维普资讯 http://www.cqvip.com

OODB 数据库 查询表达 选择查询表达 (17)

计算机科学1998 Vol. 25 №. 4

一种保持复合关系的OODB选择表达方式

68-70

Expression of Selecting Queries of OODB for Keeping Composite Relationship

TP311.13

钟 武 胡守仁

(国防科技大学计算机学院 长沙410073)

摘 要 The paper gives the idea of nested key schema lists used for navigation, by means of which the relevant composite relationships between the set objects spawned can be kept without the aid of restructuring expressions, after several set objects, between which there exist some composite relationships, are selected to produce final query results. Becaues a selecting query sentence can express quite complex query semantics, the expressions given by the paper have strong non-procedurality.

关键词 OODB, Queries, Selection operation

在表达对象间复杂的关系方面，OODB模型明显优于关系数据库模型。然而，由于对象之间的复杂关系存在着非规范性，因此在OODB的查询方面，存在着许多非完备性的查询表达方法。文[1,2]探讨了一种类似于SQL的查询表达方法；文[3,4,5]探讨了OODB的代数查询表达方法。无论哪一种查询表达方法，其选择查询操作的结果都是一个集合对象。

由于一个复杂的集合对象有可能包含若干子集合对象，且选择操作产生的集合对象必是一个无父亲的对象，因此势必产生如下待解决的问题：如何表达对多个具有某种复合关系的集合对象进行选择操作，并使操作所产生的各个新集合对象间保持原相应集合对象间的复合特性。本文就此给出一种选择查询表达方法，并用算法的形式描述了其语义。

1 复合对象

首先着重描述OODB中用元组和集合构造复合对象的问题。至于OODB中的其它概念，读者可参阅有关文献。

设：δ为无限的属性名集，Ω为系统提供的基本数据类型名(如：integer，real 等)集。这里Ω、δ不相交。在此模型中，特殊对象 null 是空对象，它表征一个当前不存在的对象。

定义1 OODB的模式Σ为三元组$(L,\leqslant,\psi)$：①类名集合为$L\cup\{Object,Set,Tuple\}$，该集合与$\Omega$、$\delta$不相交。这里，L为用户定义的类的类名集；②$\leqslant$是一个偏序关系，用于表征类名所对应类之间的继承关系：这里有：$Set\leqslant Objecdt$，$Tuple\leqslant Object$；③ψ：$L\to\Gamma$，将类名$C(\in L)$映射为一个二元组(ST,M)。其中，M是作用于ST上的方法集，ST的定义如下：

·对于属性名集$\{A_1,A_2,\cdots,A_k\}\subset\delta$，$T_1,T_2,\cdots,T_k\in\Omega\cup L$。这里，$T_i(i=1,\cdots,k)$不为C。$ST=[A_1:T_1,A_2:T_2,\cdots,A_k:T_k]$，即：ST为元组结构。此时，我们称类C为元组类；

·对于$T\in L$，且$T\leqslant Tuple$，$ST=\{T\}$，即：ST为元组集合结构。此时，我们称类C为元组集合类。

定义2 ①函数Struct用于求类的结构，函数Method用于求类的操作方法集；②对于集合类P，有$Struct(P)=\{T\}$，$T\in L$且$T\leqslant Tuple$；③对于元组类P，有$Struct(P)=[A_1:T_1,A_2:T_2,\cdots,A_k:T_k]$，函数$Att(P)=\{A_1,A_2,\cdots,A_k\}$。

定义3 ①对$D\in\Omega$，$\|D\|$是系统提供的基本类型D的值的集合；②对$C\in L$，$\|C\|=\{O|O$是类C的对象实例或是类C的子类的对象实例$\}$。由于继承性，当$C\leqslant C'$时，有$\|C\|\subseteq\|C'\|$；③若P为元组类，$Struct(P)=[A_1:T_1,A_2:T_2,\cdots,A_k:T_k]$，则$\|P\|=\{[A_1:V_1,A_2:V_2,\cdots,A_k:V_k]|V_i\in\|T_i\|(i=1,\cdots,k)\}$；④若P为集合类，$Struct(P)=\{T\}$，$T\in L$且$T\leqslant Tuple$，则$\|P\|=Powerset(\|T\|)$。

