

# 数据库系统概论

An Introduction to Database System

## 第二章 关系数据库

xx大学信息学院

# 关系数据库简介

❖ 提出关系模型的是美国IBM公司的E.F.Codd

■ 1970年提出关系数据模型

**E.F.Codd, “A Relational Model of Data for Large Shared Data Banks”, 《Communication of the ACM》, 1970**

■ 之后，提出了关系代数和关系演算的概念

■ 1972年提出了关系的第一、第二、第三范式

■ 1974年提出了关系的BC范式



# 第二章 关系数据库

**2.1 关系数据结构及形式化定义**

**2.2 关系操作**

**2.3 关系的完整性**

**2.4 关系代数**

**2.5 \*关系演算**

**2.6 小结**



# 2.1 关系数据结构及形式化定义

## 2.1.1 关系

## 2.1.2 关系模式

## 2.1.3 关系数据库

## 2.1.4 关系模型的存储结构



## 2.1.1 关系

- ❖ 单一的数据结构----关系

现实世界的实体以及实体间的各种联系均用关系来表示

- ❖ 逻辑结构----二维表

从用户角度, 关系模型中数据的逻辑结构是一张二维表

- ❖ 建立在集合代数的基础上



# 关系 (续)

1. 域 (Domain)
2. 笛卡尔积 (Cartesian Product)
3. 关系 (Relation)



# 1. 域 (Domain)

◆ 域是一组具有相同数据类型的值的集合。例：

- 整数
- 实数
- 介于某个取值范围的整数
- 指定长度的字符串集合
- {‘男’， ‘女’ }
- .....



## 2. 笛卡尔积 (Cartesian Product)

### ❖ 笛卡尔积

给定一组域  $D_1, D_2, \dots, D_n$ , 允许其中某些域是相同的。

$D_1, D_2, \dots, D_n$  的笛卡尔积为:

$$D_1 \times D_2 \times \dots \times D_n =$$

$$\{ (d_1, d_2, \dots, d_n) \mid d_i \in D_i, i=1, 2, \dots, n \}$$

■ 所有域的所有取值的一个组合

■ 不能重复



# 笛卡尔积（续）

## ❖ 元组 (Tuple)

- 笛卡尔积中每一个元素 ( $d_1, d_2, \dots, d_n$ ) 叫作一个  $n$  元组 ( $n$ -tuple) 或简称元组
- (张清玫, 计算机专业, 李勇)、
- (张清玫, 计算机专业, 刘晨) 等都是元组

## ❖ 分量 (Component)

- 笛卡尔积元素 ( $d_1, d_2, \dots, d_n$ ) 中的每一个值  $d_i$  叫作一个分量
- 张清玫、计算机专业、李勇、刘晨等都是分量



# 笛卡尔积（续）

## ❖ 基数 (Cardinal number)

- 若  $D_i$  ( $i=1, 2, \dots, n$ ) 为有限集, 其基数为  $m_i$  ( $i=1, 2, \dots, n$ ), 则  $D_1 \times D_2 \times \dots \times D_n$  的基数  $M$  为:

$$M = \prod_{i=1}^n m_i$$

## ❖ 笛卡尔积的表示方法

- 笛卡尔积可表示为一张二维表
- 表中的每行对应一个元组, 表中的每列对应一个域

# 笛卡尔积（续）

例如，给出3个域：

- ❖ **D1=导师集合SUPERVISOR=** {张清致, 刘逸}
- ❖ **D2=专业集合SPECIALITY=** {计算机专业, 信息专业}
- ❖ **D3=研究生集合POSTGRADUATE=** {李勇, 刘晨, 王敏}
- ❖ D1, D2, D3的笛卡尔积为



# 笛卡尔积（续）

❖  $D1 \times D2 \times D3 = \{$

(张清玫, 计算机专业, 李勇), (张清玫, 计算机专业, 刘晨),  
(张清玫, 计算机专业, 王敏), (张清玫, 信息专业, 李勇),  
(张清玫, 信息专业, 刘晨), (张清玫, 信息专业, 王敏),  
(刘逸, 计算机专业, 李勇), (刘逸, 计算机专业, 刘晨),  
(刘逸, 计算机专业, 王敏), (刘逸, 信息专业, 李勇),  
(刘逸, 信息专业, 刘晨), (刘逸, 信息专业, 王敏) }

