

数据库设计

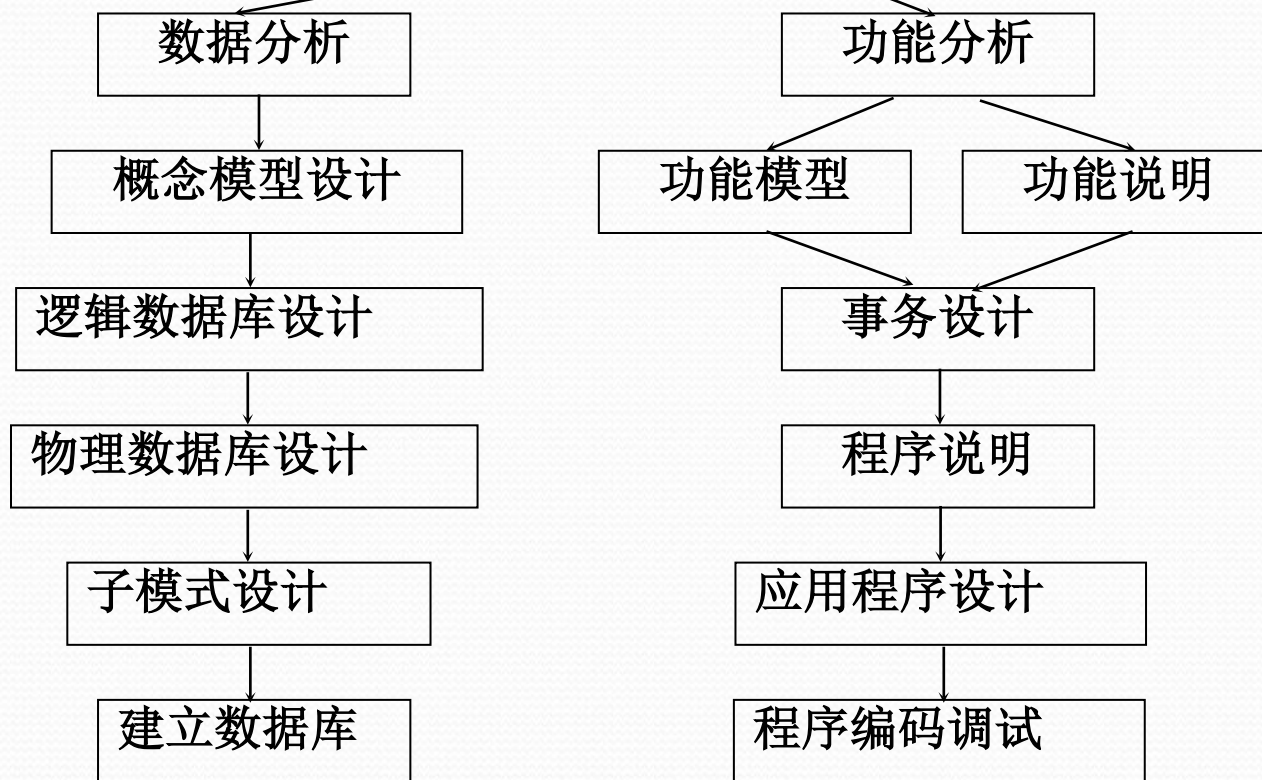
数据库设计概述

- 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。
 - 目标：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

数据库设计的特点

- 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
 - 收集、入库
 - 更新新的数据
- 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

数据库设计的特点（续）



结构和行为分离的设计

数据库设计方法

- 手工与经验相结合方法
 - 设计质量与设计人员的经验和水平有直接关系
 - 数据库运行一段时间后常常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

数据库设计方法（续）

- 新奥尔良（New Orleans）方法
 - 将数据库设计分为若干阶段和步骤
- 基于E-R模型的数据库设计方法
 - 概念设计阶段广泛采用
- 3NF（第三范式）的设计方法
 - 逻辑阶段可采用的有效方法
- ODL（Object Definition Language）方法
 - 面向对象的数据库设计方法
- Barker方法（Oracle）

数据库设计方法（续）

- 计算机辅助设计
 - ORACLE Designer 2000
 - SYBASE PowerDesigner

数据库设计的基本步骤

- 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DBMS密切相关

数据库设计的基本步骤（续）

一、数据库设计的准备工作：选定参加设计的人

1. 系统分析人员、数据库设计人员

- 自始至终参与数据库设计

2. 用户和数据库管理员

- 主要参加需求分析和数据库的运行维护

3. 应用开发人员（程序员和操作员）

- 在系统实施阶段参与进来，负责编制程序和准备软硬件环境

数据库设计的基本步骤（续）

二、数据库设计的过程(六个阶段)

1.需求分析阶段

- 准确了解与分析用户需求（包括数据与处理）
- 最困难、最耗费时间的一步

数据库设计的基本步骤（续）

2.概念结构设计阶段

- 整个数据库设计的关键
- 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DBMS的概念模型

数据库设计的基本步骤（续）

3.逻辑结构设计阶段

- 将概念结构转换为某个DBMS所支持的数据模型
- 对其进行优化

数据库设计的基本步骤（续）

4.数据库物理设计阶段

- 为逻辑数据模型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）

数据库设计的基本步骤（续）

5.数据库实施阶段

- 运用DBMS提供的数据库语言（如SQL）及宿主语言，根据逻辑设计和物理设计的结果
 - 建立数据库
 - 编制与调试应用程序
 - 组织数据入库
 - 进行试运行

数据库设计的基本步骤（续）

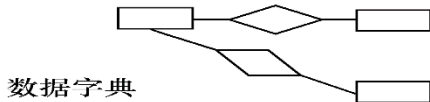
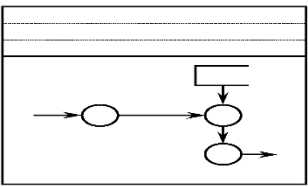
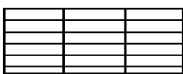
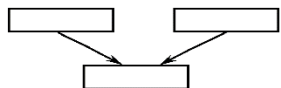
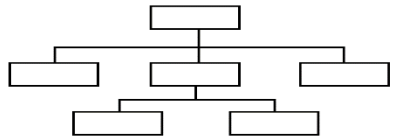


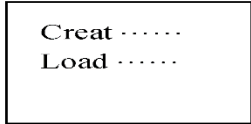
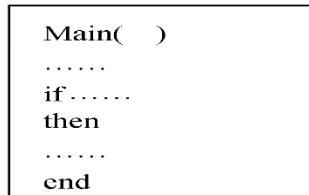
6.数据库运行和维护阶段

- 数据库应用系统经过试运行后即可投入正式运行
- 在数据库系统运行过程中必须不断地对其进行评价、调整与修改

数据库设计的基本步骤（续）

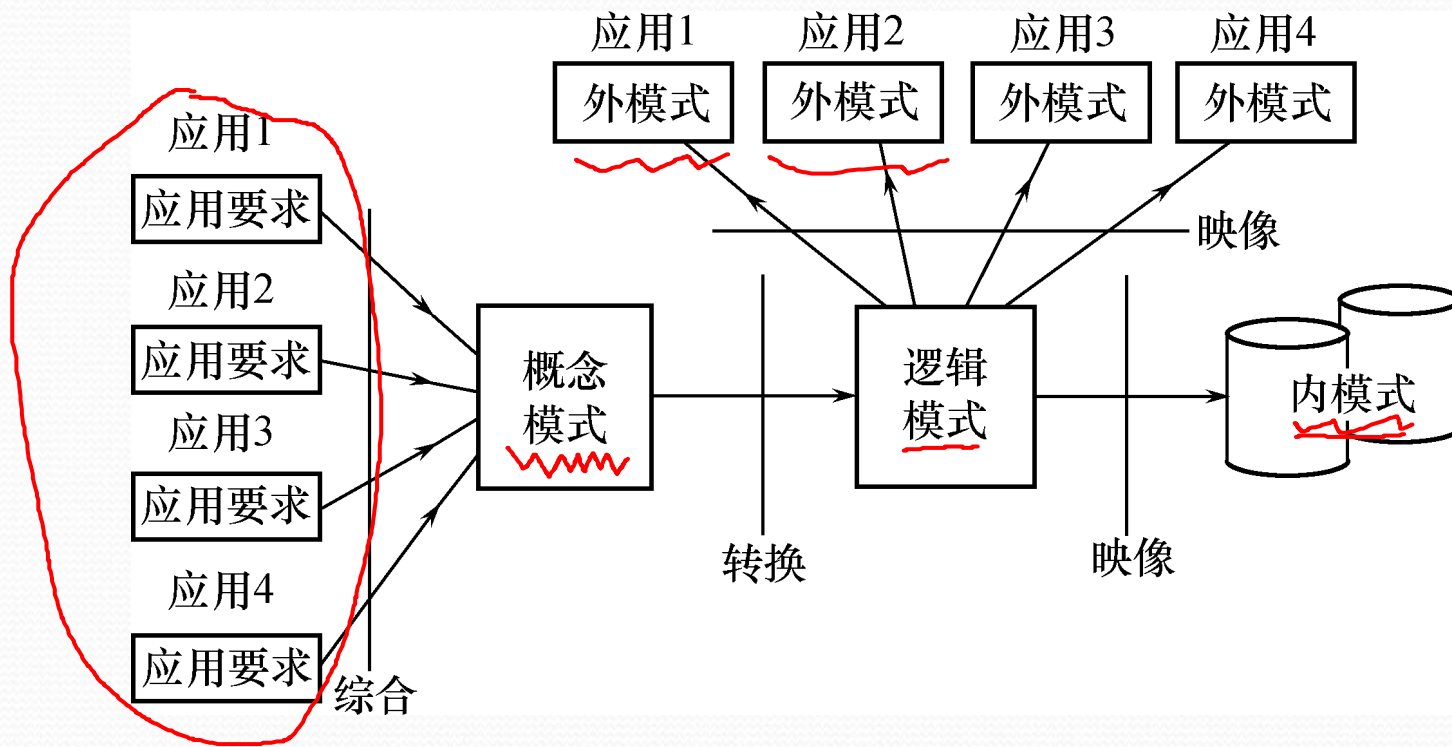
设计一个完善的数据库应用系统往往是上述六个阶段的不断反复

- 把数据库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

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）

数据库设计过程中的各级模式

数据库设计不同阶段形成的数据库各级模式



数据库的各级模式

需求分析的任务

- ❖ 需求分析的任务
- ❖ 需求分析的重点
- ❖ 需求分析的难点

需求分析的任务

- 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）
- 明确用户的各种需求
- 确定新系统的功能
- 充分考虑今后可能的扩充和改变

需求分析的重点

- 调查的重点是“数据”和“处理”，获得用户对数据库要求
 - 信息要求
 - 处理要求
 - 安全性与完整性要求

需求分析的难点

- 确定用户最终需求
 - 用户缺少计算机知识
 - 设计人员缺少用户的专业知识
- 解决方法
 - 设计人员必须不断深入地与用户进行交流

需求分析的方法

- 调查需求
- 达成共识
- 分析表达需求

调查用户需求的具体步骤

- (1) 调查组织机构情况
- (2) 调查各部门的业务活动情况。
- (3) 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。
- (4) 确定新系统的边界