定义4 在数据库模式Σ下的数据库实例为二元组(Rs,Tu)：①$Rs=\{r_1,r_2\cdots\}$，它是集合对象r_i的集合$(i=1,2,\cdots)$；②$Tu=\{t_1,t_2\cdots\}$，它是元组对象t_i的集合$(i=1,2,\cdots)$。

2 嵌套选择

本文做两条约定：1)凡$r_1,r_{11},r_{12},r',\cdots$均表示集合对象；凡$t_1,t,t_{11},\cdots$均表示元组对象；2)类似于对象的深浅拷贝，对象的状态也有深浅的问题。如：设对象r是t_1,t_2的集合对象。则r的浅状态是指它包含对象的状态，即：$\{t_1,t_2\}$；r的深状态是指它包含对象的状态以及被包含对象的状态，即：t_1,t_2的状态也属r的状态。因此，当t_1,t_2的状态改变时，我们说r的浅状态没变，而r的深状态发生了改变。在以下的论述中，我们所指的对象状态均为对象的浅状态。

对集合对象r的选择操作，本文讨论3类：第一类是从集合对象中选取元组对象构成新的集合对

维普资讯 http://www.cqvip.com

象；如：r 的状态为$\{t_1,t_2,t_3\}$时，从 r 中挑选出满足条件的对象 t_1,t_2，然后产生新的集合对象 r'，将 t_1,t_2置入 r'，使其状态为$\{t_1,t_2\}$；第二类是只从 r 的子集合对象中选择元组对象。如：r 的状态为$\{t_1,t_2\}$，而 t_1,t_2的属性 A 又分别嵌入了对象 r_1,r_2。选择操作并不直接作用于 r，而是从 r 开始，沿对象嵌套模式到达并直接作用于 r_1和 r_2；第三类是第一类和第二类的结合。

选择操作的语法为：

选择操作→σ(元组对象的集合对象　选择条件[(键模嵌套表)])
键模嵌套表→表项[,键模嵌套表]|表项
表项→键模　选择条件[(键模嵌套表)]
键模→属性名[.键模]|属性名

定义5　设 $C_i\in L(i=1,\cdots,m)$且 $C_i\leqslant Tuple(i=1,\cdots,m)$；$C_0,C_{m+1}\in L$ 且 C_0和 $C_{m+1}\leqslant Set$，其中 $Struct(C_0)$为$\{C_1\}$；属性 $A_1,A_2,\cdots,A_m\in\delta$。若 $A_i\in Att(C_i)$且其定义形式为 $A_i:C_{i+1}(i=1,\cdots,m)$，则类 C_0的键模是关于 C_{m+1}类集合对象嵌入 C_0类集合对象的一种属性嵌套，其形式为 $A_1.A_2.\cdots.A_m.C_{m+1}$称为基于 $A_1.A_2.\cdots.A_m$的嵌套集合类。

例1　在对象复合模式图1中，长方形表示元组类，"{}"是集合构造符。C_1,C_2,C_5是元组类；C_0,C_3,C_4,C_6是集合类。C_0的结构为$\{C_1\}$，C_1的属性 A_3定义为 $A_1:C_2$，C_2的属性 A_2定义为 $A_2:C_3$。于是，对任何一个 C_0类对象 r，若存在元组对象 $t\in r$，则 $t.A_1.A_2$是一个类属于 C_3的集合对象。我们称 C_0的键模是关于 C_3类集合对象嵌入 C_0类集合对象的属性嵌套 $A_1.A_2.C_3$是 C_0的基于 $A_1.A_2$的嵌套集合类。

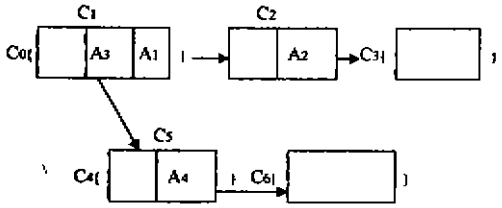

图1　对象复合模式

集合类 C_0的键模嵌套表 T 可为如下形式：

1)T 为空表，即：T 没有定义；表示选择操作不深入到 C_0类的嵌套集合类的集合对象上；

2)$T=(K_1\ W_1\ T_1,K_2\ W_2\ T_2,\cdots,K_n\ W_n\ T_n)$。其中，$K_1,K_2,\cdots,K_n$为 C_0的键模；$W_1,W_2,\cdots,W_n$分别为针对基于 $K_1,K_2,\cdots,K_n$的嵌套集合类的集合对象的选择条件。选择条件可为空。$T_1,T_2,\cdots,T_n$分别为基于 $K_1,K_2,\cdots,K_n$的嵌套集合类的键模嵌套表。键模嵌套表可为空。

