

第7章 数据库设计

The background image is a serene winter scene. In the foreground, several bare, light-colored tree trunks stand on a snow-covered bank. Beyond them is a calm body of water, possibly a lake or a wide river, which reflects the golden light of the sun. The sun is positioned low on the horizon, creating a bright glow and long, soft shadows. The sky is filled with wispy clouds, and the overall color palette is dominated by the cool blues and whites of the winter, contrasted with the warm oranges and yellows of the low sun.

本章目标

理解数据库设计的生命周期

掌握需求分析的方法和步骤

熟练掌握E-R图建模及其将其转化为关系模式的方法

熟练掌握应用关系规范化理论优化关系模式的方法

理解并掌握不同索引技术及其应用方法

理解并掌握数据库实施和维护的基本内容



- 数据库设计概述
- 需求分析
- 概念结构设计
- 逻辑结构设计
- 物理结构设计
- 数据库的实施与维护
- 本章小结



■ 数据库设计

- 数据库设计是指对于一个给定的应用环境，构造(设计)优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户的应用需求，包括信息管理要求和数据操作要求。

- 信息管理要求：在数据库中应该存储和管理哪些数据对象。
- 数据操作要求：对数据对象需要进行哪些操作，如查询、增加、删除、修改、统计和分析等操作。

■ 数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效率的运行环境：

- 数据库数据的高存取效率
- 数据库存储空间的高利用率
- 数据库系统运行维护的高效率



- 数据库设计的特点
- 数据库设计方法
- 数据库设计的基本步骤
- 数据库设计过程中的各级模式



- 1. 重视基础数据
 - 三分技术，七分管理，十二分基础数据
 - 管理
 - 数据库设计作为一个大型工程项目本身的项目管理
 - 项目所属企业（即应用部门）的业务管理
 - 基础数据
 - 数据的收集、整理、组织和不断更新
- 2. 数据库设计和数据处理设计相结合
 - 将数据库模式设计和数据处理设计密切结合



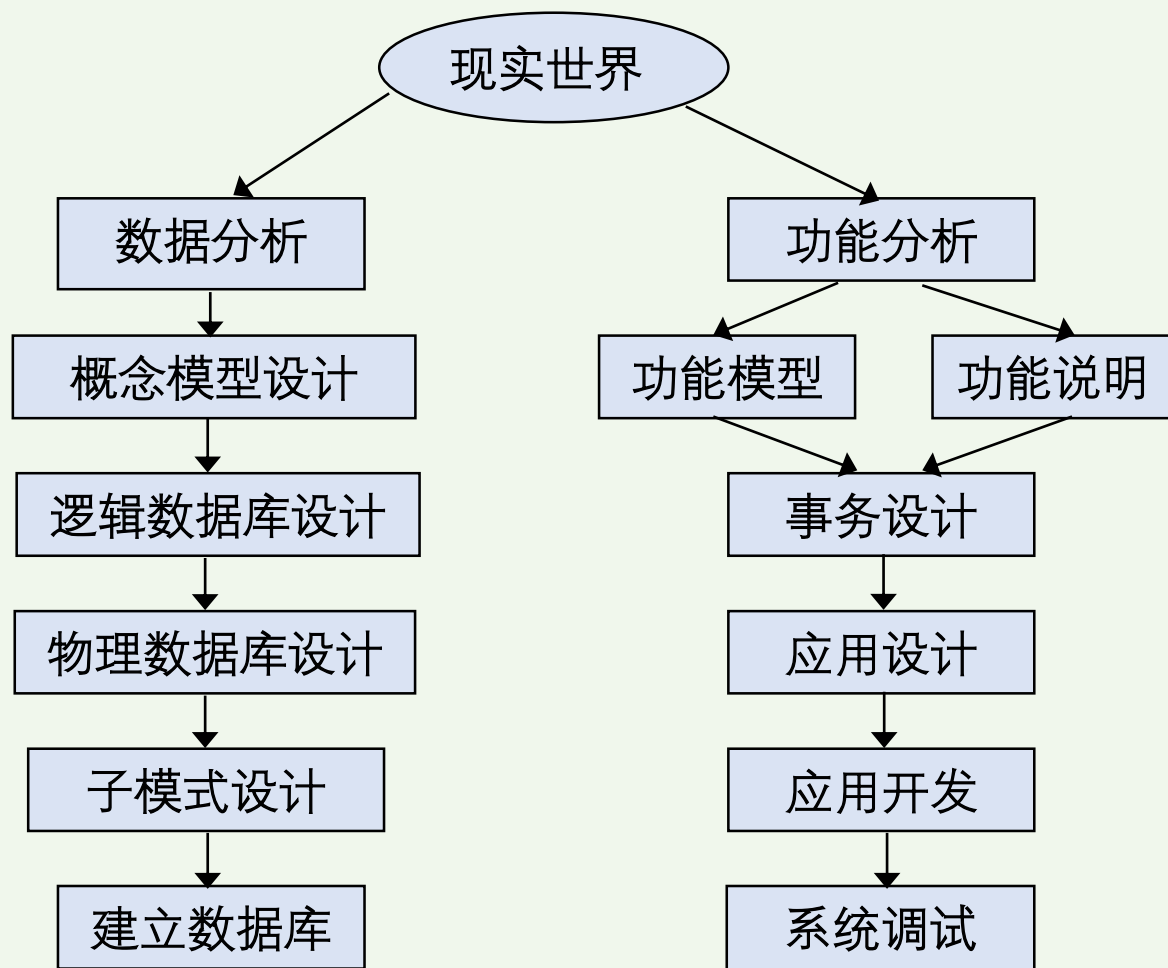


图7.1 数据库设计和应用系统设计分离的设计

■ 结构和行为分离的设计

– 传统的软件工程：

重 行为设计

- 忽视对应用中数据语义的分析和抽象，只要有可能就尽量推迟数据库结构设计的决策

– 早期的数据库设计：

重 结构设计

- 致力于数据模型和数据库建模方法研究，忽视了行为设计对结构设计的影响



- 大型数据库设计是涉及多学科的综合技术，又是一项庞大的工程项目
- 它要求多方面的知识和技术，主要包括：
 - 计算机的基础知识
 - 软件工程的原理和方法
 - 程序设计的方法和技巧
 - 数据库的基本知识
 - 数据库设计技术
 - 应用领域的知识



■ 手工与经验相结合的方法

- 设计质量与设计人员的经验和水平有直接关系
- 缺乏科学理论和工程方法支持，工程质量难以保证
- 数据库运行一段时间后，又不同程度发现各种问题，增加了维护代价

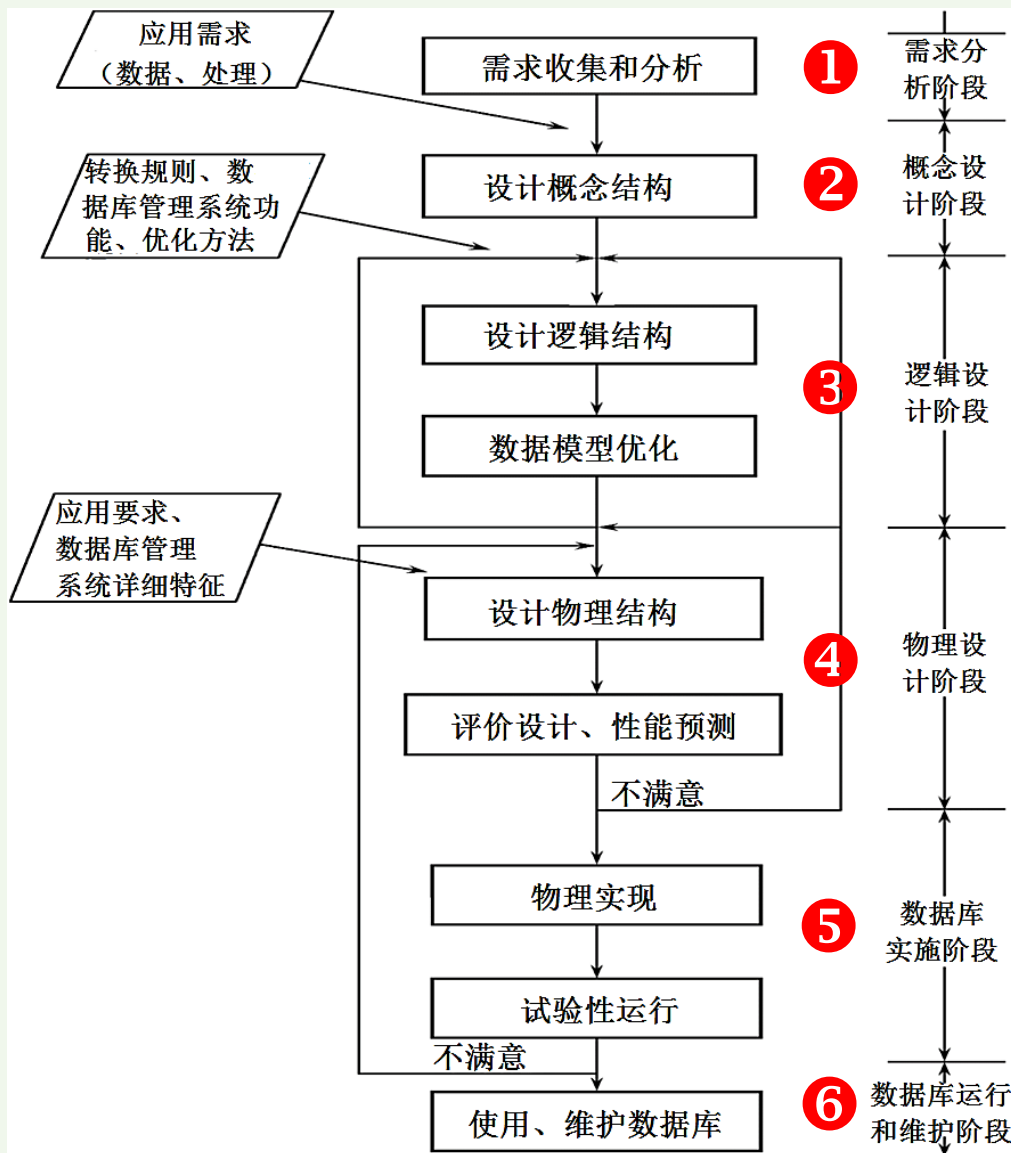
■ 规范的常用设计方法

- 新奥尔良(New Orleans)方法
- 基于E-R模型的数据库设计方法
- 3NF(第三范式)的设计方法
- 面向对象的数据库设计方法
- 统一建模语言(UML)方法

■ 为保证数据库系统的开发质量，提高开发效率，市场上有很多的数据库设计工具被普遍用于大型数据库的设计中

- 工具列表参考：<https://www.databasestar.com/data-modeling-tools/>





- 整个设计的**基础**
 - 是否做得充分与准确，决定了构建数据库的速度和质量，独立于具体DBMS
- 整个设计的**关键**
 - 通过对用户需求进行综合、归纳和抽象，形成一个独立于具体DBMS的概念模型
- 将概念结构转换为某个数据库管理系统所支持的数据模型，并对其进行优化
 - 与具体DBMS类型相关
- 为逻辑数据结构选取一个最适合应用环境的物理结构，与具体DBMS类型相关
 - 包括存储结构和存取方法
 - 由DBMS自动完成
- 设计人员运用DBMS提供的数据库语言及其宿主语言，根据逻辑设计和物理设计的结果建立数据库
 - 编写与调试应用程序，组织数据入库，试运行
 - 应用程序由程序员完成
- 数据库运行过程中的评估、调整与修改
 - 由DBA负责



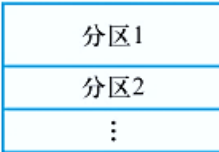
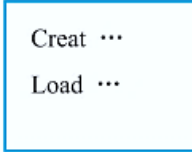


- 设计一个完善的数据库应用系统，往往是上述6个阶段的不断反复
 - 这个设计步骤既是数据库设计的过程，也包括了数据库应用系统的设计过程
 - 把数据库的设计和对数据库中数据处理的设计紧密结合起来，将这两个方面的需求分析、抽象、设计、实现在各个阶段同时进行，相互参照，相互补充，以完善两方面的设计
 - 参加数据库设计的人员
 - 系统分析员
 - 数据库设计人员
 - 应用开发人员
 - DBA
 - 用户代表
- 数据库设计的核心人员
 - 自始至终参与数据库设计，水平高低决定数据库系统的质量
 - 程序员：负责编写程序
 - 操作员：负责准备软硬件环境
 - 参与需求分析
 - 数据库的运行和维护



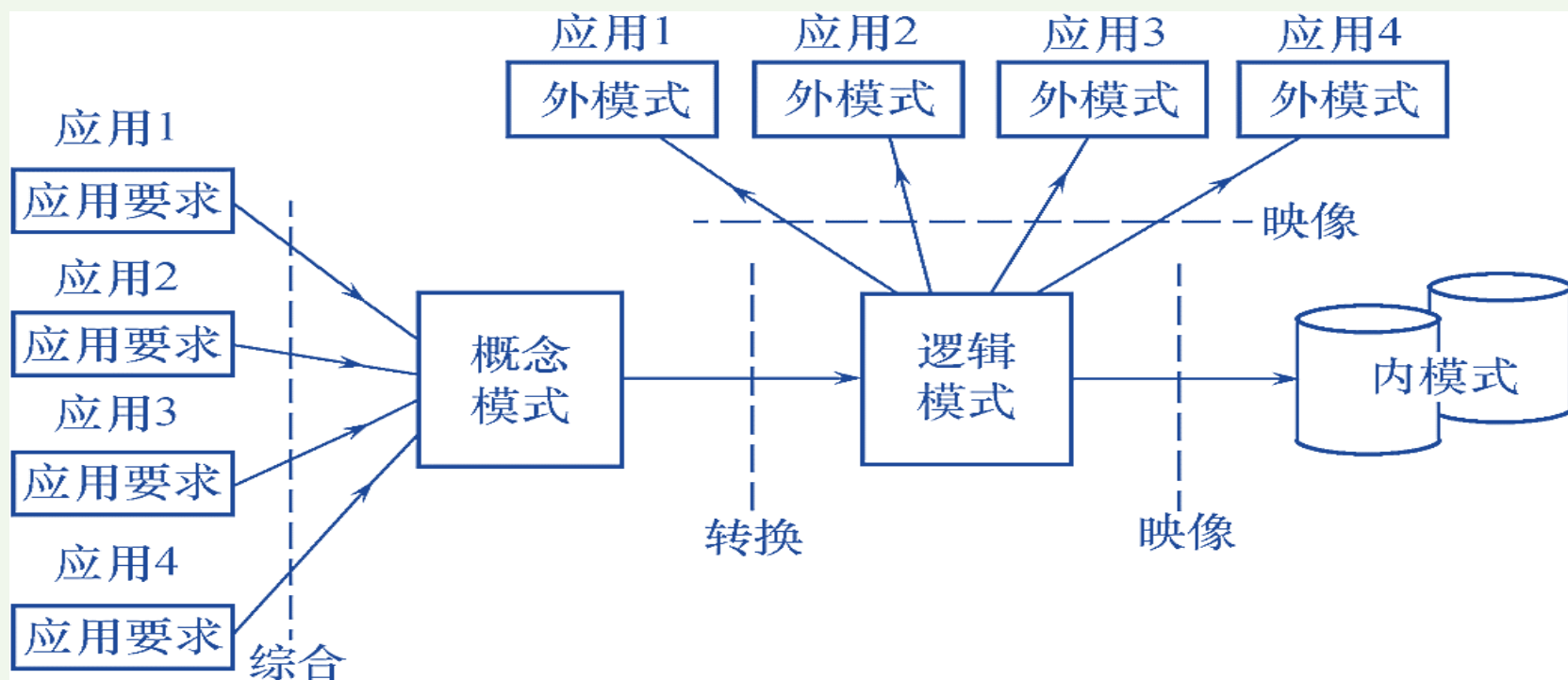
图 3

数据库设计各个阶段的数据设计描述

设计阶段	设计描述
需求分析	数据字典：全系统中数据项、数据结构、数据流、数据存储、处理过程的描述
概念结构设计	概念模型(E-R图) 
逻辑结构设计	某种数据模型 关系 非关系 
物理结构设计	存储布局 存取方法选择 存取路径建立 是否压缩存储 
数据库实施	创建数据库模式 装载数据 数据库试运行 
数据库运行和维护	性能监测、转储/恢复、数据库重组和重构



