2025年春季学期

数据库系统概论

An Introduction to Database Systems

第七章　数据库设计

中国科学技术大学
人工智能与数据科学学院

黄振亚，huangzhy@ustc.edu.cn

http://staff.ustc.edu.cn/~huangzhy/Course/DB2025.html

7.3 概念结构设计

7.3.4 UML

- 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是为软件开发的各个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。

用UML的类图建立概念模型

- UML中的类（class）大致对应E-R图中的实体。
- 用UML的类图表示E-R图
  - 实体型：用类表示，矩形框上部记上实体名，下面列出属性名。
  - 实体的码：在类图中在属性后面加“PK”（primary key）来表示码属性。
  - 联系：用类图之间的“关联”来表示。

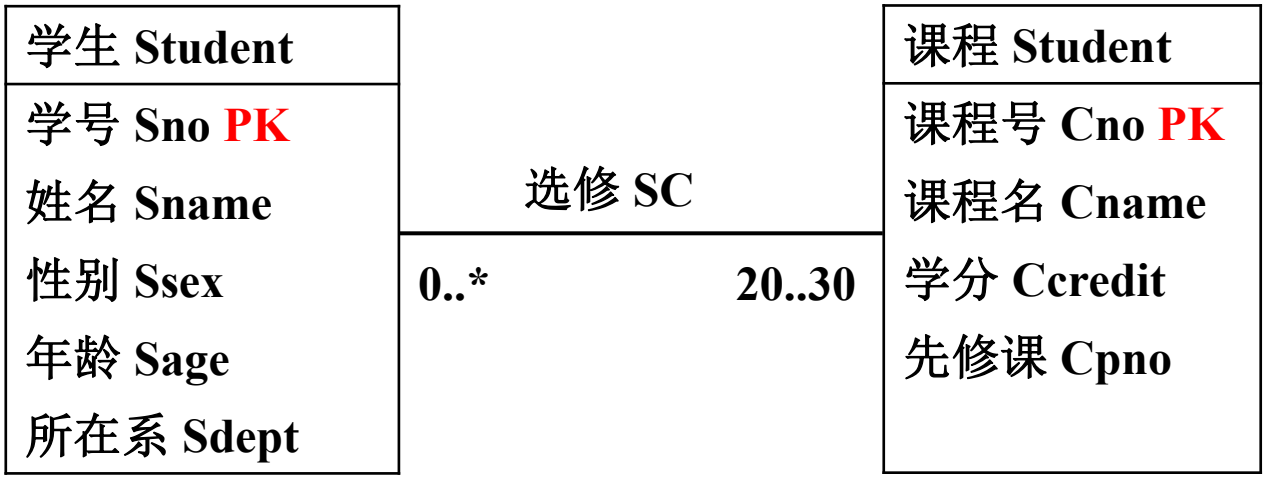


图7.18 用UML的类图表示E-R图示例

SYBASE PowerDesigner

数据库建模UML工具

用UML的类图建立概念模型（续）

- 基数约束
  - UML中关联类之间的基数约束和 E-R图中基数约束的概念类似。
- 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

SYBASE PowerDesigner

数据库建模UML工具

UML工具

- SYBASE PowerDesigner
  - https://sybase-powerdesigner.informer.com/

Sybase PowerDesigner 16.6

Visualize the impact of changes before they are applied

User rating ★★★★☆ 4 (153 votes)

Your vote ☆☆☆☆☆

Latest version: 16.6 (See all)

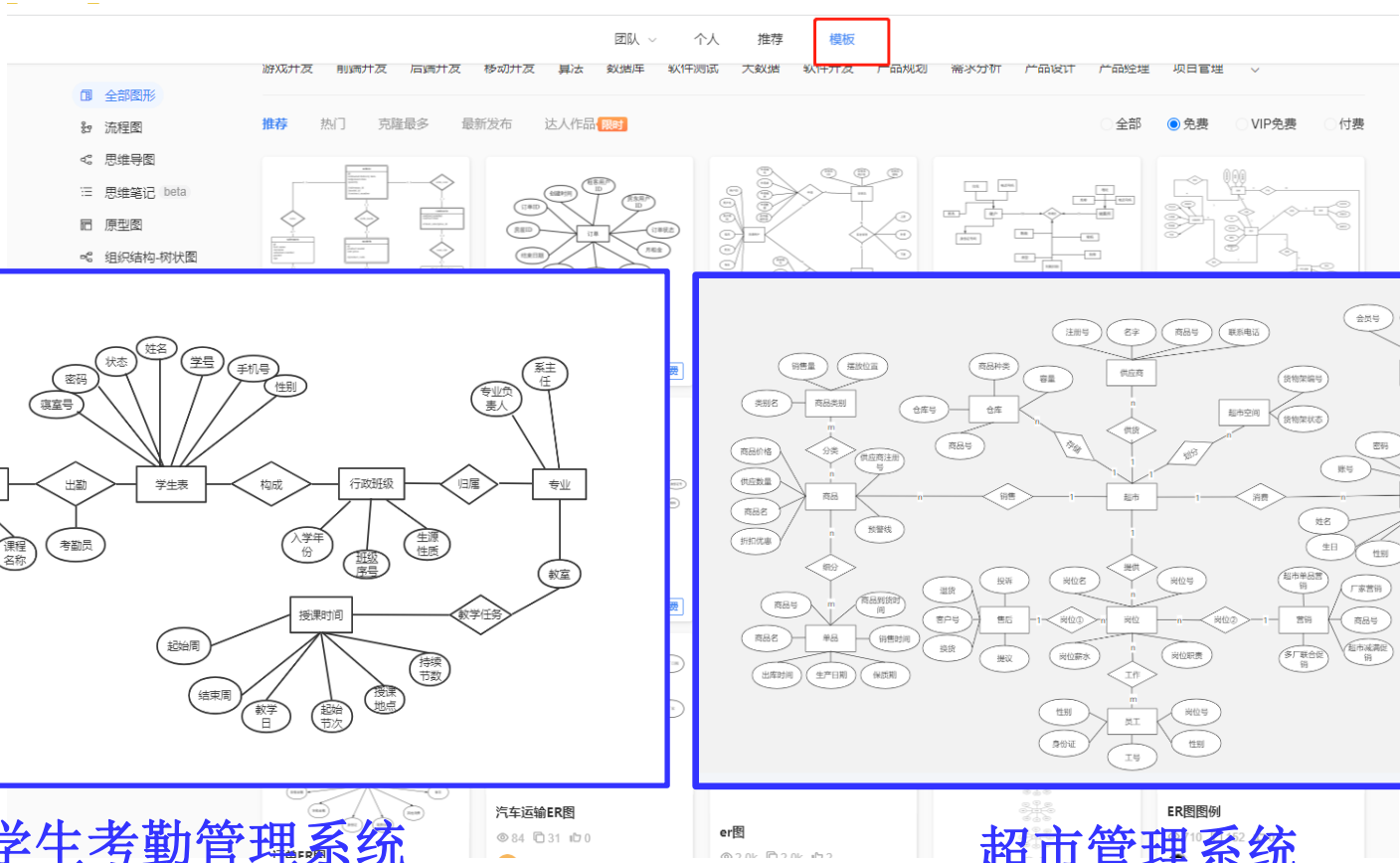
Developed by: SAP

Power Designer has integrated support for:

- Business Process Modeling supporting Business Process Modeling Notation
- Code generation for Java, C#, VB, .NET, EJB3, JSF, WinForm, and others.
- Data Warehouse Modeling
- Eclipse plug-in
- Object modeling (UML 2.0 diagrams)
- Report generation
- XML Modeling supporting XML Schema and DTD standards
- Visual Studio 2005 add-in

E-R图设计软件

- https://processon.com/
  - 含有大量ER图模板



学生考勤管理系统

超市管理系统

7.3 概念结构设计

7.3.1 概念结构

7.3.2 E-R模型

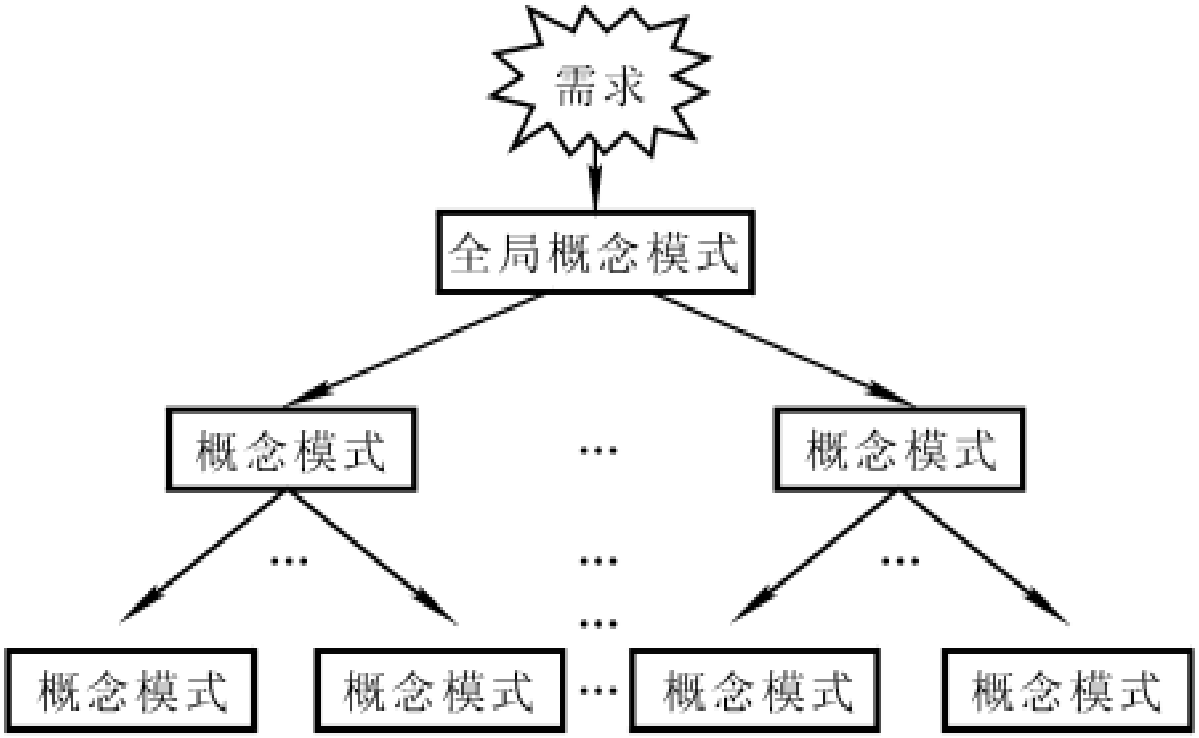
*7.3.3 扩展的E-R模型

*7.3.4 UML

7.3.5 概念结构设计

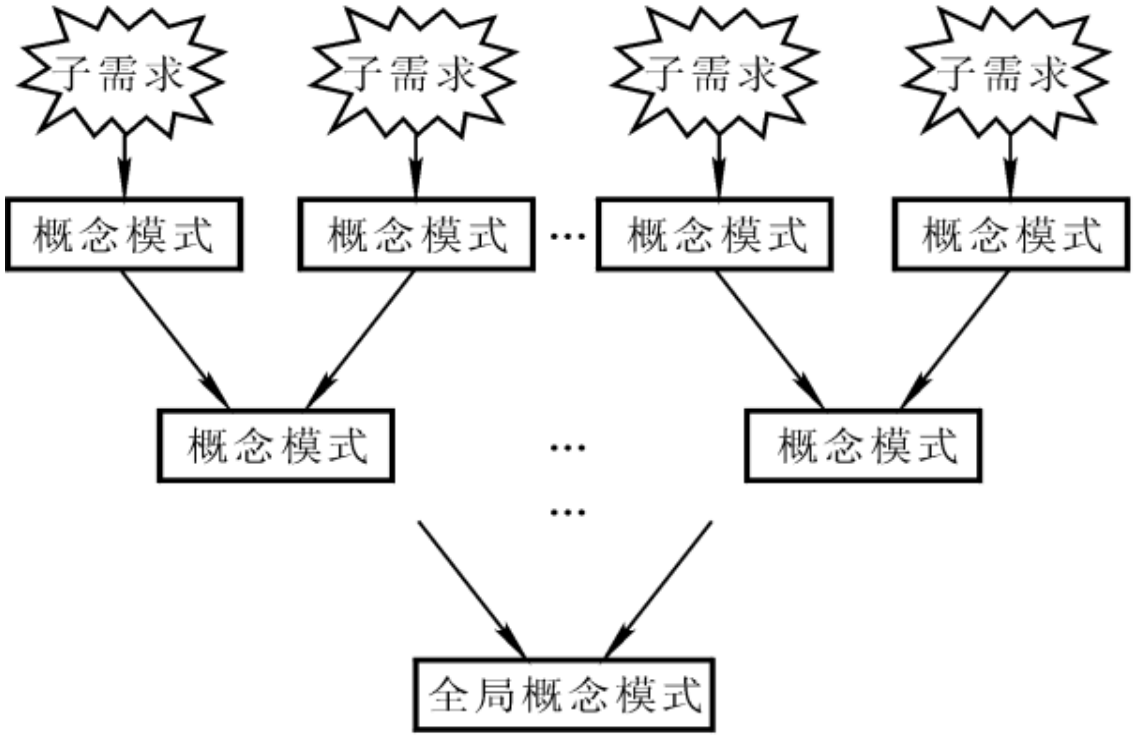
7.3.5 概念结构设计

- 概念结构设计方法与步骤（四类方法）
  - 1. 自顶向下
    - 首先定义全局概念结构的框架，然后逐步细化



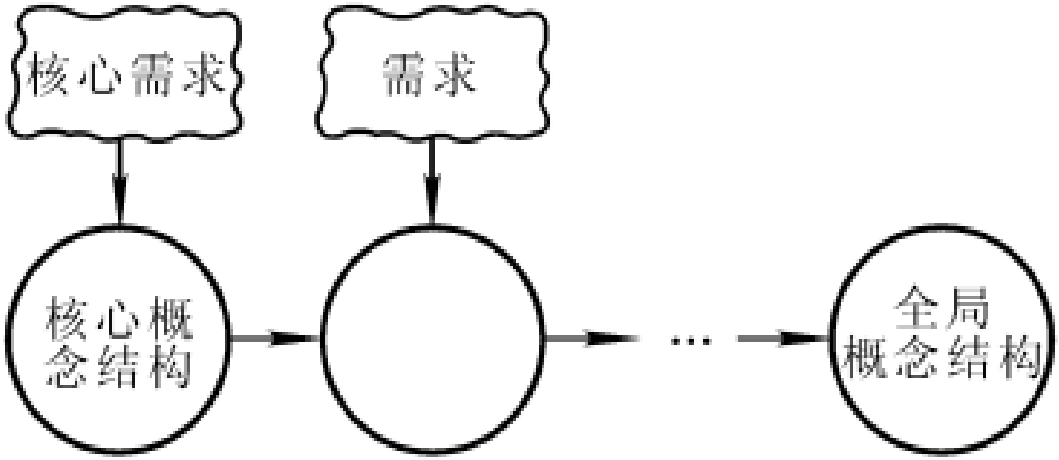
7.3.5 概念结构设计

- 概念结构设计方法与步骤（四类方法）
  - 2. 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



7.3.5 概念结构设计

- 概念结构设计方法与步骤（四类方法）
  - 3. 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



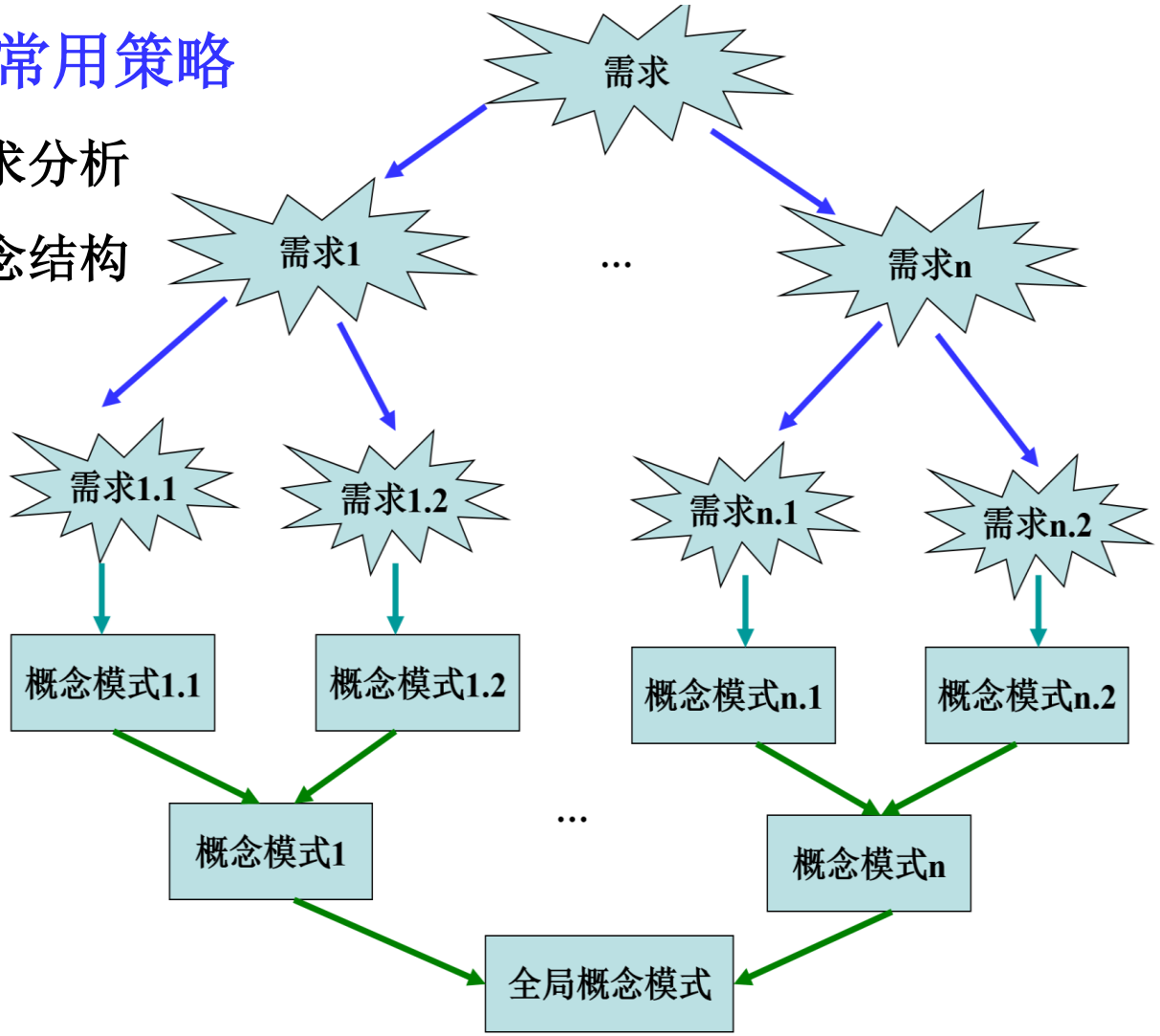
逐步扩张策略

7.3.5 概念结构设计

- 概念结构设计方法与步骤（四类方法）
  - 4. 混合策略
    - 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。