例2　就图1来说，对于集合类 C_0，其键模嵌套表 T 可为$(A_3\ W_1\ T_1,A_1.A_2\ W_2\ T_2)$。该表中共有2个表项 $A_3\ W_1\ T_1$和 $A_1.A_2\ W_2\ T_2$。基于 A_3的嵌套集合类为 C_4，基于 $A_1.A_2$的嵌套集合类为 C_3。W_1是针对 C_4类集合对象中的元组对象的选择条件，W_2是针对 C_3类集合对象中的元组的对象的选择条件。T_1是集合类 C_4的键模嵌套表，T_2是集合类 C_3的键模嵌套表。T_1可为$(A_4\ W_3\ T_3)$，T_2和 T_3为空表。

就例2来说，若 r_n是类 C_n的集合对象，则 $\sigma(r_0\ W\ T)$的操作过程的语义如下：

1)产生一个与 r_0同类的空集合对象 r'，并从 r_0中选出满足 W 条件的对象 t_0并置入 r'中，使 r'的状态为$\{t_0\mid t_0\in r_0\wedge satisfy(t_0,W)\}$。其中，谓词 $satisfy(t_0,W)$为：t_0满足 W；

2)对于 r'中的每一个元组对象 t_0，针对 T 中的表项 $A_2\ W_1\ T_1$执行 $\sigma(t_0.A_3\ W_1\ T_1)$；

2.1)产生一个与 $t_0.A_3$同类的空集合对象 r_4，r_4的类为 C_4，并从 $t_0.A_3$中选出满足 W_1条件的对象 t_4并置入 r_4中，使 r_4的状态为$\{t_4\mid t_4\in t_0.A_3\wedge satisfy(t_4,W_1)\}$；

2.2)对于 r_4中的每一个元组对象 t_4，针对 T_1中的表项 $A_4\ W_3\ T_3$执行 $\sigma(t_4.A_4\ W_3\ T_3)$；

2.3)将对象 r_4置入 $t_0.A_3$。

3)对于 r'中的每一个元组对象 t_0，针对 T 中的表项 $A_1.A_2\ W_2\ T_2$执行 $\sigma(t_0.A_1.A_2\ W_2\ T_2)$。

从上述 $\sigma(r_0\ W\ T)$的操作过程，我们可看到三点：

①选择操作可沿键模深入到子集合对象上进行选择操作。因此，选择操作在作用于多个集合对象后，被操作的集合对象之间的复合模式不变；

②操作深入的次序是深度优先法；

③由于子集合对象是嵌入到一个元组对象的属性中的，因此对该子集合对象的选择操作的结果会改变元组对象的属性值，也即：嵌套选择操作会对 OODB 中的对象产生修改的副作用。这种广义的查询方式也为查询语言 OODAPLEX 所采纳[6]。

$\sigma(r\ W\ T)$的形式化描述如下：

```
algorithm σ(r W T)
begin
当 r 为 null 时，
    返回对象 null；
当 r 不为 null，而 T 为空时；
    产生与 r 同类的空对象 r'；
    使 r' 的状态为{t|t∈r∧satisfy(t,W)}；
    返回 r'；
当 r 不为 null，T 不为空时；
    产生与 r 同类的空对象 r'；
    使 r' 的状态为{t|t∈r∧satisfy(t,W)}；
    for all t∈r' do
    for T 中的每一个表项 I do
/* I_Att 表示 I 中的键模；I_CON 表示 I 中的条件；I_T
   表示 I 中的键模嵌套表 */
   将 σ(t.I_Att I_CON I_T)返回的对象置入 t.I_Att
   中；
    返回对象 r'；
end
```