常用调查方法

(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

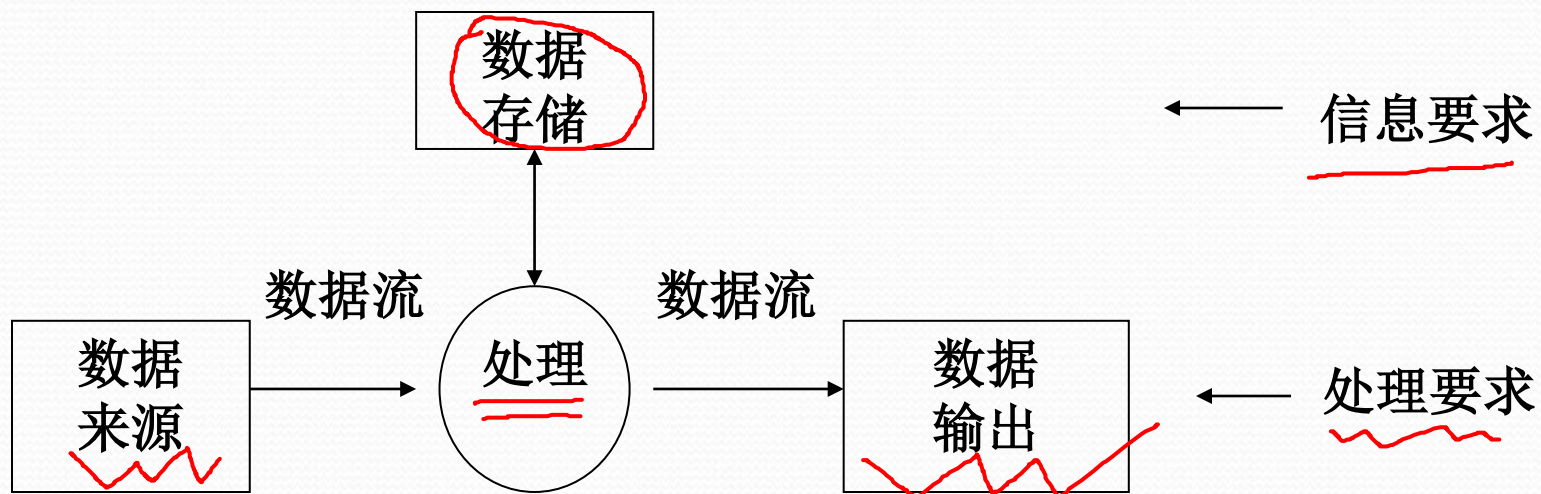
(6)查阅记录

进一步分析和表达用户需求

- 结构化分析方法（Structured Analysis, 简称SA方法）
 - 从最上层的系统组织机构入手
 - 自顶向下、逐层分解分析系统

进一步分析和表达用户需求（续）

1. 首先把任何一个系统都抽象为：



进一步分析和表达用户需求（续）

2. 分解处理功能和数据

(1) 分解处理功能

- 将处理功能的具体内容分解为若干子功能

(2) 分解数据

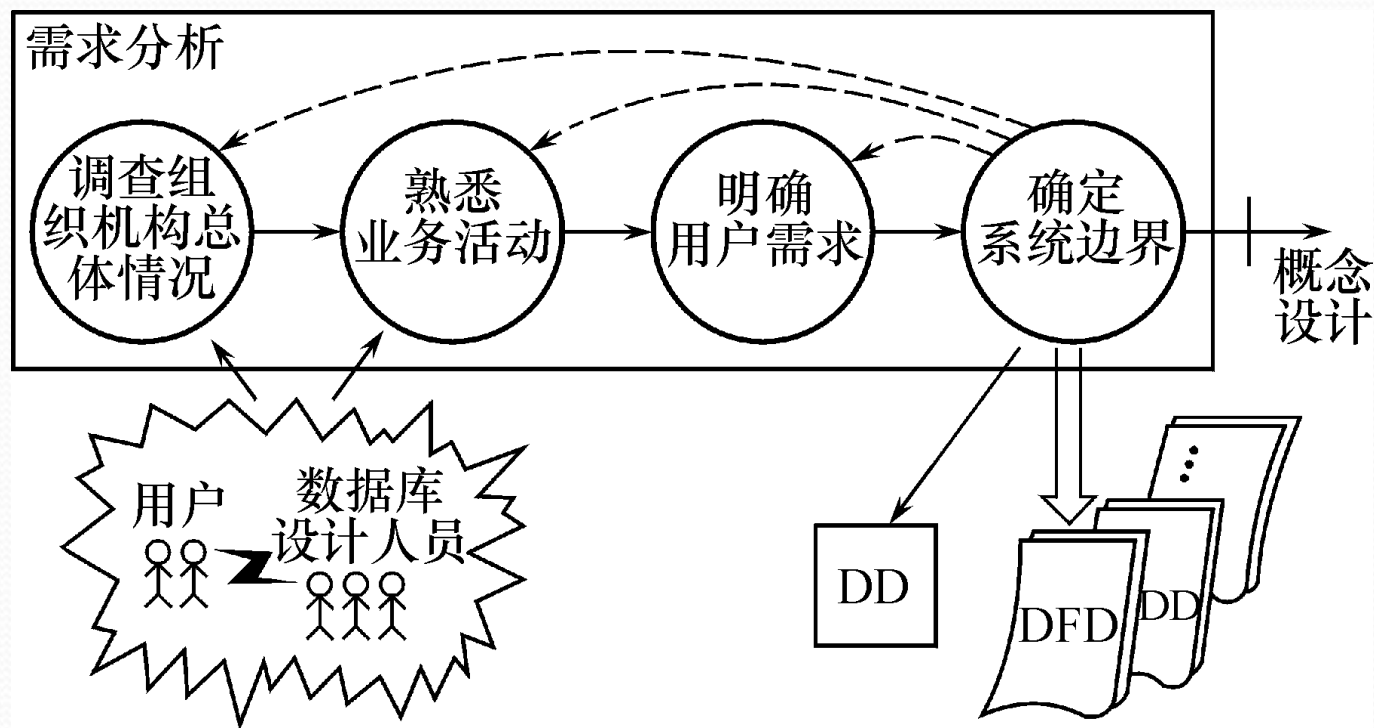
- 处理功能逐步分解同时，逐级分解所用数据，形成若干层次的数据流图

(3) 表达方法

- 处理逻辑：用判定表或判定树来描述
- 数据：用数据字典来描述

3. 将分析结果再次提交给用户，征得用户的认可

需求分析过程



需求分析过程

数据字典

- 数据字典的用途
 - 进行详细的数据收集和分析所获得的主要结果
- 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

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 }

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 }

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 }

5. 处理过程

- 具体处理逻辑一般用判定表或判定树来描述
- 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 } }

数据字典举例

例：学生学籍管理子系统的数据字典。

数据项，以“学号”为例：

数据项：学号

含义说明：唯一标识每个学生

别名：学生编号

类型：字符型

长度：8

取值范围：00000000至99999999

取值含义：前两位标别该学生所在年级，
后六位按顺序编号

与其他数据项的逻辑关系：

处理过程（续）

数据结构，以“学生”为例

“学生”是该系统中的一个核心数据结构：

数据结构： 学生

含义说明： 是学籍管理子系统的主体数据结构，
定义了一个学生的有关信息

组成： 学号，姓名，性别，年龄，所在系，年级

处理过程（续）

数据流，“体检结果”可如下描述：

数据流： 体检结果

说明： 学生参加体格检查的最终结果

数据流来源：体检

数据流去向：批准

组成：

平均流量：

高峰期流量：

处理过程（续）

数据存储，“学生登记表”可如下描述：

数据存储： 学生登记表

说明： 记录学生的基本情况

流入数据流：

流出数据流：

组成：

数据量： 每年3000张

存取方式： 随机存取

处理过程（续）

处理过程“分配宿舍”可如下描述：

处理过程：分配宿舍

说明： 为所有新生分配学生宿舍

输入： 学生，宿舍

输出： 宿舍安排

处理： 在新生报到后，为所有新生分配学生宿舍。

要求同一间宿舍只能安排同一性别的学生，

同一个学生只能安排在一个宿舍中。

每个学生的居住面积不小于3平方米。

安排新生宿舍其处理时间应不超过15分钟。

数据字典

- 数据字典是关于数据库中数据的描述，是元数据，而不是数据本身
- 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

需求分析小结

- 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

概念结构设计

- 什么是概念结构设计
 - 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
 - 概念结构是各种数据模型的基础，它比数据模型更独立于机器、更抽象，从而更加稳定
 - 概念结构设计是整个数据库设计的关键

概念结构（续）



概念结构（续）

- 概念结构设计的特点

(1) 能真实、充分地反映现实世界

(2) 易于理解

(3) 易于更改

(4) 易于向关系、网状、层次等各种数据模型转换

概念结构（续）

- 描述概念模型的工具
 - E-R模型
 - IDEF1X

IEDF1X建模方法

- IDEF是ICAM DEFinition method 的缩写,是美国 空军在70年代末80年代初, ICAM (Integrated Computer Aided Manufacturing) 工程在结构化分析和设计方法基础上发展的一套系统分析和设计方法。后来就称之为Integration Definition method, 简称不变。

IDEF建模系列

- IDEF₀ 功能模型
- IDEF_{1X} 数据模型
- IDEF₂ 仿真模型设计
- IDEF₃ 过程描述获取
- IDEF₄ 面向对象设计
- IDEF₅ 本体论描述获取
- IDEF₆ 设计原理获取
- IDEF₇ 信息系统审定
- IDEF₈ 人与系统接口设计
- IDEF₉ 经营约束的发现
- IDEF₁₀ 信息制品建模
- IDEF₁₁ 信息工具建模
- IDEF₁₂ 组织设计
- IDEF₁₃ 三模式映射设计
- IDEF₁₄ 网络设计

IDEF1X 语法和语意 (注意和ER模型的区别和联系)

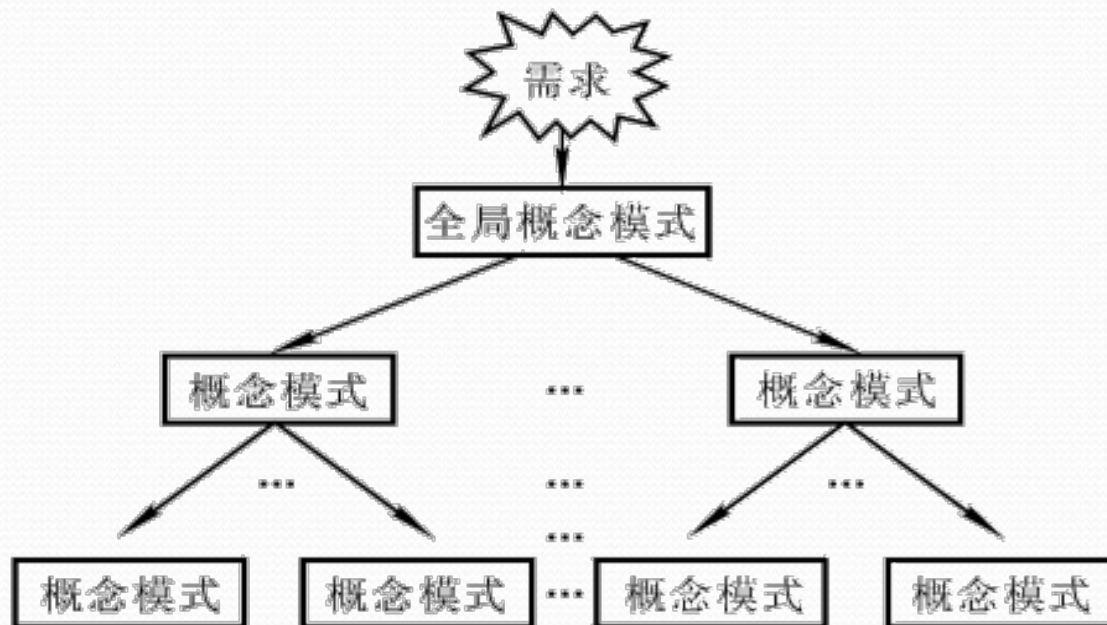
- 实体
 - 独立标识实体-四边形表示
 - 从属标识实体-圆角四边形表示
- 属性
 - 矩形框的每一行表示一个属性
 - 主键属性列在矩形最上边，用横线和其他属性分割
- 外键
 - 属性后面通过FK来标识
- 主键
 - 在属性后边以(PK)标识

