

Chapter 6: Entity-Relationship Model

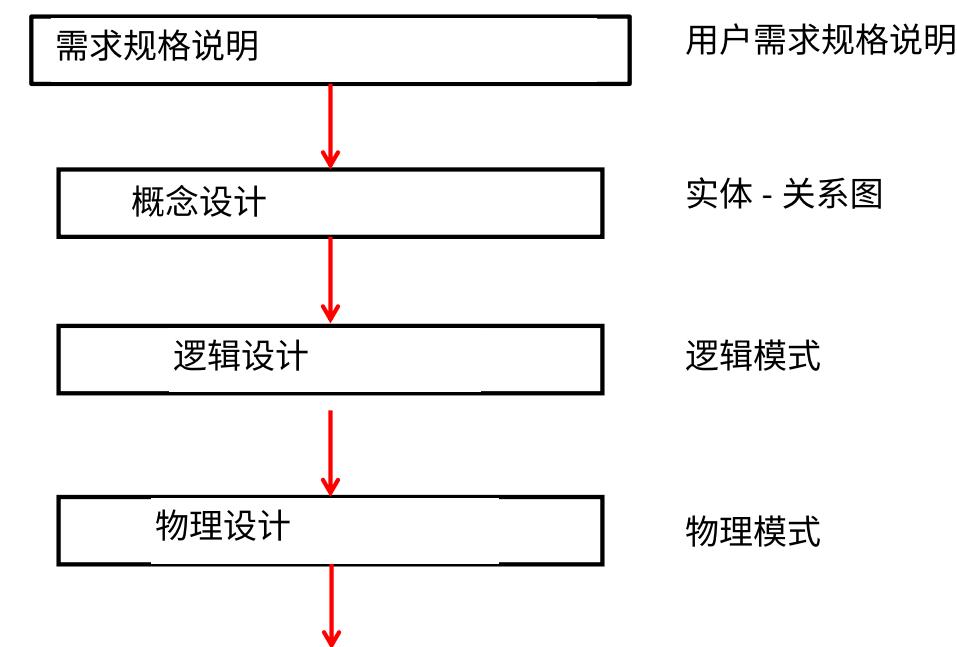
数据库设计过程、建模、约束、弱实体集、转换为关系模式、设计问题、扩展的实体 - 关系 (E - R) 特性、统一建模语言 (UML)

大学模式

大学企业的关系模式

教室 (建筑物, 房间号, 容量)
系 (系名, 建筑物, 预算)
课程(课程编号, 课程名称, 系名, 学分)
教师(教师编号, 姓名, 系名, 工资)
课程安排(课程编号, 课节编号, 学期, 年份, 教学楼, 房间号, 时段编号)
授课(教师编号, 课程编号, 课节编号, 学期, 年份)
学生(学生编号, 姓名, 系名, 总学分)
选课(学生编号, 课程编号, 课节编号, 学期, 年份, 成绩)
导师 (导师ID, 学生ID)
时间段 (时间段ID, 日期, 开始时间, 结束时间)
先修课程 (课程ID, 先修课程ID)

数据库设计过程



设计阶段

初始阶段（需求规格说明）——全面描述潜在数据库用户的数据需求。

第二阶段（概念设计）——选择数据模型

应用所选数据模型的概念
将这些需求转化为数据库的概念模式。

一个完全开发的概念模式表明了企业的功能需求。

• 描述将对数据执行的操作（或事务）类型。

设计阶段（续）

• 最终阶段（数据库设计）——从抽象数据模型过渡到数据库的实现

逻辑设计——确定数据库模式。

• 数据库设计要求我们找到一组“良好”的关系模式。

• 业务决策——我们应该在数据库中记录哪些属性？

• 计算机科学决策——我们应该有哪些关系模式，以及属性应如何在各种关系模式中分布？

物理设计——确定数据库的物理布局

设计方案

• 在设计数据库模式时，我们必须确保避免两种情况

主要陷阱：

• 冗余：糟糕的设计可能导致信息重复。

• 信息的冗余表示可能导致数据

信息的各种副本之间存在不一致性

• 不完整性：糟糕的设计可能使企业的某些方面
难以或无法进行建模。

• 仅避免糟糕的设计是不够的。可能有大量
优秀的设计供我们选择。

• 数据库设计可能是一个具有挑战性的问题

• 巨大的设计空间

设计方法

• 实体 - 关系模型（本章涵盖）

• 将企业建模为实体和关系的集合

• 实体：企业中可与其他对象区分开的“事物”或“对象”

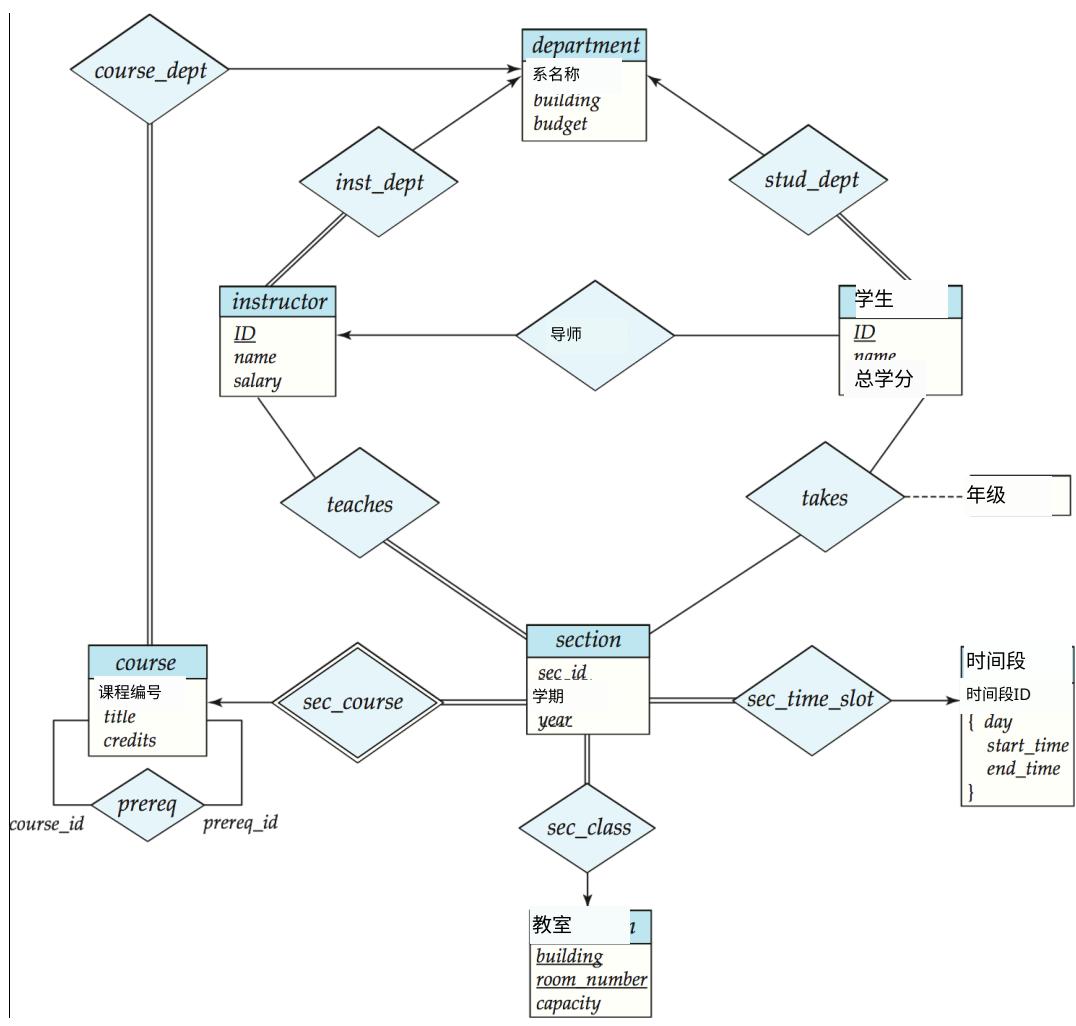
• 由一组属性描述

• 关系：多个实体之间的关联

• 通过实体 - 关系图以图表形式表示：- 规范化理论（第7章）

• 明确哪些设计是糟糕的，并对其进行测试

大学企业的实体 - 关系图



实体关系模型概述

数据库建模

数据库可以建模为：实体的集合，实体之间的关系。

实体是存在且可与其他对象区分开的对象。示例：特定的人、公司、事件、植物。实体具有属性。示例：人有姓名和地址。实体集是具有相同类型且共享相同属性的实体的集合。

示例：所有人、公司、树木、假期的集合

实体集“教师”和“学生”