7.3.5 概念结构设计

- 开发大型系统时的常用策略
  - 自顶向下地进行需求分析
  - 自底向上地设计概念结构

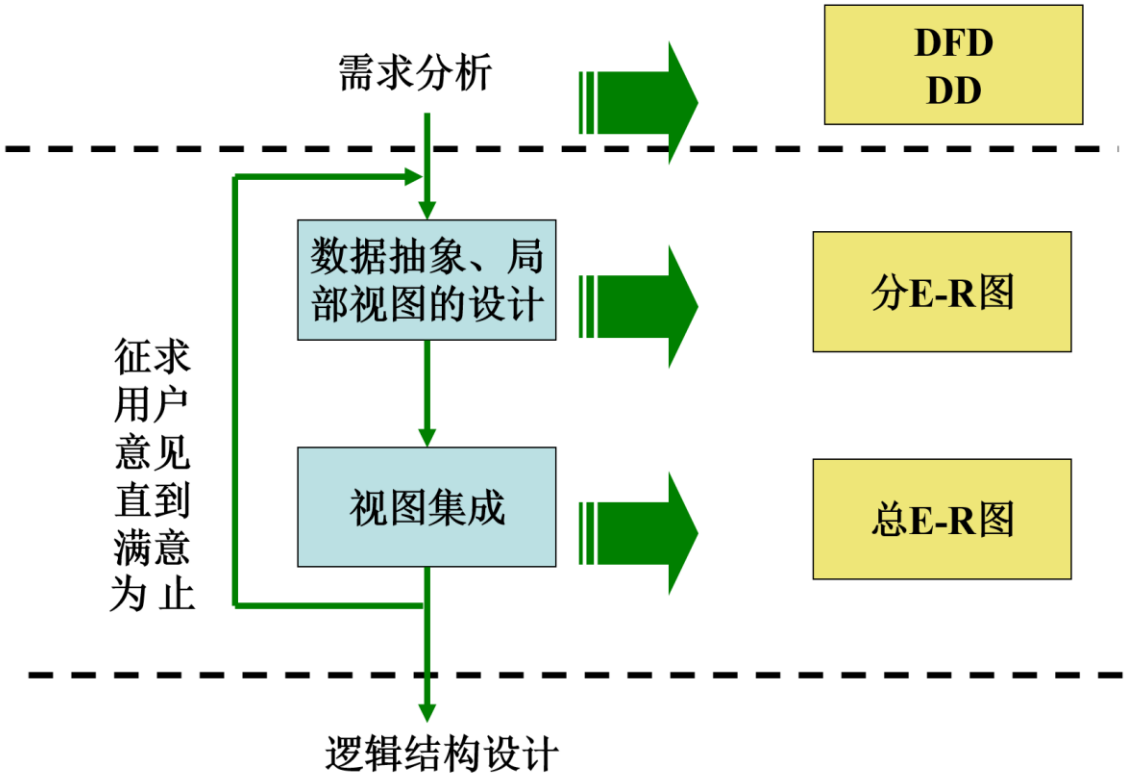


概念结构设计的方法与步骤（续）

- 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，形成子系统，即分E-R图

第2步：集成局部视图，得到全局概念结构，即总E-R图



DD数据字典
DFD用户需求规格说明书

7.3.5 概念结构设计

- 第1步：局部设计，形成（子系统）分 E-R图
  - 问题：如何确定实体和属性？
    - 没有形式上可以截然划分的界限

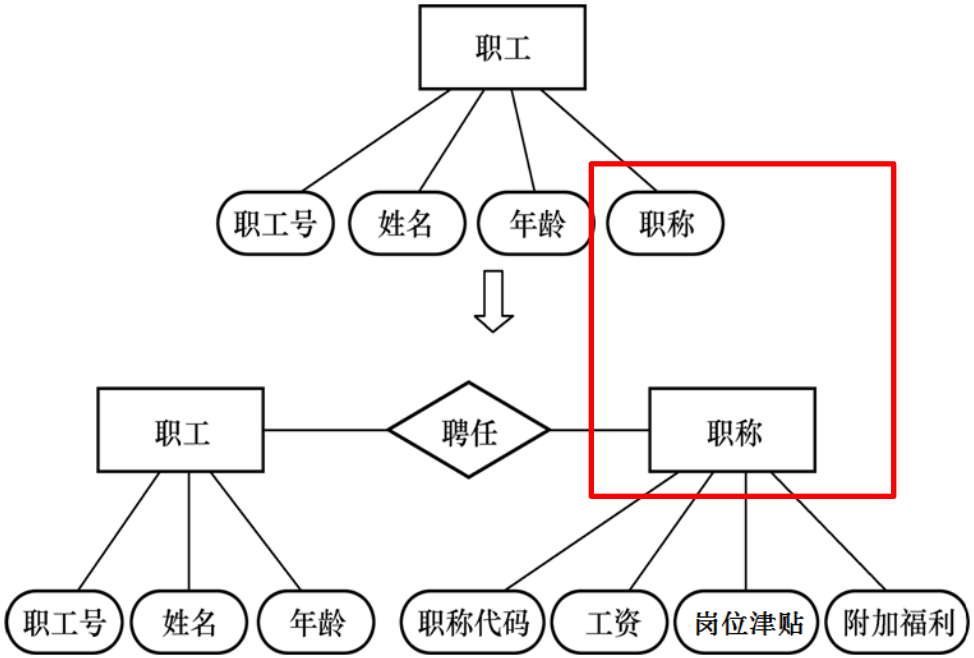
1. 实体与属性的划分原则
   - 初步：现实世界应用环境有大体划分
   - 调整原则：为了简化E-R图的处置，现实世界的事物能作为属性对待的，尽量作为属性对待
   - 两条准则：
     - 1）作为属性，不能再具有需要描述的性质。属性必须是不可分的数据项，不能包含其他属性
     - 2）属性不能与其他实体具有联系，即E-R图中所表示的联系是实体之间的联系

概念结构设计（续）

[例1] 职工是一个实体：

职工号、姓名、年龄是职工的属性，如何设计“职称”？

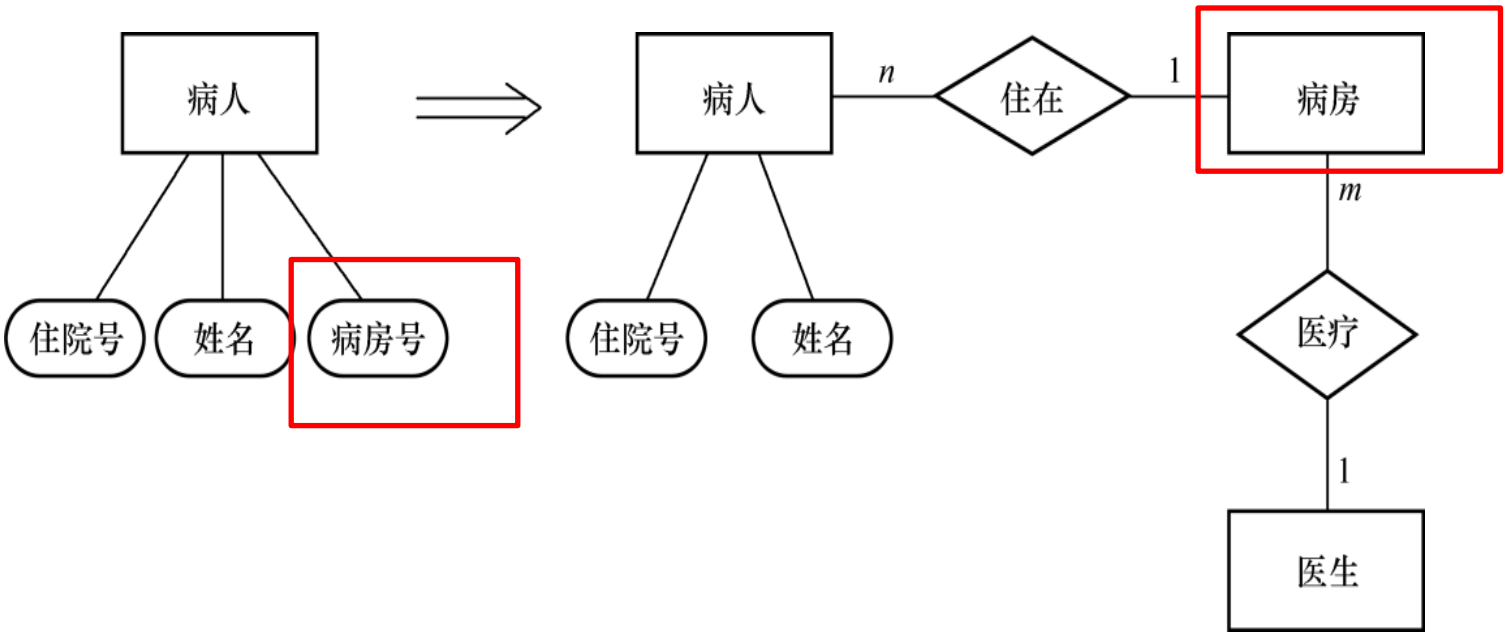
- 职称如果没有与工资、福利挂钩，无进一步描述的特点，根据准则（1）可以作为职工实体的属性
- 如果不同的职称有不同的工资、住房标准和不同的附加福利，则职称作为一个实体更恰当



概念结构设计（续）

[例2] 在医院中，如何设计“病房号”？

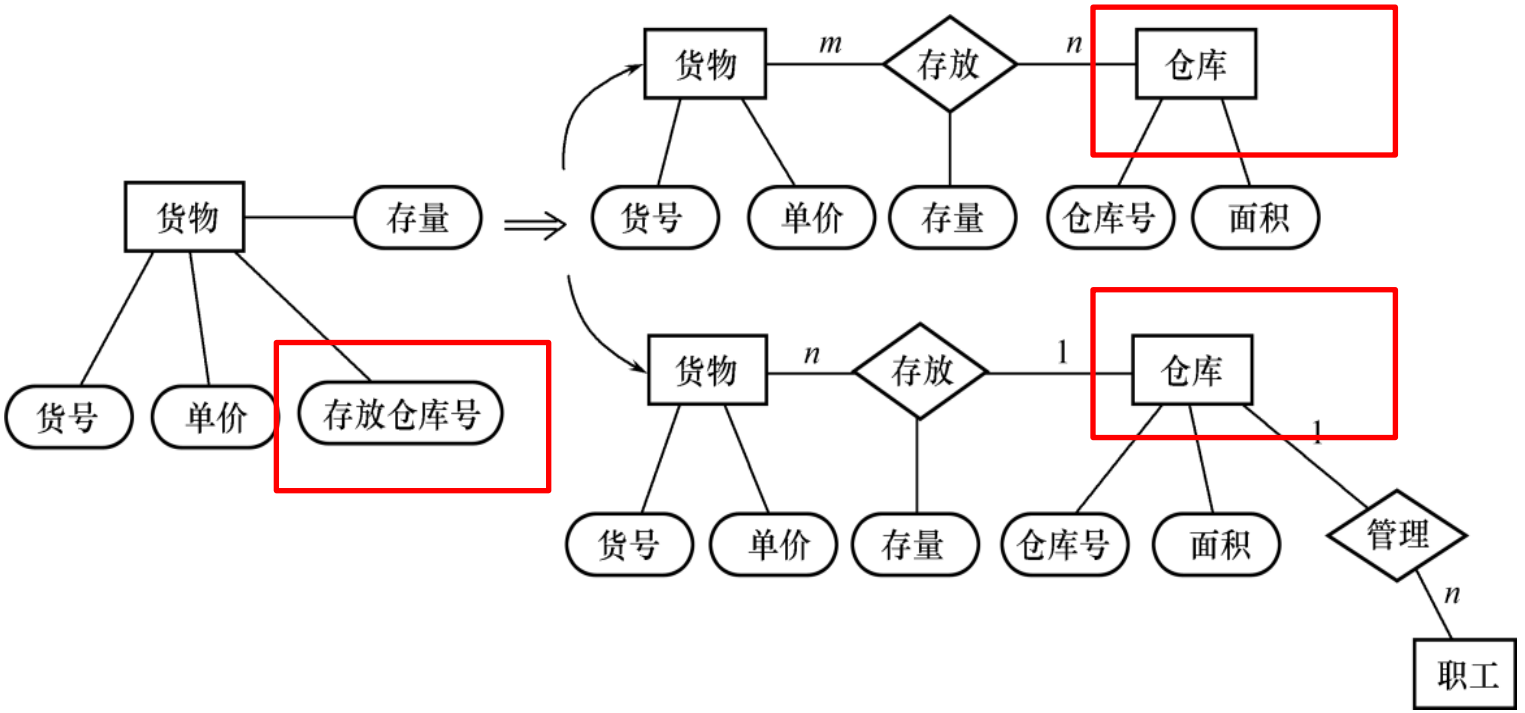
- 一个病人只能住在一个病房，病房号可以作为病人实体的一个属性
- 如果病房还要与医生实体发生联系，即一个医生负责几个病房的病人的医疗工作，则根据准则（2） 病房应作为一个实体



概念结构设计（续）

[例3] 货物实体，如何设计“仓库”？

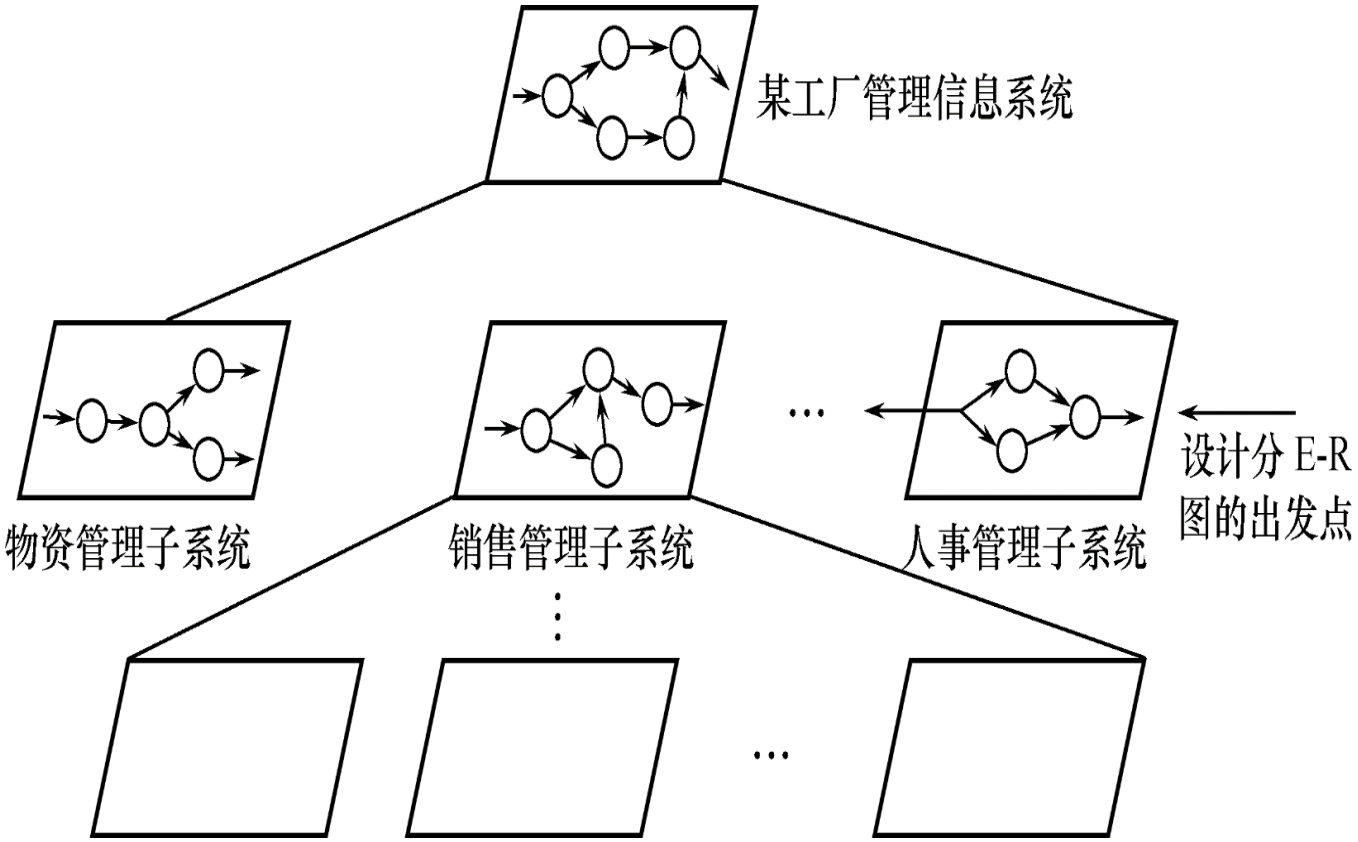
- 如果一种货物只存放在一个仓库，那么就可以把存放货物的仓库的仓库号作为描述货物存放地点的属性
- 如果一种货物可以存放在多个仓库中，或者仓库本身又用面积作为属性，或者仓库与职工发生管理上的联系，那么就应把仓库作为一个实体。