❖ 基数为  $2 \times 2 \times 3 = 12$



# 笛卡尔积（续）

表 2.1  $D_1, D_2, D_3$  的笛卡尔积

SUPERVISOR	SPECIALITY	POSTGRADUATE
张清玫	计算机专业	李勇
张清玫	计算机专业	刘晨
张清玫	计算机专业	王敏
张清玫	信息专业	李勇
张清玫	信息专业	刘晨
张清玫	信息专业	王敏
刘逸	计算机专业	李勇
刘逸	计算机专业	刘晨
刘逸	计算机专业	王敏
刘逸	信息专业	李勇
刘逸	信息专业	刘晨
刘逸	信息专业	王敏



# 3. 关系 (Relation)

## (1) 关系

$D_1 \times D_2 \times \dots \times D_n$  的子集叫作在域  $D_1, D_2, \dots, D_n$  上的关系, 表示为

$R (D_1, D_2, \dots, D_n)$

- $R$ : 关系名
- $n$ : 关系的目或度 (Degree)



# 关系 (续)

## (2) 元组

关系中的每个元素是关系中的元组，通常用 $t$ 表示。

## (3) 单元关系与二元关系

当 $n=1$ 时，称该关系为单元关系 (**Unary relation**)

或一元关系

当 $n=2$ 时，称该关系为二元关系 (**Binary relation**)



# 关系 (续)

## (4) 关系的表示

关系也是一个二维表，表的每行对应一个元组，表的每列对应一个域

## (5) 属性

- 关系中不同列可以对应相同的域
- 为了加以区分，必须对每列起一个名字，称为属性  
(Attribute)
- $n$ 目关系必有 $n$ 个属性



# 关系 (续)

## (6) 码

- 候选码 (Candidate key)

若关系中的某一属性组的值能唯一地标识一个元组，则称该属性组为候选码

简单的情况：候选码只包含一个属性

- 全码 (All-key)

最极端的情况：关系模式的所有属性组是这个关系模式的候选码，称为全码 (All-key)



# 关系 (续)

## (6) 码 (续)

- 主码

若一个关系有多个候选码，则选定其中一个为主码（Primary key）

- 主属性

候选码的诸属性称为主属性（Prime attribute）

不包含在任何候选码中的属性称为非主属性（Non-Prime attribute）或非码属性（Non-key attribute）



# 关系 (续)

❖  $D1, D2, \dots, Dn$  的笛卡尔积的某个子集才有实际含义

例：表2.1 的笛卡尔积没有实际意义

取出有实际意义的元组来构造关系

关系： **SAP(SUPERVISOR, SPECIALITY, POSTGRADUATE)**

假设：导师与专业：n:1， 导师与研究生：1:n

主码： **POSTGRADUATE** (假设研究生不会重名)

SUPERVISOR	SPECIALITY	POSTGRADUATE
张清玫	计算机专业	李勇
张清玫	计算机专业	刘晨
刘逸	信息专业	王敏



# 关系 (续)

## (7) 三类关系

基本关系 (基本表或基表)

实际存在的表，是实际存储数据的逻辑表示

查询表

查询结果对应的表

视图表

由基本表或其他视图表导出的表，是虚表，不对应实际存储的数据



# 关系 (续)

## (8) 基本关系的性质

- ① 列是同质的 (**Homogeneous**)
- ② 不同的列可出自同一个域
  - 其中的每一列称为一个属性
  - 不同的属性要给予不同的属性名
- ③ 列的顺序无所谓,，列的次序可以任意交换
- ④ 任意两个元组的候选码不能相同
- ⑤ 行的顺序无所谓, 行的次序可以任意交换



# 基本关系的性质（续）

## ⑥ 分量必须取原子值

这是规范条件中最基本的一条

表2.3 非规范化关系

SUPERVISOR	SPECIALITY	POSTGRADUATE	
		PG1	PG2
张清玫	计算机专业	李勇	刘晨
刘逸	信息专业	王敏	

小表

# 2.1 关系数据结构

## 2.1.1 关系

## 2.1.2 关系模式

## 2.1.3 关系数据库

## 2.1.4 关系模型的存储结构



## 2.1.2 关系模式

1. 什么是关系模式
2. 定义关系模式
3. 关系模式与关系



# 1. 什么是关系模式

- ❖ 关系模式（**Relation Schema**）是型
- ❖ 关系是值
- ❖ 关系模式是对关系的描述
  - 元组集合的结构
    - 属性构成
    - 属性来自的域
    - 属性与域之间的映象关系
  - 完整性约束条件



