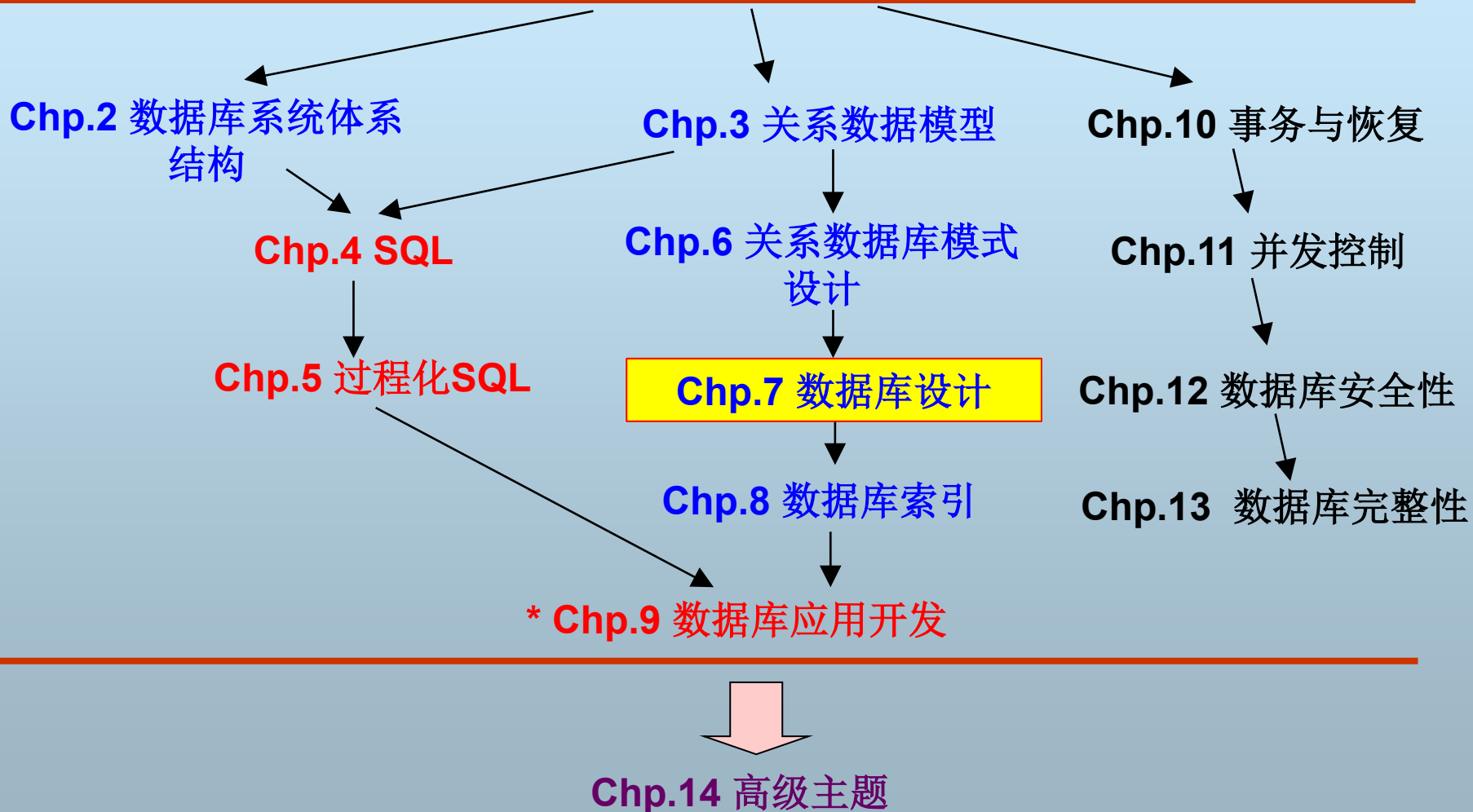


第7章 数据库设计

课程知识结构

Chp.1 数据库系统概述



一、什么是数据库设计

- 对于给定的应用环境，构造最合适的数据库模式，并利用现成的**DBMS**，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的需求
 - 面向特定应用
 - 逻辑设计
 - 物理设计

二、数据库设计方法

- 数据库设计是一种方法而不是工程技术，缺乏科学的方法论支持，很难保证质量
- 规范化设计方法：运用软件工程的思想方法进行数据库设计
 - 新奥尔良方法（New Orleans）
 - ◆ 需求分析、概念设计、逻辑设计、物理设计
 - 基于ER模型的方法
 - 基于关系模式的设计方法
 - 基于3NF的设计方法
 - 计算机辅助数据库设计方法

