



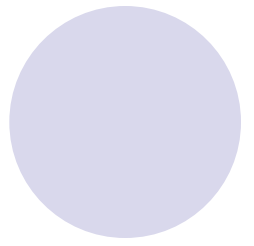
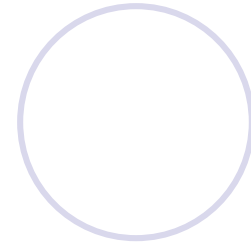
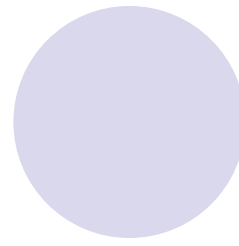
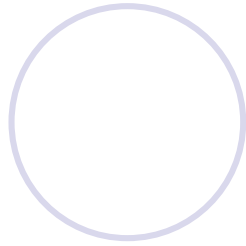
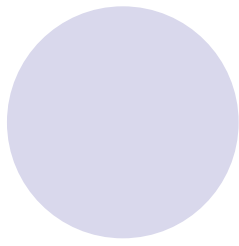
数据库系统

Database System

主讲：张仲楠 教授

Email: zhongnan_zhang@xmu.edu.cn

Office: 海韵A416



数据库系统 Database System

第七章 数据库设计

第七章 数据库设计

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 物理结构设计

7.6 数据库实施和维护

7.7 小结

7.1 数据库设计概述

- 数据库设计（广义）
 - 数据库及其应用系统的设计，即设计整个数据库应用系统
- 数据库设计（狭义） --- **关注点**
 - 设计数据库本身，即设计数据库的各级模式并建立数据库
 - 数据库应用系统设计的一部分
- 数据库与应用系统密不可分

7.1 数据库设计概述

- 数据库设计是指对于一个给定的应用环境，构造（设计）优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户的应用需求，包括信息管理要求和数据操作要求。
- 信息管理要求：在数据库中应该存储和管理哪些数据对象。
- 数据操作要求：对数据对象需要进行哪些操作，如查询、增、删、改、统计等操作。

数据库设计概述（续）

- 数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效率的运行环境。
- 高效率的运行环境
 - 数据库数据的存取效率高
 - 数据库存储空间的利用率高
 - 数据库系统运行管理的效率高

7.1 数据库设计概述

7.1.1 数据库设计的特点

7.1.2 数据库设计方法

7.1.3 数据库设计的基本步骤

7.1.4 数据库设计过程中的各级模式

7.1.1 数据库设计的特点

1. 数据库建设的基本规律

- 三分技术，七分管理，十二分基础数据
- 管理
 - 数据库建设项目管理
 - 企业（即应用部门）的业务管理
- 基础数据
 - 数据的收集、整理、组织和不断更新

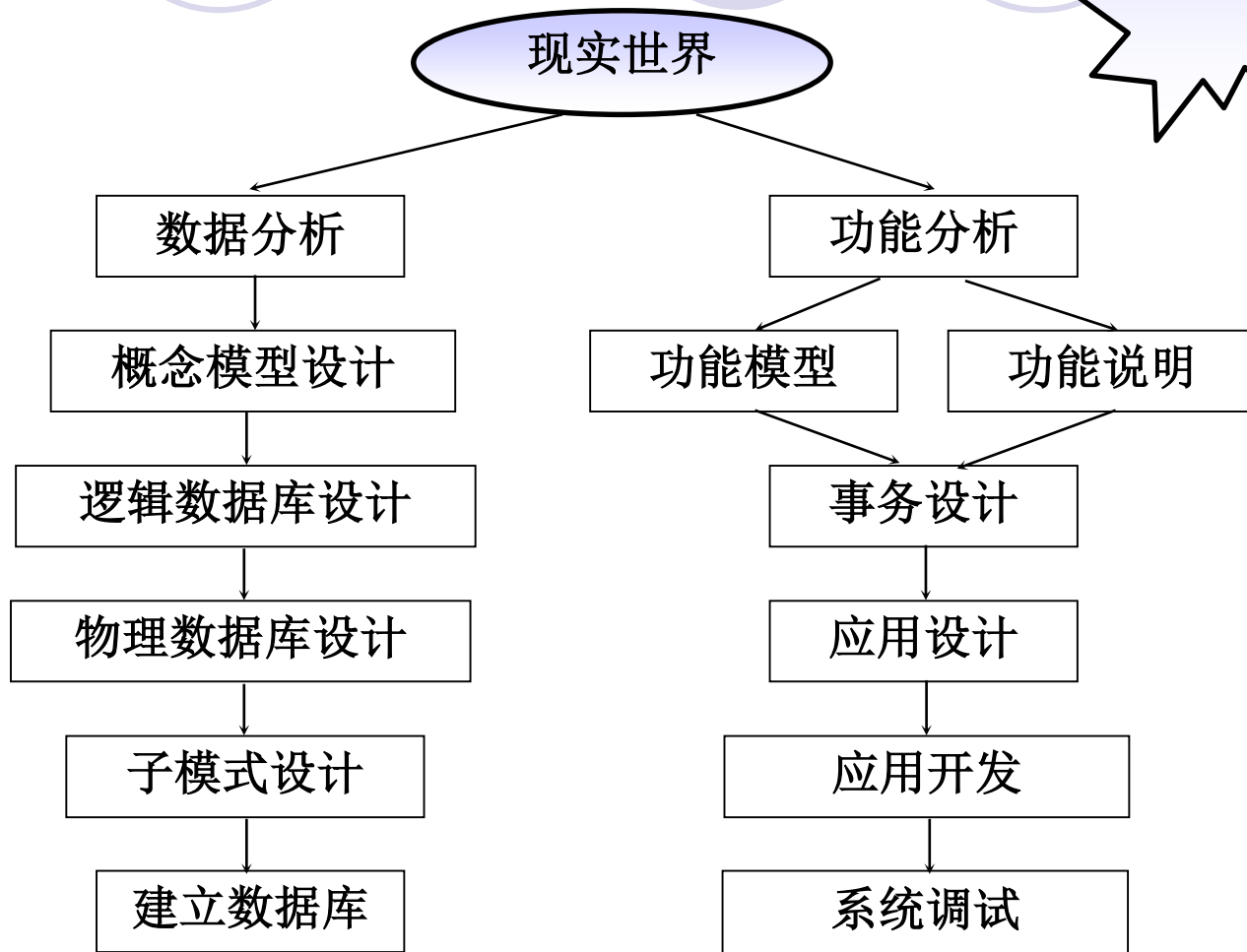
数据库设计的特点（续）

2. 结构（数据）设计和行为（处理）设计相结合

- 将数据库**结构设计**和**数据处理设计**密切结合
- 结构和行为分离的设计
 - 传统的软件工程：重 **行为设计**
 - 忽视对应用中数据语义的分析和抽象，只要有可能就尽量推迟数据结构设计的决策
 - 早期的数据库设计：重 **结构设计**
 - 致力于数据模型和数据库建模方法研究，忽视了行为设计对结构设计的影响

数据库设计的特点（续）

错误的



结构和行为分离的设计

7.1 数据库设计概述

7.1.1 数据库设计的特点

7.1.2 数据库设计方法

7.1.3 数据库设计的基本步骤

7.1.4 数据库设计过程中的各级模式

7.1.2 数据库设计方法

- 要求数据库设计人员应该具备的技术和知识
 - 计算机的基本知识;
 - 软件工程的原理和方法;
 - 程序设计的方法和技巧;
 - 数据库的基本知识和设计技术;
 - 应用领域的知识。

数据库设计方法（续）

- 手工试凑法

- 手工与经验相结合的方法
- 设计质量与设计人员的经验和水平有直接关系
- 缺乏科学理论和工程方法的支持，质量难以保证
- 数据库运行一段时间后常常又不同程度地发现各种问题，增加了维护代价

数据库设计方法（续）

- 规范设计法

- 新奥尔良（New Orleans）方法

- 基于E-R模型的数据库设计方法

- 3NF（第三范式）的设计方法

- 面向对象的数据库设计方法

- 统一建模语言（UML）方法

需求分析
概念设计
逻辑设计
物理设计

7.1 数据库设计概述

7.1.1 数据库设计的特点

7.1.2 数据库设计方法

7.1.3 数据库设计的基本步骤

7.1.4 数据库设计过程中的各级模式

7.1.3 数据库设计的基本步骤

- 按照结构化系统设计方法，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）

- 参加数据库设计的人员

- 系统分析人员和数据库设计人员

- 自始至终参与数据库设计，其水平决定了数据库系统的质量

- 数据库管理员和用户代表

- 主要参加需求分析与数据库的运行和维护

- 应用开发人员

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

数据库设计的基本步骤（续）

1. 需求分析阶段

- 包括数据与处理, 是否做得充分与准确, 决定了构建数据库的速度和质量

2. 概念结构设计阶段

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

3. 逻辑结构设计阶段

- 将概念结构转换为**某个数据库管理系统所支持的数据模型**, 并对其**进行优化**

数据库设计的基本步骤（续）

4. 物理结构设计阶段

- 为**逻辑数据结构**选取一个最适合应用环境的**物理结构**
- 包括**存储结构和存取方法**

5. 数据库实施阶段

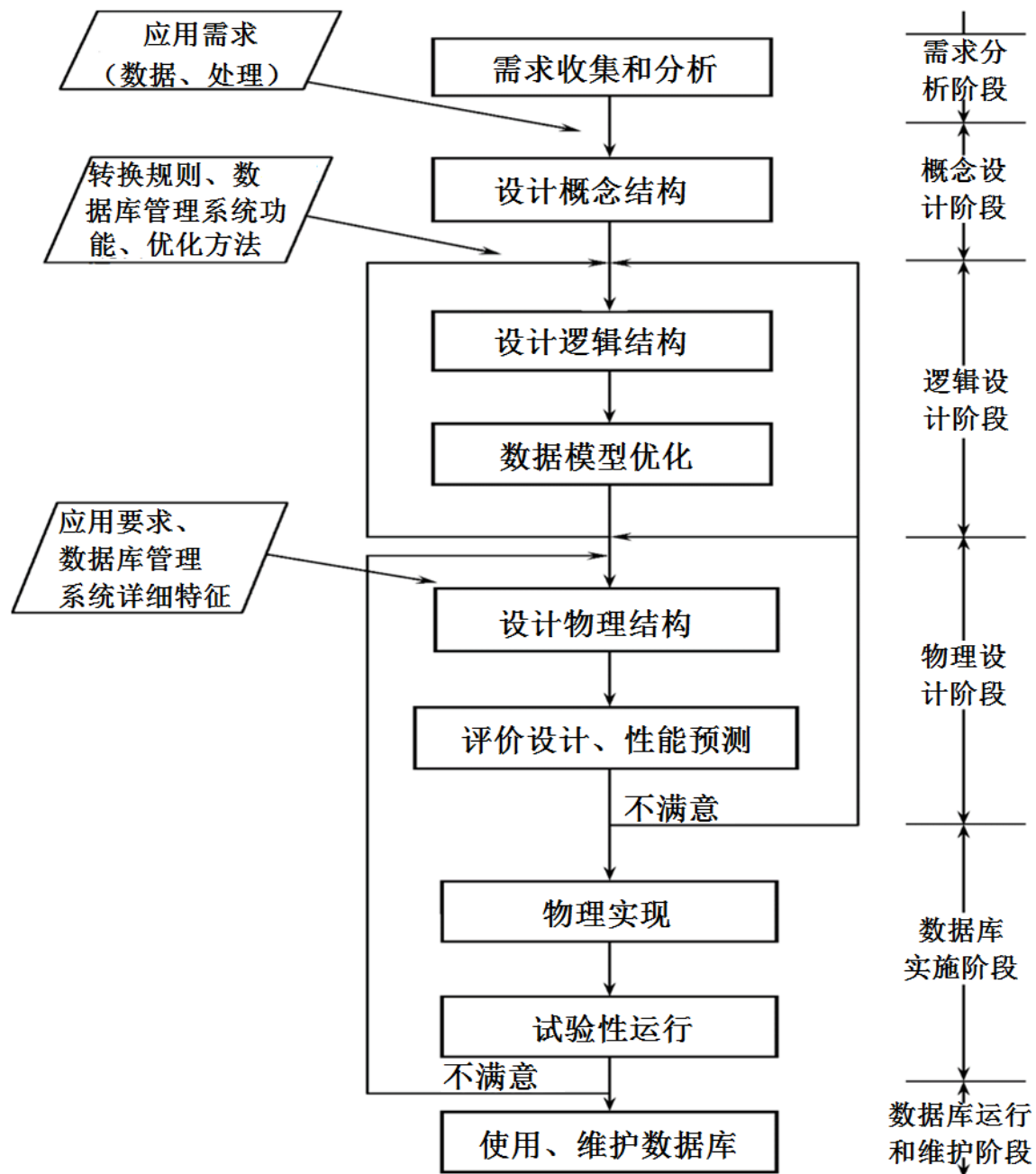
- 根据逻辑设计和物理设计的结果构建数据库
- 编写与调试应用程序
- 组织数据入库并进行试运行

6. 数据库运行和维护阶段

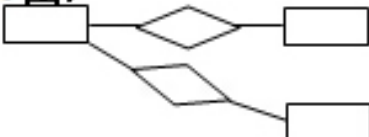
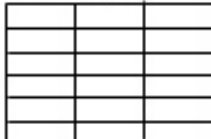
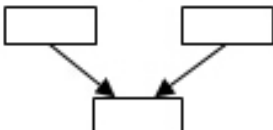
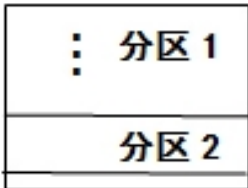

- 经过试运行后即可投入正式运行
- 在运行过程中必须不断对其进行**评估、调整与修改**

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6个阶段的**不断反复**
- 这个设计步骤既是数据库设计的过程，也包括了数据库应用系统的设计过程
- 把**数据库的设计**和对数据库中**数据处理的设计****紧密结合起来**，将这两个方面的需求分析、抽象、设计、实现在各个阶段同时进行，相互参照，相互补充，以完善两方面的设计



数据库设计步骤

设计阶段	设计描述
需求分析	数字字典、全系统中数据项、数据结构、数据流、数据存储的描述
概念结构设计	概念模型 (E-R 图)  数据字典
逻辑结构设计	某种数据模型 关系  非关系 
物理结构设计	存储安排 存取方法选择 存取路径建立 
数据库实施	创建数据库模式 装入数据 数据库试运行 
数据库运行和维护	性能监测、转储/恢复、数据库重组和重构

