ORDBMS 中动态模式修改的研究*

李红燕[1,2] 李战怀[3]

[1](北京大学信息科学中心 北京 100871)

[2](北京大学视觉与听觉信息处理国家重点实验室 北京 100871)

[3](西北工业大学计算机科学与工程系 西安 710072)

E-mail: lihy@cis.pku.edu.cn

摘要 继承性是对象-关系型数据库管理系统(ORDBMS)的一个重要特征.它提供了强有力的增量建模能力,也增加了模式演变的复杂度.通过对完全继承语义进行进一步的探讨,提出 ORDBMS 中动态模式修改(dynamic schema modification,简称 DSM)的准则化模型,并通过继承层次选择表达式(inheritance hierarchy selective expression,简称 IHSE)概念的引入,使子类在超类模式改变时,可以自由地选择级联改变或迁移策略.

关键词 对象-关系型数据库管理系统,继承性,类,动态模式修改,继承层次选择表达式.

中图法分类号 TP311

动态模式修改(dynamic schema modification,简称 DSM)又称为动态模式进化,是指在 DBMS 运行过程中,以一致性方式对数据库模式进行改变,并将这种改变传播到模式中的数据对象上去的处理过程[1].动态模式修改能力在数据库应用系统开发过程中显得非常重要.

DSM 的实现难度与数据模型密切相关.在规范关系数据库系统中,DSM 语义比较简单,并且由于关系的独立性,一个关系的改变不影响其他关系和其中的元组.在 OODBMS 中,由于类层次间的继承性,对某个类的改变不仅影响到自身,还可能影响到其所有子类和其中的实例.而在 ORDBMS[2]中,也正是由于对继承性的支持,其 DSM 实现与 OODBMS 存在类似之处.

典型的 OODBMS,如 ORION[4],GemStone[4]等对于 OODBMS 中的 DSM 实现皆有不同程度的贡献.GemStone 只支持单继承,其模式修改语义虽然简单,但却经常造成信息冗余,并且数据模式也不太直观.ORION 支持多重继承,其 DSM 内容被认为是最丰富的[5],但 ORION 对于 DSM 的描述采用不变式及规则模型.在不同数据库系统中,由于数据模型的差异,所选用的不变式及规则集也各不一样,ORION 本身就定义了 5 个不变式及 12 条规则[3].在这种模型下,系统之间往往由于缺乏统一性描述而使其相互比较变得困难.在文献[1]中提出了一种有关 DSM 的公理化模型,该模型中包括 9 条公理,可适用于纯粹的 OODB,对于 ORDB 却并不是很适合,因为 ORDB 一般都支持多重继承及完全继承语义,并且对全继承语义的理解和 DSM 实现还有一些独到之处.

本文针对 ORDBMS 的具体特征来探讨基于多重继承及完全继承语义的 ORDBMS 中 DSM 底层机制的共性.与 OODB 不同的是,文章对全继承语义进行了细分,认为 ORDB 中类的不同成员可采取不同的全继承策略,以便能更好地适应实际需求,但这也使得模式演变更为丰富和复杂.此外,我们还对 DSM 在继承层次中的传递策略进行了研究,提出了继承层次选择表达式(inheritance hierarchy selective expression,简称 IHSE)的概念,使用户可以自由选择级联改变和迁移两种策略,IHSE 还使得 SQL(structured query language)操作目标得以从传

* 本文研究得到国家 863 高科技项目基金(No. 863-511-946-002)和国家重点基础研究发展规划项目(No. G1999032705)资助.作者李红燕,女,1970 年生,博士,主要研究领域为数据库系统与 Internet 应用软件.李战怀,1961 年生,博士,教授,主要研究领域为数据库系统.

本文通讯联系人:李红燕,北京 100871,北京大学信息科学中心

本文 2000-01-04 收到原稿,2000-04-12 收到修改稿

统关系系统中某个特定表扩展到某继承层次中的表集合.在此基础上,我们提出了ORDBMS中DSM的准则化模型.

1 准则化模型

为了保证模式修改前后数据库系统的一致性,DSM应该遵循一些原则.下面,我们将给出ORDBMS中有关DSM的准则化模型.