- 数据库设计不同阶段形成的数据库各级模式



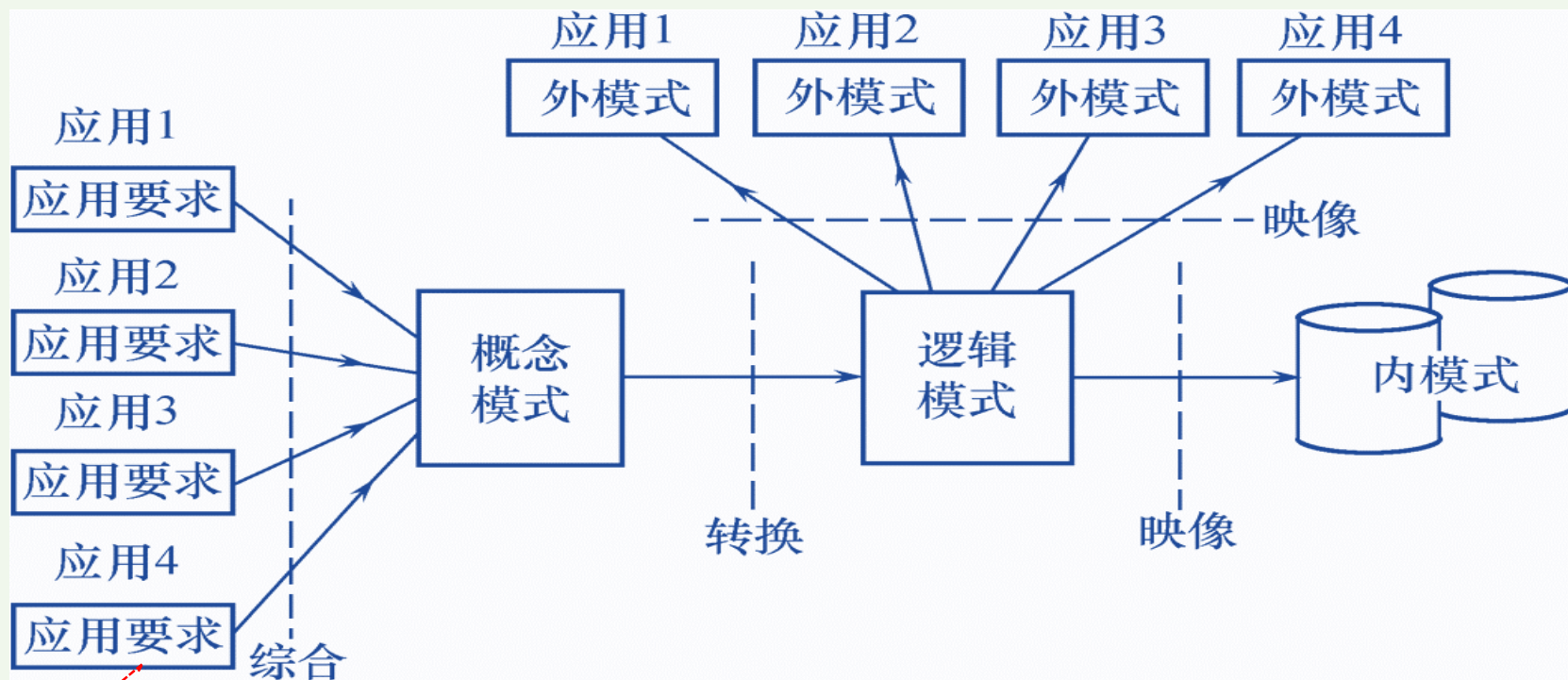
需求分析阶段

概念结构设计阶段

逻辑结构设计阶段

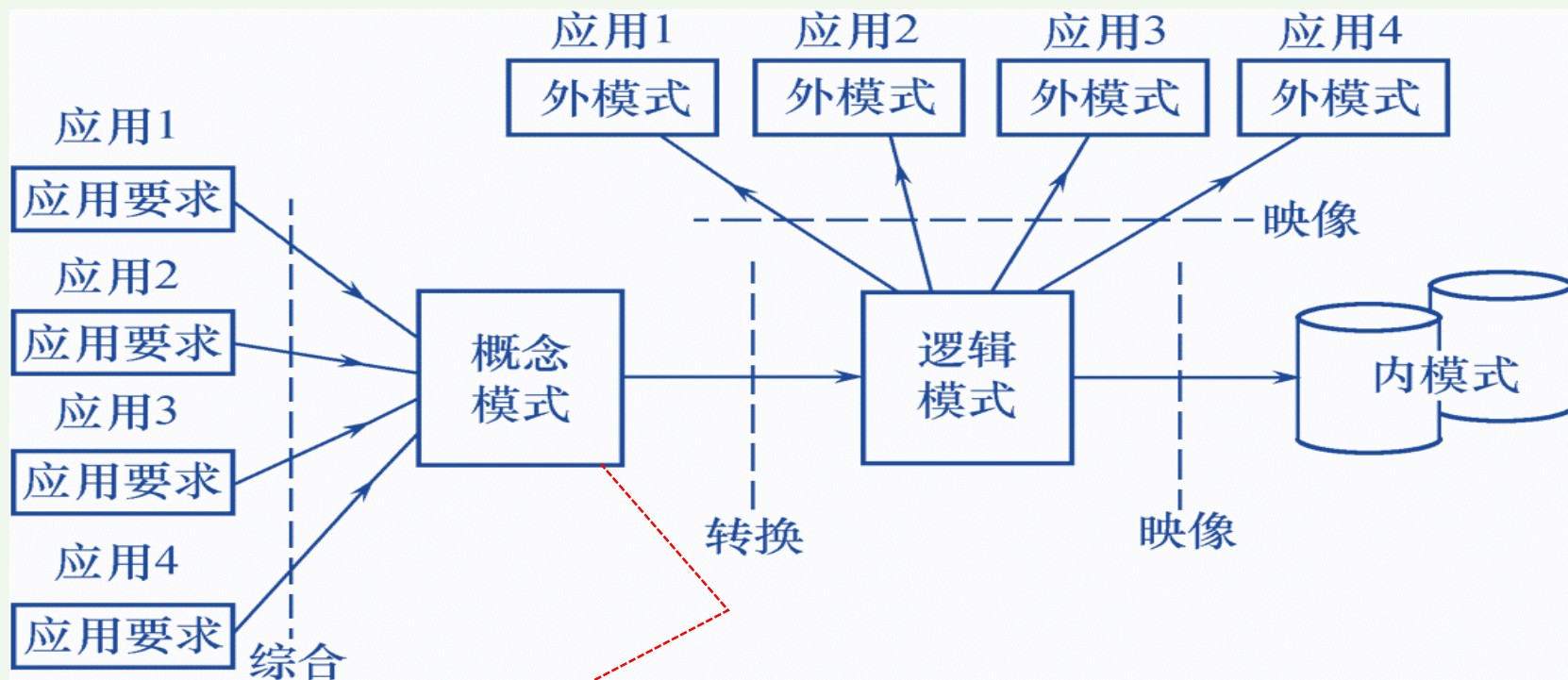
物理结构设计阶段





需求分析阶段:

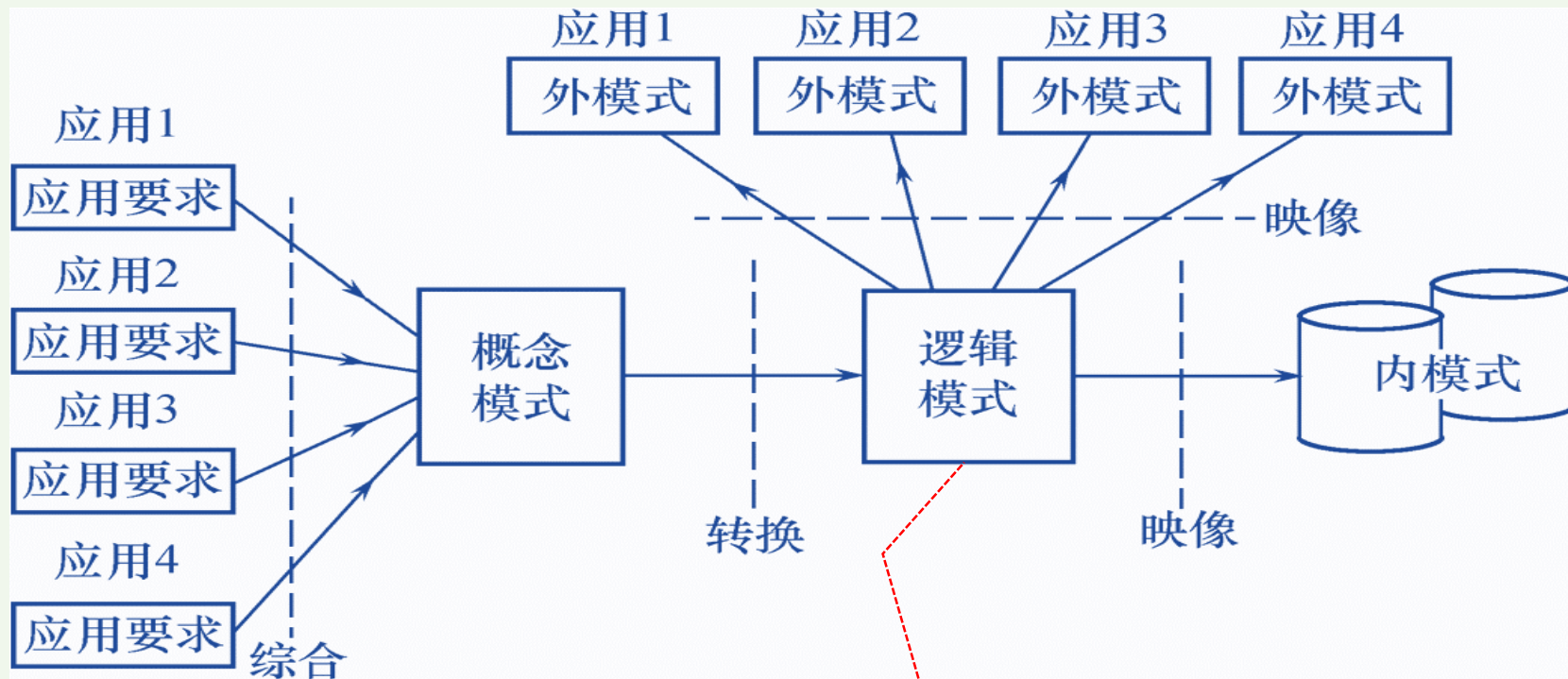
综合各个用户的应用需求



概念设计阶段:

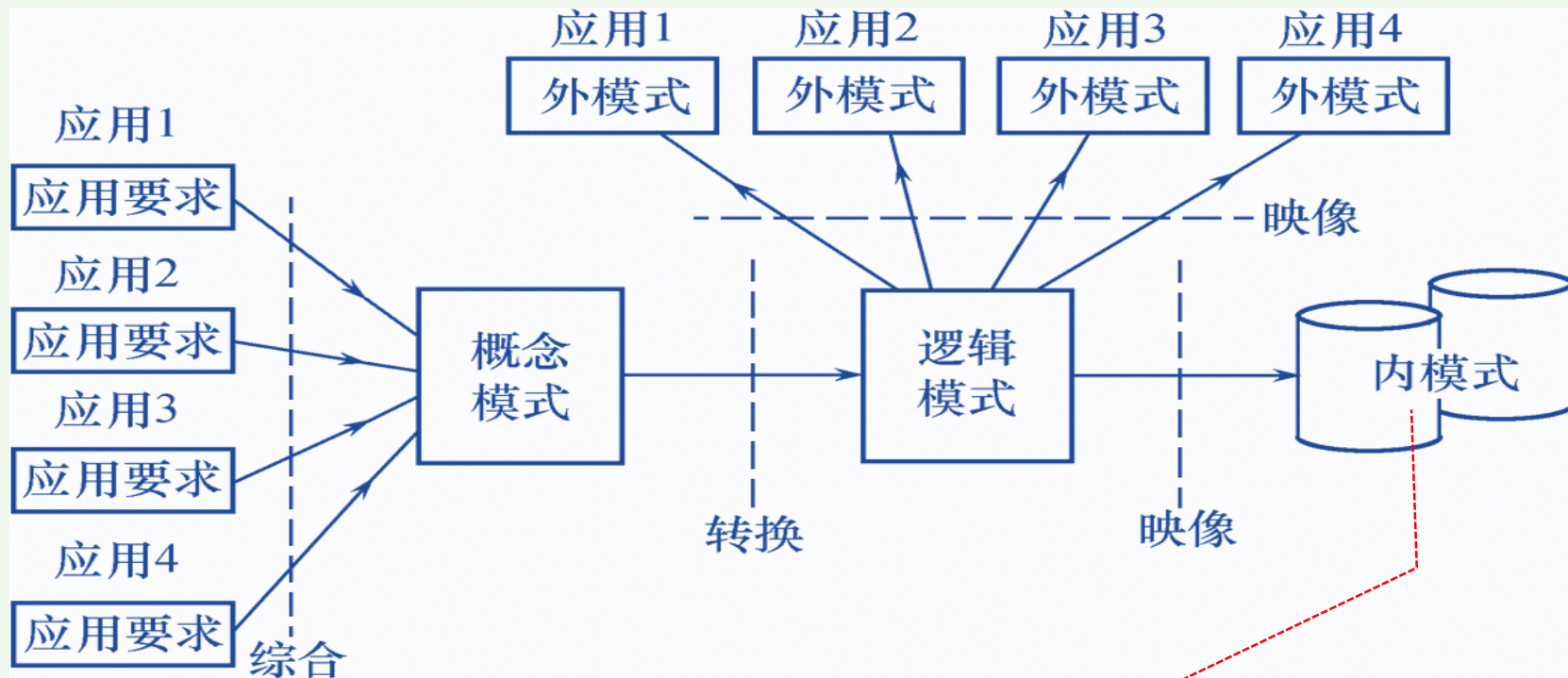
形成独立于机器特点, 独立于各个数据库管理系统产品的概念模式(E-R图)





逻辑设计阶段:

- 首先将E-R图转换成具体的数据库产品支持的数据模型, 如关系模型, 形成数据库逻辑模式
- 然后根据用户处理的要求、安全性的考虑, 在基本表的基础上再建立必要的视图(View), 形成数据的外模式



物理设计阶段:

- 根据数据库管理系统特点和处理的需要, 进行物理存储安排, 建立索引, 形成数据库内模式

- 需求分析就是分析用户的要求
 - 是设计数据库的起点
 - 需求分析结果是否准确地反映了用户的实际要求，将直接影响到后面各个阶段的设计，并影响到设计结果是否合理和实用
- 本节主要内容：
 - 需求分析的任务
 - 需求分析的方法
 - 数据字典



- 详细调查现实世界要处理的对象（组织、部门、企业等）
- 充分了解原系统（手工系统或计算机系统）工作概况
- 明确用户的各种需求
- 在此基础上确定新系统的功能
- 新系统必须充分考虑今后可能的扩充和改变
- 调查的重点是“数据”和“处理”，获得用户对数据库的要求
 - 信息要求
 - 用户需要从数据库中获得信息的内容与性质
 - 由信息要求可以导出数据要求，即在数据库中需要存储哪些数据
 - 处理要求
 - 用户要完成的处理功能
 - 对处理性能的要求
 - 安全性与完整性要求



- 确定用户最终需求的难点

- 用户缺少计算机知识，不能准确地表达自己的需求，他们所提出的需求往往不断地变化。
- 设计人员缺少用户的专业知识，不易理解用户真正需求，甚至误解用户需求

- 解决方法

- 设计人员必须不断深入地与用户进行交流，才能逐步确定用户的实际需求

- 敏捷开发方法





• 调查步骤:

- ① 调查组织机构情况
- ② 调查各部门的业务活动情况
- ③ 明确新系统的各项要求: 信息要求、处理要求、安全性与完整性要求 (调查重点)
- ④ 确定新系统边界: 哪些由计算机完成, 哪些由人工完成

• 调查方法:

- ① 跟班作业
- ② 开调查会
- ③ 请专人介绍
- ④ 询问
- ⑤ 设计调查表请用户填写
- ⑥ 查阅记录

• 调查完用户需要后, 还需进一步分析和表达用户的需求

• 结构化分析(SA)方法

- 简单实用
- 从最上层的系统组织机构入手
- 自顶向下、逐层分解方式

• 需求分析报告必须提交给用户, 征得用户的认可



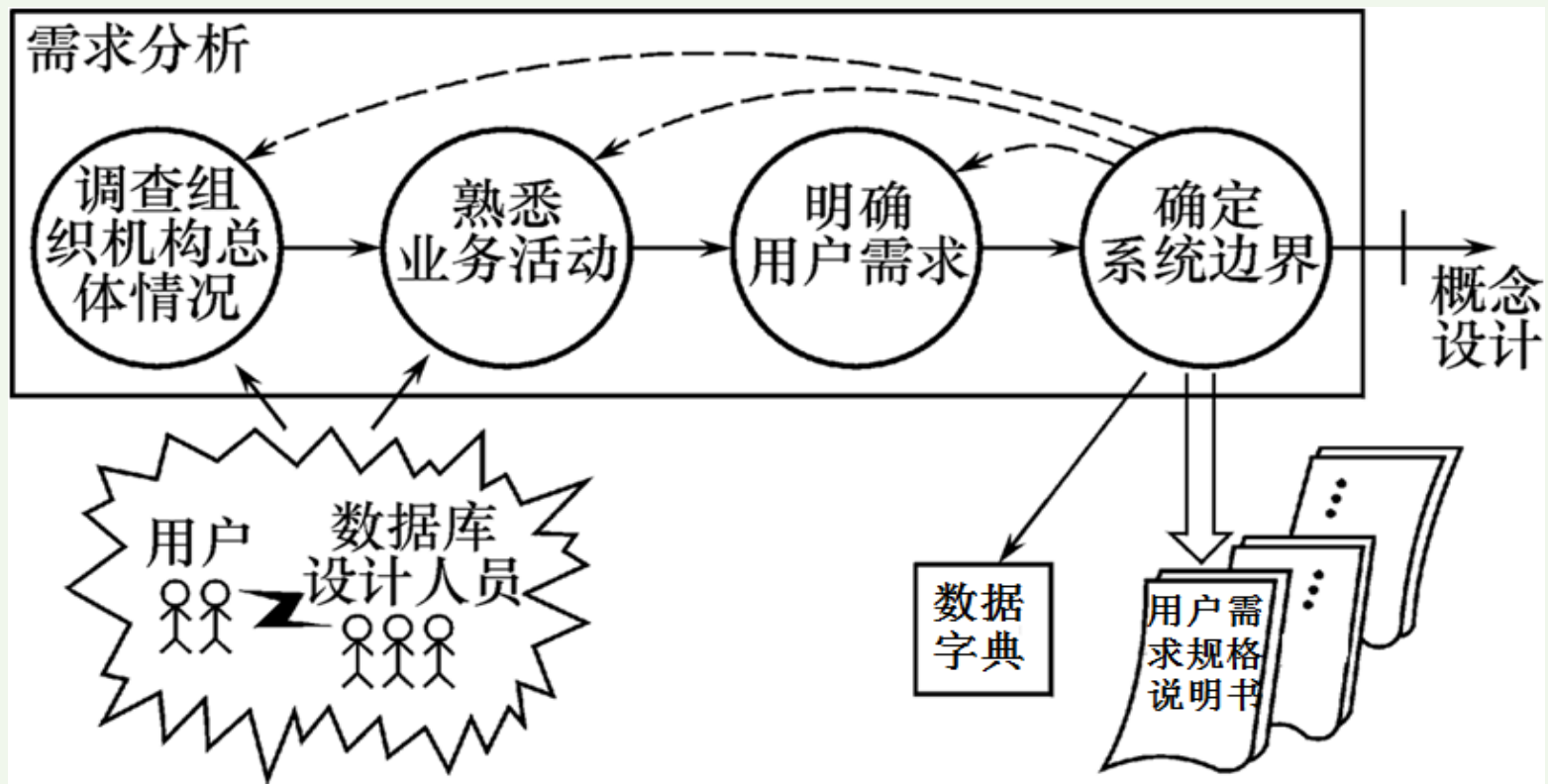


图7.5 需求分析过程

- 数据字典是关于数据库中数据的描述，即元数据，不是数据本身
- 数据字典在需求分析阶段建立，在数据库设计过程中不断修改、充实、完善
- 数据字典是进行详细的数据收集和数据分析所获得的主要成果
- 注意：和RDBMS中数据字典的区别和联系
- 数据字典的内容
 - 数据项(data item)
 - 数据结构
 - 数据流
 - 数据存储
 - 处理过程



