计算机科学2003Vol. 30№. 10(增刊)

基于对象代理模型的面向对象数据库模式进化方法*)

Schema Evolution Methodology of Object-Oriented Databases Based on Object-Deputy Model

罗 慧 彭智勇

(武汉大学软件工程国家重点实验室 武汉430072)

Abstract Most of schema evolution methods of object-oriented databases are too complicated to be incorporated. In this paper, we propose a schema evolution methodology of object-oriented database based on object-deputy model. Object-deputy model can provide a unified realization of object views, multiple roles and object migration. It can also support schema evolution easily due to its flexibility.

Keywords Object-oriented, Object-deputy, Schema evolution

1 引言

模式进化是面向对象数据库领域内的一个重要问题。多个应用共享一个数据库时,任一应用程序的改变会影响其他应用程序的正常运行,模式必须不断进化以满足动态改变,支持多个应用程序对数据库的共享。目前的面向对象数据库的模式进化方法主要分为多版本和多视图两类方法。多版本的模式进化方法又包括模式版本和类版本两种。模式版本的方法(Orion[1])保存模式的所有版本,任何数据库结构的改变都将产生一个完整的新的模式版本;类版本的方法(Encore[2])则是保存类的所有版本,类的任何修改将产生该类及其所有子类的新的类版本。多视图的模式进化方法(TSE[3])中每个应用程序有一个对应的视图模式,应用程序的改变直接反映在它对应的视图模式。

传统面向对象数据库中的 IS-A 继承关系,由于类的改变会影响它所有的子类,使它的模式进化过程非常复杂。本文提出的基于对象代理模型的面向对象数据库模式进化方法由于没有 IS-A 关系约束,使得模式进化较容易实现,能有效地支持多个应用程序。

2 对象代理模型及其模式进化操作

常见的数据模型主要有层次模型、网状模型、关系模型和面向对象模型。面向对象模型因其丰富语义而被广泛应用于 CAD、多媒体处理、办公自动化等领域,但由于 IS-A 继承的局限使它难以达到与关系数据库同等的灵活性,也缺乏模拟实体多元化和动态本质的能力。对象代理模型[4,5]是对面向对象数据模型的扩充和发展。目前已在 SmallTalk 环境下得到实现,并在数据库集成方面表现出极大优势[6]。基于对象代理模型的数据库中只有类和代理类,可通过对象代理模型定义的代数操作导出的代理类来实现传统面向对象数据库中的父类和子类以及它们之间的 IS-A 继承关系。

对象代理模型引入了对象和代理对象这两个基本概念来模拟现实世界中的实体。代理对象是对象的扩充及变换,具有相同特性的代理对象用代理类表示。一个对象可以产生一个或多个代理对象,一个代理对象可以是多个对象的代理对象,代理对象可以继续产生代理对象。若 p 的代理对象是 q,则 p 叫做 q 的源对象。代理对象可选择性地继承源对象中的属性和方法,也可以追加定义自己的属性和方法。通过切换操作,代理类不仅可以继承源类中的属性和方法,还可以改变继承的属性和方法的名字和类型,实现了对源类的虚拟继承,即使用代理对象中继承的属性和方法实际上是通过调用切换操作使用源对象中的属性和方法。

对象代理模型定义了一系列的操作来支持模式进化。类似于关系代数的对象代理代数作用于源类可派生不同语义的代理类。其中,Select, Project, Extend 和 Grouping 作用于单个源类,分别具有选择、投影、追加定义和分组功能; Union 和 Join 作用于多个源对象,分别实现对多个源类对象的并集和聚集。删除类时,必须由系统确定是否调用系统操作 delete C 来删除类 C。类似于 SQL 语言的对象代理定义语言定义了泛化、特化、聚合、分组等各种语义,可定义各种语义的源类和代理类。选择恰当的语义和代数操作就可以对模式进行修改。此外,对象代理模型中还定义了 add(C,o), delete(C,o)和 write(o, a,v)三个基本操作用于修改面向对象数据库中的对

