
第2章 数据模型

第2章 数据模型

- 2.1 数据模型的基本概念
- 2.2 数据模型的四个世界
- 2.3 概念世界与概念模型
- 2.4 信息世界与逻辑模型
- 2.5 计算机世界与物理模型

2.1 数据模型的基本概念

数据库是一个统一、集中的数据管理机构，它在保存用户所需数据的同时，也必须具有向外界提供数据服务的功能，即为用户提供**存取**数据库中数据的手段。因此如何描述现实世界中各种各样的数据和它们之间的各种复杂的关系，实现用户对数据的操作要求，并最终以计算机及数据库所允许的形式反映到数据库中去，这是一个非常重要的问题

在这里，我们使用**数据模型**这个概念来描述现实世界中的数据及其相互关系，定义在这些数据上可以执行的操作

2.1 数据模型的基本概念

□ 什么是数据模型？

➤ 数据是对于现实世界的符号抽象，而数据模型则是对数据特征的抽象，为数据库系统的信息表示和操作提供一个抽象框架，是数据库系统的核心与基础

— 数据模型应该能比较真实地模拟现实世界、易于人理解、便于在计算机上实现

数据模型

特征抽象

数据

符号化

现实世界

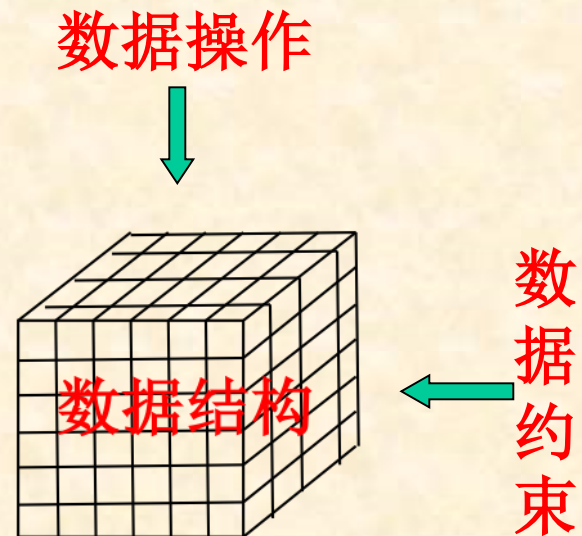
2.1 数据模型的基本概念

□ 数据模型 (data model)

【定义】描述数据的结构，定义在该数据结构上可以执行的操作以及数据之间必须满足的约束条件

➤ 数据模型的组成

- 数据结构
- 数据操作
- 数据约束



2.1 数据模型的基本概念

□ 数据模型 – 数据结构

➤ 描述数据的类型、内容、性质以及数据间的联系

– 数据结构是一个数据模型的基础，数据操作与数据约束均是建立在相应的数据结构上的

▪ 在这之前，数据模型中的数据结构被称为‘数据模式’

– 这也是不同类型数据模型的划分依据

2.1 数据模型的基本概念

□ 数据模型 – 数据操作

- 在相应数据结构上可以执行的操作类型与操作方式
 - 在不同的数据结构上可以提供不同的操作方式与操作类型

□ 数据模型 – 数据约束

- 主要描述数据结构内数据间的相互关系，包括：
 - 数据间的语法/语义联系
 - 数据间的制约与依存关系
 - 数据（间）的动态变化规则
- 其目的是确保数据的正确、有效与相容

2.1 数据模型的基本概念

- 数据模型的核心是数据结构，如何将现实世界中我们需要的数据及其复杂关系最终反映到数据库中去，这需要有一个逐步转化的过程
- 我们用建立在不同抽象层次上的‘数据模型’来表示每一步转化的结果：
 - 概念数据模型 (conceptual data model)
 - 又简称为 ‘概念模型’
 - 逻辑数据模型 (logic data model)
 - 又简称为 ‘数据模型’
 - 物理数据模型 (physical data model)
 - 又简称为 ‘物理模型’

2.1 数据模型的基本概念

□ 概念数据模型

➤ 侧重于对客观世界中复杂事物的结构描述及它们之间的内在联系的刻画，不涉及具体的描述细节和物理实现因素

— 是一种面向客观世界和用户的模型，与具体采用的DBMS及计算机实现无关

— 主要的几种概念模型

- E-R模型，EE-R模型
- 面向对象模型
- 谓词模型

2.1 数据模型的基本概念

□ 概念数据模型（cont.）

- 概念数据模型主要描述这些客观对象的数据特征及其相互关系
- 【例】 学生，教师，运动员，教练

2.1 数据模型的基本概念

□ 这些客观对象的数据特征如下：

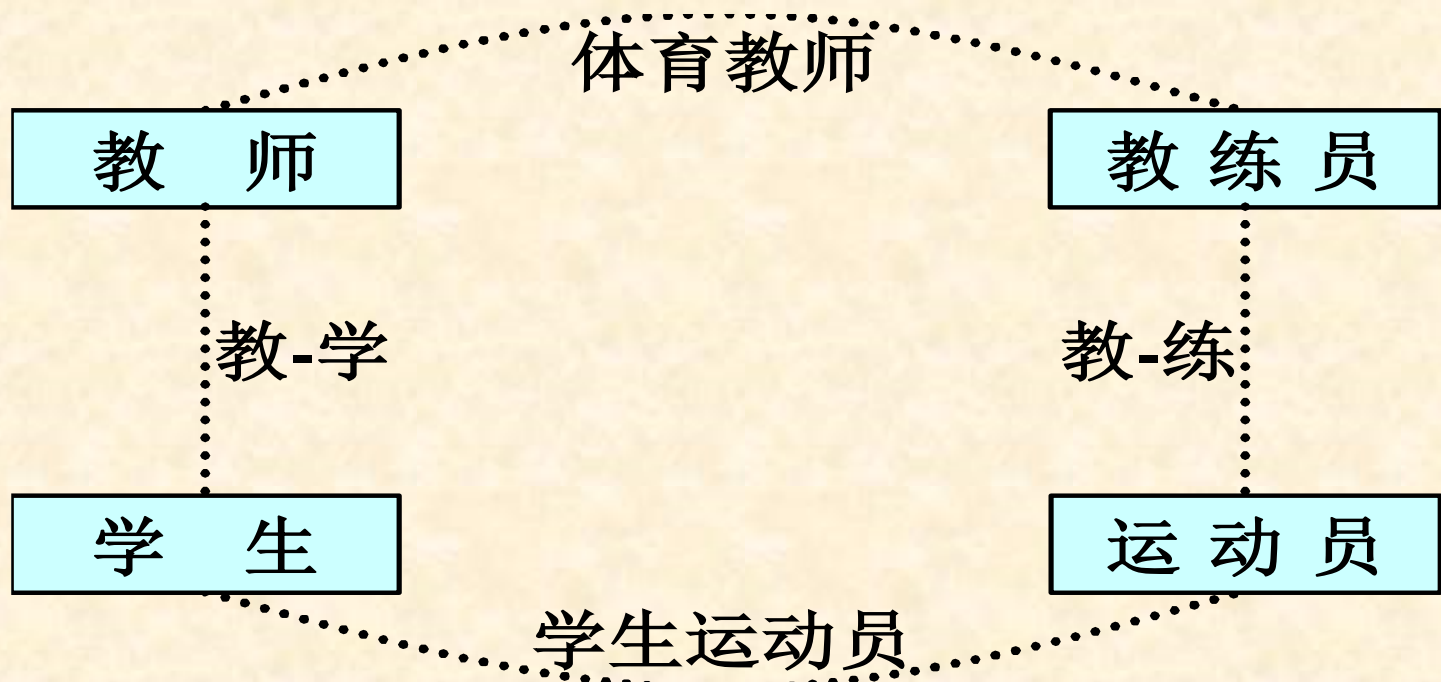
对象	数据特征
学 生	学号，姓名，系，专业，性别，入学年份
教 师	姓名，性别，出生日期，职称，学历，参加工作时间，工作证编号，身份证号
运动员	姓名，性别，出生日期，从事运动项目，身高，体重
教练员	姓名，性别，出生日期，职称，从事运动项目，参加工作时间，工作证编号，身份证号

— 要了解每个数据项的语义含义，但并不需要定义其实现细节（如数据类型，取值的约束等）

2.1 数据模型的基本概念

□ 概念数据模型 (cont.)

➤ 相互关系的描述



2.1 数据模型的基本概念

□ 逻辑数据模型

- 着重于数据模型在数据库系统一级的实现，即利用具体的DBMS所提供的工具（DDL）来定义的数据模型
 - 是一种面向数据库系统的模型，概念数据模型只有在转换成逻辑数据模型后才能在数据库中得以表示
 - 是一个中介模型，具有承上启下的作用

2.1 数据模型的基本概念

□ 逻辑数据模型（cont.）

➤ 成熟并（曾经）得到大量使用的逻辑数据模型有：

— 层次模型、网状模型

— 关系模型、面向对象模型、谓词模型

— 对象关系模型

2.1 数据模型的基本概念

□ 逻辑数据模型（cont.）

- 需要描述每个客观事物及其相互关系在选定的**DBMS**中的实现结构
- 即根据选定的**DBMS**来定义客观事物及其相互关系的实现结构

2.1 数据模型的基本概念

□ 例如：

➤ 客观事物的实现结构：

- 关系数据库：表及其属性的定义，如：
 - ✕ 属性的名称、数据类型、取值约束等
 - ✕ 表级的取值约束
- 面向对象数据库：类及其属性、方法的定义

➤ 相互关系的实现结构：

- 关系数据库：表及其外键
- 面向对象数据库：类的继承与合成关系

2.1 数据模型的基本概念

□ 物理数据模型

- 给出了数据模型在计算机内部的真正物理结构，是一种面向计算机物理实现的模型
- 一个概念数据模型将首先被转化为某一种逻辑数据模型，并通过所选择的DBMS将其进一步转化为具体的物理数据模型，才能使其在计算机中得以物理实现

2.1 数据模型的基本概念

□ 物理数据模型 (cont.)

➤ 大都由选定的某种DBMS来负责数据库物理存储结构的选择，但也向用户提供了一些与物理存储结构和存取方法有关的定义功能，如：

— 索引 (index) 的定义

— 集簇 (cluster) 的定义

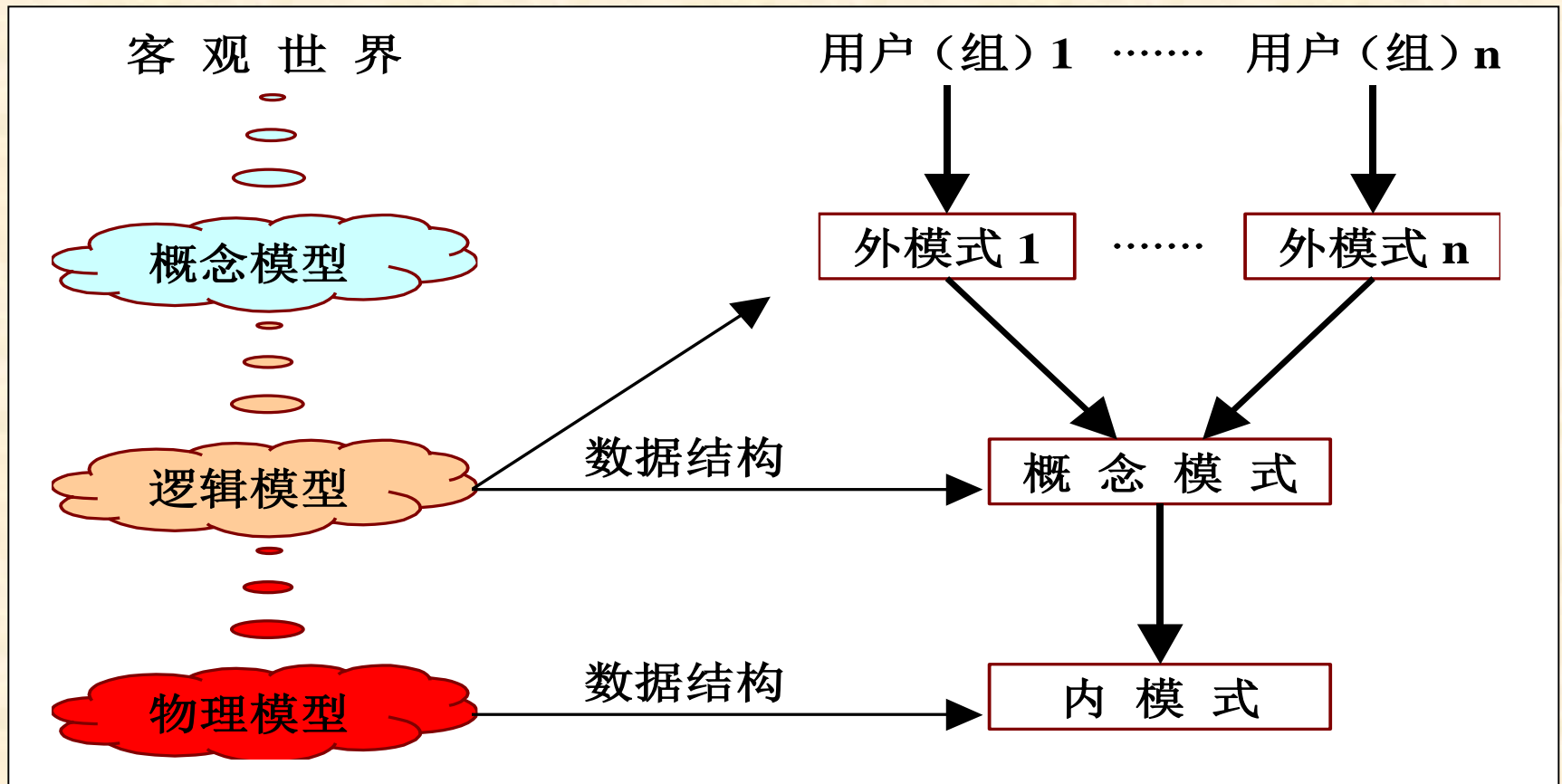
— 存储区域的选择

■ 例如：Oracle用户可以选择表中数据的物理存储所使用的表空间 (tablespace) 等

2.1 数据模型的基本概念

□ 三种‘数据模型’与‘三级模式’之间的关系

- 是不同的‘模型’与‘模式’的分层方法，请注意它们之间的区别



第2章 数据模型

2.1 数据模型的基本概念

2.2 数据模型的四个世界

2.3 概念世界与概念模型

2.4 信息世界与逻辑模型

2.5 计算机世界与物理模型

2.2 数据模型的四个世界

□ 使用‘**数据模型**’概念可以将现实世界中用户的复杂要求反映到计算机数据库中的实现，这种反映是一个逐步转化的过程，它分为四个阶段，我们称其为四个世界。每一次转化都是一个加工与提高的过程

—现实世界

—概念世界

—信息世界

—计算机世界

2.2 数据模型的四个世界

□ 现实世界

- 在客观世界中根据用户的需求目标而划定边界的一个应用环境

— 用户需求

- 数据需求
 - 处理要求
- } 从而确定了数据库应支持的应用功能和应用范围

— 现实世界为整个转换过程提供了客观基础与初始启动环境

2.2 数据模型的四个世界

□ 概念世界

- 以现实世界为基础作进一步的抽象而形成的概念模型
- DB设计人员经过调查研究会从用户那里获得一组庞大复杂的需求信息，由于用户在理解要求和描述表示上的不足，因此难免会向DB设计人员提供一些多余的、错误的或者不准确的信息，因而需要DB设计人员对其作分析提高，去粗取精，去伪存真，最后形成一些基本概念与基本关系。这些基本概念与基本关系可以用我们所选择的某一种概念数据模型中所提供的术语和方法来统一表示

2.2 数据模型的四个世界

□ 概念世界中的基本术语

【例】 E-R模型： 实体， 属性， 联系

【例】 OO模型： 对象， 类， 方法， 继承，

➤ 概念世界与具体的DBMS和计算机无关

2.2 数据模型的四个世界

□ 信息世界

- 以概念世界为基础，选用特定的**DBMS**构造而成的逻辑数据模型
 - 侧重于概念数据模型的细化和在数据库系统统一级的实现，即利用特定的**DBMS**所提供的工具来定义逻辑数据模型
 - 该模型的定义与具体的**DBMS**有关

2.2 数据模型的四个世界

□ 计算机世界

➤ 基于逻辑数据模型在计算机中的物理实现而形成的物理数据模型

— 侧重于数据库物理存储结构的描述

- 存储结构的设计
- 存取路径的设计

✧ 文件结构的选择：堆/直接/索引 文件等

✧ 索引/集簇 的设计

- 存储空间的分配

— 是DB的最终实现结构

2.2 数据模型的四个世界



图2-1：四个世界的转化示意图

第2章 数据模型

2.1 数据模型的基本概念

2.2 数据模型的四个世界

2.3 概念世界与概念模型

2.4 信息世界与逻辑模型

2.5 计算机世界与物理模型

2.3 概念世界与概念模型

□ 概念世界是一个较为抽象、概念化的世界，它给出了数据的概念化结构。概念世界一般用概念模型表示

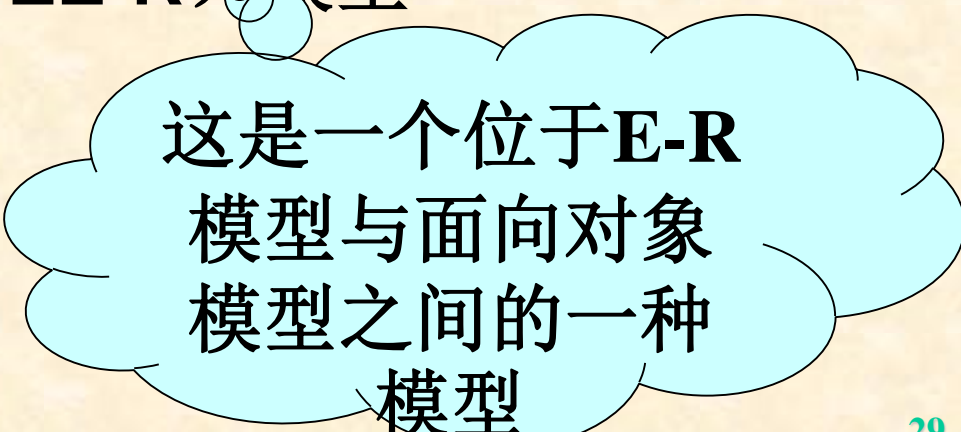
□ 常用的几种概念模型

➤ 实体-联系（E-R）模型

➤ 扩充的实体-联系（EE-R）模型

➤ 面向对象模型

➤ 谓词模型



这是一个位于E-R模型与面向对象模型之间的一种模型

2.3.1 实体-联系模型

□ 实体-联系模型

- **Entity-Relationship model**, 简称**E-R模型**
- 这是一种概念化的模型, 它将现实世界的要求转化成实体、联系、属性等基本概念及它们之间的两种基本关系, 并且用一种较为简单的图表示, 称**E-R图** (Entity-Relationship diagram)