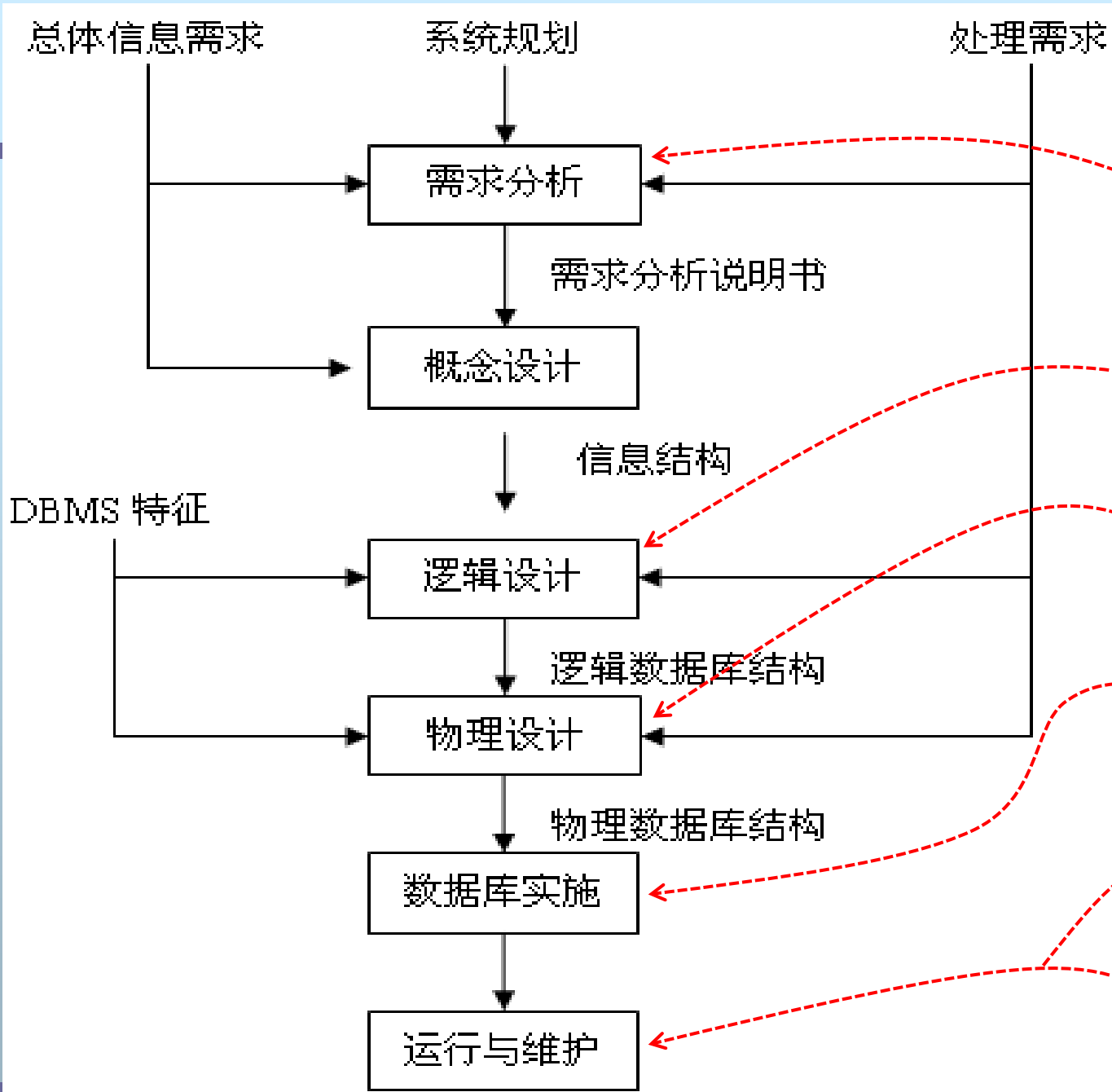
数据库设计不同阶段上的具体实现技术和方法

我们的选择

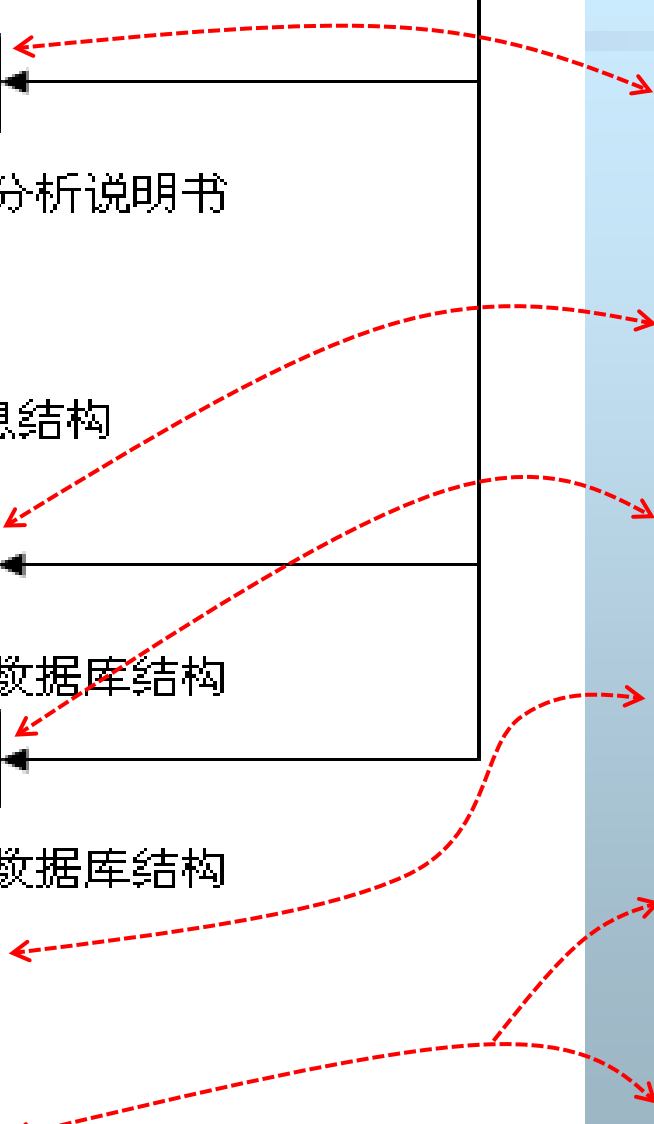
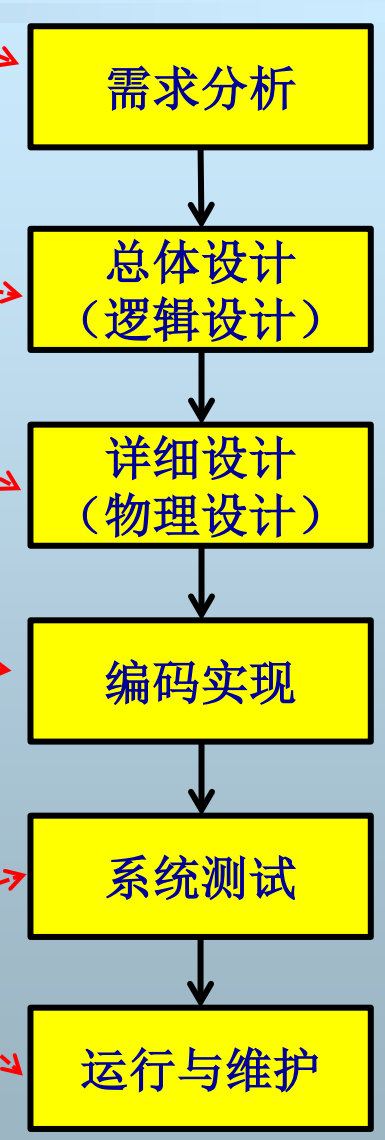
- 以新奥尔良方法为基础，基于ER模型和关系模式，采用计算机辅助进行数据库设计
 - 概念设计：基于ER模型
 - 逻辑设计：基于关系模式设计
 - 计算机辅助设计工具
 - ◆ ERWIN (CA)
 - ◆ Power Designer (Sybase, now SAP)
 - ◆ Workbench (MySQL)
 - ◆ Visible Analyst (Visible)
 - ◆ Navicat Data Modeler (PremiumSoft)
 - ◆

三、数据库设计步骤

- 需求分析
- 概念设计
- 逻辑设计
- 物理设计
- 数据库实施
- 数据库运行与维护



对比：软件工程



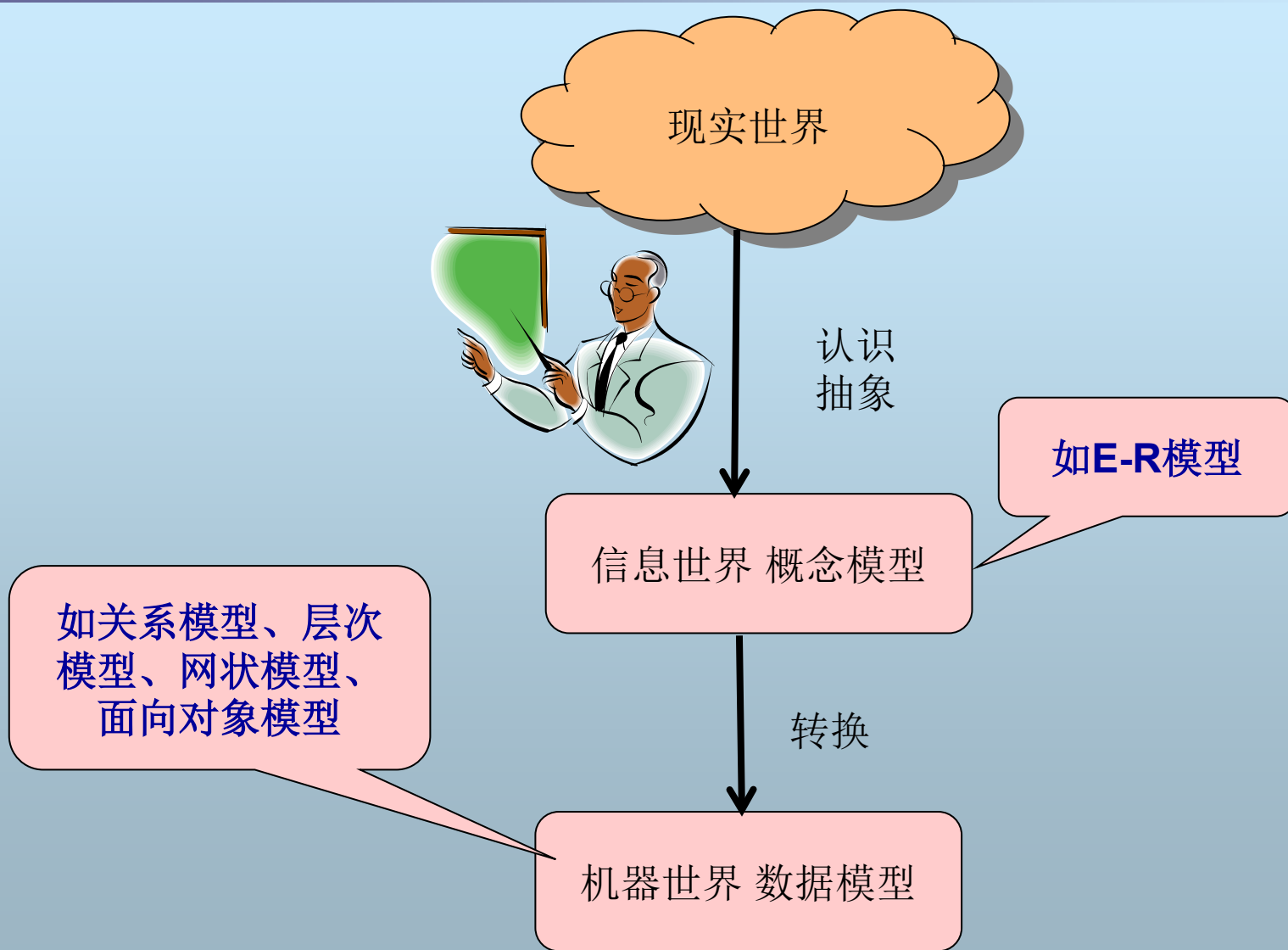
输入输出

- **输入：总体信息需求、处理需求、DBMS特征**
 - **总体信息需求：数据库应用系统的目标、数据元素的定义、数据在组织中的使用描述**
 - **处理需求：每个应用需要的数据项、数据量以及处理频率**
 - **DBMS特征：DBMS说明、支持的模式、程序语法**
- **输出：数据库设计说明书（完整的数据库逻辑结构和物理结构、应用程序设计说明）**

四、概念设计（ER模型设计）

- 产生反映组织信息需求的数据库概念结构，即概念模型
 - 概念模型独立于数据库逻辑结构、DBMS以及计算机系统
 - 概念模型以一组ER图形式表示
- 概念设计侧重于数据内容的分析和抽象，以用户的观点描述应用中的实体以及实体间的联系

数据抽象的层次



1、ER模型的概念

■ ER模型 (Entity-Relationship Model)

- 1976, Peter .P. Chen (陳品山) 提出的概念设计方法
- 以ER图的方式表达现实世界实体及实体间的联系



[Louisiana State University](#)

Peter Chen. The Entity-Relationship Model--Toward a Unified View of Data. *ACM Transactions on Database Systems*, Vol. 1(1), p.9-36,1976

One of the 38 *most influential* papers in Computer Science
One of the 35 *most cited* article in Computer Science

The **entity-relationship model**—toward a unified view of data

[PDF] [acm.org](#)

PPS Chen - ACM transactions on database systems (TODS), 1976 - [dl.acm.org](#)

... of the three **data** models [4, 19, 26, 30, 311]. This paper uses the **entity-relationship** model as a framework from which the three existing **data** models may be derived. The reader may ...