- 数据项是不可再分的数据单位，是数据的最小组成单位

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, 与其他数据项的逻辑关系, 数据项之间的联系}

- “取值范围、与其他数据项的逻辑关系” 定义了数据的完整性约束条件，是设计数据检验功能的依据
- 数据项之间的关系可以用数据依赖的概念进行分析和表示
 - 按实际语义写出每个数据项之间的数据依赖，这些数据依赖是数据库逻辑设计阶段数据模型优化的依据



数据元素条目

名称：学号

别名：S-NO

说明：本校学生编码

数据值类型：(连续 / 离散)

类型：(字符 / 数字)

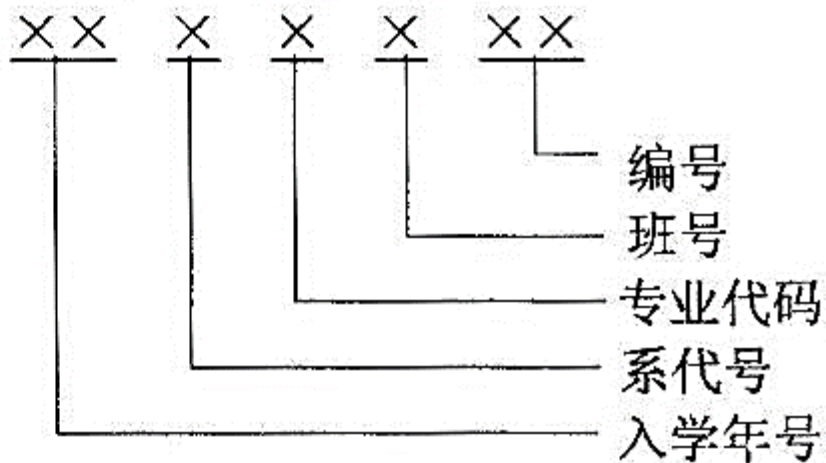
长度：7

有关数据结构：学生成绩，学生卡

总编号：1-101

编号：101

有关编码说明：



- 数据结构反映了数据项之间的组合关系
- 一个数据结构可以由若干个数据项组成，也可以由若干个数据结构组成，或由若干个数据项和数据结构混合组成
- 对数据结构的描述

数据结构描述 = { 数据结构名, 含义说明, 组成: {数据项或数据结构} }



数据结构条目

名称：学生登记卡

说明：新生入学时填写的卡片

结构：

学号

姓名

[曾用名]

入学日期

出生日期

性别

民族

家庭地址

本人简历*

开始时间

终止时间

单位

职务

总编号：2-03

编号：008

有关的数据流、数据存储：

新生登记表

学籍表

数量

每年约 1000 份



- 数据流是数据结构在系统内传输的路径

数据流描述={数据流名, 说明, 数据流来源, 数据流去向, 组成: { 数据结构}, 平均流量, 最高峰期流量}

- 数据流来源: 说明该数据流来自哪个过程
- 数据流去向: 说明该数据流将到哪个过程去
- 平均流量: 在单位时间 (每天、每周、每月等) 里的传输次数
- 高峰期流量: 在高峰时期的数据流量



数据流条目

名称：期末成绩单

总编号：3-05

简要说明：学期结束时，由任课教师填写的成绩单

编号：005

数据流来源：教师

数据流去向：P2.1、P2.2

包含的数据结构：

流通量：200 份 / 学期

科目名称

{ 考试
考查 }

学生成绩*

学号

姓名

成绩

任课教师

- 数据存储是数据结构停留或保存的地方，也是数据流的来源和去向之一
- 对数据存储的描述

数据存储描述={数据存储名, 说明, 编号, 输入的数据流, 输出的数据流, 组成: { 数据结构}, 数据量, 存取频度, 存取方式}

- 存取频度: 每小时、每天或每周存取次数及每次存取的数据量
- 存取方式: 批处理 / 联机处理, 检索 / 更新, 顺序检索 / 随机检索
- 输入的数据: 数据来源
- 输出的数据: 数据去向



数据存储条目

名称：学习成绩一览表

说明：学期结束，按班汇集学生各科成绩

结构：

班级

学生成绩*

学号

姓名

成绩*

科目名称

{ 考试
考查 }

成绩

总编号：4-02

编 号：D2

有关的数据流：

P2.1.1→D2

D2→P2.1.2

D2→P2.1.4

D2→P2.1.3

D2→P2.1.5

信息量：150 份 / 学期

有无立即查询：有



- 处理过程的具体处理逻辑一般用判定表或判定树来描述
- 数据字典中只需要描述处理过程的说明性信息
- 处理过程说明性信息的描述

处理过程描述={ 处理过程名, 说明, 输入: {数据流}, 输出: {数据流}, 处理: {简要说明}}

- 简要说明: 说明该处理过程的功能及处理要求
 - 功能: 该处理过程用来做什么
 - 处理要求: 处理频度要求, 如单位时间里处理多少事务, 多少数据量、响应时间要求等
 - 处理要求是后面物理设计的输入及性能评价的标准



处理功能条目

名称：填写成绩单

总编号：5-007

说明：通知学生成绩，有补考科目的说明补考日期

编号：P2.1.4

输入：D2→P2.1.5

输出：P2.1.5→学生(成绩通知单)

处理：查 D2(成绩一览表)，打印每个学生的成绩通知单，若有不及格科目，不够直接留级，则在“成绩通知”中填写补考科目、时间，若直接留级则注明留级。



- 把需求收集和分析作为数据库设计的第一阶段是十分重要的
- 第一阶段收集的基础数据(用数据字典来表达)是下一步进行概念设计的基础
- 强调两点
 - 设计人员应充分考虑到可能的扩充和改变, 使设计易于更改, 系统易于扩充
 - 必须强调用户的参与
 - 用户的参与是数据库设计不可分割的一部分
 - 任何调查研究没有用户的积极参与都是寸步难行的
 - 设计人员帮助用户建立数据库环境下的共同概念, 对设计的最终结果承担共同责任



- 将需求分析得到的用户需求抽象为信息结构(即概念模型)的过程就是**概念结构设计**
- 本节主要内容：
 - **概念模型**
 - **E-R模型**
 - **扩展的E-R模型**
 - **用UML中的类图表示E-R图**
 - **用E-R图进行概念结构设计**



■ 概念模型的特点：

- 能真实、充分地反映现实世界，是现实世界的一个真实模型
- 易于理解，从而可以用它和不熟悉计算机的用户交换意见
- 易于更改，当应用环境和应用要求改变时，容易对概念模型修改和扩充
- 易于向关系、网状、层次等各种数据模型转换

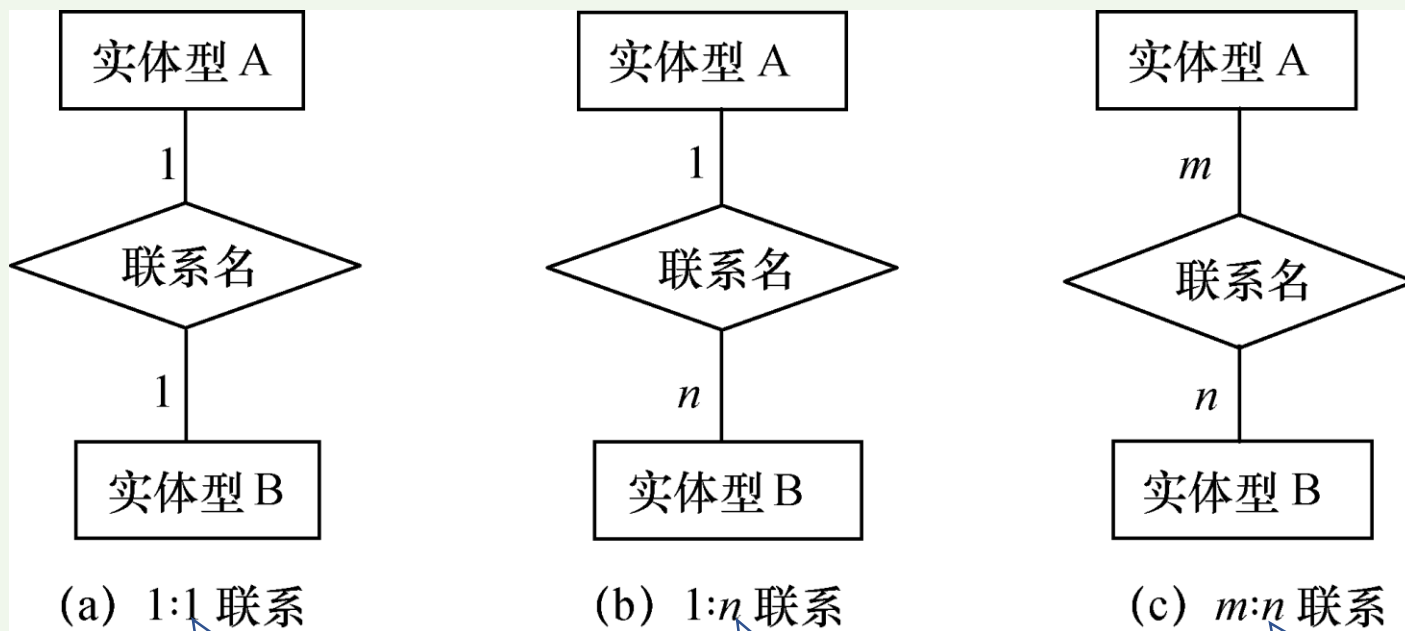
■ 描述工具

- E-R模型(Model)/图(Diagram)/方法(Approach)



1. 两个实体型之间的联系

– 1:1联系; 1:n联系; m:n联系

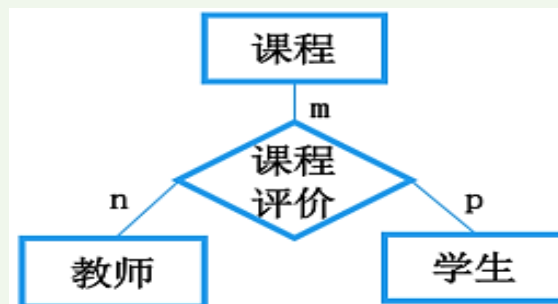


例: 学院与院长

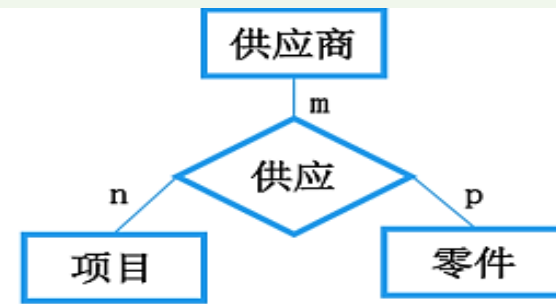
例: 学院与系

例: 学生与课程的选课

- 2.两个以上实体型之间的联系
 - 1:1联系; 1:n联系; m:n联系



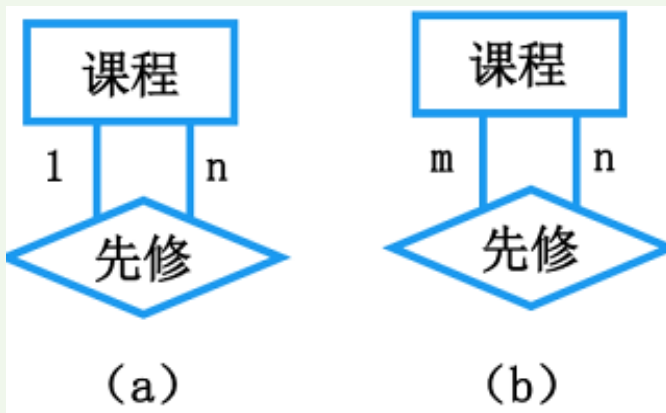
(a)



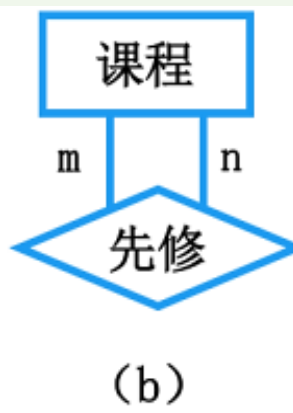
(b)

- 3.单个实体型内的联系

- 1:1联系; 1:n联系; m:n联系



(a)



(b)

- 第2章中我们假设一门课只列出直接先修课，每门课可以是多门课程的直接先修课，所以课程内部的“先修”是一对多的联系，如图7.8(a)所示

- 实际上一门课程也可以有多门直接先修课，某一门课程也可以作为多门课程的先修课，这时课程内部的“先修”是多对多的联系，如图7.8(b)所示

- 联系的度：参与联系的实体型的数目称为联系的度（二元联系；三元联系；N元联系）



■ E-R图提供了表示实体型、属性和联系的方法：

- 实体型用矩形表示
- 属性用椭圆表示
- 联系用菱形表示



图7.9 学生实体属性图

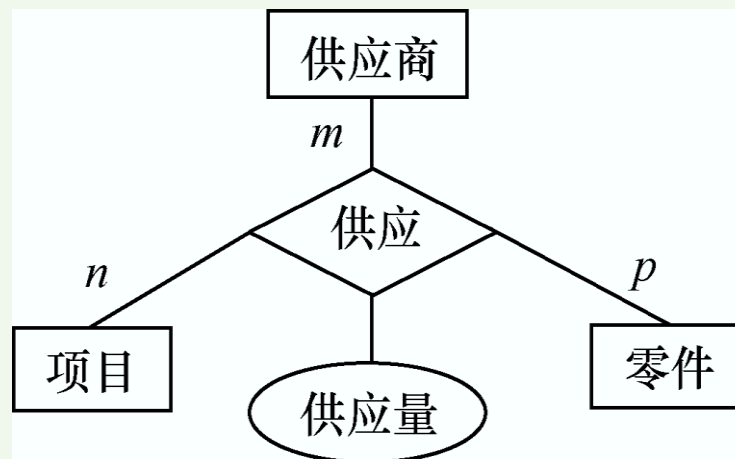


图7.10 实体型及联系E-R图

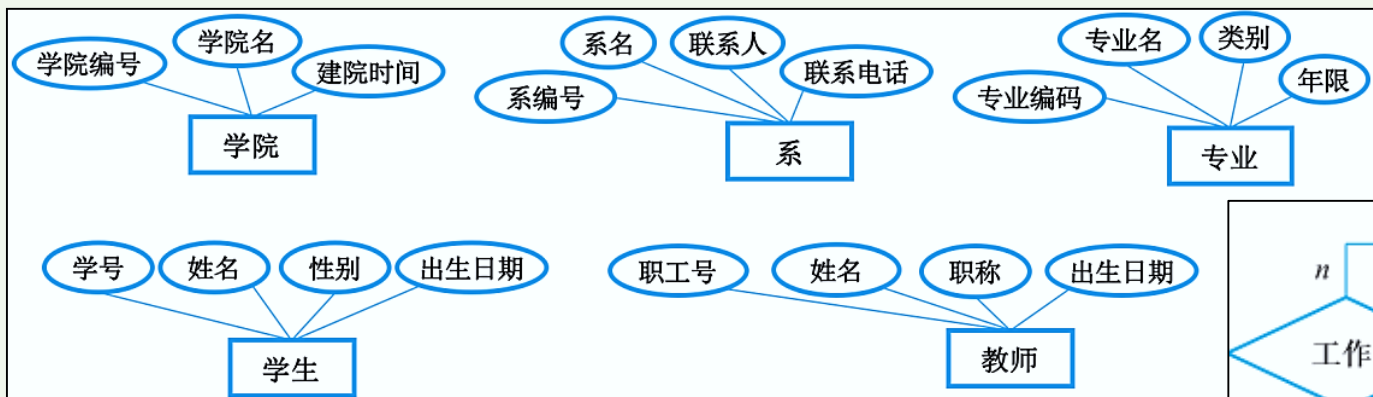
■ 设学生学籍管理涉及以下5类实体：

- 学院：学院编号、学院名、建院时间
- 系：系编号、系名、联系人、联系方式
- 专业：专业编码、专业名、类别、年限
- 学生：学号、姓名、性别、出生日期
- 教师：职工号、姓名、职称、出生日期

