

第9章 数据库设计

数据库设计

9.1 数据库设计概述

9.2 数据库设计的需求分析

9.3 数据库的概念设计

9.4 数据库的逻辑设计

9.5 数据库的物理设计

9.1 数据库设计概述

□ 数据库设计的基本任务

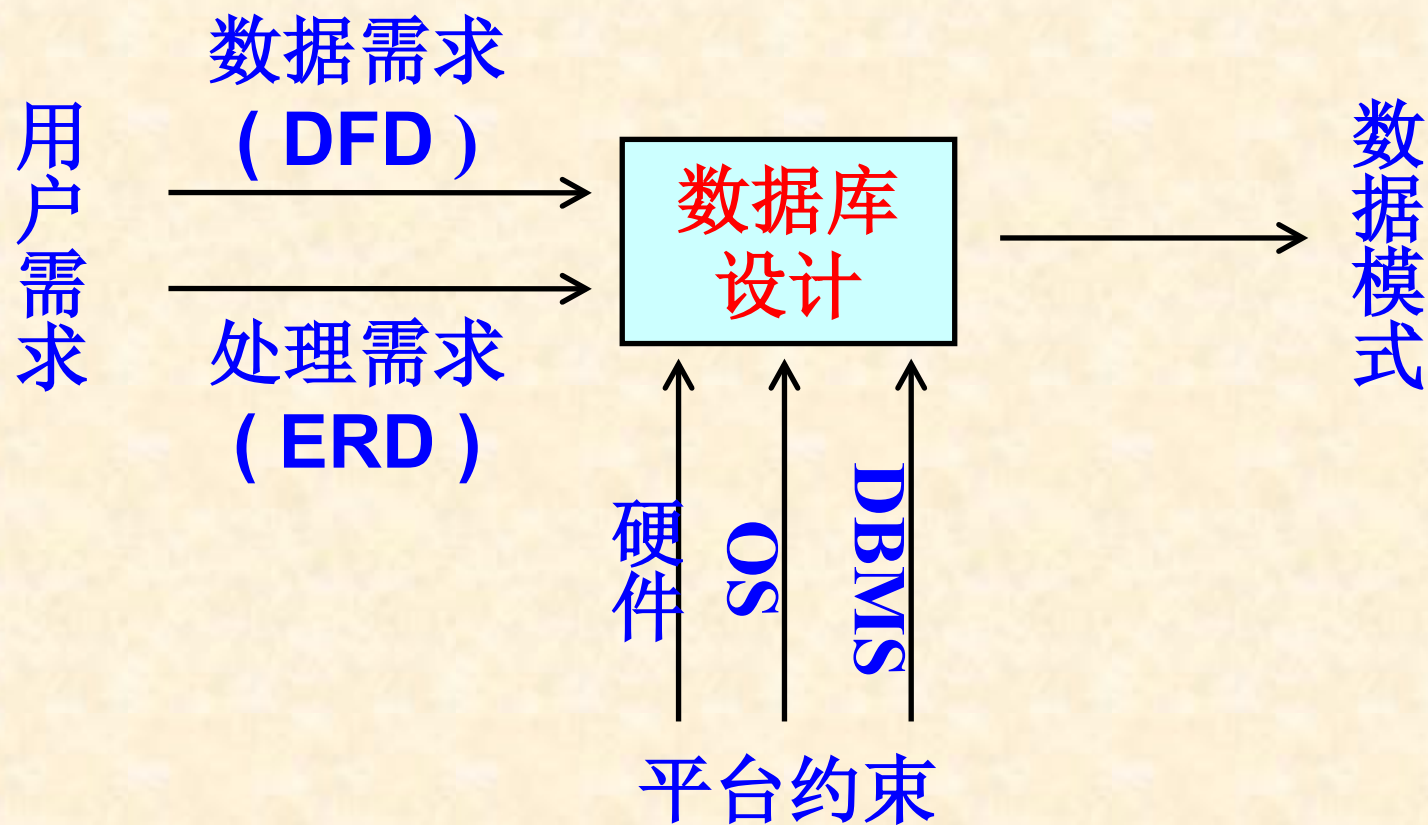
□ 数据库设计的基本方法

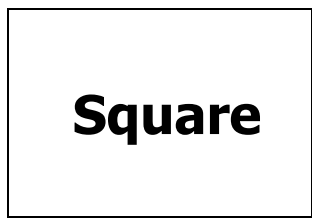
9.1 数据库设计概述

□ 数据库设计的基本任务

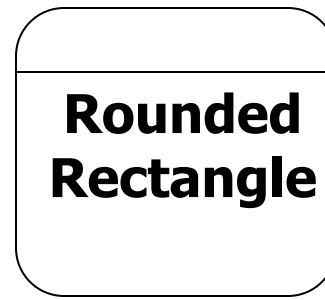
- 根据用户对象的信息需求、处理需求和数据库的支持环境（包括硬件、操作系统与DBMS）设计出数据模式。
 - 信息需求：用户的数据、结构及其要求
 - 处理需求：用户对数据的处理过程和方式
- 数据库设计即是在一定平台制约下，根据信息需求与处理需求设计出性能良好的数据模式。