☆ Save [Cite](#) Cited by 12706 [Related articles](#) [All 82 versions](#)

1、ER模型的概念

■ ER模型要素

● 实体 Entity

- ◆ 包含实体属性

● 实体与实体间的联系 Relationship

- ◆ 包含联系类型和联系属性

(1) 实体与联系

■ 实体 (Entity)

- 现实世界中可标识的对象
- 如学生、学校、发票、教室、系、班级.....
- 物理存在的实体 (教室、学生)、代表抽象概念的实体 (课程)
- 应用中的数据以实体的形式呈现
- 一个实体具有唯一的标识, 称为码 (Key)

■ 联系 (Relationship)

- 实体和实体之间发生的关联
- 一个实体一般都与一个或多个实体相关

(2) 联系的类型

■ 1对1联系 (1:1)

- 学校和校长、学生和学生简历.....
- **A和B是1:1联系指一个A只有一个B与其关联，并且一个B也只有一个A与其关联**

■ 1对多联系 (1:N)

- 公司和职工、系和学生、客户和订单.....
- **A和B是1:N联系指一个A可以有多个B与其关联，但一个B只有1个A关联**

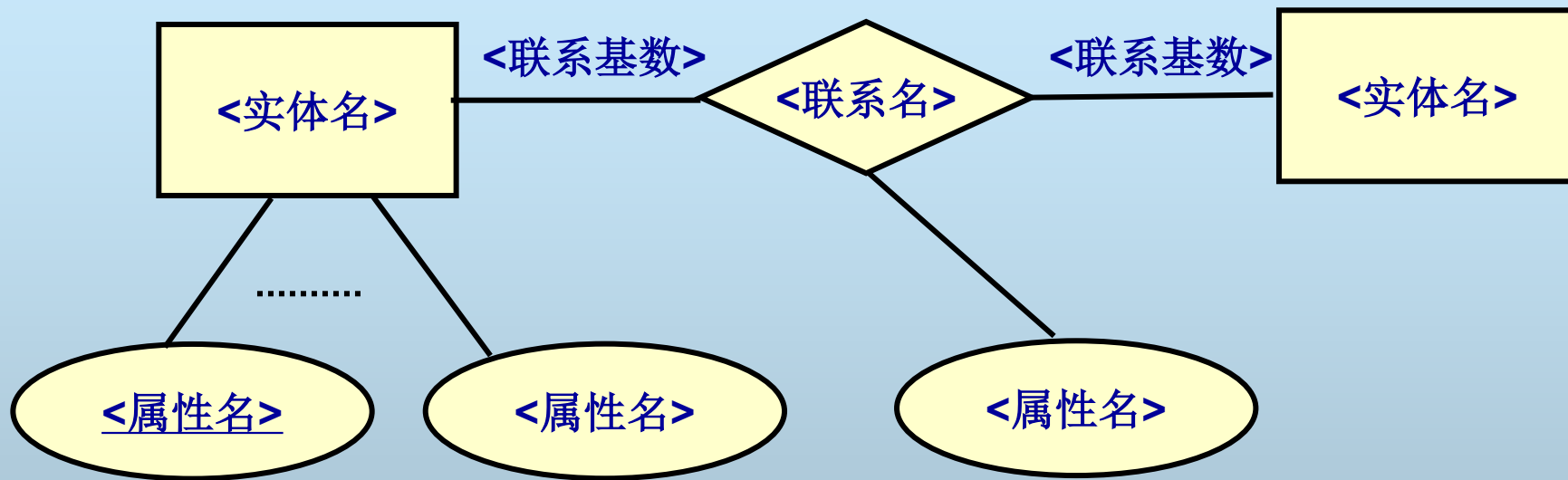
■ 多对多联系 (M:N)

- 学生和课程、教师和课程、医生和病人.....
- **一个A可有多个B对应，一个B也可有多个A对应**

(3) 联系的确定

- 联系的确定依赖于实体的定义和特定的应用，同样的实体在不同应用中可能有不同的联系
 - 部门和职工：若一个职工只能属于一个部门，则是1:N，若一个职工可属于多个部门，则是M:N
 - 图书馆和图书：若图书的码定义为索书号，则为M:N（一个索书号可能有几本相同的书）；若图书的码为图书条码，并且每本书有一个唯一条码，则为1:N联系

(4) ER图的符号



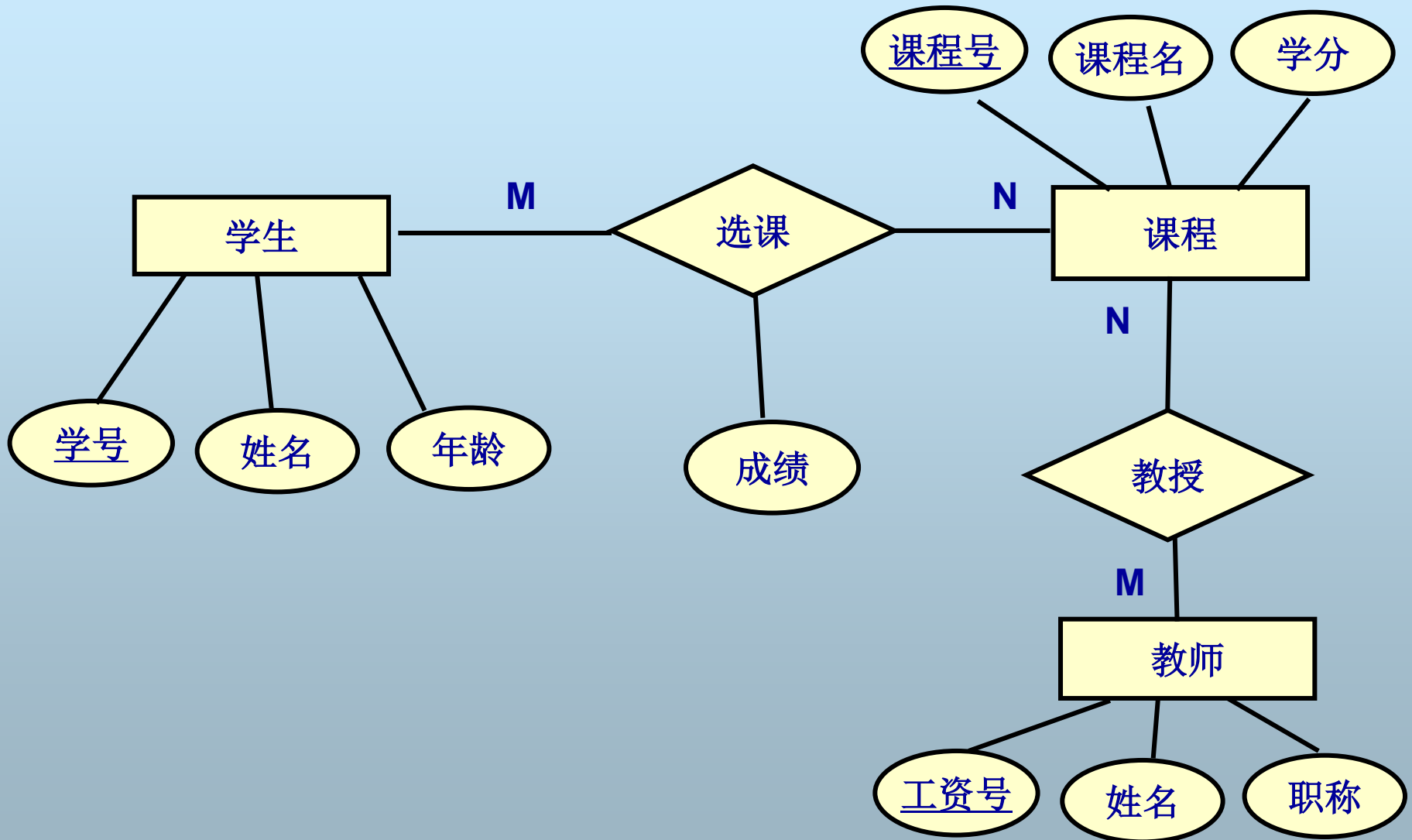
矩形：表示实体

菱形：表示联系，两端写上联系的基数

(1:N, M:N, 1:1)

