象演算和对象代数应以ODMG-93的面向对象数据模型为基础,并能建模OQL的特征.一些研

* 本文研究得到国家863高科技项目基金资助.
作者石祥滨,1963年生,博士生,副教授,主要研究领域为分布数据库,面向对象数据库,信息集成技术,软件工程.张斌,1964年生,博士,副教授,主要研究领域为信息集成技术,面向对象数据库,WEB数据库.软件工程.王国仁,1966年生,博士,副教授,主要研究领域为分布数据库,并行处理技术,面向对象数据库.于戈,1962年生,博士,教授,博士导师,主要研究领域为并行处理技术,分布数据库,面向对象数据库,信息集成技术.郑怀远,1931年生,教授,博士导师,主要研究领域为分布数据库,面向对象数据库,信息集成技术.

本文通讯联系人:石祥滨,沈阳 110006,东北大学计算机系

本文1997-05-29收到原稿,1997-12-15收到修改稿

究[2,3]认为对象包含(Object Comprehensions)是一种理想的对象演算表示方法,这种表示方法具有简明、表达力强、可优化的特点.本文发展了这种思想,用于建模OQL,并提供了相应的代数支持.本文目前的研究是基于集合类型的,我们相信通过引入类型转换子和针对其它聚集类型的操作符可以将本文的研究进一步扩展,满足处理各种聚集类型的要求.

1.1 对象演算的定义

对象演算表达式的基本形式为 $set\{e|\bar{q}\}$,$\bar{q}$ 是项的序列 $q_1,\ldots,q_n,n\geq 1$,每一个项 q_i 为:(1) 一生成子(Generator) $v\leftarrow e'$,v 是一个迭代变量;(2) 一谓词 $pred,e,e'$,$pred$ 为与OQL规定的查询表达式具有等价表达能力的表达式,可能包含方法调用.表达式 e 相当于OQL的投影表达式,生成子相当于OQL中SELECT子句中的Guery as Identifier形式的变量说明.

对象演算表达式 $set\{e|\bar{q}\}$ 表示对象的集合,根据项的序列经一系列的迭代和选择,计算表达式 e 得到该集合中的每一个对象.其数学含义可归纳为

$set\{e|\}=\{e\}$

$set\{e|x\leftarrow u,\bar{q}\}=(u,\lambda x.\ set\{e|\bar{q}\})$

其中$(u,\lambda x.\ set\{e|\bar{q}\})$采用了通常使用的迭代函数记法,即

$(u,\lambda x.\ f(x))=\{f(x_1)\}\cup\{f(x_2)\}\ldots\cup\{f(x_n)\},\quad u=\{x_1,x_2,\ldots,x_n\}$

$set\{e|pred,\bar{q}\}=$if $pred$ then $set\{e|\bar{q}\}$ else $\{\}$

根据以上定义的对象演算,OQL语句到对象演算表达式的转换是直接了当的.

OQL语句的一般形式:select e from e_1 as $x_1,\ldots,e_n$ as x_n where $pred$ 可直接转换为对象演算表达式

$set\{e|x_1\leftarrow e_1,\ldots,x_n\leftarrow e_n,pred\}$.

下面以图1所示的数据库模式举例说明对象演算表达式的表达查询的方式.

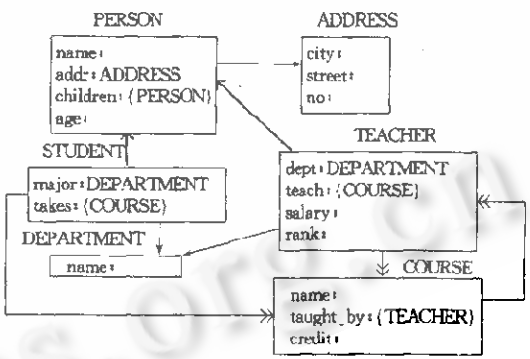

图1　示例数据库模式

例1:查询年龄最大的孩子居住在New York的人的名字和年龄最大的孩子的名字.在这个例子中,投影表达式和谓词表达式中的路径表达式存在部分重叠,并且包含对定义在类PERSON上方法oldest_child的调用.

select p.name,p.oldest_child().name **from** person as p **where** p.oldest_child().addr.city="New York"

$set\{\langle p.name,p.oldest\_child().name\rangle|p\leftarrow person,p.oldest\_child().addr.city="NewYork"\}$