教师编号 教师姓名

76766	克里克
45565	卡茨
10101	斯里尼瓦桑
98345	金
76543	辛格
22222	爱因斯坦

导师

student-ID 学生姓名

98988	田中
12345	尚卡尔
00128	张
76543	布朗
76653	葵
23121	查韦斯
44553	佩尔蒂埃

学生

在实体 - 关系图中表示实体集

- 实体集可以用图形表示如下：
- 矩形表示实体集。
- 实体矩形内列出的属性
- 下划线表示主键属性

教师	学生
ID 姓名 薪水	ID 姓名 总学分

关系集

关系是多个实体之间的关联

示例：

44553 (佩尔蒂埃) 导师 22222 (爱因斯坦)
学生实体 关系集 教师实体

关系集是 $n \geq 2$ 个实体之间的数学关系，每个实体都取自实体集

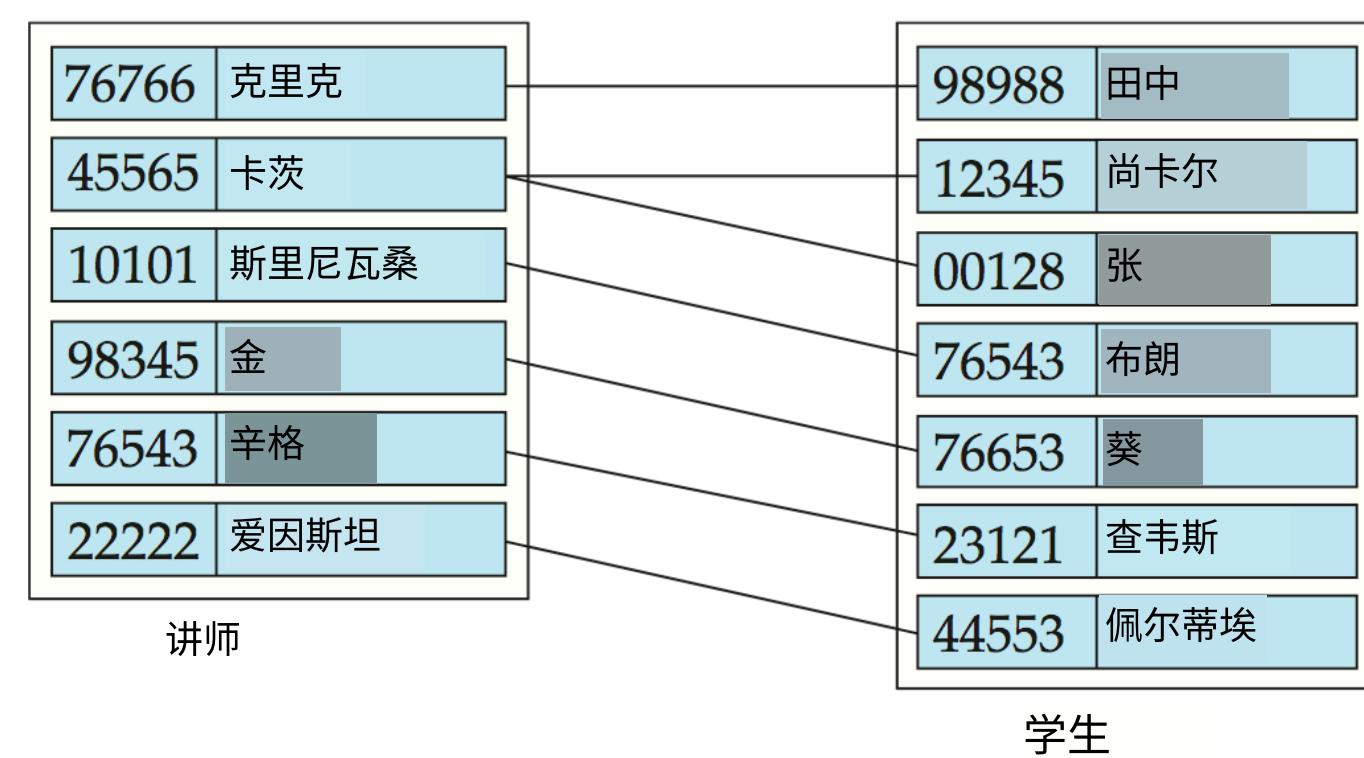
$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

其中 (e_1, e_2, \dots, e_n) 是一种关系

示例：

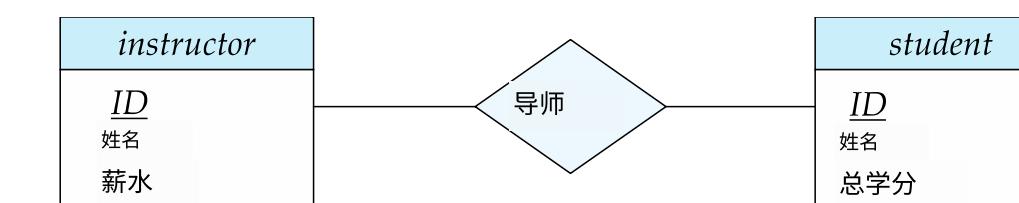
$$(44553, 22222) \in \text{导师}$$

关系集 导师



在实体 - 关系 (ER) 图中表示关系集

- 菱形表示关系集。



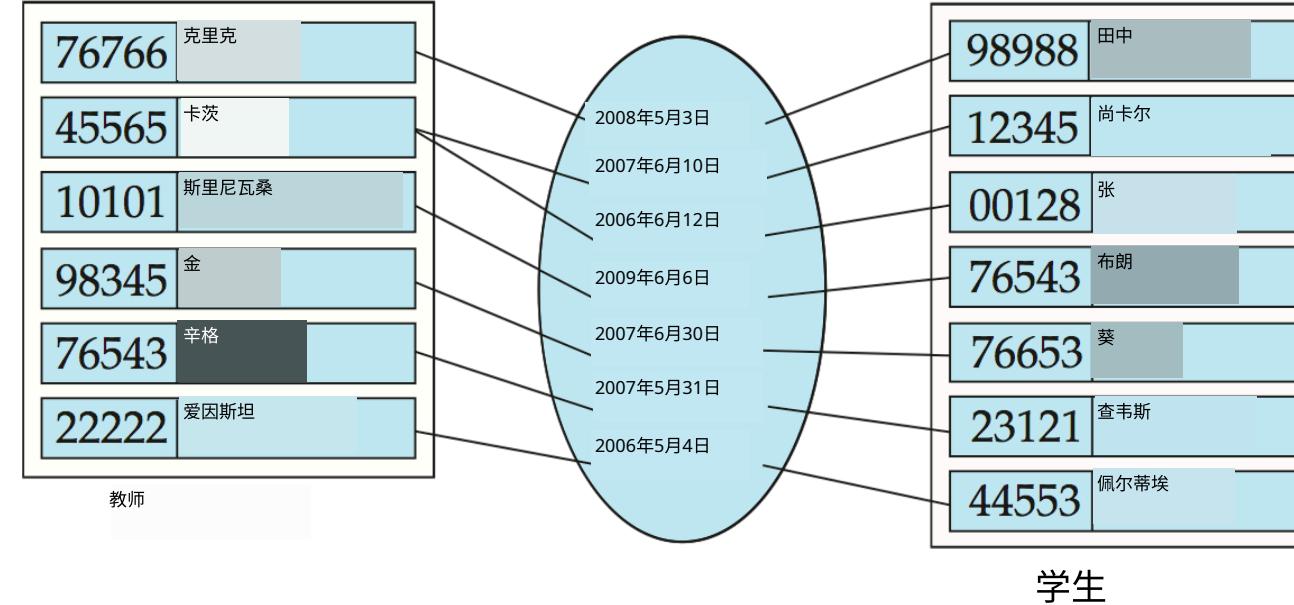
带属性的关系集

属性也可以是关系集的特性。

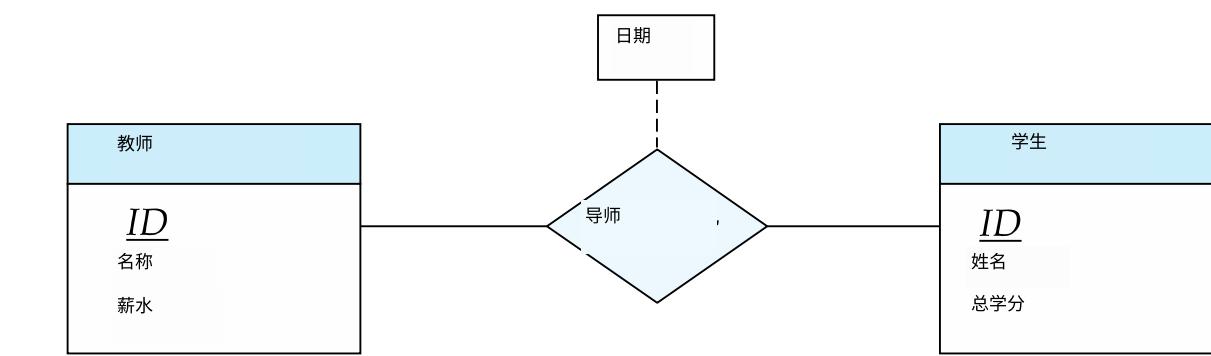
例如，实体集之间的导师关系集

教师和学生可能有日期属性，用于记录

学生何时开始与导师建立关联



带有属性的关系集

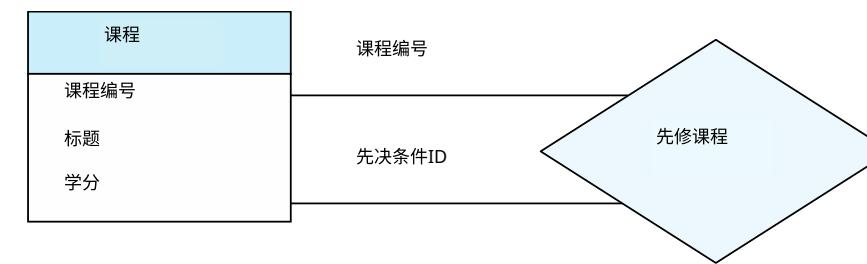


角色

一个关系的实体集不必是不同的

实体集的每次出现都在关系中扮演一个“角色”

标签 “course_id” 和 “prereq_id” 被称为角色。



关系集的度

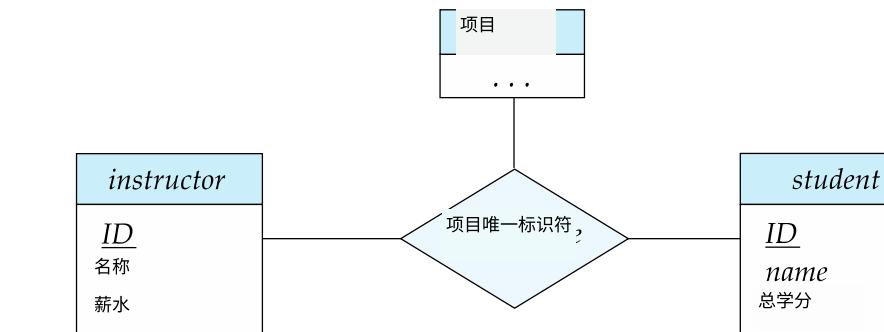
二元关系 (二元联系)

