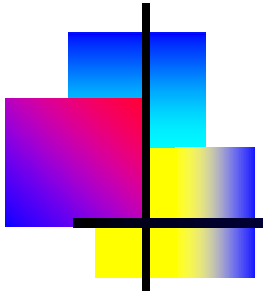
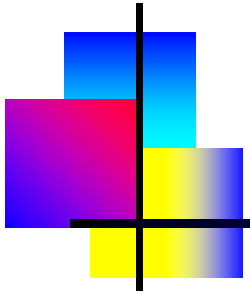


教材：数据库实用教程（第四版）

《数据库原理》课程

清华大学出版社
2024年12月30日



《数据库原理》

第十一章　面向对象数据库

清华大学出版社
董健全
2024年12月30日

第十一章　面向对象数据库

本章概念：

- ODMG标准的核心概念
- 面向对象数据模型的基本概念
- 持久化程序设计语言的基本概念
- 基于C++的面向对象数据库的对象定义语言和对象操纵语言
- OODB与ORDB的比较
- 使用UML类图来概念对象建模

§1 面向对象数据库系统概述

一、ODMG组织和标准

ODMG（Object Data Management Group）

ODMG93是用于面向对象数据管理产品接口的一个定义。

ODMG93标准有五个核心概念：

ODMG93标准有五个核心概念：

（1）对象是基本的数据结构、是存储和操作的基本单位。

（2）每个对象有一个永久的标识符。这个标识符在该对象的整个生命期中都有效，即不论该对象是存储在外存中还是内存中，该标识符都必须是有效的。

（3）对象可以被指定类型和子类型。对象被创建为一个给定的类型。对象还可以定义为其他对象的子类型。此时，它们继承父类型的所有数据特性和行为。

（4）对象状态由数据值与联系定义。

（5）对象行为由对象操作定义。

二、OODBS的概念

一个面向对象数据库系统（OODBS）应该满足两个标准：

首先它是一个数据库系统（DBS），具备DBS的基本功能，譬如查询语言、散列或成组存取方法、事务管理、一致性控制及恢复；

其次是一个面向对象系统，是针对面向对象程序设计语言的持久性对象存储管理而设计的，充分支持完整的面向对象概念和机制，譬如用户自定义数据类型、自定义函数、对象封装等必不可少的特点。

OODBS表达为：“面向对象系统 + 数据库能力”

§2 面向对象数据模型的基本概念

一、对象（object）

对象由三个部分组成：

（1）一组变量。它们包含对象的数据，

变量相当于ER模型中的属性。

（2）一组消息。这是一个对象所能响应的消息集合，每个消息可有若干参数。对象接受消息后应作出相应的响应。

（3）一组方法（Method）。每个方法是实现一个消息的代码段，一个方法返回一个值作为对消息的响应。

对象的方法可以分为只读型和更新型两种。

二、类（class）

在数据库中通常有很多相似的对象。“相似”是指它们响应相同的消息使用相同的方法、并且有相同名称和类型的变量。

将相似的对象分组形成了一个“类”（ Class)。

类是相似对象的集合。类中每个对象也称为类的实例（Instance）。

一个类中的所有对象共享一个公共的定义，尽管它们对变量所赋予的值不同。

面向对象数据模型中类的概念相当于ER模型中实体集的概念。

例： 用伪码写一个（“职员”）类的定义。定义中给出了类似的变量和类的对象所响应的消息,处理这些消息的方法在这里未给出。

```
class  employee{  /* 变量 */
                    string  name;
                    string  address;
                    data  start_date;
                    int  salary;
                  /* 消息 */
                 int  annual_salary( );
                 string  get_name( );
                 string  get_address ( ) ;
                 int  set_address (string  new_address) ;
                  int  employment_length( );
               };
```

一个类对象包括两部分内容：

（1）一个集合变量，它的值是该类的所有实例对象所组成的集合；

（2） 对消息new实施一个方法，用以创建类的一个新实例。

三、继承性

继承性允许不同类的对象共享它们公共部分的结构和特性。

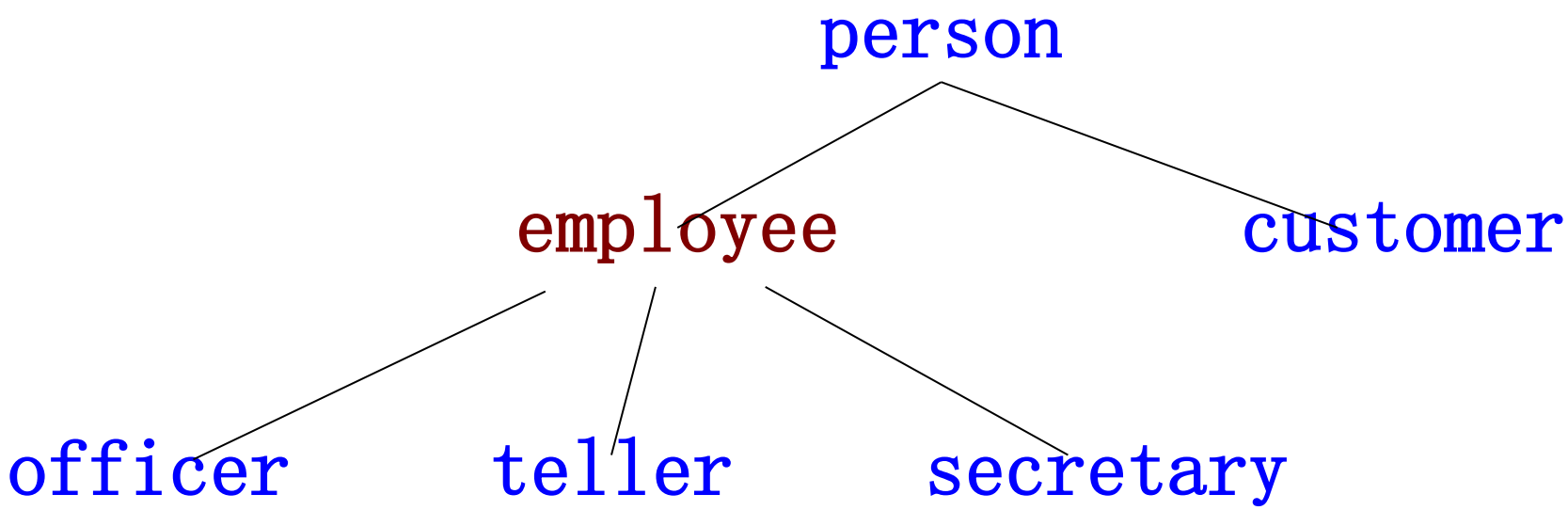
继承性可以用超类和子类的层次联系实现。

单继承性：一个子类可以继承某一个超类的结构和特性；

多重继承性：一个子类也可以继承多个超类的结构和特性。

继承性是数据间的泛化／细化联系，是一种“is a”联系，表示了类之间的相似性。

例：银行日常工作中涉及到各类人员的细化层次的类继承层次图：



图中：每个职员（**employee**）是一个人（**person**），

人是职员的泛化、抽象化，

职员是人的细化、具体化。

person是超类，**employee**是子类；等等。

伪代码定义如下：

```
class person {                                  /* 人 */
        string name;                            /* 姓名 */
        string address;                         /* 地址 */
            };
class customer isa person {                     /* 客户 */
                    int credit_rating;          /* 信用度 */
                        };
class employee isa person {                     /* 职员 */
                    date start_date;            /* 工作起始日期 */
                    int salary;                 /* 工资 */
                        };
```

