

数据库系统原理

陈岭

浙江大学计算机学院

8

数据库设计和E-R模型

- 实体集
- 联系集
- 设计问题
- 映射约束
- 键
- E-R图
- 扩展的E-R特性
- E-R模式设计
- E-R模式到表的转换

- 数据库可被建模为：
 - 实体集合
 - 实体间联系
- **实体**是客观存在的对象并且与其他对象可区分
 - 例如： 特定的人，公司，事件，植物
- **实体**具有**属性**
 - 例如： 人具有姓名和地址
- **实体集**是相同类型的实体的集合，他们具有相同的性质
 - 例如： 所有人的集合，所有公司的集合

实体集

□ 例，实体集 *instructor*

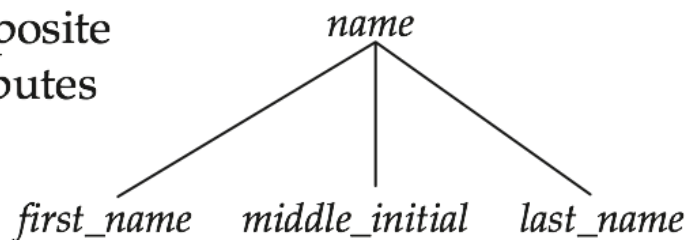
属性名称 →	ID	name	dept_name	salary
	10101	Srinivasan	Comp. Sci.	65000
	12121	Wu	Finance	90000
	15151	Mozart	Music	40000
属性值	<u>22222</u>	Einstein	Physics	95000
	32343	El Said	History	60000
	33456	Gold	Physics	87000
	45565	Katz	Comp. Sci.	75000
	58583	Califieri	History	62000
一个实体	<u>76543</u>	Singh	Finance	80000
	76766	Crick	Biology	72000
	83821	Brandt	Comp. Sci.	92000
	98345	Kim	Elec. Eng.	80000

instructor

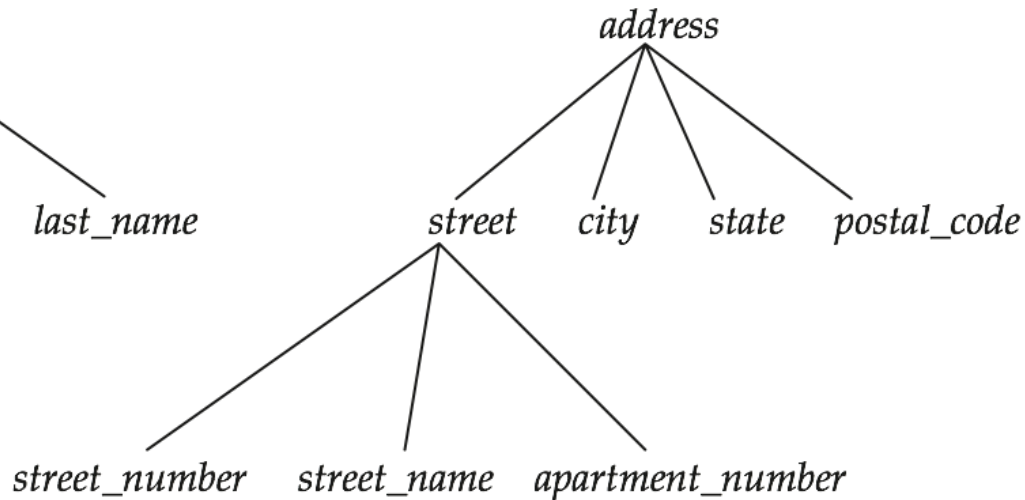
- 实体用一个属性集合来表示，即实体集中所有成员都具有的描述性特性
- 域：属性允许取值的集合
- 属性种类：
 - 简单属性与复合属性
 - 单值属性与多值属性
 - 例，多值属性： *phone-numbers*
 - 派生属性
 - 可由其他属性计算得到
 - 例，给定出生日期可计算出年龄
 - 基属性或存储属性

□ 复合属性

复合属性
composite
attributes

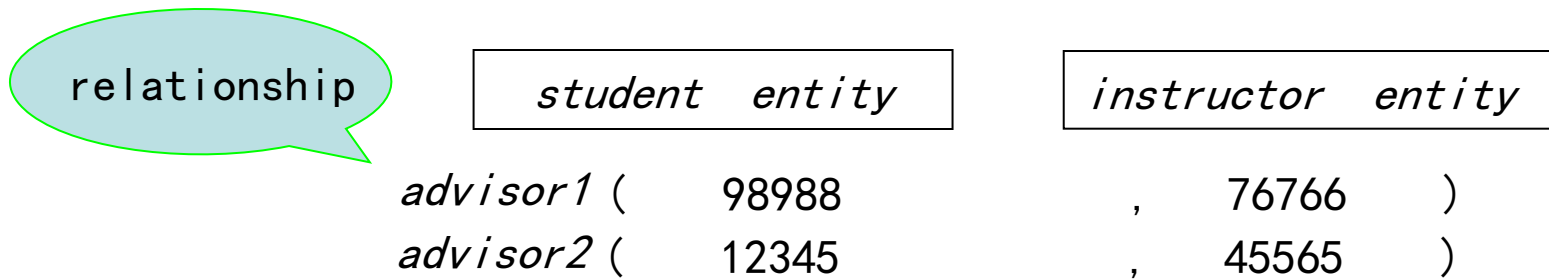


分量属性
component
attributes



联系集

□ 联系是指多个实体之间的相互关联



□ 联系集是相同类型联系的集合

- 一个联系集包含多个同类联系（或联系实例，relationship instance）
- 一个联系集表示二个或多个实体集之间的关联

□ 正规地说，联系集是 $n \geq 2$ 个实体集上的数学关系，每个实体取自一实体集

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

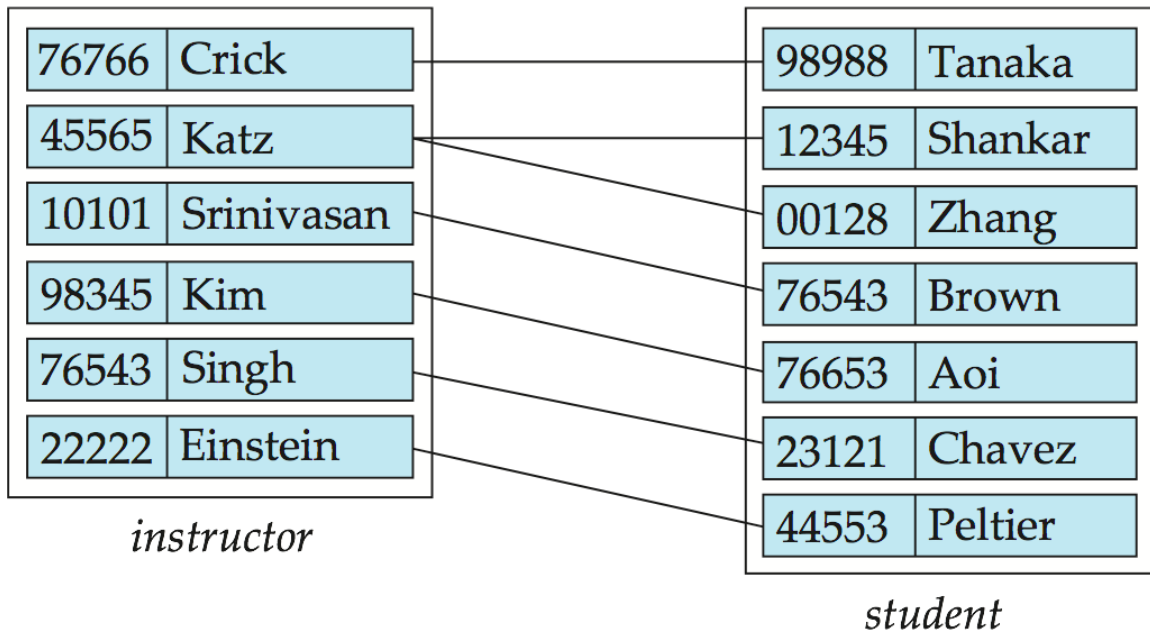
■ 其中 (e_1, e_2, \dots, e_n) 是一个联系， E_i 为实体集

■ 例， $(98988, 76766) \in \text{advisor}$,

其中， $98988 \in \text{student}$, $76766 \in \text{instructor}$

联系集

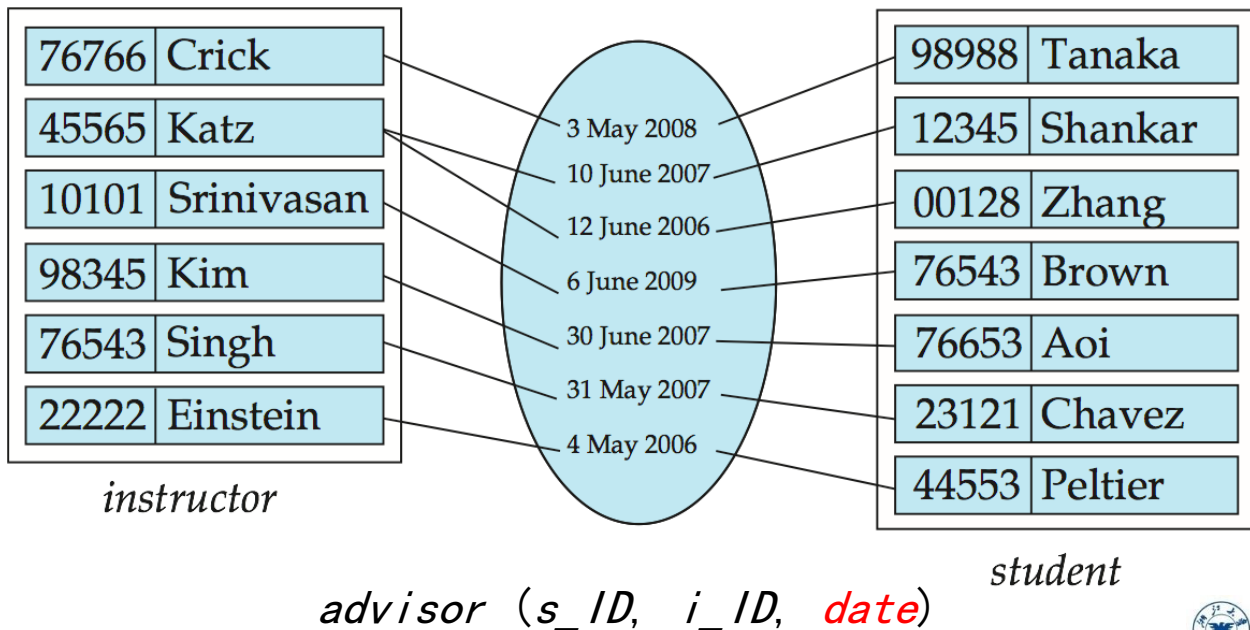
联系集 *advisor*



advisor (*s_ID*, *i_ID*)

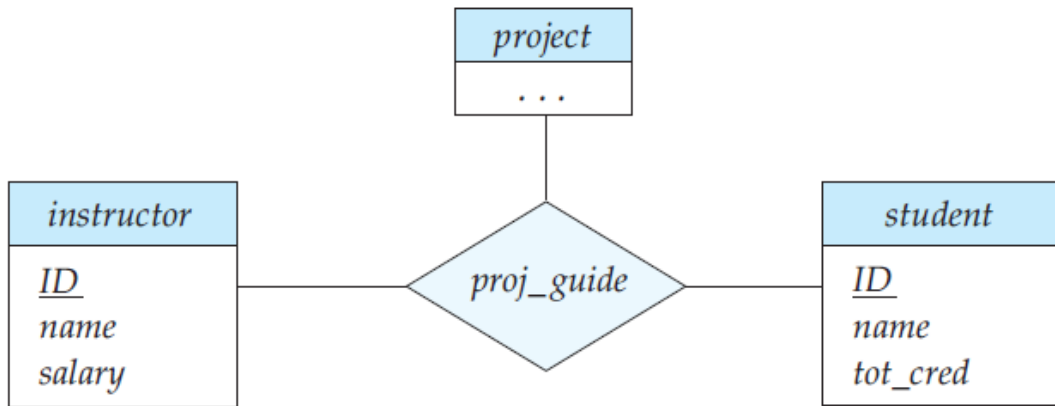
联系集

- 联系集也可具有属性
- 例如， 实体集 *instructor* 和 *student* 之间的 *advisor* 联系集可具有属性 *date*



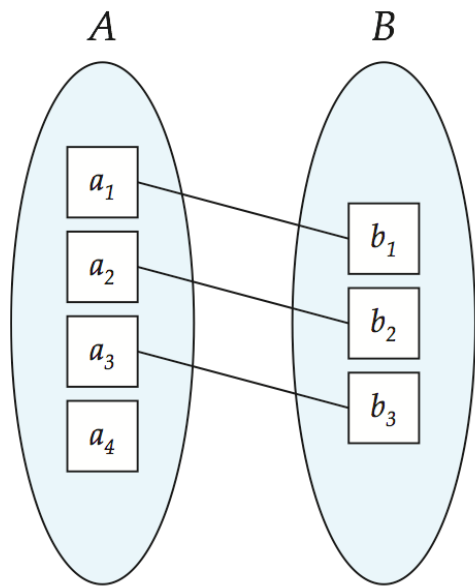
联系集的度

- 参加联系的实体集的个数
- 涉及两个实体集的联系集称为二元的（二元联系）
- 联系集可以涉及多于两个的实体集
 - 例，假设一个student在每个项目上最多只能有一位导师，如下图，包含三个实体集 *instructor*、*student*和*project*（三元联系）
 - 多于两个实体集之间的联系较少见，数据库系统中的联系集一般多为二元的

