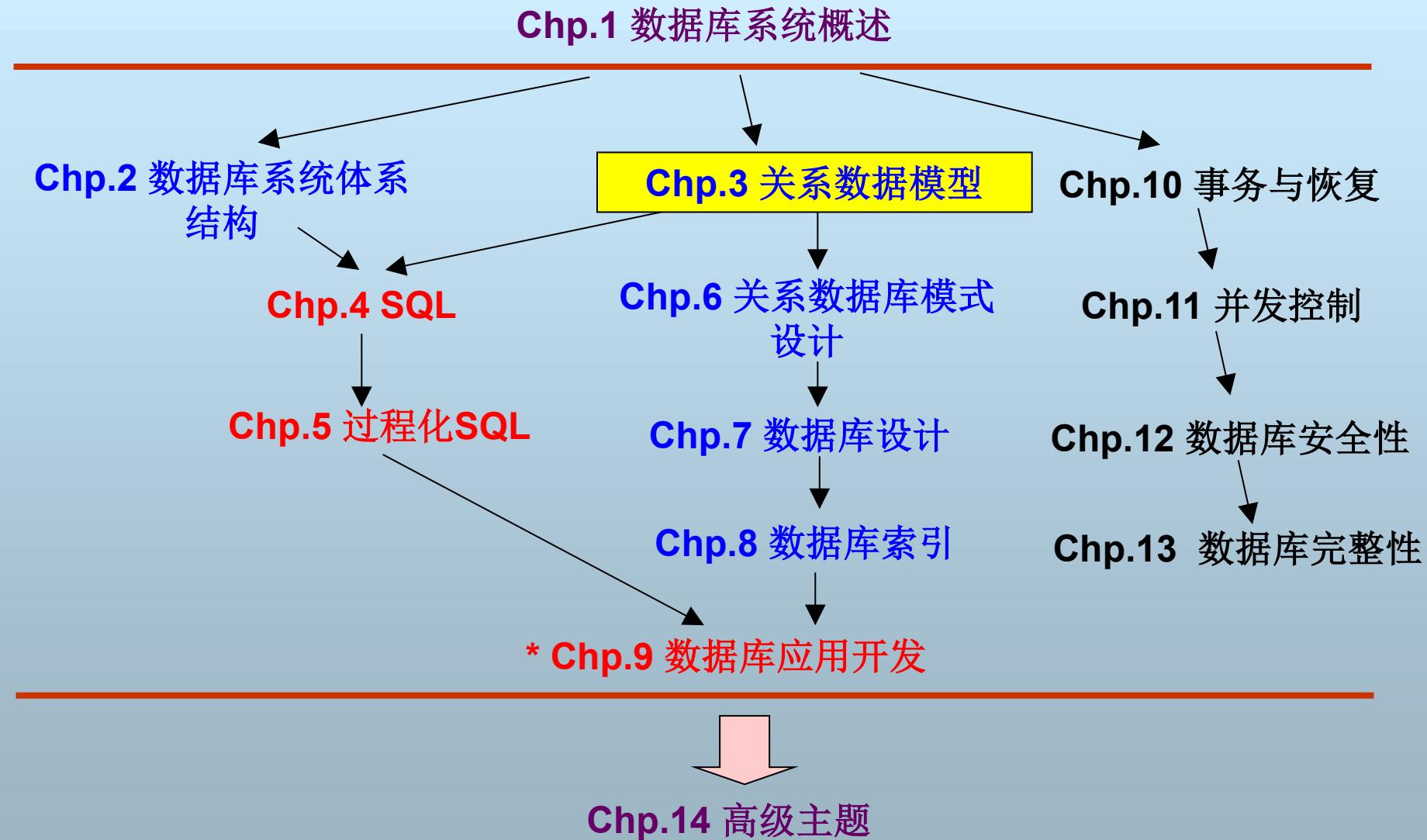


# 第3章 关系数据模型

# 课程知识结构



# 本章主要内容

- 数据模型的概念
- 关系数据模型的概念
- 关系数据模型的形式化定义
- 关系模型的三类完整性规则
- 关系代数

# 一、数据模型

- 使用数据库技术，首先必须把现实世界中的事物表示为计算机能够处理的数据
- 模型是对现实世界特征的抽象
  - 如建筑模型、沙盘模型等
- 数据模型是对现实世界**数据特征**的抽象
- 数据模型的定义
  - 描述现实世界**实体**、**实体间联系**以及**数据语义和一致性约束**的模型

# 1、数据模型的分类

## ■ 根据模型应用的不同目的

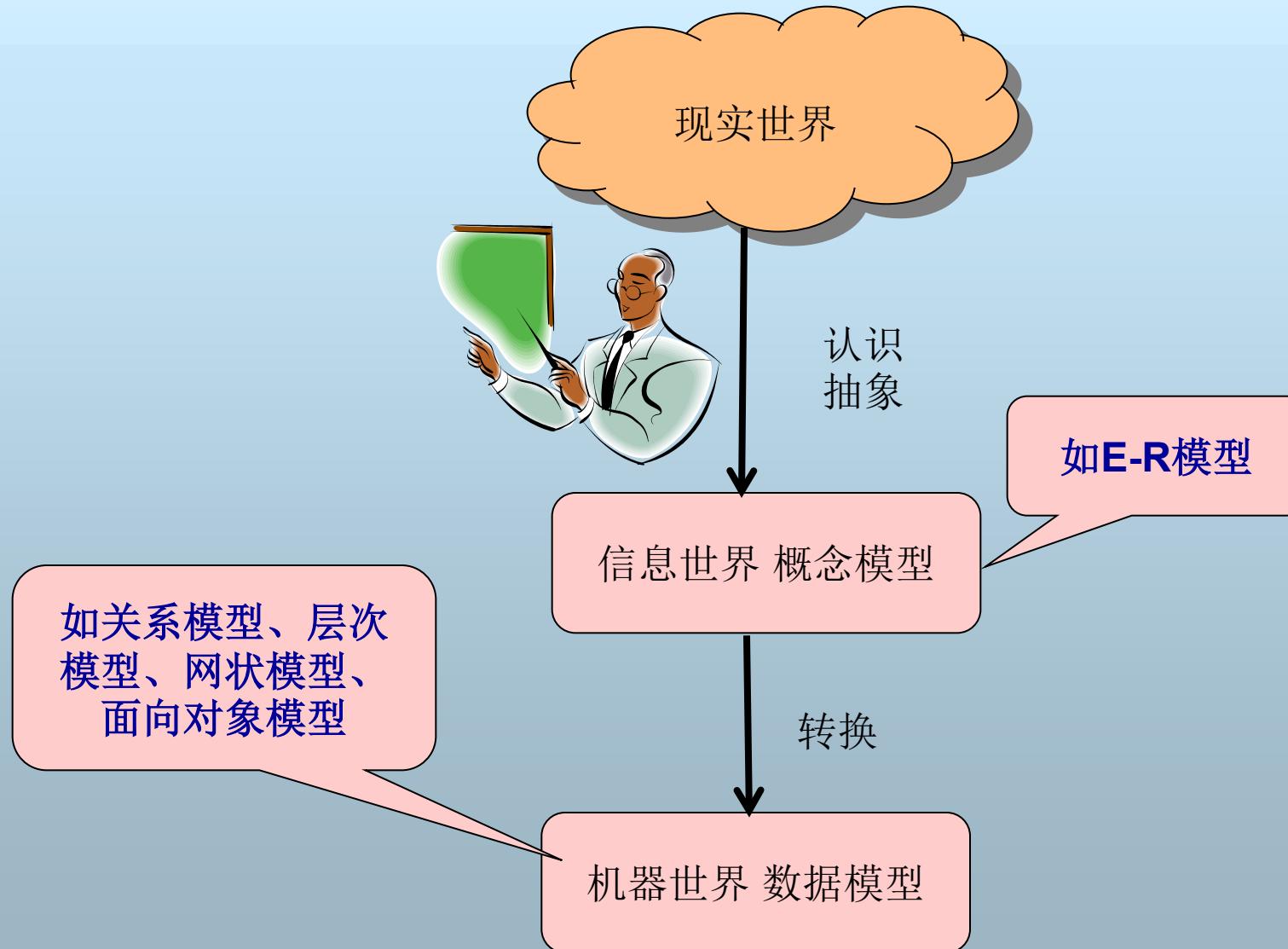
- 概念数据模型（概念模型）

- ◆ 按用户的观点对数据进行建模，强调语义表达功能
- ◆ 独立于计算机系统和DBMS
- ◆ 主要用于数据库的概念设计

- 结构数据模型（数据模型）

- ◆ 按计算机系统的观点对数据进行建模，直接面向数据库的逻辑结构
- ◆ 与计算机系统和DBMS相关
  - 回顾：DBMS支持某种数据模型
- ◆ 有严格的形式化定义，以便于在计算机系统中实现