## 2. 定义关系模式

关系模式可以形式化地表示为：

$$R (U, D, DOM, F)$$

**R**           关系名

**U**           组成该关系的属性名集合

**D**           **U**中属性所来自的域

**DOM**       属性向域的映象集合

**F**           属性间数据的依赖关系的集合



# 定义关系模式（续）

例：

导师和研究生出自同一个域——人，  
取不同的属性名，并在模式中定义属性向域  
的映象，即说明它们分别出自哪个域：

**DOM (SUPERVISOR-PERSON)**  
**= DOM (POSTGRADUATE-PERSON)**  
**= PERSON**



# 定义关系模式 (续)

关系模式通常可以简记为

$R(U)$  或  $R(A_1, A_2, \dots, A_n)$

- $R$ : 关系名
- $A_1, A_2, \dots, A_n$ : 属性名

注: 域名及属性向域的映象常常直接说明为  
属性的类型、长度



# 3. 关系模式与关系

## ❖ 关系模式

- 对关系的描述
- 静态的、稳定的

## ❖ 关系

- 关系模式在某一时刻的状态或内容
- 动态的、随时间不断变化的

## ❖ 关系模式和关系往往笼统称为关系

通过上下文加以区别



# 2.1 关系数据结构

## 2.1.1 关系

## 2.1.2 关系模式

## 2.1.3 关系数据库

## 2.1.4 关系模型的存储结构



## 2.1.3 关系数据库

### ❖ 关系数据库

- 在一个给定的应用领域中，所有关系的集合构成一个关系数据库

### ❖ 关系数据库的型与值

- 关系数据库的型：关系数据库模式，是对关系数据库的描述
- 关系数据库的值：关系模式在某一时刻对应的关系的集合，通常称为关系数据库



# 2.1 关系数据结构

## 2.1.1 关系

## 2.1.2 关系模式

## 2.1.3 关系数据库

## 2.1.4 关系模型的存储结构



## 2.1.4 关系模型的存储结构

### ◆ 关系数据库的物理组织

- 有的关系数据库管理系统中一个表对应一个操作系统文件，将物理数据组织交给操作系统完成
- 有的关系数据库管理系统从操作系统那里申请若干个大的文件，自己划分文件空间，组织表、索引等存储结构，并进行存储管理



# 第二章 关系数据库

2.1 关系模型概述

2.2 关系操作

2.3 关系的完整性

2.4 关系代数

2.5 \*关系演算

2.6 小结



## 2.2.1 基本的关系操作

### ❖ 常用的关系操作

- 查询操作：选择、投影、连接、除、并、差、交、笛卡尔积
  - 选择、投影、并、差、笛卡尔积是5种基本操作
- 数据更新：插入、删除、修改

### ❖ 关系操作的特点

- 集合操作方式：操作的对象和结果都是集合，一次一集合的方式



## 2.2.2 关系数据库语言的分类

- ❖ 关系代数语言
  - 用对关系的运算来表达查询要求
  - 代表: **ISBL**
- ❖ 关系演算语言: 用谓词来表达查询要求
  - 元组关系演算语言
    - 谓词变元的基本对象是元组变量
    - 代表: **APLHA, QUEL**
  - 域关系演算语言
    - 谓词变元的基本对象是域变量
    - 代表: **QBE**
- ❖ 具有关系代数和关系演算双重特点的语言
  - 代表: **SQL (Structured Query Language)**



# 第二章 关系数据库

2.1 关系数据结构及形式化定义

2.2 关系操作

2.3 关系的完整性

2.4 关系代数

2.5 \*关系演算

2.6 小结



# 关系的三类完整性约束

## ❖ 实体完整性和参照完整性

- 关系模型必须满足的完整性约束条件称为关系的两个**不变性**，应该由关系系统自动支持

## ❖ 用户定义的完整性

- 应用领域需要遵循的约束条件，体现了具体领域中的语义约束



# 2.3 关系的完整性

## 2.3.1 实体完整性

## 2.3.2 参照完整性

## 2.3.3 用户定义的完整性



## 2.3.1 实体完整性

### ◆ 规则2.1 实体完整性规则 (Entity Integrity)

- 若属性A是基本关系R的主属性，则属性A不能取空值
- 空值就是“不知道”或“不存在”或“无意义”的值

例：

选修 (学号, 课程号, 成绩)

“学号、课程号”为主码

“学号”和“课程号”两个属性都不能取空值



# 实体完整性 (续)

## ❖ 实体完整性规则的说明

- (1) 实体完整性规则是针对基本关系而言的。  
一个基本表通常对应现实世界的一个实体集。
- (2) 现实世界中的实体是可区分的，即它们具有某种唯一性标识。
- (3) 关系模型中以主码作为唯一性标识。
- (4) 主码中的属性即主属性不能取空值。  
主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，这与第 (2) 点相矛盾，因此这个规则称为实体完整性



