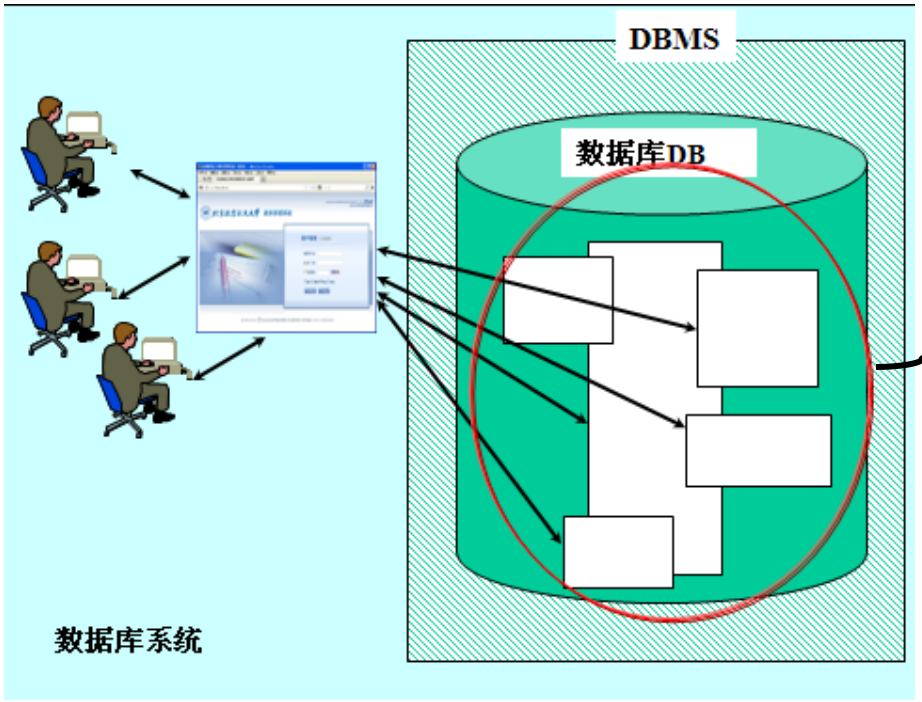
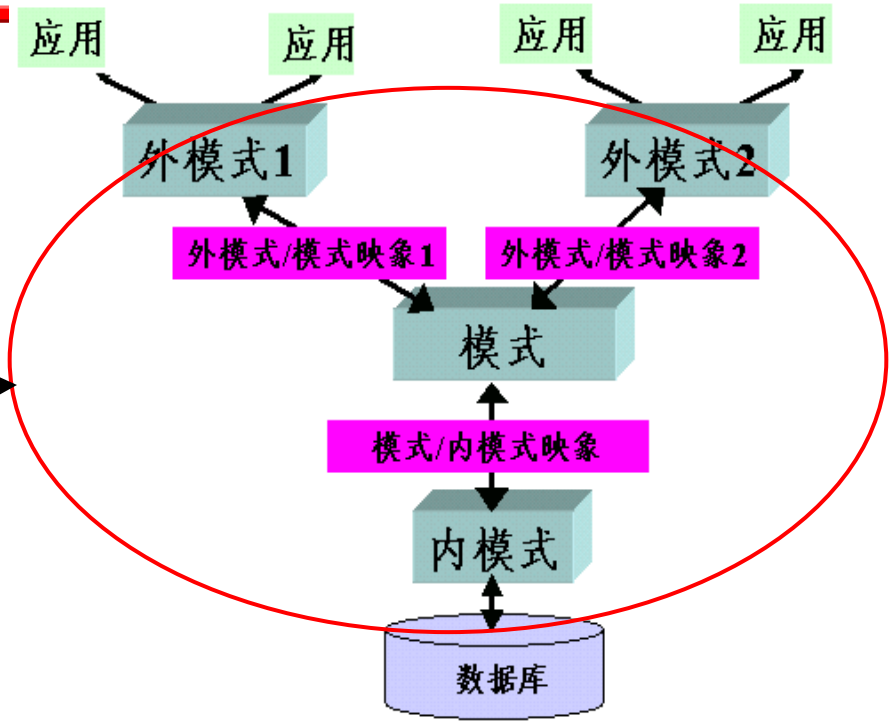
第四章 数据库设计

- 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

数据库设计概述





数据库系统体系结构

- 什么是数据库设计
  - 是指对于一个给定的应用环境，设计优化的数据库逻辑和物理结构，建立数据库，使之能够有效地存储数据，为开发满足用户需求的应用系统奠定基础。
- 数据库设计的特点
  - 要把数据设计和处理设计密切结合起来；
  - 与硬件、软件和管理紧密相关。

数据库设计概述

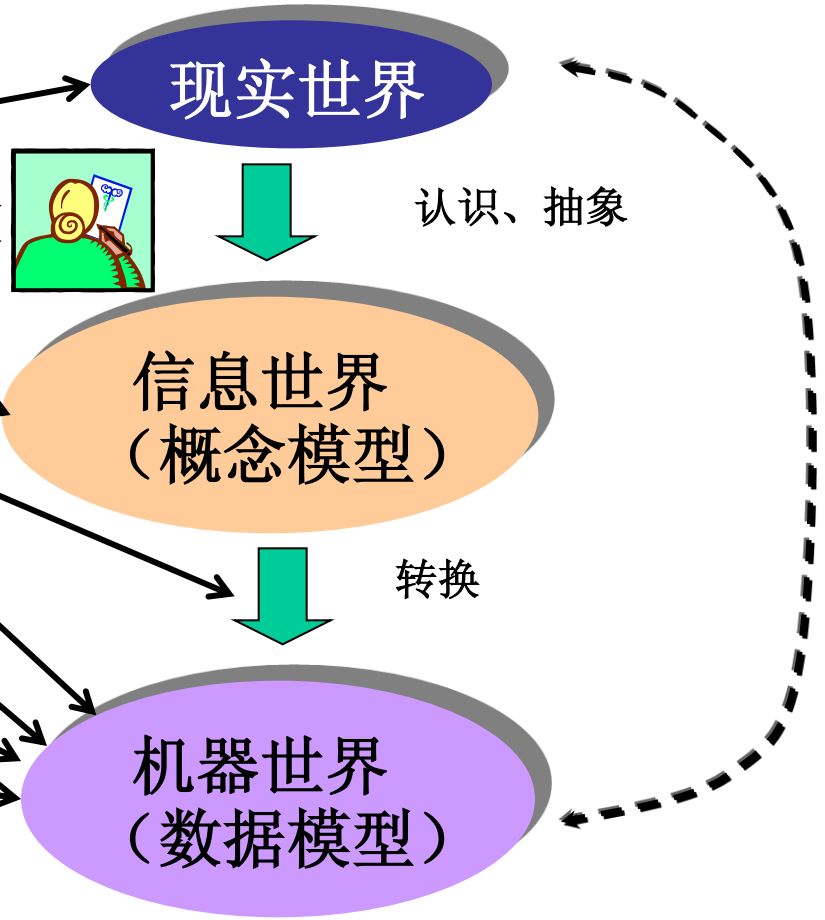
- 数据库设计的方法（1）
  - 手工试凑法（直接设计法）：根据应用的数据要求与处理要求，直接设计数据库的结构。缺点：
    - 数据间关系复杂，使用要求各式各样，很难仅凭经验进行设计；
    - 把数据的逻辑结构，物理结构、处理要求等一起考虑，很难对模式进行评价和优化。用户需求一旦发生变化，数据结构很难随之发生变化；
    - 数据库设计与具体的DBMS紧密结合，移植困难；
    - 缺乏文档资料，难于与用户交流，对设计难于评审；
    - 难以由多个人合作进行设计。

数据库设计概述

- 数据库设计的方法（2）
  - 规范设计法：运用软件工程的思想和方法，把整个设计过程划分为若干阶段，把复杂的大问题分为若干相对简单的小问题，每个阶段只解决整个设计中的部分问题。
  - 规范设计法是迭代和逐步求精的过程，每一过程完成时要进行设计分析，产生各种设计文档，并组织评审和用户交流，如不满足要求则进行修改。

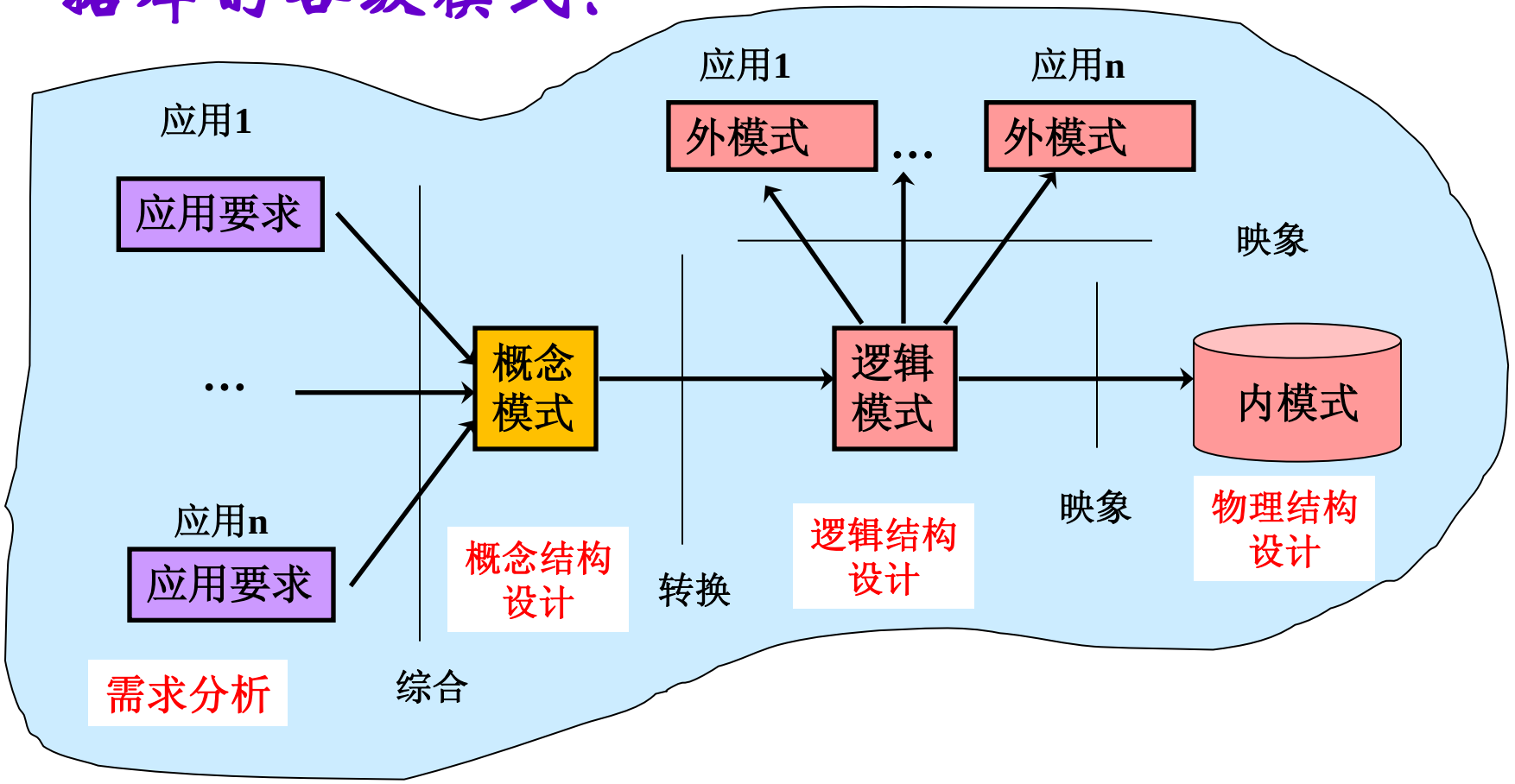
数据库设计的基本步骤

- 按照规范设计方法，数据库设计分为以下六个阶段：
  - 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。



数据库设计的基本步骤

- 在六阶段数据库设计的不同阶段形成了数据库的各级模式：



数据库设计阶段概述（1）

- 需求分析
  - 对应用环境进行详细调查，收集支持系统目标的基础数据及其处理；
- 数据库概念结构设计
  - 通过对用户需求进行综合、归纳与抽象，形成独立于数据库逻辑结构与具体DBMS的概念模型，可以用E-R图等表示；
- 数据库逻辑结构设计
  - 将概念结构转换为某个DBMS所支持的数据模型，并进行优化。再将得到的逻辑结构转换成特定的DBMS能处理的模式、子模式；

数据库设计阶段概述（2）

- 数据库物理结构设计
  - 设计数据库在物理设备上的存储结构和存取方法。一般分为两步：一是确定数据库的内模式；二是对物理结构进行时间与空间效率的评价;
- 数据库实施
  - 是建立数据库的过程。用DBMS的DDL描述三级模式，并调试产生目标模式。开发应用程序，组织数据入库并试运行;
- 数据库运行和维护
  - 在数据库正式运行后，由DBA执行对数据库经常性的维护工作，包括数据库转储与恢复、数据库控制、数据库性能监控、数据库的重组与重构。