概念结构设计的方法与步骤

- 设计概念结构的四类方法

- 自顶向下

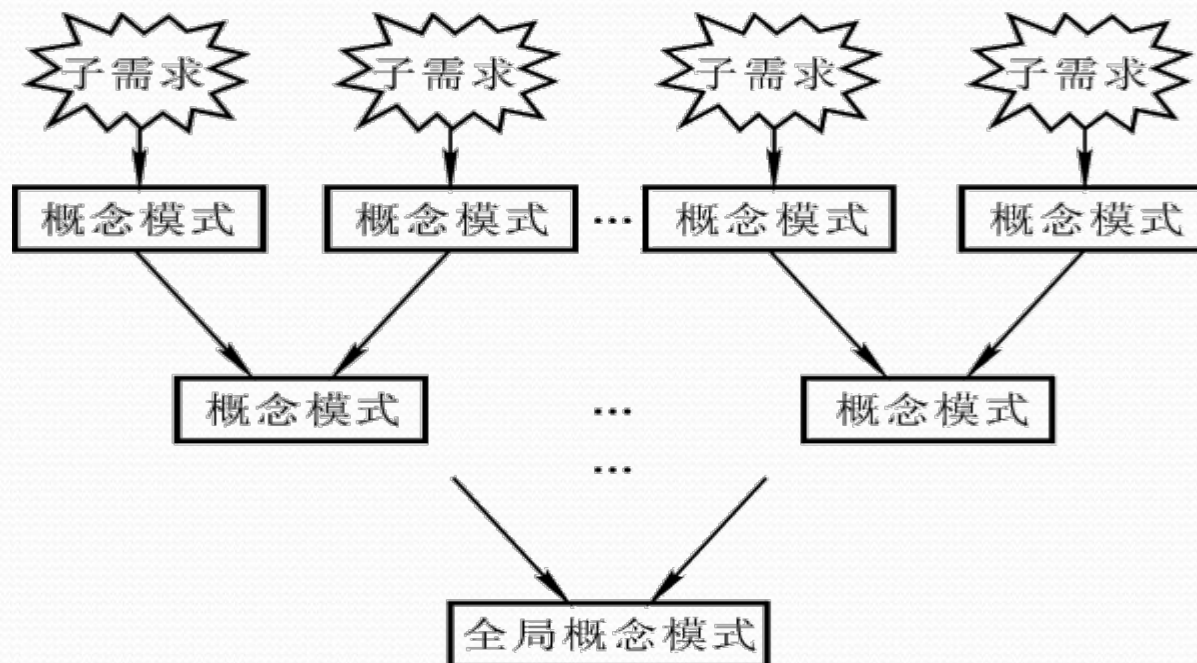
- 首先定义全局概念结构的框架，然后逐步细化



概念结构设计的方法与步骤

• 自底向上

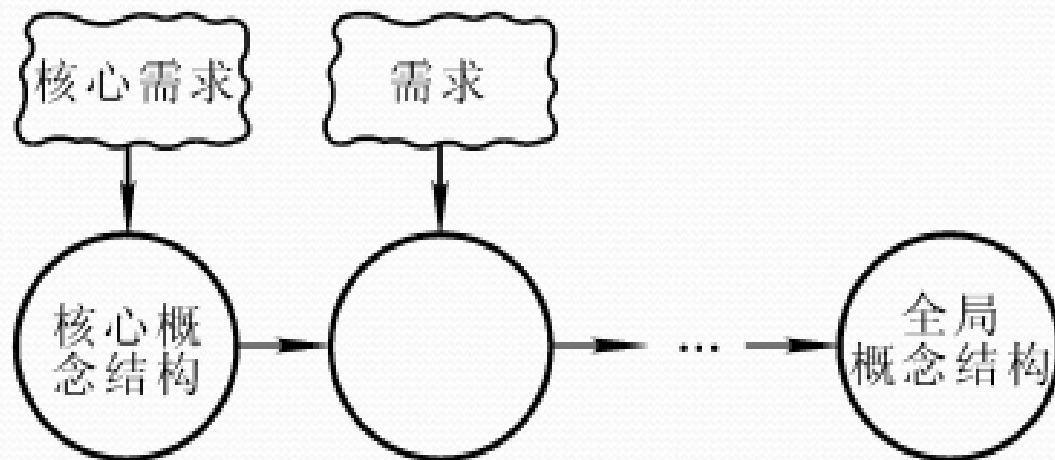
- 首先定义各局部应用的概念结构，然后将它们集成起来，得到全局概念结构



概念结构设计的方法与步骤（续）

- 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

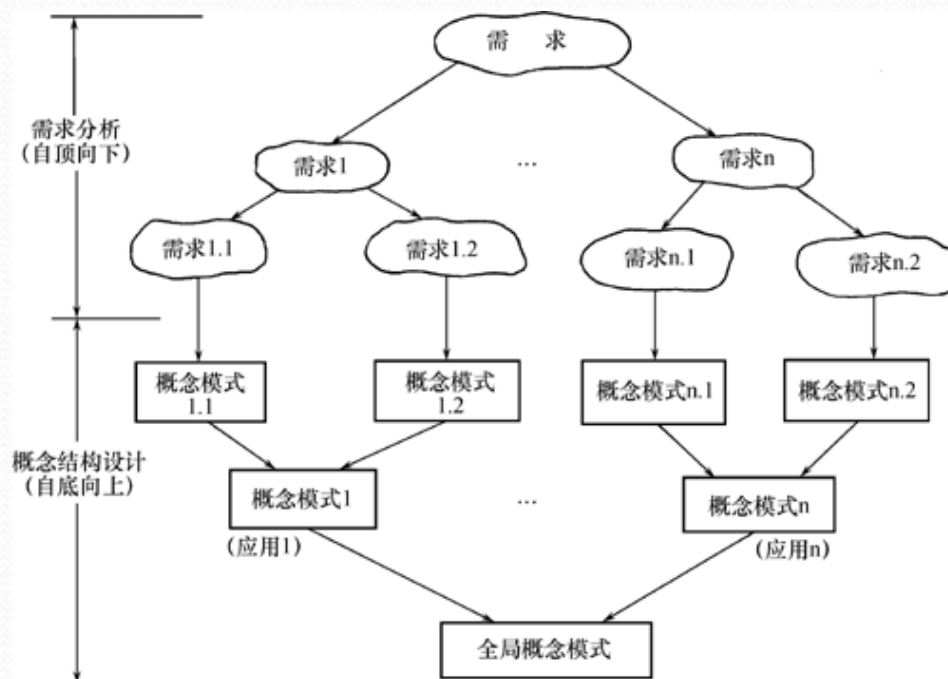
概念结构设计的方法与步骤（续）

- 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。

概念结构设计的方法与步骤（续）

- 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

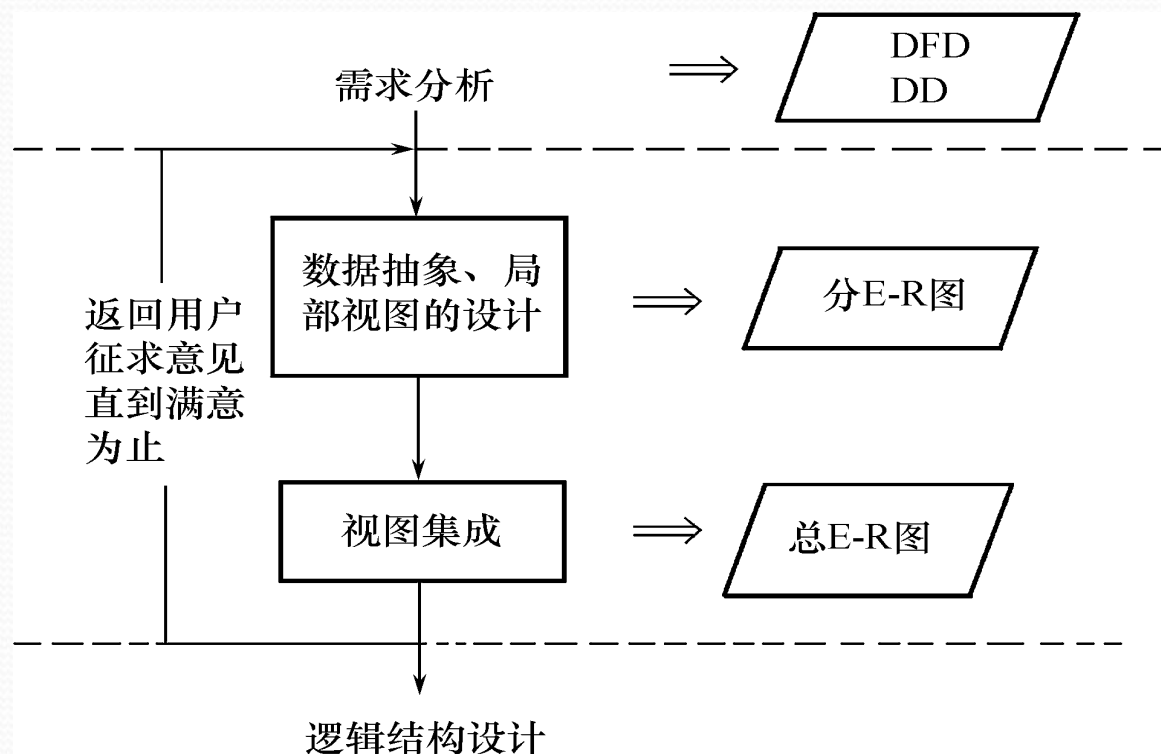


概念结构设计的方法与步骤（续）

❖ 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

第2步：集成局部视图，得到全局概念结构



数据抽象与局部视图设计

- 数据抽象
- 局部视图设计

数据抽象

- 抽象是对实际的人、物、事和概念中抽取所关心的共同特性，忽略非本质的细节，并把这些特性用各种概念精确地加以描述。
- 概念结构是对现实世界的一种抽象

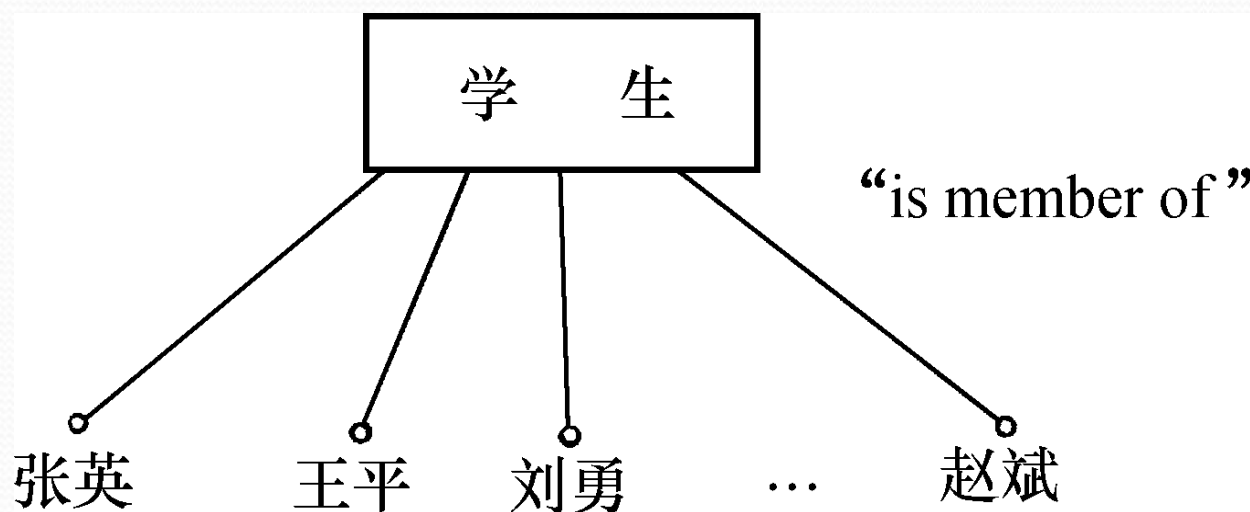
数据抽象（续）

- 三种常用抽象

1. 分类（Classification）

- 定义某一类概念作为现实世界中一组对象的类型
- 抽象了对象值和型之间的 “is member of”的语义

数据抽象（续）

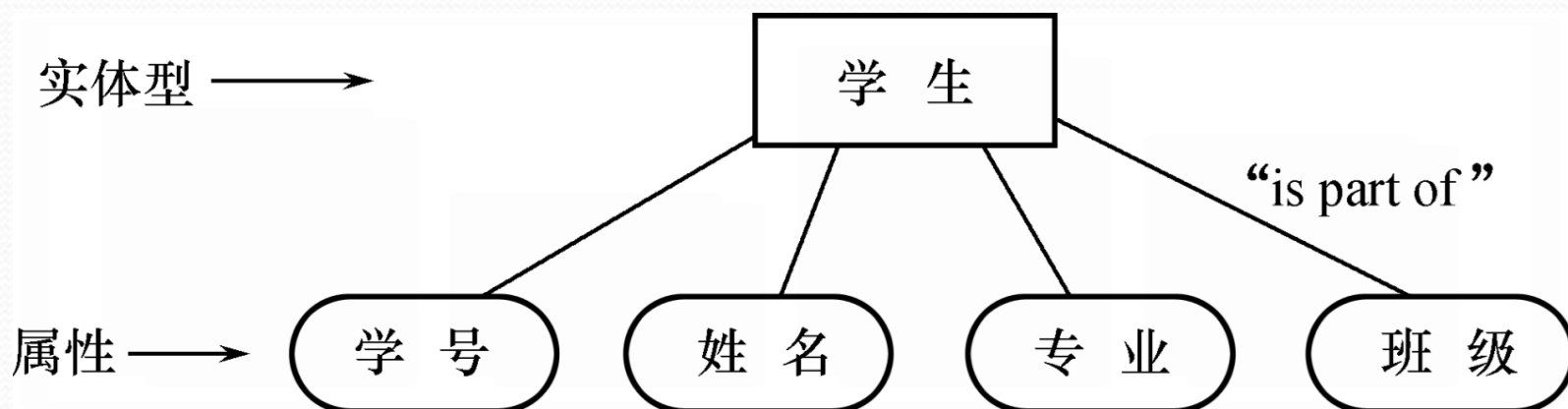


数据抽象（续）

2. 聚集（Aggregation）

- 定义某一类型的组成成分
- 抽象了对象内部类型和成分之间 “is part of”的语义

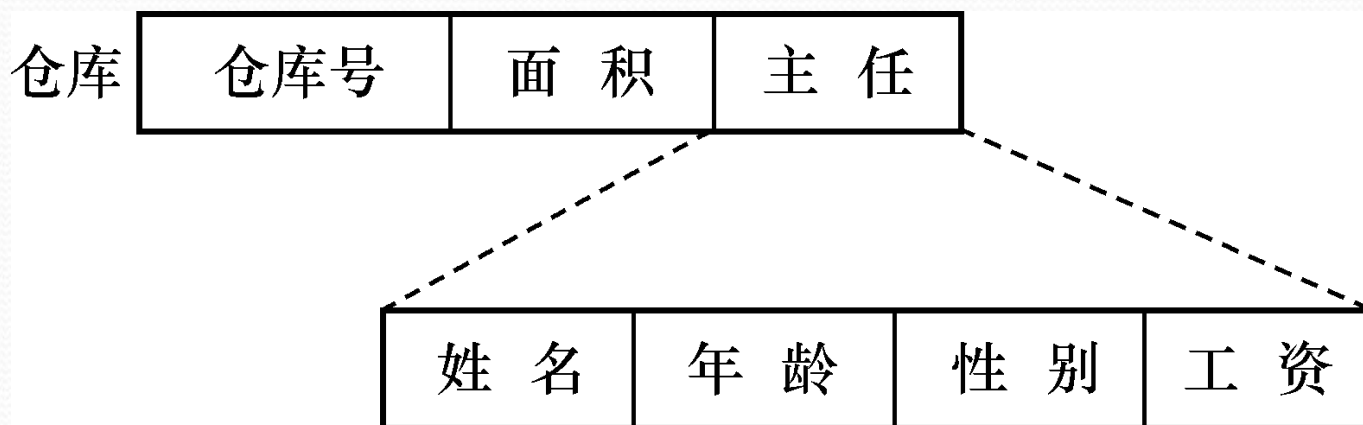
数据抽象（续）



聚集

数据抽象（续）

- 复杂的聚集，某一类型的成分仍是一个聚集



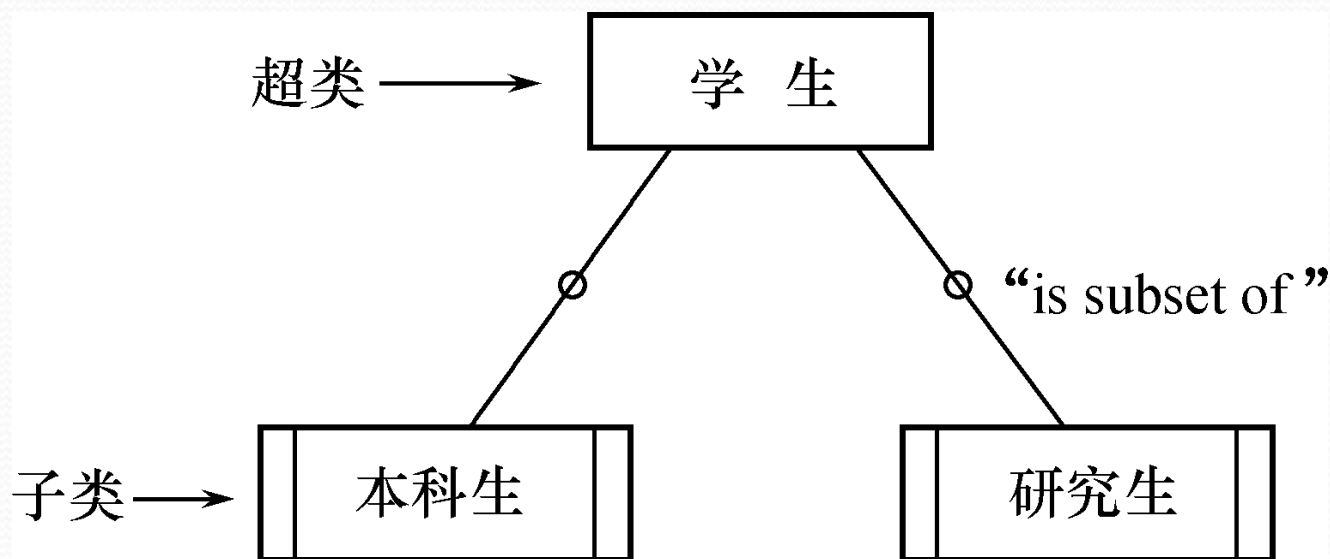
更复杂的聚集

数据抽象（续）

3. 概括（Generalization）

- 定义类型之间的一种子集联系
- 抽象了类型之间的“is subset of”的语义
- 继承性

数据抽象（续）



概括

局部视图设计

设计分E-R图的步骤:

