


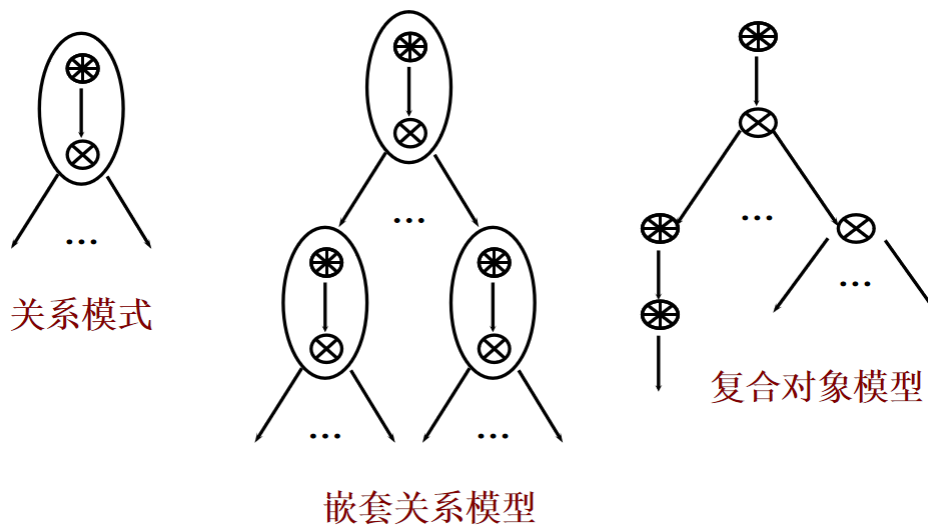
第10章 对象关系数据库

10.1 对象联系图

10.1.1 DB技术发展过程

1. 平面关系模型：属性不可分
2. 嵌套关系模型：属性：关系/表（可多次嵌套），基本类型
3. 复合对象模型：属性：元组 / 结构，关系 / 集合，基本类型
4. 面向对象模型：继承，属性：元组，关系，基本类型

集合用  表示，元组用  表示。



5. 定义方法

// 定义 Dept(dno,dname,staff(empno,ename,age)) 的嵌套关系

方法1：类型

```
type DeptRel = relation(dno:integer, dname:string,  
    staff:EmpRel); # 定义车间类型  
type EmpRel = relation(empno:integer, ename:string,  
    age:integer); #定义职员类型  
persistent var Dept:DeptRel #持久变量形式说明
```

方法2：先元组，再类型

```

type DeptTup = tuple(dno:integer, dname:string,
staff:EmpRel); # 定义车间元组
type DeptRel = set(DeptTup); # 定义车间类型（元组的集合）
type EmpTup = tuple(empno:integer, ename:string,
age:integer); # 定义职员元组
type EmpRel = set(EmpTup);
persistent var Dept:DeptRel

```

方法3: 元组 + set

```

type DeptTup = tuple(dno:integer, dname:string,
staff:set(EmpTup));
type EmpTup = tuple(empno:integer, ename:string,
age:integer);
persistent var Dept:set(DeptTup);

```

10.1.2 引用类型

嵌套关系和符合对象无法表达递归结构；利用指针，指向关系。

10.1.3 对象联系图的成分

名称	符号	含义
椭圆		对象 / 实体
小圆圈	○	基本数据类型（整型、实型、字符串型）属性
椭圆间的边		“引用”
单箭头	→	单值属性
双箭头	→→	多值属性
双线箭头	⇒	子类 ⇒ 超类
双向箭头	↔	两属性间的逆联系

10.1.4 泛化 / 细化

1. 超类是子类的泛化，子类是超类的细化。
2. 超类型：较高层的对象；子类型：较低层的对象。具有继承性。
3. 泛化/细化联系：泛化边(双线箭头)；泛化边：子类 ⇒ 超类。

10.2 面向对象的类型系统

1. 基本类型：整型，浮点型，字符，字符串，布尔型，枚举型。
2. 复合类型：

类型	类型	顺序	重复性	元素个数	例子
行 / 元组 / 结构 / 对象类型	不同	有序			日期 (1,October,2007)
数组类型	相同	有序	可重复	预置	$[1,2,1] \neq [2,1,1]$
列表类型	相同	有序	可重复	未预置	$\{1,2,1\} \neq \{2,1,1\}$
包 / 多集类型	相同	无序	可重复	未预置	$\{1,2,1\} = \{2,1,1\}$
集合 / 关系类型	相同	无序	不可重复	未预置	$\{1,2\} = \{2,1\}$

后四行统称“聚集类型”；数据类型可以嵌套。

3. 引用类型：数据类型的定义只能嵌套，若要允许递归，就要前面提到的引用类型。

10.3 ORDB的定义语言

10.3.1 ORDB

1. 对象关系数据模型：在传统的关系数据模型基础上，提供元组、数组、集合一类丰富的数据类型以及处理新的数据类型操作的能力，并且有继承性和对象标识等面向对象特点。
1. 对象关系数据库系统：基于对象关系数据模型的DBS。