■ 这些实体型之间的联系如下：

- 一个学院可以设置多个系，一个系只能归属一个学院；一个学院只有一个教师任职院长，一个教师只在一个学院中任职院长。因此，学院和系之间具有一对多联系，学院与教师之间就担任院长关联具有一对一联系
- 一个系可以设置多个专业，一个专业只能归属一个系；一个系只能由一个教师担任系主任，一个教师只能担任一个系的系主任。因此，系和专业之间具有一对多联系，系与教师之间就任职系主任关联具有一对一联系
- 一个教师只能在一个系工作，一个系由多位教师构成。系和教师之间构成一对多联系
- 一个学生只属于一个学院，一个学院有多位学生，学院和学生之间构成一对多联系
- 一个专业同时有若干个学生选择，一个学生可以选择一个专业作为主修，另外若干个专业作为辅修。因此，专业和学生之间具有多对多联系





(a) 实体属性图

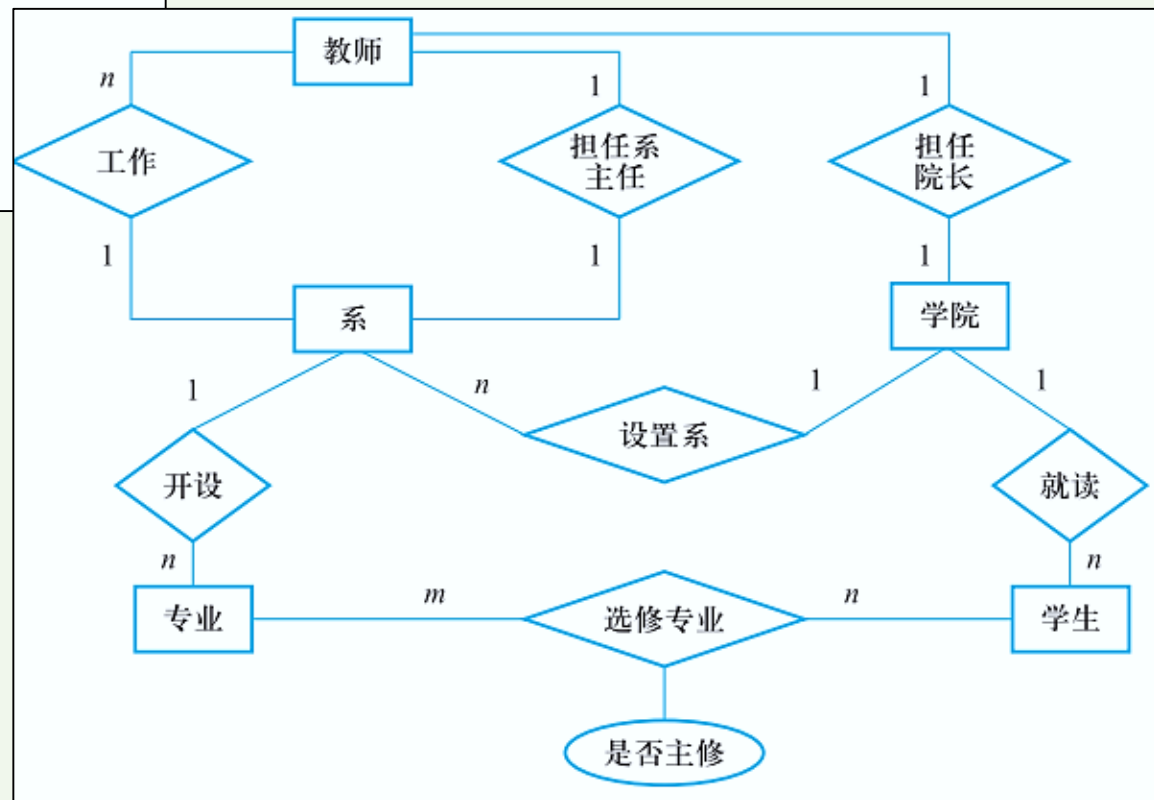


图7.11 “学生学籍管理”子系统的分E-R图



- ISA联系
- 基数约束
- Part-of联系
- 独占联系



■ ISA联系

- 某些实体型是某个实体型的子类型, 这种父类-子类联系称为**ISA**联系, 表示 “**is a**”语义
- **ISA**联系的重要性质: 子类继承了父类的所有属性, 子类也可以有自己的属性

■ (1)分类属性

- 分类属性是父实体型的一个属性
- 分类属性的值把父实体型中的实体分派到子实体型中

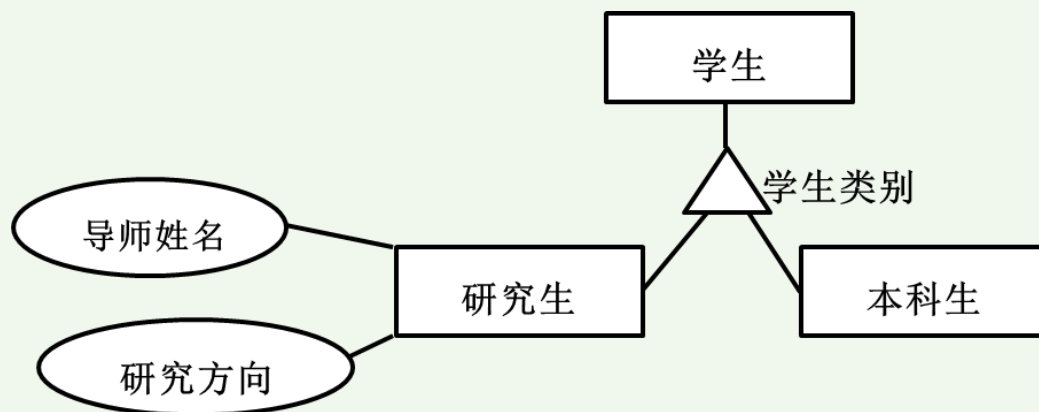


图7.12 学生的2个子类和分类属性



■ (2) 不相交约束与可重叠约束

- 不相交约束：描述父类中的一个实体不能同时属于多个子类中的实体集。即一个父类中的实体最多属于一个子类实体集，用ISA联系符号三角形的一个叉号“X”来表示
- 可重叠约束：父类中的一个实体能同时属于多个子类中的实体集。子类符号中没有叉号表示是可重叠的

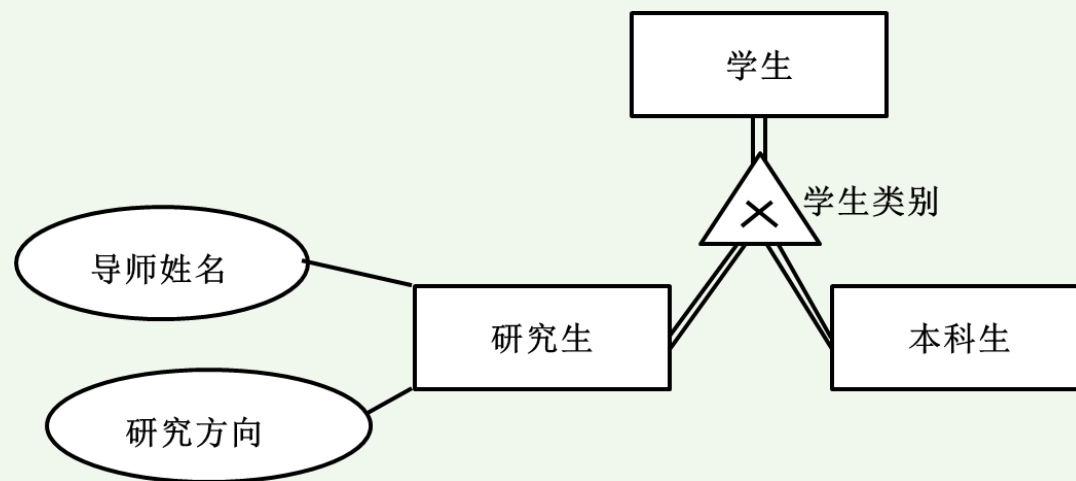


图7.13 表明一个学生不能既是本科生又是研究生

▪ (3) 完备性约束

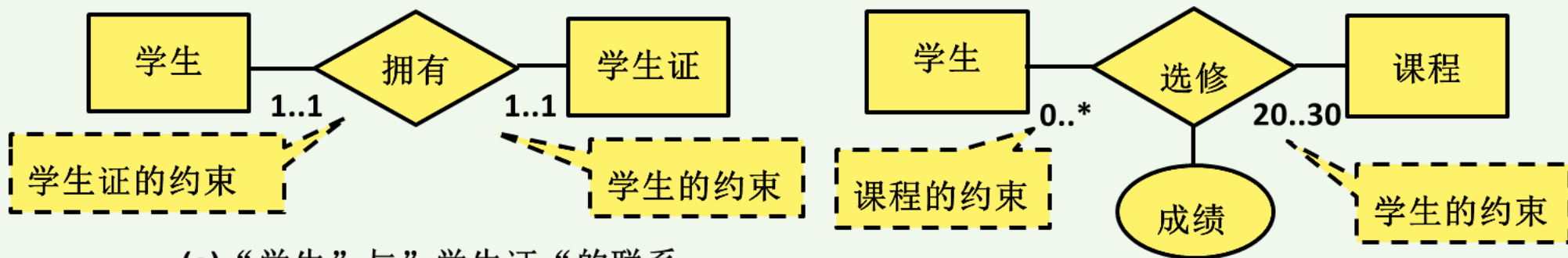
- 描述父类中的一个实体是否必须是某一个子类中的实体
 - 如果是，则叫做完全特化 (**full specialization**)
 - 否则叫做部分特化 (**partial specialization**)
- 完全特化用父类到子类的**双线连接**来表示
- 部分特化用父类到子类的**单线连接**来表示



■ 基数约束

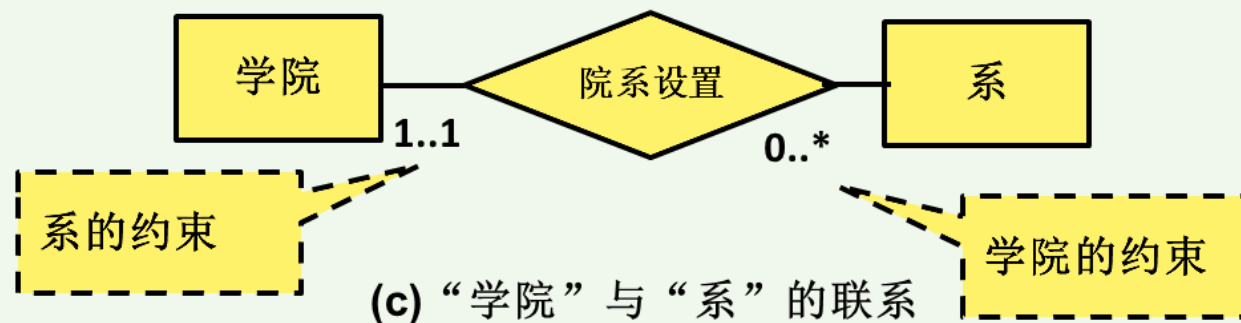
- 对实体之间一对一、一对多、多对多联系的细化
- 参与联系的每个实体型用基数约束来说明实体型中的任何一个实体可以在联系中出现的**最少次数**和**最多次数**
- 约束用一个数对**min..max**表示, $0 \leq \min \leq \max$ 。例如, 0..1, 1..3, 1..*, 其中, *代表无穷大
- **min=1**的约束叫做**强制参与约束**, 即被施加基数约束的实体型中的每个实体都要参与联系
- **min=0**的约束叫做**非强制参与约束**, 被施加基数约束的实体型中的实体可以出现在联系中, 也可以不出现在联系中





(a) “学生”与“学生证”的联系

(b) “学生”与“课程”的联系



(c) “学院”与“系”的联系

图7.14 基数约束示例

■ Part-of联系

- 表明某个实体型是另外一个实体型的一部分
- **Part-of**联系可以分为两种情况
 - 一种是整体实体如果被破坏，部分实体仍然可以独立存在，称为**非独占的Part-of联系**
 - 另一种是整体实体如果被破坏，部分实体不能存在，称为**独占的Part-of联系**，简称**独占联系**

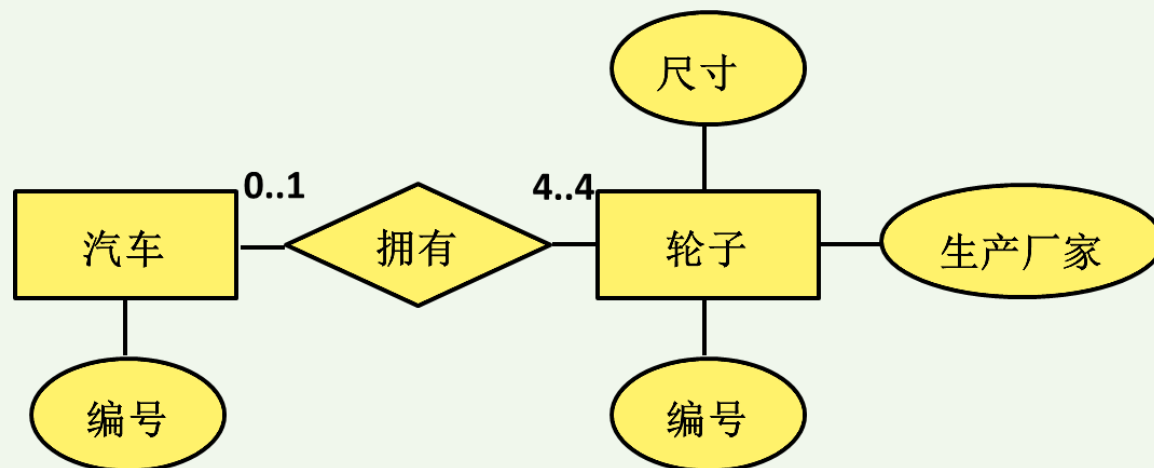


图7.15 用非强制参与联系表示非独占Part-of联系



■ 独占联系

- 如果一个实体型的存在依赖于其它实体型的存在，则这个实体型叫做**弱实体型**，否则叫做**强实体型**
- 一般地，如果不能从一个实体型的属性中找出可以作为码的属性，则这个实体型是弱实体型
- 在**E-R图**中用**双矩形表示弱实体型**，用**双菱形表示识别联系**

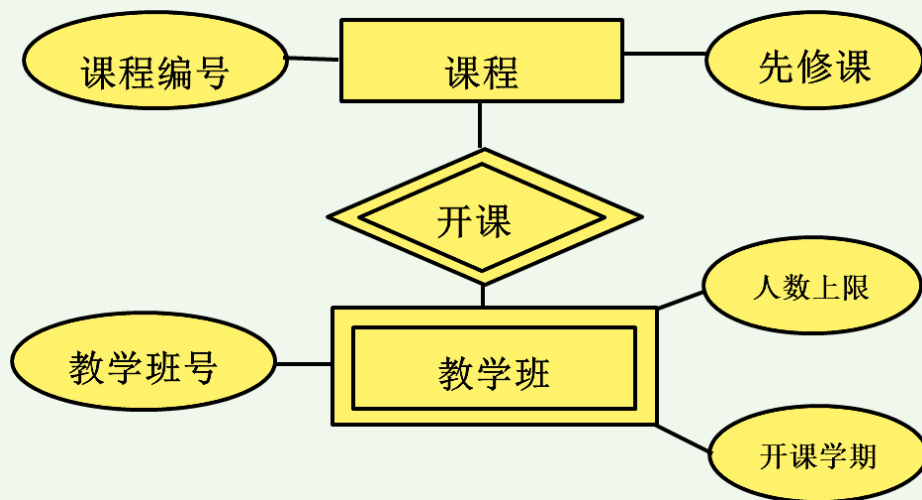


图7.16 弱实体型和识别联系

- 教学班是一个弱实体型，它只有“教学班号”“人数上限”“开课学期”三个属性。这些属性的任何组合都不能作为教学班的码。教学班的存在必须依赖于课程，没有课程自然没有教学班。