## 2、数据抽象的层次



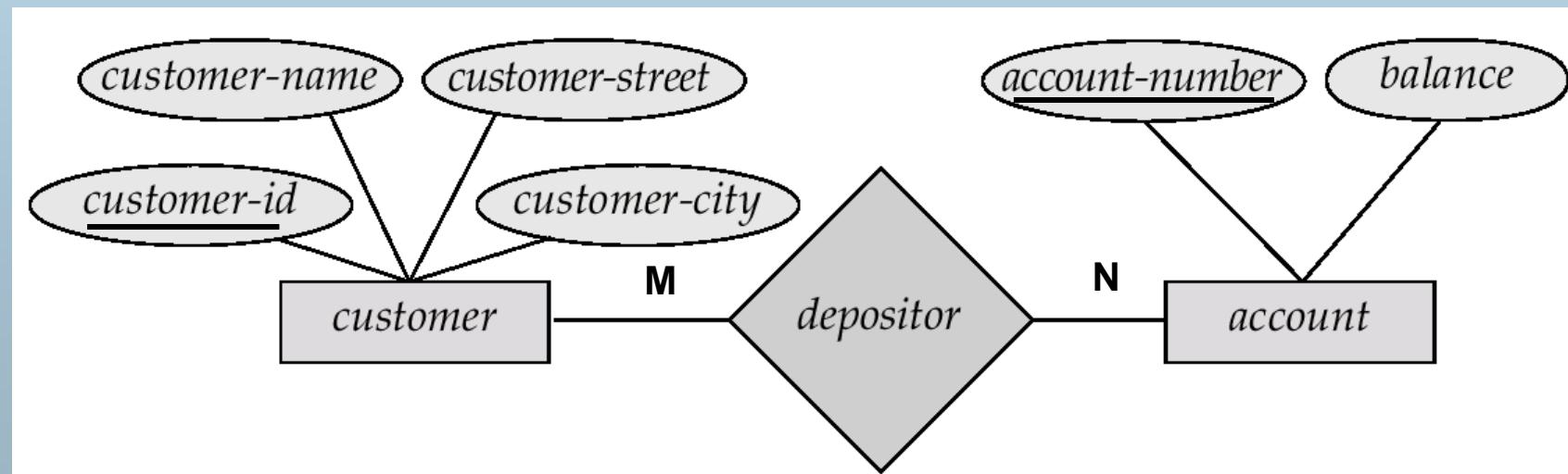
# 3、数据模型的例子

## ■ 现实世界

- 客户存款

## ■ 信息世界

- 概念模型（E-R模型）



# 3、数据模型的例子

## 机器世界

- 数据模型（关系模型）

customer-id	customer-name	customer-street	customer-city
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	account-number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

# 4、数据模型的三要素

## ■ 数据结构

- 现实世界实体及实体间联系的表示和实现

## ■ 数据操作

- 数据检索和更新的实现

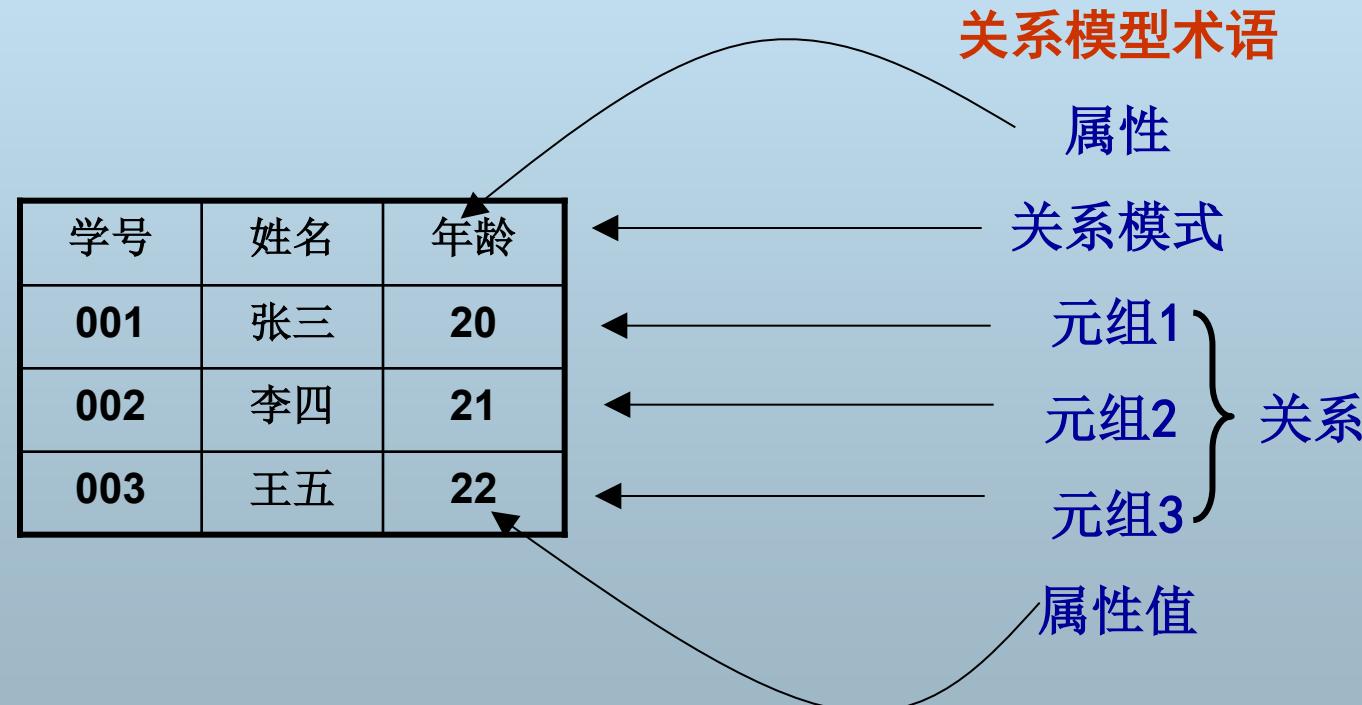
## ■ 数据的完整性约束

- 数据及数据间联系应具有的制约和依赖规则
  - ◆ 如：一个系可有多个学生，一个学生只属于一个系

## 二、关系模型概论

### ■ 关系模型

- 用二维表格结构表示实体集，外码表示实体间联系，三类完整性规则表示数据约束的数据模型



# 1、一些术语

## ■ 属性(Attribute)

- 二维表格的每一列称为关系的一个属性，列的数目称为度 (degree)

## ■ 元组(Tuple)

- 每一行称为关系的一个元组，元组的数目称为势或者基数 **cardinality**

## ■ 域(Domain)

- 一组具有相同数据类型的值的集合。每个属性有一个域

## ■ 关系(Relation)

- 元组的集合

[Guido Moerkotte: Cardinality Estimation for Having-Clauses.](#)  
[Proc. VLDB Endow. 18\(1\): 28-41 \(2024\)](#)



# 2、关系、关系模式与关系数据库

## ■ 关系模式(Relation Schema)

- 关系的逻辑结构和特征的描述
- 对应于二维表格的表头
- 通常由属性集和各属性域表示，不关心域时可省略域
  - ◆ Student(Name, Age, Class)

## ■ 关系

- 关系模式的实例，即二维表（元组的集合）

## ■ 关系数据库模式(Relational Database Schema)

- 关系模式的集合

## ■ 关系数据库：关系数据库模式的实例

# 3、关系模式的形式化定义

■ 关系模式可以形式化定义为：

- **R(U, D, dom, F)**

- ◆ R为关系模式名，U是一个属性集，D是U中属性的值所来自的域，Dom是属性向域的映射集合，F是属性间的依赖关系

■ 例：Student关系模式的定义

- **Student(U, D, dom, F)**

- ◆ U={sno, name, age}
  - ◆ D={CHAR, INT}
  - ◆ Dom={dom(sno)=dom(name)=CHAR, dom(age)=INT}
  - ◆ F={sno→name, sno→age}

■ 关系模式通常简写为R(U), R(A1,A2,...)或R

# 4、超码、候选码和主码

## ■ 超码（Super Key）

- 在关系模式中能唯一标识一个元组的属性集称为关系模式的超码

## ■ 候选码（Candidate Key）

- 不含多余属性的超码
- 包含在某个候选码中的属性称为主属性（Primary Attribute）
- 不包含在任何一个候选码中的属性称为非主属性（Non-prime Attribute）

## ■ 主码（Primary Key）

- 用户选作元组标识的一个候选码称为主码，其余的候选码称为替换码（Alternate Key）

# 4、超码、候选码和主码

## ■ Student(Sno, Name, Age, PassID)

- 超码

- ◆ (Sno, Name)
- ◆ (PassID, Name) ...

