

数据管理基础

第7章 数据库设计

智能软件与工程学院



7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 物理结构设计

7.6 数据库的实施和维护

7.7 小结

数据管理基础

7.1 数据库设计概述

智能软件与工程学院

□ ‘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、删、改、统计等操作。

数据库设计的目标

❑ 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。

❑ 高效率的运行环境

- 数据库数据的存取效率高
- 数据库存储空间的利用率高
- 数据库系统运行管理的效率高

7.1 数据库设计概述

7.1.1 数据库设计的特点

7.1.2 数据库设计方法

7.1.3 数据库设计的基本步骤

7.1.4 数据库设计过程中的各级模式

7.1.1 数据库设计的特点

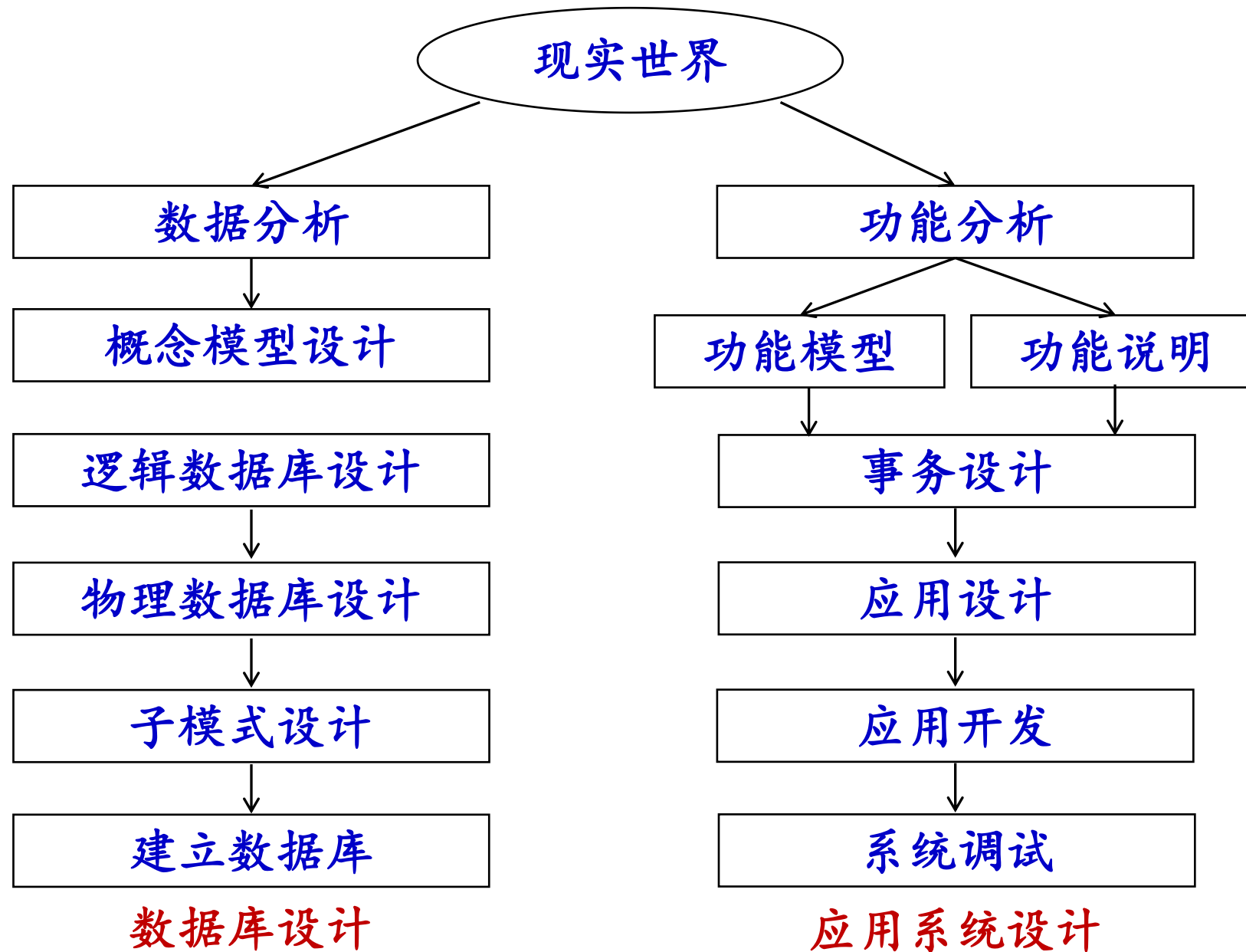
1. 数据库建设的基本规律

- 三分技术，七分管理，十二分基础数据
- 管理
 - 数据库建设项目管理
 - 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. 结构（数据）设计和行为（处理）设计相结合

- 将数据库结构设计和数据处理设计密切结合
- 结构和行为分离的设计
 - 传统软件工程：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

图7.1 结构和行为分离的设计



7.1.2 数据库设计方法 (1)

- ❑ 大型数据库设计是涉及多学科的综合性的技术，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

7.1.2 数据库设计方法 (2)

- ❑ 手工试凑法（手工与经验相结合）
 - 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的支持，工程的质量难以保证
 - 数据库运行一段时间后常常又不同程度地发现各种问题，增加了维护代价
- ❑ 规范设计法（基本思想：过程迭代和逐步求精）
 - 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

7.1.3 数据库设计的基本步骤 (1)

❑ 数据库设计分6个阶段

- 需求分析
- 概念结构设计
- 逻辑结构设计
- 物理结构设计
- 数据库实施
- 数据库运行和维护

❑ 需求分析和概念设计独立于任何数据库管理系统

❑ 逻辑设计和物理设计与选用的数据库管理系统密切相关

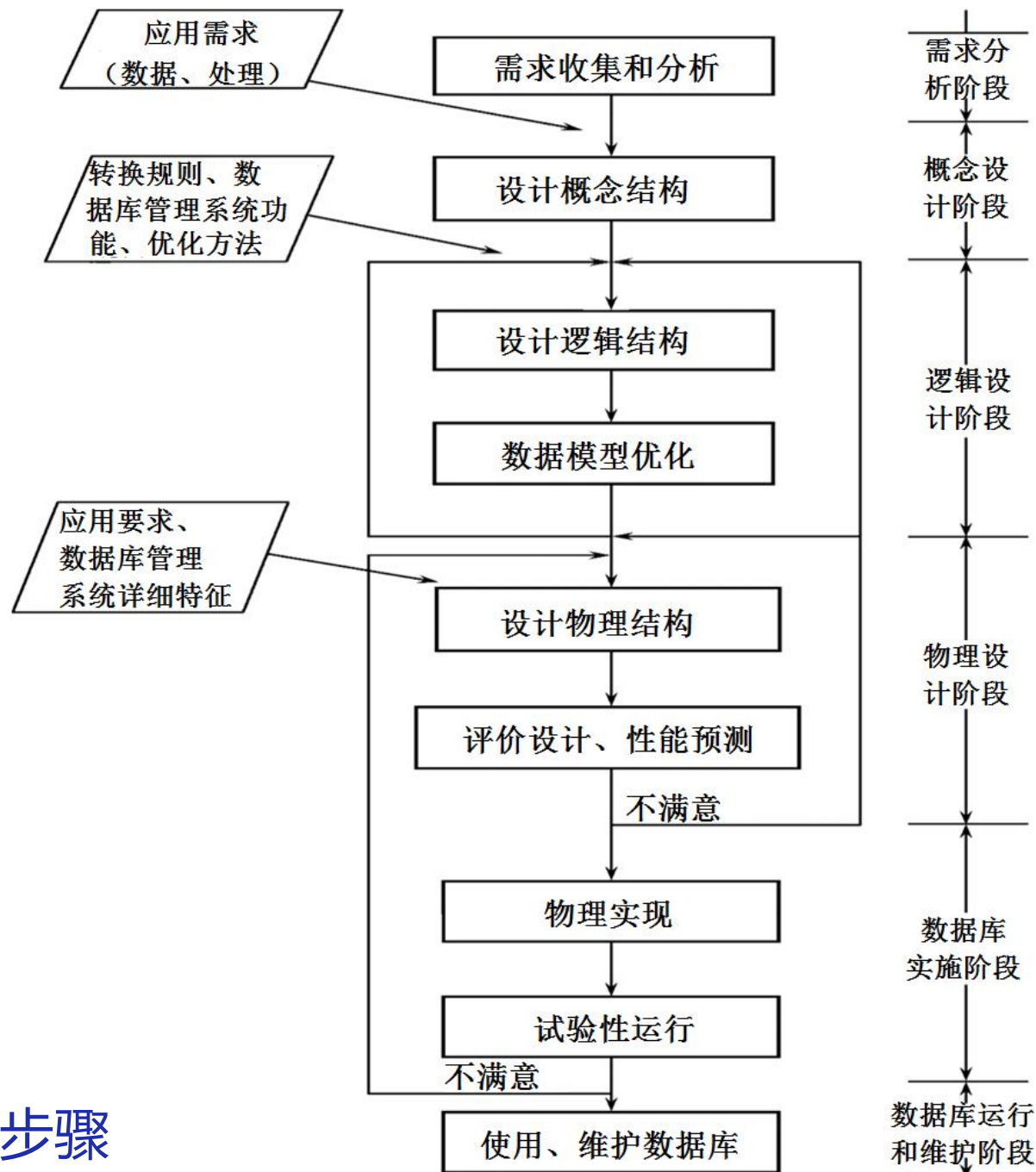


图7.2 数据库设计步骤

7.1.3 数据库设计的基本步骤 (2)

□ 参加数据库设计的人员

➤ 系统分析人员和数据库设计人员

- 自始至终参与数据库设计，其水平决定了数据库系统的质量

➤ 数据库管理员和用户代表

- 主要参加需求分析与数据库的运行和维护

➤ 应用开发人员

- 包括程序员和操作员
- 在实施阶段参与进来，分别负责编制程序和准备软硬件环境

7.1.3 数据库设计的基本步骤 (3)

1. 需求分析阶段

- 是否充分与准确，决定了构建数据库的速度和质量

2. 概念结构设计阶段

- 通过对用户需求进行综合、归纳与抽象，形成一个独立于具体数据库管理系统的概念模型

3. 逻辑结构设计阶段

- 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

4. 物理结构设计阶段

- 为逻辑数据结构选取一个最适合应用环境的物理结构，包括存储结构和存取方法

5. 数据库实施阶段


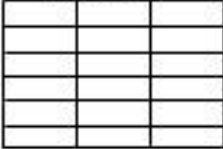
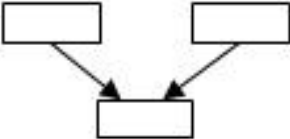


- 根据逻辑设计和物理设计的结果构建数据库
- 编写与调试应用程序
- 组织数据入库并进行试运行

6. 数据库运行和维护阶段

- 经过试运行后即可投入正式运行
- 在运行过程中必须不断对其进行评估、调整与修改

- ❑ 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。

图7.3 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	数字字典、全系统中数据项、数据结构、数据流、数据存储的描述
概念结构设计	概念模型 (E-R 图)  数据字典
逻辑结构设计	某种数据模型 关系  非关系 
物理结构设计	存储安排 存取方法选择 存取路径建立 
数据库实施	创建数据库模式 装入数据 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

7.1.4 数据库设计过程中的各级模式 (1)

❑ 数据库设计不同阶段形成的数据库各级模式

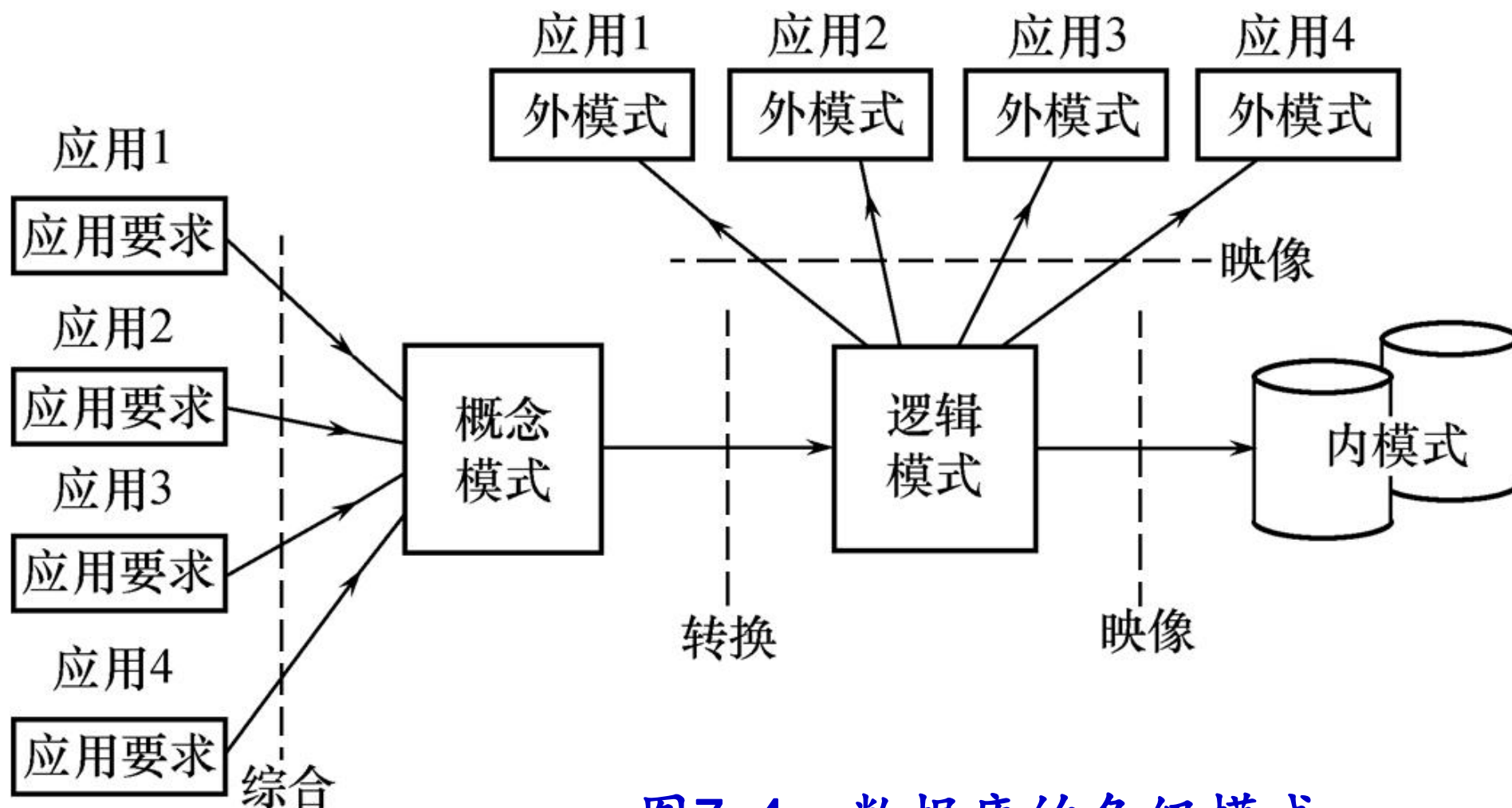
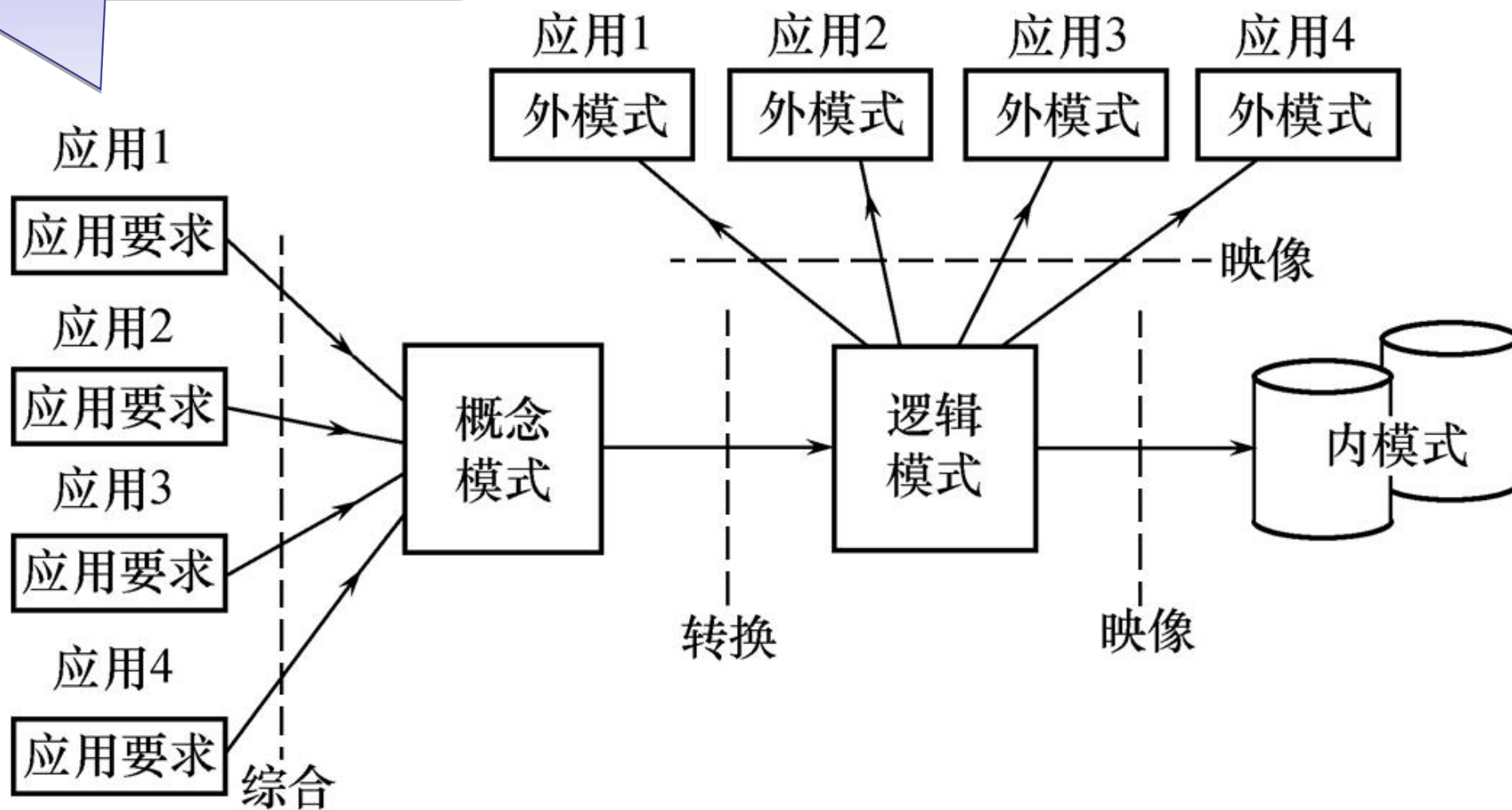


图7.4 数据库的各级模式

7.1.4 数据库设计过程中的各级模式 (2)