为便于说明,先用表1给出本文经常使用的符号及其含义.表1中最后3项是定义的3个操作.u操作是一种扩展的并操作,$A \cup B$的结果为集合A,B中的所有元素,对于相同元素可重复出现.为保证结果集中相同元素的可区分性,可对其附加一些信息,如给出各自的来源等.例如,$A=\{1,2\}$,$B=\{2,3\}$,则$A \cup B=\{1,2_{(A)},2_{(B)},3\}$.在后两种操作中,参数$Y$为一个集合,且$Y \subseteq \Gamma$;$f$为作用于变量$x$上的函数,$x$取值域为$Y$.$\alpha$操作含义为:初始化$Y''=Y$.对$Y$中每个元素作用函数$f$后,都得到一个集合,取$Y'$为这些集合的并集,如果$Y'=\varnothing$,则操作结束,得到$\alpha$操作结果集$Y''$;否则以$Y'$取代$Y$,以$Y'' \cup Y'$取代$Y''$,继续执行$\alpha$操作.$\beta$操作的含义为:对$Y$中每个元素作用函数$f$后都将得到一个集合.$\beta$操作不单独出现,根据其前面的操作符是"$\cup$"还是"u"来决定最终结果是上述所有集合的并还是扩展并.特别地,如果$Y=\varnothing$,那么α和β操作的结果集为空.

Table 1 Common symbols and their meaning

表1 常用符号及其含义

Symbol①	Meaning②
Γ	The set of all types of a system③
$P(T)$	The set of all supertypes of T④
$P_i(T)$	The set of all immediate supertypes of T⑤
$M_p(T)$	The set of private members of T⑥
$M_i(T)$	The set of inherited members of T⑦
$M(T)$	The set of members of T⑧
$O(M)$	The origin of member M⑨
$VM_{i_A}(T)$	The set of valid inherited members relevant to attributes of T⑩
$VM_A(T)$	The set of valid members relevant to attributes of T⑪
u	u operation⑫
$\alpha(f(x),Y)$	α operation⑬
$\beta(f(x),Y)$	β operation⑭

①符号,②含义,③系统中所有类的集合,④类T的所有超类集合,⑤类T的所有直接超类集合,⑧类T的私有成员集合,⑦类T的继承成员集合,⑧类T的成员集合,⑨成员M的源,⑩类T上与属性相关的有效继承成员集,⑪类T上与属性相关的有效成员集,⑫u操作,⑬α操作,⑭β操作.

准则1(类的可区分性准则). Γ中的任意两个类都是可区分的.

定义1. 对于两个类$T_1 \in \Gamma$,$T_2 \in \Gamma$来说,如果存在(T_1 IS-A T_2)关系,则称T_1为T_2的子类,或称T_2为T_1的超类,记为$T_1 \preccurlyeq T_2$.

准则2(超类集的封闭性准则). 对于$\forall T \in \Gamma$,$P(T) \subseteq \Gamma$.

准则3(非循环性准则). 对于$\forall T \in \Gamma$,$T \notin P(T)$.

定义2. 如果$T_1,T_2 \in \Gamma$,且$T_1 \preccurlyeq T_2$,那么当且仅当没有这样的x存在,使得$x \in \Gamma-\{T_1,T_2\}$,$T_1 \preccurlyeq x$,$x \preccurlyeq T_2$都成立,就称T_1为T_2的直接子类,或T_2为T_1的直接超类.若有这样的x存在,则称T_1为T_2的间接子类,或T_2为T_1的间接超类.

准则4(超类集准则). 对于$\forall T \in \Gamma$,$P(T)=\alpha(P_i(x),P_i(T))$.

子类和超类以不同抽象层次来描述事物间的IS-A联系,超类比子类抽象层次更高.子类对象除了具有其超类对象所拥有的特性(属性、用户定义函数(user defined function,简称UDF)、触发器、约束)以外,还可具有它自己所独有的特性.由于这些特性都依赖于某个类而存在,因此被称作类的成员.

类的成员按继承性被分成两类:某个类T上继承来的成员被称为继承成员,而其自身定义的成员则称作自

定义成员或私有成员,T 称作其自定义成员的源.如果 T 中某些成员来源相同,则称它们为同源成员.