数据库设计的基本任务





**Source or destination
of data**



**Process which
transforms flows
of data**

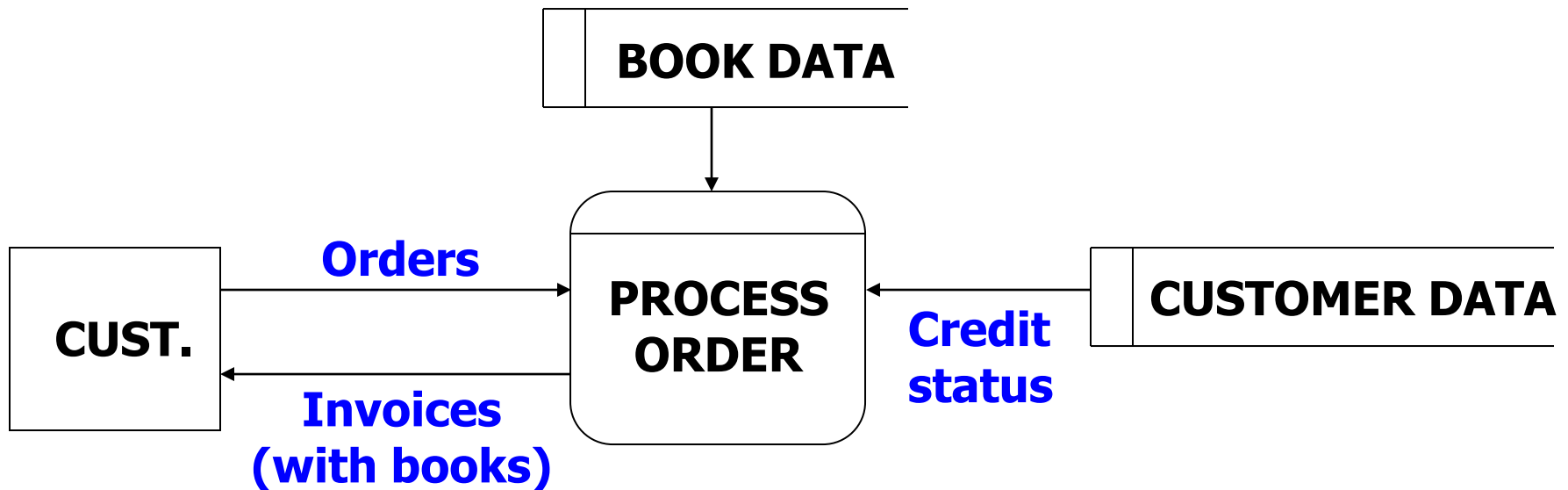


Flow of data

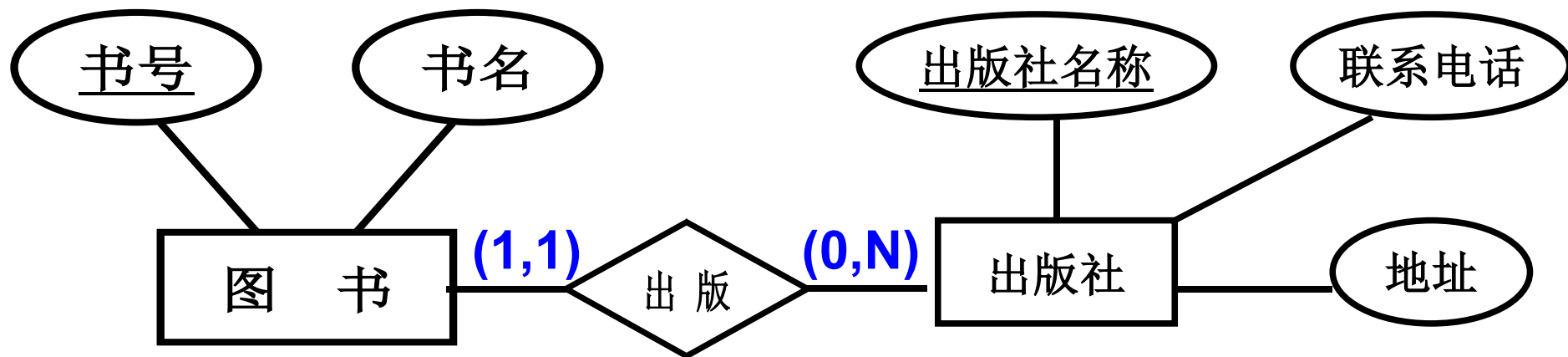


**Store of
data**

DFD symbols



Example of a data flow diagram



Example of a Entity-Relationship Diagram

9.1 数据库设计概述

□ 数据库设计的生命周期法

(1) 需求分析

(2) 概念设计

(3) 逻辑设计

(4) 物理设计

(5) 编码

(6) 测试

(7) 运行

(8) 进一步修改

数据库设计
的四个阶段



9.1 数据库设计概述

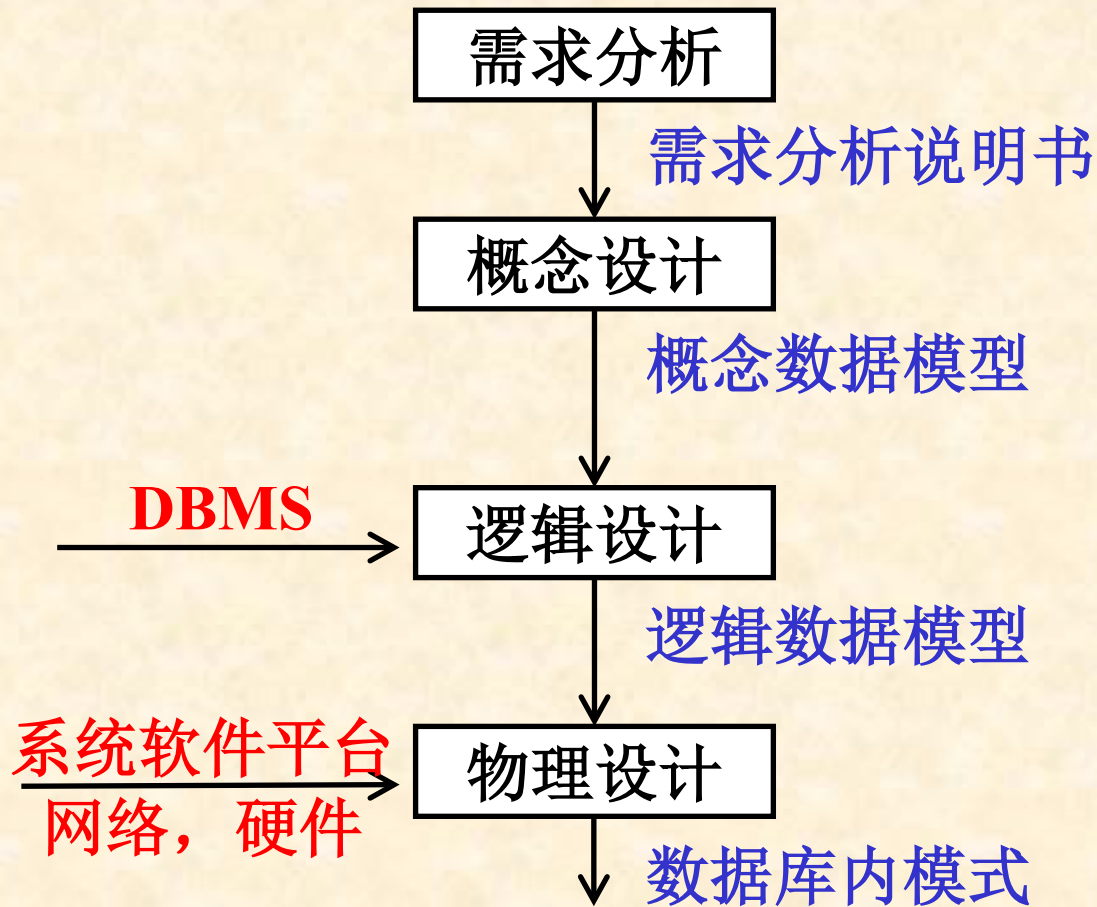


图9.1 数据库设计的四个阶段

9.1 数据库设计概述

9.2 数据库设计的需求分析

9.3 数据库的概念设计

9.4 数据库的逻辑设计

9.5 数据库的物理设计

9.2 数据库设计的需求分析

□ 需求分析

- 从调查用户单位着手，深入了解用户单位的数据流程，数据使用情况，数据的数量、流量、流向、性质，并作出分析，最终按一定规范要求以文档形式写出数据的需求说明书。

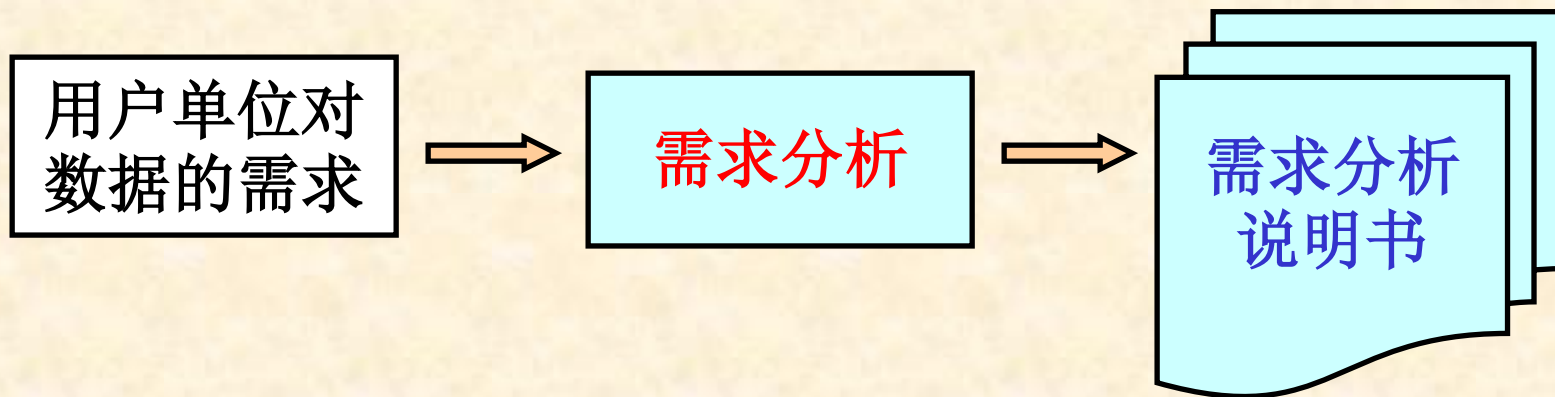


图9.2 需求分析示意图

9.2 数据库设计的需求分析

□ 数据库设计中的需求分析

➤ 确定需要在数据库保存其信息的客观事物（**things**）及其相互关系（**relationship**）

➤ **Things**

➤ **Attributes of Things**

➤ **Relationships among Things**

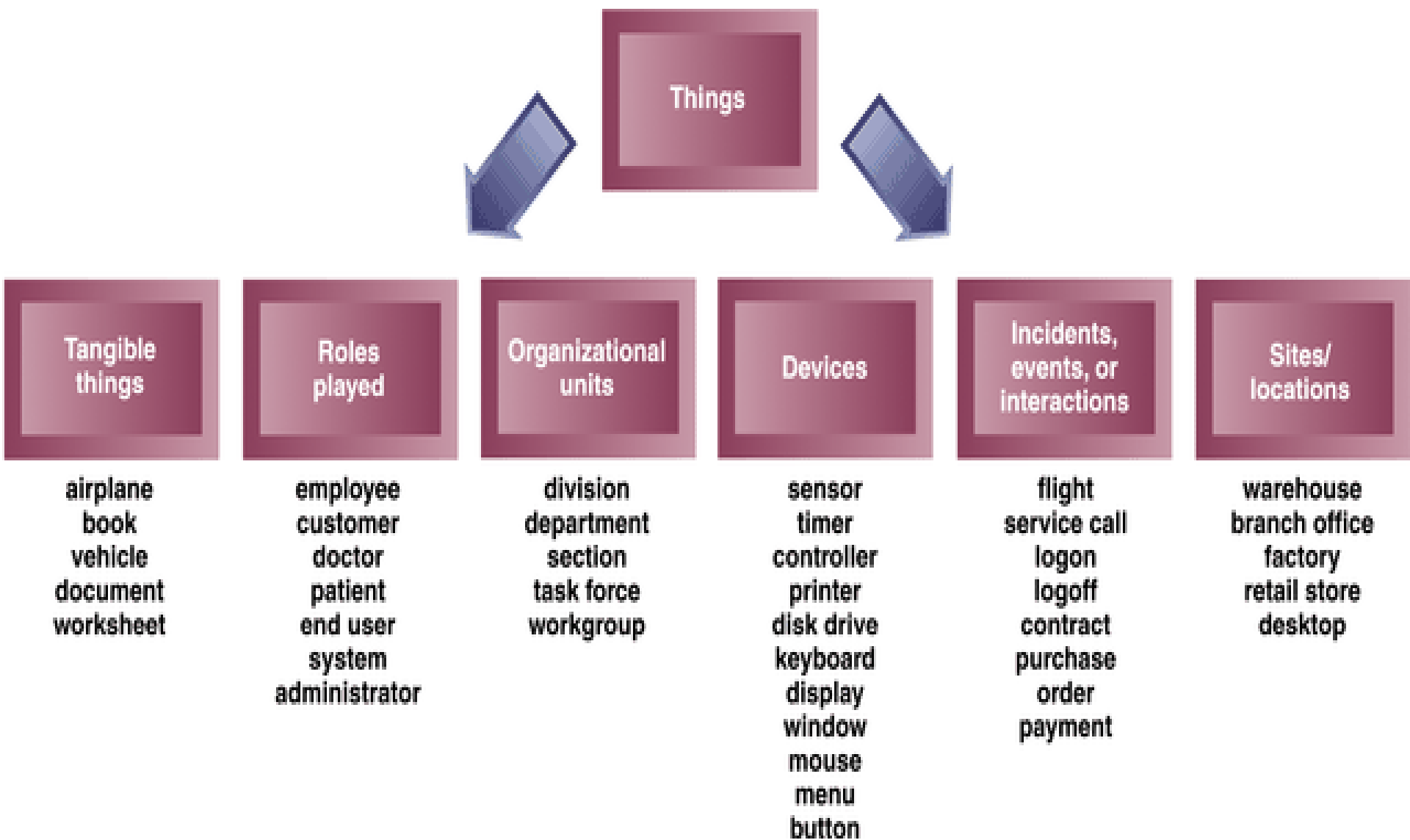


Figure: Types of Things

Relationships among Things

❑ **Relationship:** a naturally occurring association among specific things

- An order is placed by a customer
- an employee works in a department

❑ Relationships apply in two directions

- a) A customer places an order
- b) An order is placed by a customer

Relationships among Things

- Important to know the number of associations of things
 - the **cardinality** (基数) or *multiplicity* of the relationship
 - **one to one, one to many, many to many**
- Also may be important to know the range of possible values of the cardinality
 - **zero to one, zero to many**: optional relationship (可选关系)
 - **one to one, one to many**: mandatory relationships (强制关系)

More on Relationships

□ Binary relationship (二元联系)

- Relationship between two different types of things

□ Unary (recursive) relationship (一元联系)

- Relationship between two things of the same type

□ Ternary relationship (三元联系)

- A relationship between three different types of things

□ n'ary relationship (多元联系)

- A relationship between n (any number) different types of things

Attributes of Things

- ❑ **Attribute:** one piece of specific information about a thing
 - a customer has a name, phone number, credit limit etc
- ❑ **Identifier (key):** an attribute that uniquely identifies a thing
 - a person's social insurance number, or an invoice, or transaction number
- ❑ **Compound attribute:** an attribute that contains a collection of related attributes
 - a full name is made up of first and last name

Data Entities & Objects

□ Data Entities

- The things the system needs to store information about in the traditional approach
- Computer processes interact with these data entities

□ Objects – the other way to look at things

- Similar to data entities in traditional approach
- BUT the objects do the work in the system, they do not just store information (They have behavior)

9.2 数据库设计的需求分析

□ 需求分析的步骤

1. 需求调查
2. 需求分析
3. 生成数据需求分析说明书

9.2 数据库设计的需求分析

1. 需求调查

➤ 调查目的

- 了解用户单位的组织机构和业务活动情况
- 了解用户的业务活动对数据的需求
- 原始资料的收集

➤ 调查方式

- 发放调查表
- 座谈
- 搜集原始资料
- 实际观察

9.2 数据库设计的需求分析

2. 需求分析

1) 数据边界的确定

2) 绘制数据流图（DFD）

- 分析用户活动，在此基础上着重分析其数据流，进而绘制出数据流图

3) 创建数据字典

9.2 数据库设计的需求分析

➤ 数据字典的内容包括:

- 数据项
- 数据结构
- 数据流
- 数据存储
- 数据处理

9.2 数据库设计的需求分析

❑ 数据项

类型，长度，取值范围，语义约束，与其它项的关联

❑ 数据结构

数据结构的组成与约束

❑ 数据流

来源，去向，组成，平均/高峰流量

❑ 数据存储

输入/输出数据流及其组成内容

数据量，存取频度，存取方式

❑ 数据处理

输入数据，输出数据，处理的简短说明

9.2 数据库设计的需求分析

3. 生成数据需求分析说明书

- 在调查与分析的基础上依据一定的规范要求编写数据需求分析说明书。
- 数据需求分析说明书是整个应用系统需求分析文档的重要组成部分。

9.1 数据库设计概述

9.2 数据库设计的需求分析

9.3 数据库的概念设计

9.4 数据库的逻辑设计

9.5 数据库的物理设计

9.3 数据库的概念设计

9.3.1 数据库概念设计概述

- 目的：建立一个抽象的概念数据模型
- 工具
 - E-R模型：实体，属性，联系
 - EE-R模型
 - 属性、实体
 - 联系、嵌套（实体集属性）、继承
- 方法
 - 集中式模式设计法
 - 视图集成设计法

9.3 数据库的概念设计

9.3.2 数据库概念设计的过程

➤ 用户分解

- 首先将所有用户划分为功能相对独立的若干个用户组，然后针对每个用户组进行视图设计。

➤ 视图设计

- 针对每个用户组设计其数据视图，以反映该组用户对于数据的需求。

➤ 视图集成

- 将设计好的若干个局部数据视图集成为一个完整的全局数据视图。

9.3.2 数据库概念设计的过程 – 视图设计

□ 数据对象的设计次序

➤ 自顶向下

- 先从抽象级别高且普遍性强的对象开始设计，然后再逐步细化、具体化与特殊化

➤ 由底向上

- 先从具体的对象开始设计，再逐步抽象、普遍化与一般化

➤ 由内向外

- 先从最基本与最明显的对象着手设计，再逐步扩充至非基本、不明显的其他对象

□ 可以混合使用上述的三种对象设计次序。

9.3.2 数据库概念设计的过程 – 视图设计

□ 在视图设计中有两个方面的设计任务

(1) 实体与属性设计

(2) 联系与继承设计

9.3.2 数据库概念设计的过程 – 视图设计

(1) 实体与属性设计

实体与属性的设计又分为两个方面：

- 1) 区分实体与属性
- 2) 实体与属性的详细描述

9.3.2 数据库概念设计的过程 – 视图设计

1) 区分实体与属性

➤ 区分原则

- 描述信息原则

- 对实体需要有进一步的性质描述

- 依赖性原则

- 属性单向、包含性依赖于某个实体

- 一致性原则

- 组成一个实体的属性之间存在着某种内在的关联性与一致性

9.3.2 数据库概念设计的过程 – 视图设计

2) 实体与属性的详细描述

➤ 实体与属性的命名

- 简洁，便于记忆，遵守缩写规范，避免冲突

➤ 确定实体标识

- 列出所有候选关键字
- 确定实体集的主关键字

➤ 主关键字确定原则：非空值

9.3.2 数据库概念设计的过程 – 视图设计

(2) 联系与继承设计

联系与继承设计又分为：

- 1) 区分联系与继承
- 2) 联系的详细描述
- 3) 继承的详细描述

9.3.2 数据库概念设计的过程 – 视图设计

1) 区分联系与继承

➤ 两者在语义上有明确的区分

- **继承**：实体集之间的分类与包含关系
- **联系**：实体间的一种更为广泛的语义联系

➤ 同时，两者之间又具有一定的联系

- ‘继承’是一种特殊的‘联系’。因此，也可以用普通的‘联系’来表示‘继承’
- 但是，在‘子实体集’中需要重新定义所有从‘超实体集’中继承下来的信息

9.3.2 数据库概念设计的过程 – 视图设计

➤ 例如：

■ ‘继承’ 关系：

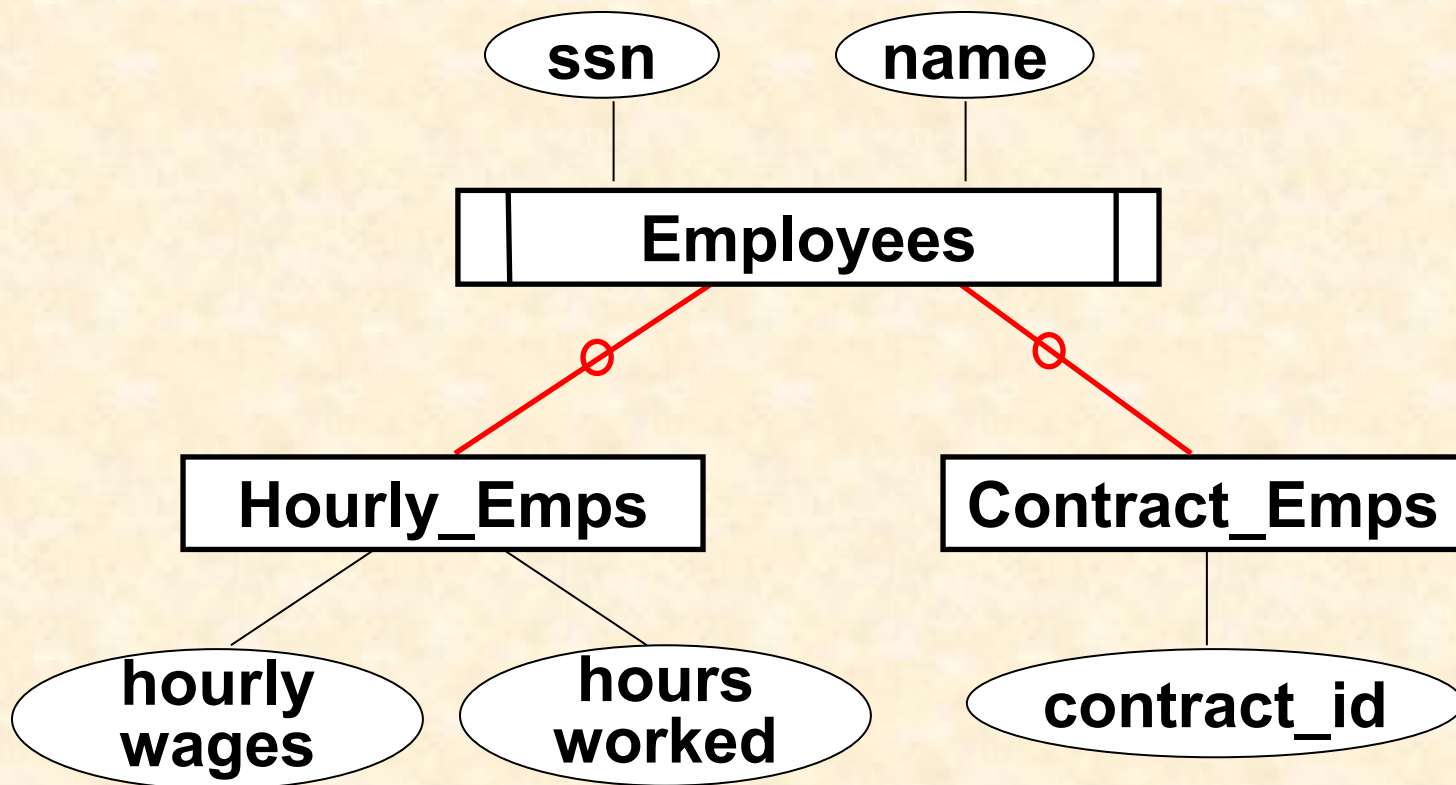
- ‘学生’ — ‘研究生’
- ‘教师’ — ‘教授’ — ‘终身教授’

■ ‘联系’：

- ‘教师’ 〈指导〉 ‘学生’
- ‘教师’ 〈助手〉 ‘教师’

9.3.2 数据库概念设计的过程 - 视图设计

□ 继承的例子



9.3.2 数据库概念设计的过程 – 视图设计

2) 联系的详细描述

➤ 联系分类

① 存在性联系

② 功能性联系

③ 事件联系

➤ 与每个联系相关的实体集的个数

➤ 联系的函数对应关系

➤ 相同实体集间的多种联系

9.3.2 数据库概念设计的过程 – 视图设计

3) 继承的详细描述

- 子实体集**继承**超实体集中的属性定义。
- 部分继承 **vs.** 全继承
 - **全继承**: 子实体集继承超实体集的**全部**属性
 - **部分继承**: 子实体集继承超实体集的**部分**属性
- 单继承 **vs.** 多重继承
 - **单继承**: 一个子实体集**只能有一个**直接的超实体集
 - **多重继承**: 一个子实体集**可以有多个**直接的超实体集
- 在继承过程中的冲突解决办法