涉及两个实体集（或度为二）。

数据库系统中的大多数关系集是二元的。

有时，将关系表示为非二元关系会更方便。

带有三元关系的实体 - 关系图



属性

实体由一组属性表示，即实体集中所有成员所拥

有的描述性属性。示例：教师

= (ID, name, street, city, salary) 课程 = (课
程ID, 课程名称, 学分) 域 - 每个属性的允许值集

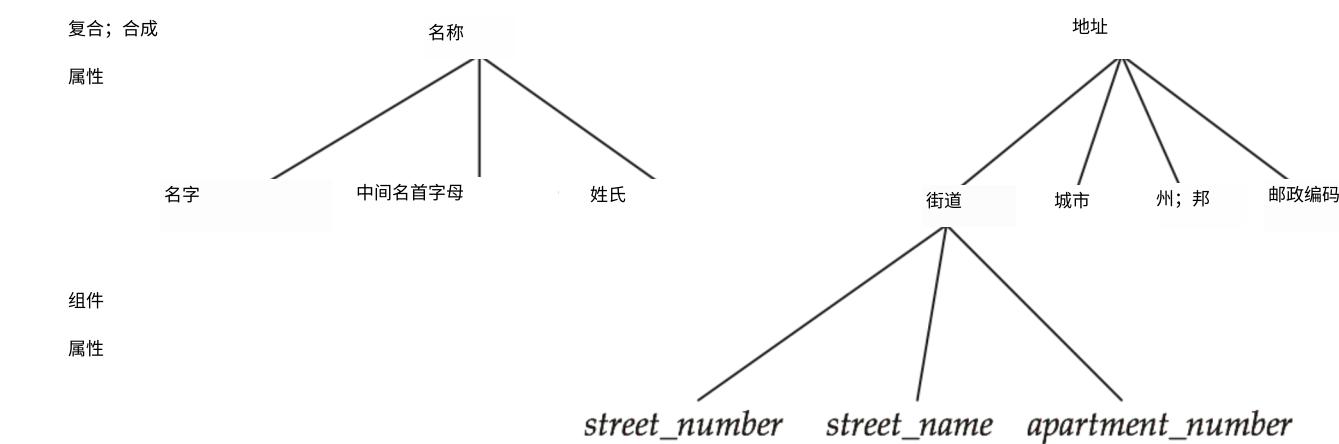
合属性类型：简单（简单）和复合（复合）属

性。单值（单值）和多值（多值）属性 - 示例：

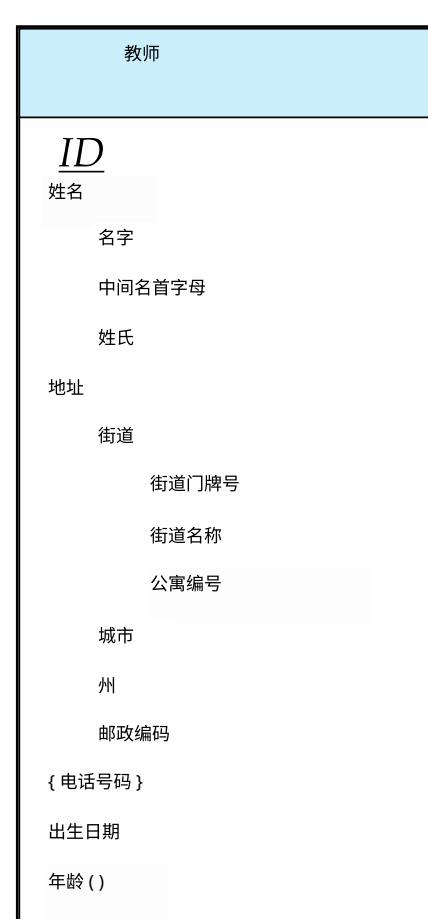
多值属性：电话号码派生（派生）属性 - 可以从
其他属性计算得出 - 示例：给定出生日期时的年

龄

复合属性



在实体 - 关系图中表示复杂属性



映射基数约束(映射基数约束)

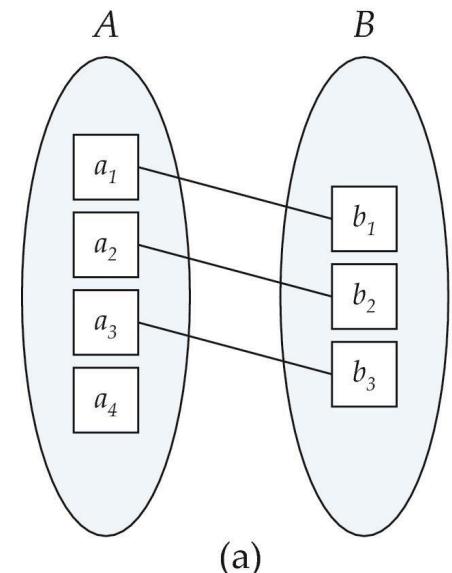
表示一个实体通过一个关系集可以关联的其他实体的数量。

在描述二元关系集时最有用。

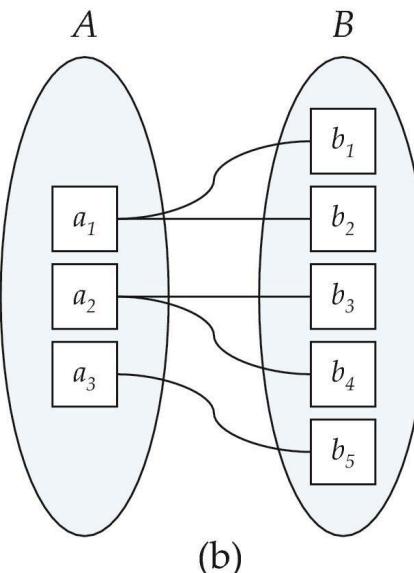
对于二元关系集，映射基数必须是以下类型之一：

- 对一
- 一对多
- 多对一
- 多对多

映射基数



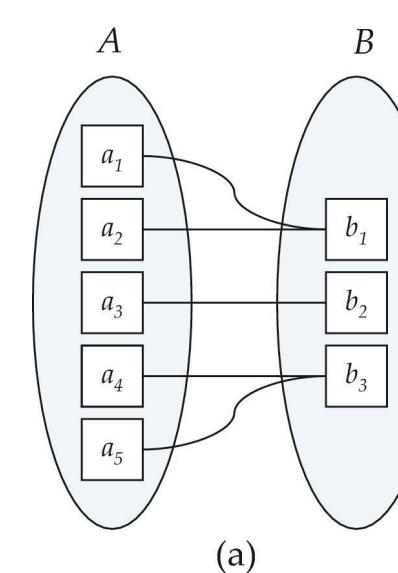
一对一



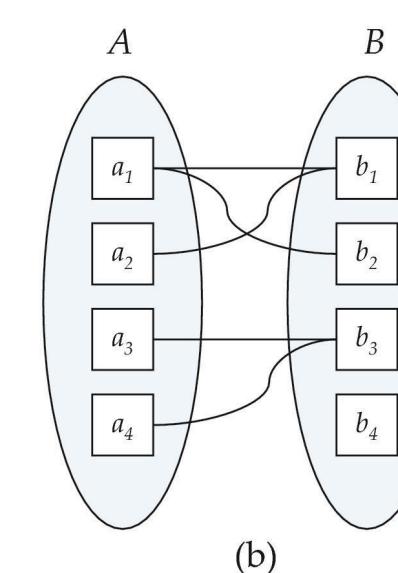
一对多

注意：集合A和B中的某些元素可能未与另一个集合中的任何元素映射

映射基数



多对
one



多对多

注意：集合A和集合B中的某些元素可能未与另一个集合中的任何元素映射

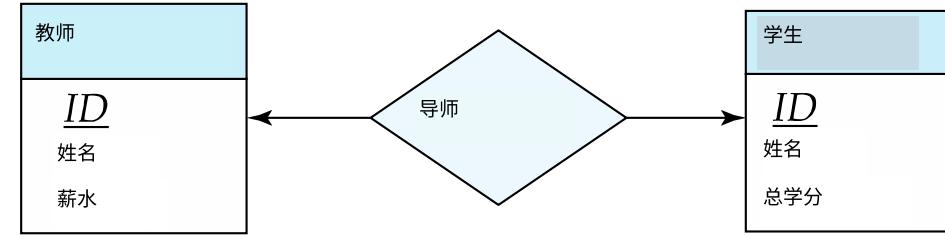
在实体 - 关系 (ER) 图中表示基数约束

我们通过在关系集和实体集之间绘制有向线 (\rightarrow) (表示“一”) 或无向线 ($-$) (表示“多”) 来表达基数约束。

教师与学生之间的一对一关系：

通过“导师”关系，一名学生最多与一名教师相关联
导师

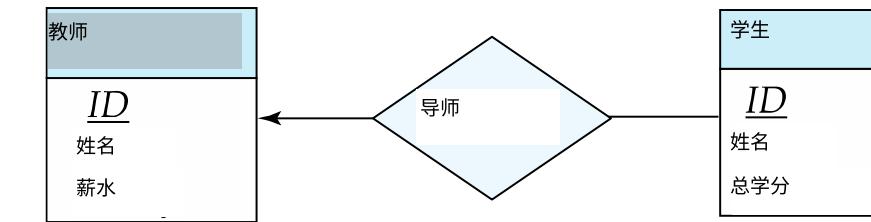
通过“导师”关系，一名教师最多与一名学生相关联



一对多关系

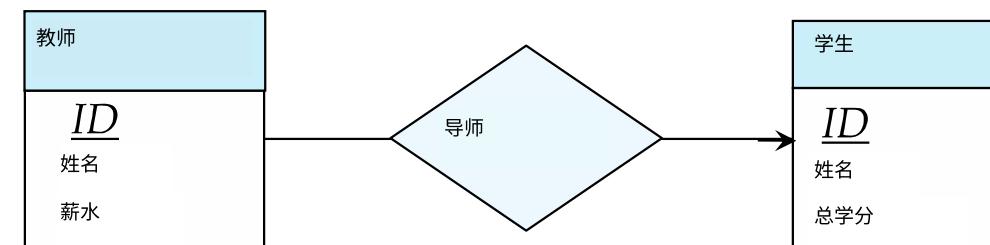
教师与学生之间的一对多关系：一名教师通过导师关系与若干（包括0名）学生相关联

一名学生通过导师关系最多与一名教师相关联



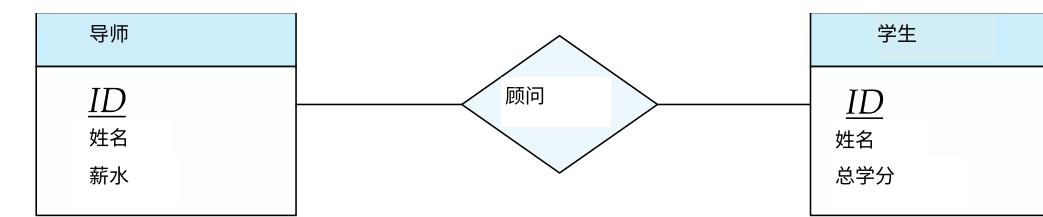
多对一关系

在教师与学生的多对一关系中，一名教师最多通过导师关系与一名学生关联，而一名学生通过导师关系与多名（包括0名）教师关联



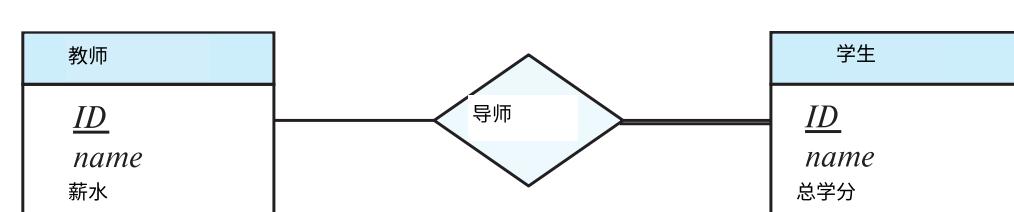
多对多关系

一位教师通过导师关系与若干（可能为0）名学生相关联；一名学生通过导师关系与若干（可能为0）名教师相关联



完全参与和部分参与

•完全参与（用双线表示）：实体集中的每个实体至少参与关系集中的一个关系



学生在导师关系中的参与是完全参与

•每个学生必须有一位关联的导师

•部分参与：某些实体可能不参与关系集中的任何关系

示例：导师在导师关系中的参与是部分参与

表达更复杂约束的表示法

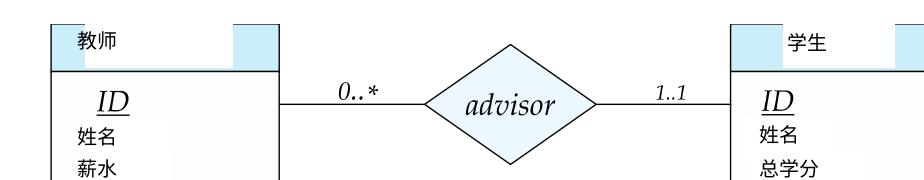
•一条线可能有相关的最小和最大基数，以 $l..h$ 的形式表示，其中 l 是最小基数， h 是最大基数