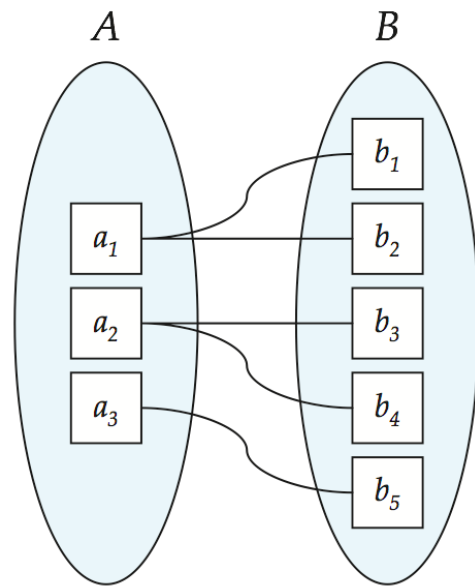


- 表达可与一个实体通过联系集进行关联的其他实体的个数
- 描述二元联系集最有用
- 二元联系集的映射基数有以下几种情况：
 - 一对一，如：就任总统（总统，国家）
 - 一对多，如：分班情况（班级，学生）
 - 多对一，如：就医（病人，医生）
 - 多对多，如：选课（学生，课程）

映射基数



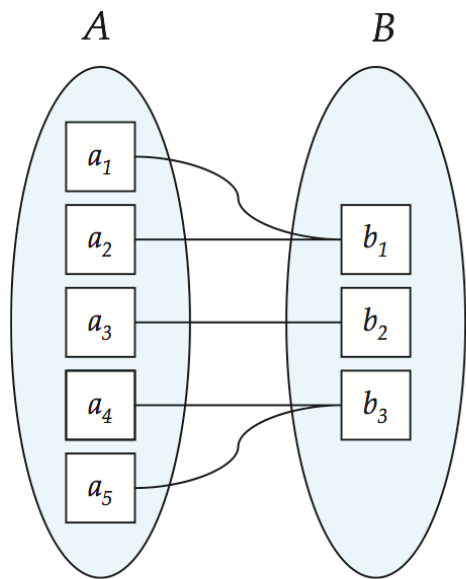
(a)
一对一



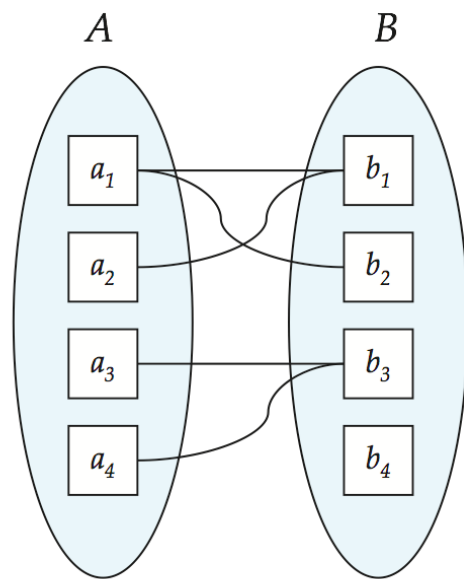
(b)
一对多

注意：A和B中的某些元素可能未被映射到另一集合中的任何元素

映射基数



(a)
多对一

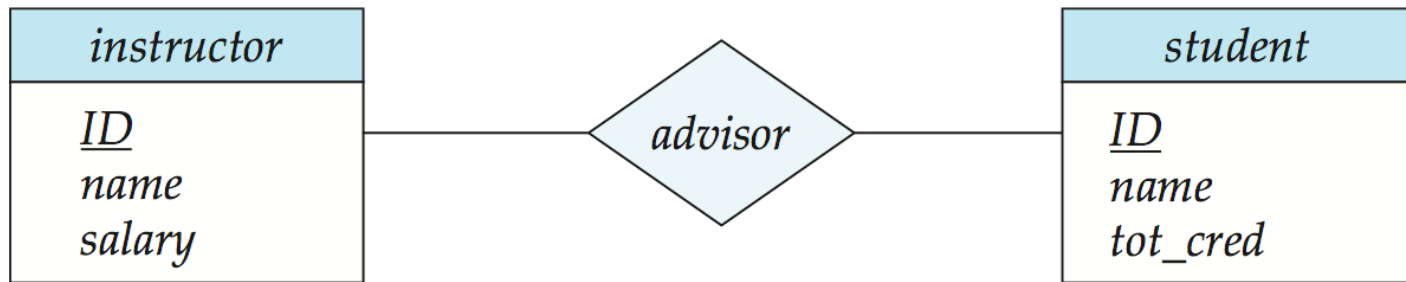


(b)
多对多

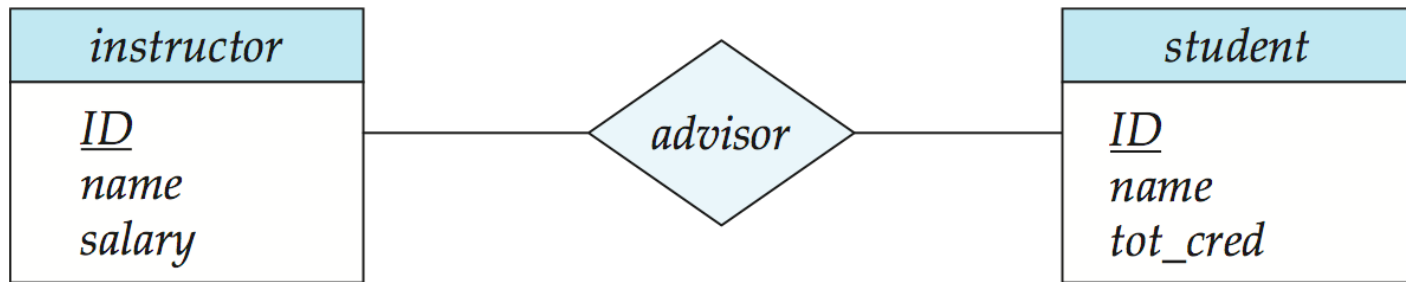
注意：A和B中的某些元素可能未被映射到另一集合中的任何元素

- ❑ 实体集的**超码**是能够唯一标识每个实体的一个或多个属性
- ❑ **候选码**是实体集的最小超码
- ❑ **候选码**可能存在多个，我们只会选择一个候选码作为**主码**或**主键**
- ❑ 例， *instructor* (*ID*, *name*, *dept_name*, *salary*)
 - 候选码： *ID*
 - 超码： { *ID* }, { *ID*, *name* }, { *ID*, ... }

- 参与一个联系集的各实体集的**码**的组合，构成该联系集的**超码**
 - (s_ID, i_ID) 是 *advisor* 的**超码**
 - 注意：这意味着一对实体在一个联系集上最多有一个联系
- 联系集的**候选码**依赖于联系集的**映射基数** (1:1, 1:n, m:n)
- 在选择主键时，如果有多个候选码，需要考虑关系集合的语义
 - 例，作为码的属性不能为空，值不应常变

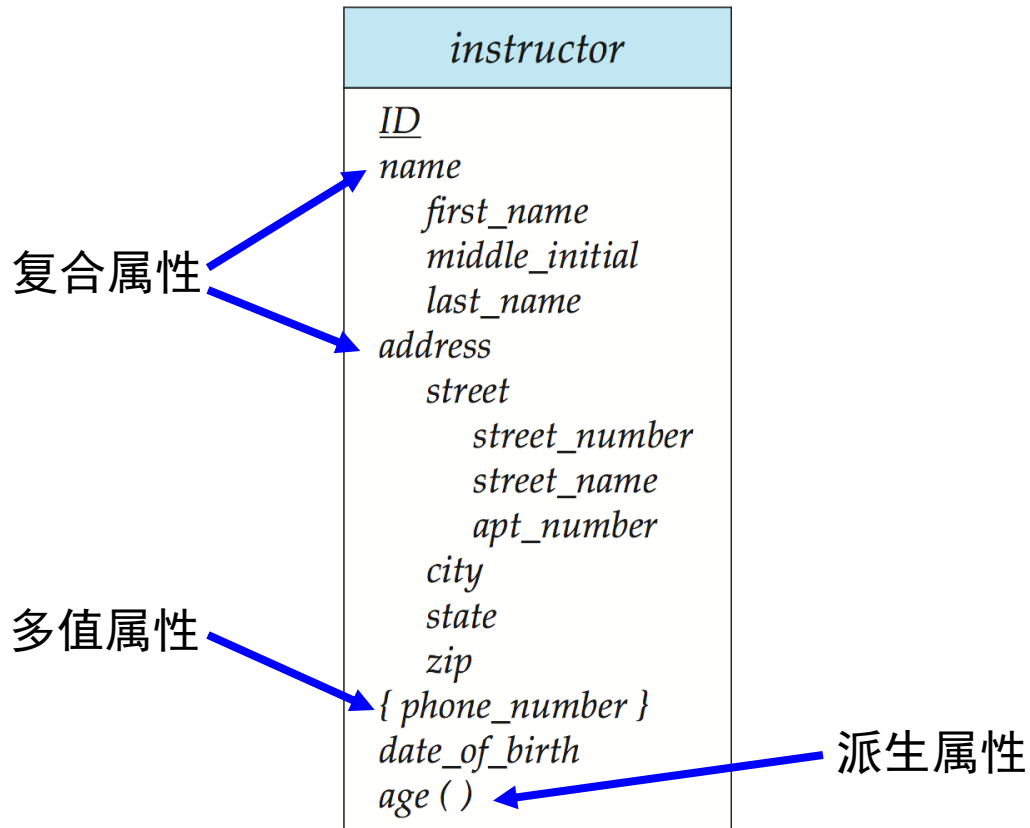


- ❑ 分成两部分的矩形代表实体集。有阴影的第一部分包含实体集的名字，第二部分包含实体集中所有属性的名字
- ❑ 菱形代表联系集
- ❑ 未分割的矩形代表联系集的属性。构成主码的属性以下划线标明
- ❑ 线段将实体集连接到联系集