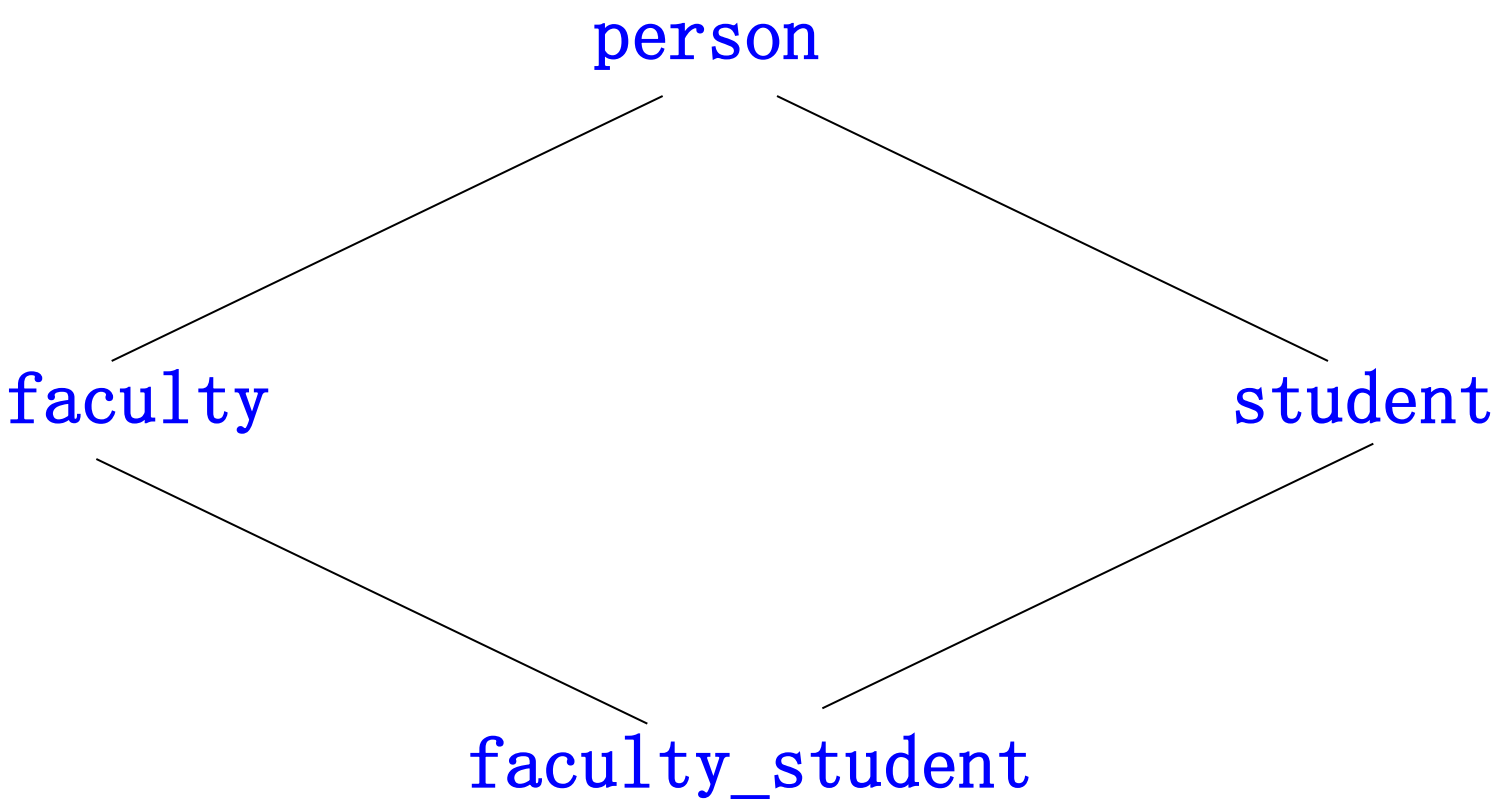
```
class officer isa employee {                    /* 高级职员 */
        int office_number;                      /* 工号 */
        int expense_account_number;             /* 经费账号 */
                };

class teller isa employee {                     /* 职员 */
        int hours_per_week;                     /* 每周工作量 */
        int station_number;                     /* 柜号 */
                };
class secretary isa employee {                  /* 秘书 */
        int hours_per_week;                     /* 每周工作量 */
        string manager;                         /* 经理姓名 */
                };
```

例：下图是人的又一个细化图：

faculty和student是person的细化子类。有的人既是教师又是学生，

∴faculty_student应是faculty和student这两个类的子类。

即：多重继承性。



四、对象标识（Object Identifier，简记为OID）

面向对象系统提供“对象标识符”的概念来标识对象。

OID与对象的物理存储位置无关，也与数据的描述方式和值无关。

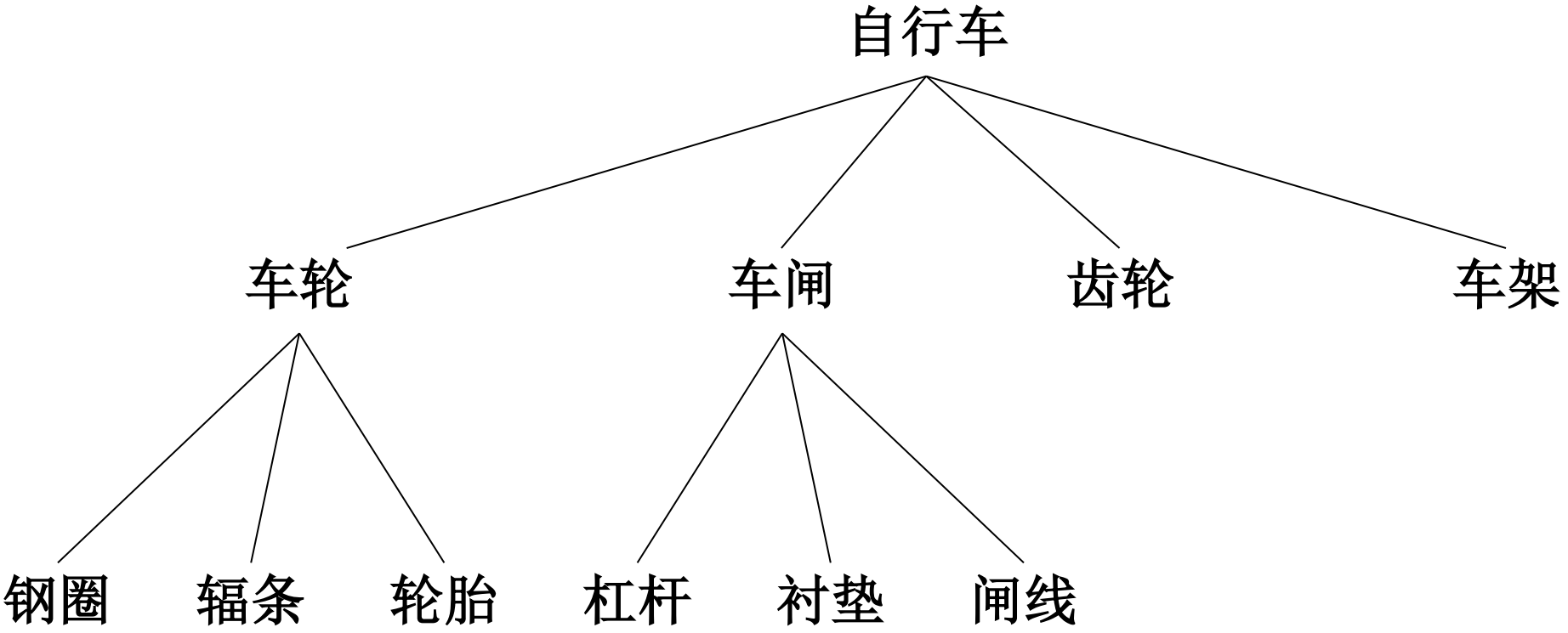
OID是惟一的，也就是说，每个对象具有单一的标识符，在对象创建的瞬间，由系统赋给对象一个OID值，它在系统内是惟一的，在对象生存期间，标识是不能改变的。

