对象代理数据库语言*)

Object Deputy Database Languages

翟博譞　彭智勇　庄继峰

(武汉大学软件工程国家重点实验室　武汉大学计算机学院　武汉430072)

Abstract Object deputy module can provide more flexibility than traditional object-oriented data model. Based on it, we are developing a database management system called object deputy database system. In this paper, we introduce object deputy database languages including object deputy definition language and object deputy manipulation language, which are designed in style of SQL.

Keywords Object deputy model, Object deputy definition language, Object deputy manipulation language

1 引言

数据模型是数据库理论的重要基础,数据模型的优劣很大程度上决定了数据库系统的好坏。20世纪70年代,出现了基于关系模型的数据库。关系模型简单易理解,且具有坚实的理论基础,概念清晰明确,数据独立性强,在支持商业数据处理应用上十分成功。至今关系数据库仍然占有着统治地位。

但是,关系模型不能用一张表模型表示出复杂对象的语义,不擅长数据类型较多、较复杂的领域,不能提供多层次的抽象。因此,数据库模型又进入了新的研究阶段——面向对象数据库的研究。面向对象模型能够很好地表示复杂对象。封装和信息隐藏特性提供了模块化机制,而且由继承的概念大大提高了重用性。同时面向对象模型还具有很强的系统扩展能力。

虽然面向对象模型有以上种种优点,但是对象的封装性和信息隐藏性也降低了系统的柔软程度。对象作为一个封装的整体,不能任意地进行分割和组合。外界只能通过限定的方法来操作对象,导致为对象建立视图十分困难。用户在使用对象时缺乏灵活性。

对象代理模型[1,2]就是针对这种情况而提出的一个新的数据模型。与传统的面向对象模型相比,对象代理模型更具柔软性,操作更灵活,使用也更方便。

为了使用户能够方便地使用数据库系统,一个结构良好的用户接口——或者说,一套设计完善的数据库语言——是必不可少的。对象代理模型是对传统面向对象数据模型的扩展,要体现出对象代理模型的新特点自然需要对以往的数据库语言进行扩展。本文介绍了一套SQL标准[3]风格的对象代理数据库语言。

2 对象代理模型

对象代理模型的核心在于数据对象之间的代理。所谓"代理",是在一个对象(源对象)的所有或部分属性上进行一定的切换(switching)操作,如换名、算术运算等,而衍生出另外一个对象(代理对象)的过程。通过对象的代理,用户可以将不同的对象进行分割组合,形成新的对象,克服了原来的面向对象模型中,难以建立对象视图,对象操作不够灵活的缺点。具体来说,对象代理数据库有以下一些特性。

在组织形式上,源对象和代理对象都属于各自的类,从而形成了源类和代理类的概念。同一个代理类中的所有对象对其源对象的代理方式是一样的。它们拥有相同的模式和代理路径,因此被聚集在同一个代理类中。一个代理类在形式上和普通类一样。

在类的继承关系上,代理类对源类的"代理"和一般意义上的"继承"是有区别的。代理类更像是源类的视图,不同的代理对象表现出了源对象扮演的不同角色。一个源类上可以同时定义多个代理类;一个代理类也不仅限于代理一个源类,它可以通过连接或并操作同时在几个源类上进行代理。代理类上也可以再定义其它的代理类。代理关系是一个有向图的形状。位于代理层次最顶端的类称为基本类,它们是没有源类的代理类(本文中,基本类和代理类统称为"类")。

在存储模式上,基本类的对象占有存储空间,存储实际数据。数据库中大部分数据都存储在基本类

*)本文研究得到国家863(2002AA4Z3450),国家自然科学基金(60273072),教育部博士点基金(20010486029)和湖北省青年杰出人才基金(2002AC003)资助。

中。代理对象中的代理属性(称为“虚属性”)只拥有模式而没有实际的物理数据元。源对象和代理对象之间由双向指针来联系,并且由切换操作来确定代理对象中虚属性的值。代理对象可以扩展属于自己属性(称为“实属性”),这些扩展属性才真正占有存储空间。

在具体的操作上,创建代理类需要一个代理规则(一条查询语句)来定义它对源类进行的形式上的变换。为了保证代理关系的明晰,必须对代理规则进行一定的限制。在对象代理模型中,只允许四种简单的代理规则。①selection 代理:只从一个类中选择源对象进行代理。②join 代理:在几个类的连接结果中进行代理。③union 代理:将若干个 selection 代理的结果并在一个代理类中。④group 代理:在一个类上进行分组的代理,一个代理对象代理一组源对象。基本类和代理类的对象操作,在形式上是一样的。系统要保证代理类中的对象在操作前后始终符合代理规则的选择结果。