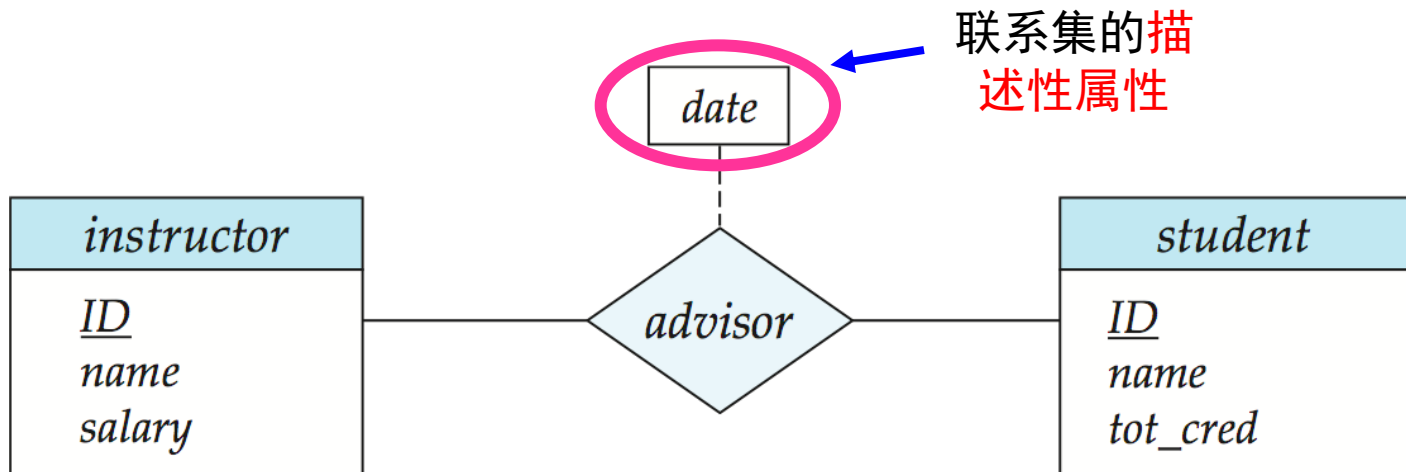


- ❑ 虚线将联系集属性连接到联系集
- ❑ 双线显示实体在联系集中的参与度
- ❑ 双菱形代表连接到弱实体集的标志性联系集

包含复合、多值和派生属性的E-R图



□ 带有属性的联系集

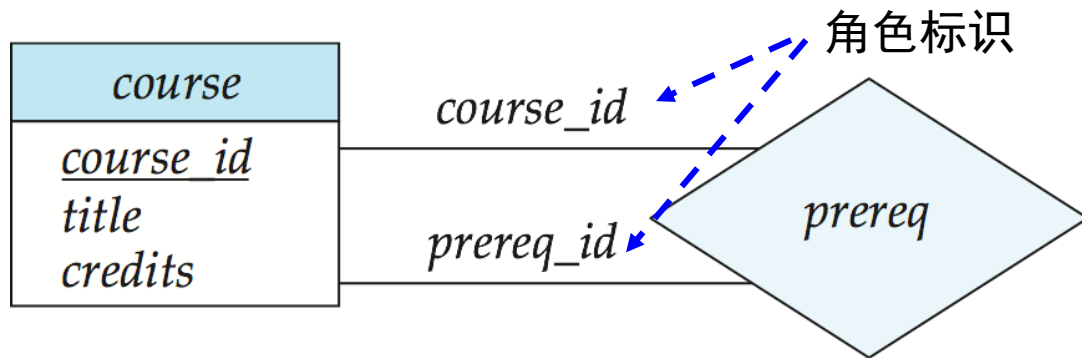


□ 参加联系的实体集不必是互不相同的

■ 例，自环联系集 (recursive relationship set)

□ 角色：实体在联系集中的作用

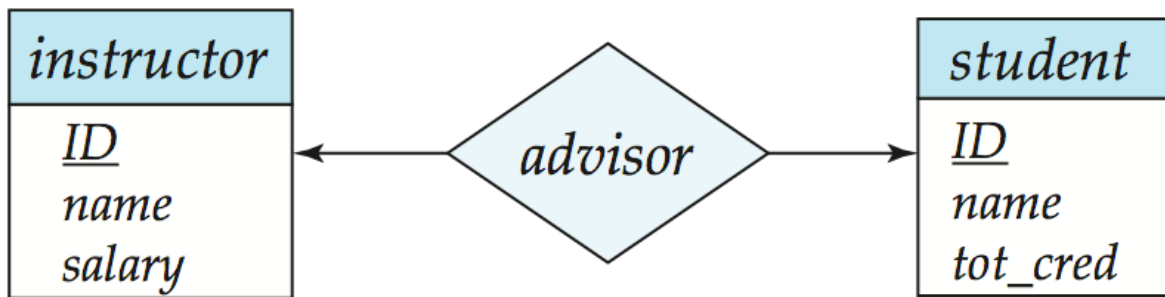
■ 例，下图给出了 *course* 实体集和 *prereq* 联系集之间的角色标识 *course_id* 和 *prereq_id*



□ 角色标记是可选的，用于明确联系的语义

基数约束

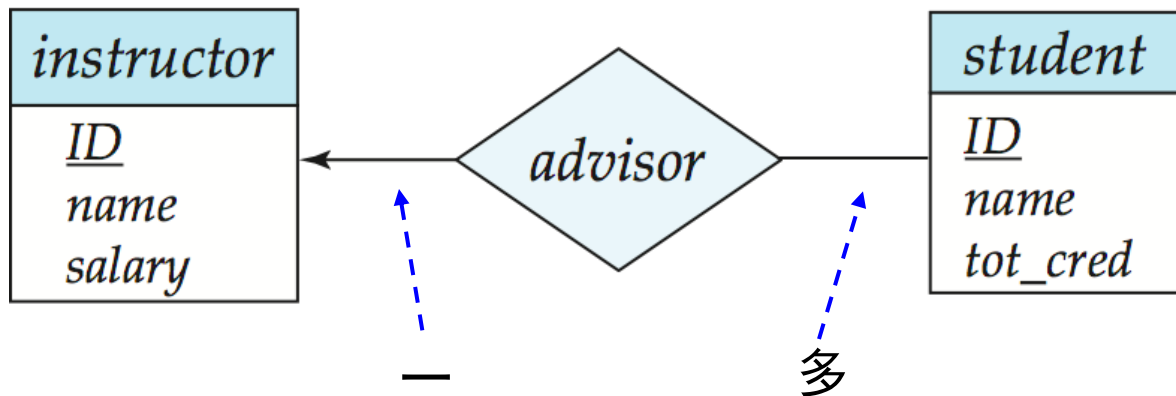
- 在联系集与实体集之间用有向直线(\rightarrow)表示“一”，无向直线($—$)表示“多”
- 一对一联系：
 - 实体集instructor和student之间的联系集可以是一对一，表示一名教师可以指导至多一名学生，并且一名学生可以有至多一位导师
 - 注：一个联系集的类型取决于被表达对象的语义约束及设计者的意图



基数约束

□ 一对多联系：

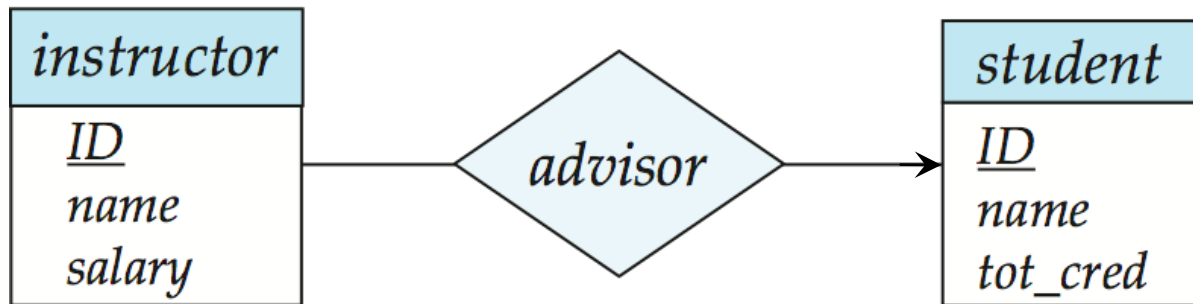
- 实体集 *instructor* 和 *student* 之间的联系集可以是一对多，表示一名教师可以指导多名学生，但一名学生可以有至多一位导师



基数约束

□ 多对一联系：

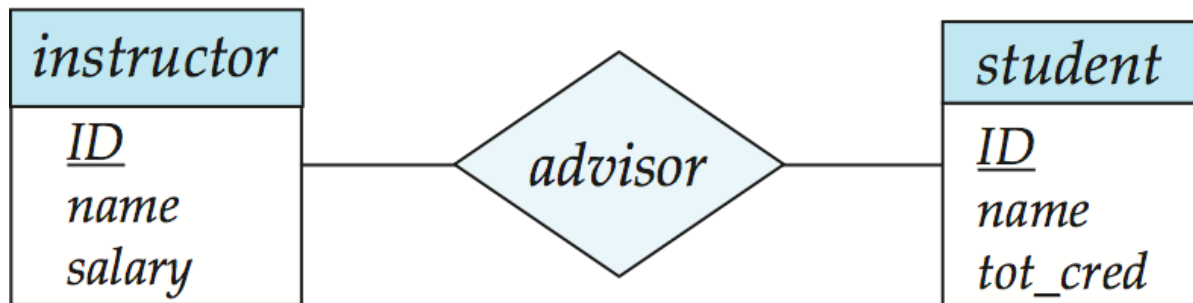
- 实体集 *instructor* 和 *student* 之间的联系集可以是多对一，表示一名教师可以指导至多一名学生，但一名学生可以有 multiple 位导师



基数约束

□ 多对多联系：

- 实体集instructor和student之间的联系集可以是多对多，表示一名教师可以指导多名学生，并且一名学生可以有多位导师

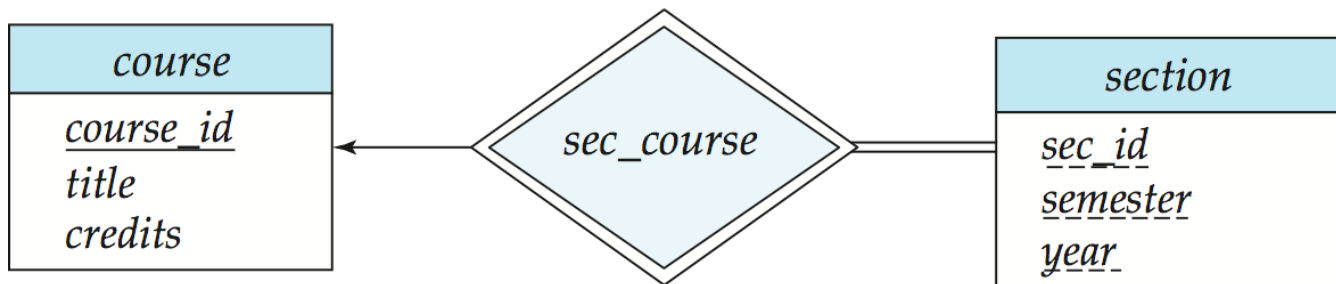


参与约束

□ 实体集参加联系集的方式

- **全参与** (用双线表示)：实体集中的每个实体都至少参加联系集中的一个联系
- **部分参与**：某些实体可能未参加联系集中的任何联系

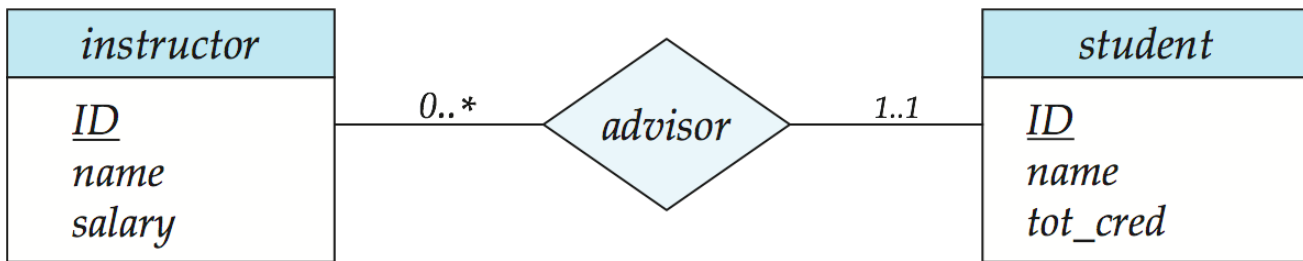
- 映射基数约束 (Mapping cardinality constraints)，限定了一个实体与发生关联的另一端实体可能关联的数目上限
- 全参与和部分参与约束，则反映了一个实体参与关联的数目下限：0次，还是至少1次



