



数据库系统概论

An Introduction to Database System

数据库设计

数据库设计

- 1 数据库设计概述
- 2 需求分析
- 3 概念结构设计
- 4 逻辑结构设计
- 5 小结

1 数据库设计概述

1.1 数据库设计的特点

1.2 数据库设计方法

1.3 数据库设计过程中的各级模式

1.4 数据库设计的基本步骤

1.1 数据库设计的特点

❖ 数据库建设的基本规律

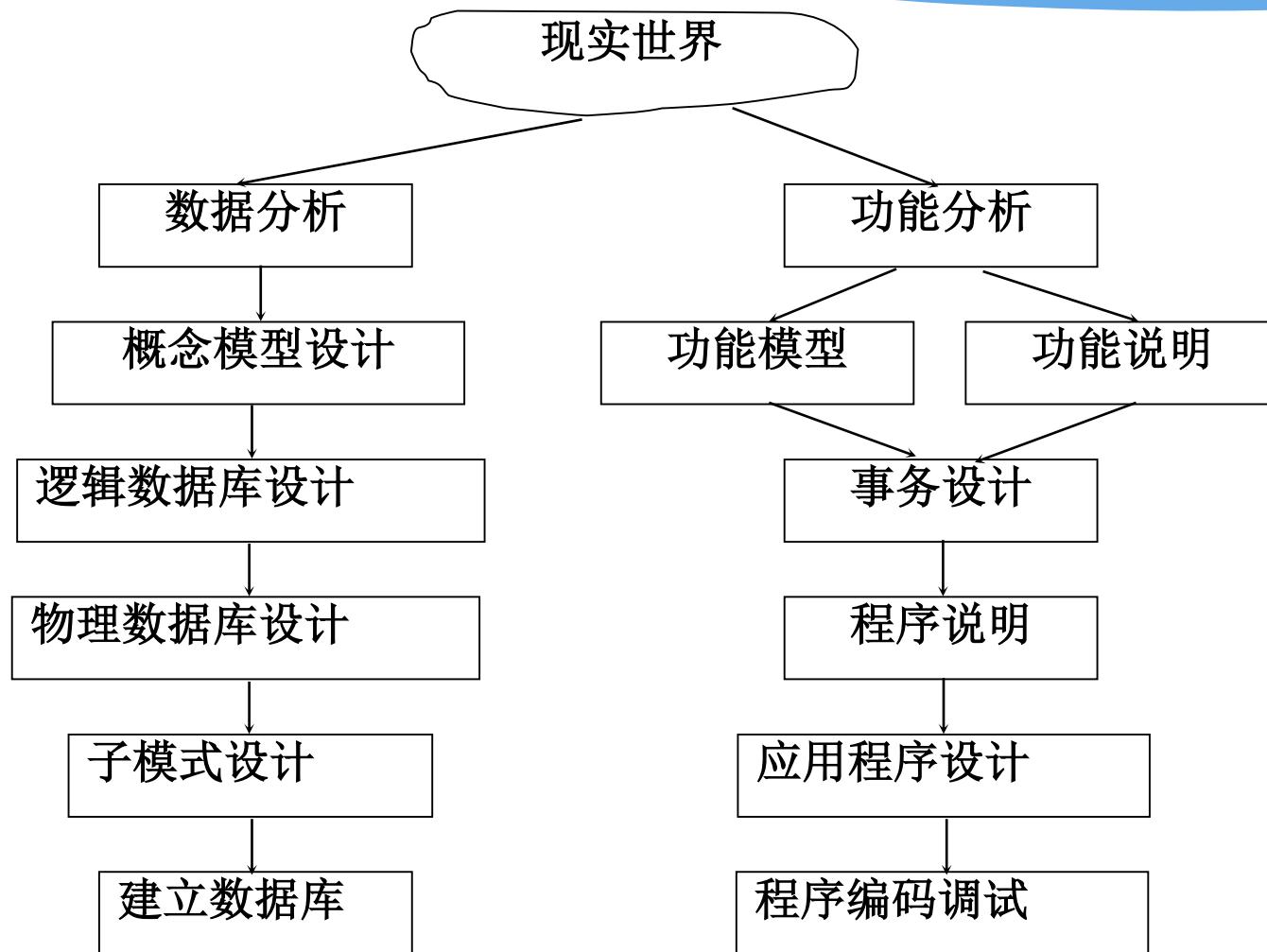
- 三分技术，七分管理，十二分基础数据
- 管理
 - 数据库建设项目管理
 - 企业（即应用部门）的业务管理
- 基础数据
 - 收集、入库
 - 更新新的数据

❖ 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

❖ 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



1.2 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

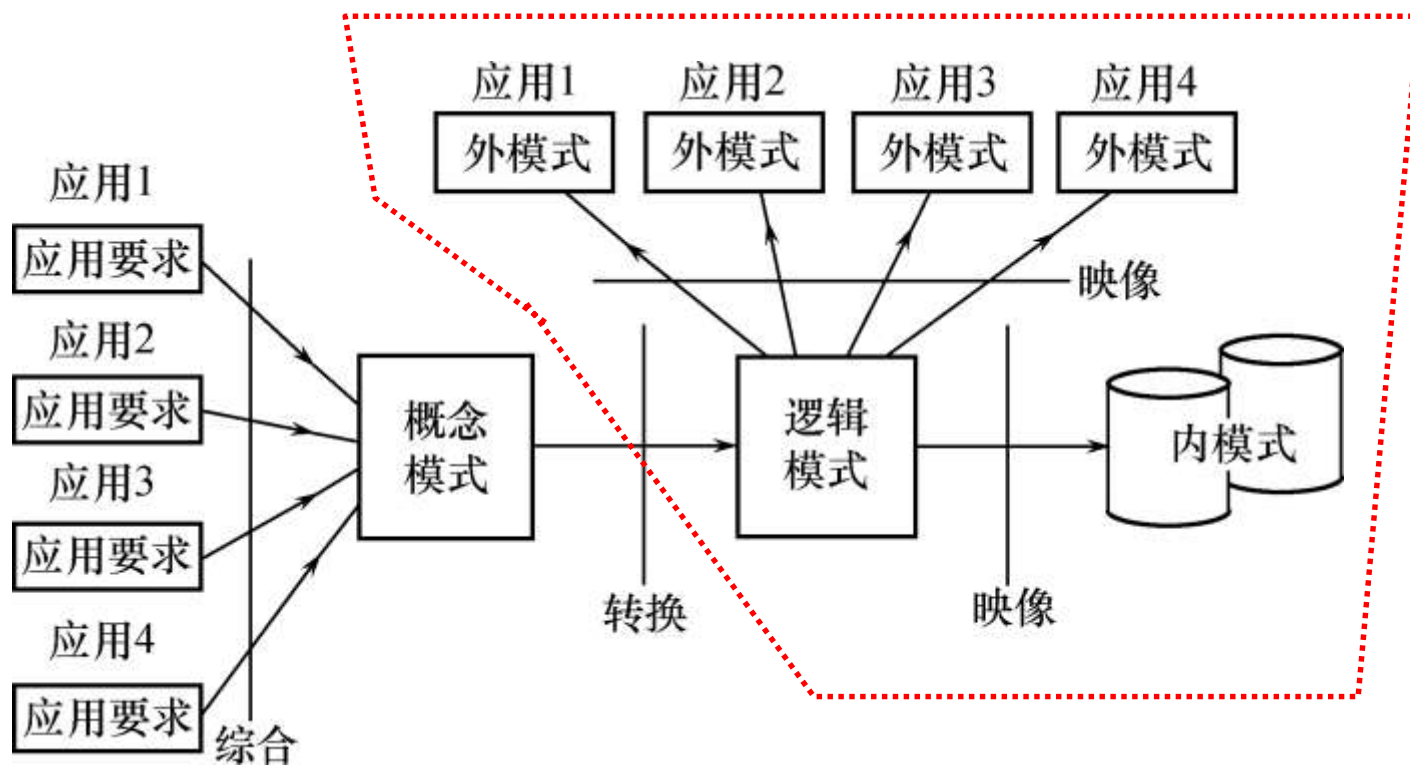
数据库设计方法（续）

❖ 计算机辅助设计

- **ORACLE Designer 2000**
- **SYBASE PowerDesigner**

1.3 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

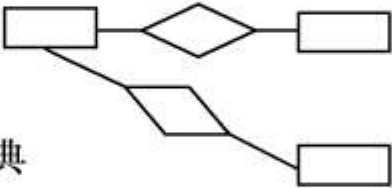
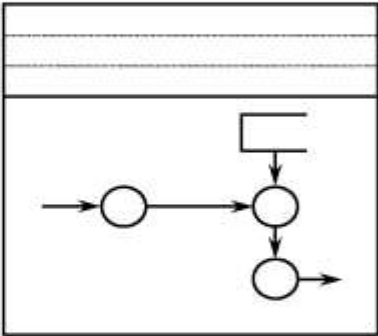

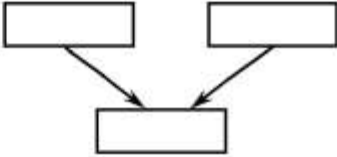
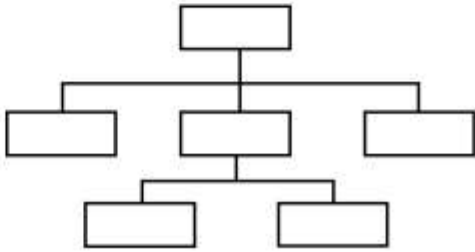
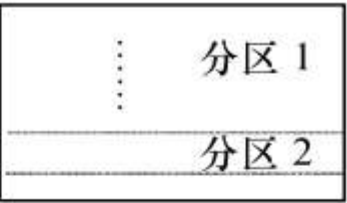
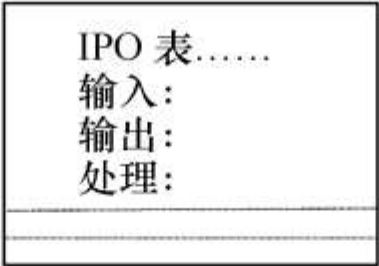
1.4 数据库设计的基本步骤

阶段	参与者
需求分析	系统分析人员、数据库设计人员、用户、数据库管理员
概念结构设计	系统分析人员、数据库设计人员
逻辑结构设计	系统分析人员、数据库设计人员
物理结构设计	系统分析人员、数据库设计人员
数据库实施	系统分析人员、数据库设计人员、应用开发人员
数据库运行和维护	系统分析人员、数据库设计人员、用户、数据库管理员

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	<p>概念模型（E-R图）</p>  <p>数据字典</p>	<p>系统说明书包括：</p> <p>① 新系统要求、方案和概图</p> <p>② 反映新系统信息流的数据流图</p> 
逻辑结构设计	<p>某种数据模型</p> <p>关系</p>  <p>非关系</p> 	<p>系统结构图</p> <p>（模块结构）</p> 
物理设计	<p>存储安排 方法选择 存取路径建立</p> 	<p>模块设计 IPO 表</p> 

2 需求分析

3 概念结构设计

3.1 概念模型与ER图

3.2 概念结构设计的方法与步骤

3.3 局部视图设计

3.4 视图的集成

3.1.概念模型与ER图

- ❖ 3.1.1 两大类数据模型
- ❖ 3.1.2 数据模型的组成要素
- ❖ 3.1.3 概念模型与ER图

3.1.1 两大类数据模型

❖ 数据模型分为两类（分属两个不同的层次）

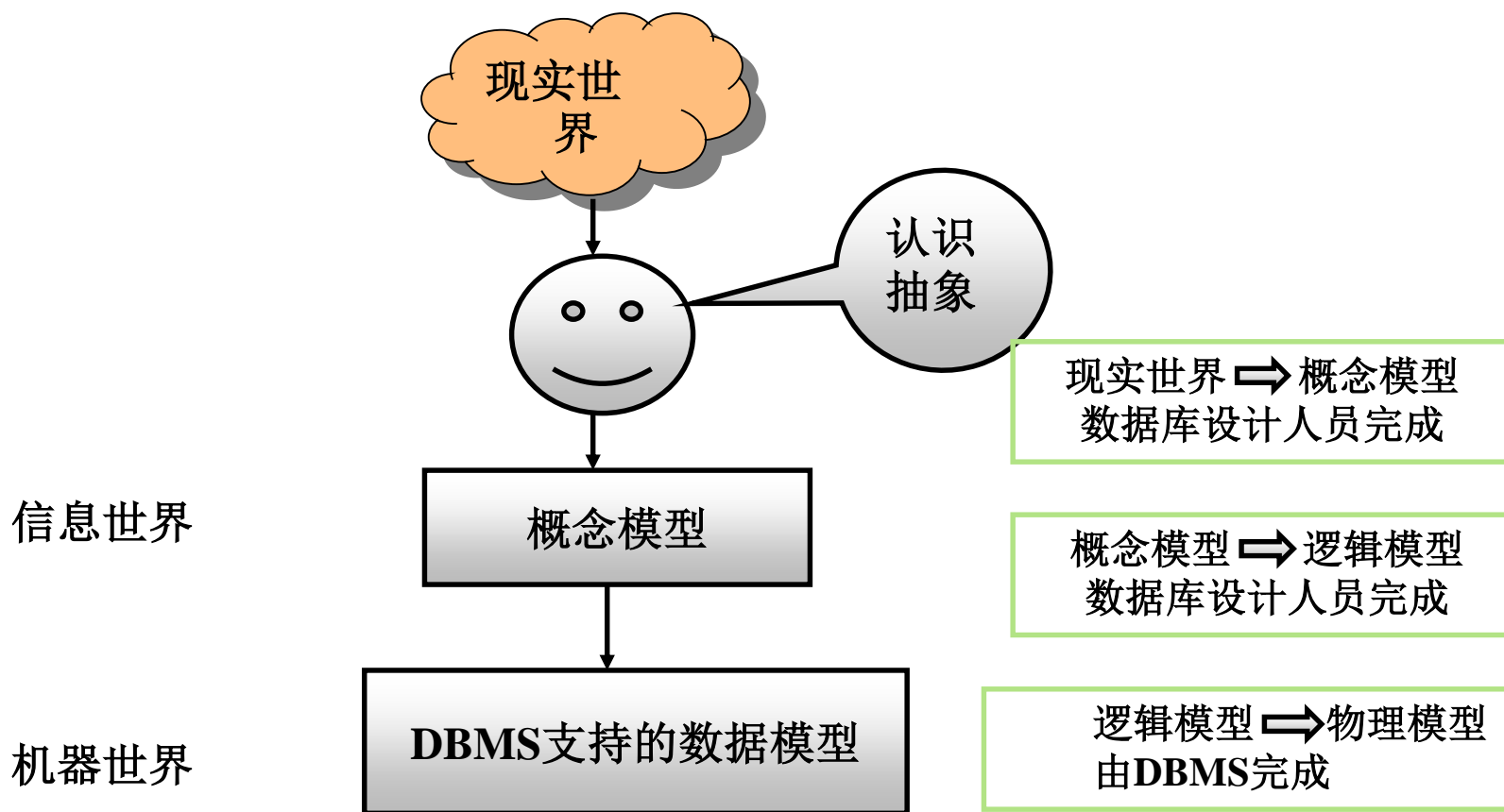
(1) 概念模型

- 也称信息模型，它是按用户的观点来对数据和信息建模，用于数据库设计。

(2) 逻辑模型和物理模型

- 逻辑模型主要包括网状模型、层次模型、关系模型、面向对象模型等，按计算机系统的观点对数据建模，用于**DBMS**实现。
- 物理模型是对数据最底层的抽象，描述数据在系统内部的表示方式和存取方法，在磁盘或磁带上的存储方式和存取方法。