— E-R模型中的基本概念

- 实体
 - 属性
 - 联系
- } 以及它们之间的连接关系

2.3.1 实体-联系模型

□ 实体(Entity)

- 客观存在且又能相互区别的事物
 - 是对现实世界中的客观事物的抽象，是概念世界中的基本单位

➤ 实体集

- 由具有共性的实体所构成的集合

2.3.1 实体-联系模型

□ 属性 (Attribute)

- 实体所具有的某种特性或特征
 - 属性可以有值
 - 一个属性可以取的值的集合，被称为该属性的**值域** (value domain)
- 一个实体可以有多个属性
 - 所谓具有共性的实体是指这些实体含有相同的属性组成
 - 不同的实体在这些属性上的取值则存在着区别

2.3.1 实体-联系模型

□ 联系 (Relationship)

- 一个实体集中的实体与另一个实体集中的实体之间的对应关系。在概念世界中，我们用两个实体集的联系来反映它们之间的这种关系
- 联系的种类（与联系相关的实体集的个数）
 - 两个实体集间的联系 (二元联系)
 - 多个实体集间的联系 (多元联系)
 - 单个实体集内部的联系

2.3.1 实体-联系模型

□ 联系的例子

➤ 两个实体集之间的联系

- 隶属关系（教师，系别）
- 学习关系（学生，课程）
- 借阅关系（学生，图书）

选手甲和选手乙都是同一个实体集‘**运动员**’中的实体

➤ 多个实体集之间的联系

- 供应关系（工厂，产品，用户）

➤ 单个实体集内部的联系

- 围棋比赛（黑方(选手甲)，白方(选手乙)）

2.3.1 实体-联系模型

□ 相同实体集之间的多种联系

➤ 在同一组实体集之间可以存在多种联系

➤ 例如：

— 职工之间

- 管理关系（上下级关系）
- 同事关系

— 教师与研究生之间

- 教学关系
- 指导关系

2.3.1 实体-联系模型

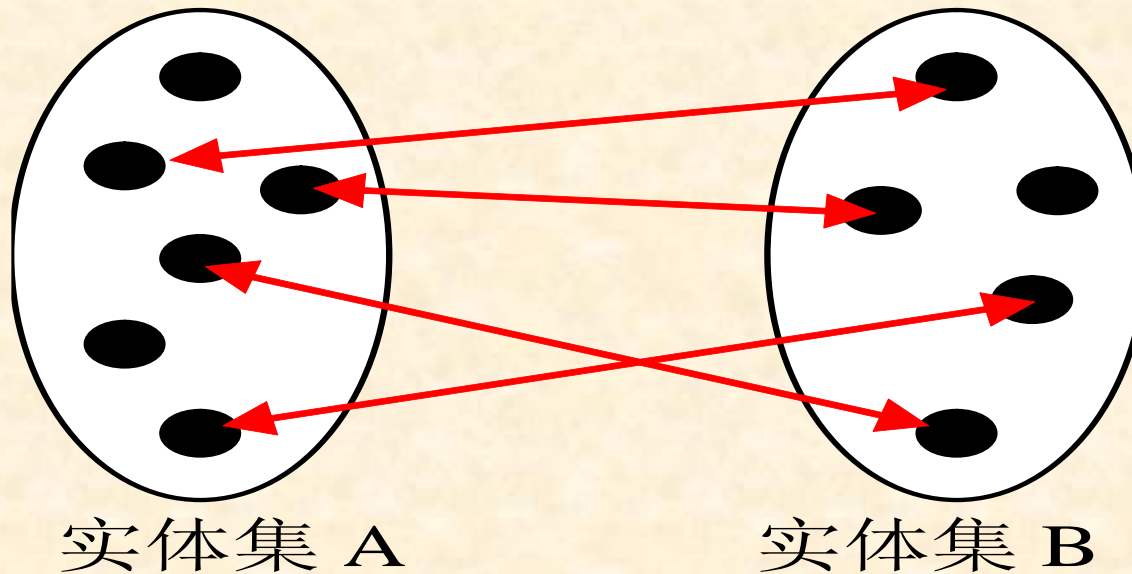
□联系的函数对应关系（图2-2）

- 一一对应（one to one）
- 一多对应（one to many）
 - 或：多一对应（many to one）
- 多多对应（many to many）

2.3.1 实体-联系模型

□ 联系的函数对应关系

➤ 一一对应 (1:1)

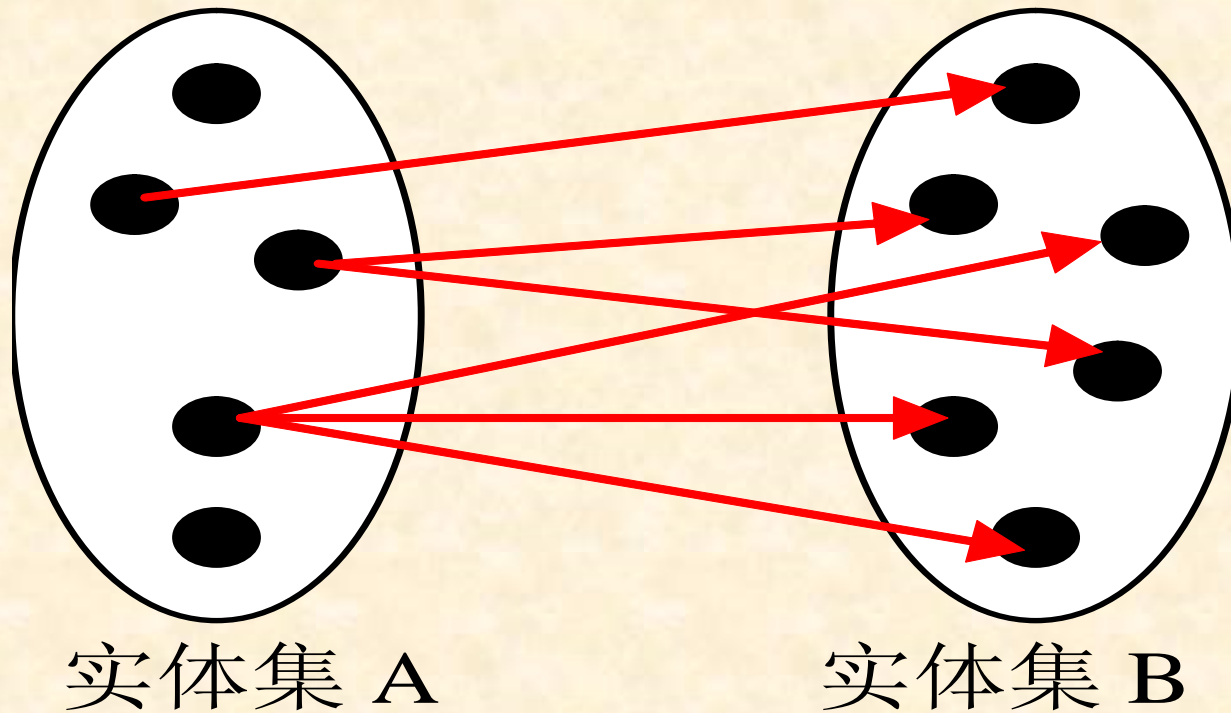


—例如：‘学校’ 与 ‘校长’
‘居民’ 与 ‘身份证’

2.3.1 实体-联系模型

□ 联系的函数对应关系

➤ 一多对应 (1 : m)

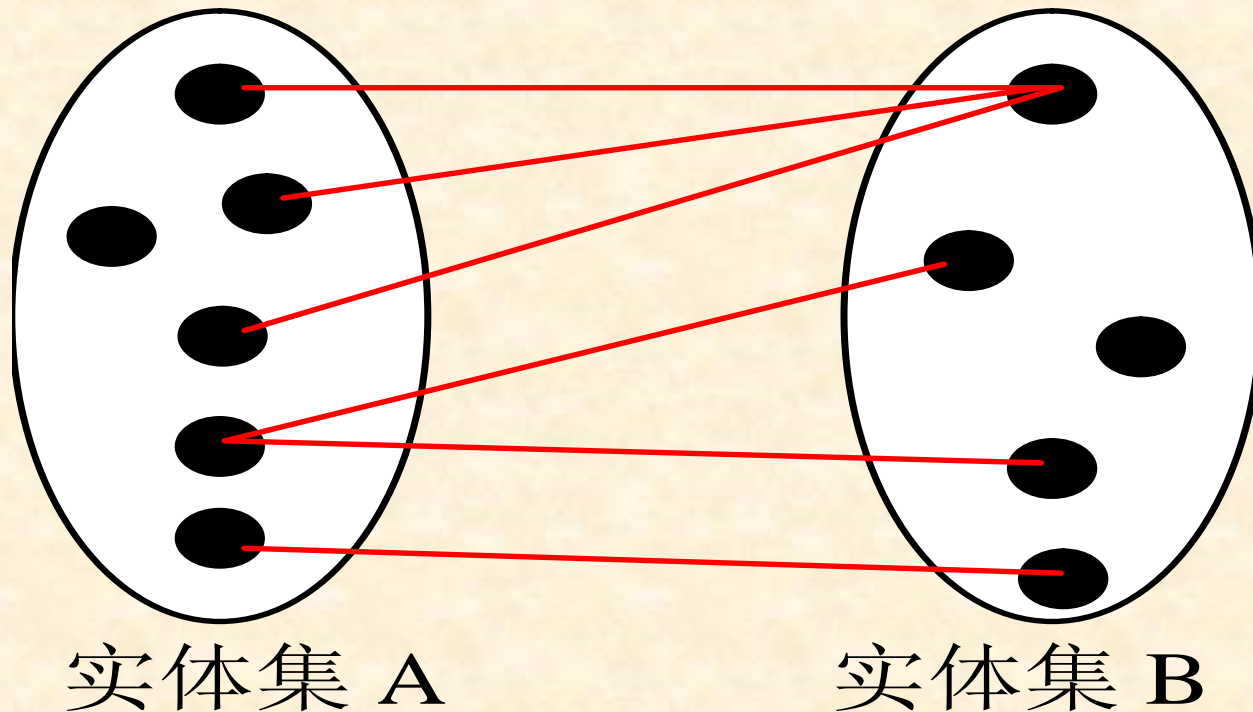


—例如：‘学生’ 与 ‘宿舍’

2.3.1 实体-联系模型

□ 联系的函数对应关系

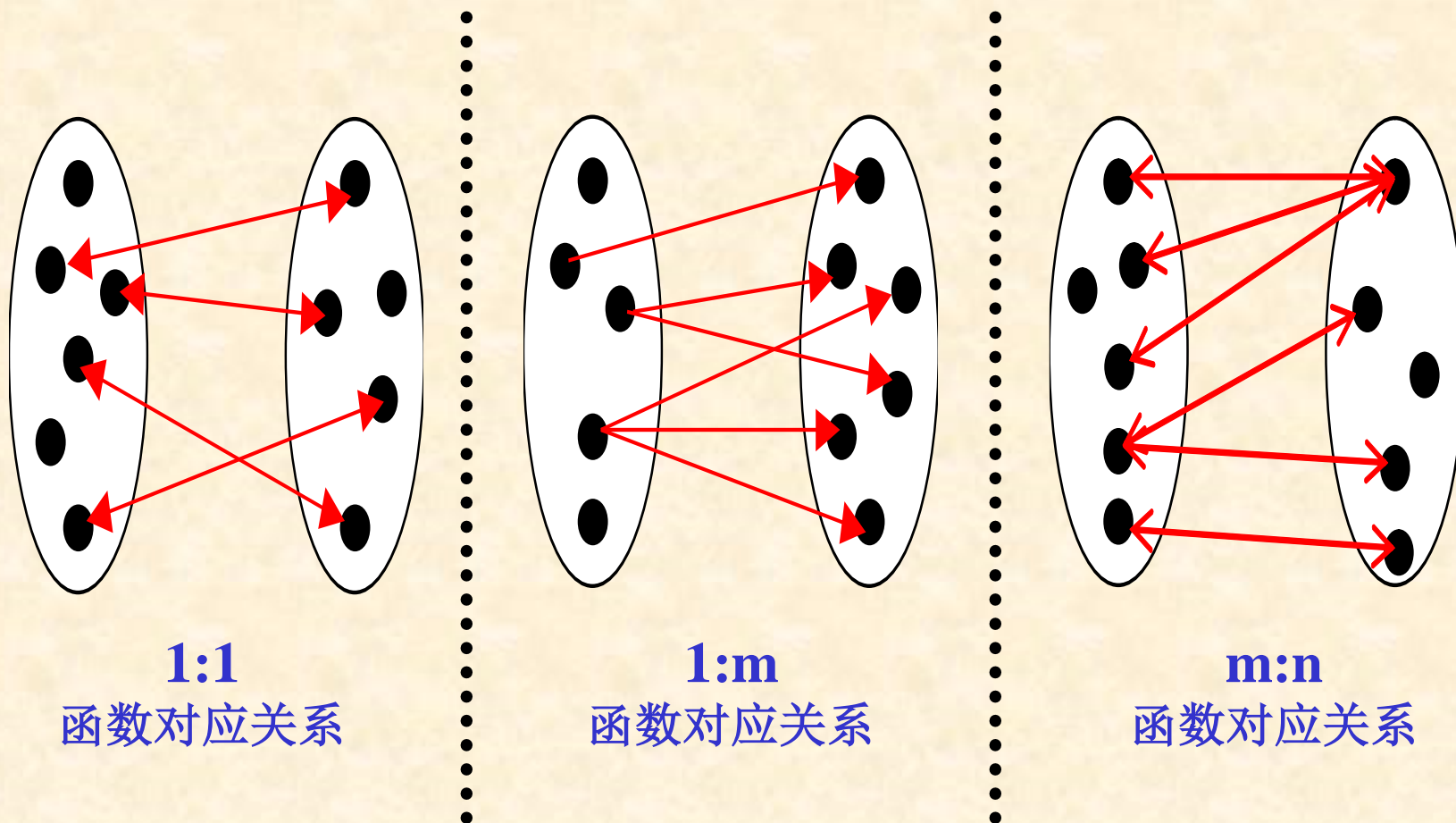
➤ 多多对应 ($m:n$)



—例如：‘学生’ 与 ‘课程’

2.3.1 实体-联系模型

□ 联系的函数对应关系



2.3.1 实体-联系模型

□联系所具有的特性

- 因联系的发生而产生的特性可以通过联系上的属性来表示

□例如：

- 学习关系（学生，课程，成绩）
- 借阅关系（学生，图书，借阅日期，归还如期）
- 比赛（甲方，乙方，比赛结果）

2.3.1 实体-联系模型

□ 基本概念之间的连接关系

- 1) 实体集（联系）与属性间的连接关系
- 2) 实体集与联系间的连接关系

□ 如何描述实体、属性、联系以及它们之间的连接关系？

2.3.1 实体-联系模型

□ 实体（集）、属性及其连接关系的描述

➤ 属性的描述：属性名，属性域

➤ 实体的描述：

— 实体名

— 实体型：实体名 + 一组属性名

▪ 用于描述实体的组成结构信息

— 实体值

▪ 实体中的每个属性都可以取值，由一个实体的所有属性的取值所构成的属性值的集合被称为该实体的实体值

▪ 在关系模型中，‘实体值’又被称为‘元组’（tuple）

2.3.1 实体-联系模型

□ 实体（集）、属性及其连接关系的描述（cont.）

➤ 实体集的描述

— 由具有相同实体型的实体所构成的集合被称为实体集

— 实体集的描述：

属性集合 + 关键字(key)

2.3.1 实体-联系模型

□ 实体（集）描述的例子，带下划线的属性为各实体集的关键字属性

实体	属性
学 生	<u>学号</u> ，姓名，系，专业，性别，入学年份
教 师	姓名，性别，出生日期，职称，学历，参加工作时间， <u>工作证编号</u> ， <u>身份证号</u>
运动员	<u>运动员证编号</u> ，姓名，性别，出生日期，从事运动项目，身高，体重， <u>身份证号</u>
教练员	姓名，性别，出生日期，职称，从事运动项目，参加工作时间， <u>工作证编号</u> ， <u>身份证号</u>