9.3.2 数据库概念设计的过程 - 视图设计

■ 例9-1：学生视图的设计例子

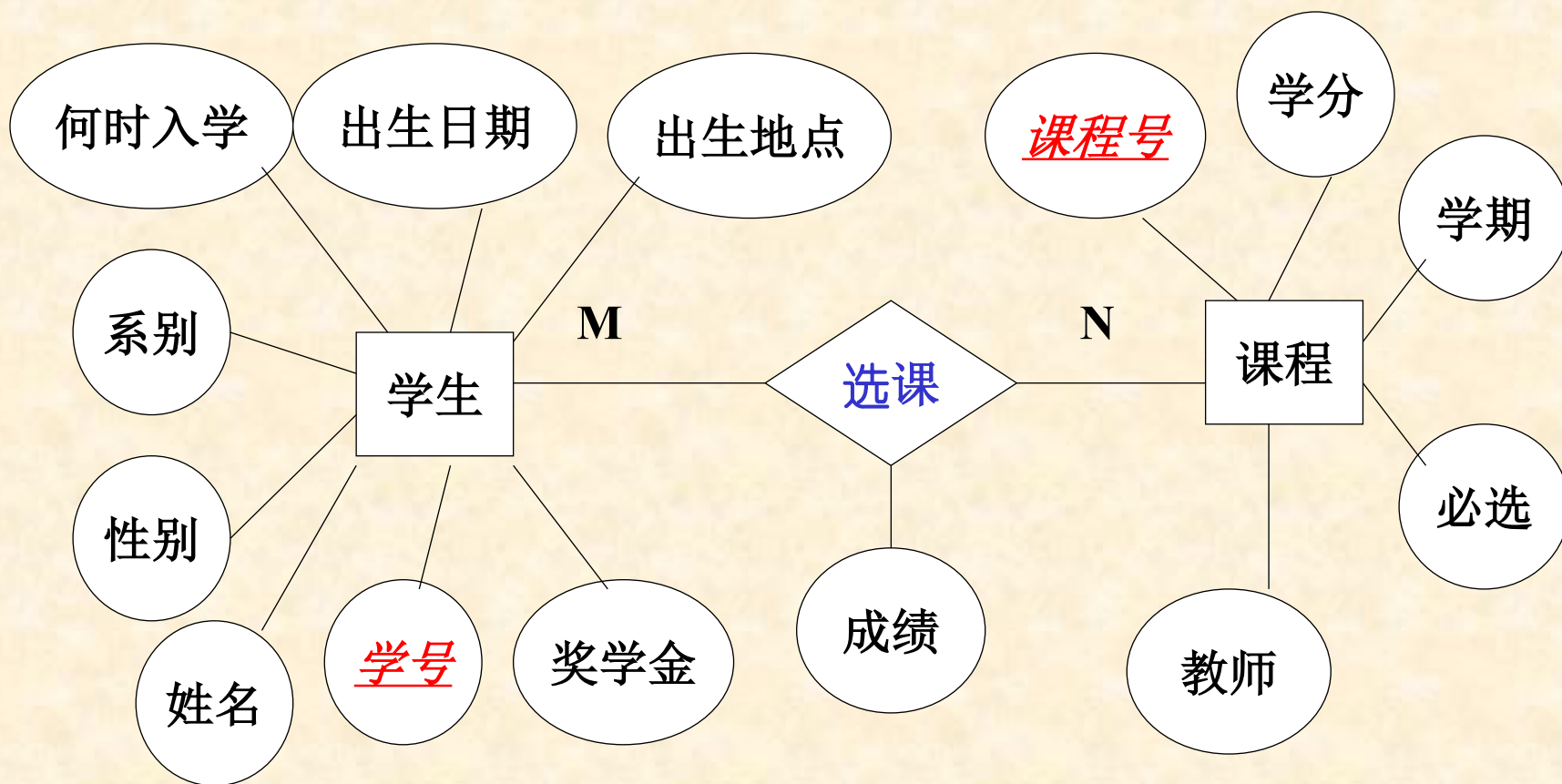


图9.5 教务处关于学生的视图

9.3.2 数据库概念设计的过程 - 视图设计

■ 例9-2：研究生视图的设计例子

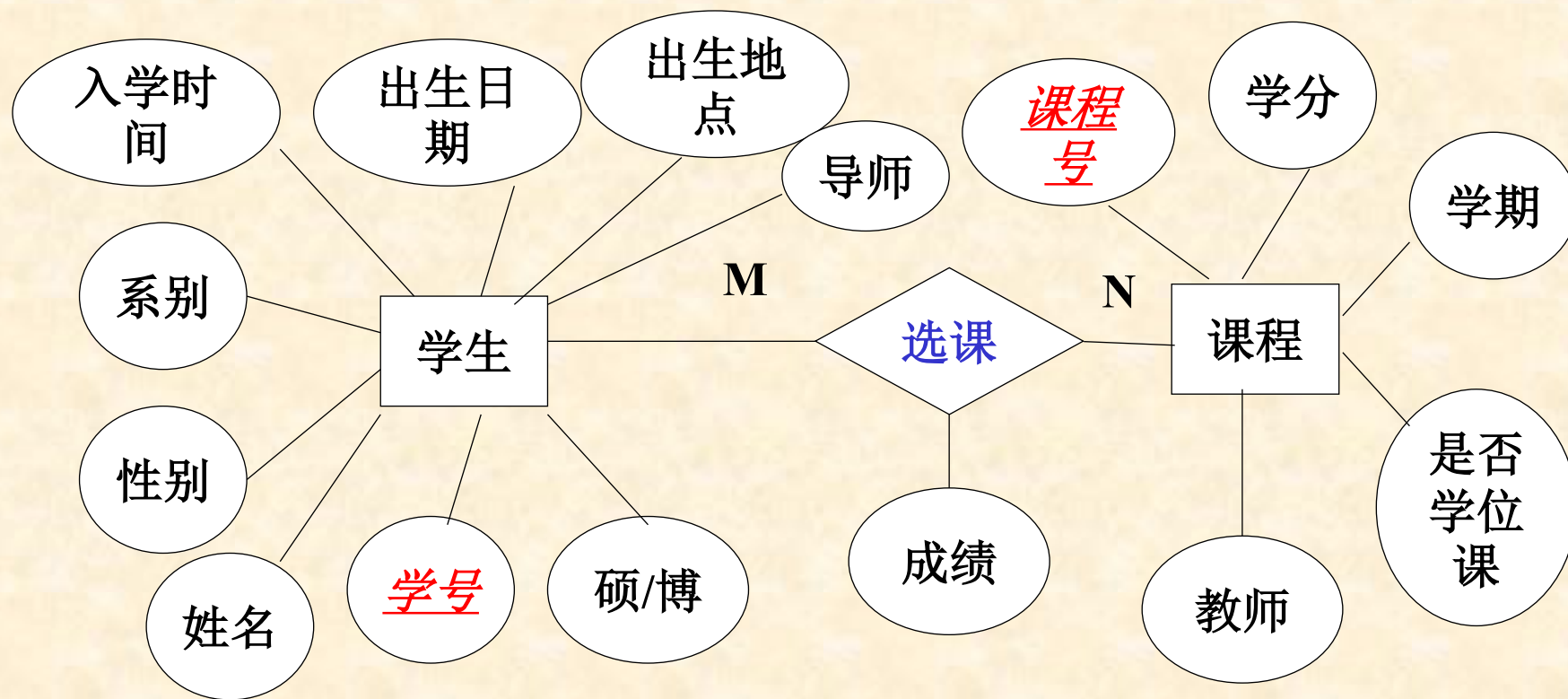


图9.6 研究生院关于研究生的视图

9.3.2 数据库概念设计的过程 – 视图设计

■ 例9-3：学校教职工视图的设计

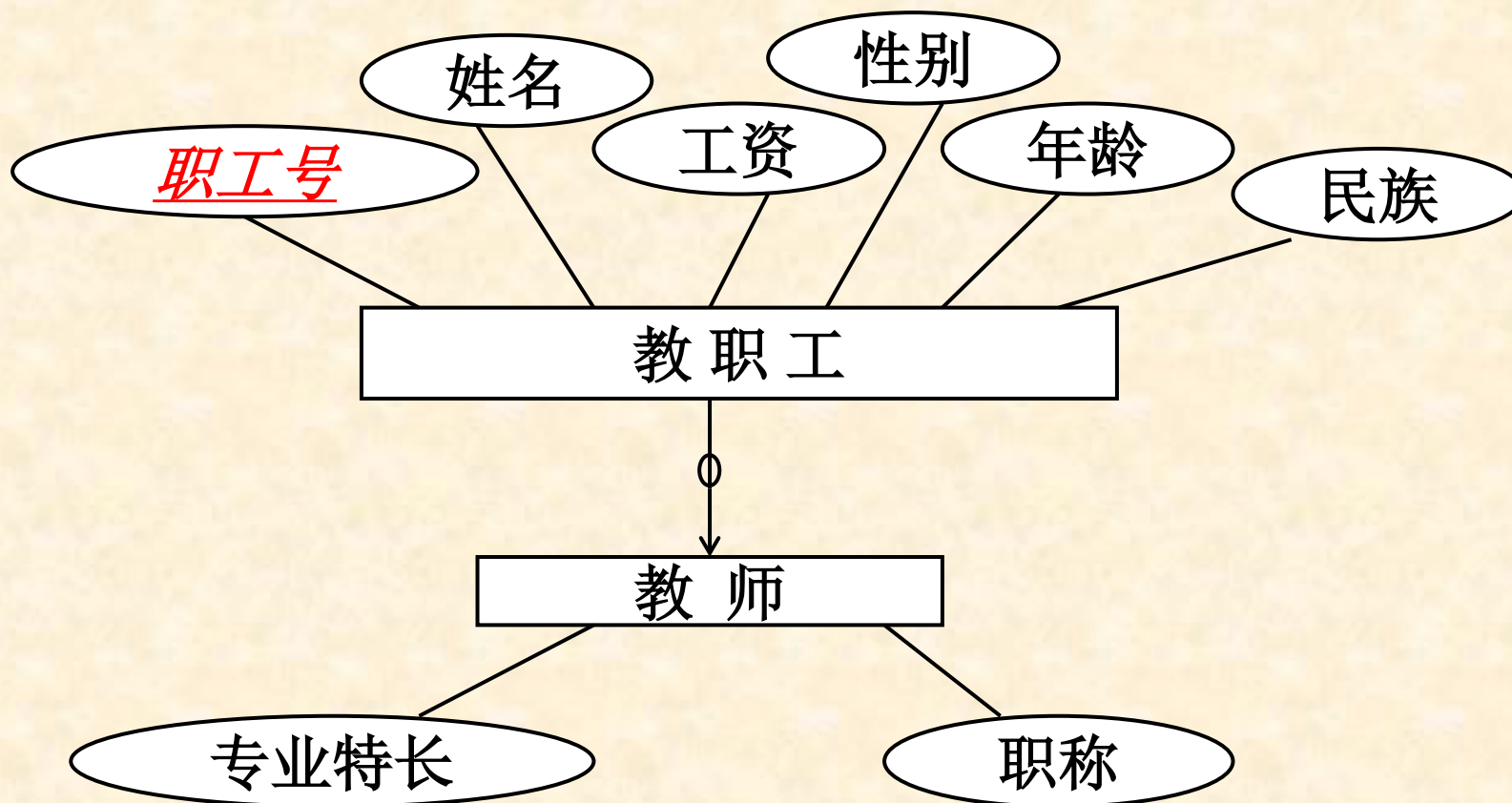


图9.7 学校教职工视图

□为什么在‘教师’实体集上没有定义标识属性？

9.3.2 数据库概念设计的过程 - 视图设计

■ 例9-4：某计算机外部设备生产厂家的视图设计

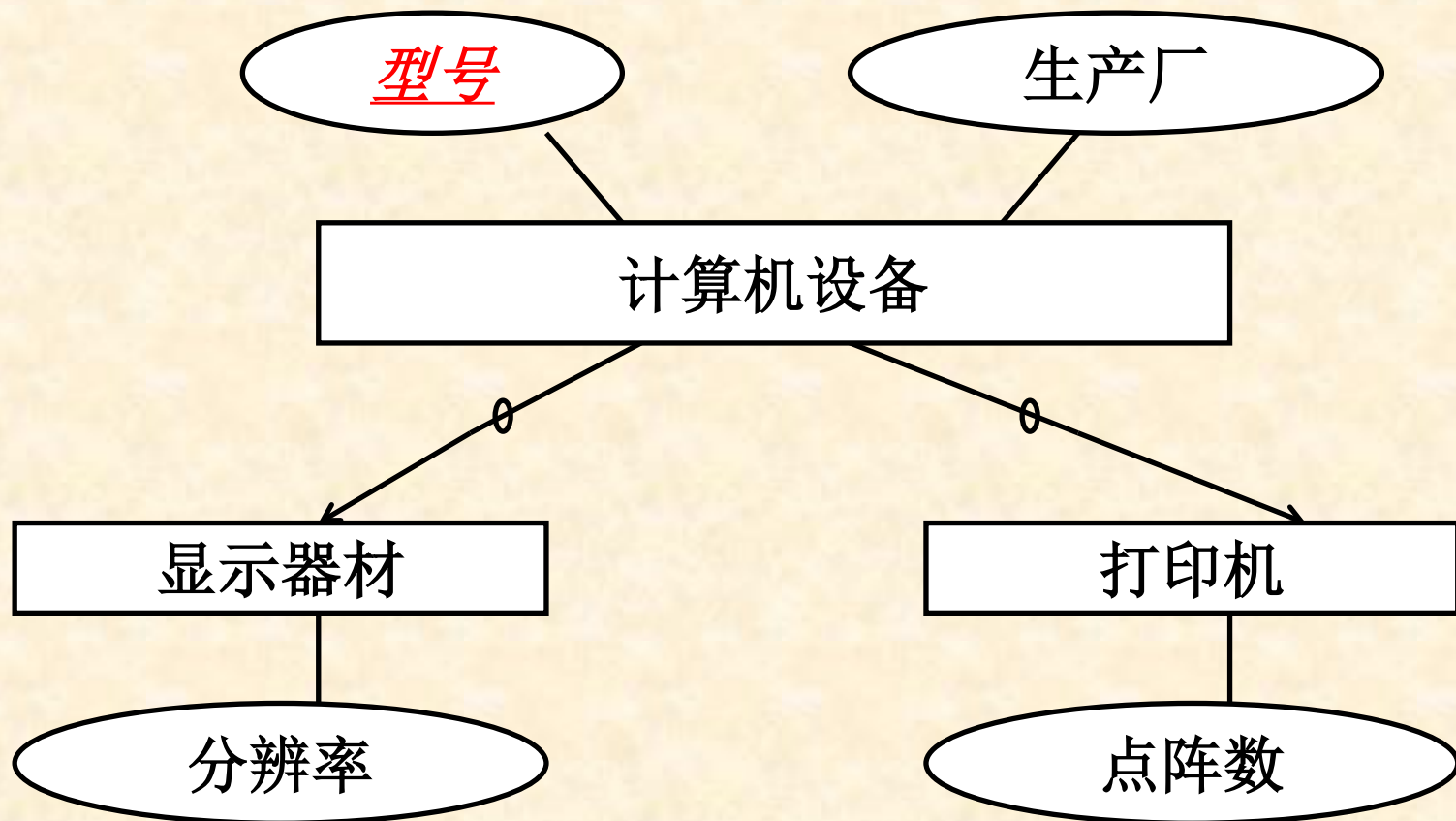


图9.8 计算机生产厂家视图

9.3 数据库的概念设计

□ 数据库设计的综合案例

- 案例1: 图书借阅管理 (\leq)
- 案例2: 篮球联赛管理 (\leq)
- 案例3: 邮件信息管理 (\leq)
- 案例4: 旅游信息管理 (\leq)
- 案例5: 学生选课管理 (\leq)