椭圆形：表示属性，实体的码加下划线，联系也可有属性

(5) ER图例子：教学应用

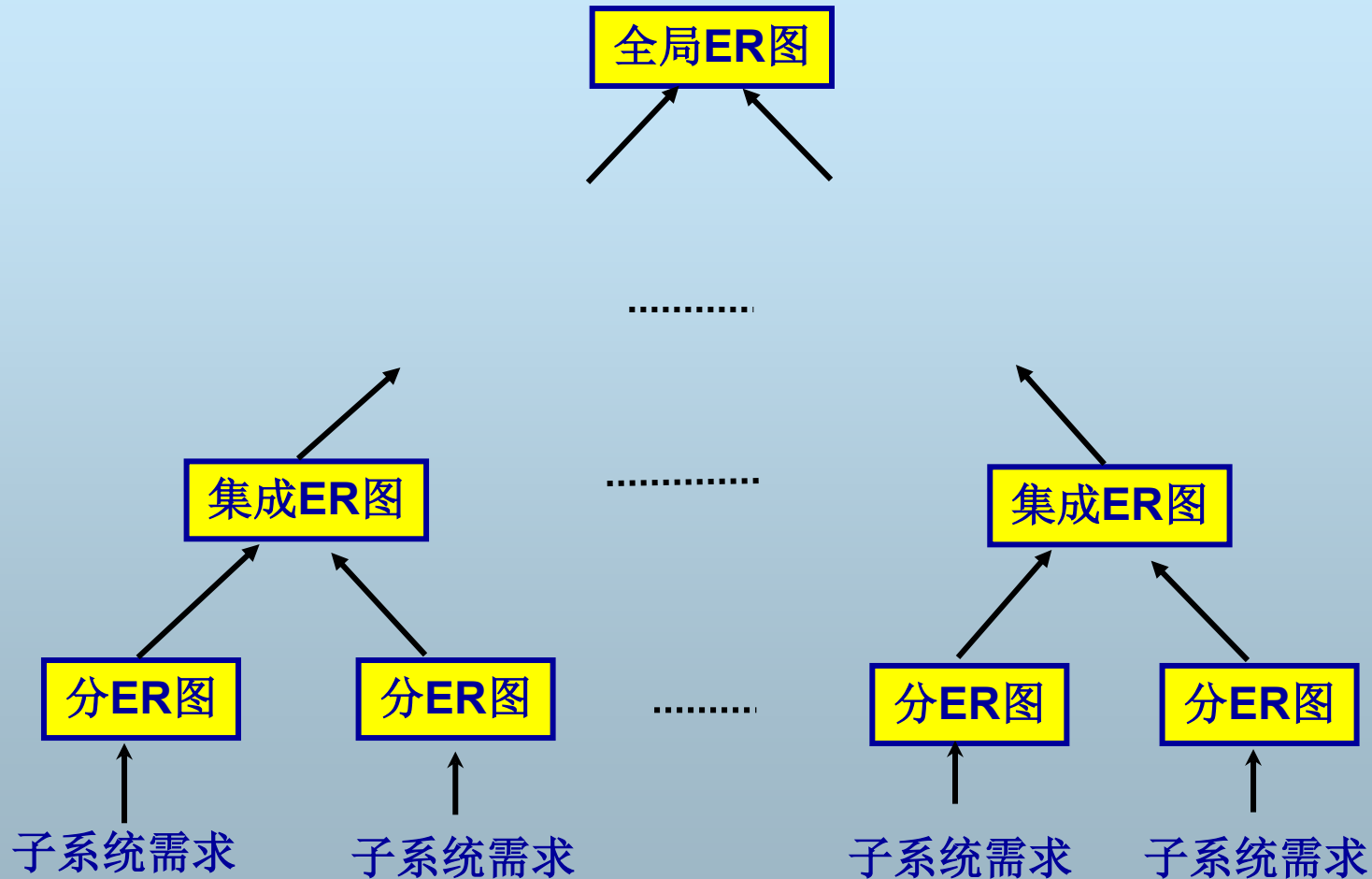


2、ER设计的步骤

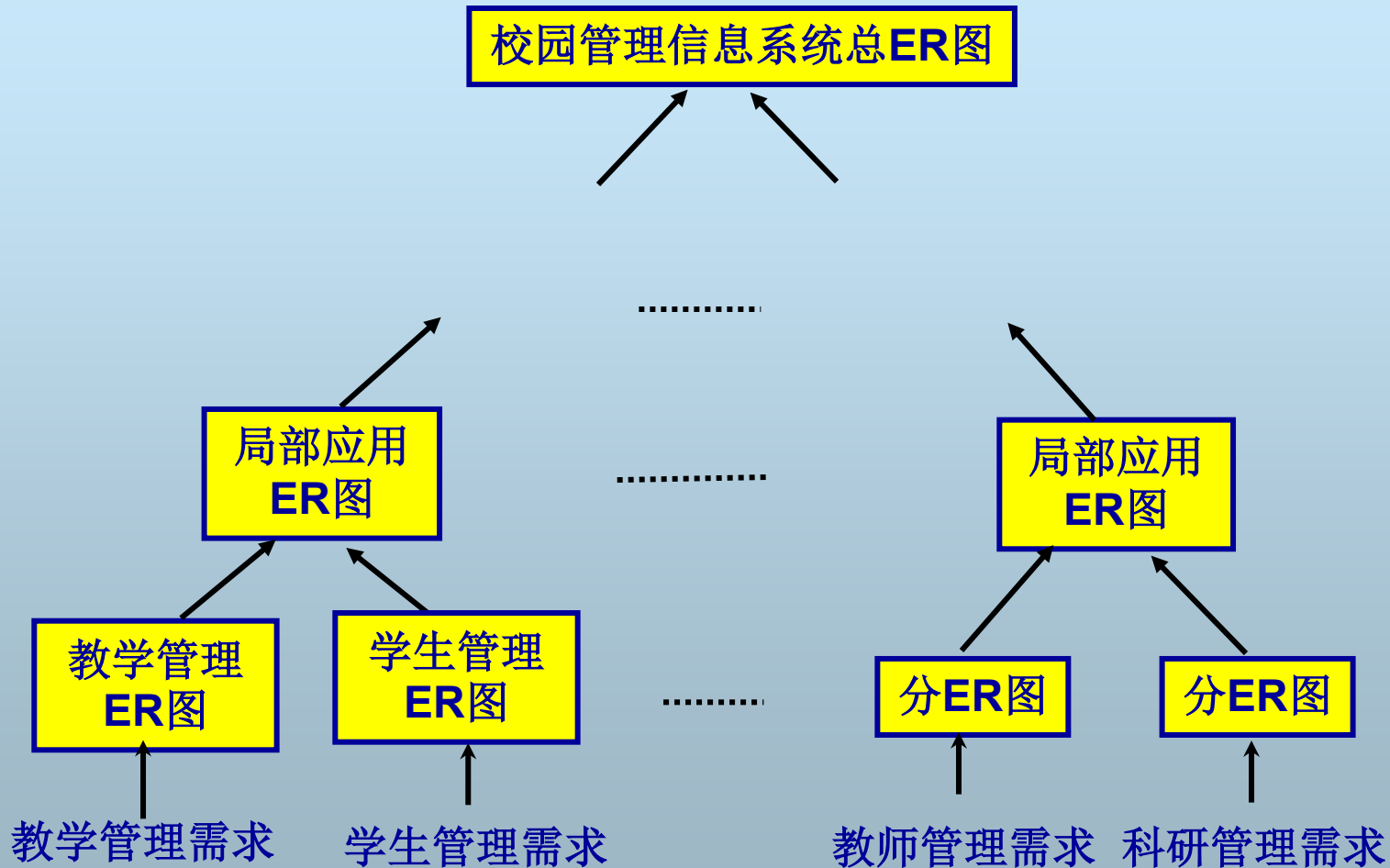
- 自顶向下进行需求分析，自底向上进行ER设计
 - 分ER模型设计（局部ER图）
 - ER模型集成
 - ER模型优化

如果应用比较简单则可以合为一个步骤

(1)ER设计的步骤示意



(2) ER设计步骤例子



(3) 分ER设计

- 通过实体、联系和属性对子系统的数据进行抽象，产生分ER图
 - 确定实体
 - 确定实体属性
 - 确定联系和联系属性
- 设计原则
 - 实体要尽可能得少
 - 现实世界中的事物若能作为属性就尽量作为属性对待

A) 确定实体

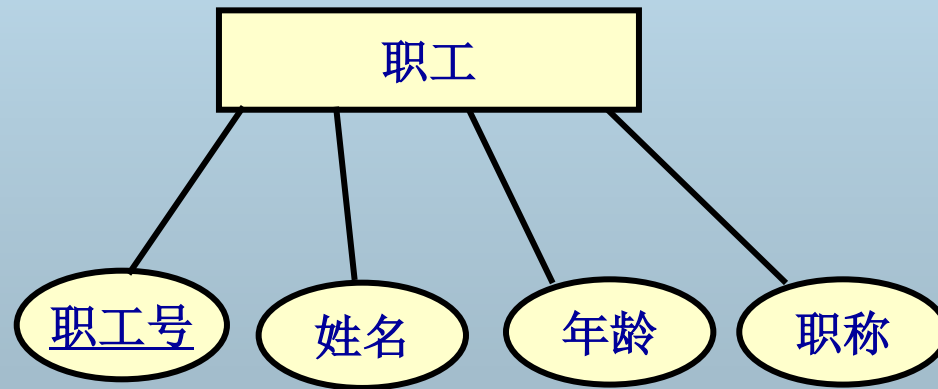
- 实体是一个属性的集合
- 需求分析阶段产生的数据字典中的数据存储器、数据流和数据结构一般可以确定为实体
 - 数据字典五个部分：数据项、数据结构、数据流、数据存储器 and 数据处理

B) 确定实体属性

- 实体和属性之间没有形式上可以截然划分的界限
 - 首先确定实体的码
 - 只考虑系统范围内的属性
 - 属性应具有域
 - 属性一般要满足下面的准则
 - ◆ 属性必须不可分，不能包含其它属性
 - ◆ 属性不能和其它实体具有联系