两大类数据模型 (续)



现实世界中客观对象的抽象过程

3.1.2 数据模型的组成要素

- ❖ 数据结构
- ❖ 数据操作
- ❖ 完整性约束条件

一、数据结构

❖ 什么是数据结构

- 描述数据库的组成对象，以及对象之间的联系

❖ 描述的内容

- 与数据类型、内容、性质有关的对象
- 与数据之间联系有关的对象

❖ 数据结构是对系统静态特性的描述

二、数据操作

❖ 数据操作

- 对数据库中各种对象(型)的实例(值)允许执行的
操作及有关的操作规则

❖ 数据操作的类型

- 查询
- 更新(包括插入、删除、修改)

❖ 数据操作是对系统动态特性的描述

三、数据的完整性约束条件

❖ 数据的完整性约束条件

- 一组完整性规则的集合。
- 完整性规则：给定的数据模型中数据及其联系所具有的制约和储存规则
- 用以限定符合数据模型的数据库状态以及状态的变化，以保证数据的正确、有效、相容。

3.1.3 概念模型

❖ 概念模型的用途

- 概念模型用于信息世界的建模
- 是现实世界到机器世界的一个中间层次
- 是数据库设计的有力工具
- 数据库设计人员和用户之间进行交流的语言

❖ 对概念模型的基本要求

- 较强的语义表达能力
- 能够方便、直接地表达应用中的各种语义知识
- 简单、清晰、易于用户理解

一、信息世界中的基本概念

- (1) 实体 (**Entity**)
- (2) 属性 (**Attribute**)
- (3) 码 (**Key**)
- (4) 域 (**Domain**)
- (5) 实体型 (**Entity Type**)
- (6) 实体集 (**Entity Set**)
- (7) 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内部的联系和实体之间的联系。
 - 实体内部的联系通常是指组成实体的各属性之间的联系
 - 实体之间的联系通常是指不同实体集之间的联系

二、两个实体型之间的联系

❖ 一对一联系 (1:1)

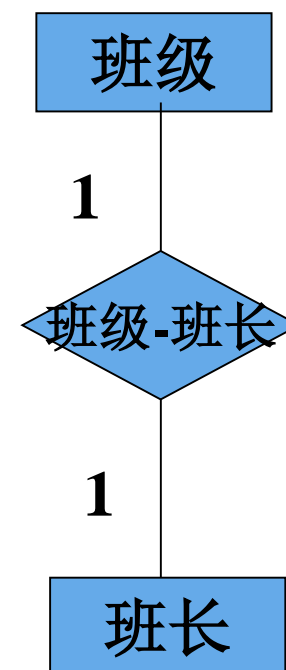
- 实例

一个班级只有一个正班长

一个班长只在一个班中任职

- 定义:

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



1:1联系

两个实体型之间的联系 (续)

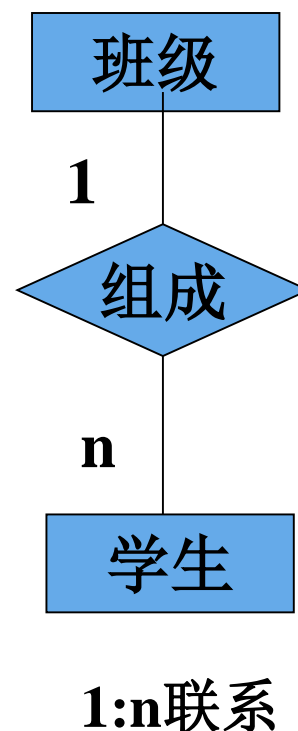
❖ 一对多联系 (1: n)

■ 实例

一个班级中有若干名学生，
每个学生只在一个班级中学习

■ 定义：

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**



两个实体型之间的联系 (续)

❖ 多对多联系 (m:n)

■ 实例

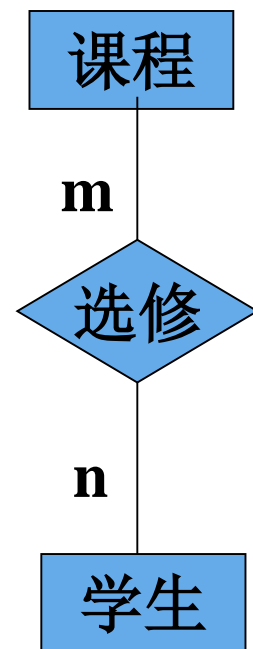
课程与学生之间的联系：

一门课程同时有若干个学生选修

一个学生可以同时选修多门课程

■ 定义：

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**B**具有多对多联系，记为**m:n**



m:n联系

三、两个以上实体型之间的联系

❖ 两个以上实体型之间一对多联系

- 若实体集 E_1, E_2, \dots, E_n 存在联系, 对于实体集 E_j ($j=1, 2, \dots, i-1, i+1, \dots, n$) 中的给定实体, 最多只和 E_i 中的一个实体相联系, 则我们说 E_i 与 $E_1, E_2, \dots, E_{i-1}, E_{i+1}, \dots, E_n$ 之间的联系是一对多的

❖ 实例

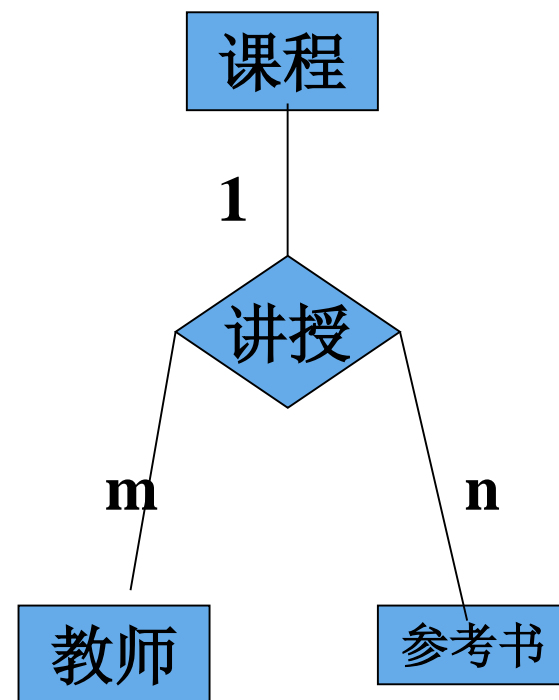
课程、教师与参考书三个实体型

一门课程可以有若干个教师讲授,

使用若干本参考书,

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

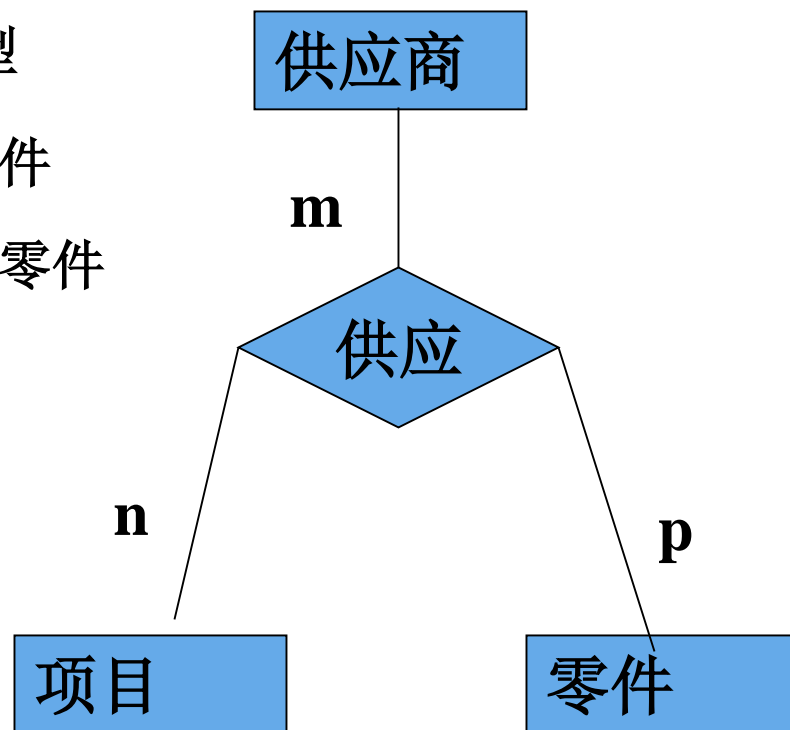


两个以上实体型间1:n联系

两个以上实体型之间的联系(续)

❖ 实例： 供应商、项目、零件三个实体型

- 一个供应商可以供给多个项目多种零件
- 每个项目可以使用多个供应商供应的零件
- 每种零件可由不同供应商供给



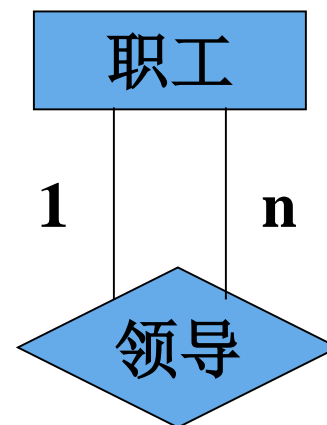
两个以上实体型间m:n联系

四、单个实体型内的联系

❖ 一对多联系

■ 实例

职工实体型内部具有领导与被领导的联系
某一职工（干部）“领导”若干名职工
一个职工仅被另外一个职工直接领导
这是一对多的联系



❖ 一对一联系

请举例

❖ 多对多联系

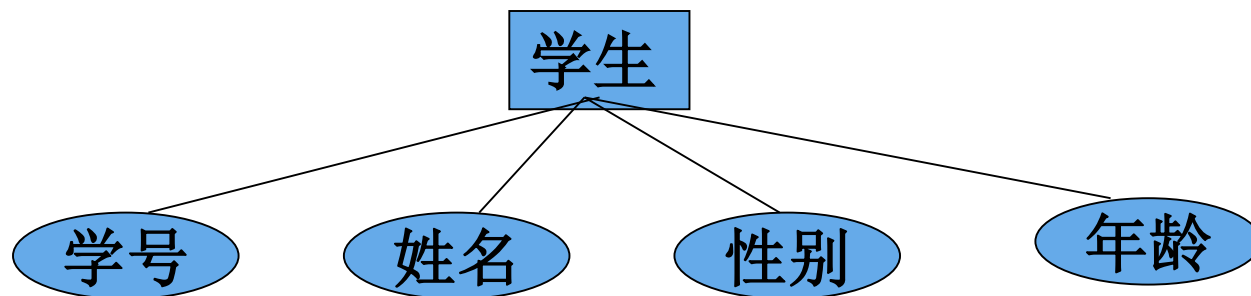
请举例

单个实体型内部
1:n联系

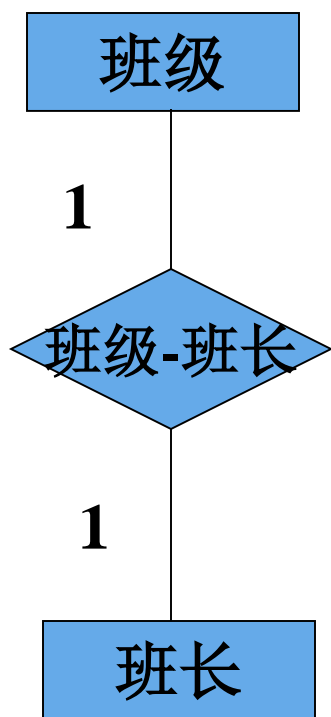
五、概念模型的一种表示方法

❖ 实体—联系方法(E-R方法)