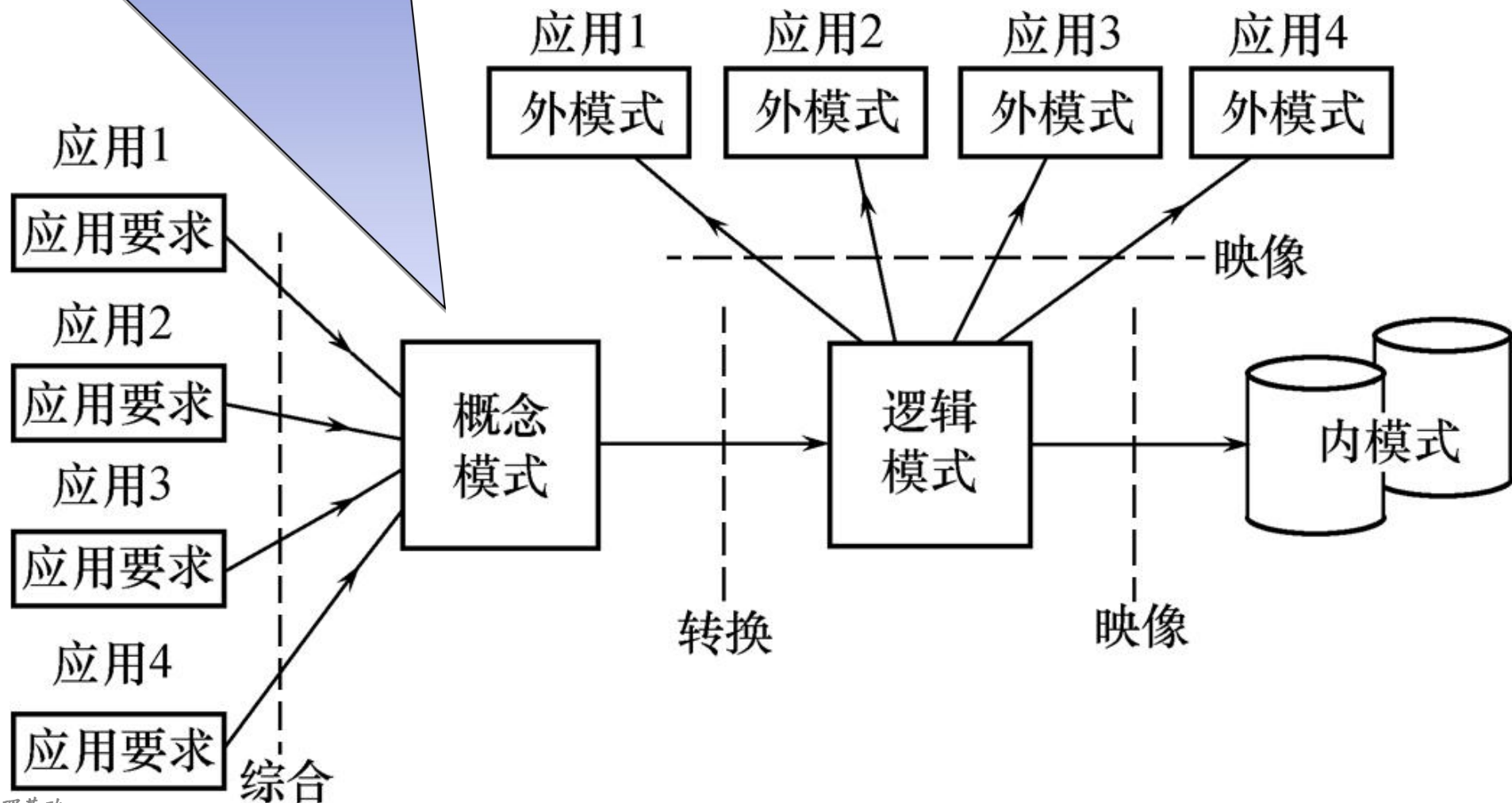
需求分析阶段：
综合各个用户的应用需求



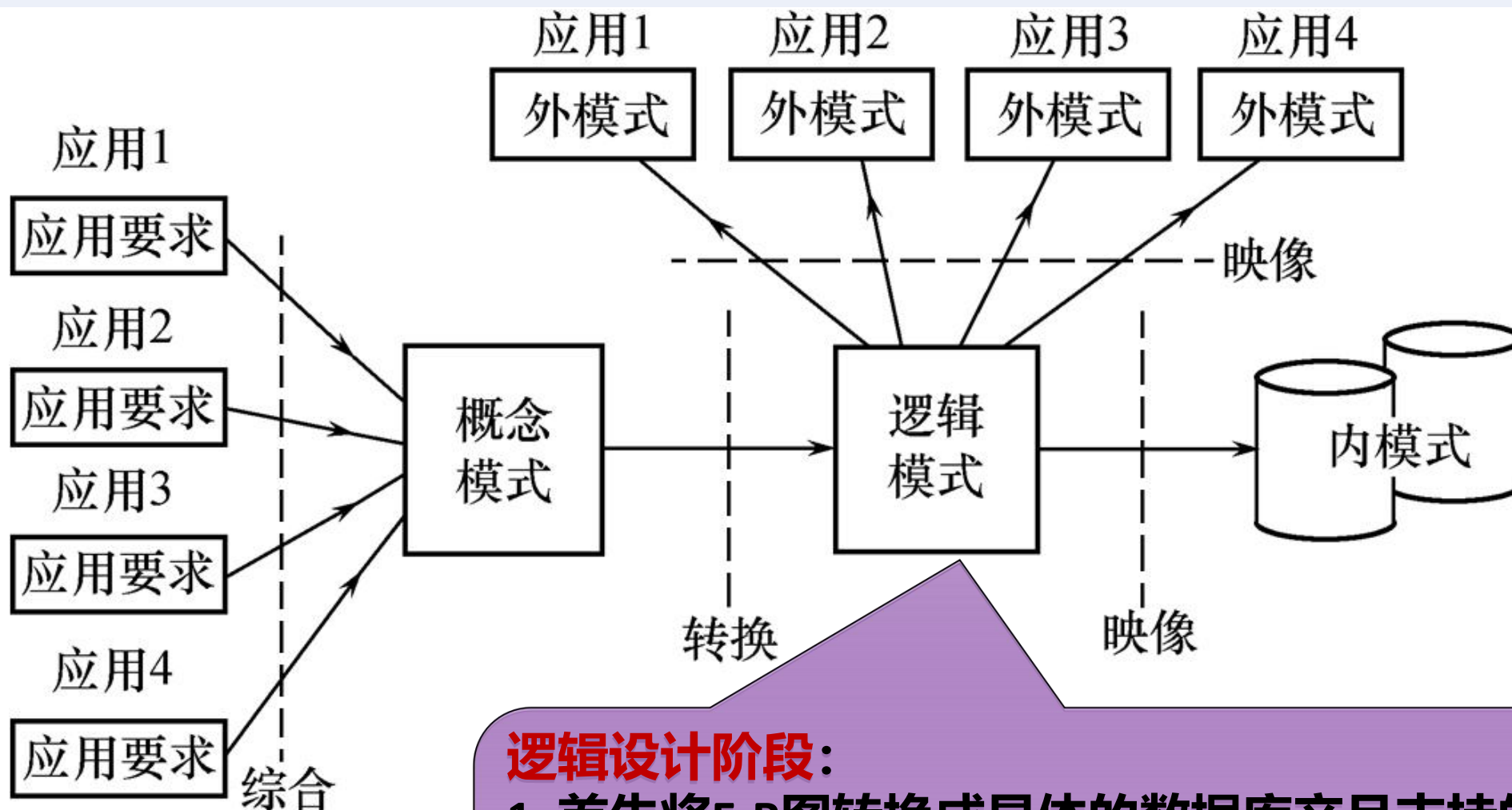
7.1.4 数据库设计过程中的各级模式 (3)

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)



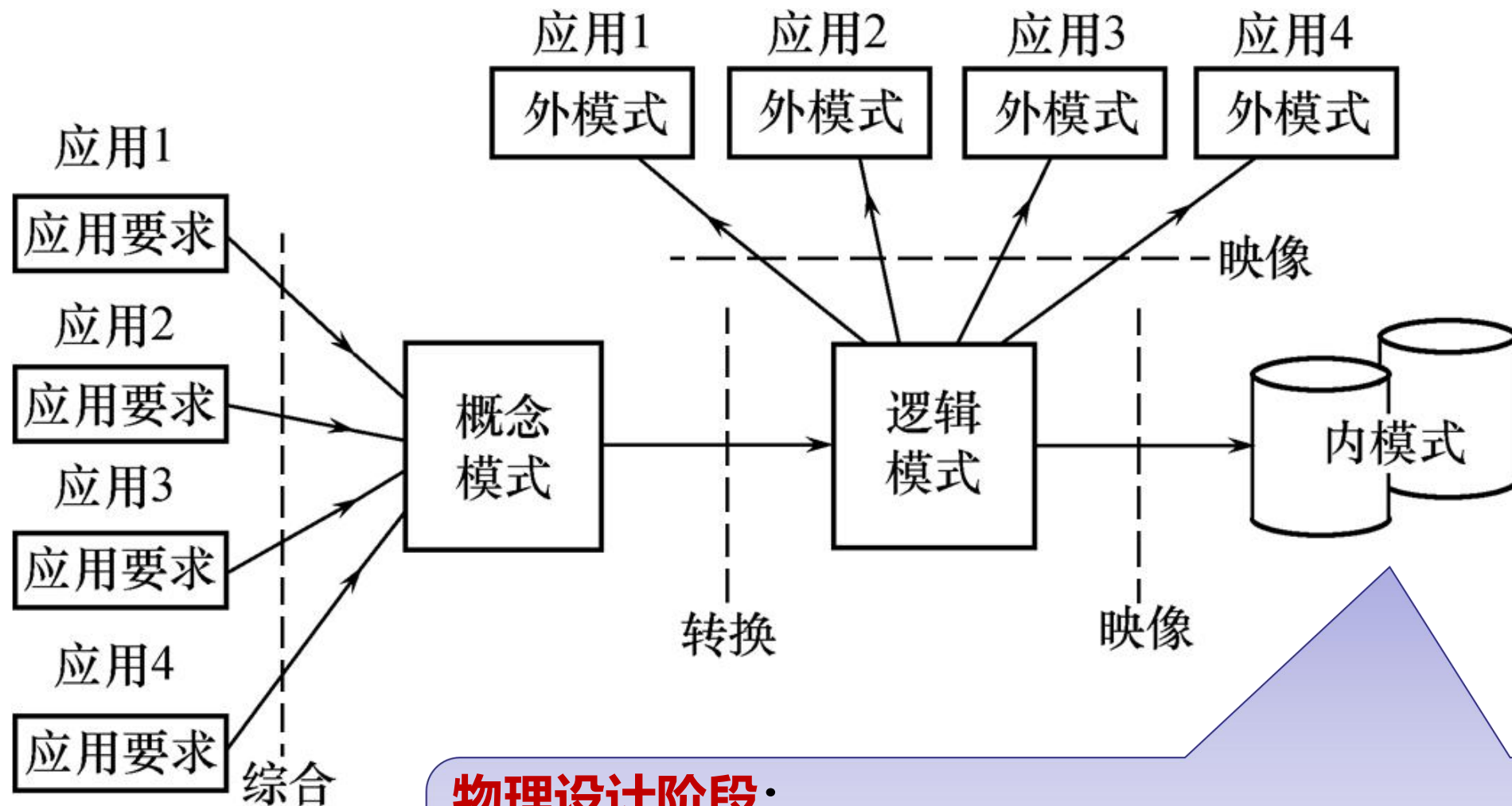
7.1.4 数据库设计过程中的各级模式 (4)



逻辑设计阶段:

1. 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**
2. 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**

7.1.4 数据库设计过程中的各级模式 (5)



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

数据管理基础

7.2 需求分析

智能软件与工程学院

7.2 需求分析

7.2.1 需求分析的任务

7.2.2 需求分析的方法

7.2.3 数据字典

❑ 需求分析就是分析用户的要求

- 是设计数据库的起点

- 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

7.2.1 需求分析的任务 (1)

- ❑ 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

7.2.1 需求分析的任务 (2)

□ 调查的重点是“数据”和“处理”，获得用户对数据库的要求

① 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

② 处理要求

- 用户要完成的处理功能
- 对处理性能的要求

③ 安全性与完整性要求

7.2.1 需求分析的任务 (3)

□ 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，不易理解用户的真正需求，甚至误解用户的需求

□ 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

7.2.2 需求分析的方法

- ❑ 调查清楚用户的实际需求并进行初步分析
- ❑ 与用户达成共识
- ❑ 分析与表达这些需求

调查用户需求的步骤

1. 调查组织机构情况

- 了解组织的部门组成情况、各部门的职责等，为分析信息流程做准备。

2. 调查各部门的业务活动情况

- 了解各部门输入和使用什么数据
- 如何加工这些数据
- 输出什么信息，输出到什么部门，输出结果的格式是什么等

3. 在熟悉业务活动的基础上，协助用户明确对新系统的各种要求

- 包括信息要求、处理要求、完全性与完整性要求等

4. 确定新系统的边界

- 对前面调查的结果进行初步分析，确定哪些功能由计算机完成或将来准备由计算机完成，哪些活动由人工完成

常用调查方法

① 跟班作业

- 通过亲身参加业务工作了解业务活动的情况

② 开调查会

- 通过与用户座谈来了解业务活动情况及用户需求

③ 请专人介绍

④ 询问

- 对某些调查中的问题，可以找专人询问

⑤ 设计调查表请用户填写

- 如调查表设计合理，则很有效

⑥ 查阅记录

- 查阅与原系统有关的数据记录

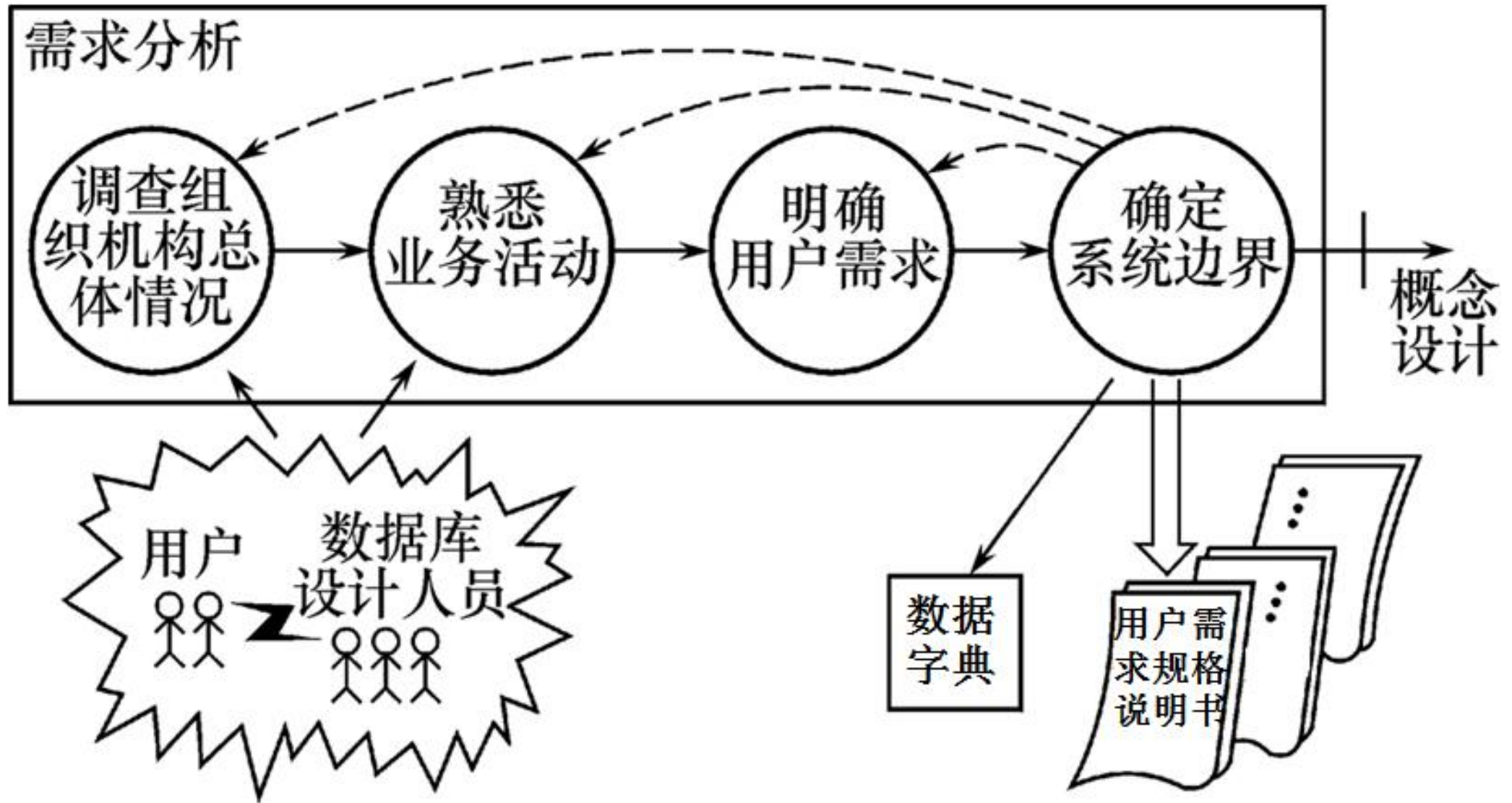
□ 分析方法

➤ 结构化分析方法（Structured Analysis，简称SA方法）

- SA方法从最上层的系统组织机构入手
- 采用自顶向下、逐层分解的方式分析系统

□ 对用户需求进行分析与表达后，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



7.2.3 数据字典

- ❑ 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结果
- ❑ 注意：和关系数据库管理系统中数据字典的区别和联系

❑ 数据字典的内容

- 数据项
- 数据结构
- 数据流
- 数据存储
- 处理过程

❑ 数据项是数据的最小组成单位

❑ 若干个数据项可以组成一个数据结构

❑ 数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容

1. 数据项

❑ 数据项是不可再分的数据单位

❑ 对数据项的描述

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
- 可以用关系规范化理论为指导，用数据依赖的概念分析和表示数据项之间的联系

2. 数据结构

- ❑ 数据结构反映了数据之间的组合关系。
- ❑ 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} }

3. 数据流

❑ 数据流是数据结构在系统内传输的路径。

❑ 对数据流的描述

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, 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

4. 数据存储

❑ 数据存储是数据结构停留或保存的地方，也是数据流来源和去向之一。

❑ 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，每次存取的数据量等信息
- 存取方法：批处理 / 联机处理；检索 / 更新；顺序检索 / 随机检索
- 输入的数据流：数据来源
- 输出的数据流：数据去向

5. 处理过程

□ 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

需求分析小结

- ❑ 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

数据管理基础

7.3 概念结构设计 (概念模型和ER模型)

智能软件与工程学院

7.3 概念结构设计

7.3.1 概念模型

7.3.2 E-R模型

7.3.3 扩展的E-R模型

7.3.4 UML

7.3.5 概念结构设计

❑ 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

❑ 概念模型的特点

- ① 能真实、充分地反映现实世界，是现实世界的一个真实模型。
- ② 易于理解，从而可以用它和不熟悉计算机的用户交换意见。
- ③ 易于更改，当应用环境 and 应用要求改变时，容易对概念模型修改和扩充。
- ④ 易于向关系、网状、层次等各种数据模型转换

❑ 描述概念模型的工具

➤ E-R模型

□ （回顾） 1.2.2 概念模型

□ 实体 (Entity)

- 客观存在并可相互区别的事物称为实体。可以是具体的人、事、物或抽象的概念。
- 每一个实体有一个‘**实体名**’，同类实体具有相同的实体名。

□ 属性 (Attribute)

- 实体具有的某一特性称为属性。一个实体可以由若干个属性来刻画。
- 每一个属性有一个‘**属性名**’，在同一个实体内属性名互不相同。

□ 联系 (Relationship)

- 现实世界中事物内部以及事物之间的联系，在信息世界中反映为**实体（型）内部的联系**和**实体（型）之间的联系**。
- 每一个联系有一个‘**联系名**’

□ 实体型 (Entity Type)

- 用实体名及其所有属性名的集合来抽象和刻画同类实体
- 实体的实体名及其所有属性名的集合，称为该实体的‘**实体型**’

□ 实体值 (Entity Value)

- 实体中的属性可以有值，一个实体的所有属性值的集合被称为该实体的‘**实体值**’。（也被称为‘**实体实例**’， **Entity Instance**）

□ 实体集 (Entity Set)

- 同一类型实体的集合，称为‘**实体集**’
- 同一个实体集中的所有实体，具有相同的实体型但实体值互不相同。

□ 码 (Key)

- 唯一标识实体的属性集称为码。
- ‘**码**’也被称为**关键字**、**键**、**标识符**(Identifier)
- 在一个实体集中，可以通过‘**码**’的取值来区分不同的实体。

两个实体型之间的联系

□ 一对一联系 (1:1)

- 如果对于实体集A中的每一个实体，实体集B中至多有一个（也可以没有）实体与之联系，反之亦然，则称实体集A与实体集B具有一对一联系，记为1:1。

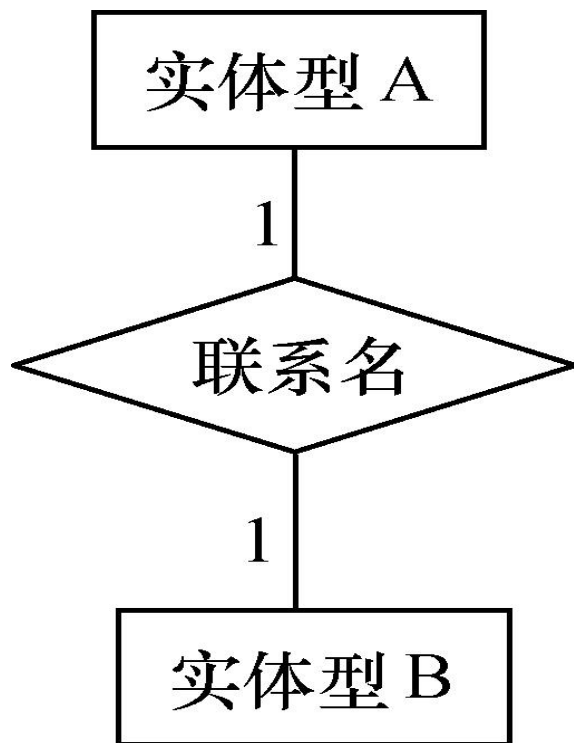
□ 一对多联系 (1:n)

- 如果对于实体集A中的每一个实体，实体集B中有n个实体 ($n \geq 0$) 与之联系，反之，对于实体集B中的每一个实体，实体集A中至多只有一个实体与之联系，则称实体集A与实体集B有一对多联系，记为1:n。