关系约束的另一种表示法

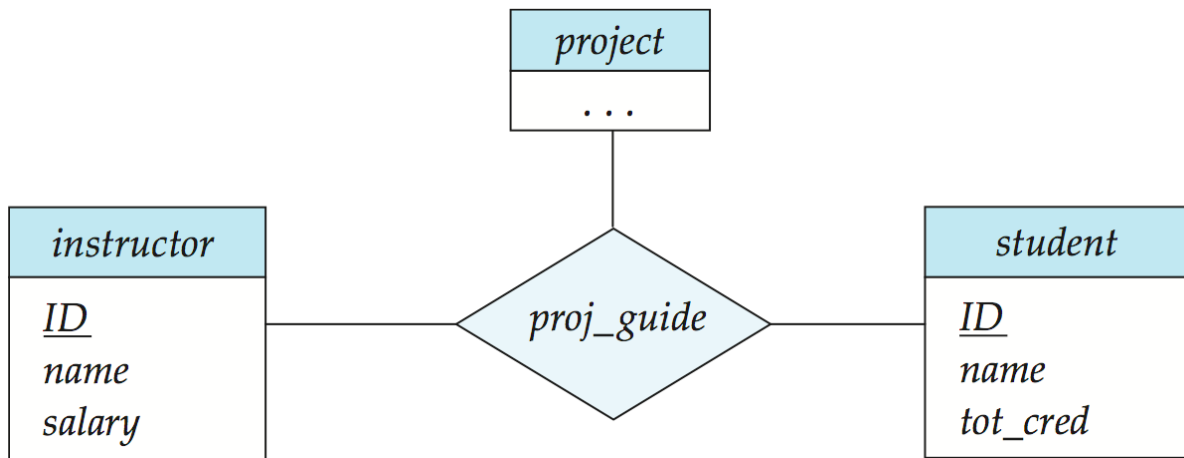
□ 用关系约束的另一种表示法来标识基数约束和参与约束

■ 例，每个学生有且仅有一个导师，教师可以有零个或多个学生



具有三元联系的E-R图

□ 例，一个教师可以指导多个学生，并可以参与到多个项目



二元与非二元联系

□ 某些看起来似乎是非二元的联系可用二元联系更好地表示

- 例，三元联系 `parents` 将孩子与其父亲和母亲相关联，可以更好地用两个二元联系 `father` 和 `mother` 代替

`parents(he, she, child) =>`
`father (he, child)`
`mother(she, child)`

- 使用二元联系可以表达部分信息（如，只知道母亲）

□ 但有些联系用非二元更自然

- 例，`proj_guide(instructor, project, student)`

非二元联系转换成二元联系

□ 任何非二元联系都可以用二元联系表示，方法是人为创建一个实体集

■ 将实体集 A, B, C 之间的联系 R 用实体集 E 和以下三个联系集代替：

- R_A ，将 E 与 A 关联
- R_B ，将 E 与 B 关联
- R_C ，将 E 与 C 关联

■ 为 E 创建一个特殊的标识属性

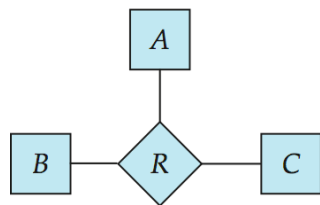
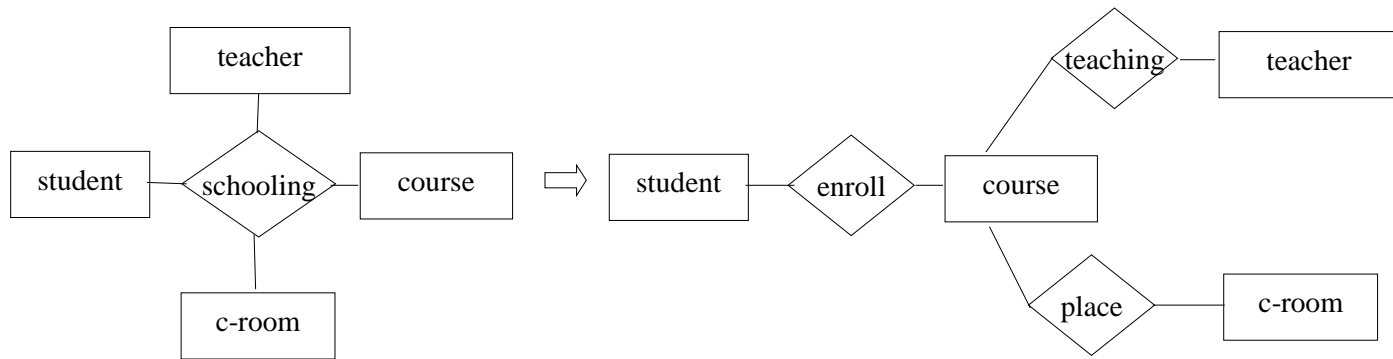
■ 将 R 的所有属性加给 E

■ 对 R 中每一个联系 (a_i, b_i, c_i)

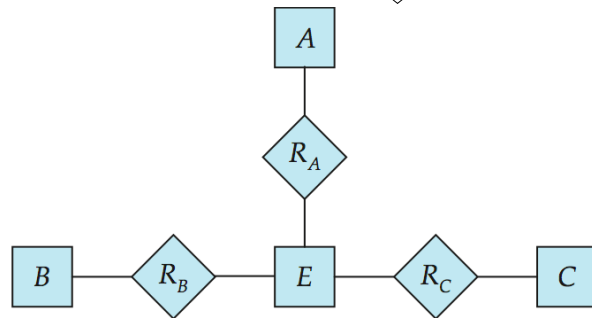
- 创建实体集 E 中的一个新实体 e_i
- 将 (e_i, a_i) 加入 R_A
- 将 (e_i, b_i) 加入 R_B
- 将 (e_i, c_i) 加入 R_C

非二元联系转换成二元联系

□ 例，将如下非二元联系 *schooling* 转换成二元联系



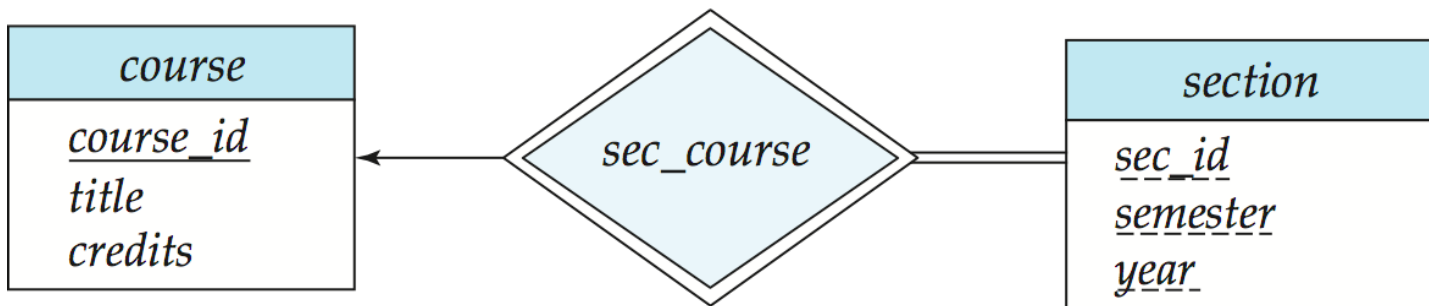
(a)



(b)

弱实体集

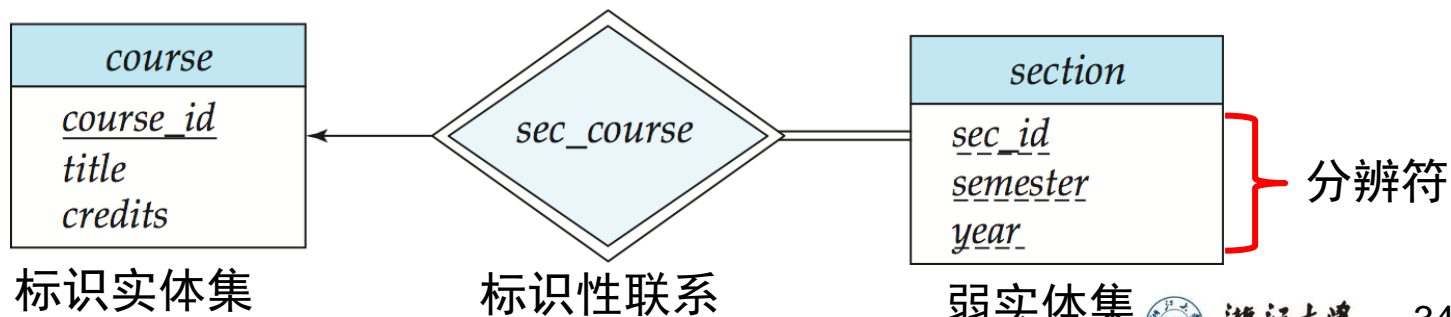
- 不具有主键的实体集称为弱实体集
- 例，考虑一个 *section* 实体，它由课程编号、学期、学年以及开课编号唯一标识。显然，开课实体和课程实体相关联。假定我们在实体集 *section* 和 *course* 之间创建了一个联系集 *sec_course*。若，实体集 *section* 为：
section(sec_id, semester, year)，则其为弱实体集



- 弱实体集的存在依赖于它的**标识实体集**（或属主实体集）的存在
 - 例, *course(course_id, title, credits)*
- **标识性联系**: 将弱实体集与其标识实体集相联的联系
 - 标识性联系是从弱实体集到标识实体集多对一的, 并且弱实体集在联系中的参与是全部的
 - 例, *sec_course*

- 弱实体集的分辨符（或称部分码）是指在一个弱实体集内区分所有实体的属性集合
 - 例，弱实体集 *section* 的分辨符由属性 *sec_id*, *year* 以及 *semester* 组成
- 弱实体集的主码由它所依赖的强实体集的主码加上它的分辨符组成
 - *section* 的主码：

$\{ \textit{course_id}, \textit{sec_id}, \textit{semester}, \textit{year} \}$



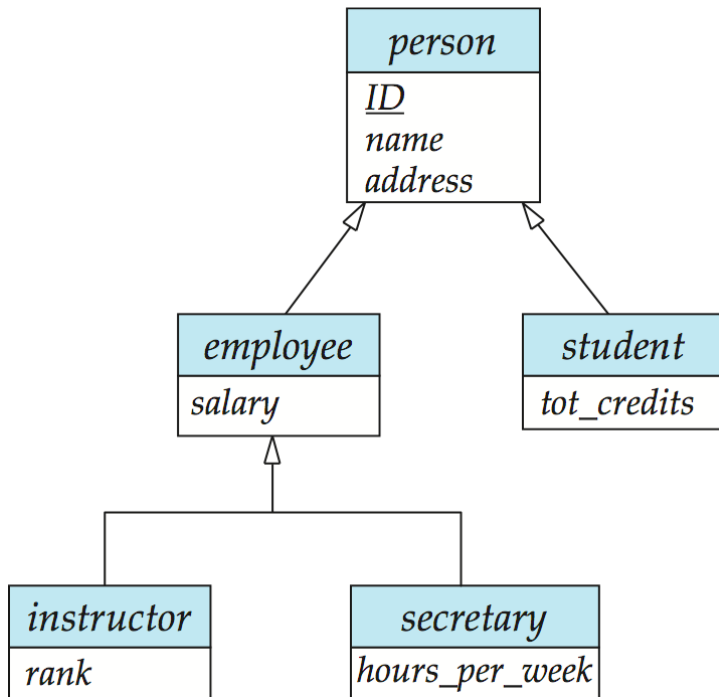
- 注意：强实体集的主码并不显式地存于弱实体集中，而是隐含地通过标识性联系起作用
- 如果 *course_id* 显式存在，*section* 就成了强实体，则 *section* 与 *course* 之间的联系变得冗余。因为 *section* 与 *course* 共有的属性 *course_id* 已定义了一个隐含的联系

□ 特化

- 自顶向下设计过程中，确定实体集中的一个具有特殊性质的子集
- 这些子集称为低层实体集，它们具有特殊的属性或者参加特殊的联系
- **属性继承**：低层实体集继承它连接的高层实体集的所有属性及参加的联系