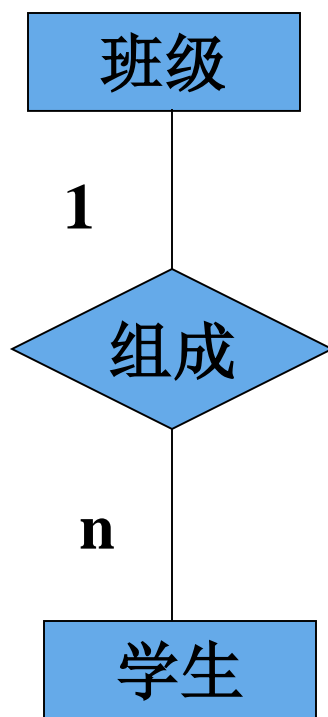
- 用E-R图来描述现实世界的概念模型
- E-R方法也称为E-R模型



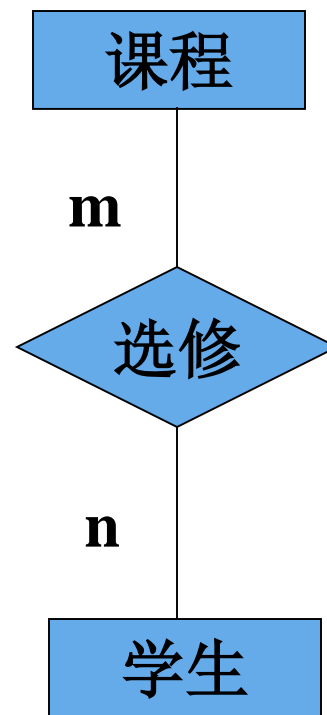
联系的表示方法示例



1:1联系



1:n联系

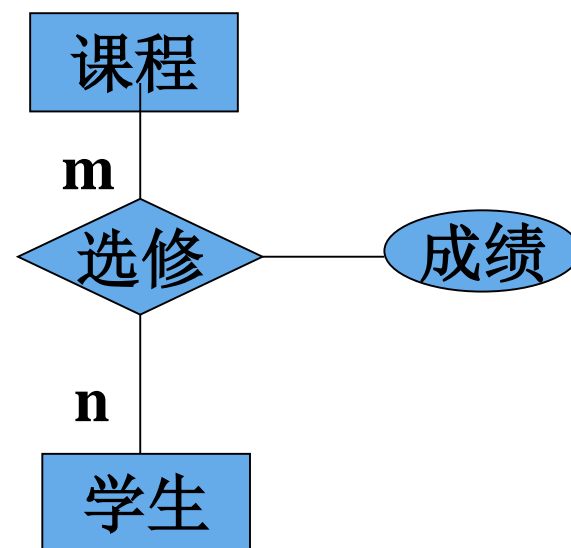


m:n联系

联系的属性

❖ 联系的属性:

联系本身也是一种实体型，也可以有属性。如果一个联系具有属性，则这些属性也要用无向边与该联系连接起来



六、一个实例

用E-R图表示某个工厂物资管理的概念模型

❖ 实体

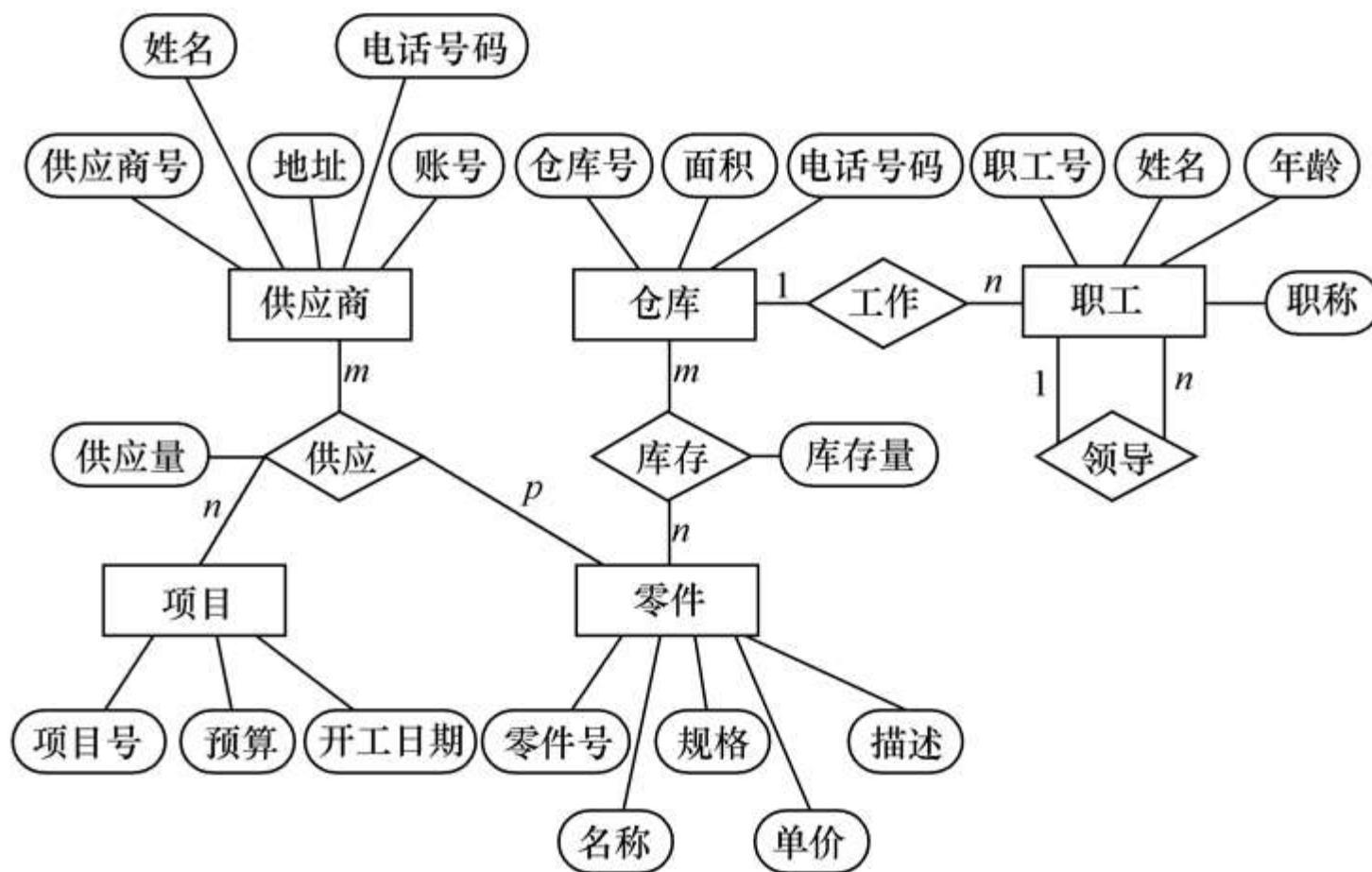
- 仓库： 仓库号、面积、电话号码
- 零件： 零件号、名称、规格、单价、描述
- 供应商： 供应商号、姓名、地址、电话号码、帐号
- 项目： 项目号、预算、开工日期
- 职工： 职工号、姓名、年龄、职称

一个实例

❖ 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。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，仓库和职工之间是一对多的联系。职工实体型中具有一对多的联系
- (3) 职工之间具有领导-被领导关系。即仓库主任领导若干保管员。
- (4) 供应商、项目和零件三者之间具有多对多的联系