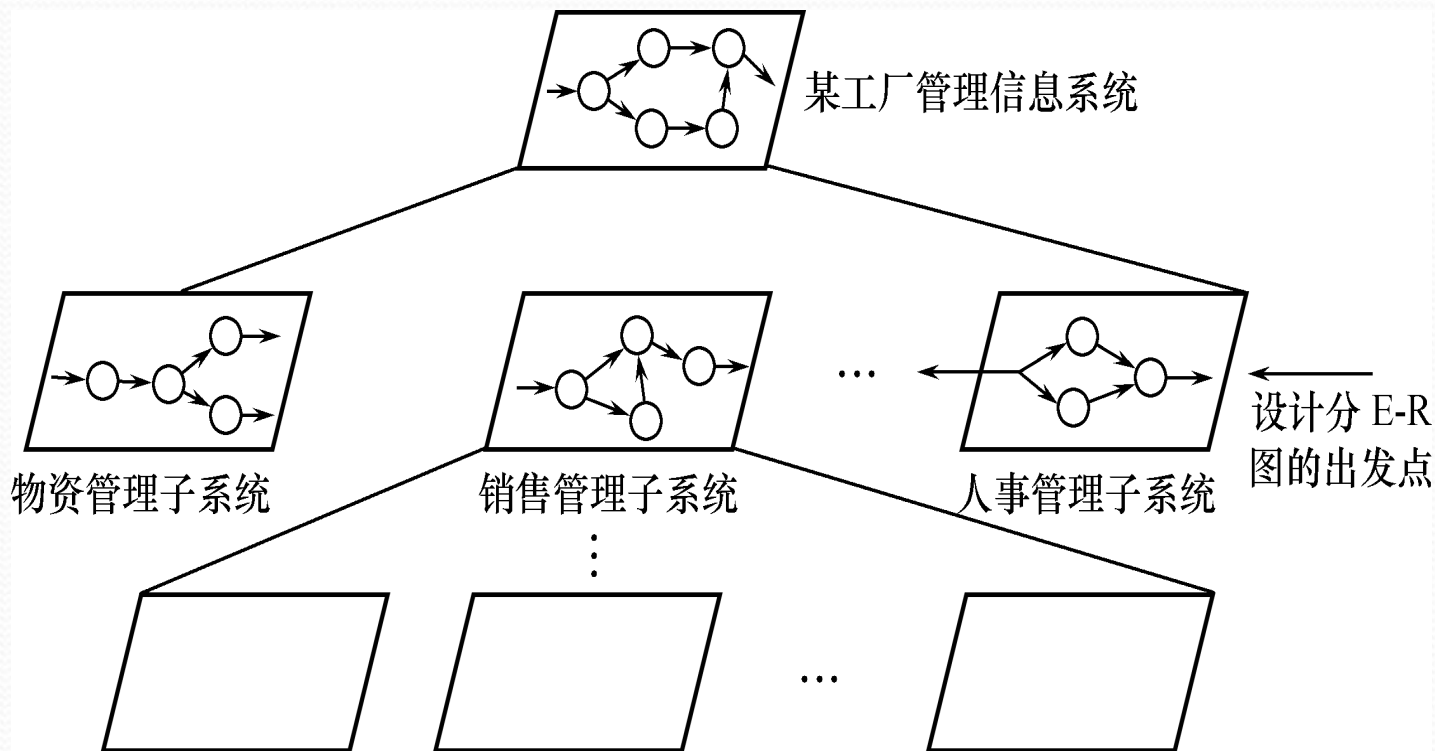
1.选择局部应用

2.逐一设计分E-R图

1. 选择局部应用

- 在多层的数据流图中选择一个适当层次的数据流图，
作为设计分E-R图的出发点
- 通常以中层数据流图作为设计分E-R图的依据

选择局部应用（续）



设计分E-R图的出发点

2. 逐一设计分E-R图

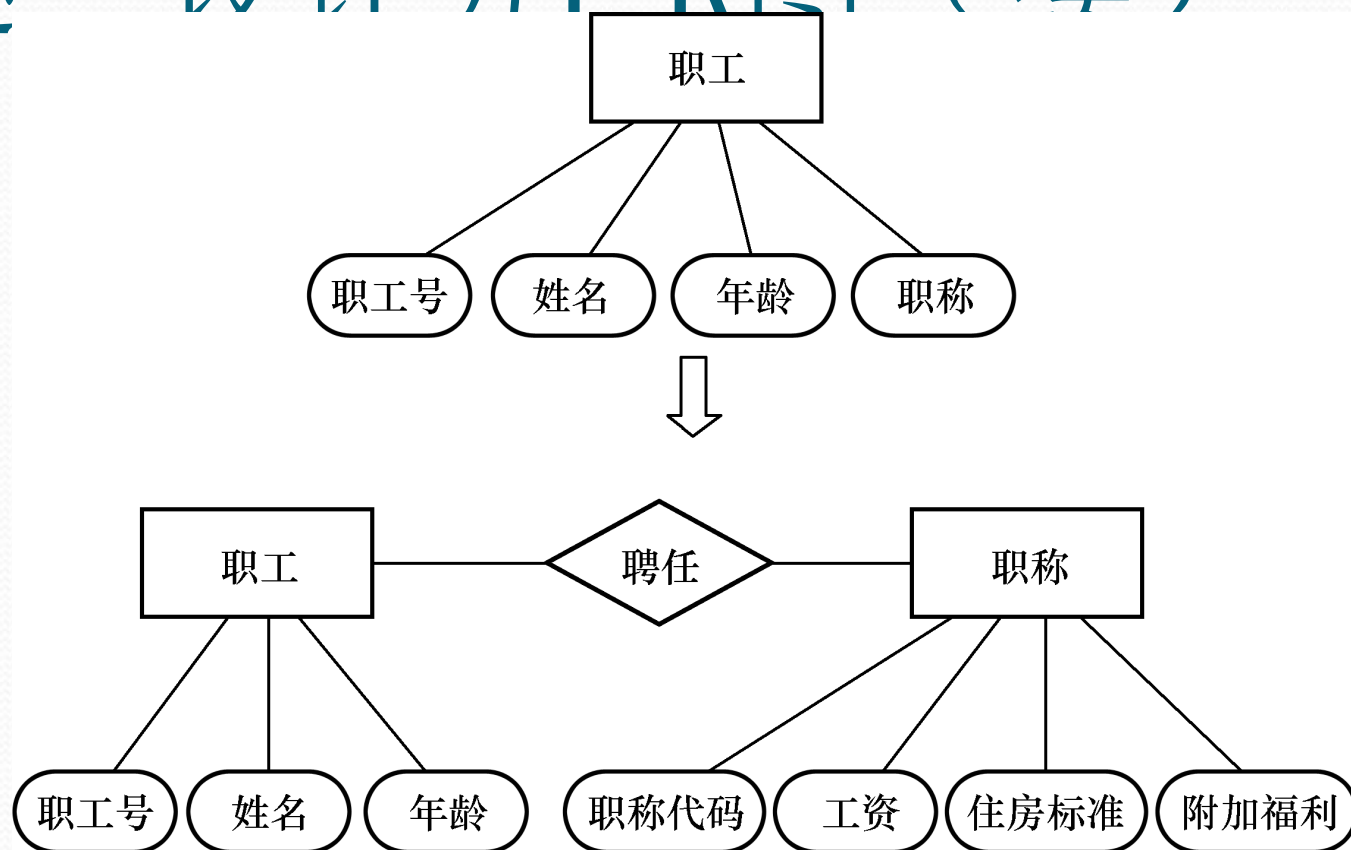
- 任务
 - 将各局部应用涉及的数据分别从数据字典中抽取出来
 - 参照数据流图，标定各局部应用中的实体、实体的属性、标识实体的码
 - 确定实体之间的联系及其类型（1:1， 1:n， m:n）

逐一设计分E-R图（续）

- 两条准则：

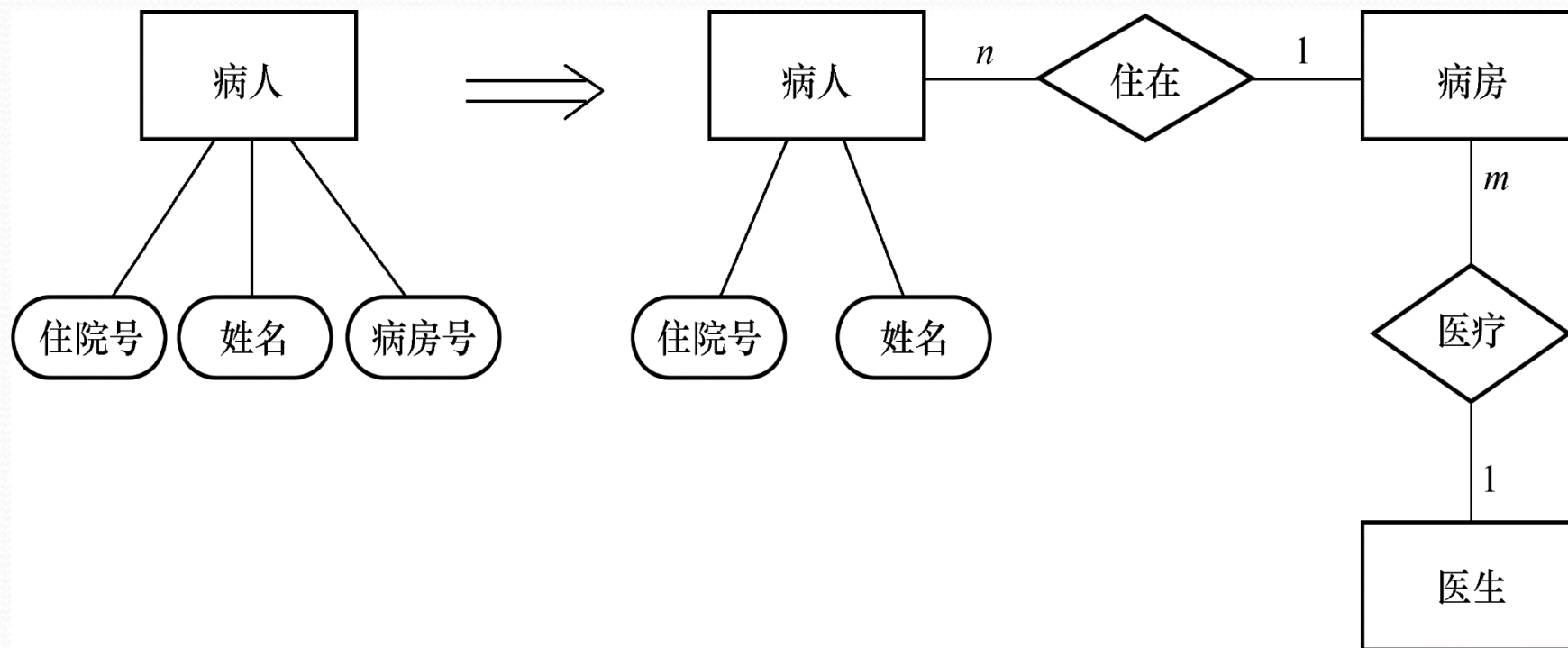
- （1）属性不能再具有需要描述的性质。即属性必须是不可分的数据项，不能再由另一些属性组成
- （2）属性不能与其他实体具有联系。联系只发生在实体之间