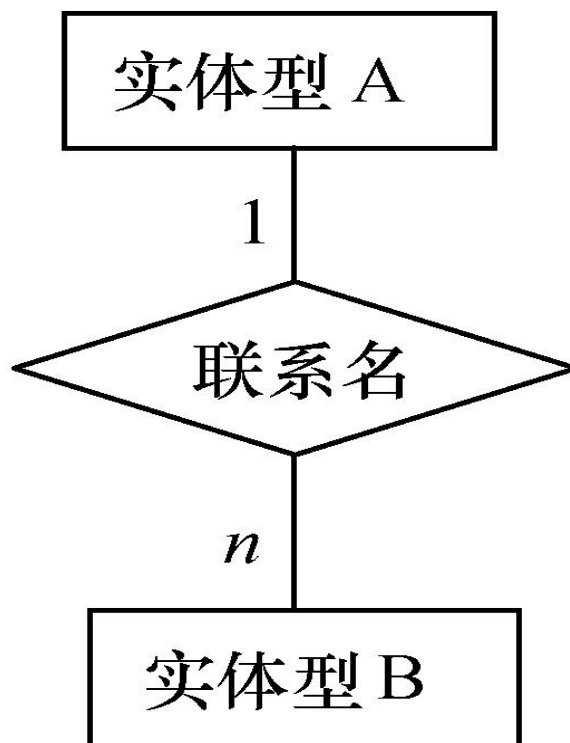
□ 多对多联系 (m:n)

- 如果对于实体集A中的每一个实体，实体集B中有n个实体 ($n \geq 0$) 与之联系，反之，对于实体集B中的每一个实体，实体集A中也有m个实体 ($m \geq 0$) 与之联系，则称实体集A与实体集B具有多对多联系，记为m:n。

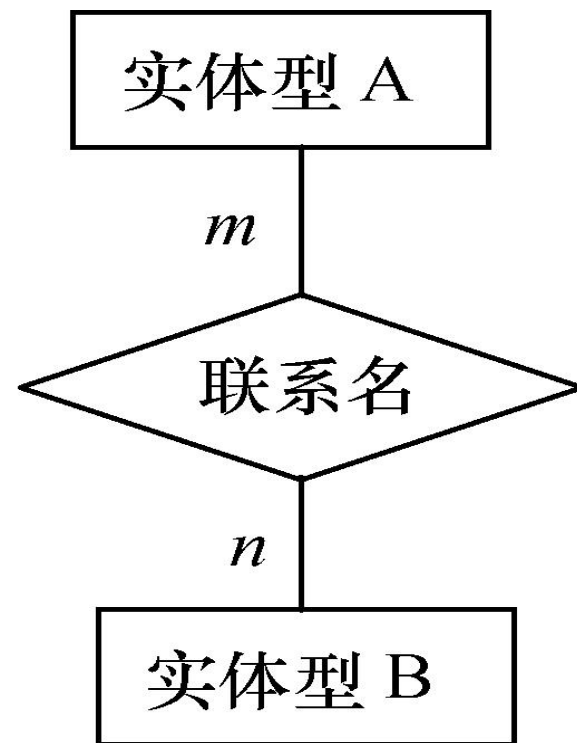
图7.6 两个实体型之间的三类联系



(a) 1:1 联系



(b) 1:n 联系



(c) m:n 联系

➤ ‘一对一’联系的例子

- ‘居民’ vs. ‘身份证’
- ‘学生’ vs. ‘学生证’

➤ 例如（从左到右‘一对多’）

- 住宿：‘宿舍’ vs. ‘学生’
- 拥有：‘人’ vs. ‘银行卡’

➤ ‘多对多’联系的例子

- 选修：‘学生’ vs. ‘课程’
- 乘坐：‘旅客’ vs. ‘航班’

两个以上的实体型之间的联系

□ 一般地，两个以上的实体型之间也存在着一对一、一对多、多对多联系。

□ 例如：

- 图7.7(a)：一门课程可以有若干个教师讲授、使用若干本参考书，而每一个教师只讲授一门课程，每一本参考书只供一门课程使用，则课程与教师、参考书之间是一对多的联系。
- 图7.7(b)：一个供应商可以向多个项目供应多种零件，每个项目可以使用多个供应商供应的多种零件，每种零件可由不同供应商供给多个项目，供应商、项目、零件三者之间是多对多的联系。

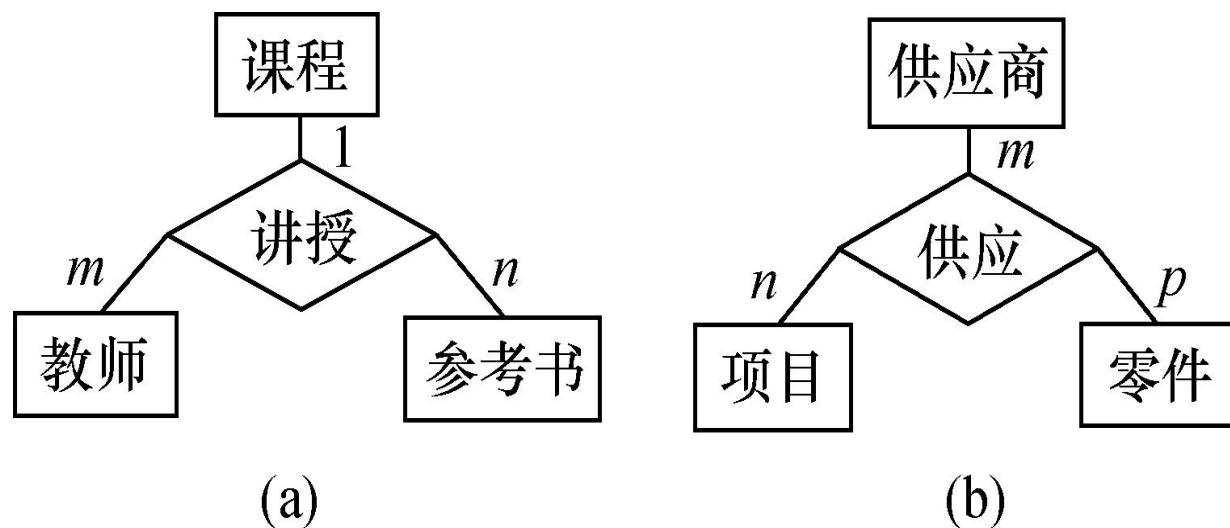


图7.7 三个实体型之间的联系示例

单个实体型内的联系

- 同一个实体集内的各实体之间也可以存在一对一、一对多、多对多的联系。
- 例如，职工实体型内部具有领导与被领导的联系，即某一职工（干部）“领导”若干名职工，而一个职工仅被另外一个职工直接领导，因此这是一对多的联系，如图7.8所示。

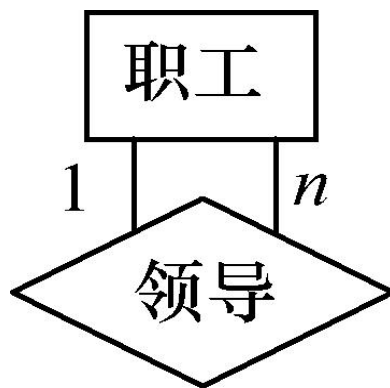


图7.8 单个实体型内的一对多联系示例

联系的度

□ **联系的度**：参与联系的实体型的数目

- 2个实体型之间的联系度为2，也称为二元联系；
- 3个实体型之间的联系度为3，称为三元联系；
- N个实体型之间的联系度为N，也称为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，矩形框内写明实体名。
- **属性**：用**椭圆形**表示，并用无向边将其与相应的实体型连接起来。
 - 例如，学生实体具有学号、姓名、性别、出生年份、系、入学时间等属性，用E-R图表示如图7.9所示

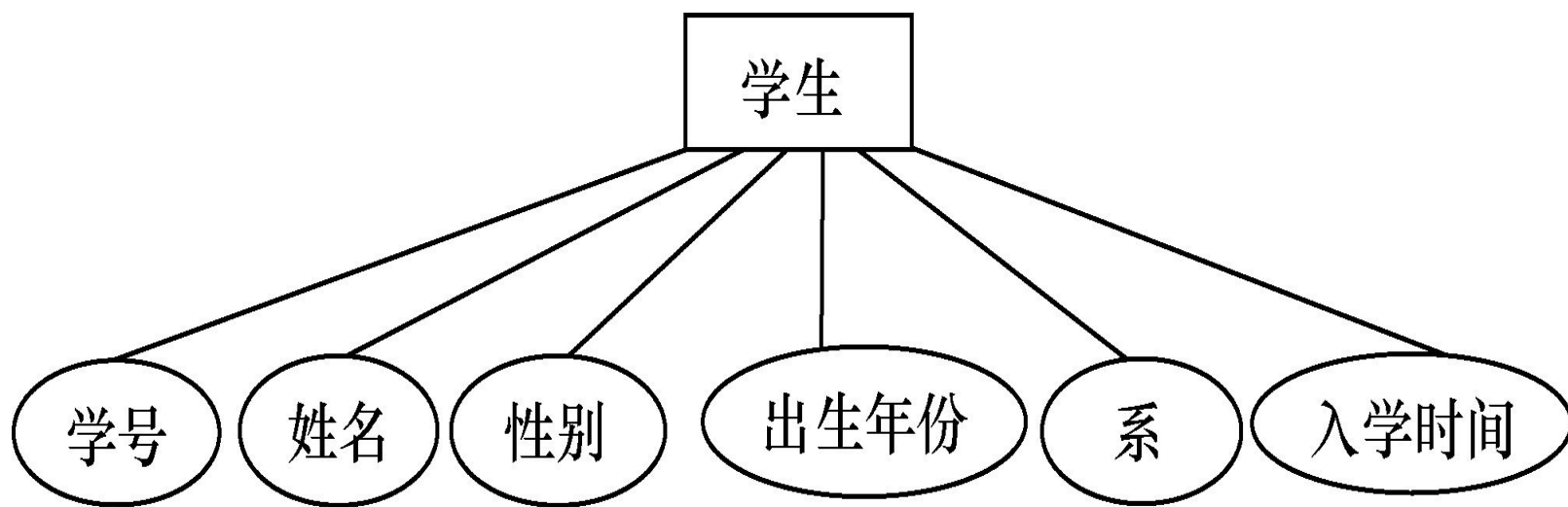


图7.9 学生实体及属性

□ 联系

- 用菱形表示，菱形框内写明联系名
- 用无向边分别与有关实体型连接起来，同时也在无向边旁标上联系的类型（1:1, 1:n 或 m:n）
- 联系可以具有属性

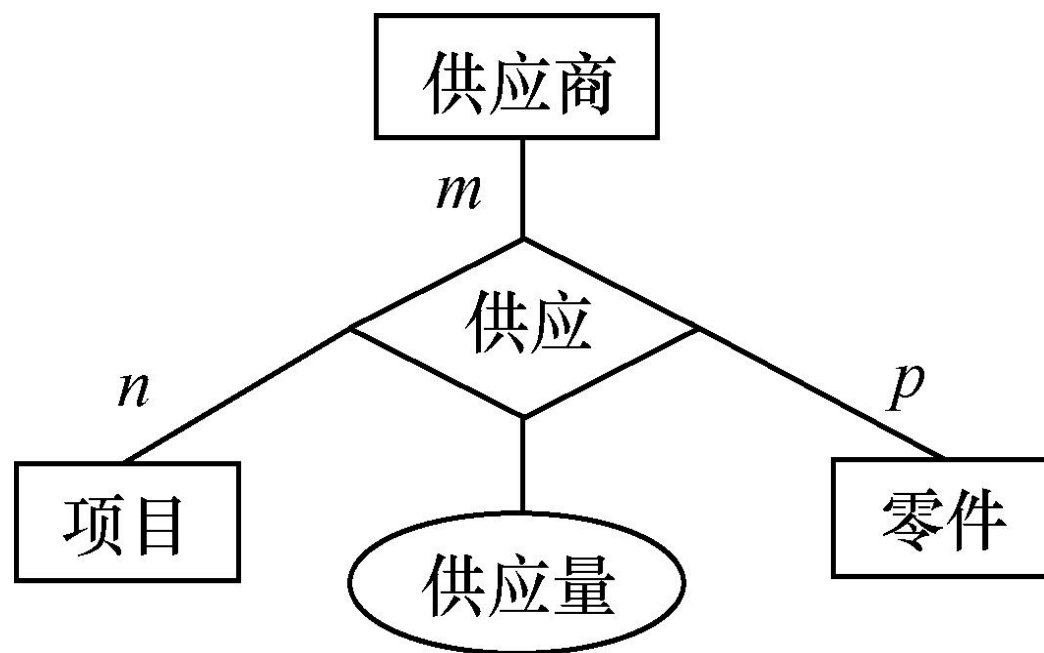
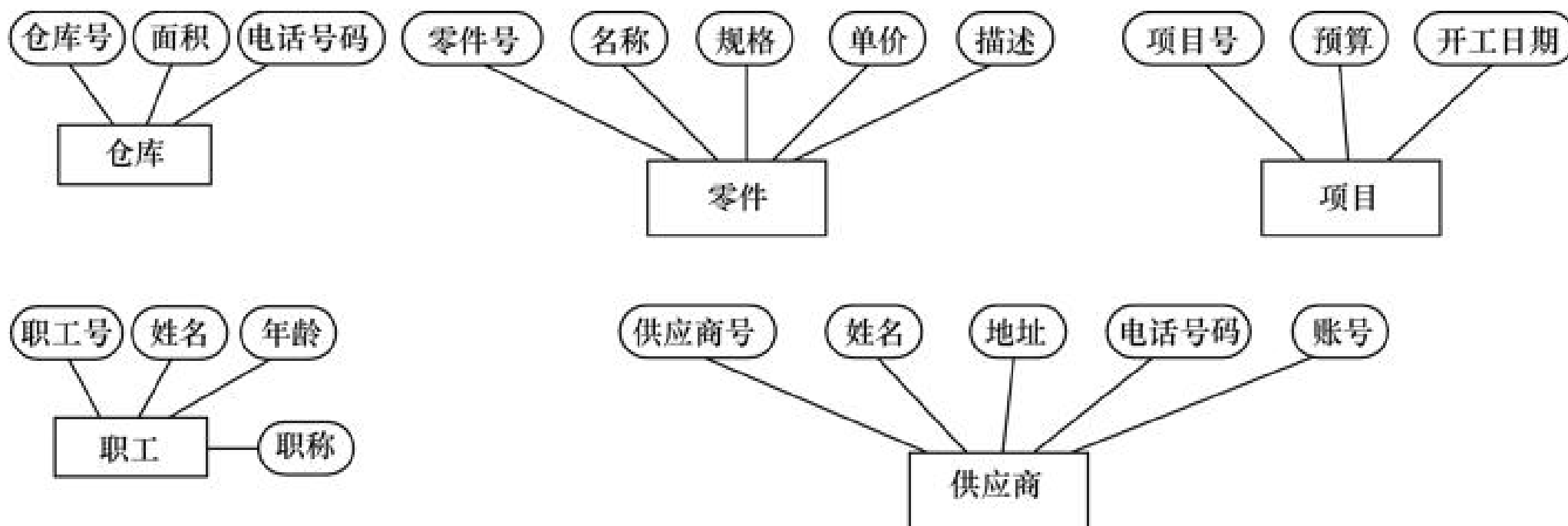


图7.10 联系的属性

E-R图实例 1

□ 某个工厂物资管理的概念模型。物资管理涉及的实体有：

- 仓库：属性有仓库号、面积、电话号码
- 零件：属性有零件号、名称、规格、单价、描述
- 供应商：属性有供应商号、姓名、地址、电话号码、账号
- 项目：属性有项目号、预算、开工日期
- 职工：属性有职工号、姓名、年龄、职称

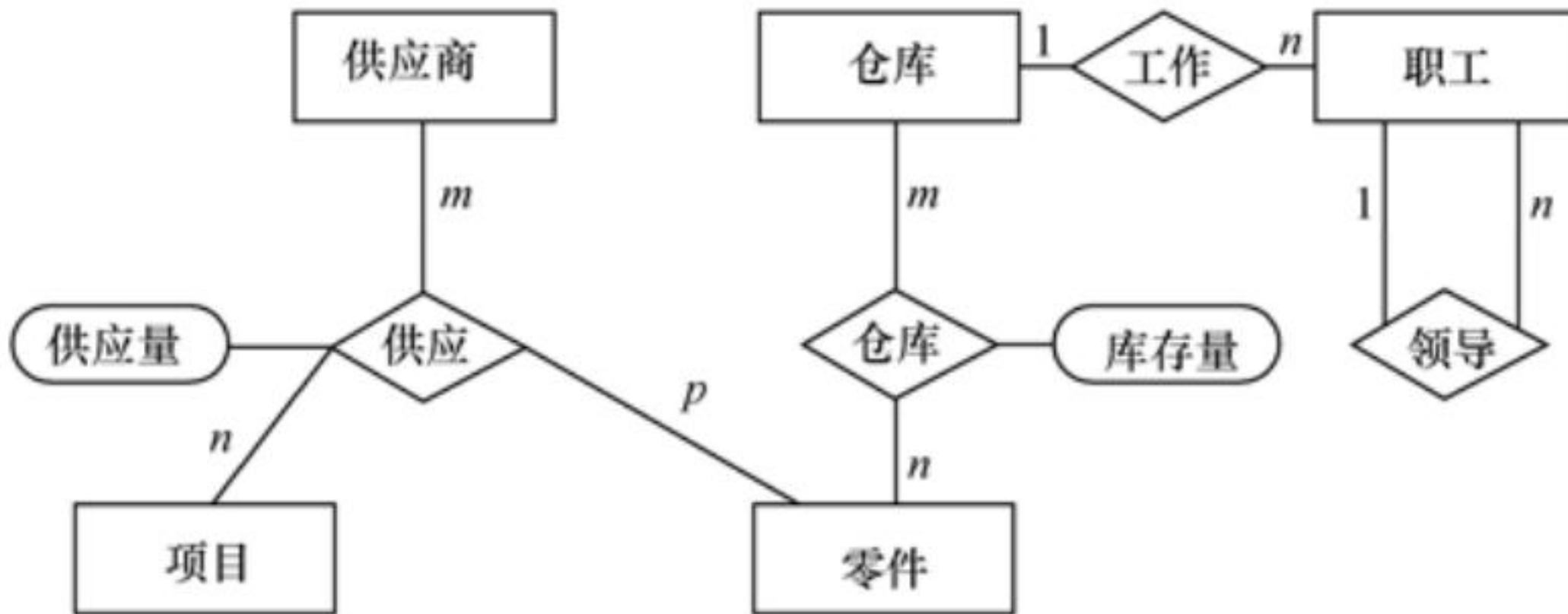


(a) 实体及其属性图

□ 这些实体之间的联系如下：

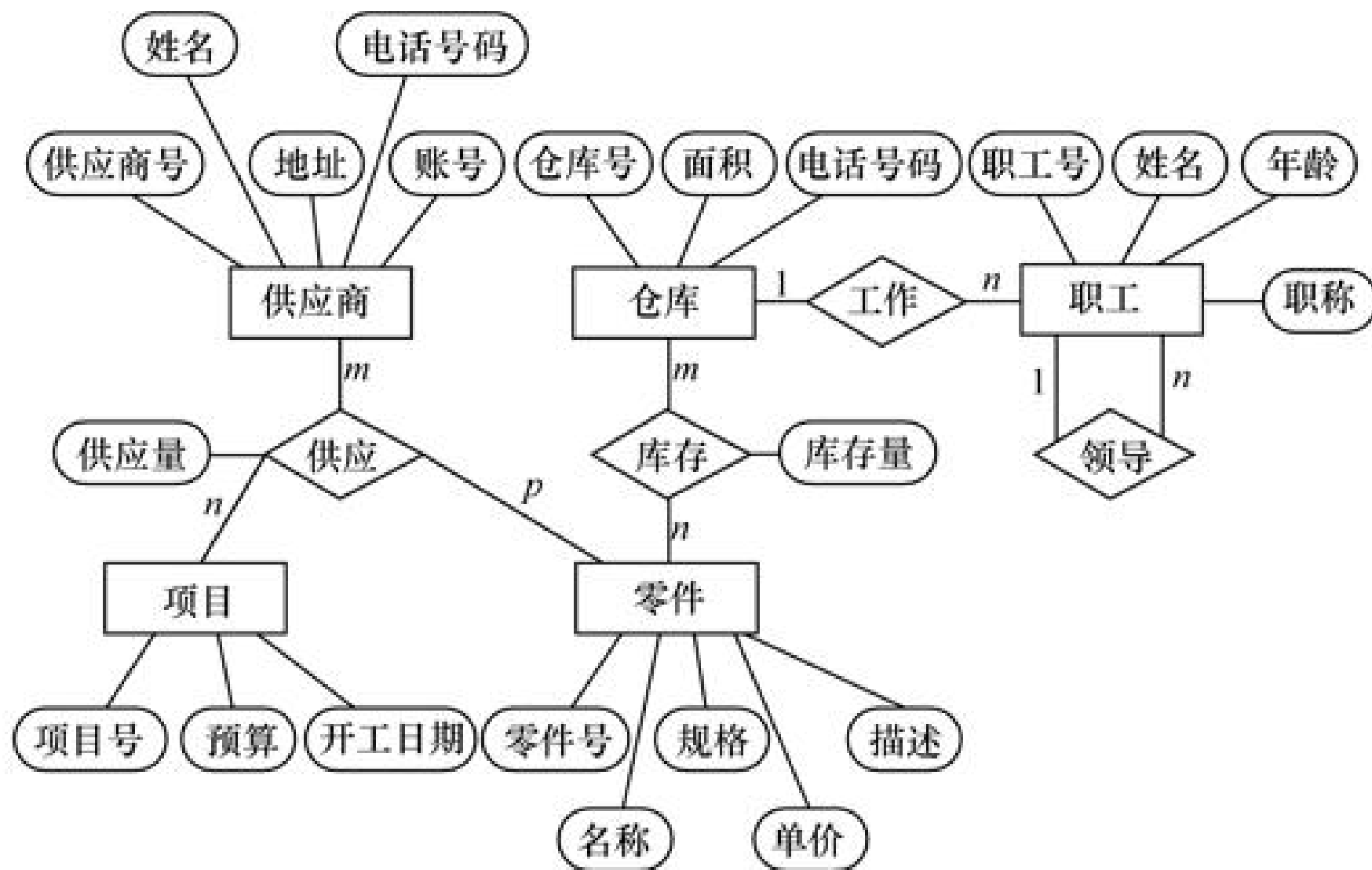
- ① 一个仓库可以存放多种零件，一种零件可以存放在多个仓库中，因此仓库和零件具有多对多的联系。用库存量来表示某种零件在某个仓库中的数量。
- ② 一个仓库有多个职工当仓库保管员，一个职工只能在一个仓库工作，因此仓库和职工之间是一对多的联系。
- ③ 职工之间具有领导与被领导关系。即仓库主任领导若干保管员，因此职工实体型中具有一对多的联系。
- ④ 供应商、项目和零件三者之间具有多对多的联系。即一个供应商可以供给若干项目多种零件，每个项目可以使用不同供应商供应的零件，每种零件可由不同供应商供给。

