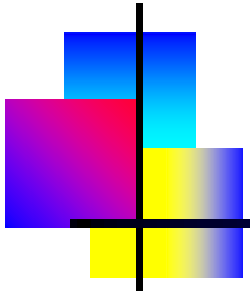
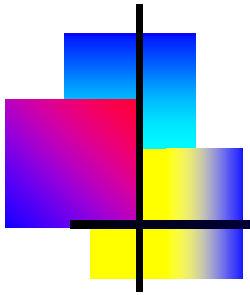


数据库实用教程（第三版）

第十章　对象关系数据库

清华大学出版社
董健全
2024年12月30日



第十三章　对象关系数据库

本章概念：

新一代**DBS**的两条途径：**ORDBS**和**OODBS**；

平面关系模型，嵌套关系模型，复合对象模型，引用类型；

对象联系图的成分及表示方法，数据的泛化/细化；

ORDB的定义语言：数据类型、继承性、引用类型的定义，

ORDB的查询语言：路径表达式、嵌套与解除嵌套。

§1　数据库技术发展的三个演变过程

- ♥ 数据模型的演变
- ♥ 查询语言的演变
- ♥ 概念建模的演变

♥ 数据库技术发展的三个演变过程：

一、数据模型的演变过程

平面关系模型：　　属性都是基本数据类型。

嵌套关系模型：　　属性可以是基本数据类型，也可以是关系类型，且数据结构可以多次嵌套。

复合对象模型：　　属性可以是基本数据类型，也可以是关系类型，还可以是元组类型，且数据结构可以多次嵌套。

面向对象类型：　　在复合对象模型的基础上，数据结构的嵌套采用引用（指针）方式，并且引入面向对象技术的继承性等概念。

返回

♥数据库技术发展的三个演变过程：

二、查询语言的演变

主要指SELECT语句的演变过程：

RDB中的SELECT语句：由六个子句构成。

ORDB中的SELECT语句（SQL3标准）：引入了路径表达式、嵌套与解除嵌套等概念。