结束语　在 OODB 的查询操作中，选择操作的表达能力至关重要。本文定义了嵌套选择操作，并描述了操作的语义。借助键模嵌套表导航的表达方式，

维普资讯 http://www.cqvip.com

计算机 体系结构 多线程体系结构 (18)

计算机科学1998 Vol. 25 №. 4

70-76

多线程体系结构现状及发展

Multithreading Architecture: Current State and Prospect

TP303

肖 刚 徐 明 周兴铭

(国防科技大学并行与分布处理国家重点实验室 长沙410073)

摘 要 This paper divides the multithreading architecture into four kinds: multiple context processor, dataflow/von Nuemann hybrid architecture, multithreading MPA node and multithreading superscalar, then surveys the their state, analyses their features, and prospects their future.

关键词 Multithreading, Dataflow, Superscalar

一、引言

多线程体系结构结合了数据流结构和传统的冯氏控制流结构，既保持了指令执行的高性能，又实现了处理器的高效率，是一种通用而高效的延迟隐藏技术。早期的多线程体系结构可以追朔到 CDC 6600 和 HEP，现今的多线程处理器中的很多技术都可以在它们那里找到踪迹。CDC 6600的 I/O 单元最早采用了多线程的技术思想，它允许10个虚拟的 I/O 处理器共享一个10级的中心流水线，处理器按轮转方式调度，每个处理器每10个周期发射一条指令。HEP 是最早的商用多线程处理器，它的结点处理器轮转地调度线程运行，每周期发射一条指令，流水线长8级。为了使流水线填满，HEP 需要大量的线程，HEP 可以支持128个线程并发执行。

如今，多线程体系结构又引起了广泛的兴趣，在 MPP、数据流处理器和微处理器领域都对它展开了更深入的研究。本文介绍多线程计算模型及其体系结构，并讨论了多线程体系结构新近的发展现状及对未来的展望。

二、多线程计算模型

多线程计算模型是数据流模型和控制流模型的结合。线程是一段顺序指令序列，线程中指令的执行按控制流方式执行，而线程作为调度的基本单位，按数据流驱动方式并行执行。

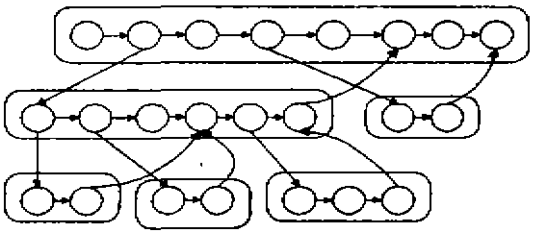

图1 多线程计算模型示例

一个多线程的计算由一组线程构成。每个线程是一个顺序任务(如指令)序列。如图1，每个块是一个线程，其中圈代表任务；水平边代表顺序关系，称为连续边。线程中的任务必须按照顺序关系从头(最左)至尾(最右)地执行。为了执行线程，给它分配一块存储器，称为活动帧，线程中的任务可以用之存储计算值。线程执行过程中可以生成其他线程。生成一个线程类似于一次过程调用，只是被调用的过程和调用过程可以并发执行。这里认为被生成线程是生成线程的子线程，而且一个线程可以生成任意多个子线程。这样，线程就由生成边连接成一个层次的激活树。每个生成边从父线程中进行生成操作的任务

实现了选择查询产生的多个集合对象之间保持原有复合关系的语义表达，避免了为保持原相应的集合对象的复合关系而使用重构这种过程性的表达方式。

主 要 参 考 文 献

[1]The Object Database Standard: ODMG-93, edited by R. G. G. Cattell, Morgan Kaufmann Publishers

[2]J. V. D. Bussche and A. Heuer, Using SQL with Object-Oriented Database, Info. Sys. 18(7)1993

[3]G. M. Shaw and S. B. Zdonik, A query algebra for object-oriented database. Proc. 6th Int. Conf. Data Engineering, 1990

[4]N. Bhalla and S. Balasundaram, Operation and Queries in Object-Oriented Database Supporting Complex Objects, Information and Software Technology, 35(1)1993

[5]钟武、胡守仁. OODB 数据模型及查询代数，计算机科学，24(2)1997

[6]U. Dayal, Queries and Views in an Object-oriented Data Model. Proc. 2nd Intel. Workshop on Database Programming Languange, 1989