

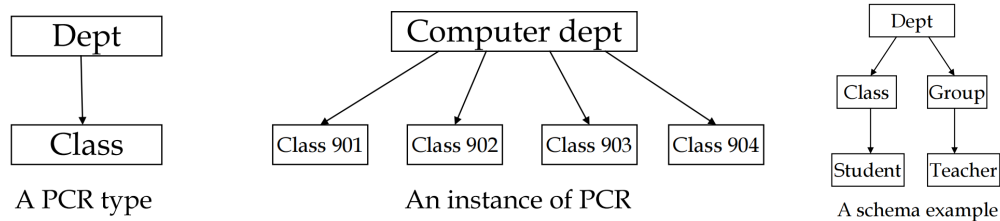
Chapter 2 Data Model

1. 分层数据模型 Hierarchical Data Model

1.1. 基本思想

由于现实世界中的许多事物都是以层次组织的，因此分层模型能够以树状结构来描述现实世界。

- 记录和字段
- 父子关系（Parent-Child relationship PCR）：分层模型中最基本的数据关系。它表达了两种记录类型之间的1对N关系。



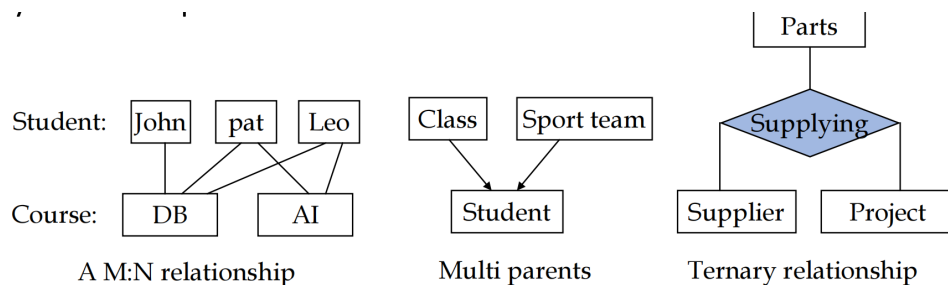
1.2. 分层数据架构 Hierarchical Data Schema

- 一个分层数据模式由PCR组成。
- 每个PCR表达一个1: N关系
- 每个记录类型只能有一个父记录

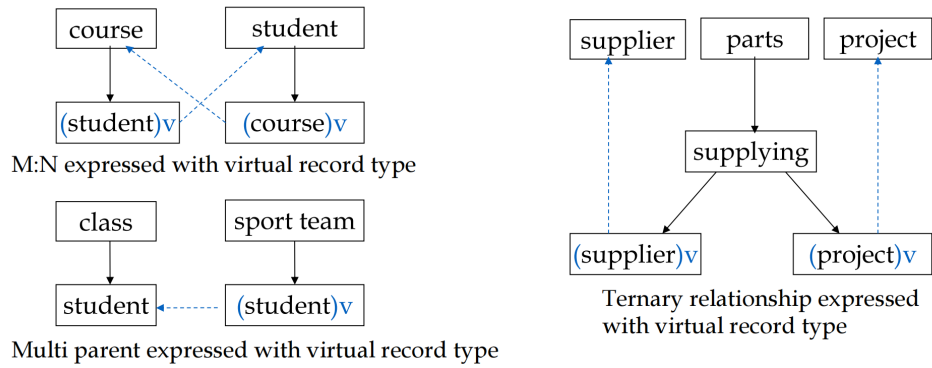
1.3. 虚拟记录Virtual Record

为什么：在现实世界中，很多数据都不是层次型的，很难直接用PCR表达它们。

- 不同记录类型之间的M: N关系
- 一种记录是两个以上PCR的子类型
- N-ary关系

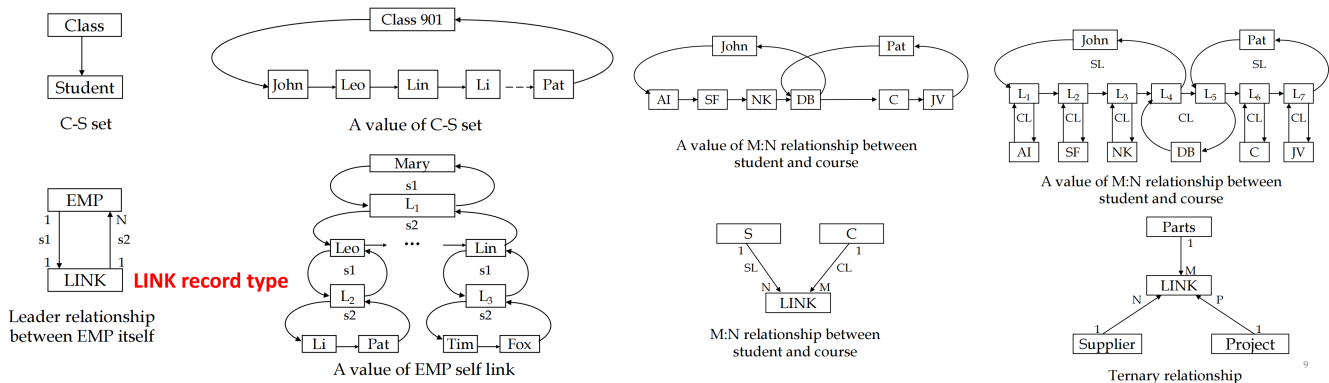


为了避免冗余，引入了虚拟记录来表达上述关系，实际上它是一种指针。



2. 网络数据模型 Network Data Model

- 基本的数据结构是“集合”，它代表了现实世界中事物之间的1: N关系。
 - “1”为所有者，“N”为成员。
- 一种记录类型可以是多个集合的所有者，也可以是多个集合的成员。许多集合形成网络结构来表达现实世界。
- 它突破了层次结构的限制，可以更轻松地表达非层次数据。
 - **记录和数据项：**数据项类似于层次模型中的字段，但可以是向量。
 - **集合：**表达两种记录类型之间的1: N关系。
 - **链接记录类型LINK record type：**用于表达自我关系(self relationship)、M: N关系和N元关系。



3. 关系数据模型 Relational Data Model

3.1. 基本概念

基本的数据结构是“表”，或关系。现实世界中的事物和它们之间的关系都以表的形式表达（all expressed as tables），因此可以用严格的数学方法进行研究，它把数据库技术提升到了理论的高度。其特点：

- 基于集合论，抽象层次高
- 屏蔽所有底层细节，简单明了，容易理解
- 可以建立新的代数系统——关系代数
- 非过程化查询语言——SQL