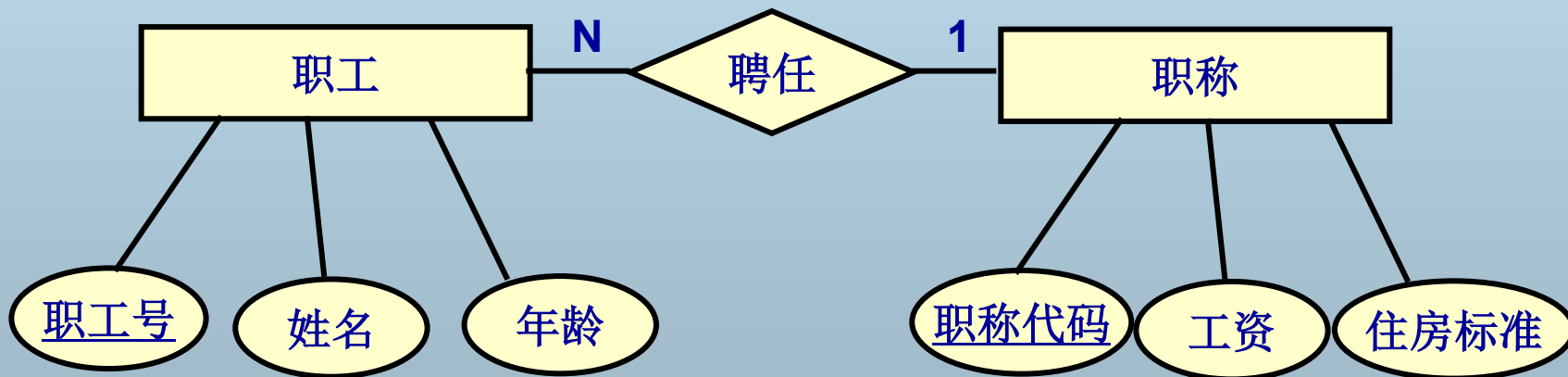
属性设计例子1

- 职工是一个实体，职工号、姓名、年龄是职工的属性，如果职工的职称没有进一步的特定描述，则可以作为职工的属性



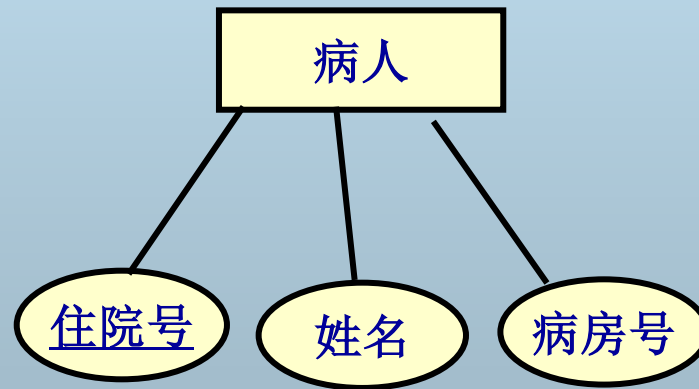
属性设计例子1

- 如果职称与工资、福利等挂钩，即职称本身还有一些描述属性，则把职称设计为实体比较恰当



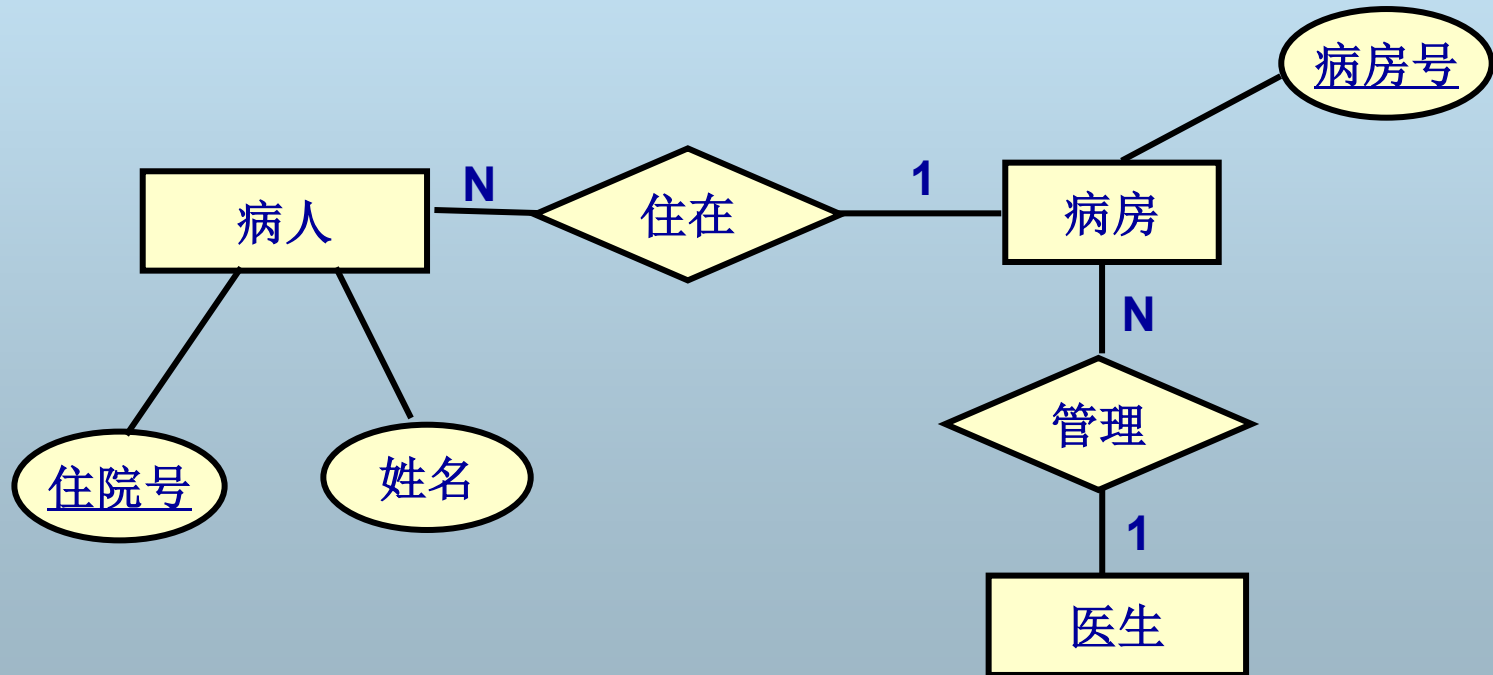
属性设计例子2

- 医院管理中，一个病人只能住在一个病房里，因此病房号可以作为病人实体的一个属性



属性设计例子2

- 但如果病房与医生实体存在负责联系，即一个医生要负责管理多个病房，而一个病房的管理医生只有一个



C) 确定联系和联系属性

■ 根据数据需求的描述确定

● 数据项描述

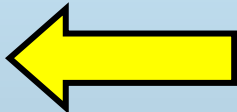
- ◆ {数据项名, 数据项含义说明, 别名, 数据类型, 长度, 取值范围, 取值含义, 与其它数据项的逻辑关系, 数据项之间的联系}

● 参考书: “系统分析与设计” 或 “软件工程”

■ 联系的基数

- 0个或1个 (国家和总统: 1个国家可以有0个或1个总统)
- 0个或1个或多个 (学院和系)
- 1个或多个 (班级和学生)
- 1个 (公司和法人)
- 确定的k个 (候选人和推荐人: 一个候选人必须有3个候选人)

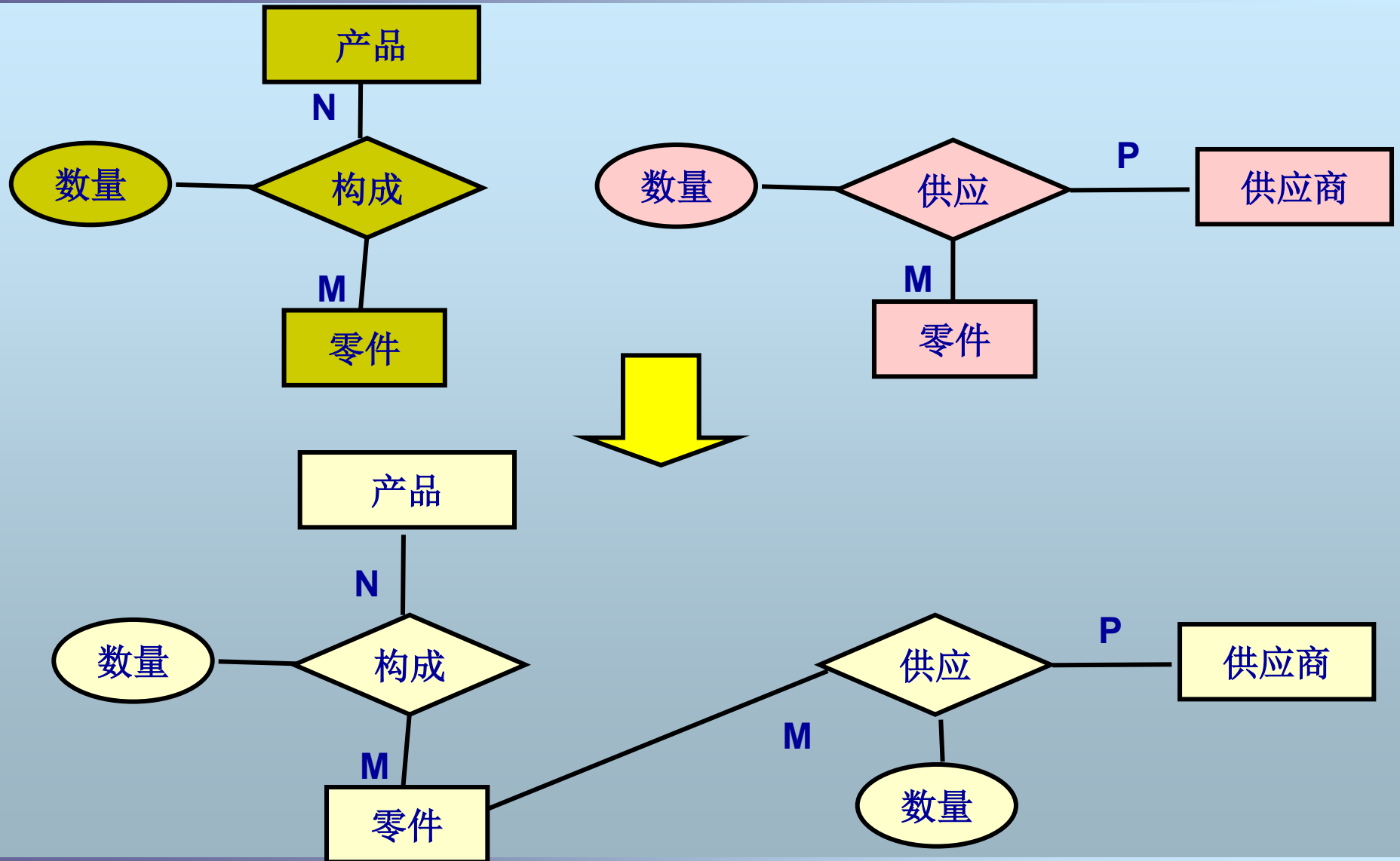
2、ER设计的步骤

- 自顶向下进行需求分析，自底向上进行ER设计
 - 分ER模型设计（局部ER图）
 - ER模型集成 
 - ER模型优化

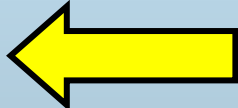
(4) ER集成

- 确定公共实体
- 合并分ER图
- 消除冲突
 - 属性冲突：类型冲突、值冲突
 - ◆ 例如性别、年龄
 - 结构冲突：实体属性集不同、联系类型不同、同一对象在不同应用中的抽象不同
 - 命名冲突：同名异义、异名同义
 - ◆ 实体命名冲突、属性命名冲突、联系命名冲突

ER集成示例



2、ER设计的步骤

- 自顶向下进行需求分析，自底向上进行ER设计
 - 分ER模型设计（局部ER图）
 - ER模型集成
 - ER模型优化 

(5) ER模型的优化

■ 目标

- 实体个数要少，属性要少，联系尽量无冗余

■ 具体优化手段

- 合并实体类型
- 消除冗余属性
- 消除冗余联系

A) 合并实体

- 一般**1:1**联系的两个实体可以合并为一个实体
- 如果两个实体在应用中经常需要同时处理，也可考虑合并
 - 例如病人和病历，如果实际中通常是查看病人时必然要查看病历，可考虑将病历合并到病人实体中
 - ◆ 减少了连接查询开销，提高效率