逐一设计分F-R图（续）



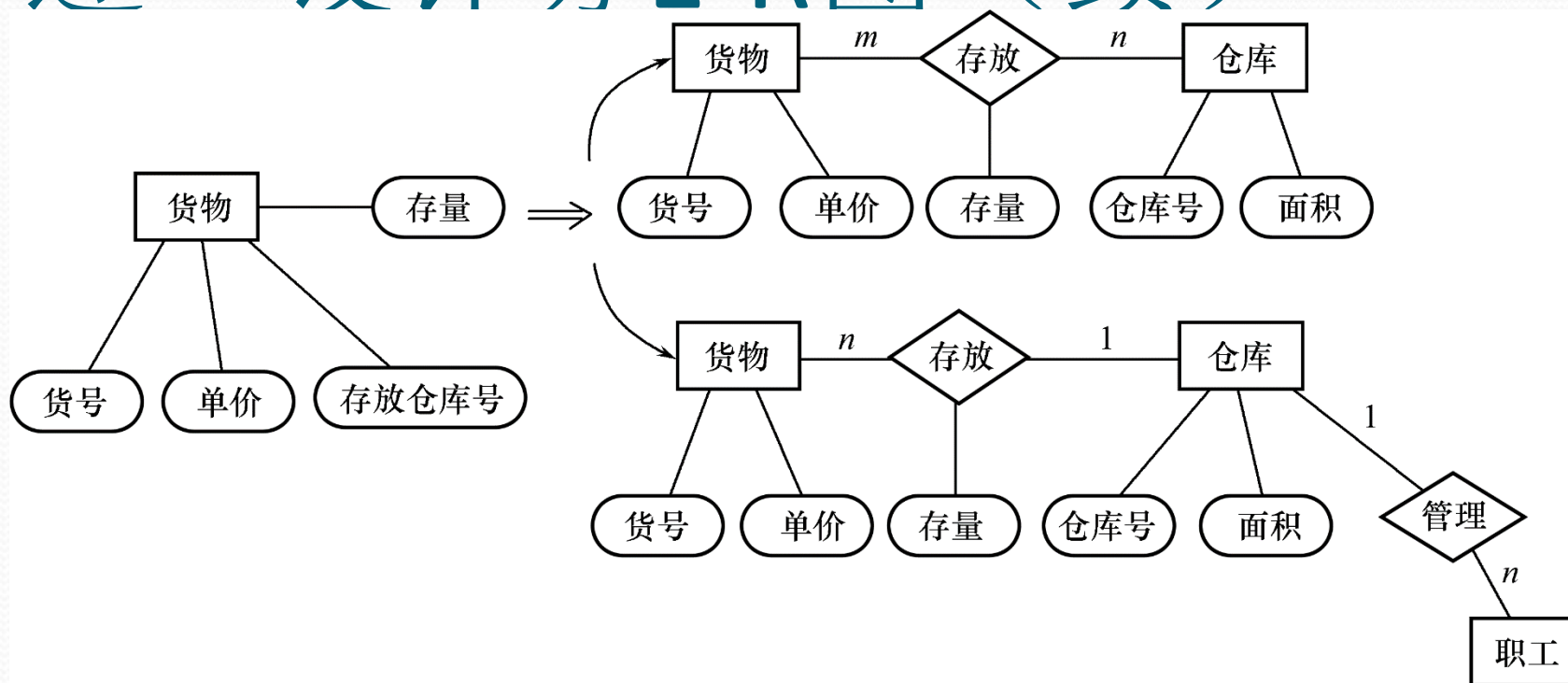
职称作为一个实体

逐一设计分E-R图（续）



病房作为一个实体

逐一设计分E-R图（续）



仓库作为一个实体

逐一设计分E-R图（续）

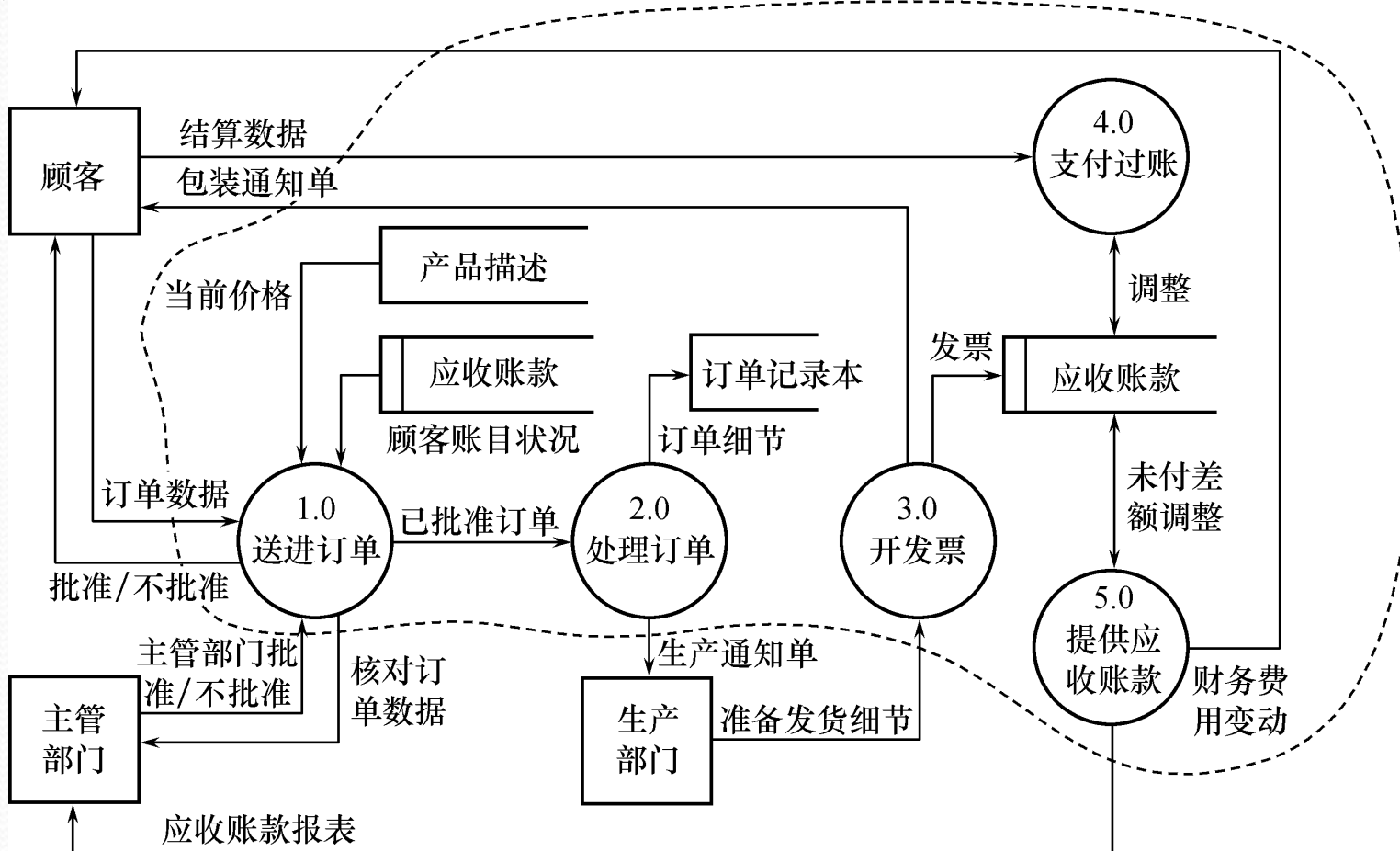
〔实例〕销售管理子系统分E-R图的设计

❖ 销售管理子系统的主要功能：

- 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

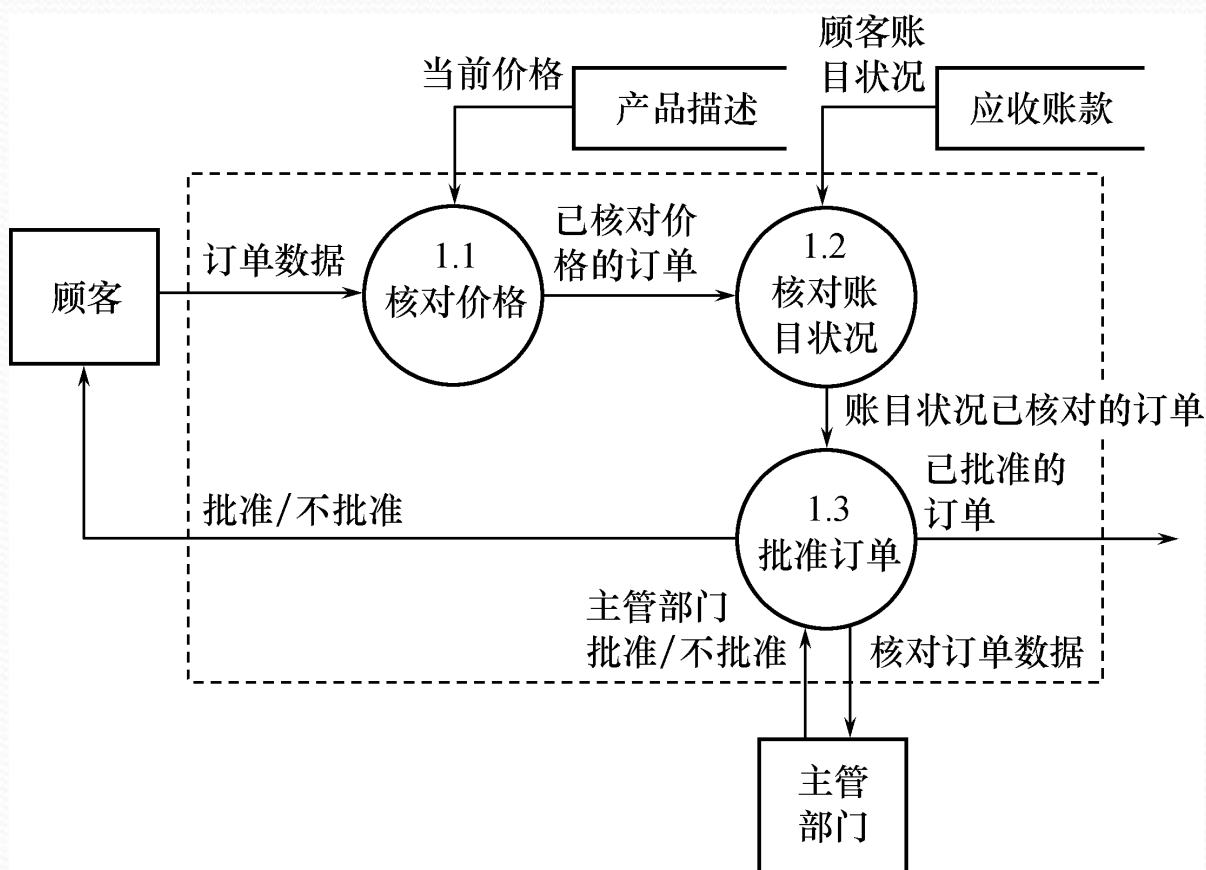
逐一设计分E-R图 (续)