2.3.1 实体-联系模型

□ 联系及其与实体集之间的连接关系的描述

➤ **联系名**：每个联系有一个名字

➤ **属 性**：联系也可以有属性

— 由一个‘联系名’ + 与该联系相关的‘实体集的名称’，以及联系上的属性，从而构成联系及其与实体集之间的连接关系的描述

➤ **函数对应关系**

□ 例如：

选课联系（学生，课程，成绩）

2.3.1 实体-联系模型

➤如果是单个实体集内部的联系，那么在描述它们之间的连接关系时，需要描述清楚参与联系的双方（或多方）在该联系中所担当的角色

➤例如：

—职工的上下级联系（上级职工，下级职工）

—围棋比赛联系（黑方，白方）

2.3.1 实体-联系模型

□ E-R模型的图示法：E-R图

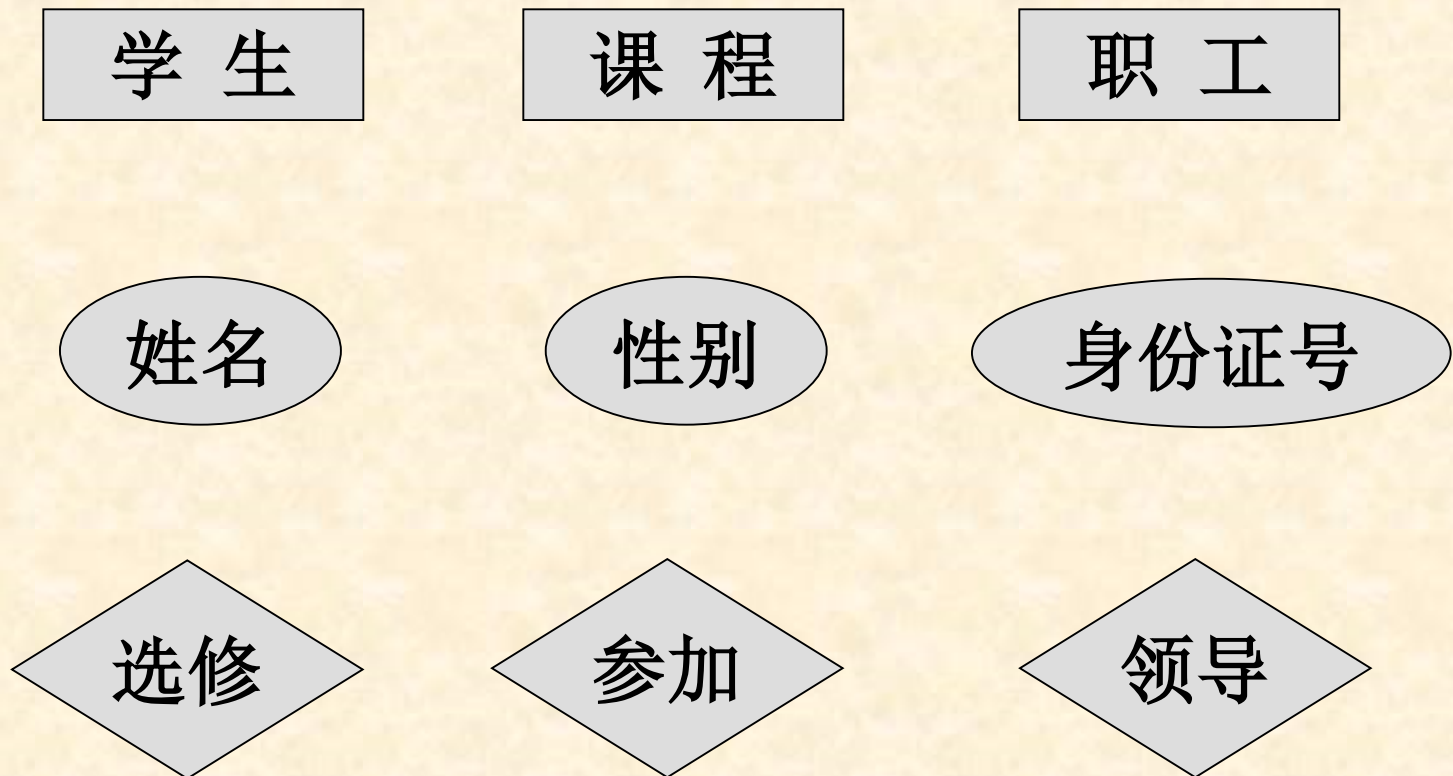
➤ 使用一些简单的图形符号来表示概念数据模型

➤ 基本概念的表示



2.3.1 实体-联系模型

□ 基本概念的名称应该写在相应的图形符号的方框范围内，例如：



2.3.1 实体-联系模型

➤ 连接关系的表示

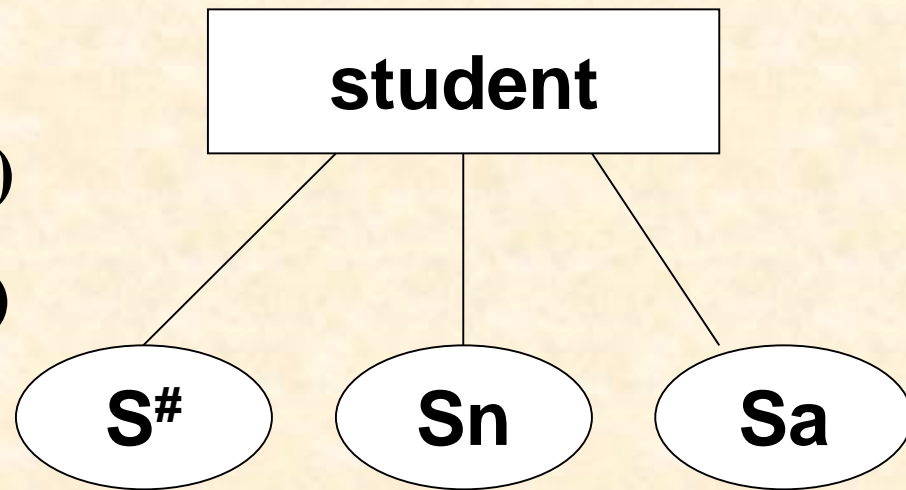
— 实体集与属性间的连接关系

- 实体集: **student**

- 属性: **S[#]** (学号)

Sn (学生姓名)

Sa (学生年龄)

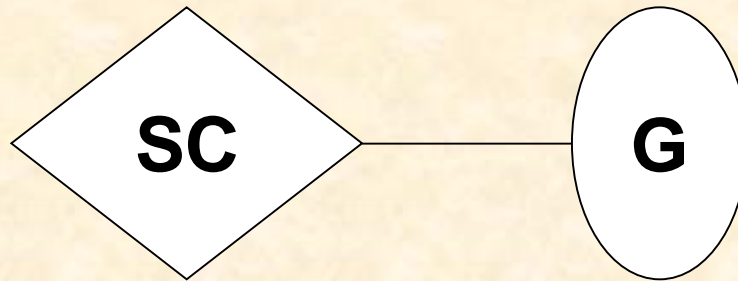


E-R图表示法

2.3.1 实体-联系模型

— 联系与属性的连接关系

- 联系：SC
- 属性：G (学生的课程成绩)

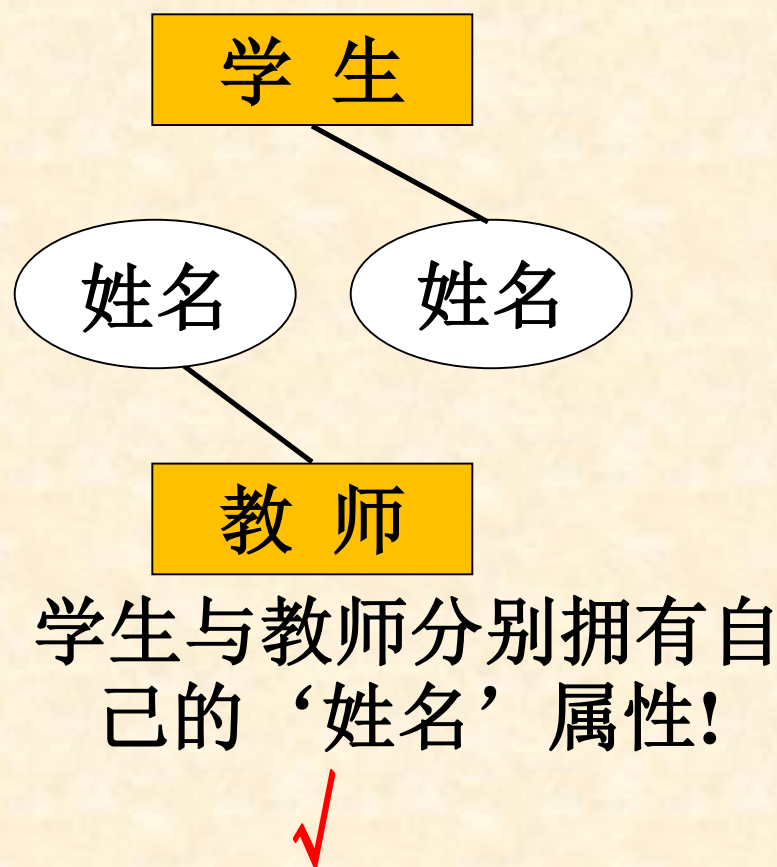
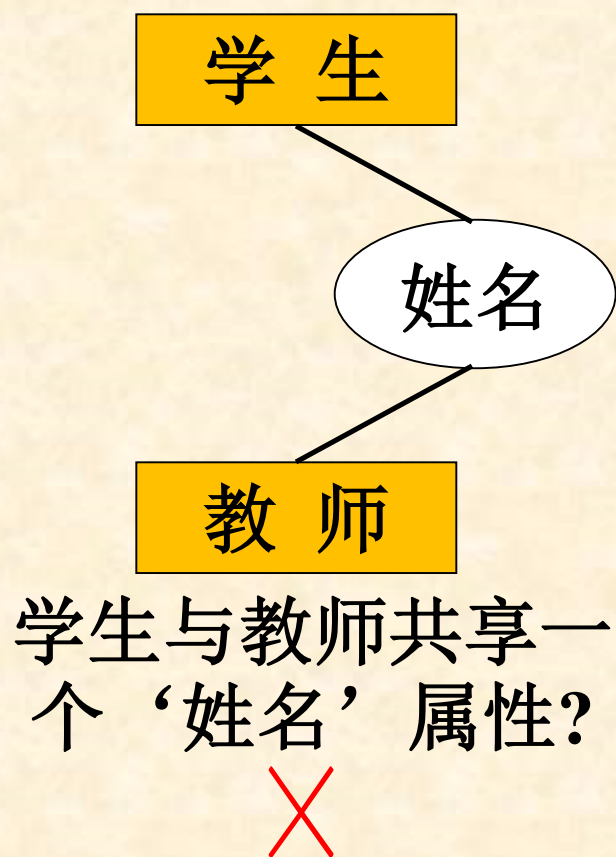


E-R图表示法

2.3.1 实体-联系模型

□ 连接关系的表示 (cont.)

- 每个实体集（联系）可以有多个属性，但每个属性只能隶属于一个实体集（联系）

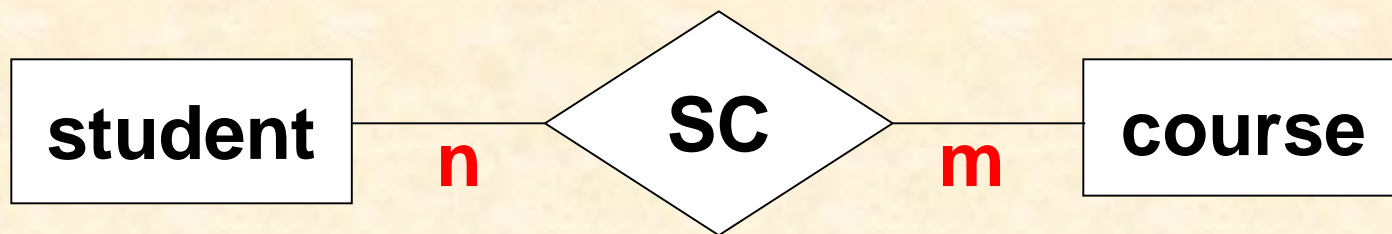


2.3.1 实体-联系模型

➤ 连接关系的表示 (cont.)

— 实体集与联系间的连接关系E-R图

- 实体集student与联系SC
- 实体集course与联系SC

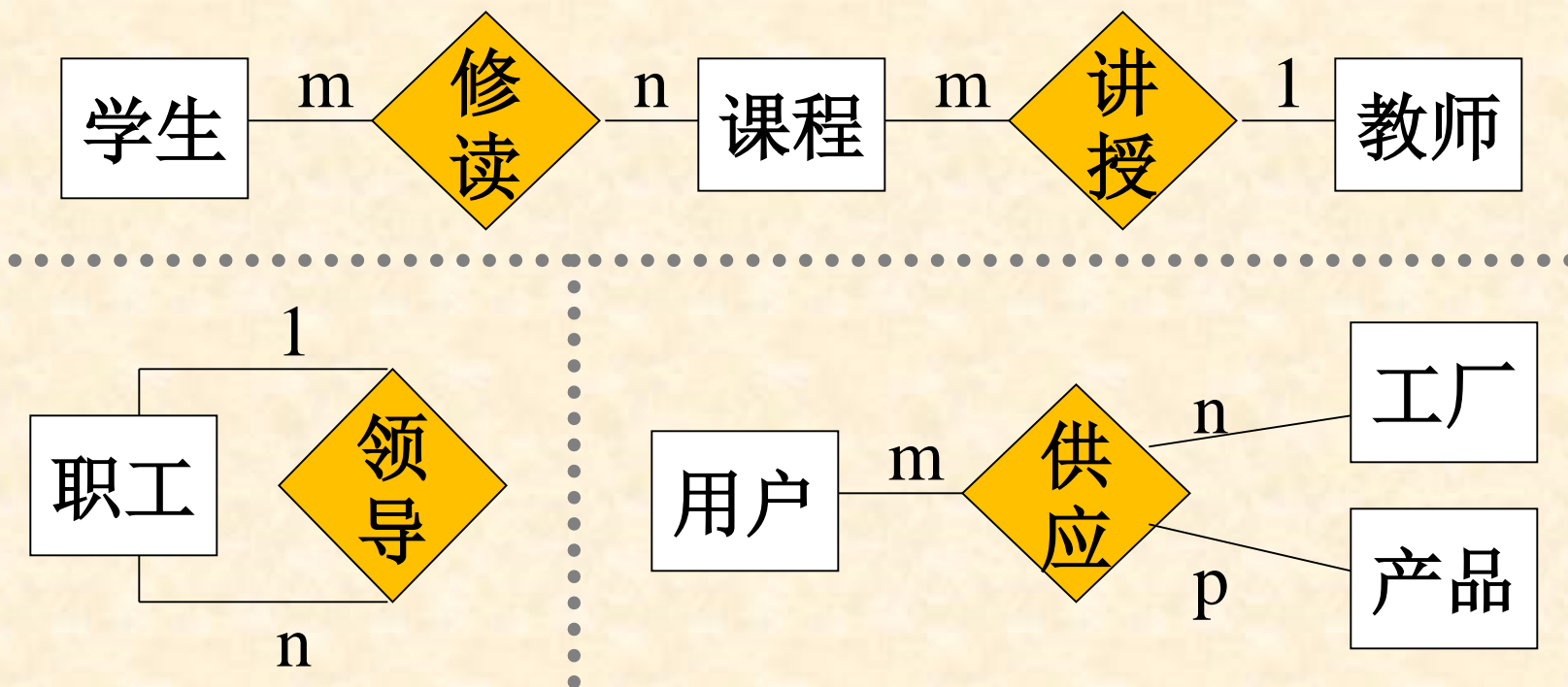


【注】为了刻画实体间的函数对应关系，必须在线段边上用1:1（一对一），1:n（一对多），n:m（多对多）等注明

2.3.1 实体-联系模型

□ 连接关系的表示 (cont.)