准则 5(成员准则). 对于$\forall T\in\Gamma, M_p(T)\cap M_i(T)=\varnothing$,并且 $M(T)=M_p(T)\cup M_i(T)$.

准则 6(成员的源确定性准则). 对于$\forall T\in\Gamma,\forall M\in M(T)$,如果 $M\in Mi(T)$,则 $O(M)\in P(T)$;如果 $M\in M_p(T)$,则 $O(M)=T$.

准则 7(成员的可区分性准则). 对于$\forall T\in\Gamma$,T 上每类成员中都必须是两两可区分的:按名访问成员必须按名区分,非按名访问成员可按名和源共同区分.

定义 3. 继承层次 Ψ 是这样的一个 Γ 的子集:对于$\forall T_1\in\Psi$,必然存在 $T_2\in\Psi-\{T_1\}$,使得 $T_1\leqslant T_2$ 或 $T_2\leqslant T_1$ 成立.

由于继承层次是基于 IS-A 联系而构成的,因此其中的类之间具备偏序关系.每个继承层次都可以构成一个带根的有向无环图(directed acyclic graph,简称 DAG).图中所有顶点均对应于系统中某个已存在的类,而有向边则表示相邻两个顶点之间所存在的 IS-A 关系.如果 T_1 为 T_2 的直接子类,则存在从顶点 T_2 到 T_1 的边;若 T_1 为 T_2 的间接子类,则存在从顶点 T_2 到 T_1 的长度大于 1 的路径.

现在利用继承层次所构成的 DAG 来证明准则 4 的正确性.

定理 1. 准则 4 是可终止的、完备的和有效的.

证明:对于$\forall T\in\Gamma$,构造 T 所在的继承层次的 DAG,设该 DAG 的根为 RT.

(1) 若 T 为 RT,则 $P_i(T)=\varnothing$,$\alpha(P_i(x),P_i(T))=\varnothing$,所以 $P(T)=\varnothing$.准则 4 是正确的.

(2) 在 T 不是 RT 的情况下,设在 DAG 中从 RT 到 T 的最大路径长度为 $L(L\geqslant 1)$,则必然在经过不超过 L 步的替换操作之后,使 $\alpha(P_i(x),P_i(T))$操作中的 $P_i(T)-\{RT\}$.由 α 操作的终止性可推知,准则 4 是可终止的.

根据 α 操作的定义与定义 2 可知,在 $\alpha(P_i(x),P_i(T))$结果集中,每个元素都是 T 的直接超类或间接超类,所以有效性成立.

又设 P 为 $\alpha(P_i(x),P_i(T))$的结果集,易见对于$\forall T'\in P_i(T)$,有 $T'\in P$ 成立.对于 T 的任意间接超类 ST,必然在 DAG 中存在一条由 ST 到 T 的路径,不妨设为 $ST\rightarrow\ldots\rightarrow T_i\rightarrow\ldots\rightarrow T_1\rightarrow T(i\geqslant 2)$,由于 $T_1\in P_i(T)$,所以 $T_1\in P$.对于路径中任意的 T_i,经过有限次运算后,必然属于 P,这将最终导致 ST 在 P 中.由 ST 和 T' 的任意性知,$P=P(T)$.完备性得证. □

由于对多重继承的支持,子类与超类,或同一子类的多个超类之间就可能因存在同名成员而导致继承冲突.

(1) 子类与超类定义了同名成员

支持继承性的 DBMS 都采用重定义(或称重载)来解决这种冲突.重定义屏蔽了超类的同名成员,使子类拥有自己的具体语义,从而表征了子类所具有的特殊性.

(2) 同一子类的多个直接超类上具有同名成员

此时又可分为两种情形:① 如果这些同名成员同源,则实际并不发生冲突,子类可以对它们直接继承;② 对于非同源类的同名成员,系统必须采用一定的策略来解决继承冲突.

根据系统对成员继承冲突的处理方式,可将完全继承语义进一步细分成以下两种.