•最小值为 1 表示完全参与。

•最大值为 1 表示该实体最多参与一个关系

•最大值为 * 表示无限制。

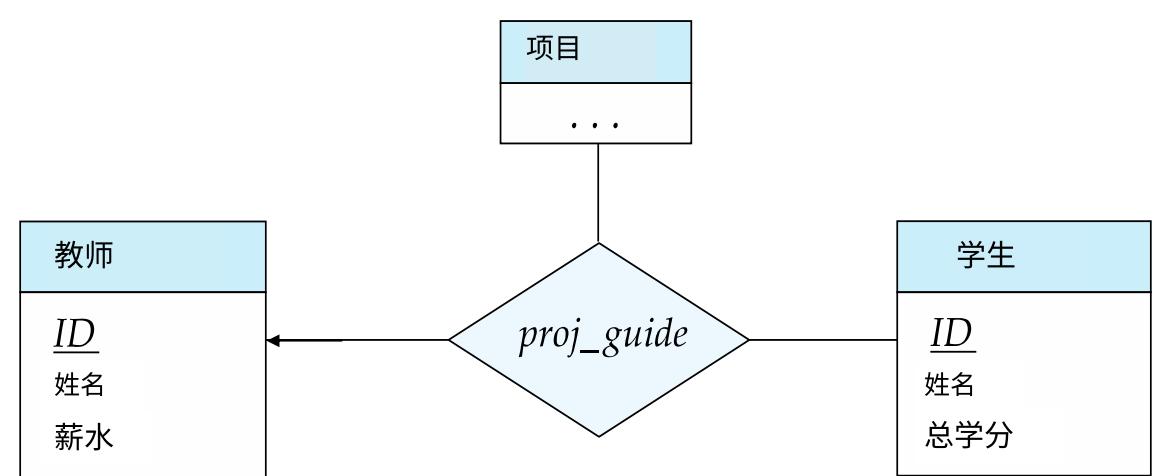
•示例



•教师可以指导0名或多名学生。一名学生必须有1名导师（教师）；不能有多名导师（教师）。

三元关系的基数约束

我们最多允许从一个三元（或更高阶）关系中引出一个箭头来表示基数约束。
例如，从proj_guide到instructor的箭头表示每个学生在一个项目中最多有一个指导老师。
为避免混淆，我们禁止使用多个箭头。



主键

主键提供了一种指定如何区分实体和关系的方法。我们将考虑：

实体集
关系集
弱实体集

实体集的主键

根据定义，单个实体是不同的。
从数据库的角度来看，它们之间的差异必须通过其属性来表达。
实体的属性值必须能够唯一地标识该实体。

实体集中的任意两个实体都不允许所有属性的值完全相同。
实体的键是一组足以区分各个实体的属性。

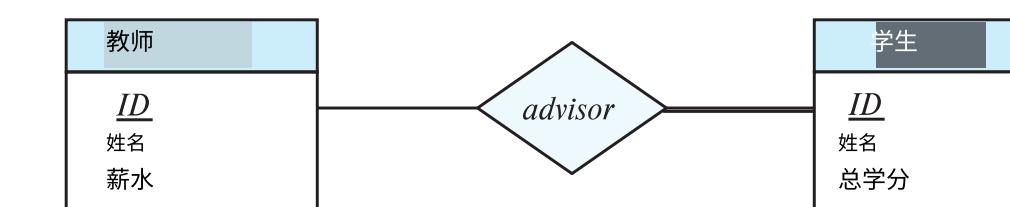
关系集的主键

为了区分关系集中的各种关系，我们使用关系集中实体的各个主键。

设 R 为涉及实体集 E_1, E_2, \dots, E_n 的关系集。
 R 的主键由实体集 E_1, E_2, \dots, E_n 的主键的并集组成

如果关系集 R 具有与之关联的属性 a_1, a_2, \dots, a_m ，那么 R 的主键还包括属性 a_1, a_2, \dots, a_m
示例：关系集“导师”。

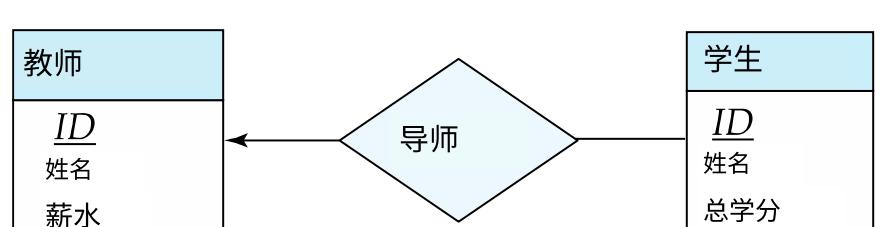
主键由教师编号和学生编号组成



关系集主键的选择取决于该关系集的映射基数。

二元关系的主键

多对多关系。前面主键的并集是最小超键，并被选为主键。
一对多关系。“多”方的主键是最小超键，并用作主键。
多对一关系。“多”方的主键是最小超键，并用作主键。
一对一关系。参与实体集之一的主键构成最小超键，并且可以选择其中之一作为主键。



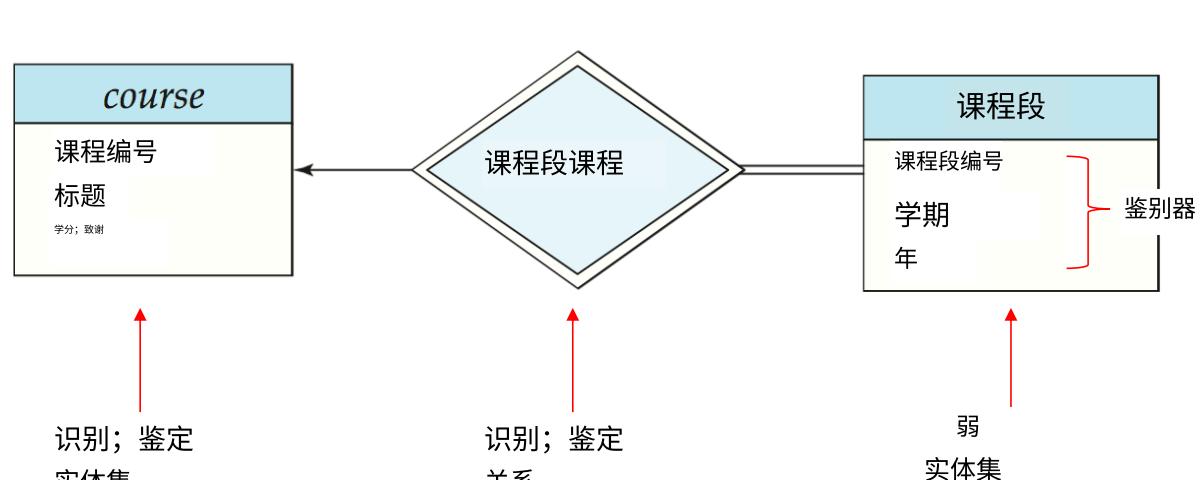
弱实体集(弱实体集)

没有主键的实体集被称为弱实体集。
弱实体集的存在依赖于标识性实体集(标识性实体集)的存在
它必须通过一个从标识性实体集到弱实体集的全参与、一对多的关系集与标识性实体集相关联
标识性联系 (标识性联系) 用双菱形表示

弱实体集的分辨符（或部分键）是这样一组属性：当已知弱实体集所依赖的标识实体时，这些属性能够区分弱实体集中的所有实体。
弱实体集的主键由弱实体集所依赖存在的强实体集的主键，加上弱实体集的分辨符构成。

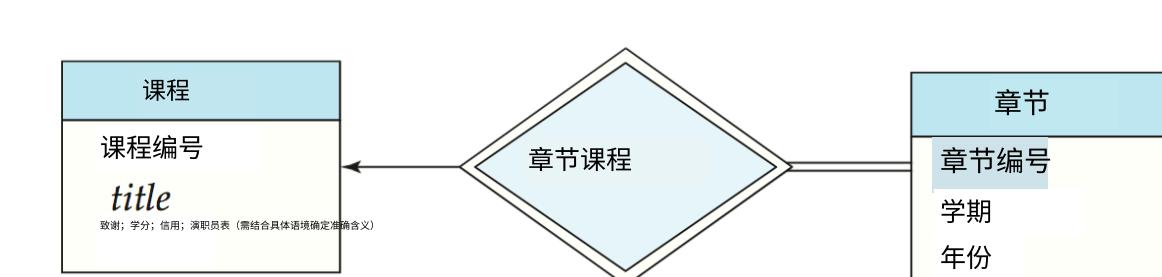
弱实体集（续）

我们用虚线标注弱实体集的鉴别符。
我们将弱实体的标识关系置于双菱形中。
课程段的主键 - (课程编号, 课程段编号, 学期, 年份)



弱实体集（续）

注意：强实体集的主键不会显式地与弱实体集一起存储，因为它隐含在标识关系中。
如果显式存储course_id, section可以成为一个强实体，但这样
一来，section和course之间的关系将被由course和section共有的属性course_id定义的隐式关系所重复