# 2.3 关系的完整性

## 2.3.1 实体完整性

## 2.3.2 参照完整性

## 2.3.3 用户定义的完整性



## 2.3.2 参照完整性

1. 关系间的引用
2. 外码
3. 参照完整性规则



# 1. 关系间的引用

- 在关系模型中实体及实体间的联系都是用关系来描述的，自然存在着关系与关系间的引用。

## 例2.1] 学生实体、专业实体

主码

学生 (学号, 姓名, 性别, 专业号, 年龄)

专业 (专业号, 专业名)

主码

- 学生关系引用了专业关系的主码“专业号”。
- 学生关系中的“专业号”值必须是确实存在的专业的专业号

# 关系间的引用（续）

例[2.2] 学生、课程、学生与课程之间的多对多联系

学生（学号，姓名，性别，专业号，年龄）

课程（课程号，课程名，学分）

选修（学号，课程号，成绩）



# 关系间的引用 (续)

例[2.3] 学生实体及其内部的一对多联系

学生 (学号, 姓名, 性别, 专业号, 年龄, 班长)

学号	姓名	性别	专业号	年龄	班长
801	张三	女	01	19	802
802	李四	男	01	20	
803	王五	男	01	20	802
804	赵六	女	02	20	805
805	钱七	男	02	19	

- “学号”是主码, “班长”是外码, 它引用了本关系的“学号”
- “班长”必须是确实存在的学生的学号



## 2. 外码 (Foreign Key)

- ❖ 设  $F$  是基本关系  $R$  的一个或一组属性，但不是关系  $R$  的码。如果  $F$  与基本关系  $S$  的主码  $K_s$  相对应，则称  $F$  是  $R$  的 **外码**
- ❖ 基本关系  $R$  称为 **参照关系** (Referencing Relation)
- ❖ 基本关系  $S$  称为 **被参照关系** (Referenced Relation) 或 **目标关系** (Target Relation)



# 外码（续）

❖ [例2.1]中学生关系的“专业号”与专业关系的主码“专业号”相对应

- “专业号”属性是学生关系的外码
- 专业关系是被参照关系，学生关系为参照关系

学生关系  $\xrightarrow{\text{专业号}}$  专业关系  
(a)



# 外码（续）

## ❖ [例2.2]中

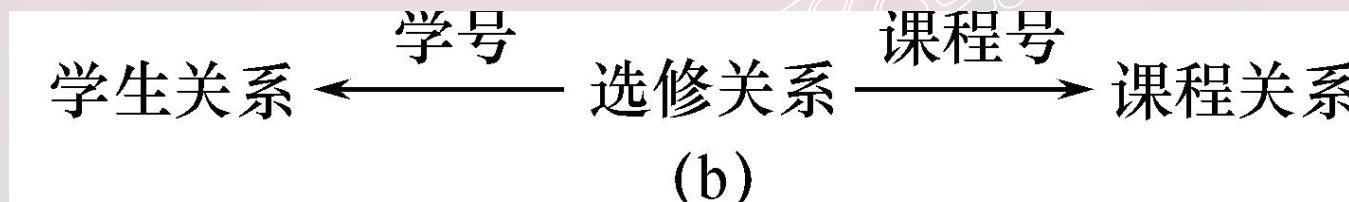
选修关系的“学号”与学生关系的主码“学号”相对应

选修关系的“课程号”与课程关系的主码“课程号”相对应

■ “学号”和“课程号”是选修关系的外码

■ 学生关系和课程关系均为被参照关系

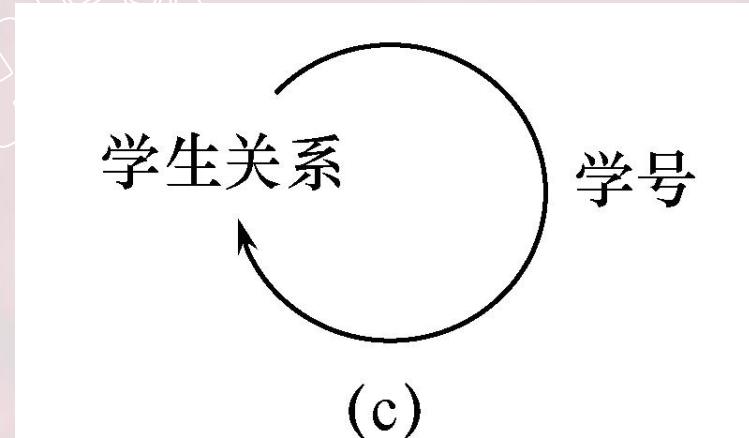
■ 选修关系为参照关系



# 外码（续）

❖ [例2.3]中“班长”与本身的主码“学号”相对应

- “班长”是外码
- 学生关系既是参照关系也是被参照关系



# 外码（续）

- ❖ 关系R和S不一定是不同的关系
- ❖ 目标关系S的主码K<sub>s</sub> 和参照关系的外码F必须定义在同一个（或一组）域上
- ❖ 外码并不一定要与相应的主码同名  
当外码与相应的主码属于不同关系时，往往取相同的名字，以便于识别