数据库设计各个阶段的数据设计描述

7.1 数据库设计概述

7.1.1 数据库设计的特点

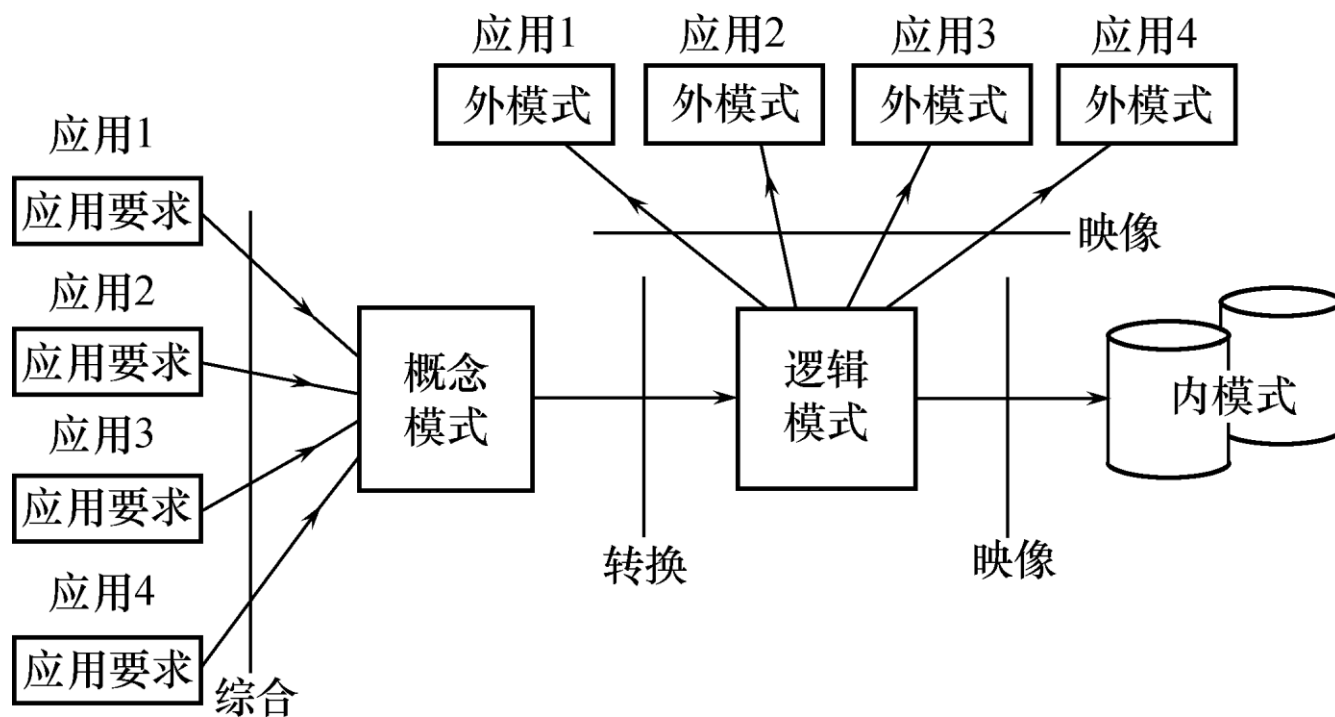
7.1.2 数据库设计方法

7.1.3 数据库设计的基本步骤

7.1.4 数据库设计过程中的各级模式

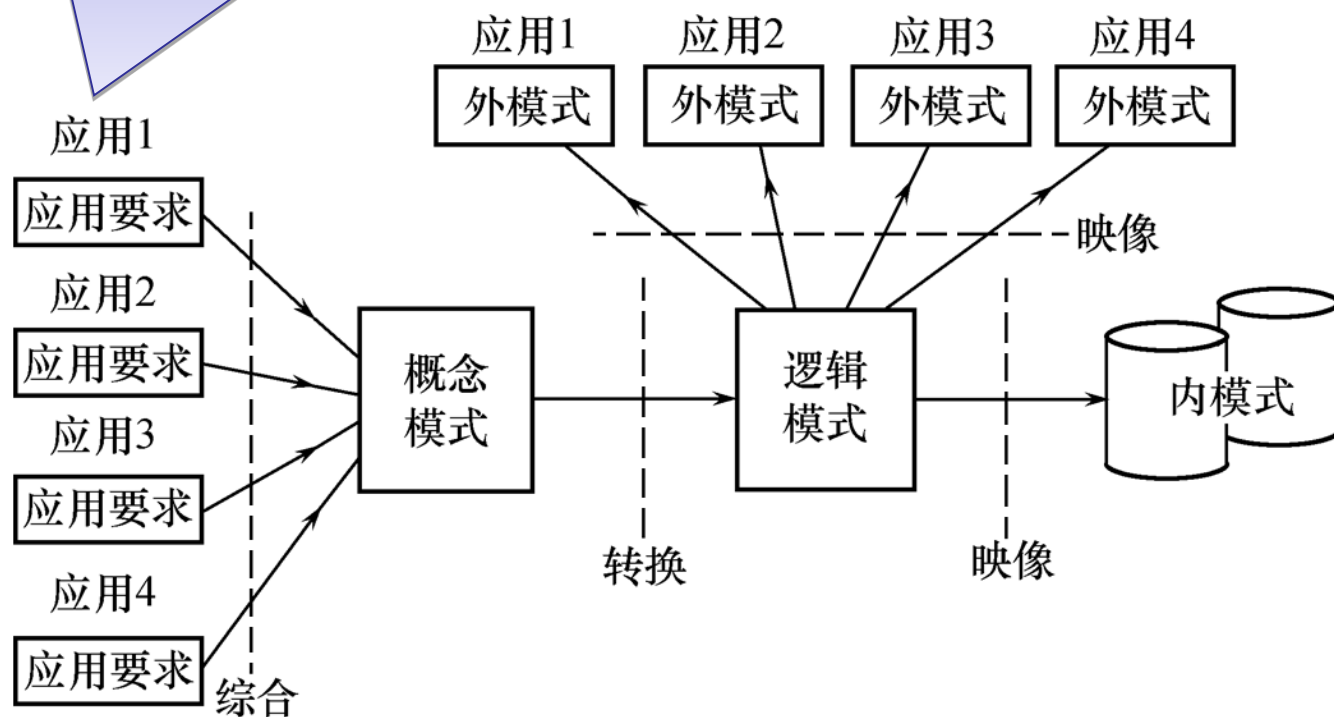
7.1.4 数据库设计过程中的各级模式

数据库设计不同阶段形成的数据库各级模式

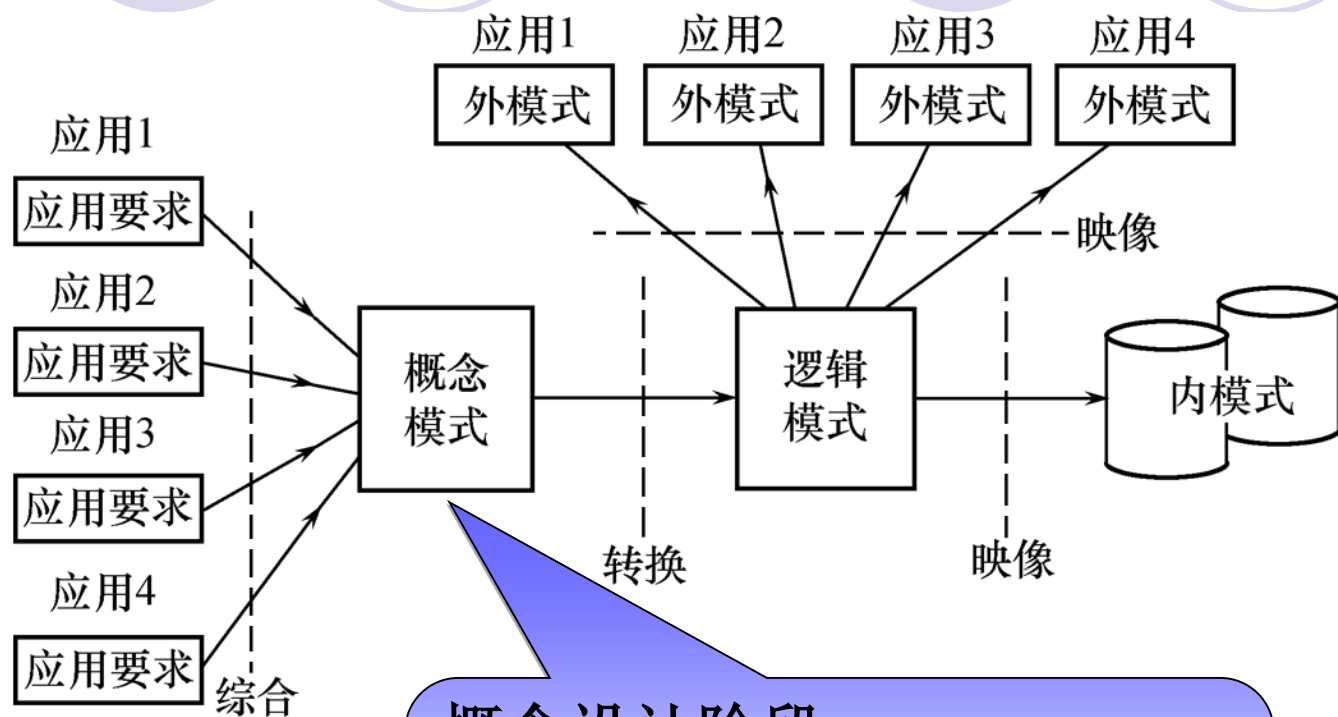


数据库设计过程中的各级模式（续）

需求分析阶段：
综合各个用户的应用需求

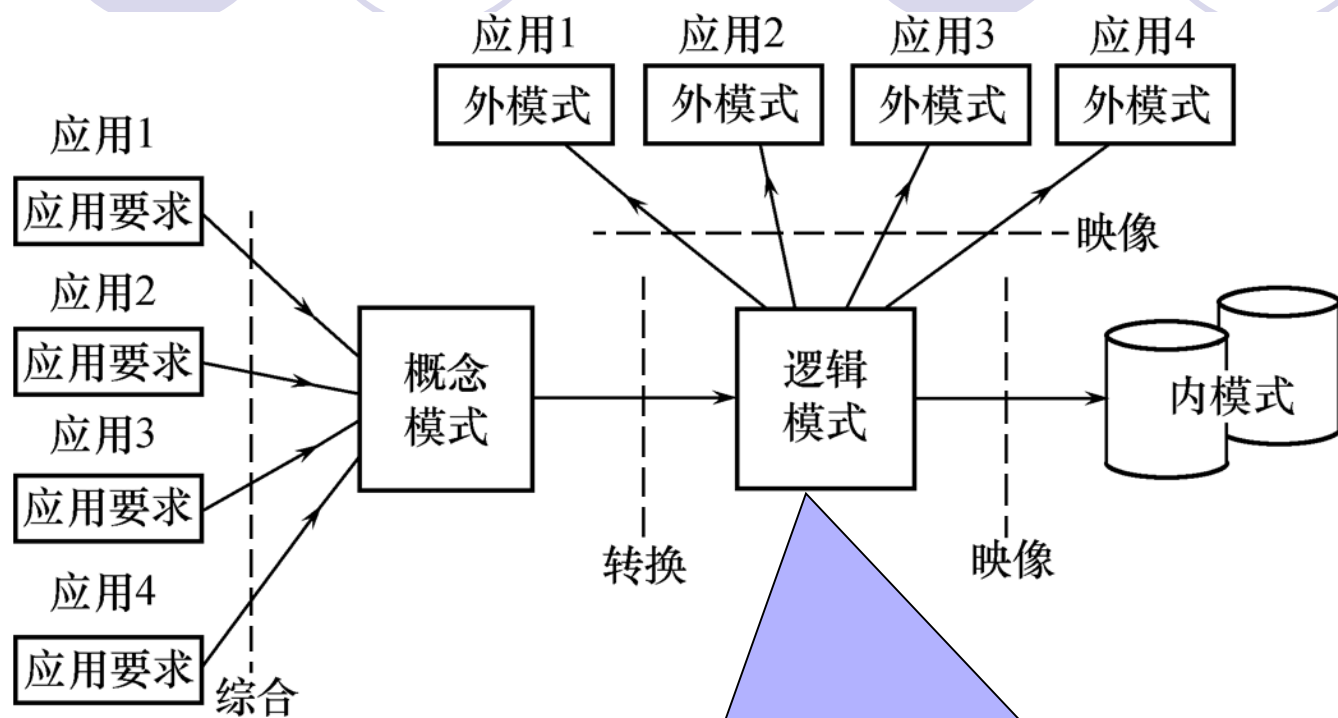


数据库设计过程中的各级模式（续）



概念设计阶段：
形成独立于机器特点，独立于各个数据库管理系统产品的**概念模式**（E-R图）

数据库设计过程中的各级模式（续）



逻辑设计阶段：

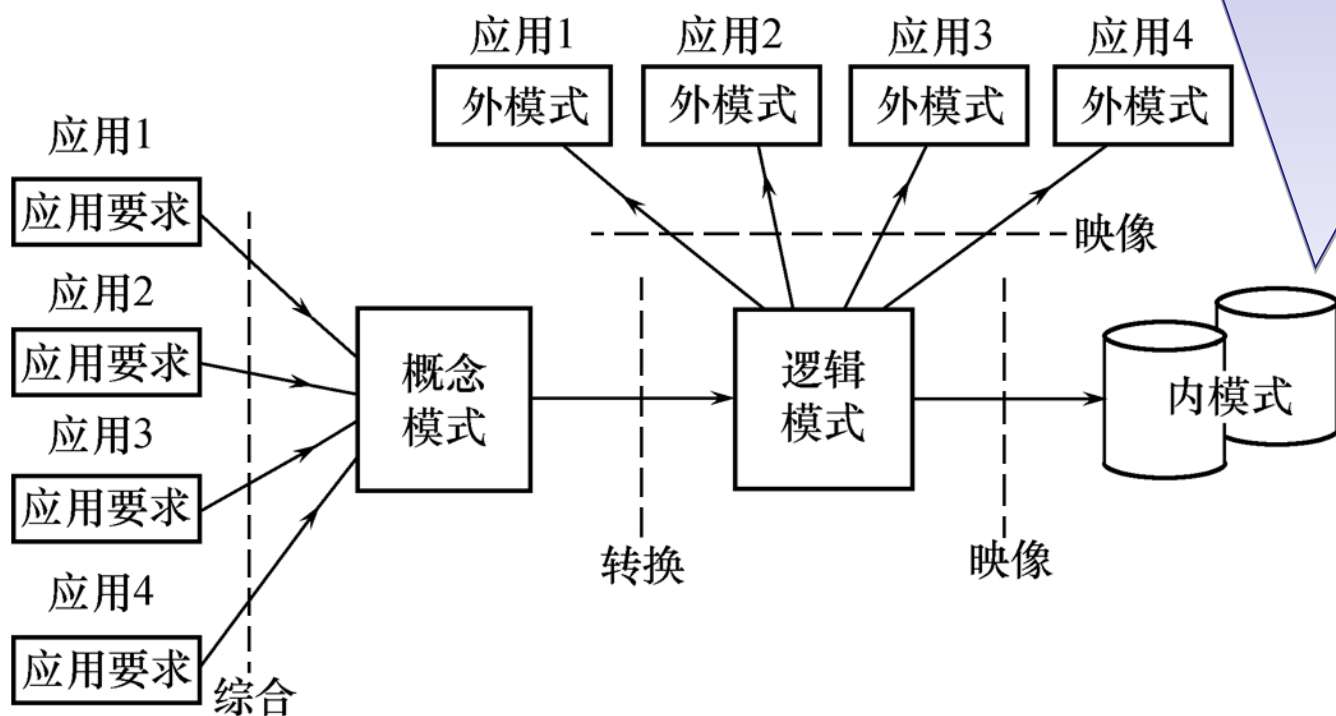
1. 首先将E-R图转换成具体的数据库产品支持的数据模型，如关系模型，形成数据库**逻辑模式**