冗余属性

假设我们有实体集：

学生，属性有：ID、姓名、总学分、系名

系，属性有：系名、教学楼、预算

我们使用关系集stud_dept来建模每个学生都有一个关系联系这

一事实

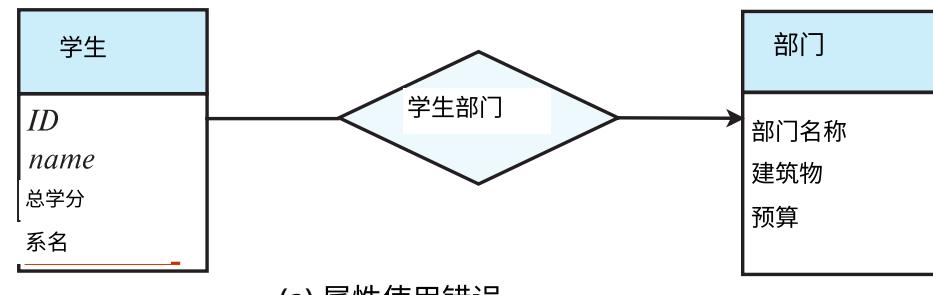
下面学生实体集中的属性系名重复了关系中已有的信息，因

此是冗余的

并且需要被移除。

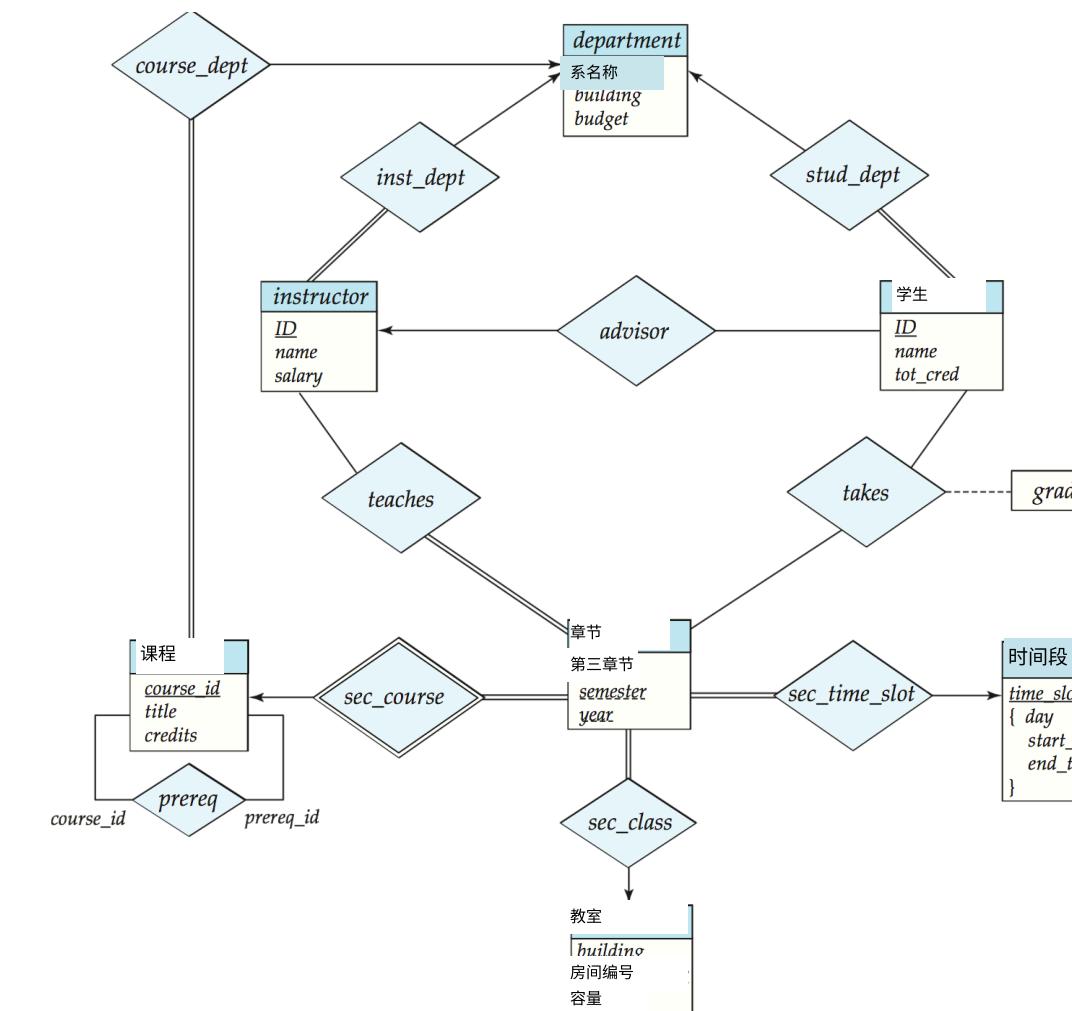
但是：当转换回表格时，在某些情况下该属性会重新引入，

我们稍后会看到。



(a) 属性使用错误

大学企业的实体 - 关系图



转换为关系模式

实体集和关系集可以统一表示为代表数据库内容的关系模式。符合实体 - 关系图的数据库可以用一组模式来表示。

对于每个实体集和关系集，都有一个唯一的模式，该模式被赋予相应实体集或关系集的名称。

每个模式都有若干列（通常对应于属性），这些列具有唯一的名称。

Reduction to Relational Schemas

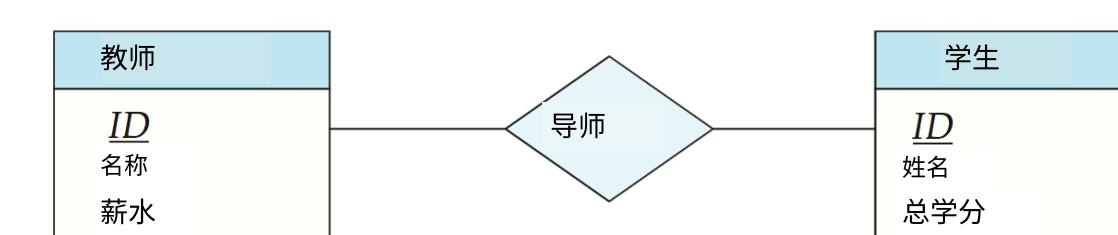
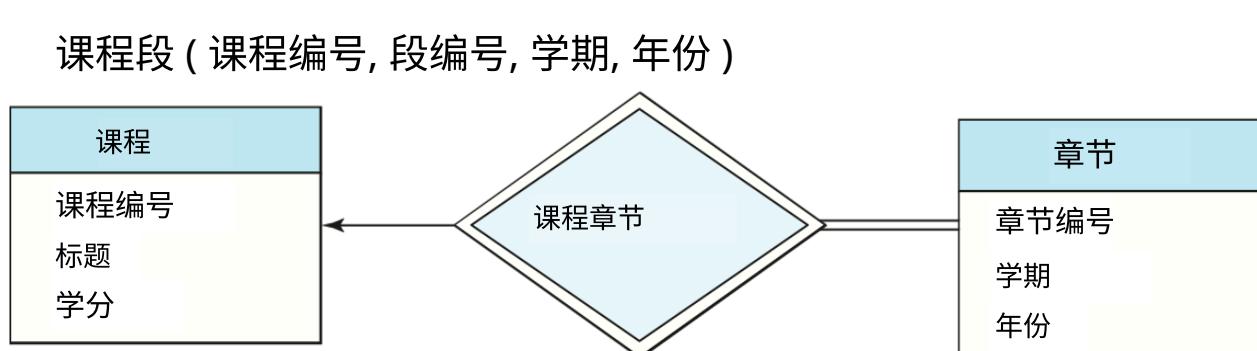
用简单属性表示实体集

一个强实体集可简化为具有相同属性的模式 课程(课程编号, 课程名称, 学分)

一个弱实体集成为一个表，该表包含一个用于标识强实体集主键的列 该表的主键是弱实体集的鉴别符和标识强实体集主键的并集

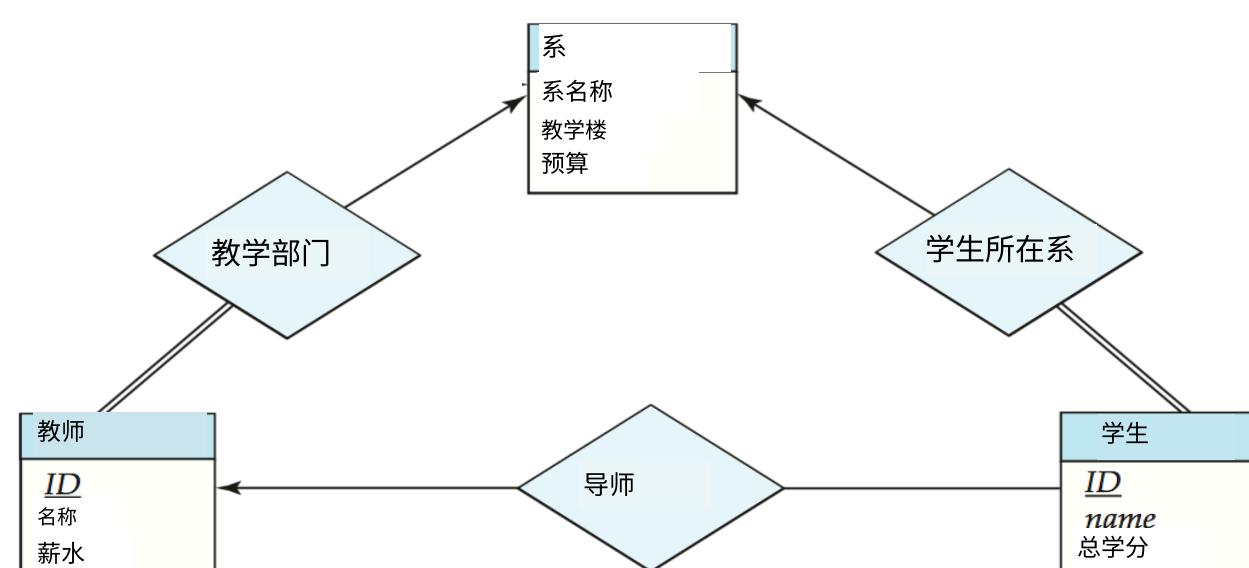
表示关系集

多对多关系集表示为一个模式，该模式包含两个参与实体集的主要属性以及该关系集的任何描述性属性。示例：关系集advisor的模式advisor = (s_id, i_id)