(1) 冲突选择式完全继承

这是指当发生成员的同名继承冲突时,根据某种策略来选择其中一个,以使成员语义保持唯一.常用的策略是按照特定优先级次序加以选择.优先级又可分为显式和隐式两类.其中后者是指在定义子类时所给出的超类列表中,越是排在前面的超类优先级越高;而前者是指采用系统提供的显式优先级定义来强制某个(些)成员的源为某特定超类.

准则 8(优先级准则). 在采用优先级方式解决成员继承冲突的情况下,系统应该采取的次序为:重定义(重载)$\xrightarrow{先于}$显式优先级$\xrightarrow{先于}$隐式优先级.

准则 9(继承性准则 1). 在冲突选择式完全继承方式下,对于$\forall T\in\Gamma$,$M_i(T)=\bigcup\beta(M(x),P_i(T))$.

在该方式中,如果子类继承了某个特定成员 C,就称 C 在该子类上对于其他超类的同名成员产生继承屏蔽.根据准则 9,对于$\forall T\in\Gamma,\forall T'\in P(T),\forall M\in(T')$,在与 T' 上 M 同类的成员中,T 上肯定存在且仅存在一个名为 M 的成员.所以,判断 T 对 T' 成员 M 是否产生继承屏蔽,可以通过检查 T 上 M 的源是否等于 T' 来定.

值得讨论的是,如果在属性继承语义上采用冲突选择式完全继承,由于属性的继承屏蔽,将可能导致与属性有关的其他继承成员的继承无效.例如,UDF 就是定义在某个类上,并以该类中的原子属性为形参的,其作用范围为该类中的对象.在使用时,UDF 以类中对象的相应属性值为实参.如果一个类上继承来了某个 UDF,但对于该 UDF 所涉及到的属性却产生了继承屏蔽,那么该 UDF 在这个类上就是无效的.以下的准则 10~12 以及算法 1 用于判定在这种属性继承方式下,其他成员的有效性问题.

准则 10(私有成员的有效性准则). 对于$\forall T\in\Gamma$,如果$M\in M_p(T)$,则 M 在 T 上是有效的.

对于$\forall T\in\Gamma$,T 上与属性相关的有效继承成员集$VM_{i_A}(T)$可用算法 1 得到.

算法 1. 计算类 T 上与属性相关的有效继承成员集$VM_{i_A}(T)$.

输入:T 所在的继承层次

输出:$VM_{i_A}(T)$

方法:① 构造 T 所在继承层次的 DAG.

② 初始化$VM_{i_A}(T)=M_i(T)$.$M_i(T)$可根据准则 9 算出.

③ 根据准则 4 计算 $P(T)$.对于$\forall T'\in P(T)$以及 T'中每个私有属性 A,检查 T 是否对 A 产生继承屏蔽.若是,则转④;否则什么也不做.

④ 在 DAG 中找出从 T'到 T 所有路径中的所有结点集合$(T_1,T_2,\ldots,T_m)(m\geqslant 1)$.注意,$T'$在该集合中,但 T 不在该集合中.然后找出每个 $T_k(1\leqslant k\leqslant m)$上与属性 A 相关的私有成员集合 $M_{p_A}(T_k)$,以 $VM_{i_A}(T)-M_{p_A}(T_1)-\ldots-M_{p_A}(T_k)-\ldots-M_{p_A}(T_m)$取代 $VM_{i_A}(T)$.

定理 2. 算法 1 是可终止的、有效的和完备的.

证明:① 对于$\forall T\in\Gamma$,$P(T)$是有限集;对于$\forall T'\in P(T)$,T'的私有属性集是有限的.加之 Γ 也是有限集,这保证了算法的可终止性.

② 有效性要保证根据算法所得到的 $VM_{i_A}(T)$中不存在错误成员.首先,根据 $VM_{i_A}(T)$的构成可知,$VM_{i_A}(T)\subseteq M_i(T)$,也即 $VM_{i_A}(T)$中不存在非继承成员.其次,证明 $VM_{i_A}(T)$中不存在无效继承成员:设 M 为 $M_i(T)$中任意一个无效成员,那么根据准则 10 或准则 5 知 $M\notin M_p(T)$,且 T 上至少要对某一个与 M 相关的属性 A 产生继承屏蔽.设 $O(A)=T_1$,$O(M)$只可能为 T_1 或 T_1 的子类,而不可能是 T 的子类,因此,$O(M)$属于从 T_1 到 T 的所有路径中的所有结点集合.这样,当步骤③中 T'的取值为 T_1 时,必然能在步骤④中将 M 从 $VM_{i_A}(T)$中删除.可见算法是有效的.