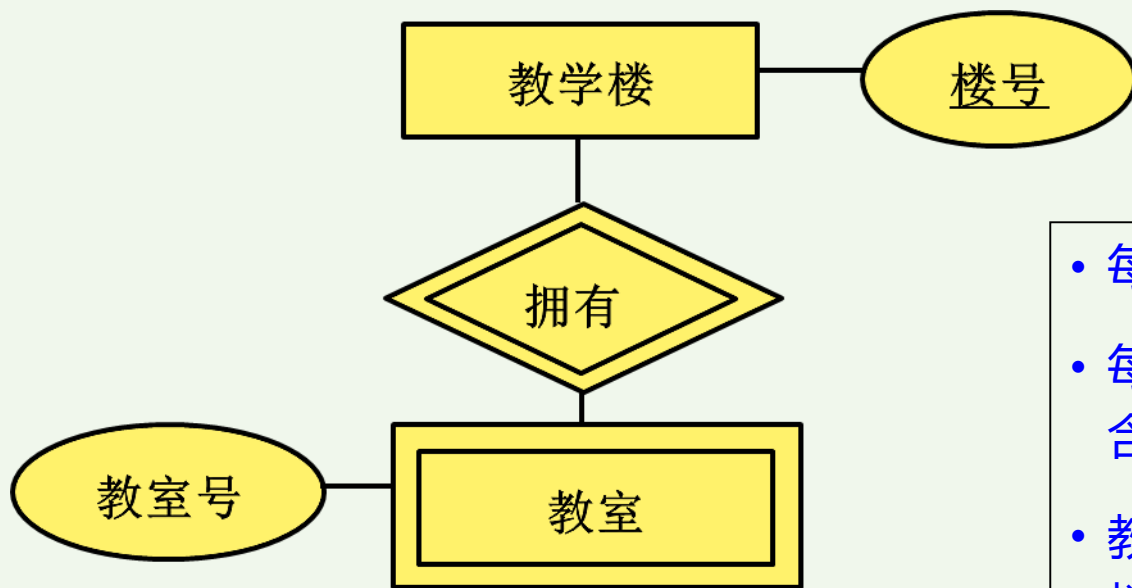


图7.17 教室是一个弱实体

- 每座教学楼都有唯一的编号或者名称
- 每个教室都有一个编号，教室号不包含楼号
- 教室号不能作为码，因为不同的教学楼中可能有编号相同的房间
- 教室是一个弱实体

- **UML称为统一建模语言**(Unified Modeling Language), 是对象管理组织(Object Management Group, OMG)的一个标准
- **UML**是为软件开发的所有阶段提供模型化和可视化支持的规范语言, 从需求规格描述到系统完成后的测试和维护
- **UML**可以用于数据建模, 业务建模, 对象建模, 组件建模等
- **UML**提供了多种类型的模型描述图(**diagram**), 借助这些图可以使应用程序更易理解



■ UML中的类(class)大致对应E-R图中的实体

- 实体型：用类表示，矩形框中实体名放在上部，下面列出属性名
- 实体的码：在类图中在属性后面加“PK” (primary key)来表示主码属性
- 联系：用类图之间的“关联”来表示
- 基数约束
 - UML中关联类之间的基数约束的概念、表示和E-R图中的基数约束类似
- UML中的子类
 - 面向对象技术支持超类-子类概念，子类可以继承超类的属性，也可以有自己的属性
 - 这些概念和E-R图的父类-子类联系、ISA联系是一致的



图7.18 用UML的类图表示E-R图示例



- 本节主要内容:

- 1. 实体与属性的划分原则

- 2. E-R图的集成



- 为简化E-R图的处置，现实世界的事物能作为属性对待的，尽量作为属性对待
- 两条准则：
 - 属性不能再具有需要描述的性质。即属性必须是不可分的数据项，不能包含其他属性
 - 属性不能与其他实体具有联系，即E-R图中所表示的联系是实体之间的联系

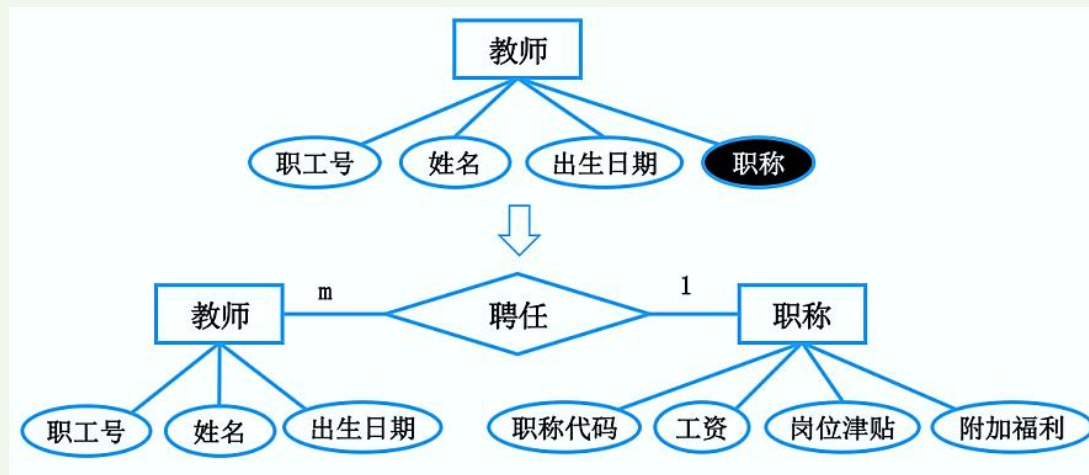


图7.19 “职称” 作为一类实体

- 教师是一个实体型，职工号、姓名、出生日期是教师的属性
- 职称如果没有与工资、福利挂钩，根据准则（1）可以作为教师实体的属性
- 如果不同的职称有不同的工资、岗位津贴和不同的附加福利，则职称作为一个实体更恰当



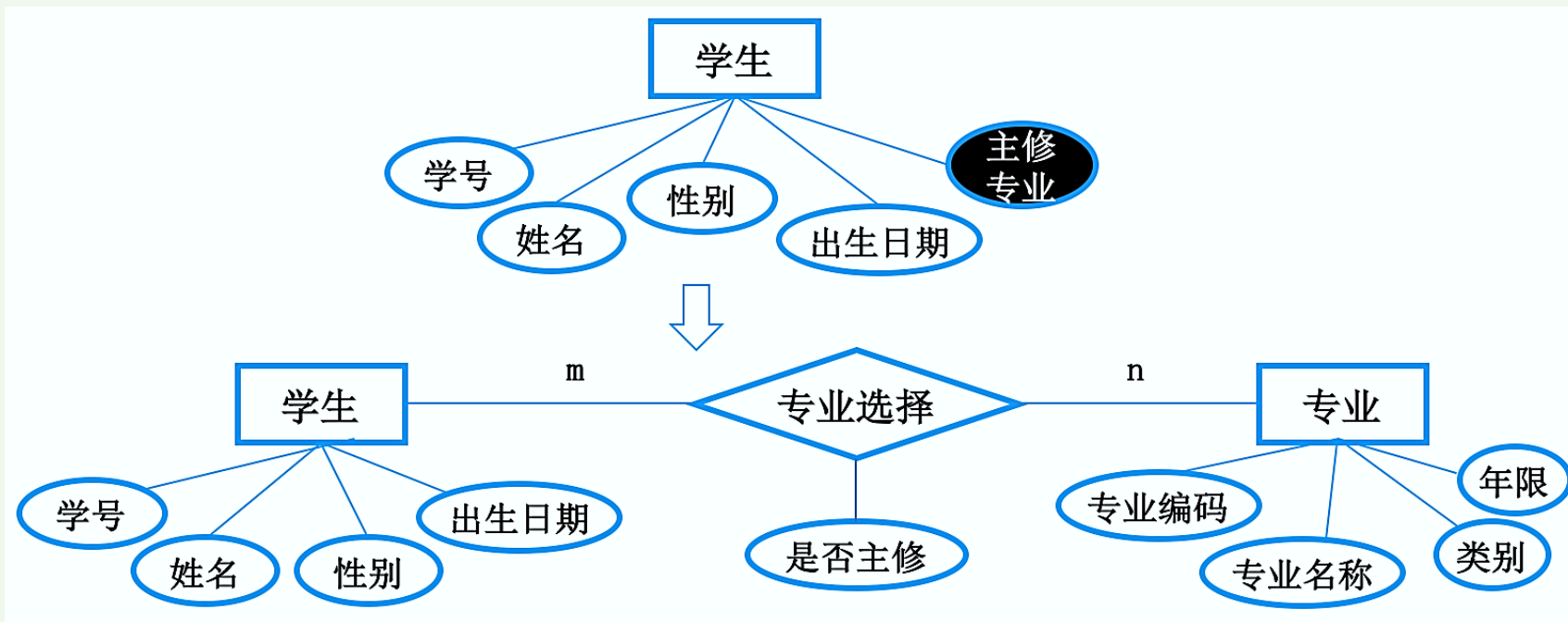


图7.20 “专业” 作为一类实体

- 一个学生如果只能选择一个专业，且专业没有跟专业类别、专业年限等挂钩，则可以把“主修专业”作为“学生”实体的属性。但如果一个学生可以选择多个专业（例如其中有一个专业为主修专业，其他专业为辅修专业），则需要把“专业”单独作为一类实体，并建立学生与专业的多对多专业选择联系

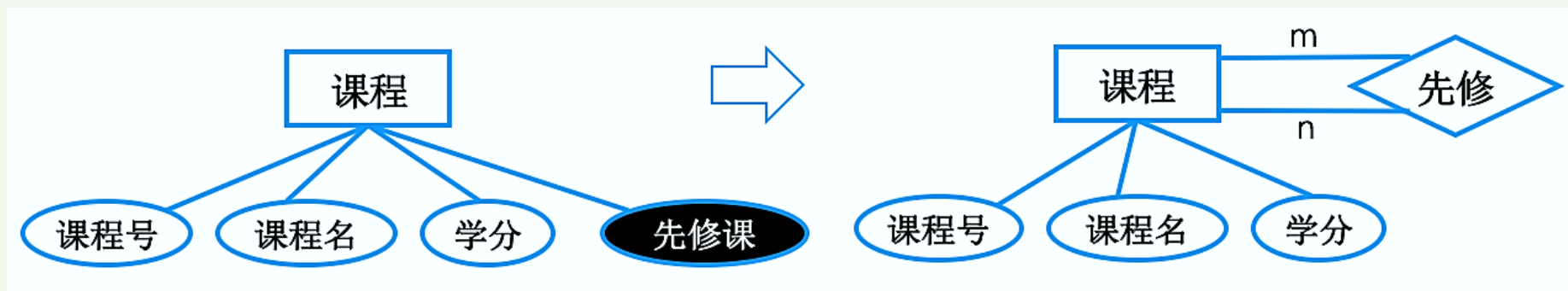


图7.21 “先修课” 作为一类实体（与“课程” 实体型相同）

- 一门课程如果只存在一门直接的先修课，则可以把先修课的课程号作为“课程”实体的属性，但如果一门课程存在多门直接的先修课，则需要把“先修课”作为单独的一类实体。由于“先修课”实体型也是课程实体型，因此“先修课”与“课程”之间是单个实体型内的多对多联系

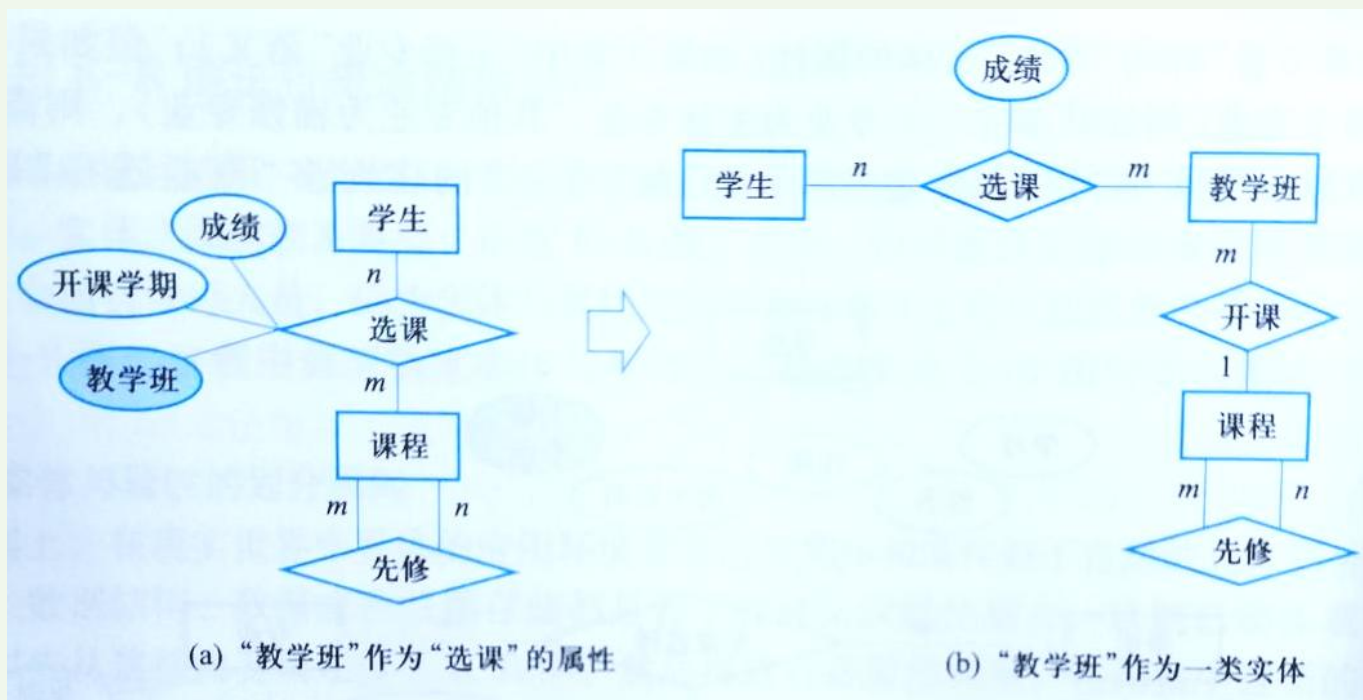


图7.22 “学生选课管理”子系统的分E-R图

- 如果一门课程只设一个教学班，则可以把教学班号作为“选课”联系的属性，属性名为“教学班”，如图7.22 (a) 所示。但这实际的教学活动中，一门课程通常可以开设多个教学班，则应把“教学班”作为一类实体，如图7.22 (b) 所示
- 注：图7.22 (b) 省略了“学生”“课程”和“教学班”实体型的属性



[例7.1] 教师教学管理子系统的分E-R图的设计

– 该子系统的主要功能是：

- 为每个教学班安排授课教师
- 为每个教学班排课，设置教室、上课时间
- 针对学生对教学班的课堂评价，授课教师需要对学生的意见进行反馈



[例7.1] 教师教学管理子系统的分E-R图的设计(cont.)

– 该子系统涉及的实体及属性如下:

- 教师实体的属性包括：职工号、姓名、出生日期、职称（为方便讨论，本子系统将职称作为教师的属性）
- 教学班：教学班号、人数上限、开课学期
- 学生：学号、姓名、性别、出生日期
- 教室实体包括教室号（教室号的编码中已经包括了楼号，例如教室号“003101”表示教3楼101教室），教学楼号、联系人、联系方式等属性
- 时间片实体包括时间片编码（例如“10102”表示星期一第1节课开始，第2节课下课结束）、星期几、开始时间、截止时间
- 注意：时间片实体也可以作为“教学班”与“教师”的“排课”联系属性。考虑到同一个教学班在一周内可以安排在相同的教室、但在不同的时间段讲授多次，本子系统将时间片单独作为实体类



[例7.1] 教师教学管理子系统的分E-R图的设计(cont.)

– 实体间的联系如下:

1. 一个教学班可以安排多个教师来讲授，一个教师也可以讲授多个教学班，因此教师与教学班具有多对多的“讲授”联系，联系的属性包括“是否为主讲教师”（一个教学班设置一个主讲教师）

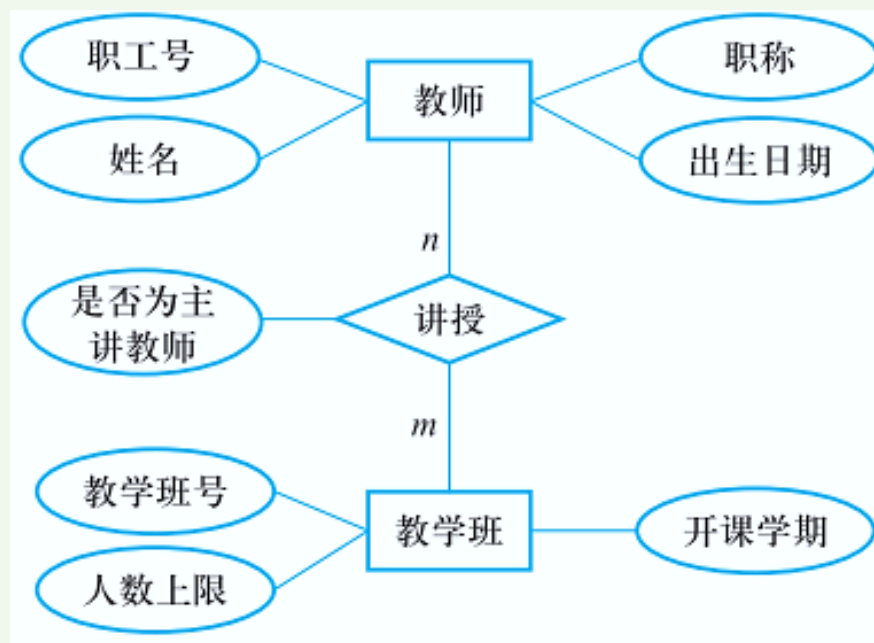


图7.23 “教师”与“教学班”实体之间的讲授联系



[例7.1] 教师教学管理子系统的分E-R图的设计(cont.)