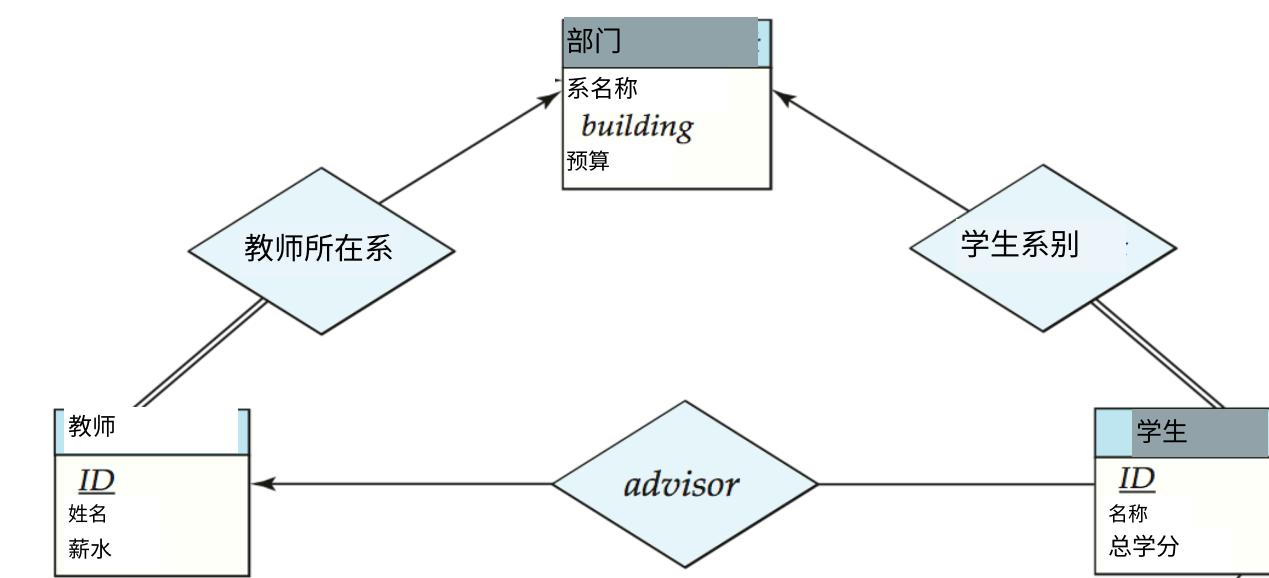


模式冗余

在多端为全参与的多对一和一对多关系集，可以通过在“多”端添加一个额外属性来表示，该属性包含“一”端的主键。示例：不针对关系集inst_dept创建模式，而是在由实体集instructor产生的模式中添加一个属性dept_name



模式冗余



部门(系名, 大楼, 预算) 教师(ID, 姓名, 薪水) 教师部门(ID, 系名) → 部门(系名, 大楼, 预算) 教师(ID, 姓名, 薪水, 系名)

模式冗余 (续)

对于一对多关系集，任一侧都可被选作“多”的一侧

也就是说，可以将额外的属性添加到与两个实体集对应的任意一个表中。如果“多”的一侧是部分参与，在与“多”的一侧对应的模式中用一个额外属性替换某个模式可能会导致出现空值。

将弱实体集与其强实体集相连的关系集所对应的模式是冗余的。

示例：章节模式已经包含了会出现在sec_course模式中的属性

复合和多值属性

实体 E 的多值属性 M 由单独的模式 EM 表示

模式 EM 具有与实体 E 的主键相对应的属性以及与多值属性 M 相对应的属性

示例：教师的多值属性电话号码由以下模式表示：

`inst_phone = (ID, 电话号码)`

多值属性的每个值都映射到模式 EM 上关系的一个单独元组

例如，主键为 22222 且电话号码为 456 - 7890 和 123 - 4567 的教师实体映射到两个元组：(22222, 456 - 7890)

(22222, 123 - 4567)

设计问题

复合属性和多值属性

教师	
ID	
姓名	
名字	
中间名首字母	
姓氏	
地址	
街道	
街道编号	
街道名称	
公寓编号	
城市	
状态	
zip	
{ 电话号码 }	
出生日期	
年龄()	

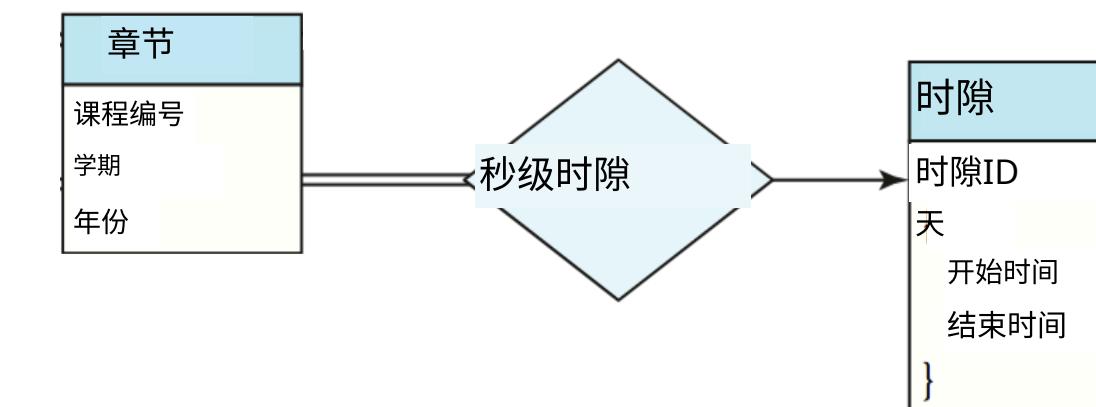
通过为每个组件属性创建单独的属性来展平复合属性

示例：给定实体集“教师”，其复合属性“姓名”包含组件属性“名”和“姓”，则该实体集对应的模式有两个属性“姓名_名”和“姓名_姓” - 如果没有歧义则省略前缀忽略多值属性，扩展的教师模式为

教师(ID,
名字, 中间名首字母, 姓氏, 街道编号, 街道名
称, 公寓编号, 城市, 州, 邮政编码, 出生日期,
年龄)

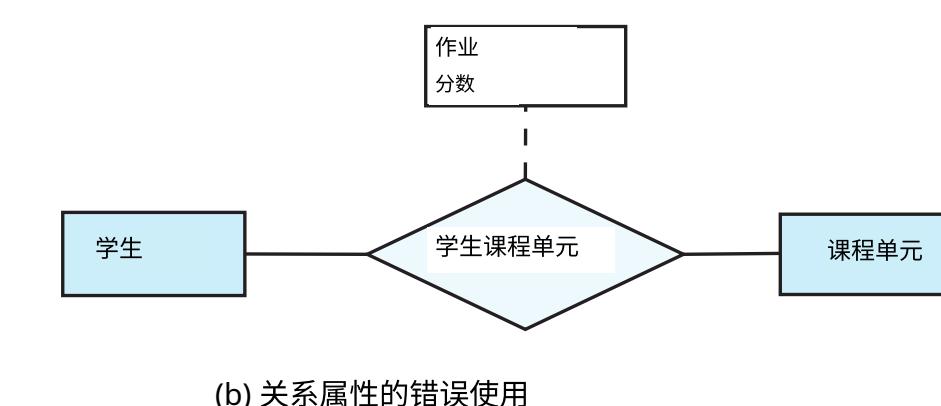
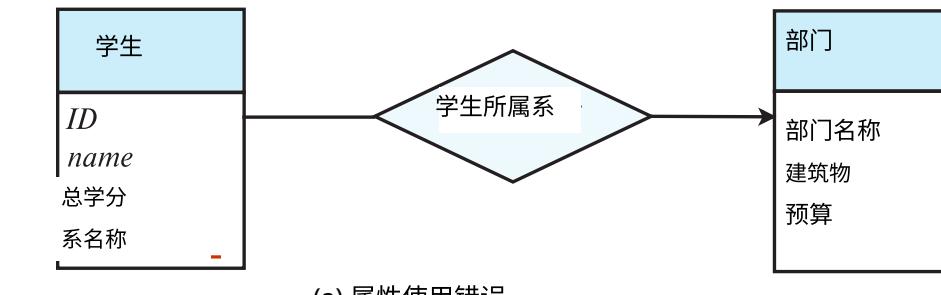
多值属性 (续)

特殊情况：实体time_slot除主键属性外只有一个属性，且该属性为多值属性time_slot(time_slot_id)时间段详情(时间段ID, 日期, 开始时间, 结束时间)优化：不创建与实体对应的关系，仅创建与多值属性对应的关系time_slot(时间段ID, 日期, 开始时间, 结束时间)注意(警告)：由于此优化，章节 (来自 sec_time_slot) 的time_slot属性不能作为外键

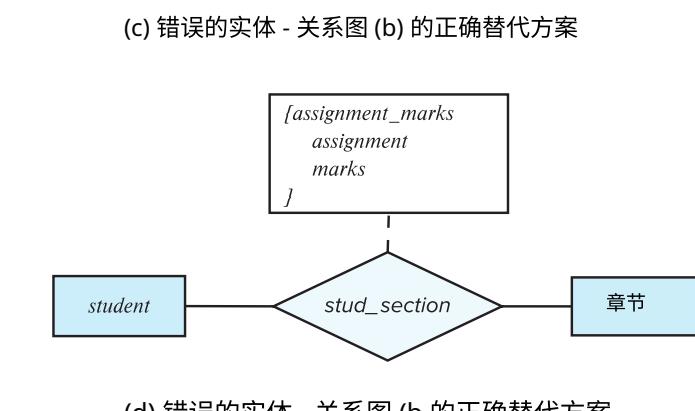
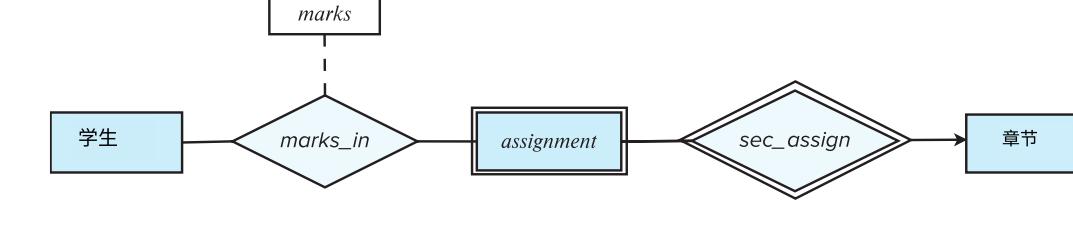
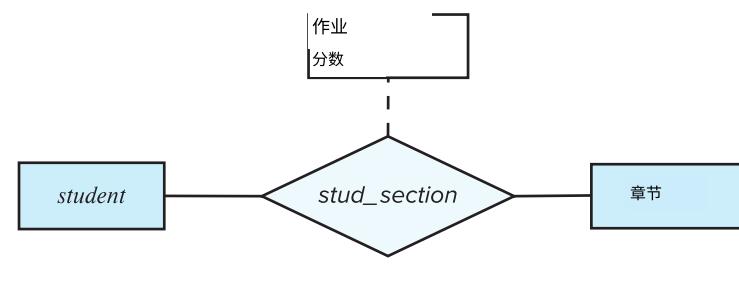