概念结构设计（续）

[例7.1] 工厂管理信息系统：多个子系统

物资管理、销售管理、劳动人事管理等



概念结构设计（续）

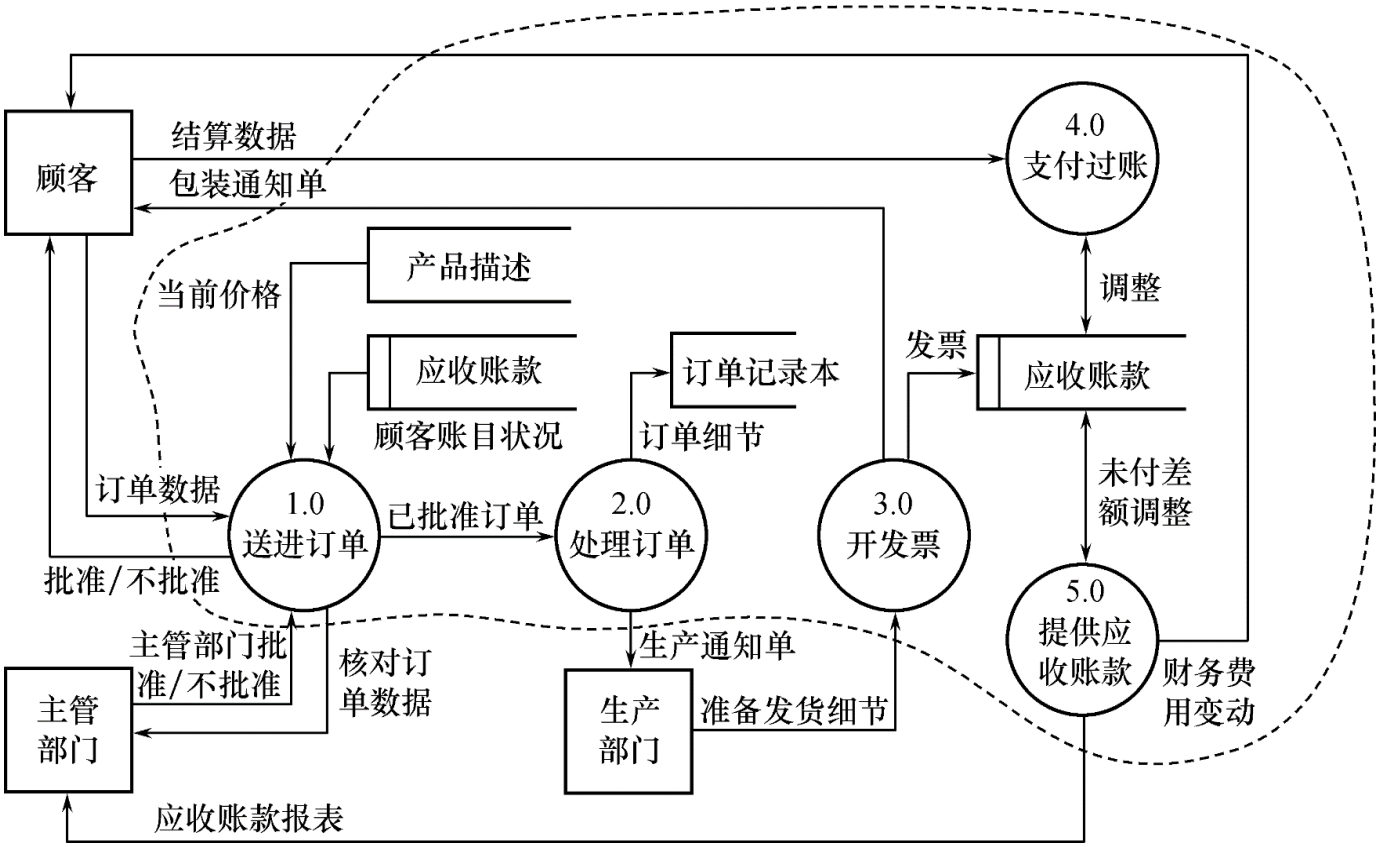
工厂管理信息系统：物资管理、销售管理、劳动人事管理等

[例7.1] 销售管理子系统E-R图的设计（第五版）

- 主要功能：
  - 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

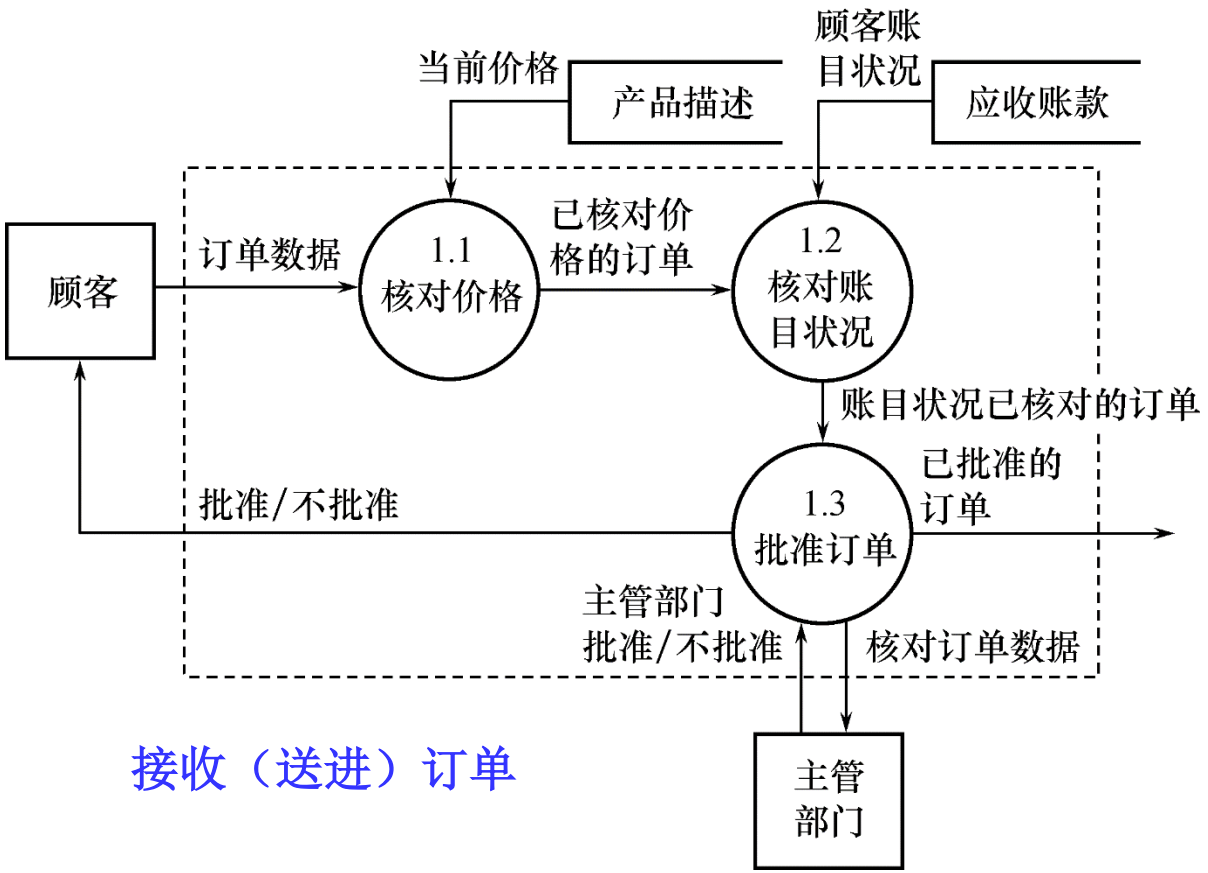
2. 逐一设计分E-R图（续）（补充）

- 销售管理子系统
  - 第一层数据流图，虚线部分划出了系统边界
    - 接收订单、处理订单、开发票、支付过账，提供应收账款



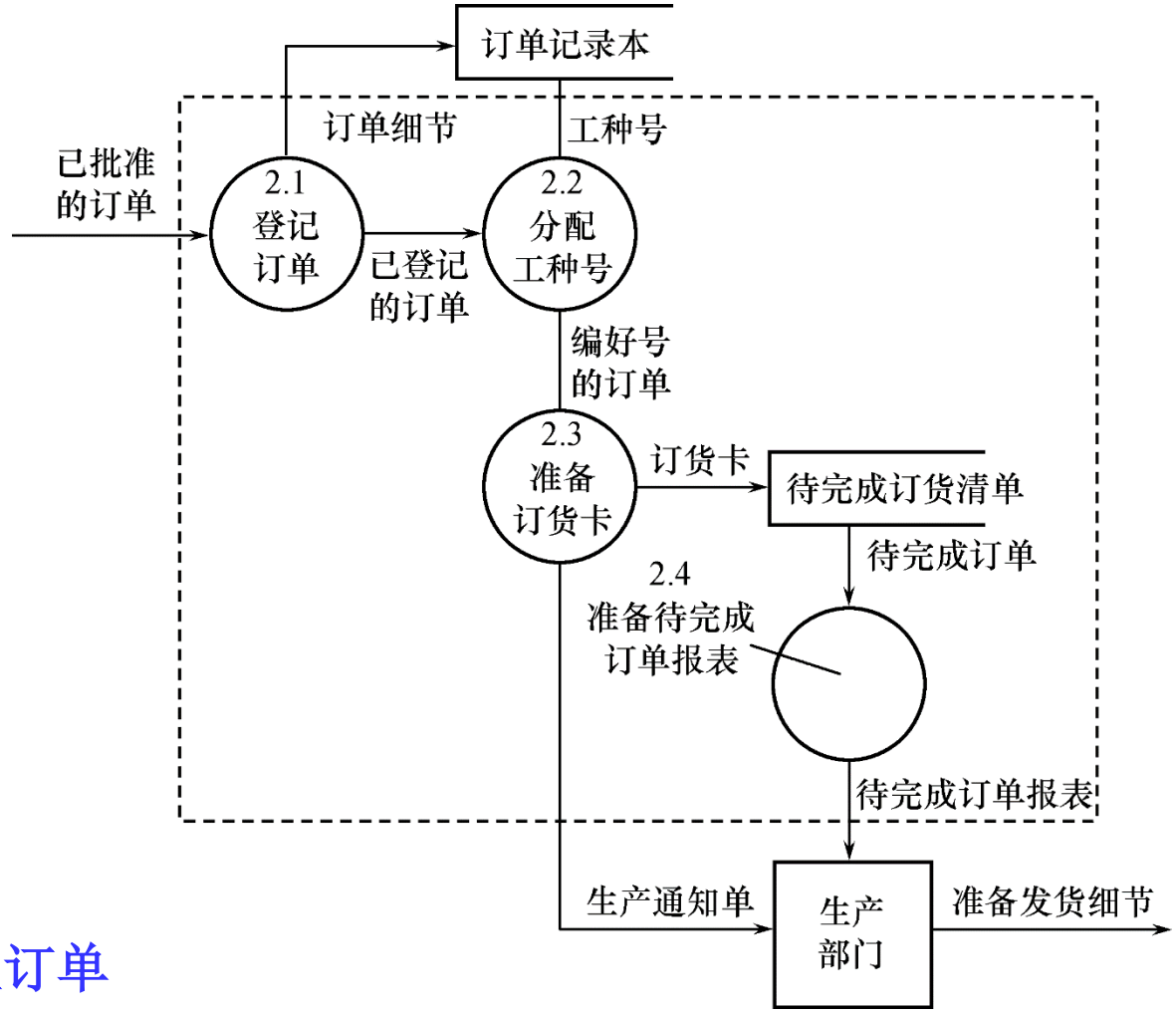
2. 逐一设计分E-R图（续）（补充）

- 上图中把**系统功能又分为4个子系统**
- 下面四个图是第二层数据流图



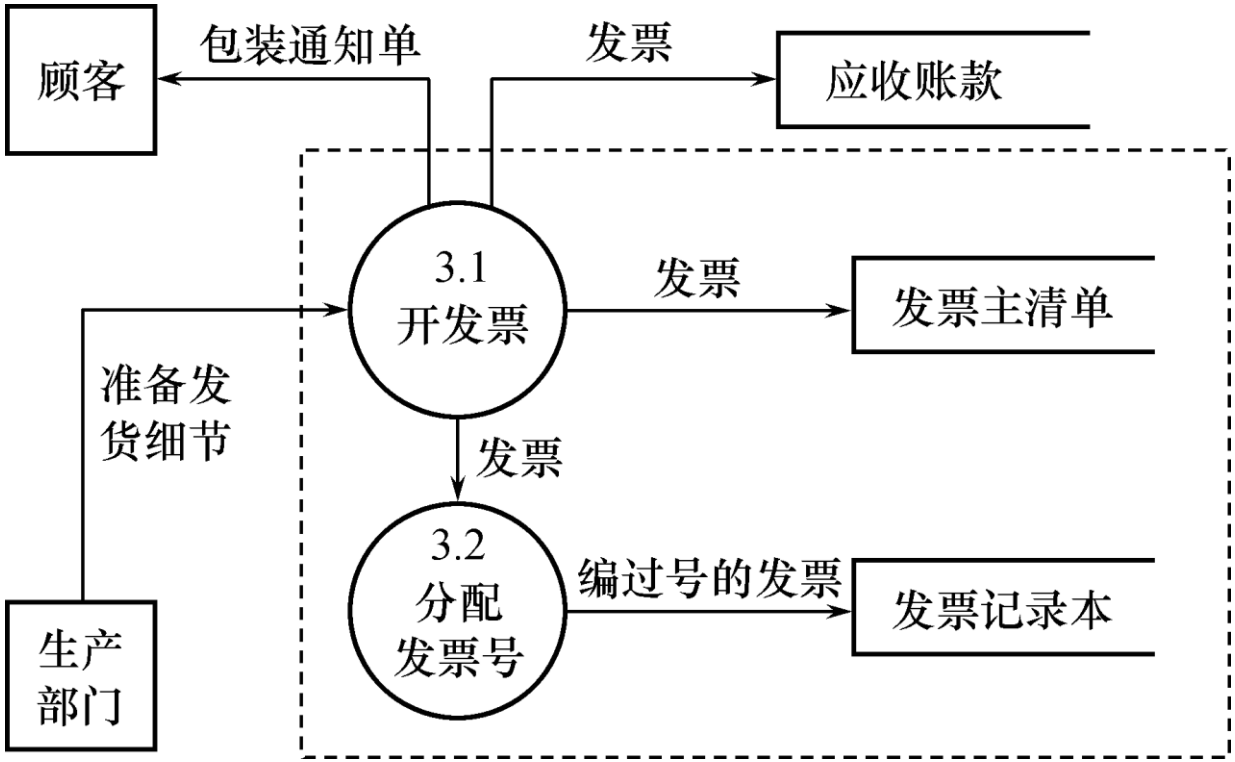
接收（送进）订单

2. 逐一设计分E-R图（续）（补充）



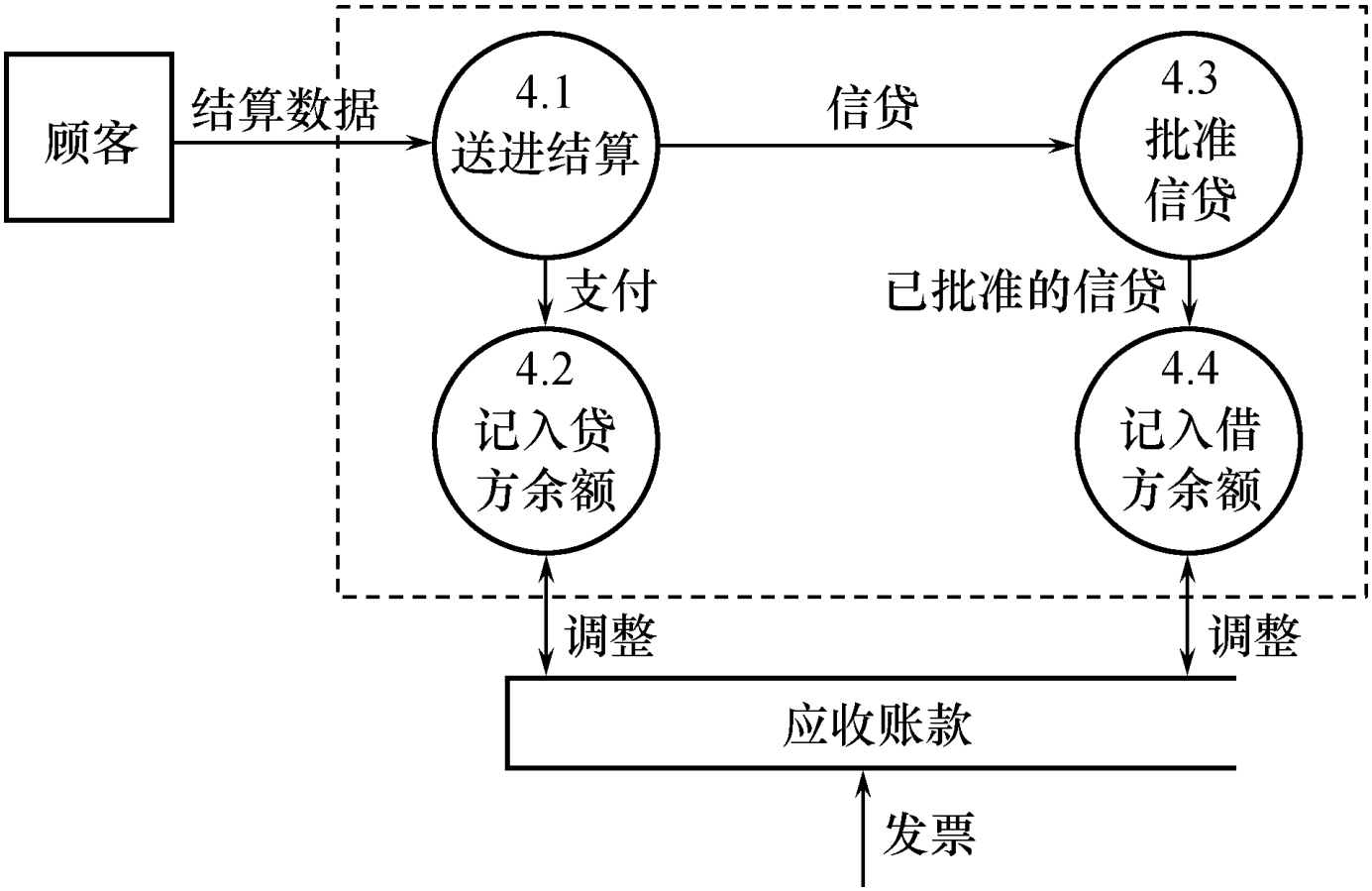
处理订单

2. 逐一设计分E-R图（续）（补充）



开发票

2. 逐一设计分E-R图（续）（补充）



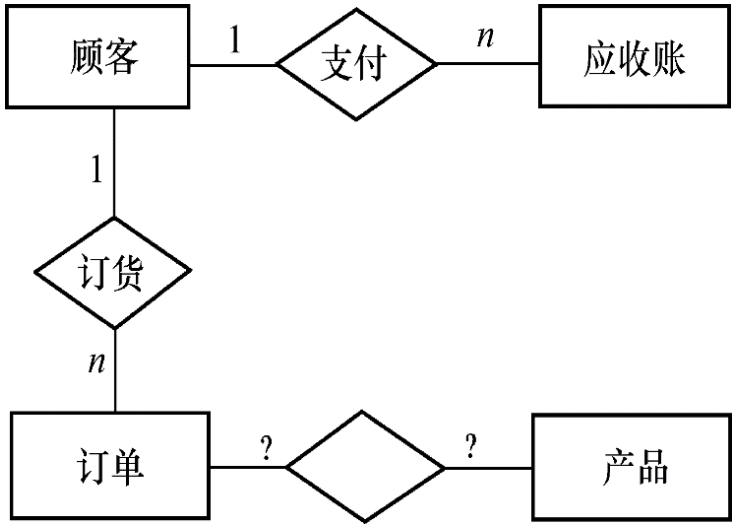
支付过账

概念结构设计（续）

工厂管理信息系统：物资管理、销售管理、劳动人事管理等

[例7.1] 销售管理子系统E-R图的设计

- 1. 画草图7.22
  - 需求分析：接收订单、处理订单、开发票、支付过账
  - 数据结构：订单、顾客，应收账目，产品



概念结构设计（续）

- 2. 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**，E-R图进行了如下**调整**：图7.23
  - （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
  - （2）**原订单和产品的联系实际上是订单细节和产品的联系**。增加联系—参照2：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，应增加实体—“折扣规则”存放这些信息，而不应把它们放在产品实体中