– 实体间的联系如下:

2. 学生可以针对其选择的教学班中的任一位老师进行课堂评价，授课教师也需要对其讲授的所有教学班中每一位学生提出的评价进行一一反馈。因此，教师、学生、教学班三者之间就课堂评价关联存在多对多联系

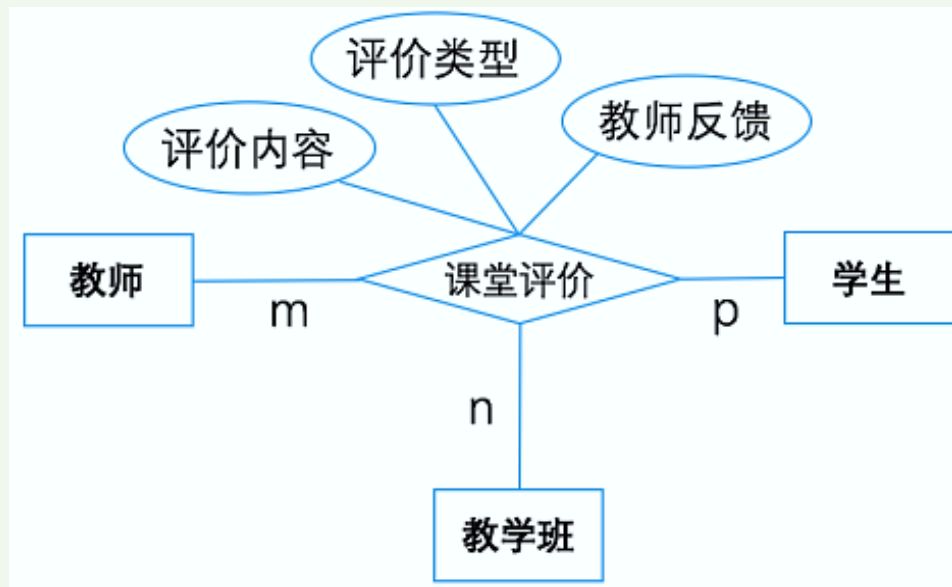


图7.24 “教师” “教学班” “学生” 三个实体之间的 “课程评价” 联系



[例7.1] 教师教学管理子系统的分E-R图的设计(cont.)

– 实体间的联系如下：

3. “教室”与“教学班”和“时间片”实体之间的“排课”联系属于一对多联系，确定了某个教学班、某个时间片，则教室可以唯一确定（一个教室在同一时间段只能安排一门课程）

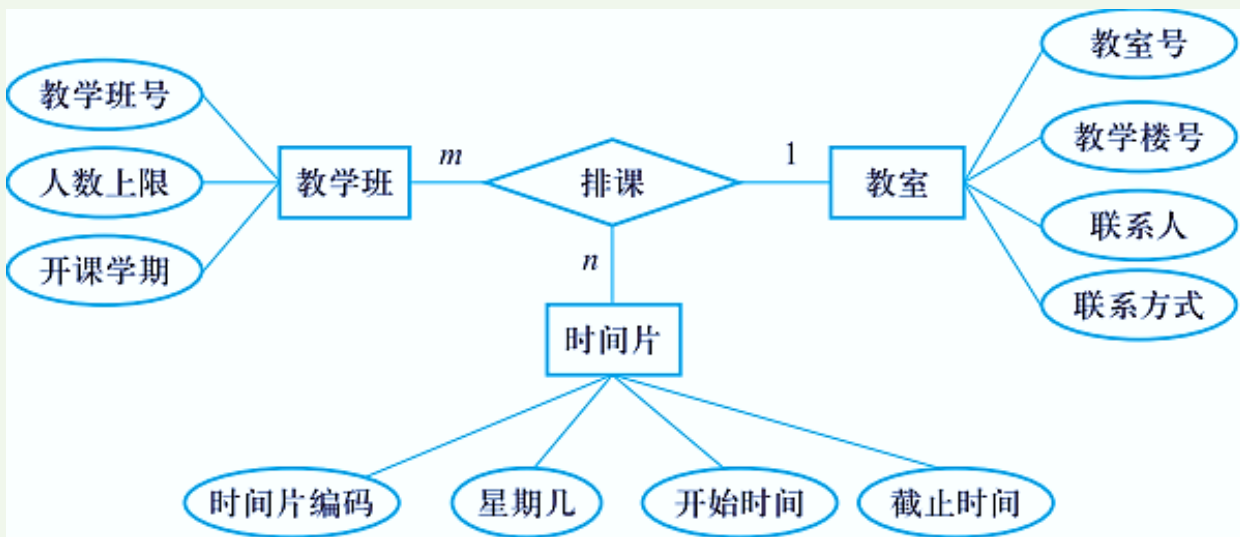
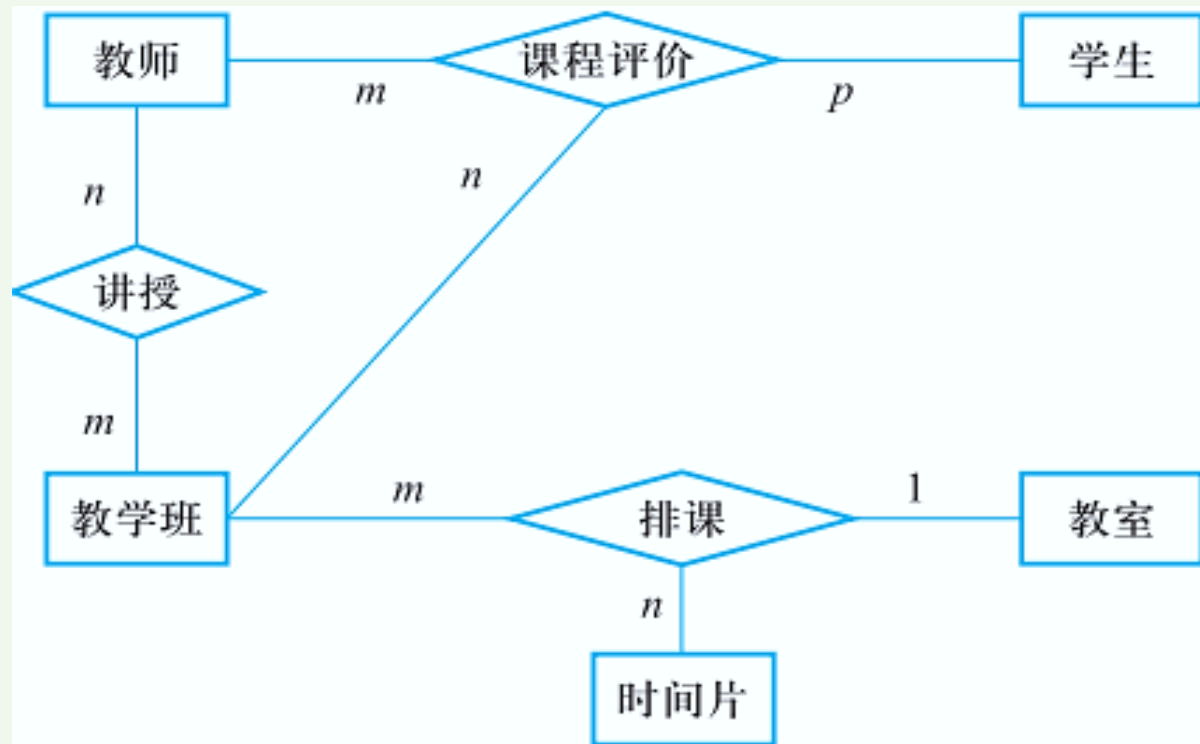


图7.25 “教室” “教学班” “时间片” 三个实体之间的“排课”联系



[例7.1] 最后得到教师教学管理子系统E-R图如下:



每个实体定义的属性如下:

- 教师: Teacher(Tno, Tname, Ttitle, Tbirthdate, Dno)
- 学生: Student(Sno, Sname, Ssex, Sbirthdate, SHno)
- 教学班表: TeachingClass(TCno, TCapacity, Semester, Cno)
- 教室表: Classroom(CRno, CRbuilding, CRcontact, CRtel)
- 时间片表: TimeSlice(TSno, TSdayoftheweek, TSstarttime, TSendtime)

多对多联系定义的属性如下:

- 排课: Schedule(TCno, Tsno, CRno)
- 讲授: Lecture(Tno, TCno, isLeading)
- 课堂评价: ClassAssess(Sno, Tno, TCno, Access, CAtype, Feedback)

图7.26 “教师教学管理”子系统的分E-R图



- 在开发一个大型信息系统时，最经常采用的策略
 - 自顶向下进行需求分析，再自底向上设计概念结构
 - 即：设计各子系统的分E-R图 \Rightarrow 集成分E-R图 \Rightarrow 得到全局E-R图
- E-R图的集成一般需要分两步：
 - 合并：解决各分E-R图之间的冲突，将分E-R图合并起来生成初步E-R图
 - 修改和重构：消除不必要的冗余，生成基本E-R图

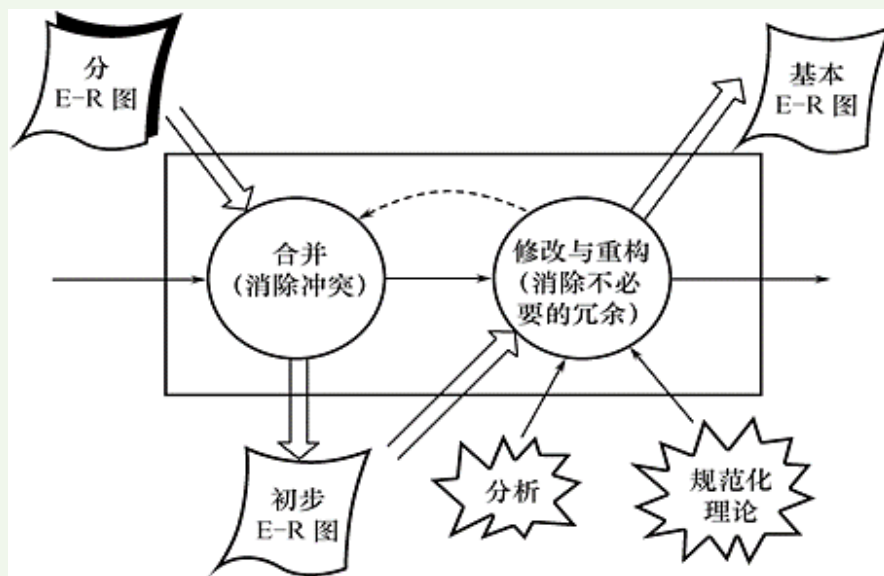


图7.27 E-R图的集成示意图

▪ 合并E-R图，生成初步E-R图

- 各个局部应用所面向的问题不同，各个子系统的E-R图之间必定会存在许多不一致的地方，称之为“冲突”
- 子系统E-R图之间的冲突主要有三类：
 - 属性冲突
 - 命名冲突
 - 结构冲突



■ 1.属性冲突

- 属性域冲突，即属性值的类型、取值范围或取值集合不同
 - 例如教学班号，有的部门把它定义为整数，有的部门把它定义为字符型，不同部门对教学班号的编码也不同
 - 年龄，某些部门以出生日期形式表示职工的年龄，而另一些部门用整数表示职工的年龄
- 属性取值单位冲突
 - 例如出生日期有的精确到年，有的精确到月，有的精确到日

■ 2.命名冲突

- 同名异义，即不同意义的对象在不同的局部应用中具有相同的名字
- 异名同义(一义多名)，即同一意义的对象在不同的局部应用中具有不同的名字
 - 如对科研项目，财务科称为项目，科研处称为课题，生产管理处称为工程
- 命名冲突
 - 可能发生在实体、联系一级上
 - 也可能发生在属性一级上
 - 通过讨论、协商等行政手段加以解决

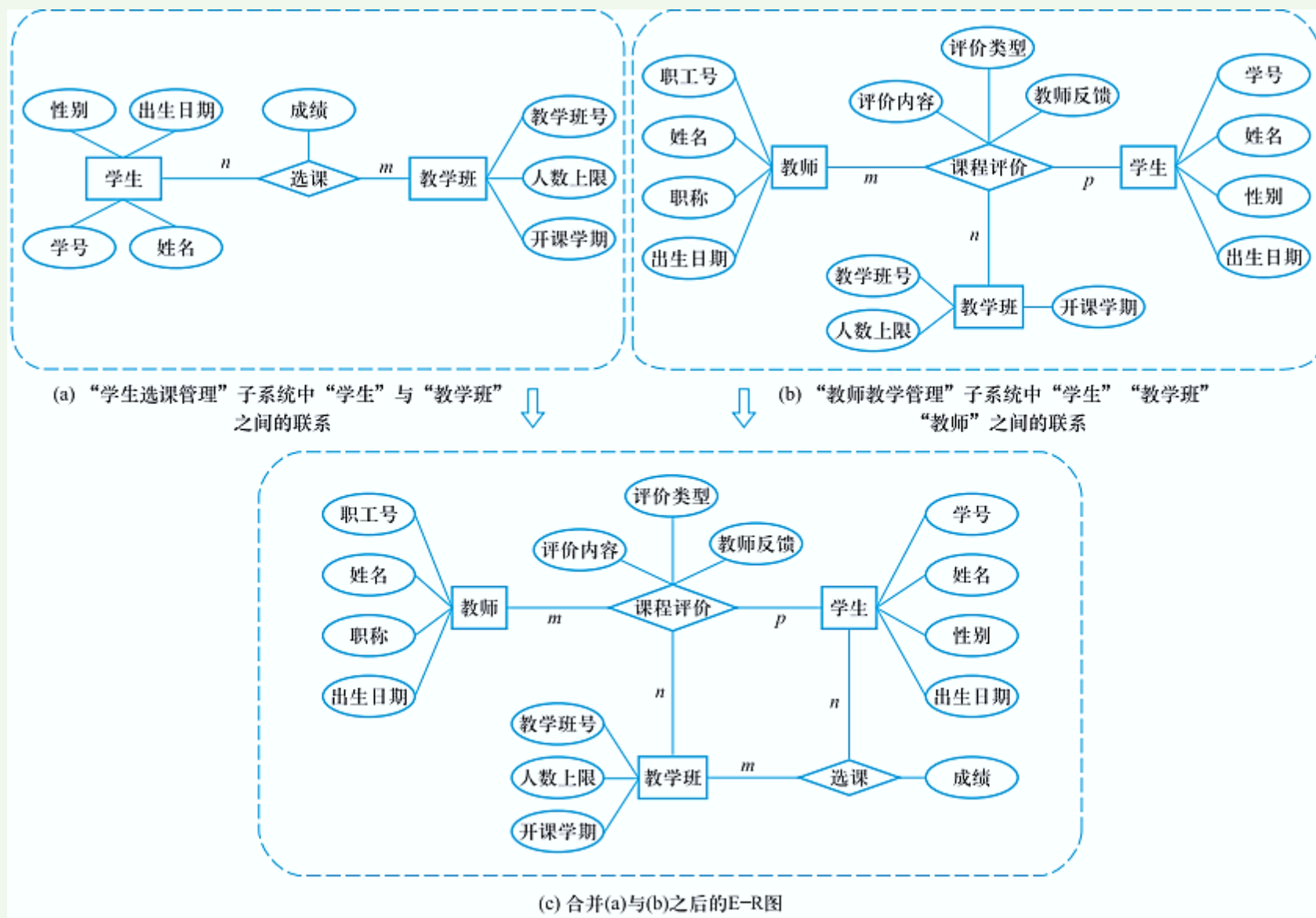


■ 3.结构冲突

- 同一对象在不同应用中具有不同的抽象
 - 例如，职工在某一局部应用中被当作实体，而在另一局部应用中则被当作属性
 - 解决方法：把属性变换为实体或把实体变换为属性，使同一对象具有相同的抽象
- 同一实体在不同子系统的**E-R图**中包含的属性个数和属性排列次序不完全相同
 - 解决方法：使该实体的属性取各子系统的**E-R图**中属性的并集，再适当调整属性的次序
- 实体间的联系在不同的**E-R图**中为不同的类型
 - 实体**E1**与**E2**在一个**E-R图**中是多对多联系，在另一个**E-R图**中是一对多联系
 - 解决方法是根据应用的语义对实体联系的类型进行综合或调整



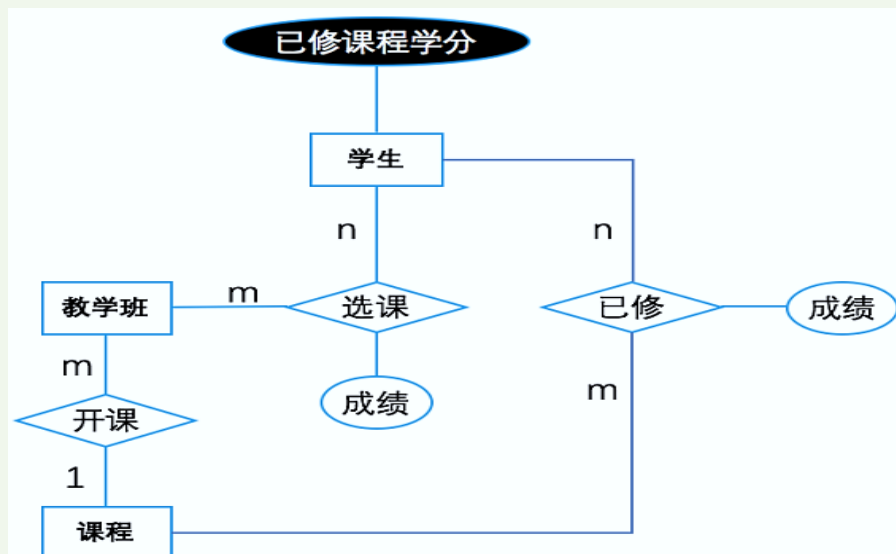
图7.28 合并两个E-R图示例



■ 消除不必要的冗余，设计基本E-R图

- 所谓**冗余的数据**是指可由基本数据导出的数据，**冗余的联系**是指可由其他联系导出的联系
- **消除冗余主要采用分析方法**，即以数据字典和数据流图为依据，根据数据字典中关于数据项之间逻辑关系的说明来消除冗余