- 候选码

- ◆ Sno
- ◆ PassID

- 主码

- ◆ 若选Sno，则Sno为主码，PassID为替换码
- ◆ 若选PassID，则PassID为主码，Sno为替换码

# 5、关系的性质

## ■ 一个关系是一个规范化的二维表格

- 属性值不可分解

- ◆ 不允许表中有表

- 元组不可重复

- ◆ 因此一个关系模式至少存在一个候选码

- 没有行序，即元组之间无序

- ◆ 关系是元组的集合

- 没有列序，即属性之间无序

- ◆ 关系模式是属性的集合

学号	课程
001	数据库
002	{数据库,C语言}

更新二义性：若001现也选了C语言，则DBMS在更新时面临二义性：

- 1.修改第1个元组的课程
- 2.修改第2个元组的学号

# 6、关系模型的形式化定义

## ■ 数据结构

- 关系：数据库中全部数据及数据间联系都以关系来表示

## ■ 数据操作

### ● 关系运算

- ◆ 关系代数
- ◆ \*关系演算（元组关系演算、域关系演算）

## ■ 数据的完整性约束

- 关系模型的三类完整性规则

# 7、关系模型的三类完整性规则

## ■ 关系数据库的数据和操作必须遵循的规则

- 实体完整性(Entity Integrity)
- 参照完整性(Referential Integrity)
- 用户自定义完整性(User-Defined Integrity)

# (1) 实体完整性

- 关系模式R的主码不可为空
  - 指组成主码的所有属性均不可取空值

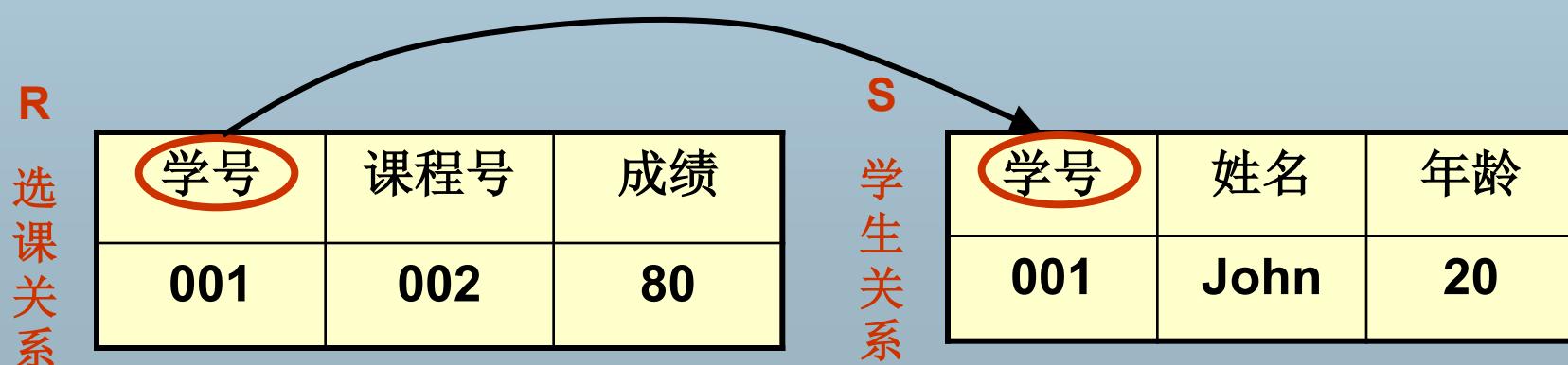


学号	课程号	成绩
S001	C001	80
S001		90
		80

## (2) 参照完整性

### ■ 外码 (Foreign Key)

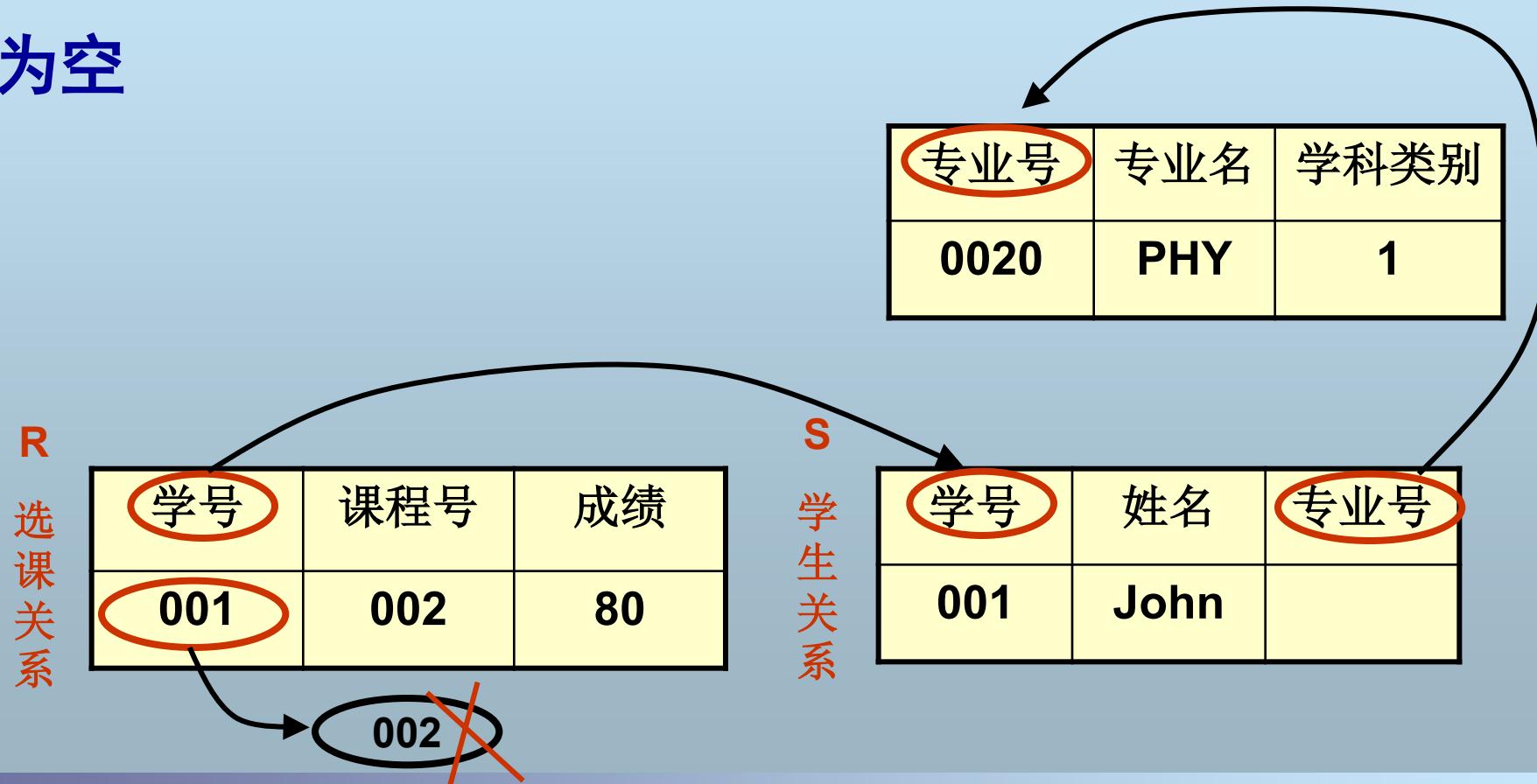
- 关系模式R的外码是它的一个属性集FK，满足：
  - ◆ 存在带有候选码CK的关系模式S，且
  - ◆ R的任一非空FK值都在S的CK中有一个相同的值
- S称为被参照关系 (Referenced Relation) , R称为参照关系 (Referential Relation)