OODB中的SELECT语句（ODMG标准OQL）：有了更多的扩充，并与宿主语言语句混合起来，可以表达更为复杂的查询操作。

返回

♥数据库技术发展的三个演变过程：

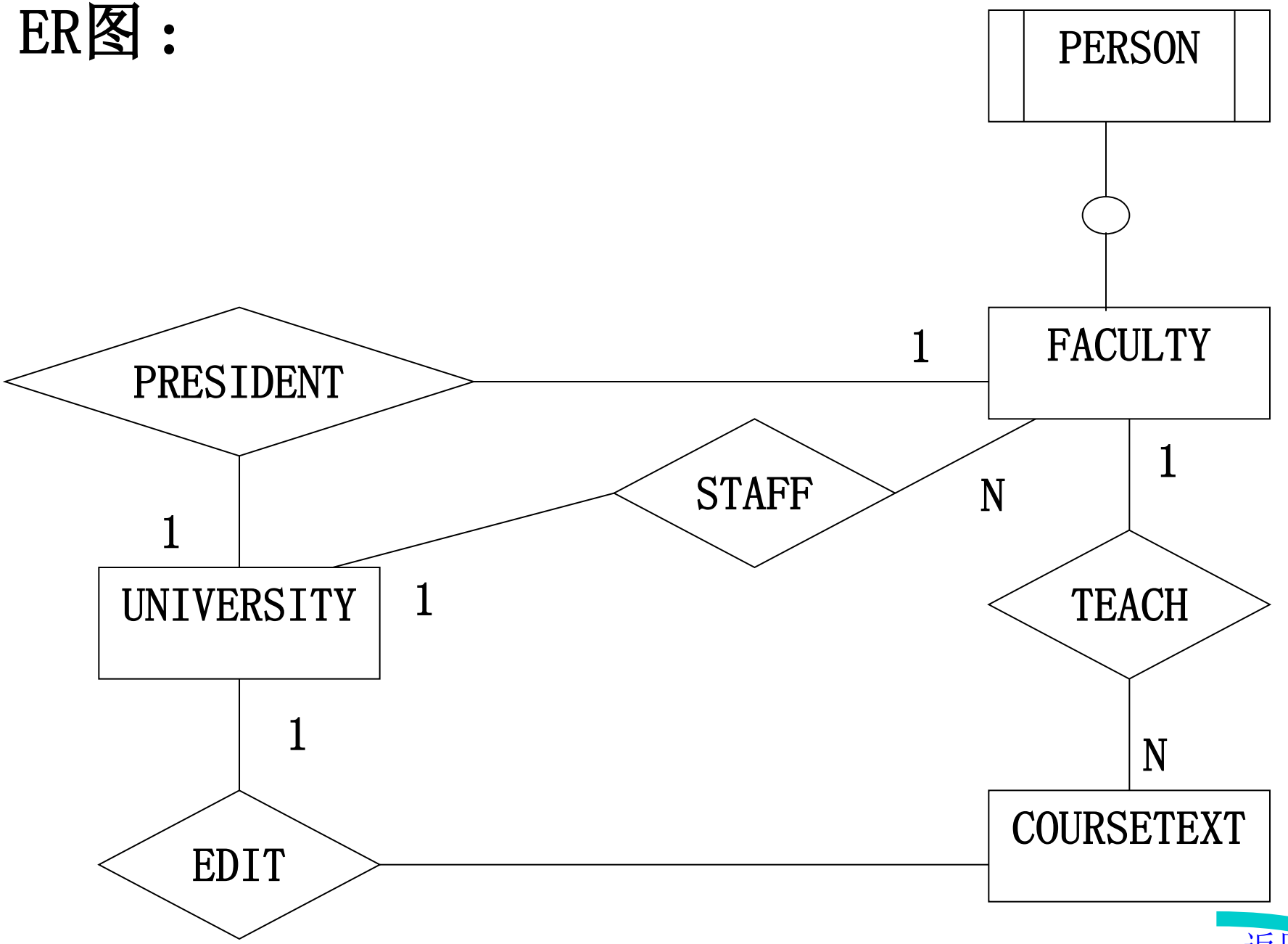
三、概念建模的演变

ER图：　主要用于关系数据库的设计。

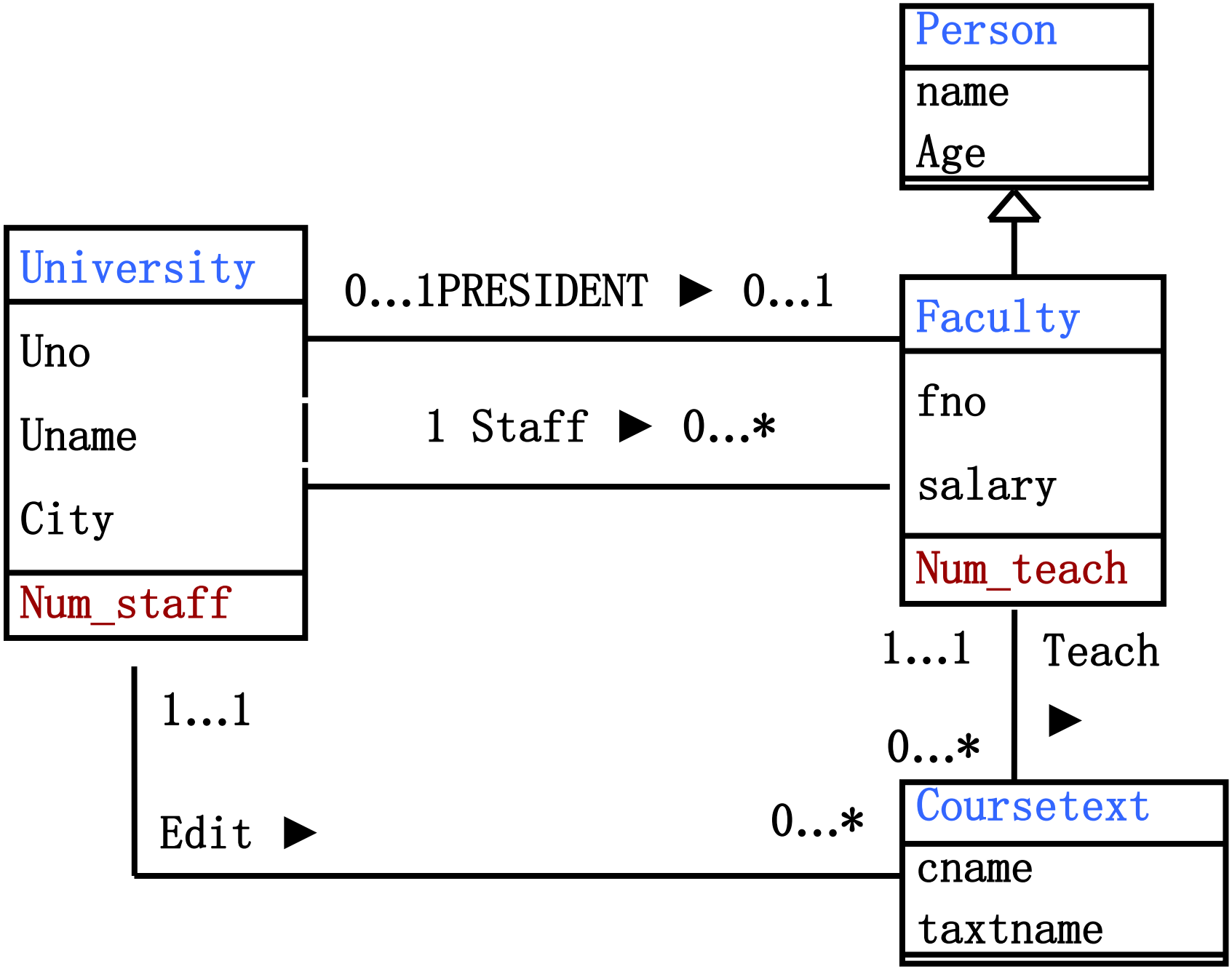
对象联系图：它是ER图的扩充，使之能表达对象之间的引用。

UML的类图：是一种纯OO技术的结构，体现了现实世界数据之间面向对象的各种联系方式。

ER图：


PERSON
FACULTY
PRESIDENT
1
STAFF
N
1
1
UNIVERSITY
1
TEACH
1
N
EDIT
COURSETEXT

返回

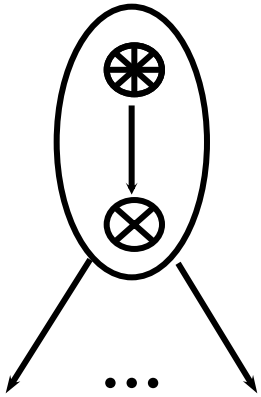

Person
name
Age
University
Uno
Uname
City
Num_staff
0...1PRESIDENT ▶ 0...1
1 Staff ▶ 0...*
Faculty
fno
salary
Num_teach
1...1
Teach
▶
0...*
1...1
Edit ▶
0...*
Coursetext
cname
taxtname

ER图中的术语	类图中的术语
实体集(Entity Set)	
实体(Entity)	对象(object)
联系(relationship)	
联系元数	关联元数
实体的基数(cardinality)	重复度(mulitiplicity)