- 软链接——与前文数据模型的本质区别

3.2. 关系模式和实例 Relation Schema and Instance

- A_1, A_2, \dots, A_n 属性 (*attributes*)
- $R = (A_1, A_2, \dots, A_n)$ 是关系模式 (*relation schema*)
 - 例如: $instructor = (ID, name, deptname, salary)$
- 在关系模式 R 上定义的关系实例 r 用 $r(R)$ 表示
- 关系当前的数值由一个表指定
- 关系 r 的一个元素 t 称为一个元组 (tuple)，表示为表中的一行

3.3. 属性和域 Attributes & Domain

- 在关系模型中，实体的特性被表示为属性。
- 属性值（通常）必须是原子 (atomic) 的，即不可分割的
- 属性值的范围被称为它的域。
 - 基本数据单元
 - 空值 *null*：是每个域的成员，表示该值是“未知”的，导致许多操作的定义复杂化

3.4. 关系和元组 Relation & Tuple

- 一个现实世界的实体可以表示为一个或多个关系
- 关系是在其属性域上定义的N元关系
- 具有属性 A_1, A_2, \dots, A_n ，相应域为 D_1, D_2, \dots, D_n 的关系 R
 - 表达为: $R = (A_1/D_1, A_2/D_2, \dots, A_n/D_n)$ 或者 $R = (A_1, A_2, \dots, A_n)$
 - 这被称为 *the schema of R*， n 是属性数，称为 R 的度
 - A_i ($1 \leq i \leq n$) 是属性名

- R 的一个实例可以表示为 r 或 $r(R)$, $r = \{t_1, t_2, \dots, t_m\}$
 - 其中每一个 tuple 都可以表示为:
 - $t = \langle v_1, v_2, \dots, v_n \rangle, v_i \in D_i, 1 \leq i \leq n$
 - $t \in D_1 \times D_2 \times \dots \times D_n, 1 \leq i \leq n$ (笛卡尔积 *Cartesian Product*)
 - $r \subseteq D_1 \times D_2 \times \dots \times D_n, 1 \leq i \leq n$
- 关系也被称为表格, 属性被称为列 (columns) , 元组被称为行 (rows)