例2:查询选修了由职称为教授的教师所讲授的课程的学生的名字和教授的名字,这个例子表达了OQL中典型的导航查询.

select student:s.name,professor:t.name **from** student as s,s.takes as c,c.taught_by as t **where** t.rank="professor"

$set\{\langle student:s.name,professor:t.name\rangle|s\leftarrow student,c\leftarrow s.takes,t\leftarrow c.taught\_by,t.rank="professor"\}$

例3:查询计算机系学生所选修的超过4学分的课程的名字,这是在FROM子句中存在嵌套的例子.

select c.name **from** (**select** s.takes **from** student as s **where** (s.major.name="cs") as c_1,c_1 as c **where** c.credit>4

$set\{c.name|c_1\leftarrow set\{s.takes|s\leftarrow student,\ s.major.name="CS"\},c\leftarrow c_1,c.credit>4\}$

例4:查询每个学生的名字及选修课程的名字,这是在SELECT子句中存在嵌套的例子.

select sname:s.name,cnames:(**select** c.name **from** s.takes as c) **from** student as s

$set\{\langle sname:s.name,cnames:set\{c.name|c\leftarrow s.takes\}\rangle|s\leftarrow student\}$

例5:查询选修了smith所选修的超过4学分的课程的学生的名字和课程的名字,这是一个在WHERE子句中存在嵌套的例子.

select sname:s.name,cname:c.name **from** student as s,s.takes as c **where** c **in** (**select** c_1 **from** student as s_1,s_1.takes as c_1 **where** s_1.name="smith" **and** c_1.credit>4)

$set\{\langle sname:s.name, cname:c.name\rangle \mid s \leftarrow student, c \leftarrow s.takes, c \text{ in } \{c_1 \mid s_1 \leftarrow student,\ c_1 \leftarrow s_1.takes, c_1.credit > 4\}$

1.2 对象演算的规范化规则

前面已经提到,在研究[2,3]中认为对象包含表达式是可优化的,这种优化是通过对象包含表达式的等价变换代替关系代数中基于代数重写的优化策略.本文把通过规则对对象演算表达式进行等价变换的过程称为规范化.对象演算的规范化要达到以下目标:(1) 运用谓词规则消除冗余的谓词项,将谓词表达式等价变换为析取范式,以达到将一个查询等价转换为一系列合取子查询的目的,这个目标是通常对象演算规范化的基本目标[11];(2) 将嵌套的查询转换为非嵌套的形式或转换为可以在对象代数中求解的形式;(3) 实现路径表达式的重新组合以避免路径表达式的重复计算;(4) 优先进行选择.我们认为在规范化的过程中取得(2)~(4)目标的优点是:①为求解嵌套和路径表达式等特定问题提供了容易理解和实现的形式化基础,保证求解结果的正确性;②由于可将各种问题在规范化的过程中综合进行考虑,因此,比基于代数重写的策略更容易产生优化的结果.

我们定义的规范化规则及解释如下:

规则 1. $set\{e \mid \bar{q}, pred_1 \vee pred_2\} \Rightarrow set\{e \mid \bar{q}, pred_1\} \cup set\{e \mid \bar{q}, pred_2\}$

规则 2. $set\{e \mid \bar{q}, v_1 \leftarrow (e_1 \cup e_2), \bar{r}\} \Rightarrow set\{e \mid \bar{q}, v \leftarrow e_1, \bar{r}\} \cup set\{e \mid \bar{q}, v \leftarrow e_2, \bar{r}\}$

规则 3. $set\{e \mid \bar{q}, v_1 \leftarrow (e_1 \cap e_2), \bar{r}\} \Rightarrow set\{e \mid \bar{q}, v \leftarrow e_1, v \in e_2, \bar{r}\}$

规则 4. $set\{e \mid \bar{q}, v_1 \leftarrow (e_1 - e_2), \bar{r}\} \Rightarrow set\{e \mid \bar{q}, v \leftarrow e_1, v \notin e_2, \bar{r}\}$

规则 5. $set\{e \mid q, v \leftarrow set\{e_1 \mid \bar{r}\}, \bar{s}\} \Rightarrow set\{e[v/e_1] \mid \bar{q}, \bar{r}, \bar{s}[v/e_1]\}$,其中 $e[v/e_1]$,$\bar{s}[v/e_1]$表示将 e,$\bar{s}$ 中的 v 替换为表达式 e_1,其作用是消除 SFW as x 类型的嵌套.