一个实例

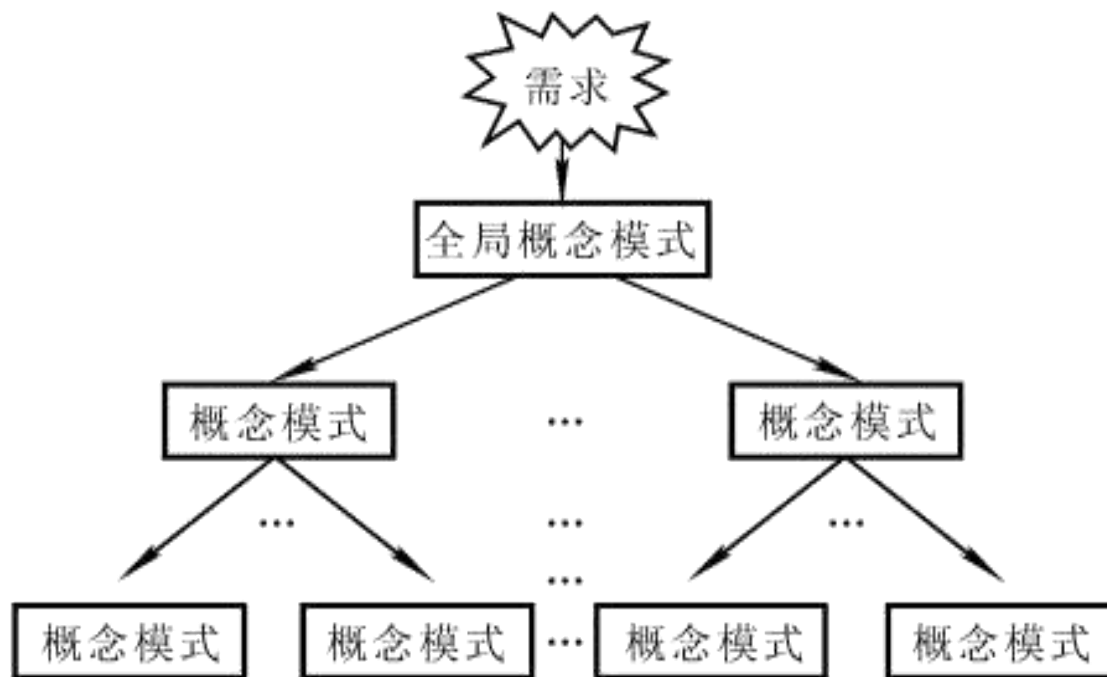


(c) 完整的实体-联系图

3.2 概念结构设计的方法与步骤

❖ 设计概念结构的四类方法

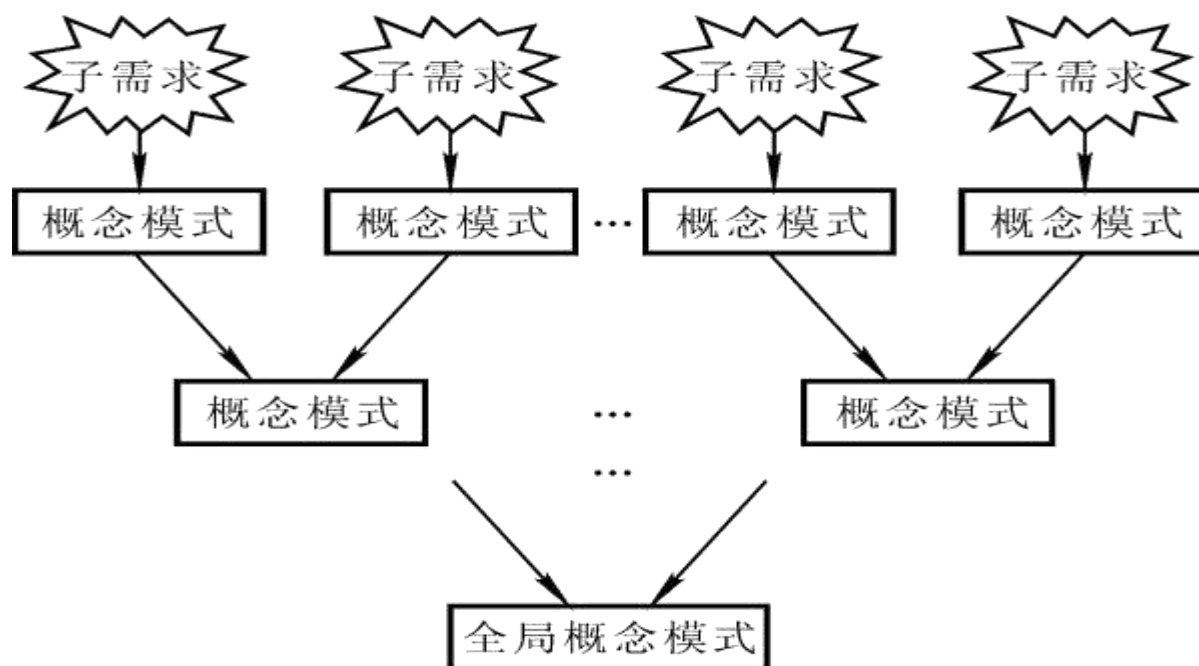
- 自顶向下：首先定义全局概念结构的框架，然后逐步细化



概念结构设计的方法与步骤（续）

■ 自底向上

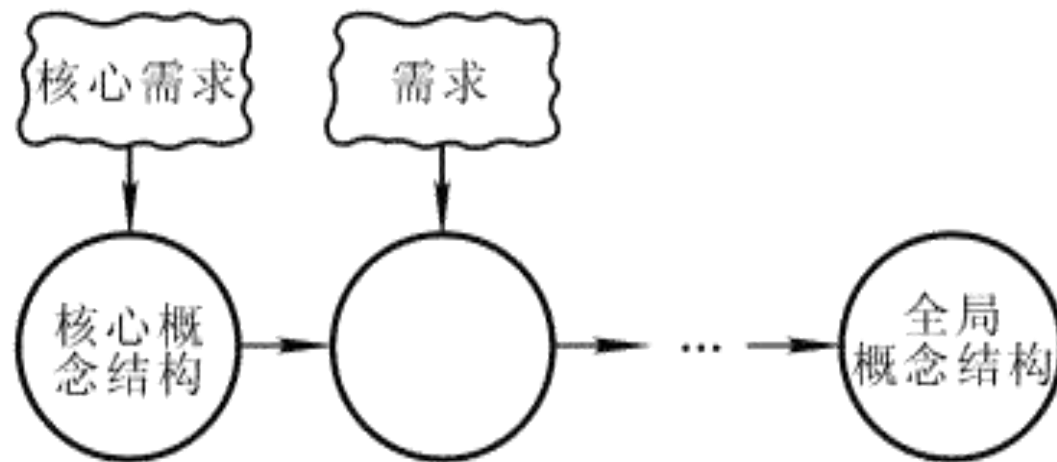
- 首先定义各局部应用的概念结构，然后将它们集成起来，得到全局概念结构



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

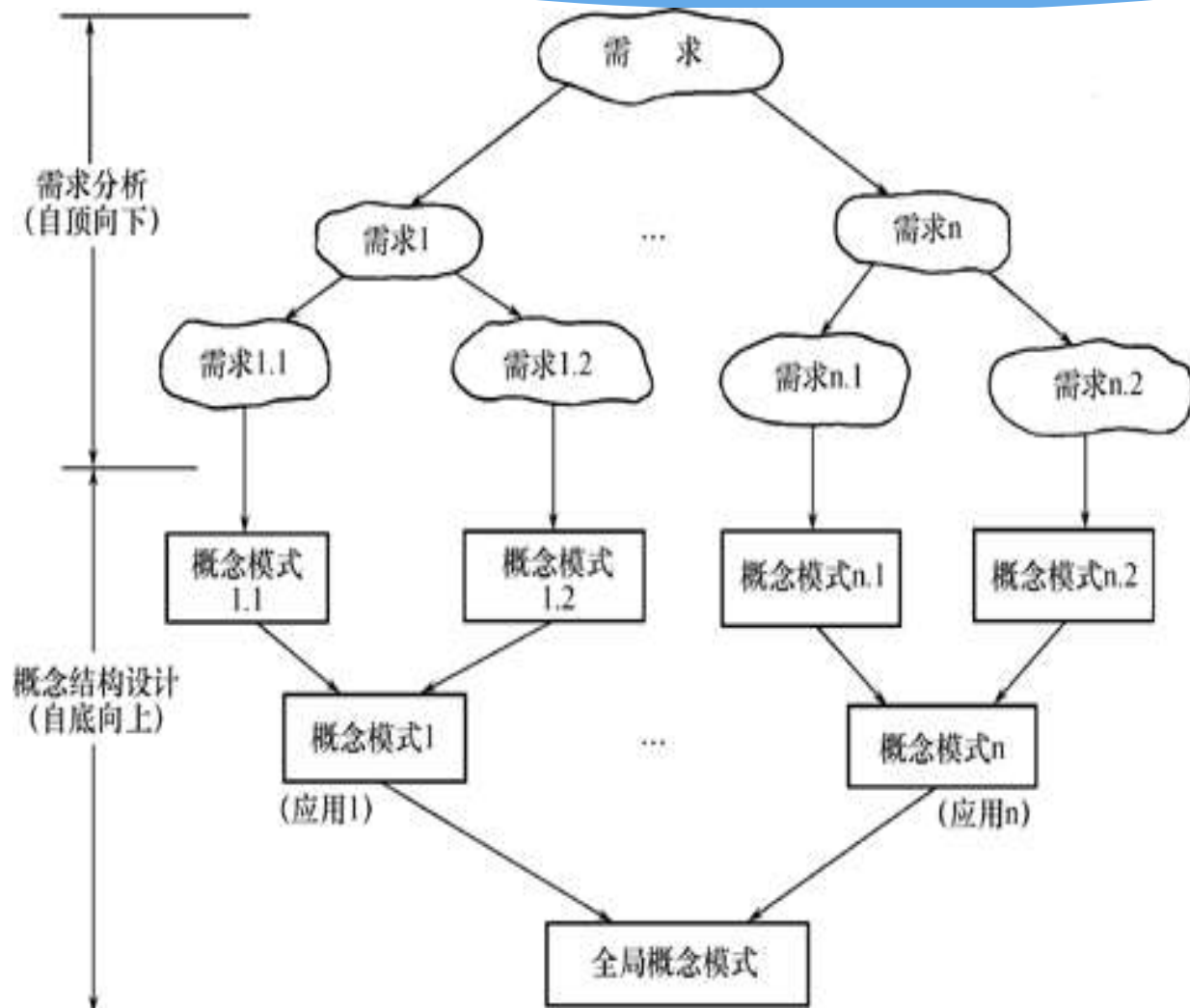
- 将自顶向下和自底向上相结合，用自顶向下策略设计一个全局概念结构的框架，以它为骨架集成由自底向上策略中设计的各局部概念结构。

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

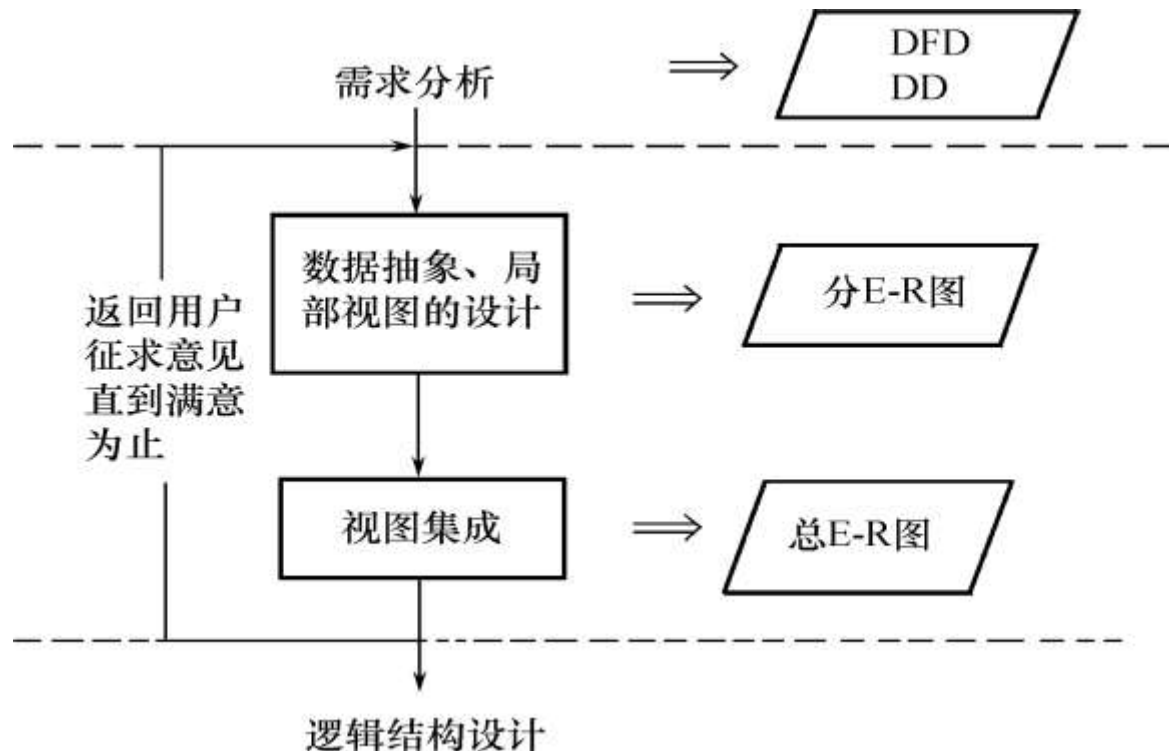


概念结构设计的方法与步骤（续）

❖ 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



3.3 局部视图设计

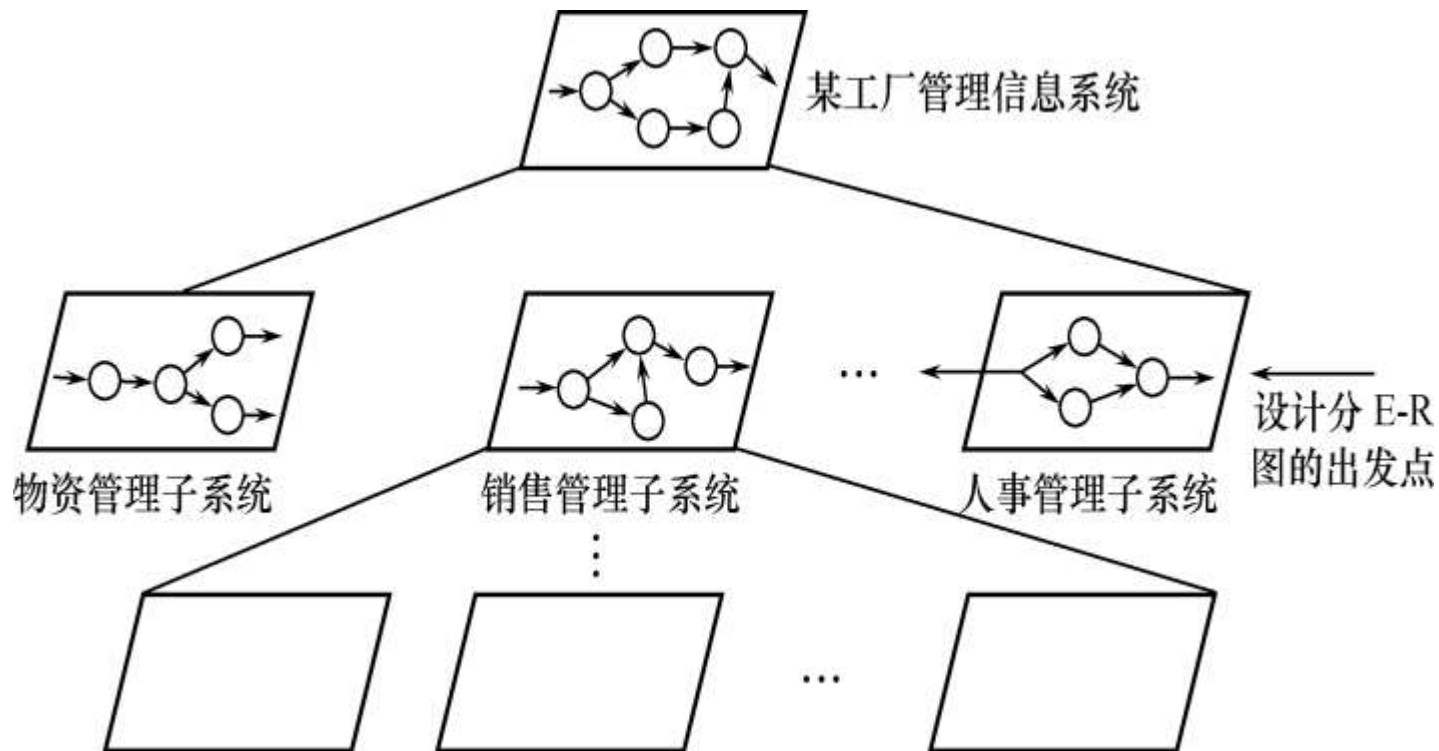
设计分E-R图的步骤:

- 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

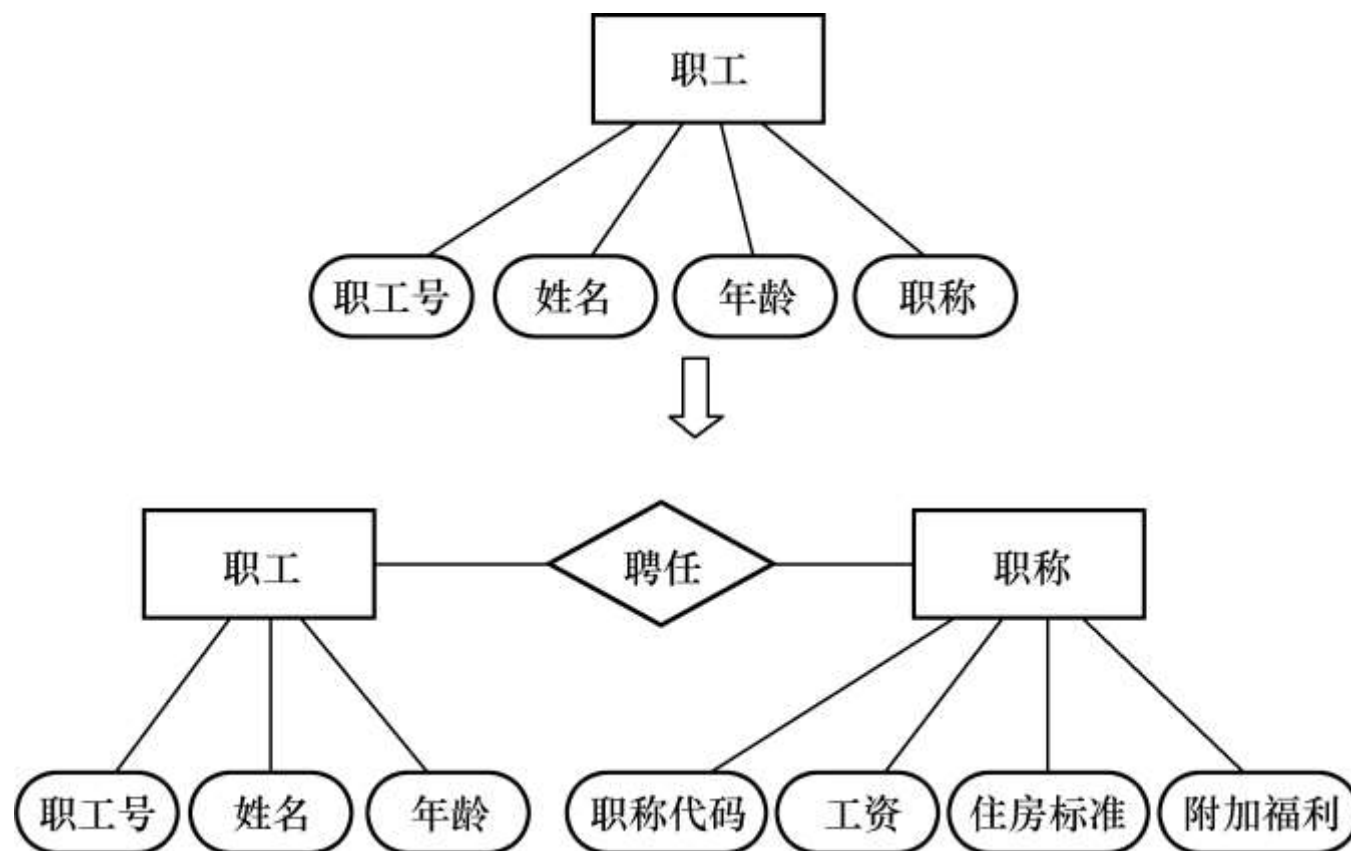
- 将各局部应用涉及的数据分别从数据字典中抽取出来
- 参照数据流图，标定各局部应用中的实体、实体的属性、标识实体的码
- 确定实体之间的联系及其类型（1:1，1:n，m:n）