3 对象代理数据库语言

对象代理数据库语言分为对象代理定义语言和对象代理操作语言。

3.1 对象代理定义语言

对象代理定义语言的功能是让用户可以根据应用的需要来创建基本类,并在基本类上定义各种代理类,以及删除类。

(1) 定义基本类

基本类是没有源类的类,其所有属性都是实属性。在对象代理数据库系统中,它们最先被创建出来,是其它各种代理类的代理基础。基本类本质上和关系表一样,但是拥有方法。定义一个基本类的命令包括:定义类名、属性名和属性类型,建立约束,创建方法。其形式命令如下:

CREATE CLASS 〈class_name〉
[ATTRIBUTE] ({〈column〉 〈type〉
〈attr_constrain〉}),([〈class_constrain〉])
[METHOD {〈method_definition〉}];

其中:

〈class_name〉:类的名字;

〈column〉:属性名;

〈type〉:属性的数据类型;

〈attr_constrain〉:各种列约束语句;

〈class_constrain〉:各种类(表)约束语句;

〈method_definition〉:方法定义体,它由关键字 METHOD 引起。一个方法定义子句中可以有多个方法定义体,它们之间用逗号隔开。如果不需要方法,则省略方法定义子句。方法定义体的形式命令如下:

〈method_definition〉::=
CREATE 〈method_name〉 ({〈parameter_type〉})
RETURNS 〈return_type〉 AS '〈method_body〉' LANGUAGE '〈language〉'

其中:

〈method_name〉:方法名;

〈parameter_type〉:输入参数的数据类型表;

〈return_type〉:函数返回的结果类型;

〈method_body〉:方法体(程序代码);

〈language〉:方法体所使用的语言。

(2)定义代理类

代理类的定义包括定义类名,代理类型,扩展属性和方法,以及代理规则。其形式命令如下:

CREATE
SELECTDEPUTYCLASS | JOINDEPUTYCLASS | UNIONDEPUTYCLASS | GROUPDEPUTYCLASS 〈class_name〉 [[ATTRIBUTE] ({〈column〉 〈type〉[attr_constrain]},[〈class_constrain〉])]
[METHOD {〈method_definition〉}]
AS〈deputy_rule〉;

定义代理类语句和定义基本类语句的不同之处在于:第一,定义代理类名字前的四个关键字分别对应了四种代理类型;第二,ATTRIBUTE 子句用来定义代理类的扩展属性。它可以完全省略,表示不进行属性扩展。第三,代理类必须有代理规则部分。代理规则要符合一定限制。例如 selection 型代理规则中不能有分组聚集操作和集合操作等,而且只能在一个源类上进行选择。从代理规则的目标表达式中,系统可以提取出代理类的虚属性模式,以及虚属性的切换操作。

(3)删除类

通过删除类的命令,用户可以将任何层次上的类从数据库中删除。其形式命令如下:

DROP CLASS 〈类名〉;