### 3. 参照完整性规则

#### ❖ 规则2.2 参照完整性规则

若属性（或属性组） $F$ 是基本关系 $R$ 的外码它与基本关系 $S$ 的主码 $K_s$ 相对应（基本关系 $R$ 和 $S$ 不一定是不同的关系），则对于 $R$ 中每个元组在 $F$ 上的值必须为：

- 或者取空值（ $F$ 的每个属性值均为空值）
- 或者等于 $S$ 中某个元组的主码值



# 参照完整性规则（续）

[例2.1]中

学生关系中每个元组的“专业号”属性只取两类值：

- (1) 空值，表示尚未给该学生分配专业
- (2) 非空值，这时该值必须是专业关系中某个元组的“专业号”值，表示该学生不可能分配一个不存在的专业



# 参照完整性规则（续）

[例2.2] 中

选修（学号，课程号，成绩）

“学号”和“课程号”可能的取值：

- (1) 选修关系中的主属性，不能取空值
- (2) 只能取相应被参照关系中已经存在的主码值



# 参照完整性规则（续）

[例2.3] 中

学生（学号，姓名，性别，专业号，年龄，班长）

“班长”属性值可以取两类值：

- (1) 空值，表示该学生所在班级尚未选出班长
- (2) 非空值，该值必须是本关系中某个元组的学号值



# 2.3 关系的完整性

## 2.3.1 实体完整性

## 2.3.2 参照完整性

## 2.3.3 用户定义的完整性



## 2.3.3 用户定义的完整性

- ❖ 针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求
- ❖ 关系模型应提供定义和检验这类完整性的机制，以便用统一的系统的方法处理它们，而不需由应用程序承担这一功能



# 用户定义的完整性（续）

例：

课程（课程号，课程名，学分）

- “课程号”属性必须取唯一值
- 非主属性“课程名”也不能取空值
- “学分”属性只能取值{1, 2, 3, 4}



# 第二章 关系数据库

2.1 关系模型概述

2.2 关系数据结构

2.3 关系的完整性

2.4 关系代数

2.5 \*关系演算

2.6 小结



## 2.4 关系代数

- ❖ 关系代数是一种抽象的查询语言，它用对关系的运算来表达查询
- ❖ 关系代数
  - 运算对象是关系
  - 运算结果亦为关系
  - 关系代数的运算符有两类：集合运算符和专门的关系运算符
- ❖ 传统的集合运算是从关系的“水平”方向即行的角度进行
- ❖ 专门的关系运算不仅涉及行而且涉及列



## 2.4 关系代数

表2.4 关系代数运算符

运 算 符	含 义	
集合 运算符	$\cup$	并
	$-$	差
	$\cap$	交
	$\times$	笛卡尔积
专门的 关系 运算符	$\sigma$	选择
	$\pi$	投影
	$\bowtie$	连接
	$\div$	除