2. 然后根据用户处理的要求、安全性的考虑，在基本表的基础上再建立必要的**视图（View）**，形成数据的**外模式**

数据库设计过程中的各级模式（续）

物理设计阶段：

根据数据库管理系统特点和处理的需要，进行物理**存储安排**，建立**索引**，形成数据库**内模式**



第七章 数据库设计



7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

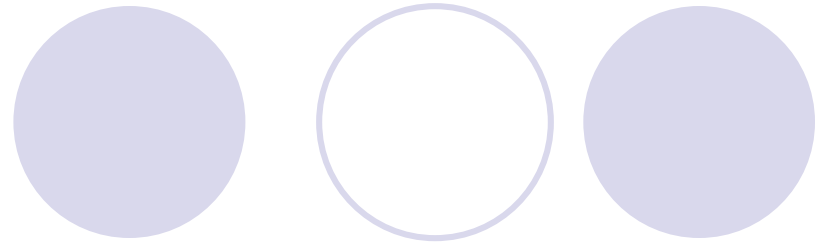
7.4 逻辑结构设计

7.5 物理结构设计

7.6 数据库实施和维护

7.7 小结

7.2 需求分析



7.2.1 需求分析的任务

7.2.2 需求分析的方法

7.2.3 数据字典

需求分析

- 需求分析就是分析用户的需要与要求
 - 需求分析是设计数据库的起点
 - 需求分析的结果是否准确地反映了用户的实际需求，将直接影响到后面各个阶段的设计，并影响到设计结果是否合理和实用

7.2.1 需求分析的任务

- 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）工作概况
- 明确用户的**各种需求**
- 在此基础上确定**新系统的功能**
- 新系统必须充分考虑今后可能的**扩充和改变**

需求分析的重点

- 需求分析的重点是“数据”和“处理”
- 信息要求
 - 用户需要从数据库中获得信息的内容与性质
 - 由用户的信息要求可以导出数据要求，即在数据库中需要存储哪些数据
- 处理要求
 - 对处理功能的要求
 - 对处理的响应时间的要求
 - 对处理方式的要求(批处理/联机处理)
- 安全性与完整性要求

需求分析的难点

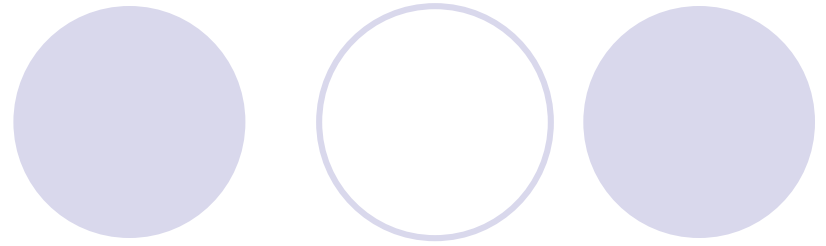
● 确定用户最终需求的难点

- 用户缺少计算机知识，不能准确地表达自己的需求，他们所提出的需求往往不断地变化。
- 设计人员缺少用户的专业知识，不易理解用户的真正需求，甚至误解用户的需求

● 解决方法

- 设计人员必须不断深入地与用户进行交流，才能逐步确定用户的实际需求

7.2 需求分析



7.2.1 需求分析的任务

7.2.2 需求分析的方法

7.2.3 数据字典

7.2.2 需求分析的方法

- 调查清楚用户的实际需求并进行初步分析
- 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

- 部门的组成情况
- 各部门的职责等

(2) 调查各部门的业务活动情况

- 各个部门输入和使用什么数据
- 如何加工处理这些数据
- 输出什么信息
- 输出到什么部门
- 输出结果的格式是什么

调查用户需求的具体步骤

(3) 熟悉业务活动，协助用户明确对新系统的各种要求

- 信息要求
- 处理要求
- 完全性与完整性要求

(4) 确定新系统的边界

- 确定哪些功能由计算机完成或将来准备让计算机完成
- 确定哪些活动由人工完成

由计算机完成的功能就是新系统应该实现的功能

常用调查方法

(1) 跟班作业

- 通过亲身参加业务工作了解业务活动的情况

(2) 开调查会

- 通过与用户座谈来了解业务活动情况及用户需求

(3) 请专人介绍

(4) 询问

- 对某些调查中的问题，可以找专人询问

(5) 设计调查表请用户填写

- 调查表设计合理，则很有效

(6) 查阅记录

- 查阅与原系统有关的数据记录

常用调查方法

- 做需求调查时，往往需要同时采用多种方法
 - 无论使用何种调查方法，都必须有用户的积极参与和配合
 - 设计人员应该和用户取得共同的语言，帮助不熟悉计算机的用户建立数据库环境下的共同概念，并对设计工作的最后结果共同承担责任

进一步分析和表达用户需求

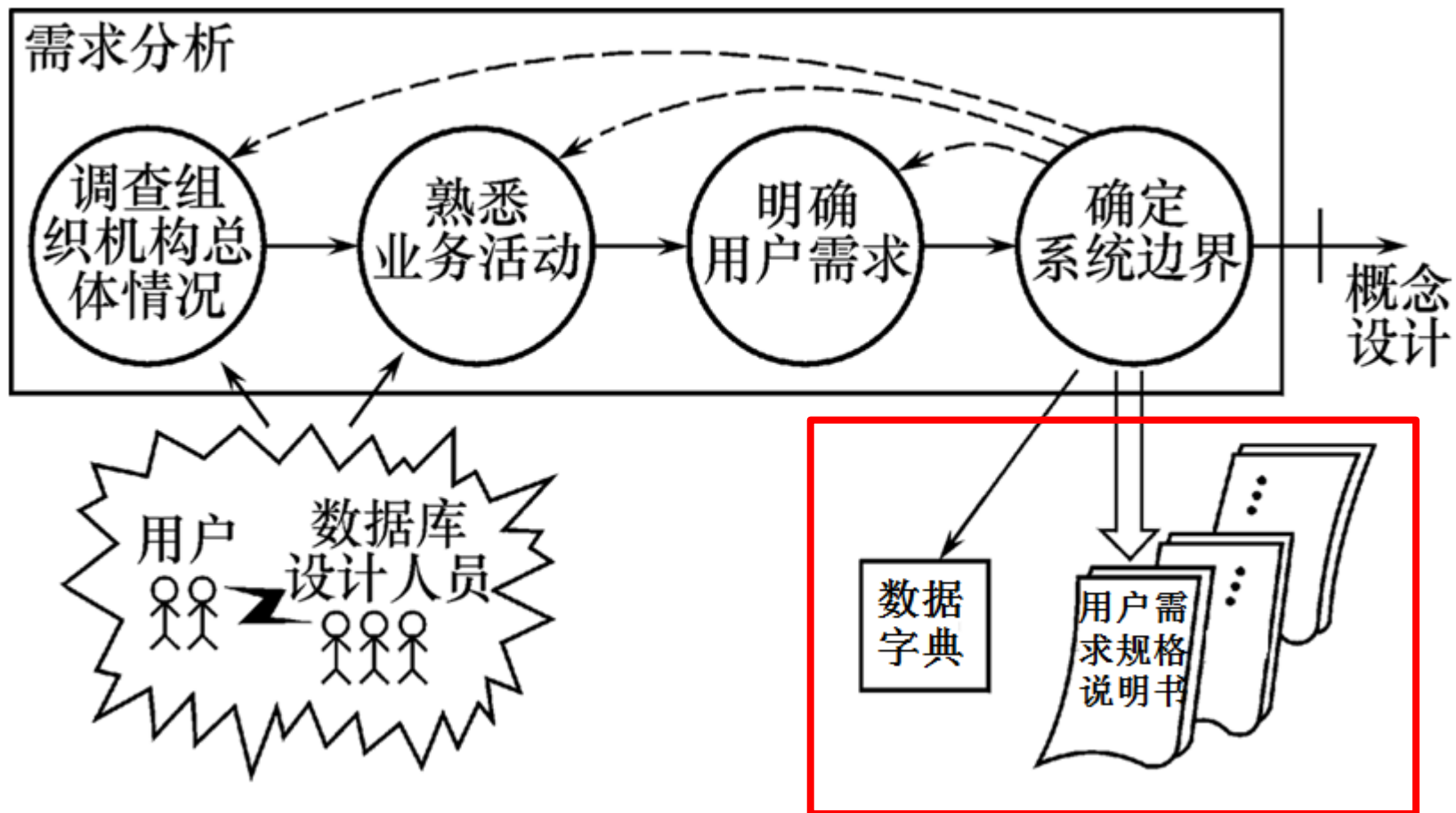
- 分析方法

- 结构化分析（**Structured Analysis, SA**）

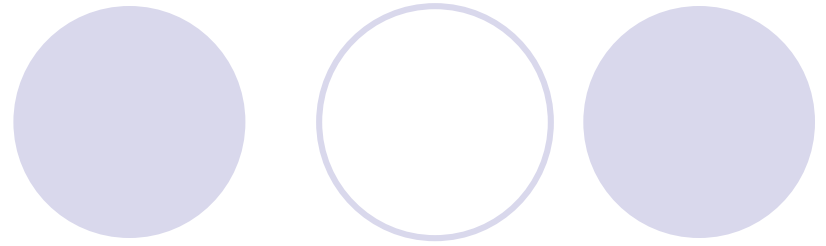
- 面向对象分析（**OOA**方法）

- 对用户需求进行分析与表达后，需求分析报告必须提交给用户，征得用户的认可

需求分析过程



7.2 需求分析



7.2.1 需求分析的任务

7.2.2 需求分析的方法

7.2.3 数据字典

7.2.3 数据字典

- 数据字典是关于数据库中数据的描述，即**元数据**，不是数据本身
- 数据字典在**需求分析阶段建立**，在数据库设计过程中**不断修改、充实、完善**
- 数据字典是进行详细的数据收集和数据分析所获得的主要成果

注意： 和关系数据库管理系统中数据字典的区别和联系

数据字典

- 数据字典的内容
 - 数据项
 - 数据结构
 - 数据流
 - 数据存储
 - 处理过程
- 数据项是数据的**最小组成单位**
- 若干个数据项可以组成一个数据结构
- 数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容。

1. 数据项

- 数据项是**不可再分**的数据单位

- 对数据项的描述

数据项描述 = { 数据项名, 数据项含义说明, 别名,
数据类型, 长度, 取值范围, 取值含义,
与其他数据项的逻辑关系, 数据项之间的
联系 }

取值范围、与其他数据项的逻辑关系定义了数据的完整性约束条件

数据项之间的联系：数据依赖

2. 数据结构

- 数据结构反映了数据之间的**组合关系**。
- 一个数据结构可以由若干个数据项组成，也可以由若干个数据结构组成，或由若干个数据项和数据结构混合组成。
- 对数据结构的描述

数据结构描述 = { 数据结构名, 含义说明,

组成: { 数据项或数据结构 } }

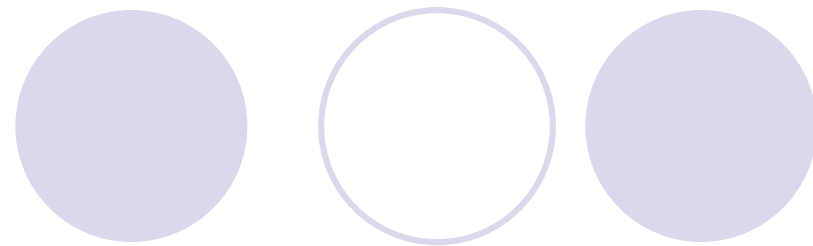
3. 数据流

- **数据流**是数据结构在系统内**传输的路径**。
- 对数据流的描述
数据流描述 = { 数据流名, 说明, 数据流来源,
数据流去向, 组成: { 数据结构 },
平均流量, 高峰期流量 }
 - 数据流来源是说明该数据流来自哪个过程(存储)
 - 数据流去向是说明该数据流将到哪个过程(存储)去
 - **平均流量**是指在单位时间（每天、每周、每月等）里的传输次数
 - **高峰期流量**则是指在高峰时期的数据流量

4. 数据存储

- 数据存储是数据结构停留或保存的地方，也是数据流的来源和去向之一。
- 可以是手工文档，也可以是计算机文档
- 数据存储描述={数据存储名,说明,编号,输入的数据流,输出的数据流,组成:{数据结构},数据量,存取频度,存取方式}
 - 存取频度：每小时、每天或每周存取次数等信息
 - 存取方法：批处理 / 联机处理；检索 / 更新；顺序检索 / 随机检索
 - 输入的数据流：数据来源
 - 输出的数据流：数据去向

5. 处理过程



- 具体处理逻辑一般用判定表或判定树来描述
- 数据字典中保存处理过程的**说明性信息描述**

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 } }

处理过程（续）

- 简要说明：主要说明该处理过程的功能及处理要求
 - 功能：该处理过程用来做什么
 - 处理要求：**处理频度要求**(如单位时间里处理多少事务)，多少数据量，**响应时间要求**等
 - 处理要求是后面物理设计的输入及性能评价的标准