E-R图实例 3



(b) 实体及其联系图

E-R图实例 4



(c) 完整的实体-联系图

7.3.3 扩展的E-R模型

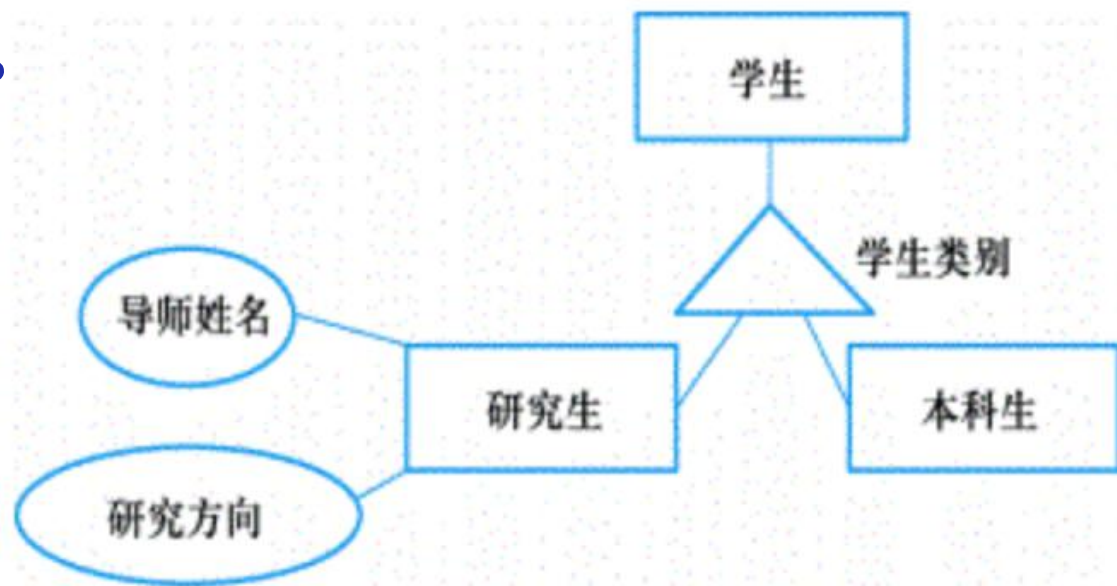
- ❑ ISA联系
- ❑ 基数约束
- ❑ part_of 联系
- ❑ 弱实体

ISA联系

❑ 有的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。用△表示。

❑ **ISA联系**的性质：

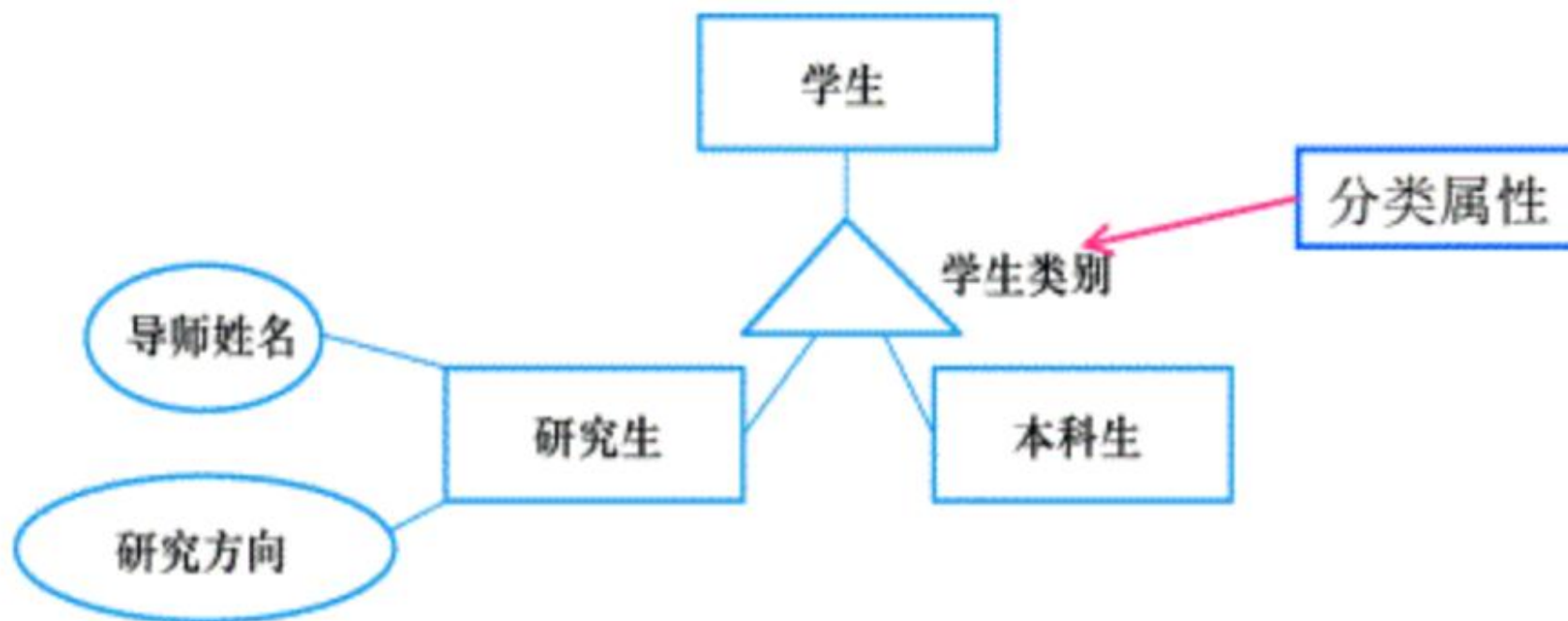
- 子类继承了父类的所有属性
- 子类也可以有自己的属性。



学生的2个子类

ISA联系-分类属性

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



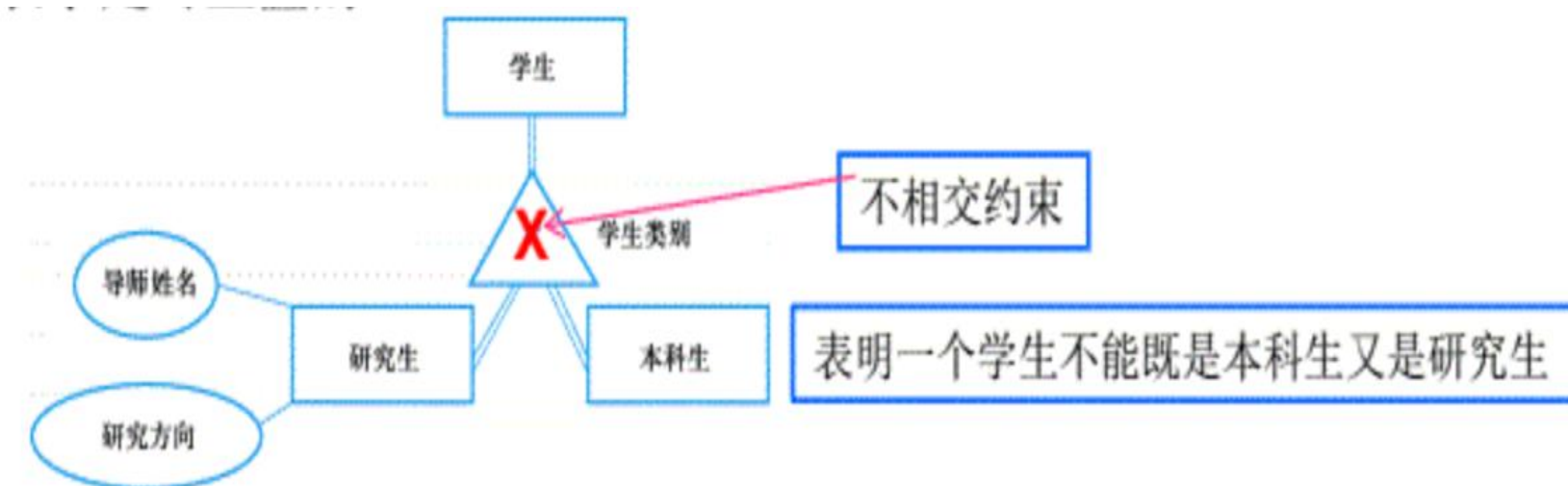
ISA联系-不相交约束与可重叠约束

□ 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。

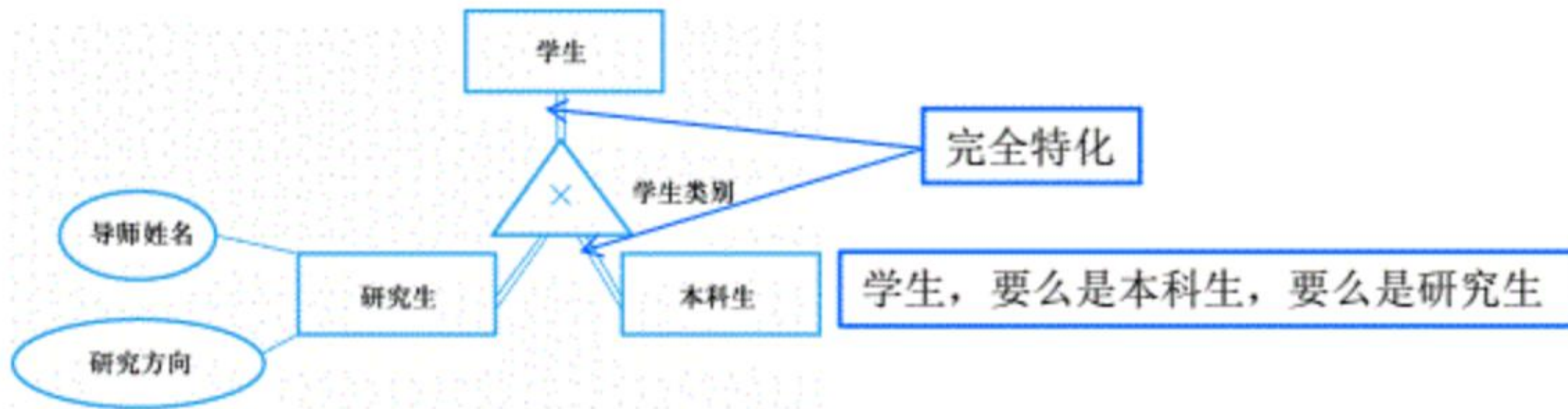
□ 可重叠约束:

- 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。



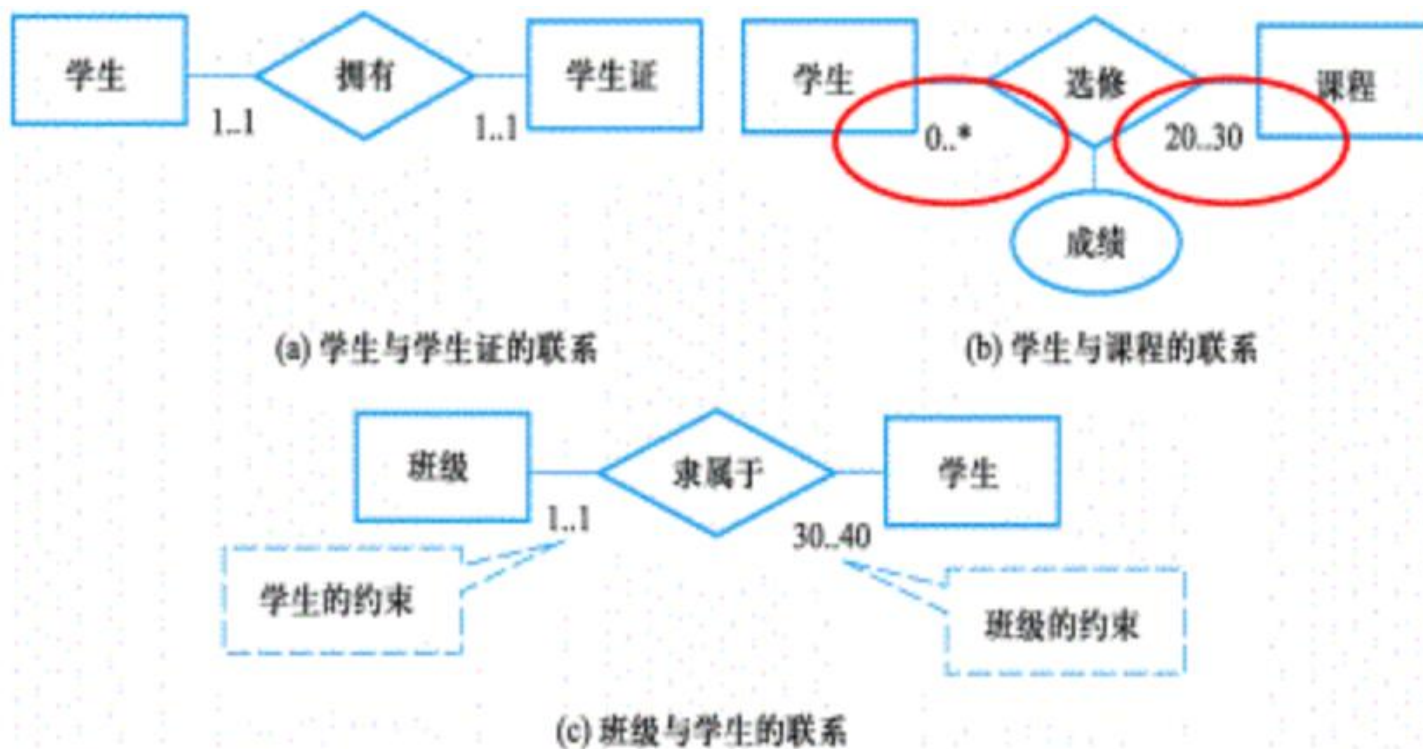
ISA联系-完备性约束

- ❑ 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**(total 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 1

- 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**。
- 是对实体之间一对一、一对多、多对多联系的细化。
- 约束用一个数对 $\text{min}..\text{max}$ 表示, $0 \leq \text{min} \leq \text{max}$ 。例如, $0..1$, $1..3$, $1..*$, , 其中 $*$ 代表无穷大。

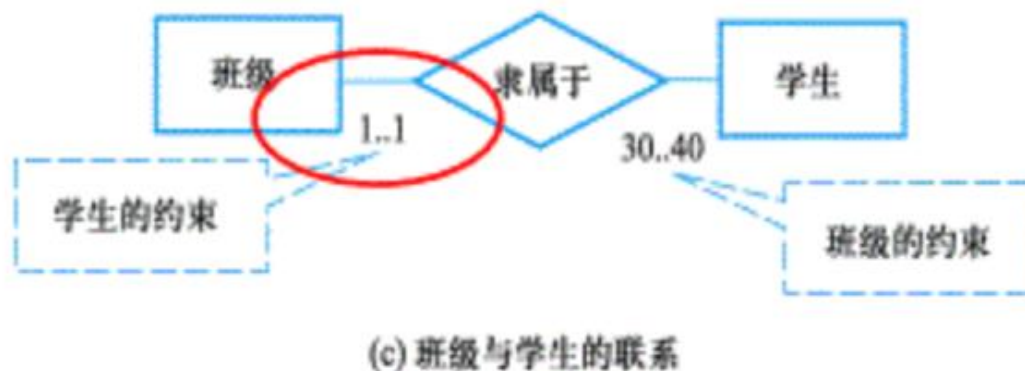
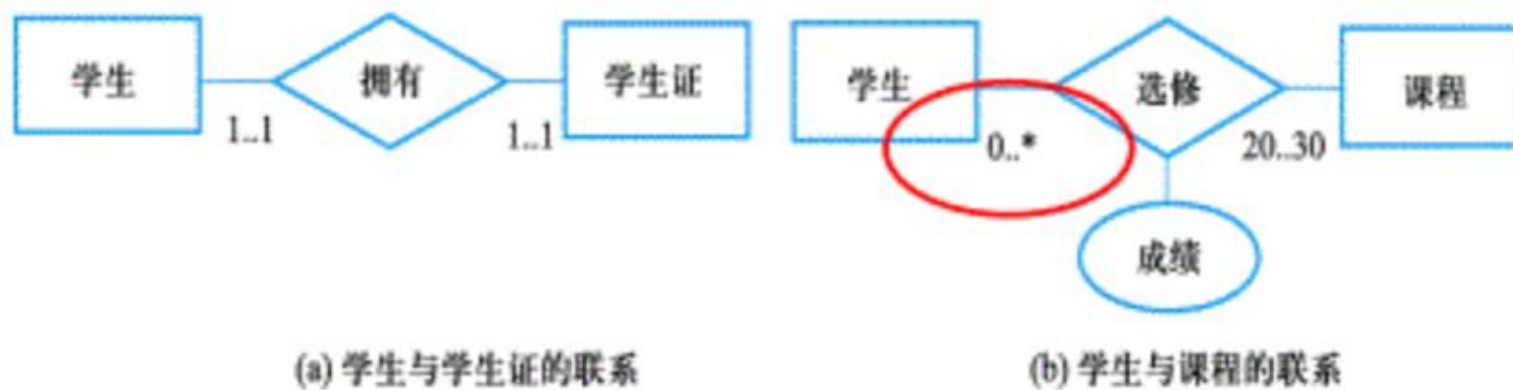


学生实体型的基数约束是 $20..30$ 表示每个学生必须选修20~30门课程:

课程的一个基数约束是 $0..*$, 即一门课程可以被很多同学选修也可能还没有同学选修, 如新开课。

基数约束 2

- ❑ $\min=1$ 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



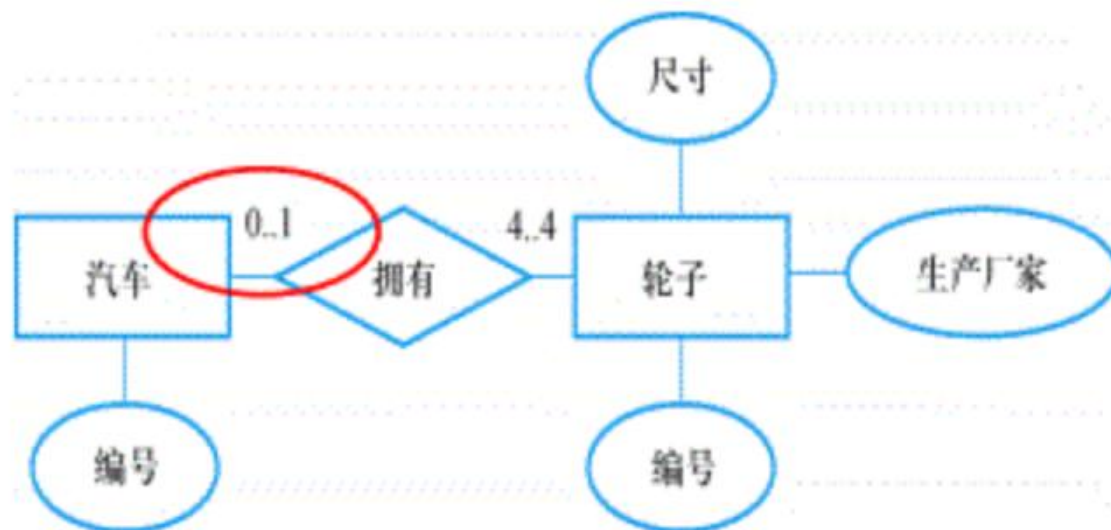
□ **Part-of 联系**：描述某个实体型是另外一个实体型的一部分。

□ **Part-of 联系**可以分为两种情况

- 非独占的**Part-of**联系，简称**非独占联系**
 - 整体实体如果被破坏，另一部分实体仍然可以独立存在
- 独占的**Part-of**联系，简称**独占联系**
 - 整体实体如果被破坏，部分实体不能存在