# 2.4 关系代数

## 2.4.1 传统的集合运算

## 2.4.2 专门的关系运算



# (1) 并 (Union)

## ❖ **R**和**S**

- 具有相同的目  $n$  (即两个关系都有  $n$  个属性)
- 相应的属性取自同一个域

## ❖ **R** **U** **S**

- 仍为  $n$  目关系, 由属于 **R** 或属于 **S** 的元组组成

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



# 并 (续)

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

RUS

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1
a1	b3	c2



## (2) 差 (Difference)

### ❖ $R$ 和 $S$

- 具有相同的目  $n$
- 相应的属性取自同一个域

### ❖ $R - S$

- 仍为  $n$  目关系，由属于  $R$  而不属于  $S$  的所有元组组成

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



# 差 (续)

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

R-S

A	B	C
a1	b1	c1



# (3) 交 (Intersection)

## ❖ $R$ 和 $S$

- 具有相同的目  $n$
- 相应的属性取自同一个域

## ❖ $R \cap S$

- 仍为  $n$  目关系，由既属于  $R$  又属于  $S$  的元组组成

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

$$R \cap S = R - (R - S)$$



# 交 (续)

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

$R \cap S$

A	B	C
a1	b2	c2
a2	b2	c1



# (4) 笛卡尔积 (Cartesian Product)

- ❖ 严格地讲应该是广义的笛卡尔积 (Extended Cartesian Product)
- ❖  $R$ :  $n$ 目关系,  $k_1$ 个元组
- ❖  $S$ :  $m$ 目关系,  $k_2$ 个元组
- ❖  $R \times S$ 
  - 列:  $(n+m)$  列元组的集合
    - 元组的前  $n$  列是关系  $R$  的一个元组
    - 后  $m$  列是关系  $S$  的一个元组
  - 行:  $k_1 \times k_2$  个元组
    - $R \times S = \{ \overbrace{t_r \ t_s} | t_r \in R \wedge t_s \in S \}$



# 笛卡尔积 (续)

$R \times S$

R.A	R.B	R.C	S.A	S.B	S.C
a1	b1	c1	a1	b2	c2
a1	b1	c1	a1	b3	c2
a1	b1	c1	a2	b2	c1
a1	b2	c2	a1	b2	c2
a1	b2	c2	a1	b3	c2
a1	b2	c2	a2	b2	c1
a2	b2	c1	a2	b2	c2
a2	b2	c1	a2	b2	c1

# 2.4 关系代数

## 2.4.1 传统的集合运算

## 2.4.2 专门的关系运算



## 2.4.2 专门的关系运算

先引入几个记号

(1)  $R, t \in R, t[A_i]$

设关系模式为  $R(A_1, A_2, \dots, A_n)$

它的一个关系设为  $R$

$t \in R$  表示  $t$  是  $R$  的一个元组

$t[A_i]$  则表示元组  $t$  中相应于属性  $A_i$  的一个分量



# 专门的关系运算 (续)

(2)  $A$ ,  $t[A]$ ,  $\bar{A}$

若  $A=\{A_{i1}, A_{i2}, \dots, A_{ik}\}$ , 其中  $A_{i1}, A_{i2}, \dots, A_{ik}$  是  $A_1, A_2, \dots, A_n$  中的一部分, 则  $A$  称为属性列或属性组。

$t[A]=(t[A_{i1}], t[A_{i2}], \dots, t[A_{ik}])$  表示元组  $t$  在属性列  $A$  上诸分量的集合。

$\bar{A}$  则表示  $\{A_1, A_2, \dots, A_n\}$  中去掉  $\{A_{i1}, A_{i2}, \dots, A_{ik}\}$  后剩余的属性组。



# 专门的关系运算（续）

(3)  $\widehat{t_r t_s}$

$R$ 为 $n$ 目关系,  $S$ 为 $m$ 目关系。

$t_r \in R$ ,  $t_s \in S$ ,  $\widehat{t_r t_s}$ 称为元组的连接。

$\widehat{t_r t_s}$ 是一个 $n + m$ 列的元组, 前 $n$ 个分量为 $R$ 中的一个 $n$ 元组, 后 $m$ 个分量为 $S$ 中的一个 $m$ 元组。



# 专门的关系运算（续）

## （4）象集 $Z_x$

给定一个关系  $R (X, Z)$ ，  $X$  和  $Z$  为属性组。

当  $t[X]=x$  时，  $x$  在  $R$  中的 **象集** (Images Set) 为：

$$Z_x = \{t[Z] \mid t \in R, t[X]=x\}$$

它表示  $R$  中属性组  $X$  上值为  $x$  的诸元组在  $Z$  上分量的集合



# 专门的关系运算 (续)

$R$	
$x_1$	$Z_1$
$x_1$	$Z_2$
$x_1$	$Z_3$
$x_2$	$Z_2$
$x_2$	$Z_3$
$x_3$	$Z_1$
$x_3$	$Z_3$

象集举例

❖  $x_1$ 在 $R$ 中的象集

$$Z_{x1} = \{Z_1, Z_2, Z_3\},$$

❖  $x_2$ 在 $R$ 中的象集

$$Z_{x2} = \{Z_2, Z_3\},$$

❖  $x_3$ 在 $R$ 中的象集

$$Z_{x3} = \{Z_1, Z_3\}$$



# 专门的关系运算 (续)

1. 选择

2. 投影

3. 连接

4. 除运算



# 专门的关系运算 (续)

学生-课程数据库：

学生关系**Student**、课程关系**Course**和选修关系**SC**

## **Student**

学号 <b>Sno</b>	姓名 <b>Sname</b>	性别 <b>Ssex</b>	年龄 <b>Sage</b>	所在系 <b>Sdept</b>
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