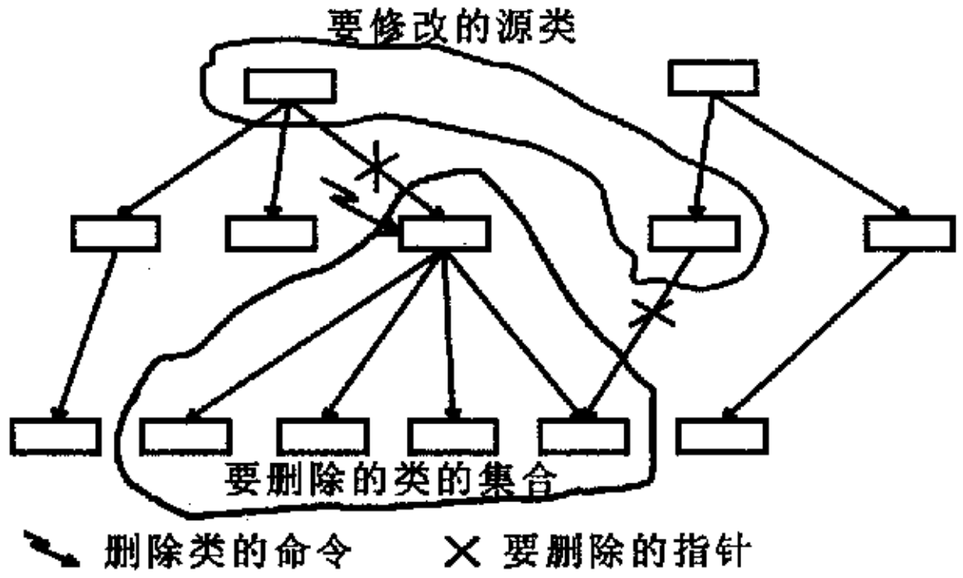

图1

在对象代理数据库中删除一个类时,系统先采

用图的广度优先遍历的方法扫描代理关系结构，得到该类的所有代理类的集合。然后删除这个集合中的所有类的基本信息，同时修改涉及到的源类，如图1所示。

3.2 对象代理定义语言举例

例1：定义一个基本类 person 表示所有的人。它的5个属性分别表示人的姓名、性别、出生年份、住址以及职业。方法 person_age(INT)用当前的年份减去出生的年份，计算人的年龄。

```
CREATE CLASS person
ATTRIBUTE
(
  name        CHAR() NOT NULL UNIQUE,
  sex         CHAR NOT NULL,
  birth       INTEGER,
  address     CHAR(),
  occupation  CHAR()
)
METHOD
(CREATE person_age(INT)RETURNS INT AS
'select CURYEAR—$1 :: INTEGER' LANGUAGE 'sql');(注:CURYEAR 表示当前的年份)
```

定义一个基本类表示公司，属性为公司名字、地址、资产和老板的姓名，不含方法。

```
CREAT CLASS company
ATTRIBUTE
(
  name      CHAR() NOT NULL UNIQUE,
  address   CHAR(),
  asset     INTEGER,
  boss      CHAR()
);
```

例2：为 person 建立一个 selection 代理类表示所有的雇员。它选择所有职业为 employee 的人进行代理。

```
CREATE SELECTDEPUTYCLASS employee
ATTRIBUTE
(salary INTEGER, company CHAR())
AS
SELECT name, sex, person_age (birth) AS age
FROM person WHERE occupation = 'employee';
```

其中的 salary 和 company 属性为扩展的实属性，表示雇员的工资和所属公司的名字。由代理规则可以提取出三个虚属性：name，sex 和 age。

例3：在 employee 上建立一个 group 类型的代理类 avg_salary，计算各个公司的平均工资：

```
CREATE GROUPDEPUTYCLASS avg_salary
AS
SELECT company, avg (salary) as avg_salary
FROM employee WHERE company 〈〉 NULL
GROUP BY company;
```

该代理类中的虚属性 avg_salary 表示了各个公司的平均工资。当 employee 中的相关属性值被修改后，平均工资的值也会自动地修改。

例4：建立一个 join 型代理类，表示公司的老板。

```
CREATE JOINDEPUTYCLASS boss
AS
SELECT name, company FROM employee, company WEHER employee. name = company. boss;
```

该类中的每个对象都有两个源对象，分别来自 employee 和 company 两个源类。

例5：建立一个 union 型代理类，表示所有的人和公司的地址。

```
CREATE UNIONDEPUTYCLASS address
AS
(SELECT name, address FROM person )UNION
(SELECT name, address FROM company);
```

该类也有两个源类，但是其中的某个代理对象只有一个源对象，来自 person 或者 company。

3.3 对象代理操作语言

对象代理操作语言支持新建、删除、修改和查询对象或代理对象的操作。对象代理数据库中对象的形式和关系数据库中元组的形式十分类似[4]，所以新建、删除和修改对象的命令可以沿用关系数据库中插入、删除和更新元组的命令。

(1)新建对象

对象代理模型目前只允许向基本类中插入对象。在某个基本类中新建对象后，系统会自动的按照新对象的属性值和代理规则生成代理对象，以及这些代理对象的代理对象。这些自动生成的对象中的扩展属性的初值为空值或者缺省值。例如，新建一个 person 对象：

```
INSERT INTO person VALUES
('zhai', 'm', 1980, 'WuHan', 'employee');
```

该对象符合了 employee 类的代理规则，因此会自动生成该类的代理对象：

```
(NULL, NULL, 'zhai', 'm', 24)
```

其中前两个扩展属性的值为空，而虚属性 age 的值则是根据定义的切换操作计算得到的。类似地，在 address 类中也会产生一个代理对象。

(2)更新对象

更新对象的操作可以任何层次的类上进行。如果是实属性的更新则可以在对象上直接进行。而对于虚属性，则需要根据对象指针回溯到其源对象上进行更新。在更新之后要重新检查代理规则，如果新

(下转第616页)

架构。然后提出一个基于兴趣查询的主动聚类策略，并且在此基础上给出一个适合于聚类后网络拓扑的自适应的路由策略以提高相似性查询的效率。最后，文本用一个基本的实验证明了所给出的策略是有效的。关于对等计算的研究目前仍十分热烈，仍有许多问题等待解决。对于节点聚类过程的研究目前还并不多，在以后的研究工作中，我们会花更多精力在这方面。