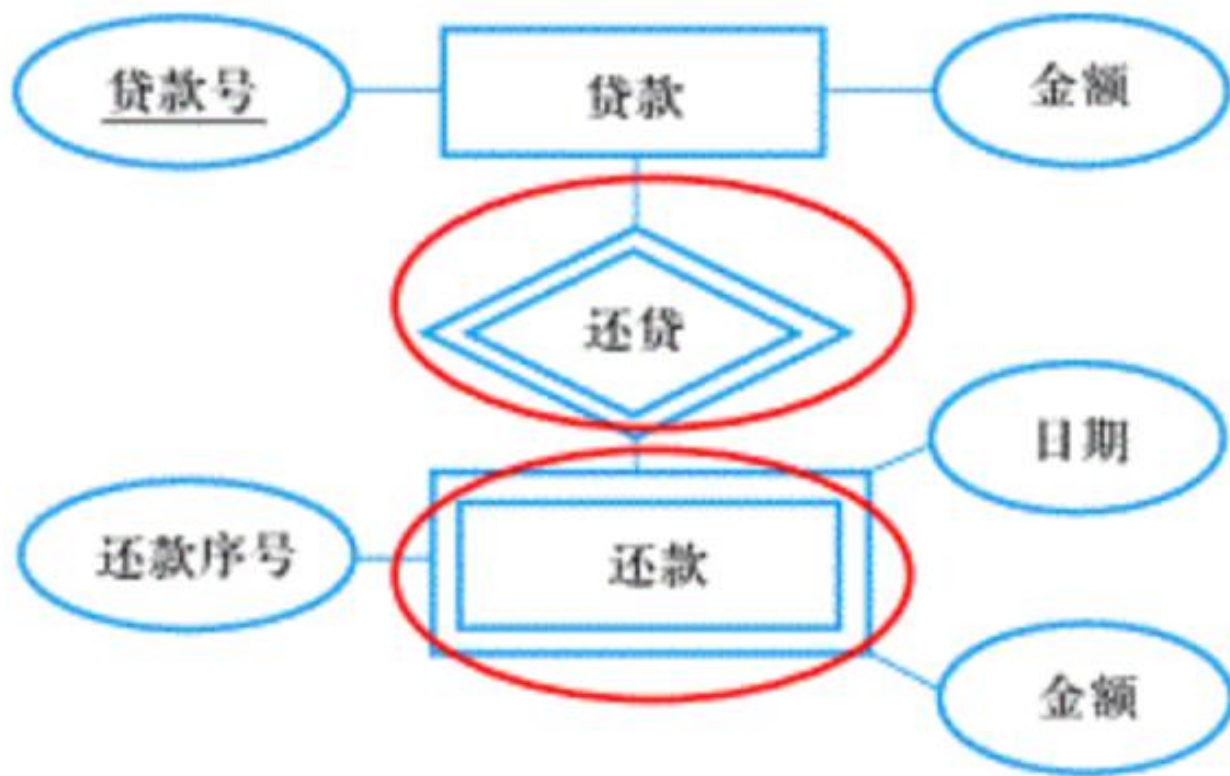
□ **Part-of联系**如何表示？

- 用非强制参与联系表示非独占的**Part-of**联系
- 用弱实体类型和识别联系来表示独占联系编号编号



弱实体型和独占联系

- ❑ 如果一个实体的存在依赖于其它实体的存在，则这个实体叫做弱实体型，否则叫做强实体型。
- ❑ 用弱实体类型和识别联系来表示独占联系双矩形表示弱实体型，用双菱形表示识别联系。



数据管理基础

7.3.5 概念结构设计 1

智能软件与工程学院

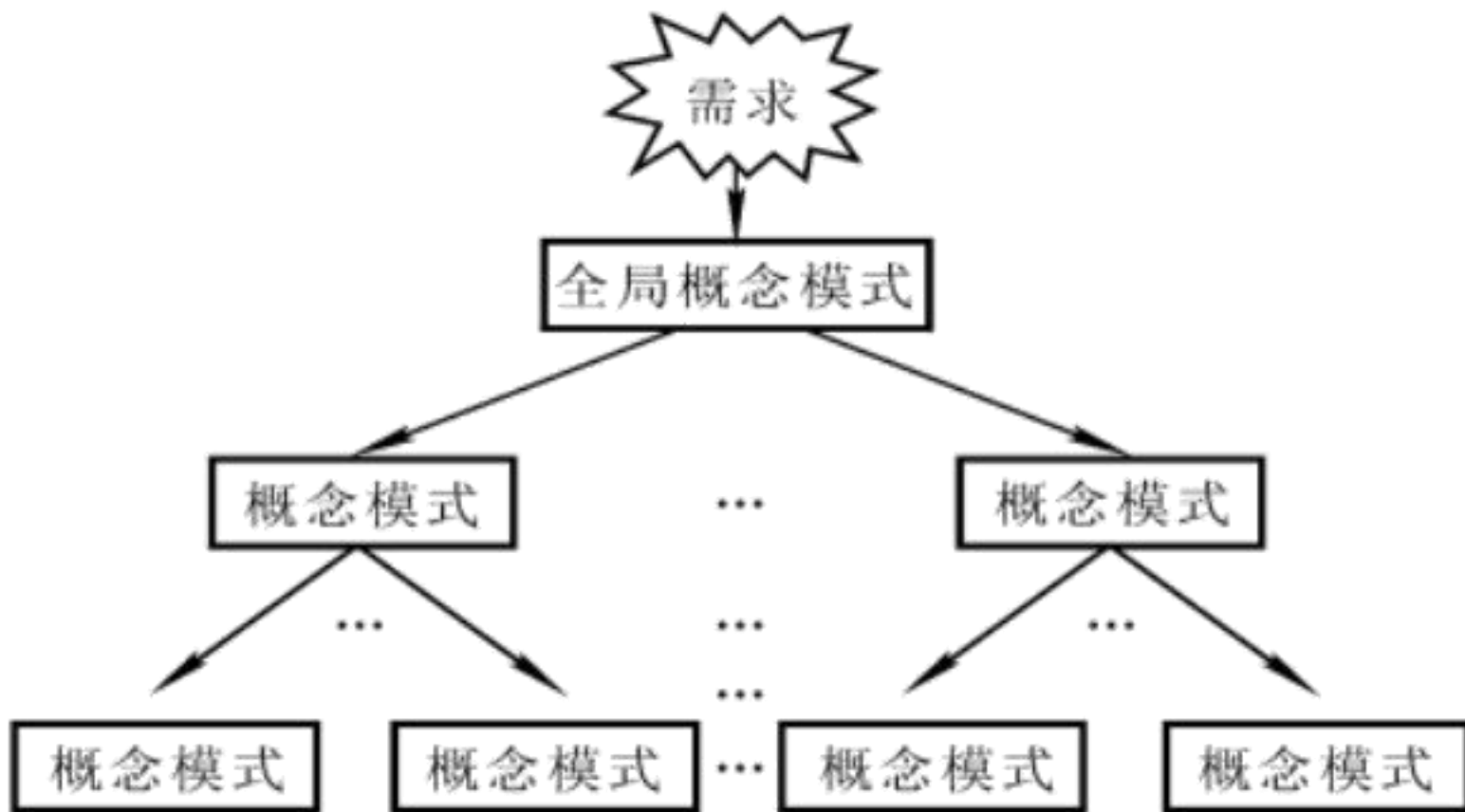
- ❑ 概念结构设计的方法
- ❑ 实体与属性的划分原则
- ❑ E-R图的集成

概念结构设计的方法

- ❑ 自顶向下
 - ❑ 由底向上
 - ❑ 逐步扩张
- } 混合策略

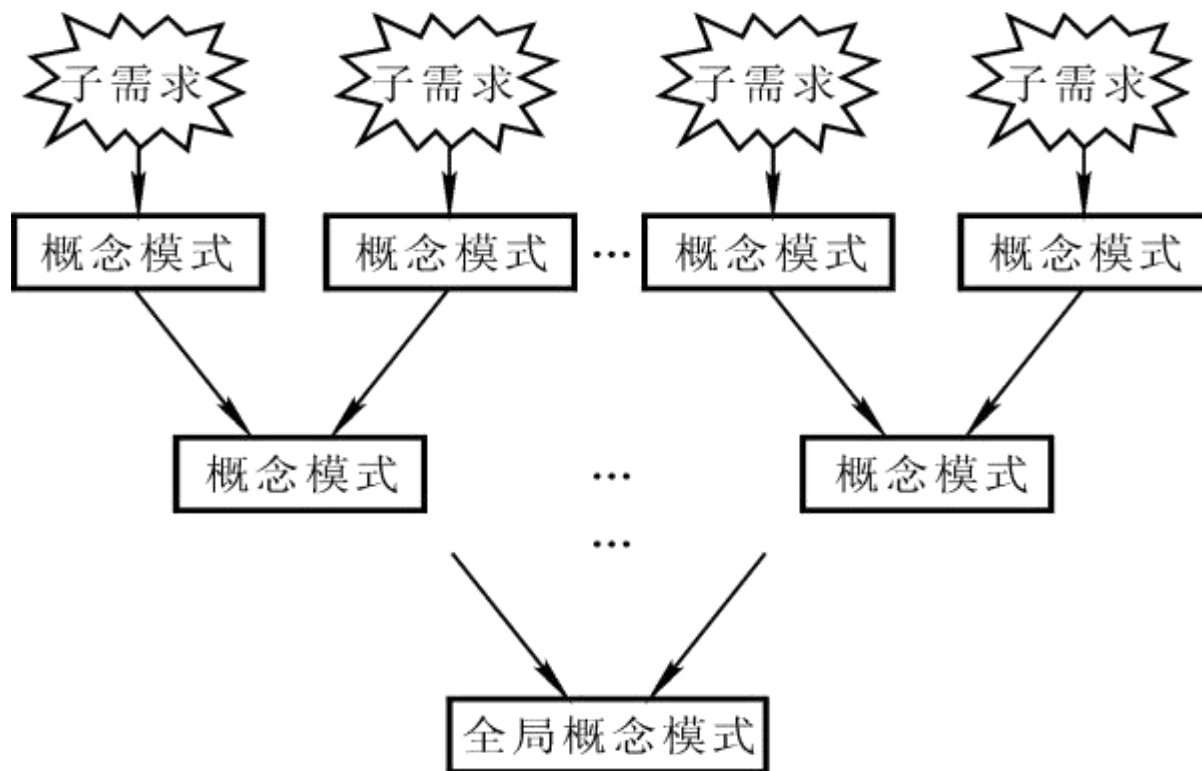
概念结构设计的方法-自顶向下

- 首先定义全局概念结构的框架，然后逐步细化



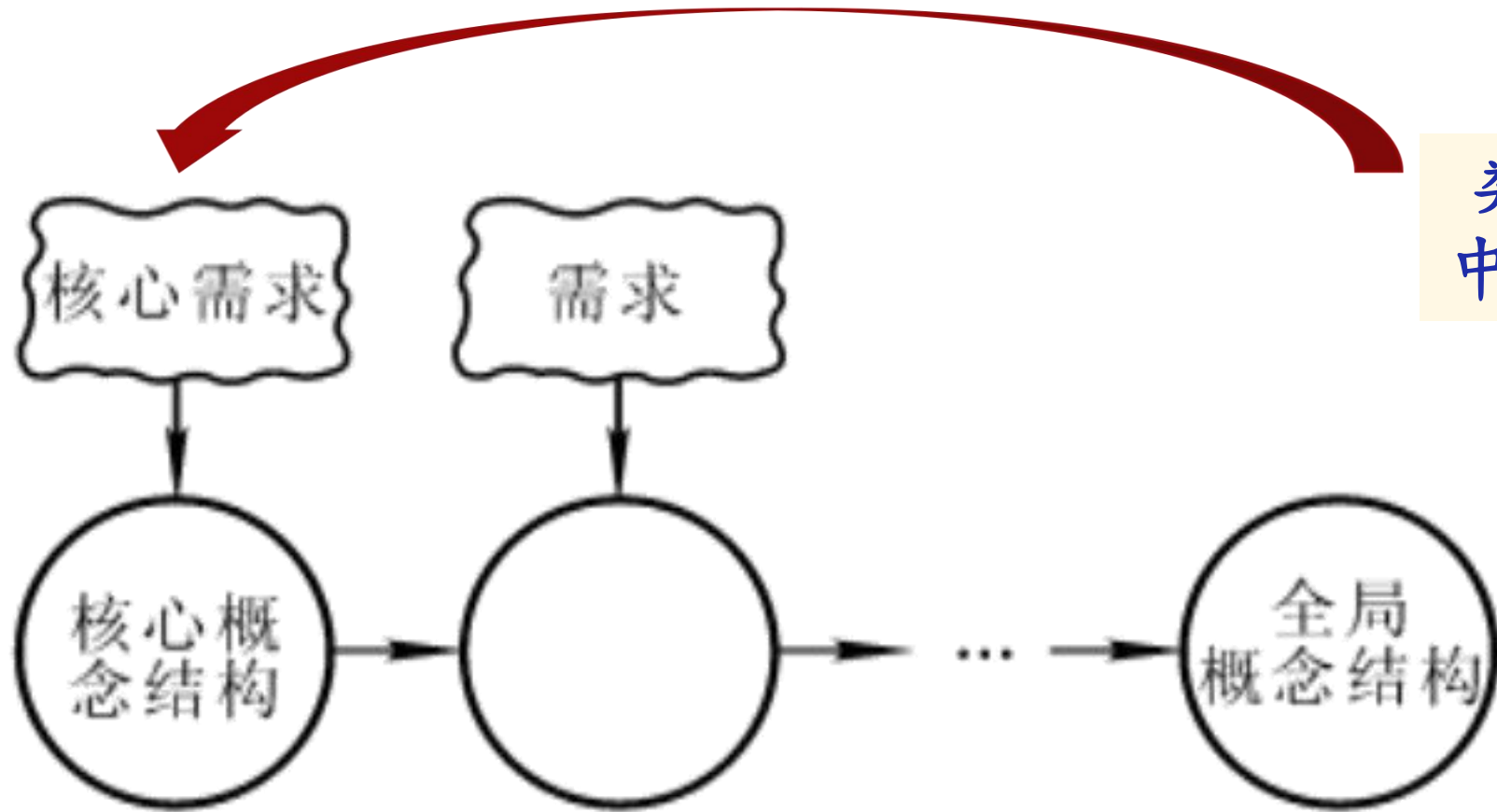
概念结构设计的方法-自底向上

- 首先定义各局部应用的概念结构，然后将它们集成起来，得到全局概念结构



概念结构设计的方法-逐步扩张

- 首先定义最重要的核心概念结构，然后向外扩充，以滚雪球的方式逐步生成其他概念结构，直至总体概念结构



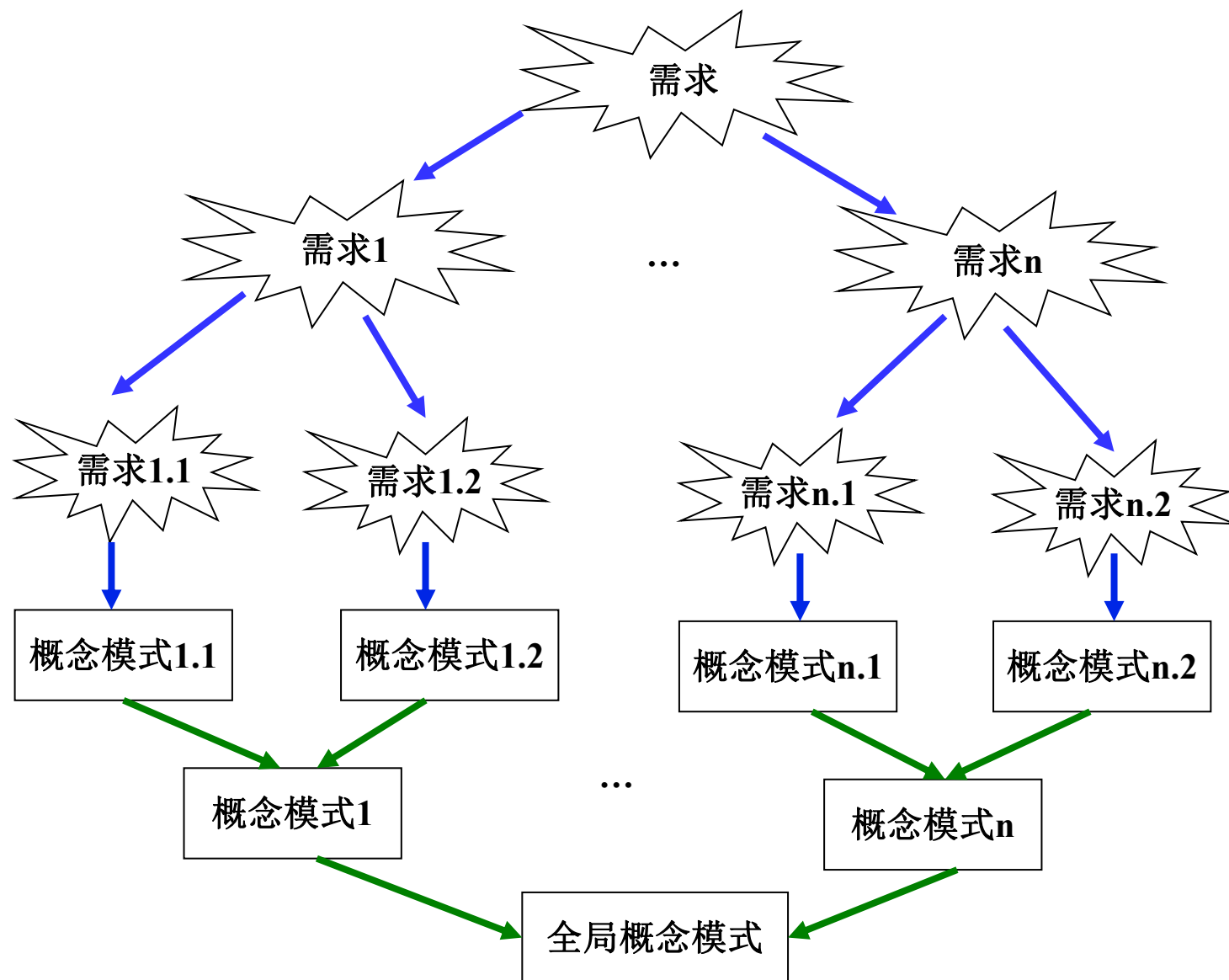
类似软件工程中的‘螺旋模型’