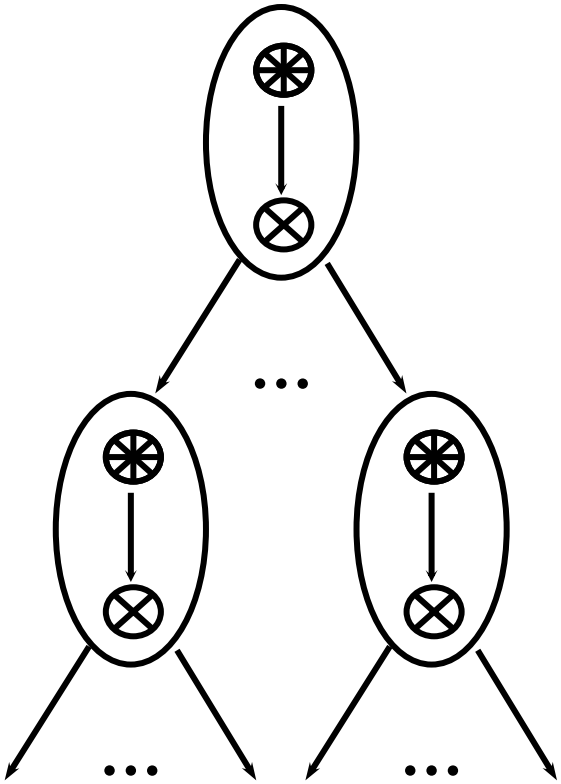
§2 对象联系图

一、三种模型的表示

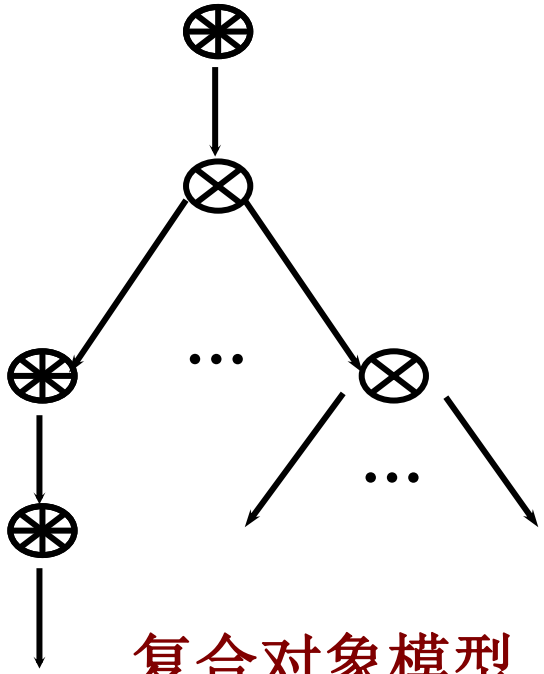
集合用 ⊛ 表示，元组用 ⊗ 表示。



关系模式



嵌套关系模型



复合对象模型

例:　在教育系统中,大学(University)与

教师(Faculty) 组成了嵌套关系：

University（ uno, uname, city, Staff (fno，fname，age))

属性Staff是一个关系类型，表示一所大学中的所有

教师。

嵌套的关系结构描述（先定义关系类型，再定义关系）：

```
type   UniversityRel = relation（uno：  integer，
                                 uname ：string，
                                 city ：  string，
                                 staff：  FacultyRel）；
type  FacultyRel = realtion（fno：integer，
                             fname：string，
                             age：integer）；
persistent  var University ：  UniversityRel；
```

这里嵌套关系用持久变量(persistent variant)形式说明,供用户使用。

也可以先定义元组类型**University**Tup与**Facult**Tup,

再定义关系类型**University**Rel与**Facult**Rel分别为**University**Tup的集合与**Facult**Tup的集合：

```
type  UniversityTup = tuple (uno: integer,
                             uname: string,
                             staff:  FacultRel);
type  FacultTup = tuple  (fno: integer,
                          fname: string,
                          age: integer) ;
type  UniversityRel = set ( UniversityTup ) ;
type  FacultRel = set ( FacultTup ) ;
persistent  var  University:  UniversityRel ;
```

也可以不定义关系类型，直接使用集合set形式：

```
type UniversityTup = tuple (uno: integer,
                            uname: string,
                            staff： set(FacultTup))
type  FacultTup = tuple (fno: integer,
                         fname: string,
                         age: integer) ;
persistent  var  University:  set(UniversityTup);
```

如果大学中还需要校长（president）信息,那么可设计成如下关系：

University (uno,uname, staff (fno,fname,age),

president [fno,fname,age])