【例】 学生与课程的“已修”关系(记录学生选修合格及以上的选课信息)，可以由学生与教学班之间的选课关系、以及教学班与课程的开课关系导出，因此，已修关系为冗余关系，可以消去



- 并不是所有的冗余数据与冗余联系都必须加以消除，有时为了提高效率，不得不以冗余信息作为代价。



■ 除了分析方法之外，也可用规范化理论来消除冗余

– 确定分E-R图实体之间的数据依赖

- 实体之间一对一、一对多、多对多的联系可以用实体码之间的函数依赖来表示。于是有函数依赖集 F_L
- 例如学院和系之间一对多的联系可表示为系编号 \rightarrow 学院编号，系和专业之间一对多的联系可表示为专业编码 \rightarrow 系编号。专业和学生之间多对多的联系可表示为（学号，专业编码） \rightarrow 是否主修等

– 求 F_L 的最小覆盖 G_L ，差集为 $D=F_L-G_L$

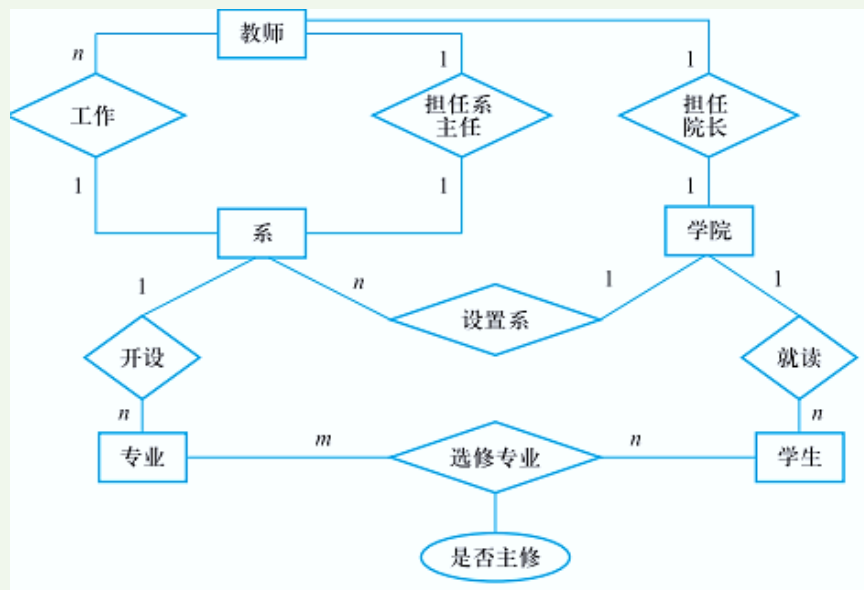
- 逐一考察 D 中的函数依赖，确定是否是冗余的联系，若是，就把它去掉

– 由于规范化理论受到泛关系假设的限制，应注意下面两个问题：

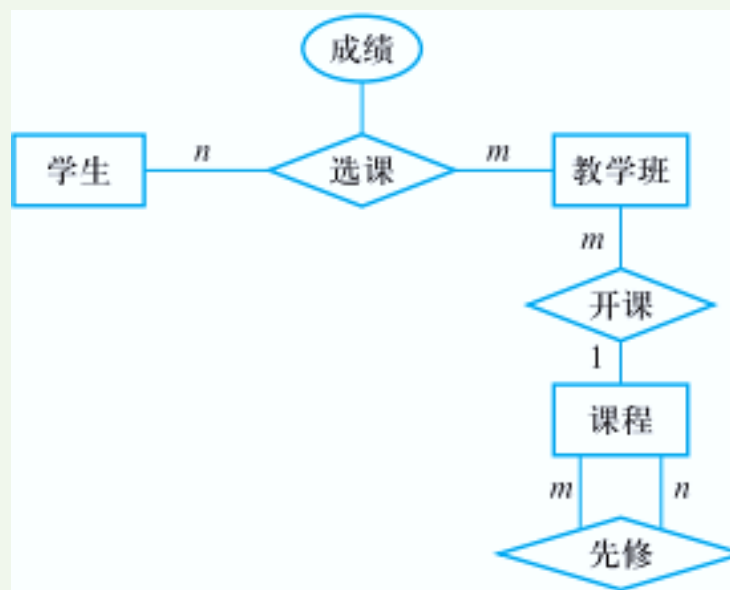
- 冗余的联系一定在 D 中，而 D 中的联系不一定是冗余的
- 当实体之间存在多种联系时，要将实体之间的联系在形式上加以区分。例如教师和系之间一对一的联系“系主任”就要表示为系主任.职工号 \rightarrow 系编号，系编号 \rightarrow 系主任.职工号



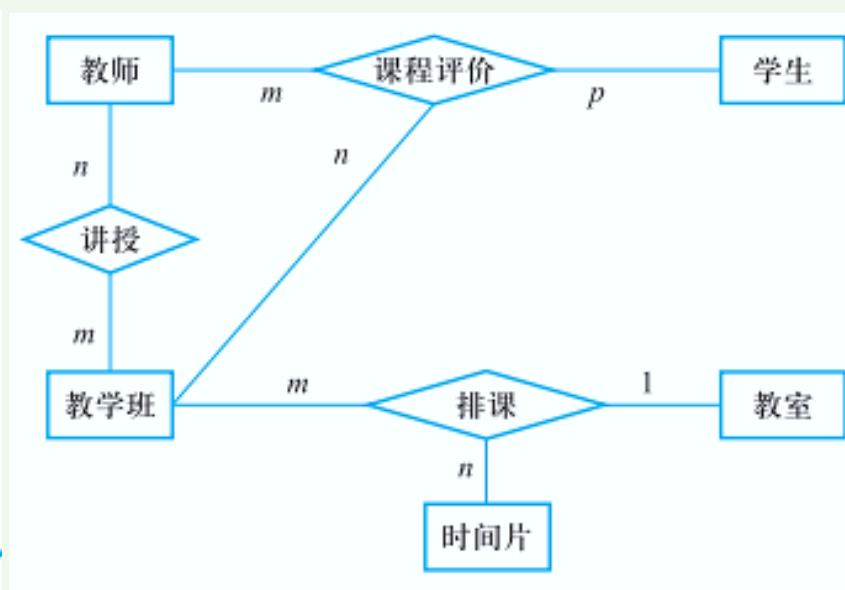
[例7.2] “高校本科教务管理” 信息系统的视图集成



学生学籍管理子系统分E-R图



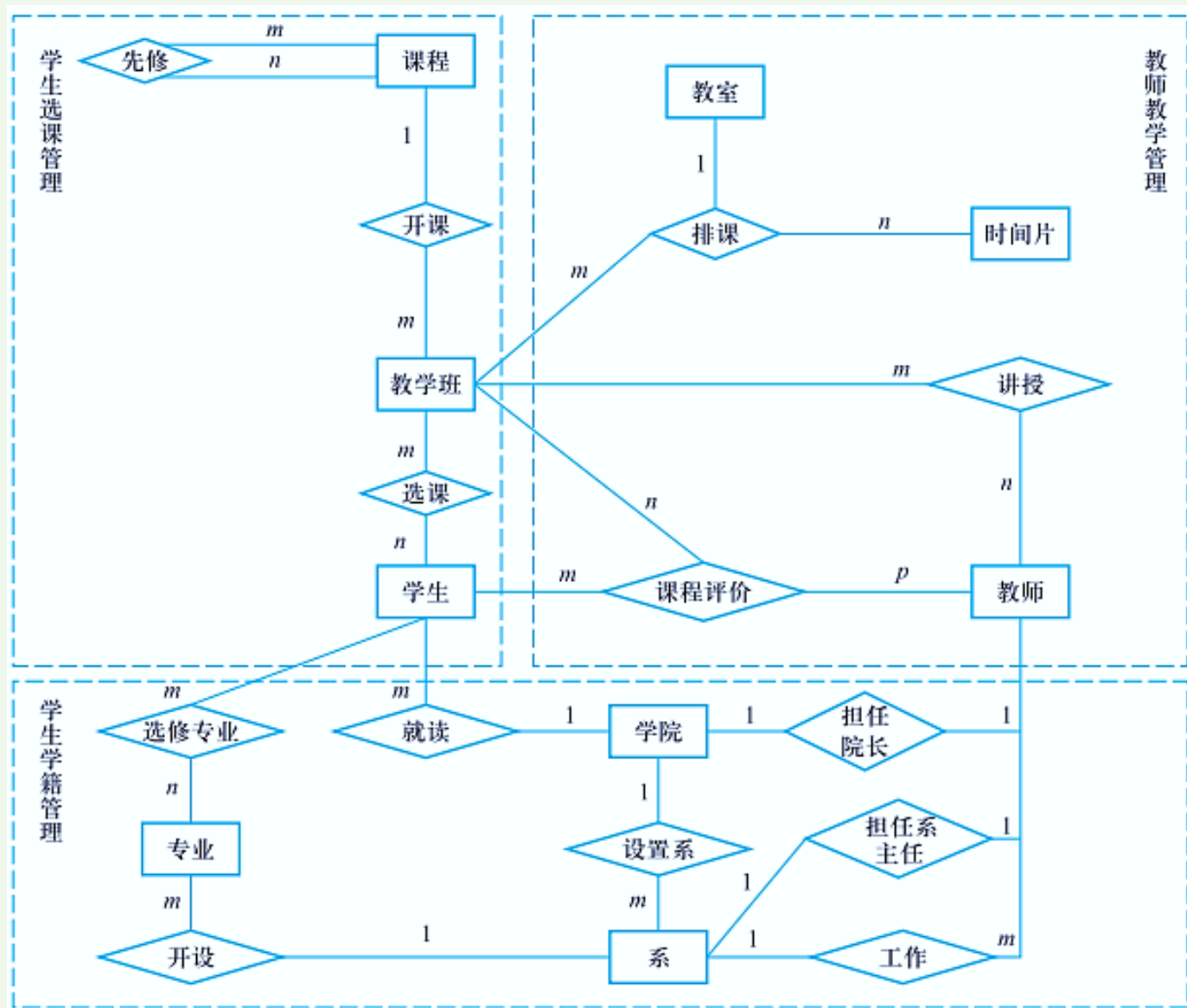
学生选课管理子系统的分E-R图



教师教学管理子系统的分E-R图



图7.30 高校本科教务管理信息系统的基本ER图



■ 逻辑结构设计的任务

- 把概念结构设计阶段设计好的基本E-R图转换为与选用的DBMS产品所支持的数据模型相符合的逻辑结构
- 主要介绍E-R图向关系数据模型的转换

■ 本节主要内容：

- E-R图向关系模型的转换
- 数据模型的优化
- 设计用户外模式



■ 转换内容

- E-R图由实体型、实体的属性和实体型之间的联系三个要素组成
- 关系模型的逻辑结构是一组关系模式的集合
- 将E-R图转换为关系模型：将实体型、实体的属性和实体型之间的联系转化为关系模式

■ 转换原则

- 1. 一个实体型转换为一个关系模式。
 - 关系的属性：实体的属性
 - 关系的码：实体的码



■ 转换原则

– 2. 实体型间的联系有以下不同情况：1对1联系

- 一个一对一联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并

(1) 转换为一个独立的关系模式

- 关系的属性：与该联系相连的各实体的码以及联系本身的属性
- 关系的候选码：每个实体的码均是该关系的候选码

(2) 与某一端实体对应的关系模式合并

- 合并后关系的属性：加入对应关系的码和联系本身的属性
- 合并后关系的码：不变



■ 转换原则

– 2. 实体型间的联系有以下不同情况：多对多联系

- 多对多联系转换为一个独立的关系模式

- 关系的属性：与该联系相连的各实体的码以及联系本身的属性
- 关系的码或关系码的一部分：各实体码的组合

- **[示例]** “选修”联系是一个多对多联系，可以将它转换为如下关系模式，其中学号与教学班号为关系的组合码：

- 选修 (学号, 教学班号, 成绩)



■ 转换原则

– 3. 实体型间的联系有以下不同情况：三个或三个以上实体间的联系

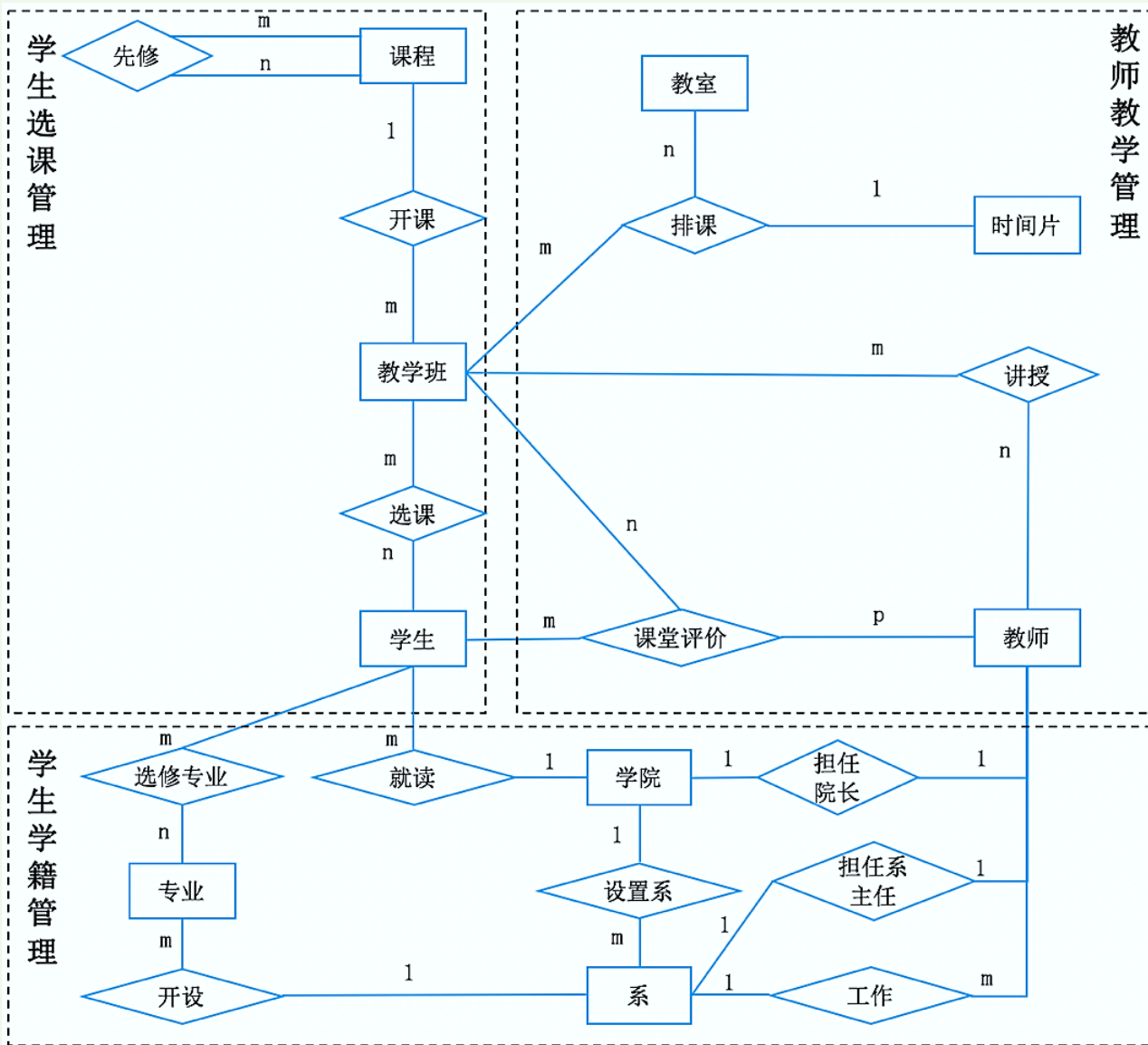
- 三个或三个以上实体间的一个多元联系转换为一个关系模式
 - 关系的属性：与该多元联系相连的各实体的码以及联系本身的属性
 - 关系的码或关系码的一部分：各实体码的组合

– 具有相同码的关系模式可合并

- 目的：减少系统中的关系个数
- 合并方法：
 - 将其中一个关系模式的全部属性加入到另一个关系模式中
 - 然后去掉其中的同义属性（可能同名也可能不同名）
 - 适当调整属性的次序



[例3] 将学生学籍管理子系统的公共子图转换为关系模型。关系的码用下划线标出



- “学生学籍管理” 子系统包括如下6个关系模式
 - 学院 (学院编号, 学院名, 建院时间, 院长)
 - 系 (系编号, 系名, 联系人, 联系方式, 系主任, 所在学院)
 - 专业 (专业编码, 专业名, 类别, 年限, 开设系)
 - 教师 (职工号, 姓名, 职称, 出生日期, 所在系)
 - 学生 (学号, 姓名, 性别, 出生日期, 所在学院)
 - 选修专业 (学号, 专业编码, 是否主修)