## (2) 参照完整性

### ■ 参照关系R的任一个外码值必须

- 等于被参照关系S中所参照的候选码的某个值
- 或者为空



### (3) 用户自定义完整性

- 针对某一具体数据的约束条件，反映某一具体应用所涉及的数据必须满足的特殊语义
- 由应用环境决定

学号	课程号	成绩
001	002	80

成绩 $\geq 0$  and 成绩 $\leq 100$



# 回顾：关系模型的形式化定义

## ■ 数据结构

- 关系：数据库中全部数据及数据间联系都以关系来表示

## ■ 数据的完整性约束

- 关系模型的三类完整性规则：实体完整性、参照完整性、用户自定义完整性

## ■ 数据操作

- 关系运算

  - ◆ 关系代数

  - ◆ 关系演算（元组关系演算、域关系演算）

### 三、关系代数(Relational Algebra)

#### ■ Algebra= $\langle A, O \rangle$

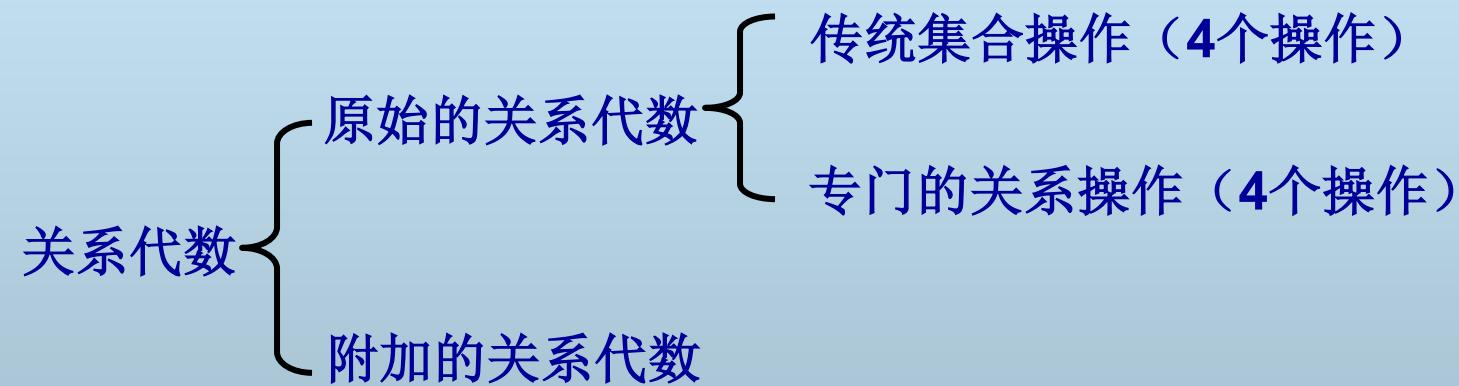
- $A$ : 一个类型的集合
- $O$ :  $A$ 上的代数运算集合，须满足在 $A$ 上的封闭性

#### ■ 关系代数： Relational Algebra= $\langle A, O \rangle$

- $A$ : 关系，关系代数中只存在一种类型，即关系
- $O$ : 关系代数运算，也称关系代数操作，以关系为运算对象的一组运算集合
- 满足封闭性：任何关系代数运算的结果仍是关系

# 三、关系代数(Relational Algebra)

## ■ 以集合运算为基础



# 1、一元操作和二元操作

## ■ 一元操作 (Unary Operation)

- 只有一个变元的代数操作
- 如选择、投影

## ■ 二元操作 (Binary Operation)

- 具有两个变元的代数操作
- 如并、交、差、笛卡儿积、连接、除

## 2、原始的关系代数

### ■ 传统的集合操作

- 并（Union）：返回两个关系中所有元组
- 交（Intersection）：返回两个关系共同的元组
- 差（Difference）：返回属于第一个关系但不属于第二个关系的元组
- 笛卡儿积（Cartesian Product）：返回两个关系的元组的任意组合所得到的元组集合

## 2、原始的关系代数

### ■ 专门的关系操作

- **选择 (Select)** : 返回指定关系中满足给定条件的元组
- **投影 (Project)** : 返回指定关系中去掉若干属性后所得的元组
- **连接 (Join)** : 从两个关系的笛卡儿积中选取属性间满足给定条件的元组
- **除 (Divide)** : 除的结果与第二个关系的笛卡儿积包含在第一个关系中

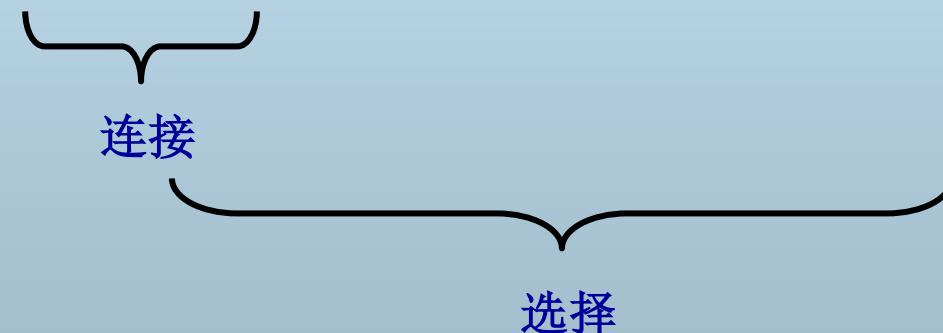
# 3、关系代数的封闭性

## ■ 关系代数的封闭性

- 任意关系代数操作的结果仍是一个关系

## ■ 关系代数的封闭性保证了关系代数操作的可嵌套性

- 例如：(S Join P) Where City='Athens'



# 4、关系代数表达式的语法

## ■ 数学符号表示

- 并  $\cup$ 、交  $\cap$ 、差  $-$ 、笛卡儿积  $\times$
- 选择  $\sigma$ 、投影  $\pi$ 、连接  $\bowtie$ 、除  $\div$

## ■ 英语关键字表示

- 并 **Union**、交 **Intersect**、差 **Minus**、笛卡儿积 **Times**
- 选择 **Where...**、投影 **{All But...}**、连接 **Join**、除 **Divideby**