扩展的E-R特性

- 特化用从特化实体指向另一方实体的空心箭头来表示。这种关系为ISA关系，代表“is a”（“是一个”）。例如，一个教师“是一个”雇员



□ 概化

- 自底向上设计过程中，将若干共享相同特性的实体集组合成一个高层实体集
- 特化与泛化简单互逆：它们在E-R图中以相同方式表示

对特化/概化的设计约束

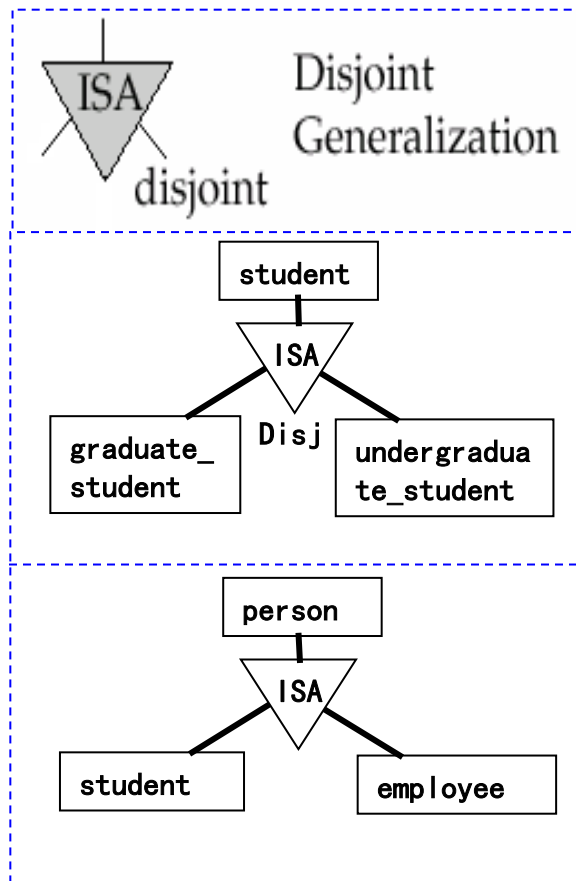
□ 关于哪些实体可以是给定低层实体集的成员 的约束

■ 条件定义的

- 只有满足 *student_type* = “研究生” 的实体才允许属于 *graduate_student* 实体

■ 用户定义的

- 大学雇员属于不同的工作组



对特化/概化的设计约束

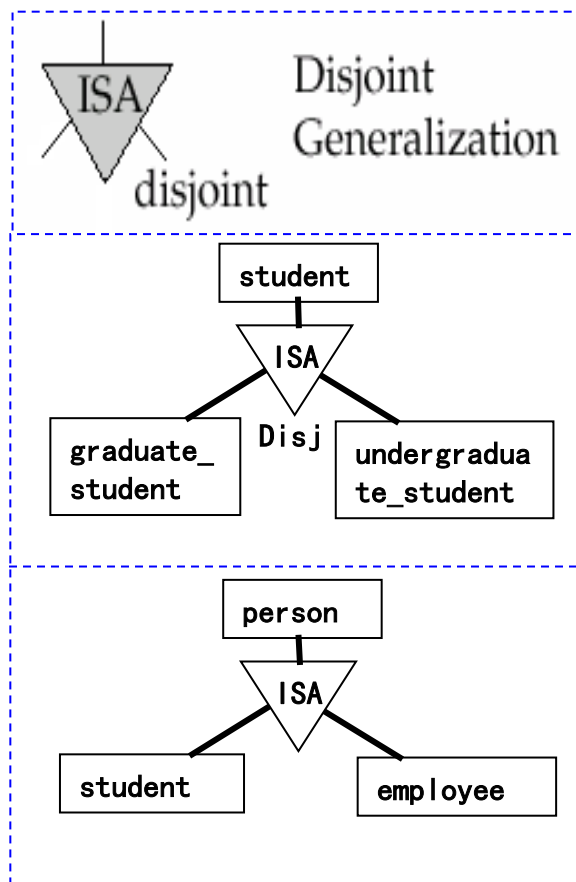
□ 关于实体在单个概化中是否可以属于多于一个低层实体集的约束

■ 不相交

- 一个实体只能属于一个低层实体集
- 在E-R图中ISA三角形旁边加注disjoint

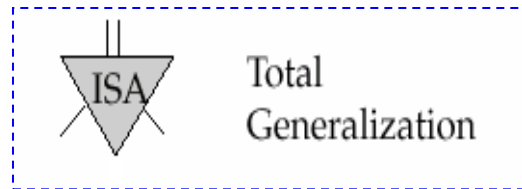
■ 重叠

- 一个实体可以属于多个低层实体集



对特化/概化的设计约束

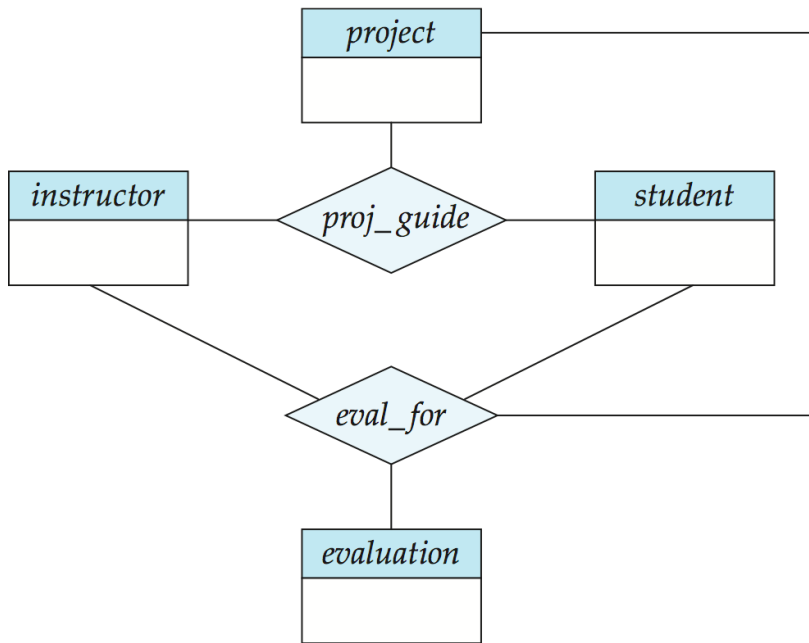
- 完备性约束：说明高层实体集中的实体是否必须至少属于一个低层实体集
 - 全部概化或特化：每个高层实体必须属于一个低层实体集
 - 部分概化或特化：允许一些高层实体不属于任何低层实体集（默认的）



扩展的E-R特性

□ 聚集

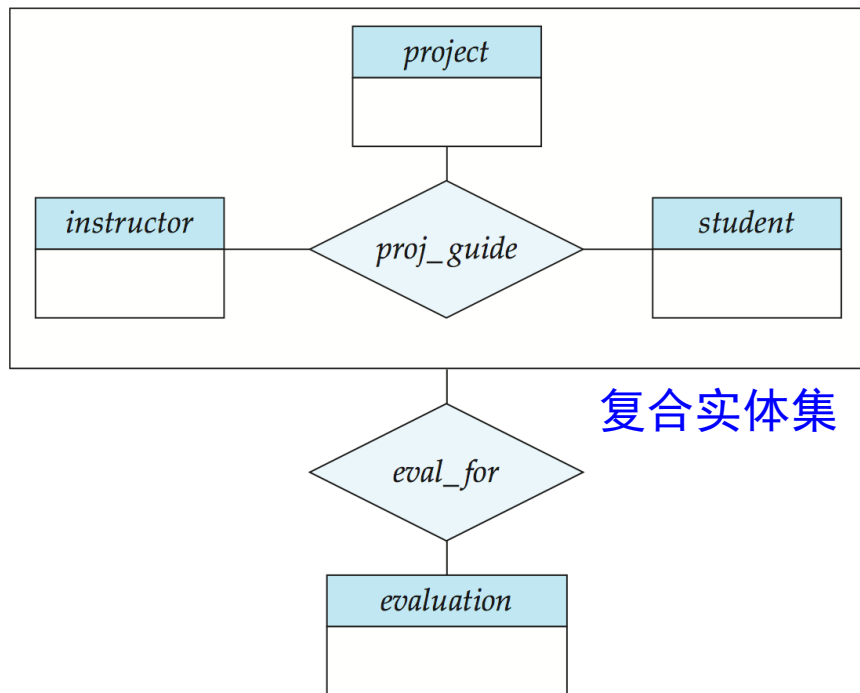
- 考虑三元联系 *proj_guide*
- 现在假设每位在项目上指导学生的教师需要记录月评估报告



- ❑ 联系集 *eval_for* 和 *proj_guide* 表达了重叠信息
 - 每个 *eval_for* 联系对应一个 *proj_guide* 联系
 - 然而, 某些 *proj_guide* 联系可能不对应任何 *eval_for* 联系, 因此我们不能丢掉 *proj_guide* 联系
- ❑ 通过聚集消除这种冗余
 - 将联系视为一个抽象实体
 - 从而允许联系之间的联系
 - 联系抽象为新实体