需求分析的目标

- 需求分析阶段的目标是收集支持系统应用目标的基础数据及其处理。调查的重点是“数据”和“处理”，包括：
  - 处理要求：指用户要完成什么处理功能，对处理的响应时间和处理方式的要求。
  - 信息要求：指系统中所涉及的数据及数据之间的联系。具体收集数据的名称、类型、长度等，确定数据之间联系的类型。
  - 安全性与完整性的要求

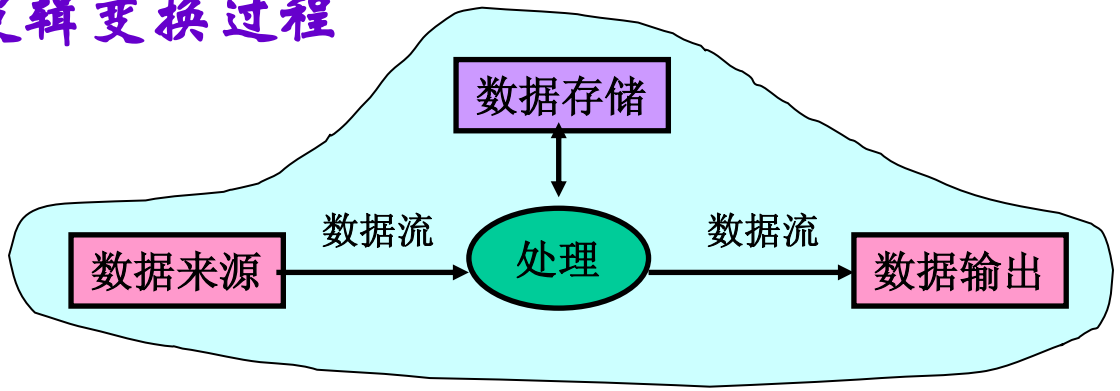
需求分析的方法

- 需求分析分为两个阶段：
  - 调查用户实际要求，与用户达成共识
  - 分析、表达用户的需求
    - 用**数据流图**表达数据和处理之间的关系；
    - 用**数据字典**描述系统中各类数据。

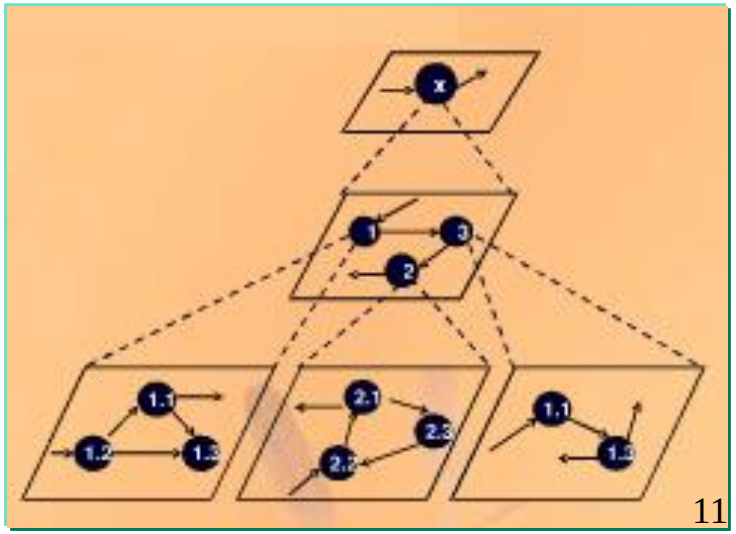
需求的表达（1）

- 数据流图（Data Flow Diagram, DFD图）

  以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程



- 采用结构化分析方法SA从最上层的系统组织机构入手，自顶向下逐步分解处理功能以及它们所用的数据，形成若干层次的数据流图。



需求的表达（2）

- 数据字典
  - 是对数据的集中的系列说明。包含每一个数据元素的名字、含义等各方面的描述。
    - 从数据流图中提取出所有原子数据项；
    - 把有联系的数据项组合起来形成数据组；
    - 以数据组为单位，写出数据项的如下定义：
      - 语义定义：名字，实际含义等
      - 类型定义：数据类型、数据宽度、小数位数等
      - 完整性约束定义：值约束、空值约束以及其他比较复杂的完整性约束。
    - 根据用户和实际领域的信息模型需要补充其他数据项及其定义。

需求的表达（3）

数据组名：				
特　征	数据项名字	数据项名字	数据项名字	数据项名字
	编号：	编号：	编号：	编号：
数据类型				
数据宽度				
小数位数				
单　位				
值 约 束				
允许空值否				
值 个 数				

需求的表达（4）

- 数据字典还可以包括数据在系统内的传输路径、数据存储位置和处理过程信息等
- 需求分析阶段的数据字典，可看成是数据元素表

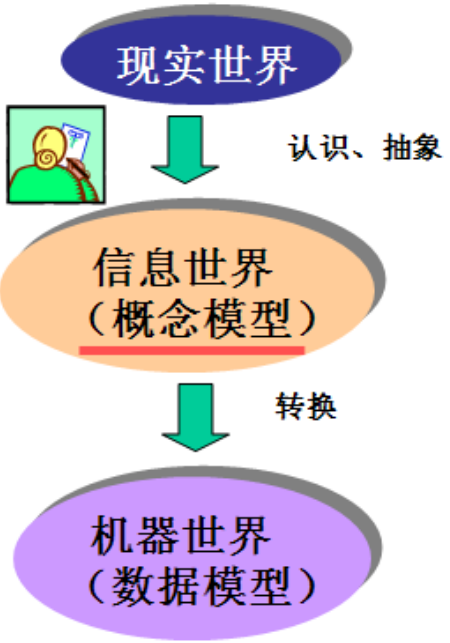
  在数据库实施阶段建立起的数据字典，是数据库系统重要组成部分

概念结构设计

- 概念结构设计的工具—— E-R法
- 应用系统的E-R图设计方法

E-R法概述

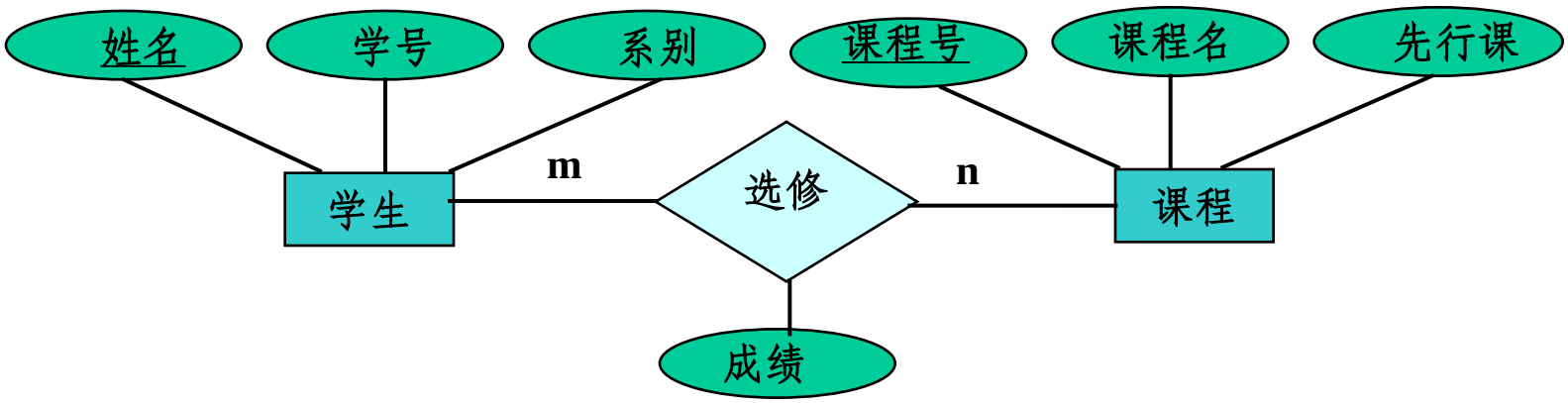
- E-R法的思想是用E-R图描述现实世界的信息，这种信息结构称为概念结构（概念模型），然后根据具体系统的要求将概念结构转换成特定系统所能接受的逻辑结构（层次、网状、关系）
- 概念模型用于信息世界建模，是现实世界到信息世界的抽象，是用户和数据库设计人员进行交流的语言
- E-R法由两部分组成：
  - 用E-R图描述现实世界；
  - 将E-R图转换成相应的数据模型。



E-R图（1）

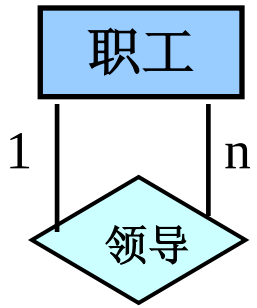
- E-R图的组成：实体、联系、属性
  - 实体：用长方形表示实体型，在框内写上实体名。
  - 属性：用椭圆形表示实体的属性，并用无向边把实体与其属性连接起来。
  - 联系：用菱形表示实体间的联系，菱形框内写上联系名。用无向边把菱形分别与有关实体相连，在无向边旁标上联系的类型。若联系也具有属性，则属性和菱形也用无向边连接上。

例：

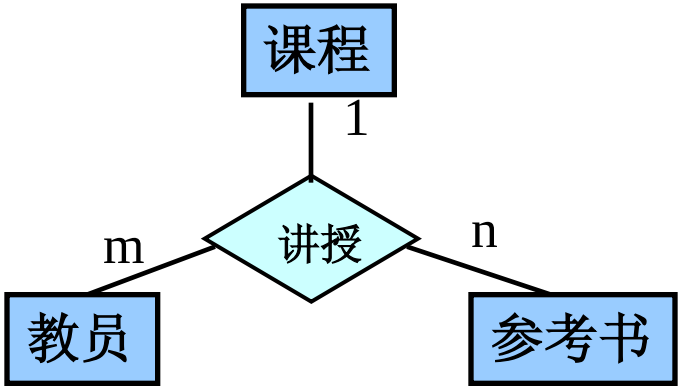


E-R图（2）

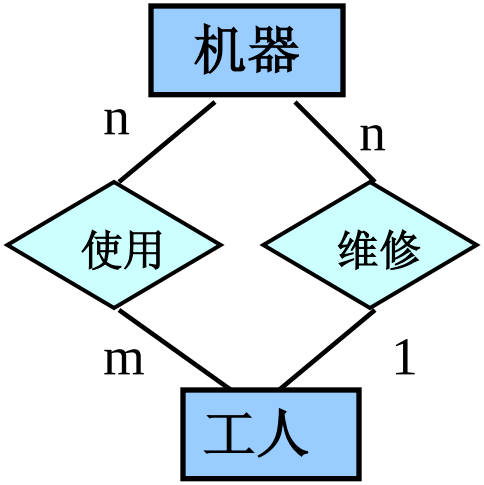
- 同一个实体集内部各实体之间也可以存在一对一、一对多、多对多的联系。（a)
- 三个或多个实体型间可能具有联系。（b）
- 两个实体型之间可具有多种联系。（c）



（a)