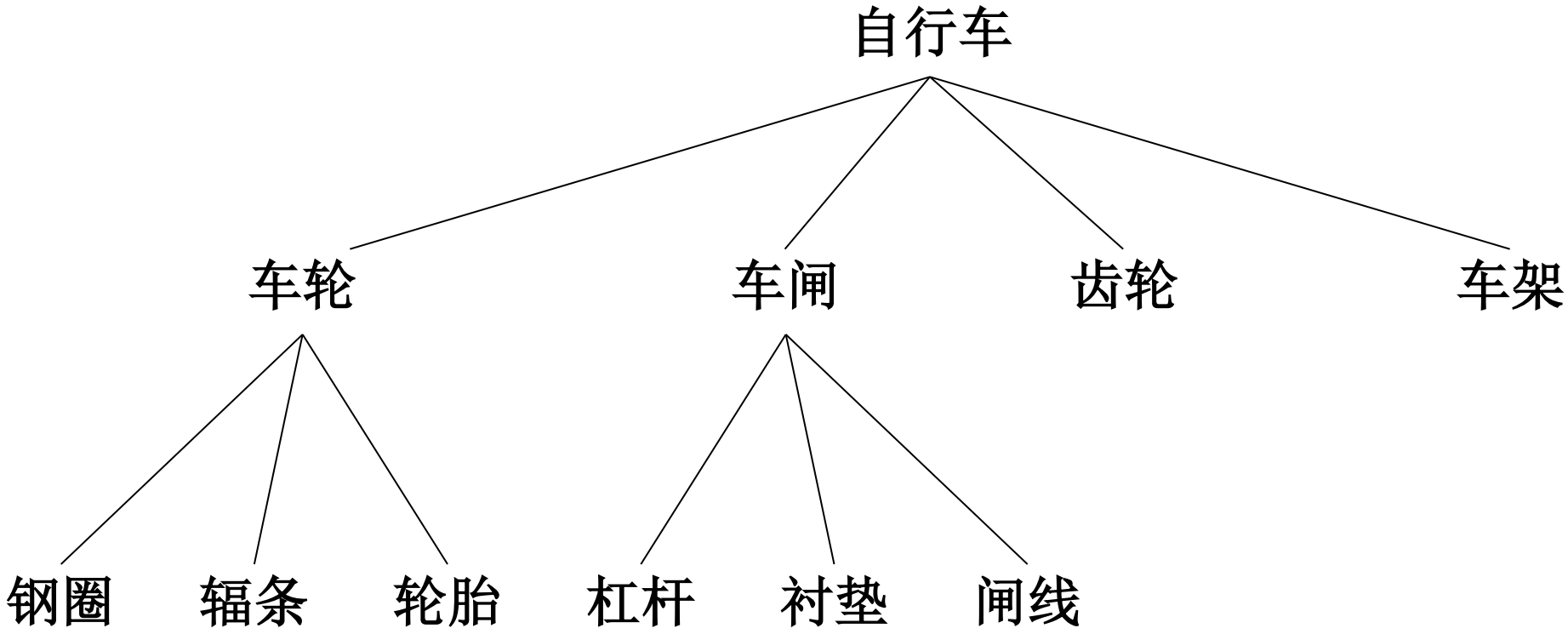
对象标识是指针一级的概念，是一个强有力的数据操纵原语，也是对集合、元组和递归等复合对象操纵的基础

五、对象包含

不同类的对象之间可能存在着包含关系（即组合关系）。

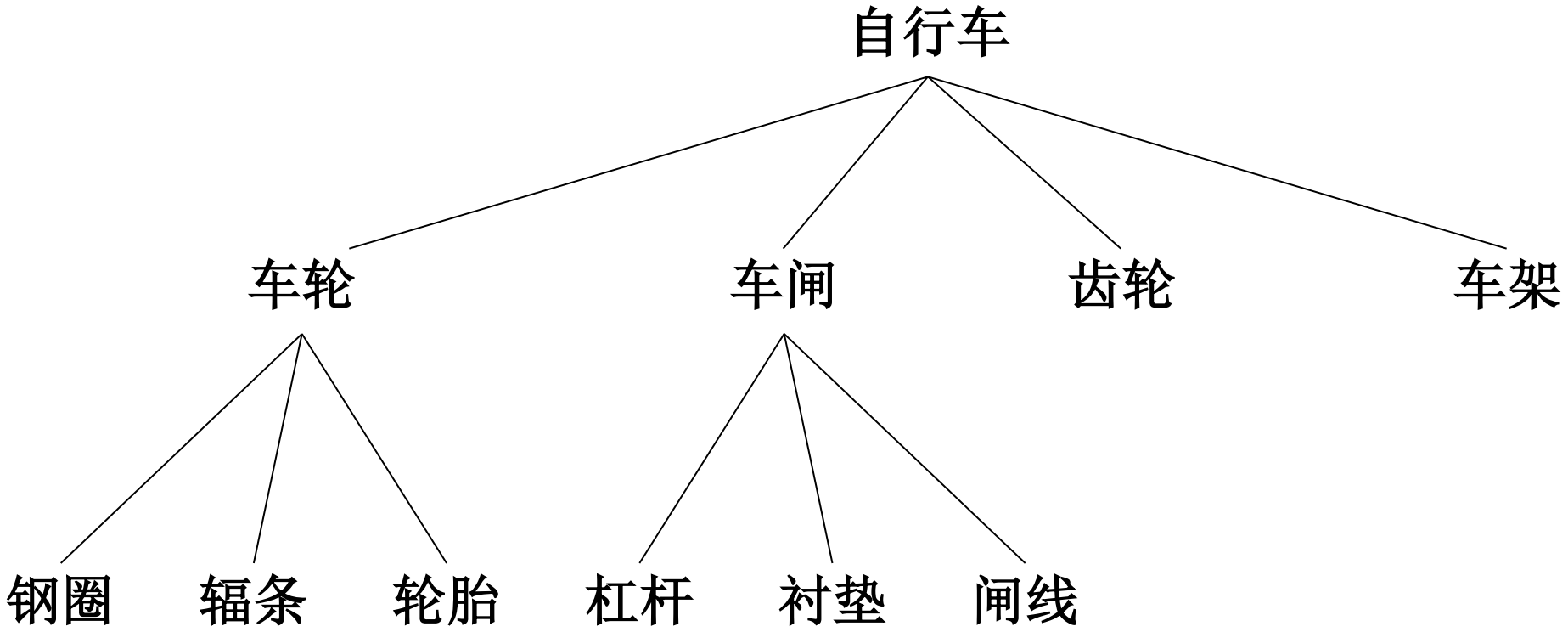
下图：自行车结构的数据库的类包含层次图





图中：自行车是车轮、车闸、齿轮、车架的组合。

车轮又包括钢圈、辐条和轮胎。结构的每一个构件都描述为一个对象，同时构件间的包含也可以用对象间的包含来描述。



包含其他对象的对象称为复合对象。包含关系可以有多层，形成类包含层次图。包含是一种“是一部分”（is part of）联系。

如：车轮是自行车的一部分，而不能说“车轮是一辆自行车”。

因此，包含与继承是两种不同的数据联系。

§3 ODMG 93和持久化C++系统

一、持久化程序设计语言

1.持久化语言与嵌入式语言的区别

（1）在嵌入式语言中，宿主语言的类型系统与SQL的类型系统不同，程序员要负责宿主语言与DML之间的类型转换。

持久化程序设计语言的查询语言与宿主语言完全集成在一块，具有相同的类型系统。创建对象并将之存储在数据库中，不需要任何显式的类型或格式改变。任何格式转换对程序员都是透明的。