③ 完备性用于保证 T 的所有有效继承成员都在算法所得到的结果 $VM_{i_A}(T)$中.设 M 为 T 中任意一个有效继承成员,A 为与 M 相关的任意属性.当步骤③中 $T'=O(A)$时,由于 T 必然继承 A(否则与 M 的有效性相矛盾),因此步骤④就不会执行.由 M 与 A 的任意性知,算法是完备的. □

准则 11(与属性相关成员的有效性准则). 对于$\forall T\in\Gamma$,设 $M_{i_A}(T)$为 T 上与属性相关的私有成员集,则 $VM_A(T)=VM_{i_A}(T)\cup M_{p_A}(T)$.

准则 12(与属性相关有效成员的作用域准则). 对于$\forall T\in\Gamma$,$\forall M\in VM_A(T)$,如果 $M\in M_p(T)$,那么 M 的作用域为 T 的属性集;如果 $M\in VM_{i_A}(T)$,那么 M 的作用域仅为 T 的继承属性集.

(2) 无条件式完全继承

在该方式下,无论冲突与否,子类都将继承其所有超类的所有成员.这可能会导致子类上存在多个同名成员的情况.

在大多数 OODBMS 中,所有成员的冲突处理都采用冲突选择式完全继承方式.我们认为,在 ORDBMS 中,应根据不同成员采取不同方式.比如说,对于按名访问的成员(如属性、UDF),拟采用冲突选择式完全继承;而对于非按名访问的成员(如约束、触发器)则更适宜于采用无条件式完全继承.

准则 13(继承性准则 2). 在无条件式完全继承方式下,对于$\forall T\in\Gamma$,$M_i(T)=u\beta(M(x),P_i(T))$.

准则 14(继承成员的非修改性准则). 在任何类上,都不能对其继承成员进行修改或删除操作.在子类中,只有对继承成员的继承优先级改变是合法操作.最后这条准则是为了保证全继承语义的正确性.

2 模式修改在继承层次中的传递

当模式发生变化时,模式中所存放的数据项(数据对象)也应作相应转变,以与模式保持一致,这种由模式改变而导致的数据对象改变被称为模式修改的传播.有3种技术实现模式修改的传播:屏蔽(screening)、转换(conversion)和过滤(filtering).文献[1]对此有详尽阐述.另一方面,当超类表模式发生改变时,将可能影响到子类表模式结构以及其中数据对象.一般说来,在一个确定的继承层次中,超类表模式的改变在子类中的传递可以采取两种策略.一种策略是级联改变.子类模式将依据超类做同样的改变.另一种策略是迁移.超类模式改变不影响到子类,也即超类的原有模式将迁移到其直接子类上.

这两种策略各有其现实意义.在支持全继承语义的OODBMS(如ORION)中,由于屏蔽策略将导致某些模式修改(如超类成员的增加)对全继承语义造成破坏,故只支持级联改变策略,这在很多情况下很不灵活.在ORDBMS中,用户应该可以根据实际需要,在不违反继承语义的前提下,自由选择上述两种策略.而继承层次选择表达式正是达到这一目的的一种有效手段.

定义4. 继承层次选择表达式用于从某个特定的继承层次中选出需要同时被操作的类.其文法表示可按如下规则递归地定义为

⟨IHSE⟩::=⟨tname⟩'*'[but '('⟨iname_list⟩')']

⟨iname_list⟩::=⟨iname⟩|⟨iname_list⟩','⟨iname⟩

⟨iname⟩::=⟨tname⟩|⟨IHSE⟩

IHSE的结果是一组类的集合,该集合中每个元素都必须是相应继承层次所构成的DAG中的顶点.IHSE的含义为:① tname后跟*表示类tname及其所有子类的集合;② but子句中的逗号分隔符表示其前后两个集合的并运算;③ but子句表示其前后两个集合的差运算.