实体 - 关系图中的常见错误

• 错误的实体 - 关系图示例

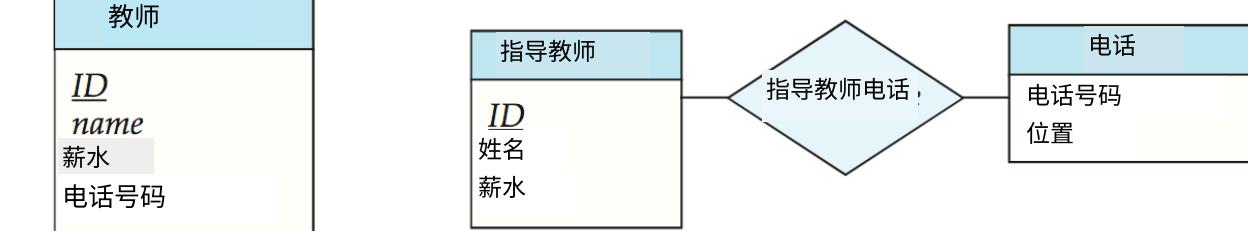


E - R 图中的常见错误



设计问题

实体集与属性的使用

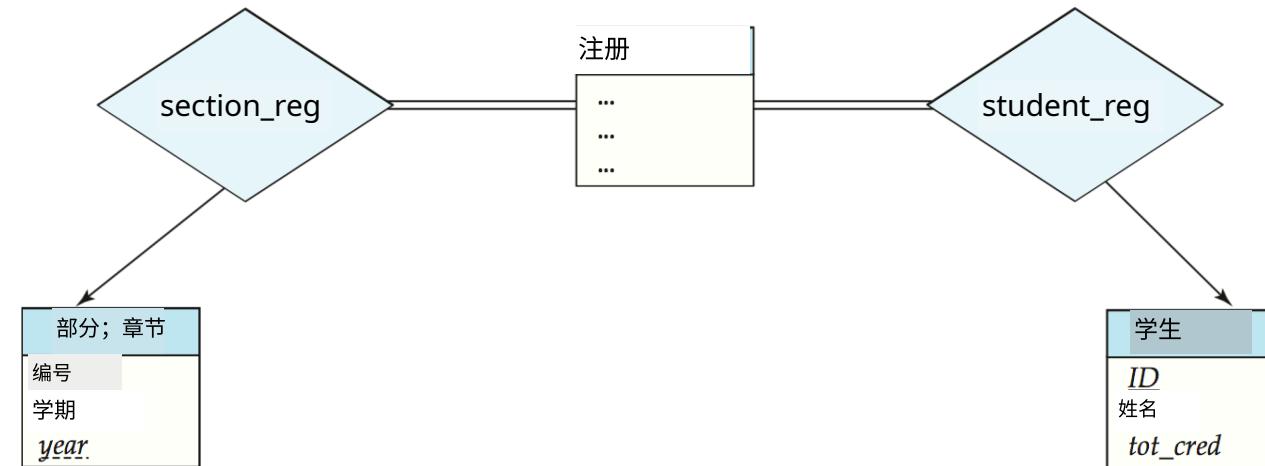


将电话作为一个实体使用可以获取有关电话号码的额外信息（包括多个电话号码）

设计问题

实体集与关系集的使用

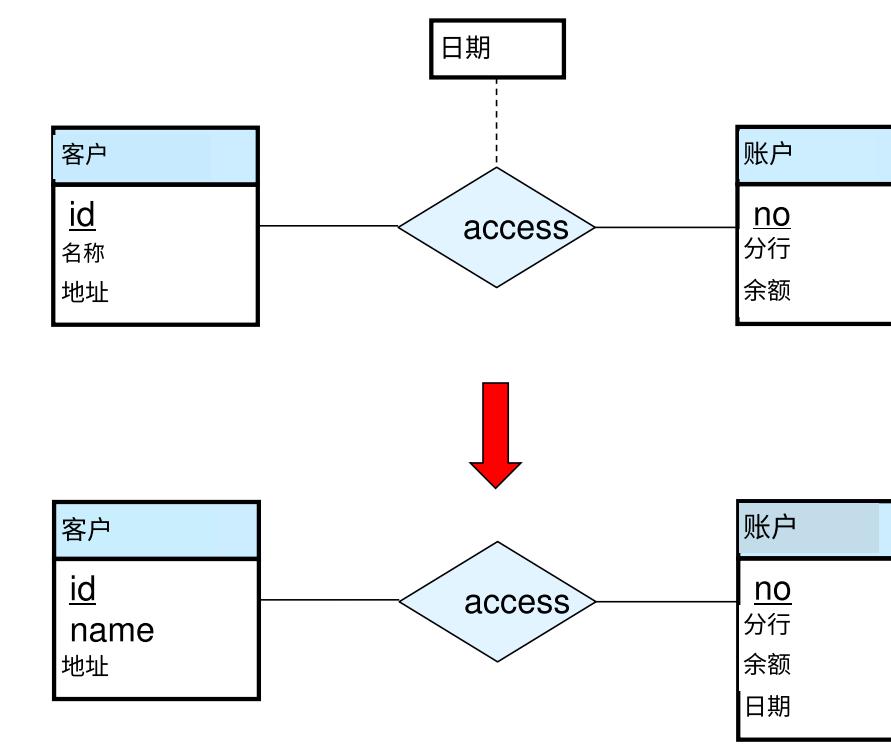
可能的指导原则是指定一个关系集来描述实体之间发生动作



设计问题

关系属性的放置

例如，将日期属性作为访问的属性或作为账户的属性



二元关系与非二元关系

二元与n元关系集

尽管可以用多个不同的二元关系集来替代任何非二元 (n元, 其中 $n > 2$) 关系集, 但n元关系集更清晰地表明多个实体参与了一个单一的关系。

一些看似非二元的关系可能用二元关系来表示会更好

例如, 将孩子与其父亲和母亲相关联的三元关系“父母”, 最好用两个二元关系“父亲”和“母亲”来替代

• 使用两个二元关系可以处理部分信息 (例如, 只知道母亲的情况)

但有些关系本质上是非二元的——例如: proj_guide

将非二元关系转换为二元形式

一般来说, 通过创建一个人工实体集, 任何非二元关系都可以用二元关系来表示。

用实体集 E 替换实体集 A, B 和 C 之间的 R , 以及三个关系集:

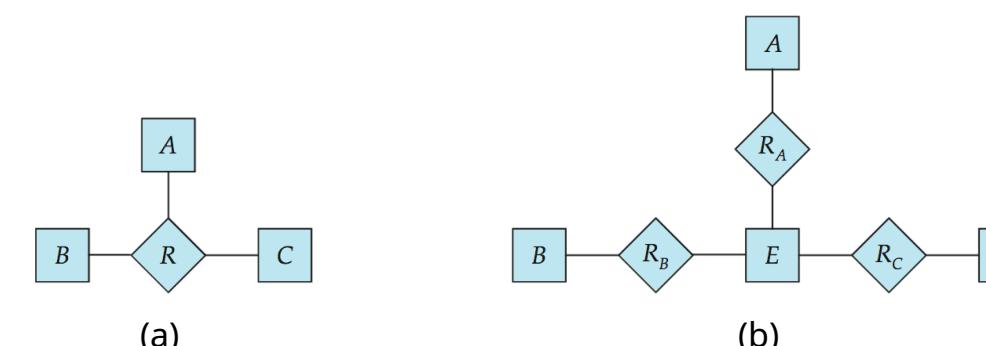
1. R_A , 关联 E 和 A
2. R_B , 关联 E 和 B
3. R_C , 关联 E 和 C

为 E 创建一个特殊的标识属性

将 R 的任何属性添加到 E 中

对于 R 中的每个关系 (a_i, b_i, c_i) , 创建

1. 在实体集 E 中创建一个新实体 e_i 。将 (e_i, a_i) 添加到 R_A
3. 将 (e_i, b_i) 添加到 R_B
4. 将 (e_i, c_i) 添加到 R_C



转换非二元关系 (续)

实体 - 关系 (E - R) 设计决策

还需要翻译约束条件

可能无法翻译所有约束条件

在转换后的模式中, 可能存在一些实例无法对应 R 的任何实例

• 练习: 为关系 R_A, R_B 和 R_C 添加约束条件, 以确保新创建的实体在每个实体集 A, B 和 C 中恰好对应一个实体

我们可以通过将 E 设为一个弱实体集 (稍后描述), 由三个关系集来标识, 从而避免创建标识属性

使用属性或实体集来表示对象。

现实世界中的概念是最好用实体集还是关系集来表达。

使用三元关系还是二元关系。

使用强实体集还是弱实体集。

扩展的实体 - 关系 (E - R)

特性: 特化/泛化

特化(特化)

自顶向下的设计过程; 我们指定实体集中与该集合中其他实体不同的子分组。

属性继承——较低层级的实体集继承与其关联的较高层级实体集的所有属性和关系参与。

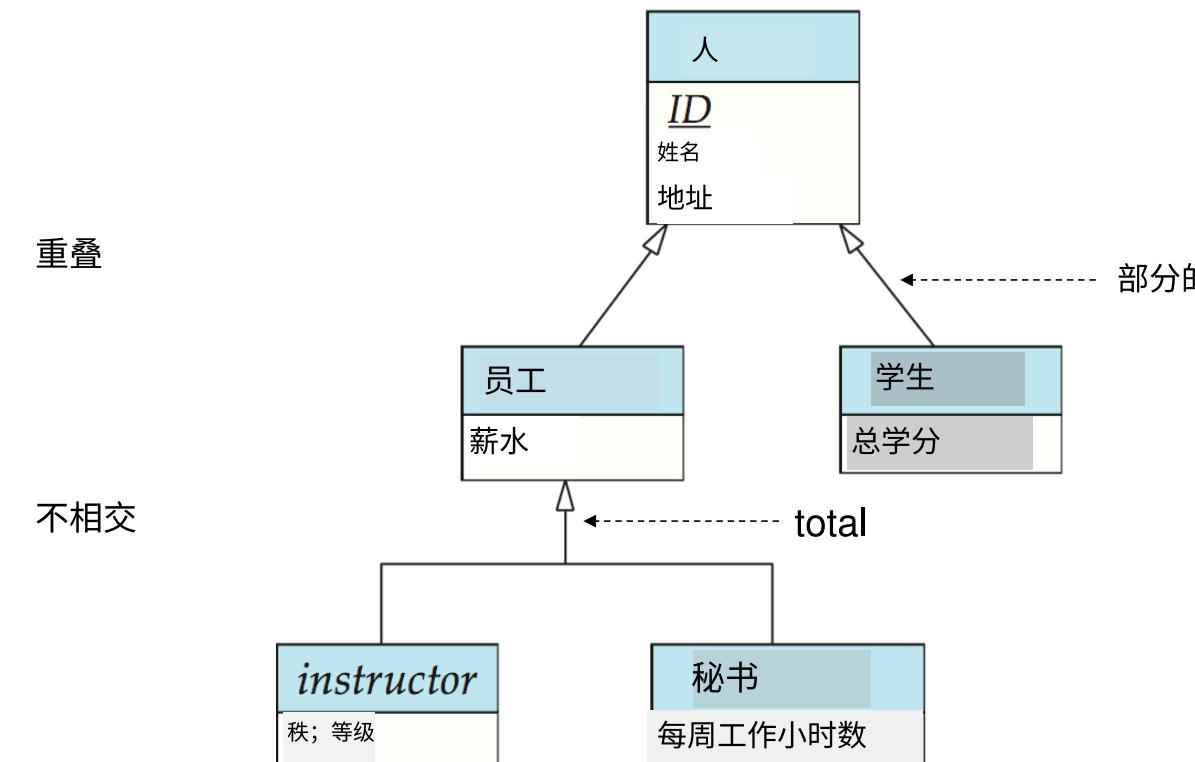
概化 (概化)