（2）使用嵌入式查询语言的程序员要负责编写程序把数据从数据库中取出放到内存中。在更新时，程序员还需编写程序段将更新过的数据写回数据库。而在持久化语言中，程序员可以直接操纵持久数据，而不必为存取数据编写程序。

2. 持久化语言的三个基本概念

（1） 对象的持久性：要把OOPL变成持久化语言，第一步就是提供一种办法，把对象区分成是持久的还是暂留的。在程序运行结束后，新创建的持久对象将被保存，而暂留对象将消失。

（2）对象标识和指针：当一个持久对象被创建时，它就要被分配一个持久的对象标识符。当创建的对象为暂留时，被分配一个暂留的对象标识符，在程序终止后，对象被删掉，标识符失去意义。

（3）持久对象的存储和访问：逻辑上，实现类的方法的程序代码应该和类的类型定义一起作为数据库模式的一部分存储。但现在往往将程序代码存储在数据库之外的文件中，目的是避免对编译器和DBMS软件进行集成。

查找数据库中对象的方法有三种：

第一种方法是根据对象名找对象。实现时，每个对象有一个对象名（如同文件名一样）。这种方法对少量的对象是有效的，但对上百万个对象就不适用了。

第二种方法是依据对象标识找对象。而对象标识存储在数据库之外。

第三种方法是将对象按聚集形式存放,然后利用程序循环找所需对象。

聚集形式包括集合（**Set**）、多集（**Multiset**）等。

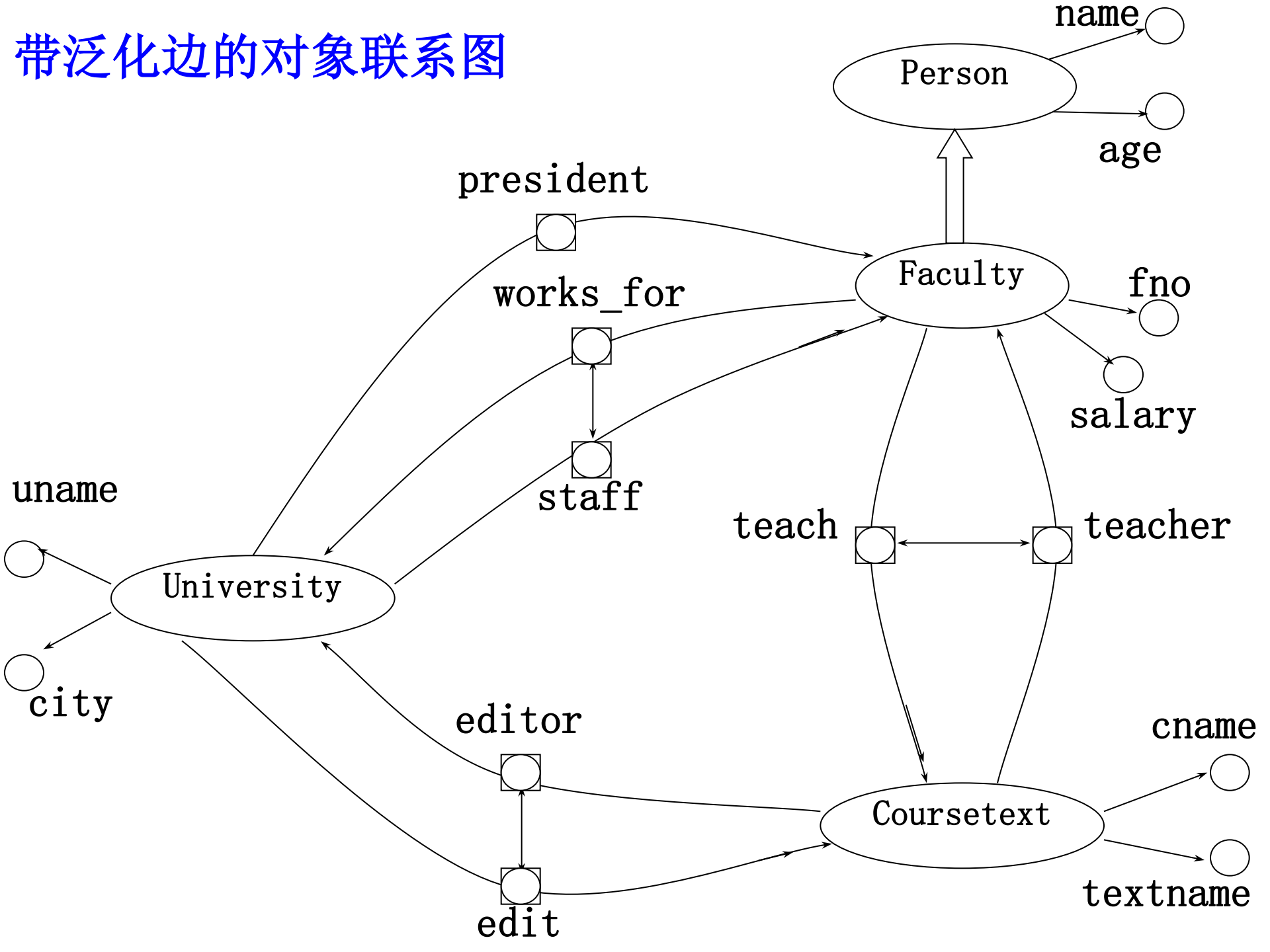
大多数**OODBS**都支持这三种访问数据库的方法。

二、ODMG C++对象定义语言

ODMG C++ 对象定义语言（C++ ODL）扩充了C++的类型定义语法。

例：在第10章的对象联系图中，定义Faculty为新的类Person的子类，那么可得到下图的带泛化边的对象联系图。

带泛化边的对象联系图


name
Person
age
president
Faculty
fno
works_for
salary
staff
uname
teach
teacher
University
city
editor
cname
Coursetext
edit
textname

该对象联系图用C++ ODL写的代码示例为：

```
class Person： public Persistent_Object {
public： string name；
        int age；
};
class Faculty： public  Person {
private： int salary；
public： int fno；
        Ref <University> works_for inverse
   University： ： staff；
        Set <Ref<Coursetext>> teach inverse
   Coursetext： ： teacher；
};
```

```
class University： public Persistent_Object {
public： string uname；
        string city；
        Ref <Faculty> president；
        Set <Ref<Faculty>> staff inverse Faculty：：
   works_for；
Set <Ref<Coursetext>> edit inverse Coursetext：： editor
};
class Coursetext： public Persistent_Object {
public： string cname；
string textname
        Ref <Faculty> teacher inverse Faculty：： teach；
Ref <University> editor inverse University：： edit；
};
```