（b）

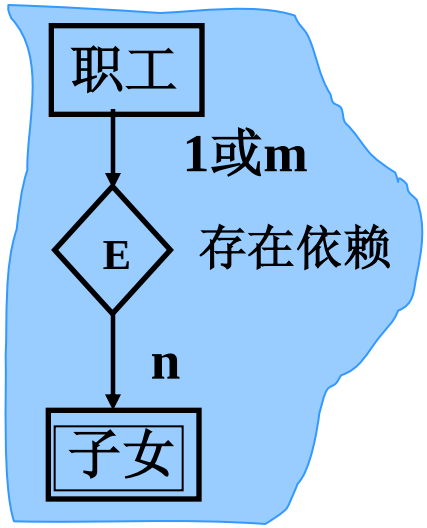


（c）

E-R图（3）

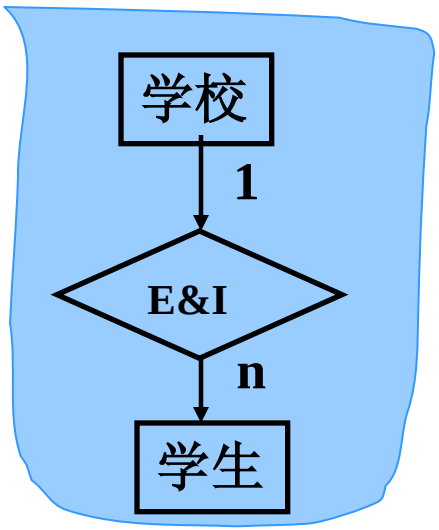
- E-R图实体之间联系的语义扩充

（1）存在依赖



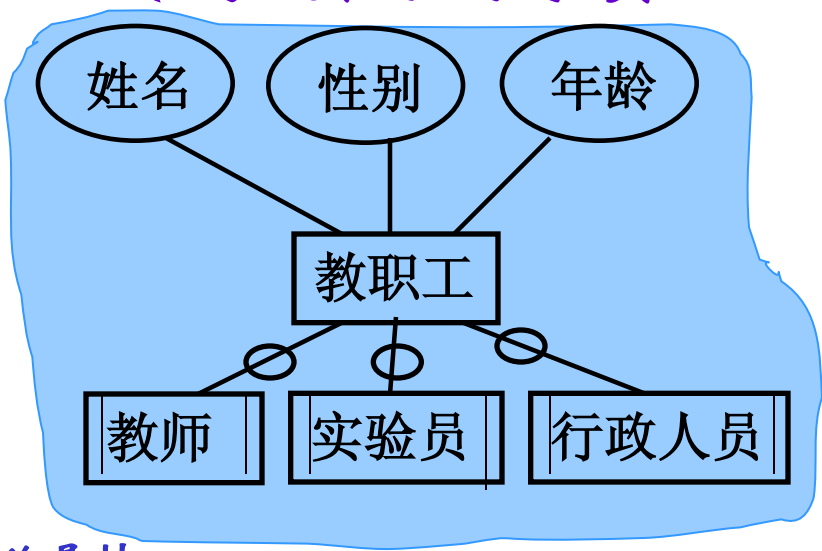
子女实体存在依赖于职工实体的存在，则称子女实体是弱实体。

（2）标识依赖



如果实体不能由它自己的属性来唯一标识，而必须通过与它相联系的另一实体一起来标识，那么称该实体标识依赖于另一个实体。

（3）实体的子类



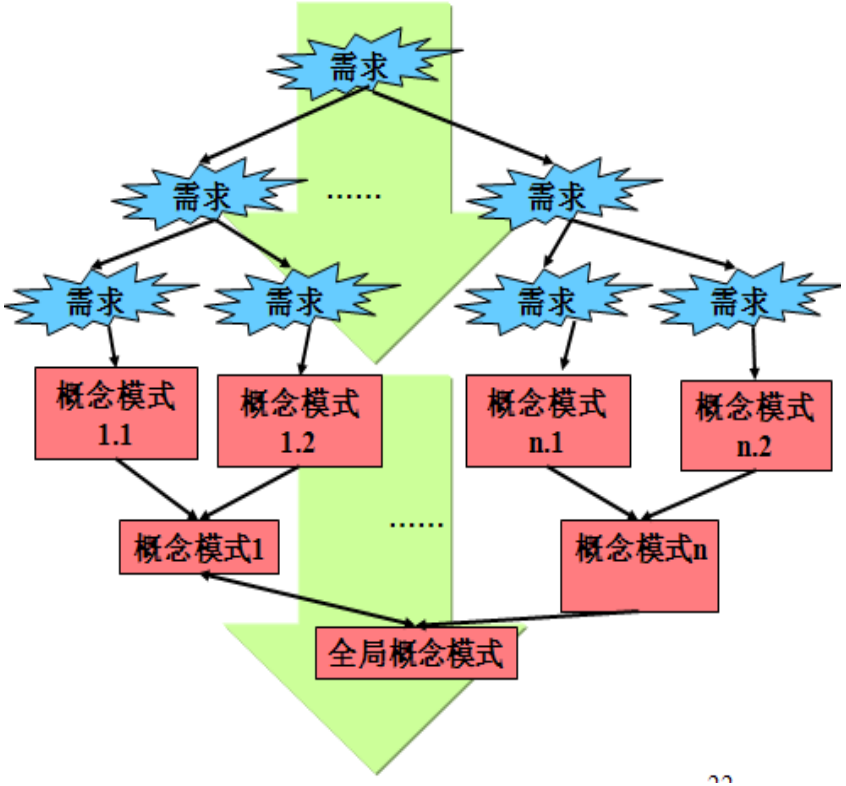
- 子类可继承父类的属性，子类也可以附加某些属性；
- 子类之间的交不一定为空。

应用系统的概念结构设计方法

- 设计概念结构通常有四类方法：
  - 自顶向下：首先定义全局概念结构的框架，然后逐步细化。
  - 自底向上：首先定义各局部应用的概念结构，然后将其集成起来。
  - 逐步扩张：首先定义核心概念结构，然后向外扩充。
  - 混合策略：自顶向下设计全局概念框架，自底向上设计各局部概念结构。
- 最常用的方法是自底向上方法

自底向上的E-R图设计方法

- 自底向上方法
  - 自顶向下进行需求分析；
  - 自底向上设计概念结构。
- 分为两个阶段
  - 局部E-R图设计
  - 综合局部E-R图形成总E-R图

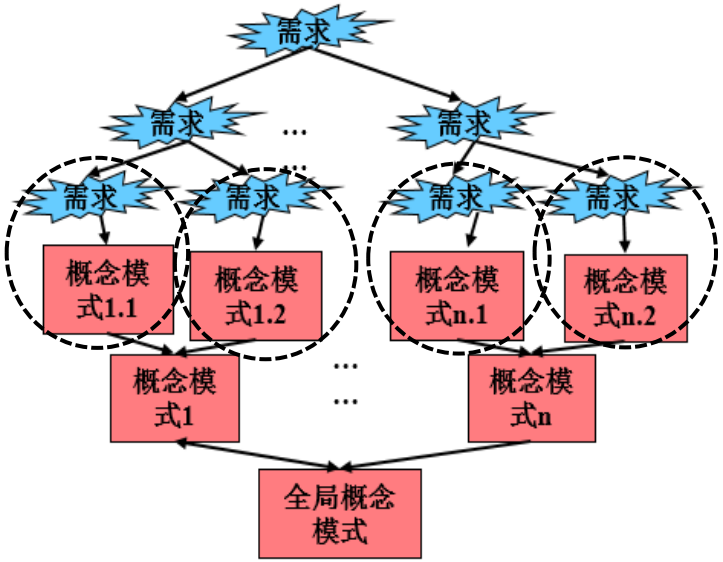


数据抽象

- 数据抽象机制用来对需求分析阶段收集的数据进行分类、组织
- 数据抽象机制包括：
  - 分类
    - 定义某一概念作为现实世界中一组对象的类型。这些对象具有某些共同的特性和行为。如实体型。
  - 聚集
    - 定义某一类型的组成成分。如属性的聚集组成了实体型。
  - 概括
    - 定义类型之间的一种子集联系。

局部E-R图设计步骤

- 选择局部应用；
- 以需求分析中得到的数据元素表为基础，利用数据抽象机制，建立实体模型；
- 确定实体之间的联系类型。用E-R图表示这些实体与实体之间的联系，形成分E-R图。