□ 在没有引入冗余的情况下，下图表达了：

- 一个学生在某个项目上由某个导师指导
- 一个学生，导师，项目的组合可能有一相关的评估

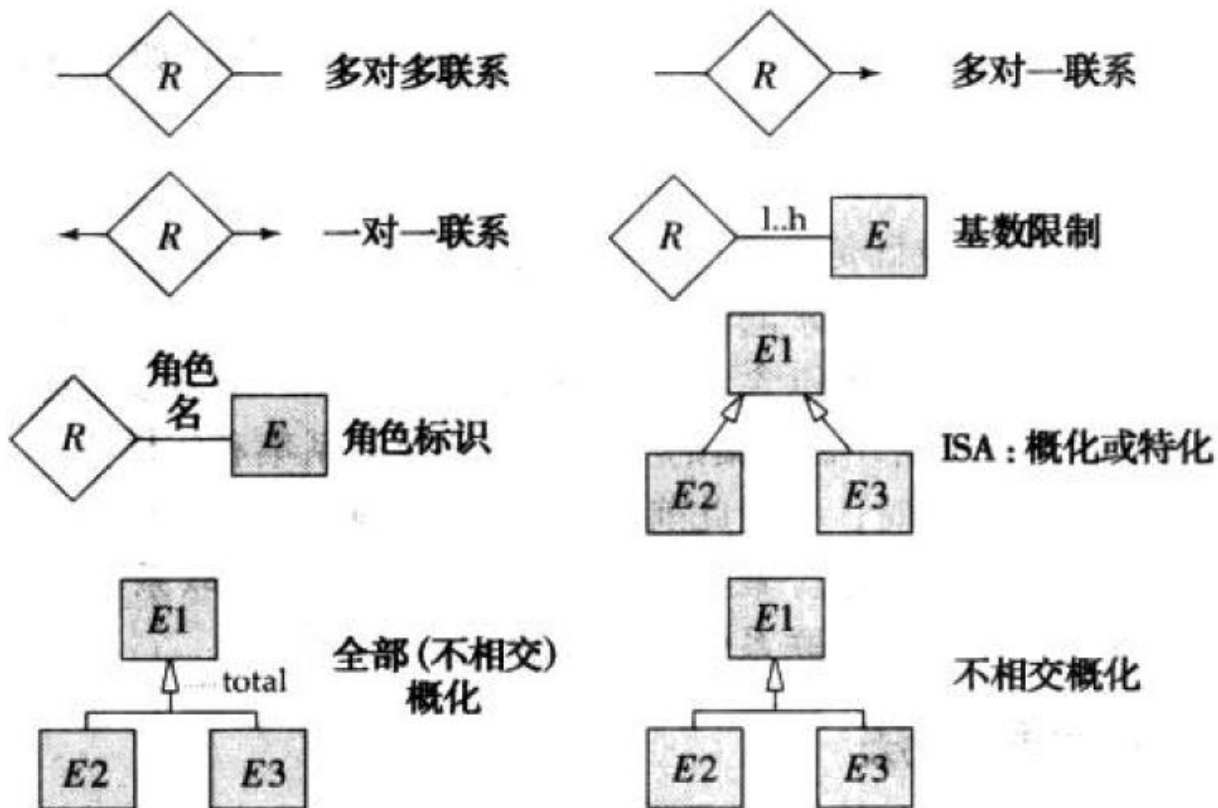


复合实体集

E-R图表示法中使用的符号小结

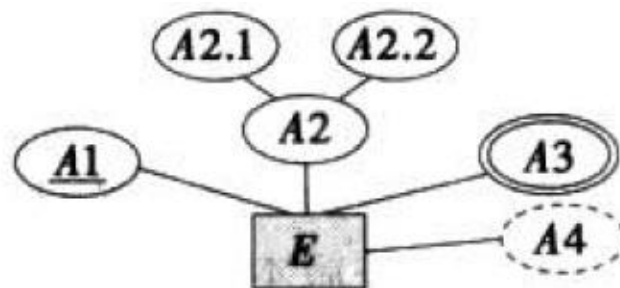


E-R图表示法中使用的符号小结



其他可选择的E-R图表示法

实体集E包含
简单属性A1、
复合属性A2、
多值属性A3、
派性属性A4、
以及主码A1



弱实体集



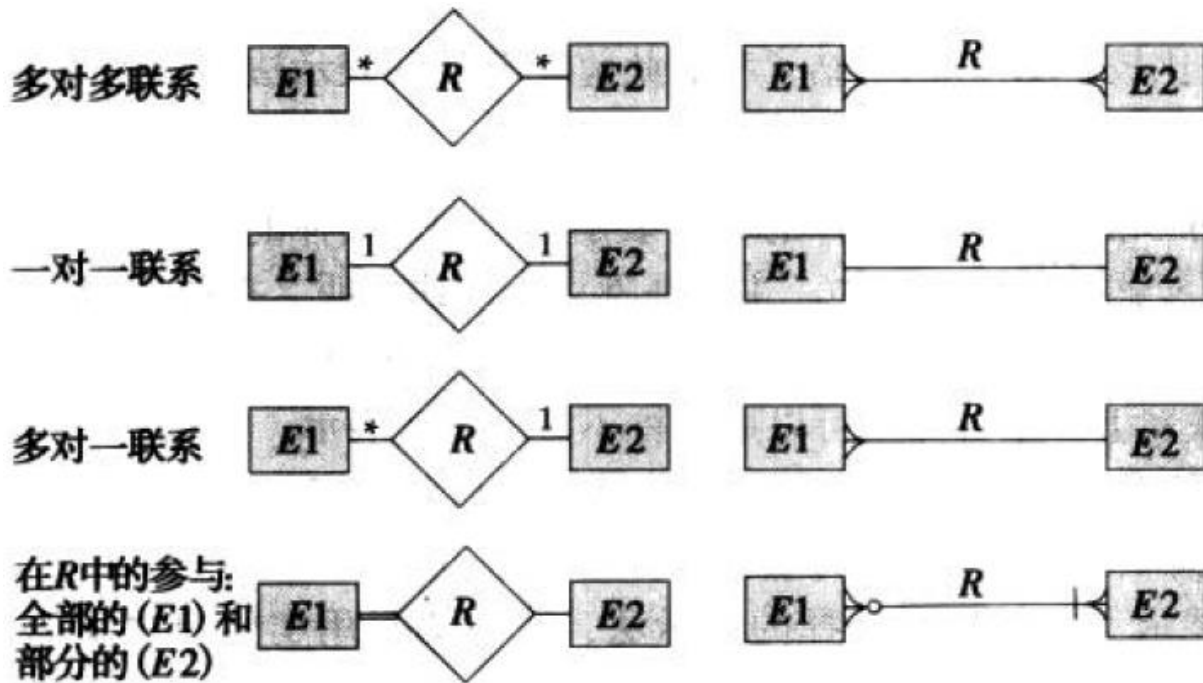
概化



全部概化



其他可选的E-R图表示法



设计数据库的E-R模式

□ 需求分析

- 需要什么样的数据、应用程序和业务

□ 概念数据库设计 ★

- 使用E-R模型或类似的高层次数据模型，描述数据

□ 逻辑数据库设计

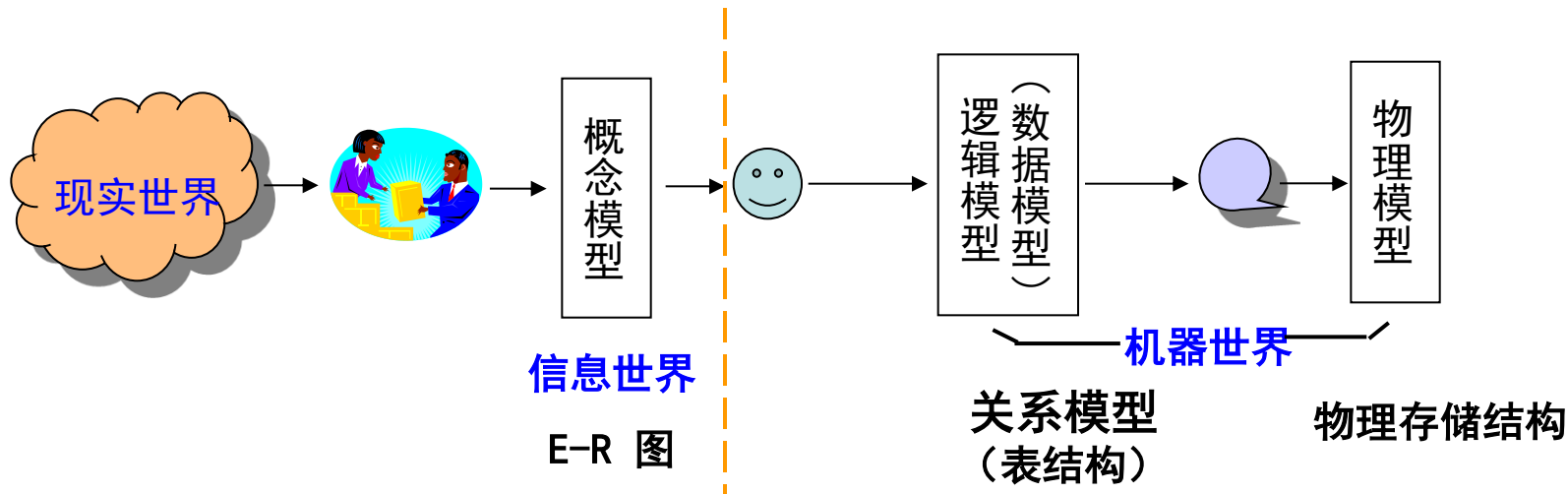
- 将概念设计转换为某个DBMS所支持的数据模型
- 关系标准化，检查冗余和相关的异常关系结构

□ 物理数据库设计

- 索引，集群和数据库调优

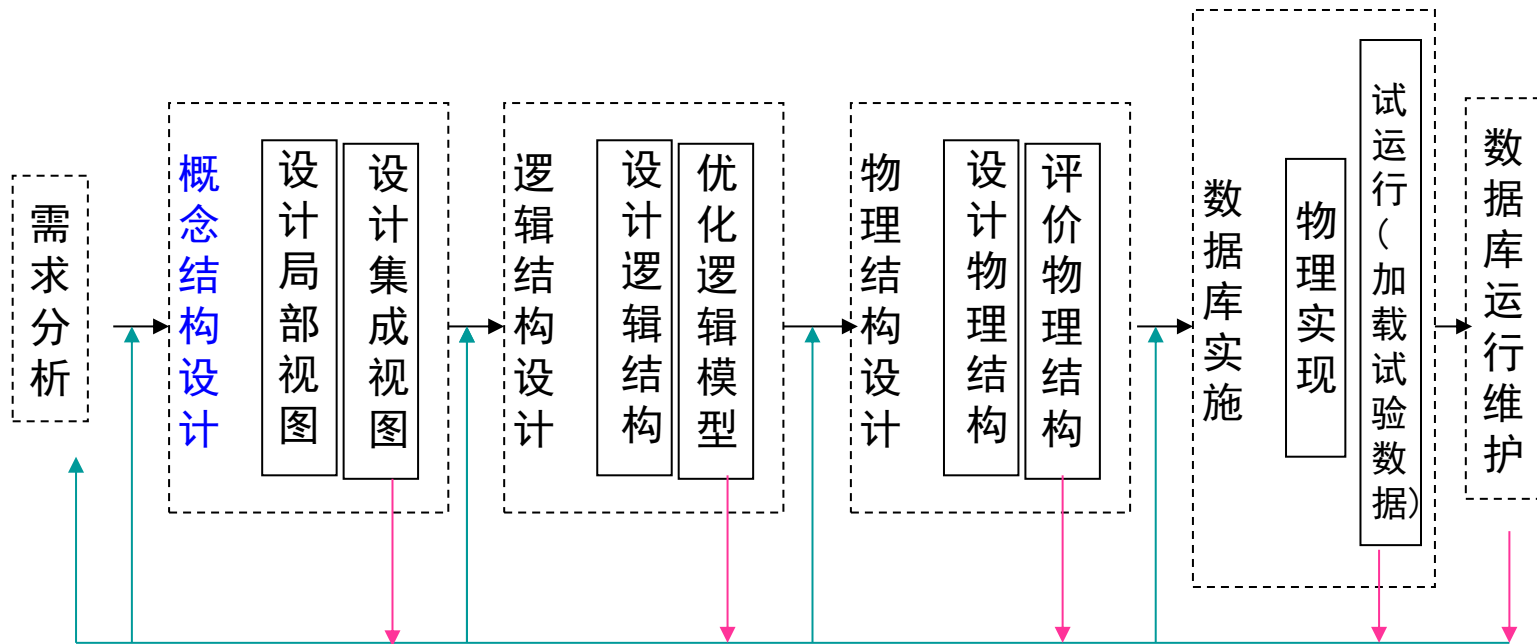
设计数据库的E-R模式

□ 数据库的设计步骤（补充）



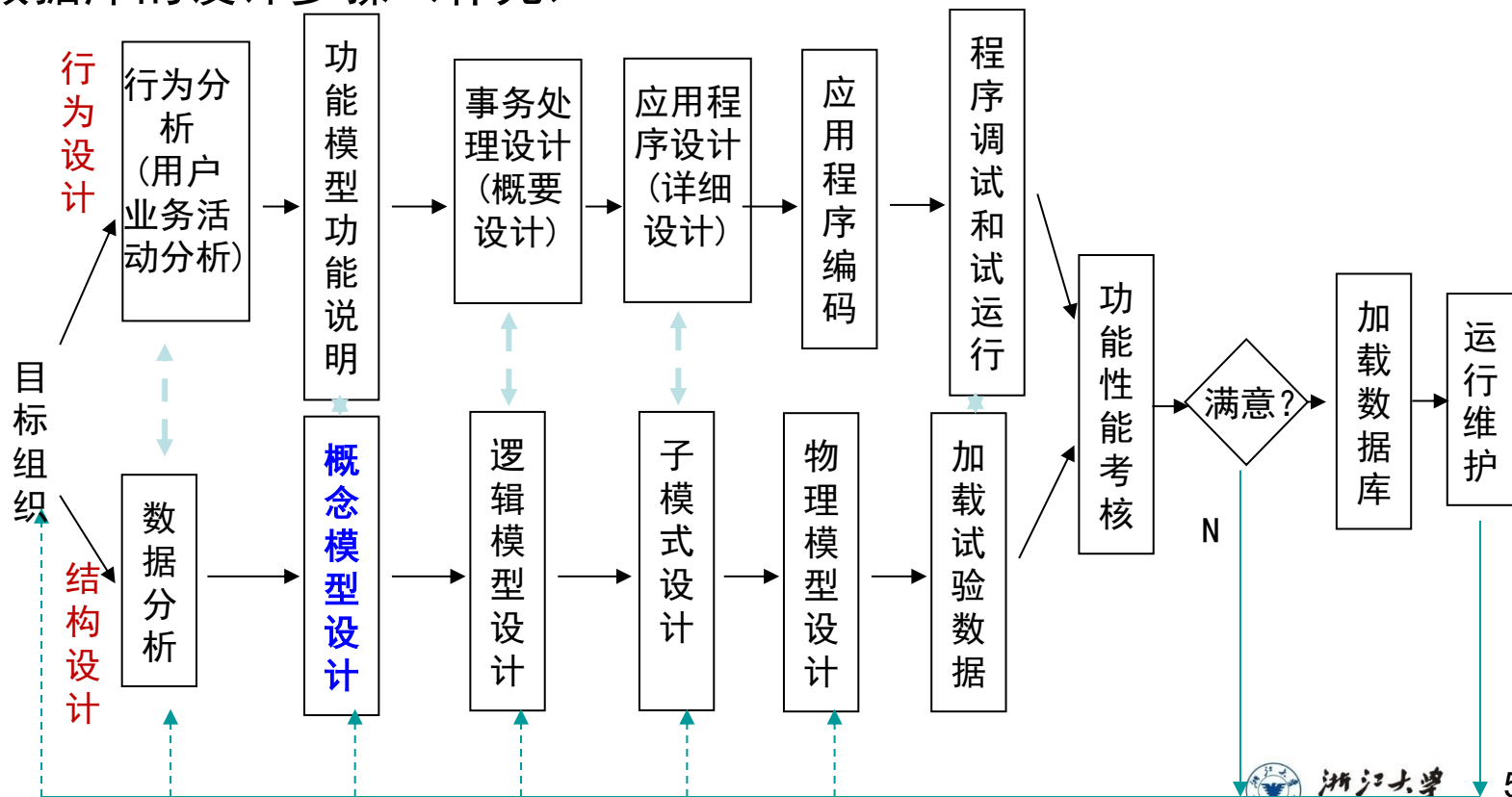
设计数据库的E-R模式

□ 数据库的设计步骤（补充）



设计数据库的E-R模式

□ 数据库的设计步骤（补充）

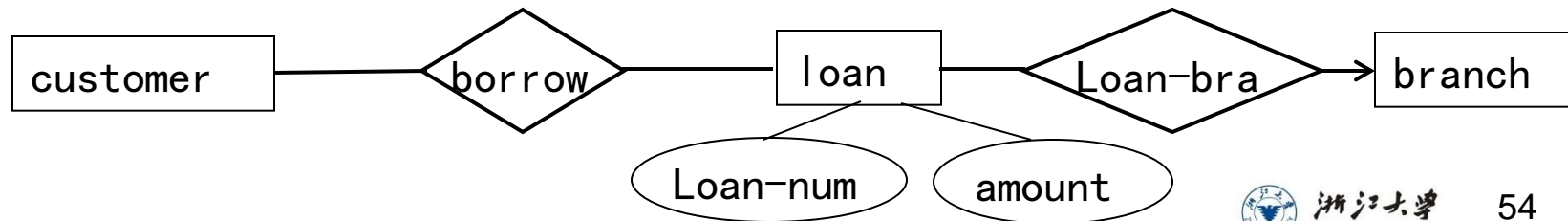
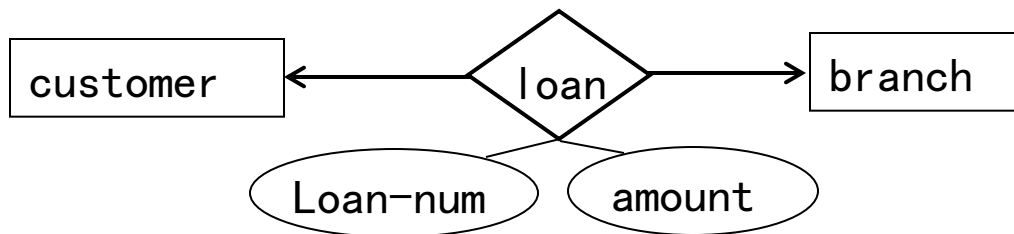