三、ODMG C++对象操纵语言

例：在上述数据库中，插入教师开课信息：某教师开设了一门课及所使用教材的编写学校 (fno，cname，textname，uname) 。

假设教师及学校的数据均已在数据库中存在，插入操作算法为：

① 打开数据库；

② 事务开始；

③ 查询工号为fno值的Faculty对象ofa；

④ 查询校名为uname值的University对象oun；

⑤ 创建Coursetext对象oco，送入cname和textname值；

⑥ 在oco的teacher中插入Faculty对象ofa；

⑦ 在oco的editor中插入University对象oun；

⑧ 事务提交（commit）。

§4　ODMG 97和对象语言

一、 ODMG数据模型

ODMG数据模型是OODBMS的基础。OODB是对象的集合，每个对象有一个惟一的对象标识符（OID），类是具有类似性质的对象的汇集。类的定义由三部分组成：

（1）属性(Attributes):属性可以是基本类型，也可以是复合类型。复合类型有struct、array、list、bag、set等五种。

（2）联系(Relationships):联系是指对象之间的引用或引用的汇集。ODMG模型中的联系与ER模型中的二元联系相类似。

（3）方法(Methods):方法是能应用到类的对象上的函数。

二、ODMG ODL

ODMG数据库模式定义为一系列接口（interface）的汇集。

关键字interface（接口）用来定义一个类。对于每个接口，可以说明一个extent（范围），它是代表类的当前对象集的一个名字。

实际上接口和范围类似于关系模型中的关系模式和关系实例。

例:对于前面的模式和对象联系图，我们可以用ODMG ODL来定义,形式如下：

```
interface Person
        (extent People)
{ attribute string name;
  attribute integer age;
};
interface Faculty: Person              /* 类Faculty是类Person的子类 */
        (extent Faculties key fno)
{ attribute integer fno;
  attribute integer salary;
  relationship University works_for inverse University::staff;
  relationship Set<Coursetext> teach inverse Coursetext::teacher;
  integer num_teach () raises (noTeach);  /*统计教师授课门数的一个方法*/
                              /* raises（引发）表示该方法可能引发的异常 */
};
```

```
interface University
        (extent Universities key uno)
{ attribute integer uno;
  attribute string uname;
  attribute string city;
  relationship Faculty president;
  relationship Set<Faculty> staff inverse Faculty::works_for;
  relationship Set<Coursetext> edit inverse Coursetext::editor;
  integer num_staff ( ) ;    /* 统计学校人数的一个方法 */
};
interface Coursetext
        (extent Coursetexts)
{ attribute string cname;
  attribute string textname;
  relationship Faculty teacher inverse Faculty::teach;
relationship University editor inverse University::edit;
};
```

三、ODMG OQL

1. OQL中的SELECT语句

OQL允许人们用传统的SELECT查询语句来写表达式，也具有消除重复、子查询、排序等功能。

例：用OQL的SELECT语句可以写出下列查询操作。

① 检索大学里授课门数超过3门的教师。要求显示大学校名和教师姓名，显示时属性名为university_name和faculty_name：

```
SELECT university_name:F.works_for.uname
       faculty_name:F.name
FROM Faculty F
WHERE F.num_teach()> 3;
```

② 检索上海地区大学中教师开设课程的课程名。

```
SELECT DISTINCT C.cname
FROM University U, U.staff F, F.teach  C
WHERE U.city =′ shanghai′ ;
```

也可以用子查询形式表达，但子查询是出现在FROM子句中：

```
SELECT DISTINCT C.cname
FROM SELECT U
       FROM University U
        WHERE U.city =′ shanghai′ ) D1,
      (SELECT F
       FROM D1.staff F) D2,
       D2.teach C;
```

这个语句也可写成WHERE子句中嵌子查询的形式：

```
SELECT DISTINCT C.cname
FROM Coursetext C
WHERE C.teacher IN
      ( SELECT F
        FROM Faculty F
        WHERE F.works_for IN
                  ( SELECT U
                    FROM University U
                    WHERE U.city =′ shanghai′ ));
```

③ 检索复旦大学的教师，要求按年龄降序排列，若年龄相同按工资升序排列。

```
SELECT F

FROM University U, U.staff  F

WHERE U.uname =′ Fudan University′

ORDER BY F.age DESC, F.salary;
```

OQL中SELECT语句查询结果是集合（set）或包（bag），但加了ORDER BY子句后，输出结果就成为列表（List）。在集合、包中，行序是无所谓的，但在列表中，行序是重要的。

④ 下面查询返回的是列表值而不是集合或多集：

```
(SELECT F.fno, F.name
 FROM Faculty  F
 ORDER BY F.age  DESC) [0:4];
```

该查询返回的是年龄最大的五位教师姓名值。

表达式[0:4]表示抽取年龄最大的5个教师。

⑤ 检索上海地区各大学中教师开课的课程名，要求显示校名、教师名、课程名：

```
SELECT  Struc(uname，set(name，set(cname)))
        FROM  University  U，U.staff  F，F.teach  C
        WHERE  U.city = ′ shanghai′ ；
```

SELECT子句中的表达式不必都是简单的变量，可以是任何表达式（包括用类型构造符构成的表达式）。上式中用了struct类型构造符和set类型构造符。SELECT子句后的struct字样，是一种显式地定义结构类型的方式，在实际使用时也可省略.

2．OQL表达式的附加格式

OQL在SELECT语句格式中提供了全称量词（FOR ALL）和存在量词（EXISTS）等谓词，以及聚集运算符、分组子句和集合运算符（并、交和差）。

（1） 量词表达式　全称量词表达式的句法：

FOR ALL x IN S：C(x)

该表达式用于检测集合S的所有成员x是否都满足条件C(x)。如果S中每个成员x都满足C(x)，则该表达式结果为true，否则为false。

存在量词的表达式的句法：

EXISTS x IN S：C（x）

该表达式用于检测集合S中是否至少有一个成员x满足条件C（x）。若存在，则该表达式结果为true，否则为false。

例：写出下列查询操作的SELECT语句。

① 检索存在60岁以上教师的大学校名。

```
SELECT DISTINCT U.uname
FROM University  U
WHERE EXISTS F IN U.staff:
F.age>=60;                    /* F是元组变量 */
```

② 检索教师年龄全在50岁以下的大学校名。

```
SELECT U.uname
FROM University  U
WHERE FOR ALL F IN U.staff:
  F.age<50; ※
```

（2） 使用聚集操作和分组子句的SELECT语句

OQL使用与SQL相同的五种聚集运算符：AVG，COUNT，SUM，MIN和MAX。在传统的SQL中，这些运算符只能应用于表中指定列，而OQL中，同样运算符可应用于其成员为合适类型的聚集操作。即：COUNT可应用于任何聚集，SUM和AVG可以用于基本类型的聚集，MAX和MIN可以用于任何可比较类型的聚集。

OQL的SELECT语句也使用分组子句和组条件表达式子句，但增加了新意。

例:写出下列查询操作的SELECTS语句。

① 检索每个年龄段教师平均授课门数。

```
SELECT F.age，avgNum:AVG（SELECT P.F.num_teach（）
                                    FROM partition  P）
    FROM Faculty F
    GROUP BY F.age;
```

（3）集合运算符 像传统的SQL一样，OQL中也有并、交、差等集合操作。

例:检索教师人数不到1000人，但工资低于1500元的人数超过500人的那些大学的编号和校名。

```
(SELECT U.uno, U.uname
 FROM University U
 GROUP BY U.uno, U.uname
HAVING U.num_staff () < 1000 )
 EXCEPT
(SELECT U.uno, U.Uname
 FROM University U, U.staff  F
  WHERE F.salary < 1500
  GROUP BY U.uno, U.uname
HAVING U.unm_staff () > 500 ) ;
```

3．OQL中对象的赋值和建立

（1）对宿主语言变量赋值

传统SQL需要在元组分量和宿主语言变量之间传递数据，而OQL则不同，可以很方便地把表达式的结果值赋给任何合适类型的宿主语言变量。

例：检索大于60岁的教师可用下列语句：

```
SELECT  F   FROM  Faculty  F   WHERE  F.age > 60;
```

该查询结果的类型是set〈Faculty〉。如果oldFaculties是同类型的宿主语言变量，用扩充了OQL的C++可以写成下列形式：

```
oldFaculties = SELECT  F
               FROM  Faculty  F
               WHERE  F.age > 60;
```

并且oldFaculties的值将成为这些Faculty对象的集合。※

（2）从聚集中提取元素

获取集合或者包的每个成员是比较复杂的，但比传统SQL基于游标的方法要简单。首先，我们需要把集合或者包转换成列表，这可以用带ORDER BY子句的方法。

例：检索大于60岁的教师，要求查询结果按工资、年龄降序排列，

可用下列语句实现：

```
facultyList = SELECT F
              FROM Faculty F
              WHERE F.age > 60
              ORDER BY F.salary DESC，F.age DESC；
```