□ 将自顶向下和自底向上相结合，用自顶向下策略设计一个全局概念结构的框架，以它为骨架集成由自底向上策略中设计的各局部概念结构。

□ 常用策略

- 自顶向下地进行需求分析
- 自底向上地设计概念结构

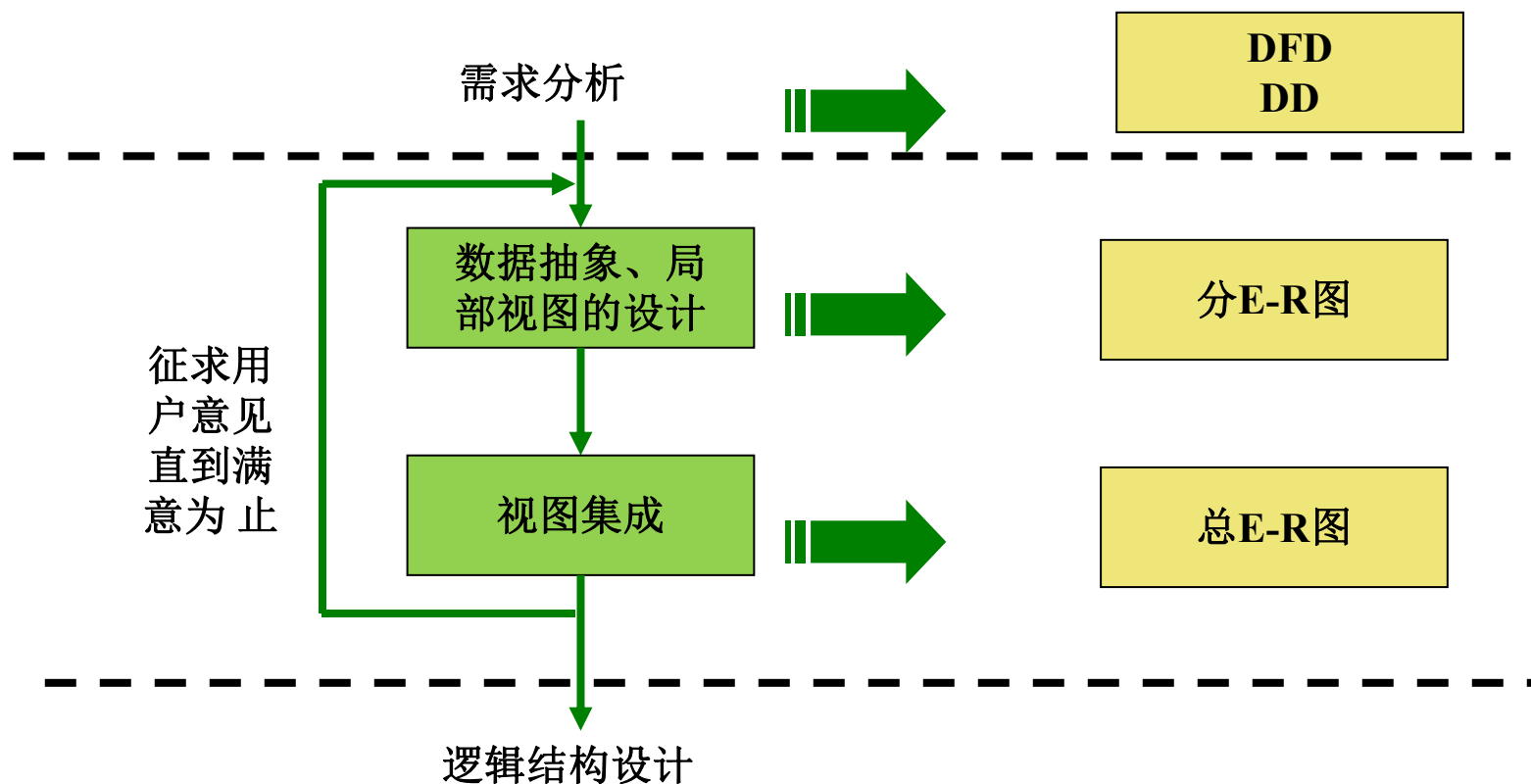
概念结构设计的方法-混合策略 2



概念结构设计的步骤

□ 自底向上设计概念结构的步骤

- 第1步：抽象数据并设计局部视图
- 第2步：集成局部视图，得到全局概念结构



- ❑ 概念结构设计的方法
- ❑ 实体与属性的划分原则
- ❑ E-R图的集成

实体与属性的划分 1

□ 实体与属性的划分原则

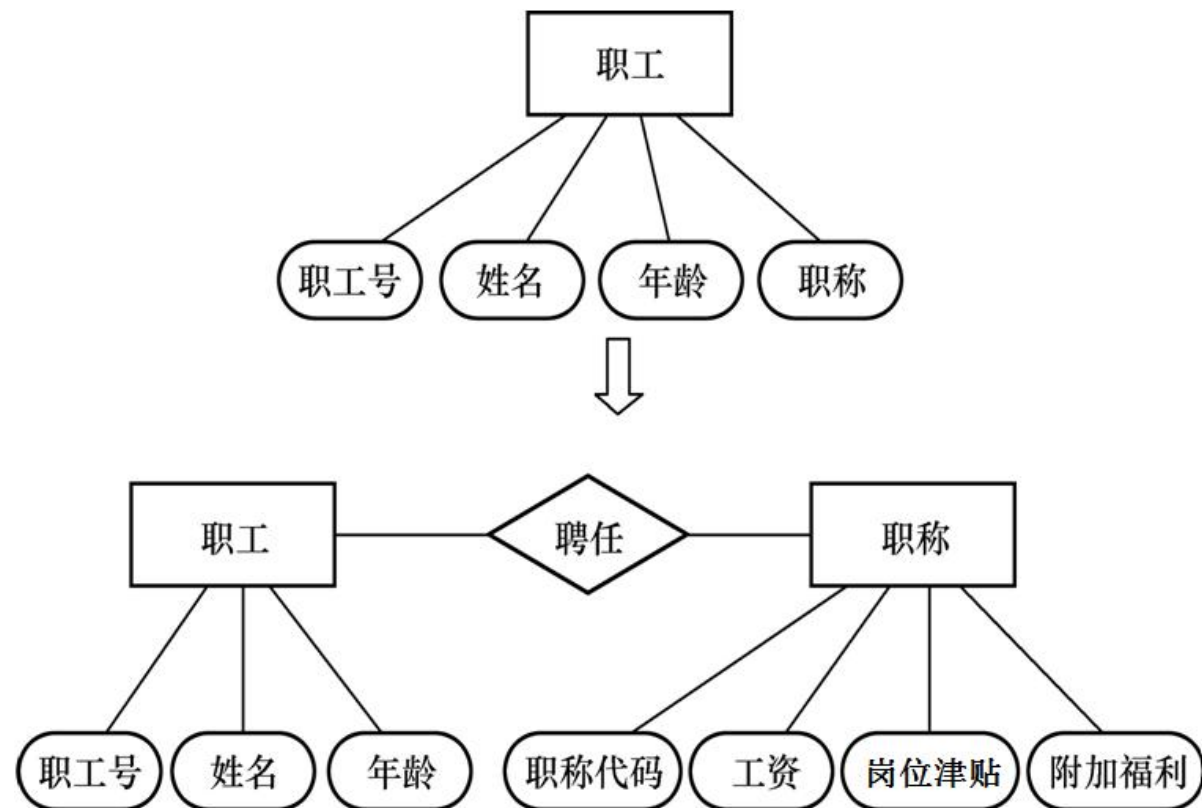
- 为了简化E-R图的处置，现实世界的事物能作为属性对待的，尽量作为属性对待。
- 两条准则：
 1. 作为属性，不能再具有需要描述的性质。属性必须是不可分的数据项，不能包含其他属性。
 2. 属性不能与其他实体具有联系，即E-R图中所表示的联系是实体之间的联系。
- 说明：在扩展E-R模型中，属性可以不必遵循原子性。

实体与属性的划分 2

❑ [例1] 职工是一个实体，职工号、姓名、年龄是职工的属性。

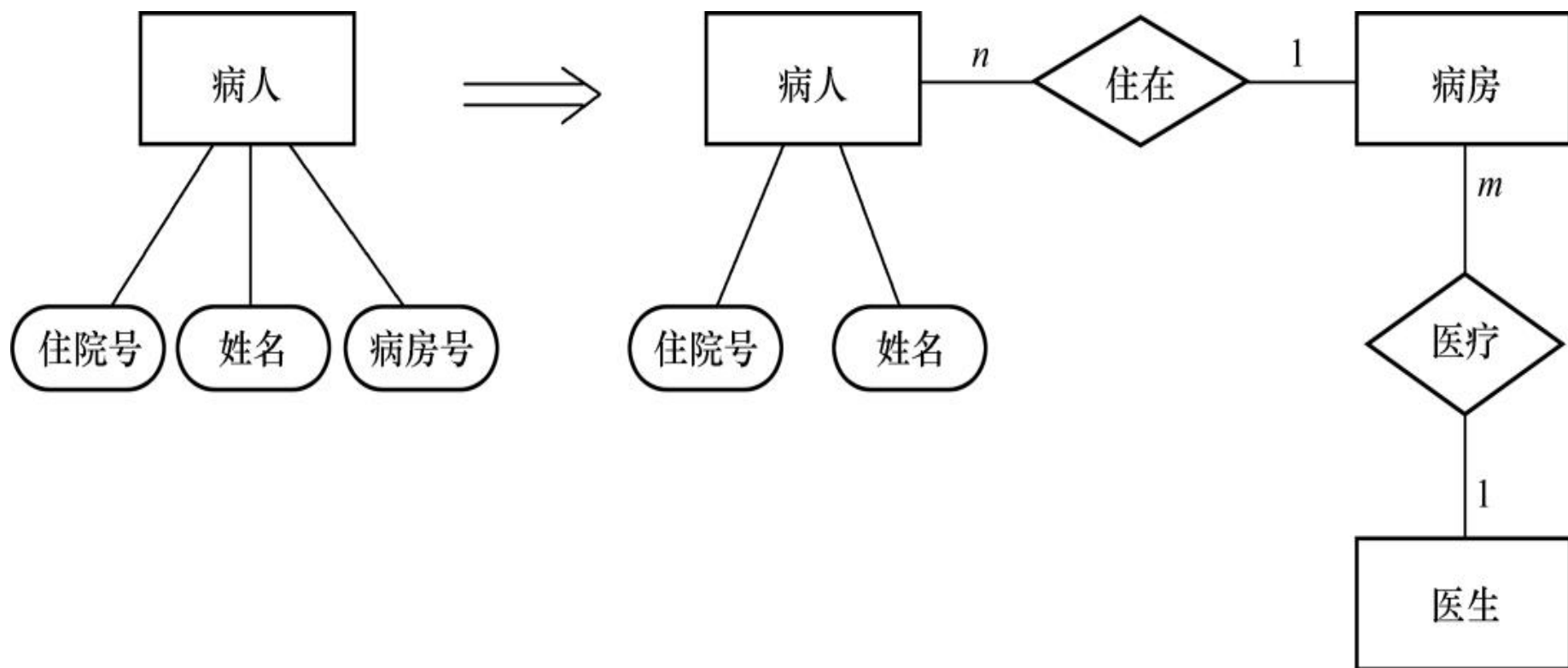
➤ 职称如果没有与工资、福利挂钩，没有进一步描述的特性，根据准则（1）可以作为职工实体的属性

➤ 如果不同的职称有不同的工资、住房标准和不同的附加福利，则职称作为一个实体更恰当



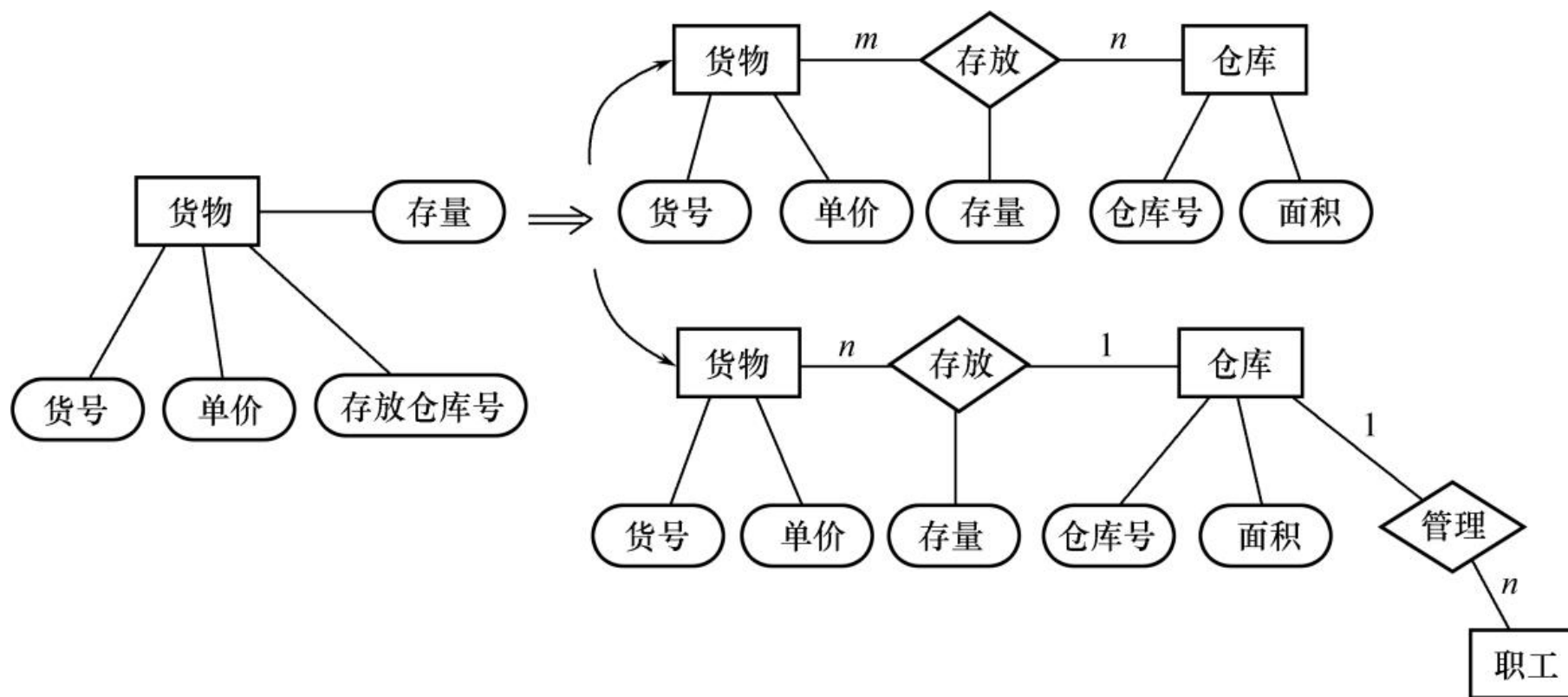
实体与属性的划分 3

❑ [例2] 在医院中，一个病人只能住在一个病房，病房号可以作为病人实体的一个属性；如果病房还要与医生实体发生联系，即一个医生负责几个病房的病人的医疗工作，则根据准则（2）病房应作为一个实体。



实体与属性的划分 4

- ❑ [例3] 如果一种货物只存放在一个仓库，那么就可以把存放货物的仓库的仓库号作为描述货物存放地点的属性。如果一种货物可以存放在多个仓库中，或者仓库本身又用面积作为属性，或者仓库与职工发生管理上的联系，那么就应把仓库作为一个实体。



□ [例7.1] 销售管理子系统E-R图的设计。

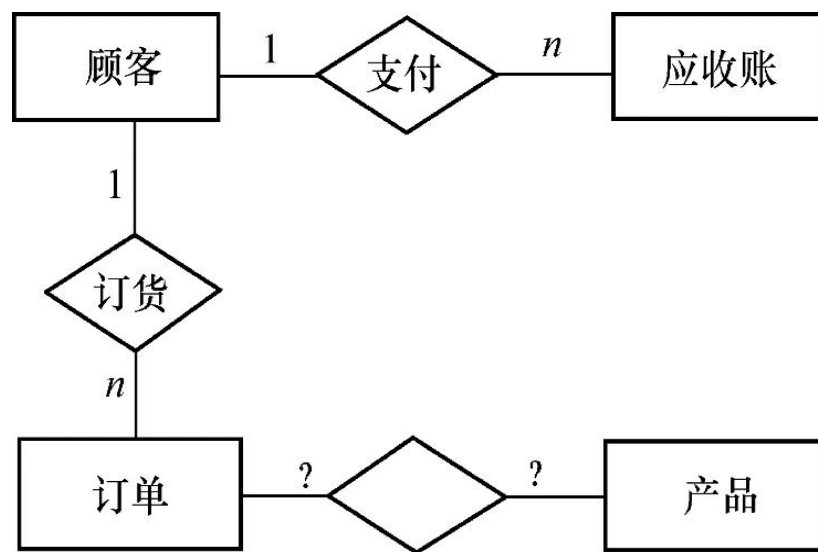
➤ 该子系统的主要功能是：

- 处理顾客和销售员送来的订单
- 工厂是根据订货安排生产的
- 交出货物同时开出发票
- 收到顾客付款后，根据发票存根和信贷情况进行应收款处理

实体与属性的划分 6

□ 参照需求分析和数据字典中的详尽描述，遵循前面给出的两个准则，进行了如下调整：

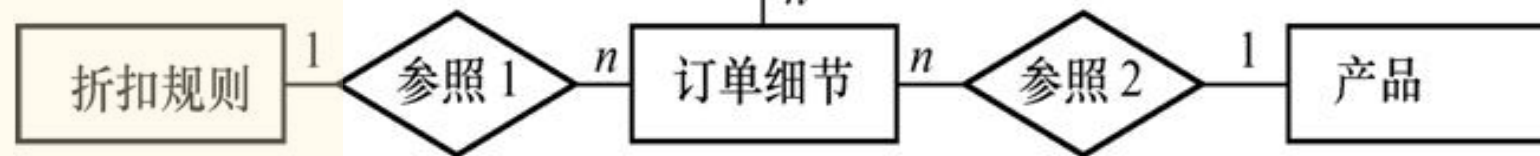
1. 每张订单由订单号、若干头信息和订单细节组成。订单细节又有订货的零件号、数量等来描述。按照准则（2），订单细节就不能作订单的属性处理而应该上升为实体。一张订单可以订若干产品，所以订单与订单细节两个实体之间是1: n的联系。



实体与属性的划分 7

2. 原订单和产品的联系实际上是订单细节和产品的联系。每条订货细节对应一个产品描述，订单处理时从中获得当前单价、产品重量等信息。

3. 工厂对大宗订货给予优惠。每种产品都规定了不同订货数量的折扣，应增加一个“折扣规则”实体存放这些信息，而不应把它们放在产品实体中。



最后得到销售管理子系统E-R图如图7.23所示。

图7.23 销售管理子系统的E-R图

实体与属性的划分 8

□ 对每个实体定义的属性如下：

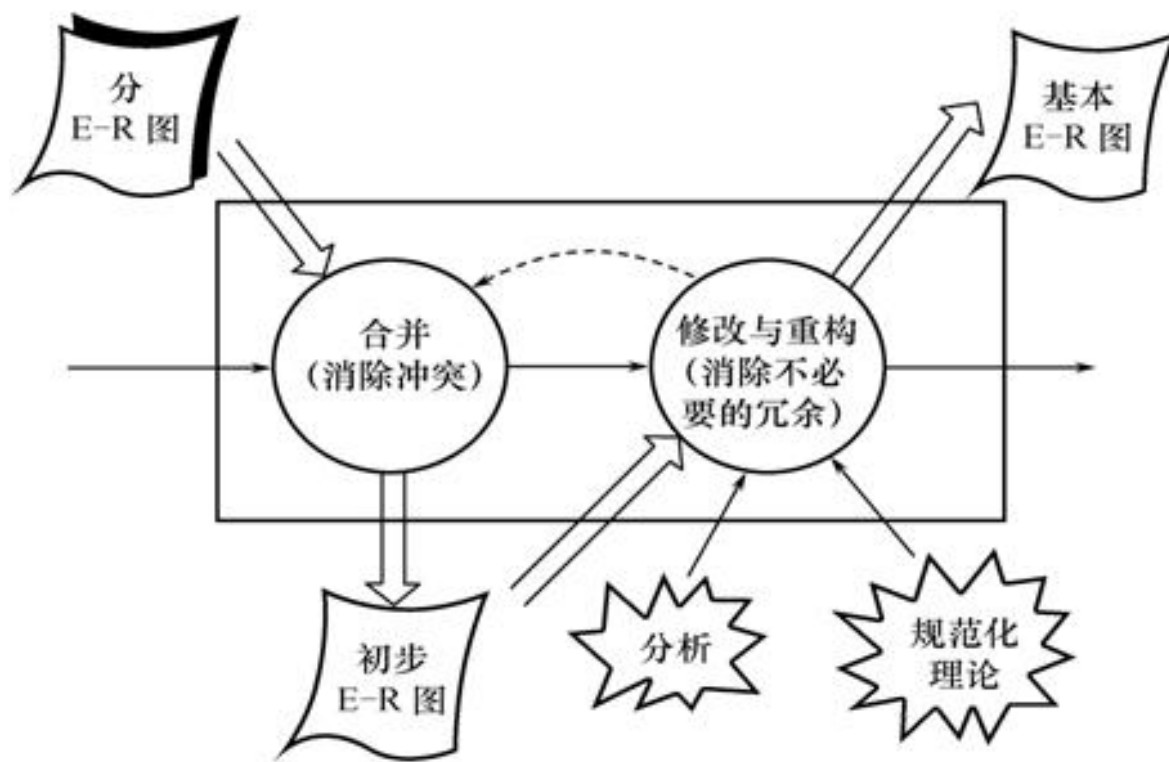
- **顾客**：{顾客号，顾客名，地址，电话，信贷状况，账目余额}
- **订单**：{订单号，顾客号，订货项数，订货日期，交货日期，工种号，生产地点}
- **订单细则**：{订单号，细则号，零件号，订货数，金额}
- **应收账款**：{顾客号，订单号，发票号，应收金额，支付日期，支付金额，当前余额，贷款限额}
- **产品**：{产品号，产品名，单价，重量}
- **折扣规则**：{产品号，订货量，折扣}

- ❑ 概念结构设计的方法
- ❑ 实体与属性的划分原则
- ❑ E-R图的集成

E-R图的集成

□ 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。



□ 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：

① 属性冲突

② 命名冲突

③ 结构冲突

□ 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

- 零件号：有的部门把它定义为整数，有的部门把它定义为字符型。
- 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