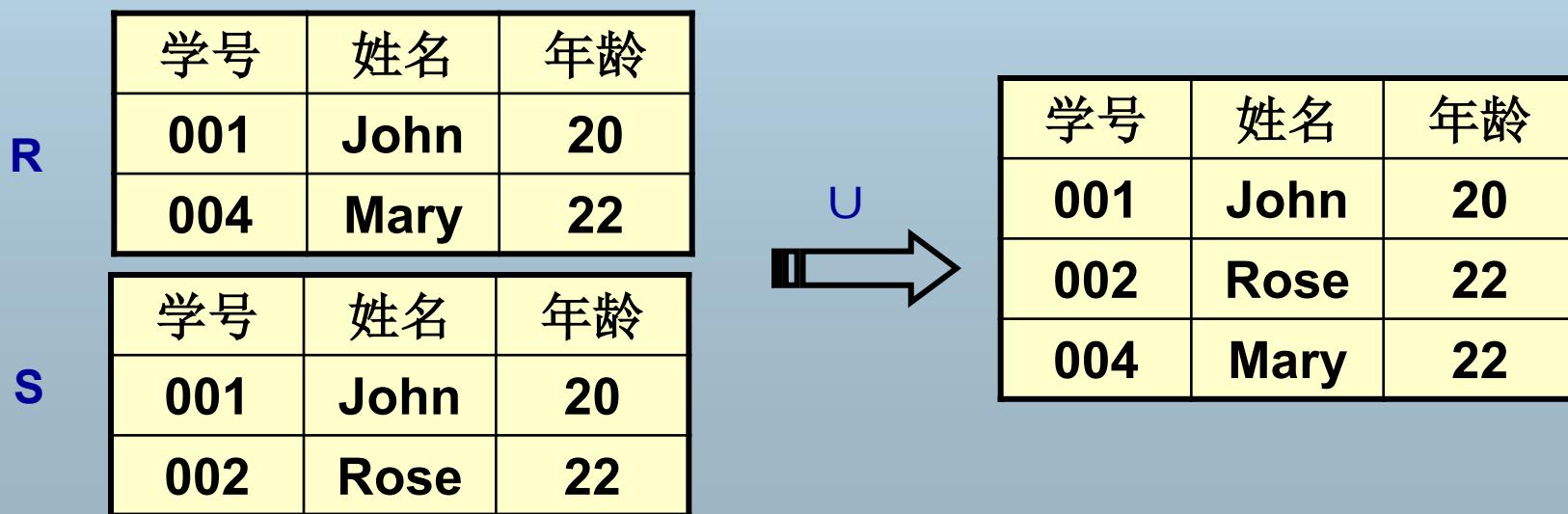
# 5、原始关系代数操作的语义

- 并
- 交
- 差
- 笛卡儿积（积）
- 选择
- 投影
- 连接
- 除

# (1) 并

$$R \cup S = \{ t | t \in R \vee t \in S \}$$

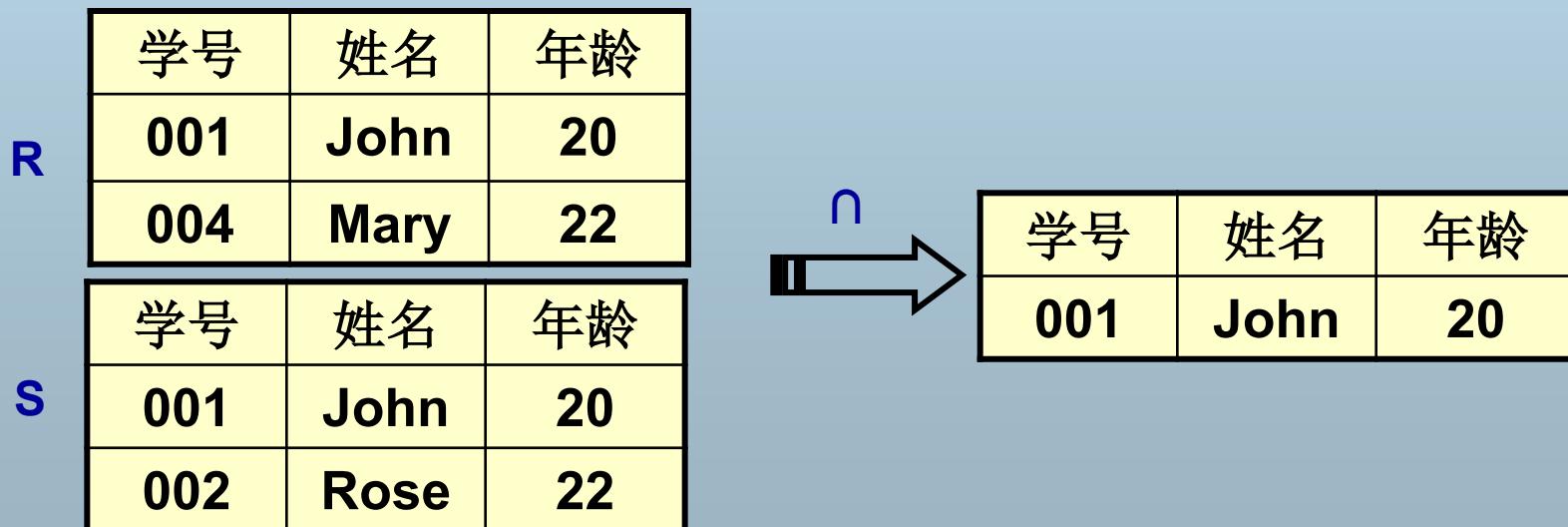
- **t是元组变量**
- **R和S是关系代数表达式**
- **R与S的degree必须相同**
- **R与S的类型必须相同**



## (2) 交

$$R \cap S = \{ t | t \in R \wedge t \in S \}$$

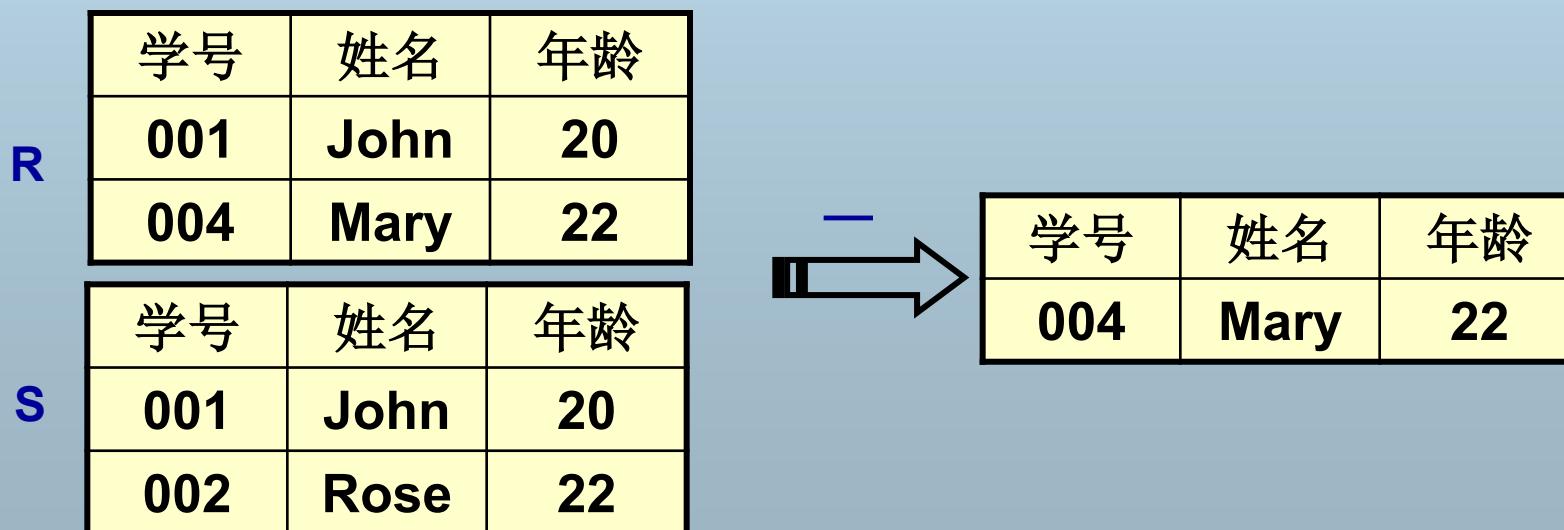
- $t$ 是元组变量
- $R$ 和 $S$ 是关系代数表达式
- $R$ 与 $S$ 的degree必须相同
- $R$ 与 $S$ 的类型必须相同



### (3) 差

$$R - S = \{ t | t \in R \wedge t \notin S \}$$

- **t是元组变量**
- **R和S是关系代数表达式**
- **R与S的degree必须相同**
- **R与S的类型必须相同**



# (4) 积

■  $R \times S = \{ t | t = \langle t^r, t^s \rangle \wedge t^r \in R \wedge t^s \in S \}$

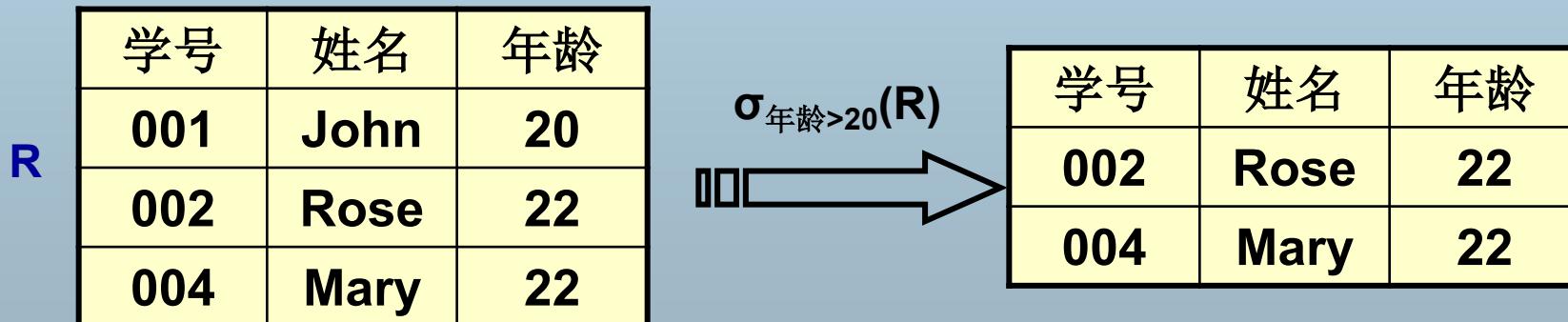


R			S			R × S					
学号	姓名	年龄	学号	姓名	年龄	R.学号	R.姓名	R.年龄	S.学号	S.姓名	S.年龄
001	John	20	001	John	20	001	John	20	001	John	20
004	Mary	22	002	Rose	22	004	Mary	22	002	Rose	22

# (5) 选择

$$\sigma_F(R) = \{ t | t \in R \wedge F(t) = \text{TRUE} \}$$

- 水平划分关系
- $F$ 是一个逻辑表达式，表示所选的元组应满足的条件
- $F$ 由逻辑运算符 $\neg$ (NOT)、 $\wedge$ (AND)、 $\vee$ (OR)连接算术表达式构成
  - 算术表达式形为 $X \theta Y$ ,  $\theta$ 可以是 $>$ ,  $<$ ,  $=$ ,  $\leq$ ,  $\geq$ 或 $\neq$ ,  $X$ 和 $Y$ 可以是属性名、常量或简单函数



## (5) 选择：一些例子

### ■ 设有学生关系

- 学生（学号，姓名，年龄，性别，系名）

### ■ 查询‘计算机系’的男学生

- $\sigma_{\text{系名} = \text{'计算机系'} \wedge \text{性别} = \text{'男'}}$  (学生)