3.5. 关系是无序的

3.6. 数据库模式 Database Schema

- 数据库模式是数据库的逻辑结构
- 数据库实例是数据库在某一特定时刻的数据快照

3.7. 键 Keys

$$K \subseteq R$$

- **超键 (superkey)** : 如果 K 的值足以标识每个可能的 $r(R)$ 关系中唯一的元组。
 - $\{ID\}$ 和 $\{ID, name\}$ 都是 *instructor* 的超键。
- **候选键 (candidate key)** : 所有超键中最小的 K
 - ID 是 *instructor* 的候选键之一。
- **主键 (primary key)** : 候选键中选择一个
 - 如果多余一个候选键, 其余的成为**替代键 (alternate key)**
 - 如果主键由一个关系的所有属性组成, 则称为**全键 (all key)**
- **外键约束 (Foreign key constraint)** : 一个关系中的值必须在另一个关系中存在
 - 参考关系
 - 被参考关系

3.8. 外键和参照完整性 Foreign Keys & Referential Integrity

- **外键**: 一个关系中的属性集, 用于“引用”另一个关系中的元组
 - 必须与第二个关系的主键相对应
 - 例如, *sid* 是一个外键 referring to Students:

- *Enrolled(sid : string, cid : string, grade : string)*
- 如果所有的外键约束都被强制执行，则实现了参照完整性，即没有悬挂引用。

Enrolled

<i>sid</i>	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

3.9. 完整性约束

- 实体完整性约束 (Entity Integrity Constraint) :

每个关系都应当有一个主键，主键列的值在表中必须唯一且不为空，从而保证数据的一致性和有效性。

- 参照完整性约束 (Referential Integrity Constraint) :

外键的值必须是关联表的主键值，参照完整性约束了无效的引用和不一致的数据关系。当涉及到关联查询和表之间的关系时，参照完整性约束非常重要。

- 域完整性约束 (Domain Integrity Constraint) :

域完整性约束用于定义列上的数据类型、范围和格式等规则。它确保列的值符合预定义的规范。域完整性约束有助于保证数据的一致性和有效性。

3.10. 关系查询语言 Relational Query Languages

- 关系查询语言可以分为两种主要类型：过程性语言和非过程性语言（也称为声明性语言）
 - 过程性语言：通过提供逐步指令来指定如何实现所需的结果
 - 非过程性语言：描述所需的结果，而不需要指定具体的实现步骤。
- 在关系查询语言中，存在着三种“纯”语言，在计算能力上是等效的：
 - 关系代数 (Relational Algebra)
 - 元组关系演算 (Tuple Relational Calculus)
 - 域关系演算 (Domain Relational Calculus)

3.11. 关系代数

- 关系代数不具备图灵机的计算能力，由六个基本操作组成，用于对关系数据库进行查询和操作

- 一种过程语言，由一组操作组成，这些操作将一个或两个关系作为输入，并产生一个新关系作为结果
- 六个基本操作：
 - 选择 (select) σ : 从关系中选择满足指定条件的元组。
 - 投影 (project) π : 从关系中选择指定的属性列，生成一个新的关系。
 - 并 (union) \cup : 将两个关系合并为一个包含两个关系的所有元组的新关系。
 - 差 (set difference) $-$: 从一个关系中删除另一个关系中存在的元组，生成一个新关系。
 - 笛卡尔积 (Cartesian product) \times : 将两个关系的所有可能组合的元组生成一个新关系。
 - 重命名 (rename) ρ : 用于为关系中的属性或关系本身指定新的名称。

4. 关系代数 Relation Algebra

关系实例 (relation instances) 的操作代数

- 封闭性质：结果也是关系实例
 - 允许丰富的组合
- 类型化：输入模式决定输出
 - 可以静态检查查询是否合法。

关系代数与集合

- 纯关系代数具有集合语义：
 - 关系实例中没有重复的元组 \rightarrow 不同于SQL (具有多重集语义)

基本操作符 (3.11)

- $\{\sigma, \pi, \cup, -, \times\}$ 是一个完整的操作集，任何其他关系代数运算都可以从它们推导出来。
- 附加运算：
 - 交集Intersection、连接join、除法division、外连接outer join、外部联合out union

4.1. Selection

- 选择满足选择条件的行。
- 结果中没有重复 (No duplicates)
- 结果模式与输入关系相同
- 结果关系可以作为另一个关系代数操作的输入

4.2. Projection

- 结果模式只包含投影列表中的字段，并且与输入关系中的字段具有相同的名称。
- 投影操作符必须消除重复项
 - 注意：实际系统通常不会执行重复消除，除非用户明确要求这样做。