- 一般的数据模型还需要向特定**DBMS**统规定的模型进行转换
- 转换的主要依据是所选用的**DBMS**的功能及限制，没有通用规则
- 对于关系模型来说，这种转换通常都比较简单
- 数据库逻辑设计的结果不是唯一的
- 得到初步数据模型后，还应该适当地修改、调整数据模型的结构，以进一步提高数据库应用系统的性能，这就是数据模型的优化
- 关系数据模型的优化通常以规范化理论为指导



1. 确定数据依赖

- 按需求分析阶段所得到的语义，分别写出每个关系模式内部各属性之间的数据依赖以及不同关系模式属性之间数据依赖

2. 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系

3. 按照数据依赖的理论对关系模式进行分析，考察是否存在部分函数依赖、传递函数依赖、多值依赖等，确定各关系模式分别属于第几范式



4. 按照需求分析阶段得到的处理的要求，分析对于这样的应用环境这些模式是否合适，**确定是否要对某些模式进行合并或分解**

- 并不是规范化程度越高的关系就越优
 - 当查询经常涉及两个或多个关系模式的属性时，系统必须经常地进行连接运算
 - 连接运算的代价是相当高的
 - 在这种情况下，第二范式甚至第一范式也许是适合的
- 非BCNF的关系模式虽然会存在不同程度的更新异常或冗余，但如果在实际应用中对此关系模式只是查询，并不执行更新操作，就不会产生实际影响
- 对于一个具体应用来说，到底规范化进行到什么程度，需要权衡响应时间和潜在问题两者的利弊才能决定



5. 对关系模式进行必要分解，提高数据操作效率和存储空间的利用率

– 常用分解方法：

- 水平分解
- 垂直分解

– 什么是水平分解？

- 把(基本)关系的元组分为若干子集合，定义每个子集合为一个子关系，以提高系统的效率

– 如何进行水平分解？

- 对符合“80/20原则”的，把经常被使用的数据（约20%）水平分解出来，形成一个子关系
- 关系R上n个事务，多数事务存取数据不相交，R可分解为少于或等于n个子关系，使每个事务存取的数据对应一个子关系



5. 对关系模式进行必要分解，提高数据操作效率和存储空间的利用率

– 什么是垂直分解？

- 把关系模式 R 的属性分解为若干子集合，形成若干子关系模式。

– 垂直分解的原则

- 经常在一起使用的属性从 R 中分解出来形成一个子关系模式

– 垂直分解的优点

- 可以提高某些事务的效率

– 垂直分解的缺点

- 可能使另一些事务不得不执行连接操作，降低了效率。

– 垂直分解的适用范围

- 取决于分解后 R 上的所有事务的总效率是否得到了提高

– 进行垂直分解的方法

- 简单情况：直观分解
- 复杂情况：用第6章中的模式分解算法
- 垂直分解必须不损失关系模式的语义（保持无损连接性和保持函数依赖）



■ 定义用户外模式时应注重三个方面：

– 1.使用更符合用户习惯的别名

- 合并各分E-R图曾做了消除命名冲突的工作，以使数据库系统中同一关系和属性具有唯一的名字。这在设计数据库整体结构时是非常必要的
- 用视图机制可以在设计用户视图时可以重新定义某些属性名，使其与用户习惯一致，以方便使用

– 2.针对不同级别的用户定义不同的视图，以保证系统的安全性

- 如针对教师教学管理子系统内的教师、学生、教学班三者之间的教学评价关系，建立两个视图：
 - 教师-教学班评价视图(职工号、教师.姓名、教学班号、课程号、课程名、评价内容、评价类型、教师反馈)
 - 学生-教学班-评价视图 (学号、学生.姓名、教师.姓名、教学班号、课程号、课程名、评价内容、评价类型、教师反馈)
 - 教师-教学班评价视图包含教师对应教学班的所有课程评价信息，隐藏了提供教学评价意见的学生信息，从而保证了学生信息的安全性；学生-教学班-评价视图可以看到学生评价信息，也能看到教师反馈

– 3.简化用户对系统的使用

- 如果某些局部应用中经常要使用某些很复杂的查询，为了方便用户，可以将这些复杂查询定义为视图



■ 什么是数据库的物理设计？

- 数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于选定的数据库管理系统
- 为一个给定的逻辑数据模型选取一个最适合应用要求的物理结构的过程，就是数据库的物理设计

■ 数据库物理设计的步骤

- 确定数据库的物理结构
 - 在关系数据库中主要指存取方法和存储结构
- 对物理结构进行评价
 - 评价的重点是时间和空间效率
 - 如果评价结果满足原设计要求，则可进入到物理实施阶段。否则就需要重新设计或修改物理结构，有时甚至要返回逻辑设计阶段修改数据模型



- 本节主要内容：
 - 数据库物理设计的内容和方法
 - 选择关系模式存取方法
 - 确定数据库的存储结构
 - 评价数据库的物理结构

- 设计物理数据库结构的准备工作
 - 对要运行的**事务**进行详细分析，获得选择物理数据库设计所需参数
 - 充分了解所用**RDBMS**的内部特征，特别是系统提供的存取方法和存储结构
- 选择物理数据库设计所需参数
 - **数据库查询事务**
 - 查询的关系
 - 查询条件所涉及的属性
 - 连接条件所涉及的属性
 - 查询的投影属性
 - **数据更新事务**
 - 被更新的关系
 - 每个关系上的更新操作条件所涉及的属性
 - 修改操作要改变的属性值
 - 每个事务在各关系上运行的**频率和性能要求**，如事务**T**必须在**100ms**内结束
- 关系数据库物理设计的内容
 - 为关系模式选择存取方法（建立存取路径）
 - 设计关系、索引等数据库文件的物理存储结构



- 数据库系统是多用户共享的系统，对同一个关系要建立多条存取路径才能满足多用户的多种应用要求
- 物理结构设计任务之一是根据RDBMS支持的存取方法确定选择哪些存取方法
 - 存取方法是快速存取数据库中数据的技术
- RDBMS常用存取方法：
 - B⁺树索引
 - 哈希索引
 - 聚簇索引

} 两种经典的索引方法，使用最为普遍



- 根据应用要求确定
 - 对哪些属性列建立索引
 - 对哪些属性列建立组合索引
 - 对哪些索引要设计为唯一索引
- B⁺树索引存取方法的一般规则
 - 如果一个(或一组)属性经常在查询条件中出现, 则考虑在这个(或这组)属性上建立索引(或组合索引)
 - 如果一个属性经常作为最大值和最小值等聚集函数的参数, 则考虑在这个属性上建立索引
 - 如果一个(或一组)属性经常在连接操作的连接条件中出现, 则考虑在这个(或这组)属性上建立索引
- 关系上定义的索引数过多会带来较多的额外开销
 - 维护索引的开销
 - 查找索引的开销



- 选择使用哈希索引方法的两种情形：
 1. 如果一个关系的属性主要出现在等值连接条件中，或
 2. 主要出现在等值比较选择条件中，且满足两个条件之一：
 - 该关系的大小可预知，而且不变
 - 该关系的大小动态改变，但所选用的DBMS提供了动态哈希存取方法



1. 什么是聚簇?

- 为了提高某个属性(或属性组)的查询速度, 把这个或这些属性(称为聚簇码)上具有相同值的元组集中存放在连续的物理块中称为聚簇
- 该属性(或属性组)称为聚簇码(cluster key)
- 许多RDBMS都提供了聚簇功能
- 聚簇存放与聚簇索引的区别

2. 聚簇索引

- 建立聚簇索引后, 基表中数据也需要按指定的聚簇属性值的升序或降序存放。也即聚簇索引的索引项顺序与表中元组的物理顺序一致
- 在一个基本表上最多只能建立一个聚簇索引



3. 聚簇索引的适用条件

- 很少对基表进行增删操作
- 很少对其中的变长列进行修改操作

4. 聚簇的作用

- 大大提高按聚簇属性进行查询的效率

【例】假设学生关系按所在系建有索引，现在要查询信息系的所有学生名单。

- 计算机系的500名学生分布在500个不同的物理块上时，至少要执行500次I/O操作。
- 如果将同一系的学生元组集中存放，则每读一个物理块可得到多个满足查询条件的元组，从而显著地减少了访问磁盘的次数。

- 节省存储空间

- 聚簇以后，聚簇码相同的元组集中在一起了，因而聚簇码值不必在每个元组中重复存储，只要在一组中存一次就行了



5.聚簇的适用范围

- 既适用于单个关系独立聚簇，也适用于多个关系组合聚簇。
- 当通过聚簇码进行访问或连接是该关系的主要应用，与聚簇码无关的其他访问很少或者是次要的时，可以使用聚簇。
 - 尤其当SQL语句中包含有与聚簇码有关的**ORDER BY, GROUP BY, UNION, DISTINCT**等子句或短语时，使用聚簇特别有利，可以省去或减化对结果集的排序操作。

6.聚簇的局限性

- 聚簇只能提高某些特定应用的性能
- 建立与维护聚簇的开销相当大
 - 对已有关系建立聚簇，将导致关系中元组的物理存储位置移动，并使此关系上原有的索引无效，必须重建。
 - 当一个元组的聚簇码改变时，该元组的存储位置也要做相应改变



7.选择聚簇存取方法

a. 设计候选聚簇

- ① 常在一起进行连接操作的关系可以建立组合聚簇
- ② 如果一个关系的一组属性经常出现在相等比较条件中, 则该单个关系可建立聚簇
- ③ 如果一个关系的一个(或一组)属性上的值重复率很高, 则此单个关系可建立聚簇。

b. 检查候选聚簇中的关系, 取消其中不必要的关系

- ① 从聚簇中删除经常进行全表扫描的关系
- ② 从聚簇中删除更新操作远多于连接操作的关系
- ③ 不同的聚簇中可能包含相同的关系, 一个关系可以在某一个聚簇中, 但不能同时加入多个聚簇。
要从这多个聚簇方案(包括不建立聚簇)中选择一个较优的, 即在这个聚簇上运行各种事务的总代价最小



- 确定数据库物理结构主要指确定数据的**存放位置**和**存储结构**，包括：确定关系、索引、聚簇、日志、备份等的存储安排和存储结构，确定系统配置等
- 确定数据的存放位置和存储结构要综合考虑**存取时间**、**存储空间利用率**和**维护代价**三个方面的因素
 - 这三个方面常常是相互矛盾的，因此需要进行权衡，选择一个**折中方案**

1.确定数据的存放位置

- 根据应用情况将数据
 - **易变**部分与**稳定**部分分开存放
 - **经常存取**部分与**存取频率较低**部分分开存放
- 例如：
 - 可以将比较大的表分别放在两个磁盘上，加快存取速度，这在多用户环境下特别有效。
 - 可以将日志文件与数据库对象（表、索引等）放在不同的磁盘以改进系统的性能。

2.确定系统配置

- 数据库管理系统一般都提供了一些存储分配参数
 - 合理的默认值/初始值，但不一定适合每一种应用环境
- 同时使用数据库的用户数
- 同时打开的数据库对象数
- 内存分配参数
- 缓冲区分配参数（使用的缓冲区长度、个数）
- 存储分配参数
- 物理块的大小
- 物理块装填因子
- 时间片大小
- 数据库的大小
- 锁的数目等
- 系统都为这些变量赋予了合理的缺省值。
- 在进行物理设计时需要根据应用环境确定这些参数值，以使系统性能最优。
- 在物理设计时对系统配置变量的调整只是初步的，要根据系统实际运行情况做进一步的调整，以期切实改进系统性能



- 对数据库物理设计过程中依据**时间效率**、**空间效率**、**维护代价**和各种用户要求所产生的多种方案进行**评价**，从中选择一个较优的方案作为数据库的物理结构
- 评价数据库物理结构设计的方法**完全依赖于所选用的RDBMS**，评价维度：
 - **存储空间**
 - **存取时间**
 - **维护代价**
- 对估算结果进行权衡、比较，选择出一个较优的、合理的物理结构
 - 如果该结构不符合用户需求，则需要修改设计



- 完成数据库的物理设计之后，设计人员就要使用目标RDBMS提供的数据库定义语言或具备数据库定义语言功能的工具(如MySQL提供了Workbench工具)，直接或间接地将数据库的逻辑结构设计和物理结构设计结果严格描述出来，生成目标RDBMS可以接受的SQL语句，再经过调试产生目标模式，然后就可以组织数据入库，这就是数据库的实施阶段
- 本节主要内容：
 - 数据的载入和应用程序的编码与调试
 - 数据库的试运行
 - 数据库的运行和维护

- **组织数据载入**就是要将各类源数据从各个局部应用中抽取出来，输入计算机，再分类转换，最后综合成符合新设计的数据库结构的形式，输入数据库
 - 这样的数据转换、组织入库的工作是相当费力、费时的
 - 因为数据的组织方式、结构和格式都与新设计的数据库系统有相当的差距
 - 特别是原系统是手工数据处理系统时，各类数据分散在各种不同的原始表格、凭证和单据之中
- 为提高数据录入的工作的效率和质量，应该针对具体的应用环境设计一个**数据录入子系统**，由计算机来完成数据入库的任务
- 现代**RDBMS**一般都提供不同**RDBMS**之间**数据转换的工具**，应充分利用新系统的数据转换工具
 - **ORACLE与SQL SERVER数据转换示例**: <http://www.cnblogs.com/jxgzCHforever/p/8650056.html>
- **数据库应用程序的设计应该与数据库设计同时进行**，因此在组织数据入库的同时还要调试应用程序
 - 应用程序的设计、编码和调试的方法、步骤参见软件工程相关课程



- 在系统的数据有一小部分已输入数据库后，就可以开始对数据库系统进行联合调试，这也称为**数据库的试运行**
- **数据库的试运行包括：**
 - **功能测试：**实际运行数据库应用程序，执行对数据库的各种操作，测试应用程序的各项功能是否满足设计要求。如果不满足，对应用程序部分则要修改、调整，直至达到设计要求为止
 - **性能测试：**测量系统的性能指标，分析其是否达到设计目标
 - **原因：**设计时得到的只是近似估计，与实际系统运行有一定差距
 - 如果测试的结果与设计目标不符，则要返回物理设计阶段重新调整物理结构，修改系统参数，某些情况下甚至要返回逻辑设计阶段修改逻辑结构
- **注意：**
 - 如果试运行后还要修改数据库的设计，则需要重新组织数据入库。因此，**应分期分批组织数据入库**。
 - 要做好数据库的转储和恢复工作，一旦故障发生，能使数据库尽快恢复，尽量减少对数据库的破坏。
 - 这是因为，试运行阶段的数据库系统还不稳定，软硬件故障随时都可能发生，系统的操作人员对新系统还不熟悉，误操作不可避免



- 数据库试运行合格后，数据库开发工作就基本完成，可以投入正式运行。但由于应用环境不断变化，数据库运行过程中物理存储也会不断变化，对数据库设计进行**评价、调整、修改等维护工作**是一个长期的任务，也是设计工作的延伸和提高
- 在数据库运行阶段，对数据库**经常性的维护工作**主要是由**DBA**完成的
- 数据库的维护工作主要包括：
 - 数据库的转储和恢复
 - 数据库的安全性、完整性控制
 - 数据库性能的监督、分析和改造
 - 主流**RDBMS**一般会提供监测系统性能参数的工具
 - 数据库的重组与重构
 - **DBMS**一般都提供了供重组数据库使用的实用程序，帮助**DBA**重新组织数据库
 - 重构造数据库的程度是有限的。若重构数据库的代价太大，则表明现有数据库应用系统的生命周期已经结束，应该**重新设计新的数据库应用系统**了



- 详细介绍了数据库设计各个阶段的目标、方法和步骤，重点是概念结构的设计和逻辑结构的设计
- 数据库各级模式的形成：
 - 需求分析阶段：综合各个用户的应用需求(现实世界的需求)
 - 概念设计阶段：概念模式(信息世界模型)，用E-R图来描述
 - 逻辑设计阶段：逻辑模式、外模式
 - 物理设计阶段：内模式
- 数据库应用程序的设计应与数据库设计同时进行
- 数据库运维过程中DBA的主要职责和工作内容



- openGauss场景化综合应用实验

- 数据库外模式是在下列哪个阶段设计()
A.数据库概念结构设计 B.数据库逻辑结构设计 C.数据库物理结构设计 D.数据库实施和维护
- 生成DBMS系统支持的数据模型是在下列哪个阶段完成()
A.数据库概念结构设计 B.数据库逻辑结构设计 C.数据库物理结构设计 D.数据库实施和维护
- 根据应用需求建立索引是在下列哪个阶段完成()
A.数据库概念结构设计 B.数据库逻辑结构设计 C.数据库物理结构设计 D.数据库实施和维护
- 员工性别的取值，有的为“男”，“女”，有的为“1”，“0”，这种情况属于()
A.属性冲突 B.命名冲突 C.结构冲突 D.数据冗余
- 集成局部E-R图要分为两个步骤，分别是_____和_____。
- 数据库常见的存取方法主要有_____、_____和hash方法。
- 在进行概念结构设计时，将事物作为属性的基本准则是什么？
- 将E-R图转换为关系模式时，可以如何处理实体型之间的联系？



- 教材第七章全部习题.
- 自行研读附录“高校本科教务管理”信息系统的E-R图和关系模式.
- 要求：作业在布置后一周完成并提交到课程网站