该语句将把按工资、年龄降序排列的所有Faculty对象的列表赋给宿主语言变量facultyList。

§5 OODB与ORDB的比较

1. 各种DBS的长处和优势可以概括如下：

（1）关系系统：数据类型简单，查询语言功能强大，高保护性。

（2）基于持久化语言的OODBS：支持复合数据类型，与程序设计语言集成一体化，高性能。

（3） ORDBS：支持复合数据类型，查询语言功能强大，高保护性。

该总结具有普遍性，但对有些DBS而言它们的分界线是模糊的。例如，有些以持久化语言为基础的OODBS是在一个关系DBS之上实现的，这些系统的性能可能比不上那些直接建立在存储系统之上的OODBS，但这些系统却提供了关系系统所具有的较强保护能力。

2. OODB与ORDB的主要区别如下所示：

OODB	ORDB
从OOPL　C++出发，引入持久数据的概念，能操作DB，形成持久化C++系统	从SQL出发，引入复合类型、继承性、引用类型等概念（SQL 3）
ODMG OQL（类似于SQL）	SQL3
有导航式查询,也有非过程性查询	结构化查询，非过程性查询
符合面向对象语言	符合第4代语言
显式联系	隐式联系
惟一的对象标识符	有主键概念，也有对象标识概念
能够表示“关系”	能够表示“对象”
对象处于中心位置	关系处于中心位置