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname, rating}(\sigma_{rating > 8}(S2))$$

↓

sname	rating
yuppy	9
rusty	10

4.3. Cross-Product

- S_1 的每一行与 R_1 的每一行配对。
- 结果模式具有 S_1 和 R_1 中每个属性各一个属性，如果可能则继承属性名称
 - 若 S_1 和 R_1 都有一个名为 sid 的属性，一般考虑重命名

4.4. Renaming

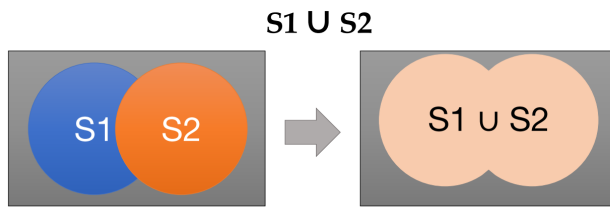
- 更改关系及其属性的名
- 关系代数不需要名称，我们只需要使用位置参数即可
 - $\rho(C(1 \rightarrow sid_1, 5 \rightarrow sid_2), S_1 \times R_1)$

sid1				sid2		
(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

4.5. Union

- 两个输入关系必须兼容Union Compatible：
 - 字段数量相同
 - 相应位置的字段具有相同的类型

- SQL Expression: UNION vs. UNION ALL



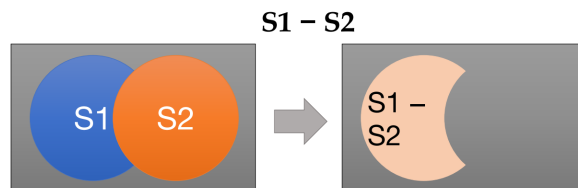
sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

- 关系代数采用的是Union \Leftrightarrow Union all 重复行保留重复

4.6. Set Difference

- 输入关系必须兼容 (Compatible) 。
- SQL表达式: EXCEPT



sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

4.7. Intersection

- 输入关系必须兼容
- SQL表达式: INTERSECT



sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

4.8. Joins

条件连接 (Condition join)

- Condition Join** : $R \bowtie_c S = \sigma_c (R \times S)$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

← $S1 \bowtie_{S1.sid < R1.sid} R1$


- 结果模式与交叉积相同

- 交集的元组数少于交叉积，可能能够更有效地计算
- 有时也被称为theta-join: \bowtie_{θ}

等值连接 (Equi-join)

- 条件连接的一种特殊情况，其中条件 c 只包含等值 (equalities) 。

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

 $S1 \bowtie_{sid} R1$

- 结果模式类似于交叉积，但等值属性只出现一次
- 特殊情况——自然连接：基于所有公共属性的等值连接

自然连接 (Natural join)

- 等值连接的特殊情况，其中为所有匹配字段指定相等性并且消除重复字段
- $R \bowtie S = \pi_{uniquefld}.\sigma_{eq.matchingfld}(R \times S)$
 - 计算 $R \times S$
 - 选择在两个关系中出现的字段具有相等值的行
 - 投影到所有唯一字段的集合上

外连接 (Outer joins)

- 保留未匹配的元组，空的部分设置为Null
 - 左外连接 $* \bowtie$: 保留左关系中的所有元组
 - 右外连接 $\bowtie *$: 保留右关系中的所有元组
 - 全外连接 $* \bowtie *$: 保留左和右关系中的所有元组

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96
31	Lubber	8	55.5	null	null

$S1 * \bowtie R1$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

$S1 \bowtie^* R1 =$
 $S1 \bowtie R1$ (Why?)

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96
31	Lubber	8	55.5	null	null

$S1 * \bowtie^* R1$

4.9. Outer Unions

- 扩展联合操作：可以联合两个不联合兼容（not union-compatible）的关系
- 结果集中的属性集是两个操作数属性集的并集
- 在原始元组中不存在的属性的值被填充为NULL

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

sid	bid	day
22	101	10/10/96
58	103	11/12/96

sid	sname	rating	age	bid	day
22	dustin	7	45.0	null	null
31	lubber	8	55.5	null	null
58	rusty	10	35.0	null	null
22	null	null	null	101	10/10/96
58	null	null	null	103	11/12/96