逐一设计分E-R图（续）

❖ 两条准则：

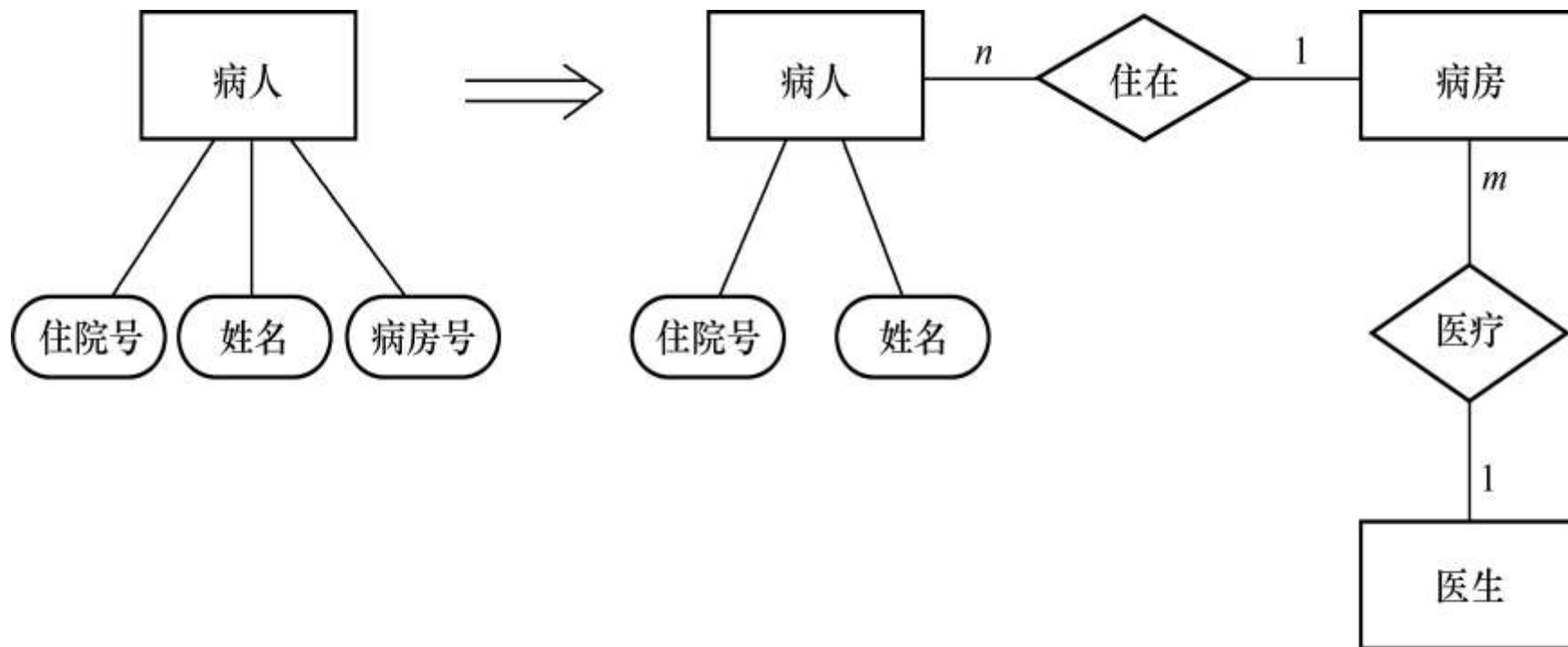
- （1）属性不能再具有需要描述的性质。即属性必须是不可分的数据项，不能再由另一些属性组成
- （2）属性不能与其他实体具有联系。联系只发生在实体之间

逐一设计分E-R图（续）



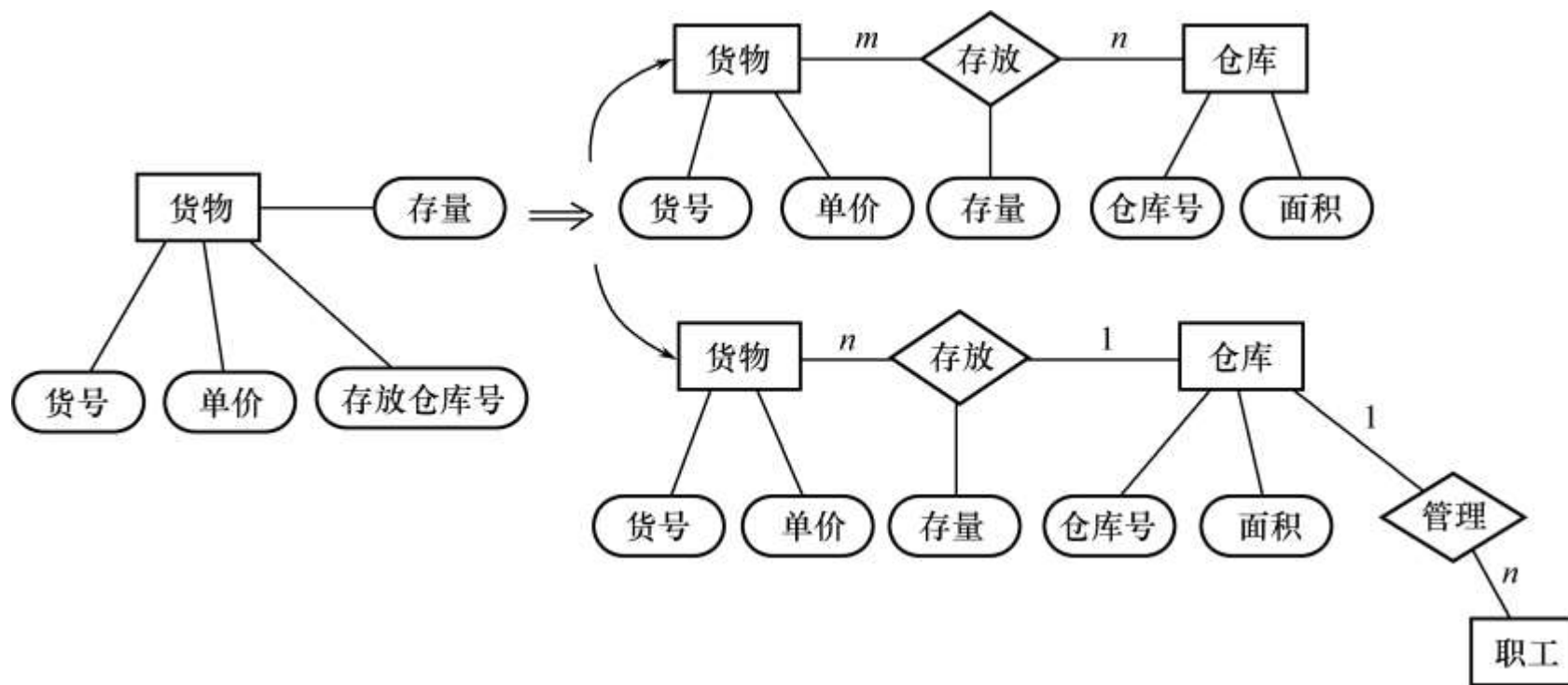
职称作为一个实体

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

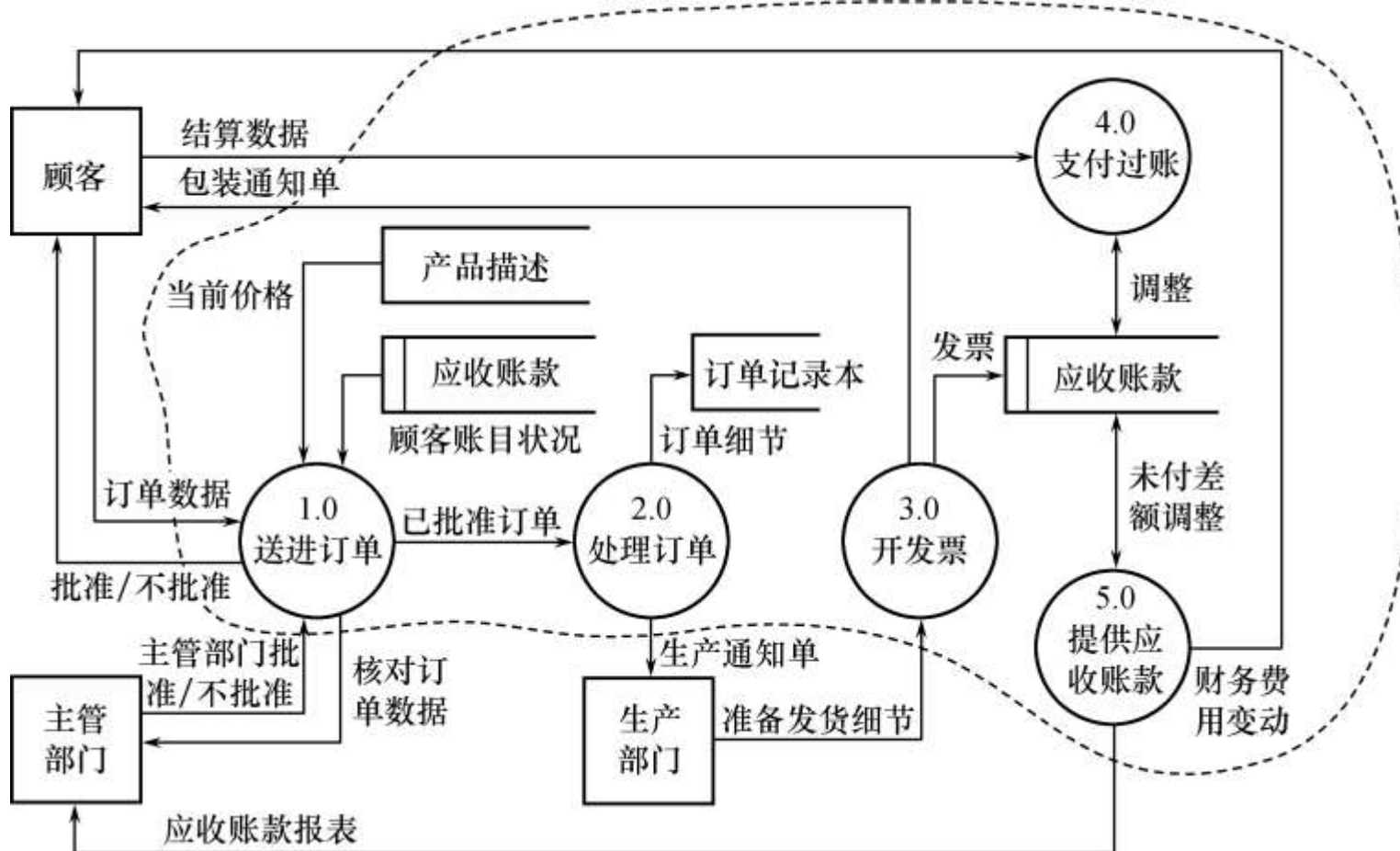
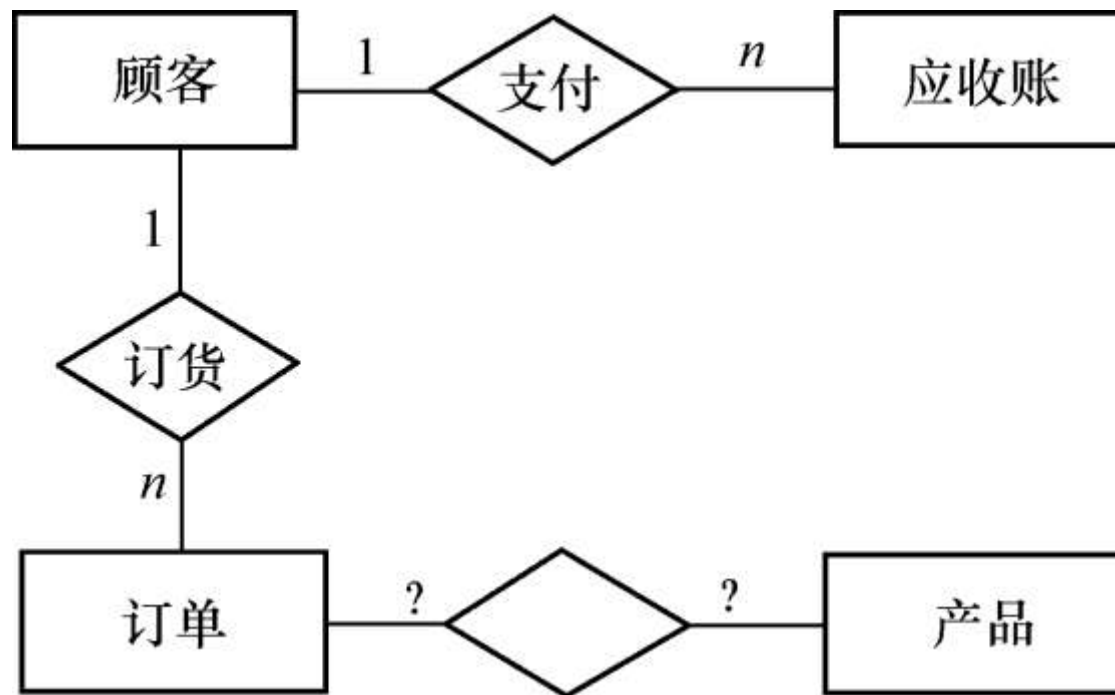