概念结构设计（续）

- 最后得到销售管理子系统E-R图如图7.23所示。

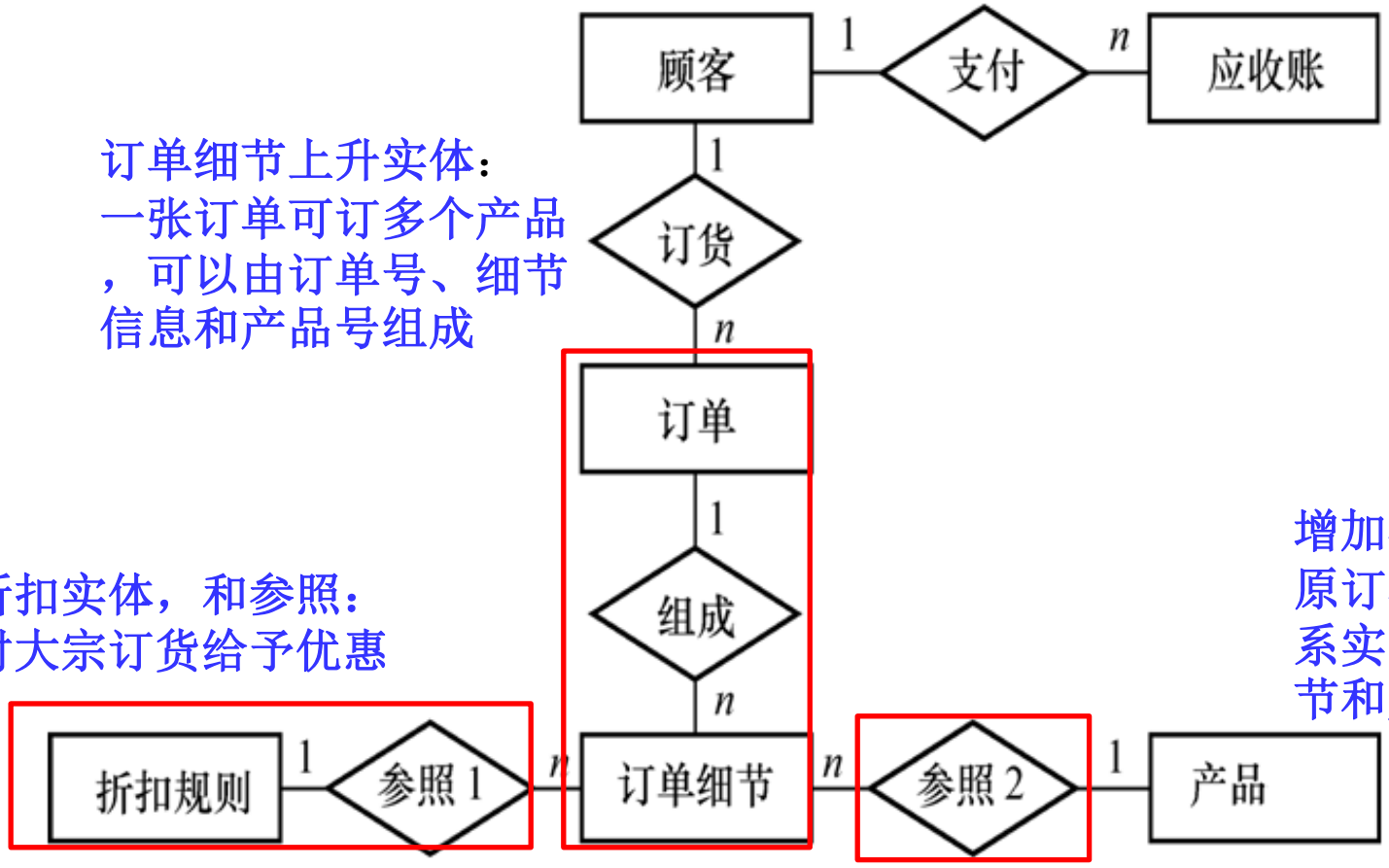


图7.23　销售管理子系统的E-R图

概念结构设计（续）

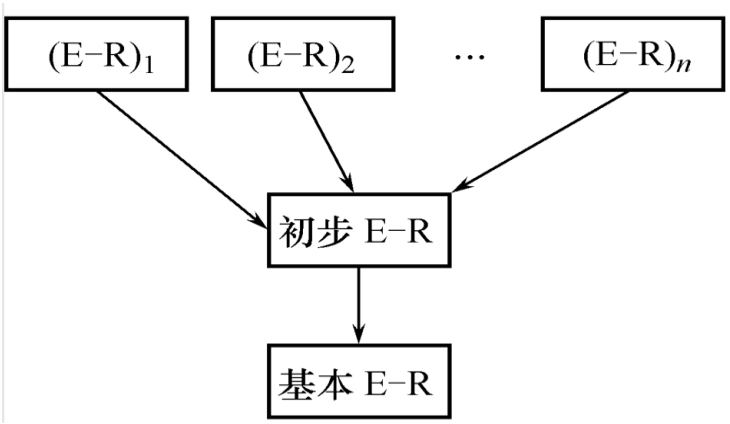
- 对每个实体定义的属性如下：
  - 顾客：{顾客号，顾客名，地址，电话，信贷状况，账目余额}
  - 订单：{订单号，顾客号，订货项数，订货日期，交货日期，工种号，生产地点}
  - 订单细则：{订单号，细则号，零件号，订货数，金额}
  - 应收账款：{顾客号，订单号，发票号，应收金额，支付日期，支付金额，当前余额，货款限额}
  - 产品：{产品号，产品名，单价，重量}
  - 折扣规则：{产品号，订货量，折扣}

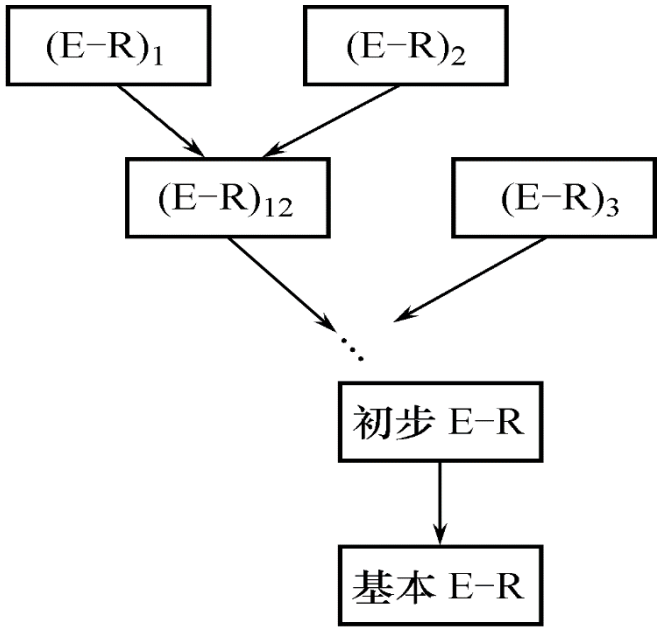
课后画实体-属性图？

概念结构设计（续）

2. E-R图的集成

- 一次集成：多个分E-R图一次集成 （局部视图简单时）
- 逐步集成：用累加的方式一次集成两个分E-R图

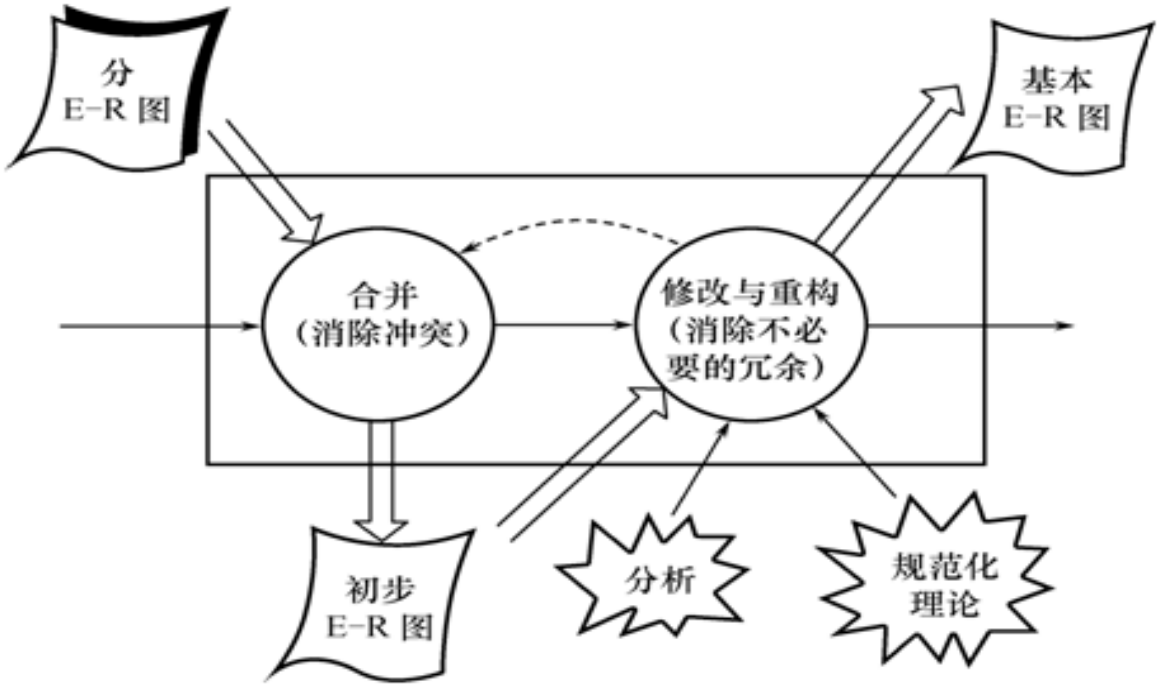


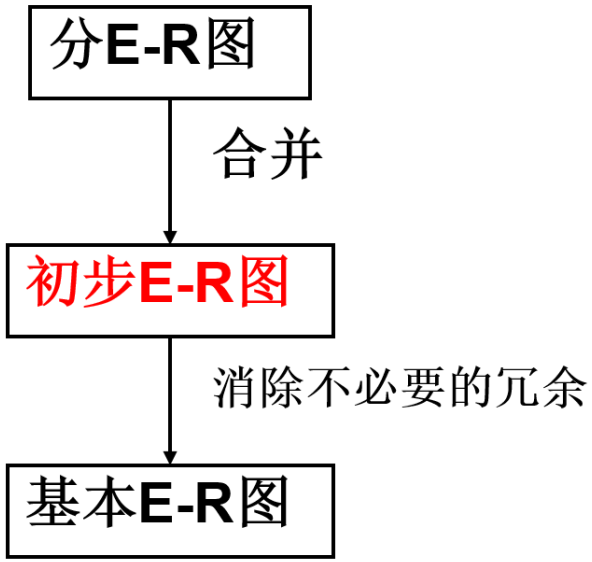


概念结构设计（续）

2. E-R图的集成

- E-R图的集成一般需要分两步
  - （1）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
  - （2）修改和重构。消除不必要的冗余，生成基本E-R图。





概念结构设计（续）

2. E-R图的集成

- （1）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。
  - 合并E-R图：消除E-R图中的不一致，全系统可理解
  - 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

概念结构设计（续）

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
  - 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。

概念结构设计（续）

② 命名冲突

- 同名异义：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）：同一意义的对象在不同的局部应用中具有不同的名字
  - 例如，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

概念结构设计（续）

③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。（变换遵循7.3.5的两个准则）
- 2. 同一实体在不同子系统的E-R图中所包含的属性个数和属性排列次序不完全相同
  - 常见：原因是不同的应用关心实体的不同侧面
  - 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

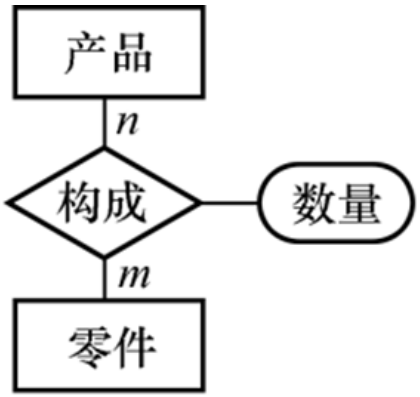
概念结构设计（续）

③结构冲突（续）

- 3. 实体间的联系在不同的E-R图中为不同的类型
  - 例如，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

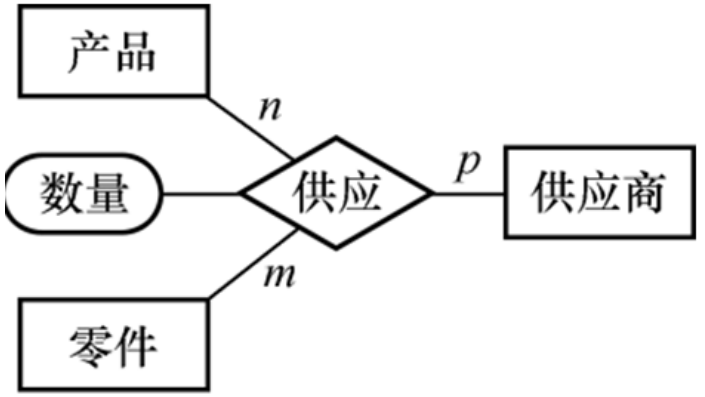
概念结构设计（续）

图7.25(a)中零件与产品之间存在多对多的联系“构成”



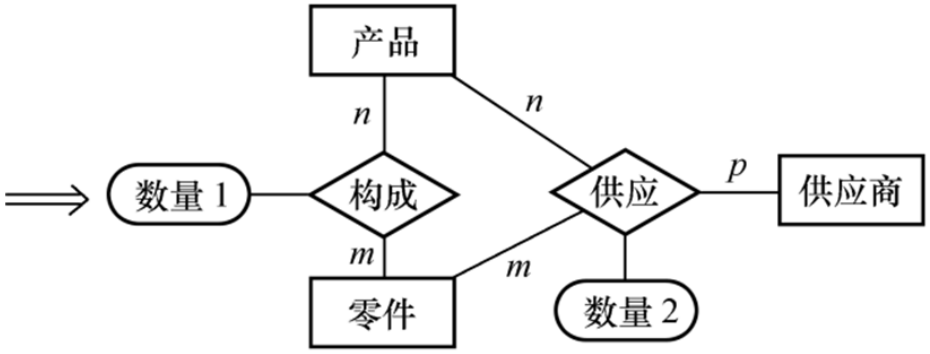
(E-R)1

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”



(E-R)2

图7.25(c)，合并两个E-R图



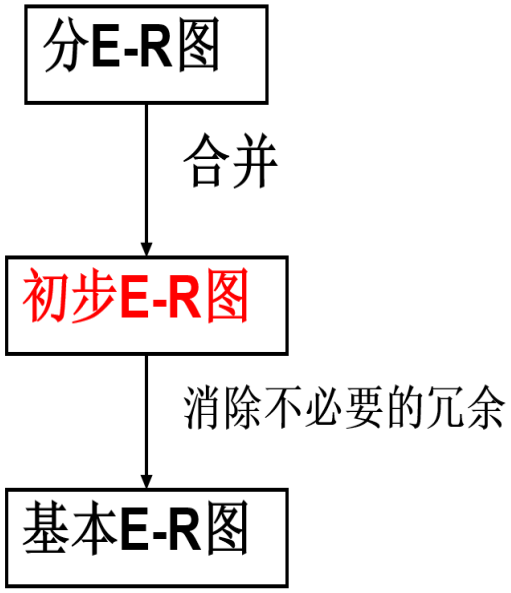
(E-R)12

概念结构设计（续）

E-R图的集成

在(1)初步E-R图中，仍存在一些冗余的数据和实体间冗余的联系

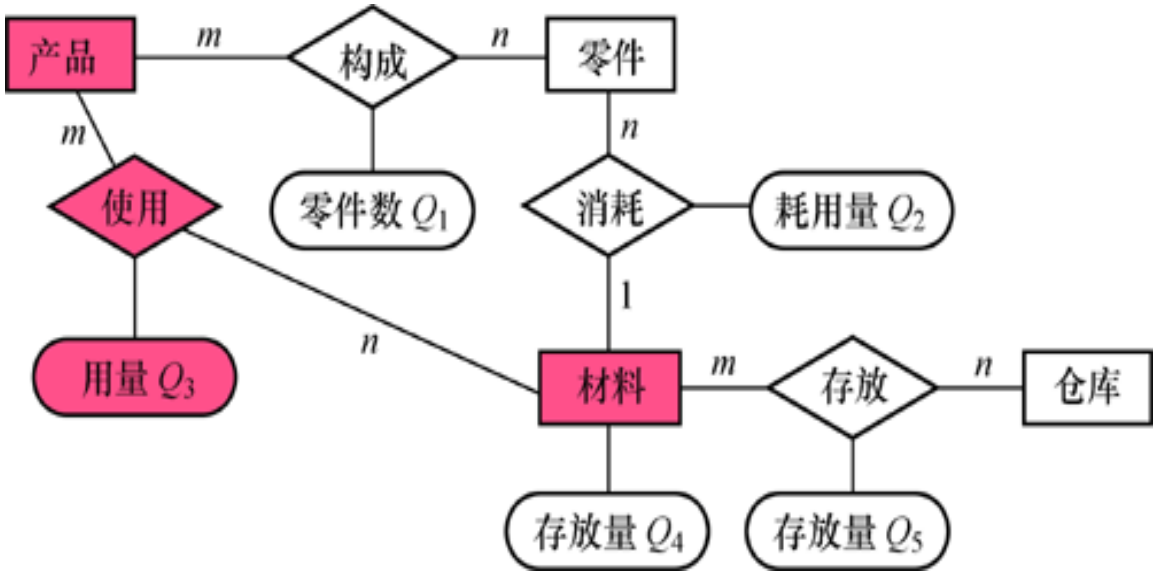
- （2）消除不必要的冗余，设计基本E-R图
  - 破坏完整性，给数据库维护增加困难
    - 冗余的数据：可由基本数据导出的数据
    - 冗余的联系：可由其他联系导出的联系
  - 消除冗余方法
    - 主要采用分析方法
    - 规范化理论