- 每个联系可以与一个或多个实体集相关，每个实体集也可以与一个或多个联系相关
- 例如（省略了实体集及联系上的属性）



2.3.1 实体-联系模型

□ 例2.1

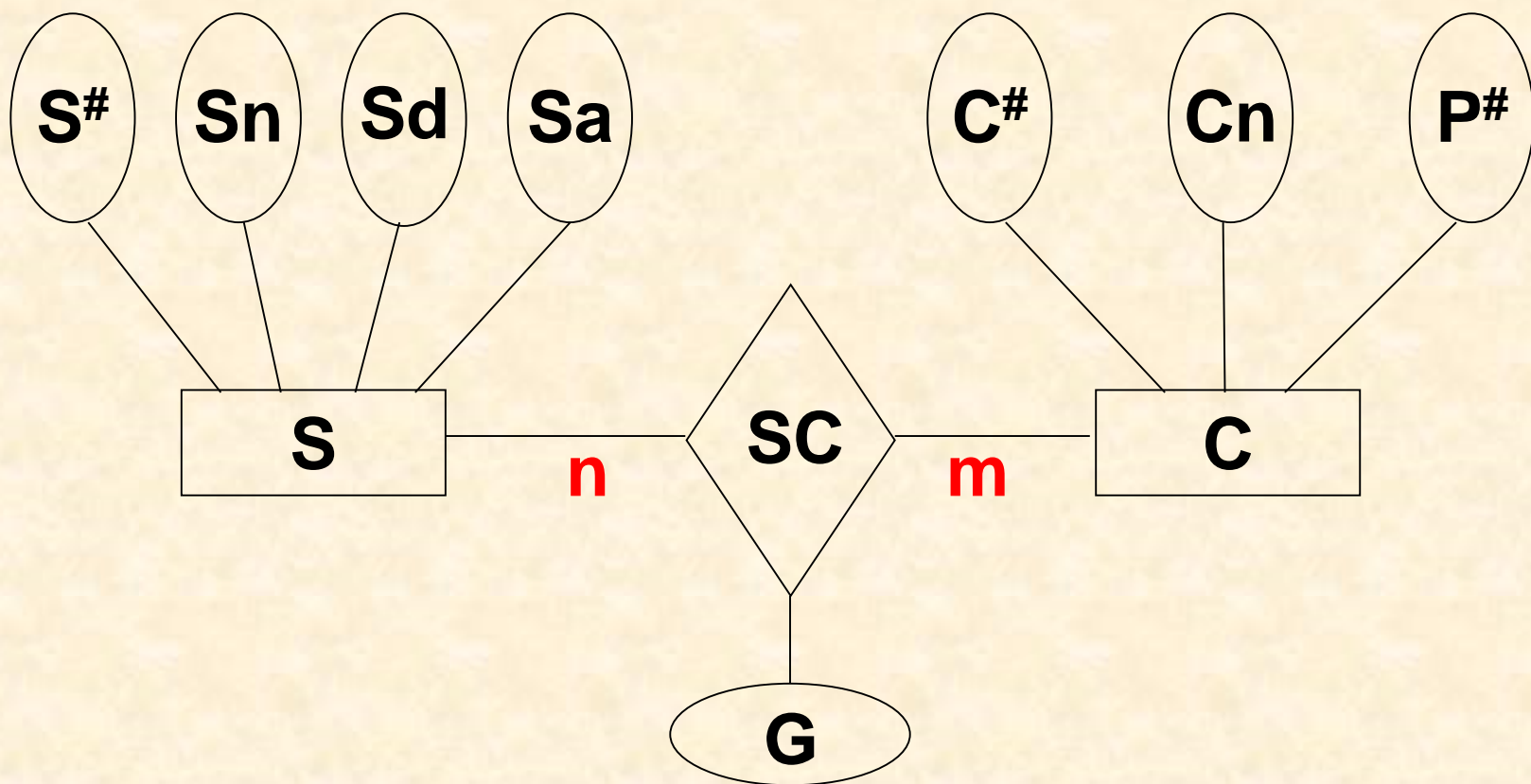


图2-12 E-R图的一个实例

2.3.1 实体-联系模型

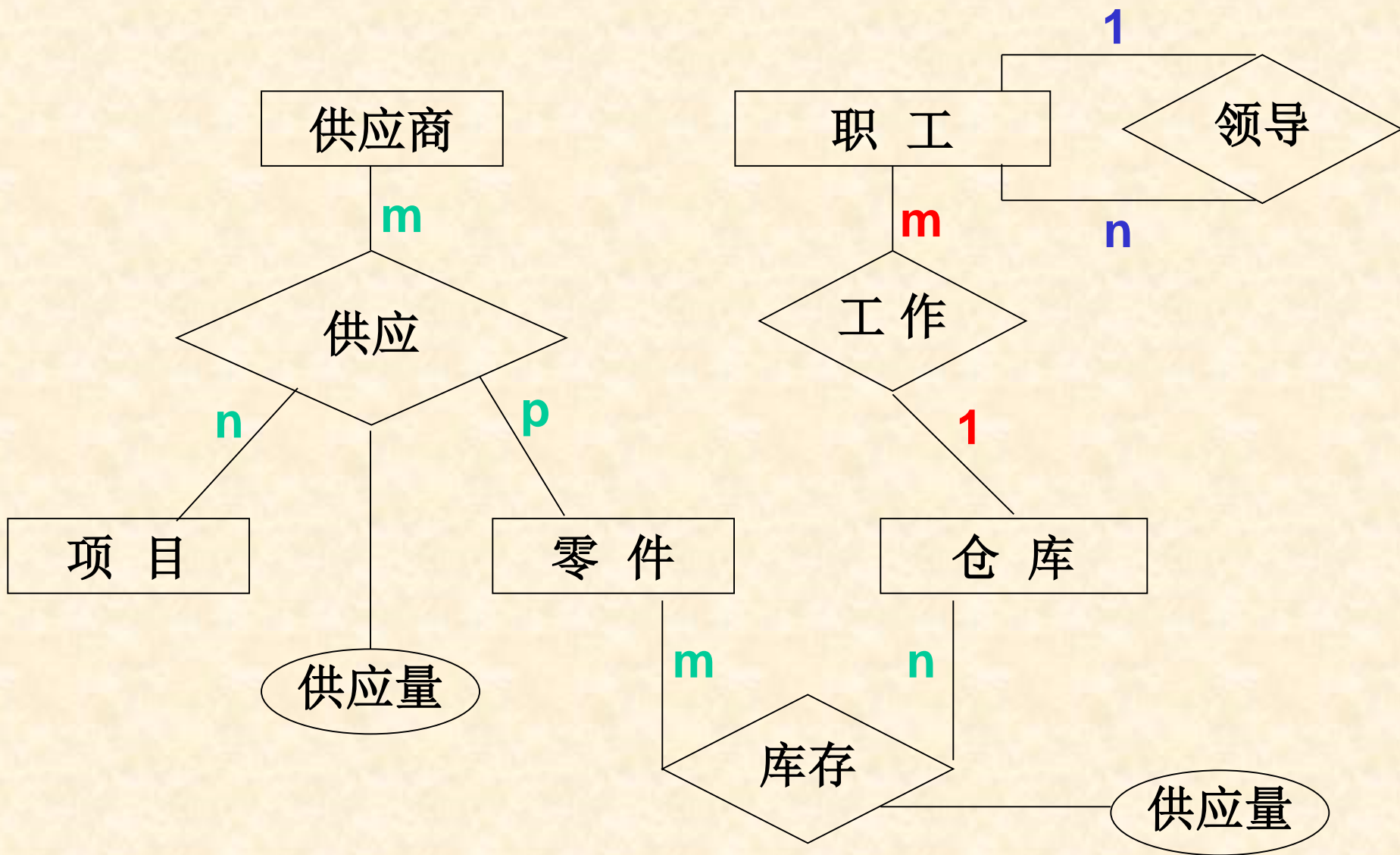
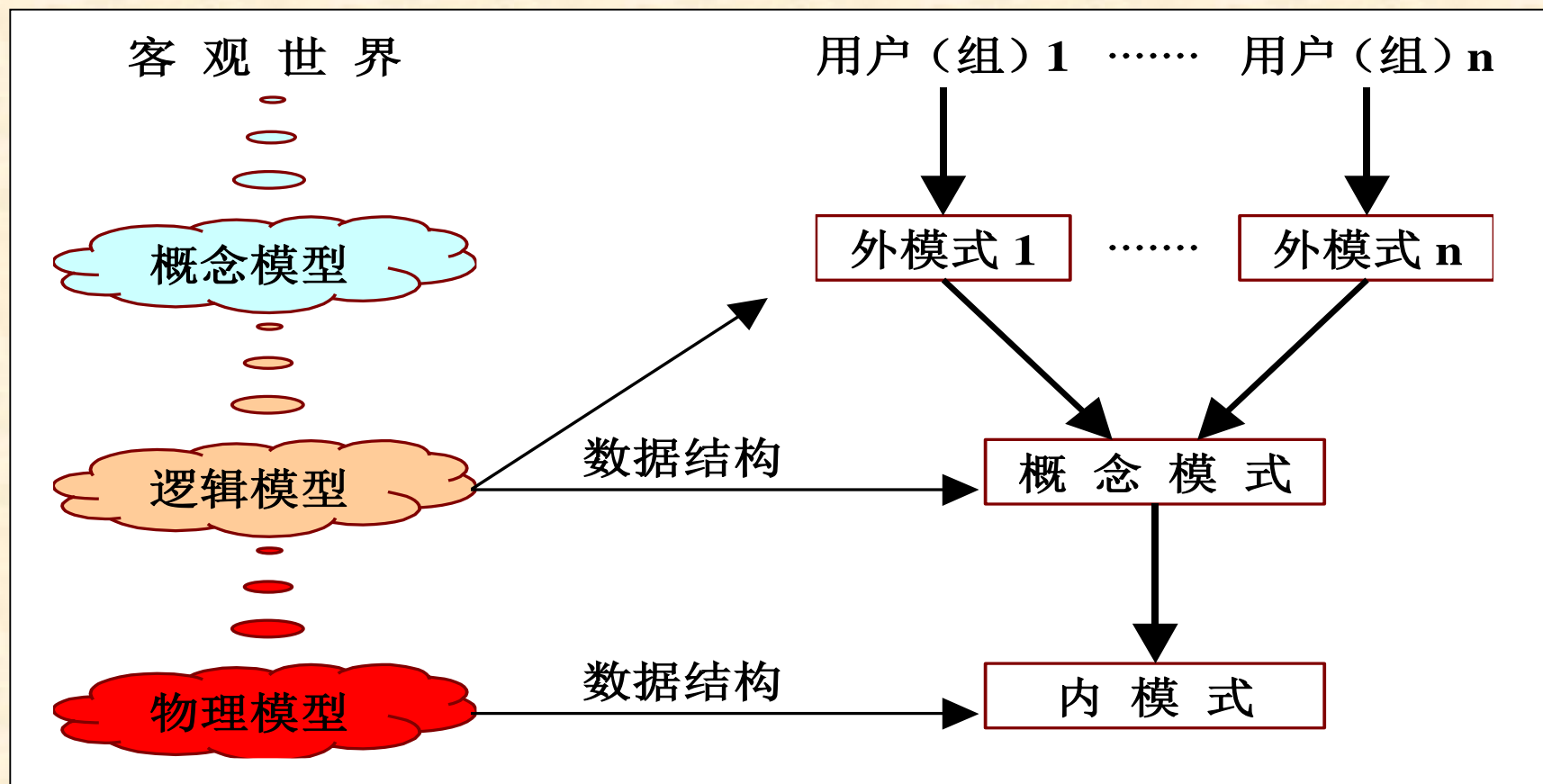


图2-13 某工厂的物资管理E-R图（省略了属性）

回顾

- 数据模型：数据结构、数据操作、数据约束
- ‘数据模型’与‘三级模式’之间的关系



- E-R模型：实体、属性、联系（函数对应关系）

2.3.1 实体-联系模型

□ E-R模型的设计

- 标识系统中的 实体、属性与 联系
- E-R图表示

□ E-R模型的设计选择

- 实体 or 属性？
- 实体 or 联系？
- 二元联系 or 三（多）元联系？
- 联系的函数对应关系？
- 属性的依附对象：实体 or 联系？

2.3.1 实体-联系模型

□ 实体 or 属性 ?

- 实体: 需要进一步多方面的描述信息
- 属性: 单一的描述值（非结构化的单值信息）

□ 实体 or 联系 ?

- 实体: 可以独立存在的持久对象
- 联系: 因为某种需要或某件事情的发生而产生的信息，通常与现实世界中的多个对象有关

2.3.1 实体-联系模型

□二元联系 or 三（多）元联系？

- 基于一个联系的语义描述需要，以及因此而涉及到的实体的个数
- 在三(多)元联系中，在下述情况下也可以考虑采用若干个二元联系来实现
 - 用户只需要使用它们之间的两两联系
 - 不会出现二义性（歧义性）

□联系的函数对应关系？

- 基于系统的语义约束来定义

2.3.1 实体-联系模型

□ 属性的依附对象：实体 or 联系？

➤ 实体(集)中的属性

- 是该实体的内在特征，不会因为某些联系的出现而产生改变或消亡

➤ 联系上的属性

- 用于描述因联系的发生而需要记录、存储下来的信息
- 其值会随着联系的产生而出现，也会随着联系的消亡而消亡

模型设计实例

□ 图书借阅系统: Case One

□ 篮球联赛系统: Case Two

第2章 数据模型

2.1 数据模型的基本概念

2.2 数据模型的四个世界

2.3 概念世界与概念模型

2.3.1 E-R模型

2.3.2 EE-R模型

2.3.3 面向对象模型

2.3.4 谓词模型

2.4 信息世界与逻辑模型

2.5 计算机世界与物理模型

2.3.2 扩充E-R模型

□ E-R模型在表示概念世界中使用较为普遍，但其在表示能力上尚有欠缺，如复杂的语义表达能力。因此很多人对E-R模型进行了扩充，构成了扩充的实体-联系模型，简称为EE-R模型

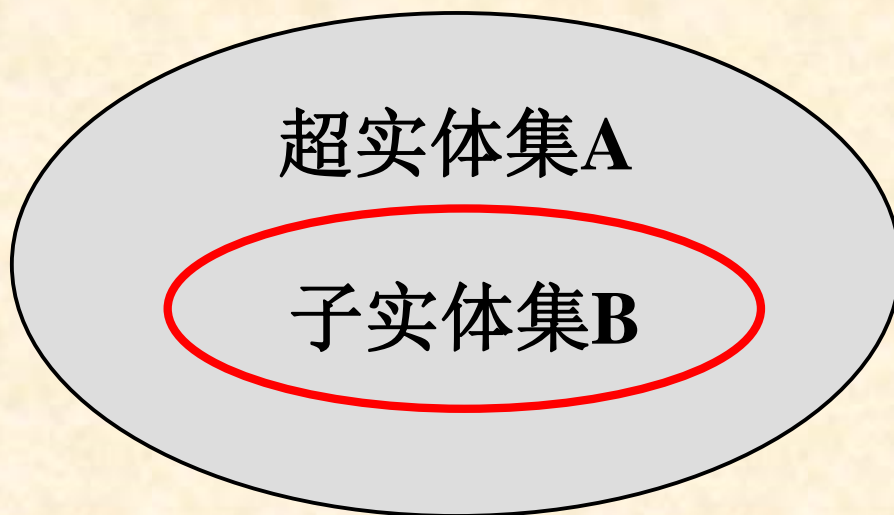
□ EE-R模型

- 扩充的E-R模型版本有很多，但其对E-R模型的扩充成分主要是“IS-A联系”
- EE-R模型也可以用一些基本的图形符号来表示，我们称其为EE-R图

2.3.2 扩充E-R模型

□ IS-A联系

- 如果实体集B是实体集A的一个子集，且具有比实体集A更多的属性，则我们称在实体集A与实体集B之间存在着一种特殊的‘IS-A联系’。其中：
 - 实体集A被称为实体集B的 超(实体)集
 - 实体集B被称为实体集A的 子(实体)集



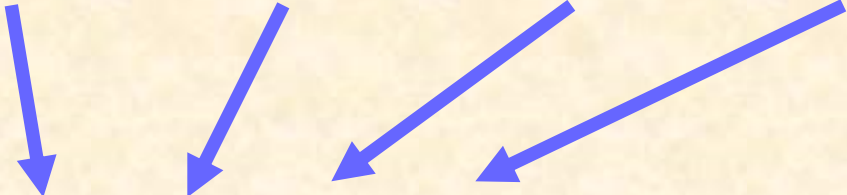
□ 子集B可以通过IS-A联系继承超集A中的所有属性

2.3.2 扩充E-R模型

【例2.3和例2.4】 ‘学生’ 与 ‘研究生’

Student(学号S[#], 姓名Sn, 系别Sd, 年龄Sa)

G_student(S[#], Sn, Sd, Sa, 导师姓名, 研究方向)



- ‘研究生’是‘学生’的一个子集，每个研究生同时也是一个学生，也拥有学生所具有的特性（属性）
- 因此，我们在它们之间建立一种特殊的IS-A联系，使得‘研究生’实体集能够从‘学生’实体集那里继承得到它们的共性，而将注意力集中于‘研究生’自己的特性定义上

2.3.2 扩充E-R模型

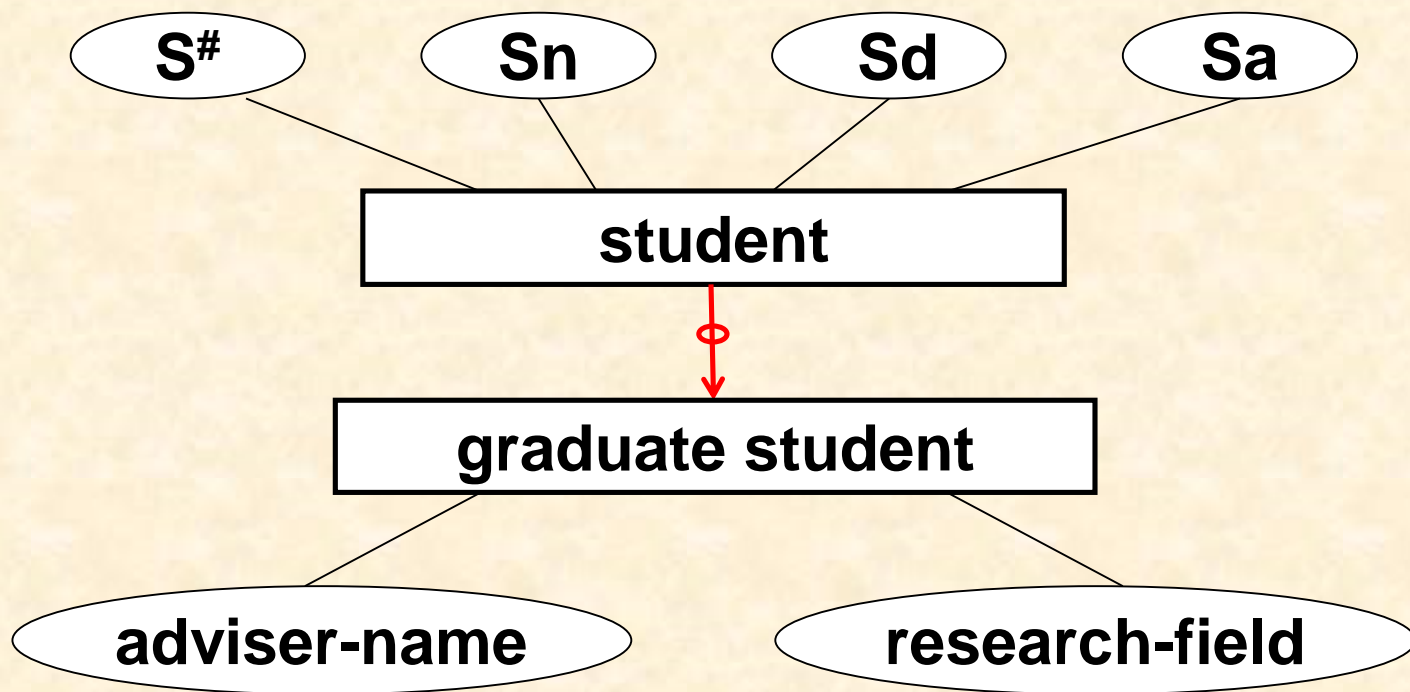
□ EE-R图

- 实体集、属性以及联系的图形表示方法与E-R图一样，这里只需要增加扩充部分的图形表示

2.3.2 扩充E-R模型

□ IS-A联系的表示方法

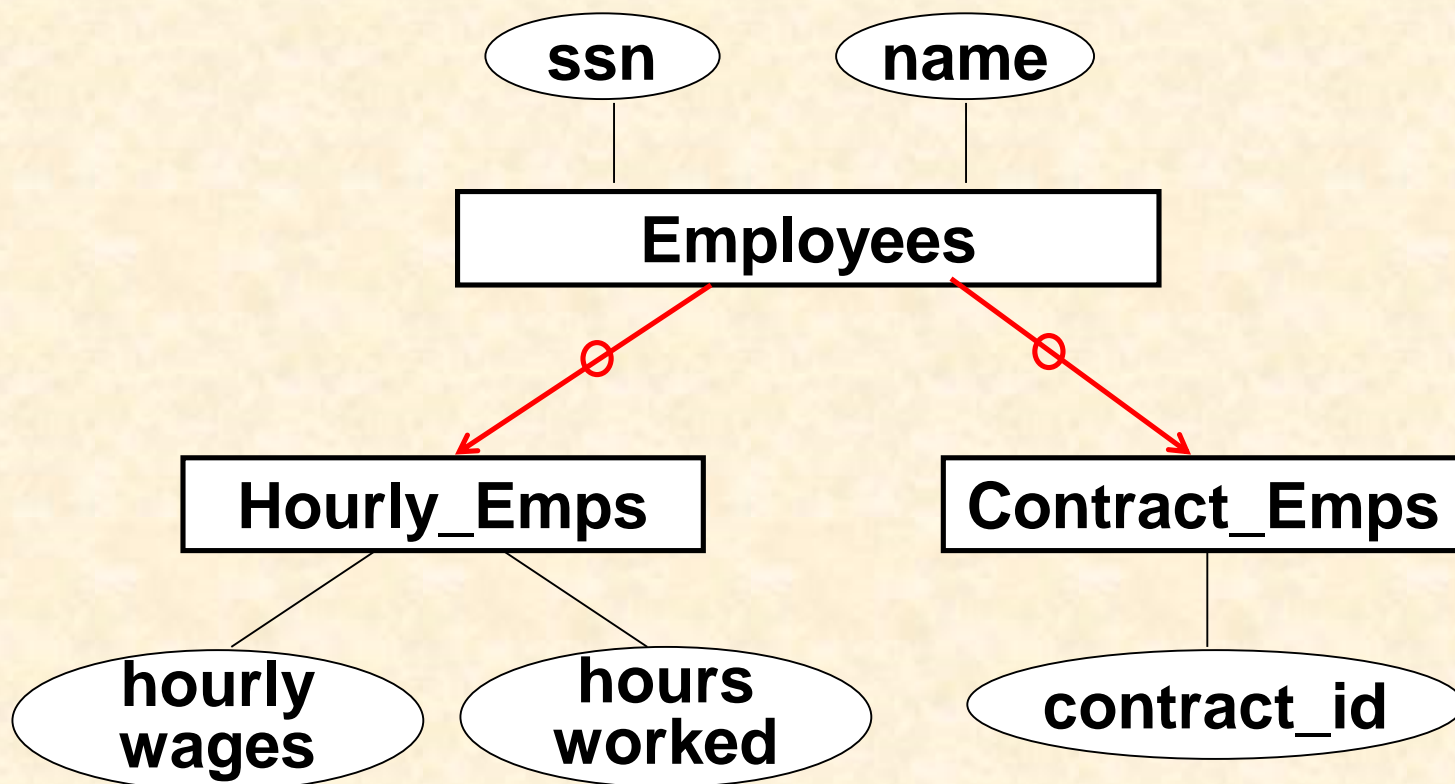
【例2.4】图2.14



2.3.2 扩充E-R模型

□ IS-A 联系的表示方法

【例】多个子实体集的例子



2.3.2 扩充E-R模型

□除了上述的扩充成分之外，还有一种比较有用的扩充成分被称为‘弱实体’

□弱实体 (Weak Entity)