*)本文研究得到国家863项目(2002AA4Z3450),国家自然科学基金项目(60273072)及国家留学回国人员基金项目资助。

象。分别实现为类 C 增加对象 o，删除类 C 的对象 o，为对象 o 的属性 a 赋值 v，以保证模式进化后数据库的一致性。

3 对多应用的支持

基于对象代理模型的面向对象数据库的所有应用程序共享一个全局模式，任何模式的修改直接在全局模式上进行；因新增或修改应用程序引起的模式进化不影响其它应用程序的正常运行，修改后的模式仍然支持所有的应用程序。

3.1 定义新应用程序的模式进化方法

新增应用程序中使用的类与原应用程序的关系有下面三种情况，分别用不同的方法进行处理。

(1)新应用程序中使用的类已在原应用程序中被使用，则模式无需另作修改。

(2)新应用程序中使用的类在原应用程序中未被使用，需在全局模式中加入该类的定义。如果新增类是源类则直接定义为一个新的根类，否则从全局模式中已定义的类派生出所需的新代理类。

(3)新应用程序中使用的类与原应用程序中使用的类的名字相同但定义不同，全局模式中已存在与新应用程序所使用类的同名而定义不同的类，需要增加新的类定义。向全局模式中加入同名类时要重新命名以保证类名的唯一，应用程序中的名字冲突可通过名字空间解决。若新应用程序使用的类与原模式中的同名类相关，如新类继承了原模式中同名类的属性和方法，则由原模式中同名类派生出新代理类加入全局模式。若新应用程序中使用的类与原模式中的同名类无关，即只有名字相同，则根据需要将其定义为适当的根类或者代理类。

下例说明新增应用程序时的模式进化方法。本文所有图中仅标出第一次出现或者发生变化的属性，且代理类继承源类定义的所有属性和方法，a1是一个已定义的面向对象数据库的应用程序，a1使用了类 Person 和 Student，Student 是 Person 的代理类，当前的全局模式如图1-a。现新增一个应用程序 a2，使用类 Person，Employee 和 Student。其中，a2和 a1共享类 Person；Employee 是 a2中新定义的代理类；a2中的 Student 继承了 a1中 Student 类的属性和方法并追加了属性 major 和 advisor，因此在全局模式中定义为 Student 类的代理类，并重命名为 Student'。增加新应用程序 a2后的全局模式如图1-b，可同时支持应用程序 a1和 a2。

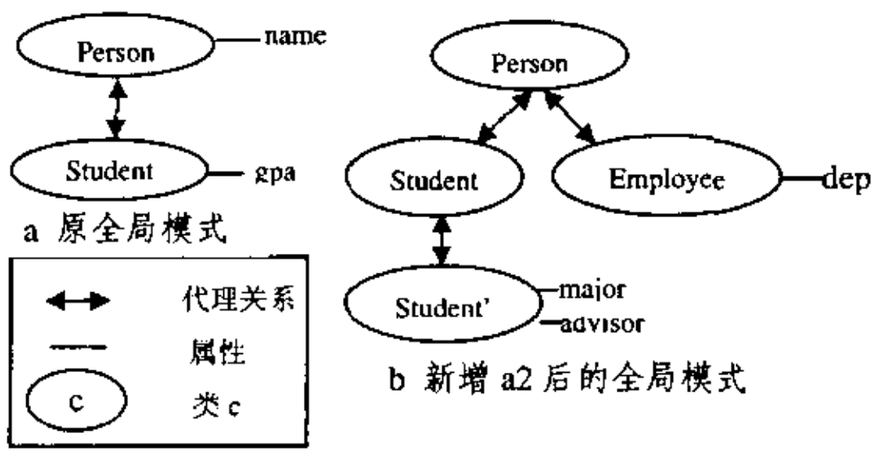

图1

3.2 修改已定义应用程序的模式进化方法

修改已定义的面向对象数据库应用程序，可能发生的操作有三种：修改应用程序中的类；在应用程序中增加类；从应用程序中删除类。

3.2.1 *修改类* 包括增加或删除类的属性和方法，改变属性和方法的名字或类型以及改变代理关系等。

如果被修改的类及其所有代理类仅在被修改的应用程序中使用，则直接在全局模式中对原来的类进行修改或者使用模式中已定义的类。若类被修改为新的定义，则原应用程序中被修改类的所有代理类也要进行相应的修改；若类被修改为模式中已定义的类，则修改原应用程序中被修改类的所有代理类后，从模式中删除被修改的类。