$S1 \cup R1$

4.10. Division

- 不支持作为基本操作符
- 假设 A 有两个字段 x, y ， B 有一个字段 y ：
 - $A/B = \{ \langle x \rangle \mid \exists \langle x, y \rangle \in A, \forall \langle y \rangle \in B \}$
 - A/B 包含的所有 x （即水手）在 B 的每个 y （即船）上都有一个对应的 x, y 元组。
- 一般来说， x 和 y 可以是任何字段列表； y 是 B 中的字段列表， $x \cup y$ 是 A 中的字段列表。

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

$B1$

pno
p2
p4

$B2$

pno
p1
p2
p4

$B3$

sno
s1
s2
s3
s4

$A/B1$

sno
s1
s4

$A/B2$

sno
s1

$A/B3$

- 使用基本运算符表示 A/B
 - $Disqualified_x = \pi((\pi_x(A) \times B) - A)$
 - A 中 x 和 B 做交叉积，得出所有可能性的集合
 - 把已经存在于 A 中可能性删去
 - 若某个 x 对应的可能性全部被删去，则在 π_x 时不会保留他
 - $A/B = \pi_x(A) - Disqualified_x$
 - 用 A 的 π_x 减去上面结果，保留的即为 A/B

4.11. 运算符分类

- 一元运算符Unary: σ, π, ρ
- 二元操作符Binary: $\cup, -, \times$
- 复合运算符Compound: \cap, \bowtie

5. 关系演算 Relation Calculus

- 只需要指出结果必须满足的逻辑条件
- 两种类型:
 - 元组关系演算 (TRC: Tuple relation calculus)
 - 变量遍历元组 (即被绑定到元组上)
 - 域关系演算 (DRC: Domain relation calculus)
 - 变量遍历域元素 (属性值)
 - 都是一阶逻辑的简单子集
- 演算有变量、常量、比较运算符、逻辑联接符和量词
- 演算中的表达式称为公式: 一个答案元组基本上是将常量分配给变量, 使公式评估为真。

5.1. 域关系演算DRC

5.1.1. 基本概念

- 查询的形式为:
 - $\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n, x_{n+1}, \dots, x_{n+m}) \}$
- 其中 $x_1, x_2, \dots, x_n, \dots, x_{n+m}$ 被称为域变量
 - 答案: 使 $P(x_1, x_2, \dots, x_n, x_{n+1}, \dots, x_{n+m})$ 为真的 $\langle x_1, x_2, \dots, x_n \rangle$ 元组
 - 公式是递归定义的, 从简单的原子公式开始, 使用逻辑连接操作符构建

5.1.2. DRC公式

- 原子公式 Atomic formula:
 - $\langle x_1, x_2, \dots, x_n \rangle \in Rname$ 或 $X op Y$ 或 $X op const$
 - op 是 $<, >, =, \leq, \geq, \neq$ 之一、
- 公式 Formula:
 - 原子公式
 - $\neg p, p \wedge q, p \vee q$, 其中 p 和 q 是公式

- $\exists X(p(X))$, 其中变量 X 在 $p(X)$ 中是自由的
- $\forall X(p(X))$, 其中变量 X 在 $p(X)$ 中是自由的
- 使用量词 $\exists X$ 和 $\forall X$ 会使 X 受约束, 未受约束的变量是自由的。

Find sailors rated > 7 who've reserved a red boat Find sailors who've reserved all boats

• $\{ \langle I, N, T, A \rangle \mid \langle I, N, T, A \rangle \in \text{Sailors} \wedge T > 7 \wedge$
 $\exists Ir, Br, D [\langle Ir, Br, D \rangle \in \text{Reserves} \wedge Ir = I \wedge$
 $\exists B, BN, C [\langle B, BN, C \rangle \in \text{Boats} \wedge B = Br \wedge C = \text{'red'}]] \}$

• $\{ \langle I, N, T, A \rangle \mid \langle I, N, T, A \rangle \in \text{Sailors} \wedge$
 $\forall B, BN, C (\neg (\langle B, BN, C \rangle \in \text{Boats}) \vee$
 $(\exists Ir, Br, D (\langle Ir, Br, D \rangle \in \text{Reserves} \wedge Ir = I \wedge Br = B))) \}$

- 对于一个水手若想保留下来必须对所有船使得条件 P 成立
- P 想为真 \Rightarrow 这个船要么不在表里, 要么被他借过

5.2. 不安全查询, 表达力 Unsafe Queries, Expressive Power

- 可以写出语法上正确的算术查询, 但有无限数量的答案! 这样的查询是不安全的。
 - e.g. $\{S \mid \neg(S \in \text{Sailors})\}$
 - 在关系代数中可以表达的每个查询都可以在DRC/TRC中表达为安全查询; 反之亦然。
- 关系完备性:
 - $\{\sigma, \pi, \cup, -, \times\}$ 是一个完备的操作集合, 关系计算可以表达这五个操作, 因此也是关系完备的。
 - SQL语言基于关系计算, 因此它可以表达任何在关系代数/计算中可表达的查询。