下图是第一层数据流图，虚线部分划出了系统边界

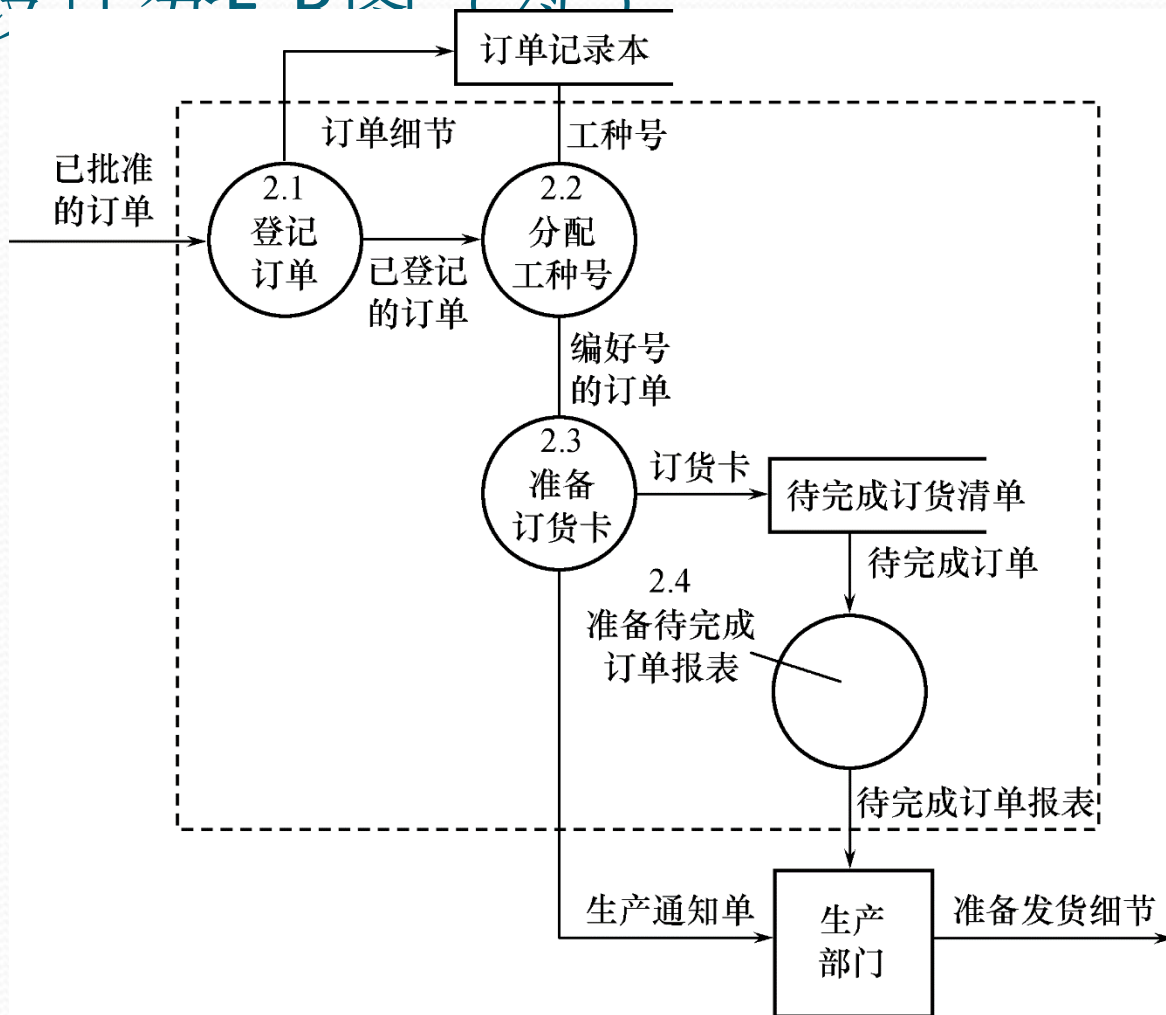


逐一设计分E-R图（续）

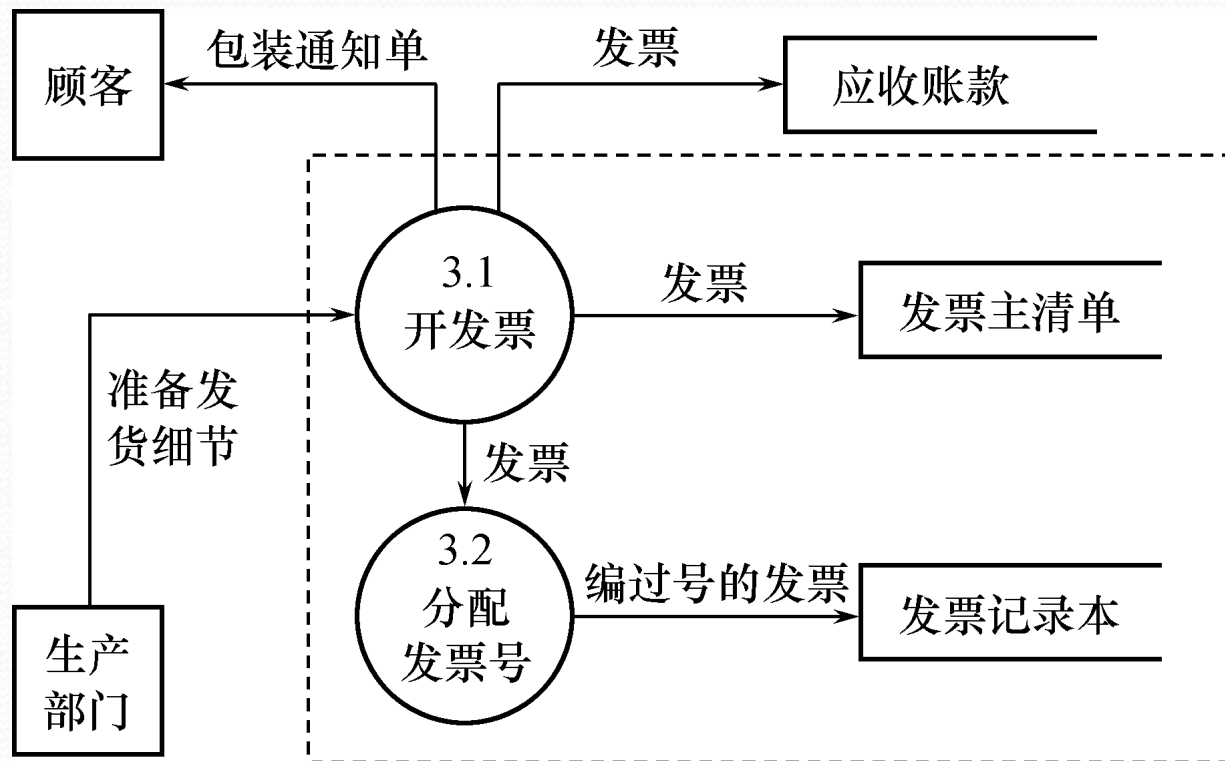
- 上图中把系统功能又分为4个子系统，下面四个图是第二层数据流图



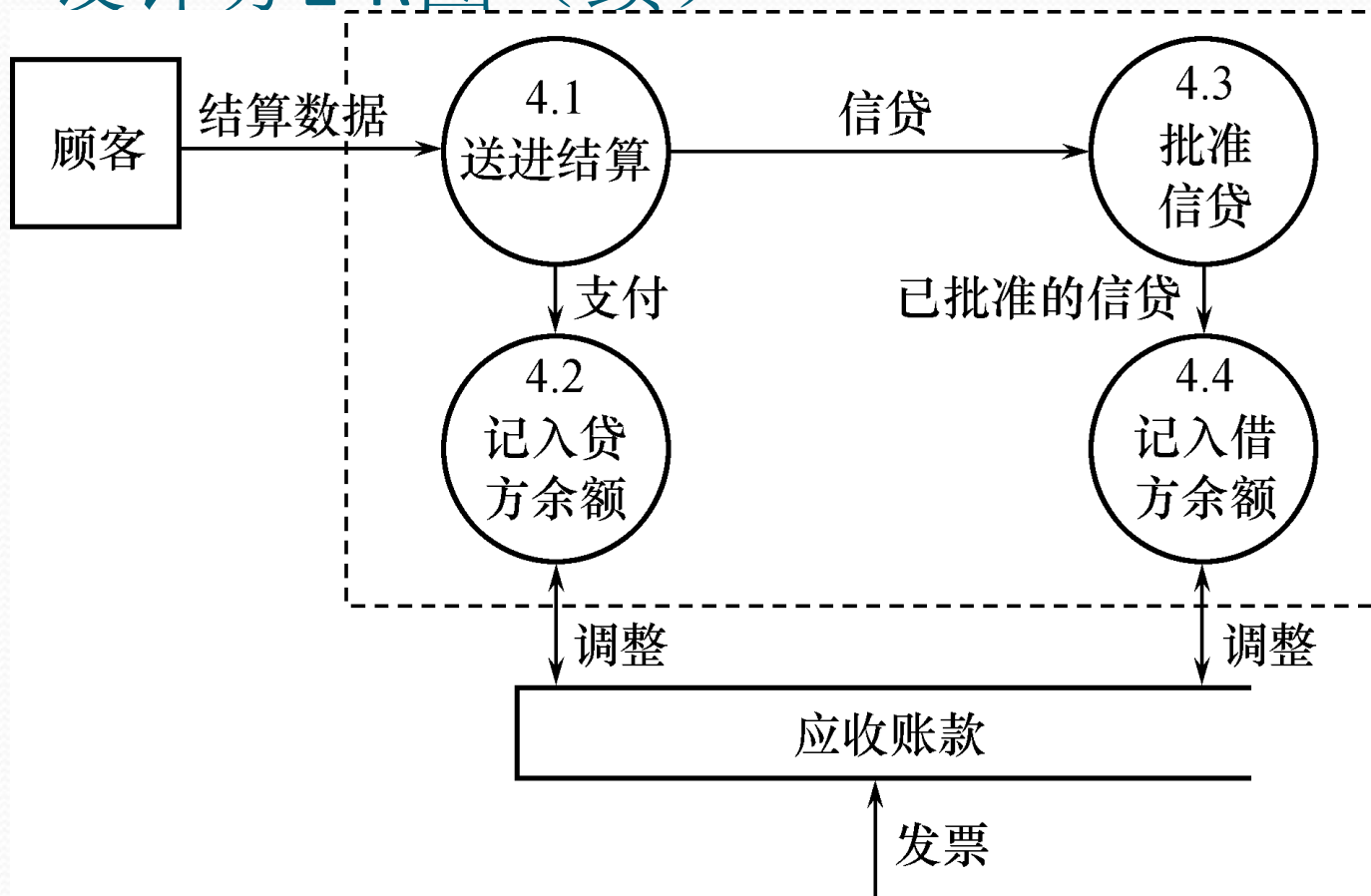
逐一设计分C D图 (续)



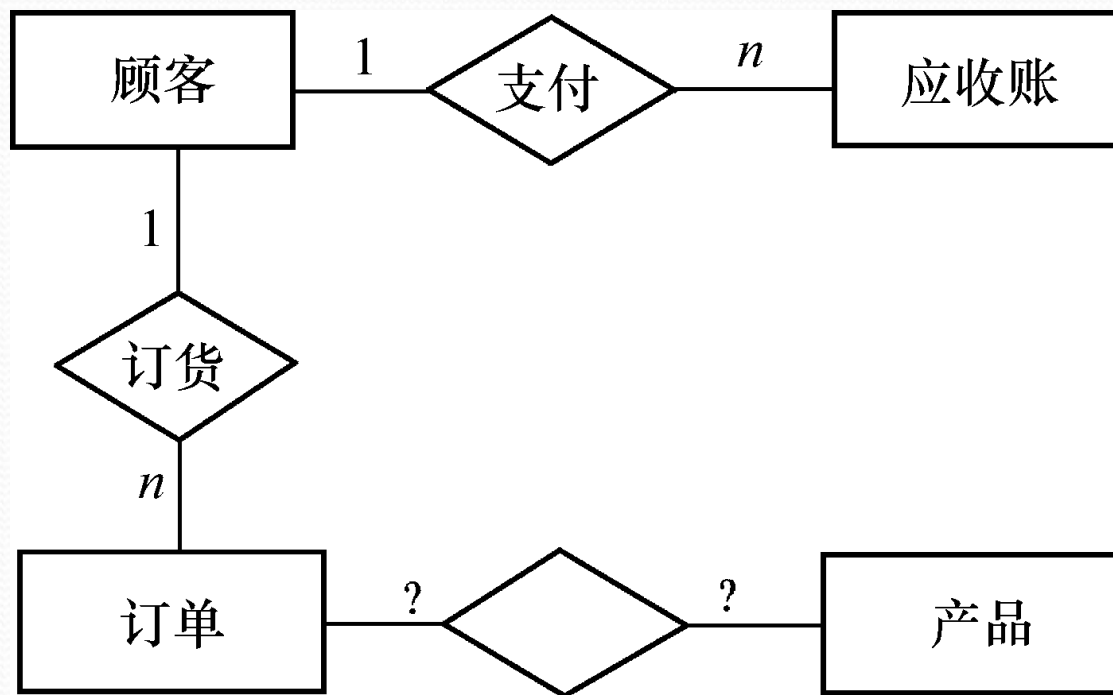
逐一设计分E-R图（续）



逐一设计分E-R图（续）



逐一设计分E-R图（续）



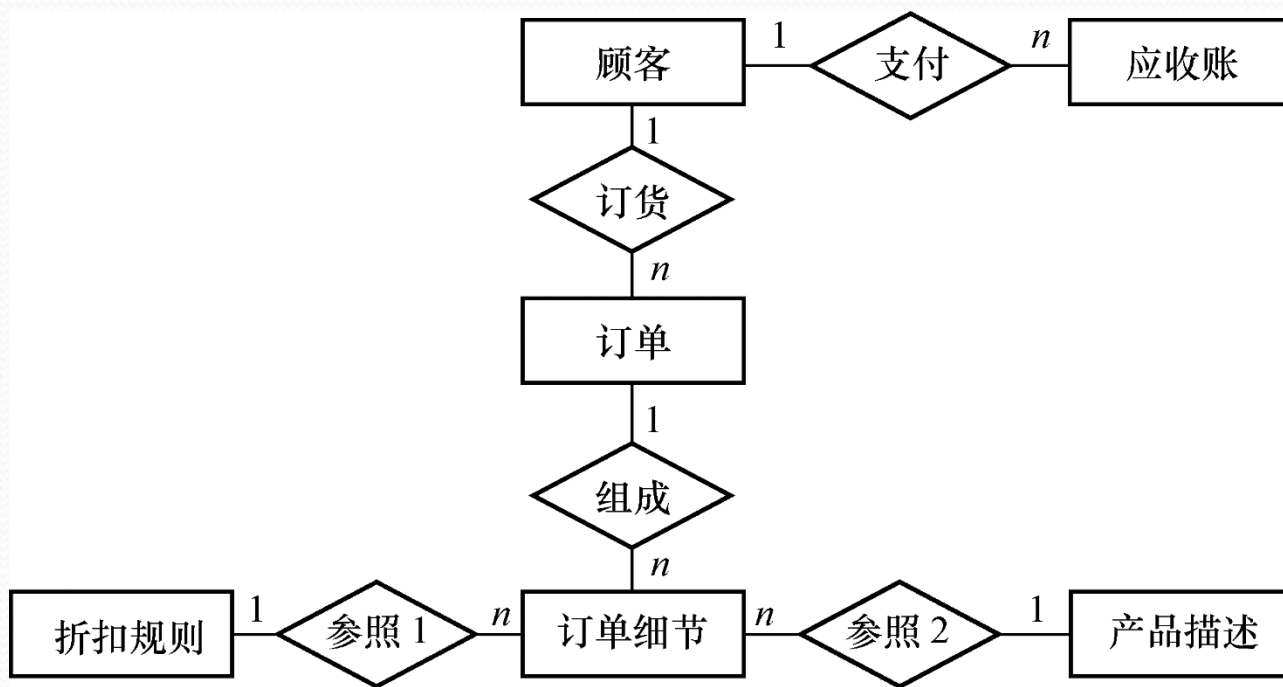
分E-R图的框架

逐一设计分E-R图（续）

- 参照第二层数据流图和数据字典，遵循两个准则，进行如下调整：
 - (1) 订单与订单细节是 $1:n$ 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。
 - (3) “发票主清单”是一个数据存储，不必作为实体加入分E-R图
 - (4) 工厂对大宗订货给予优惠

逐一设计分E-R图（续）

- 得到分E-R图如下图所示



销售管理子系统的分E-R图

逐一设计分E-R图（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，贷款限额}
- 产品描述：{产品号，产品名，单价，重量}
- 折扣规则：{产品号，订货量，折扣}