图18 销售管理子系统第一层数据流图

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）

❖ 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

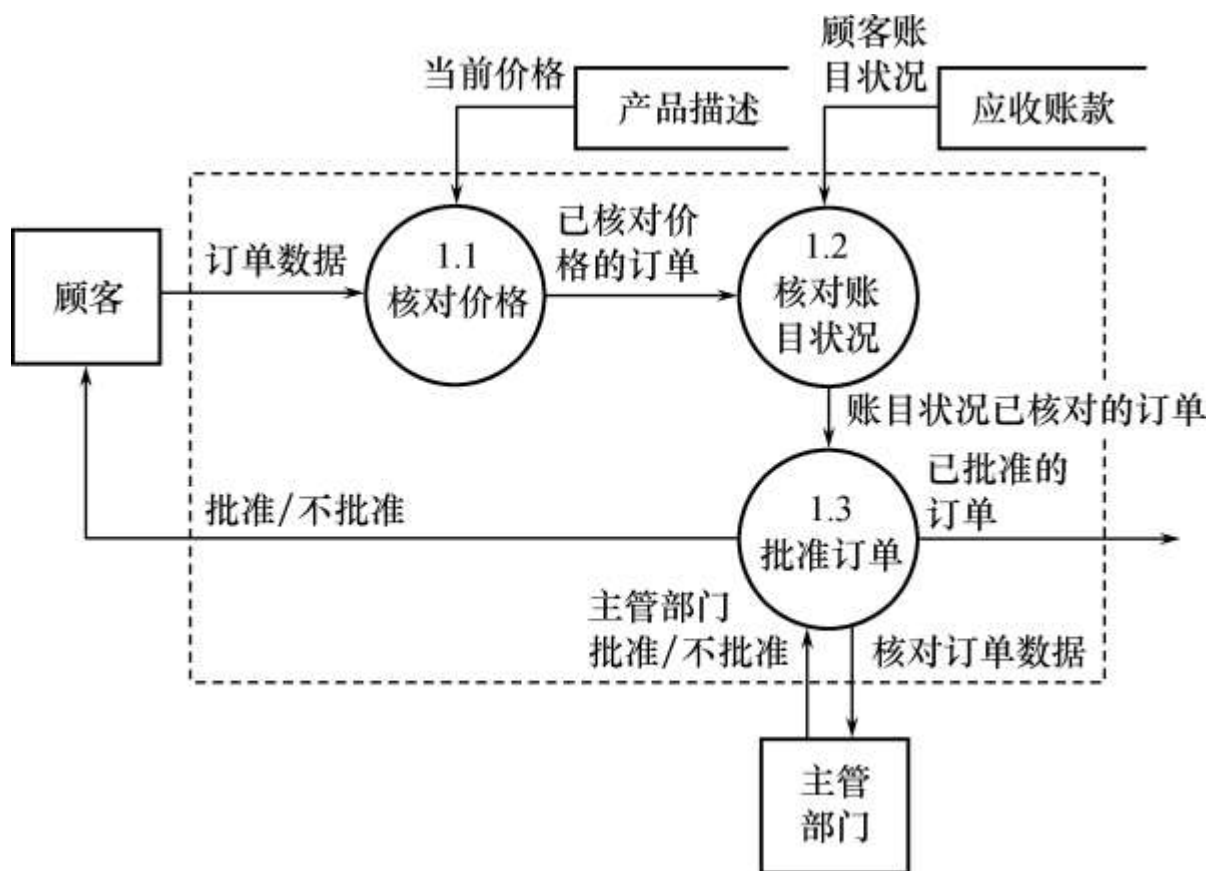


图19 接收订单

逐一设计分E-R图（续）

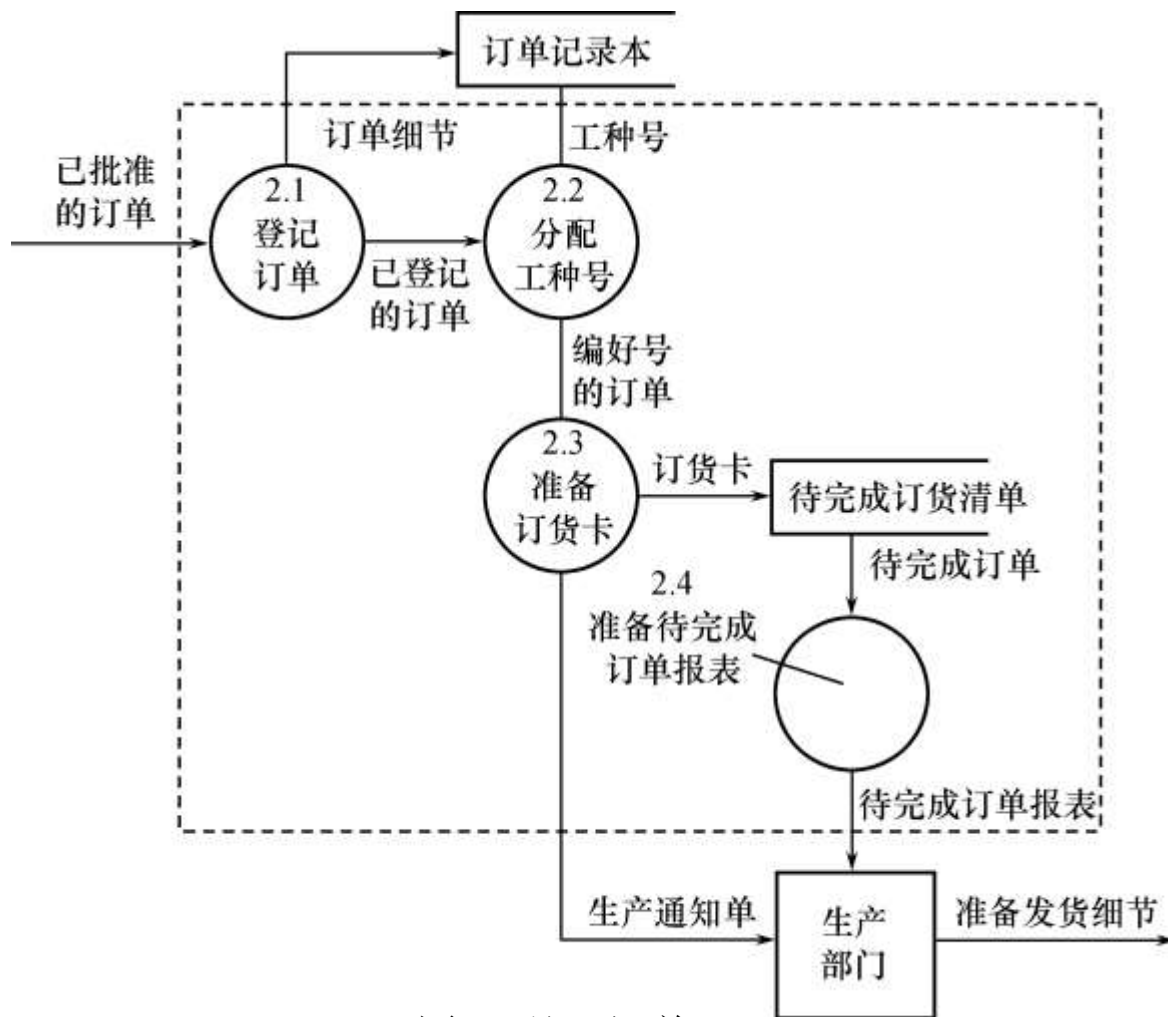


图20 处理订单

逐一设计分E-R图（续）

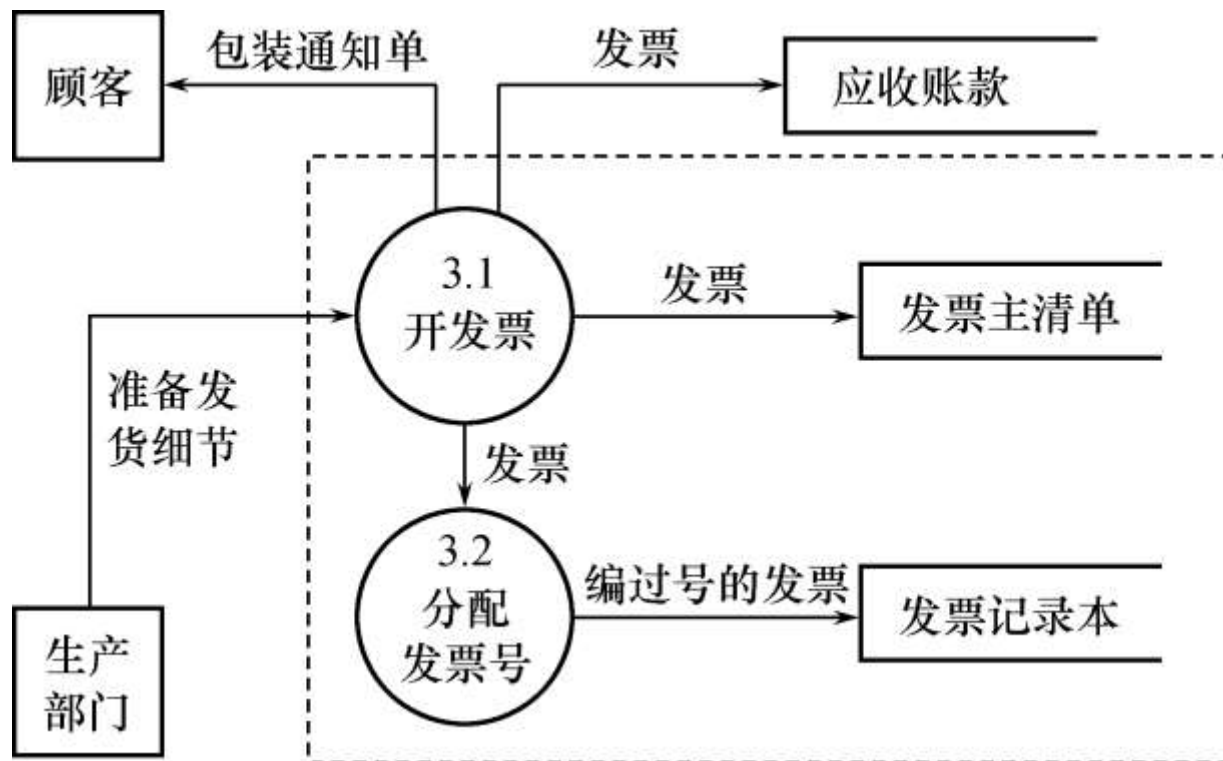


图21 开发票

逐一设计分E-R图（续）

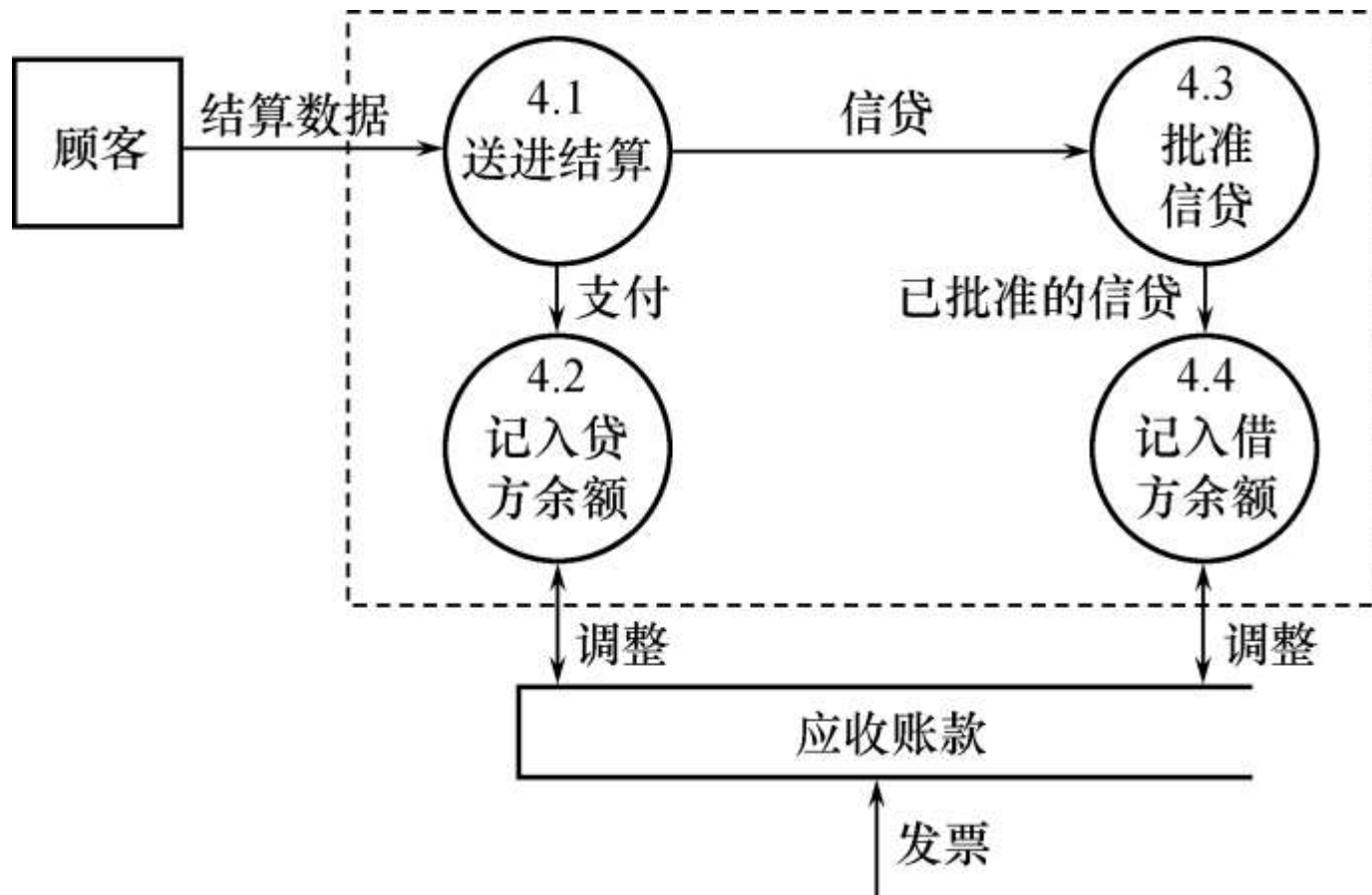


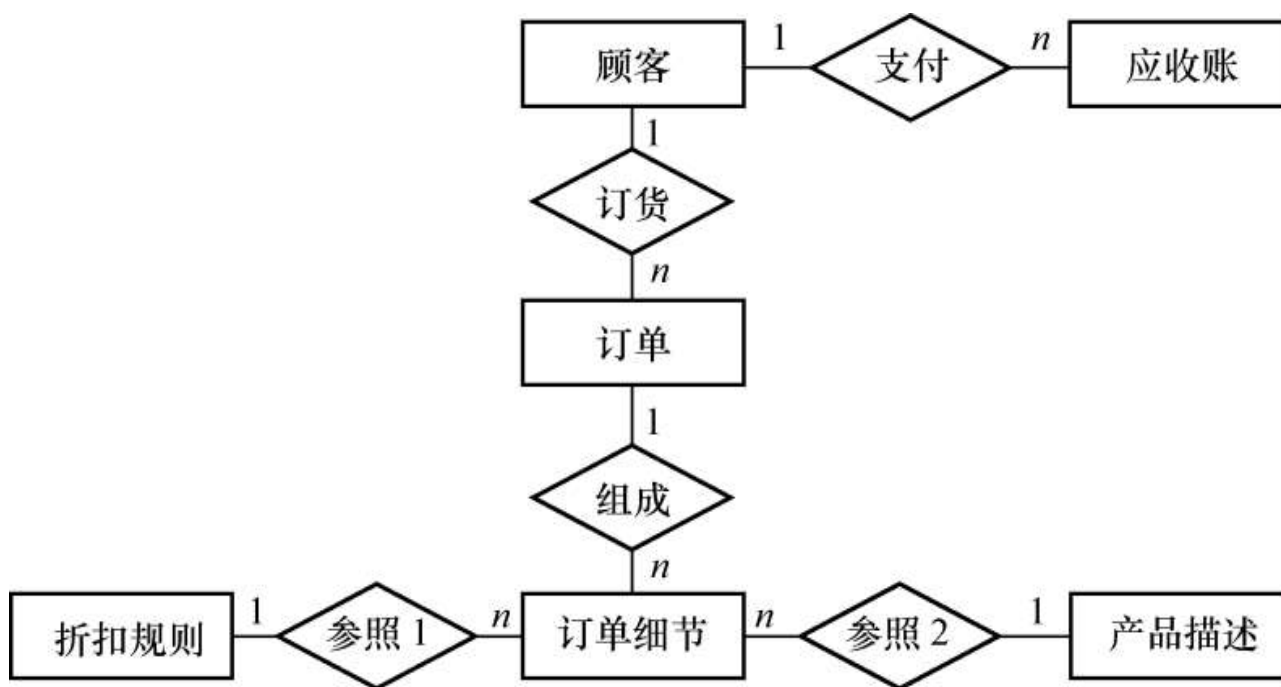
图22 支付过账

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) 图21中“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，货款限额}
- 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}