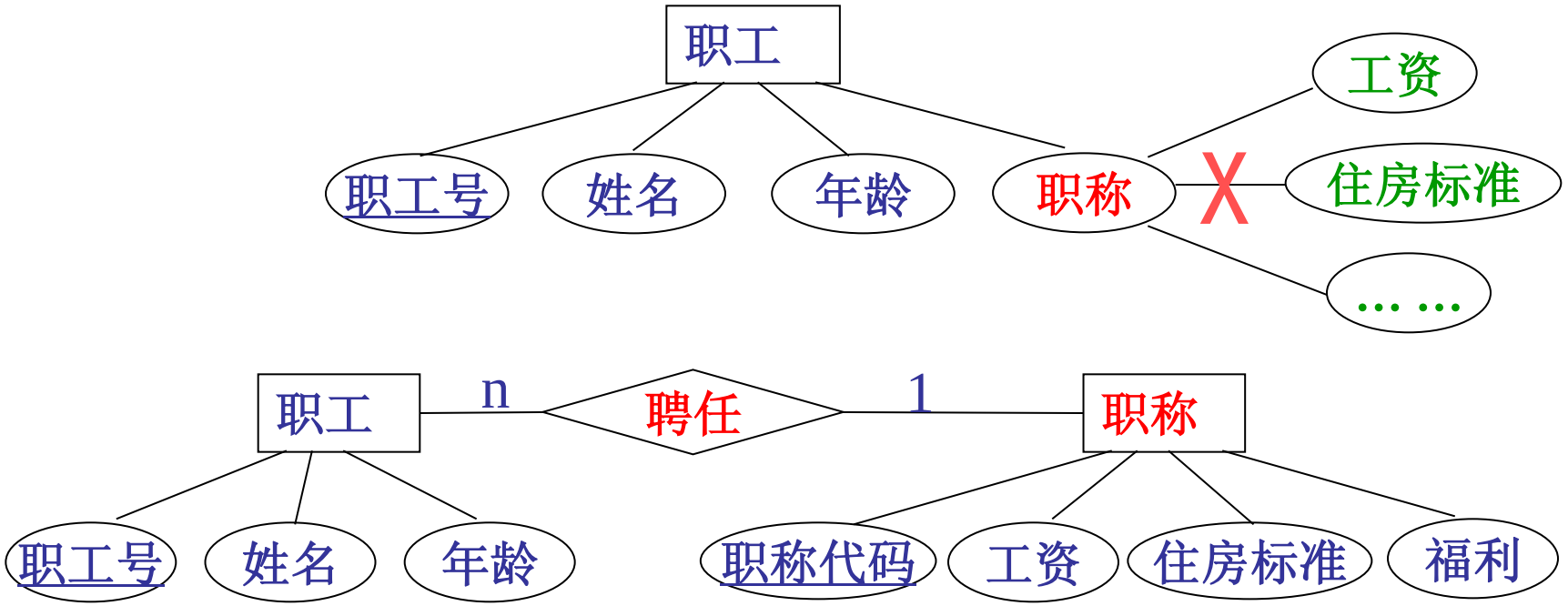


局部E-R图中的实体模型设计

- 建立实体模型的关键是确定实体及其属性
- 实体模型的建立方法：
  - 对数据字典进行初步抽象，得到实体和属性，然后再按原则进行必要调整

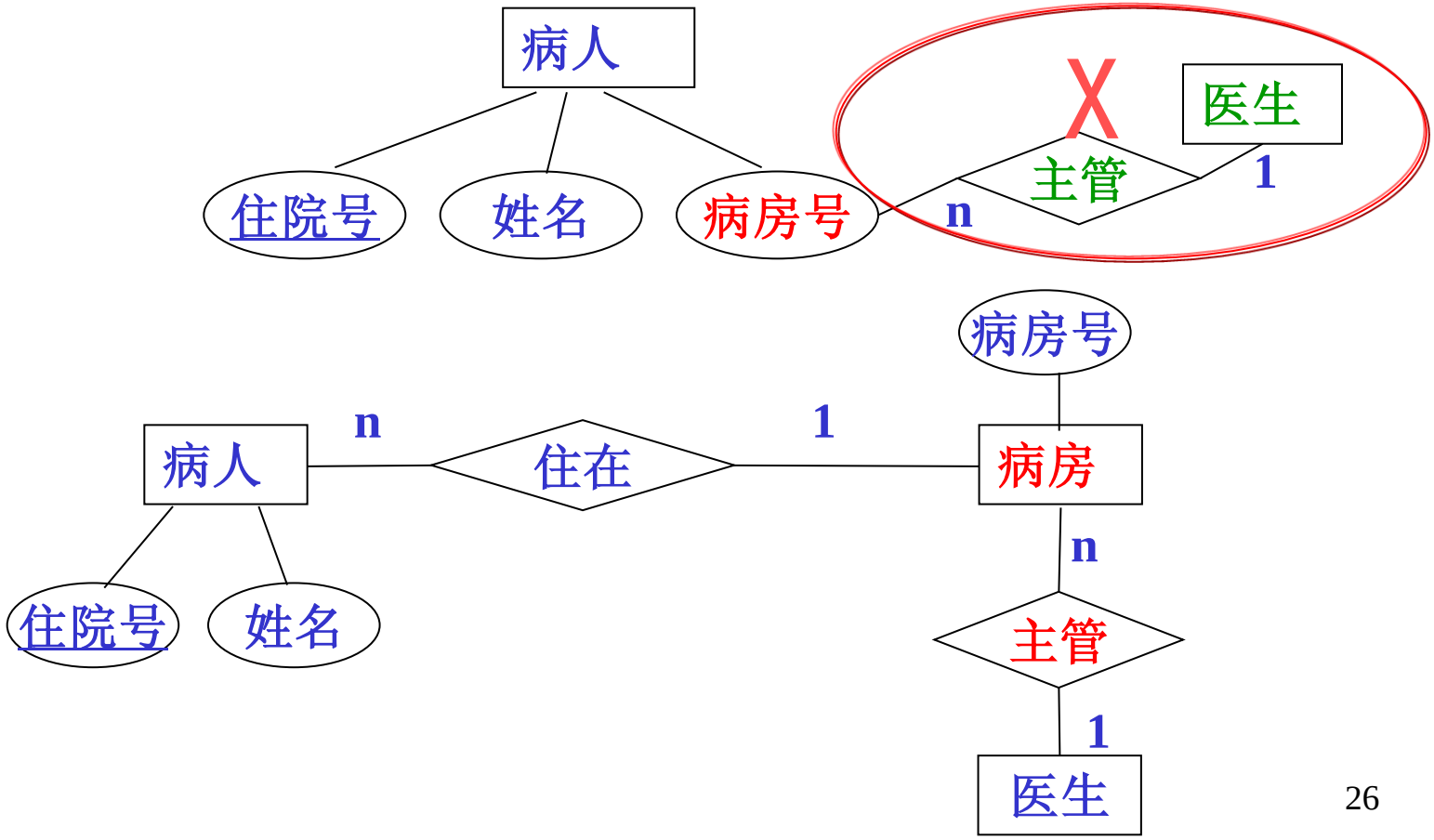
实体模型的调整原则（1）

- 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，不能是另一些属性的聚集。



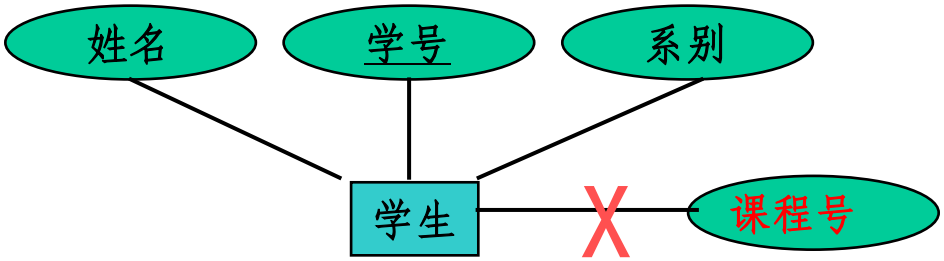
实体模型的调整原则（2）

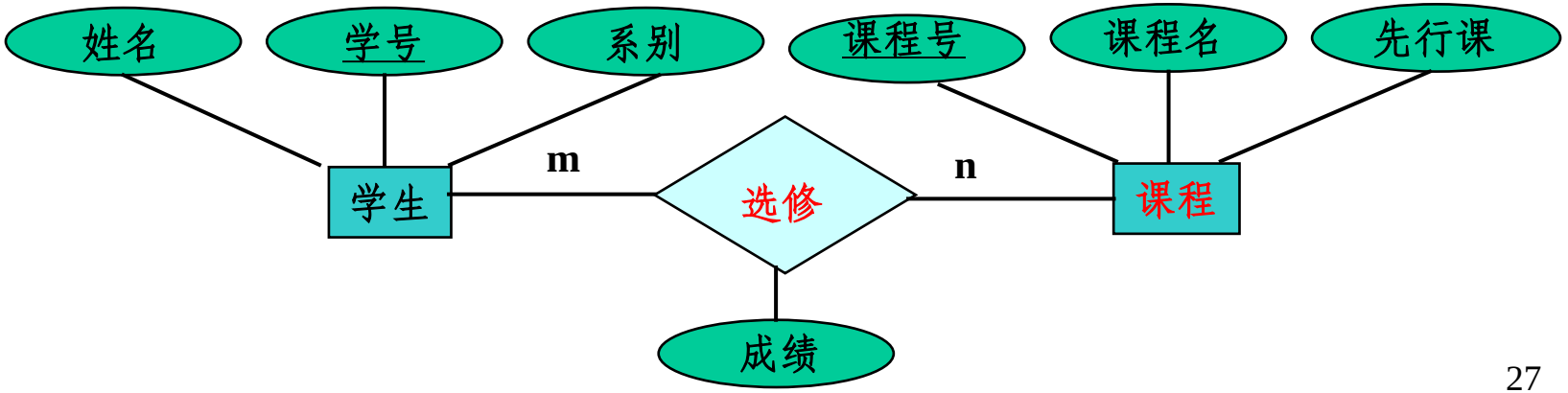
- 属性不能与其他实体具有联系，即E-R图中所表示的联系是实体之间的联系



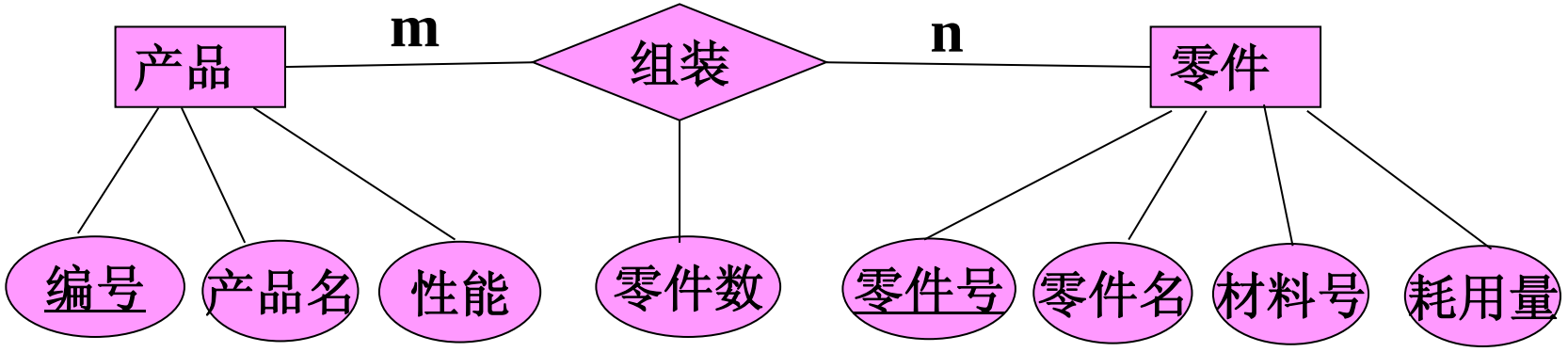
实体模型的调整原则（3）

- 实体和描述它的属性之间保持1：1或n：1的联系。对于1:n 或 n:m联系，要进行调整，一般可将该属性上升为实体

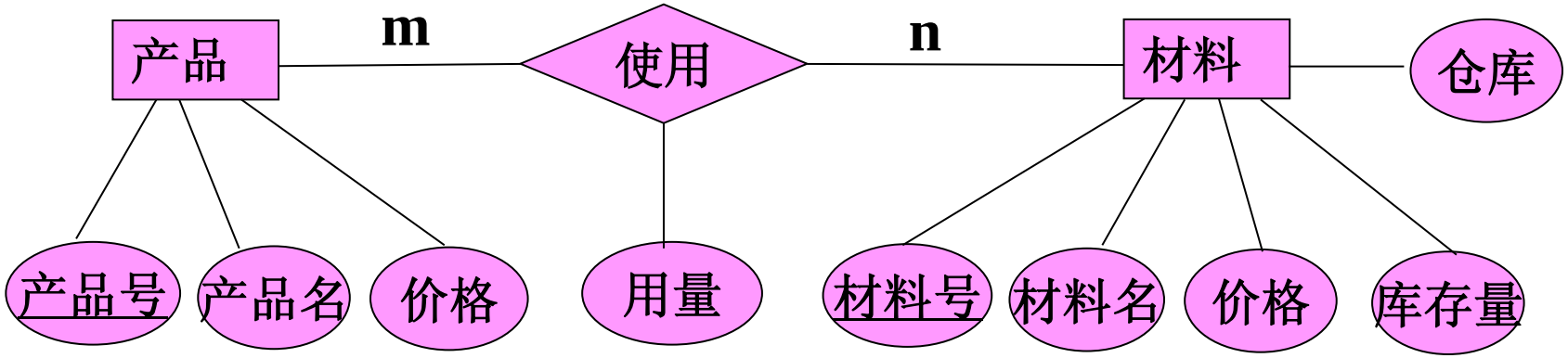




示例—某工厂信息系统的局部E-R图



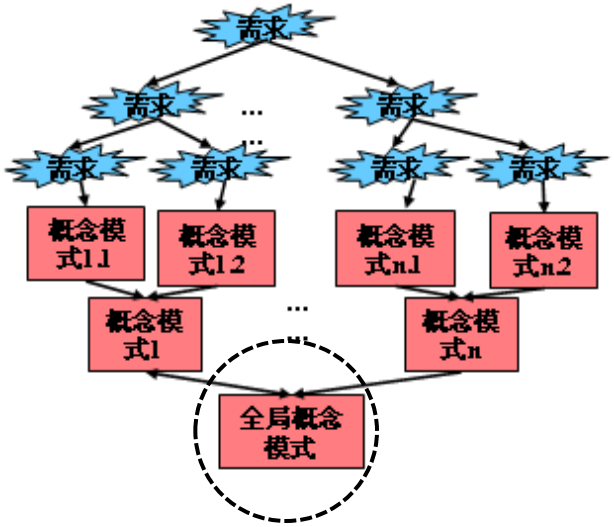
生产部门的局部E-R图



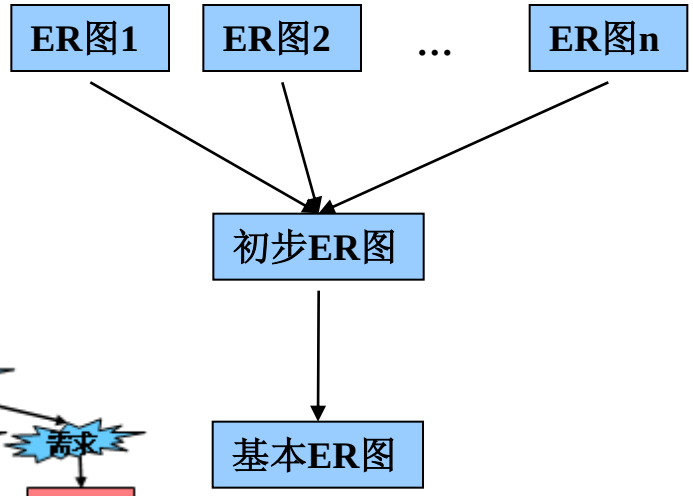
供应部门的局部E-R图

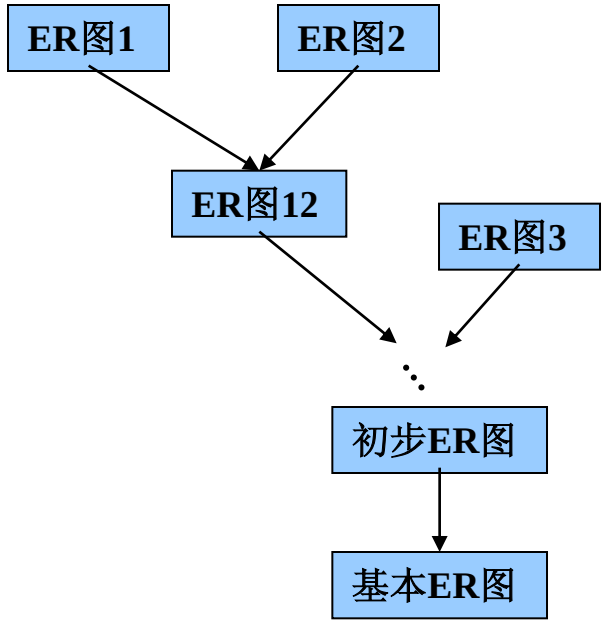
综合分E-R图形成总E-R图

- 设计总E-R图（全局概念模式）可以有两种方法：
  - 多个分E-R图一次集成（a)。
  - 逐步集成，用累加的方式一次集成两个分E-R图（b）。



(a)





(b)