5.3. 元组关系演算 Tuple Relational Calculus

- 查询的形式为: $\{t \mid \langle \text{attribute list} \rangle \mid P(t)\}$
- t 被称为元组变量
- 答案包括所有使公式 $P(t)$ 为真的元组。
 - e.g. 查找所有评分高于7且年龄小于50的海员的名字;
 - $\{t[N] \mid t \in \text{Sailors} \wedge t.T > 7 \wedge t.A < 50\}$

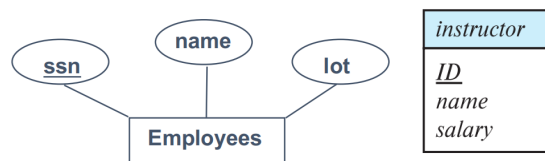
6. 实体关系模型 Entity-Relationship Model

- 基于图形的模型
 - 可以被看作是图或者关系之上的一个粉饰
 - 直观感受更灵活, 结构更少
 - 与对象关系映射 (ORM) 软件包相对应

- ORM软件包：Ruby on Rails, Django, Hibernate, Sequelize等

6.1. ER 数据模型

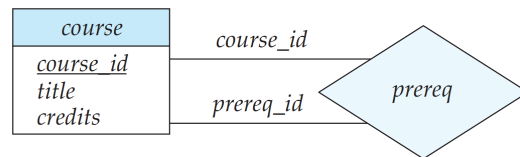
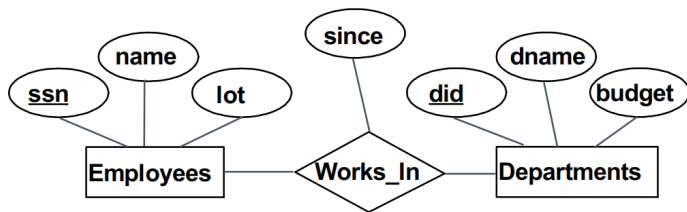
- 实体 (Entity)：现实世界中与其他对象区分开来的对象，由一组属性来描述
- 实体集 (Entity Set)：相似实体的集合
 - 实体集中的每个实体都具有相同的属性集（至少在我们考虑ISA层次结构之前）
 - 每个实体集都有一个key（主键？）
 - 每个属性都有一个域
 - 允许组合或多值属性



- 实体的表示
 - 矩形代表实体集
 - 实体集内列出的属性
 - 下划线表示主键属性

6.2. 关系 Relationship

- 关系：两个或多个实体之间的关联
 - 关系可以有属性
- 关系集 (Relationship Set)：类似关系的集合
 - 一个 n 元关系集 R 涉及 n 个实体集 $E_1 \dots E_n$
 - 每个 R 中的关系涉及实体 $e_1 \dots e_n$
 - 同一个实体集可以参与不同的关系集，或者在同一个关系集中担任不同“角色”
- ER图通过菱形代表关系集
- 关系集的度
 - 二元关系集：涉及两个实体集（度为2）——在数据库系统中，大多数关系集是二元关系集
 - 在多个实体集之间的关系集很少见
 - e.g. 学生可以在指导老师的指导下进行研究项目
 - 关系集proj_guide是指导老师、学生和项目之间的三元关系集。