其中**Staff**是关系类型， **president**是元组类型；

[ ]表示元组类型。

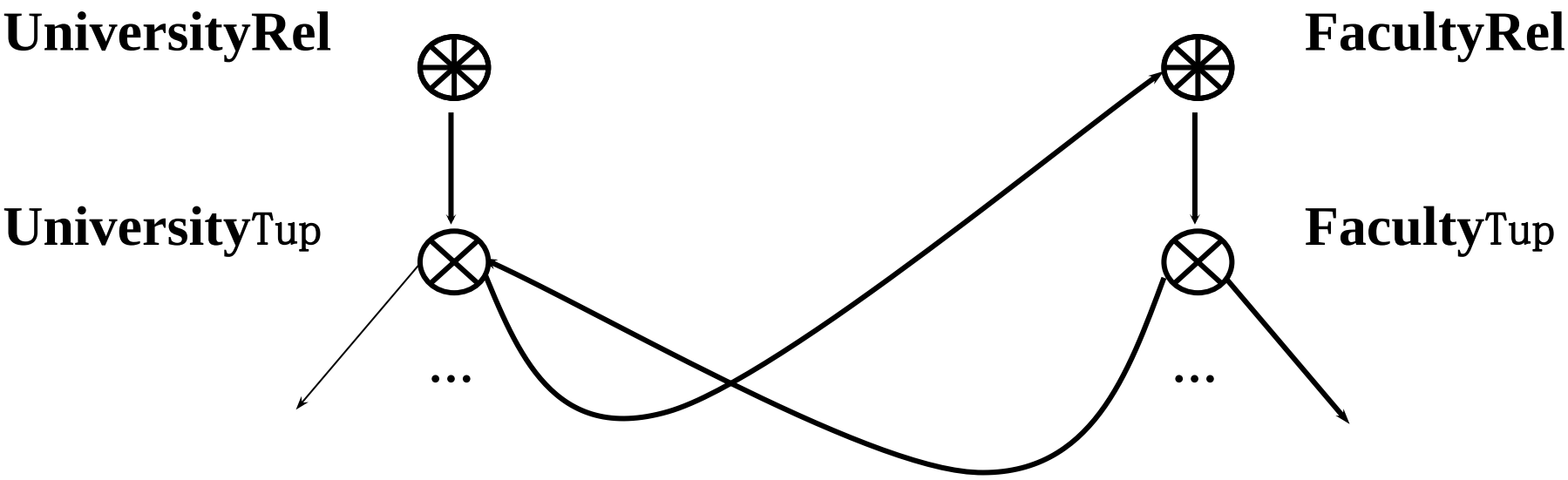
University关系是一个复合关系，不是嵌套关系，

二、引用类型

嵌套关系和复合对象的明显弱点是:

无法表达递归的结构，类型定义不允许递归。

禁用的类型构造 ：

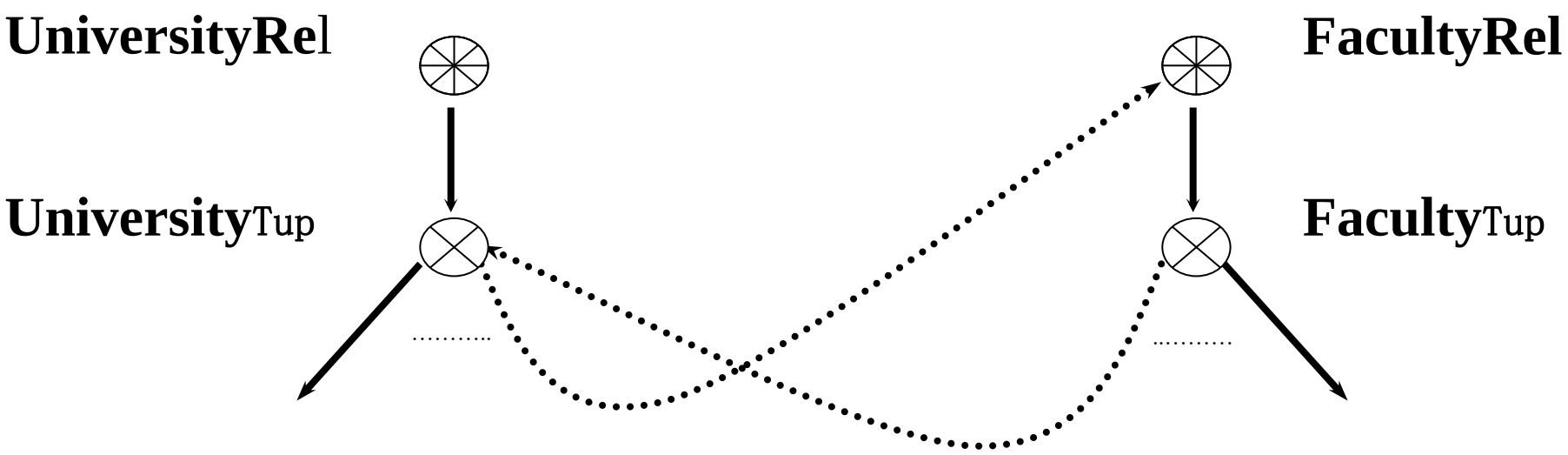


采用“引用” 技术解决类型定义中的递归问题。

在属性的类型中，除了基本数据类型、元组类型、关系类型外，还可以出现“引用类型”。**引用类型相当于程序设计中指针的概念，**

在面向对象技术中称为“对象标识”。

引入“引用”概念的类型构造：



三、对象联系图的成分

椭圆：　　表示对象类型（相当于实体类型）；

小圆圈：　　表示基本数据类型（整型、实型、字符串型）属性；

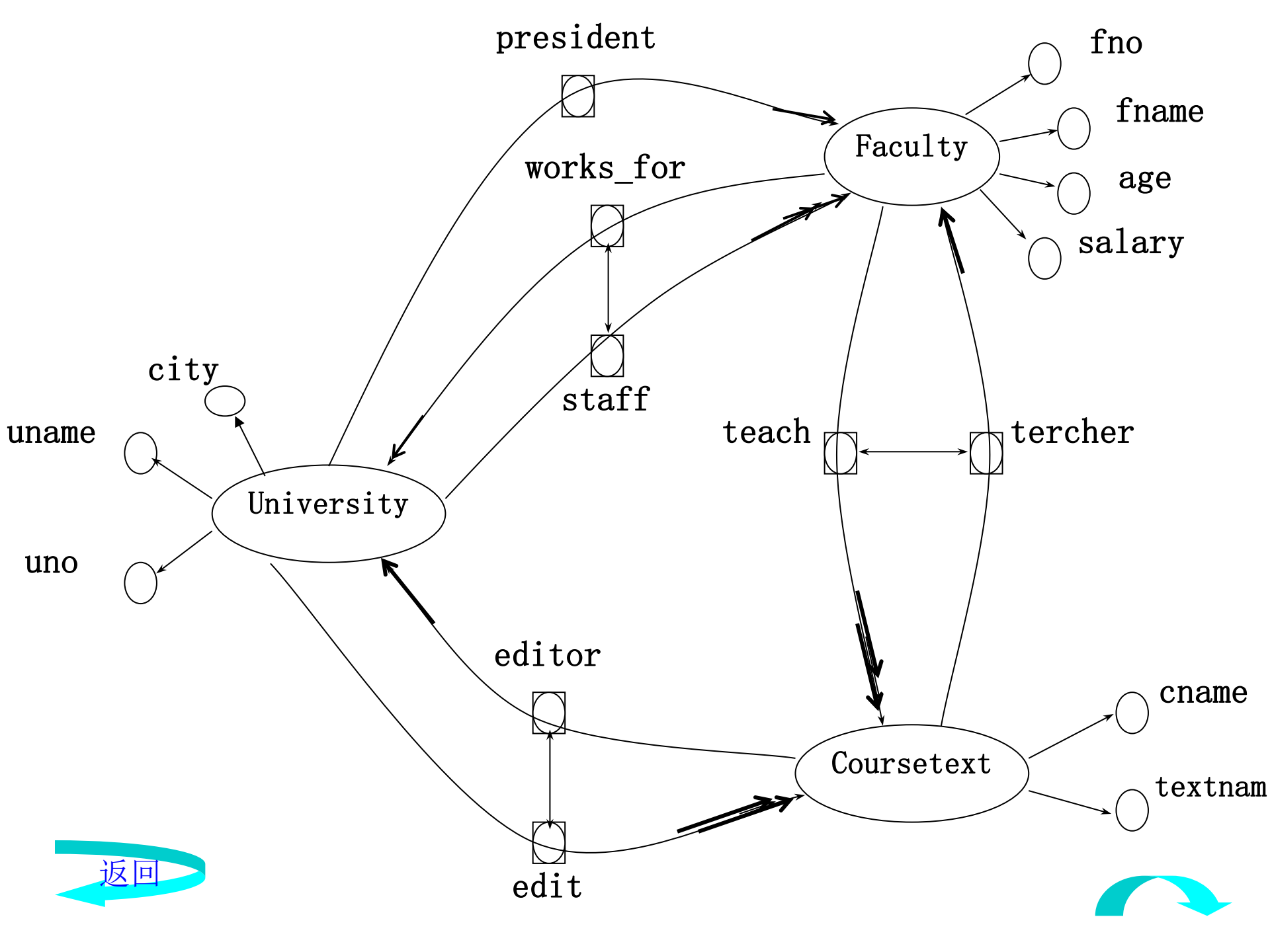
椭圆之间的边：　　表示对象之间的“引用”；

单箭头（→）：　　表示属性值是单值（属性可以是基本数据类型，也可以是另一个对象类型）；

双箭头（→→）：　　表示属性值是多值（属性可以是基本数据类型，也可以是另一个对象类型）；

双线箭头（⇒）：　　表示对象类型之间的超类与子类联系（从子类指向超类）；

双向箭头（←→）：　　表示两个属性之间值的联系为逆联系。


president
fno
fname
age
salary
Faculty
works_for
staff
city
uname
University
uno
teach
tercher
editor
cname
Coursetext
textnam
edit
返回