B) 消除冗余属性

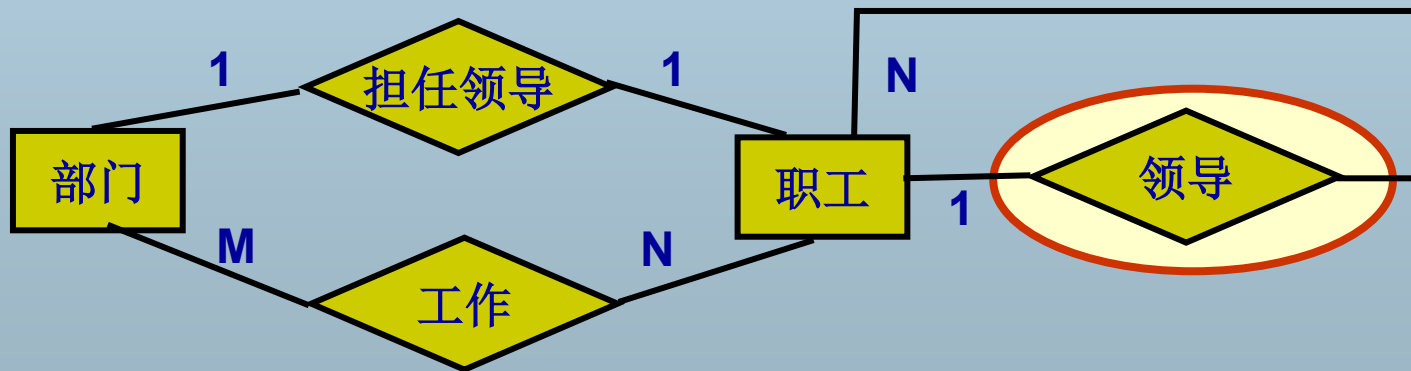
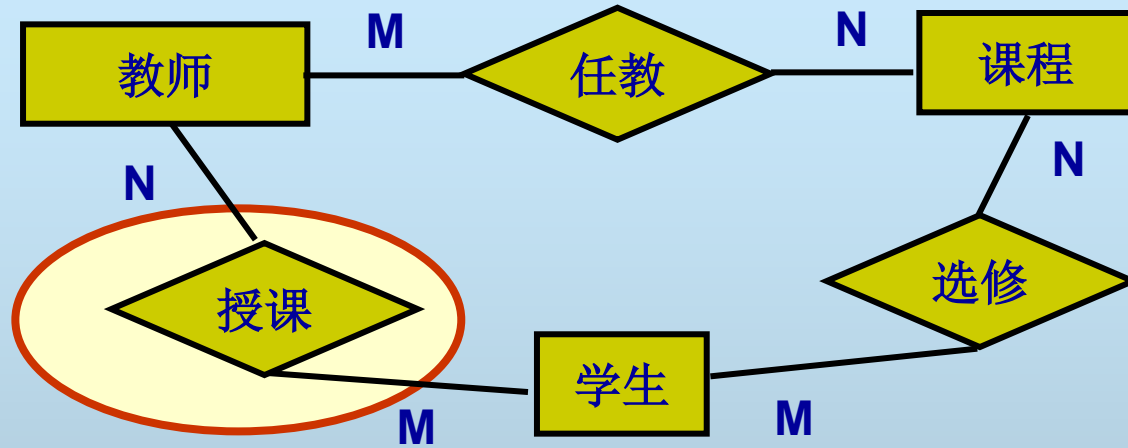
- 分ER图中一般不存在冗余属性，但集成后可能产生冗余属性
 - 例如，教育统计数据库中，一个分ER图中含有高校毕业生数、在校学生数，另一个分ER图中含有招生数、各年级在校学生数
 - 每个分ER图中没有冗余属性，但集成后“在校学生数”冗余，应消除

B) 消除冗余属性

■ 冗余属性的几种情形

- 同一非码属性出现在几个实体中
- 一个属性值可从其它属性值中导出
 - ◆ 例如出生日期和年龄

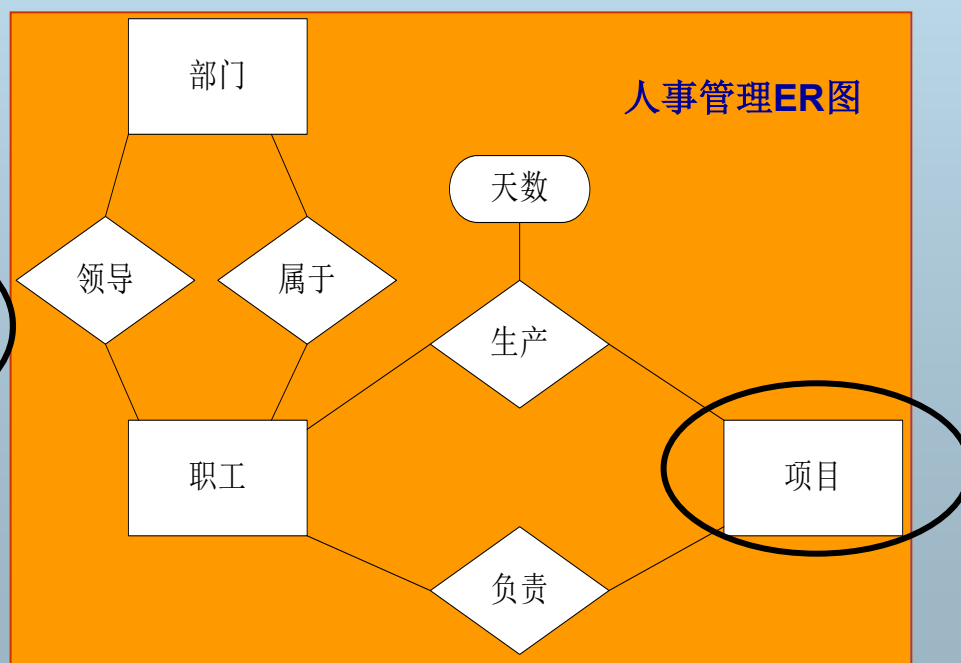
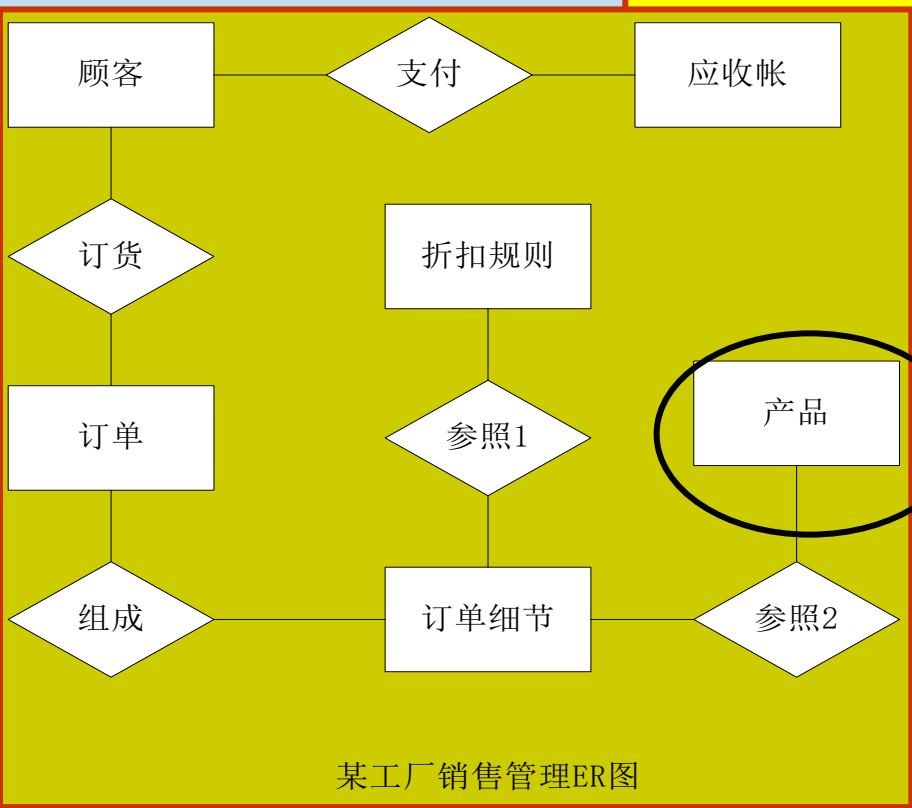
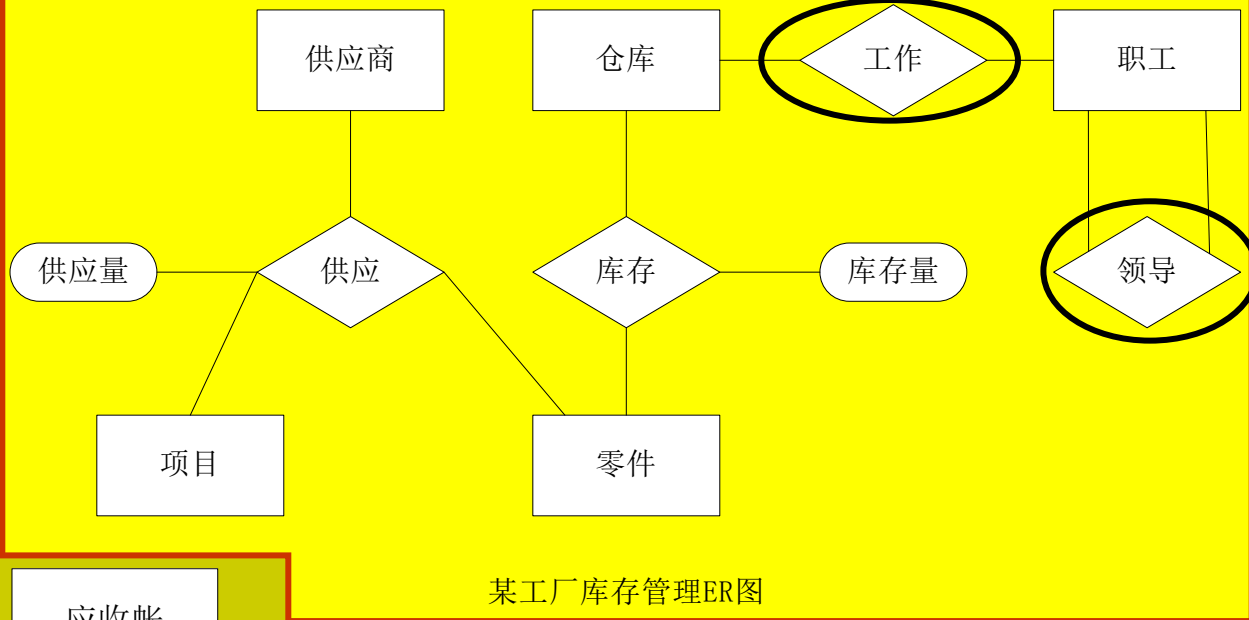
C) 消除冗余联系