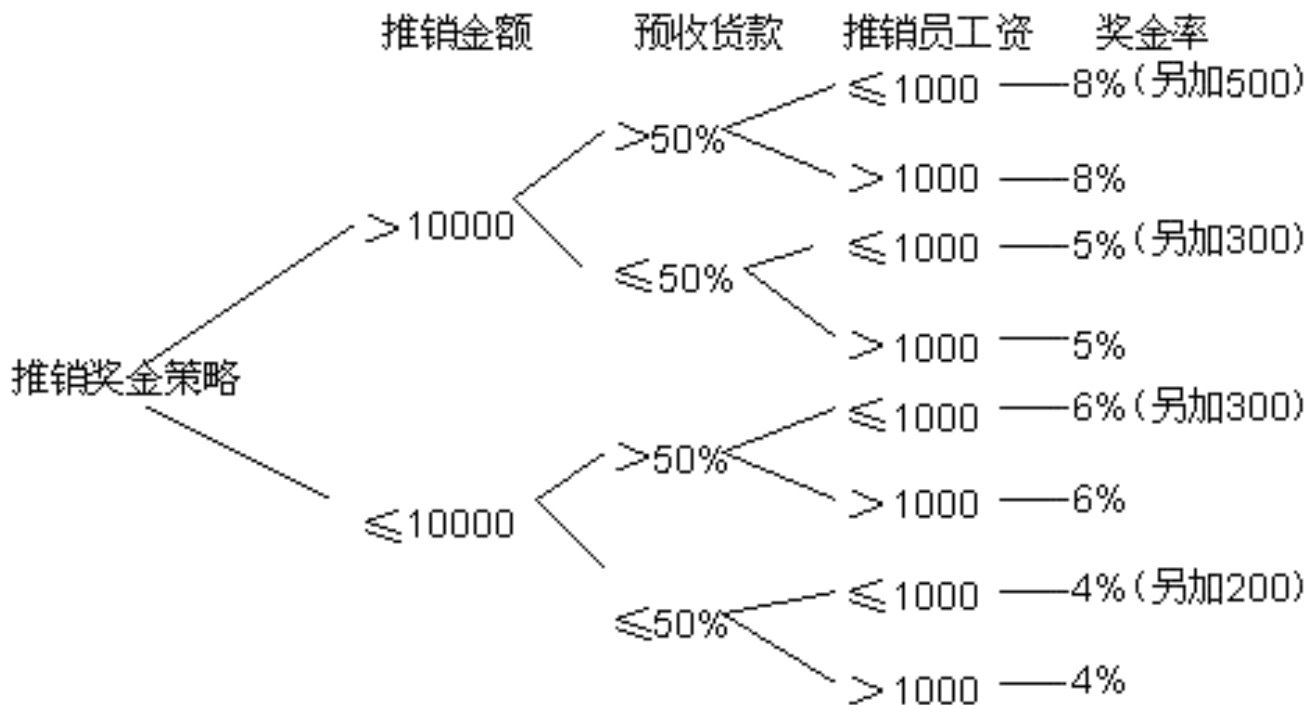
判定表

- 判定表是分析和表达多逻辑条件下执行不同操作的情况的工具。



判定树

- 又称决策树，是一种描述处理的图形工具，适合描述问题处理中具有多个判断，而且每个决策与若干条件有关。



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。

需求分析小结

- 把需求收集和分析作为数据库设计的第一阶段是十分重要的。
- 第一阶段收集的基础数据（用数据字典来表达）是下一步进行概念设计的基础。
- 强调两点
 - （1）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**
 - （2）必须强调用户的参与

第七章 数据库设计

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

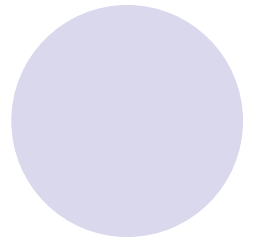
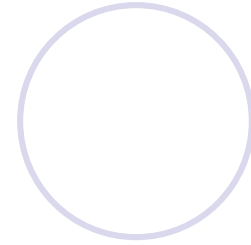
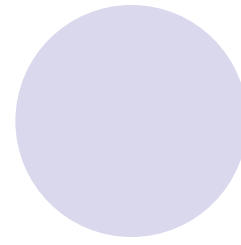
7.4 逻辑结构设计

7.5 物理结构设计

7.6 数据库实施和维护

7.7 小结

7.3 概念结构设计



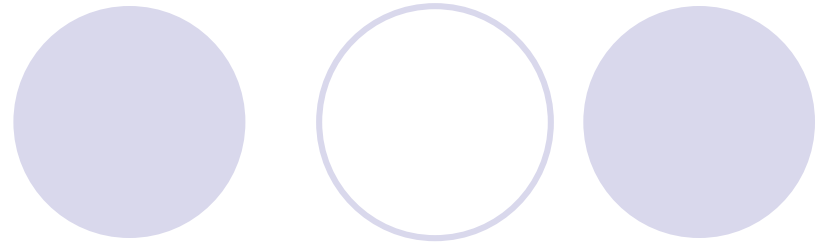
7.3.1 概念模型

7.3.5 概念结构设计

7.3.1 概念模型

- 将需求分析得到的用户需求抽象为信息结构（即概念模型）的过程就是概念结构设计
- 概念模型的特点
 - （1）能真实、充分地反映现实世界，是现实世界的一个真实模型。
 - （2）易于理解，从而可以用它和不熟悉计算机的用户交换意见。
 - （3）易于更改，当应用环境和应用要求改变时，容易对概念模型修改和扩充。
 - （4）易于向逻辑数据模型转换
- 描述概念模型的工具
 - E-R模型

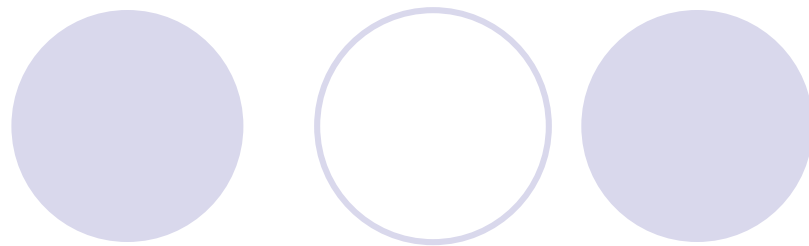
7.3 概念结构设计



7.3.1 概念结构

7.3.5 概念结构设计

7.3.5 概念结构设计



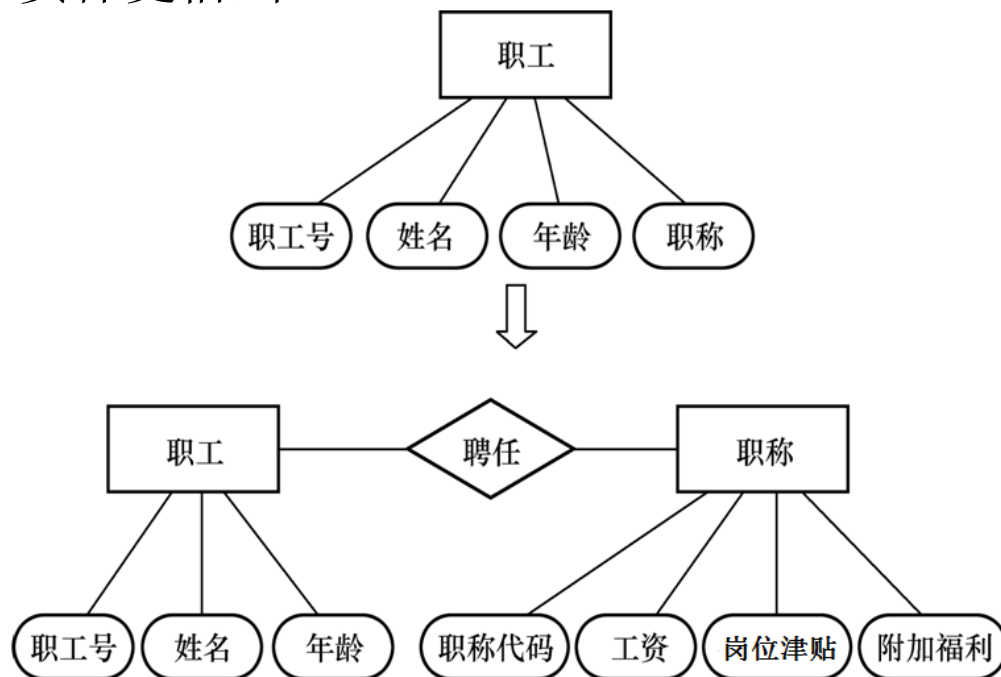
1. 实体与属性的划分原则

- 为了简化E-R图的处置，现实世界的事物能作为属性对待的，尽量作为属性对待。
- 两条准则：
 - (1) 作为属性，**不能再具有需要描述**的性质。属性必须是**不可分**的数据项，不能包含其他属性。
 - (2) **属性不能与其他实体具有联系**，即E-R图中所表示的联系是实体之间的联系。

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。

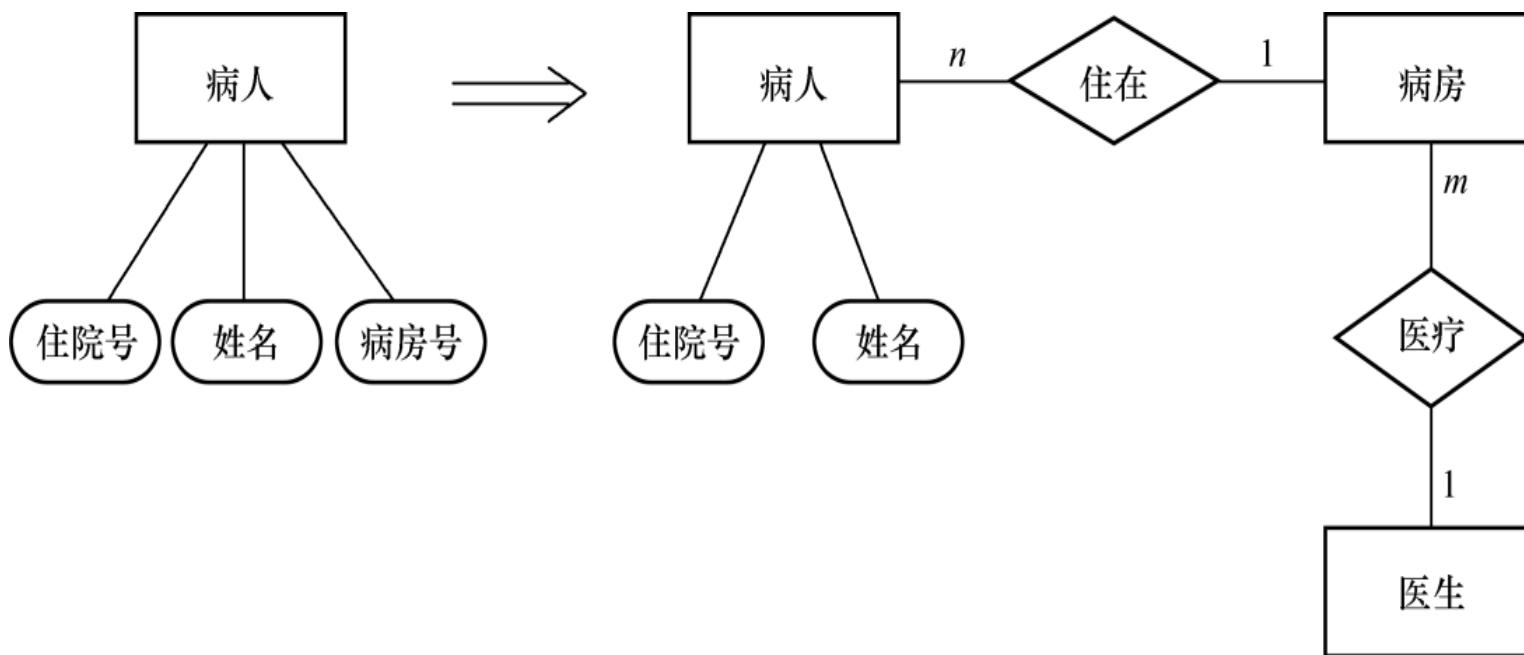
- 职称如果没有与工资、福利挂钩，根据准则（1）可以作为职工实体的属性
- 如果不同的职称有不同的工资、住房标准和不同的附加福利，则职称作为一个实体更恰当



概念结构设计（续）

[例2] 在医院中，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；

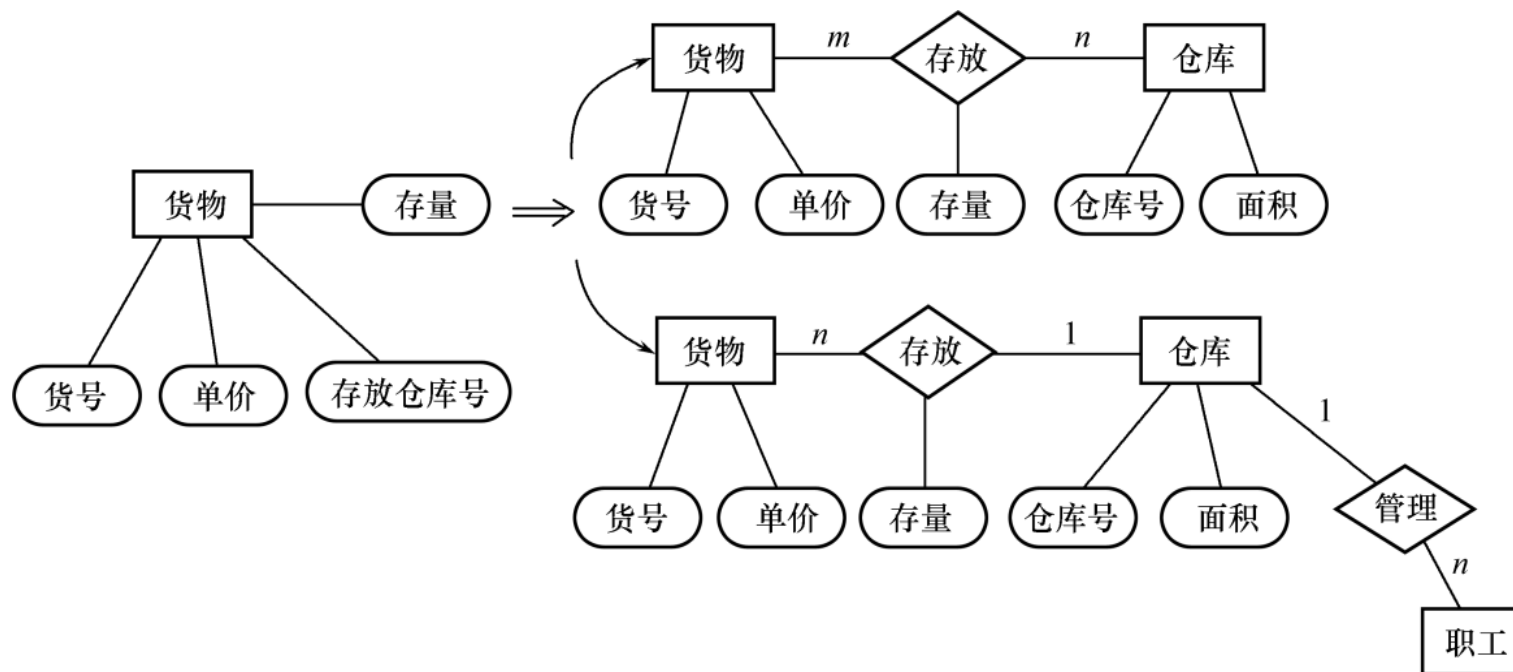
如果病房还要与医生实体发生联系，即一个医生负责几个病房的病人的医疗工作，则根据准则（2）病房应作为一个实体。