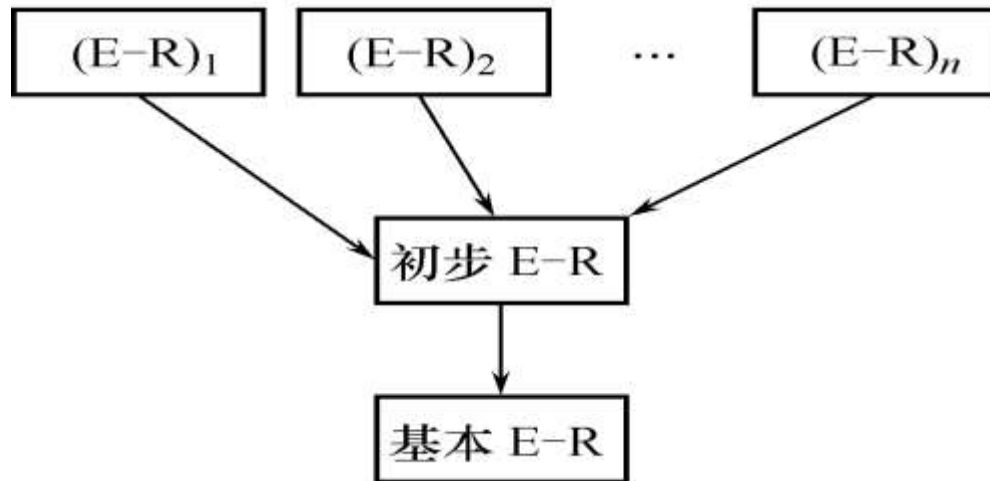
3.4 视图的集成

- ❖ 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

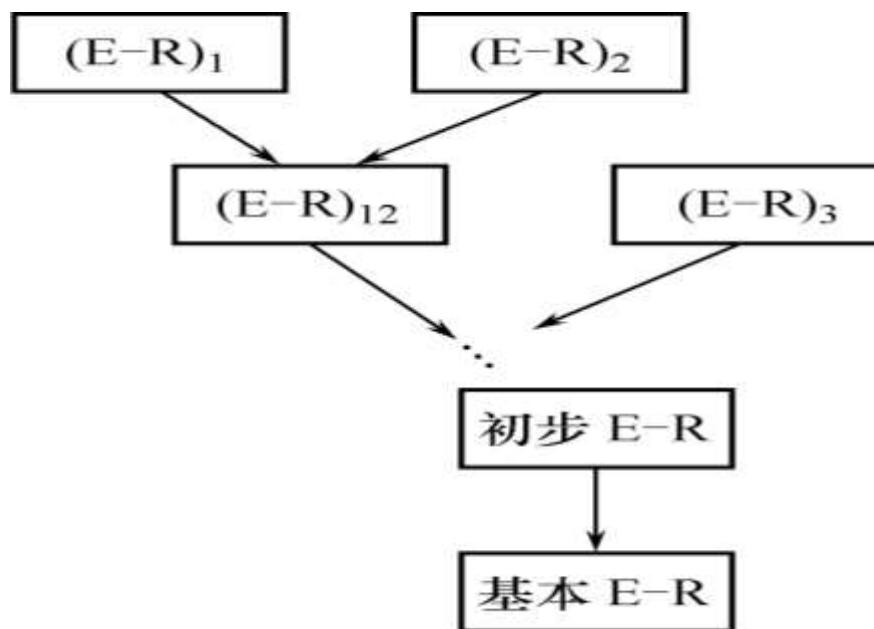
- 一次集成多个分E-R图
- 通常用于局部视图比较简单时



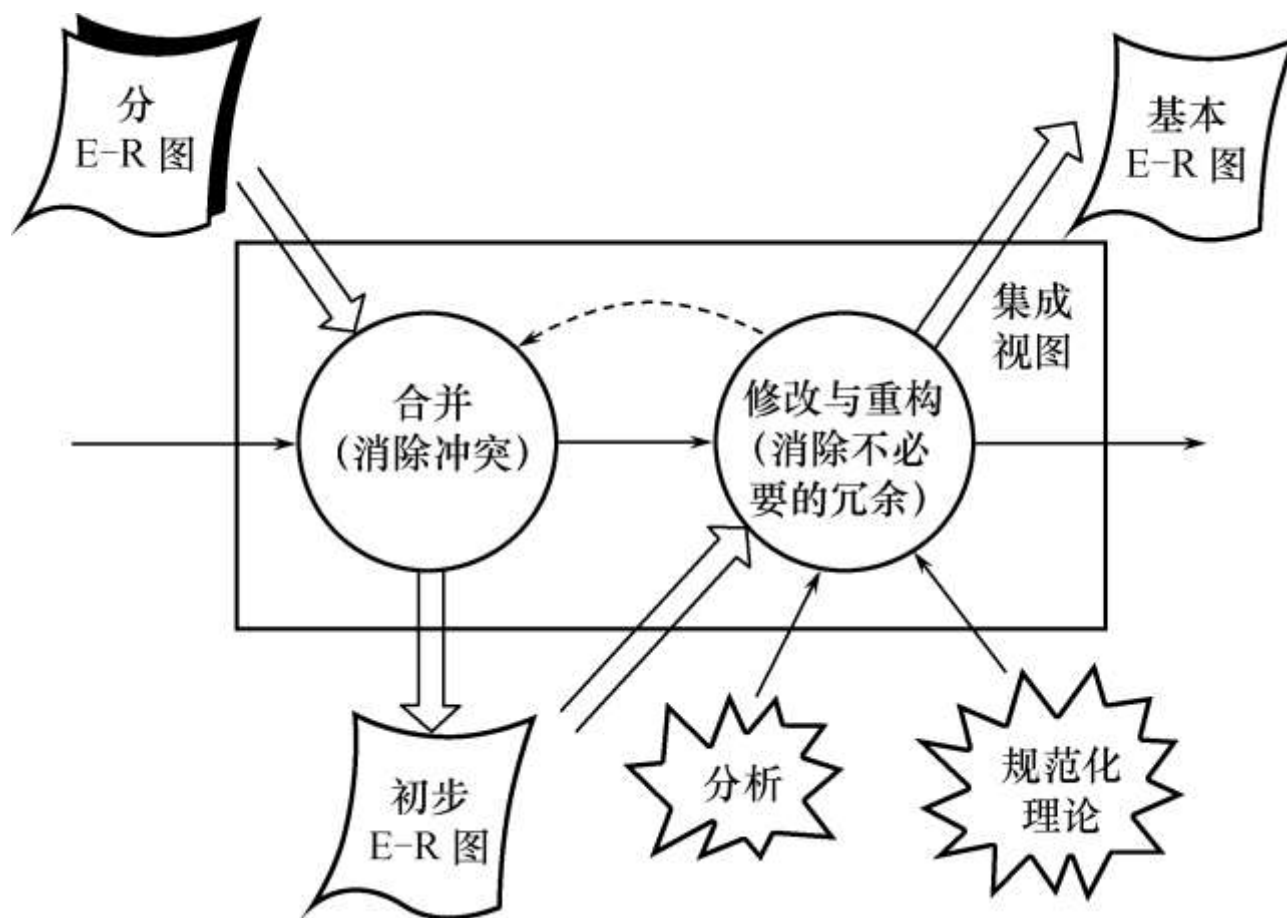
视图的集成（续）

❖ 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）

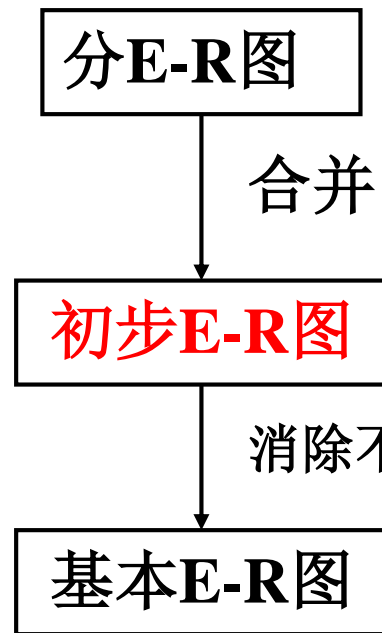


视图集成

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

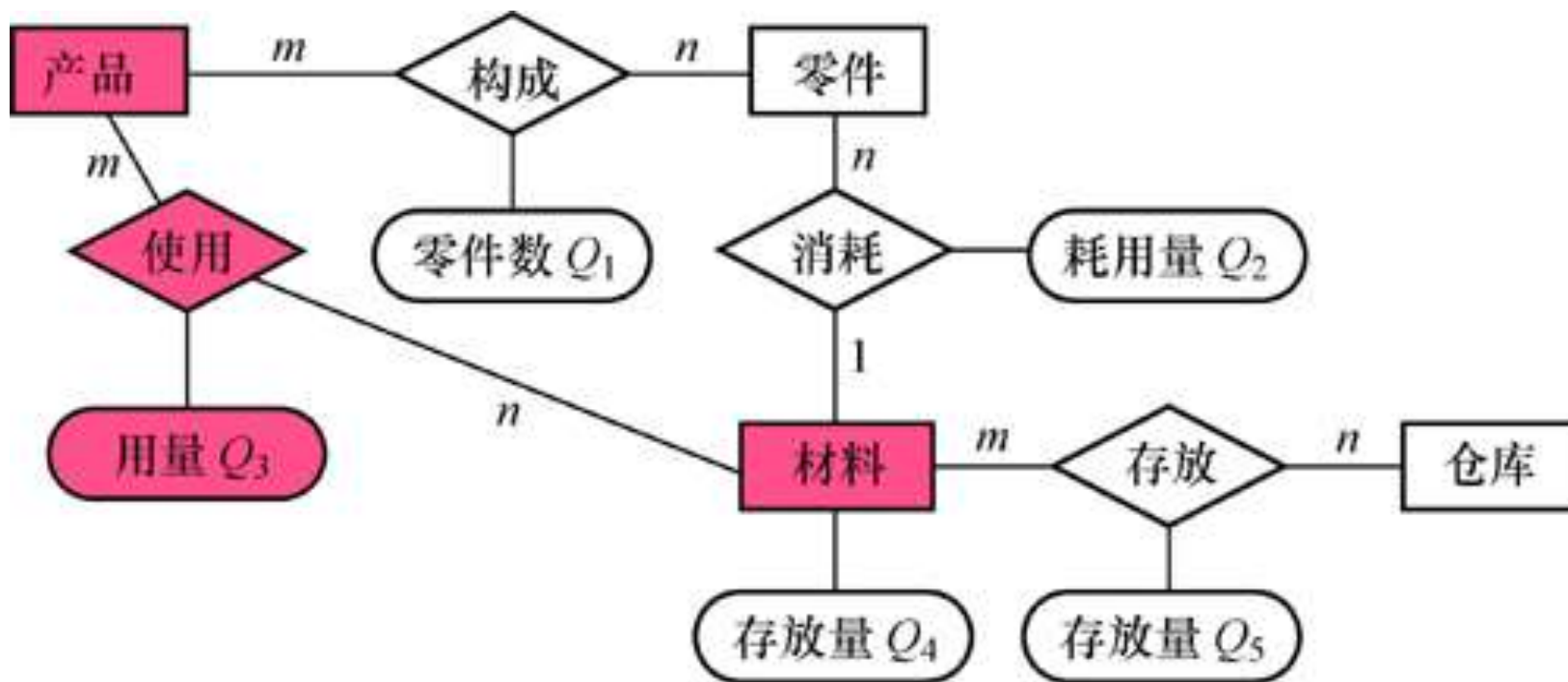
❖ 分析方法

- 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

❖ 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
 - $Q4 = \sum Q5$
 - 一旦**Q5**修改后就应当触发完整性检查，对**Q4**进行修改