如果被修改的类或者它的代理类被其它应用程序使用，则保持全局模式中被修改类原来的定义。若类被修改为新的定义，则从原模式中的被修改类派生出一个新的代理类，新代理类的定义就是修改后的定义，同样的，被修改应用程序中所有以被修改类为源类的代理类要进行相应的修改；若类被修改为模式中已定义的类，只需修改应用程序中所有以被修改类为源类的代理类。

a1，a2是面向对象数据库的两个应用程序，数据库的全局模式如图2-a 所示，a1使用类 People、Student、Grad 和 TA，a2使用类 People、Student 和 UnderGrad。a1，a2共享 People 和 Student。现修改 a1。如将 a1中的类 Grad 和 Student 修改为与全局模式1中任何类的定义都不同，对于 Grad，由于它及其代理类 TA 只在 a1中使用，因此直接修改 Grad 和 TA，修改后全局模式如图2-b 所示；对于 Student，由于它同时被 a1，a2使用，因此不能直接修改 Student，而是由 Student 派生出一个新的代理类 Student1，又 a1中 Student 的代理类 Grad 和 TA 仅在 a1中使用，因此直接在代理类 Grad 和 TA 上进行修改，修改后的全局模式如图2-c。

3.2.2 *增加类* 应用程序中增加的类既可以是在其它应用程序中已被使用的类，也可以是所有应用程序中都未曾使用的新类，还可以是与其它应用程序中使用的类同名但定义不同的类。在模式中增加类的方法同3.1。如果新增类是被修改应用程序中某个代理类的源类，则需要用3.2.1中的方法修改相应的代理类。

应用程序 a1，a2共享全局模式(图2-a)。现需要在 a2中为 UnderGrad 创建一个代理类 TA，它与全

局模式中已定义的 TA 定义不同，且与它无关，因此在全局模式中新增一个代理类 TA_1 以支持 a2(图2-d 中实线部分)。继续修改 a2，为 TA_1 增加一个源类 Grad，Grad 在全局模式中已被定义(图2-d 中实线部分)，TA_1 的源类发生改变，且 TA_1 只在 a2中使用，因此修改 TA_1 直接增加代理关系(图2-d)即可。