自下而上的设计过程——将多个具有相同特征的实体集合并为一个较高层级的实体集。

特化和概化是彼此的简单逆过程; 它们在实体 - 关系图中的表示方式相同。

术语“特化”和“概化”可互换使用。

Extended ER Features



关于实体是否可以在单个泛化中属于多个低级实体集的约束。

不相交(不相交)

- 一个实体只能属于一个低级实体集
- 在E-R图中，通过将多个低级实体集链接到同一个三角形来表示

重叠(重叠)

- 一个实体可以属于多个低级实体集

关于.....的设计约束

特化/泛化 (续)

完全性约束 (完全性约束) - 指定高层实体集中的实体是否必须属于泛化内的至少一个低层实体集。

全部：实体必须属于一个低层实体集 部分：实体不必属于一个低层实体集

通过模式表示特化

·方法1：

- 为高级实体创建一个模式
- 为每个低级实体集创建一个模式，包括高级实体集的主键和本地属性

模式	属性
人员	ID, 姓名, 街道, 城市
学生	ID, 总学分
员工	ID, 薪资

·缺点：获取员工信息需要访问两个关系，一个对应低级模式，另一个对应高级模式

将特化表示为模式

·方法2：

- 为每个实体集形成一个包含所有本地和继承属性的模式

模式	属性
人员	ID, 姓名, 街道, 城市
学生	ID, 姓名, 街道, 城市, 总学分
员工	ID, 姓名, 街道, 城市, 薪资

·缺点：对于既是学生又是员工的人，姓名、街道和城市可能会被重复存储

将特化表示为模式

方法3：

对所有超/子实体集使用单一模式，并使用类型属性来区分实体。

模式	属性
人员	ID、姓名、街道、城市、人员类型、总学分、薪资

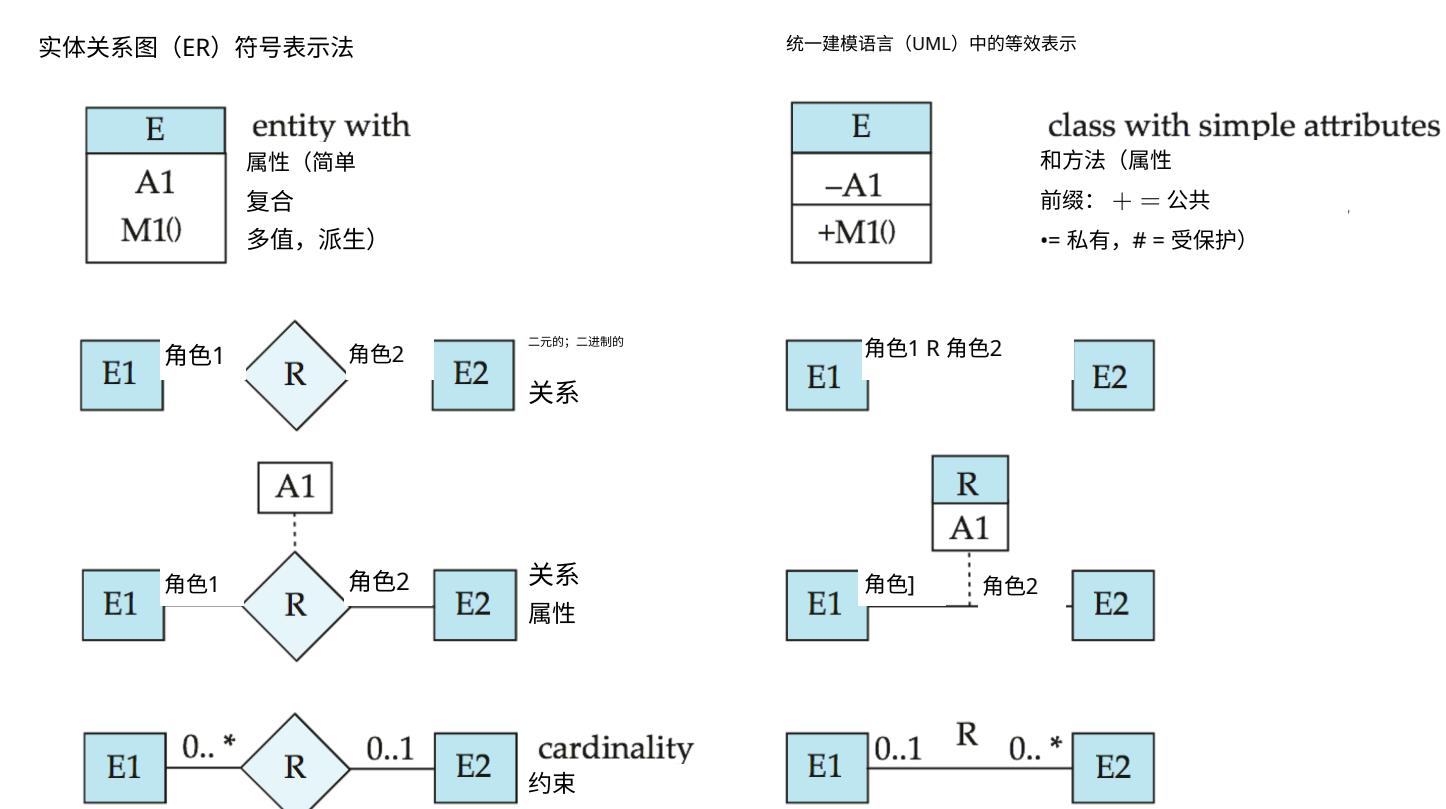
UML

UML：统一建模语言

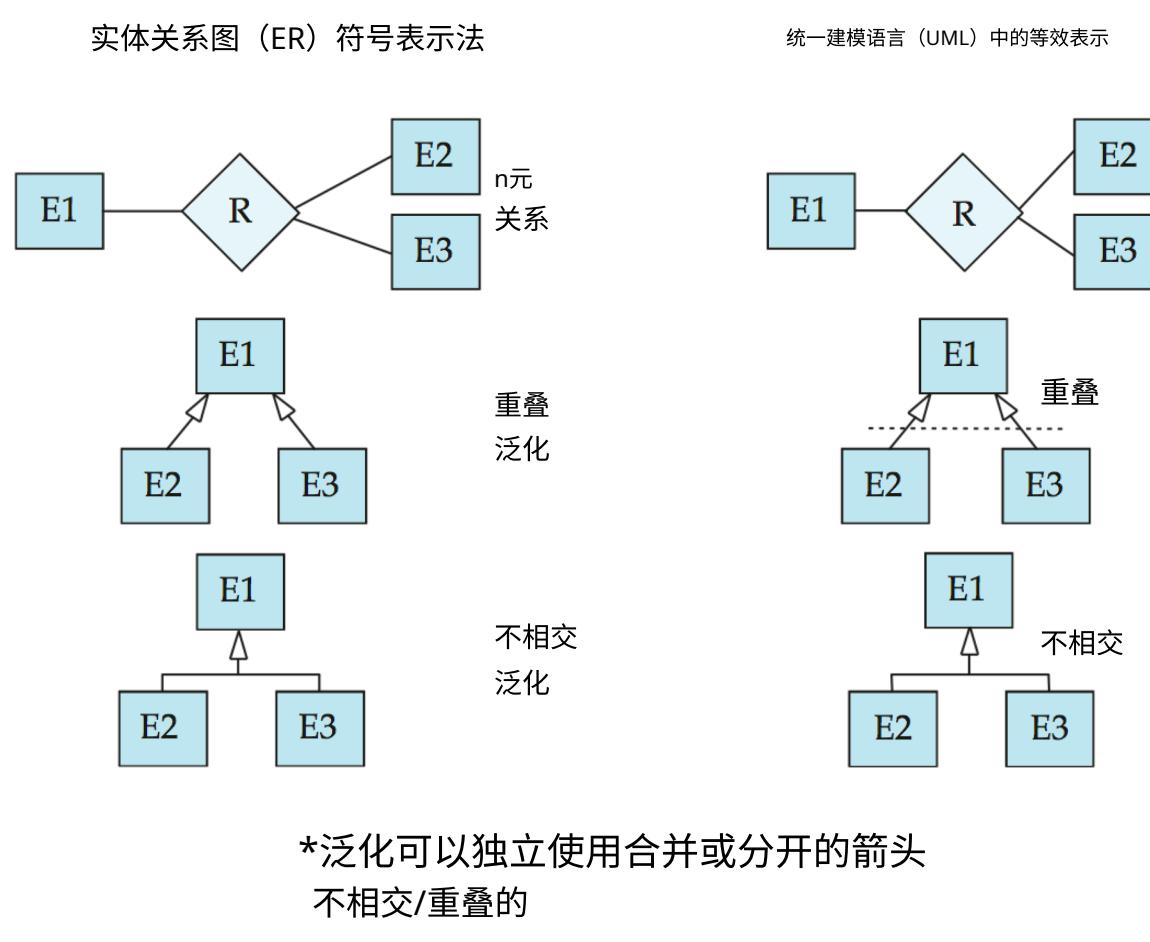
UML有许多组件，可对整个软件系统的不同方面进行图形化建模

UML类图对应于实体 - 关系图，但存在一些差异。

实体关系图 (ER) 与统一建模语言 (UML) 类图对比



*注意基数约束描述中位置的反转



在UML中，二元关系集通过绘制一条连接实体集的线来表示。关系集名称写在该线旁边。
实体集在关系集中所扮演的角色也可以通过在连接实体集的线上写上角色名称来指定。
或者，可以将关系集名称与关系集的属性一起写在一个框中，然后使用虚线将该框连接到描绘关系集的线上。

End of Chapter 6