四、数据的泛化 / 细化

数据的泛化 / 细化：对概念之间联系进行抽象的一种方法。

“泛化”：称较高层上抽象是较低层上抽象的“泛化”。（在较低层

上抽象表达与之联系的较高层上的抽象， ）

“细化”：称较低层上抽象是较高层上抽象的“细化”。这种细化联

系是一种“是”（is a）的联系。

在有泛化 / 细化联系的对象类型之间：

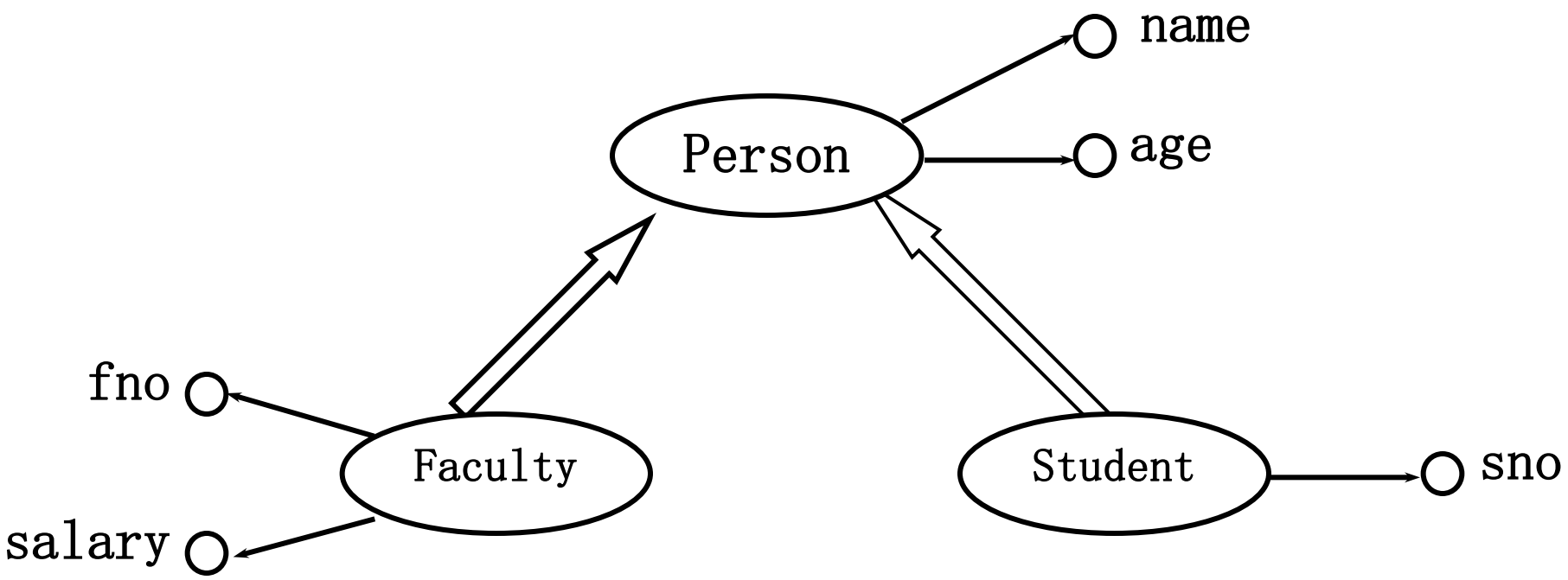
较高层的对象类型称为“超类型”（supertype），

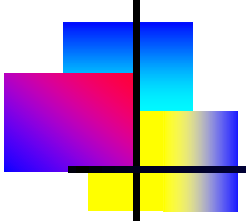
较低层的对象类型称为“子类型”（subtype）。

子类具有继承性，继承超类的特征，而子类本身又有其它的特征。

泛化/细化联系用泛化边(双线箭头)表示，泛化边从子类指向超类。

泛化/细化联系用泛化边(双线箭头)表示,泛化边从子类指向超类。





§3　面向对象的数据类型

一、基本数据类型:整型、浮点型、字符型、字符串、布尔型和枚举型；

二、复合类型：复合类型有下列五种：

1. 结构（或行）类型：不同类型元素的有序集合称为结构。

2. 数组类型：同类元素的有序集合，称为数组（array）。

3. 包(多集)类型：同类元素的无序集合并且允许有重复的元素。

4. 集合类型：相同类型元素的无序集合，并且所有的元素必须是不同的（set）。

5. 列表类型：类型相同并且允许有重复的元素的有序集合。

复合类型中： 数组、列表、包、集合

———— 统称为聚集类型。

聚集类型的差异

类型	元素	元素的重复性	元素个数	例子
数组	有序	允许一个元素出现多次	预置	[1，2，1]和[2，1，1]是不同的数组
列表	有序	允许一个元素出现多次	未预置	{1，2，1}和{2，1，1}是不同的列表
包（多集）	无序	允许一个元素出现多次	未预置	{1，2，1}和{2，1，1}是相同的包
集合（关系）	无序	每个元素只能出现一次	未预置	{1，2}和{2，1}是相同的集合

数据类型可以嵌套。

例：　课程成绩集

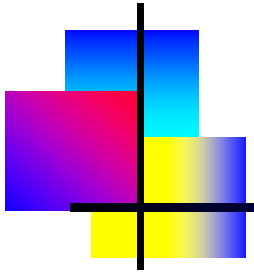
{(MATHS,80),(PHYSICS,90),(PL,70),(OS,80),(DB,80)}

外层是集合类型，里层是结构类型。

三、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



§4 ORDB的定义语言

一、ORDB的定义:

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。

二、ORDB数据类型的定义

在对象关系模型中，属性可以是基本数据类型或复合类型。

复合类型有下列四种：

（1）结构类型：不同类型元素的有序集合称为结构。

（2）数组类型：同类元素的有序集合，称为数组（array）。

（3）多集类型：同类元素的无序集合（有的成员可多次出现），称为多集（multiset）。

（4）集合类型：同类元素的无序集合（每个成员只能出现一次），称为集合（set）。

在ORDB中，数据类型可以嵌套。

例如课程成绩集：

{（MATHS，80），（PHYSICS，90），（PL，70），（OS，80），（DB，0）},

外层是集合类型，里层是结构类型。

例：设学生选课和成绩的嵌套关系如下：

SC(name,cg(course,grade,date))

其属性表示：学生姓名、课程名、成绩、日期等含义。

```
CREATE TYPE MyString char varying; /*定义MyString是变长字符串类型*/
CREATE TYPE MyDate (day  integer,
                    month char(10),
                    year  integer); /*定义MyDate是结构类型*/
CREATE TYPE CourseGrade(course  MyString,
                        grade  integer,
                        date MyDate); /*定义CourseGrade是结构类型*/
CREATE  TYPE  StudentGrade  setof(courseGrade);
                                    /*定义StudentGrade集合类型*/
CREATE  TYPE  StudentcourseGrade (name  MyString,
                                  cg StudentGrade);
                                /*定义StudentcourseGrade是结构类型*/
```

在上述基础上再定义关系SC:

```
CREATE  TABLE  sc  of TYPE StudentcourseGrade;
    /*定义表sc包含了类型为结构StudentcourseGrade的元组*/
```

```
CREATE TYPE MyString char varying;
                                /*定义MyString是变长字符串类型*/
CREATE TYPE MyDate ( day   integer,
                     month char(10),
                     year  integer);
                                /*定义MyDate是结构类型*/
 CREATE TYPE CourseGrade ( course  MyString,
                           grade   integer,
                           date MyDate);
                               /*定义CourseGrade是结构类型*/
```

```
CREATE  TYPE  StudentGrade  setof(courseGrade);
                    /*定义StudentGrade集合类型*/
CREATE  TYPE  StudentcourseGrade (name  MyString,
                                  cg  StudentGrade);
                  /*定义StudentcourseGrade是结构类型*/
```

在上述基础上再定义关系SC:

```
CREATE TABLE  sc  of TYPE  StudentcourseGrade;
/*定义表sc包含了类型为结构StudentcourseGrade的元组*/
```

也可以不创建中间类型，直接创建所需的表：

```
CREATE  TABLE  sc（name  MyString，
                    cg  setof（course  MyString，
                                grade  integer，
                                 date  MyDate））；
```