§6 使用UML类图来概念对象建模

一、 统一建模语言（Unified Modeling Language: UML）

UML适用于各类系统的建模，为了实现这种大范围应用能力

UML被定义成比较粗放和具有普遍性，以满足不同系统的建模。

通过提供不同类型生动的图，UML能表达系统多方面的透视，这些图有使用：

事件图（Use-Case Diagram）、

类图（Class Diagram）、

状态图（State Diagram）、

组件图（Component Diagram）

等9种。

下面只描述强调系统的数据、某些行为和面貌的类图

二、用类图表达类和关联

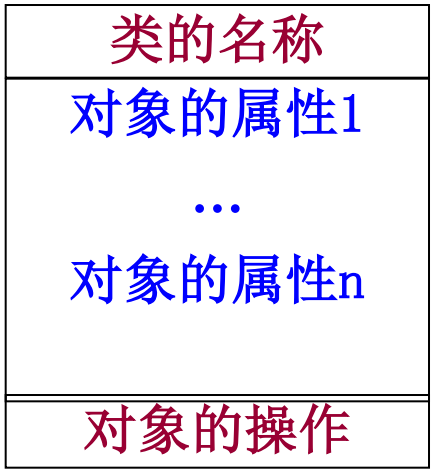
类图描述了系统的静态结构，包括类和类间的联系。

类图与ER图、对象联系图有很多类似的地方，但所用的术语和符号有所不同。

1. 类图与ER图中术语的区别

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

2. 类图中的基本成分是类和关联：



（1）类被表示为由三个部分组成的方框：

- 上面部分给出了类的名称；
- 中间部分给出了该类的单个对象的属性；
- 下面部分给出了一些可以应用到这些对象的操作。

（2）关联是对类的实例之间联系的命名，相当于ER模型中的联系类型。

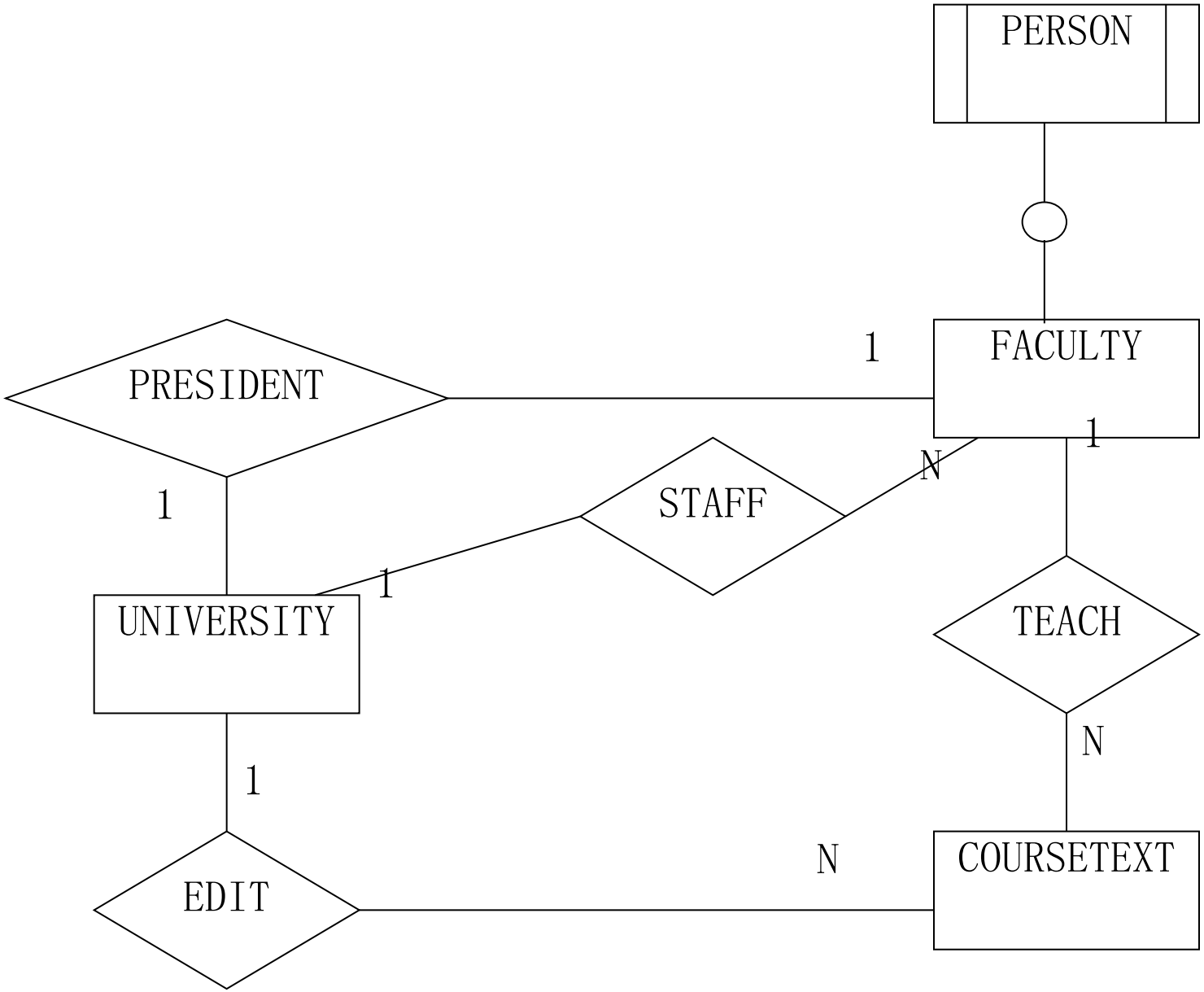
与关联有关的内容有：

- 关联元数(degree)：与关联有关的类的个数，称为关联元数或度数；
- 关联角色(role)：关联的端部，也就是与关联相连的类，称为关联角色。

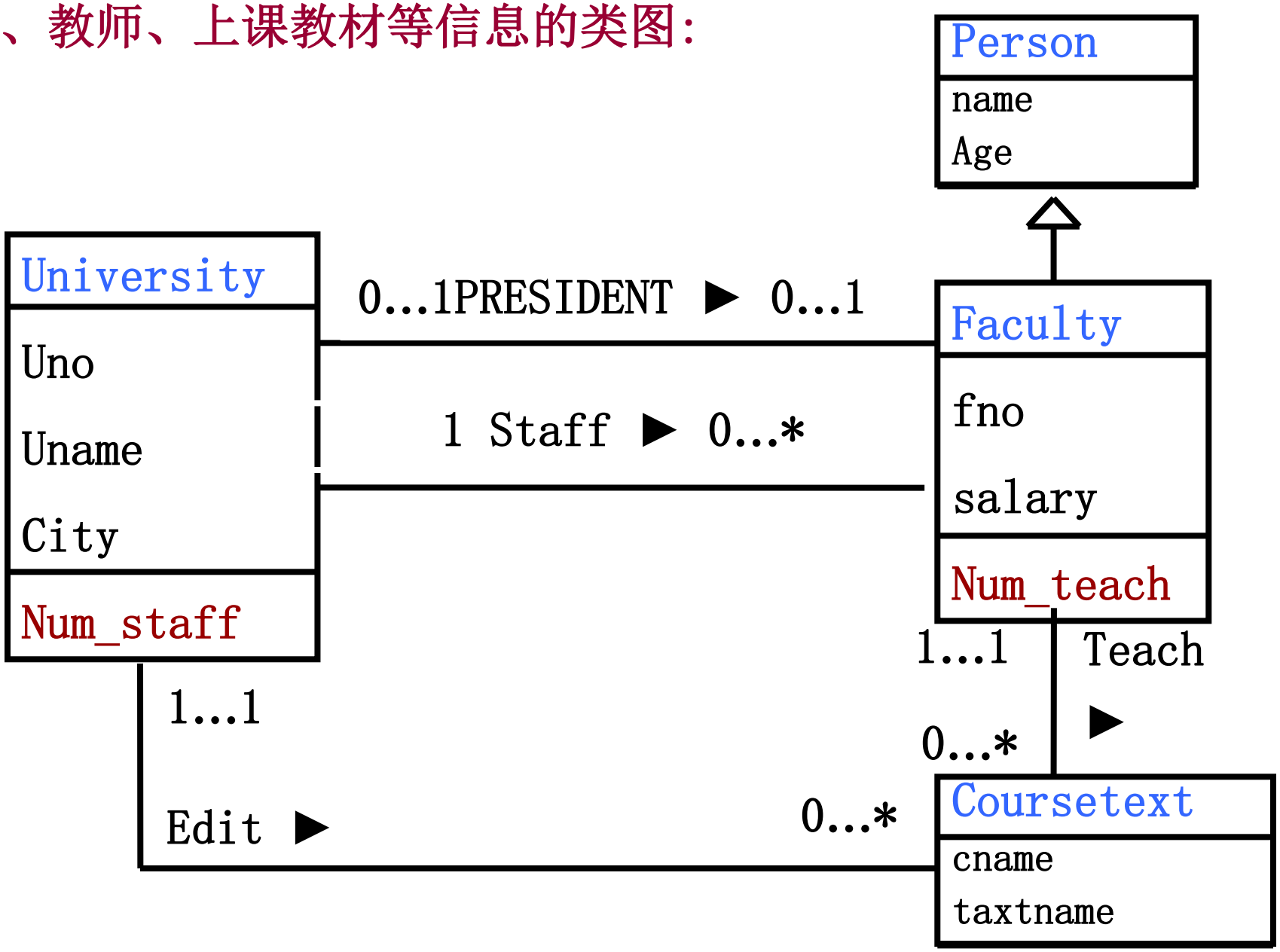
  角色名可以命名，也可以不命名，就用类的名字作为角色名。
- 重复度(multiplicity)：重复度是指在一个给定的联系中有多少对象参与。

  即是关联角色的重复度。

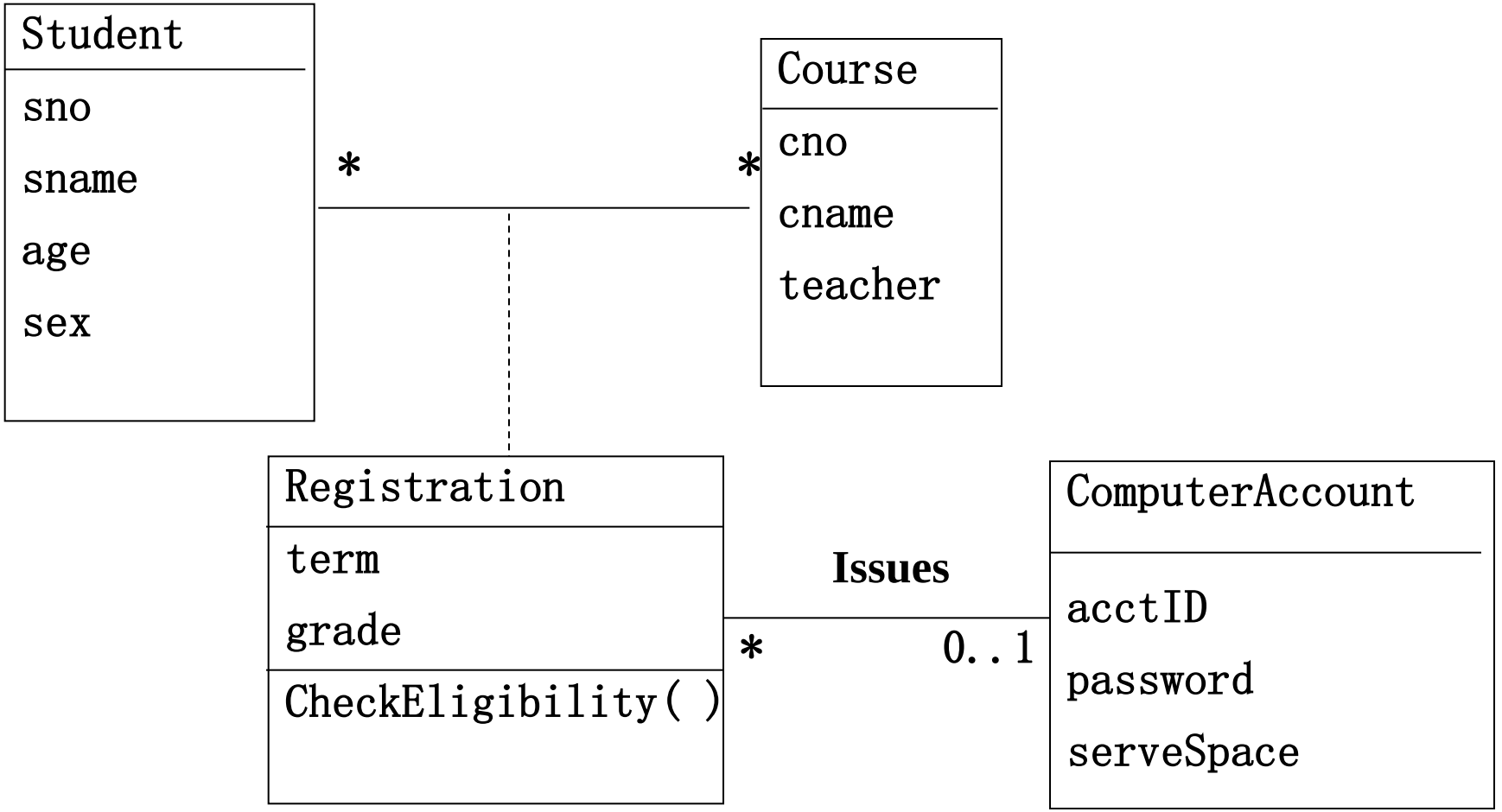
例 :大学、教师、上课教材等信息的ER图



大学、教师、上课教材等信息的类图:

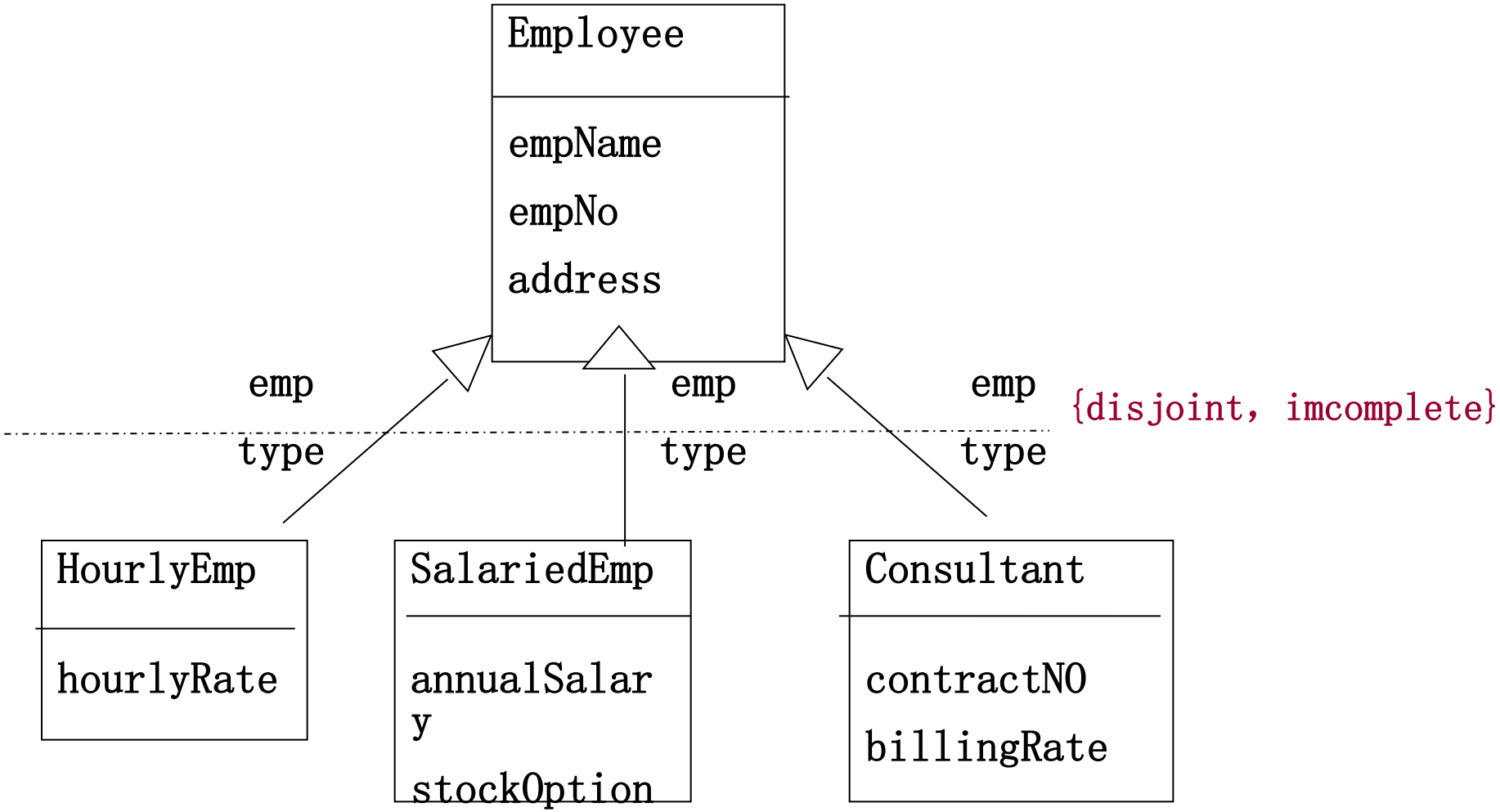


3. 用类图表达关联类

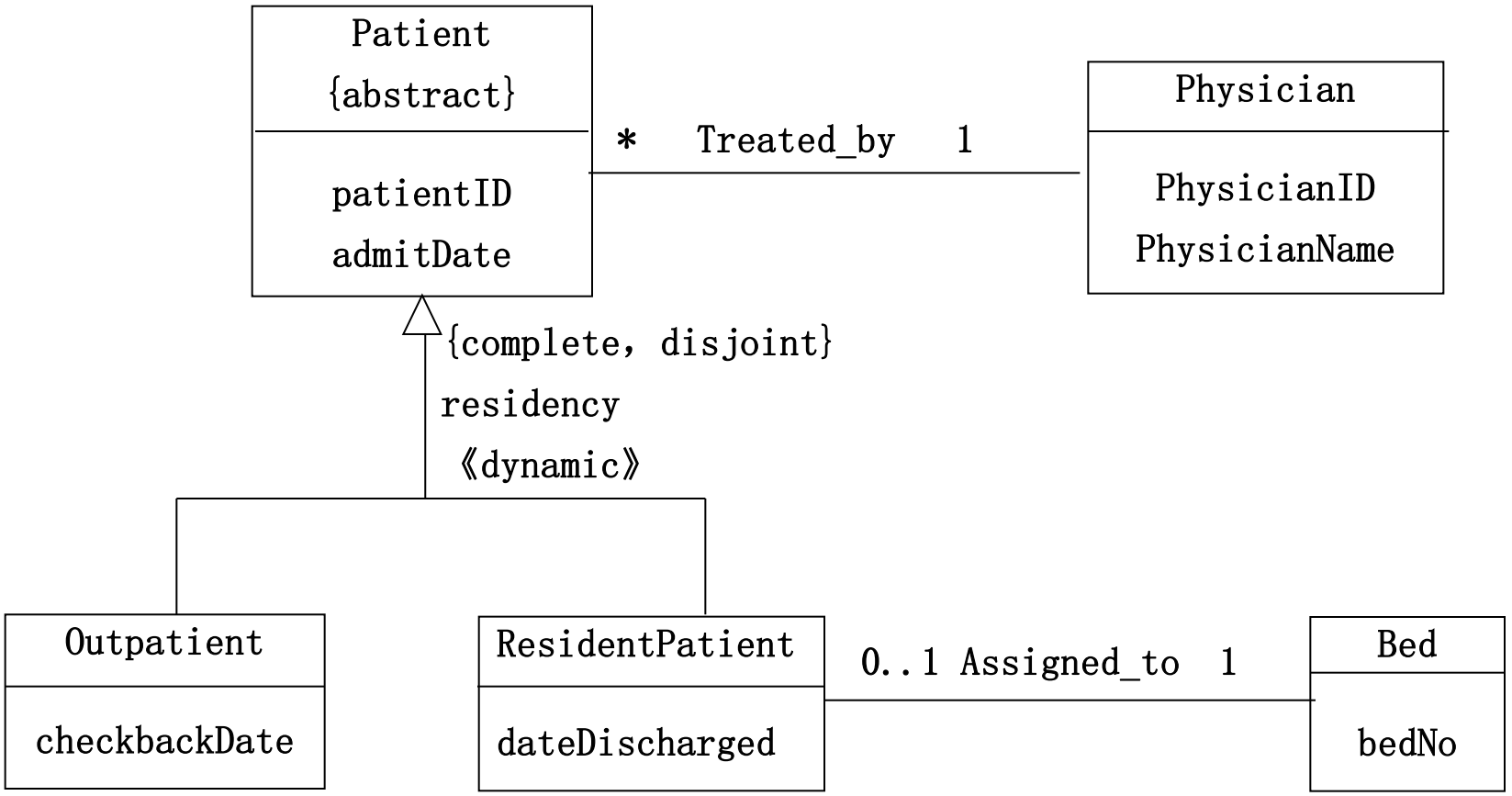

Student
sno
sname
age
sex
*
*
Course
cno
cname
teacher
Registration
term
grade
CheckEligibility( )
Issues
*
0..1
ComputerAccount
acctID
password
serveSpace

4．用类图表达泛化/细化

(1)带有三个子类的Employee超类



(2) 带有两个具体子类的抽象的Patient类



精读和习题要求

精读教材： P.232～253

习 题11： P.254 11、12