规则 6. $set\{\langle e_1, a: set\{e_2 \mid \bar{r}\}\rangle \mid \bar{q}\} \Rightarrow set\{c.e_1, a: set\{b.e_2 \mid b \leftarrow X, b.e_1 = c.e_1, b.x = c.x\} \mid c \leftarrow X\}, X = set\{\langle e_1, a: e_2, x\rangle \mid \bar{q}, \bar{r}\}$,假定 $\bar{q}$ 中的叠代变量为 x,其作用是消除 a:SFW 类型的嵌套.

规则 7. $set\{e \mid \bar{q}, v \in set\{e_1 \mid \bar{s}\}, \bar{r}\} \Rightarrow set\{e \mid \bar{q}, \bar{s}, v = e_1, \bar{r}\}$,其作用是消除 x in SFW 类型的嵌套.

由于查询表达式的正交嵌套特征,OQL 中的嵌套是非常复杂的.需要针对不同的情况设计不同的规范化规则,篇幅所限,本文只列出 3 种典型嵌套的规范化规则 5~7.

规则 8. $set\{e \mid \bar{q}, pred, \bar{r}\} \Rightarrow set\{e[v_1/v.e_1] \mid \bar{q}, v_1 \leftarrow v.e_1, pred[v_1/v.e_1], \bar{r}[v_1/v.e_1]\}$,$v.e_1.e_2 \ldots e_m (m \geqslant 2)$是谓词 $pred$ 中的路径表达式,$\{x_1, x_2, \ldots, x_l\} (l \geqslant 1)$是 $\bar{q}$ 中的叠代变量,$v \in \{x_1, x_2, \ldots, x_l\}$,并且 v 不依赖 $\bar{r}$ 中的叠代变量.该规则的作用是展开谓词中的路径表达式,以达到路径表达式重组的目的,避免路径表达式的重复计算.

规则 9. $set\{e \mid \bar{q}\} \Rightarrow set\{e[v_1/v.e_1] \mid \bar{q}, v_1 \leftarrow v.e_1\}$,$v.e_1.e_2 \ldots e_m (m \geqslant 2)$是 e 中的路径表达式,$\{x_1, x_2, \ldots, x_n\} (n \geqslant 1)$是 q 中的叠代变量,$v \in \{x_1, x_2, \ldots, x_n\}$,该规则用于逐层展开投影表达式的路径表达式.其解释与规则 8 类似.

规则 10. $set\{e \mid \bar{q}, v_1 \leftarrow e_1, pred, \bar{r}\} \Rightarrow set\{e \mid \bar{q}, pred, v_1 \leftarrow e_1, \bar{r}\}$,谓词表达式 $pred$ 不依赖于迭代变量 v_1.这个规则可以达到通常在进行连接前优先进行选择的目的.

按前面对象演算的定义,以上规范化规则很明显是保持语义的变换.

1.3 对象演算规范化步骤

根据前面定义的规范化规则,对象演算表达式的规范化采取的基本步骤如下:

(1) 将 OQL 语句中的谓词表达式 $pred$ 归约为析取范式 $pred_1 \vee \ldots \vee pred_n$.

(2) 运用规则 1 将 $set\{e \mid \bar{q}\}$转换为 $set\{e \mid \bar{q}_1\} \cup \ldots \cup set\{e \mid \bar{q}_n\}$.

对于经步骤(1),(2)得到的对象演算表达式 $set\{e \mid \bar{q}_i\}$执行以下步骤:

(3) 用规则 2~4 重写集合表达式.

(4) 用规则 6 消除 a:SFW 类型的嵌套.

(5) 用规则 5 消除 SFW as x 类型的嵌套.

(6) 用规则 7 消除 x in SFW 类型的嵌套.

(7) 用规则 8 逐层展开谓词的每个路径表达式.

(8) 用规则 9 逐层展开投影表达式的每个路径表达式.

(9) 运用规则 10 尽可能前移谓词项,直到不能前移为止.

经过规范化后的对象演算表达式具有如下性质.

性质 1. 经规范化的对象演算表达式 $set\{e \mid \bar{q}\}$中项的序列 $\bar{q}$ 中的第 1 项一定是一由命名对象集合定义的生成子 $x \leftarrow e_1$;即 e_1 是具有名字的对象集合,为类外延或持久化变量名.