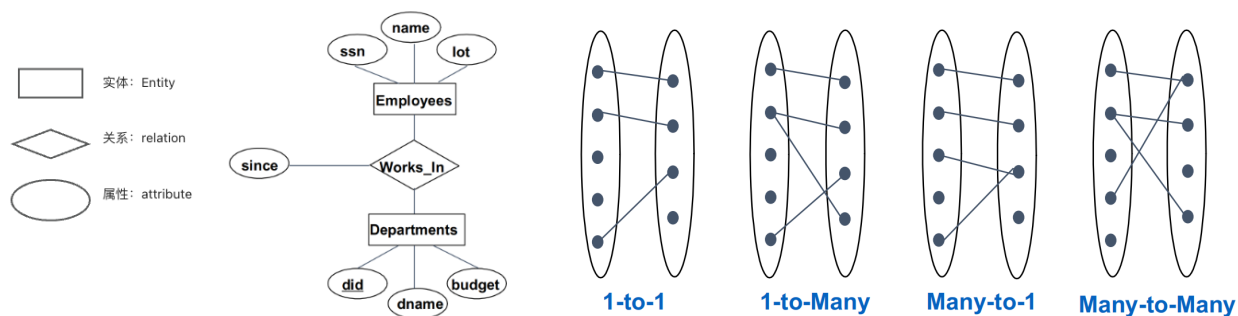
6.3. 角色 Roles

关系中的实体集可以是唯一的

- 实体集的每个实例在关系中扮演不同的“角色”
- 标签“course_id”和“prereq_id”被称为角色。

6.4. ER 图

- 概念模型：实体-关系，与实际DBMS无关。



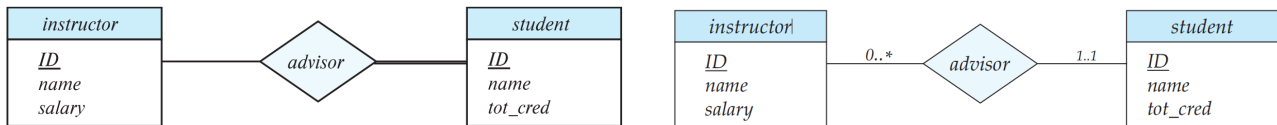
6.5. 基数比率约束 Cardinality Ratio Constraints (Mapping Cardinality Constraints)

- 关系可以区分为1: 1, 1: N 和 M: N
- 表示：
 - 在关系集和实体的集合之间画一条箭头线(→)来表示“一个”
 - 画一条无向线(—)来表示“许多”

6.6. 参与约束 Participation Constraints

- 总体参与（由双线表示）：实体集中的每个实体至少参与关系集中的一个关系。
 - 每个学生必须有一个导师
- 部分参与：某些实体可能不会参与任何关系中的关系设置

- 有的老师可以不参与指导



- 一条线可以有相关的最小和最大基数，以形式l..h表示，其中l是最小基数，h是最大基数。
 - 导师可以指导0个或多个学生，学生必须有一个导师，不能有多个导师。

6.7. ER 数据模型中的主键

6.7.1. 实体集中的主键

根据定义，每个实体是独一无二的

- 从数据库的角度来看，必须以属性来表达它们之间的差异
- 在一个实体集中，没有两个实体允许在所有属性上具有完全相同的值
- 主键是一个足以区分实体的属性集

6.7.2. 关系集中的主键

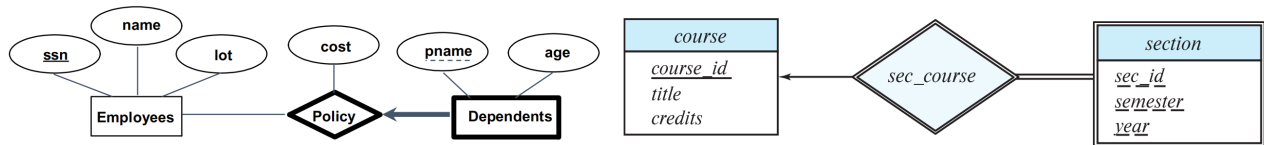
为了区分关系集中的各种关系，我们使用关系集中实体的各自主键

- 假设 R 是一个涉及实体集 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 的主键的并集构成
- 关系集中主键的选择取决于关系的映射基数。

6.8. 高级实体关系模型

6.8.1. 弱实体 Weak Entities

- 弱实体集是指其存在依赖于另一个实体（称为其识别实体）的实体集。
- 我们不用将主键与弱实体关联，而是使用识别实体以及被称为区分器的额外属性来唯一标识弱实体。
- 只有考虑另一个实体（所有者实体）的主键才能唯一地识别一个弱实体
 - 所有者实体集和弱实体集必须参与一个一对多关系集（一个所有者，多个弱实体）
 - 弱实体集必须在这个标识关系集中具有完全参与性。