概念结构设计（续）

[例3] 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。



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

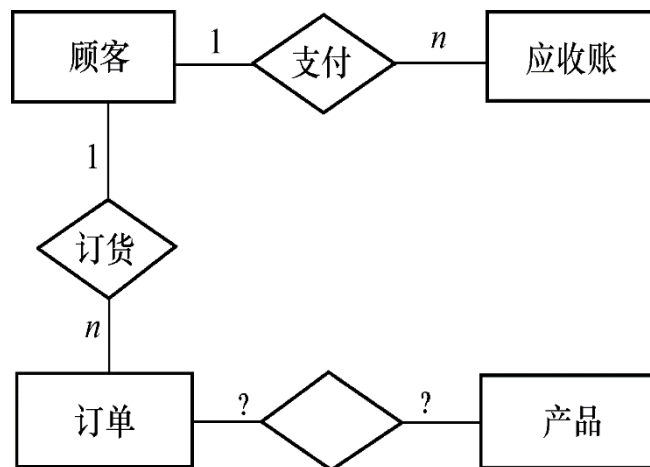
❖ 销售管理子系统的主要功能：

- 处理顾客和销售员送来的订单
- 工厂根据订货安排生产
- 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

概念结构设计（续）

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

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



概念结构设计（续）

- （2）原订单和产品的联系实际上是订单细节和产品的联系。每条订货细节对应一个产品描述，订单处理时从中获得当前单价、产品重量等信息。
- （3）工厂对大宗订货给予优惠。每种产品都规定了不同订货数量的折扣，应增加一个“折扣规则”实体存放这些信息，而不应把它们放在产品实体中。

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。

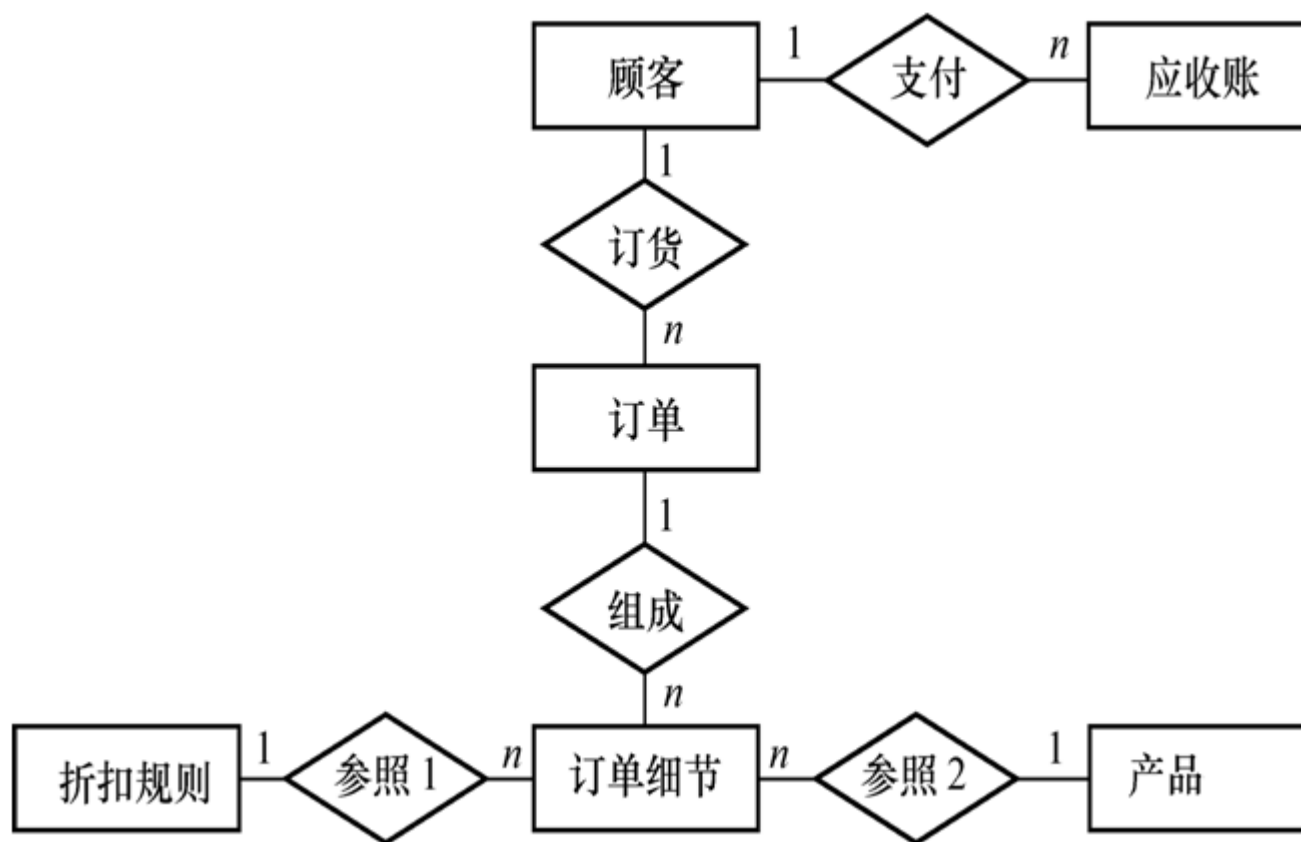


图7.23 销售管理子系统的E-R图

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，贷款限额}
 - 产品：{产品号，产品名，单价，重量}
 - 折扣规则：{产品号，订货量，折扣}

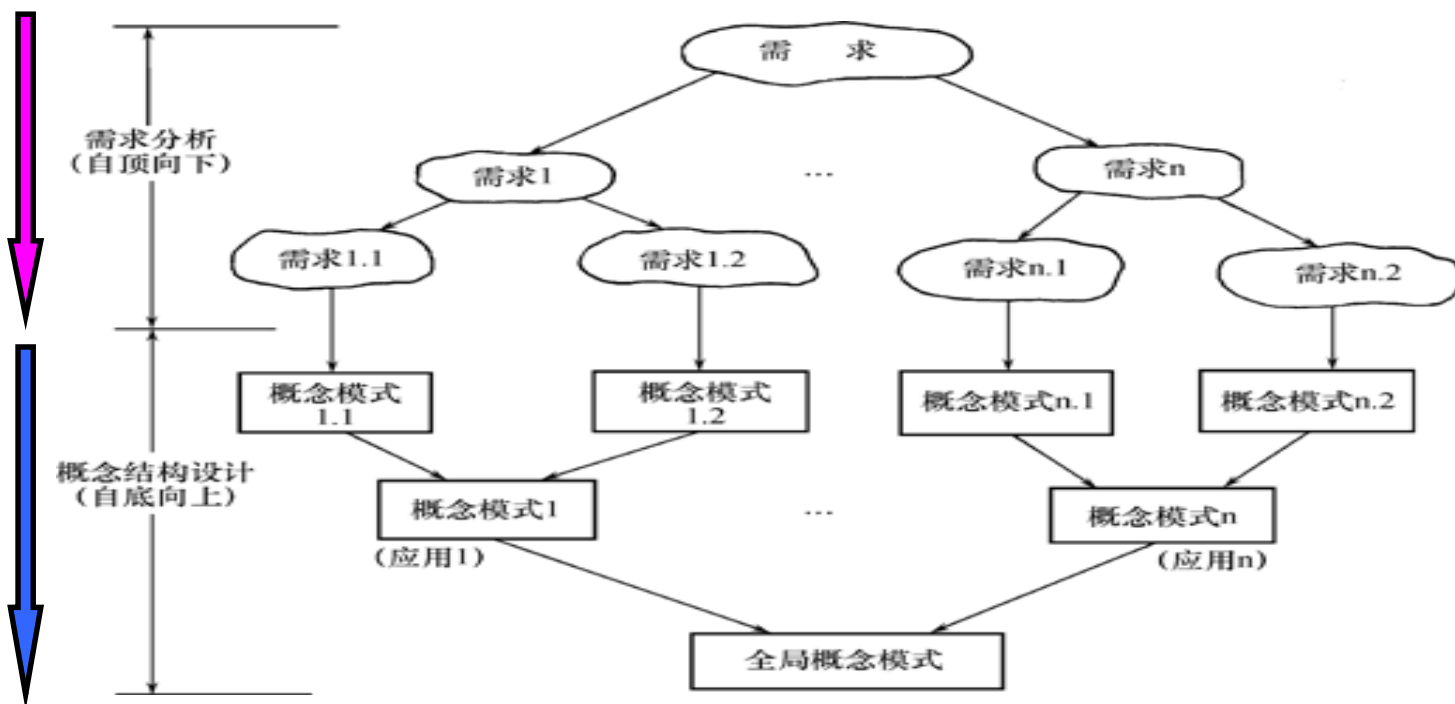
概念结构设计（续）

2. E-R图的集成

- 常用策略：

- 自顶向下地进行需求分析

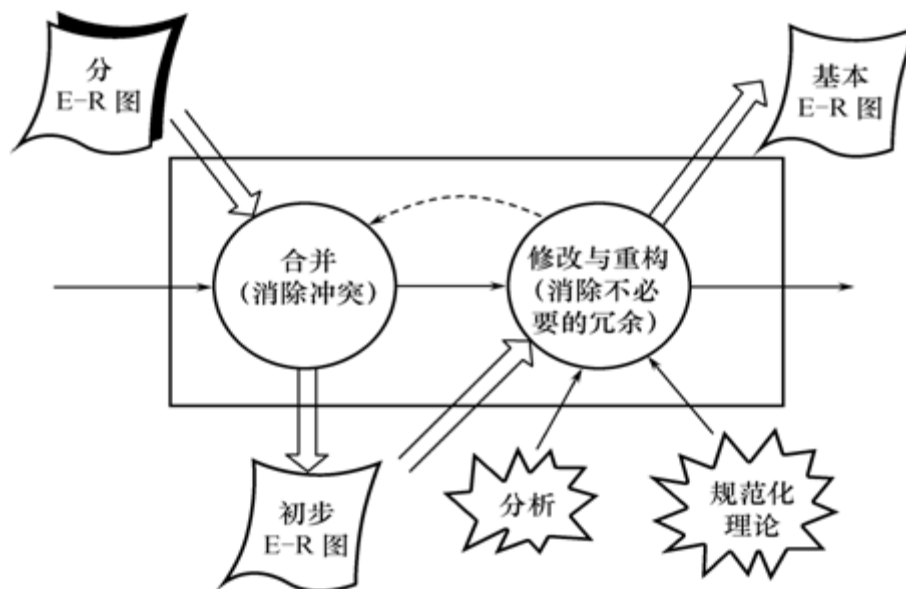
- 自底向上地设计概念结构



2. E-R图的集成

○E-R图的集成一般需要分两步

- 合并。解决各分E-R图之间的冲突，将分E-R图合并起来生成初步E-R图。
- 修改和重构。消除不必要的冗余，生成基本E-R图。



概念结构设计（续）

（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**。
- 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。
 - 例如零件号，有的部门把它定义为整数，有的部门把它定义为字符型。
 - 年龄，某些部门以出生日期形式表示职工的年龄，而另一些部门用整数表示职工的年龄。
- 属性取值单位冲突。
 - 例如，零件的重量有的以公斤为单位，有的以斤为单位，有的以克为单位。

概念结构设计（续）

②命名冲突

- **同名异义**，即不同意义的对象在不同的局部应用中具有相同的名字。
- **异名同义**（一义多名），即同一意义的对象在不同的局部应用中具有不同的名字。
 - 如对科研项目，财务科称为项目，科研处称为课题，生产管理处称为工程。
- 命名冲突
 - 可能发生在实体、联系一级上
 - 也可能发生在属性一级上
 - 通过讨论、协商等行政手段加以解决

概念结构设计（续）

③结构冲突

- 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，使同一对象具有相同的抽象。
- 同一实体在不同子系统的**E-R**图所包含的属性个数和属性排列次序不完全相同。
 - 解决方法：使该实体的属性取各子系统的**E-R**图中属性的并集，再适当调整属性的次序。

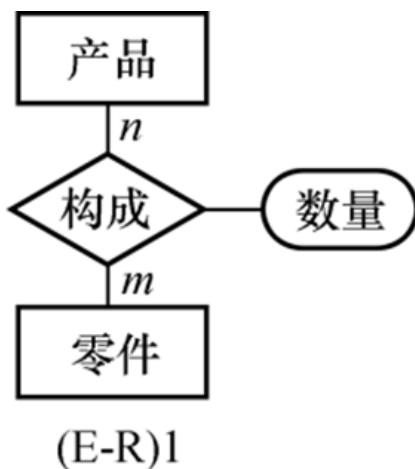
概念结构设计（续）

③结构冲突（续）

- 实体间的联系在不同的E-R图中为不同的类型。
 - 实体E1与E2在一个E-R图中是多对多联系，在另一个E-R图中是一对多联系
 - 解决方法是根据应用的语义对实体联系的类型进行综合或调整。

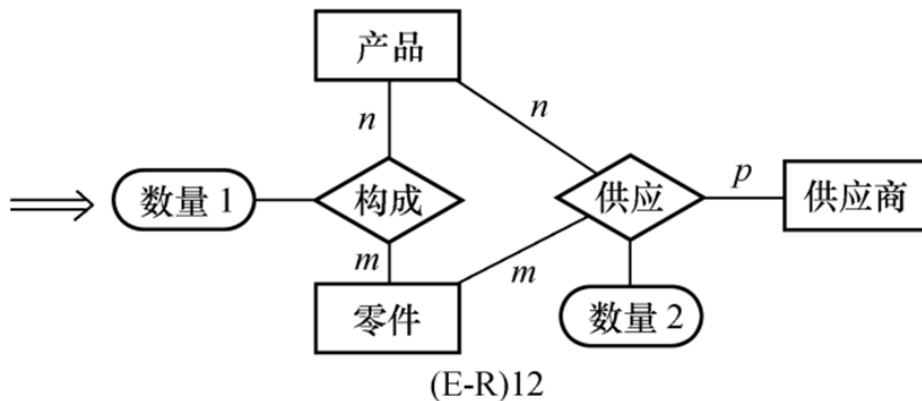
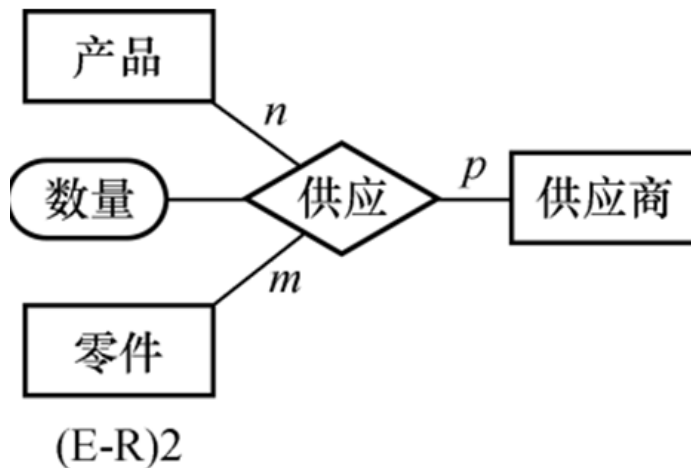
概念结构设计（续）

图7.25(a)中零件与产品之间存在多对多的联系“构成”



合并两个E-R图，如图7.25(c)

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”