9.3.2 数据库概念设计的过程

□视图集成

- 所有局部视图的统一与合并，最终形成一个完整的全局模式。

- 在视图集成过程需要完成的工作
 - ① 确定所采用的原理与策略
 - ② 规划视图的集成步骤
 - ③ 发现并解决可能存在的冲突现象

视图集成的原理与策略

□ **等同**：指两个或多个数据对象具有相同的语义。

- 属性等同，实体等同，语义相关等同（属性-实体）
- 包括：同义同名，同义异名

□ **聚合**：数据对象之间的一种组成关系。

- 由属性聚合成实体
- 由属性和实体聚合成新的实体

□ **抽取**

- 将不同实体中的相同属性提取成一个新的实体，并构造具有继承关系的结构。

9.3.2 数据库概念设计的过程 – 视图集成

□ 常见的几种冲突现象

➤ 命名冲突

- 同义异名，同名异义

➤ 概念冲突

- 同一概念在一处为实体，而在另一处则为属性或联系

➤ 域冲突

- 例如：类型冲突，度量单位的冲突

➤ 约束冲突

➤ 冲突的解决办法：视图修改

9.3.2 数据库概念设计的过程 - 视图集成

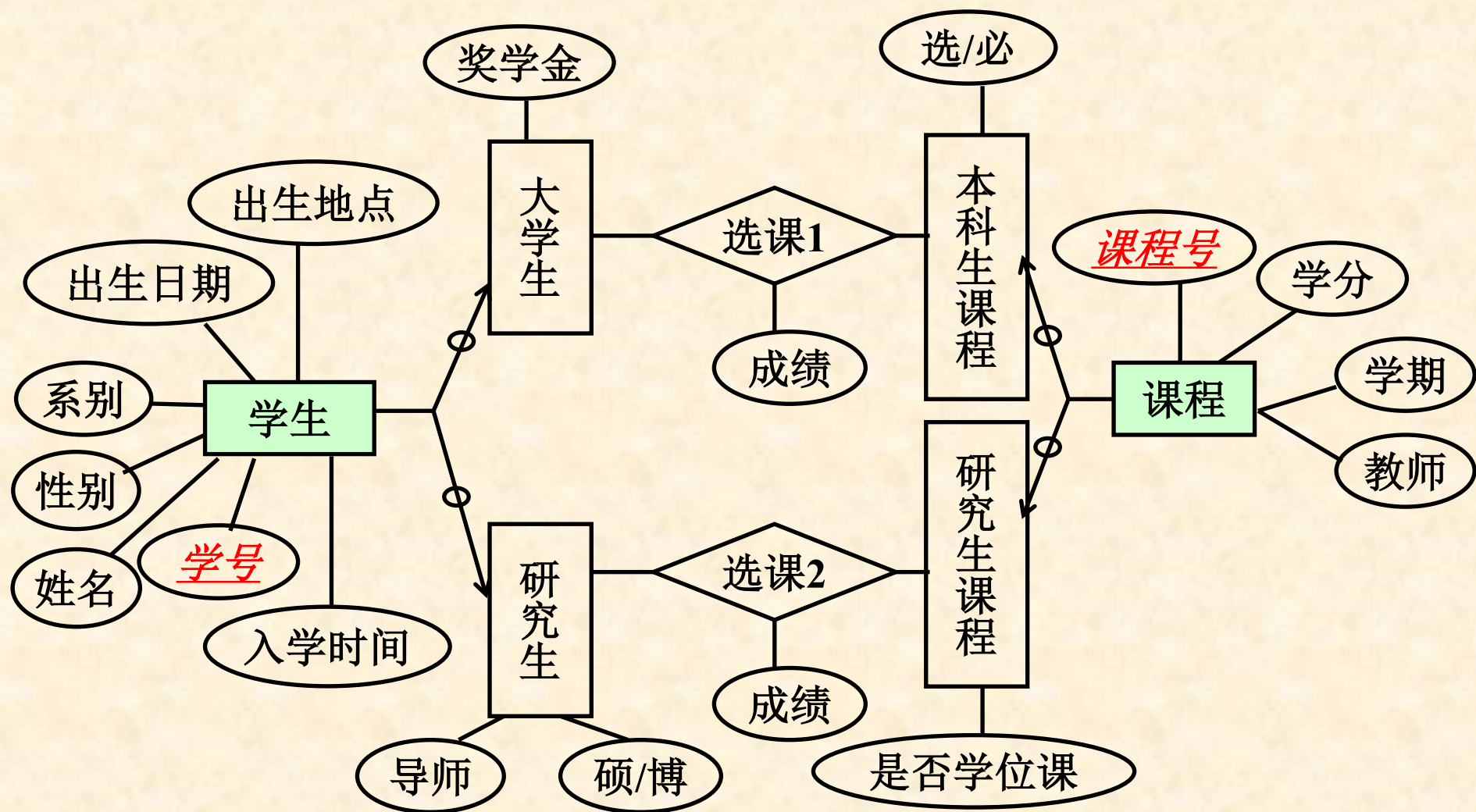


图9.10 对图9-5和图9-6进行视图集成后的全局视图

9.3.2 数据库概念设计的过程 – 视图集成

□ 在图9-5和图9-6的集成过程中所使用到的集成技术

➤ ‘命名冲突’的解决办法

- ‘学生’ 改为 ‘研究生’ 和 ‘大学生’
- ‘课程’ 改为 ‘本科生课程’ 和 ‘研究生课程’
- ‘选课’ 改为 ‘选课1’ 和 ‘选课2’
- ‘何时入学’ 和 ‘入学时间’ 统一为 ‘入学时间’

➤ ‘抽取’形成新的实体集

- 由 ‘研究生’ 和 ‘大学生’ 抽取出 ‘学生’
- 由 ‘本科生课程’ 和 ‘研究生课程’ 抽取出 ‘课程’

9.3.2 数据库概念设计的过程 – 视图集成

□例：图9-10与图9-5进行集成存在‘概念冲突’：

教师

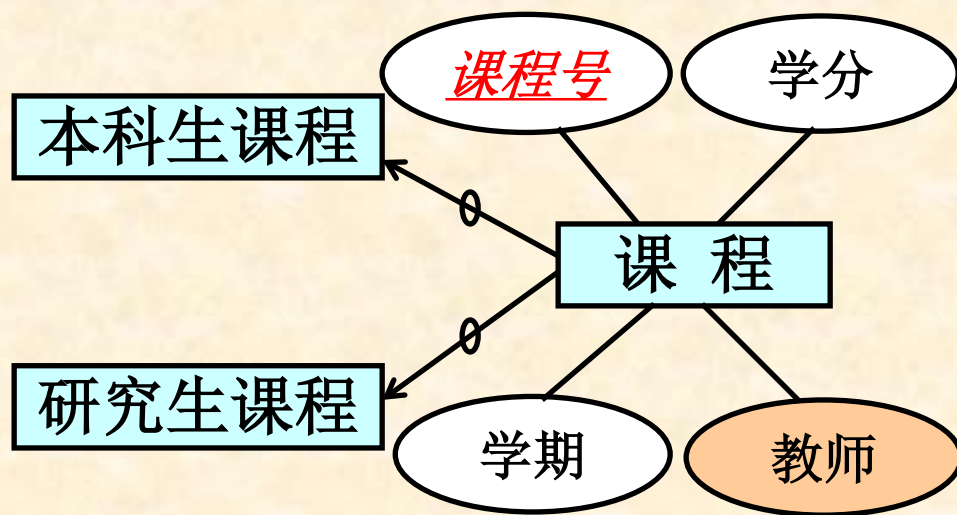


图9-10 （局部）

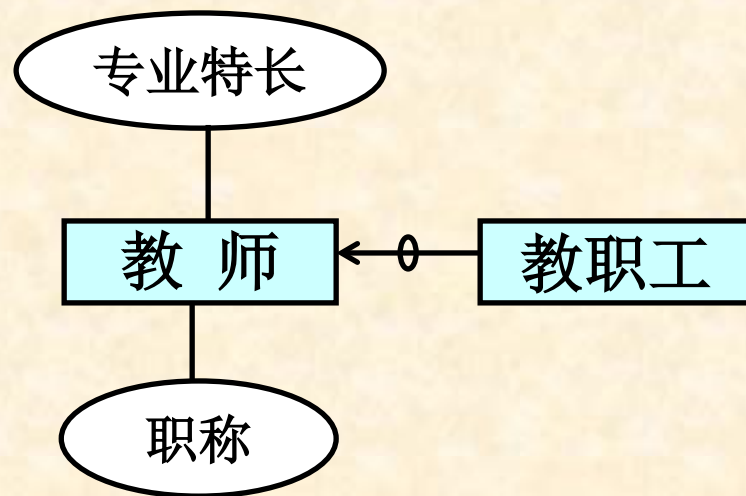


图9-5 （局部）

9.3.2 数据库概念设计的过程 – 视图集成

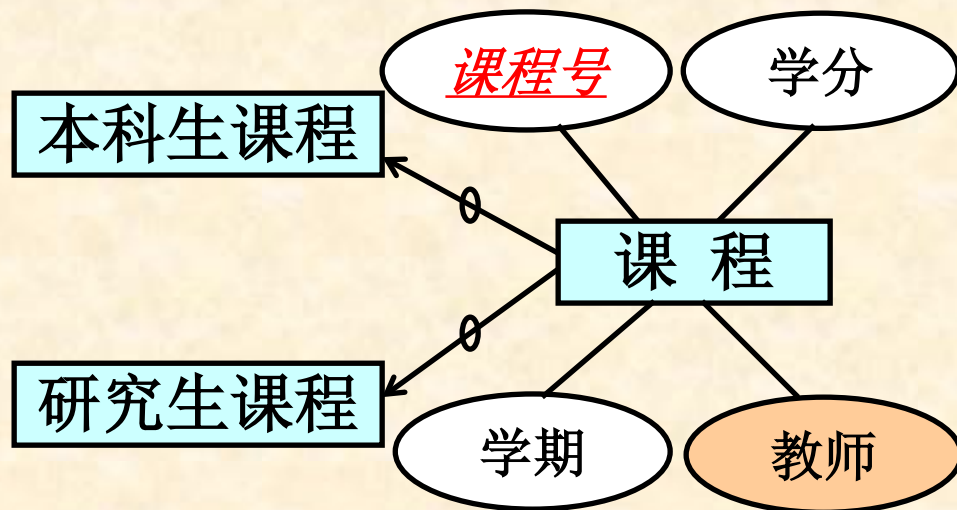


图9-10（局部）

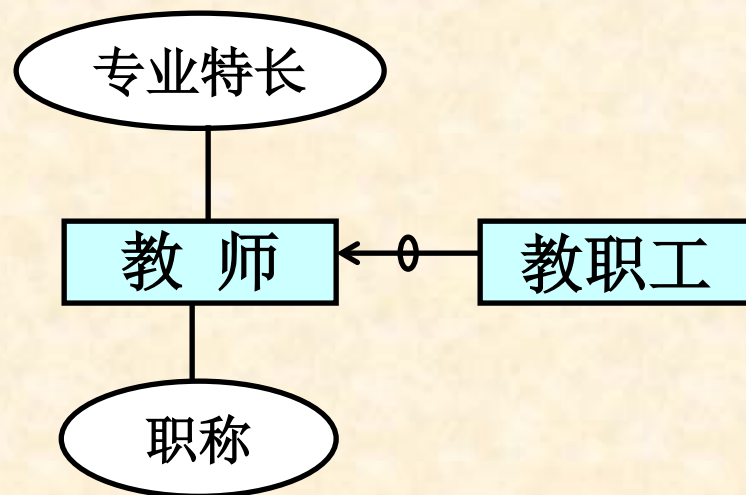


图9-5（局部）

□ 解决办法

- 在图9-10的‘课程’实体集中删去‘教师’属性
- 在图9-10的‘课程’实体集与图9-5的‘教师’实体集之间建立一个‘讲课’联系

9.1 数据库设计概述

9.2 数据库设计的需求分析

9.3 数据库的概念设计

9.4 数据库的逻辑设计

9.5 数据库的物理设计

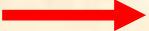
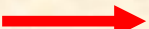
9.4 数据库的逻辑设计

□ 基本任务

- 将前一阶段设计得到的概念数据模型(**EE-R模型**)转换成用户所选择的数据库管理系统(**DBMS**)支持的逻辑数据模型。
- 由于在概念设计阶段选择的是**EE-R模型**，而目前最流行的是关系数据库系统，因此，数据库的逻辑设计的基本任务就是：
 - 将**EE-R模型**转换成相等价的关系数据库模式

9.4 数据库的逻辑设计

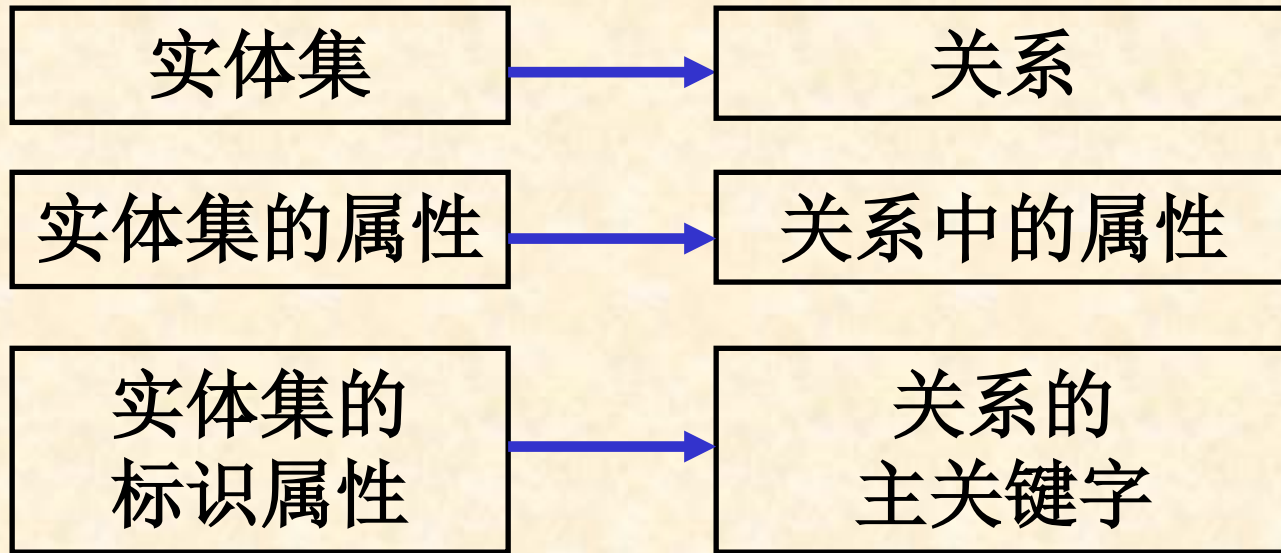
□ 基本方法

- 每个实体集  一个关系模式
- 每个联系  一个关系模式

9.4 数据库的逻辑设计

□ 实体集的转换方法

- 每个实体集被转换成一个关系（模式）



- 关系及其属性的命名采用原实体集及其属性的名称

9.4 数据库的逻辑设计

□ 联系的转换方法

- 在一般情况下，每个联系也被转换成一个关系模式，联系名被用作转换得到的关系模式的关系名，该关系模式中的属性由两部分组成：
 - 联系自身所具有的属性
 - 与该联系相关的实体集的标识属性
- 来自于相关实体集的标识属性也是相关实体集转换得到的关系模式的主关键字，因此它们也是联系转换得到的关系模式中的外关键字

9.4 数据库的逻辑设计

□ 在从概念数据模型（EE-R）到逻辑数据模型（关系模型）的转换过程中，还可能存在一些需要处理的问题。

- 命名与属性域的处理
- 非原子属性处理
- 联系的特殊转换
- 继承的转换
- 规范化
- RDBMS性能调整
- 约束条件设置

9.4.1 逻辑设计的基本方法

1. 命名与属性域的处理

➤ 关系及属性的命名

- 尽量采用在**EE-R**模型中原有的名称
- 可以重新命名，但要避免命名的冲突现象
 - 在同一个数据库模式中，关系名具有唯一性
 - 在同一个关系模式中，属性名具有唯一性

➤ 属性域的定义

- 根据**DBMS**的选型进行必要的数据类型转换

9.4.1 逻辑设计的基本方法

2. 非原子属性的处理

- 集合属性
- 元组属性

❖ 集合属性的例子

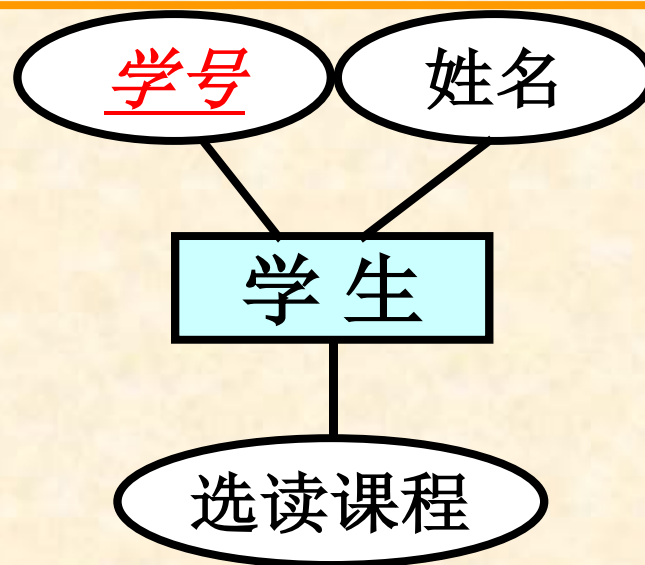


表 9-1 之 EE-R 图

学号	姓名	选读课程
S1307	王承志	Database Operating System Computer Network

表 9-1 学生实体

9.4.1 逻辑设计的基本方法

□集合属性的处理

- 关系模式不变，但原有关系的一个元组将被纵向展开成多个元组
- 在上述转换过程中，虽然实体集(或关系模式)中的属性没有增加，但转换得到的关系模式的主关键字由原实体集的标识属性和该集合属性联合构成（红色的属性名集合）。