下面给出例 1,3,4 的规范化过程.

$set\{\langle p.name, p.oldest\_child().name\rangle \mid p \leftarrow person, p.oldest\_child().addr.city = \text{“}NewYork\text{”}\} \xrightarrow{rule4} set\{\langle p.name, t.name\rangle \mid p \leftarrow person, t \leftarrow p.oldest\_child(), t.addr.city = \text{“}NewYork\text{”}\} \xrightarrow{rule9} set\{\langle p.name, t.name\rangle \mid p \leftarrow person, t \leftarrow p.oldest\_child(), w \leftarrow t.addr, w.city = \text{“}NewYork\text{”}\}$

$set\{c.name \mid c_1 \leftarrow set\{s.takes \mid s \leftarrow student, s.major.name = \text{“}CS\text{”}\}, c \leftarrow c_1, c.credit > 4\} \xrightarrow{rule5} set\{c.name \mid s \leftarrow student, s.major.name = \text{“}CS\text{”}, c \leftarrow s.takes, c.credit > 4\} \xrightarrow{rule3} set\{c.name \mid s \leftarrow student, d \leftarrow s.major, d.name = \text{“}CS\text{”}, c \leftarrow s.takes, c.credit > 4\}$

$set\{\langle sname: s.name, cnames: set\{c.name \mid c \leftarrow s.takes\}\rangle \mid s \leftarrow student\} \xrightarrow{rule6} set\{\langle sname: s.name, cnames: set\{y.cnames \mid y \leftarrow S, y.sname = x.sname, y.s = x.s\}\rangle \mid x \leftarrow S\}$

$S = set\{\langle sname: s.name, cnames: c.name, s\rangle \mid s \leftarrow student, c \leftarrow s.takes\}$

$set\{\langle sname: s.name, cname: c.name\rangle \mid s \leftarrow student, c \leftarrow s.takes, c \text{ in } \{c_1 \mid s_1 \leftarrow student, c_1 \leftarrow s_1.takes, c_1.credit > 4\}\} \xrightarrow{rule7} set\{\langle sname: s.name, cname: c.name\rangle \mid s \leftarrow student, c \leftarrow s.takes, s_1 \leftarrow student, c_1 \leftarrow s_1.takes, c_1.credit > 4, c = c_1\}$

2 对象代数

下面给出我们所使用的主要代数操作符的签名，并重点讨论经规范化的对象演算表达式到对象代数表达式的转换问题.

2.1 对象代数的定义

我们所使用的代数操作符的签名为：

(1) $Select(\sigma)$: $set[T], pred \rightarrow Set[T]$

(2) $Image(\alpha)$: $set[T], f: T \rightarrow Q, parameter\_list \rightarrow set[Q]$

(3) $Project(\pi)$: $set[T], \langle A_1, f_1: T \rightarrow T_1, \ldots, A_n, f_n: T \rightarrow T_n\rangle \rightarrow set[\langle A_1: T_1, \ldots, A_n: T_n\rangle]$

(4) $Union(\cup), Intersection(\cap), Difference(-)$: $set[T], set[R] \rightarrow set[S]$，其中 S 为 T 和 R 的最特定的共同超类型.

(5) $Join(\infty)$: $set[T], set[R], A_1, A_2, pred \rightarrow set[\langle A_1: T, A_2: R\rangle]$

(6) $Flatten(\rho)$: $set[set[T]] \rightarrow set[T]$

(7) $Nest(\mu)$: $set[\langle A_1: T_1, \ldots, A_i: T_i, \ldots, A_n: T_n\rangle], A_i \rightarrow set[\langle A_1: T_1, \ldots, A_i: set[T_i], \ldots, A_n: T_n\rangle]$

(8) $Unest(\nu)$: $set[\langle A_1: T_1, \ldots, A_i: set[T_i], \ldots, A_n: T_n\rangle], A_i \rightarrow set[\langle A_1: T_1, \ldots, A_i: T_i, \ldots, A_n: T_n\rangle]$

(9) $Navigation\_join(\otimes)$: $set[T_1], f_1: T_1 \rightarrow T_2, \ldots, f_n: T_n \rightarrow T_{n+1}, A_1, \ldots, A_n, pred \rightarrow set[\langle A_1: T_1, \ldots, A_n: T_n\rangle]$

2.2 对象演算与对象代数的对应规则

对象演算表达式与以上代数操作的对应规则如下所示. 在这些规则中出现的代数操作符使用了前缀记法，其中的迭代变量只起标识作用.