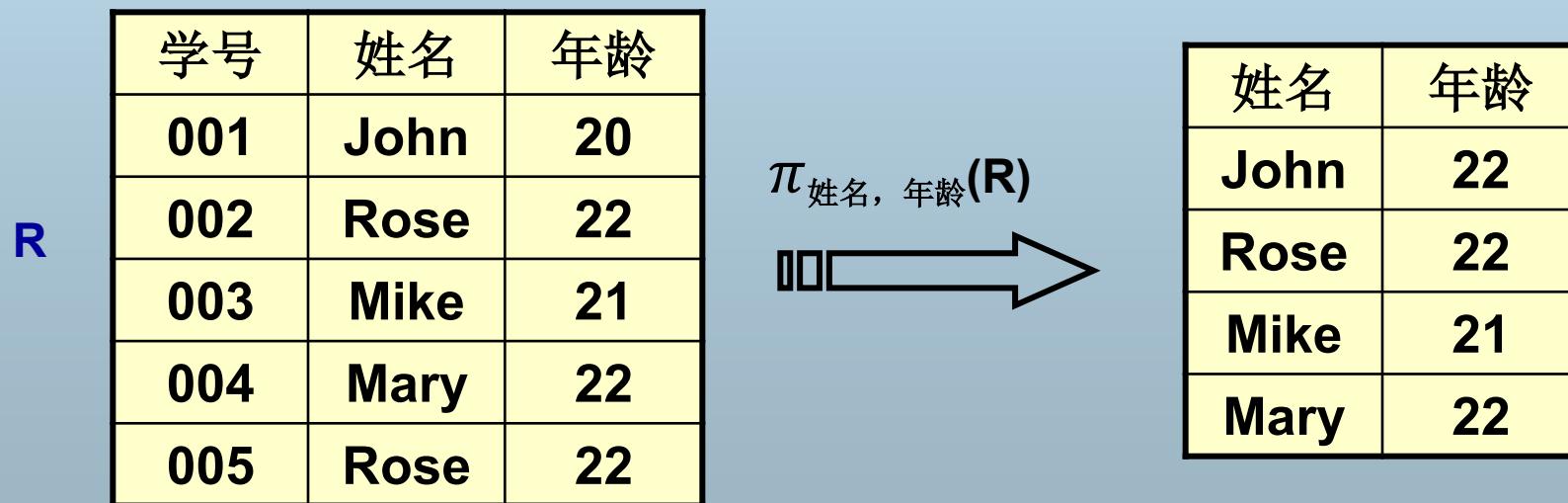
### ■ 查询年龄为20或21岁的学生

- $\sigma_{\text{年龄} = 20 \vee \text{年龄} = 21}$  (学生)

# (6) 投影

■  $\pi_A(R) = \{ t[A] | t \in R \}$ , 其中A是R的属性子集

- 垂直划分关系, 选取若干列所构成的关系
- A中的属性不可重复



# (7) 连接

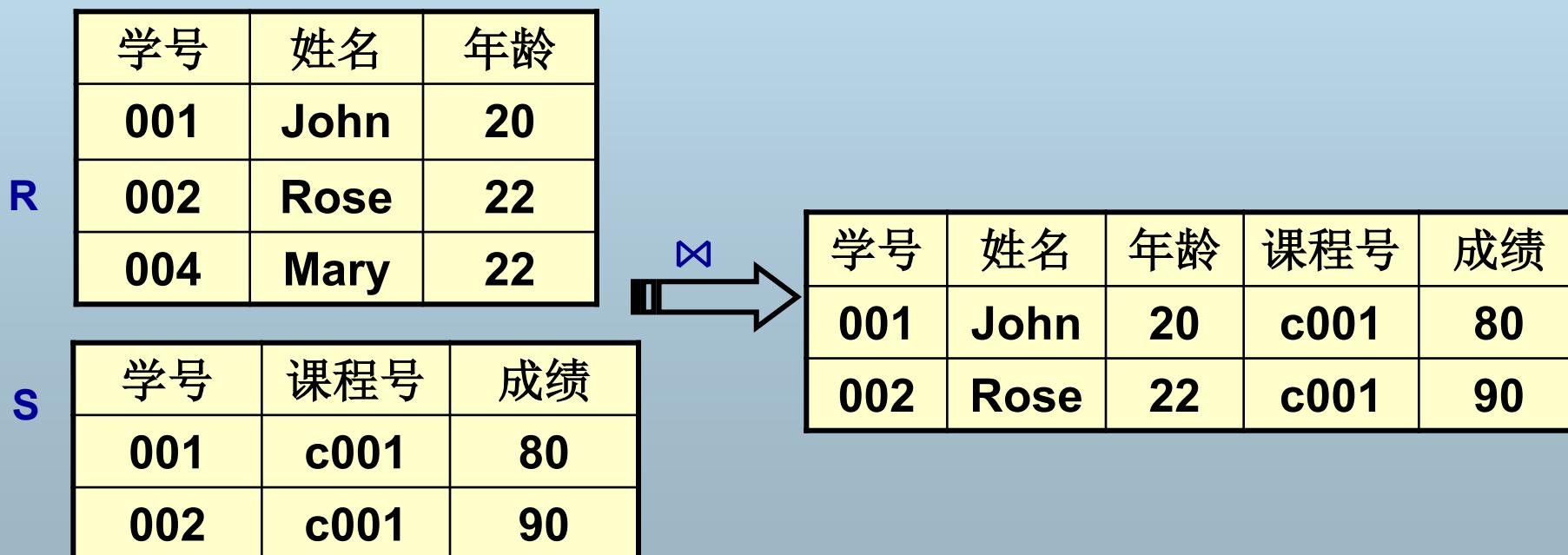
- 自然连接
- $\theta$  连接
- 等值连接

## (7) 连接：自然连接

■ 设R的属性集为{X,Y}， S的属性集为{Y,Z}

■  $R \bowtie S = \{t \mid t = \langle X, Y, Z \rangle \wedge t[X, Y] \in R \wedge t[Y, Z] \in S\}$