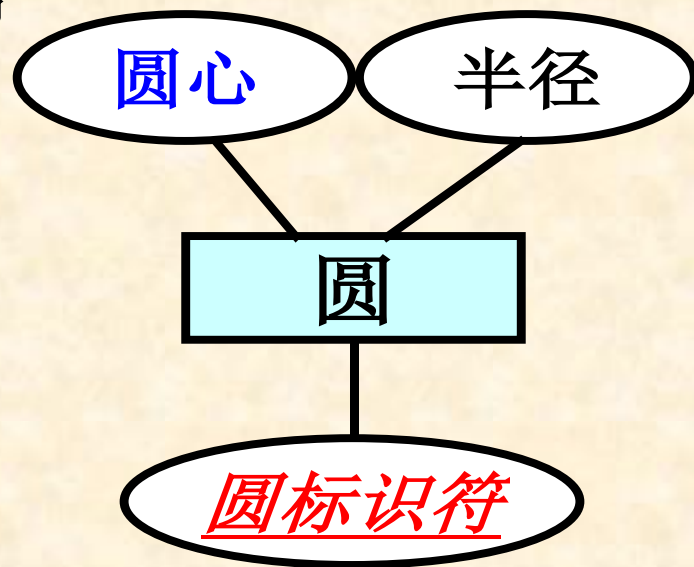
学号	姓名	选读课程
S1307	王承志	Database
S1307	王承志	Operating System
S1307	王承志	Computer Network

表 9-1 之学生关系

9.4.1 逻辑设计的基本方法

□ **元组属性的处理**：将一个元组属性横向展开成多个属性

➤ **例9.6**：实体集‘圆’有三个属性：圆标识符，圆心和半径，而圆心又由其X坐标轴和Y坐标轴的值组成(左图)。



例 9.6 之 EE-R 图

➤ 转换得到的关系模式如下图

圆标识符	X轴	Y轴	半径
------	----	----	----

例9.6 之关系模式 – 圆

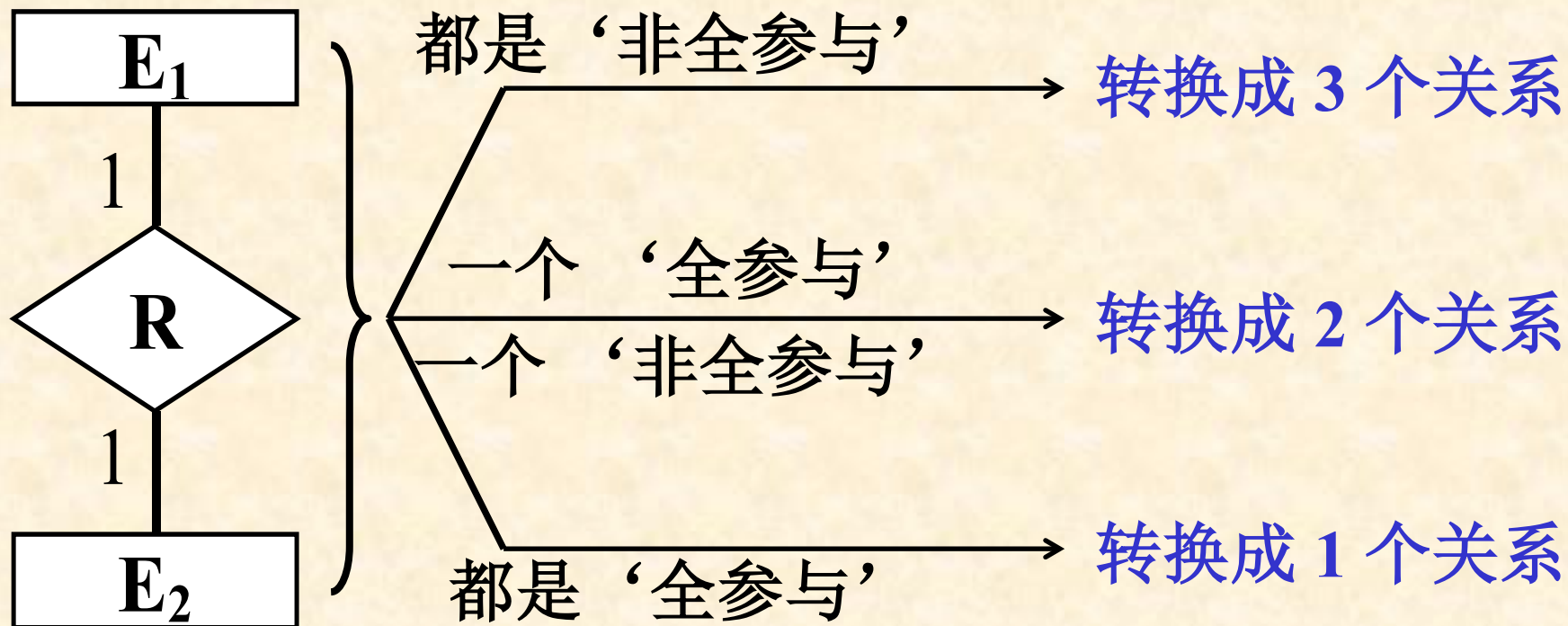
9.4.1 逻辑设计的基本方法

4. 联系的转换

- 在一般情况下，一个联系可以被转换成一个关系
- 但是在有些情况下，联系也可被归并到相关联的实体所对应的关系模式中去，即将联系与某个（或几个）相关联的实体集共同转换成一个关系模式。
- 下面以两个实体集之间的一个二元联系为例，介绍在不同情况下其逻辑设计的方法。

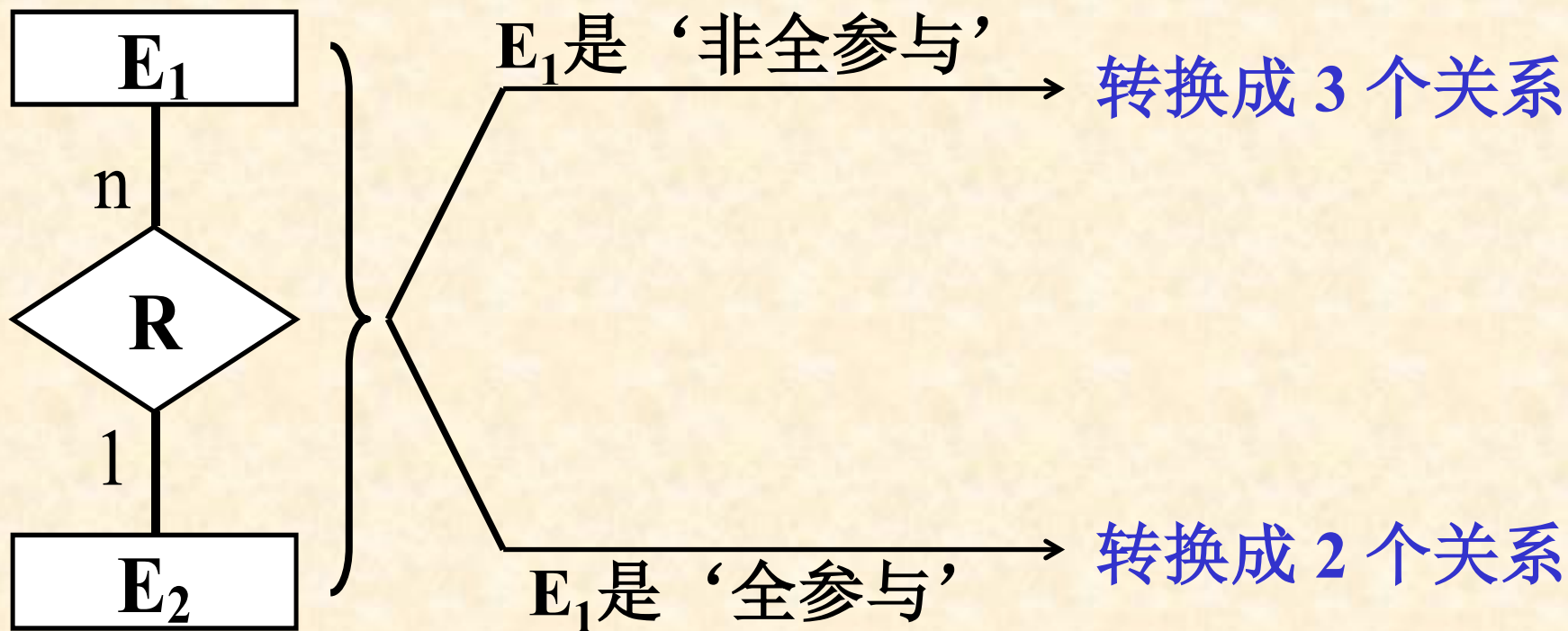
9.4.1 逻辑设计的基本方法

□ 两个实体集之间的一个二元联系，可以根据该联系的函数对应关系，以及每个实体参与该联系的参与方式将其转换成1、2或3个关系模式



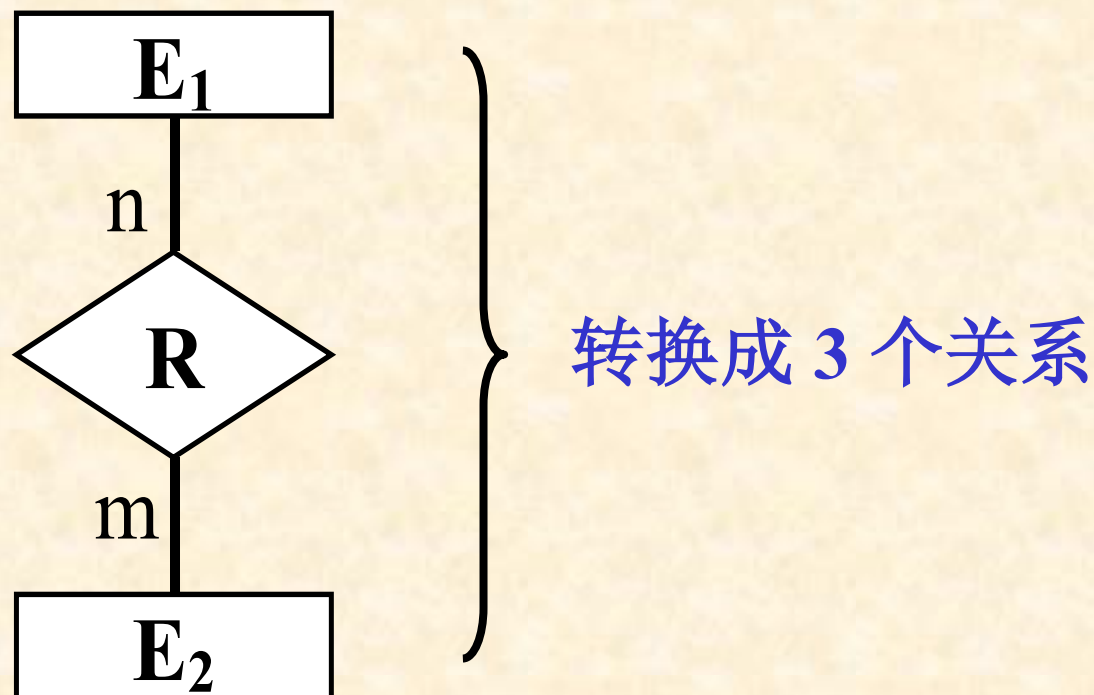
‘一对一’ 联系的转换方式

9.4.1 逻辑设计的基本方法



‘多对一’ 联系的转换方式

9.4.1 逻辑设计的基本方法



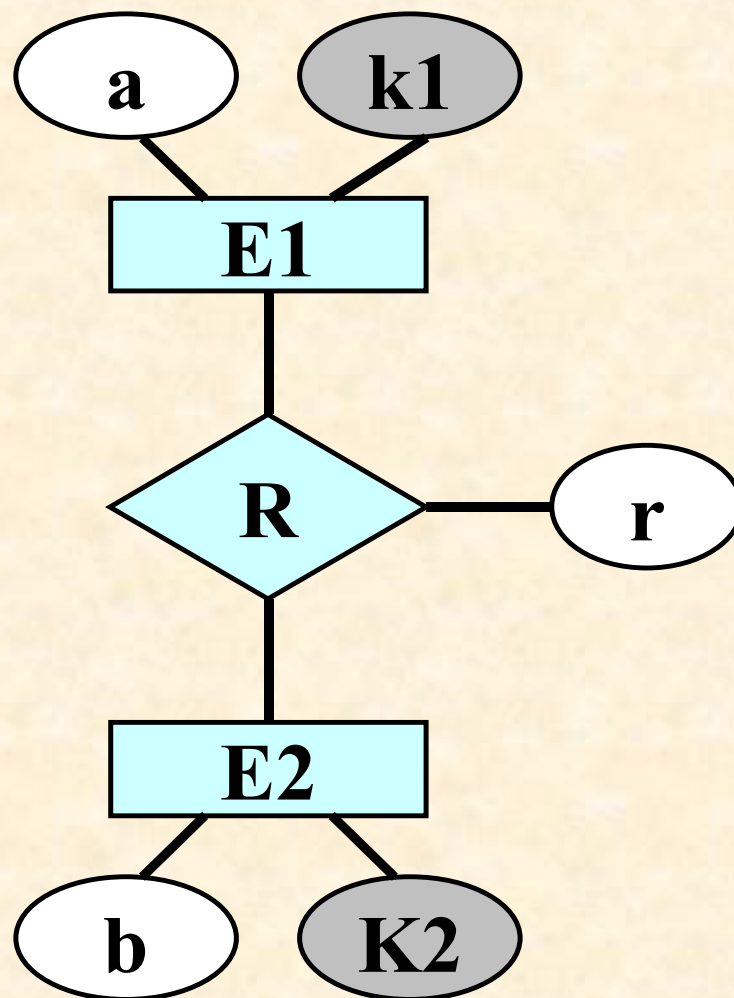
‘多对多’ 联系的转换方式

9.4.1 逻辑设计的基本方法

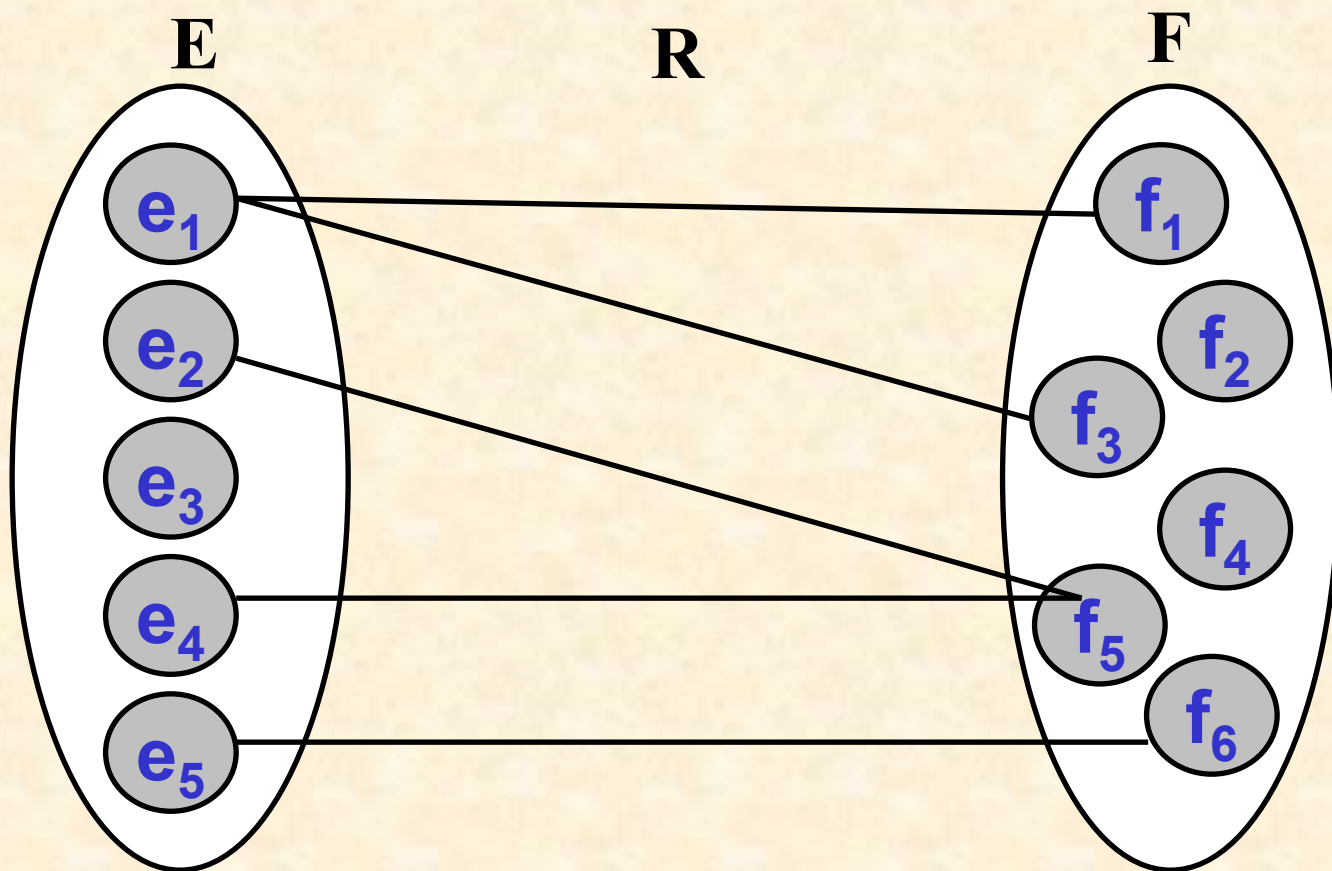
■ 实体集与联系的‘全参与’