10.3.2 数据类型

```

CREATE TYPE DATE(day integer, month char(10), year integer);    /*
定义结构类型*/
CREATE TYPE MyString char varying;                               /*定义变长字符串类型*/
CREATE TYPE NameArray MyString[10];                             /*定义数组类型*/
CREATE TYPE StudentGrade multiset(integer);                     /*定义包类型*/
CREATE TYPE CourseList setof(MyString);                         /*定义集合类型*/

/*使用中间变量，建立表类型*/
CREATE TYPE CourseGrade (course MyString, grade integer)
CREATE TYPE StudentCourse (name MyString, cg setof(CourseGrade));
/*定义嵌套类型*/
CREATE TABLE sc of Type StudentCourse;

/*不使用中间变量，建立表类型*/
CREATE TABLE sc(name MyString, cg setof(course MyString, grade
integer));

```

10.3.3 继承性

1. 类型级

```

/*超类型*/
CREATE TYPE Person(name MyString, social_number integer);
/*子类型*/
CREATE TYPE Student(degree MyString, department MyString)
        under Person;
CREATE TYPE Teacher(salary integer, department MyString)
        under Person;

```

2. 表级

```

/*超表*/
CREATE TABLE People(name MyString, social_number integer);
/*子表*/
CREATE TABLE Students(degree MyString, department MyString)
        under People;
CREATE TABLE Teachers(salary integer, department MyString)
        under People;

```

3. 一致性要求

- a. 超表 中每个元组最多可以与每个子表中的一个元组对应。
- b. 子表 中每个元组在超表中恰有一个元组对应，并在继承的属性上有相同的值。

10.3.4 引用类型

利用ref(表名)表示引用类型，即地址。实现递归。若集合类型，需加setof()

```
CREATE TYPE MyString char varying;  
CREATE TABLE ddept(dno integer, dname MyString, staff  
setof(ref(emp)));  
CREATE TABLE emp(empno integer, ename MyString, age integer,  
works_for setof(ref(dept)));
```

10.4 ORDB的查询语言

10.4.1 SELECT语句

1. 扩充SQL对SELECT语句的使用规定

- a. 允许用于计算关系的表达式出现在任何关系名可以出现的地方。
- b. 每个基本表设置一个元组变量，然后才可引用，在**FROM**子句中要为每个关系定义一个元组变量。

e.g. FROM university as U

- c. 当属性值为单值或结构值时，属性的引用方式仍和传统的关系 模型一样，在层次之间加点“.”。

U.president.fname ✓ university 的 **president** 属性，只有一个 **faculty** 元组与之对应。

- d. 当路径中某个属性值为集合时，就不能连着写下去，必须定义元组变量。

U.staff.frame × university 的 **staff** 属性，有多个 **faculty** 元组与之对应，不能直接获取属性值。

10.4.2 嵌套与解除嵌套

“接触嵌套”：将一个嵌套关系转换成1NF的过程。

例： 检索使用本校教材开课的教师工号、姓名及所在学校：

```
SELECT  A.uname, B.fno, B.fname
FROM    university as A, A.staff as B, B.teach as C
WHERE   C.editor.uname = A.uname;
```

uname	fno	fname
Fudan University	1357	ZHAO
Fudan University	2468	LIU
Jiaotong University	4567	WEN
Jiaotong University	5246	BAO
Jiaotong University	3719	WU

图 10.9 1NF 关系

“嵌套”：将一个1NF关系转化为嵌套关系。

```
/* 利用set 和 group by 实现嵌套*/
SELECT  A.uname, set (B.fno, B.fname) as teachers
FROM    university as A, A.staff as B, B.teach as C
WHERE   C.editor.uname = A.uname
GROUP BY A.uname;
```

uname	teachers
	(fno, fname)
Fudan University	{ (1357, ZHAO), (2468, LIU) }
Jiaotong University	{ (4567, WEN), (5246, BAO), (3719, WU) }

图 10.10 非 1NF 关系（嵌套关系）

*10.4.3 函数的定义和使用

```
/*定义类型 和 表*/
CREATE TYPE StudentCourseGrade(name MyString, cg setof(course
MyString, grade integer));
CREATE TABLE sc of TYPE StudentCourseGrade

/*定义函数*/
CREATE FUNCTION course_count(one_student StudentCourseGrade) /*参数
类型:Student CourseGrade*/
RETURNS integer AS /*返回值*/
SELECT count(B.cg)
FROM one_student as B
```

10.4.4 复合值的创建和查询

1. INSERT 复合值

```
INSERT INTO sc VALUES ('ZHANG', set(('DB',80),('OS',85)));
```

2. SELECT 复合值

```
/*检索 WANG LIU ZHANG三位学生选修的课程门数 */  
SELECT A.name, count(A.cg)  
FROM sc as A  
WHERE A.name IN set('WANG','LIU','ZHANG')  
GROUP BY A.name
```