概念结构设计（续）

消除冗余方法：采用分析方法

- 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



$Q_3=Q_1\times Q_2$，$Q_4=\sum Q_5$。所以Q_3和Q_4是冗余数据，可以消去。

并且由于Q_3消去，产品与材料间$m:n$的冗余联系也应消去

图7.26 消除冗余（Q3=Q1*Q2是冗余，Q4是Q5的合也冗余）

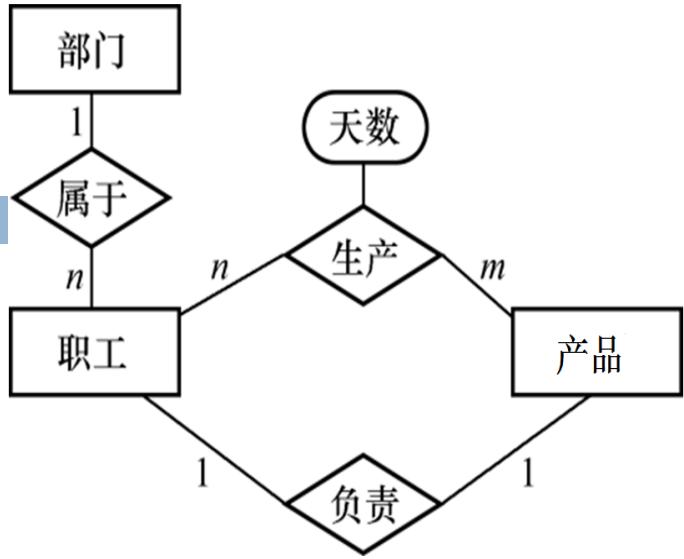
注意：并不是所有的冗余数据与冗余联系都必须加以消除，有时为了提高效率，不得不以冗余信息作为代价。

消除冗余的方法（续）

- 效率VS冗余信息
  - 需要根据用户的整体需求来确定
- 若人为地保留了一些冗余数据，则应把数据字典中数据关联的说明作为完整性约束条件
  - $Q4=\sum Q5$
  - 一旦Q5修改后就应当触发完整性检查，对Q4进行修改

此处如何实现？

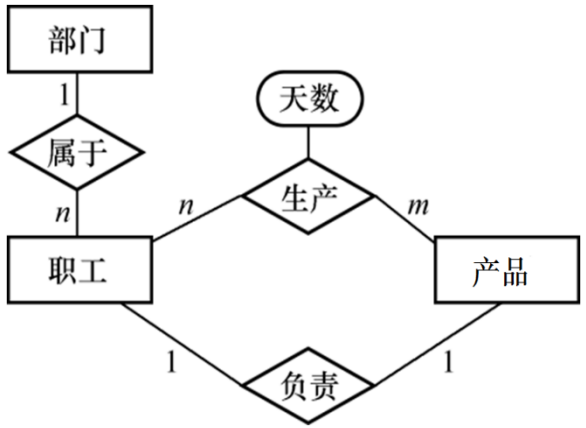
概念结构设计（续）



- 消除冗余的方法：
  - 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：有函数依赖集F_L
      - 部门和职工之间一对多的联系可表示为：职工号→部门号
      - 职工和产品之间多对多的联系可表示为：（职工号，产品号）→工作天数等。

概念结构设计（续）

- ②求FL的最小覆盖GL，差集为 D=FL-GL。
  - 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
- 规范化理论受到泛关系假设的限制，应注意两个问题
  - 冗余的联系一定在D中，而D中的联系不一定是冗余
  - 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
    - 图7.27中，职工和产品之间存在另一个1对1联系表示为
      - 负责人.职工号→产品号
      - 产品号→负责人.职工号



概念结构设计（续）

- [例7.2] 某工厂管理信息系统的视图集成。

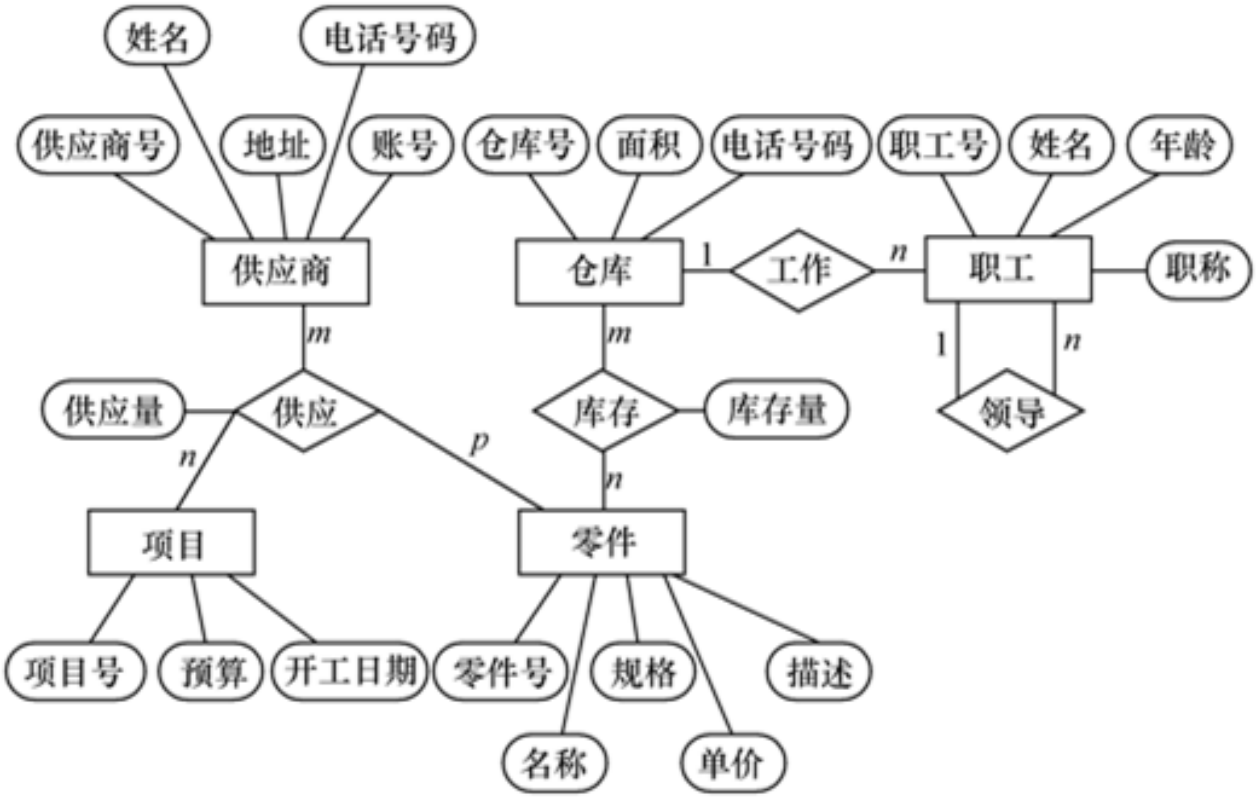


图7.11 子系统：工厂物资管理E-R图

概念结构设计（续）

- ［例7.2］某工厂管理信息系统的视图集成

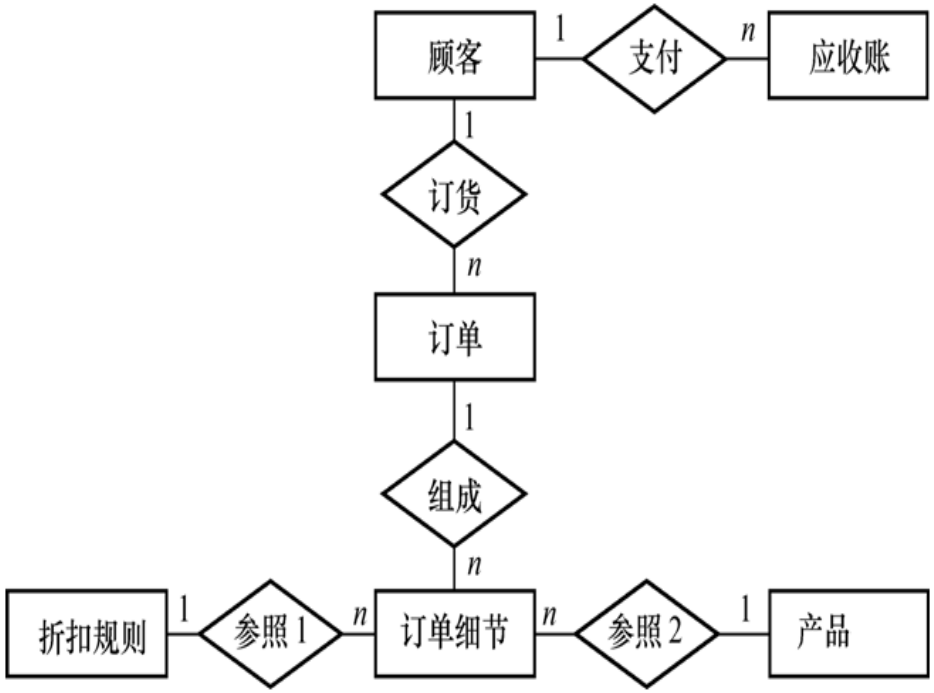


图7.23 销售管理子系统的E-R图

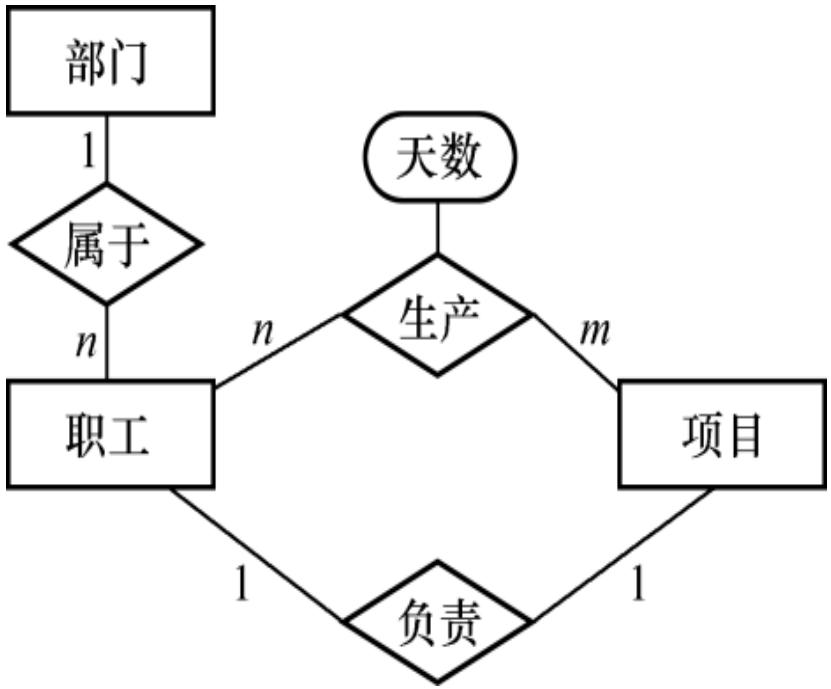
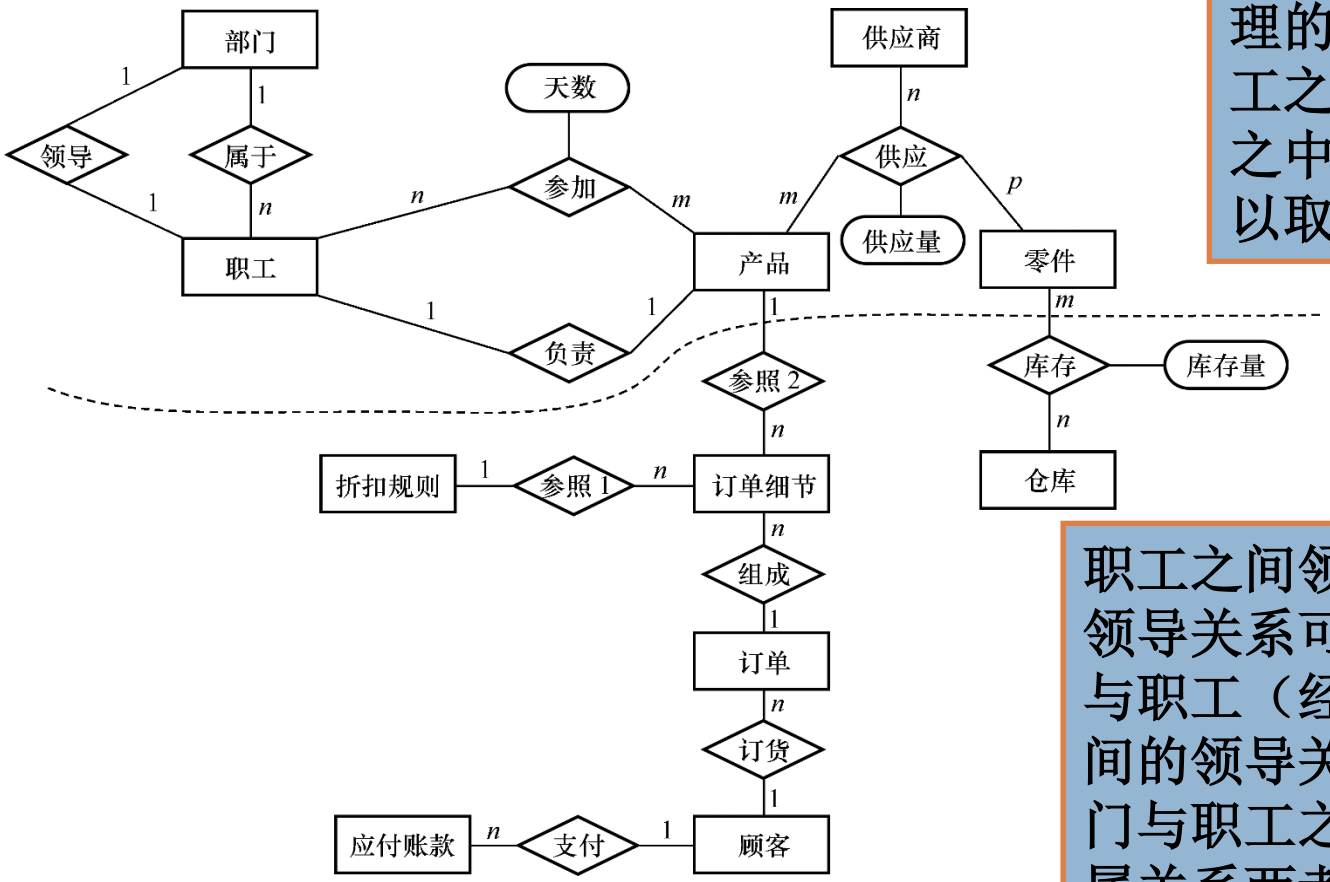


图7.27 劳动人事管理的分E-R图

概念结构设计（续）

- [例7.2] 某工厂管理信息系统的视图集成。



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。

图7.28 某工厂管理信息系统的基本E

消除冗余，设计生成基本E-R图实例（续）

集成过程，解决了以下问题：

- 异名同义，项目和产品含义相同（统一为产品）
- 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

验证整体概念结构

- 视图集成后形成一个整体的数据库概念结构，对该整体概念结构还必须**进行进一步验证**，确保它能够满足下列条件：
  - 整体**概念结构内部必须具有一致性**，不存在互相矛盾的表达
  - 整体**概念结构能准确地反映原来的每个视图结构**，包括属性、实体及实体间的联系
  - 整体**概念结构能满足需要分析阶段所确定的所有要求**

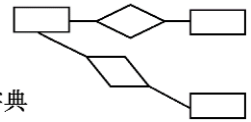
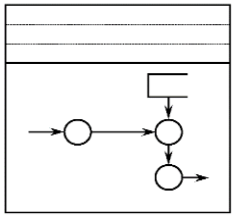
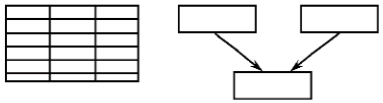
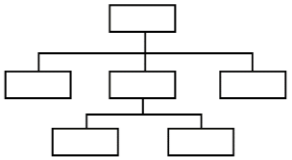
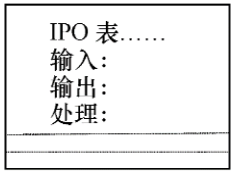
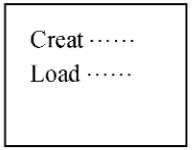
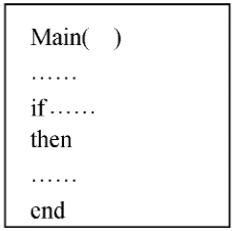
- 整体概念结构最终还应该**提交给用户**，征求用户和有关人员的意见，进行评审、修改和优化，然后把它确定下来，作为数据库的概念结构，作为进一步设计数据库的依据。

第七章 数据库设计

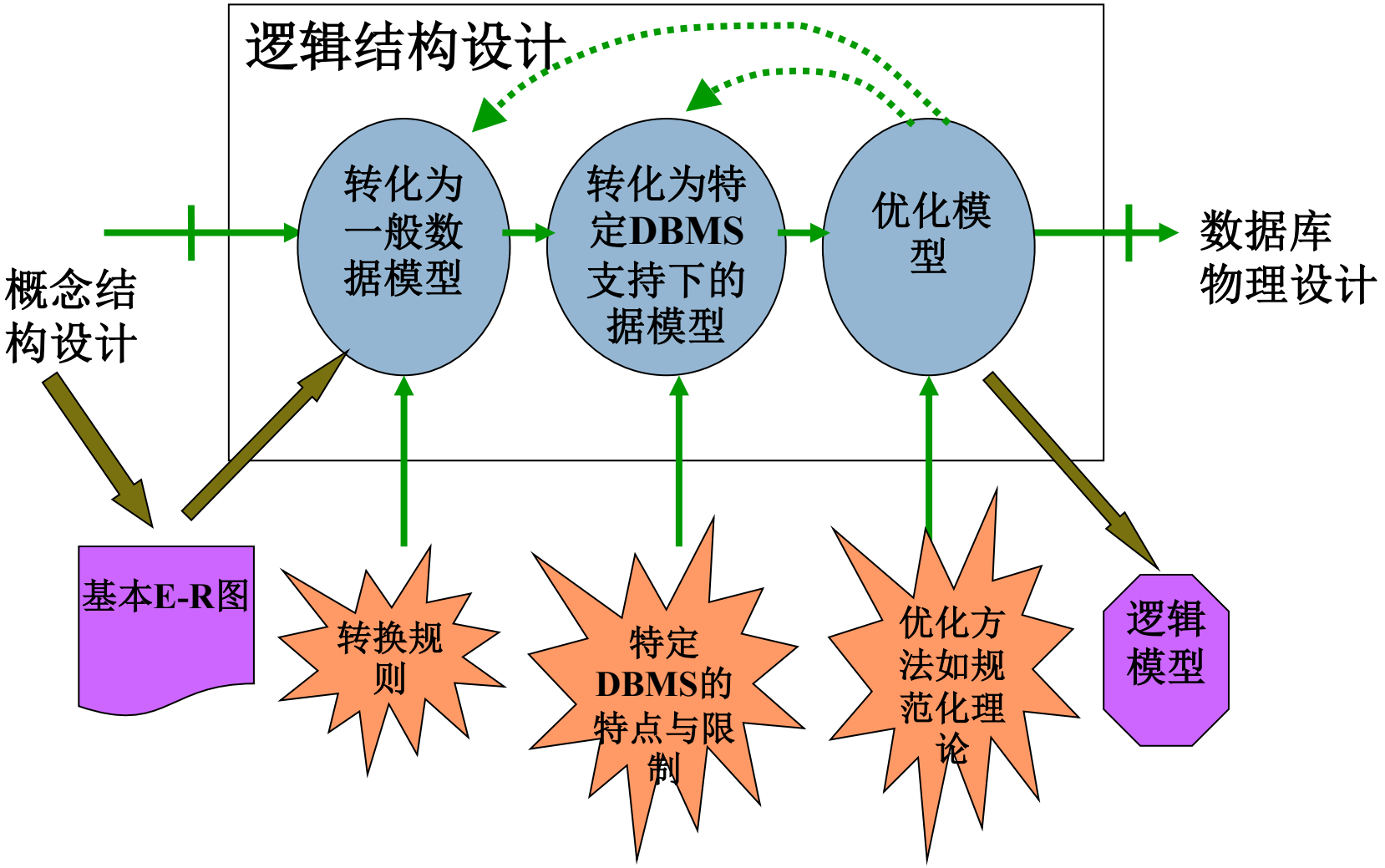
7.4 逻辑结构设计

- 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DBMS产品所支持的数据模型**相符合的逻辑结构
  - 关系模型：一组关系模式
- 逻辑结构设计的步骤
  - 将概念结构转化为一般的关系、网状、层次模型
  - 将转换来的关系、网状、层次模型向特定DBMS支持下的数据模型转换
  - 对数据模型进行优化