ORDB中数据类型系统还支持数组和多集类型，例如：

```
CREATE  TYPE  NameArray  MyString[10];
                                    /*定义NameArray是数组类型*/
CREATE  TYPE  Grade  multiset(integer);
            /*定义Grade是多集类型，每个成员是整数*/
```

多集类型中的元素是无序的。如果要求元素有序,则用列表类型：

```
  CREATE  TYPE  Grade  listof(integer);
```

三、继承性的定义

继承性可以发生在类型一级或表一级。

1.类型级的继承性：

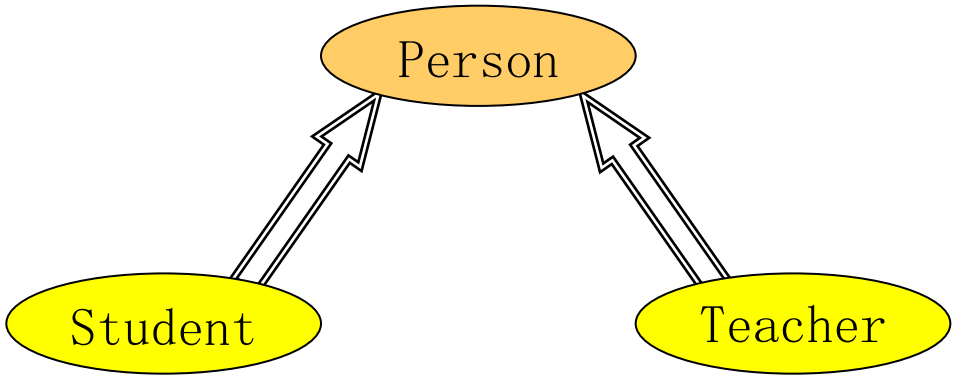
数据类型之间的子类型与超类型的联系。

例：有关人的类型定义：

```
CREATE TYPE  Person（name  MyString,
                     social_number  integer）;
```

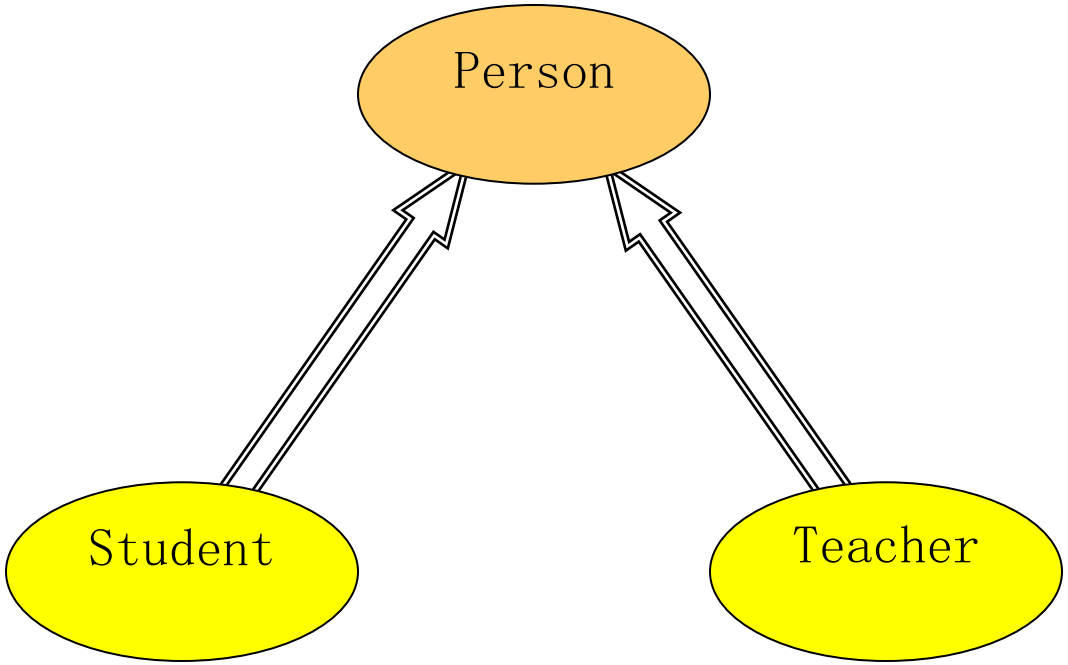
可用继承性定义学生类型和教师类型：

```
CREATE  TYPE  Student（ degree  MyString,
                        department  MyString）
                        under  Person;
CREATE  TYPE  Teacher（ salary  integer,
                        department  MyString）
                        under  Person;
```



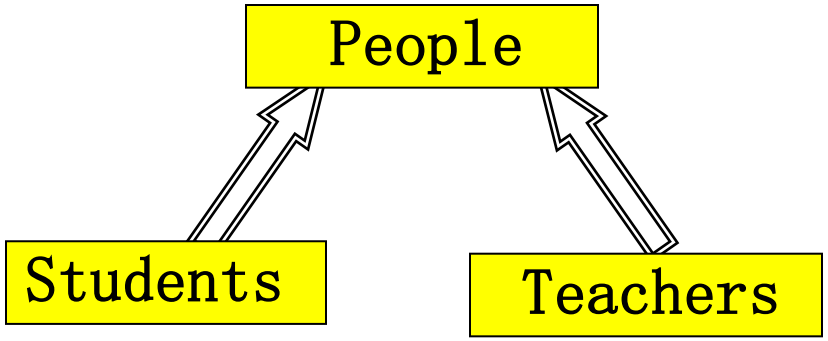
Student和Teacher两个类型都继承了Person类型的属性：name和social_number。称Student和Teacher是Person的子类型（subtype），Person是Student的超类型（supertype），也是Teacher的超类型。