IHSE刻画了某次操作中一个特定超类及其相关子类之间的概括语义.在使用时,IHSE的适用范围是表名适用范围的真子集.使用IHSE的典型命令有:查询、删除、修改、DDL(data definition language)和DCL(data control language)命令.这样,IHSE使ORDB中的SQL操作目标得以从传统RDB中的某个特定表扩展到某继承层次中的表集合.

2.1 迁移传递

例如:"ALTER TABLE T DROP a_i"表示仅从T表上删除属性a,但T的所有子类表模式均不作改动,系统将采取迁移策略,自动将所有继承了该属性的直接子类上的a属性类型由继承属性改为私有属性.

2.2 级联改变传递

"ALTER TABLE T* DROP a_i"则表示上述删除操作将级联作用到T的所有子类表(包括间接子类表)上,对所有未对此子表属性进行重定义或继承屏蔽的子类表上的relations子表结构进行删除之后,在这些表数据中相应于relations子表的取值项也同样予以删除.

2.3 级联改变-迁移混合传递

如果用户想使上述删除操作只作用于某些子类表上,而屏蔽掉另一些子类表,就可以采用另一种继承层次选项——"*(BUT)"来综合运用级联删除及迁移策略.例如,"ALTER TABLE T* (BUT ST*) DROP a_i"的作用就是在除ST及其所有子类以外的其余所有在继承层次中的表上进行属性删除操作,同时,a属性将迁移到ST上,成为其私有属性.

3 DSM的分类

在ORDBMS中,对表模式改变的操作可有如下分类.

3.1 对表成员的改变

3.1.1 增加表成员的操作类

当往表中增加成员时,根据准则7,需满足这样的限制:在任意表上所有属性不能重名,所有UDF不能重

名,所有私有触发器不能重名,所有有名的私有约束不能重名.此外,根据准则 5、准则 9 或准则 13 可知,当往超类 T 中增加一个新成员 M 时,T 的所有子类上都将增加一个相应的继承成员.在 M 采用冲突选择式完全继承的方式下,M 在 T 的子类上导致的继承冲突通过准则 8 解决.在增加了 M 的各个类中,根据准则 6 可知,$O(M)=T$.

3.1.2 修改/删除表成员的操作类

该类操作需满足准则 14 的限制.此外,在保证准则 5、准则 9 或准则 13 的前提下,超类成员的修改或删除在继承层次中的传递策略可通过 IHSE 来自由选择.特别地,当在一个类上删掉某属性之后,系统应根据准则 10~12、准则 4 及算法 1 自动识别并删除继承层次中的无效成员.另外,当某属性从超类迁移到子类时,与该属性相关的成员也可以一并迁移,如果它们在子类上不产生继承无效性的话.此外,如果类 T 上被删除成员 C 采用冲突选择式完全继承语义,并且 T 还有定义了与 C 同名成员的超类(即在删除前,T 上采用子类重定义方式),则需根据准则 8 确定出 T 上将继承哪个超类上定义的 C,并且在 T 的所有采取级联删除策略的子类上,都将增加新的 C 的继承定义.

3.2 对继承层次结构的改变

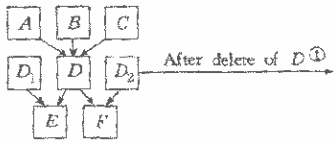
3.2.1 增加新类

当往某继承层次中增加一个新类 T 时,满足准则 1~3 的限制.对于 T 的任意一类成员的集合,可根据准则 5~13 确定.对于与属性相关的有效成员的判断,可根据准则 10~12、准则 4 及算法 1 来实现.

3.2.2 删除已有类

当删除一个继承层次中的类 T 时,需保持原有继承层次的完整性.设 T 的直接子类集为 $Sub(T)$,直接超类集为 $Sup(T)$.那么在 T 被删除后,对于 $\forall T' \in Sub(T)$,$Sup(T')=(Sup(T')-\{T\})\cup Sup(T)$;对于 $\forall T' \in Sup(T)$,$Sub(T')=(Sub(T')-\{T\})\cup Sub(T)$,并且在新的直接子类的直接超类中,原有优先级次序保持不变.