数据库描述

设计阶段	设计描述	
	数据	处理
需求分析	数据字典、全系统中数据项、数据流、数据存储的描述	数据流图和判定表（判定树）、数据字典中处理过程的描述
概念结构设计	概念模型（E-R图）  数据字典	系统说明书包括： ① 新系统要求、方案和概图 ② 反映新系统信息流的数据流图 
逻辑结构设计	某种数据模型 关系 非关系 	系统结构图 （模块结构） 
物理设计	存储安排 方法选择 存取路径建立 	模块设计 IPO 表 
数据库实施阶段	编写模式 装入数据 数据库试运行 	程序编码、编译联结、测试 
数据库运行和维护	性能监测、转储 /恢复 数据库重组和重构	新旧系统转换、运行、维护（修正性、适应性、改善性维护）

逻辑结构设计(续)


逻辑结构设计
转化为一般数据模型
转化为特定DBMS支持下的据模型
优化模型
概念结构设计
数据库物理设计
基本E-R图
转换规则
特定DBMS的特点与限制
优化方法如规范化理论
逻辑模型

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.1 E-R图向关系模型的转换

- E-R图向关系模型的转换要解决的问题
  - 如何将实体型和实体间的联系转换为关系模式
  - 如何确定这些关系模式的属性和码

E-R图向关系模型的转换（续）

- 转换内容
- 转换原则

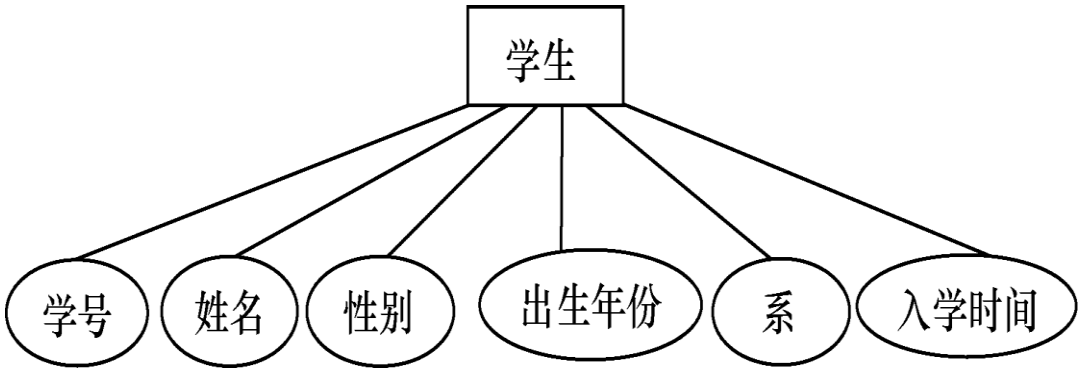
转换内容

- 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、实体的属性和实体之间的联系转换为关系模式。

转换原则

- 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



学生（学号，姓名，性别，出生年份，系别，入学时间）

转换原则（续）

2. 实体型间的联系有以下不同情况：

(1) 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（续）

[例] “管理”联系为1：1的联系

（1）转换为一个独立的关系模式

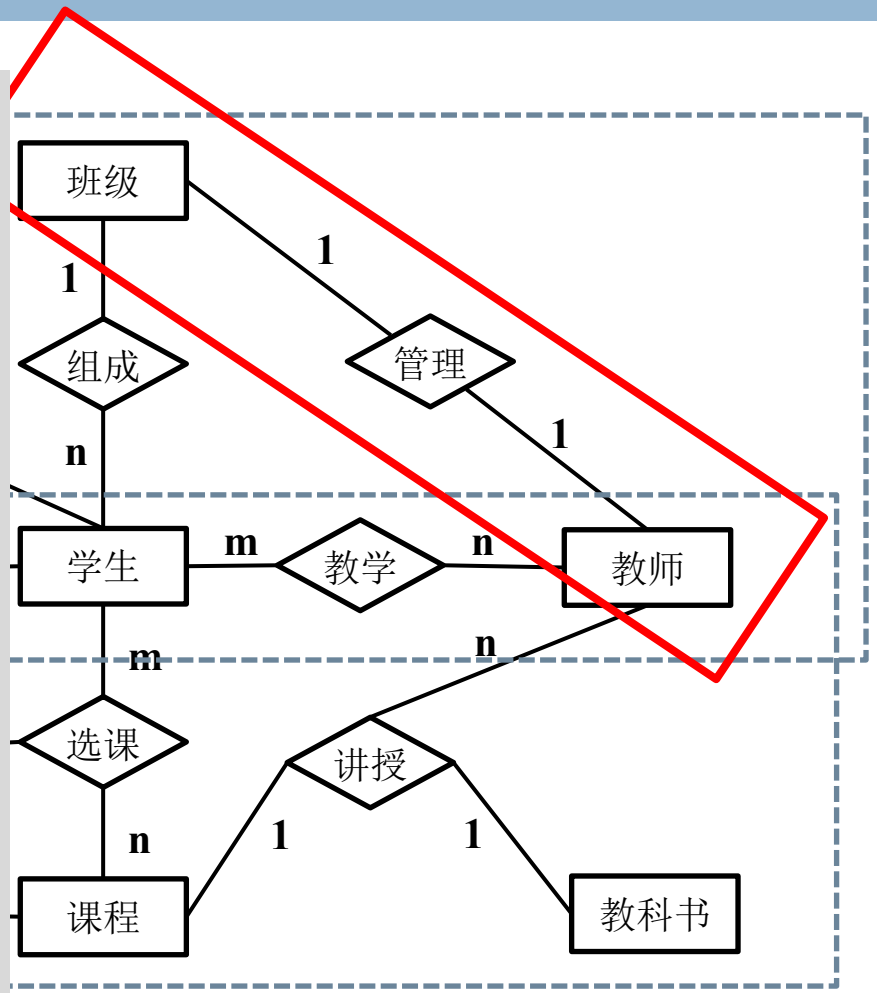
- 管理（职工号，班级号）
- 管理（职工号，班级号）

（2）“管理”与“班级”关系模式合并

- 班级（班级号，学生人数，职工号）
- 在“班级”中加入教师的码，即职工号

（3）“管理”与“教师”关系模式合并

- 教师（职工号，姓名，性别，职务，班级号，是否为优秀班主任）
- 在“教师”中加入“班级”的码，即班级号



学生管理子系统E-R图

转换原则（续）

2. 实体型间的联系有以下不同情况：

(2) 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，一般情况下更倾向于采用这种方法

转换原则（续）

[例] “组成”联系为1：n的联系

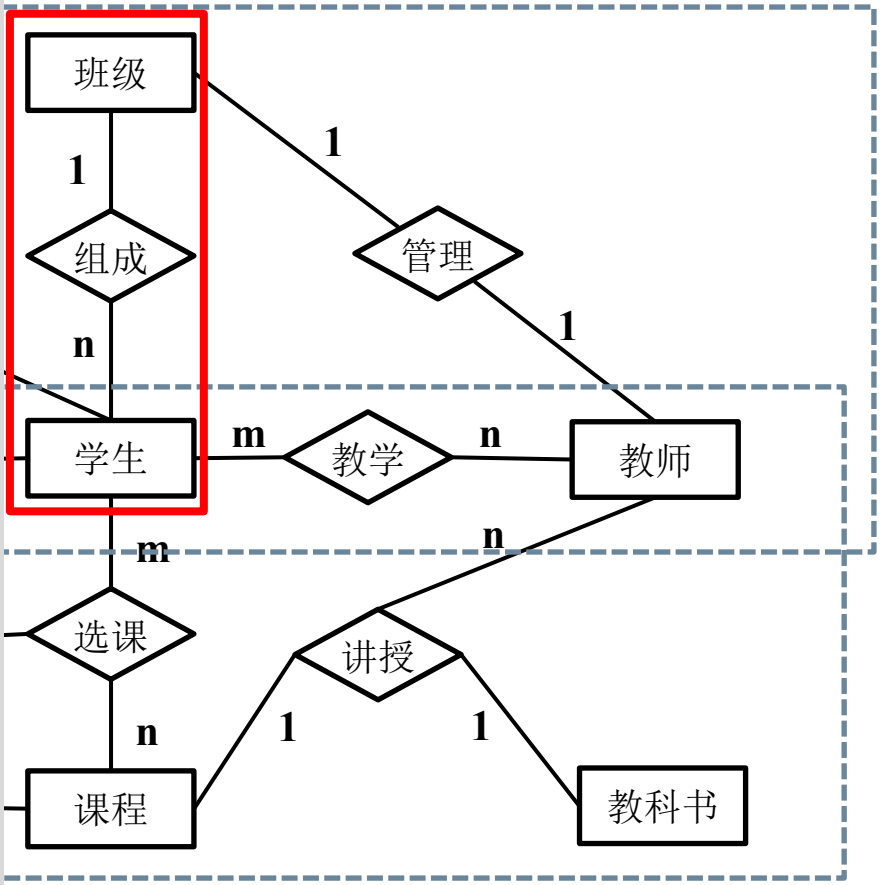
将其转换为关系模式的两种方法

（1）使其成为一个独立的关系模式：

- 组成（学号，班级号）

（2）将其与“学生”合并

- 学生（学号，姓名，性别，出生日期，所在系，年级，班级号，平均成绩）



学生管理子系统E-R图

转换原则（续）

2. 实体型间的联系有以下不同情况：

(3) 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

- 不能合并：部分函数依赖

[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（续）

2. 实体型间的联系有以下不同情况：

(3) 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：

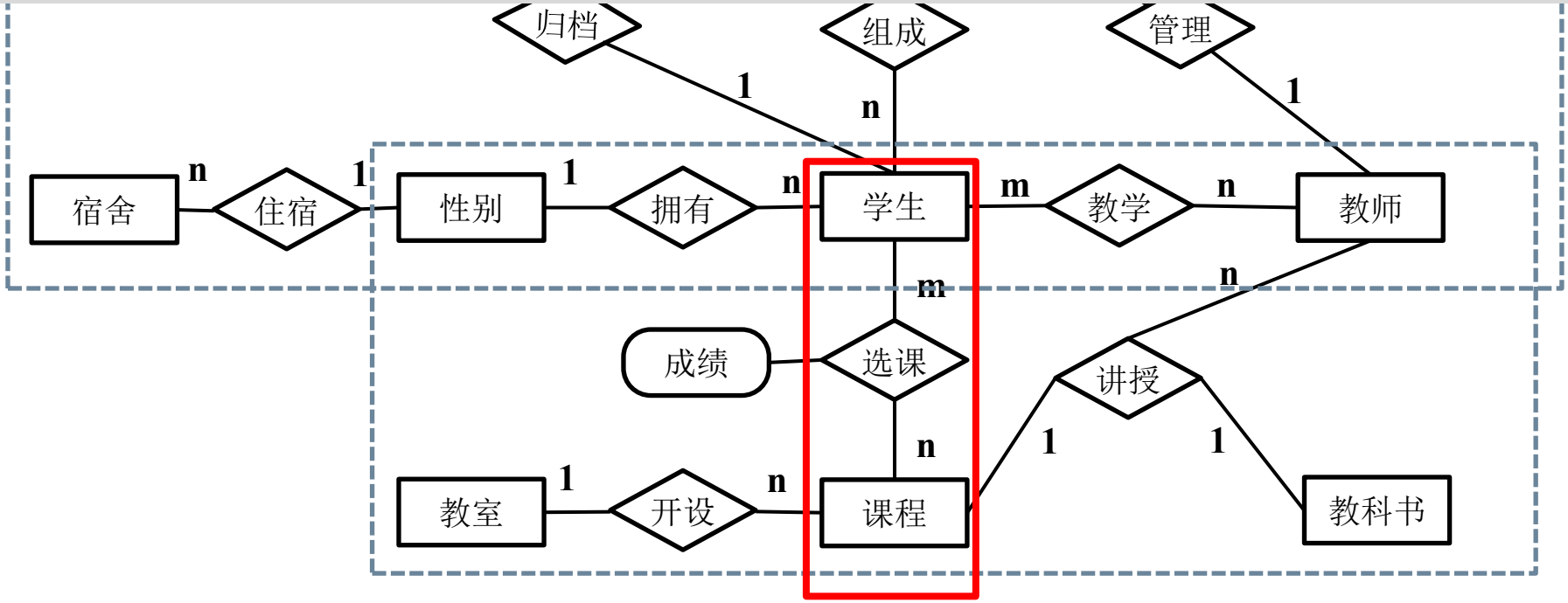
选修（学号，课程号，成绩）

转换原则（续）

[例] “选修”联系为m：n的联系,可以将它转换为

选修（学号，课程号，成绩），

其中，学号和课程号为关系的组合码



学生管理子系统E-R图

转换原则（续）

2. 实体型间的联系有以下不同情况：

(4) 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

[例]“讲授”联系是一个三元联系，可以将它转换为如下关系模式，其中课程号、职工号和书号为关系的组合码：

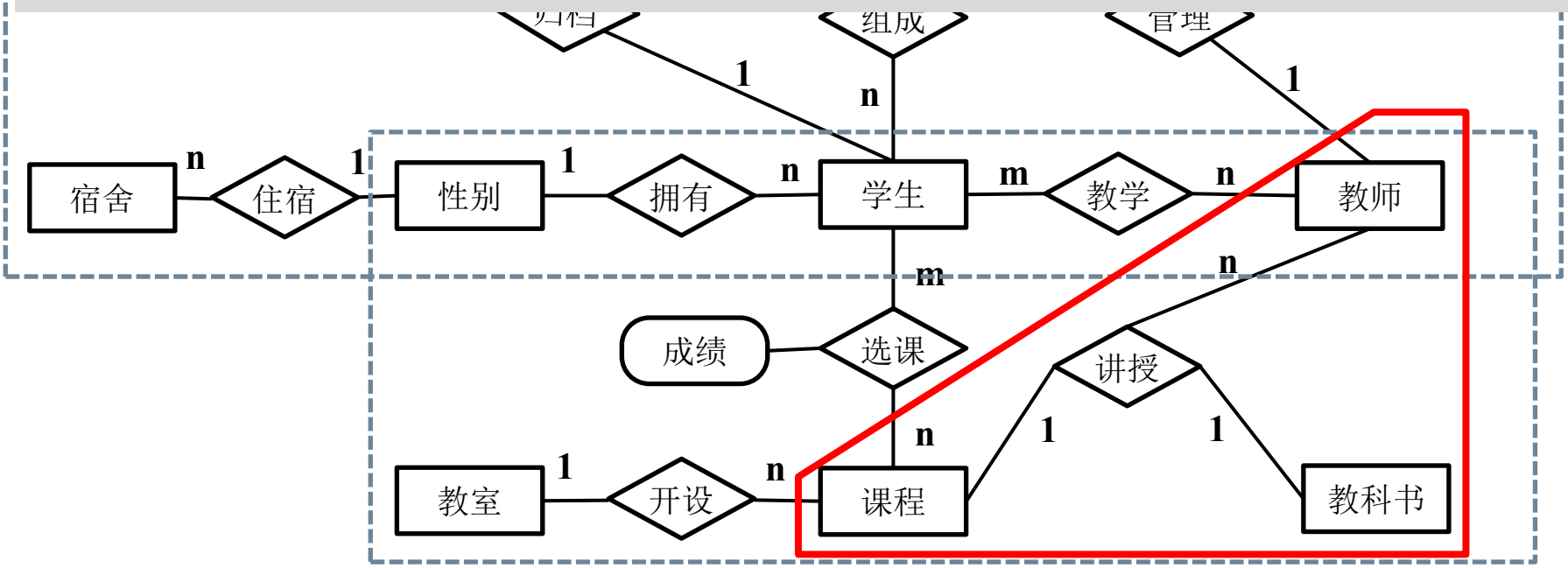
讲授（课程号，职工号，书号）

转换原则（续）

[例] “讲授”联系是一个三元联系，可以将它转换为

讲授（课程号，职工号，书号）

其中，课程号、职工号和书号为关系的组合码



学生管理子系统E-R图

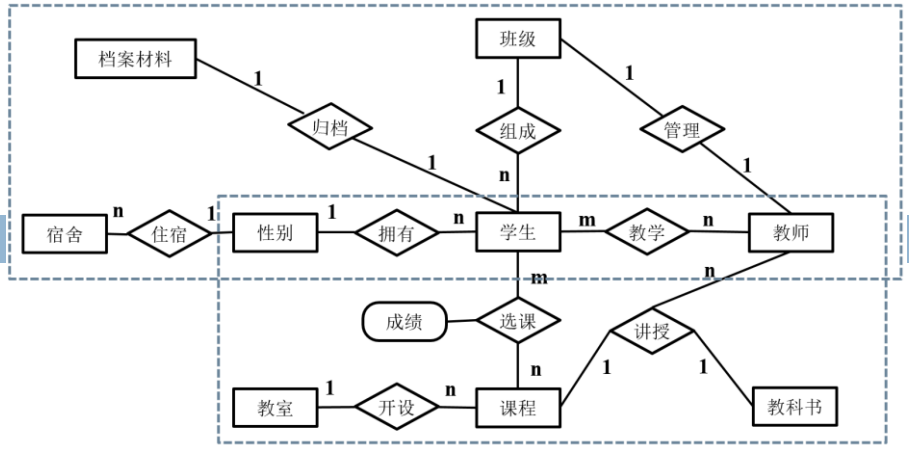
转换原则（续）

2. 实体型间的联系有以下不同情况：

(5) 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

转换原则（续）

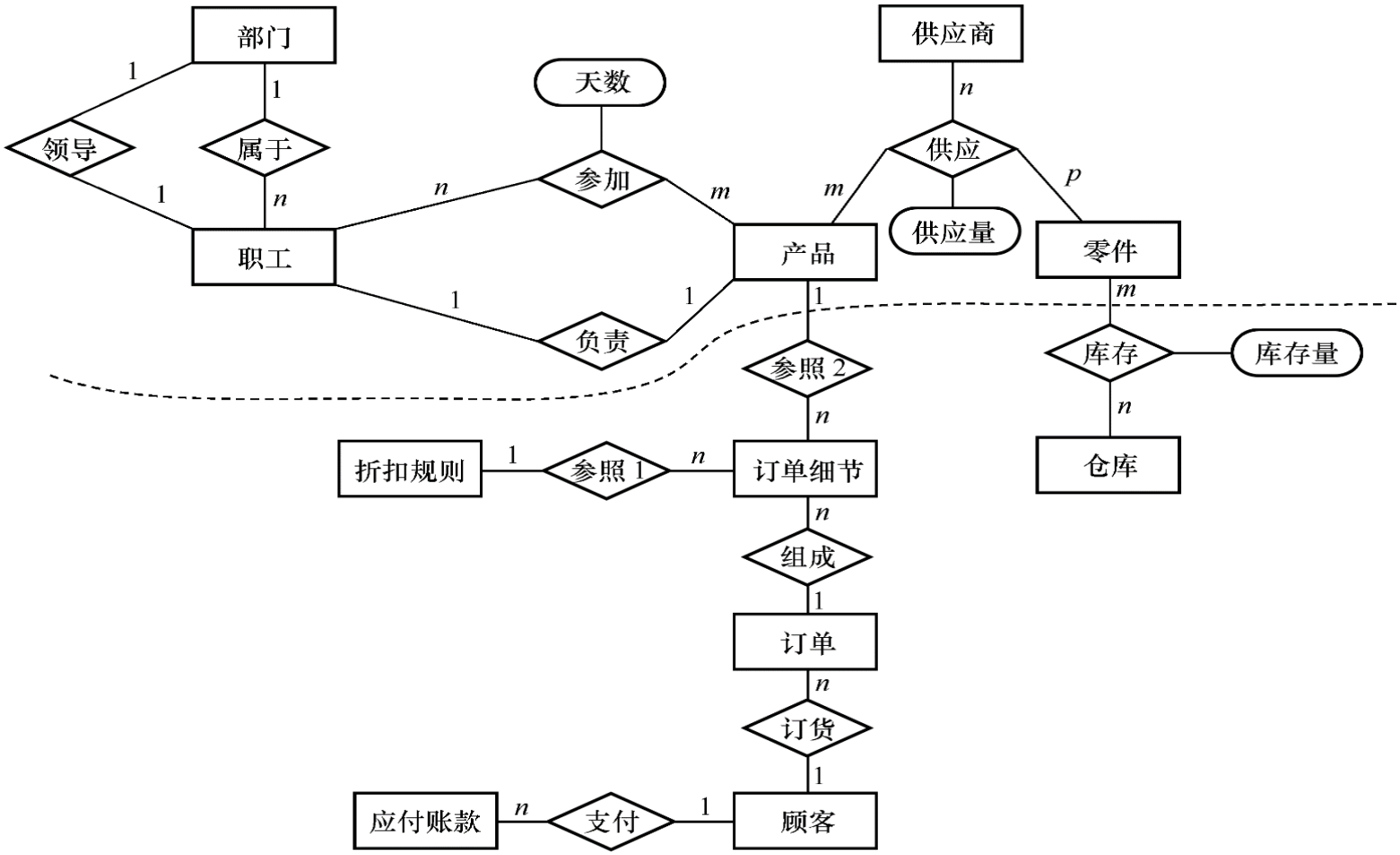


按照上述转换原则，学生管理子系统中的实体和联系转换为下列关系模型

- **实体与联系**
  - 学生（学号，姓名，性别，出生日期，所在系，年级，班级号，平均成绩，档案号）
  - 性别（性别，宿舍楼）
  - 宿舍（宿舍编号，地址，性别，人数）
  - 班级（班级号，学生人数）
  - 教师（职工号，姓名，性别，职称，班级号，是否优秀班主任）
  - 教学（职工号，学号）
  - 课程（课程号，课程名，学分，教室号）
  - 选修（学号，课程号，成绩）
  - 教科书（书号，书名，价格）
  - 教室（教室编号，地址，容量）
  - 讲授（课程号，教师号，书号）
  - 档案材料（档案号，…）
- 该关系模型由12个关系模式组成，其中：
  - 学生关系模式包含了“拥有”、“组成”和“归档”联系所对应的关系模式
  - 教师关系模式包含了“管理”联系所对应的关系模式
  - 宿舍关系模式包含了“住宿”联系所对应的关系模式
  - 课程关系模式包含了“开设”联系所对应的关系模式

E-R图向关系模型的转换（续）

[例] 把图7.28中虚线上部的E-R图转换为关系模型



E-R图向关系模型的转换（续）

[例] 把图7.28中虚线上部的E-R图转换为关系模型

- **部门实体的关系模式**

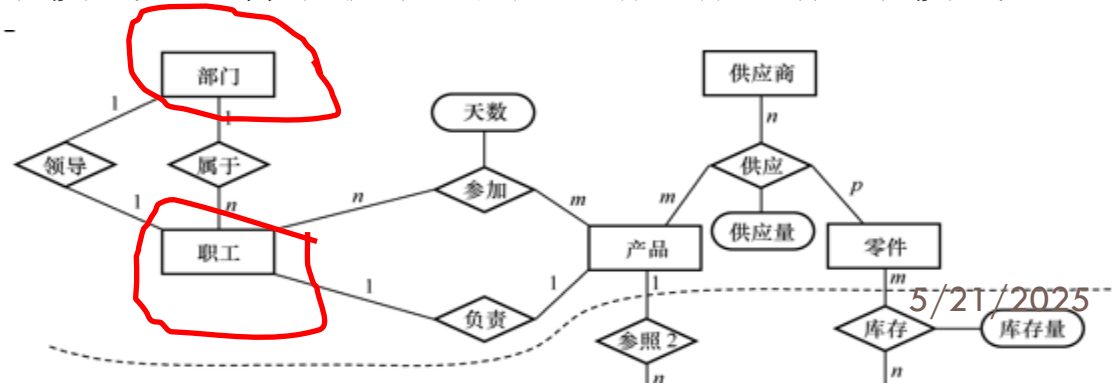
  部门（部门号，部门名，经理的职工号，…）

  - 此关系模式已包含了联系“领导”所对应的关系模式
  - 经理的职工号是关系的候选码

- **职工实体的关系模式**

  职工（职工号、部门号，职工名，职务，…）

  - 该关系模式已包含了联系“属于”所对应的关系模式



E-R图向关系模型的转换（续）

[例] 把图7.28中虚线上部的E-R图转换为关系模型

- 产品实体的关系模式

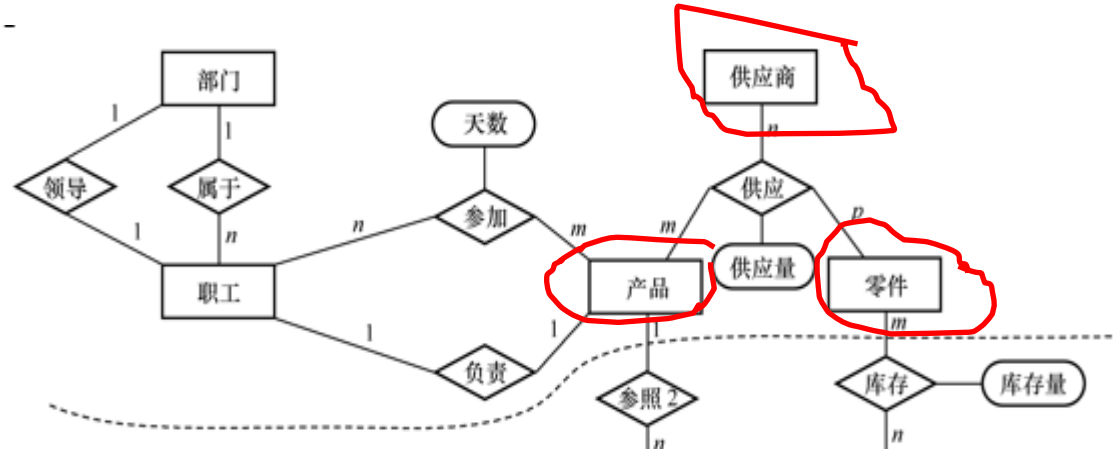
  产品（产品号，产品名，产品组长的职工号，…）

- 供应商实体的关系模式

  供应商（供应商号，姓名，…）

- 零件实体的关系模式

  零件（零件号，零件名，…）



E-R图向关系模型的转换（续）

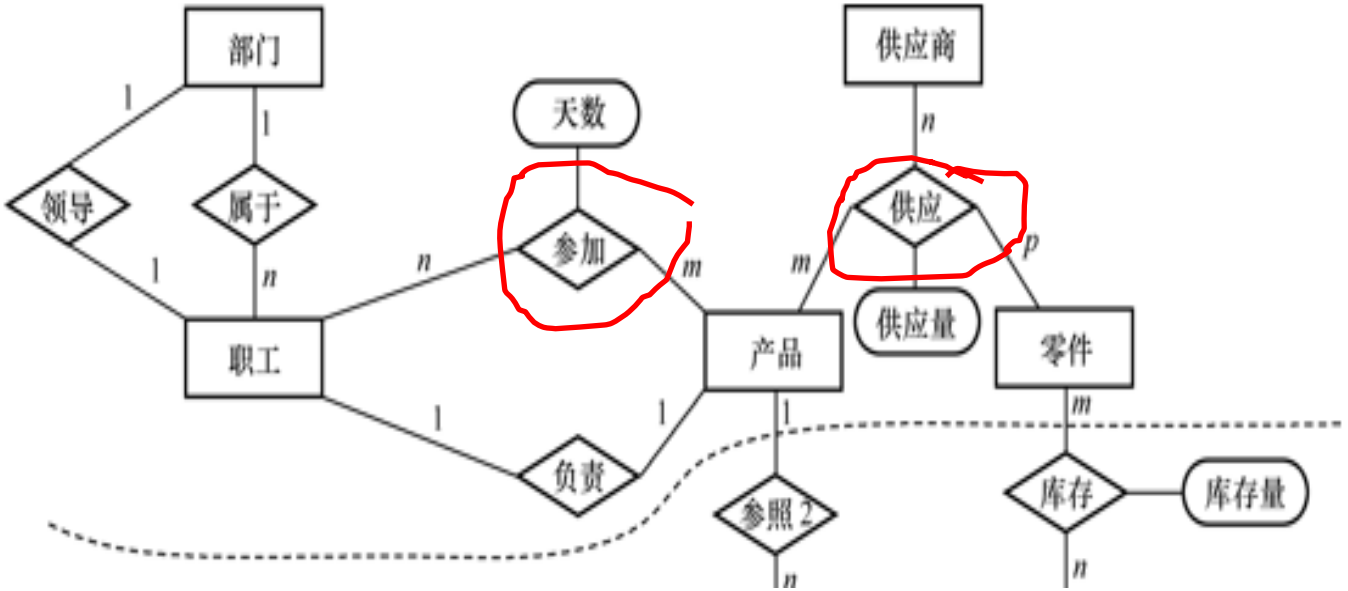
[例] 把图7.28中虚线上部的E-R图转换为关系模型

- “参加”联系的关系模式

  职工工作（职工号，产品号，工作天数，…）