- 相当于在 $R \times S$ 中选取R和S的所有公共属性值都相等的元组，并在结果中去掉重复属性

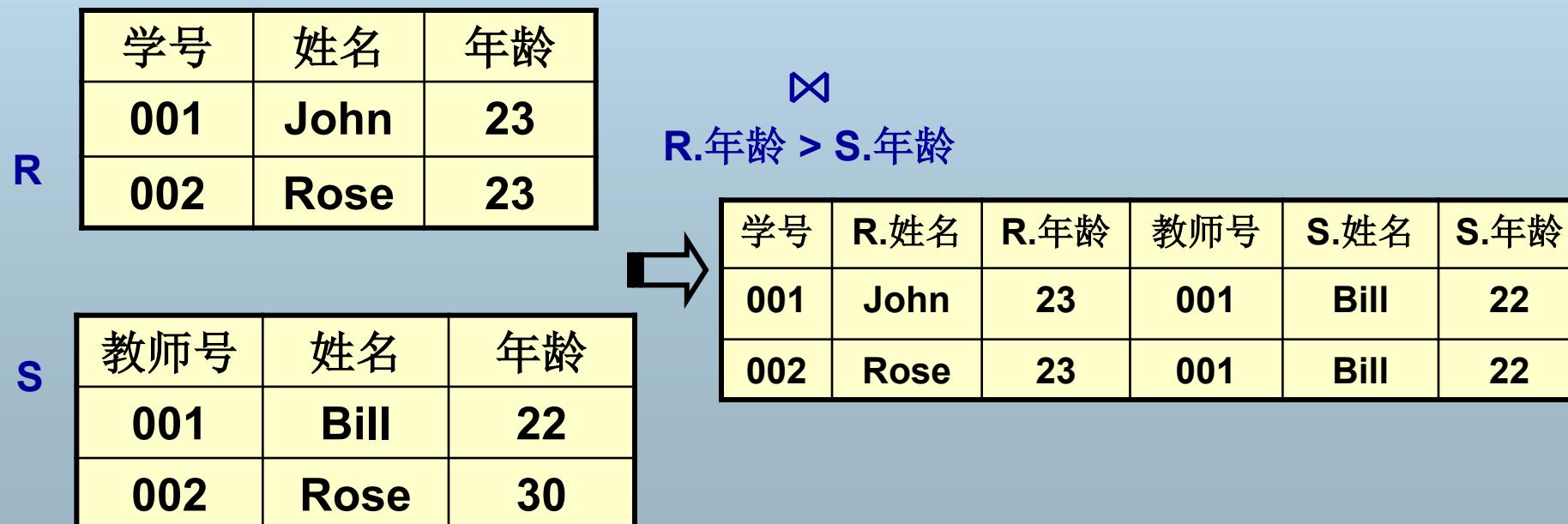


## (8) 连接: $\theta$ 连接

■ 设R的属性集为{X,Y}, S的属性集为{Y,Z}

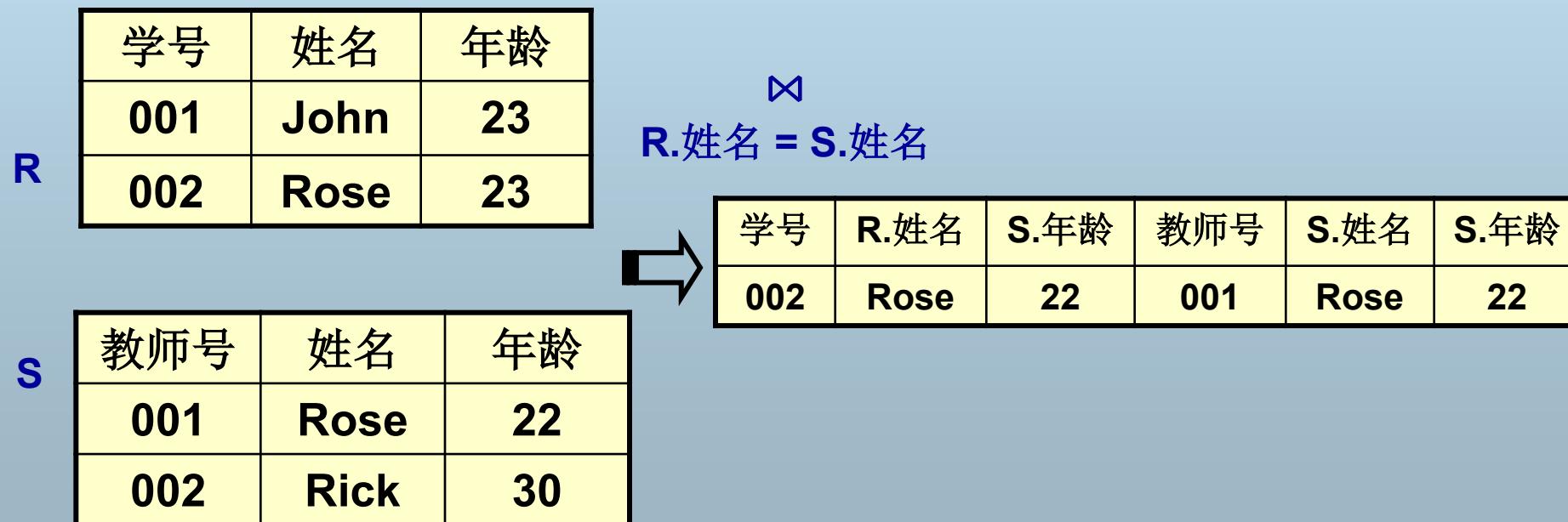
■  $R \bowtie_{A \theta B} S = \{t | t = \langle t^r, t^s \rangle \wedge t^r \in R \wedge t^s \in S \wedge t^r[A] \theta t^s[B]\}$

- 相当于在 $R \times S$ 中选取R的属性A值与S的属性B值满足比较关系  $\theta$  的元组。



# (9) 连接：等值连接 (equijoin)

- 在 $\theta$ 连接中，当 $\theta$ 为等号时，称“等值连接”
- 等值连接是 $\theta$ 连接中比较常见的形式



# (10) 除 (divide)

- 设关系R的属性集为{X,Y}, S的属性集为{Y}, 则 $R \div S$ 的结果是一个关系P, P的属性集为{X}, 并且 $P \times S$ 包含在R中
  - P的degree=R的degree-S的degree
- $R \div S$ 的计算方法
  - $T = \pi_X(R)$
  - $W = (T \times S) - R$
  - $V = \pi_X(W)$
  - $R \div S = T - V$

计算{X}和{Y}的积中不包含在R中的{X}, T减去这部分{X}, 剩下的就是{X}×{Y}包含在R中的{X}, 就是最终的结果

# (10) 除：例子

## ■ 学生选课关系：

- R (sno, cno, score)

sno	cno	score
PB00001001	c001	80
PB00001001	c002	80
PB00001002	c001	80
PB00001002	c002	90

## ■ 求选了PB00001001所选全部课程的学生学号

- PB00001001所选全部课程

◆  $\pi_{cno}(\sigma_{sno='PB00001001'}(R))$

cno
c001
c002

- 选了PB00001001所选全部课程的学生学号

◆  $\pi_{sno, cno}(R) \div \pi_{cno}(\sigma_{sno='PB00001001'}(R))$

sno
PB00001001
PB00001002

# 补充：重命名操作 (Rename)

- 数据库中的关系都具有名字，但关系代数表达式的结果没有可供引用的名字
- 引入重命名操作使关系代数表达式的结果可以方便地在其它位置引用
- 定义
  - $\rho_x(E)$ : 将关系代数表达式E重命名为X
  - $\rho_{x(A_1, A_2, \dots, A_n)}(E)$ : 将关系代数表达式E重命名为X，并且各属性更名为A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>n</sub>
  - 例: 求每个学生的学号, 姓名, 课程号和成绩  
$$\pi_{R.sno, R.name, S.cno, S.score}(\rho_{R(sno, name, age)}(\text{学生}) \bowtie \rho_{S(sno, cno, score)}(\text{选课}))$$

# 6、（原始）关系代数的基本运算

■ 基本运算有5个

- 并、差、积、选择、投影
- 重命名\*

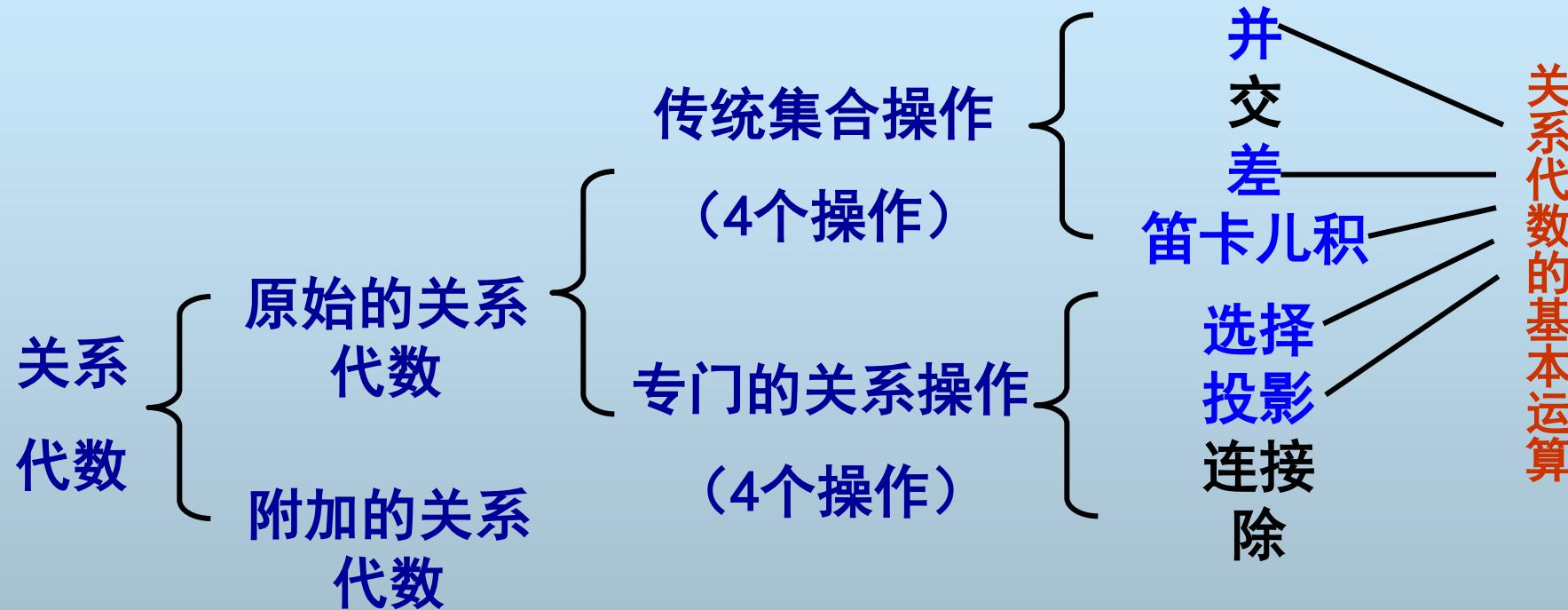
■ 其它操作都可以通过这些基本操作来表示

- 交：
- 连接（自然连接、 $\theta$ 连接）：
- 除：

# 7、关系代数表达式的定义

- 关系代数中的基本表达式是关系代数表达式，基本表达式由如下之一构成：
  - 数据库中的一个关系
  - 一个常量关系
- 设E1和E2是关系代数表达式，则下面的都是关系代数表达式：
  - $E1 \cup E2$ 、 $E1 - E2$ 、 $E1 \times E2$
  - $\sigma_P(E1)$ , 其中P是E1中属性上的谓词
  - $\pi_S(E1)$ , 其中S是E1中某些属性的列表
  - $\rho_x(E1)$ , 其中x是E1结果的新名字