集成局部E-R图的步骤

- 合并
  - 解决各分E-R图之间的冲突，将各分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。

局部E-R图的合并（1）

- 消除冲突，合并分E-R图，生成初步E-R图。
- 冲突主要包括：属性冲突、命名冲突和结构冲突

（1）属性冲突：属性的类型、取值范围或取值集合不同，或属性取值单位冲突。

解决——讨论协商解决。

（2）命名冲突：包括属性名、实体名、联系名之间的同名异义，异名同义。

- 同名异义：不同意义的对象具有相同的名字。
- 异名同义：相同意义的对象具有不同名字。

解决—— 建立命名表，统一命名，异名同义的名字可标为别名。

局部E-R图的合并（2）

（3）结构冲突：

- 同一对象在不同应用中有不同抽象。如在一应用中为实体，在另一应用中为属性。

解决——遵守实体与属性的划分原则，把属性变为实体或实体变为属性，使同一对象具有相同的抽象。

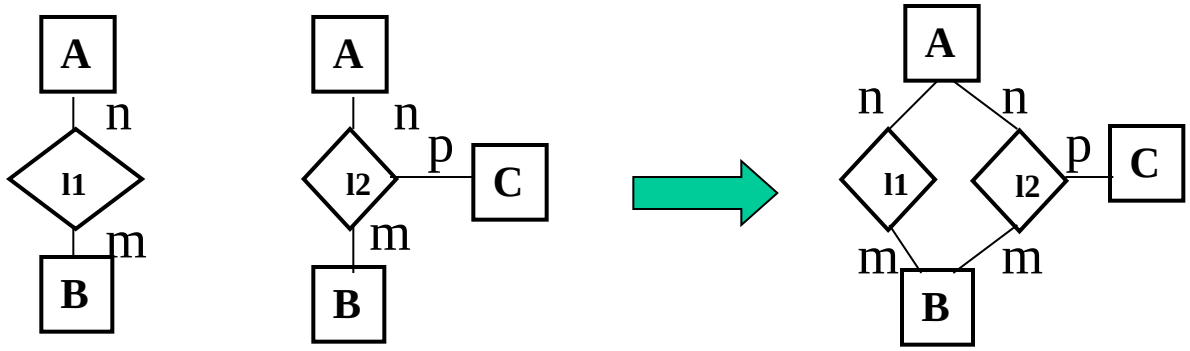
- 同一实体在不同分E-R图中属性个数、次序不同。

解决——同一实体的属性通常取分E-R图中属性的并，再适当调整次序。

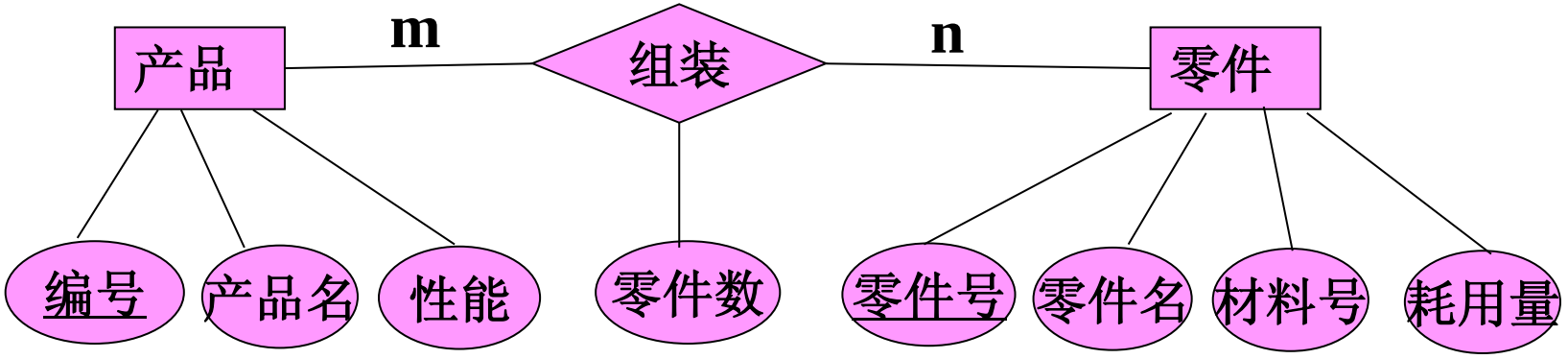
- 实体之间的联系在不同分E-R图中呈现不同类型。

解决——根据语义加以综合或调整。

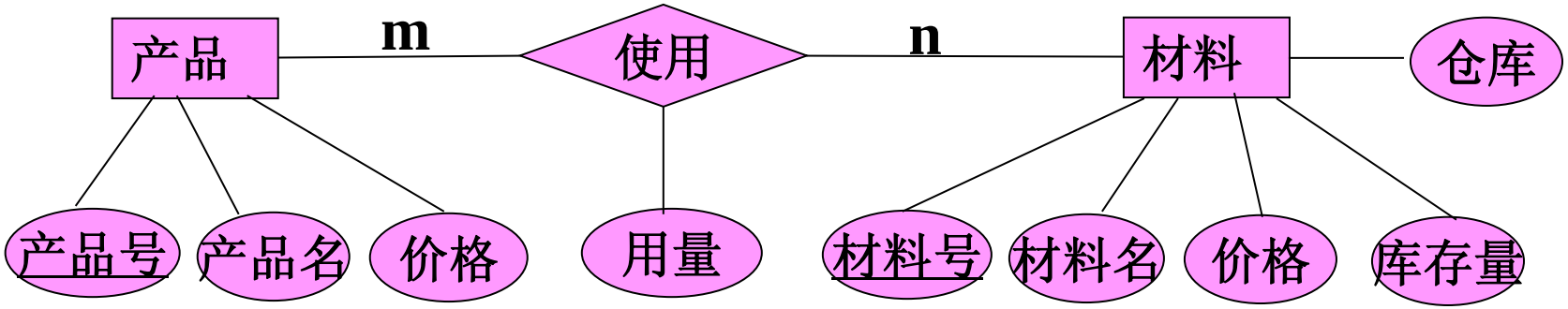
例：



示例

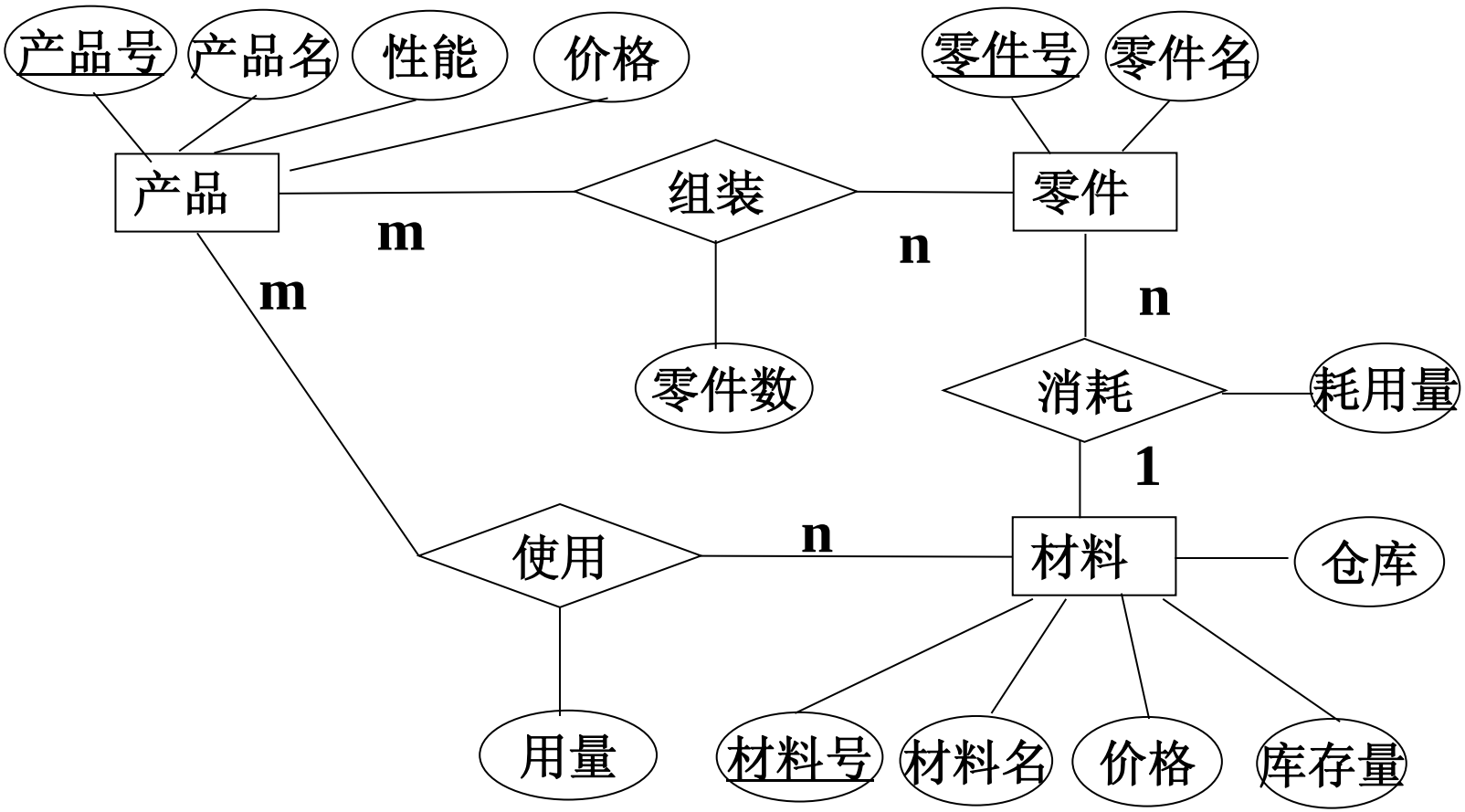


生产部门的局部E-R图



供应部门的局部E-R图

局部E-R图的合并（3）



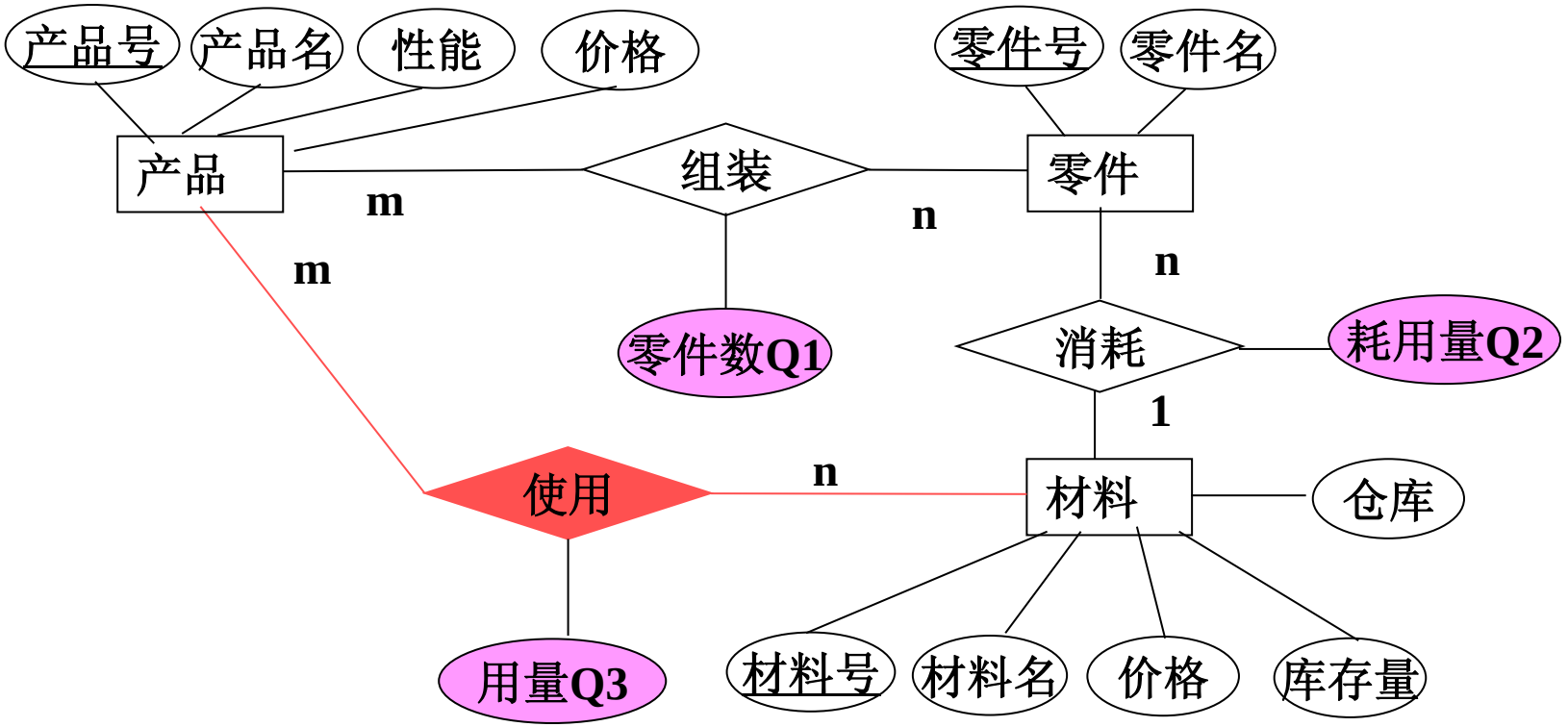
初步E-R图

修改初步E-R图设计基本E-R图(1)

- 消除不必要的冗余，设计基本E-R图
  - 初步E-R图中可能存在冗余的数据和冗余的联系。
    - 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它联系导出的联系。
  - 消除冗余有两种方法：
    - 分析法
    - 规范化方法