□ 用属性还是实体集来表示对象

- *instructor(ID, name, ..., phone)*, 优点: 简单。但多个电话怎么处理? 电话的其他属性?
- *instructor(ID, name, ...)*
phone(phone-num, location, type, color)
ins-phone(i_ID, phone-num)
- 若一个对象只对其名字及单值感兴趣, 则可作为属性, 如 **性别**; 若一个对象除名字外, 本身还有其他属性需描述, 则该对象应定义为实体集。如 **电话**, **住址**, **部门**
- 一个对象不能同时作为实体和属性
- 一个**实体集**不能与另一实体集的**属性**相关联, 只能实体与实体相联系

□ 用实体集还是用联系集

- 二个对象之间发生的动作 “relationship set” 表示
- 还需要考虑映射基数的影响
- 例，考虑 *branch*、*loan*、*customer*，如果一个客户在一个分支机构有多个贷款账户，那么将会影响E-R设计



□ 用实体属性还是用联系

- *student*(*sid*, *name*, *sex*, *age*, ..., *supervisor-id*, *supervisor-name*,
supervisor-position, ..., *class*, *monitor*)

- 要从对象的语义独立性和减少数据冗余考虑

```
student(sid, name, sex, age, ...);  
supervisor(sup-id, name, position, ...);  
stu-sup(sid, sup-id, from, to);  
class(classno, specialty, monitor, stu-num);  
stu-class(sid, classno);
```

- 用n元联系还是二元联系
- 用强实体集还是弱实体集
- 特化/概化的使用，有助于设计的模块化
- 聚集的使用，将聚集实体集视为单个单元，从而不必关心其内部结构的细节

□ 获取系统需求

- *classroom, department, course, instructor, section, student, time_slot*

□ 实体集设计

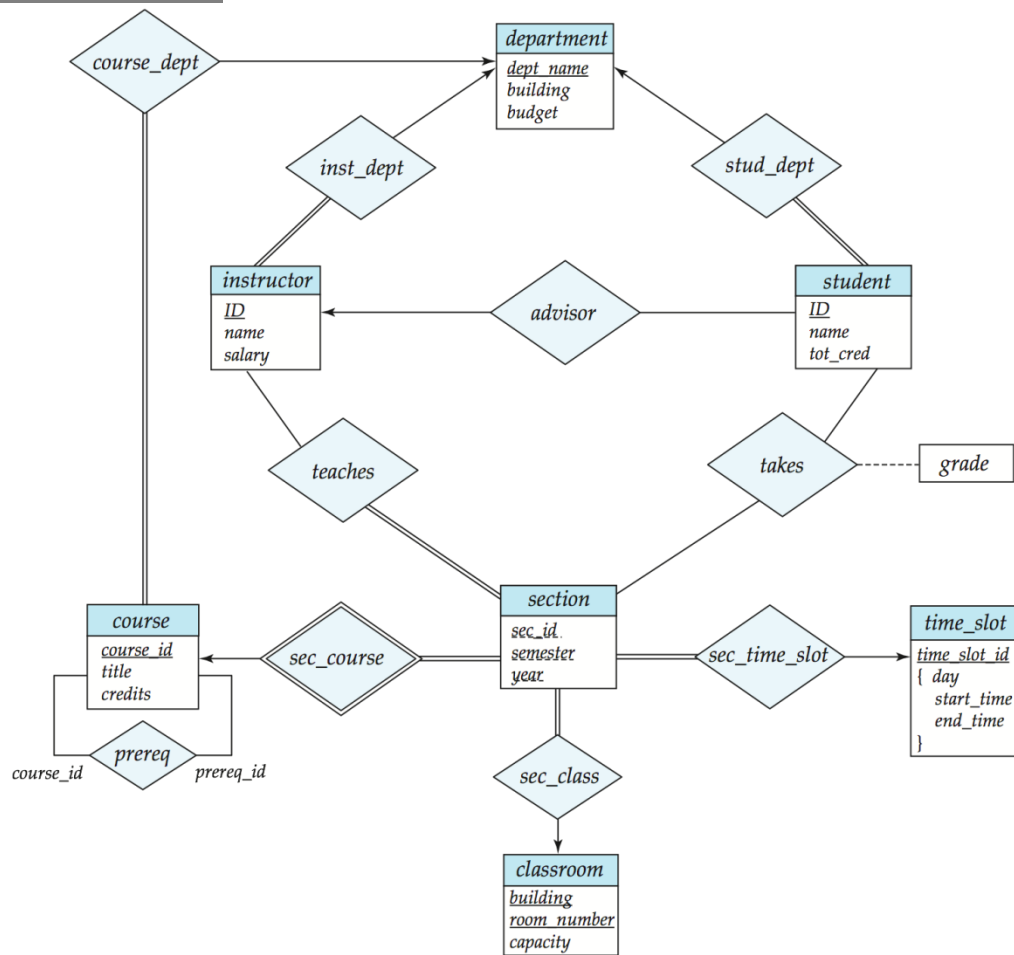
- *classroom (building, room_number, capacity)*
- *department (dept_name, building, budget)*
- *course (course_id, title, credits)*
- *instructor (ID, name, salary)*
- *section (course_id, sec_id, semester, year)*
- *student (ID, name, tot cred)*
- *time_slot (time_slot_id, {(day, start_time, end_time)})*

□ 联系集设计

- *inst_dept*: 关联教师和系
- *stud_dept* : 关联学生和系
- *teaches*: 关联教师和开课
- *takes*: 关联学生和开课, 包含描述性属性 *grade*
- *course_dept*: 关联课程和系
- *sec_course*: 关联开课和课程
- *sec_class*: 关联开课和教室
- *sec_time_slot*: 关联开课和时段
- *advisor*: 关联学生和教师
- *prereq*: 关联课程和先修课程

大学数据库的设计

□ E-R图



将E-R模式转换为数据库表

- ❑ E-R图转换成表格式是从E-R图导出关系数据库设计的基础
- ❑ 符合E-R图的数据库可以表示成若干表的集合
- ❑ 对每一个实体集及联系集都有一个唯一的表，该表的名字就是对应实体集或联系集的名字

实体集表示为表

□ 强实体集转换到具有相同属性的表

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

department (*dept name*, *building*, *budget*)

还有: *classroom*(*building*, *room number*, *capacity*)

course(*course id*, *title*, *credits*)

instructor(*ID*, *name*, *salary*)

student(*ID*, *name*, *tot cred*)

有复合属性的实体集

- 复合属性可通过为每个组成属性创建一个单独属性
- 例，给定带有复合属性 *name* (组成属性为 *first-name* 和 *last-name*) 的实体集 *instructor*，对应的表具有两个属性

name.first-name 和 *name.last-name*

或是，*first-name* 和 *last-name*

有多值属性的实体集

□ 实体集E的多值属性 M 用一个单独的表EM表示

■ 例, *instructor*的多值属性`phone_number`表示为表
instructor(*ID*, *name*, *salary*, *phone_number*)

⇒ *instructor*(*ID*, *name*, *salary*)

instructor_phone(*ID*, *phone_number*)

□ 多值属性的每个值映射到表EM中的单独行

■ 主键为“22222”的*instructor*实体及其电话号码“555-1234”和“555-4321”映射到两行:

— *instructor_phone*联系中的(22222 , 555-1234)和(22222 , 555-4321)

弱实体集表示

- 弱实体集转换成的表还包含对应于其标识强实体集的主键的列

<u>course_id</u>	<u>sec_id</u>	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

section(course_id, sec_id, semester, year, building,
room_number, time_slot_id)

联系集表示成表

- ❑ 联系集表示成的表具有对应于两个参加实体集的主键的列，以及对应于联系集自己的描述性属性的列
- ❑ 对于多对多的联系集
 - *takes*(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
 - *teaches*(*ID*, *course_id*, *sec_id*, *semester*, *year*)
 - *prereq*(*course_id*, *prereq_id*)

联系集表示成表

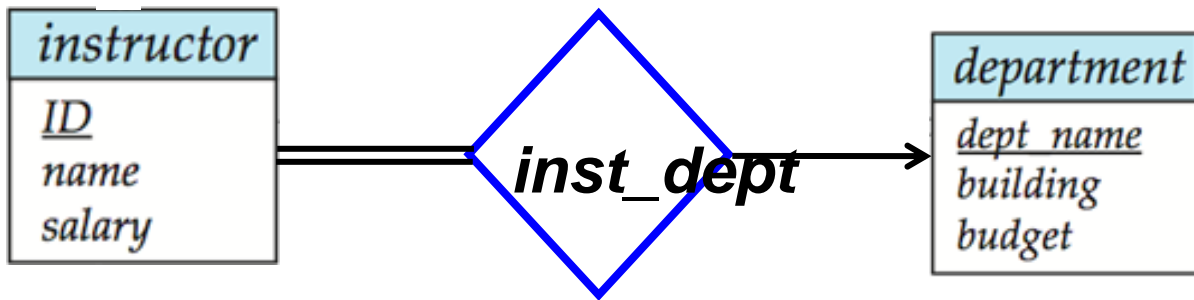
□ 对于多对一的联系集

- *advisor*(*s_ID*, *i_ID*)
- *sec_course*(*course_id*, *sec_id*, *semester*, *year*)
- *sec_time_slot*(*course_id*, *sec_id*, *semester*, *year*, *time_slot_id*)
- *sec_class*(*course_id*, *sec_id*, *semester*, *year*, *building*, *room number*)
- *inst_dept*(*ID*, *dept_name*)
- *stud_dept*(*ID*, *dept_name*)
- *course_dept*(*course_id*, *dept_name*)

□ 对于一对多的联系集（和多对一类似）

表的冗余

- 如果多对一和一对多联系集在“多”端是完全的，则可不必要为联系集创建表，而是在对应于“多”端的表中加入对应于“一”端表的主键的额外属性



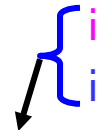
instructor(*ID*, *name*, *salary*); *department*(*dept_name*, *building*, *budget*)

inst_dept(*ID*, *dept_name*)

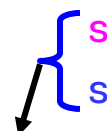
instructor(*ID*, *name*, *dept_name*, *salary*);

- 对于一对一联系集，任何一边都可选作为“多”端
 - 即，额外属性可加入到任何一个对应于两个实体集的表中
- 如果“多”端参加联系是部分的，上述方法可导致空值
 - 例，如果 *inst_dept* 是部分参与的，那么我们将为那些没有相关联的系的教师在属性 *dept_name* 中存放空值

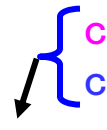
表的冗余

 inst_dept(ID, dept_name)
instructor(ID, name, salary)

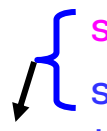
instructor(ID, name, dept_name, salary)

 stu_dept(ID, dept_name)
student(ID, name, tot_cred)

student(ID, name, dept_name, tot_cred)

 course_dept(course_id, dept_name)
course(course_id, title, credits)

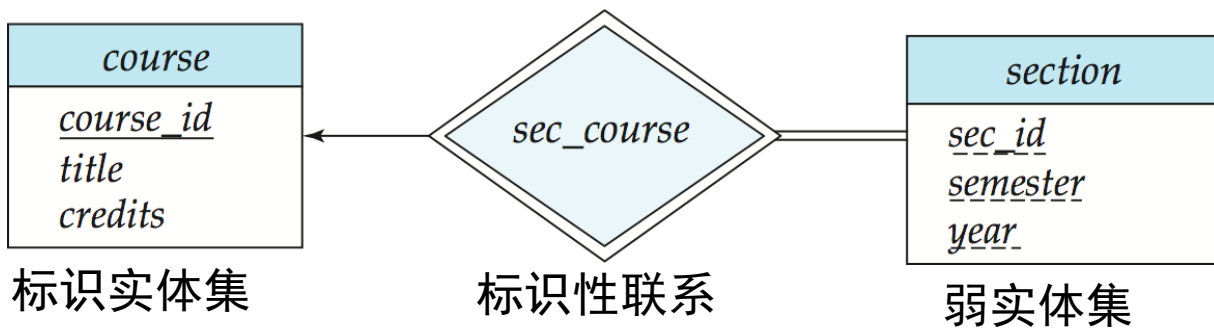
course(course_id, title, dept_name, credits)

 sec_class(course_id, sec_id, semester, year, building, room_number)
section(course_id, sec_id, semester, year);
section(course_id, sec_id, semester, year, building, room_number)