□ 假设在实体集 E_1 与实体集 E_2 之间存在一个二元联系 R (如左图)。

□ 如果 E_1 中的每个实体均与 E_2 中的某些实体有关联, 则称实体集 E_1 在该联系中是‘全参与’, 否则是‘非全参与’。

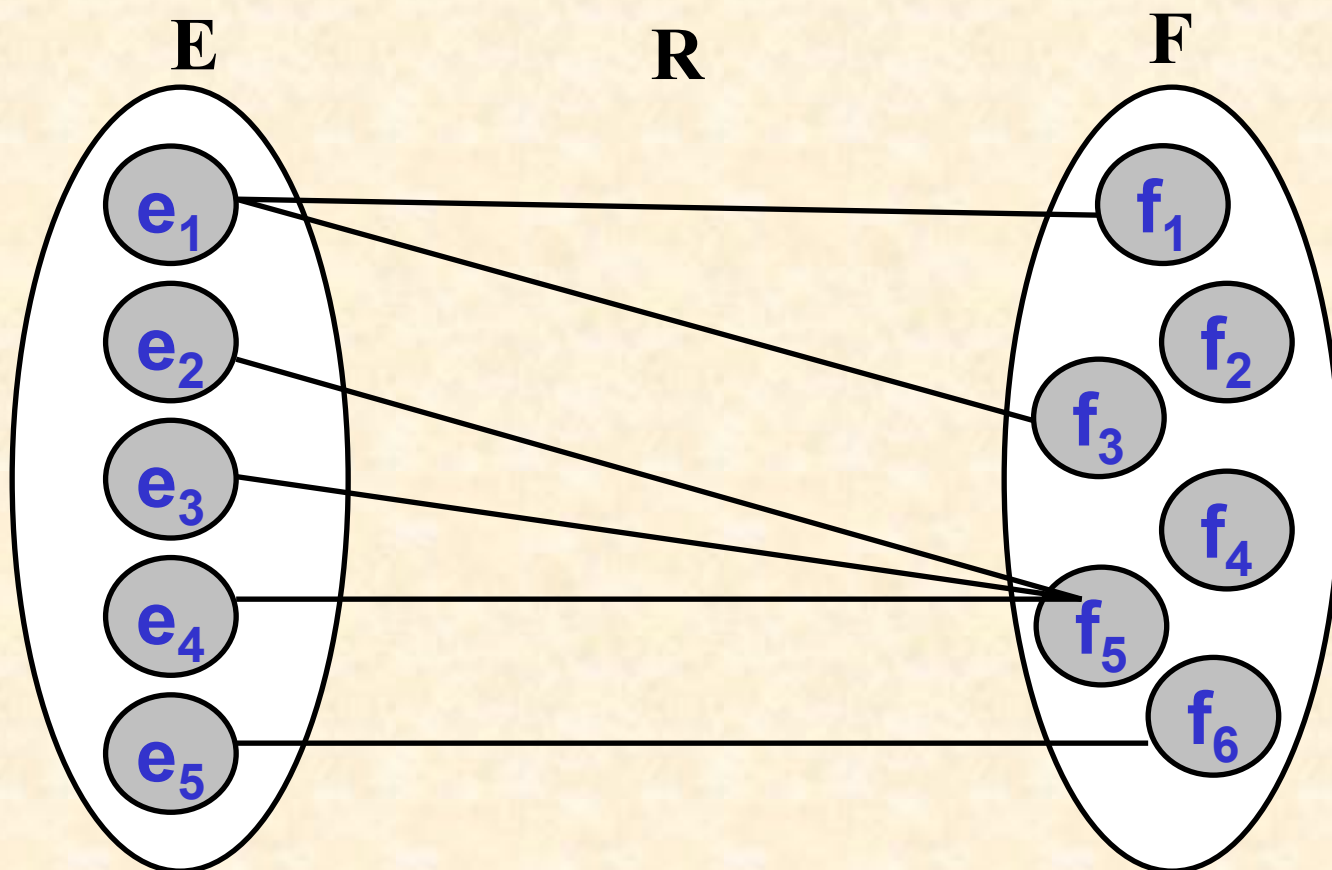


实体集E和实体集F与它们之间的二元联系R的参与关系



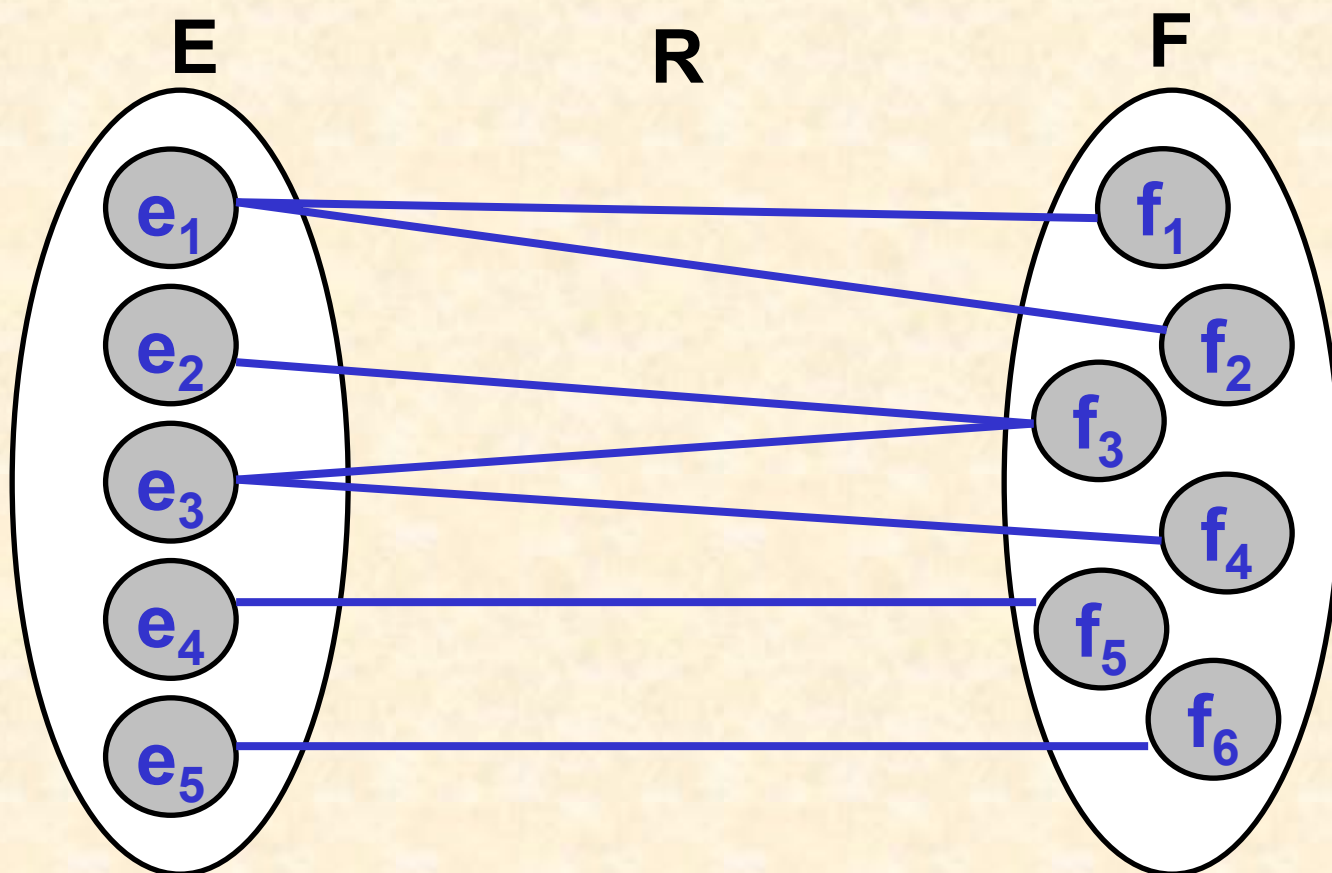
实体集E与实体集F都是 ‘非全参与’

实体集E和实体集F与它们之间的二元联系R的参与关系



实体集E是‘全参与’，而实体集F是‘非全参与’

实体集E和实体集F与它们之间的二元联系R的参与关系

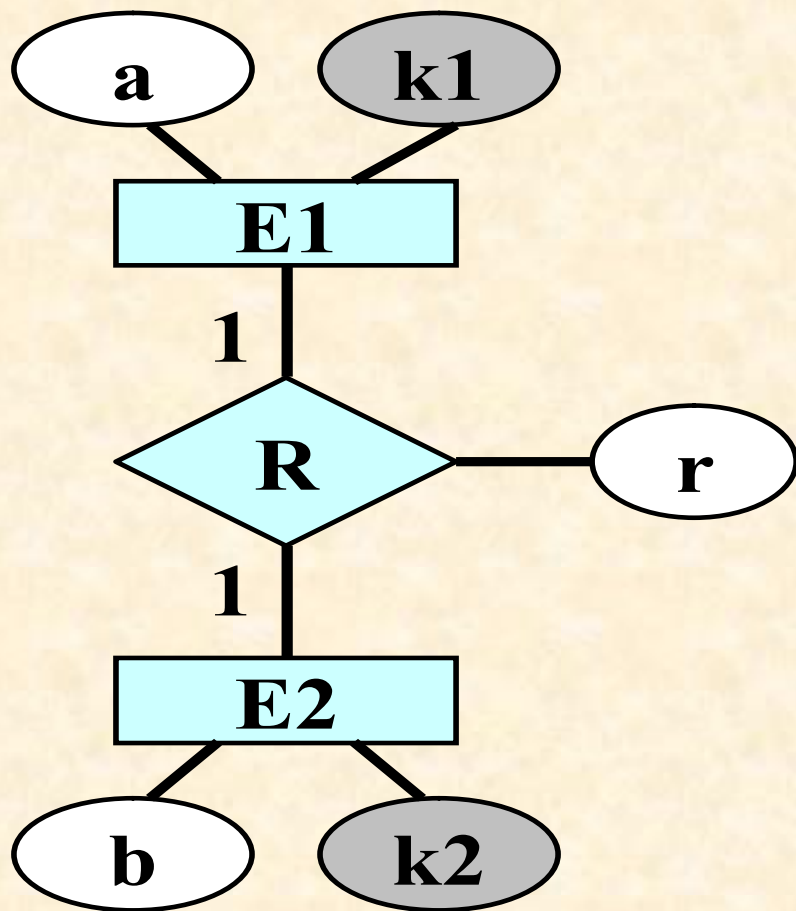


实体集E与实体集F都是 ‘全参与’

“一对一”二元联系的转换

1) 如果 E_1 和 E_2 都是 ‘非全参与’：

将图9.12转换成三个关系模式



E1 (k1, a)

➤ 关键字是k1

E2 (k2, b)

➤ 关键字是k2

R (k1, k2, r)