# 回顾：关系代数



# 8、关系代数表达式示例

## ■ 供应商和零件数据库

- **Supplier:**
  - ◆ **S (S#, SNAME, STATUS, CITY)**
- **Part:**
  - ◆ **P (P#, PNAME, COLOR, WEIGHT)**
- **Supply:**
  - ◆ **SP (S#, P#, QTY)**, 其中 **S#** 参照 **S.S#**, **P#** 参照 **P.P#**. 都是外码

# (1) 简单表达式

■ 求London城市中的供应商的全部信息

- $\sigma_{CITY='London'}(S)$

■ 求London城市中的供应商的供应商号,名称和状态

- $\pi_{S\#, SNAME, STATUS}(\sigma_{CITY='London'}(S))$

■ 求红色并且重量不超过15的零件号和零件名

- $\pi_{P\#, PNAME}(\sigma_{COLOR='Red' \wedge WEIGHT <= 15}(P))$

## (2)复杂查询

■ 求提供零件P2的供应商名称

- $\pi_{SNAME}(\sigma_{P\#=P2}(S \bowtie SP))$

■ 求提供红色零件的供应商名称

- $\pi_{SNAME}(\sigma_{COLOR='Red'}(S \bowtie SP \bowtie P))$
- $\pi_{SNAME}(S \bowtie SP \bowtie (\sigma_{COLOR='Red'}(P)))$

■ 求住在同一城市的供应商号码对

- $\pi_{S1\#, S2\#}(\sigma_{S1.S1\# < S2.S2\#}(\rho_{S1(S1\#, CITY)}(\pi_{S\#, CITY}(S)) \bowtie \rho_{S2(S2\#, CITY)}(\pi_{S\#, CITY}(S))))$

## (3)复杂查询

■ 求提供所有零件的供应商名称

- $\pi_{SNAME}(S \bowtie (\pi_{S\#, P\#}(SP) \div \pi_{P\#}(P)))$

■ 求提供了S2提供的所有零件的供应商名称

- $\pi_{SNAME}(S \bowtie (\pi_{S\#, P\#}(SP) \div \pi_{P\#}(\sigma_{S\#=‘S2’}(SP))))$

■ 求不提供零件P2的供应商名称

- $\pi_{SNAME}(S) - \pi_{SNAME}(\sigma_{P\#=‘P2’}(S \bowtie SP))$

# 9、附加的关系代数操作

- 扩展投影（广义投影）
- 聚集函数
- 分组
- 排序
- 赋值

# (1) 扩展投影(Extended Project)

- 在传统的投影操作  $\pi_L(R)$  当中，  $L$  是  $R$  的一些属性的集合
- 扩展投影仍用  $\pi_L(R)$  来表示投影操作， 其中， 投影列表可以是以下所列出的元素之一：
  - $R$  的一个属性
  - 形如  $x \rightarrow y$  的表达式， 其中，  $x$  和  $y$  都是属性的名字。元素  $x \rightarrow y$  表示把  $R$  中的  $x$  属性取来并重命名为  $y$ 。
  - 形如  $E \rightarrow z$  的表达式， 其中  $E$  是一个涉及  $R$  的属性、常量、代数运算符或者串运算符的表达式。 $z$  是表达式  $E$  得到的结果属性的新名字
    - ◆  $a+b \rightarrow x$  作为一个列表元素表示  $a$  和  $b$  属性的和，并被重命名为  $x$ 。元素  $c||d \rightarrow e$  表示连接串类型的属性  $c$  和  $d$ ，并重命名为  $e$ 。

# (1) 扩展投影例子

## ■ Employee(E#, FirstName, LastName, salary)

- $\pi_{E\#, FirstName \parallel LastName \rightarrow Name}(Employee)$
- $\pi_{E\#, salary * 0.95 \rightarrow Taxed\_salary}(Employee)$
- $\pi_{E\# \rightarrow EmployeeNum}(Employee)$

## (2)聚集函数(aggregate)

■ 聚集函数输入一个值的集合，返回一个单一值

- **SUM** (汇总)
- **COUNT** (计数)
- **AVG** (求均值)
- **MAX** (求最大值)
- **MIN** (求最小值)
  - ◆  $\pi \text{ count(E#)} \rightarrow \text{employee\_count}(\text{Employee})$
  - ◆  $\pi \text{ AVG(salary)} \rightarrow \text{avg\_salary}(\text{Employee})$

## (3)分组 (group)

- 通常，人们不仅希望对简单的一整列求平均值或者是其他的聚集操作。还需要按照其他某一属性分组，然后考虑各分组中元组的聚集操作。
- $\gamma_L(R)$ 
  - $L$ 中只能包括两类对象：
    - ◆ 分组属性： $\gamma$ 操作依据某个属性的值将 $R$ 分组。这个用来分组的属性就被称为是分组属性。
    - ◆ 聚集函数：即应用到关系的某个属性上的聚集操作。作为聚集函数参数的属性被称作是聚集属性。

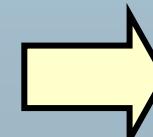
# (3)分组 (group)

■  $\gamma_L(R)$ 返回的关系是这样产生的:

- 把关系R的元组分组。每一组由具有L中分组属性上具有特定值的所有元组构成。
- 对于每一组，产生一个如下内容的元组：
  - ◆ 该组的分组属性值。
  - ◆ 该组中所有元组对列表L的属性聚集操作的结果

学号	课程号	成绩
001	c001	80
002	c001	90
001	c002	80
.....	.....	.....

$\gamma_{\text{课程号}, \text{AVG}(\text{成绩}) \rightarrow \text{平均成绩}}(R)$



课程号	平均成绩
c001	80
c002	85
.....	.....

## (4)排序(sort)

### ■ $\tau_L(R)$

- 返回按属性列表L排序的关系R，结果中的所有元组是按照L排序。如果L是 $A_1, A_2, \dots, A_n$ ，那么R的元组就先按属性 $A_1$ 的值排序，对于 $A_1$ 属性相等的元组，就按 $A_2$ 的值排序，依此类推

### ■ 例 对学生按年龄排序

- $\tau_{\text{年龄}}(\text{学生})$

# (5) 赋值

- 通过赋值操作可以给临时关系变量赋值，使关系代数表达式可以分开写
- $R \leftarrow E$ : 把关系代数表达式E的结果赋给R
- 例：求选了PB00001001所选全部课程的学生学号
  - PB00001001所选全部课程
    - ◆  $\text{Temp1} \leftarrow \pi_{\text{cno}}(\sigma_{\text{sno} = 'PB00001001'}(R))$
  - 全部的选课信息
    - ◆  $\text{Temp2} \leftarrow \pi_{\text{sno, cno}}(R)$
  - 选了PB00001001所选全部课程的学生学号
    - ◆  $\text{Result} = \text{Temp2} \div \text{Temp1}$

赋值操作并不把结果显示给用户，最后一句表示表达式结果被作为结果显示

# 四、数据更新

- 数据更新仍通过关系代数实现
- 删除
  - $R \leftarrow R - E$ :  $R$ 是关系,  $E$ 是关系代数查询
  - 例 “从数据库中删除姓名为‘Rose’的学生”
    - ◆  $\text{Student} \leftarrow \text{Student} - \sigma_{\text{name} = 'Rose'}(\text{Student})$

# 五、数据更新

## ■ 插入

- **R $\leftarrow R \cup E$** : R是关系, E是关系代数表达式
- 如果E是常量关系, 则可以插入单个元组
- 例1: 设关系S1和S2分别存放本科生数据和研究生数据, “把所有的本科生都全部升级为研究生”
  - ◆ **S2 $\leftarrow S2 \cup S1$**
- 例2: 插入一个新的本科生
  - ◆ **S1 $\leftarrow S1 \cup \{ ('001', 'Rose', 19) \}$**

# 五、数据更新

## ■ 修改

- $R \leftarrow \pi_{F_1, F_2, \dots, F_n}(R)$ : 通过广义投影实现
- $F_i$ : 当第*i*个属性没有被修改时是*R*的第*i*个属性; 当被修改时是第*i*个属性和一个常量的表达式
- 例1: “将每个学生的学号前加上字母S”
  - ◆  $\text{Student} \leftarrow \pi_{\cdot S'} \parallel sno, name, sex, age(\text{Student})$
- 如果只想修改*R*中的部分元组, 可以用下式
- $R \leftarrow \pi_{F_1, F_2, \dots, F_n}(\sigma_p(R)) \cup (R - \sigma_p(R))$
- 例2: “将所有男学生的学号前加上字母M”
  - ◆  $\text{Student} \leftarrow \pi_{\cdot M'} \parallel sno, name, sex, age(\sigma_{sex='M'}(\text{Student})) \cup (\text{Student} - \sigma_{sex='M'}(\text{Student}))$

# 本章小结

- 关系数据模型概念
- 关系数据模型的形式化定义
- 关系模型的三类完整性规则
- 关系代数