➤如果一个实体A的存在需要依赖于其他实体集中的某个实体的存在，那么实体A被称为弱实体。例如：

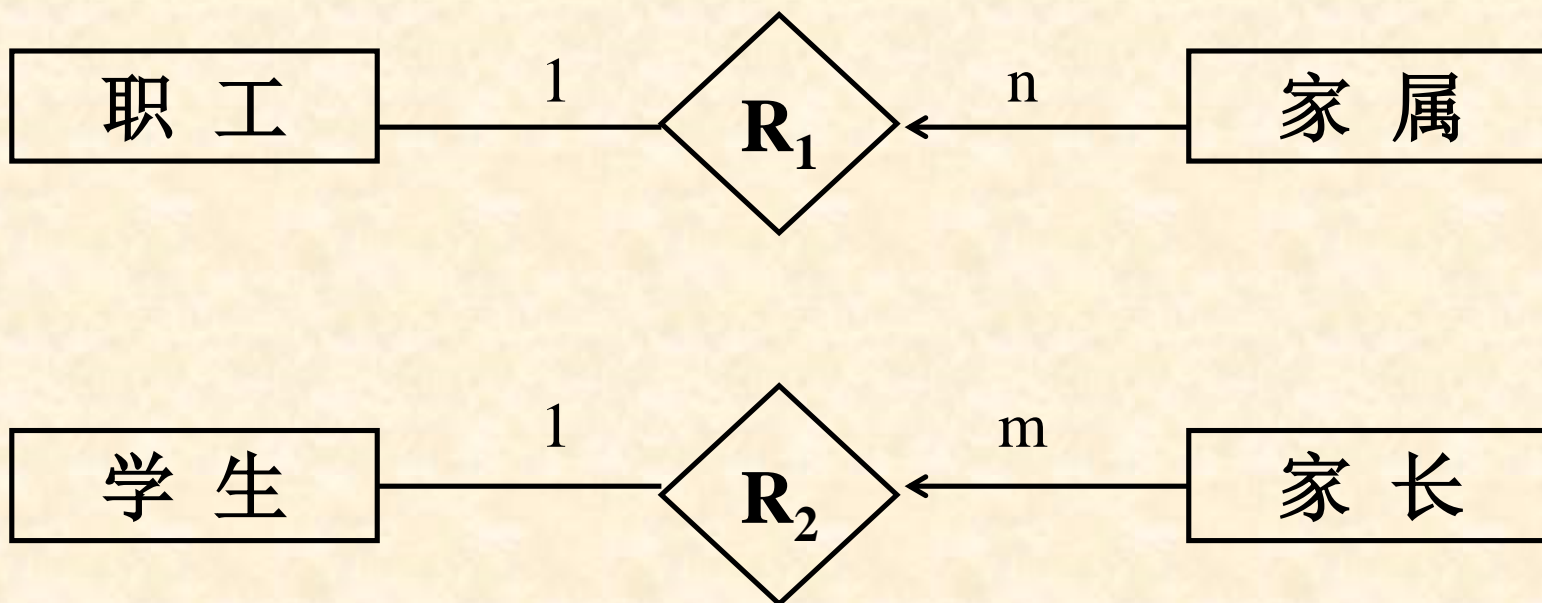
- 职工 vs 家属
- 学生 vs 家长

➤弱实体(集)与所依赖的实体(集)之间的函数对应关系应该是“多对一”的关系

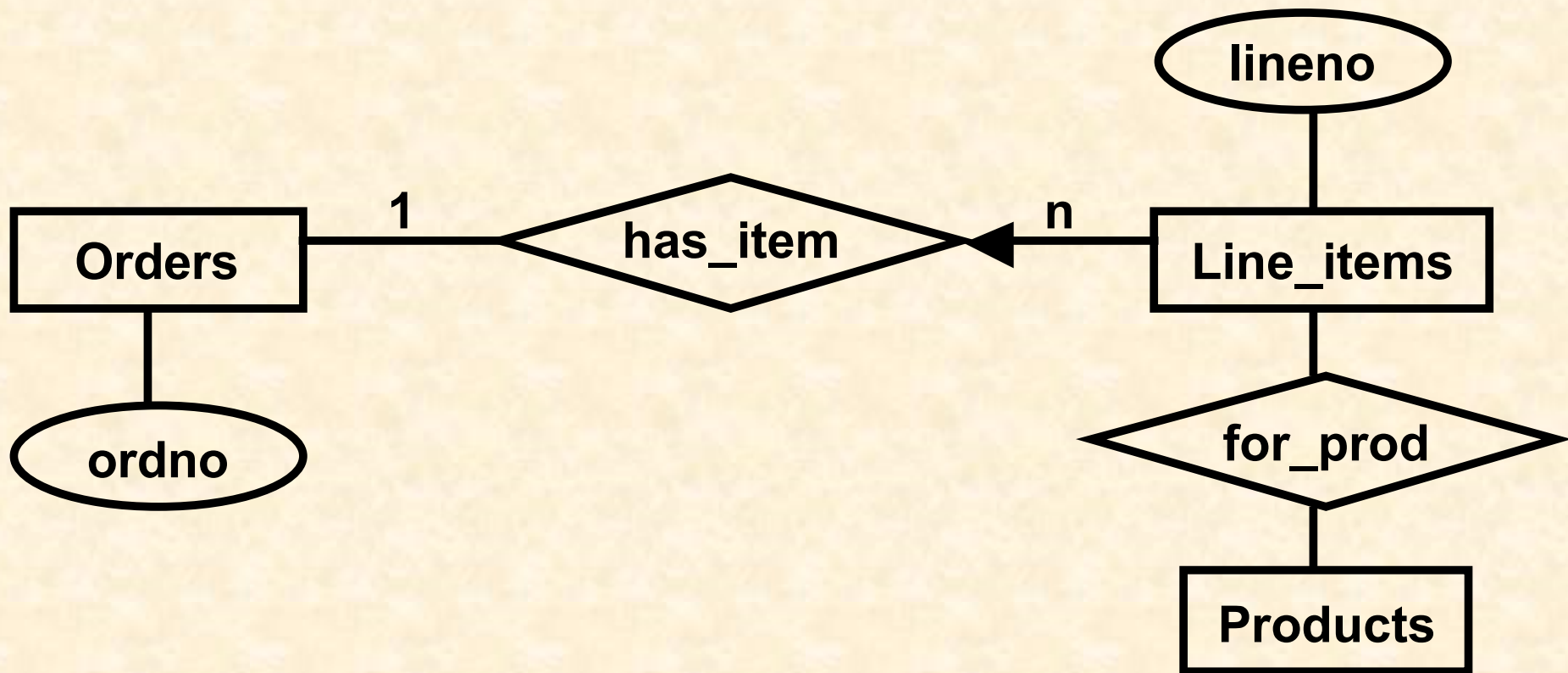
2.3.2 扩充E-R模型

□ 弱实体的表示

➤ 从弱实体到联系的有向箭头



弱实体的例子



订单、订单项与所订购的商品之间的EE-R图

第2章 数据模型

2.1 数据模型的基本概念

2.2 数据模型的四个世界

2.3 概念世界与概念模型

2.3.1 E-R模型

2.3.2 EE-R模型

2.3.3 面向对象模型

2.3.4 谓词模型

2.4 信息世界与逻辑模型

2.5 计算机世界与物理模型

2.3.3 面向对象模型

□ 面向对象模型是将面向对象技术引入到数据库领域，借鉴面向对象的设计方法而发展起来的一种新的数据模型。该模型采用了面向对象方法中的基本概念和方法来构造数据库的数据模型，从而大大提升了模型的表达能力，特别是在表示非传统的、复杂数据关系方面具有极强的表达力

□ 面向对象模型（简称OO模型）的基本内容

➤ 对象

— 对象标识符，对象的组成，对象的特点

➤ 类

— IS-A, IS-PART-OF

➤ 消息

2.3.3 面向对象模型

□对象 (Object)

- 客观世界中能够相互区别开来的事物
- ‘对象’是OO模型中的最基本的概念，相当于E-R模型中的‘实体’，但它们两者之间又存在着很大的区别

2.3.3 面向对象模型

□ ‘对象’ 与 ‘实体’ 的区别:

- 在E-R模型中，是通过一个实体所固有的内在属性的属性值来描述一个实体的特征的，侧重于实体结构的描述。而在OO模型中，不仅需要描述对象的属性组成，还可以描述该对象所具有的行为特征
- 在E-R模型中，是通过一个或一组属性上的取值来区分不同的实体的。而在OO模型中，每个对象都有一个系统定义的对象标识符，可以且只能通过对象标识符来区分不同的对象

2.3.3 面向对象模型

□对象的组成

➤对象标识符 (Object Identifier, 简称OID)

—每个对象均具有的一个能相互区别的名字

➤对象的静态特性

—对象中的属性，用于反映对象的状态与特性，
是每个对象固有的静态表示

➤对象的动态特性

—可以施加在该对象上的方法 (method)

—方法又称为‘操作’ (operation)

2.3.3 面向对象模型

□ 方法

- 是可以作用在对象上的一段程序
- 方法用于反映对象的行为特征，是对象的固有动态行为的表示，可用于审视并改变对象的内部状态（属性值）

【例】‘学生’对象的组成

□OID

- 这是一个系统定义的属性，也是由系统自动为每个对象进行赋值的。OID的取值取决于每个面向对象系统的具体实现

□属性

- 学号，姓名，年级，系别，出生日期，所修课程

□方法

- 修改学生的年级值： $\text{年级} + 1$
- 计算一个学生的年龄
- 统计一个学生的总学分数
- 统计一个学生的平均学分数

2.3.3 面向对象模型

□ 对象的特点

- 1) 对象的封装 (encapsulate)
- 2) 对象标识符的独立性
- 3) 对象属性值的多值性

2.3.3 面向对象模型

1) 对象的封装 (encapsulate)

- 对象的属性与方法被封装在一起构成一个整体
- 对象的封装特性将一个对象划分为两个部分
 - 对象的内部表示
 - 对象中的属性组成与方法实现
 - 对象的外部表示
 - 方法的调用接口，亦称 ‘对象界面’
- 对象封装的优点
 - 可以屏蔽对象的内部状态及其实现细节，有利于数据的保护
 - 有利于对象代码及数据结构的维护
 - 提高对象的可靠性与可重用性

2.3.3 面向对象模型

2) 对象标识符的独立性

- 对象标识符独立于对象的属性，它是由系统定义并赋值的，可以唯一标识一个对象
 - 完全不同于E-R模型中用户定义的关键字的概念
- 不同的面向对象系统可以有不同的实现方法，但它们都必须保证对象标识符的如下特性：
 - 唯一性
 - 持久性
 - 不可重用性

3) 对象属性值的多值性

对象属性的取值可以是一个单值，也可以是一个值的集合，甚至是另外一个对象

➤ 单值

- 简单值：数值型、字符型（如年级、姓名、学号等）
- 复杂值：自定义数据类型（如日期、时间等）

➤ 集合值

- 如：电话号码
- 也可以有各种类型的集合：Set、Bag、List、Array

➤ 对象值

- 如：系别
- 也可以是对象值的集合，如：所修课程

2.3.3 面向对象模型

□ 类 (Class)

- 具有相同属性、方法的对象集合
 - 可以通过对类的定义来描述类中对象的静态特性和动态特性
 - 实例 (Instance)
 - 类也可以被抽象成一个对象，我们称其为‘**类对象**’
 - 为了将‘类对象’与类中的对象区别开来，我们将类中的对象又称为实例
 - 元类 (meta class)
 - 由所有‘类对象’所构成的对象集合

2.3.3 面向对象模型

□ 类与类之间的关系

- 在面向对象方法中，类与类之间可以建立两种关系：‘IS-A’和‘IS-PART-OF’，并且可以通过这两种关系来描述实现世界中的复杂对象及其相互之间的复杂关系

—IS-A关系

- 子类与超类之间的继承关系

—IS-PART-OF关系

- 类的聚合与分解关系

2.3.3 面向对象模型

□ 继承 (inheritance)

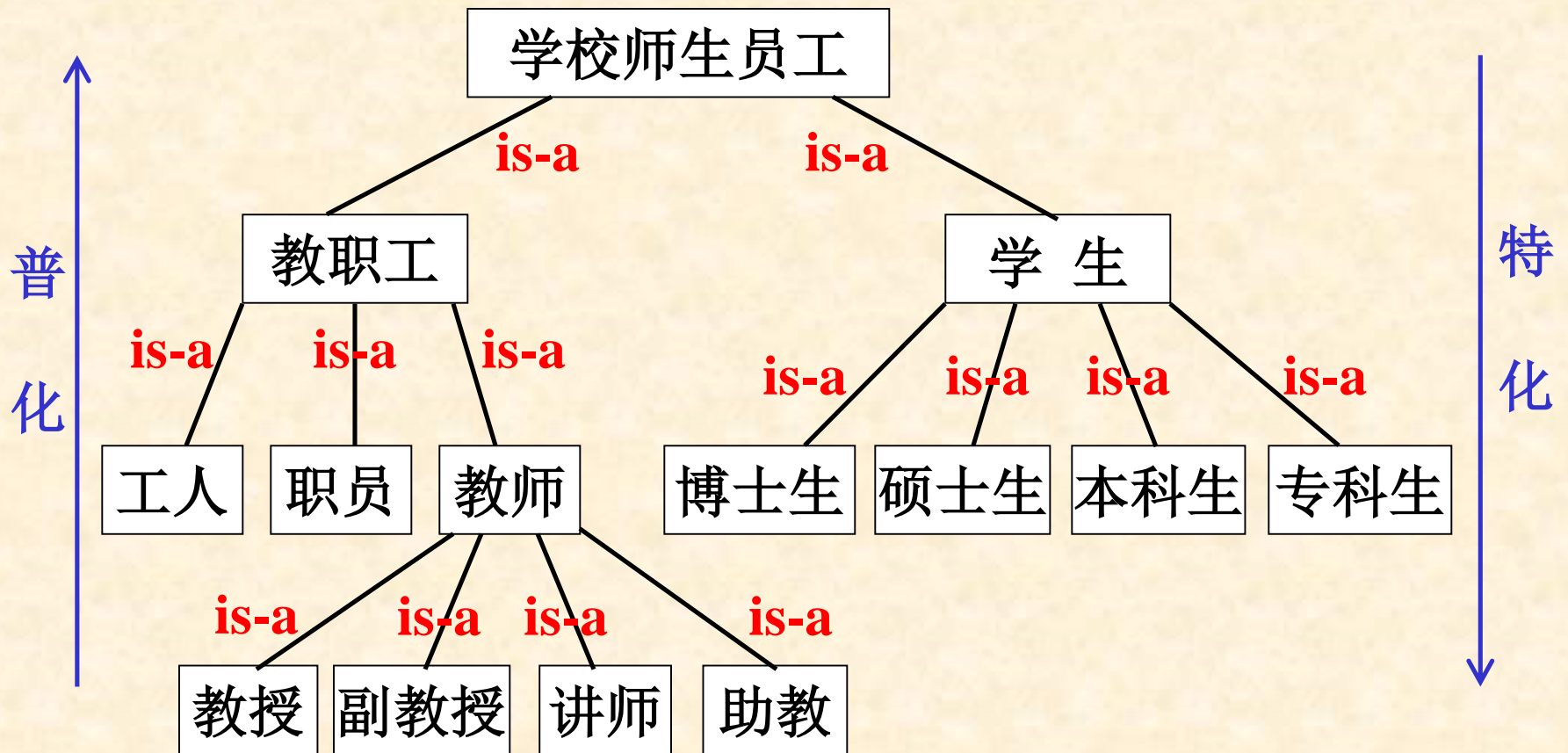
- 一个类的定义和实现建立在其它类的基础之上，并共享其它类的定义和实现
- 两个类之间的继承关系我们可以用一种特殊的IS-A联系来描述
- 其中被继承的类称为‘超类’，获得继承的类被称为‘子类’
- 子类与超类之间的继承关系可以构成一个单向、不循环的层次结构