表的冗余

- ❑ 联系弱实体集及其标识性实体集的联系集对应的表是冗余的
- ❑ 例，*section*表已经包含了出现在*sec_course*表中的信息（如，*course_id*，*sec_id*, *semester*和*year*列）



表的冗余

section(course_id, sec_id, semester, year, building, room_number, time_slot_id)

<u>course_id</u>	<u>sec_id</u>	<u>semester</u>	<u>year</u>	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

删除: sec_course (course_id, sec_id, semester, year)



特化表示成表

□ 方法1:

- 为高层实体集构造表
- 为每个低层实体集构造表，包括高层实体集的主键和局部属性

表	表属性
<i>person</i>	<i>ID, name, street, city</i>
<i>student</i>	<i>ID, tot_cred</i>
<i>employee</i>	<i>ID, salary</i>

- 缺点：获得 *employee*之类的实体的信息需要访问两个表

□ 方法2:

- 为每个实体集构造表，其属性包括所有局部属性和继承来的属性

表	表属性
<i>person</i>	<i>ID, name, street, city</i>
<i>student</i>	<i>ID, name, street, city, tot_cred</i>
<i>employee</i>	<i>ID, name, street, city, salary</i>

- 如果特化是全部特化，则没有必要为一般实体 *person* 创建表
 - 可以被定义为包含特化联系的“视图”
 - 由于外键约束的需要，可能仍然需要定义关系模式 *person*
- 缺点：对于既是学生又是雇员的人，其 *name*、*street* 和 *city* 被冗余存储

大学数据库模式

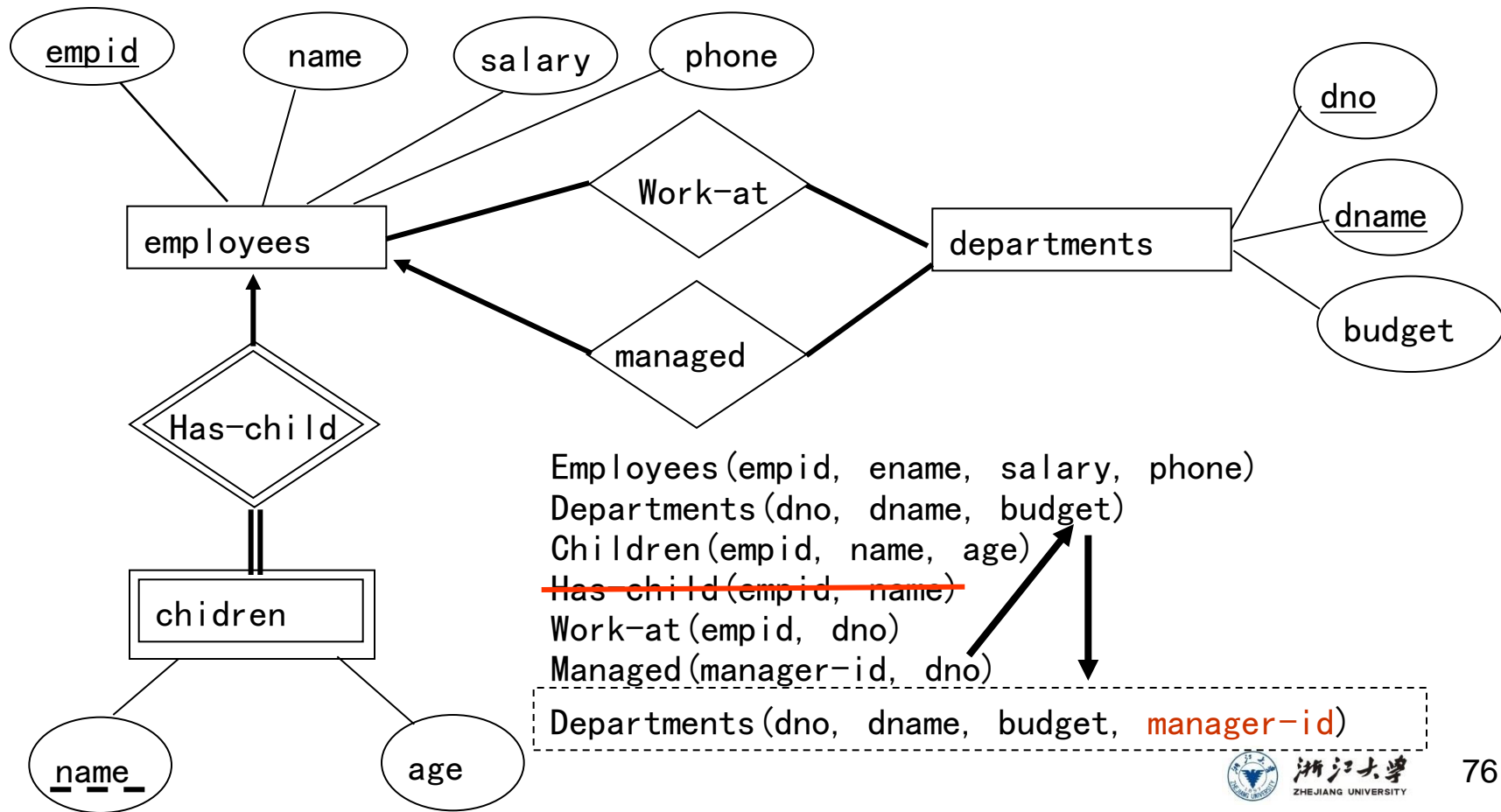
- ❑ *classroom*(building, room_number, capacity)
- ❑ *department*(dept_name, building, budget)
- ❑ *course*(course_id, title, dept_name, credits)
- ❑ *instructor*(ID, name, dept_name, salary)
- ❑ *section*(course_id, sec_id, semester, year, building, room_number, time_slot_id)
- ❑ *teaches*(ID, course_id, sec_id, semester, year)
- ❑ *student*(ID, name, dept_name, tot_cred)
- ❑ *takes*(ID, course_id, sec_id, semester, year, grade)
- ❑ *advisor*(s_ID, i_ID)
- ❑ *time_slot*(time_slot_id, day, start_time, end_time)
- ❑ *prereq*(course_id, prereq_id)



示例一

- ❑ 一个公司的数据库需要存储有关雇员（由empid唯一标识，有姓名、工资和手机号码作为属性），部门（由dno唯一标识，有部门名称与预算作为属性）和雇员的孩子信息（有姓名和年龄属性）。员工在部门工作，每个部门由一个管理员管理，孩子必须被唯一名称标识，其父母（假定只有父母中的一位在公司工作）是已知的。一旦父母离开公司，我们将不再关注孩子的信息。绘制ER图，然后将其转换为关系模式。

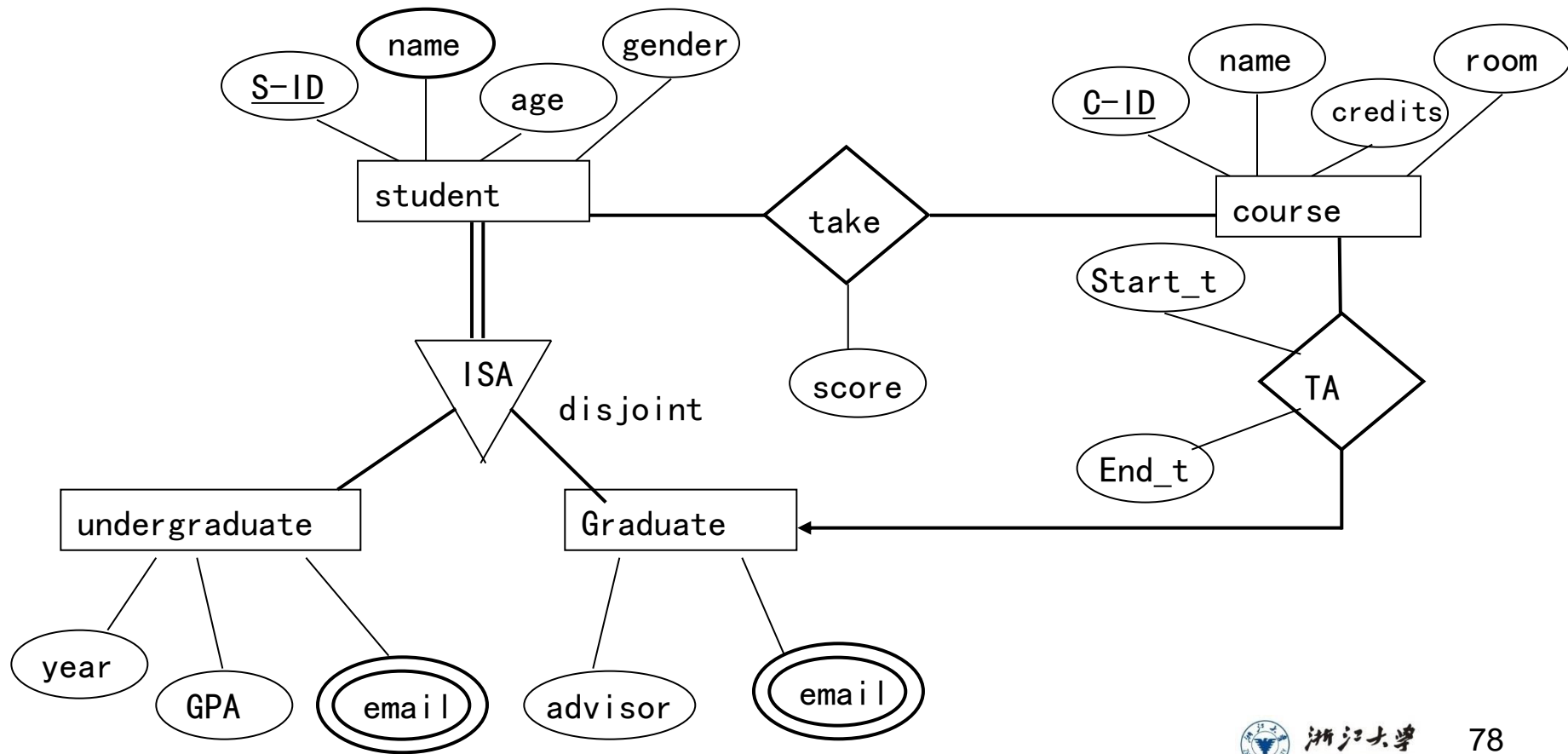
示例一



示例二

- ❑ 学生学习某门课程。每个学生都有学号，姓名，年龄和性别属性。每门课程都有课程编号，课程名，学分和上课教室编号属性。每个学生可以是本科生或研究生。对于每一个本科生，我们想记录他/她的学习年份，平均绩点，以及（可能是多个）电子邮箱。对于每一个研究生，我们要记录他/她的导师的名字，以及（可能是多个）电子邮箱。此外，每门课程有一名研究生作为课程助教，我们要记录助教的开始和结束时间（例如，开始于2015年3月10日，结束于2015年6月30日）。绘制E-R图。

示例二



- ❑ 数据库设计主要涉及数据库模式的设计
- ❑ E-R模型主要用于数据库设计过程、E-R图用于表示数据库模式的逻辑结构
- ❑ 实体、实体集
- ❑ 联系、联系集
- ❑ 超码、候选码以及主码同样适用于实体和联系集
- ❑ 映射基数
- ❑ 弱实体集、强实体集
- ❑ 将E-R图转换为关系模式

- ❑ 数据库设计主要涉及数据库模式的设计
- ❑ E-R模型主要用于数据库设计过程，E-R图用于表示数据库模式的逻辑结构，以及E-R图的其他表示方法
- ❑ 实体、实体集
- ❑ 联系、联系集
- ❑ 超码、候选码以及主码同样适用于实体和联系集
- ❑ 映射基数
- ❑ 弱实体集、强实体集
- ❑ 特化和概化定义了一个高层实体集和一个或多个低层实体集之间的包含关系

- ❑ 聚集是一种抽象，其中联系集（和跟它们相关的实体集一起）被看作高层实体集，并且可以参与联系
- ❑ E-R设计问题
- ❑ 将E-R图转换为关系模式

谢谢！