(1) $set\{x \mid x \leftarrow e, pred_1(x), \ldots, pred_m(x)\} \Rightarrow \sigma((e,x), pred_1(x) \wedge \ldots \wedge pred_m(x))$

(2) $set\{f(p_1, \ldots, p_n) \mid x \leftarrow e\} \Rightarrow \alpha((e,x), f(p_1, \ldots, p_n))$

(3) $set\{\langle A_1: e_1(x), \ldots, A_n: e_n(x)\rangle \mid x \leftarrow e\} \Rightarrow \pi((e,x), A_1: e_1, \ldots, A_n: e_n)$

(4) $set\{x \mid x \leftarrow e_1, x \in e_2, \ldots, x \in e_n\} \Rightarrow e_1 \cap e_2 \ldots \cap e_n$

(5) $set\{x \mid x \leftarrow e_1, x \notin e_2\} \Rightarrow e_1 - e_2$

(6) $set\{\langle t,r\rangle \mid t \leftarrow e_1, r \leftarrow e_2, pred_1(t,r), \ldots, pred_m(t,r)\} \Rightarrow \infty((e_1,t), (e_2,t), pred_1(t,r) \wedge \ldots \wedge pred_m(t,r))$

(7) $set\{x \mid s \leftarrow e, x \leftarrow s\} \Rightarrow \rho(e)$

(8) $set\{\langle A_1: x.A_1, \ldots, A_{i-1}: x.A_{i-1}, A_i: set\{Y.A_i \mid y \leftarrow e, y.A_1 = x.A_1, \ldots, y.A_{i-1} = x.A_{i-1}, y.A_{i+1} = x.A_{i+1}, \ldots, y.A_n = x.A_n\}, A_{i+1}: x.A_{i+1}, \ldots, A_n: x.A_n\rangle \mid x \leftarrow e\} \Rightarrow \mu(e, A_i)$

(9) $set\{\langle A_1: x.A_1, \ldots, A_{i-1}: x.A_{i-1}, A_i: y, A_{i+1}: x.A_{i+1}, \ldots, A_n: x.A_n\rangle \mid x \leftarrow e, y \leftarrow x.A_i\} \Rightarrow \nu(e, A_i)$

(10) $set\{\langle x_1, \ldots, x_n\rangle \mid x_1 \leftarrow e, x_2 \leftarrow x_1.e_1, \ldots, x_n \leftarrow x_{n-1}.e_{n-1}, pred_1, \ldots, pred_m\} \Rightarrow \otimes((e,x_1), (e_1,x_2), \ldots, (e_{n-1}, x_n), pred_1 \wedge \ldots \wedge pred_m)$

2.3 对象演算到对象代数的转换

对象演算到对象代数的转换应保持原有的语义，该过程基于如下定理.

定理 1. 如果 $set\{e|\bar{q}\}$ 中，$x_1,\ldots,x_n,n\geqslant 1$ 是 $\bar{q}$ 中的全部叠代变量，则

$set\{e(x_1,\ldots,x_n)|\bar{q}\}\Rightarrow set\{e(t.x_1,\ldots,t.x_n)|t\leftarrow set\{(x_1,\ldots,x_n)|\bar{q}\}\}$

定理 2. $set\{(x_1,\ldots,x_n)|\bar{r},\bar{s}\}$ 中，$x_1,\ldots,x_i,i\geqslant 1$ 是 $\bar{r}$ 中的叠代变量，$x_{i+1},\ldots,x_n$ 是 $\bar{s}$ 中的叠代变量，则 $set\{(x_1,\ldots,x_n)|\bar{r},\bar{s}\}\Rightarrow set\{(y.x_1,\ldots,y.x_i,x_{i+1},\ldots,x_n)|y\leftarrow set\{(x_1,\ldots,x_i)|\bar{r}\},\bar{s}\}$.

由于 $(x_1,\ldots,x_n).x_i\Rightarrow x_i$ 运用规范化规则 5，这两个定理是很容易证明的.

根据定理 1，$set\{e|\bar{q}\}$ 向对象代数的转换问题，可以首先将 $set\{(x_1,\ldots,x_n)|\bar{q}\}$ 转换为对象代数 E，然后对于 E 运用投影操作计算表达式 $e(x_1,\ldots,x_n)$.