2.3.3 面向对象模型

□关于继承

- ‘普化’ 与 ‘特化’
- ‘单继承’ 与 ‘多继承’
- ‘全继承’ 与 ‘部分继承’
- 继承的作用
- 继承所带来的问题

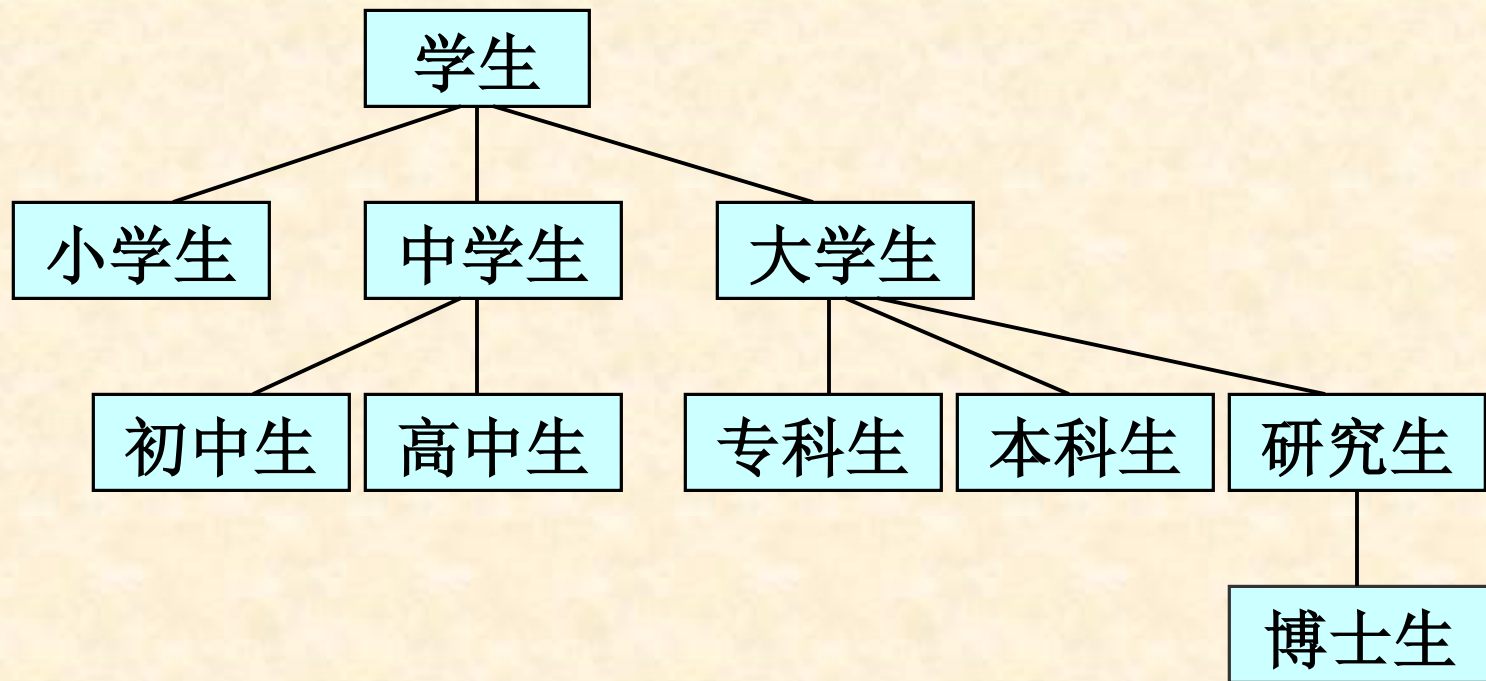
【图2-16】子类、超类示意图



- **普化**：子类到超类的抽象过程，也是一个对象集合不断合并的过程
- **特化**：从超类到子类的细化过程，是一个对对象集合不断进行分解的过程

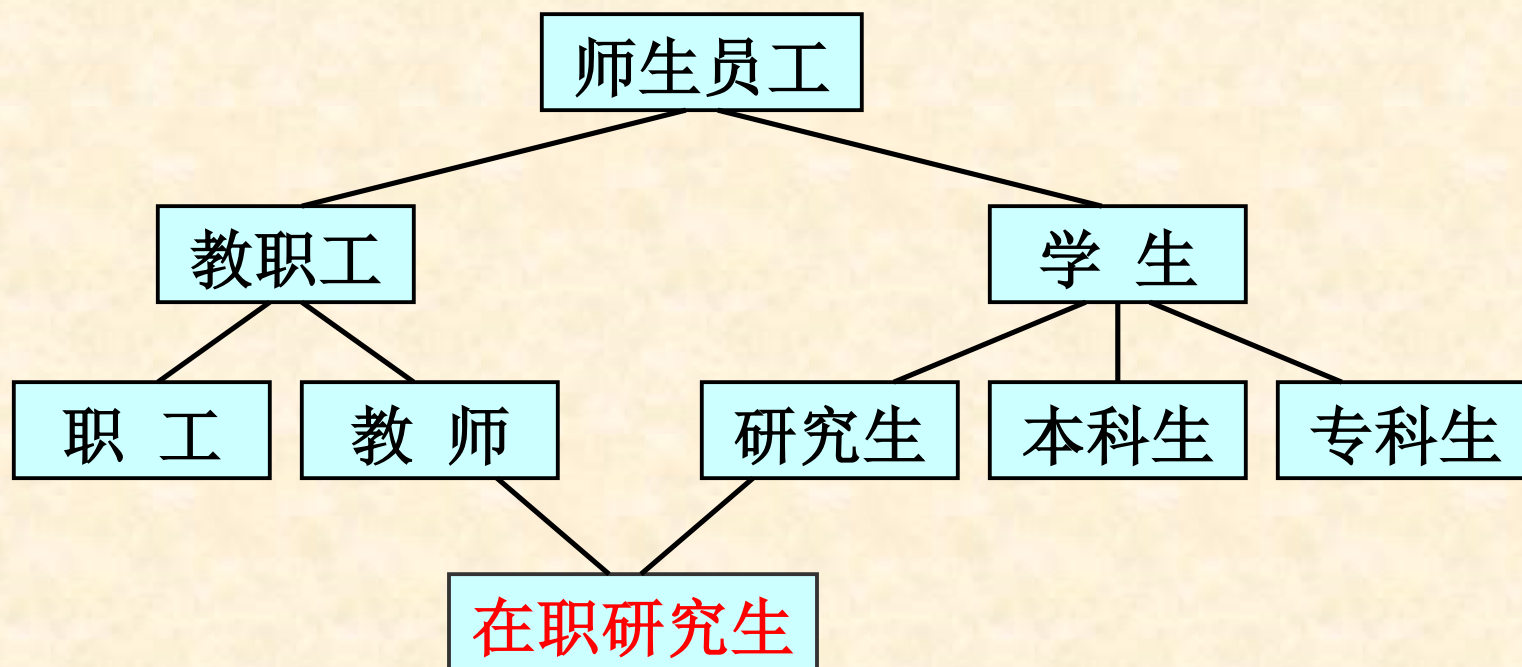
2.3.3 面向对象模型

- **单继承**：每个子类只有唯一的一个直接超类
- **多继承**：一个子类可以有多个直接超类



单继承：树状类继承层次结构

2.3.3 面向对象模型



多继承：格状类继承层次结构

2.3.3 面向对象模型

➤ 继承的作用

- 支持代码的共享与重用
- 有助于系统的扩充

➤ 在继承过程中需要解决的问题

— 重写 (overriding)

- 用于解决在继承过程中，超类与子类（或超类与超类）之间的冲突
- 冲突：方法或属性的名字相同但语义或实现不同

— 命名冲突

2.3.3 面向对象模型

□ 类的聚合与分解

➤ 类的聚合

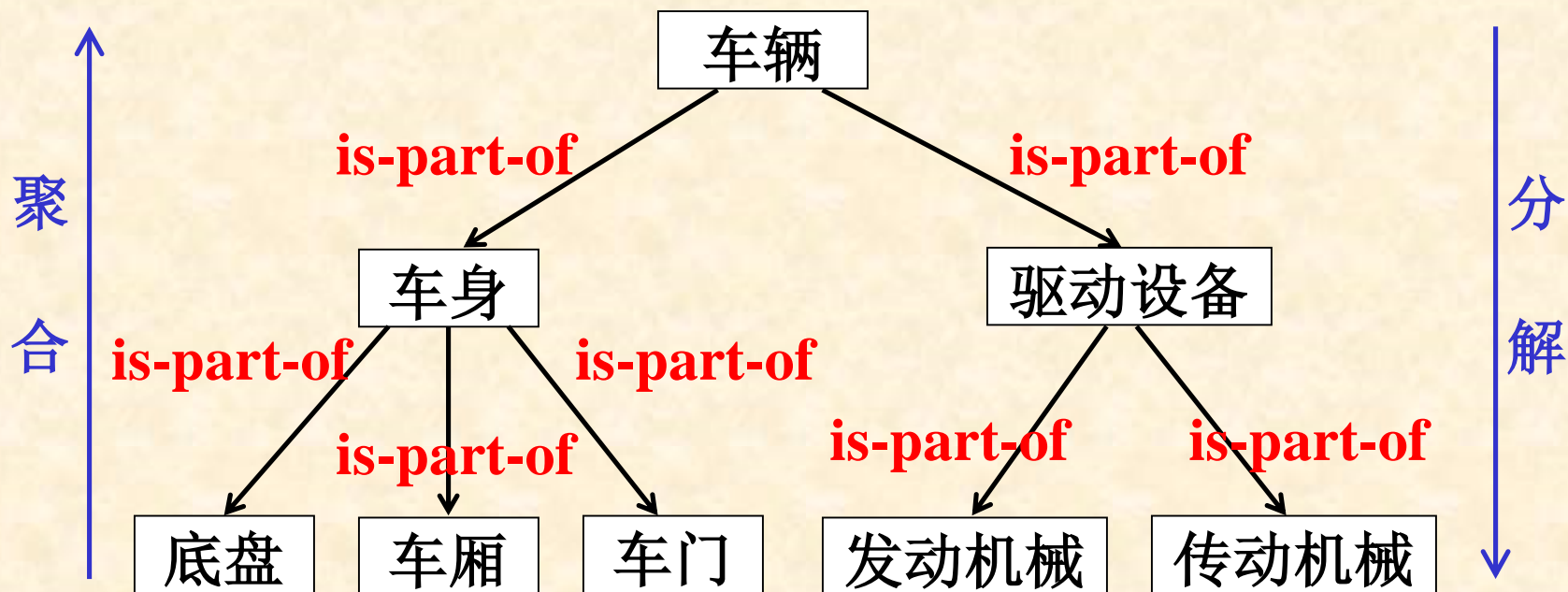
— 由若干个简单类聚合成一个复杂的类的过程

- 类与类之间的这种聚合联系构成了类与类的Is-part-of关系

➤ 类的分解

— 由复杂类分解成若干层次上的简单类的过程

【图2-17】类的聚合与分解关系

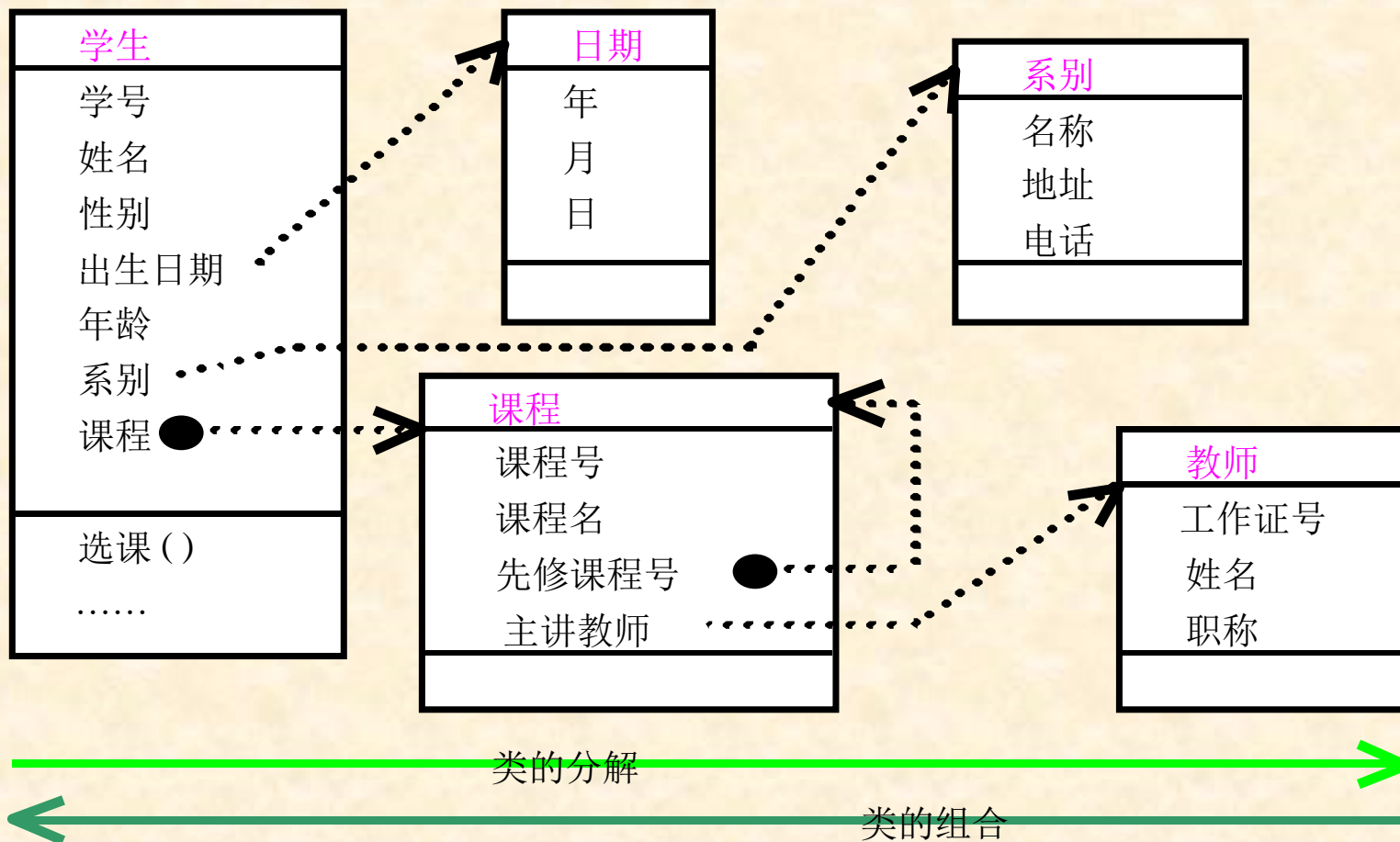


- 类的聚合与分解关系具有三种语义
 - 组成语义，嵌套语义，联系语义

- 实质上是反映了类中对象之间的组合与分解关系

2.3.3 面向对象模型

□ 【例】类的聚合与分解关系



类的聚合与分解层次结构图

2.3.3 面向对象模型

- 在面向对象方法中，IS-A与IS-PART-OF联系都具有特定的语义信息。因此，在OO模型中，我们可以采用这两种联系来构成一个类层次结构，以便于对复杂的数据关系的描述
- 客观世界中的任何联系都可以用 ‘IS-A’ 与 ‘IS-PART-OF’ 这两种方式构造出来

2.3.3 面向对象模型

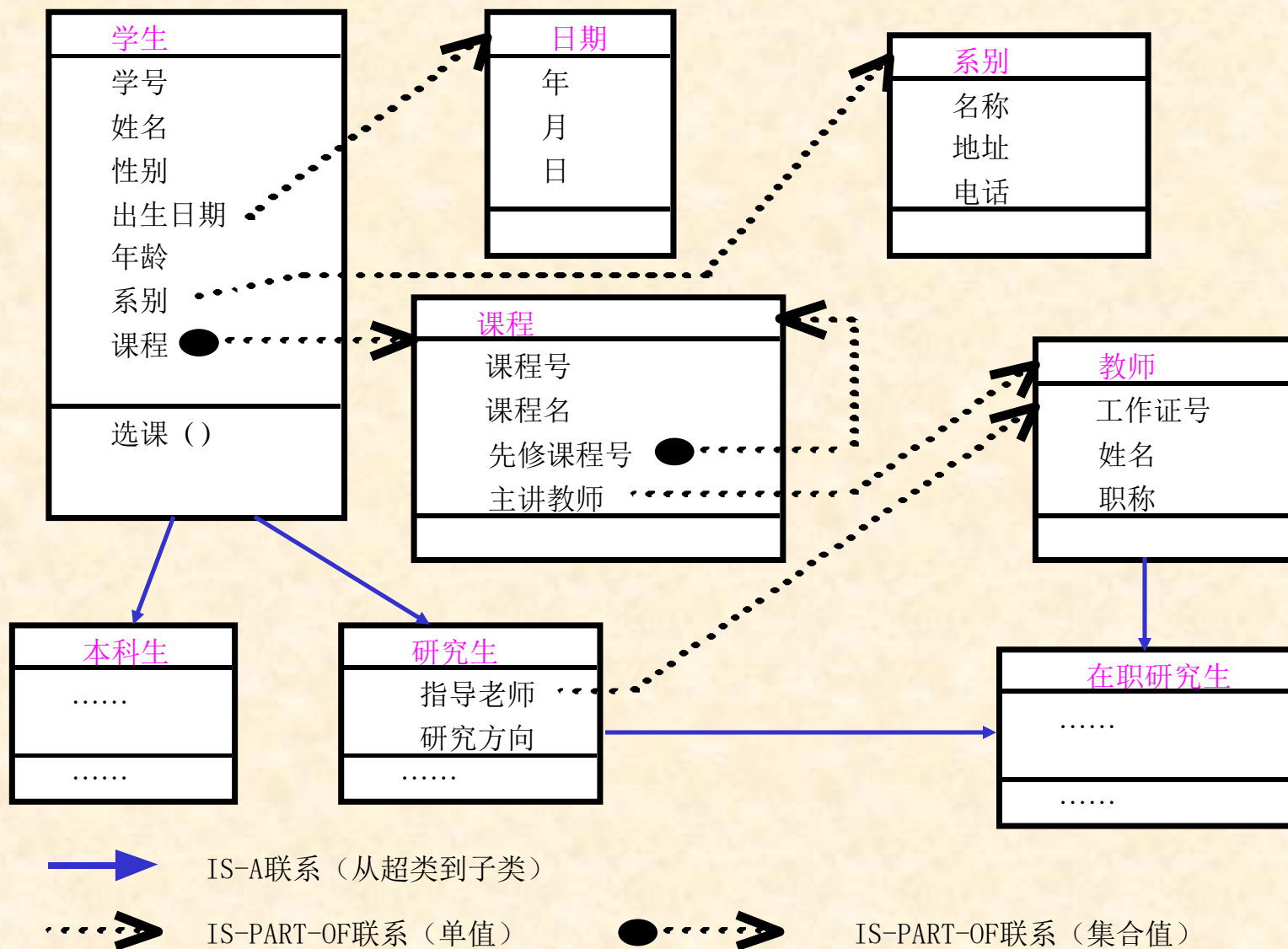


图 面向对象模型图

2.3.3 面向对象模型

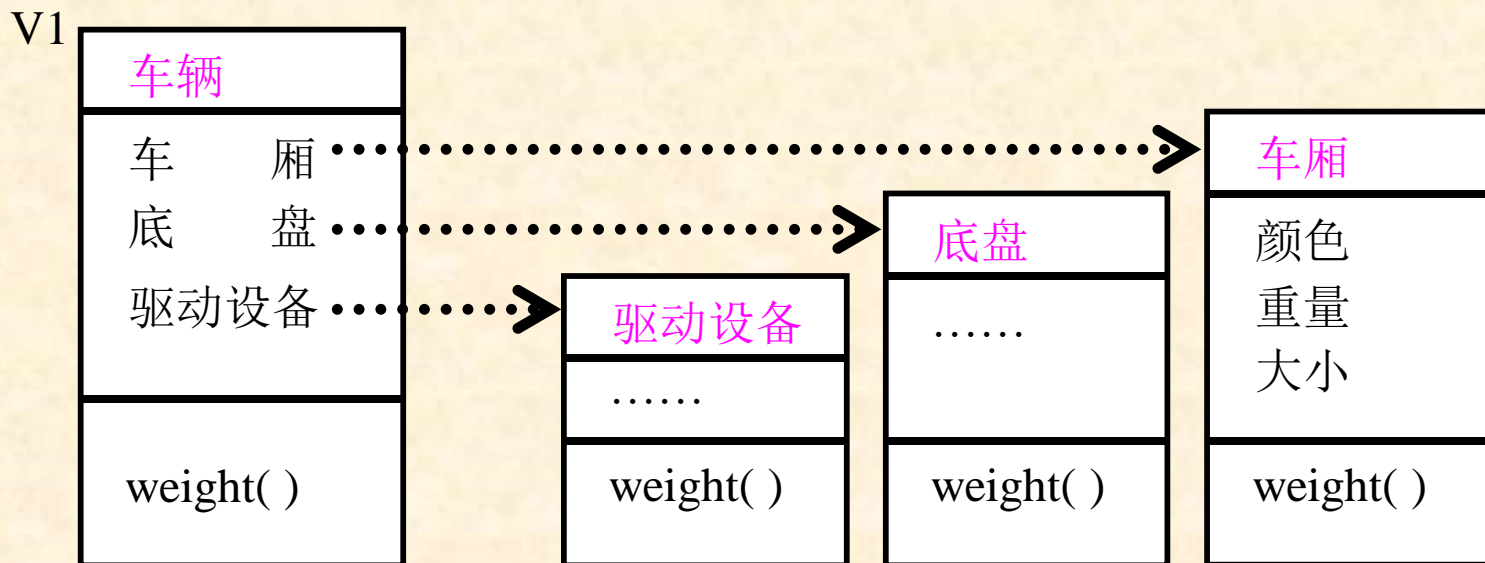
□ 消息 (message)

- 对象间的一种协作机制
- 一个对象可以通过向另一个对象发送消息来调用另一个对象中的方法，以获得其协作来共同完成某一个任务
- 消息仅作用于对象界面，再通过对象界面调用相应的方法来进一步影响与改变对象自身
- 用户对一个对象所做的操作也可以看成是一条发向该对象的消息，并通过该消息执行对象中的方法，以实现用户的操作要求或获得操作结果