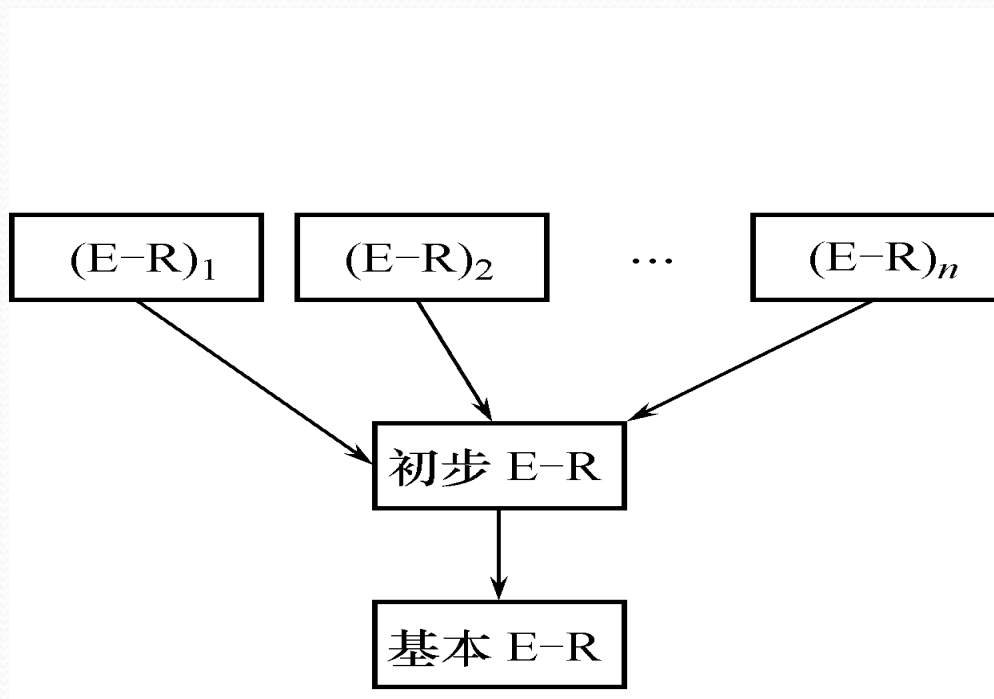
视图的集成

- 各个局部视图即分E-R图建立好后，还需要对它们进行合并，集成为一个整体的数据概念结构即总E-R图。

视图集成的两种方式

- 多个分E-R图一次集成

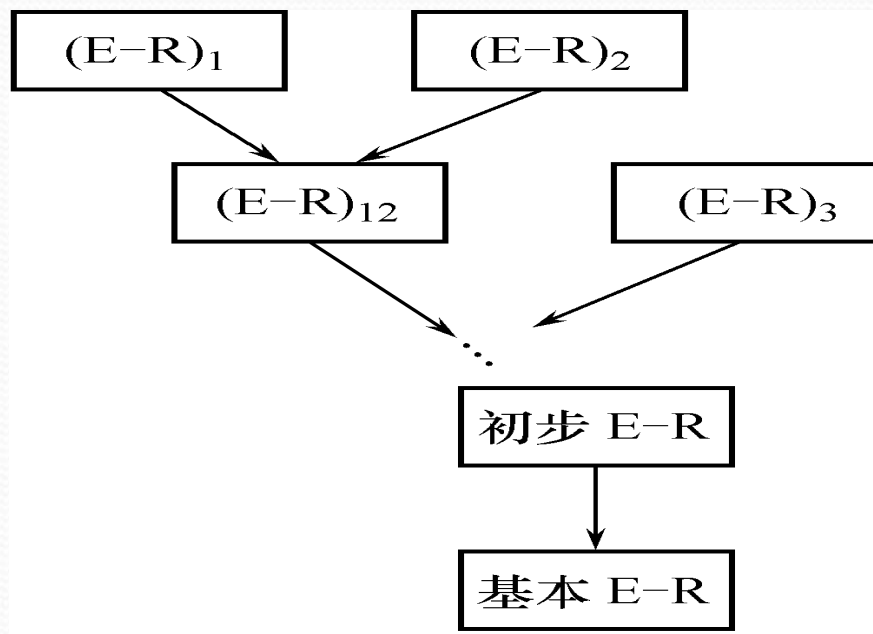
- 一次集成多个分E-R图
- 通常用于局部视图比较简单时



视图的集成（续）

● 逐步集成

- 用累加的方式一次集成两个分E-R图



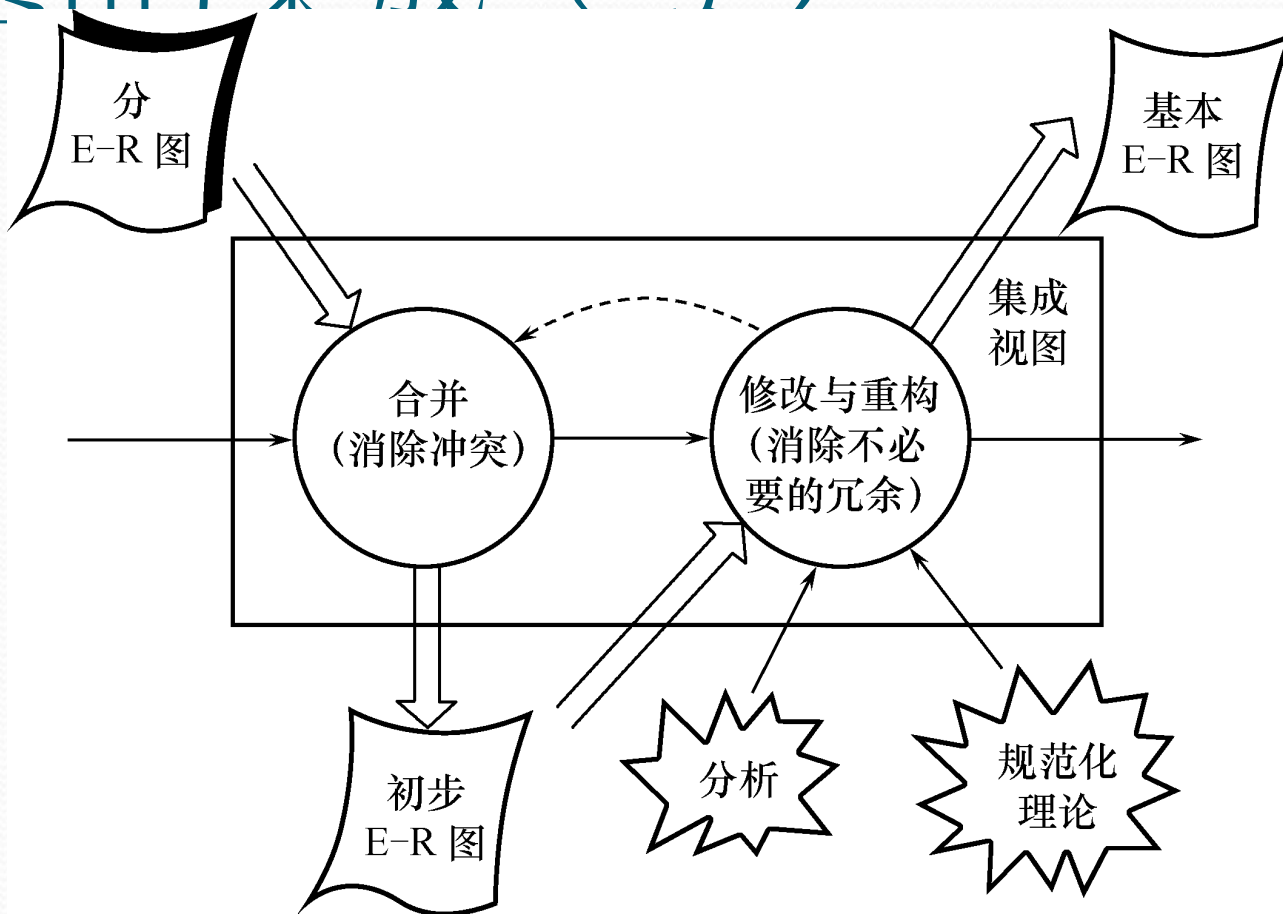
视图的集成（续）

- 集成局部E-R图的步骤

1. 合并

2. 修改与重构

视图的集成 (续)



视图集成

合并分E-R图，生成初步E-R图

- 各分E-R图存在冲突
 - 各个分E-R图之间必定会存在许多不一致的地方
- 合并分E-R图的主要工作与关键
 - 合理消除各分E-R图的冲突

合并分E-R图，生成初步E-R图（续）

- 冲突的种类
 - 属性冲突
 - 命名冲突
 - 结构冲突

属性冲突

- 两类属性冲突
 - 属性域冲突
 - 属性值的类型
 - 取值范围
 - 取值集合不同
 - 属性取值单位冲突

命名冲突

- 两类命名冲突

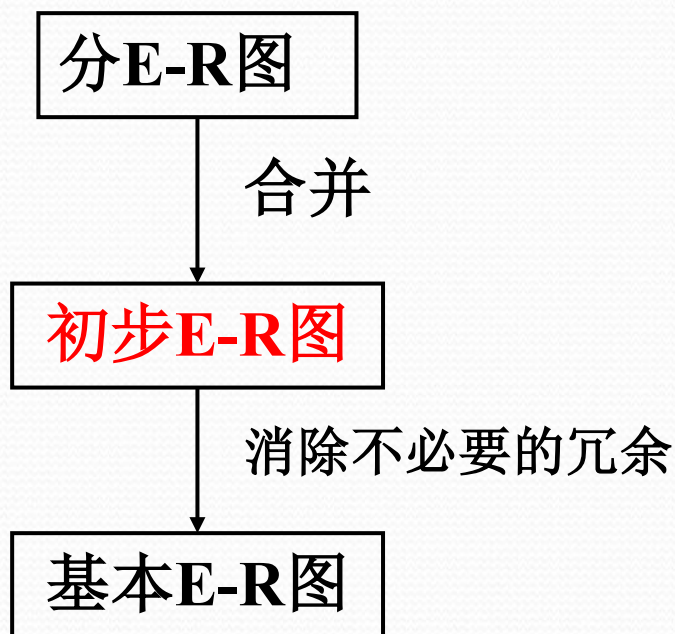
- 同名异义：不同意义的对象在不同的局部应用中具有相同的名字
- 异名同义（一义多名）：同一意义的对象在不同的局部应用中具有不同的名字

结构冲突

- 三类结构冲突
 - 同一对象在不同应用中具有不同的抽象
 - 同一实体在不同分E-R图所包含的属性个数和属性排列次序不完全相同
 - 实体之间的联系在不同局部视图中呈现不同的类型

消除不必要的冗余，设计基本E-R图

- 基本任务
 - 消除不必要的冗余，设计生成基本E-R图



可能存在冗余的数据
和冗余的实体间联系

消除不必要的冗余，设计基本E-R图（续）

- 冗余
- 消除冗余的方法

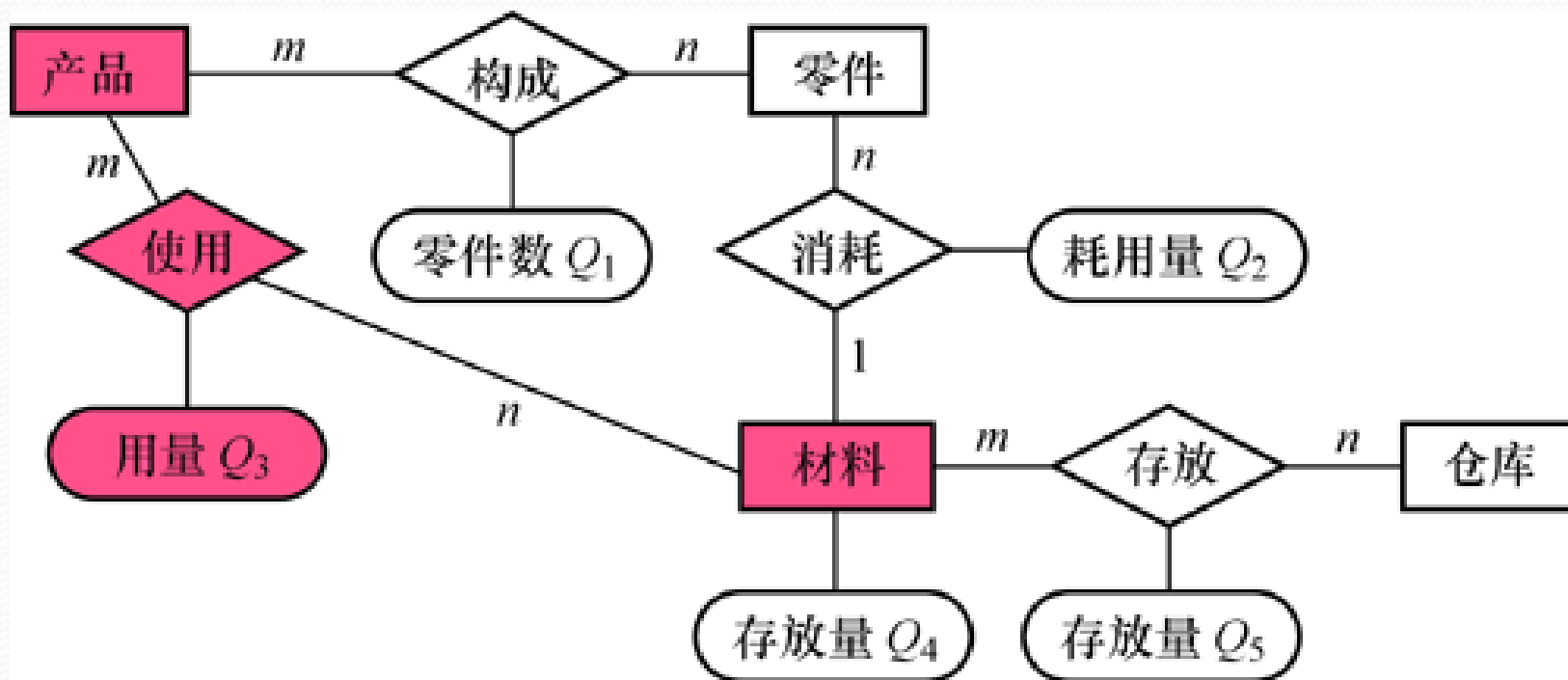
1. 冗余

- 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
- 冗余数据和冗余联系容易破坏数据库的完整性，给数据库维护增加困难
- 消除不必要的冗余后的初步E-R图称为基本E-R图

消除冗余的方法

- 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的逻辑关系

消除冗余的方法（续）



消除冗余

消除冗余的方法（续）

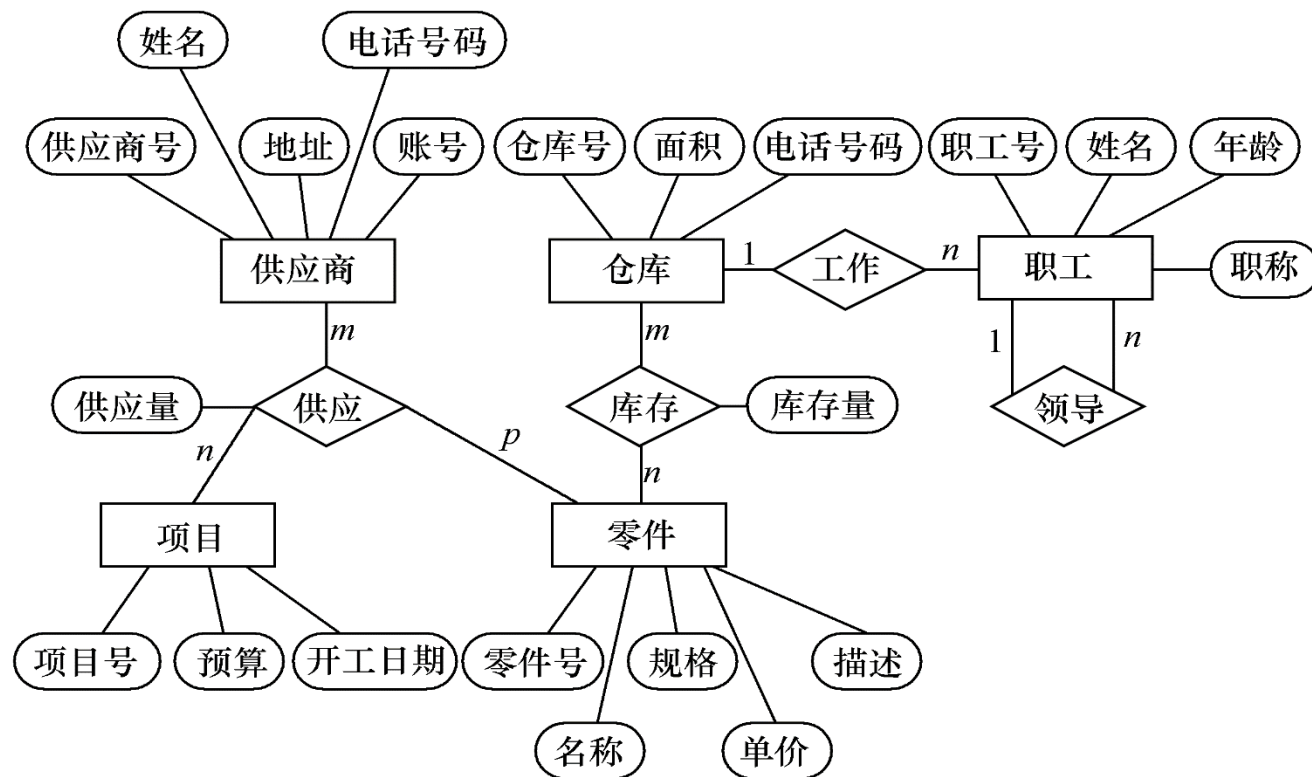
- 效率VS冗余信息
 - 需要根据用户的整体需求来确定
- 若人为地保留了一些冗余数据，则应把数据字典中数据关联的说明作为完整性约束条件
 - $Q_4 = \Sigma Q_5$
 - 一旦 Q_5 修改后就应当触发完整性检查，对 Q_4 进行修改