根据定理 2，$set\{(x_1,\ldots,x_n)|\bar{r},\bar{s}\}$ 向对象代数的转换可以首先将 $set\{(x_1,\ldots,x_i)|\bar{r}\}$ 转换为对象代数表达式 E'，然后将 $set\{(y.x_1,\ldots,y.x_i,x_{i+1},\ldots,x_n)|y\leftarrow E',\bar{s}\}$ 转换为对象代数表达式，即可以根据前面定义的对应性规则采取对 $\bar{q}$ 中的项的序列从前向后依次匹配的策略，只要保证每一次匹配的中间结果的类型为：

$set\{(x_1:type\_of\ \ element(x_1),\ldots,x_i:type\_of\_element(x_i))\},x_1,\ldots,x_i$ 为已转换的代数表达式中涉及的原对象演算表达式中的叠代变量. 为了便于按这种思路转换，我们定义了以下宏代数操作符.

(1) $Left\_project\_join(\vec{\infty})$:

$set[\langle A_1:T_1,\ldots,A_n:T_n\rangle],set[T_{n+1}],A_{n+1},pred\rightarrow set[\langle A_1:T_1,\ldots,A_{n+1}:T_{n+1}\rangle]$

(2) $Left\_project\_navigation\_join(\vec{\otimes})$:

$set[T],f_1:T_1\rightarrow T_2,\ldots,f_n:T_n\rightarrow T_{n+1},A_1,\ldots,A_n,pred\rightarrow set[\langle B_1:R_1,\ldots,B_m:R_m,A_1:T_1,\ldots,A_n:T_n\rangle]$，类型 T 为 $set[\langle B_1:R_1,\ldots,B_m:R_m\rangle]$

这两个宏代数操作符的作用是，使得再次匹配连接和导航连接时，中间结果的正确性可明显地用其它代数操作符等价地表达出来. 这两个宏代数操作的使用方法如下：

如果待转换的项的序列为：$y\leftarrow E',x_{i+1}\leftarrow x_j.e_{i+1},x_{i+2}\leftarrow x_{i+1}.e_{i+2},\ldots,x_m\leftarrow x_{m-1}.e_m,pred,\bar{s}$

E' 是已转换的中间结果对象代数表达式，如果 E' 中包含的原对象演算的叠代变量为 $x_1,\ldots,x_i,i\geqslant 1,x_j\in\{x_1,\ldots,x_i\}$.

如果 $i=1$，则将 s 前的序列匹配为 $\otimes((E',y),(e_{i+1},x_{i+1}),\ldots,(e_m,x_m),pred)$；否则，$\bar{s}$ 前的序列匹配为宏操作 $\vec{\otimes}((E',y),(e_{i+1},x_{i+1}),\ldots,(e_m,x_m),pred)$.

$\vec{\infty}$ 操作符的使用方法与 $\vec{\otimes}$ 操作符类似.

根据对象演算与对象代数的对应性规则和上述的转换思想可以容易地写出转换算法(算法略). 以下只对例 3 的转换过程作简要说明.

初始待转换项的序列为

$$s\leftarrow student,d\leftarrow s.major,d.name=\text{“}CS\text{”},c\leftarrow s.takes,c.credit>4,$$

按定理 1，2 和对应规则 10，将 $s\leftarrow student,d\leftarrow s.major,d.name=\text{“}CS\text{”}$ 匹配为

$$E'=\otimes((student,s),(s.major,d),d.name=\text{“}CS\text{”}),$$

待转换项的序列变为：$x\leftarrow E',c\leftarrow s.takes,c.credit>4$，按宏操作 $\vec{\otimes}$ 的定义及匹配方法，继续匹配为

$$E'=\vec{\otimes}((\otimes((student,s),(s.major,d),d.name=\text{“}CS\text{”}),x),(s.takes,c),c.credit>4,$$

最后，将 $set\{c.name|y\leftarrow E'\}$ 匹配为

$$\pi((\vec{\otimes}((\otimes((student,s),(s.major,d),d.name=\text{“}CS\text{”}),x),(s.takes,c),c.credit>4),y),c.name).$$

其余例子的规范化对象演算表达式可转换为如下代数表达式：

例 6: $\pi((\otimes((person,p),(p.oldest\_child(),t),(w,t.addr),w.city=\text{“}NewYork\text{”}),x),p.name,t.name)$