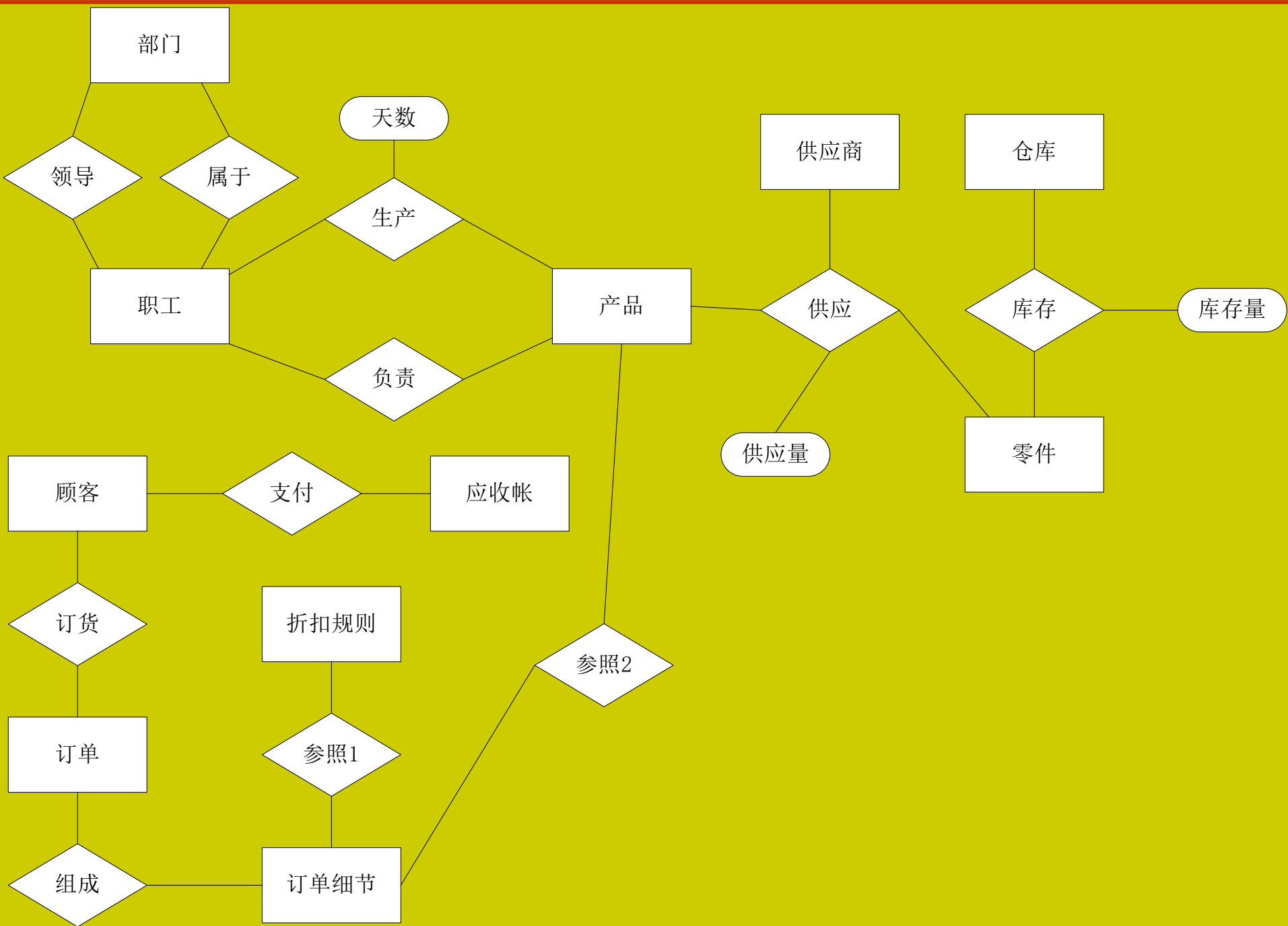


项目=产品

职工与仓库的联系冗余

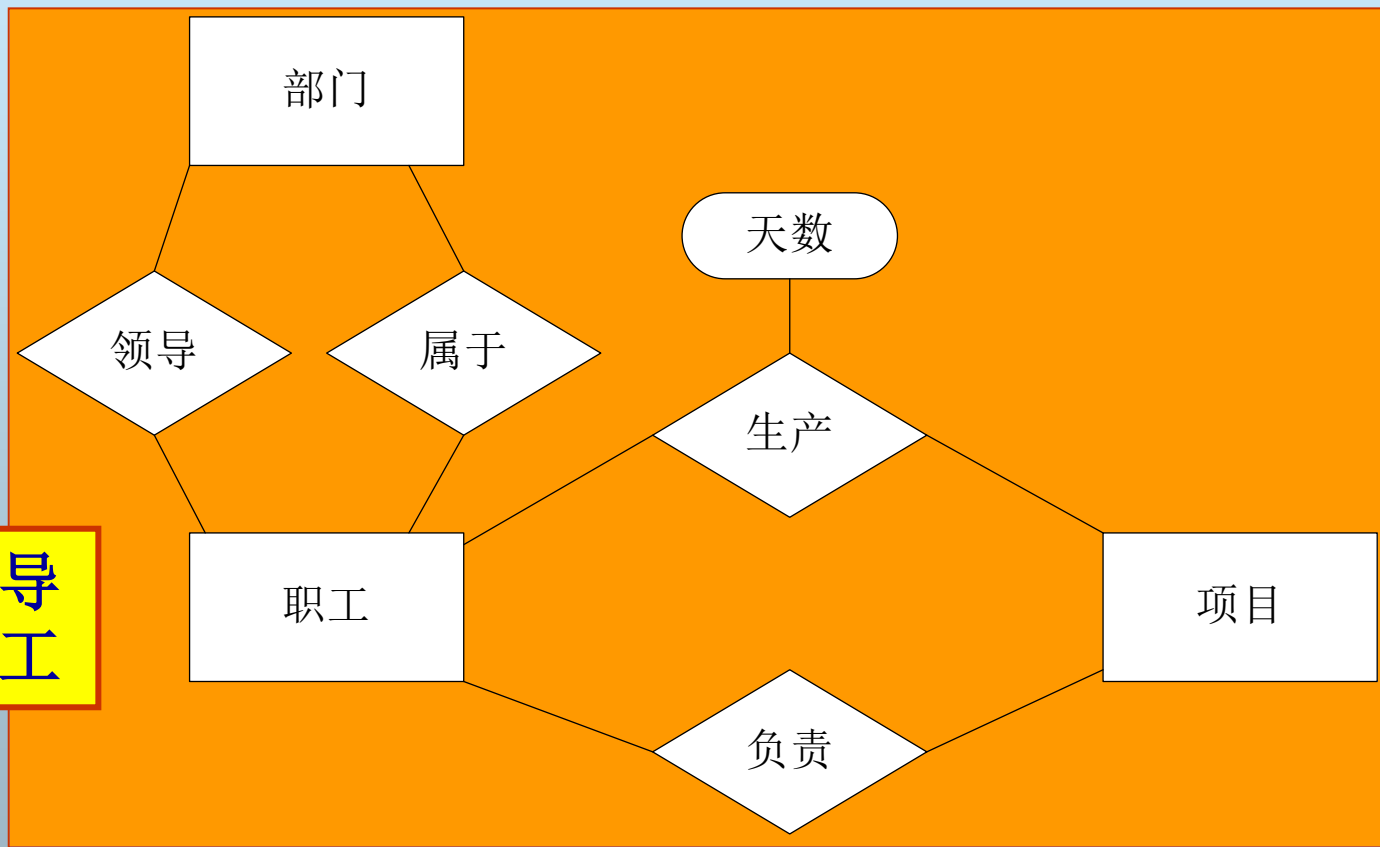
职工与职工的领导联系冗余





3、ER模型的扩展

- 传统的ER模型无法表达一些特殊的语义



无法区分领导者和一般职工

3、ER模型的扩展

- 弱实体
- 子类（特殊化）与超类（一般化）

(1) 弱实体 (weak entity)

- 一个弱实体的存在必须以另一实体的存在为前提
 - 弱实体所依赖存在的实体称为**常规实体** (regular entity) 或**强实体** (strong entity)
 - 弱实体有自己的标识，但它的标识只保证对于所依赖的强实体而言是唯一的。在整个系统中没有自己唯一的实体标识

(1) 弱实体 (weak entity)

■ 弱实体的例子

- 一个公司的人事系统中，需要管理职工和职工子女信息
- 子女是弱实体，职工是强实体
- 是否弱实体要看具体应用：例如在社区人口管理系统中，子女就不是弱实体，即使双亲都不存在了，子女仍应存在于人口系统中