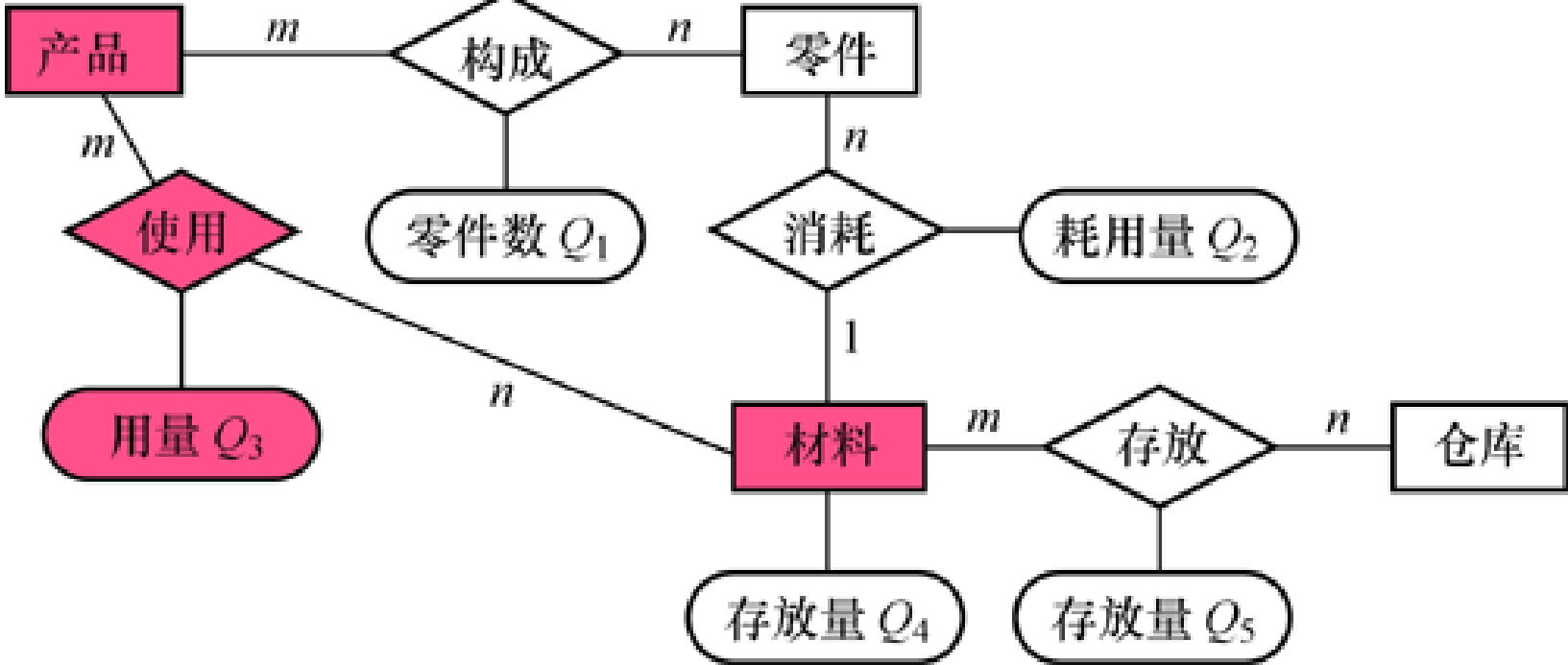
概念结构设计（续）

（2）消除不必要的冗余，设计基本E-R图

- 所谓冗余的数据是指可由基本数据导出的数据，冗余的联系是指可由其他联系导出的联系。
- 冗余数据和冗余联系容易破坏数据库的完整性
- 消除冗余主要采用分析方法，即以数据字典和数据流图为依据，根据数据字典中关于数据项之间逻辑关系的说明来消除冗余。

除冗余的方

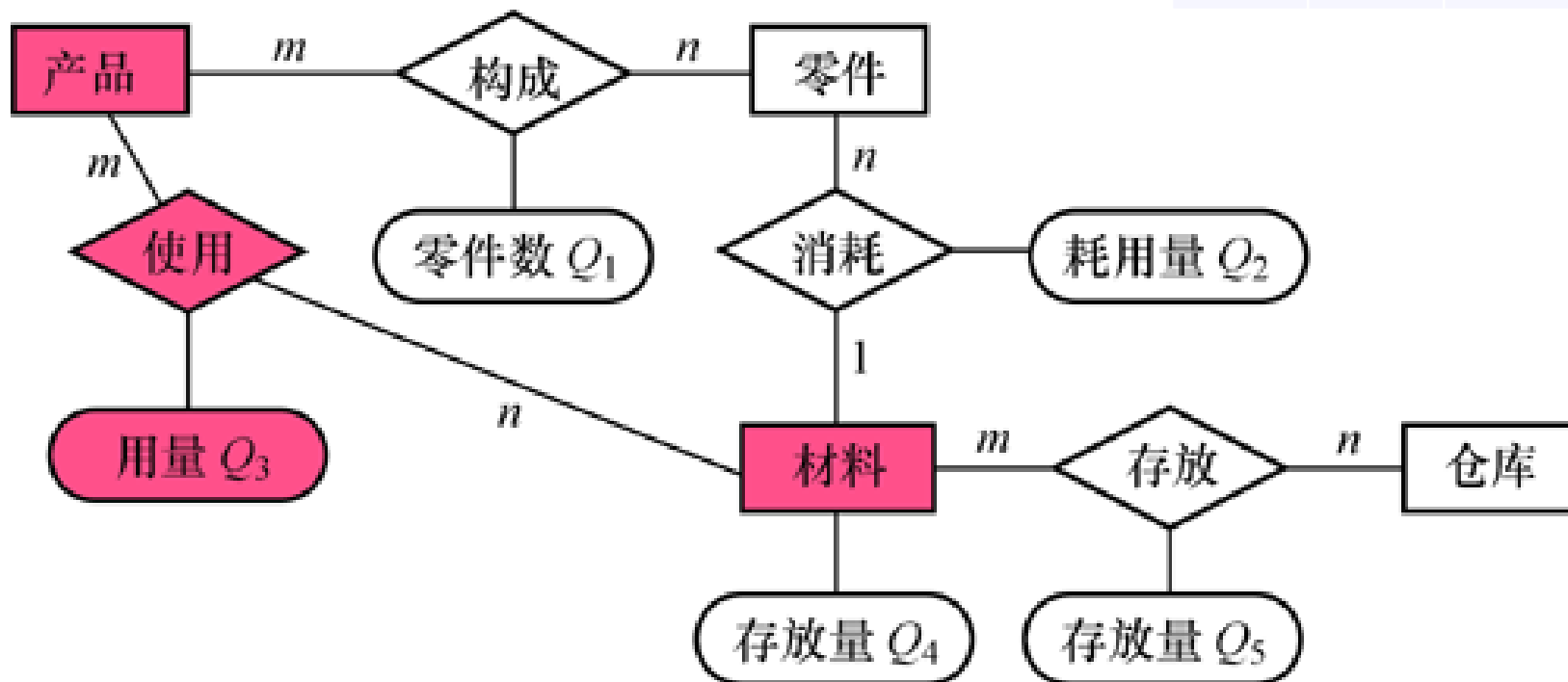
零件	材料	Q2
X	W	Q2x
Y	W	Q2y



对于某一特定**产品A**，包含与材料W相关的零件为X和Y：
 在A中使用到的X的数量为Q1x, 生产每一个X需要W 的数量为Q2x
 A中同时使用到零件的Y的数量为Q1y, 生产每一个Y需要W 的数量为Q2y
 则，A中用到W的总量Q3为 $Q1x * Q2x + Q1y * Q2y$
 以此类推，总的来说 $Q3 = \sum Q1 * Q2$

消除冗余的方法（续）

材料	仓库	Q5
A	X	Q5x
A	Y	Q5y



对于某一特定材料A，分别存放在仓库X和Y：
 在仓库X中的存放量为 $Q5x$ ，
 在仓库Y中的存放量为 $Q5y$ ，
 则，A的总存放量 $Q4$ 为 $Q5x+Q5y$
 以此类推，总的来说 $Q4=\sum Q5$

消除冗余的方法（续）

○效率VS冗余信息

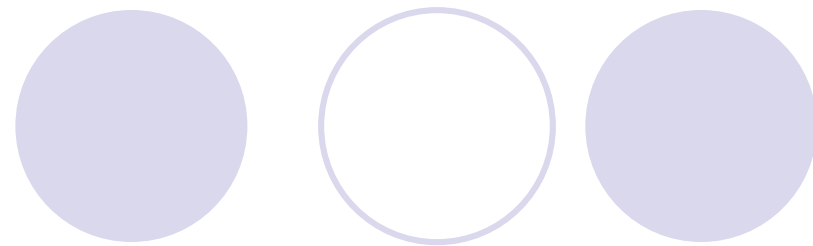
- 需要根据用户的整体需求来确定

○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

验证整体概念结构



- 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

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

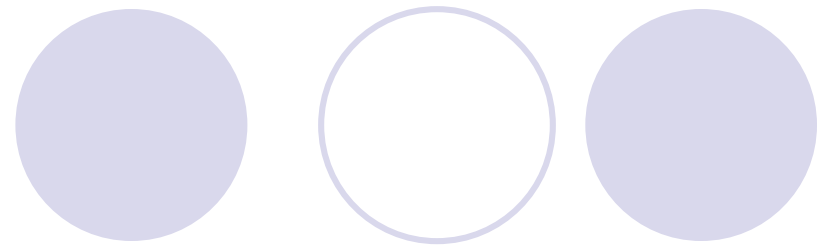
7.4 逻辑结构设计

7.5 物理结构设计

7.6 数据库的实施和维护

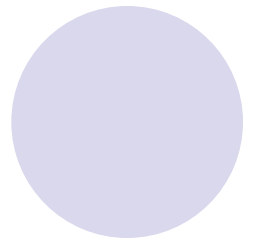
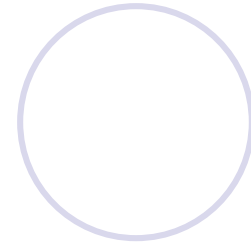
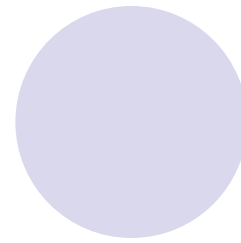
7.7 小结

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**

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

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

- 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：将实体型、实体的属性和实体型之间的联系转化为关系模式

E-R图向关系模型的转换（续）

- 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

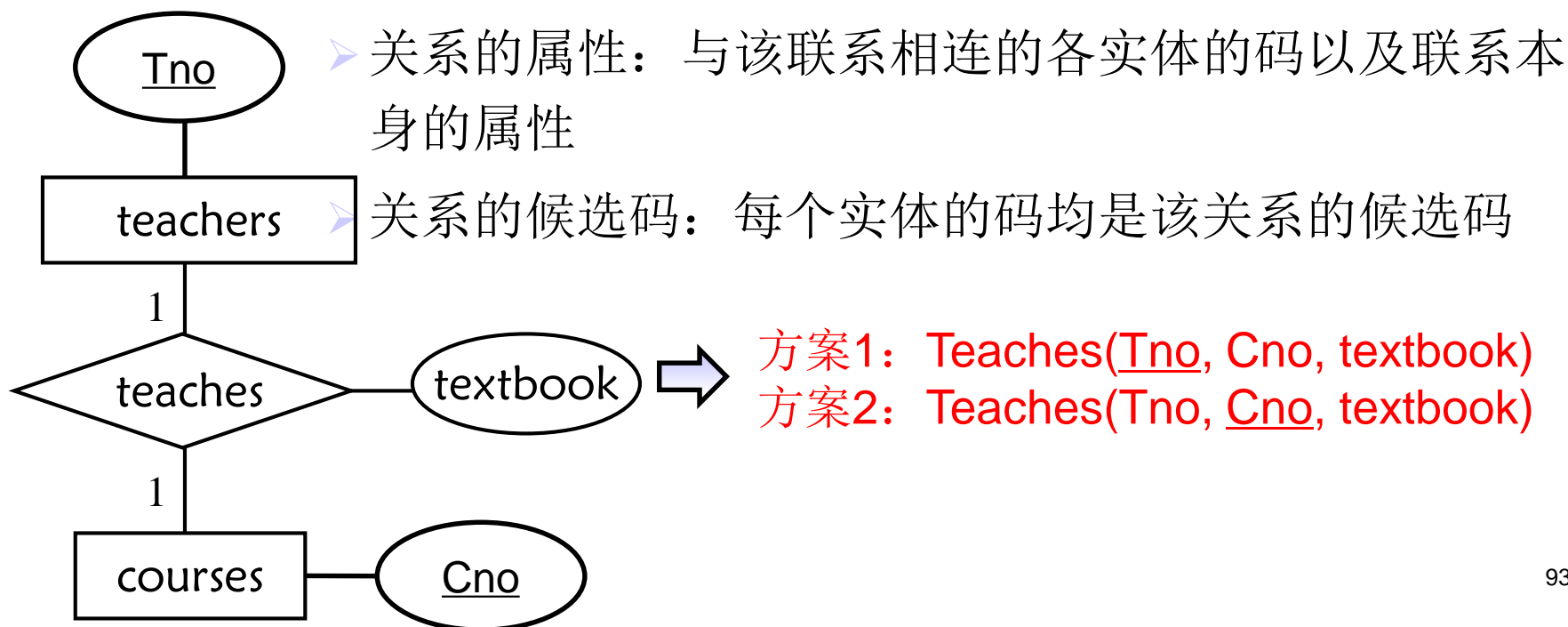
- 关系的码：实体的码

E-R图向关系模型的转换（续）

2. 实体型间的联系有以下不同情况

（1）一个1:1联系的转换

① 转换为一个独立的关系模式

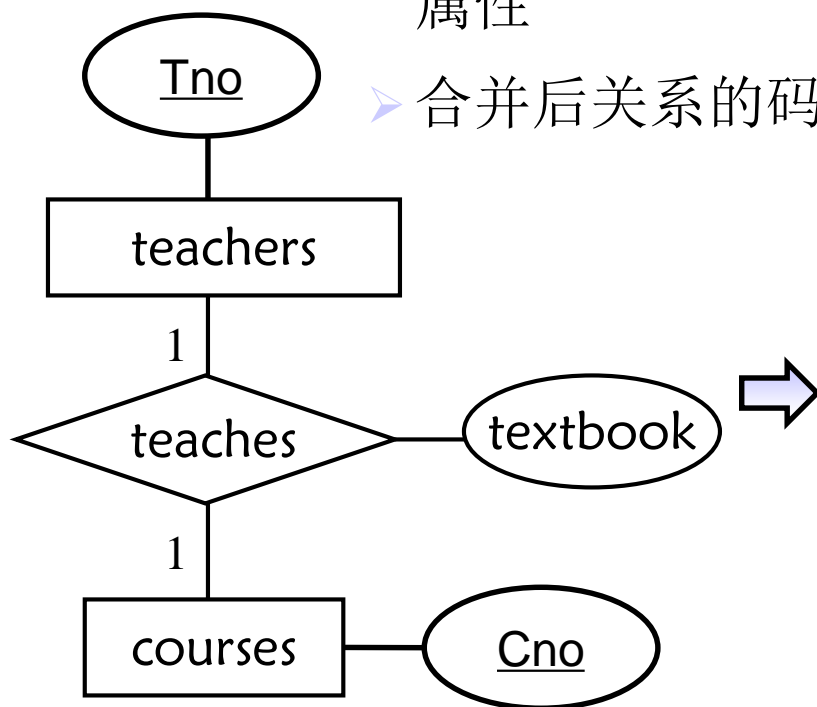


E-R图向关系模型的转换（续）

（1）一个1:1联系的转换（续）

②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



方案3：与Teachers(Tno, Tname) 合并
Teachers(Tno, Tname, Cno, textbook)