例 7: $\pi((\otimes((student,s),(s.takes,c),(c.taught\_by,t),t.rank=\text{“}professor\text{”}),x),student:s.name,professor:t.name)$

例 8: $\pi((\mu((\pi((\otimes((student,s),(s.takes,c)),x),sname:s.name,cnames:c.name,s),y),cnames),z),sname:s.name,cnames:c.name)$

例 9: $\pi((\infty((\otimes((student,s),(s.takes,c)),x),(\otimes((student,s_1),(s_1.takes,c_1),c_1.credit>4,y),c=c_1,z),sname:s.name,cname:c.name)$

3 结 论

本文是针对SCOPE/CIMS系统中实现全局查询语言OQL而提出的形式化基础,目前已应用于系统的实现中,但这种形式化基础完全可以用于其它类型的面向对象的数据库系统中.

虽然目前本文提出的形式化基础还不能完全解决实现OQL的全部问题,如包含各种聚集的查询的处理、有效地处理全称量词等,但我们相信,以目前的基础,通过适当地扩充规范化规则和代数操作符,完全解决实现OQL的全部问题并不存在不可逾越的障碍.

本文的后续研究主要是进一步完善该形式化基础和针对多数据库查询处理特点的研究,如查询分解、全局查询计划生成(包括已有应用(方法)的集成策略及优化策略)、局部查询优化等.

参考文献

1 Object Database Management Group. The object database standard: ODMG-93(release 1.2). San Francisco, CA: Morgan Kaufmann, 1996

2 Beeri C. New data models and languages — the challenge. In: Bocca J B *et al* eds. Proceedings of the 11th ACM Symposium on Principles of Database Systems. New York: ACM Press, 1992. 1~15

3 Danie. K C *et al*. Evaluating object-oriented query languages. The Computer Journal, 1994,37(10):858~872

4 Yu L, Osborn S L. An evaluation framework for algebraic object-oriented query models. In: Kamaiyashi Y *et al* eds. Proceedings of the 7th International Conference on Data Engineering. Los Alamitos Califinia: IEEE Press, 1991. 670~677

5 Elisa Bertino. Object-oriented query languages: the notion and the issues. IEEE Transactions on Knowledge Data Engineering, 1992,4(3):223~237

6 Alhajj R, Arkun M E. Object-oriented query language facilitating construction of new objects. Information and Software Technology, 1993,35(9):519~529

7 Shaw G M, Zdonik S B. Object-oriented query-equivalence and optimization. In: Kim M *et al* eds. Deductive and Object-oriented Databases. North-Holland: Elsevier Science Publisher, 1990. 281~295

8 Kemper Alfons, Moerkotte Guido. Advanced query proceeding in object bases using access support relations. In: Dennis Mcleeod *et al* eds. Proceedings of the 16th Very Large Database Conference. Polo Alto, Califinia: Morgan Kaufmann, 1990. 290~301

9 Hannie J S *et al*. From nested-loop to join query in OODB. In: bocca J B *et al* eds. Proceedings of the 20th Very Large Database Conference. Santiago, Chile, 1994. 618~629

10 Cluet Sophie *et al*. Reloop, an algebra based query language for an object-oriented database system. In: Kim W *et al* eds. Deductive and Object-orientec Databases. North-Holland: Elsevier Science Publisher, 1990. 313~332

11 Dave D S, Ozsu M Tamper. Queries and query proceeding in object-oriented database systems. ACM Transactions on Information System, 1990,8(4):387~430

12 Fegaras L, Maier D. Toward an effective calculus for object query languages. In: Michael J Carey *et al* eds. Proceedings of the ACM SIGMOD International Conference'95 on Management of Data. New York: ACM Press, 1995. 47~58

A Formal Foundation for Implementing Object Query Language

SHI Xiang-bin ZHANG Bin WANG Guo-ren YU Ge ZHENG Huai-yuan

(*Department of Computer Science Northeastern University Shenyang* 110006)

Abstract In this paper, the authors select OQL(object query language) of the object database standard: ODMG-93 suggested by ODMG (object database management group) as global query language in object-oriented multidatabase SCOPE/CIMS and present a formal foundation for implementing OQL, including an object calculus and an object algebra suitable for modeling OQL, normalization rules and normalization steps of the object calculus; mapping rules and translation methods between the calculus and the algebra.

Key words Database, object-oriented database, query model, query language.