类型层次图：



箭头方向为超类型，箭尾方向为子类型。

2. 表级的继承性　在表级实现继承性（子表与超表的联系）。



这里People称为超表，students和teachers称为子表。子表继承超表的全部属性。

先定义表People：CREATE TABLE People(name MyString,
social_number integer);

然后再用继承性定义表students和teachers：

```
CREATE  TABLE students (degree  MyString,
                        department  MyString)
                        under  People;
CREATE  TABLE teachers (salary  integer,
                        department  MyString)
                        under  People;
```

子表和超表应满足下列两个一致性要求：

① 超表 中每个元组最多可以与每个子表中的一个元组对应。

② 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。

四、引用类型的定义

数据类型可以嵌套定义，但要实现递归，就要使用“引用”类型。

在嵌套引用时，不是引用对象本身的值，是引用对象标识符

（即 “指针”的概念）。

引用类型有两种方式：

1.“ref”表示引用的是元组的标识符，不是元组值。

例如，大学中属性“校长”是对教师对象元组标识符的引用：

president ref（faculty）

2.“setof”表示属性值是集合类型。

例如：大学中属性“员工”是教师对象的集合：

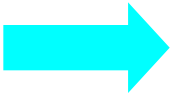
staff setof（ref（faculty））/*对faculty对象的引用的集合*/

例：大学和教师对象联系图所表示的数据库可用下列形式定义：

```
CREATE  TYPE  MyString  char  varying;
CREATE  TABLE  university ( uno  integer,
                            uname  MyString,
                            city  MyString,
                            president  ref (faculty) ,
                            staff  setof (ref (faculty) ) ,
                            edit setof (ref (coursetext) ) ) ;
CREATE  TABLE  faculty (fno  integer,
                        fname  MyString,
                        age  integer,
                        works_for  ref (university) ,
                        teach  setof (ref (coursetext) ) ) ;
CREATE  TABLE  coursetext (cname  MyString,
                           textname  MyString,
                           teacher  ref (faculty) ,
                           editor  ref (university) ) ;
```

元组的地址(即指针)

staff是集合类型，
每个成员是关系
faculty元组的标识符



例：大学和教师对象联系图所表示的数据库可用下列形式定义：

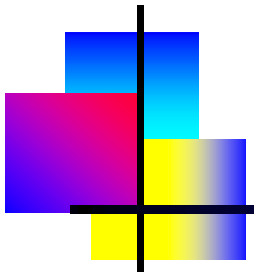
```
CREATE  TYPE  MyString  char  varying;

CREATE  TABLE  university (uno  integer,
                           uname  MyString,
                           city  MyString,
                           president  ref(faculty),
                           staff setof(ref(faculty)),
                           edit setof(ref(coursetext)));
```

staff是集合类型，每个成员是关系faculty元组的标识符

元组的地址（即指针）

```
CREATE  TABLE  faculty (fno  integer,
                        fname  MyString,
                        age  integer,
                        works_for  ref (university) ,
                        teach  setof (ref(coursetext)));
CREATE  TABLE  coursetext(cname  MyString,
                          textname  MyString,
                          teacher  ref(faculty),
                          editor  ref(university));
```



§5 ORDB的查询语言

一、扩充的SQL语言对SELECT语句的使用规定：

1. 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，比如FROM子句或SELECT子句中。这种可自由使用子表达式的能力使得充分利用嵌套关系结构成为可能。

例:检索上海地区各大学超过50岁的教师人数,可用下列语句表达:

```
SELECT  A.uname，count（SELECT  *
                          FROM  A.staff as B
                          WHERE  B.age＞50）
FROM  university  as A
WHERE  A.city = ´shanghai´;
```

2. 在ORDB中规定要为每个基本表设置一个元组变量，然后才可引用(不允许出现：表名.属性名)，在FROM子句中要为每个关系定义一个元组变量。

3. 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，在层次之间加园点“.”。

如：检索某大学的校长姓名时，可写成“U.president.fname”，

这里U是为关系university设置的元组变量。

4、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。

例：检索讲授MATHS课，采用“Mathematical Analysis”教材的教

师工号和姓名。

```
SELECT   A.fno，A.fname
FROM   faculty as A
WHERE (‘MATHS’,’Mathematical Analysis’) IN A.teach;
```

这里使用了以关系为值的属性teach，属性teach在无嵌套关系的SQL中是要求一个SELECT查询语句。

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;
```

也可表达为：

```
SELECT  B.work_for.uname，B.fno，B.fname
FROM  faculty as  B，  B.teach  as  C
 WHERE  B.works_for.uname = C.editor.uname;
```

此查询结果为一个1NF关系：

查询结果为 1NF关系:

uname	fno	fname
Qinhua University	1357	ZHAO
Qinhua University	2468	LIU
Beijin University	4567	WEN
Beijin University	5246	BAO
Beijin University	3719	WU

二、嵌套与解除嵌套

在使用SELECT语句时，我们可以要求查询结果以嵌套关系形式显示，也可以以1NF（非嵌套）形式显示。将一个嵌套关系转换成1NF的过程称为“解除嵌套”。

将一个1NF关系转化为嵌套关系称为“嵌套”。

嵌套的实现：用对SQL分组的一个扩充来完成。

在SQL分组的常规使用中，需要对每个组（逻辑上）创建一个临时的多重集合关系，然后在这个临时关系上应用一个聚集函数。

如果不应用聚集函数而只返回这个多重集合，我们就可以创建一个嵌套关系。

“检索使用本校教材开课的教师工号、姓名及所在学校”。

如果我们希望查询结果为嵌套关系，那么可在属性（fno，fname）上对关系进行嵌套，语句如下：

```
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;
```

此语句的查询结果为一个非1NF的嵌套关系。

查询结果为非1NF关系：

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

精读和习题要求

精读教材：P.216～228

习 题10： P.230 3 ～5

6、7