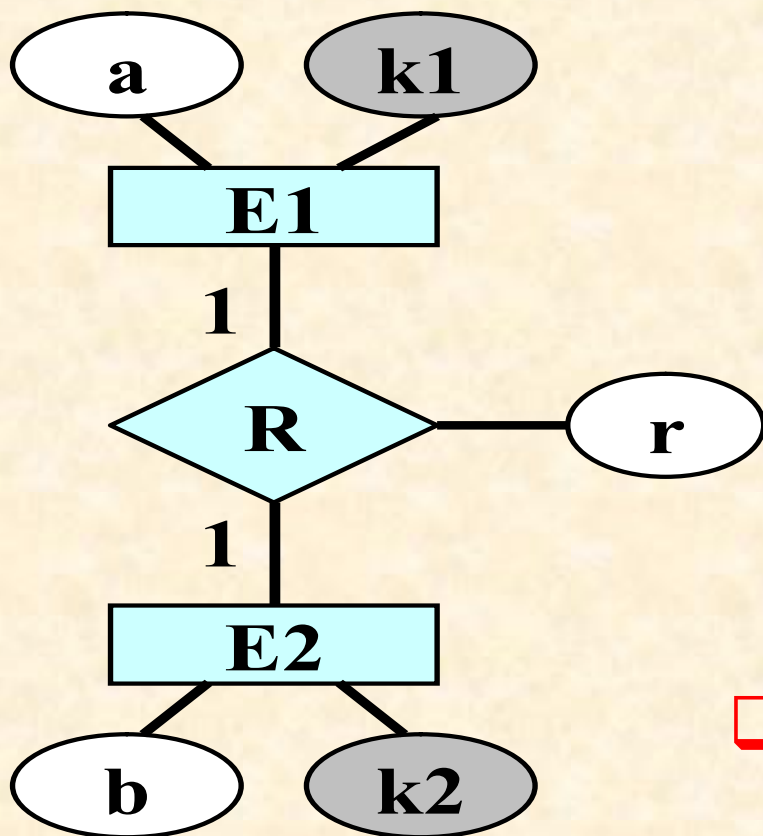
➤ 关键字是 k1 或 k2

图9.12 1:1二元联系

“一对一”二元联系的转换

2) 如果 E_1 是‘全参与’， E_2 是‘非全参与’：

- 可以将联系 R 与实体集 E_1 合并，将图9.12转换成二个关系模式。



$E_1 (k_1, a, k_2, r)$

➤ 关键字是 k_1

➤ k_2 是关系 E_1 的外关键字

$E_2 (k_2, b)$

➤ 关键字是 k_2

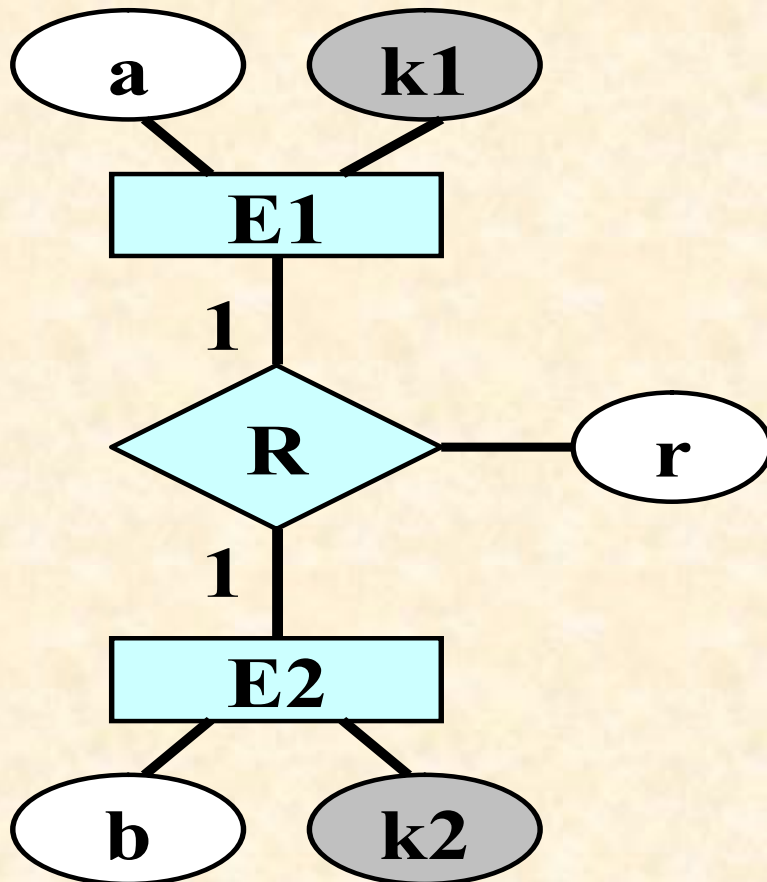
❑ 如果 E_2 是‘全参与’， E_1 是‘非全参与’，则转换方式与此类似。

图9.12 1:1二元联系

“一对一”二元联系的转换

3) 如果 E_1 和 E_2 都是 ‘全参与’：

将三者全部合并，将图9.12转换成一个关系模式。



$E(k_1, a, k_2, b, r)$

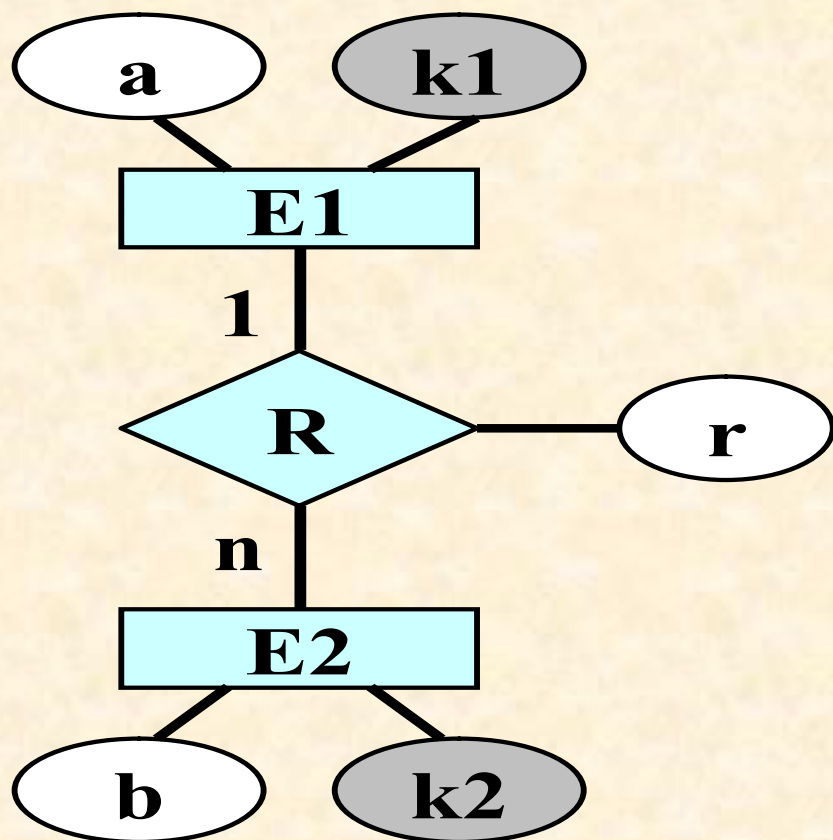
➤ k_1, k_2 是该关系的
两个候选关键字

图9.12 1:1二元联系

“一对多” 二元联系的转换

1) 如果多端 E_2 是 ‘全参与’ :

- 可以将联系 R 与实体集 E_2 合并, 将图9.13 转换成 2 个关系模式



$E_1(k_1, a)$

- 关键字是 k_1

$E_2(k_2, b, k_1, r)$

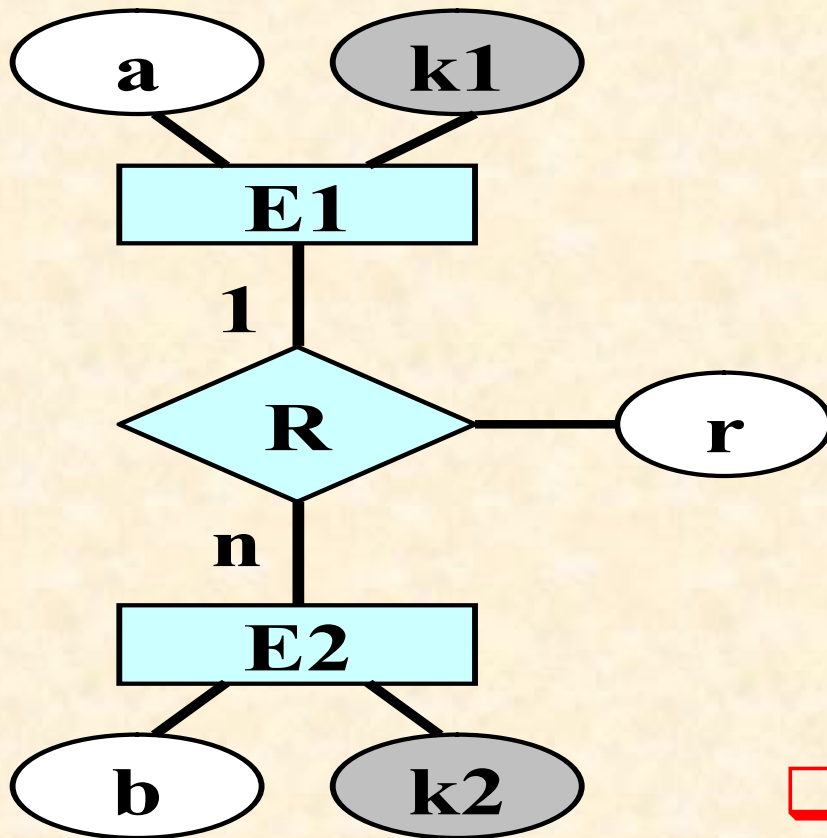
- 关键字是 k_2
- k_1 是关系 E_2 的外关键字

图9.13 1:n二元联系

“一对多” 二元联系的转换

2) 如果多端 E_2 是 ‘非全参与’：

➤ 必须将图9.13 转换成3个关系模式



E1 (k1, a)

➤ 关键字是k1

E2 (k2, b)

➤ 关键字是k2

R (k2, k1, r)

➤ 关键字是k2

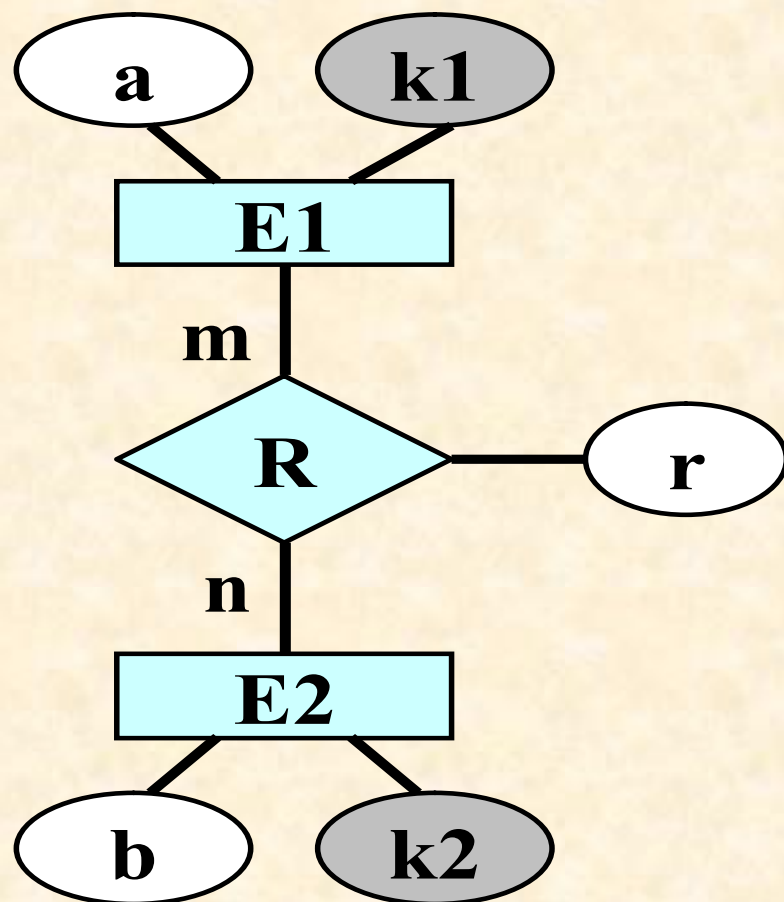
➤ k1是关系R的外关键字

❑ 在这里，关系E1与关系R是不能合并的。

图9.13 1:n二元联系

“多对多” 二元联系的转换

- 将转换成3个关系模式



m:n二元联系

E1 (k1, a)

➤ 关键字是k1

E2 (k2, b)

➤ 关键字是k2

R (k1, k2, r)

➤ (k1, k2)共同构成关系R
的关键字

➤ k1和k2是关系R的两个
外关键字

4. 联系的转换（续）

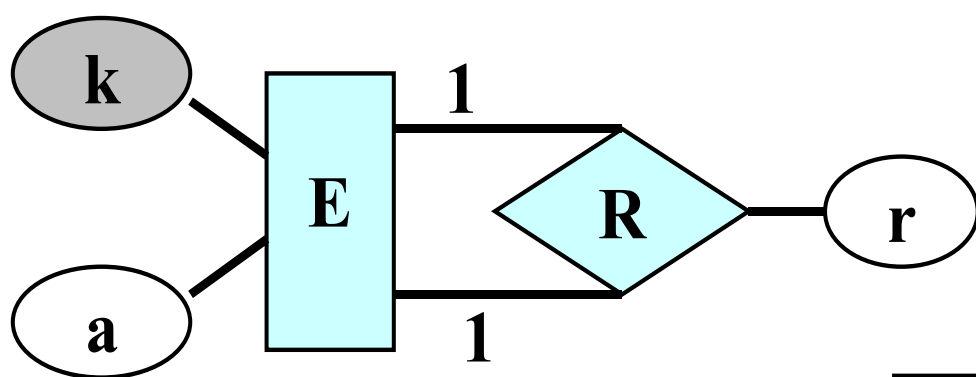
■ 多个实体集之间的多元联系

- 每个实体集转换成一个关系模式
- 联系被单独转换成一个关系模式
 - 其属性包括：
 - a) 联系自身的属性
 - b) 参与联系的每个实体集对应关系的主关键字
- 转换得到的关系模式的关键字一般由所有参与联系的实体集所对应关系模式的关键字联合组成

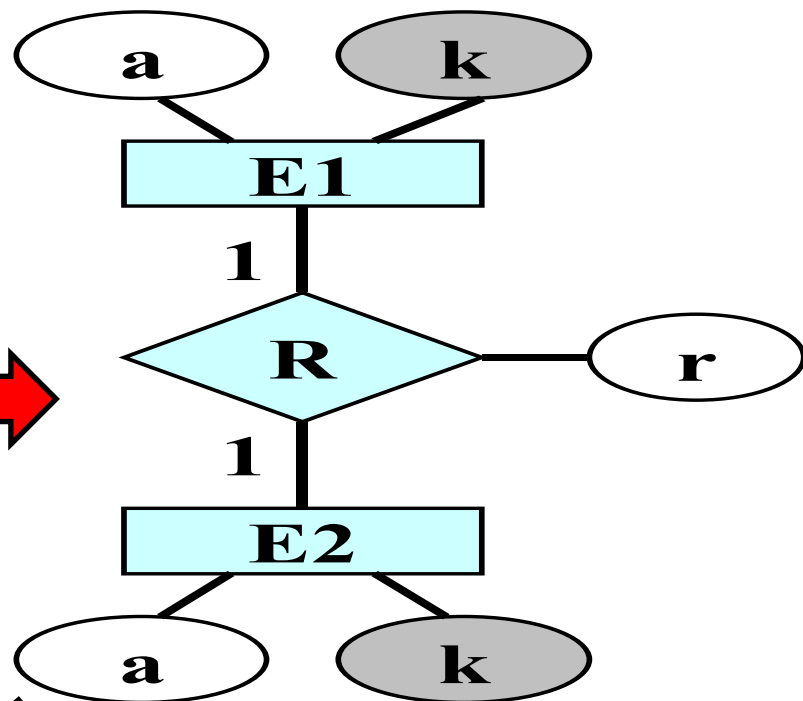
4. 联系的转换（续）

■ 单个实体集内部的联系

- 首先，将该联系转换成两个实体集之间的二元联系
- 再按照二元联系的处理方式转换成关系模式
- 将两个实体集转换得到的关系模式合并为一个关系



单个实体集内部的 1:1 联系



将实体集 E 分解为 E1, E2 两个实体集

□ 转换成三个关系模式

E1 (k, a)

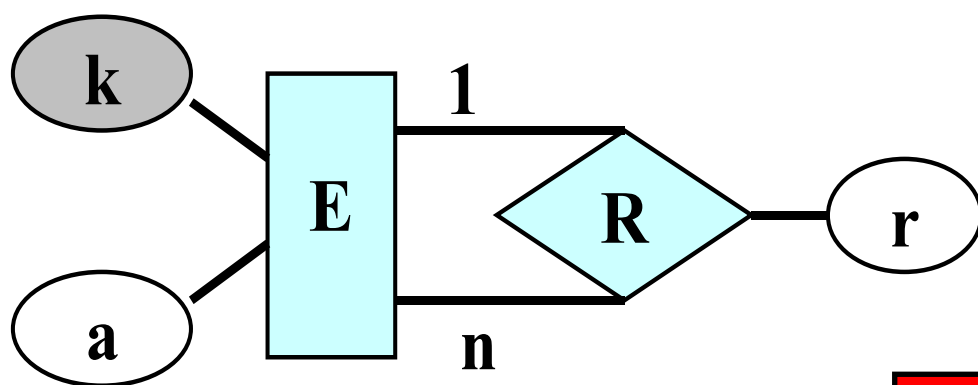
E2 (k, a)

R (k1, k2, r)

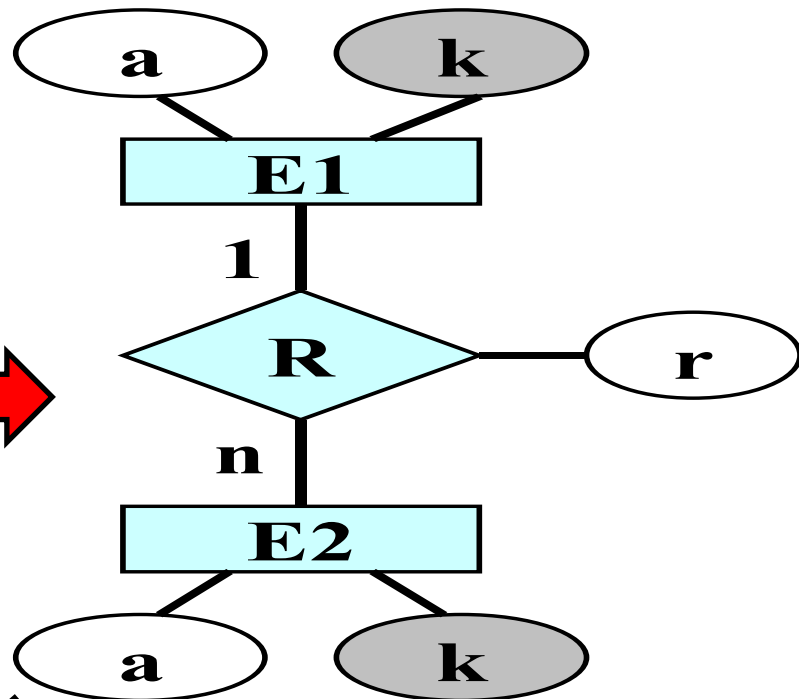
其中：k1和k2是R的两个候选码，k1对应E1的主码，k2对应E2的主码

□ 合并成一个关系模式

E (k, a, k', r)