2.3.3 面向对象模型

- 消息的组成: $\text{Type A} . \text{Op} (\text{O1}, \text{O2}, \dots, \text{On})$
- 接收者: 对象A
 - 操作名: Op
 - 操作参数及其返回结果的类型: O1, O2, ..., On及Type
 - 例:

```
int V1.weight( ) { return ( V1.车厢.weight( ) +  
                           V1.底盘.weight( ) +  
                           V1.驱动设备.weight( ) ); }
```



2.3.3 面向对象模型

□ 消息与方法的比较

- 方法是对象的内部操作，它包括方法的外部调用接口和内部实现细节两个部分
- 消息则是一个跨对象的对象间的操作

□ 由于消息本身并不具有某些特定的语义信息，因此在OO模型中我们主要考虑 IS-A 和 IS-PART-OF 两种联系，以它们为主要手段来构造面向对象的数据模型

2.3.3 面向对象模型

□ C++与OODB的区别

- 1) C++中没有OID的概念
- 2) C++主要讨论（管理）对象，而OODB则主要讨论（管理）类
- 3) C++主要关心类与类之间的继承关系，而OODB也很重视类的合成关系
- 4) C++中的类与对象都是挥发性的，而OODB中的类与对象则均是持久性的
- 5) OODB中的持久类需要长期保存，并具有共享性，因此，OODB非常重视类（对象）的安全性、完整性、并发控制和故障恢复能力，而C++中则没有这些功能

统一建模语言¹⁹⁹¹

□ UML (统一建模语言, Unified Modeling Language)

最初是开发用来在面向对象风格中作为描述软件设计的一种图形化的标注

➤ 现在已经做了一些扩展和更改, 作为一种流行的数据库设计描述的标注

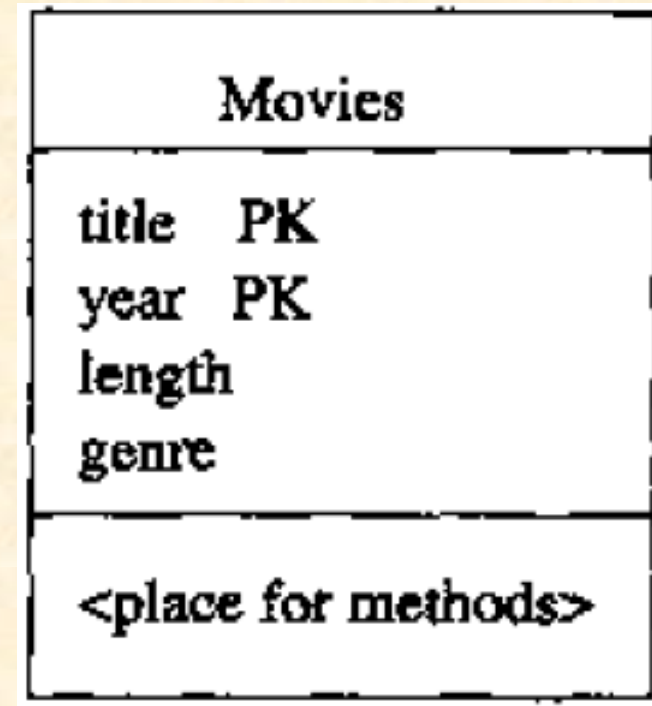
UML	E/R模型
类	实体集
关联	二元联系
关联类	联系的属性
子类	isa层次
聚集	多对一联系
组合	具有引用完整性的多对一联系

□ UML中的类与E/R模型中的实体集类似

- 类的标注却是非常不同

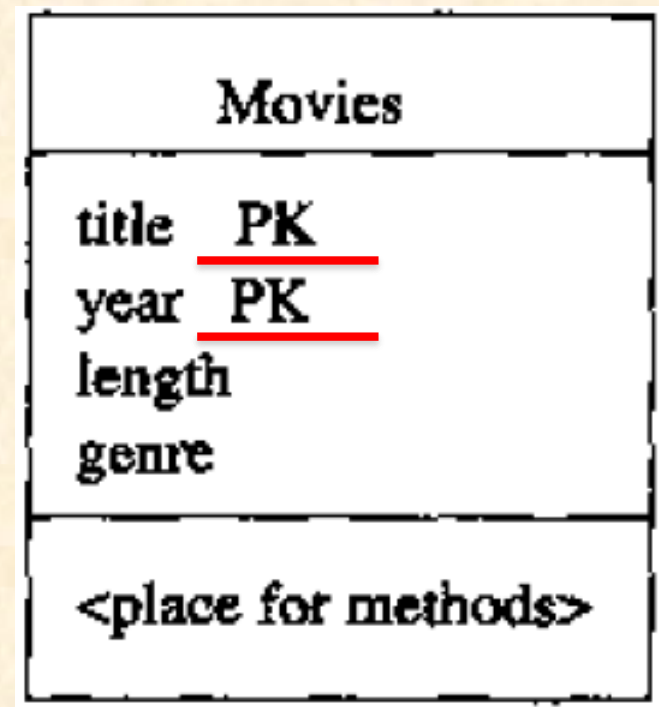
□ 一个类框分为三个部分

- 顶部是类的名字
- 中间是它的属性
- 底部是方法
 - E/R模型和关系模式都不提供方法。然而这是一个重要的概念，通常出现在现代的关系系统中，称作“对象关系” DBMS



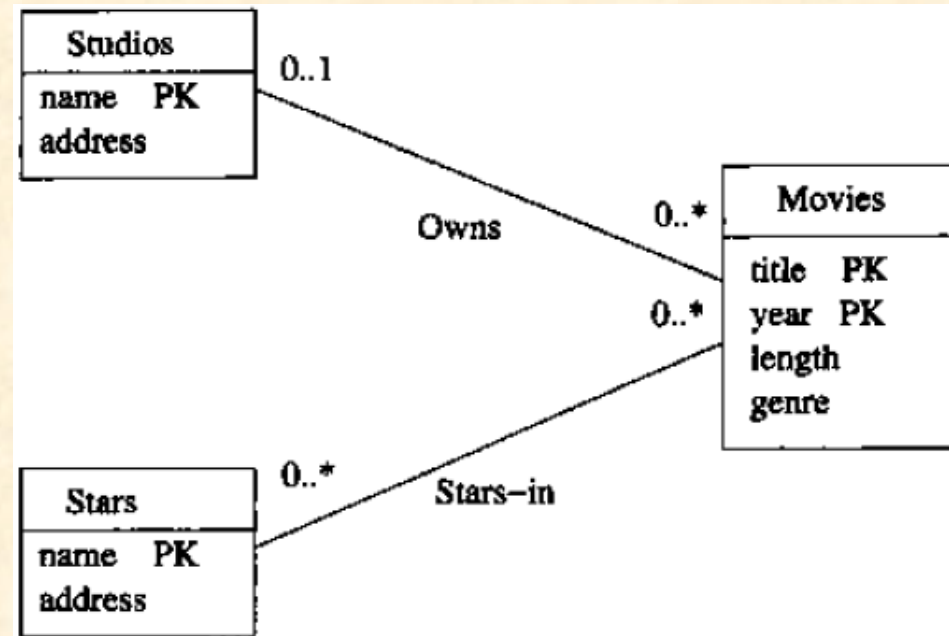
UML类的键¹⁰³

- 像实体集那样，可以给UML类指定一个键
- 其方法是，在每个键属性的后面用字母PK标明，表示“主键”



□ 在类之间的二元联系称为**关联** (association)

- **UML**中没有相似的多路联系，一个多路联系却可以拆成几个二元的联系
- 关联是对象对的集合，每个对象来自它连接的类
- 两个类之间构建一个**UML**的关联只需简单地通过在这两个类之间划一条线，并给这条线一个名字
- 每个与其他关联的类在连接的对象的数量上有一定约束。这种约束通过在每个连接线的末端用一个***m..n***标签形式来表明
 - 这一端至少有***m***个对象，至多有***n***个对象与另外一端的对象连接
 - 用*代替***n***，表示“无限”
 - 单独的*表示区间***0..****



□ 一个关联的两端可以连接同一个类，这样的关联被称为 **自关联**

- 为了区分一个类在自关联中表现的不同角色，分别给这个关联的两端一个名字

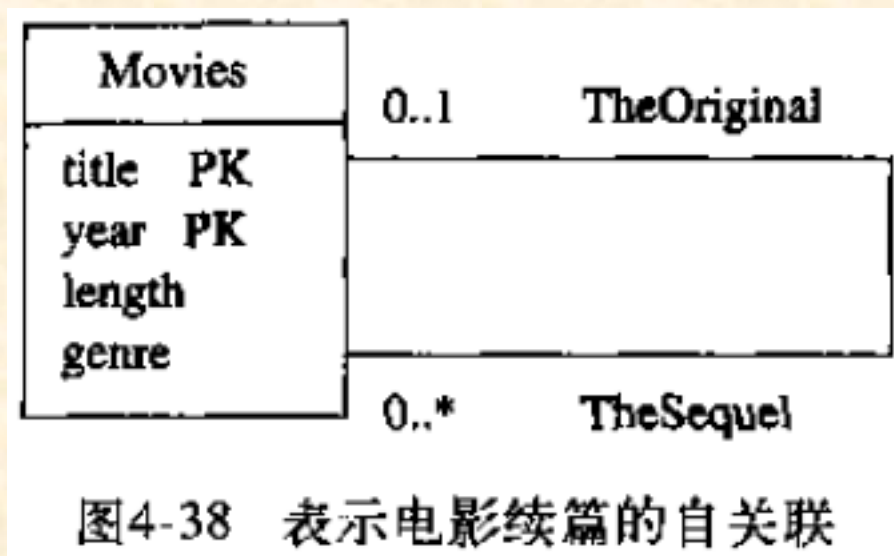
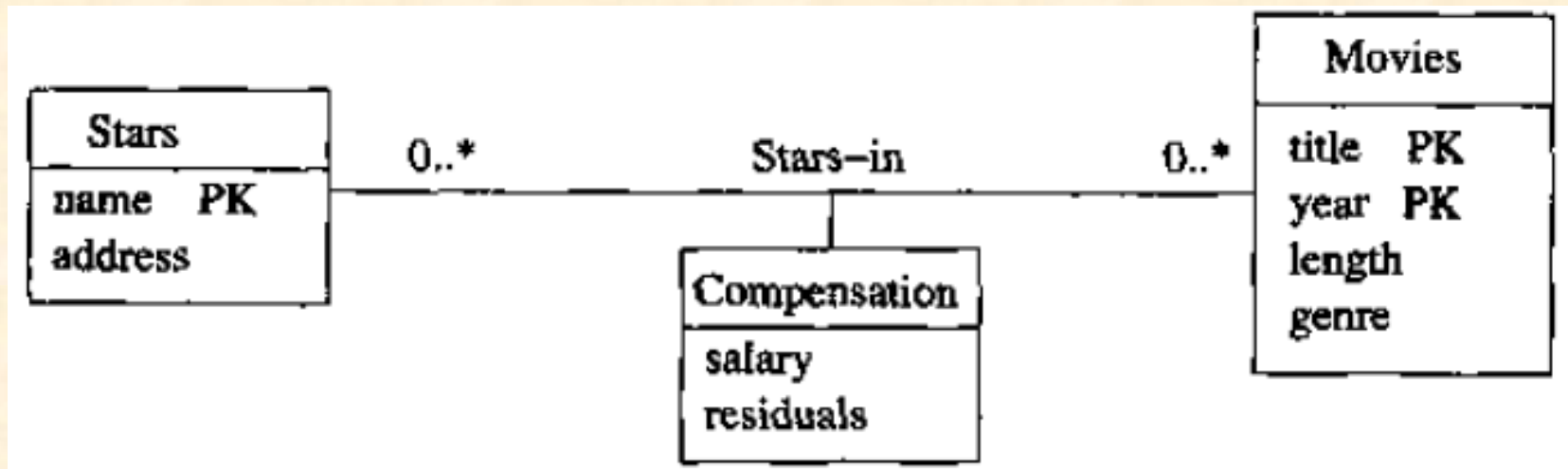


图4-38 表示电影续篇的自关联

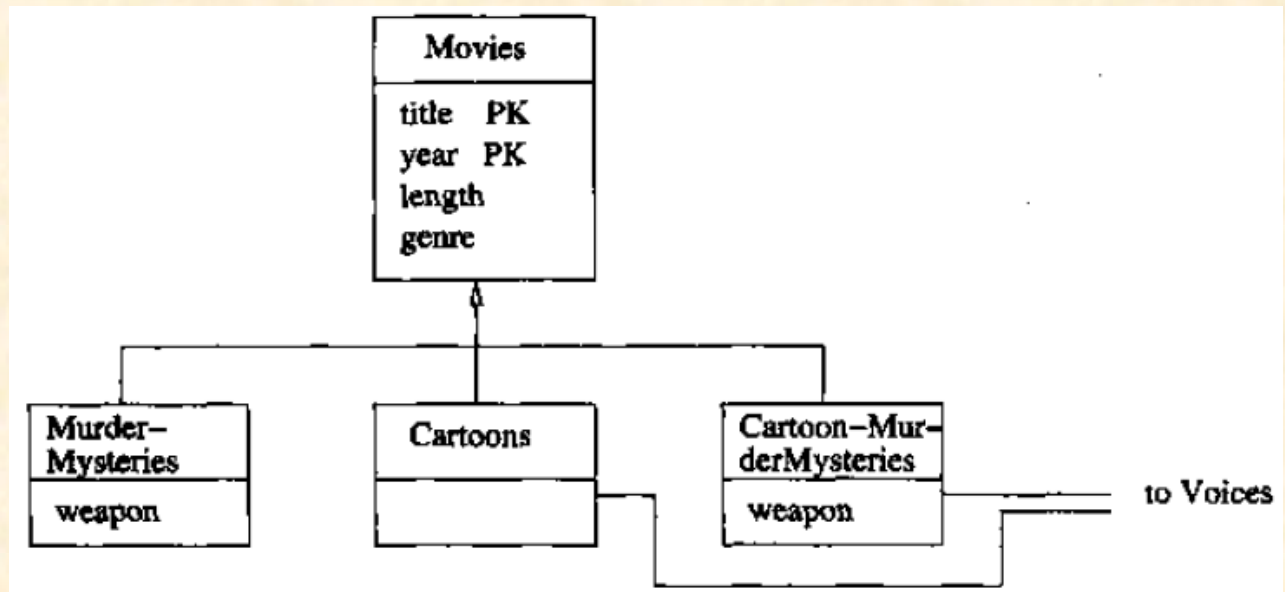
- 可以用曾在E/R模型中的方法将属性附加到一个关联中
- 在UML中，我们创建一个新类，称作**关联类** (association class)，并且将它放置在关联的中间
 - 关联类具有它自己的名字，但它的属性可认为是它依附的关联的属性



UML中的子类¹⁹⁷

□ 任何UML类在它下面可以有一个子类的层次

- 主键来自根层次，就像是E/R模型的层次
- 和任何类一样，子类由矩形来表示
 - 一个子类从它的超类继承了特征（属性和关联）
 - 任何额外的属于子类的属性都显示在那个子类的方框中，并且子类可以有自己的、额外的与其他类的关联



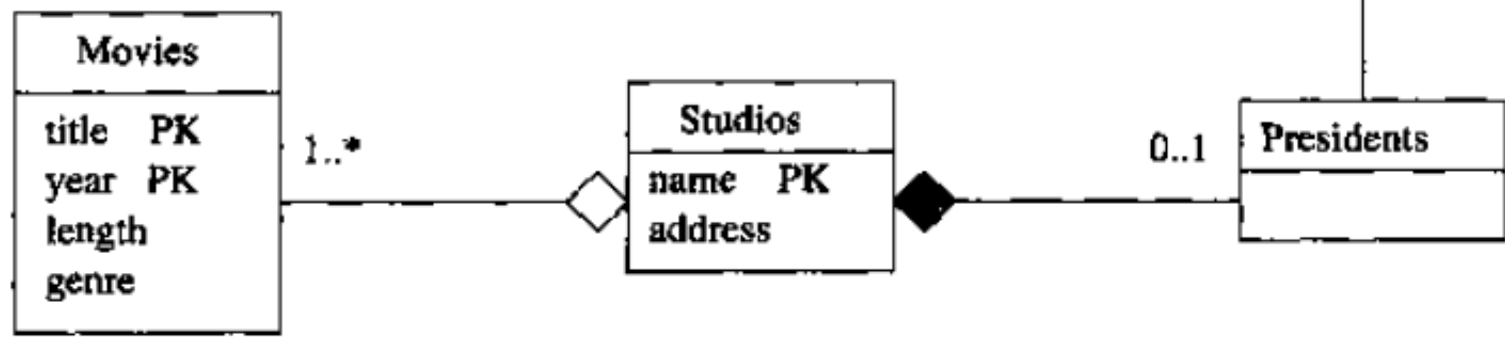
□ 对于多对一的关联有两个特殊的标记，它们的含义相当微妙

➤ 聚集 (aggregation)

- 在两个类之间的一条线，线的末端是一个空的菱形，含义是一定要为0..1

➤ 组合 (composition)

- 与关联相似，但在菱形端的标注一定要为1..1。也就是说，与菱形对应相连的类的每个对象都要与菱形端的一个对象相连接。组合的菱形是实心的黑色



UML图到关系的转化

□ 许多将E/R图转化为关系的想法对于UML也是一样

- 类到关系
- 关联到关系
- 从UML子类到关系
- 从聚集与组合到关系
- UML与弱实体集的类比

第2章 数据模型

2.1 数据模型的基本概念

2.2 数据模型的四个世界

2.3 概念世界与概念模型

2.3.1 E-R模型

2.3.2 EE-R模型

2.3.3 面向对象模型

2.3.4 谓词模型

2.4 信息世界与逻辑模型

2.5 计算机世界与物理模型

2.3.4 谓词模型

- 谓词模型又称谓词逻辑模型，它使用数理逻辑中的一阶谓词演算公式来表示数据模型
 - 数理逻辑是一种用数学方法来研究逻辑推理和证明的学科
 - 而一阶谓词演算则是数理逻辑的一个分支，它具有较强的表示能力，在计算机科学中被广泛应用于：人工智能、演绎数据库、知识库等领域
- 谓词模型主要用于构造知识库系统的‘概念数据模型’和‘逻辑数据模型’

2.3.4 谓词模型

□ 谓词

- 一种由一个谓词标识符号 P 和若干个变元(个体变量) x_1, x_2, \dots, x_n 所组成的符号。如:

$$P(x_1, x_2, \dots, x_n)$$

- 谓词是一个能够根据变元的取值来判断其是否成立的逻辑表达式，相当于一个带有变量的命题公式

—例如:

- $F(x, y, z) : x^2 + y^2 = z^2$, x, y, z 是自然数
- $P(x) : x$ 是联合国常任理事国

2.3.4 谓词模型

1. 实体集

- 假设一个实体集 R 有 n 个属性 A_1, A_2, \dots, A_n , 那么我们可以用一个含有 n 个变元的谓词 $P(x_1, x_2, \dots, x_n)$ 来表示该实体集
 - 实体集 R 中的元组使 $P(x_1, x_2, \dots, x_n)$ 为真, 而非实体集 R 的元组则使 $P(x_1, x_2, \dots, x_n)$ 为假, 即:

$$P(a_1, a_2, \dots, a_n) = \begin{cases} \text{true,} & \text{当实体}(a_1, a_2, \dots, a_n) \in R \\ \text{false,} & \text{当实体}(a_1, a_2, \dots, a_n) \notin R \end{cases}$$

2.3.4 谓词模型

编号	姓名	性别	年龄	籍贯	政治面貌
138	徐英健	女	18	浙江	团员
139	赵文虎	男	23	江苏	党员
140	沈亦奇	男	20	上海	群众
141	王 宾	男	21	江苏	群众
142	李红梅	女	19	安徽	团员

表2-1：人事档案简表

【例】可以为上面的‘人事档案简表’实体集定义一个谓词 $P(x_1, x_2, x_3, x_4, x_5, x_6)$ ，且只有该表中的5个元组才能使谓词P成立，其它的都不能使谓词P成立。如：

$P(140, \text{沈亦奇}, \text{男}, 20, \text{上海}, \text{群众}) = \text{true}$ ，而

$P(140, \text{沈亦奇}, \text{男}, 21, \text{上海}, \text{群众}) = \text{false}$

2.3.4 谓词模型

2. 属性

- 可以用谓词中的变元 x_i ($i=1, 2, \dots, n$)来表示实体中的属性
- 属性的域也可用谓词来表示
 - 如属性 x_3 为整型则可用： $\text{Int}(x_3)$ 表示之
 - 这里的 $\text{Int}(x)$ 是系统定义的内部谓词，用于约束变元 x 的取值为整型值

2.3.4 谓词模型

- 一般，对谓词中的每个变元均有一定的约束，它们可用统一的约束谓词 $C(x_i)$ ($i=1, 2, \dots, n$) 表示
- 因此一个具有 n 个属性的实体集往往可用下面的谓词公式来表示：

$$P(x_1, x_2, \dots, x_n) \wedge C(x_1) \wedge C(x_2) \wedge \dots \wedge C(x_n)$$

2.3.4 谓词模型

3. 联系

- 也可以用谓词来表示联系，谓词中的变元由参与该联系的实体（通常用实体的关键字属性代替）以及联系本身所具有的属性组成

2.3.4 谓词模型

【例】‘学生’与‘课程’之间的‘选课’联系
(图2-12) 可以表示为:

$$SC(s, c, g) = \begin{cases} \text{true} & \text{如果学生 } s \text{ 选修了课程 } c \text{ 且成绩为 } g \\ \text{false} & \text{如果学生 } s \text{ 没有选修课程 } c, \text{ 或者课程 } c \text{ 的成绩不是 } g \end{cases}$$

2.3.4 谓词模型

【例】‘工厂’，‘产品’与‘用户’之间的‘供应’联系（图2-10）可以表示为：

$$\text{FPU}(f, p, u) = \begin{cases} \text{true} & \text{如果工厂 } f \text{ 向用户 } u \\ & \text{供应了产品 } p \\ \text{false} & \text{如果工厂 } f \text{ 没有向} \\ & \text{用户 } u \text{ 提供产品 } p \end{cases}$$

2.3.4 谓词模型

4. 操作

- 也可以用谓词来表示模型上的操作

【例】如果有操作 $\sum_{i=1}^n x_i = X$

- 则我们可以用下面的谓词来表示该操作：

$$\text{Op}(x_1, x_2, \dots, x_n, X)$$

其中： x_1, x_2, \dots, x_n 是操作对象， X 是操作结果

2.3.4 谓词模型

【例】要在表2-1所示的‘人事档案简表’实体集中查询籍贯是‘江苏’的实体

- 设该实体集所对应的谓词为 $P(x_1, x_2, x_3, x_4, x_5, x_6)$
- 系统提供了一个已经定义好的内部谓词

$$E(x, y): x = y$$

- 那么该查询可以表示为如下定义的一个谓词 $ES(x_1, x_2, x_3, x_4, x_5, x_6)$:

$$P(x_1, x_2, x_3, x_4, x_5, x_6) \wedge E(x_5, \text{'江苏'})$$

2.3.4 谓词模型

5. 完整性约束

- 可以用谓词或谓词公式来表示属性间的完整性约束条件

【例】要在表2-1所示的‘人事档案简表’实体集中定义一个完整性约束条件：职工的年龄必须在18到60岁之间

- 首先引入两个内部谓词

$G(x, y): x \geq y$ 和 $L(x, y): x \leq y$

- 那么上述的约束条件可以表示为：

$G(\text{年龄}, 18) \wedge L(\text{年龄}, 60)$

- 也可以表示为：

$G(\text{年龄}, 18) \wedge G(60, \text{年龄})$

2.3.4 谓词模型

【例】 男职工的年龄必须在18到60岁之间，而女职工的年龄必须在18到55岁之间

➤ 该约束条件可以表示为：

$$(E(\text{性别}, '男') \wedge G(\text{年龄}, 18) \wedge L(\text{年龄}, 60)) \\ \vee (E(\text{性别}, '女') \wedge G(\text{年龄}, 18) \wedge L(\text{年龄}, 55))$$

□ 综上所述，我们可以用谓词或谓词公式来描述数据模型中的实体集、属性、联系、操作以及完整性约束等相关内容，并且可以描述更为复杂的模型语义信息

例2.9 一个关于‘圆’的数据模型

□ 圆的模式定义

$$C_r(\text{cno}, x, y, r) \wedge C_1(\text{cno}) \wedge C_2(x) \wedge C_3(y) \wedge C_4(r)$$

–其中：cno是圆的标识属性，x和y是圆心的横坐标和纵坐标值，r是圆的半径

□ 圆上操作的定义

➤放大：X (xno, r')

–其中：xno是被放大的圆的标识，r'是半径的增大数

➤移动：M (mno, a, b)

–其中：mno是被移动的圆的标识，a, b是圆心x, y轴修改量

回顾

□ E-R模型

➤ 实体、属性、联系

□ EE-R模型

➤ IS-A、弱实体

□ 面向对象模型

➤ 对象、类、消息

□ 谓词模型

□ 作业：P43页，2.10、2.11题

第2章 数据模型

2.1 数据模型的基本概念

2.2 数据模型的四个世界

2.3 概念世界与概念模型

2.4 信息世界与逻辑模型

2.5 计算机世界与物理模型

2.4 信息世界与逻辑模型

□ 信息世界是数据库的世界，它着重于数据模型在数据库系统一级的构造与操作

➤ 信息世界用逻辑数据模型来进行描述

□ 逻辑模型的分类

➤ 由于在不同时期以及不同的数据库厂商之间采用的实现手段和方法的不同，因此逻辑模型的种类很多

➤ 在数据库技术的发展历程中，具有比较重要的影响和历史地位的逻辑模型有：

— 层次模型和网状模型

— 关系模型和对象关系模型

— 面向对象模型

— 谓词模型

2.4 信息世界与逻辑模型

□ The '60s: 数据库技术的萌芽阶段

➤ 1961: IDS (Integrated Data Store)

– Designed by Bachman

– 奠定了网络数据模型(Network Data Model)的基础

▪ Conference on Data Systems Languages
Database Task Group (CODASYL DBTG)

2.4 信息世界与逻辑模型

- 1965-1970: IMS (Information Management System)
 - Developed by IBM
 - 奠定了层次数据模型(Hierarchical Data Model)的基础
 - 允许对数据的多用户存取

2.4 信息世界与逻辑模型

□ The '70s: 数据库技术飞速发展阶段

➤ 1970: 关系数据模型 (The Relational Model)

– Developed by E. F. Codd

➤ 1971: CODASYL Database Task Group
Report

2.4 信息世界与逻辑模型

- 1975: 著名的国际会议
 - ACM Special Interest Group on Management Of Data organized first SIGMOD international conference
 - Very Large Data Base Foundation organized first VLDB international conference
- 1976: Entity-Relationship (ER) model

2.4 信息世界与逻辑模型

□ 有影响的研究工作

- **System R** (IBM)
- **INGRES** (University of California, Berkeley)
- **System 2000** (University of Texas, Austin)
- **Socrate Project** (University of Grenoble, France)
- **ADABAS** (Technical University of Darmstadt, W. Germany)

□ 数据库语言

➤ SQUARE, SEQUEL (**SQL**), QBE, QUEL

2.4 信息世界与逻辑模型

□ The '80s: 商用关系数据库管理系统的兴起

➤ DBMS在PC平台上的实现

DBASE (Foxbase, Foxpro等) ,
PARADOX

➤ 1983: 商用关系数据库管理系统的出现与流行

DB2, ORACLE, SYBASE, INFORMIX,

➤ 1985:

发布最初的SQL标准

2.4 信息世界与逻辑模型

➤ 其它方向:

- Expert Database Systems
- Object-oriented DBMSs

2.4 信息世界与逻辑模型

□ The '90s

- 专用数据库系统

spatial, temporal, multimedia
data,

- 商用面向对象数据库管理系统

- 对象关系数据库管理系统

□ 新世纪以来

- 数据仓库

- 安全数据库

- XML数据库

- 嵌入式、移动、实时、.....

2.4 信息世界与逻辑模型

□ NoSQL (= Not only SQL)

➤ 泛指非关系型的数据库

— 键值存储数据库

— 列存储数据库

— 文档型数据库

— 图数据库

□ NewSQL

□ 是对各种新的可扩展/高性能数据库的简称

2.4 信息世界与逻辑模型

□ 概念模型与逻辑模型对应关系表

概念模型	E – R模型			EE – R模型	面向对象模型	谓词模型
逻辑模型	层次模型	网状模型	关系模型	对象关系模型	面向对象模型	谓词模型

2.4.1 关系模型与关系模型数据库系统

□ 关系模型 (Relational model) 是完全不同于层次模型和网状模型的一种新的逻辑模型

➤ 关系模型的基本数据结构

— 二维表，简称‘表’ (Table)

➤ 关系模型的数据操纵

— 是建立在二维表上的操作，它包括对一张表及多张表间的查询，以及对一张表的删除，插入及修改等操作

2.4.1 关系模型与关系模型数据库系统

□ 二维表 (Table)

- 二维表由表框架与元组所组成，表框架由若干个属性组成
- 存放于框架内的每‘一行数据’都被称为‘一个元组’ (Tuple)，或称‘行’ (Row)
- 一张二维表是由一个有 n 个属性的框架及 m 个元组组成

2.4.1 关系模型与关系模型数据库系统

□ 关系

➤ 由行和列组成的二维表格

➤ 关系的约束

1) 同一表中的属性名各不相同

2) 表中的属性与属性的排放次序无关

3) 表中的元组均不相同

4) 表中的元组与元组的排列次序无关

5) 表中的每一分量必须是一个不可分割的基本数据项

2.4.1 关系模型与关系模型数据库系统

学生

学号	姓名	系别	年龄
S1001	张曼英	计算机	19
S1002	李 红	数学	20
S1003	丁一珉	中文	18
S1004	王爱民	计算机	20

课程

课程编号	课程名	主讲教师
C101	C++	T02
C102	OS	T01
C103	DB	T02

2.4.1 关系模型与关系模型数据库系统

□ 关系中的基本概念

- 关系模式
- 关系数据库模式
- 元组
- 关键字（或简称为‘键’-key）
 - 主关键字
 - 外关键字

2.4.1 关系模型与关系模型数据库系统

□ 关系模式

- 一个关系的关系名及其属性名的集合构成该关系的关系模式

□ 关系数据库模式

- 该关系数据库中所有关系的关系模式的集合

□ 元组

- 关系中的每一行

2.4.1 关系模型与关系模型数据库系统

□ 关键字

- 关系中的一个属性集的值能唯一标识关系中的一个元组，且又不含多余的属性值，则称该属性集为该关系的关键字
- 每一个关系都有关键字
- 一个关系也可以有多个关键字，所以关键字也被称为‘**候选关键字**’

2.4.1 关系模型与关系模型数据库系统

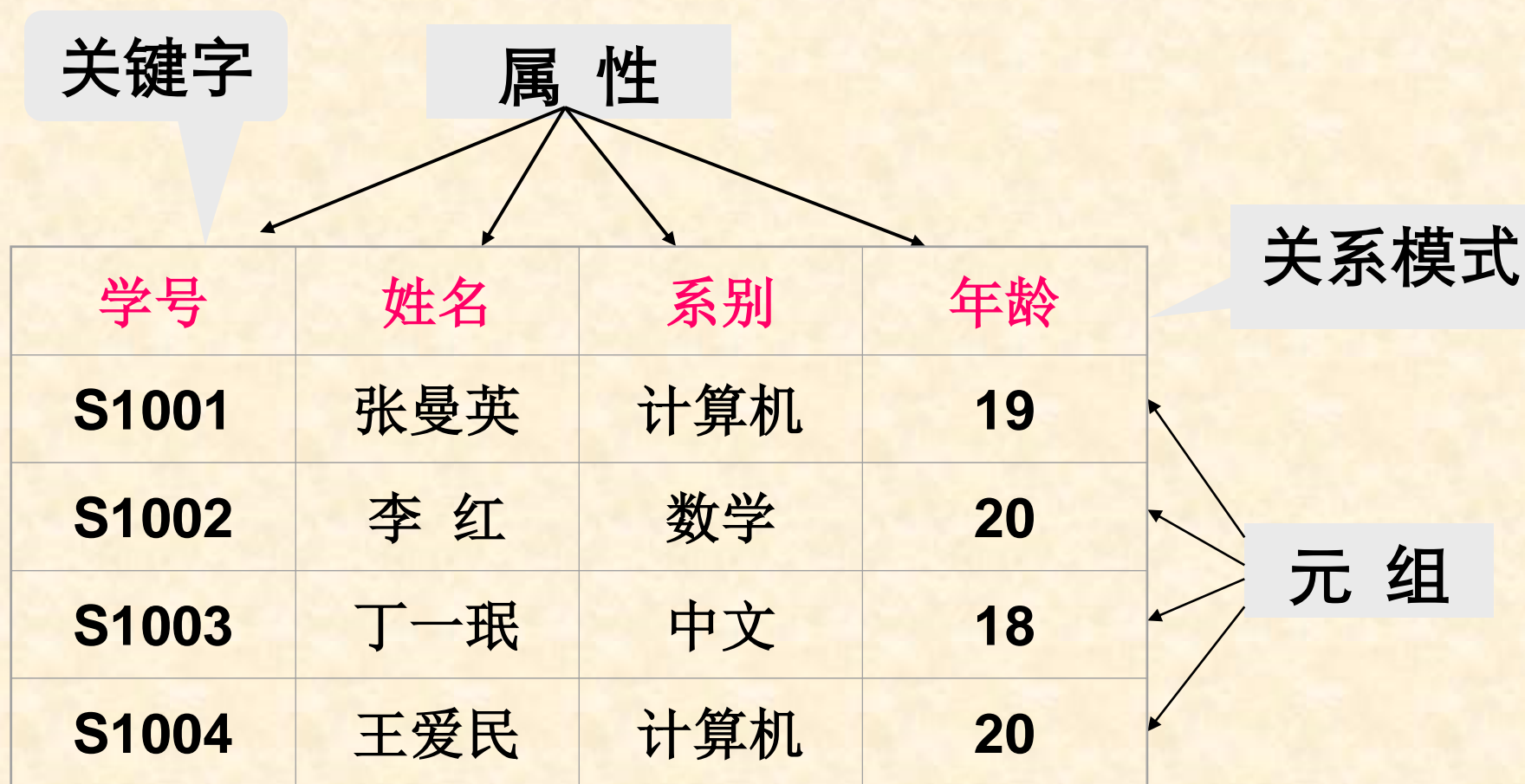
□主关键字

- 可以从关系的候选关键字中选取一个作为该关系的主关键字

□外关键字

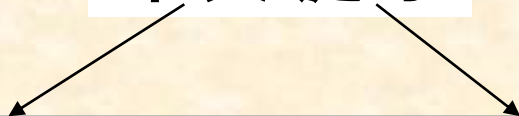
- 设关系R中的属性集F，其取值来自于关系S中的主关键字K，则称属性集F是关系R的外关键字
- 关系R和关系S可以是同一个关系

2.4.1 关系模型与关系模型数据库系统



2.4.1 关系模型与关系模型数据库系统

外 关 键 字



学 号	课程编号	成 绩
S1001	C101	85
S1001	C102	91
S1001	C103	95
S1003	C101	90
S1002	C101	76
S1003	C102	87
S1004	C101	88

2.4.1 关系模型与关系模型数据库系统

□ 关系模型上的数据操作

- 关系模型数据操作的对象是‘关系’
- 关系模型数据操作的结果也是一个‘关系’
- 关系模型的五种基本操作：
 - 属性指定
 - 元组选择
 - 关系的合并
 - 元组插入
 - 元组删除

第2章 数据模型

2.1 数据模型的基本概念

2.2 数据模型的四个世界

2.3 概念世界与概念模型

2.4 信息世界与逻辑模型

2.5 计算机世界与物理模型

2.5 计算机世界与物理模型

□ 物理模型是面向计算机的模型，它构造数据库系统的物理实现

➤ 主要涉及操作系统级文件组织，有时还会涉及到硬件级数据组织

2.5 计算机世界与物理模型

□ 文件系统的组成

- **项 (Item)** : 文件系统中最小基本单位, 项内符号是不能继续分割的
- **记录 (Record)** : 由若干项组成, 记录内的各项间有内在语义联系
 - 记录有型与值的区别
- **文件 (file)** : 记录的集合
 - 一般讲, 一个文件所包含的记录都是同型的
- **文件集 (file set)** : 由若干个文件构成

2.5 计算机世界与物理模型

□ 提高文件读写操作效率的方法

- 索引 (Index)
- Hash法
- 集簇 (Cluster)

2.5 计算机世界与物理模型

□ 索引 (Index)

- 将文件中的记录与其物理地址（即磁盘块）间建立一张对应关系表以便于快速查找，这就是索引
- 索引一般也是一个文件。当数据文件中的记录数很大时，索引文件本身也还需要建立索引，这叫二级索引
- 依此类推，可以建立多级索引

2.5 计算机世界与物理模型

□ Hash法

- 一种函数转换法，其主要思想是：通过一个hash函数将要查找的记录转换成该记录所在的物理地址，然后可以直接进行记录的定位读取操作

□ 集簇 (Cluster)

- 在记录查找中往往需要按某项的项值查找，将具有相同或相邻项值的记录聚集在相同磁盘块内或圆柱体内以减少读盘次数，提高查找速度，这被称为集簇

本章小结

- 本章讨论数据模型，它是数据库系统的核心

- 数据模型

- 基本概念
- 三个抽象层次上的数据模型

- 概念数据模型

- 四种概念模型
- E-R模型，E-R图
- EE-R模型：IS-A联系，弱实体
- 面向对象模型

- 逻辑数据模型

- 关系模型