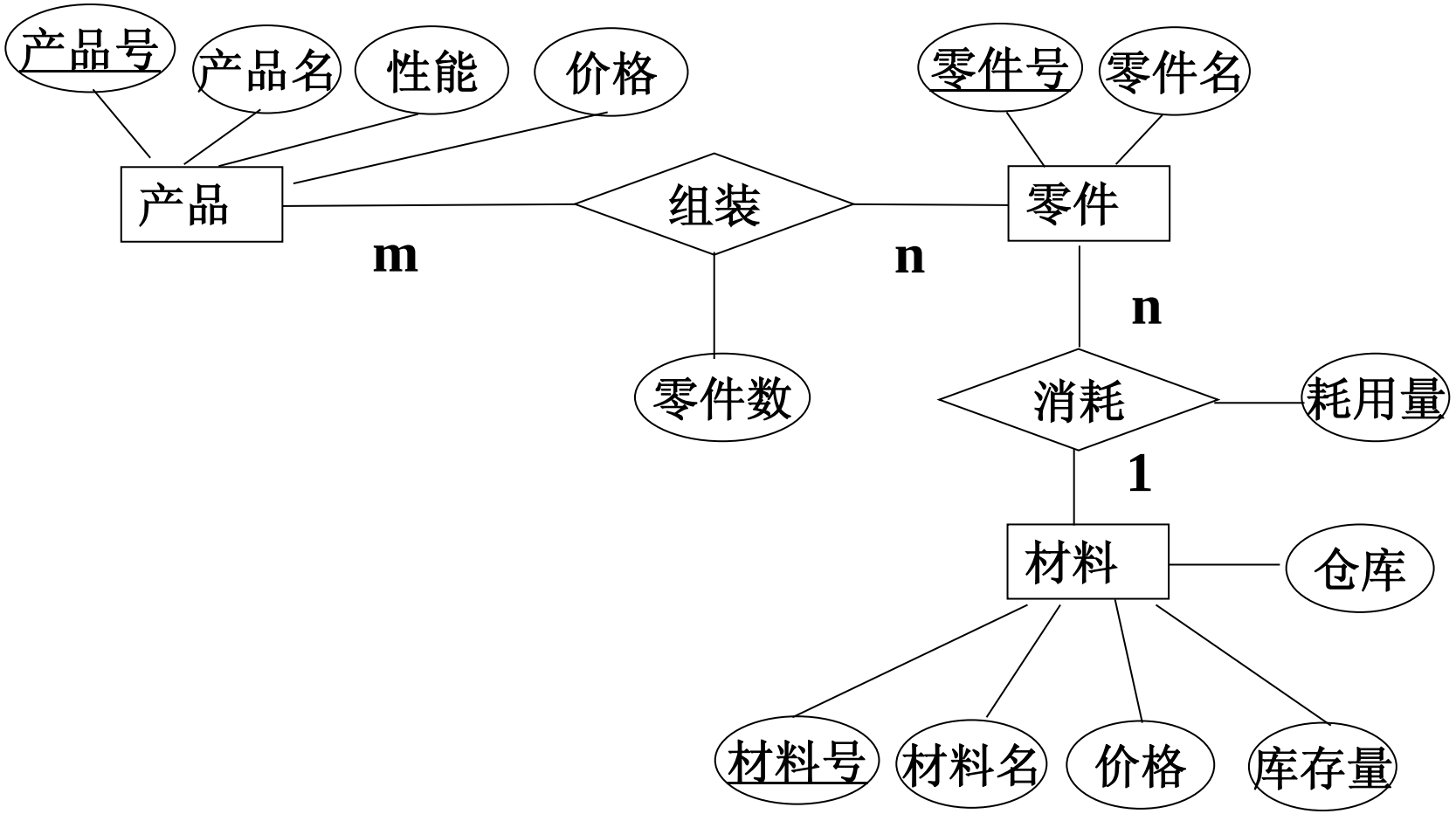
修改初步E-R图设计基本E-R图(2)

- 分析法消除冗余



- Q3=Q1*Q2，所以Q3是冗余的数据；
- 产品和材料之间的联系“使用”可以从产品与零件、零件与材料的联系导出，所以是冗余的联系也可消去。

修改初步E-R图设计基本E-R图(3)

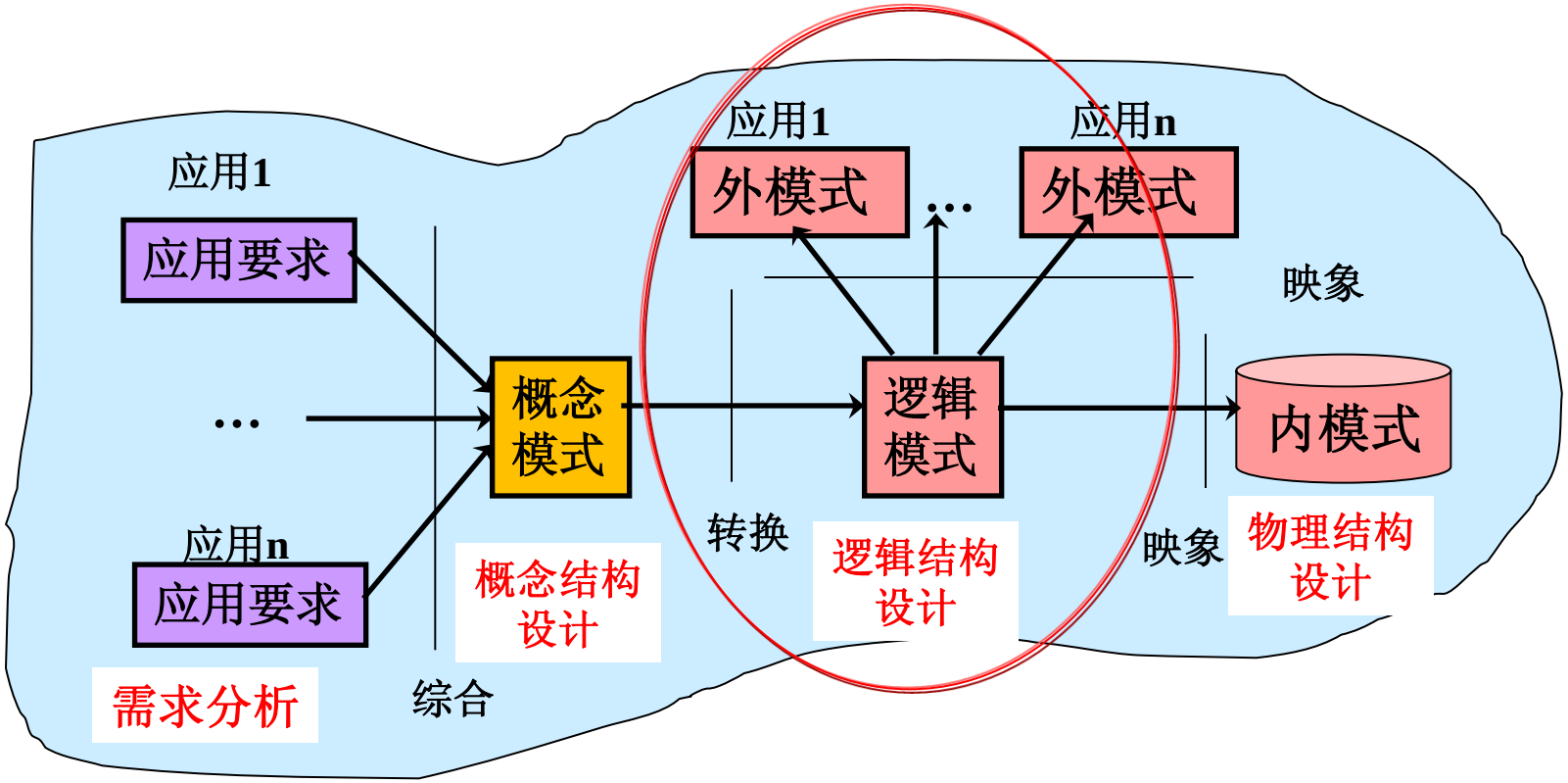


修改初步E-R图设计基本E-R图(4)

- 规范化方法消除冗余联系
  - 把E-R图中实体用符号表示;
  - 对每一对n:1、1:1或n:m联系表示为实体码之间的函数依赖表达式X→Y;
  - 利用函数依赖集的最小覆盖算法进行极小化处理。设原函数依赖表达式集合为F，最小覆盖集为G，则 D=F-G;
  - 考察D中每一个函数依赖表达式，确定是否冗余联系;
  - 去掉冗余联系后形成基本E-R图。