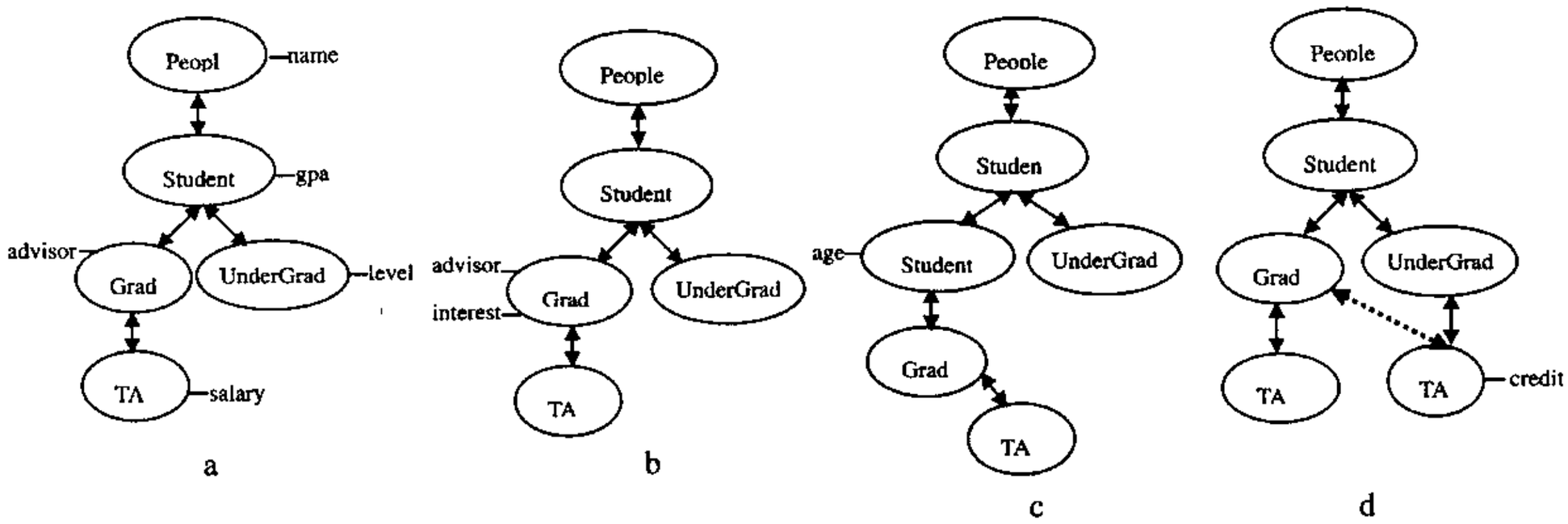

图2

3.2.3 *删除类* 仅当被删除的类及所有的代理类只在被修改的应用程序中使用时才允许从全局模式中删除相应的类定义，且删除该类后要修改全局模式中该类的所有代理类的定义。如果从应用程序中删除的类或者它的代理类在其它应用程序中被使用，则不允许从全局模式中删除这个类，而是对被修改应用程序中被删除类的所有代理类进行修改(方法同3.2.1)。这样修改后的模式仍然可以支持所有的应用。

a1，a2共享全局模式(图2-a)。如果从 a1中删除类 Grad，因为类 Grad 及其代理类 TA 只在 a1中使用，因此可直接从全局模式1中删除类 Grad，而代理类 TA 因为 Grad 的删除也要从全局模式中删除，修改后的全局模式如图3-a。如果要从 a1中删除 Student 类，因为 a1，a2共享类 Student，因此不允许直接删除类 Student，而必须修改 a1中以 Student 为源类的代理类 Grad 和 TA。因为 Grad 和 TA 只在 a1中使用，因此直接修改 Grad 和 TA，得到如图3-b 所示的全局模式。

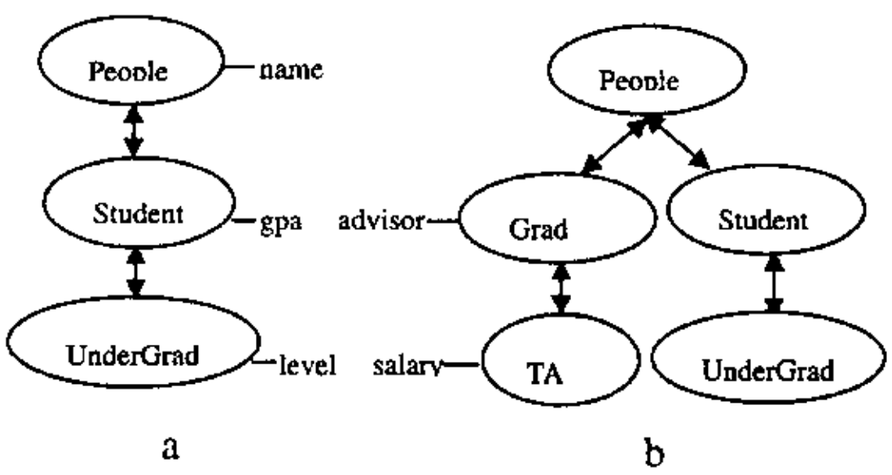

图3

结论 本文提出了基于对象代理模型的面向对象数据库的模式进化方法，能有效地支持多个应用，实现了数据的共享，使模式的修改大为简化，提高了面向对象数据库系统的柔软性。

参 考 文 献

1 Kim W, Chou H-T. Versions of Schema for Object-Oriented Database. In: Proc. of the 14th VLDB Conf. 1988. 148～149

2 Skarra A H, Zdonik S B. The Management of Changing Types in an Object-Oriented Database, Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications, 1986. 483～494

3 Ra Y-G, Rundensteiner E A. A Transparent Schema-Evolution System Based on Object-Oriented View Technology. IEEE, 1997. 600～624

4 Peng Zhiyong, Kambayashi Y. Deputy Mechanisms for Object-Oriented Databases. In: Proc. of IEEE 11th Int. Conf. on Data Engineering, 1994. 333～340

5 Peng Zhiyong, Kambayashi Y. Realization of Computer Supported Cooperative Work Environments Using the Object Deputy Model. In: Proc. of the Intl. Database Engineering and Applications Symposium, 1998. 276～284

6 Peng Z, Kambayashi Y. Resolving Conflicts and Handling Replication during Integration of Multiple Database by Object Deputy Model, ER 2001, LNCS 2224, 2001. 285～298