(2) 弱实体的表示



- 弱实体用双线矩形表示，存在依赖联系用双线菱形表示，箭头指向强实体

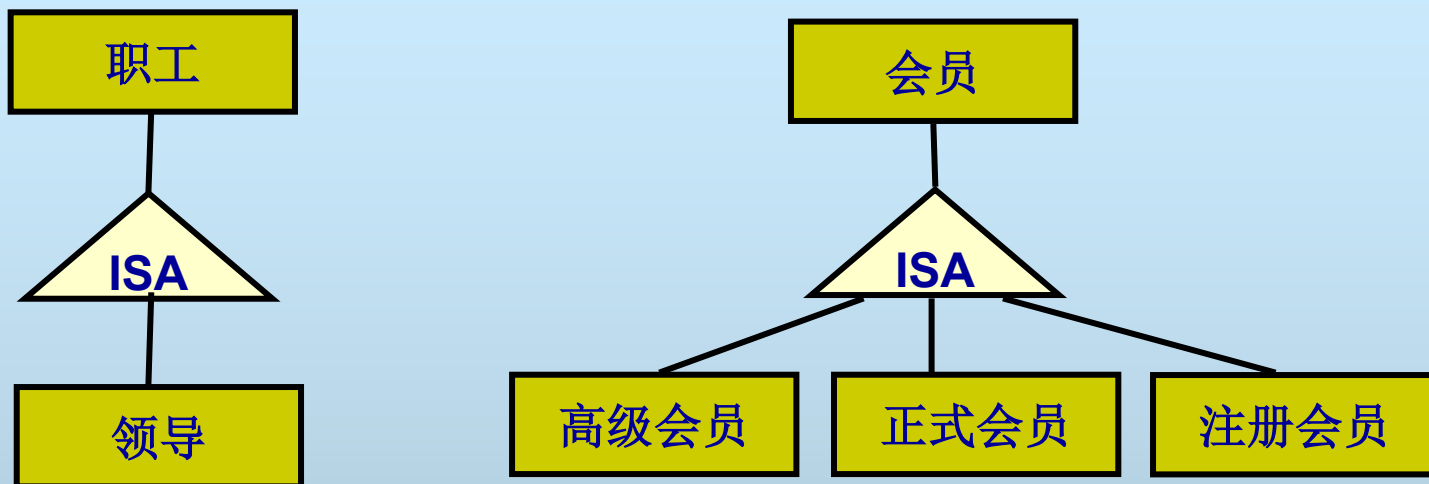
(3) 子类（特殊化）与超类（一般化）

■ 子类（Subtype）和超类（Supertype）

- 两个实体A和B并不相同，但实体A属于实体B，则A称为实体子类，B称为实体超类
- 子类是超类的特殊化，超类是子类的一般化
- 子类继承了超类的全部属性，因此子类的标识就是超类的标识
- 例如，研究生是学生的子类，经理是职工的子类

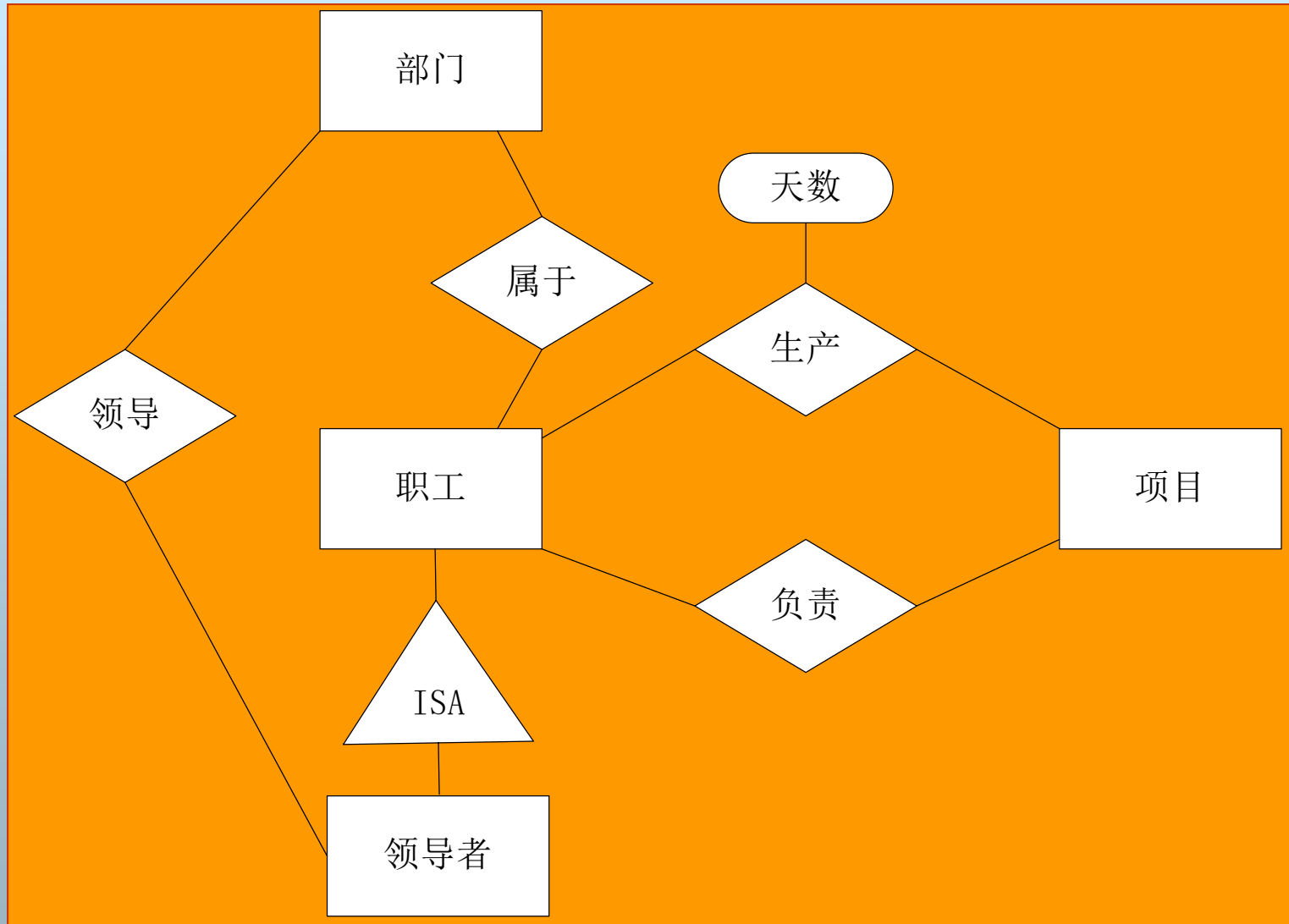
■ 在ER设计时，可以根据实际情况增加子类，也可以根据若干实体抽象出超类

(4) 子类符号

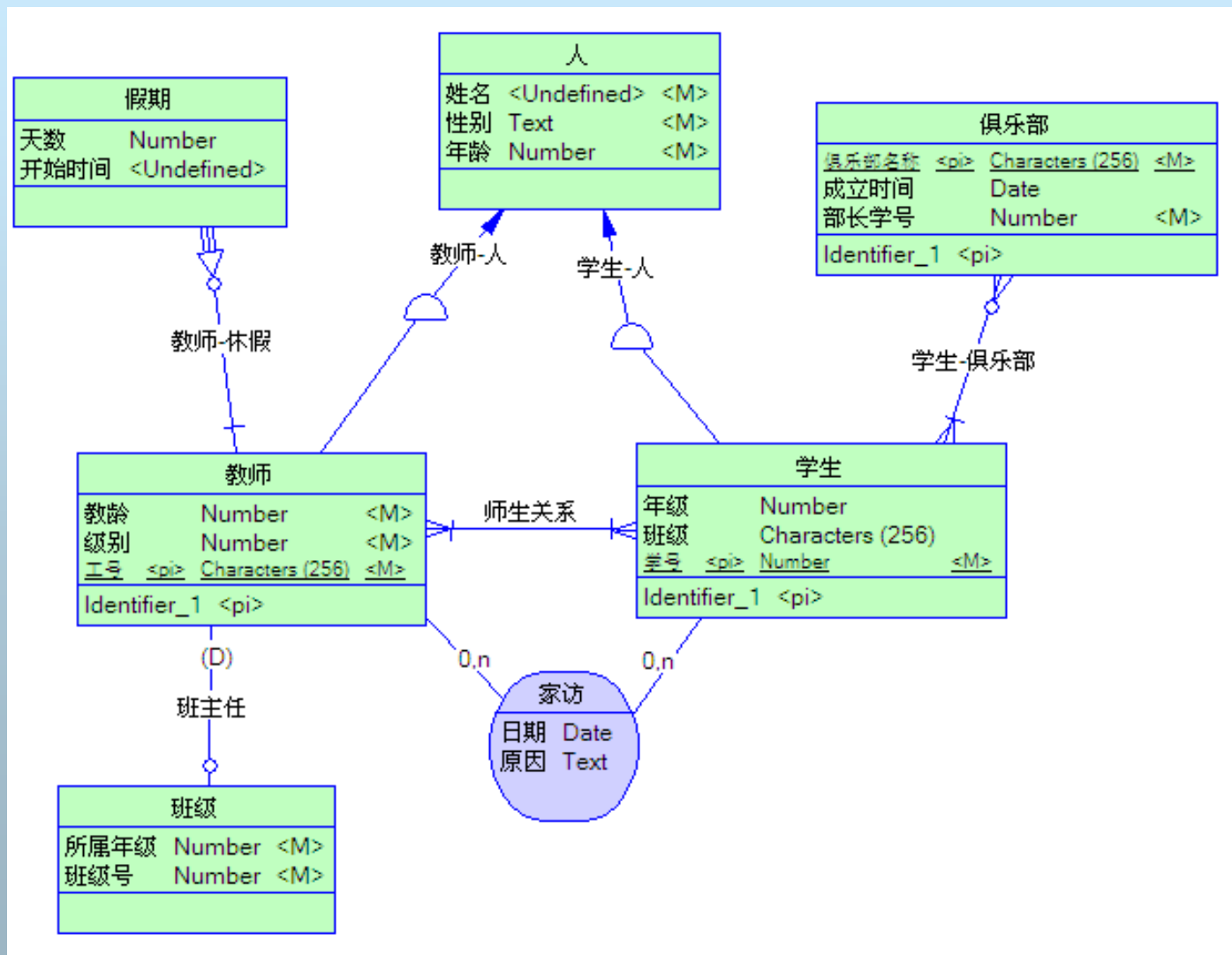


■ ISA表示子类与超类关系

(5) 子类例子



Power Designer中的符号



Power Designer中的符号