(a)

# 专门的关系运算 (续)

## Course

课程号 <b>Cno</b>	课程名 <b>Cname</b>	先行课 <b>Cpno</b>	学分 <b>Ccredit</b>
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

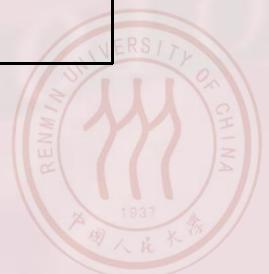
(b)

# 专门的关系运算 (续)

SC

学号 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

(c)



# 1. 选择 (Selection)

❖ 选择又称为限制 (Restriction)

❖ 选择运算符的含义

■ 在关系  $R$  中选择满足给定条件的诸元组

$$\sigma_F(R) = \{t | t \in R \wedge F(t) = \text{'真'}$$

■  $F$ : 选择条件, 是一个逻辑表达式, 取值为“真”或“假”

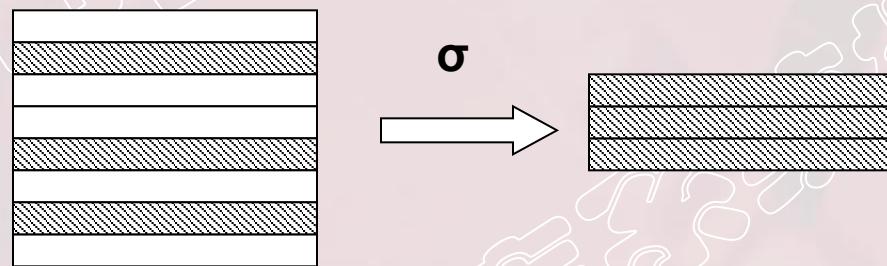
- 基本形式为:  $X_1 \theta Y_1$

- $\theta$  表示比较运算符, 它可以是  $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ,  $=$  或  $\neq$



# 选择 (续)

- ◆ 选择运算是从关系  $R$  中选取使逻辑表达式  $F$  为真的元组，是从行的角度进行的运算



# 选择 (续)

[例2.4] 查询信息系 (IS系) 全体学生。

$\sigma_{Sdept = 'IS'} (Student)$

结果：

Sno	Sname	Ssex	Sage	Sdept
201215125	张立	男	19	IS



# 选择 (续)

[例2.5] 查询年龄小于20岁的学生。

$\sigma_{\text{Sage} < 20}(\text{Student})$

结果：

Sno	Sname	Ssex	Sage	Sdept
201215122	刘晨	女	19	IS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS



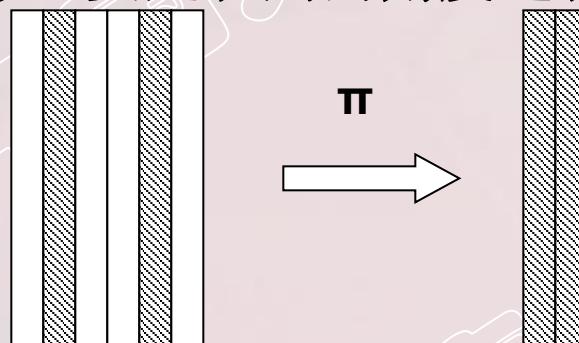
## 2. 投影 (Projection)

- 从  $R$  中选择出若干属性列组成新的关系

$$\pi_A(R) = \{ t[A] \mid t \in R \}$$

$A$ :  $R$  中的属性列

- 投影操作主要是从列的角度进行运算



- 投影之后不仅取消了原关系中的某些列，而且还可能取消某些元组（避免重复行）



# 投影 (续)

❖ [例2.6] 查询学生的姓名和所在系。

即求**Student**关系上学生姓名和所在系两个属性上的投影

$\pi_{Sname, Sdept}(Student)$

结果：

Sname	Sdept
李勇	CS
刘晨	CS
王敏	MA
张立	IS



# 投影 (续)

[例2.7] 查询学生关系Student中都有哪些系。

$\pi_{Sdept}(Student)$

结果：

Sdept
CS
IS
MA



# 3. 连接 (Join)

- ❖ 连接也称为 $\theta$ 连接
- ❖ 连接运算的含义

从两个关系的笛卡尔积中选取属性间满足一定条件的元组

$$R \bowtie_{A\theta B} S = \{ \overbrace{t_r t_s}^{\wedge} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] \theta t_s[B] \}$$