- 零件的重量：有的以公斤为单位，有的以斤为单位，有的以克为单位。

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

结构冲突 1

□ 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。但仍然要遵循实体与属性的划分原则。

□ 同一实体在不同子系统的E-R图所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：根据应用的语义对实体联系的类型进行综合或调整。

结构冲突 2

图7.25(a)中零件与产品之间存在多对多的联系“构成”

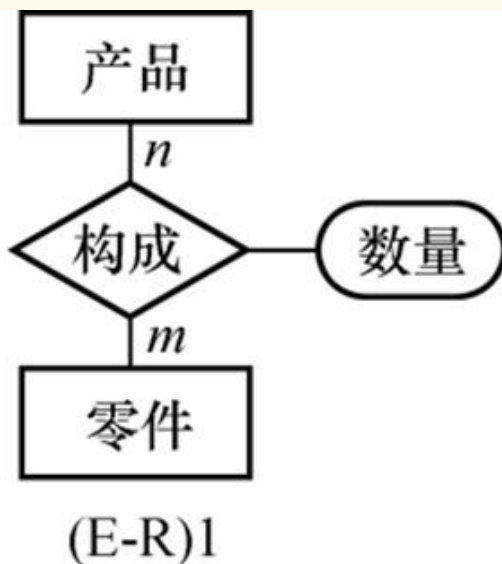
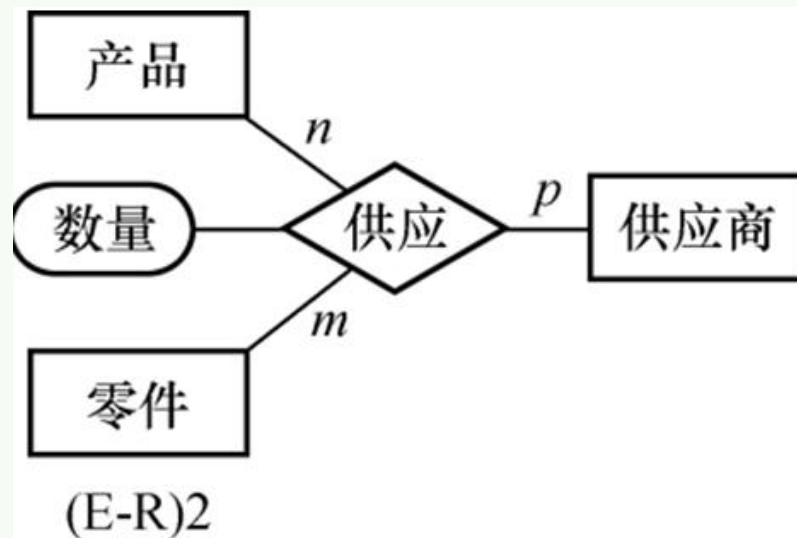
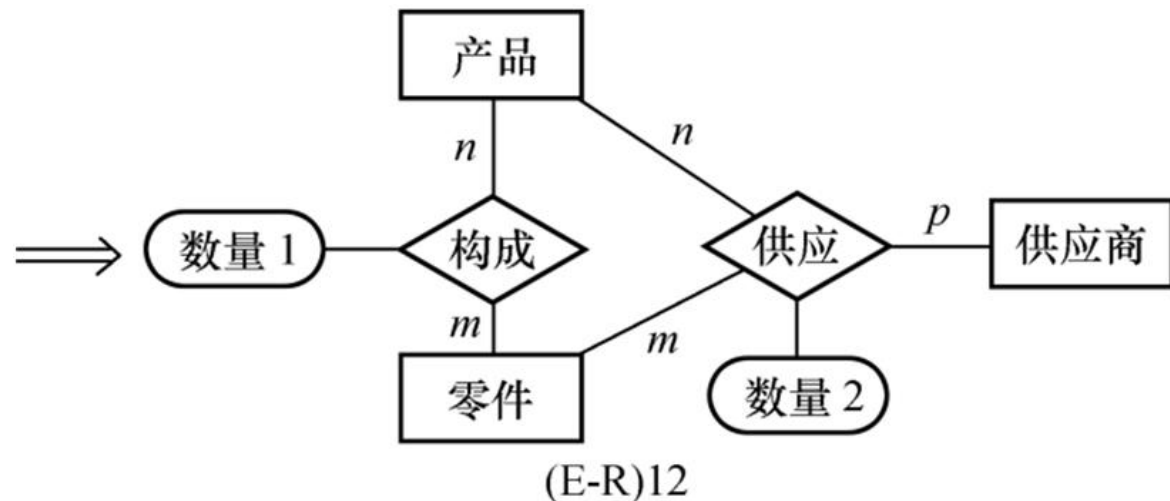


图7.25(b)中产品、零件与供应商三者之间还存在多对多的联系“供应”



合并两个E-R图，如图7.25(c)



□ 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。

E-R图的修改和重构 2

- ❑ 如图7.26中， $Q_3=Q_1 \times Q_2$ ， $Q_4=\sum Q_5$ 。所以 Q_3 和 Q_4 是冗余数据，可以消去。
- ❑ 并且，由于 Q_3 被消去，产品与材料间 $m:n$ 的冗余联系也应被消去。

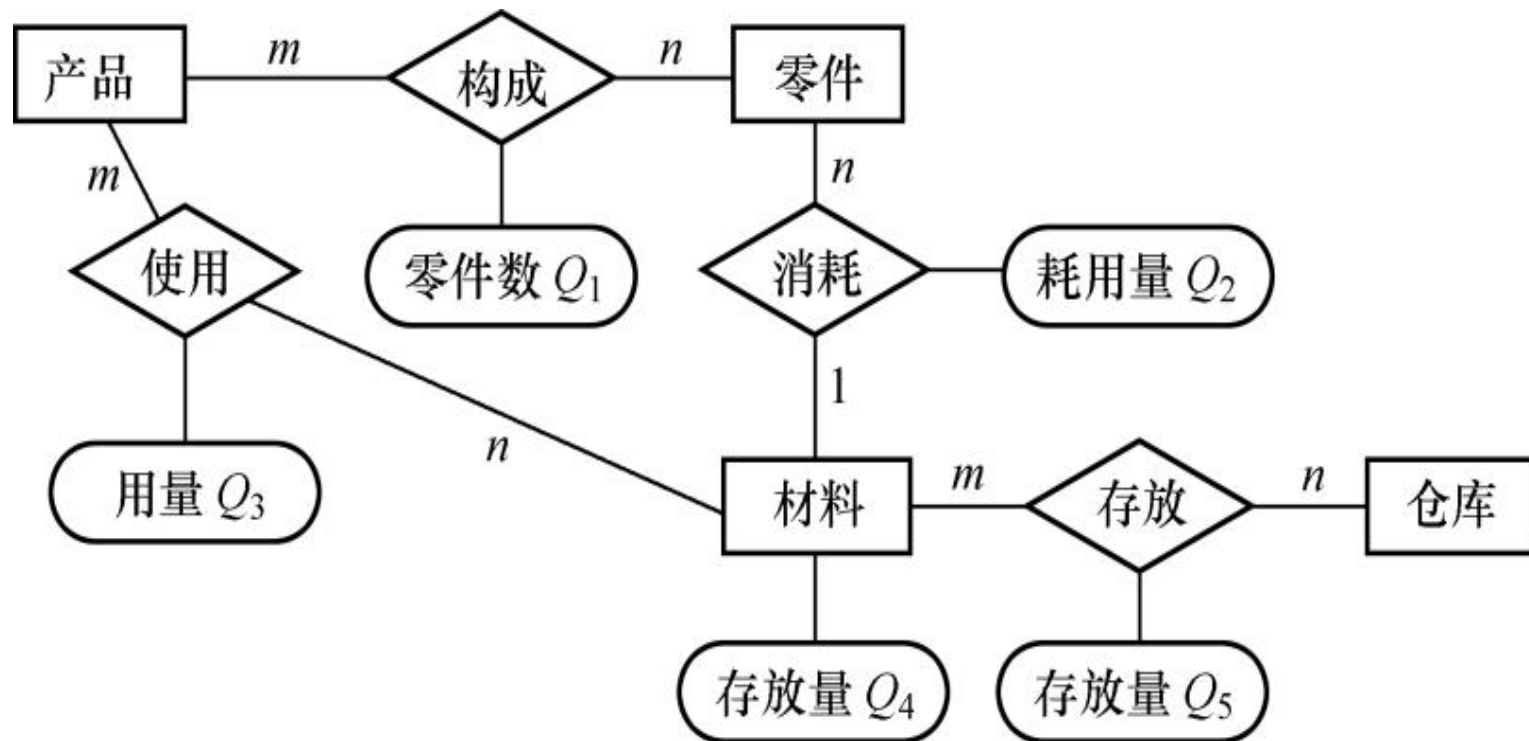


图7.26 消除冗余

- 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。

E-R图的修改和重构 3

□ 用规范化理论来消除冗余

① 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 。

- 如图7.27中：部门和职工之间一对多的联系‘属于’可表示为：
职工号 \rightarrow 部门号

- 职工和产品之间多对多的联系‘生产’可表示为：
(职工号, 产品号) \rightarrow 工作天数

② 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。

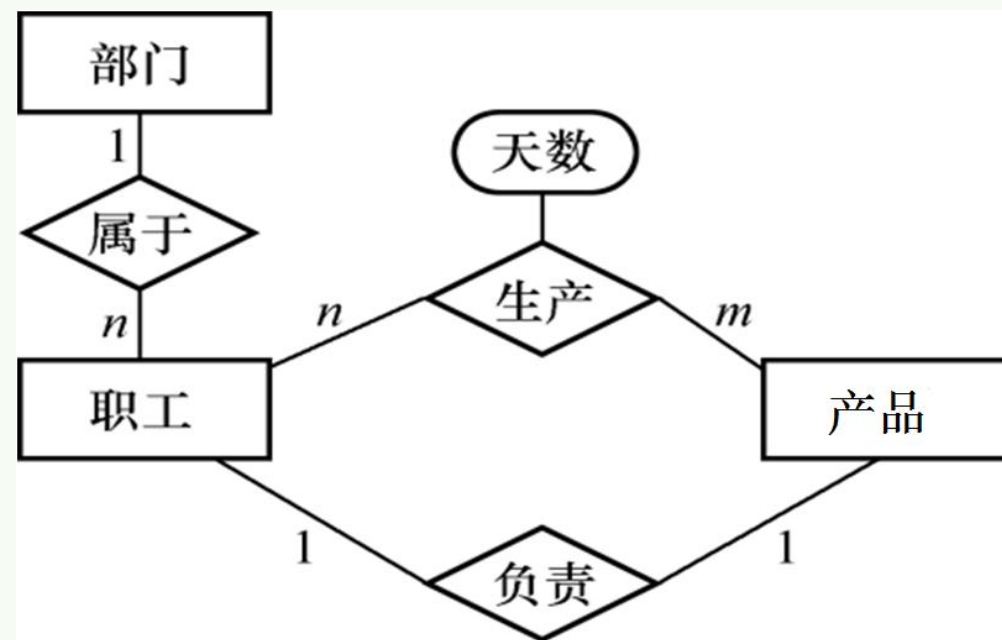


图7.27 劳动人事管理的分E-R图

- ❑ 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 $D = F_L - G_L$ 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。
 - 如部门和职工之间另一个一对一的联系‘负责’可以表示为：
负责人.职工号 \rightarrow 部门号
部门号 \rightarrow 部门号负责人.职工号

□ [例7.2] 某工厂管理信息系统的视图集成。

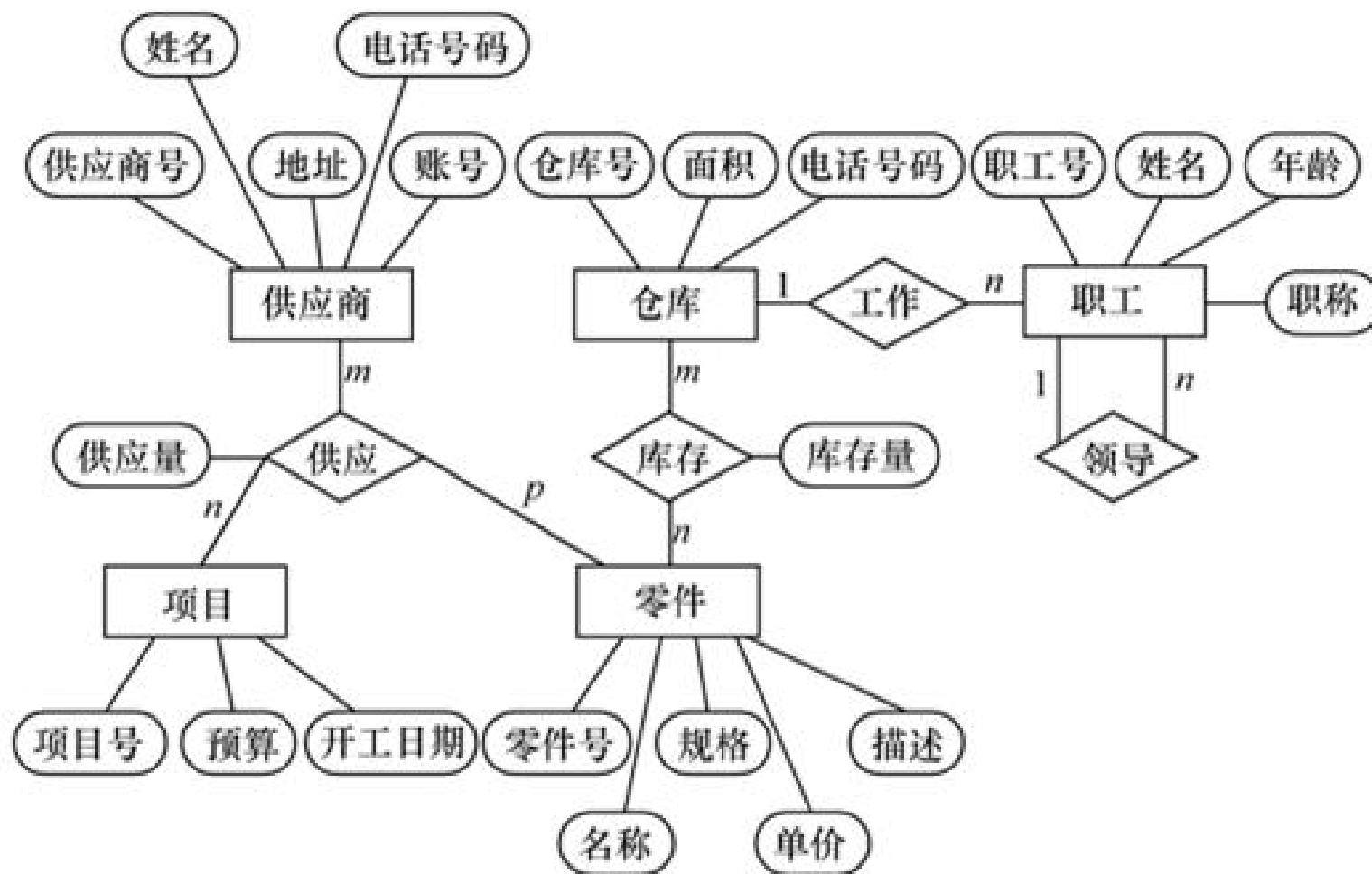


图7.11 工厂物资管理E-R图

□ [例7.2] 某工厂管理信息系统的视图集成。

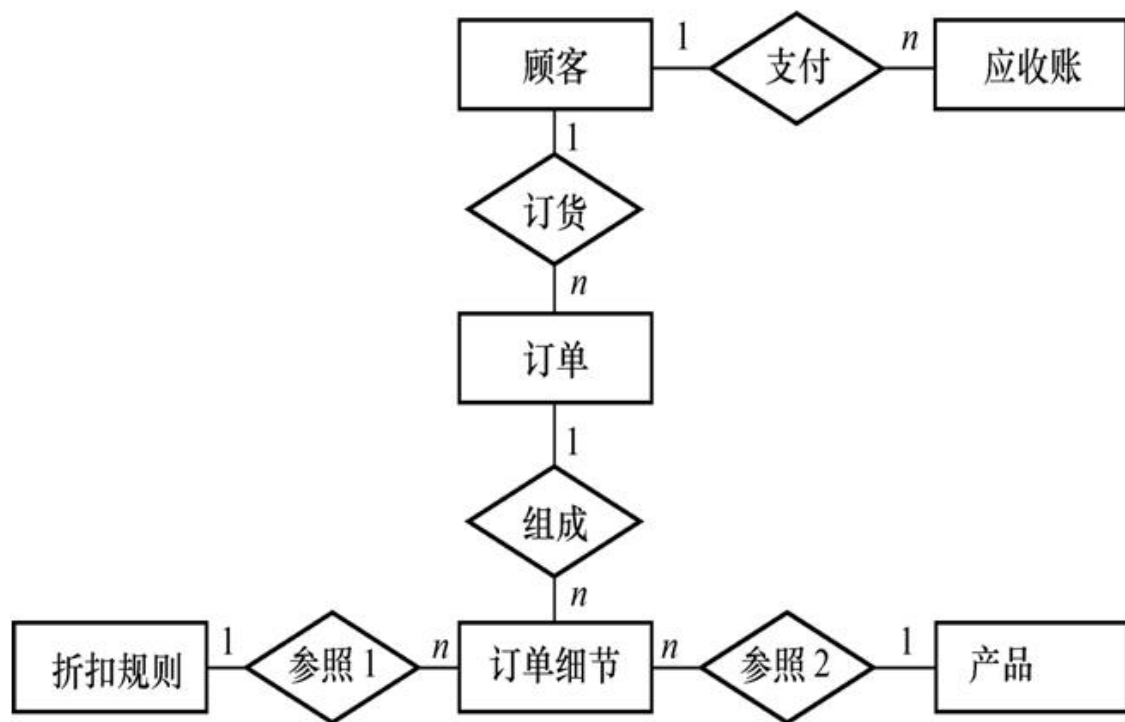


图7.23 销售管理子系统的E-R图

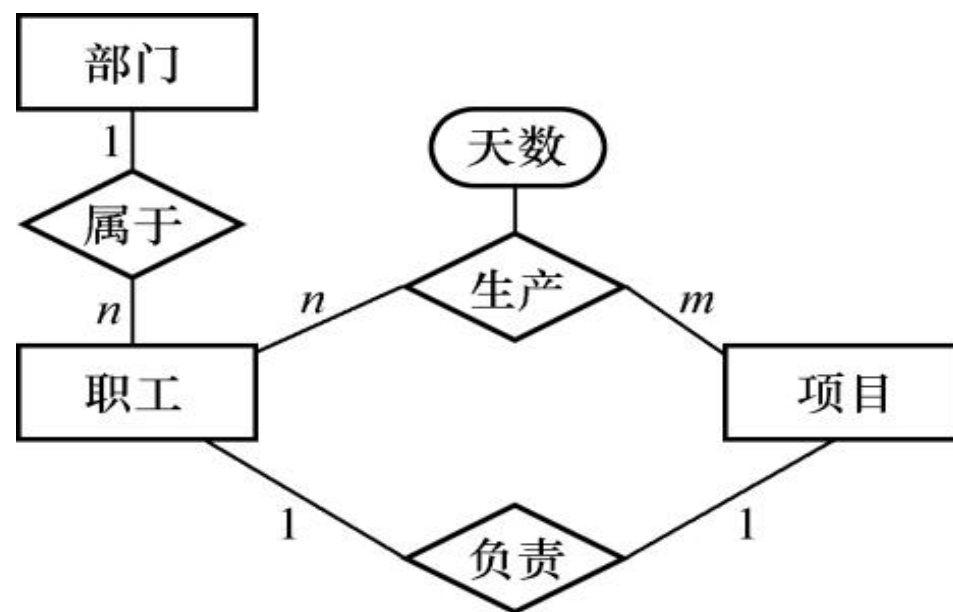
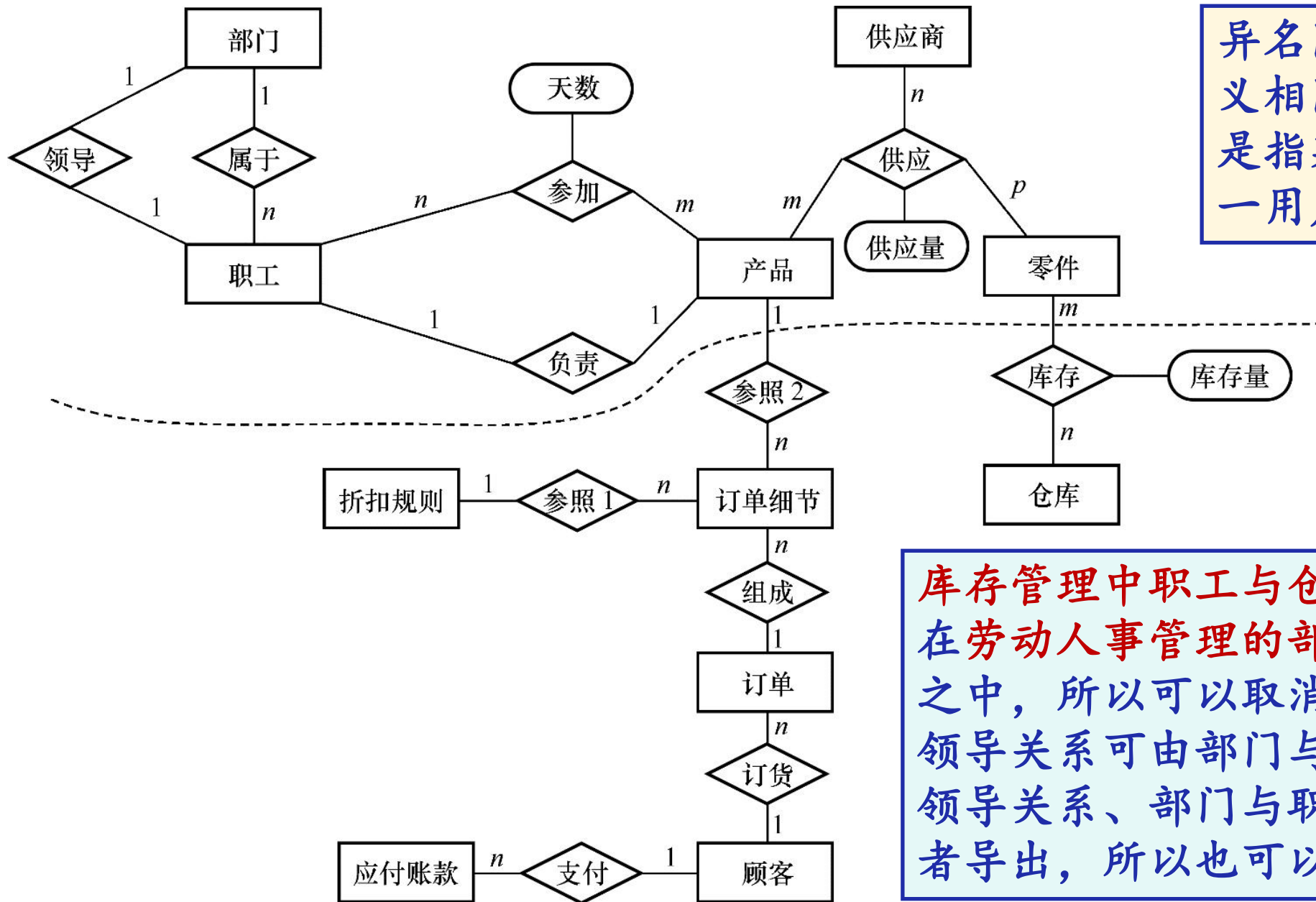