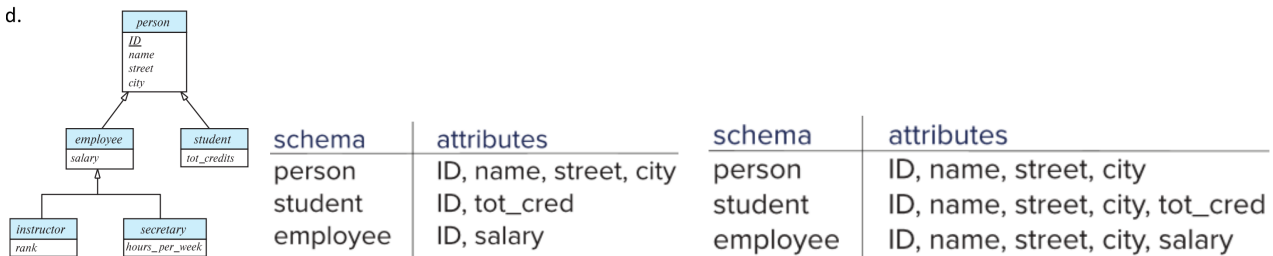


- 弱实体只有“部分密钥partial key”（虚线下划线）
 - 例如section主键为：course_id, sec_id, semester, year
- 弱实体用双矩形表示

6.8.2. 特殊化 Specialization

- 通过架构表达特化
 - 方法一：
 - 为上层实体构造一个模式
 - 为每个下层实体集构造一个模式，包括上层实体集的主键和局部属性
 - 缺点：获取员工信息需要访问两个关系，一个是与低级模式对应的，一个是与高级模式对应的
 - 方法二：
 - 为每个实体集构造一个模式，其中包含所有局部和继承属性
 - 缺点：对于既是学生又是雇员的人，姓名、街道和城市可能会被冗余存储

d.



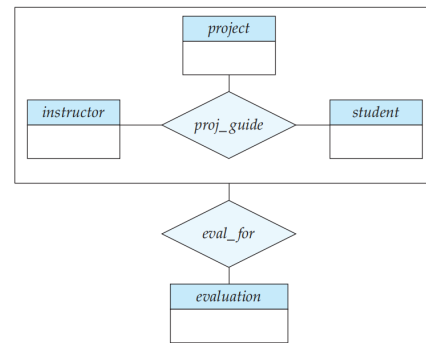
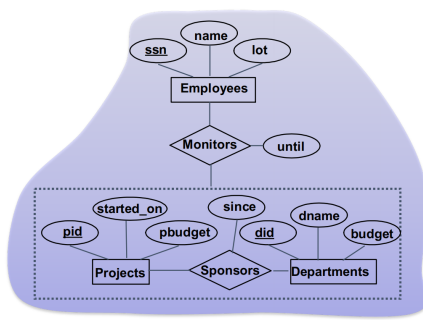
6.8.3. 泛化 Generalization

- 自下而上的设计过程——将具有相同特征的多个实体集组合成更高层次的实体集
- 专业化和一般化是彼此的简单倒置；它们在ER图中以相同的方式表示
- 专业化和一般化这两个术语可以互换使用

6.8.4. 聚合 Aggregation

- 允许我们将关系集合视为实体集合，以便参与（其他）关系
 - 通过聚合消除冗余而不引入冗余，以下图表表示：
 - 学生由特定导师指导进行特定项目

- 学生、导师和项目组合可能具有关联评估
- 允许关系具有关系

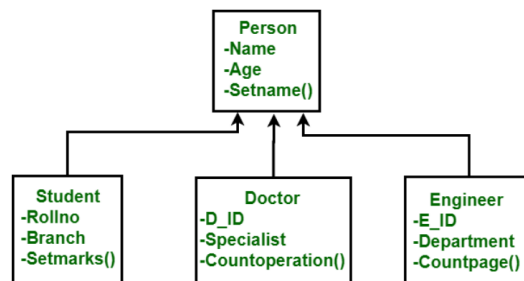


6.8.5. 类别 Category

- 允许我们表达实体集合由不同类型实体组成，即混合实体集合

7. 面向对象的数据模型 Object-Oriented Data Model

- 关系数据模型的不足
 - 违背1NF
 - 面向对象的分析和编程
 - 对象持久化的需求
- 对象关系DBMS：原生的面向对象DBMS



8. 其他数据模型

- 基于逻辑的数据模型（演绎DBMS） Logic-based data model (Deductive DBMS)
 - 扩展DBMS的查询功能（特别是递归查询recursive query功能）
 - 提高DBMS的演绎能力
- 实时数据模型 Temporal data model

- 空间数据模型 Spatial data model
- XML数据模型

9. 总结

- 数据模型是DBMS的核心
- 数据模型是一种在数据库中模拟现实世界的方法论
- 事实上，每种DBMS都实现了某种数据模型