数据库逻辑结构设计

- 逻辑结构设计的任务就是把概念结构转换为选用的DBMS所支持的数据模型的过程。

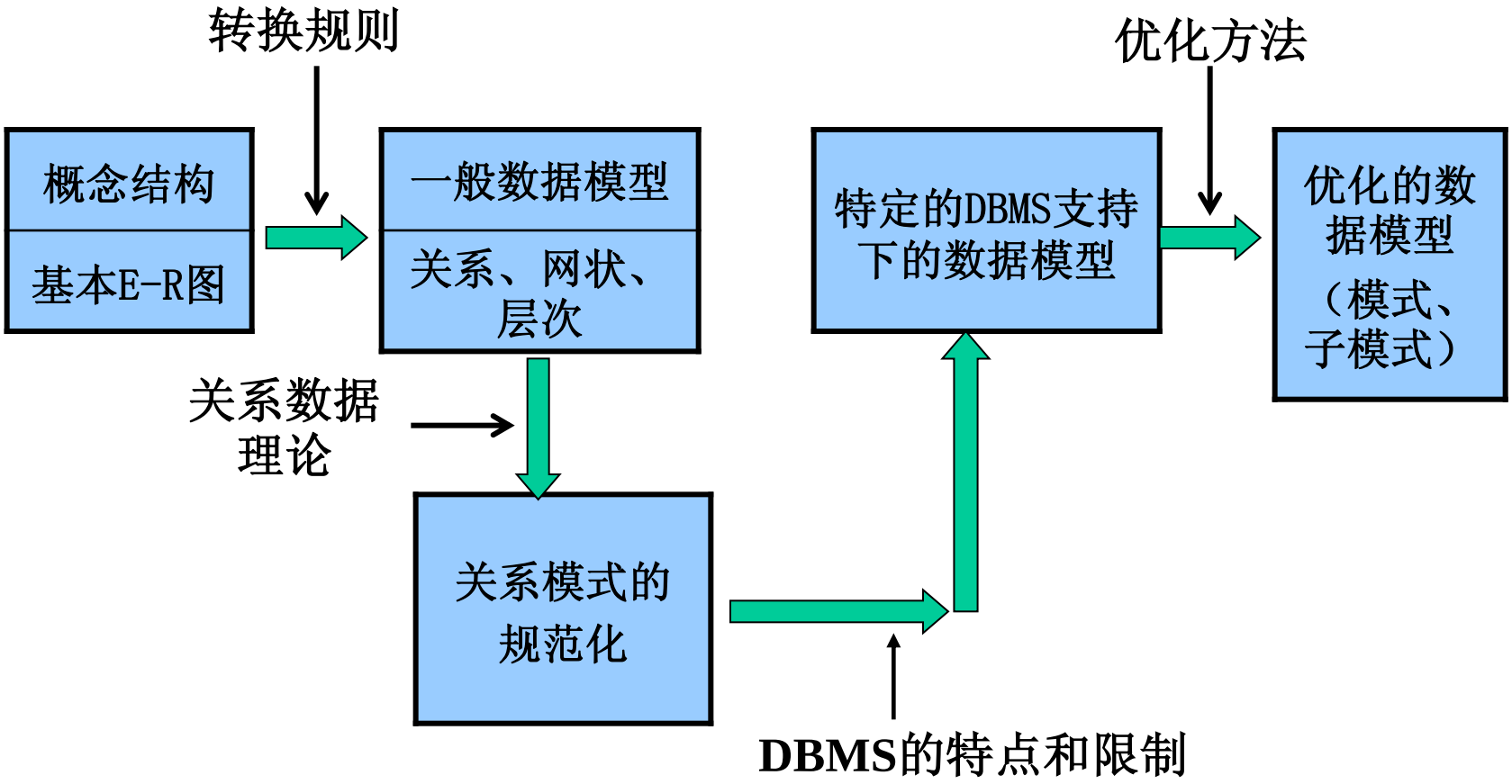


数据库逻辑结构设计

- 关系数据库逻辑结构设计的任务具体包括：
  - 形成初始关系数据库模式
  - 关系模式规范化
  - 关系模式优化
  - 子模式定义

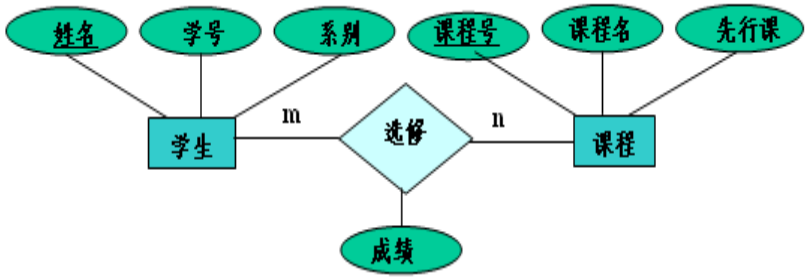
数据库逻辑结构设计

- 逻辑结构设计步骤如下：

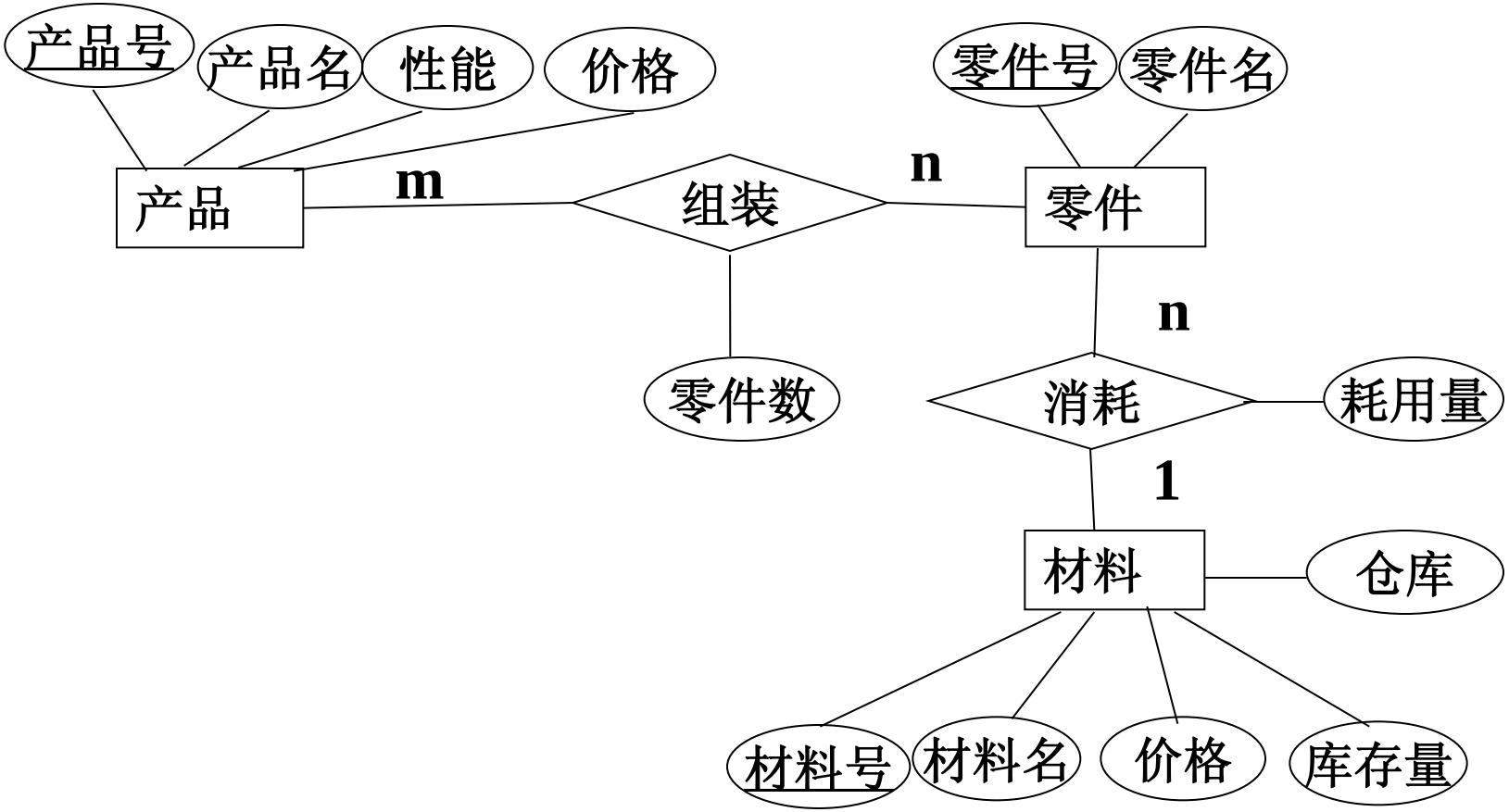


E-R图向关系模型的转换规则

- 一个实体型转换为一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。
- 一个联系转换为一个关系模式。与该联系相连的各实体的码以及联系的属性转换为关系的属性：
  - 若联系为1：1，则每个关系的码均是该关系的候选码；
  - 若联系为1：n，则该关系的码是n端实体的码；
  - 若联系为n：m，则该关系的码是诸实体码的组合。
- 三个或三个以上实体间的多元联系，转换为一个关系模式，与该多元联系相连的各实体的码以及联系的属性转换为关系的属性，而关系的码为各实体码的组合。
- 具有相同码的关系可以合并



E-R图向关系模型示例

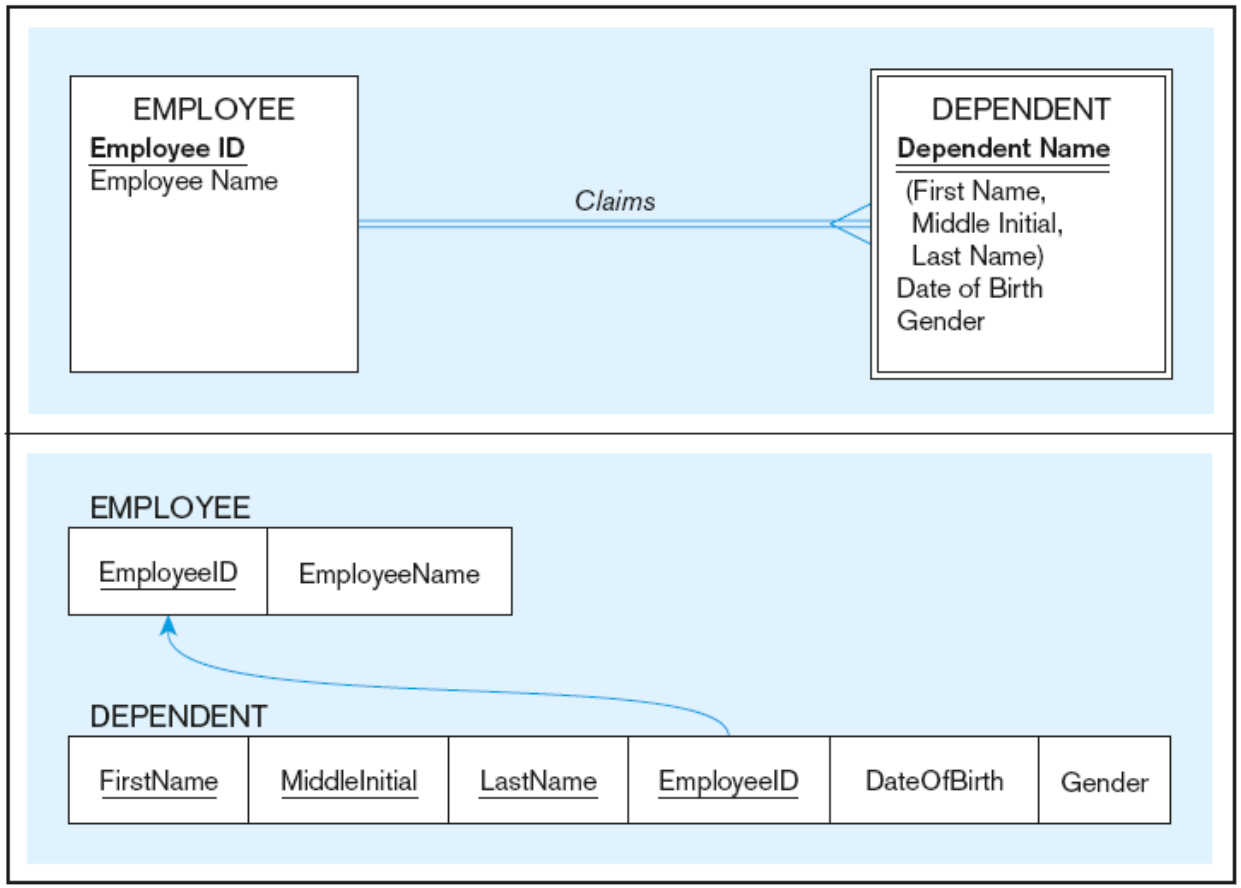


E-R图向关系模型的转换规则

- 弱实体类型的转换
  - 对于每个弱实体类型，创建一个新的关系，该关系中包含所有弱实体类型的属性。
  - 把标识关系的主码添加到新关系中，并将其作为新关系的外码。
  - 新关系的主码是标识关系的主码和弱实体类型的部分标识(码）的组合。

弱实体类型的转换示例

FIGURE 4-11 Example of mapping a weak entity
(a) Weak entity DEPENDENT

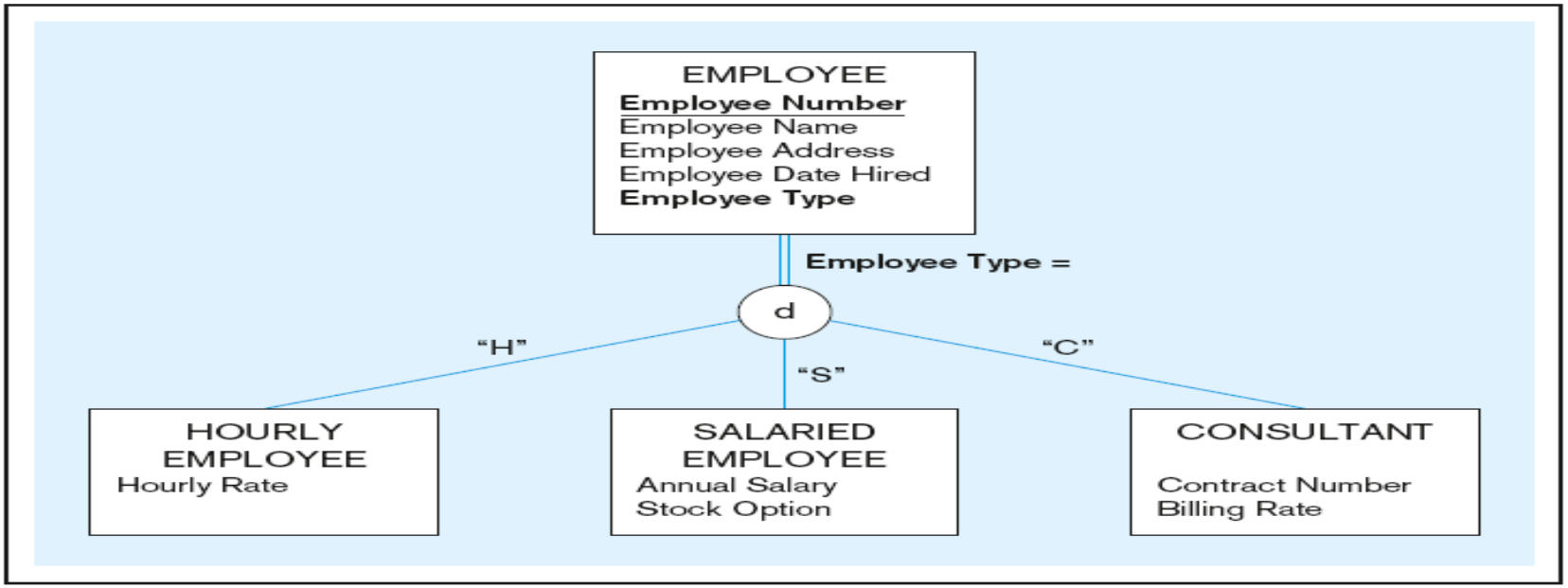


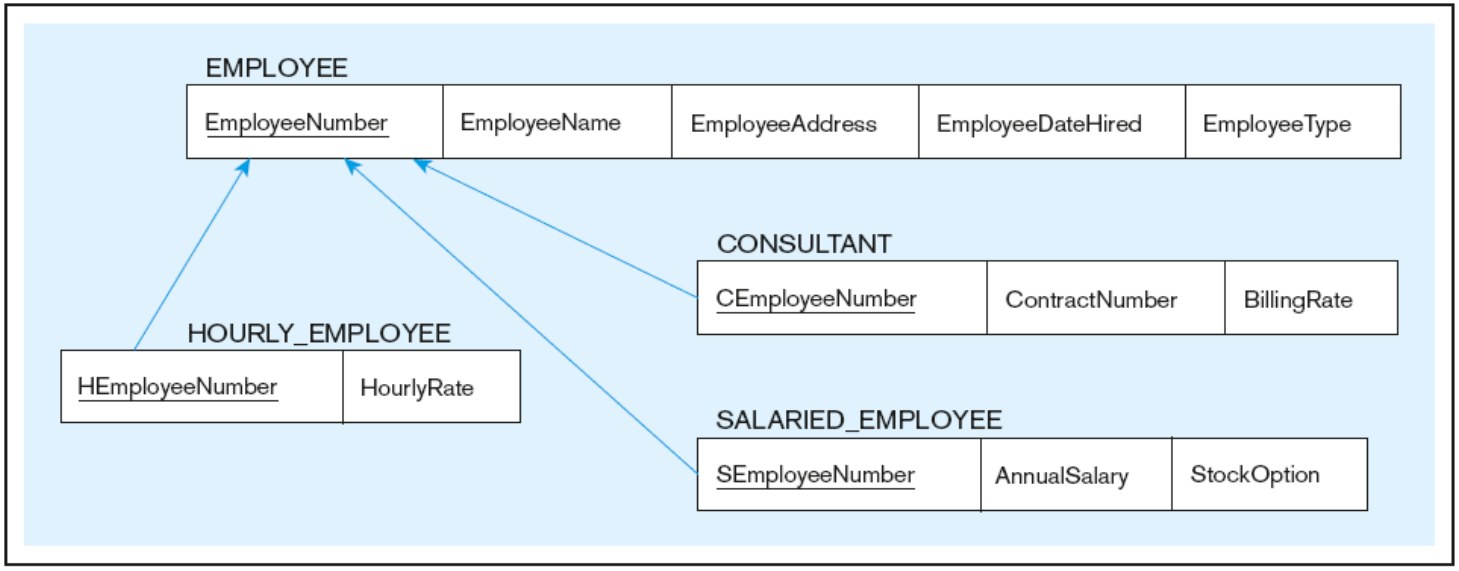
(b) Relations resulting from weak entity

E-R图向关系模型的转换规则

- 超类/子类联系的转换
  - 为超类和每个子类创建单独的关系。
  - 在超类所创建的关系中，包含所有子类成员都共有的属性，包括主码。
  - 在超类中包含一个（或多个）属性作为**子类判定符**。
  - 在为每个子类所创建的关系中，**包含超类的主码**以及子类特有的属性。

超类/子类联系转换的示例





MODERN DATABASE MANAGEMENT , Jeffrey A. Hoffer, et al.