方案4：与Courses(Cno, Cname)合并
Courses(Cno, Cname, Tno, textbook)

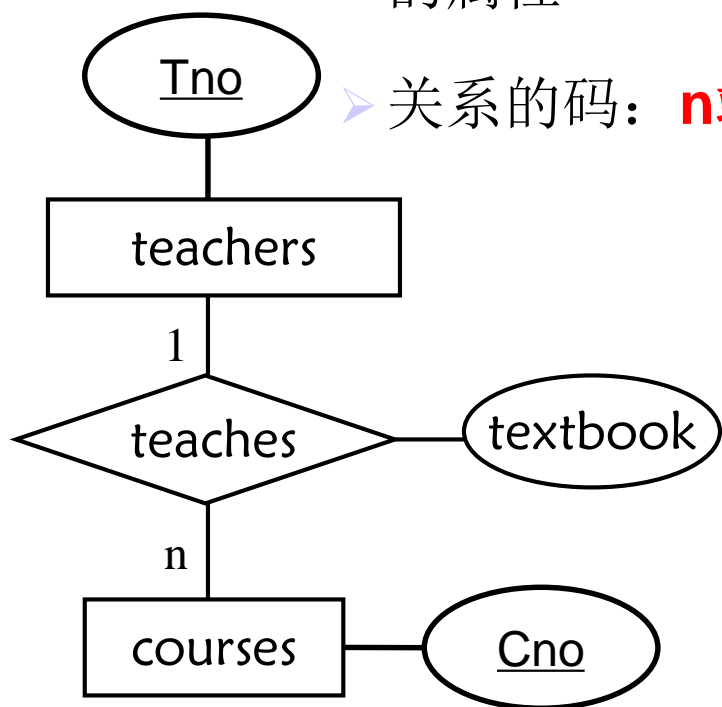
E-R图向关系模型的转换（续）

（2）一个1:n联系的转换

①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**



⇒ 方案1: Teaches(Tno, Cno, textbook)

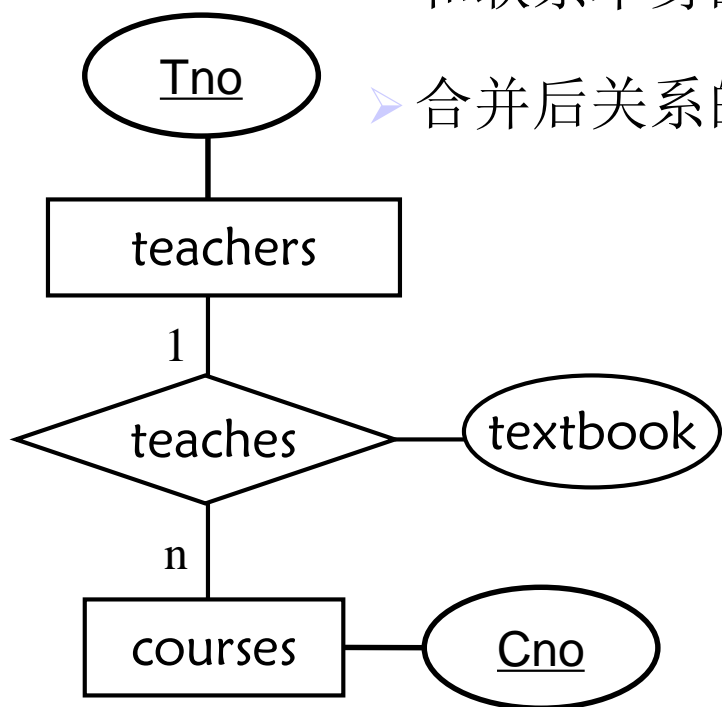
E-R图向关系模型的转换（续）

（2）一个1:n联系的转换（续）

②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

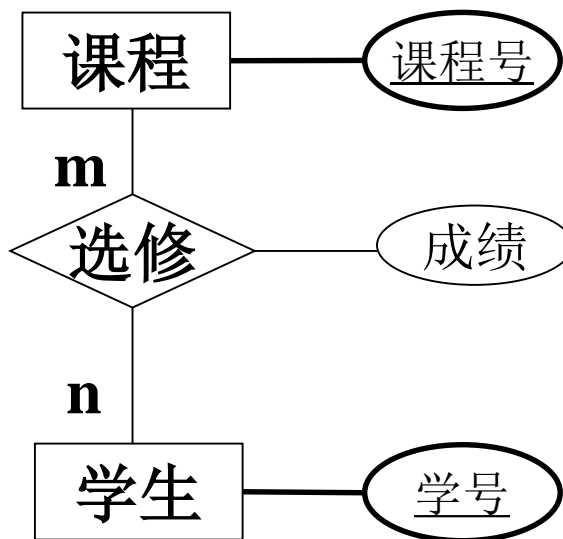


方案2：与Courses(Cno, Cname)合并
Courses(Cno, Cname, Tno, textbook)

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

- 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

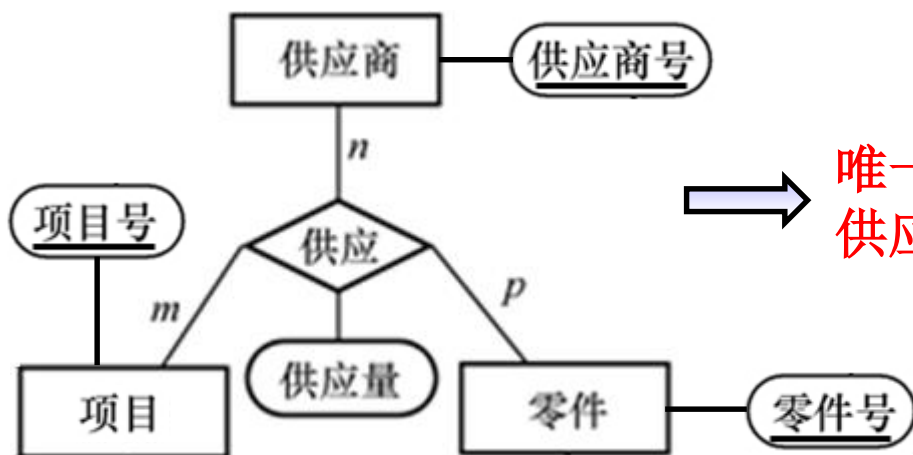


唯一方案：
选修（学号，课程号，成绩）

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。

- 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



唯一方案：

供应（供应商号, 项目号, 零件号, 供应量）

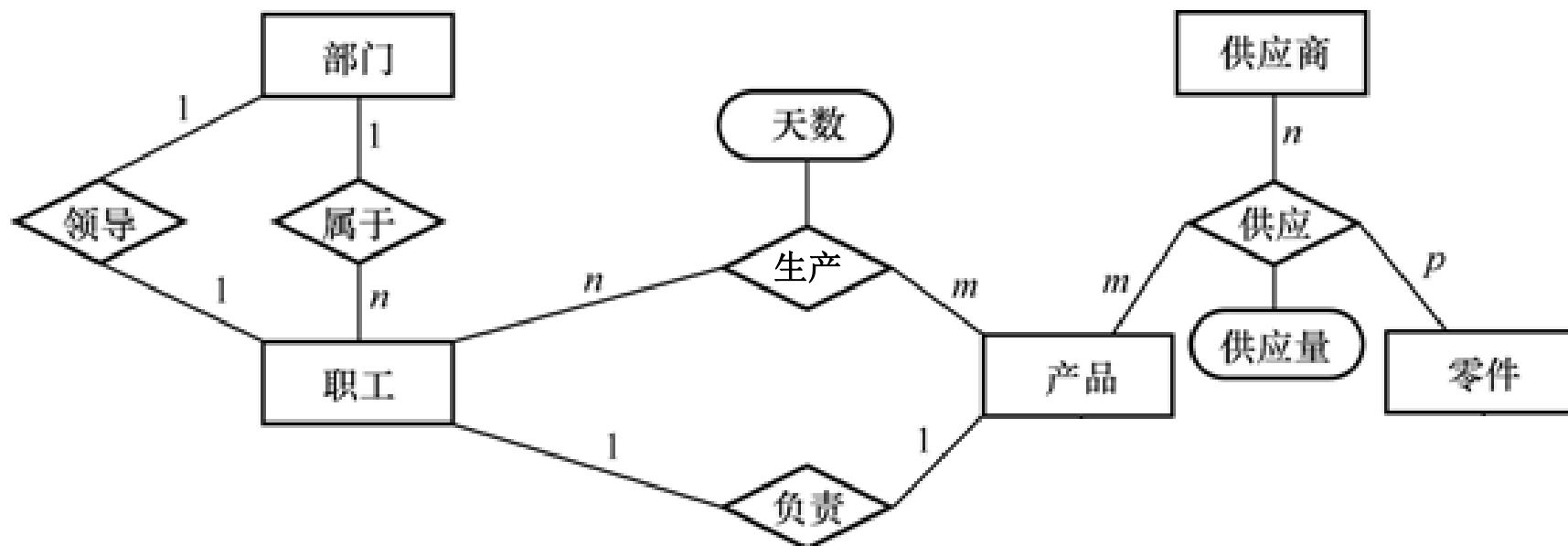
E-R图向关系模型的转换（续）

（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）
 - 适当调整属性的次序

E-R图向关系模型的转换（续）

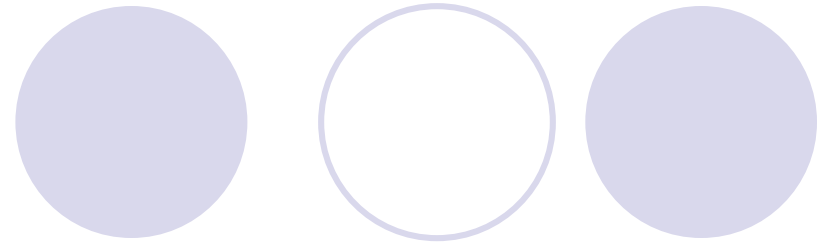
[例] 把以下的E-R图转换为关系模型



E-R图向关系模型的转换（续）

- 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，...）
 - 生产（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）
- } 合并后的关系

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

- 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）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) 按照数据依赖的理论对关系模式进行分析，考察是否存在部分函数依赖、传递函数依赖、多值依赖等，**确定各关系模式分别属于第几范式。**
- (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）

○并不是规范化程度越高的关系就越优

- 当查询经常涉及两个或多个关系模式的属性时，系统必须经常地进行连接运算
- 连接运算的代价是相当高的
- 因此在这种情况下，第二范式甚至第一范式也许是适合的。

数据模型的优化（续）

- 非BCNF的关系模式虽然会存在不同程度的更新异常，但如果在实际应用中对此关系模式只是查询，并不执行更新操作，就不会产生实际影响。
- 对于一个具体应用来说，到底规范化进行到什么程度，需要权衡响应时间和潜在问题两者的利弊才能决定

数据模型的优化（续）

（5）对关系模式进行必要分解，**提高数据操作效率和存储空间的利用率。**

○常用分解方法

- 水平分解
- 垂直分解

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。

●如何分解

- ✓ 对符合80/20的，把经常被使用的数据（约20%）水平分解出来，形成一个子关系。
- ✓ 水平分解为若干子关系，使每个事务存取的数据对应一个子关系。

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

- 垂直分解的优点

- 可以提高某些事务的效率

- 垂直分解的缺点

- 可能使另一些事务不得不执行连接操作，降低了效率

数据模型的优化（续）

○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

○进行垂直分解的方法

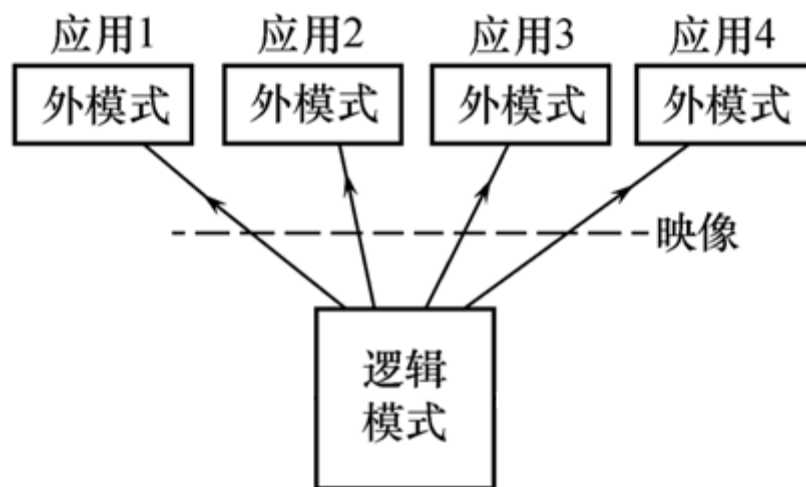
- 简单情况：直观分解
- 复杂情况：用第6章中的模式分解算法
- 垂直分解必须**不损失关系模式的语义**（保持无损连接性和保持函数依赖）

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

- 定义数据库模式主要是从系统的**时间效率**、**空间效率**、**易维护**等角度出发。
- 定义用户外模式时应该更注重考虑用户的**习惯与方便**。包括三个方面：