单个实体集内部的 1:n 联系



将实体集 E 分解为 $E1, E2$ 两个实体集

□ 转换成两个关系模式

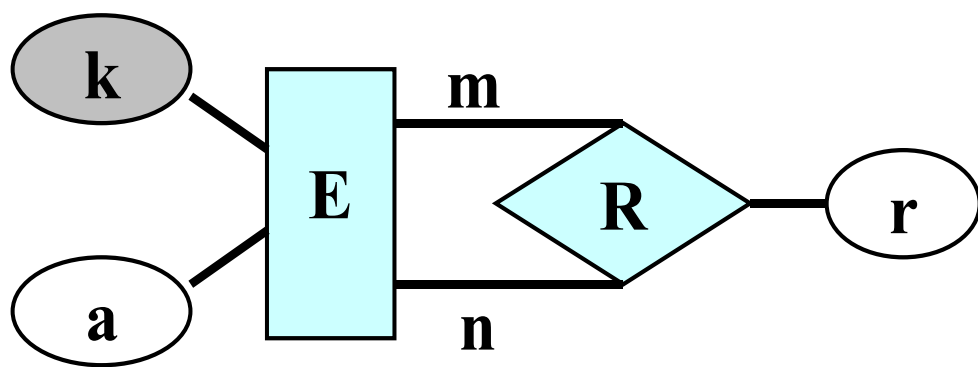
$E1(k, a)$

$E2(k, a, k1, r)$

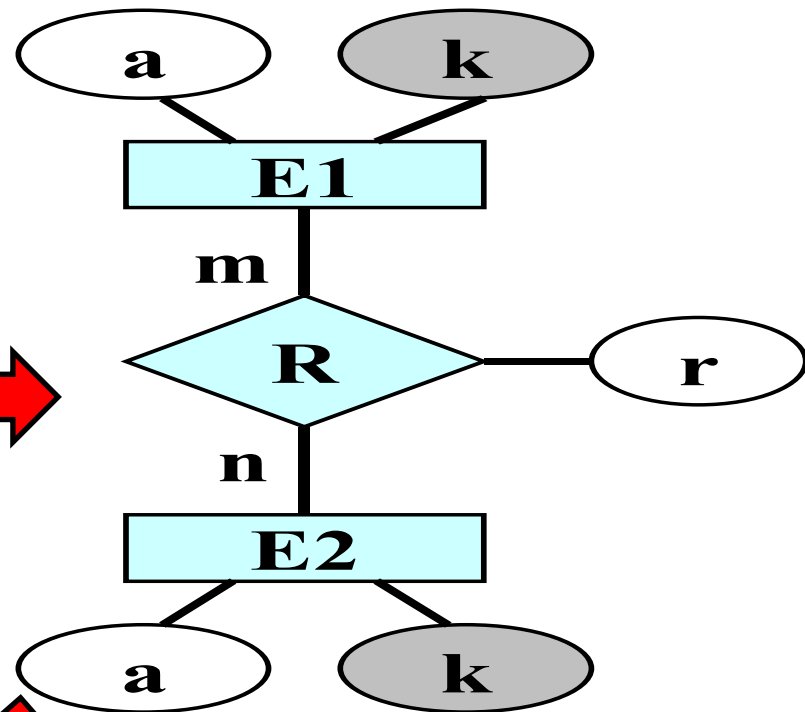
其中: $k1$ 对应 $E1$ 的主码

□ 合并成一个关系模式

$E(k, a, k', r)$



单个实体集内部的 $m:n$ 联系



将实体集 E 分解为 $E1, E2$ 两个实体集

□ 转换成三个关系模式

$E1(\underline{k}, a)$

$E2(\underline{k}, a)$

$R(k1, k2, r)$

其中: $(k1, k2)$ 是 R 的主码,
 $k1$ 对应 $E1$ 的主码, $k2$ 对应 $E2$ 的主码

□ 合并成两个关系模式

$E(\underline{k}, a)$

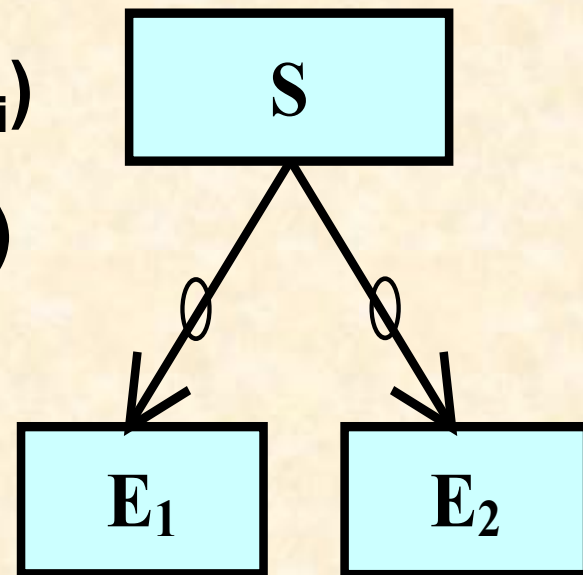
$R(k1, k2, r)$

9.4.1 逻辑设计的基本方法

5. 继承的转换

□ 假设超实体集**S**与其两个子实体集**E₁**和**E₂**构成如右图所示的继承关系。其中：

- 实体集**S**的属性有(**k**,**A₁**,**A₂**,...,**A_n**), **k**是其标识属性
- 实体集**E₁**的属性有(**B₁₁**,**B₁₂**,..., **B_{1i}**)
- 实体集**E₂**的属性有(**B₂₁**,**B₂₂**,...,**B_{2j}**)

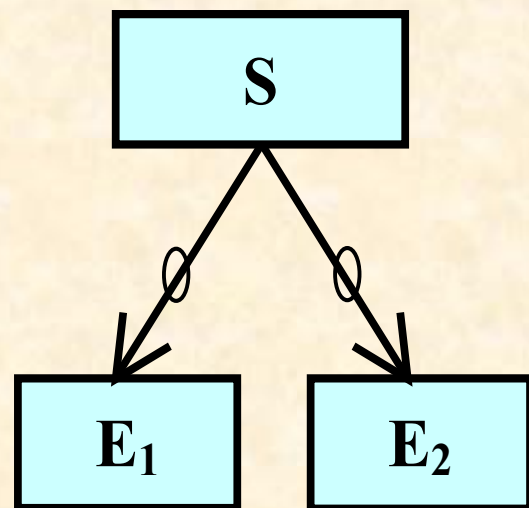


9.4.1 逻辑设计的基本方法

□ 实体集S的属性有 $(k, A_1, A_2, \dots, A_n)$ ， k 是其标识属性

□ 实体集 E_1 的属性有 $(B_{11}, B_{12}, \dots, B_{1i})$

□ 实体集 E_2 的属性有 $(B_{21}, B_{22}, \dots, B_{2j})$



转换方式1: 每一个实体集都将被转换为一个关系

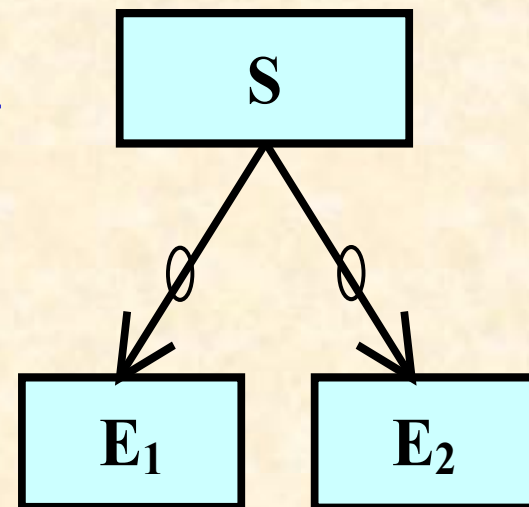
S (**k**, **A**₁, **A**₂, ..., **A**_n)

E₁ (**k**, **B**₁₁, **B**₁₂, ..., **B**_{1i})

E₂ (**k**, **B**₂₁, **B**₂₂, ..., **B**_{2j})

9.4.1 逻辑设计的基本方法

- 实体集S的属性有 $(k, A_1, A_2, \dots, A_n)$ ， k 是其标识属性
- 实体集 E_1 的属性有 $(B_{11}, B_{12}, \dots, B_{1i})$
- 实体集 E_2 的属性有 $(B_{21}, B_{22}, \dots, B_{2j})$



转换方式2： 只有最底层的叶子结点才会被转换为关系，并从其所有超实体集中继承属性。

$E_1(\underline{k}, A_1, A_2, \dots, A_n, B_{11}, B_{12}, \dots, B_{1i})$

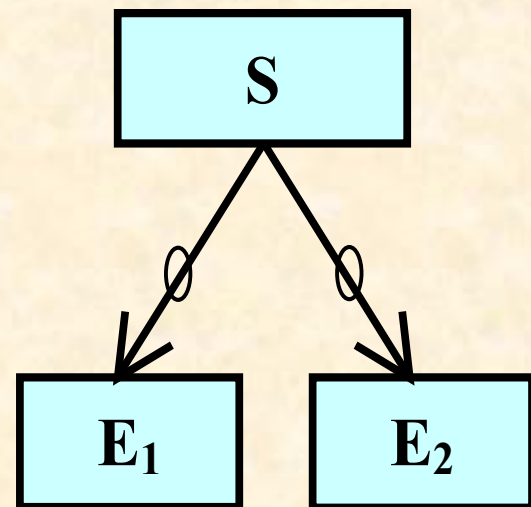
$E_2(\underline{k}, A_1, A_2, \dots, A_n, B_{21}, B_{22}, \dots, B_{2j})$

9.4.1 逻辑设计的基本方法

□ 实体集S的属性有 $(k, A_1, A_2, \dots, A_n)$ ， k 是其标识属性

□ 实体集 E_1 的属性有 $(B_{11}, B_{12}, \dots, B_{1i})$

□ 实体集 E_2 的属性有 $(B_{21}, B_{22}, \dots, B_{2j})$



转换方式3: 被转换为单个关系，其中含有所有实体集中的属性。

$E_1(\underline{k}, A_1, A_2, \dots, A_n, B_{11}, B_{12}, \dots, B_{1i}, B_{21}, B_{22}, \dots, B_{2j})$

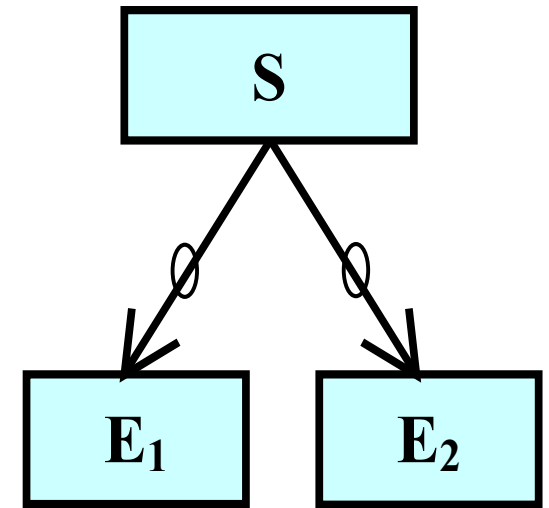
三种不同转换方式之间的对比

转换方式1:

$S(\underline{k}, A_1, A_2, \dots, A_n)$

$E_1(\underline{k}, B_{11}, B_{12}, \dots, B_{1i})$

$E_2(\underline{k}, B_{21}, B_{22}, \dots, B_{2j})$



转换方式2:

$E_1(\underline{k}, A_1, A_2, \dots, A_n, B_{11}, B_{12}, \dots, B_{1i})$

$E_2(\underline{k}, A_1, A_2, \dots, A_n, B_{21}, B_{22}, \dots, B_{2j})$

转换方式3:

$E_1(\underline{k}, A_1, A_2, \dots, A_n, B_{11}, B_{12}, \dots, B_{1i}, B_{21}, B_{22}, \dots, B_{2j})$

9.4.1 逻辑设计的基本方法

6. 规范化

- 通过关系数据库的规范化过程，使设计得到的结果关系模式至少需满足第三范式(3NF)

9.4.1 逻辑设计的基本方法

7. 关系数据库管理系统 (DBMS)

➤ 为满足RDBMS在性能、存储空间等方面的要求及其它限制条件所做的调整与修改。包括：

(1) 逆规范化

— 减少关系的连接运算次数，提高系统性能

(2) 关系的分割

(3) 尽量使用快照

9.4.1 逻辑设计的基本方法

(2) 关系的分割

➤调整每个关系的大小，提高存取效率

➤常用分割方法：

■ 水平分割

— 将一个关系的元组集合划分为若干个不相交的子集，每个子集对应一个子关系模式

■ 垂直分割

— 将一个关系模式纵向分解成若干个子关系模式，并具有无损联接性。

9.4.1 逻辑设计的基本方法

(3) 尽量使用快照

➤ 快照(snapshot)

- 用数据查询命令定义，由系统事先生成查询结果后并保留在数据库中的实关系。

➤ 快照的维护

- 周期性地刷新
- 由用户手工刷新

9.4.1 逻辑设计的基本方法

8. 约束条件的设置

- 完整性约束
- 安全性约束
- 数据类型约束
- 数据量的约束
- 重新设置每个表的候选键、主键及外键

9.4.2 关系视图的设计

□ 关系视图

- 在关系模式基础上所设计的直接面向操作用户的视图即用户的‘**外模式**’，它可以根据用户的需求随时构造，一般 **RDBMS** 均提供关系视图的功能。
- 关系视图的作用
 - 提供数据的逻辑独立性
 - 能适应用户对数据的不同需求
 - 有一定数据保密功能

数据库逻辑设计案例

- 案例1: 图书借阅管理 (<=)
- 案例2: 篮球联赛管理 (<=)
- 案例3: 邮件信息管理 (<=)
- 案例4: 旅游信息管理 (<=)
- 案例5: 学生选课管理 (<=)

数据库设计的综合案例

□案例1:

➤ 题目

参考答案

□案例2:

➤ 题目

参考答案

9.1 数据库设计概述

9.2 数据库设计的需求分析

9.3 数据库的概念设计

9.4 数据库的逻辑设计

9.5 数据库的物理设计

9.5 数据库的物理设计

□ 设计目标

- 对数据库内部物理结构作调整并选择合理的存取路径，以提高数据库访问速度及有效利用存储空间。

□ 在关系数据库设计中，用户参与物理设计的内容有：

- 存取方法的设计
- 存储结构的设计

9.5.1 存取方法的设计

□ 在数据库系统中，物理数据的存取方法有以下三种

- 索引 (Index)
- 集簇 (Cluster)
- HASH

9.5.1 存取方法的设计

□ 索引设计的考虑因素

- 在主关键字及外关键字上建立索引
 - 提高关系联接查询的速度
 - 有利于实体完整性及引用完整性的检查
- 以读为主的关系应尽可能多地建立索引
- 如果根据某属性的等值查询所得到的结果元组数量较少，则可以考虑对该属性建立索引
- 对经常用于统计查询的属性建立索引
 - 可以根据索引数据直接获取统计结果，不必再去访问对应关系的数据块。

9.5.1 存取方法的设计

□ 集簇设计的考虑

➤ 集簇 (Cluster)

- 将有关的数据元组集中存放于一个或相邻的物理块内或同一柱面内以提高查询效率。（图）

➤ 集簇的优点与缺点

■ 优点：

- 可以明显提高某些特定应用的查询效率

■ 缺点：

- 可能会影响到其它应用的访问效率
- 建立集簇的开销较高

9.5.1 存取方法的设计

□ 建立集簇时的考虑因素

- “通过集簇属性的访问” 是相关表上的主要应用
- 一个集簇属性值所对应的元组数量不能太少，也不宜过多
- 集簇属性上的值应该相对稳定

9.5.2 存储结构的设计

□ 存储结构的设计又包括两个方面：

- 数据存放位置的设计（又称分区设计）
- 系统参数配置

9.5.2 存储结构的设计

□ 数据存放位置设计的指导原则

- 减少访盘冲突，提高I/O并行性
 - 提高磁盘访问的并发度
- 分散热点数据，均衡I/O负担
 - 均衡磁盘访问负载，有效利用多磁盘之间的并发访问能力
- 保证关键数据快速访问，缓解系统瓶颈
 - 避免并发访问操作以及访盘冲突对关键数据访问操作的影响

9.5.2 存储结构的设计

□ 系统参数配置

➤ 设置与调整数据库管理系统的参数配置

➤ 常见的参数如：

- 数据库用户数
- 内存分配参数
- 存储分配参数
- 数据库大小
-

- 同时打开数据库数
- 缓冲区分配参数
- 时间片大小
- 锁的数目等
-