参 考 文 献

1 Crespo A. Garcia-Molina HRouting indices for peer-to-peer systems. In ICDCS, 2002

2 Gnutella Development Home Page

3 MSN Home Page

4 Napster Home Page

5 Ng W S, Ooi B C, Tan K L. Bestpeer: A self-con. gurable peer-to-peer system. In: Proc. of the 18th Intl. Conf. on Data Engineering, San Jose, CA, April 2002 (Poster Paper)

6 Ng W S, Ooi B C, Tan K L, Zhou A. Peerdb: A p2p-based system for distributed data sharing. In: Proc. of the 19th Intl. Conf. on Data Engineering, 2003

7 Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proc. of the ACM Special Interest Group on Data Communication (SIGCOMM), 2001

8 SETI@home Home Page

9 Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. In: Proc. of the ACM Special Interest Group on Data Communication (SIGCOMM), 2001

10 Zhao B Y, Kubiatowicz J D, Joseph A D. Tapestry: An infrastructure for fault-resilient widearea location and routing

(上接第571页)

对象不再符合原来的代理规则，就要删除旧的代理对象；如果新对象符合了新的代理，就要生成新的代理对象。这个过程称为对象的更新迁移。

例如，我们对上例中生成的 employee 对象的实属性进行更新，输入工资和公司名：

UPDATE employee SET salary = 2000, company = 'Microsoft' WHERE name = 'zhai';

这时更下层的 avg_salary 中 Microsoft 公司的平均工资也会自动地修改。如果对虚属性 name 进行更新：

```
UPDATE employee SET name = 'zbx'
WHERE name = 'zhai';
```

那么在相关的 person 类和 boss 类以及 address 类中的 name 属性都会相应地被修改。

(3)删除对象

删除对象目前也只能在基本类上进行。如果删除掉上面定义的 person 类对象：

DELETE FROM person WHERE name = 'zhai';

则对应的 employee 和 address 代理对象也会自动被删除。但是在 avg_salary 中的代理对象不会被删除，而是被修改。

(4)查询对象

对象代理数据库中，当查询涉及到代理对象中的虚属性时，系统要根据定义的切换操作取得其对应的实属性值，加以计算，将结果返回给用户。而这一切对用户是透明的。因此，对代理类中的虚属性的查询在形式上和实属性的查询一样。

对象代理模型使用一种特殊的路径表达式来实现跨类的属性查询。对象代理数据库中的对象之间由双向指针相互联系，使得用户能够由一个对象直接得到相关的另外一个对象，从而形成一个对象指针路径。路径表达式就是表示这种路径的语法成分，它可以作为各种表达式的变量。形式上一个路径表达式是这样的：

〈类名〉{→〈类名〉}.〈属性名〉

例如查询 boss 类中某个对象的性别，则使用如下查询：

```
SELECT boss→employee.sex
WHERE boss.company = 'Microsoft';
```

其中斜体部分就是路径表达式，它的涵义是取 boss 对象对应的 employee 类源对象中的 sex 属性值。

结论 我们已经在 PostgreSQL 系统[5,6]上扩展了对象代理数据库定义语言，上面所介绍的命令都已经得到了实现。但是这些功能还不足以完全达到对象代理模型的要求。在完整的对象代理模型中，代理类上不仅可以进行查询，还可以进行插入删除等操作。这些操作需要写方式的切换操作。另外对于方法的继承，实际上也可以加入切换操作。目前我们正在研究这些切换操作的实现方法。

参 考 文 献

1 Peng Z. An Object Deputy Model for Advanced Database Applications. DASFAA, 1995. 1～15

2 Peng Z, Kambayashi Y. Deputy Mechanisms for Object-Oriented Databases. In: Proc. of the Eleventh Intl. Conf. on Data Engineering, March 1995

3 Sybase E, Sandy J M. SQL: 1999, formly known as SQL3. In: Proc. of the 1999 ACM SIGMOD Conf. ACM, 1999. 131～138

4 Stonebraker M. Object management in POSTGRES using procedures. In: Proc. on Object-oriented database systems, 1986

5 Stonebraker M, Rowe L A. The design of POSTGRES. In: Proc. of the 1986 ACM SIGMOD intl. conf. on Management of data, June 1986

6 Stonebraker M, Kemnitz G. The POSTGRES next generation database management system. Communications of the ACM, Oct. 1991

7 Garcia-Molina H, Ullman J D, Widom J. Database System Implementation. China Machine Press, 2002