消除冗余，设计生成基本E-R图实例

[实例] 某工厂管理信息系统的视图集成。

书中图1.14(c)、图24、图29分别为该厂物资、销售和劳动人事管理的分E-R图

图30为该系统的基本E-R图

消除冗余，设计生成基本E-R图实例（续）

该厂物资管理分E-R图

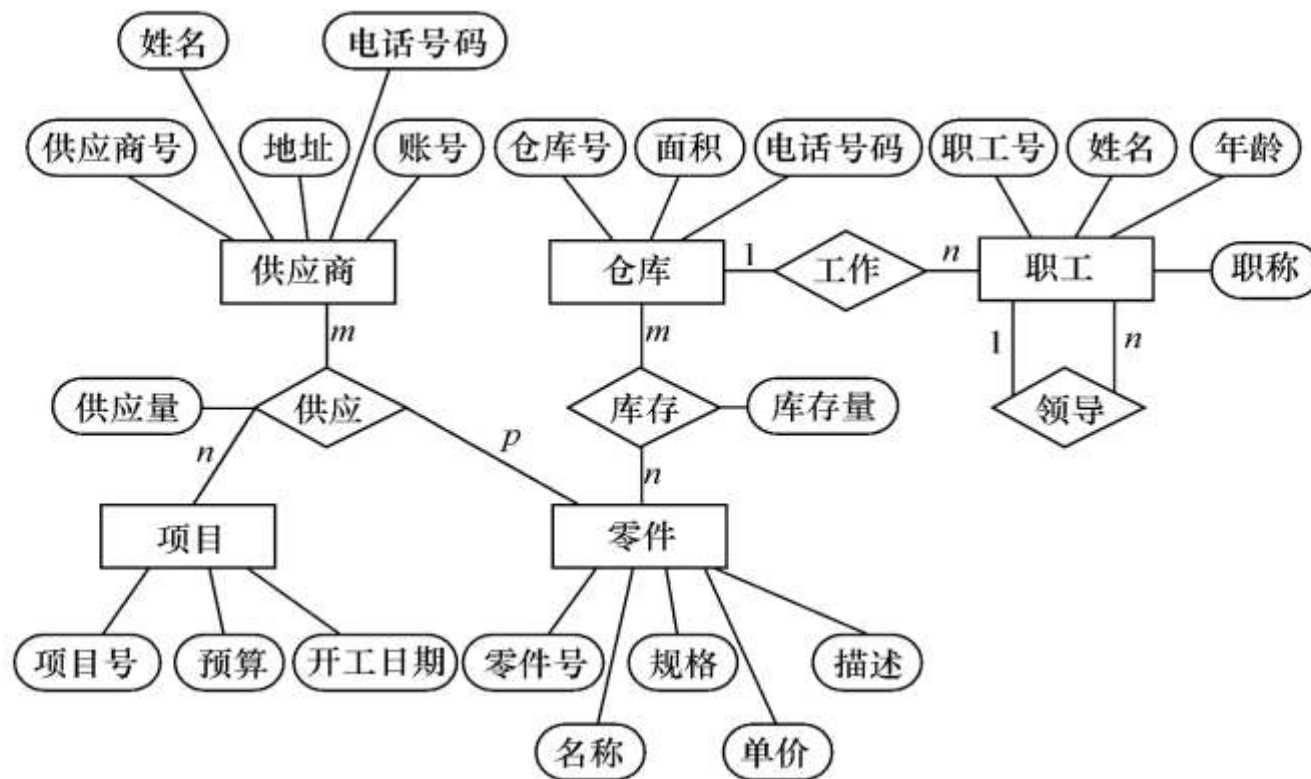


图1.14(c) 工厂物资管理E-R图

消除冗余，设计生成基本E-R图实例（续）

该厂销售管理分E-R图

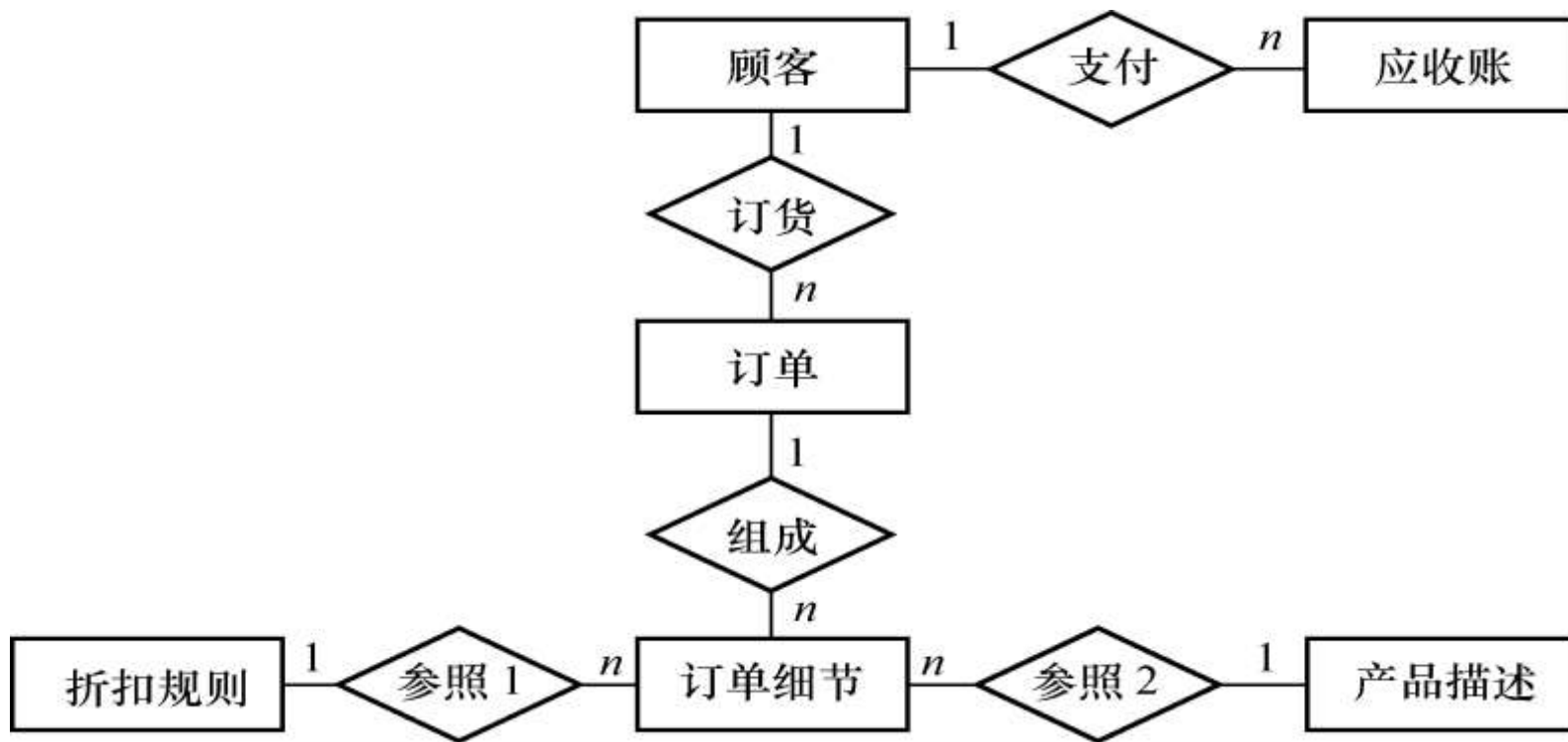


图24 销售管理子系统的分E-R图

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

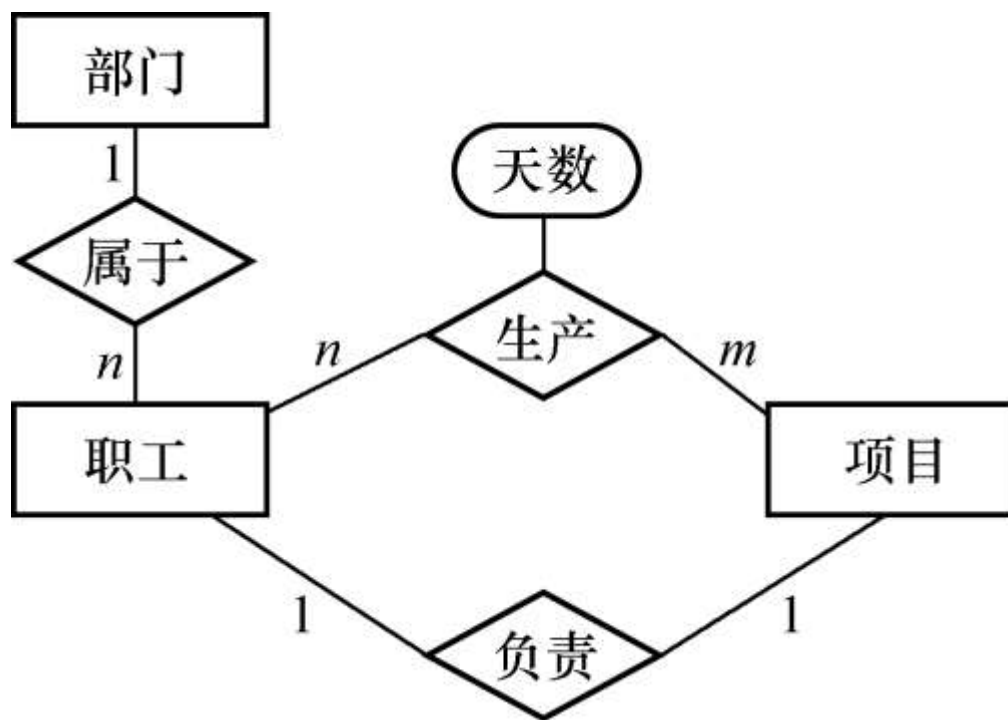
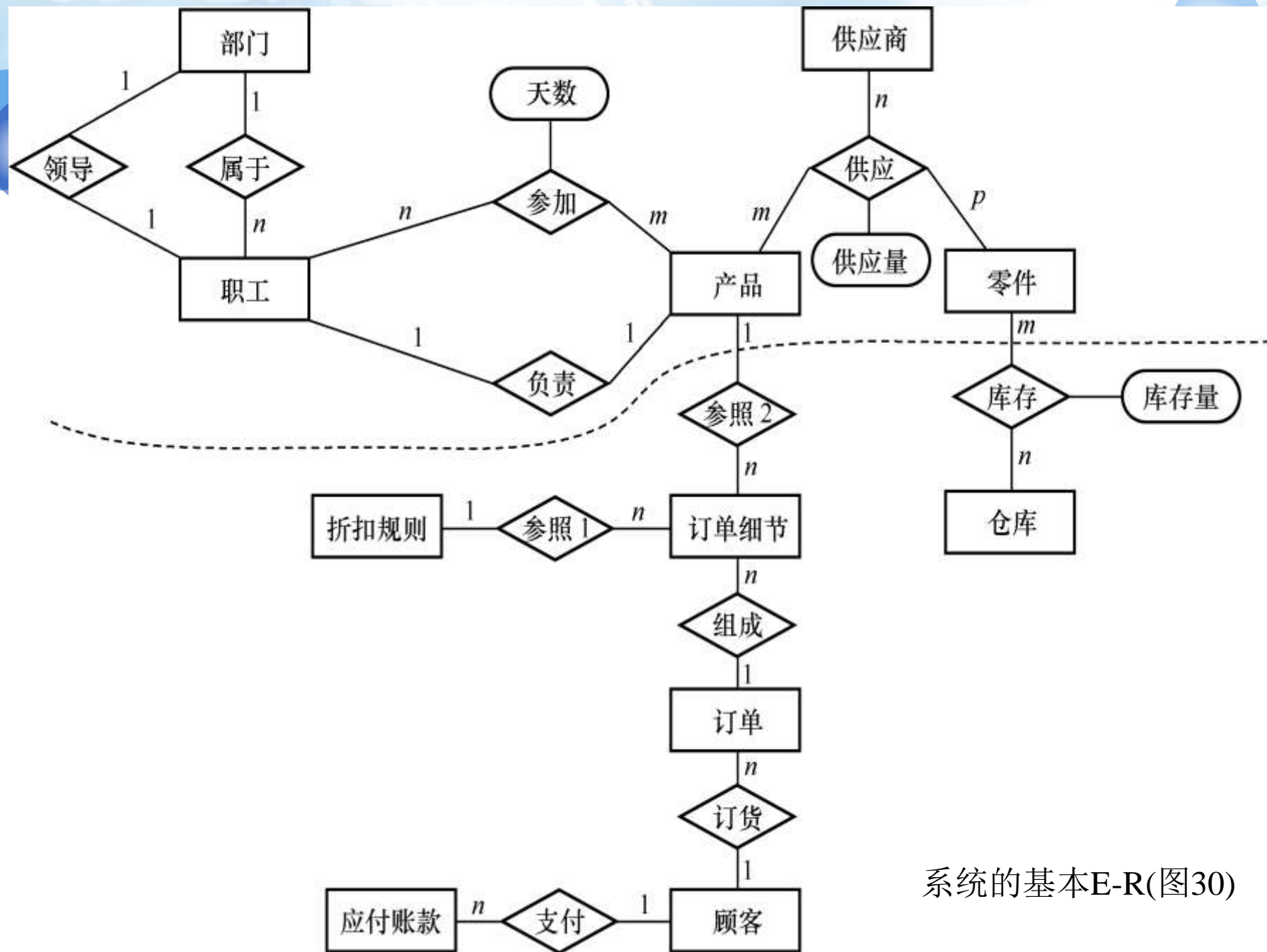


图29 劳动人事管理的分E-R图



系统的基本E-R(图30)

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

概念结构设计小结

❖ 设计局部视图

- 1. 选择局部应用
- 2. 逐一设计分E-R图
 - 标定局部应用中的实体、属性、码，实体间的联系
 - 用E-R图描述出来

概念结构设计小结

❖ 集成局部视图

■ 1. 合并分E-R图，生成初步E-R图

➤ 消除冲突

- 属性冲突
- 命名冲突
- 结构冲突

■ 2. 修改与重构

➤ 消除不必要的冗余，设计生成基本E-R图

- 分析方法
- 规范化理论

4 逻辑结构设计

4.1 E-R图向关系模型的转换

4.2 数据模型的优化

4.3 设计用户子模式

4.1 E-R图向关系模型的转换

❖ 转换内容

❖ 转换原则

E-R图		关系模型
实体		R
属性		U
联系	1:1	R
		合并到相邻实体
	1:n	R
		合并到n端实体
	m:n	R
	多元	R

例：基本E-R(图30)

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），并适当调整属性的次序

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] 把图30中虚线上部的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] 把图30中虚线上部的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] 把图30中虚线上部的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）

4 逻辑结构设计

4.1 E-R图向关系模型的转换

4.2 数据模型的优化

4.3 设计用户子模式

4.2 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。

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，一般说来，第三范式就足够了

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

4 逻辑结构设计

4.1 E-R图向关系模型的转换

4.2 数据模型的优化

4.3 设计用户子模式

4.3 设计用户子模式

❖ 定义用户外模式时应该注重的的问题

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

逻辑结构设计小结

- ❖ **E-R图向关系模型的转换内容**
- ❖ **E-R图向关系模型的转换原则**

逻辑结构设计小结

❖ 优化数据模型的方法

1. 确定数据依赖
2. 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。
3. 确定各关系模式分别属于第几范式。
4. 分析对于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。
5. 对关系模式进行必要的分解或合并

逻辑结构设计小结

❖ 设计用户子模式

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外模式，以满足系统对安全性的要求。
3. 简化用户对系统的使用