消除冗余的方法（续）

- 规范化理论
 - 函数依赖的概念提供了消除冗余联系的形式化工具

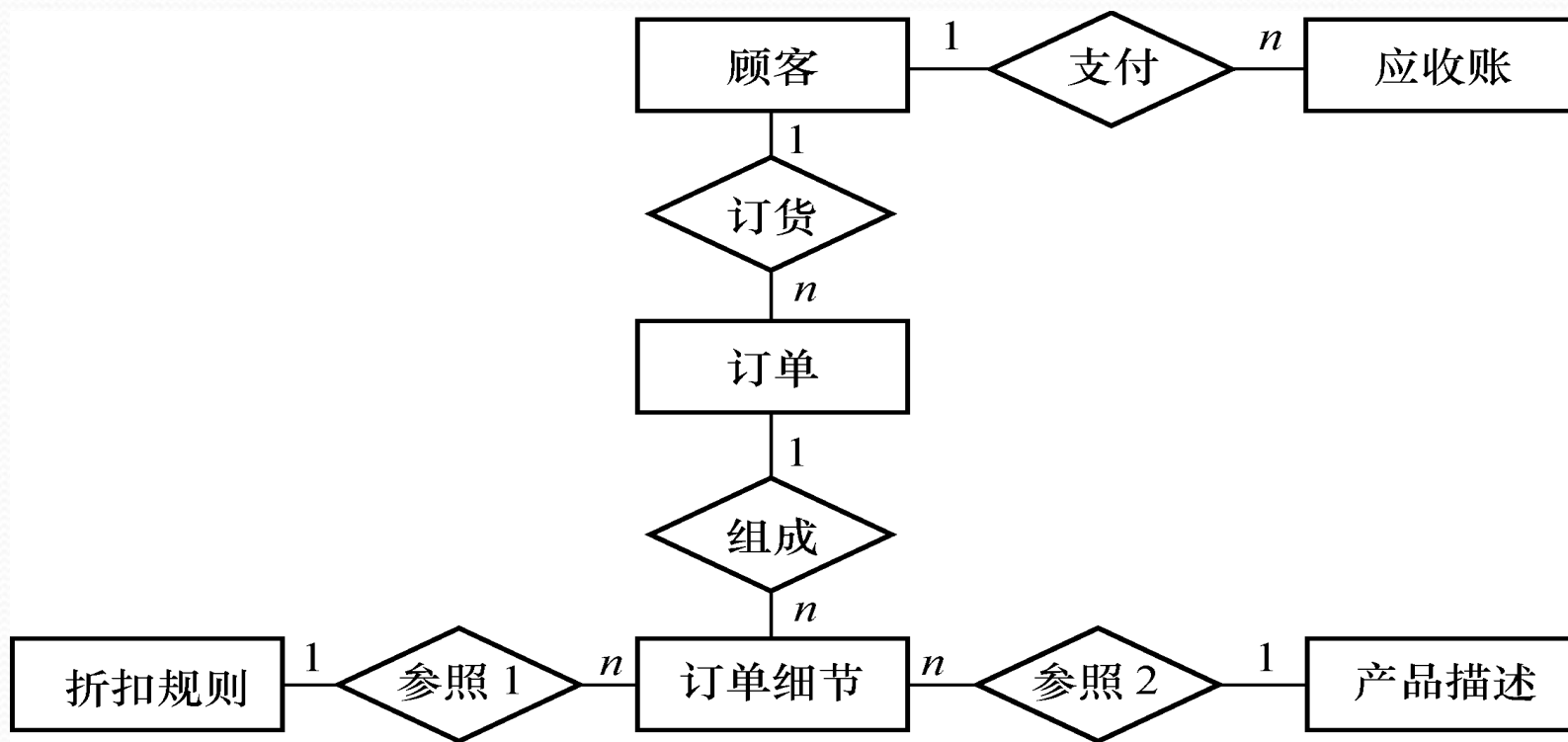
消除冗余，设计生成基本E-R图实例

消除冗余设计生成基本E-R图实例（续）



工厂物资管理E-R图

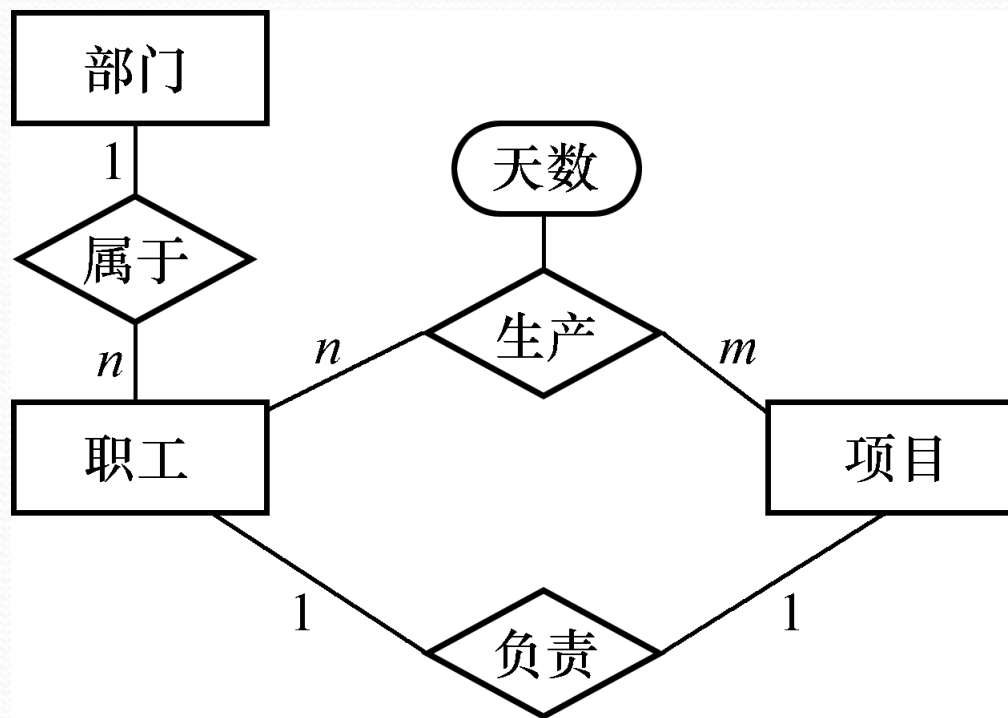
消除冗余设计生成基本E-R图实例（续）



销售管理子系统的分E-R图

消除冗余，设计生成基本E-R图实例（续）

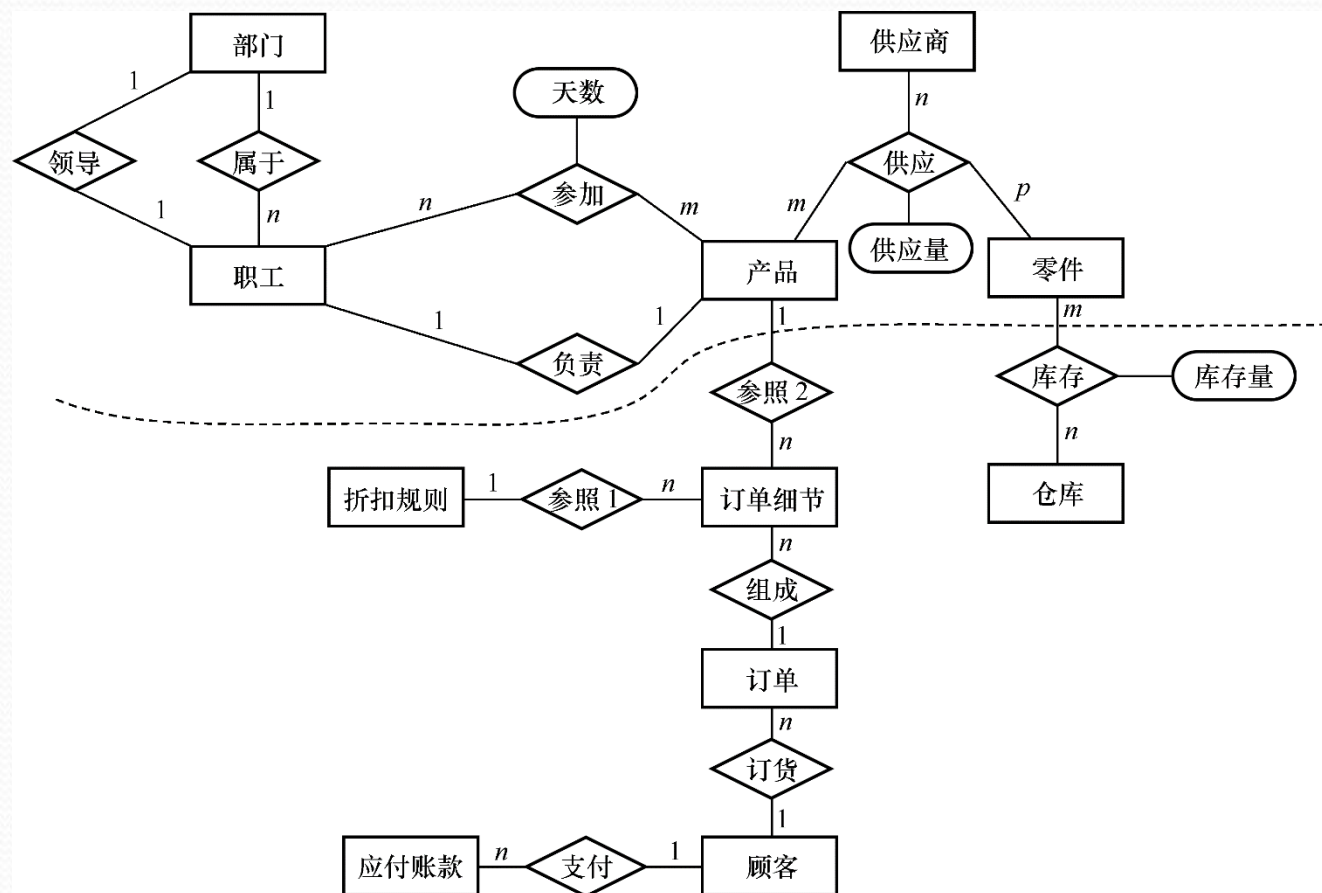
该厂劳动人事管理分E-R图



劳动人事管理的分E-R图

系统的基本E-R

消除冗余，设计生成基本E-R图实例（续）



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
- 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

验证整体概念结构（续）

- 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。

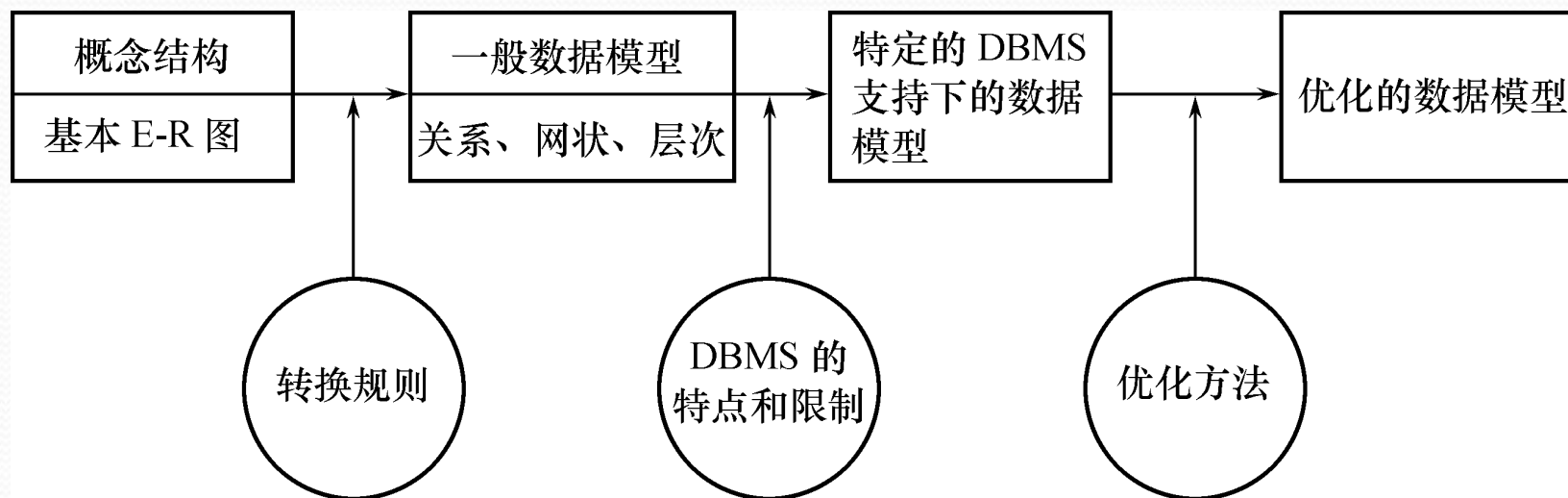
概念结构设计小结

- 概念结构设计的步骤
 - 抽象数据并设计局部视图
 - 集成局部视图，得到全局概念结构
 - 验证整体概念结构

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
- 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

逻辑结构设计(续)



逻辑结构设计时的3个步骤

E-R图向关系模型的转换

- 转换内容
- 转换原则

E-R图向关系模型的转换（续）

- 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
- 转换内容
 - 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。

E-R图向关系模型的转换（续）

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
- 与某一端实体对应的关系模式合并

(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
- 与n端对应的关系模式合并

E-R图向关系模型的转换（续）

(3) 一个 $m:n$ 联系转换为一个关系模式。

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）

E-R图向关系模型的转换（续）

(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。

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）

E-R图向关系模型的转换（续）

(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）

E-R图向关系模型的转换（续）

注意：

- 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。

E-R图向关系模型的转换（续）

[例] 部分E-R图转换为关系模型

- 部门实体对应的关系模式

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

- 职工实体对应的关系模式

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] 把部分E-R图转换为关系模型（续）

- 产品实体对应的关系模式

产品（产品号，产品名，产品组长的职工号，...）

- 供应商实体对应的关系模式

供应商（供应商号，姓名，...）

- 零件实体对应的关系模式

零件（零件号，零件名，...）

E-R图向关系模型的转换（续）

[例] 把部分E-R图转换为关系模型（续）

- 联系“参加”所对应的关系模式

职工工作（职工号，产品号，工作天数，...）

- 联系“供应”所对应的关系模式

供应（产品号，供应商号，零件号，供应量）

数据模型的优化

- 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

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。

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

数据模型的优化（续）

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。

注意：并不是规范化程度越高的关系就越优，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 \rightarrow 英语

学号 \rightarrow 数学

学号 \rightarrow 语文

学号 \rightarrow 平均成绩

(英语, 数学, 语文) \rightarrow 平均成绩

数据模型的优化（续）

显然有：

学号 \rightarrow (英语,数学,语文)

因此该关系模式中存在传递函数信赖，是2NF关系

虽然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

- 水平分解的适用范围

- 满足“80/20原则”的应用
- 并发事务经常存取不相交的数据

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

- 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

设计用户子模式

- 定义用户外模式时应该注重的问题

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）

[例] 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）

- 顾客视图中只包含允许顾客查询的属性
- 销售部门视图中只包含允许销售部门查询的属性
- 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

逻辑结构设计小结

- 任务
 - 将概念结构转化为具体的数据模型
- 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
 - 将转化来的关系、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
 - 设计用户子模式