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的视图，以保证系统的安全性。

- 假设有关系模式产品（产品号，产品名，规格，单价，生产车间，生产负责人，产品成本，产品合格率，质量等级），可以在产品关系上建立两个视图：

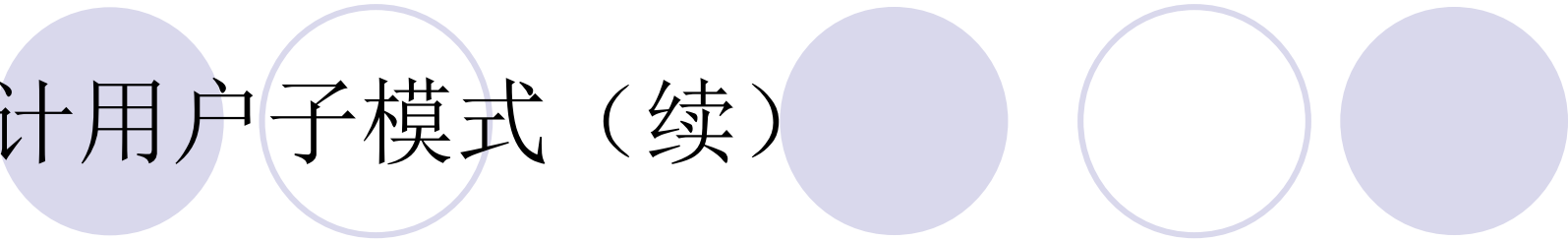
- 为一般顾客建立视图：

产品1（产品号，产品名，规格，单价）

- 为产品销售部门建立视图：

产品2（产品号，产品名，规格，单价，车间，
生产负责人）

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

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

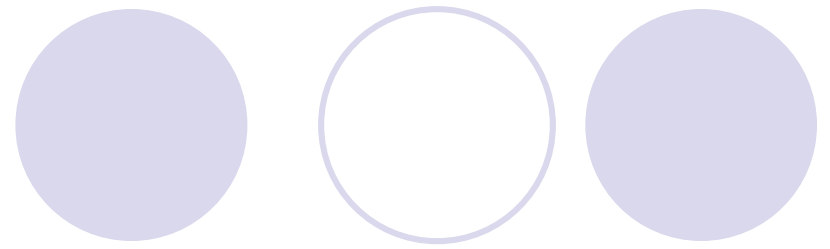
7.4 逻辑结构设计

7.5 物理结构设计

7.6 数据库的实施和维护

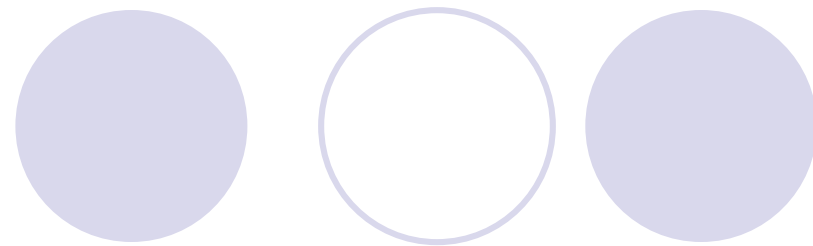
7.7 小结

7.5 物理结构设计



- 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

物理结构设计(续)

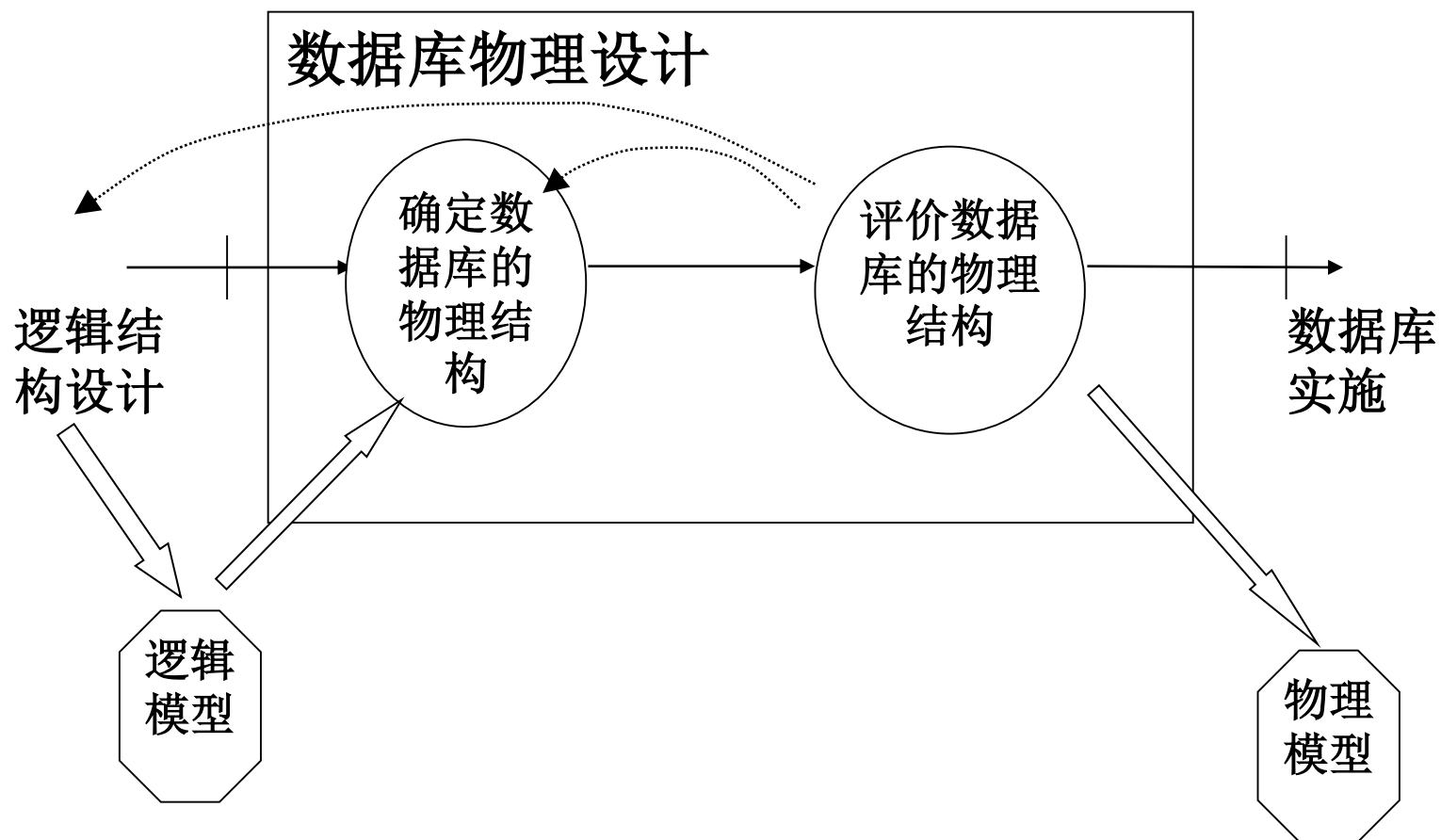


● 数据库物理设计的步骤

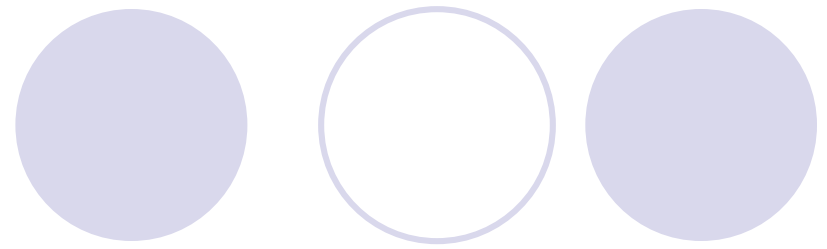
- 确定数据库的物理结构，在关系数据库中主要指**存取方法和存储结构**
- 对物理结构进行评价，评价的重点是**时间**和**空间**效率

如果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

物理结构设计(续)



7.5 物理结构设计



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

- 对要运行的事务进行详细分析，获得选择数据库物理设计所需**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**

数据库的物理设计的内容和方法（续）

- 选择数据库物理设计所需参数

- 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

数据库的物理设计的内容和方法（续）

- 选择数据库物理设计所需参数(续)

- 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

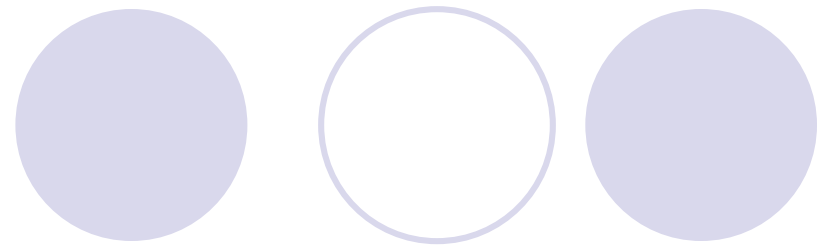
- 每个事务在各关系上运行的**频率和性能要求**

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 物理结构设计



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
- 存取方法是快速存取数据库中数据的技术

关系模式存取方法选择（续）

- DBMS常用存取方法

- 索引方法

- 目前主要是B+树索引方法

- 经典存取方法，使用最普遍

- 聚簇（Cluster）方法

- HASH方法

一、索引存取方法的选择

- 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

```
CREATE UNIQUE INDEX SCno  
ON SC(Sno ASC, Cno DESC);
```

索引存取方法的选择（续）

- 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
- 如果一个关系的更新频率很高，这个关系上定义的索引数不能太多

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相等比较选择条件中

- 该关系的大小可预知，而且不变；

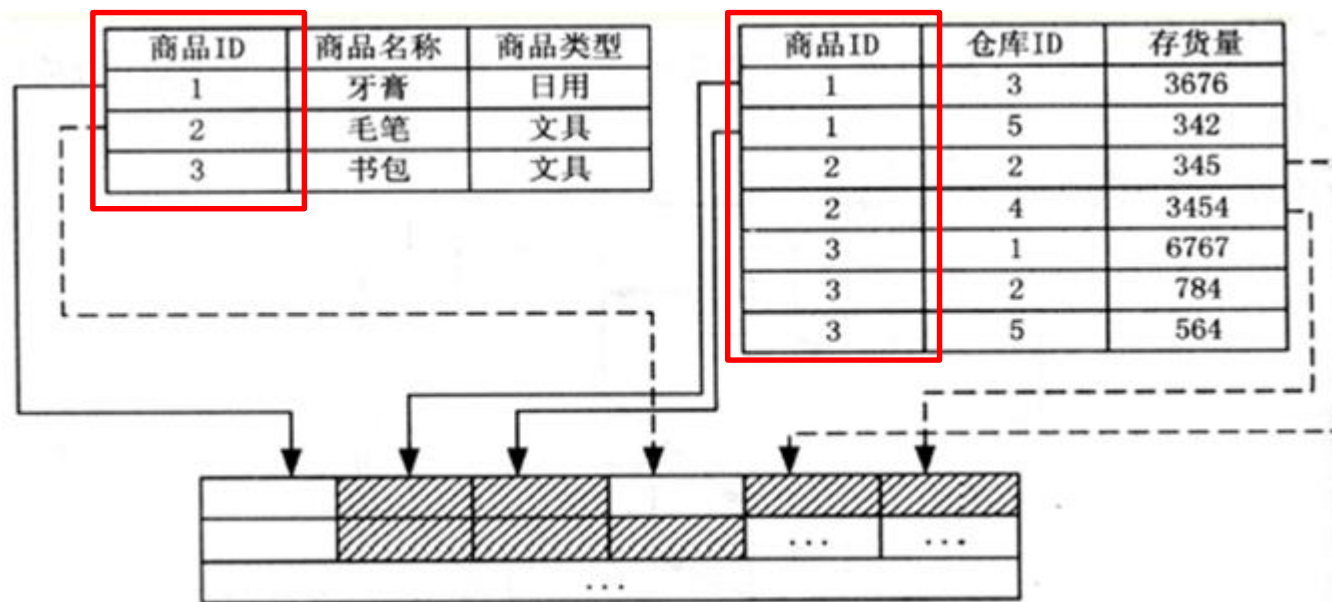
或

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



聚簇存取方法的选择（续）

- 聚簇的用途: 大大提高按聚簇码进行查询的效率

例：假设学生关系按所在系建有索引，现在要查询信息系的所有学生名单。

- 信息系的**500**名学生分布在**500**个不同的物理块上时，至少要执行**500**次I/O操作
- 如果将同一系的学生元组集中存放，则每读一个物理块可得到多个满足查询条件的元组，从而显著地减少了访问磁盘的次数

聚簇存取方法的选择（续）

- 聚簇的适用范围

- 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，这一查询涉及学生关系和选修关系的连接操作，即需要**按学号连接**这两个关系，为提高连接操作的效率，可以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。这就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）

- 一个数据库可以建立多个聚簇，但一个关系只能加入一个聚簇
- 选择聚簇存取方法，即确定需要建立多少个聚簇，每个聚簇中包括哪些关系

聚簇存取方法的选择（续）

● 设计候选聚簇

- 对经常在一起进行**连接操作**的关系可以建立聚簇
- 如果一个关系的一组属性经常出现在**相等比较**条件中，则该单个关系可建立聚簇
- 如果一个关系的一个(或一组)属性上的**值重复率**很高，则此单个关系可建立聚簇。即对应每个聚簇码值的平均元组数不太少。太少了，聚簇的效果不明显

聚簇存取方法的选择（续）

● 优化聚簇设计

- 从聚簇中删除经常进行全表扫描的关系；
- 从聚簇中删除更新操作远多于连接操作的关系；
- 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

聚簇存取方法的选择（续）

● 聚簇的局限性

○ 1. 聚簇只能提高**某些特定应用**的性能

○ 2. 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

聚簇存取方法的选择（续）

● 总结

- 当通过聚簇码进行访问或连接是该关系的主要应用，与聚簇码无关的其他访问很少或者是次要的时，可以使用聚簇。

➤ 尤其当SQL语句中包含有与聚簇码有关的**ORDER BY**，**GROUP BY**，**UNION**，**DISTINCT**等子句或短语时，使用聚簇特别有利，可以省去对结果集的排序操作

7.5 物理结构设计

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

- 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

确定数据的存放位置（续）

例：

- 如果计算机有多个磁盘或磁盘阵列，可以考虑将表 and 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

确定数据的存放位置（续）

例（续）：

- 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
 - 使用的缓冲区长度、个数
 - 存储分配参数
 -

7.5 物理结构设计

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

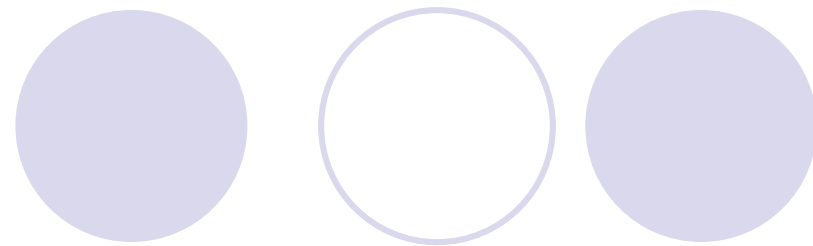
7.5.4 评价物理结构

7.5.4 评价物理结构

- 评价内容

- 对数据库物理设计过程中产生的多种方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构

评价物理结构(续)



- 评价方法（完全依赖于所选用的**DBMS**）
 - 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

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 数据库的物理设计

7.6 数据库的实施和维护

7.7 小结

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
- 物理结构设计
- 实施和维护

小结（续）

- 数据库各级模式的形成

- 数据库的各级模式是在设计过程中逐步形成的
- 需求分析阶段综合各个用户的应用需求（现实世界的需求）
- 概念设计阶段形成独立于机器特点、独立于各个DBMS产品的**概念模式**（信息世界模型），用E-R图来描述

小结（续）

- 在逻辑设计阶段将**E-R**图转换成具体的数据库产品支持的数据模型如关系模型，形成数据库**逻辑模式**。然后根据用户处理的要求、安全性的考虑，在基本表的基础上再建立必要的视图（**VIEW**）形成数据的**外模式**
- 在物理设计阶段根据**DBMS**特点和处理的需要，进行物理存储安排，设计索引，形成数据库**内模式**