图7.27 劳动人事管理的分E-R图

图7.28 某工厂管理信息系统的基本E-R图 (E-R图集成案例)



异名同义，**项目**和**产品**含义相同。某个项目实质上是指某个产品的生产。统一用产品作实体名。

库存管理中职工与仓库的工作关系已包含在劳动人事管理的部门与职工之间的联系之中，所以可以取消。职工之间领导与被领导关系可由部门与职工（经理）之间的领导关系、部门与职工之间的从属关系两者导出，所以也可以取消。

数据管理基础

7.4 逻辑结构设计

智能软件与工程学院

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图转换为与选用数据库管理系统产品所支持的数据模型相符合的逻辑结构

❑ 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：
 - ① 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
 - ② 模型优化
 - ③ 设计用户子模式

□ 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-1:1联系

□ 一个1:1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。

□ 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

□ 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-1:n联系

❑ 一个1:n联系可以转换为一个独立的关系模式，也可以与n端对应的关系模式合并。

❑ 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：n端实体的码

❑ 与n端对应的关系模式合并

- 合并后关系的属性：在n端关系中加入1端关系的码和联系本身的属性
- 合并后关系的码：不变
- 可以减少系统中的关系个数

转换原则-m:n联系

□ 一个m:n联系转换为一个关系模式

- 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：各实体码的组合

□ [例] “选修”联系是一个m:n联系，可以将它转换为如下关系模式，其中学号与课程号为关系的组合码：

- 选修（学号，课程号，成绩）

转换原则-多元联系

□ 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：各实体码的组合

□ 具有相同码的关系模式可合并

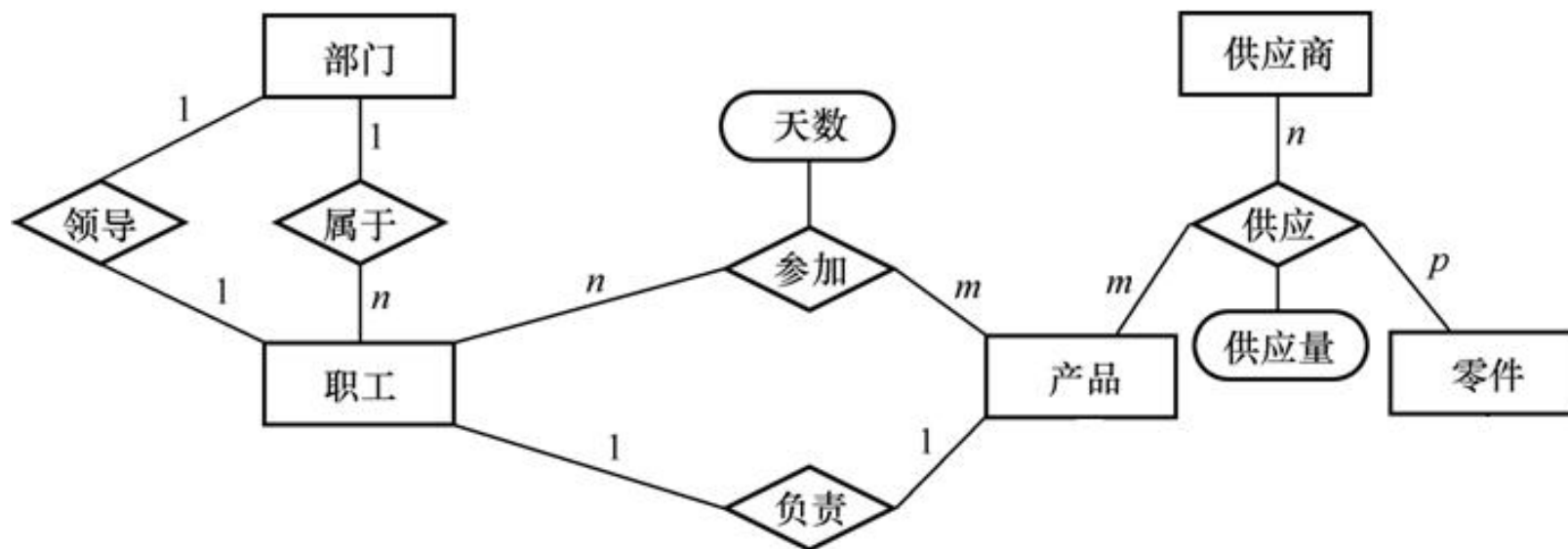
➤ 目的：减少系统中的关系个数

➤ 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

E-R图向关系模型的转换

- ❑ 部门 (部门号, 部门名, 经理的职工号, ...)
- ❑ 职工 (职工号, 部门号, 职工名, 职务, ...)
- ❑ 产品 (产品号, 产品名, 产品组长的职工号, ...)



- ❑ 供应商 (供应商号, 姓名, ...)
- ❑ 零件 (零件号, 零件名, ...)
- ❑ 参加 (职工号, 产品号, 工作天数, ...)
- ❑ 供应 (产品号, 供应商号, 零件号, 供应量)

数据模型的优化

- ❑ 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。

□ 关系规范化设计：确定数据依赖 + 规范化设计

- 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。
- 按照数据依赖的理论对各个关系模式进行分析，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，并按要求开展规范化设计。

□ 不同关系之间冗余函数依赖的检查

- 按需求分析阶段所得到的语义，分别写出不同关系模式属性之间数据依赖。
- 对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。

□ 按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，确定是否要对它们进行合并或分解。

❑ 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
- 因此在这种情况下，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，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

关系模式的分解

□ 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

水平分解

□ 把(基本)关系的元组集合划分为若干子集合，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。
- 水平分解为若干子关系，使每个事务存取的数据对应一个子关系。

垂直分解

❑ 把关系模式R的属性集划分分解为若干子集合，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关系R中分解出来形成一个子关系模式

❑ 垂直分解的优点与缺点

➤ 优点：可以提高某些事务的效率

➤ 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

➤ 取决于分解后关系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）

设计用户子模式 1

- ❑ 定义数据库模式主要是从系统的时间效率、空间效率、易维护等角度出发。
- ❑ 定义用户外模式时应该更注重考虑用户的习惯与方便。包括三个方面：
 - ① 使用更符合用户习惯的别名
 - 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
 - ③ 简化用户对系统的使用

设计用户子模式 2

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。

- 假设有关系模式产品（产品号，产品名，规格，单价，生产车间，生产负责人，产品成本，产品合格率，质量等级），可以在产品关系上建立两个视图：
 - 为一般顾客建立视图：
产品1（产品号，产品名，规格，单价）
 - 为产品销售部门建立视图：
产品2（产品号，产品名，规格，单价，车间，生产负责人）

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

数据库管理系统适应性转换

- ❑ 一般的数据模型还需要向特定数据库管理系统规定的模型进行转换。
- ❑ 转换的主要依据是所选用的数据库管理系统的功能及限制。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。

数据管理基础

7.5 物理结构设计

智能软件与工程学院

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
- ❑ 若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。

- ❑ 数据库物理设计的内容与方法
- ❑ 关系模式存取方法选择
- ❑ 确定数据库的存储结构
- ❑ 评价物理结构

数据库物理设计的内容和方法

□ 设计物理数据库结构的准备工作

- 充分了解应用环境，详细分析要运行的事务，以获得选择物理数据库设计所需参数。
- 充分了解所用关系型数据库管理系统的内部特征，特别是系统提供的存取方法和存储结构。

□ 关系数据库物理设计的内容

- 为关系模式选择存取方法
- 设计关系、索引等数据库文件的物理存储结构

□ 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

关系模式存取方法选择

- ❑ 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关系数据库管理系统支持的存取方法确定选择哪些存取方法。
- ❑ 数据库管理系统常用存取方法
 - B+树索引存取方法
 - Hash索引存取方法
 - 聚簇存取方法

B+树索引存取方法的选择

□ 选择索引存取方法的主要内容：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

□ 选择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、查找索引的开销）

□ 选择Hash存取方法的规则

- 如果一个关系的属性主要出现在等值连接条件中或主要出现在等值比较选择条件中，而且满足下列两个条件之一
 - 该关系的大小可预知，而且不变；
 - 该关系的大小动态改变，但所选用的数据库管理系统提供了动态Hash存取方法。

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）
- ❑ 聚簇的用途：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。

❑ 聚簇

- 既适用于单个关系独立聚簇
- 也适用于经常进行连接操作的多个关系
 - 把多个连接的元组按连接属性值聚集存放
 - 从而实现多个关系的“预连接”，提高连接操作的效率。

❑ 选择聚簇存储方法，即确定需要建立多少个聚簇，每个聚簇中包含哪些关系

- 一个数据库可以建立多个聚簇，一个关系只能加入一个聚簇。

□ 设计候选聚簇

- 常在一起进行连接操作的关系可以建立组合聚簇
- 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；
- 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。

□ 检查候选聚簇中的关系，取消其中不必要的关系

- 从聚簇中删除经常进行全表扫描的关系
- 从聚簇中删除更新操作远多于连接操作的关系
- 从聚簇中删除重复出现的关系
 - 当一个关系同时加入多个聚簇时，必须从这多个聚簇方案（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。

❑ 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所有索引无效，必须重建。
 - 当一个元组的聚簇码改变时，该元组存储位置也要做相应改变。

❑ 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

确定数据库的存储结构 1

- ❑ 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。

- ❑ 影响数据存放位置和存储结构的因素
 - 硬件环境
 - 应用需求
 - 存取时间
 - 存储空间利用率
 - 维护代价

这三个方面常常是相互矛盾的
必须进行权衡，选择一个折中方案

□ 基本原则

➤ 根据应用情况将

- 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]

- 可以将比较大的表分别放在两个磁盘上，以加快存取速度，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。

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

- 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
-

❑ 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，以切实改进系统性能。

评价物理结构

□ 对数据库物理设计过程中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。

□ 评价方法

➤ 定量估算各种方案

- 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。

数据管理基础

7.6 数据库的实施和维护

智能软件与工程学院

7.6 数据库的实施和维护

- ❑ 数据的载入和应用程序的调试
- ❑ 数据库试运行
- ❑ 数据库的运行和维护
 - 转储与恢复
 - 安全性和完整性控制
 - 数据库性能的监督、分析和改造
 - 数据库的重组和重构

数据的载入

❑ 数据库结构建立好后，就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。

❑ 数据装载方法

➤ 人工方法

➤ 计算机辅助数据入库

应用程序的调试

- ❑ 数据库应用程序的设计应该与数据设计并行进行
- ❑ 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、步骤在软件工程等课程中有详细讲解，这里就不赘述了

❑ 应用程序调试完成，并且已有一小部分数据入库后，就可以开始对数据库系统进行联合调试，也称数据库的试运行。

❑ 主要工作包括：

- **功能测试**：实际运行应用程序，执行对数据库的各种操作，测试应用程序的各种功能。
- **性能测试**：测量系统的性能指标，分析是否符合设计目标。

❑ **数据库性能指标的测量**

- 数据库物理设计阶段在评价数据库结构估算时间、空间指标时，作了许多简化和假设，忽略了许多次要因素，因此结果必然很粗糙。
- 数据库试运行则是要实际测量系统的各种性能指标（不仅是时间、空间指标），如果结果不符合设计目标，则需要返回物理设计阶段，调整物理结构，修改参数；有时甚至需要返回逻辑设计阶段，调整逻辑结构。

□数据库的试运行的注意事项

① 数据的分期入库

- 重新设计物理结构甚至逻辑结构，会导致数据重新入库
- 由于数据入库工作量实在太太，所以可以采用分期输入数据的方法
 - 先输入小批量数据供先期联合调试使用
 - 待试运行基本合格后再输入大批量数据
 - 逐步增加数据量，逐步完成运行评价

② 数据库的转储和恢复

- 在数据库试运行阶段，系统还不稳定，硬、软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，误操作也不可避免
- 因此必须做好数据库的转储和恢复工作，尽量减少对数据库的破坏

❑ 在数据库运行阶段，对数据库经常性的维护工作主要是由数据库管理员完成的，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、分析和改进
- 数据库的重组织与重构造

数据库的转储和恢复

- ❑ 数据库管理员要针对不同的应用要求制定不同的转储计划，定期对数据库和日志文件进行备份。
- ❑ 一旦发生介质故障，即利用数据库备份及日志文件备份，尽快将数据库恢复到某种一致性状态。

□ 初始定义

- 数据库管理员根据用户的实际需要授予不同的操作权限
- 根据应用环境定义不同的完整性约束条件

□ 修改定义

- 当应用环境发生变化，对安全性的要求也会发生变化，数据库管理员需要根据实际情况修改原有的安全性控制
- 由于应用环境发生变化，数据库的完整性约束条件也会变化，也需要数据库管理员不断修正，以满足用户要求

❑ 在数据库运行过程中，数据库管理员必须监督系统运行，对监测数据进行分析，找出改进系统性能的方法。

- 利用监测工具获取系统运行过程中一系列性能参数的值
- 通过仔细分析这些数据，判断当前系统是否处于最佳运行状态
- 如果不是，则需要通过调整某些参数来进一步改进数据库性能

数据库的重组织

□ 为什么要重组织数据库

- 数据库运行一段时间后，由于记录的不断增、删、改，会使数据库的物理存储变坏，从而降低数据库存储空间的利用率和数据的存取效率，使数据库的性能下降。

□ 重组织的形式

- 全部重组织
- 部分重组织
 - 只对频繁增、删的表进行重组织

□ 重组织的目标

- 提高系统性能

□ 重组织的工作

- 按原设计要求
 - 重新安排存储位置
 - 回收垃圾
 - 减少指针链
- 数据库的重组织不会改变原设计的数据逻辑结构和物理结构

- 数据库管理系统一般都提供了供重组织数据库使用的实用程序，帮助数据库管理员重新组织数据库。

□ 为什么要进行数据库的重构造

- 数据库应用环境发生变化，会导致实体及实体间的联系也发生相应的变化，使原有的数据库设计不能很好地满足新的需求
 - 增加新的应用或新的实体
 - 取消某些已有应用
 - 改变某些已有应用

□ 重构造的主要工作

- 根据新环境调整数据库的模式和内模式
 - 增加或删除某些数据项
 - 改变数据项的类型
 - 增加或删除某个表
 - 改变数据库的容量
 - 增加或删除某些索引

□ 重构造数据库的程度是有限的

- 若应用变化太大，已无法通过重构造数据库来满足新的需求，或重构造数据库的代价太大，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。

复习思考题 1

(在设计例子时，请不要直接使用课件中的案例)

1. 试述数据库设计过程。
2. 试述数据库设计过程中，各个设计步骤的设计结果。
3. 请理解下述各组概念的定义及其相互关系
 - ① entity 与 entity instance
 - ② attribute 与 domain
 - ③ identifier 与 descriptor
 - ④ single-valued attribute/composite attribute/multi-valued attribute
 - ⑤ relationship 与 relationship instance
 - ⑥ relationship 与 IS-A联系
 - ⑦ '基数约束' 与 '函数关系'
 - ⑧ single-valued participation 与 multi-valued participation
 - ⑨ mandatory participation 与 optional participation

复习思考题 2

4. 正确理解联系上的‘函数关系’概念，并按以下要求分别举例说明（课件中的例子除外）：分别设计一个一对一、一对多、多对多的二元联系，并给出联系上的语义约束。
5. 请设计一个多元联系的例子，参与联系的部分实体来自于同一个实体集。
6. 在ER模型的设计中，①是否可以只使用‘二元联系’这一种联系类型？②如果想采用若干个二元联系来代替一个多元联系的设计方案，有哪些需要注意的问题，并请举例说明。
7. 请设计一个弱实体的例子。
8. 请设计一个含有多级继承（IS-A联系）的例子。
9. 请设计一个模型，里面含有四种不同类型的实体与联系间的基数约束：强制参与/非强制参与，单值参与/多值参与。
10. 关系规范化理论对数据库设计有什么意义？
11. 试述数据库物理设计的内容和步骤。
12. 什么是数据库的重组织和重构造？为什么要进行数据库的重组织和重构造？

复习思考题 3

13. 设有一个图书借阅管理数据库，已知：图书的属性有书号、书名；读者的属性有借书证号、姓名、身份证号、居住地址、联系电话；出版社的属性有出版社名称、办公地址、联系电话。其中：

- ① 每一本图书都有且只有一个书号和书名，书号是图书的标识属性，允许不同的图书具有相同的书名；
- ② 每个读者都有且只有一个借书证号、身份证号、姓名，最多登记一个居住地址，必须留一个或多个联系电话，借书证号和身份证号都可以单独作为读者的标识属性；
- ③ 每一个出版社都有且只有一个名称、办公地址、联系电话，出版社名称是出版社的标识属性；
- ④ 每本图书只能由一个出版社出版发行；
- ⑤ 每个读者可以同时借阅多本图书，也可以在不同时候借阅同一本图书；系统需要记录每一本图书每一次被借阅的借阅日期和归还日期（借阅日期和归还日期的数据类型是时间戳）。

(1) 请用EE-R模型来表示该数据库系统的概念数据模型；

(2) 请将上述概述数据模型转换成关系数据模型；

(3) 请写出每个关系上的最小函数依赖集、所有关键字。

复习思考题 4

14. 设有一个期末考试监考安排系统，其中需要存储的信息如下：每一门课程的课程号、课程名；每一位教师的工作证编号、姓名；每一场考试的考试时间（时间戳类型）、考试教室、监考老师。如果规定：
- ① 课程号、工作证编号、考试教室分别是课程、教师、教室的标识属性(identifier)；
 - ② 每一门课程至少安排一位或多位主讲教师；一位老师可以不担任主讲教师任务，也可以担任多门课程的主讲教师；
 - ③ 每一门课的期末考试只安排一场，考试时间包括开始时间和结束时间，可分在多个教室中同时进行；但在同一时间内，一间教室中只能安排一门课程的考试；
 - ④ 在一场考试中，课程主讲教师作为主考教师必须参加自己主讲课程的监考；同一位老师主讲的不同课程，不能安排在同一段时间内考试；
 - ⑤ 除了主考老师外，在每一间考试教室中还需要安排一位或多位‘监考老师’；一位老师可以在不同的时间担任不同课程的监考任务，但在同一时间内，一位‘监考老师’只能在指定的一间教室中履行监考任务。
- (1) 请用EE-R模型来表示该数据库系统的概念数据模型；
- (2) 请将上述概述数据模型转换成关系数据模型；
- (3) 请写出每个关系上的最小函数依赖集、所有关键字。
- (4) 上述关系是否满足3NF？如不满足，请用到3NF的模式分解算法直接对其进行模式分解。