- “供应”联系的关系模式

  供应（产品号，供应商号，零件号，供应量）



E-R图向关系模型的转换（续）

[例] 总结 把图7.28中E-R图转换为关系模型

- 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，…）
- 职工工作（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）

E-R图向关系模型的转换（续）

一些TIP：

- 从理论上讲，1:1联系可以与任意一端对应的关系模式合并
  - 但在一些情况下，与不同的关系模式合并效率会大不一样。因此究竟应该与哪端的关系模式合并需要依应用的具体情况而定
- 由于连接操作是最费时的操作，所以一般应以尽量减少连接操作为目标
  - 例如，如果经常要查询某个班级的班主任姓名，则将管理联系与教师关系合并更好些

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.2 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
- 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。
- 关系数据模型的优化通常以规范化理论为指导

数据模型的优化（续）

- 优化数据模型的方法
  - 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. 消除冗余的联系

    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。

数据模型的优化（续）

3. 确定所属范式

- 按照数据依赖的理论对关系模式逐一进行分析
- 考查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

❖ 关系模式规范化的基本步骤

1NF

↓ 消除非主属性对码的部分函数依赖

消除决定属性集非码的非平凡函数依赖

2NF

↓ 消除非主属性对码的传递函数依赖

3NF

↓ 消除主属性对码的部分和传递函数依赖

BCNF

↓ 消除非平凡且非函数依赖的多值依赖

4NF

4. 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。

数据模型的优化（续）

注意：并不是规范化程度越高的关系就越优

- 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
- 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。若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。若有更新操作，需要考虑完整性约束（如触发器）
- 一般说来，第三范式就足够了

数据模型的优化（续）

例：在关系模式

学生成绩单(学号,英语,数学,语文,平均成绩) 中存在下列函数依赖：

学号→英语

学号→数学

学号→语文

学号→平均成绩

(英语, 数学, 语文)→平均成绩

数据模型的优化（续）

观察“平均成绩”：

学号→(英语,数学,语文) →平均成绩

因此该关系模式中存在传递函数依赖，是2NF关系

平均成绩可以由其他属性推算出来，

但如果应用中需要经常查询学生的平均成绩，为提高效率，仍然可保留该冗余数据，对关系模式不再做进一步分解

数据模型的优化（续）

5. 按照需求分析阶段得到的各种应用对数据处理的要求，对关系模式进行必要的分解，以提高数据操作的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

数据模型的优化（续）

- 水平分解
  - 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
    - 例如，物流系统：跨域快递，同城快递
  - 水平分解的适用范围
    - 满足“80/20原则”的应用。把经常被使用的数据（约20%）水平分解出来，形成一个子关系
    - 并发事务经常存取不相交的数据
    - 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

数据模型的优化（续）

- 垂直分解
  - 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
    - 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
      - 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

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.3 设计用户子模式

- 定义数据库模式—全局模式
  - 考虑系统应用的时间效率、空间效率、易维护等
- 定义用户子模式（外模式）—视图机制
  - 考虑局部应用的特殊需求和用户体验

(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View，以满足系统对安全性的要求

(3) 简化用户对系统的使用

设计用户子模式（续）

（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。

设计用户子模式（续）

2. 针对不同级别的用户定义不同视图，保证系统安全性

- 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）
  - 生产领导部门则可以查询全部产品数据
- 可以防止用户非法访问不允许他们查询的数据，保证系统的安全性

设计用户子模式（续）

（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。

第七章 数据库设计

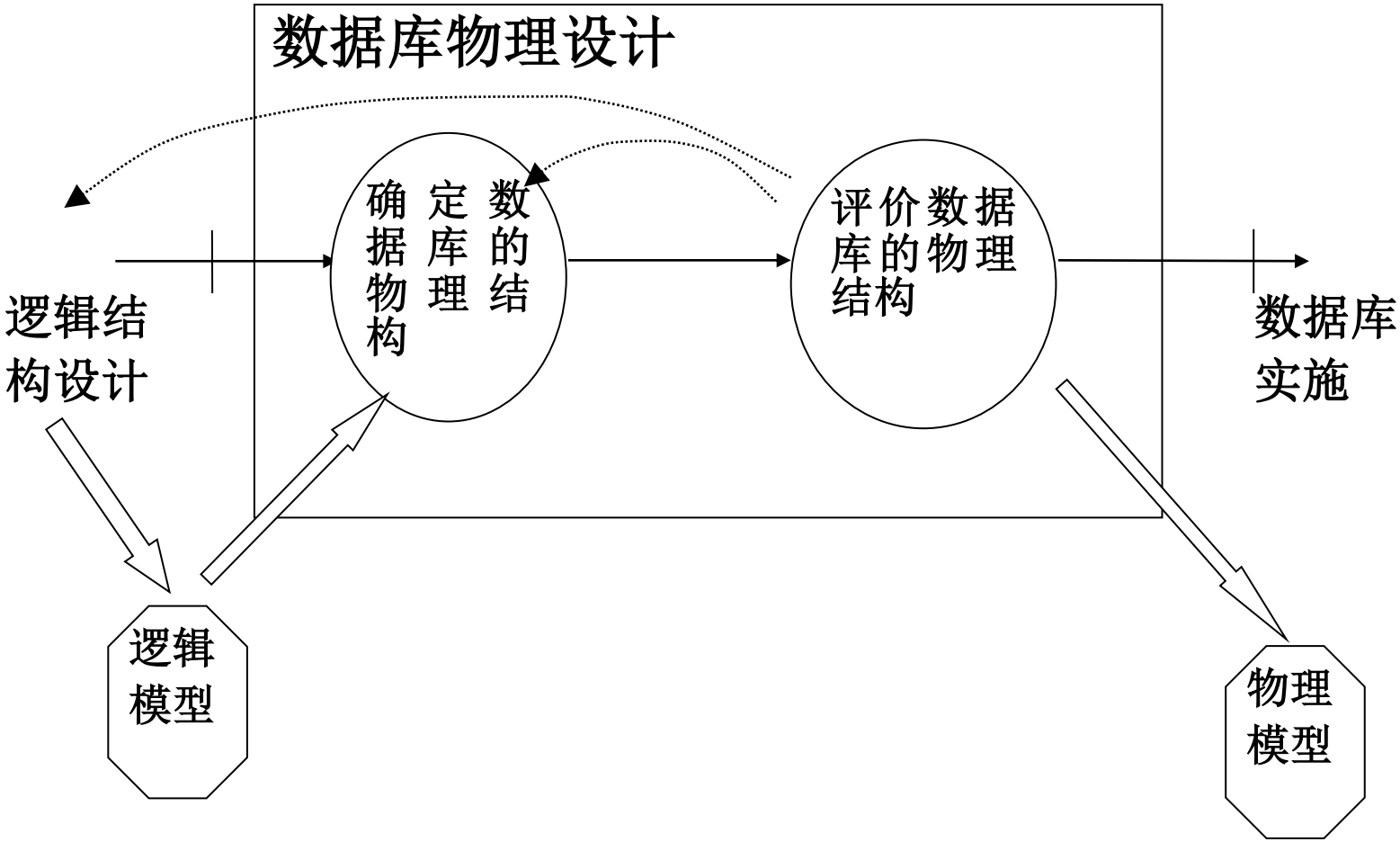
7.5 数据库的物理设计

- 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环境的物理结构的过程，就是数据库的物理设计

数据库的物理设计(续)

- 数据库物理设计的步骤
  - 1. 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;
  - 2. 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。

数据库的物理设计(续)



回顾：数据字典

- **4. 数据存储**
  - 输入输出
  - 数据量
  - 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
- **5. 处理过程**
  - 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

物理设计的输入及性能评价的标准

7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.1 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，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
    - 有哪些索引(B+树，HASH，聚簇)，如何建立索引
    - 有哪些存储结构（行存储，列存储，块存储），如何选择