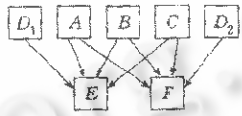
图 1 是该操作的一个示例.假设图 1(a)中 D 上直接超类优先级次序为 A,B,C;E 上直接超类优先级次序为 D_1,D;F 上直接超类优先级次序为 D,D_2.则在 D 被删除后,A,B,C 分别成为 E,F 的直接超类,且 E 的直接超类优先级次序为 D_1,A,B,C;而 F 的直接超类优先级次序则为 A,B,C,D_2.



①删除 D 之后.

(a) The inheritance hierarchy before delete of D

(a) 删除 D 前的继承层次



(2) The inheritance hierarchy after delete of D

(b) 删除 D 后的继承层次

Fig. 1 The example for deletion of class

图 1 类删除操作示例

此外,在删除一个类时,该类所有成员被一并删除.在保证继承语义正确性的前提下,系统可根据用户使用的 IHSE 判断该类成员是否迁移到其子类中去,以及迁移到哪些子类中去.

4 结 论

与 ORION 等典型 OODBMS 相比,本文提出了基于多重继承和完全继承语义的 ORDBMS 中 DSM 的准则化模型,并对完全继承语义进行了进一步的细分,提出了 IHSE 的概念以及对 DSM 在继承层次中传递策略的选择手段.相关工作已在 ANGEL 这个由西北工业大学软件工程中心完全从底层研制开发的 ORDBMS 中全部实现.限于篇幅,有关ANGEL系统的详细介绍请参阅文献[6].本文所提出的 DSM 准则化模型完全适用于其他 ORDBMS 以及基于完全继承、多重继承或单继承(可视为多继承的特例)语义的 OODBMS.

参考文献

1 Randal J, Peters M, Özsu T. An axiomatic model of dynamic schema evolution in objectbase systems. ACM Transactions on Database Systems, 1997,22(1):75～114

2 Stonebraker M. Object-Relational DBMSs—— The Next Great Wave. San Francisco: Morgan Kaufmann Publishers, Inc., 1996

3 Kim W, Garza J F, Ballou N *et al*. Architecture of the ORION next-generation database system. IEEE Transactions on Knowledge & Data Engineering, 1990,2(1):109～124

4 Breitl R, Maier D, Otis A *et al*. The gemstone data management system. In: Kim W, Lochovsky F H eds. Object-Oriented Concepts, Databases, and Applications. Reading, MA: Addison-Wesley Publishing Company, 1989. 283～308

5 He Yan-xiang, Zheng Zhen-mei, Shi Shu-gang. Object-Oriented Database. Wuhan: Wuhan University Press, 1995
(何炎祥,郑振楣,石树刚. 面向对象数据库. 武汉:武汉大学出版社,1995)

6 Li Hong-yan, Li Zhan-huai, Xu Qiu-yuan. The semantics and implementation of inheritance in ORDBMS. Journal of Computer-Aided Design & Computer Graphics, 2000,12(2):116～120
(李红燕,李战怀,徐秋元. ORDBMS 的继承语义及其实现. 计算机辅助设计与图形学学报,2000,12(2):116～120)

Research on Dynamic Schema Modification in ORDBMS

LI Hong-yan[1,2] LI Zhan-huai[3]

[1](*National Laboratory on Machine Perception Beijing University Beijing* 100871)

[2](*Center for Information Science Beijing University Beijing* 100871)

[3](*Department of Computer Science and Engineering Northwestern Polytechnical University Xi'an* 710072)

Abstract As an important property of ORDBMS, inheritance provides powerful capability for incremental modeling of composite objects. It also makes the DSM (dynamic schema modification) more complex. In this paper, the semantics of full inheritance is subdivided, and a criterion model of DSM in ORDBMS is presented. With Inheritance Hierarchy Selective Expression, the cascade strategy or migrate strategy used on subclasses can be chosen freely when the schema of their super classes is modified.

Key words ORDBMS, inheritance, class, DSM (dynamic schema modification), inheritance hierarchy selective expression.