关系模型的规范化与优化（1）

- 规范化
  - 按照数据依赖的理论，逐一分析转换所得关系模式，判断是否存在部分函数依赖、传递函数依赖、多值依赖等，确定它们的范式等级；
- 优化
  - 按应用系统的处理要求，确定是否进行模式合并或分解；
  - 为了提高存取效率和存储空间的利用率，可以对关系模式进行必要的分解。
    - 水平分解
    - 垂直分解

关系模型的规范化与优化（2）

- 水平分解
  - 是把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，以提高系统效率。
    - 80/20原则

      可以把经常使用的那一部分数据分解出来作为一个关系，其他数据作为另一个关系。
    - 数据分片

      如果关系R上具有n个事务，而且多数事务存取的数据不相交，则R可以分解为少于或等于n个子关系。

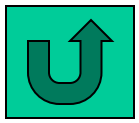
关系模型的规范化与优化（3）

- 垂直分解
  - 是把关系模式R的属性分解为若干子集合，形成若干子关系模式。
    - 垂直分解的原则是，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。
    - 垂直分解必须确保无损连接性和保持函数依赖。

设计用户子模式

- 根据局部应用的需求，结合具体DBMS的特点，设计用户外模式（视图）
  - 使用更符合用户习惯的别名；
  - 可以对不同级别的用户定义不同的视图，以保证系统的安全性；
  - 通过定义视图降低复杂查询的难度，简化用户对系统的使用。

Create View <视图名> [(<列名>[,<列名>] …)]
As <子查询>



数据库的物理设计

- 确定数据库的存储结构
- 选择关系的存取方法
  - 存取方法是使事务能够快速存取数据库中数据的技术。

确定数据库的存储结构

- 确定存放位置
  - 经常存取部分与和存取频率较低部分分开存放
  - 数据和日志备份放于不同的磁盘上
- 确定系统配置
  - 确定系统配置变量、存储分配参数，进行物理优化

关系的存取方法

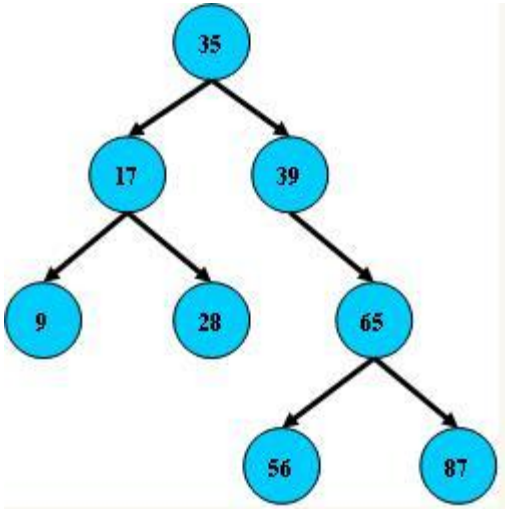
- 常用的存取方法：
  - 索引方法
  - 聚集方法
  - HASH方法

索引方法

- 基本概念
  - 索引记录/索引项，是索引文件的记录，包括两个域：
    - 索引域：存储数据文件中一个或一组域的一个值（K）
    - 指针：指向索引域值为K的记录所在磁盘块的地址。
- 分类
  - 按照索引文件结构
    - 稀疏索引：把所有数据记录按索引域值分成许多组，每组设立一个索引项；
    - 稠密索引：为每个记录设立一个索引项，记录存放是任意的，但索引是有序的。

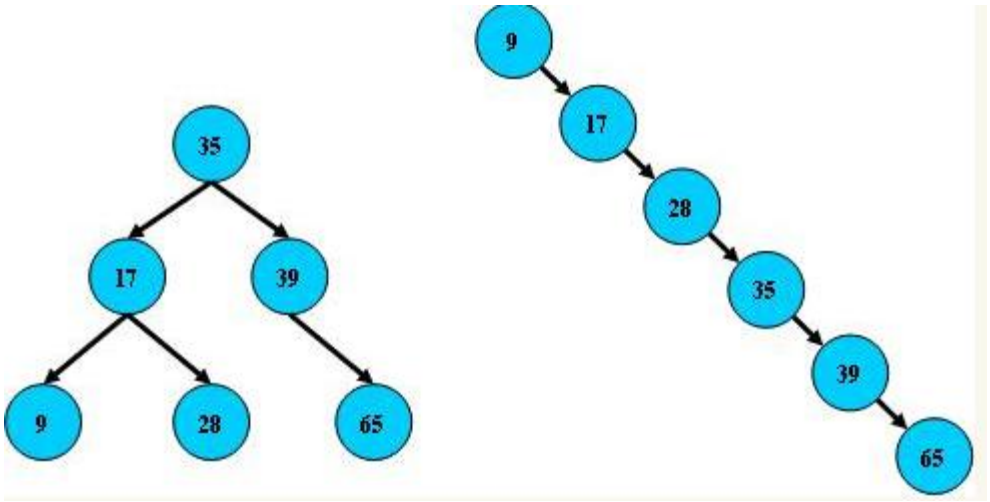
多级索引

- 二叉树索引
  - 每个节点只有一个关键字，有两个指针，分别指向关键字值小于或大于当前节点值的节点；
- 多枝树索引
  - 每个节点有D（D>=2)个关键字值，则有D+1个指针,每两个指针之间对应相临关键字之间取值的区间；



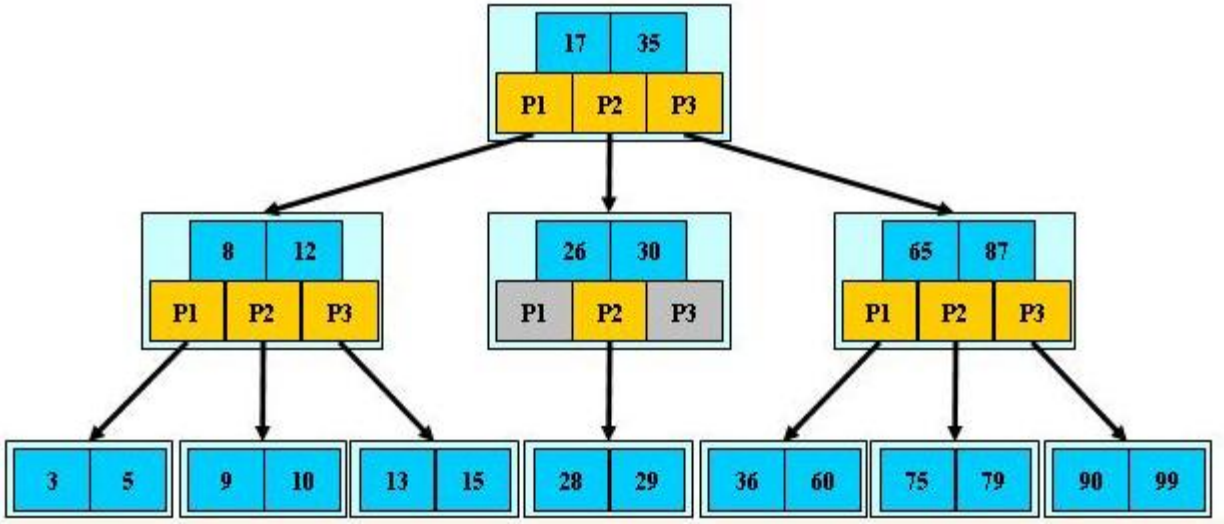
多级索引

- 二叉树索引的问题



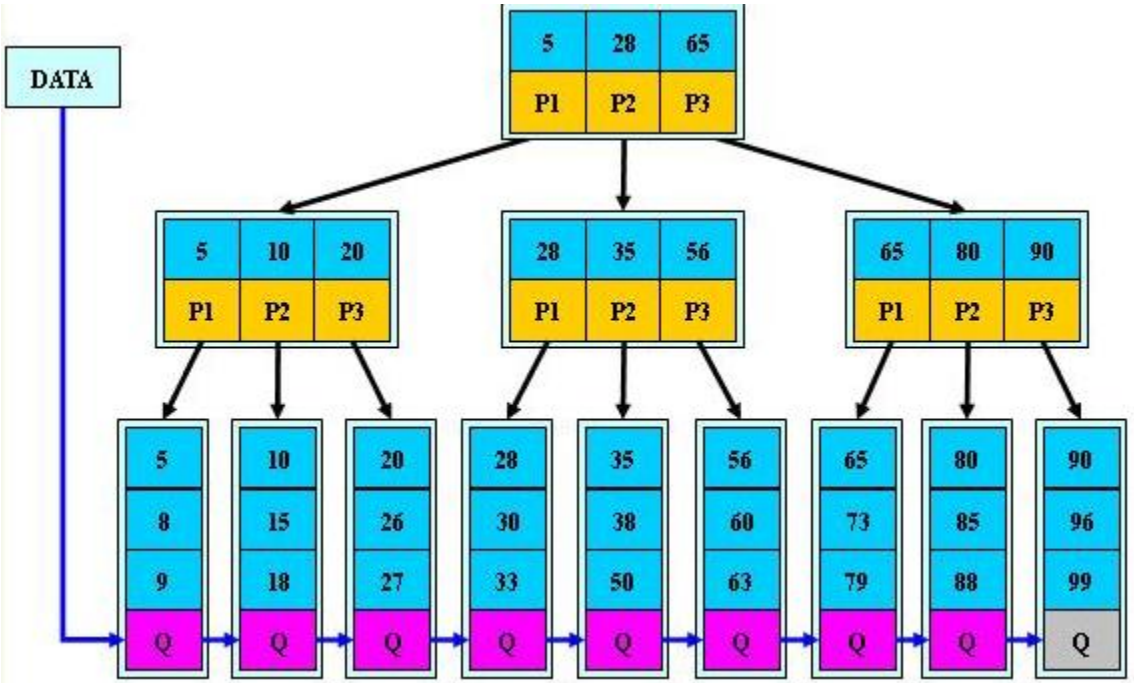
多级索引

- B树（平衡树）索引
  - 是附加限制条件的索引树。限制了每个节点放置关键字与指针的最小和最大个数，并且所有的叶节点都在同一层上。B树的关键字是散布在各层上。



多级索引

- B^+树索引
  - 是B树的改进。把树中所有关键字都按递增次序从左到右安排在叶节点上，并且链接起来。B^+树能同时进行随机查找和顺序查找。



索引存取方法的选择

- 选择索引域原则：
  - 经常在查询条件中出现的属性
  - 经常作为最大值和最小值库函数的参数
  - 经常作为连接属性
- 索引并非越多越好

聚集方法

- 基本概念
  - 把关系中某个属性/组(聚集键)值相同的记录集中存放在连续的物理块，称为聚集。能够提高该属性的查询速度
- 一个关系只能参加一个聚集。

聚集存取方法的选择

- 一般原则
  - 经常进行连接操作的关系可建立聚集
  - 单个关系的某组属性经常进行相等比较
  - 关系的某个属性组值重复率高
- 注意问题
  - 建立与维护聚集系统开销很大，对于更新操作远远多于连接操作的关系不应使用聚集方法。

HASH文件

- 一种支持快速存取的文件存储方法
- 基本概念

  通过HASH函数将记录关键字转换成地址。

  设F是一个文件，A是F的HASH域，H是定义在A的值域上的HASH函数，即有H(A)。对于记录r（其A的值为a)的存储地址由H(a)确定。

HASH存取方法的选择

- 如果关系的属性主要出现在等连接条件中，或出现在相等比较条件中，而且满足下列条件之一，可以选择该方法：
  - 关系的大小可预知，而且不变；
  - 如果关系大小动态改变，则须DBMS提供动态HASH存取方法。

小结

- 数据库设计的阶段以及各阶段的任务
- E-R法以及系统概念结构设计自底向上的方法
- 逻辑结构设计与物理结构设计