数据库的物理设计的内容和方法（续）

- 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

数据库的物理设计的内容和方法（续）

- 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

7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.2 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
- 物理设计的任务之一就是要确定选择哪些存取方法，即建立哪些存取路径

建立索引

- 语句格式

CREATE [UNIQUE] INDEX <索引名>

ON <表名>

[USING 索引方法]（列名1，列名2，[, …]）；

■ **UNIQUE：此索引的每一个索引值只对应唯一的数据记录**

例：

CREATE UNIQUE INDEX Studentname ON Student USING Hash(sname);

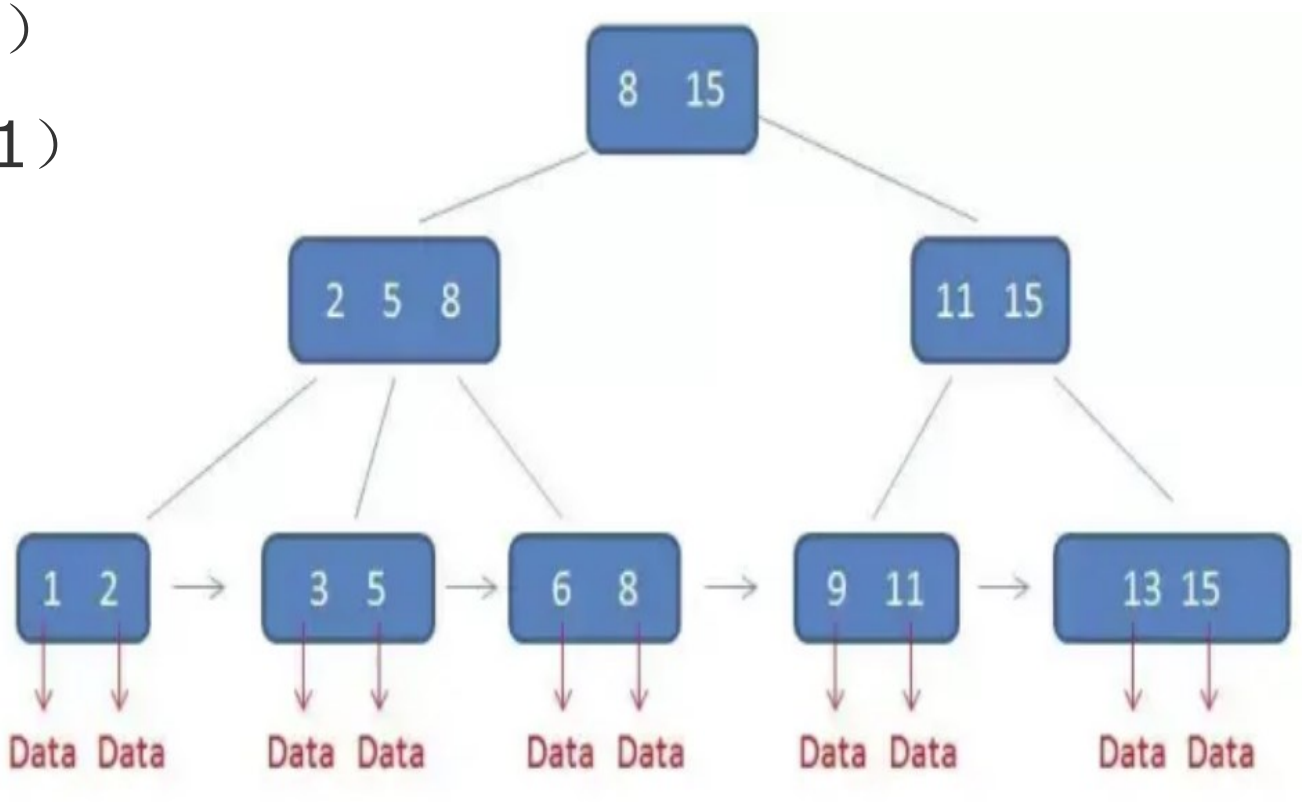
关系模式存取方法选择（续）

- DBMS常用存取方法
  - B+树索引方法
    - 经典存取方法，使用最普遍
  - HASH方法
  - 聚簇（Cluster）方法

B+树索引方法

B+树

- 索引与数据
- 单值查询（369）
- 范围查询（3-11）



一、索引存取方法的选择(B+树)

- 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

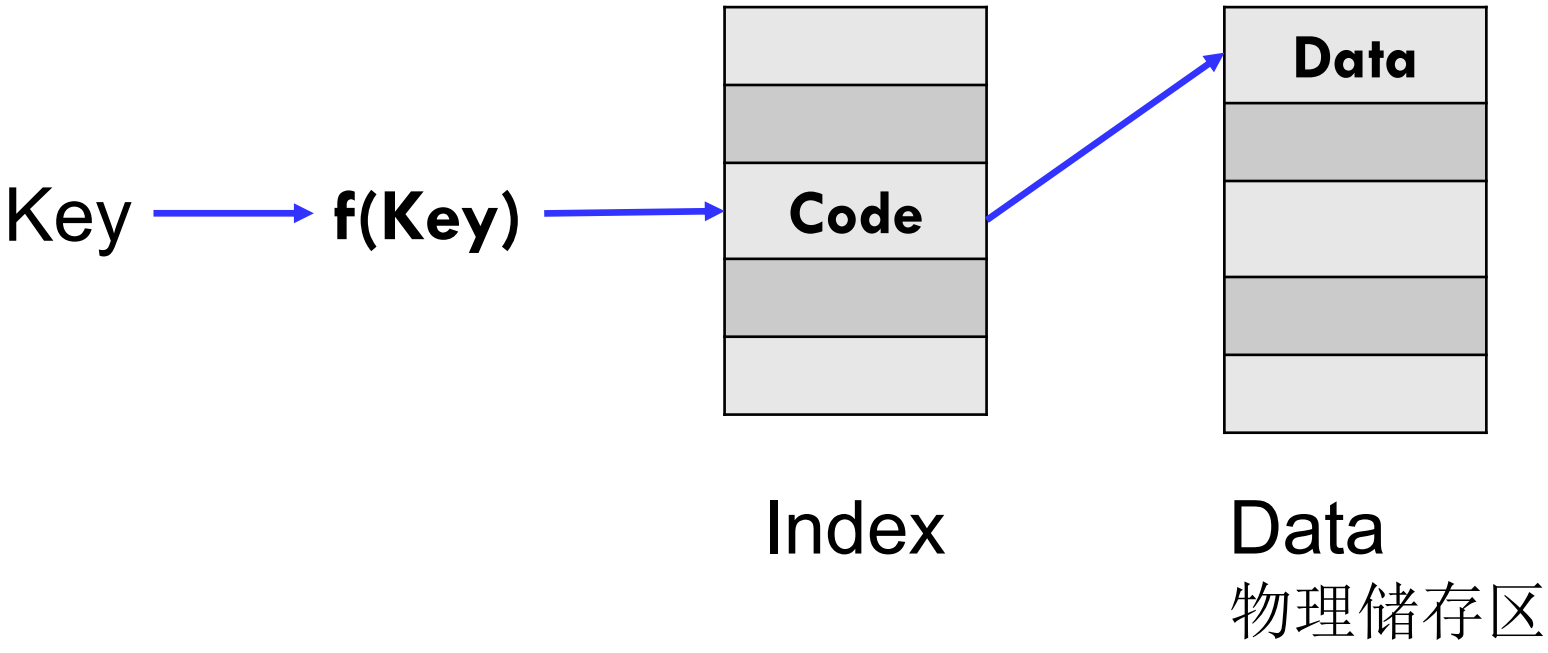
B+树索引存取方法的选择

- 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
  - 维护索引的开销
  - 查找索引的开销

HASH索引

HASH

哈希函数 *f* (key)



二、HASH存取方法的选择

- 选择HASH存取方法的规则
  - 当一个关系满足下列两个条件时，可以选择HASH存取方法
    - 该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
    - 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HASH存取方法

三、聚簇存取方法的选择

- 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称为聚簇
    - 该属性（或属性组）称为聚簇码（cluster key）
    - 许多关系型数据库管理系统都提供了聚簇功能
    - 聚簇存放与聚簇索引的区别

聚簇存取方法的选择（续）

- 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
- 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

建立索引

- 语句格式

CREATE [CLUSTER] INDEX <索引名>

ON <表名> (<列名>)

- **CLUSTER：** 表示要建立的索引是聚簇索引

不同DBMS的实现方式不同

回顾：建立索引

[例]在Student表的Sage列上建立一个聚簇索引

```
CREATE CLUSTER INDEX Stuage
 ON  Student(Sage);
```

“年龄”属性建立聚簇索引，是合适的

[例] CREATE CLUSTER INDEX Stusname

```
 ON  Student(Sname);
```

在Student表的Sname（姓名）列上建立一个聚簇索引，而且Student表中的记录将按照Sname值的升序存放

聚簇存取方法的选择（续）

- 聚簇的用途
  - 1. 大大提高按聚簇码进行查询的效率

    例：学生关系按所在系建有索引，现查询DS系的学生名单。

    - 随机存放：DS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（注意索引与数据地址）
    - 按照系名聚簇存放：将同一系的学生元组聚簇存放，则每读一个物理块可得到多个满足查询条件的元组，显著减少访问磁盘次数。DS系500名学生聚簇存放50个物理块，只要执行50次I/O
  - 2. 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

聚簇存取方法的选择（续）

- 聚簇的适用范围

1. 既适用于单个关系独立聚簇，也适用于多个关系组合聚簇

例：假设用户经常要按系别查询学生成绩单，

Select sname, cno, grade from student, sc where student.sno=sc.sno

- 查询涉及Student关系和SC关系的连接操作，按学号连接
  - 把具有相同学号值的学生元组和选修元组在物理上聚簇在一起。
  - 相当于把多个关系按“预连接”的形式存放，
  - 从而大大提高连接操作的效率。

聚簇存取方法的选择（续）

- 聚簇的适用范围

2. 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

- **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，可以省去对结果集的排序操作**

聚簇存取方法的选择（续）

- 聚簇的局限性
  - 一个基表上最多建立一个聚簇索引
  - 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，将导致关系中元组移动其物理存储位置，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移动
      - 例，学生从CS换到DS系

聚簇存取方法的选择（续）

- 聚簇的适用条件：设计候选聚簇
  - 经常在一起进行连接操作的关系可以建立聚簇
  - 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
  - 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
    - 即对应每个聚簇码值的平均元组数不太少。聚簇的效果不明显

聚簇存取方法的选择（续）

- 聚簇的适用条件：优化聚簇设计

  （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）从聚簇中删除重复出现的关系

  - 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
    - 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

7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.3 确定数据库的存储结构

- 确定数据库物理结构的内容
  - 1. 确定数据的存放位置和存储结构
    - 关系
    - 索引
    - 聚簇
    - 日志
    - 备份

    - 内存/磁盘
    - 列存放，行存储
    - 集中存放，分散存放
    - 顺序存放，随机存放，聚簇存放
  - 2. 确定系统配置

1. 确定数据的存放位置

- 确定数据存放位置和存储结构的因素
  - 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

  例：消除一切冗余数据虽能够节约存储空间和减少维护代价，但往往会导致检索代价的增加

  必须进行权衡，选择一个折中方案

确定数据的存放位置（续）

- 基本原则
  - 根据应用情况将
    - 易变部分与稳定部分分开存放
    - 存取频率较高部分与存取频率较低部分，分开存放本
    - 日志与数据库对象（表，索引等）分开存放

确定数据的存放位置（续）

例：

- 数据库数据备份、日志文件备份等由于只在故障恢复时才使用，而且数据量很大，可以考虑存放在磁带上
- 如果计算机有多个磁盘或磁盘阵列，可以考虑将表和索引分别放在不同的磁盘上，在查询时，由于磁盘驱动器并行工作，可以提高物理I/O读写的效率
- 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

2. 确定系统配置

- **DBMS产品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
  - 锁的数目等

2. 确定系统配置

- 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。

7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

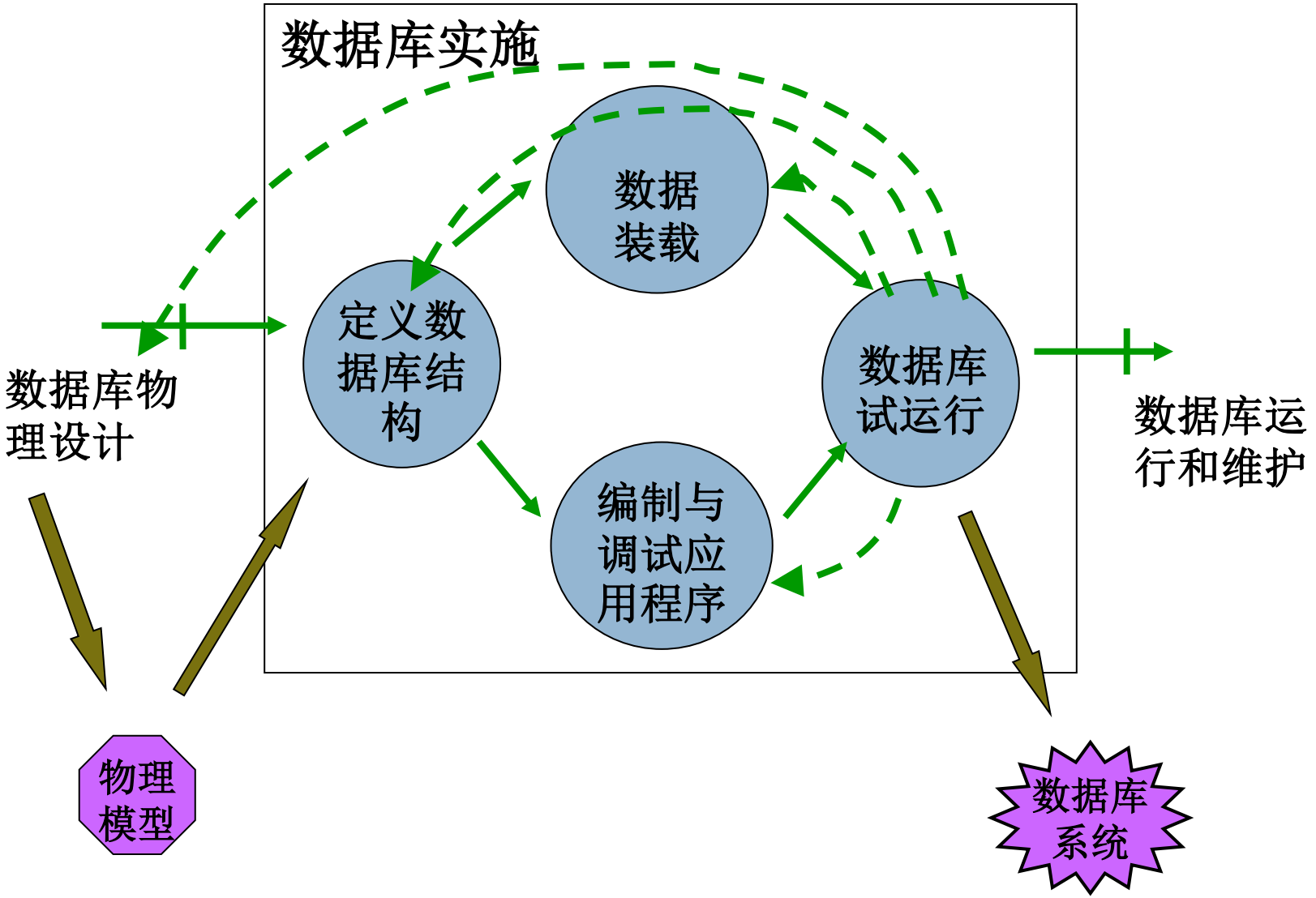
7.5.4 评价物理结构

7.5.4 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
  - 如果该结构不符合用户需求，则需要修改设计

第七章 数据库设计

7.6数据库实施和维护


数据库实施
数据装载
定义数据库结构
数据库试运行
编制与调试应用程序
数据库物理设计
数据库运行和维护
物理模型
数据库系统

7.6数据库实施和维护

7.6.1 数据的载入和应用程序的调试

7.6.2 数据库的试运行

7.6.3 数据库的运行和维护

7.6.1 数据的载入和应用程序的调试

- 数据的载入
- 应用程序的编码和调试

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ETL
  - 数据抽取
  - 数据转换
  - 数据载入
- 使用ETL工具辅助完成

ETL工作是费时、费力的　　数据预处理是费时、费力的

应用程序的编码和调试

- 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
- 软件工程：应用程序的设计、编码和调试的方法、步骤

7.6数据库实施和维护

7.6.1 数据的载入和应用程序的调试

7.6.2 数据库的试运行

7.6.3 数据库的运行和维护

7.6.2 数据库的试运行

- 在原有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，称为数据库的试运行
- 主要工作包括：

1）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，直到达到设计要求

2）性能测试

- 测量系统的性能指标，分析是否达到设计目标
- 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

数据库的试运行（续）

- 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（续）

强调两点：

1. 分期分批组织数据入库
   - 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太大，费时、费力，所以应分期分批地组织数据入库
     - 先输入小批量数据供调试用
     - 待试运行基本合格后再大批量输入数据
     - 逐步增加数据量，逐步完成运行评价

数据库的试运行（续）

2. 数据库的转储和恢复（第十章）

- 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。

7.6 数据库实施和维护

7.6.1 数据的载入和应用程序的调试

7.6.2 数据库的试运行

7.6.3 数据库的运行和维护

7.6.3 数据库的运行与维护

- 数据库试运行合格后，数据库即可投入正式运行。
- 数据库投入运行标志着开发任务的基本完成和维护工作的开始
- 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，也是设计工作的继续和提高。
  - 应用环境在不断变化
  - 数据库运行过程中物理存储会不断变化

7.6.3 数据库的运行与维护

- 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，包括：
  1. 数据库的转储和恢复
  2. 数据库的安全性、完整性控制
  3. 数据库性能的监督、分析和改进
  4. 数据库的重组织和重构造

7.6.3 数据库的运行与维护

- 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：

1. 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。

数据库的运行和维护

2. 数据库的安全性、完整性控制

- 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

数据库的运行和维护

3. 数据库性能的监督、分析和改进

- 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

数据库的运行和维护（续）

4. 数据库的重组织与重构造

（1）数据库的重组织

- 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。

数据库的运行与维护（续）

- 重组织的形式
  - 全部重组织
    - 索引重组、单表重组、表空间重组
  - 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

数据库运行与维护（续）

（2）数据库重构造

实体，联系发生了变化，根据新环境调整数据库的模式和内模式

- 增加新的数据项
- 改变数据项的类型
- 改变数据库的容量
- 增加或删除索引
- 修改完整性约束条件

第七章 数据库设计

7.7 小结

- 数据库的设计过程
  - 需求分析
  - 概念结构设计
  - 逻辑结构设计
  - 物理设计
  - 实施和维护

小结（续）

- 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。