- **A**和**B**: 分别为**R**和**S**上度数相等且可比的属性组
  - **$\theta$** : 比较运算符
- 连接运算从**R**和**S**的广义笛卡尔积  $R \times S$  中选取**R**关系在**A**属性组上的值与**S**关系在**B**属性组上的值满足比较关系 **$\theta$** 的元组

# 连接 (续)

## ❖ 两类常用连接运算

### ■ 等值连接 (equijoin)

- $\theta$  为 “=” 的连接运算称为等值连接
- 从关系  $R$  与  $S$  的广义笛卡尔积中选取  $A$ 、 $B$  属性值相等的那些元组，即等值连接为：

$$R \underset{A=B}{\bowtie} S = \{ \hat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B] \}$$



# 连接 (续)

## ■ 自然连接 (Natural join)

- 自然连接是一种特殊的等值连接

- 两个关系中进行比较的分量必须是相同的属性组
- 在结果中把重复的属性列去掉

- 自然连接的含义

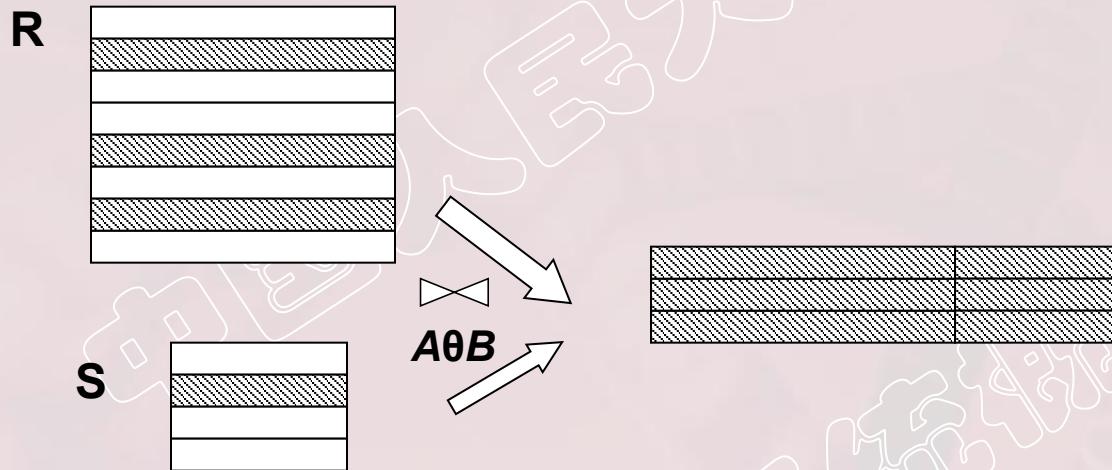
$R$ 和 $S$ 具有相同的属性组 $B$

$$R \bowtie S = \{ \widehat{t_r t_s} [U-B] \mid t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B] \}$$



# 连接 (续)

- ◆ 一般的连接操作是从行的角度进行运算。



自然连接还需要取消重复列，所以是同时从行和列的角度进行运算。



# 连接 (续)

❖ [例2.8]关系R和关系S如下所示:

R

A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	12

S

B	E
b1	3
b2	7
b3	10
b3	2
b2	2



# 连接 (续)

一般连接  $R \bowtie_{C \leq E} S$  的结果如下：

A	R.B	C	S.B	E
a1	b1	5	b2	7
a1	b1	5	b3	10
a1	b2	6	b2	7
a1	b2	6	b3	10
a2	b3	8	b3	10

# 连接 (续)

等值连接  $R \bowtie S$  的结果如下:

$R.B=S.B$

A	R.B	C	S.B	E
a1	b1	5	b1	3
a1	b2	6	b2	7
a2	b3	8	b3	10
a2	b3	8	b3	2



# 连接 (续)

自然连接  $R \bowtie S$  的结果如下：

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2

# 连接 (续)

## ❖ 悬浮元组 (Dangling tuple)

- 两个关系  $R$  和  $S$  在做自然连接时, 关系  $R$  中某些元组有可能在  $S$  中不存在公共属性上值相等的元组, 从而造成  $R$  中这些元组在操作时被舍弃了, 这些被舍弃的元组称为悬浮元组。



# 连接 (续)

## ❖ 外连接 (Outer Join)

- 如果把悬浮元组也保存在结果关系中，而在其他属性上填空值(Null)，就叫做外连接
- 左外连接(**LEFT OUTER JOIN**或**LEFT JOIN**)
  - 只保留左边关系R中的悬浮元组
- 右外连接(**RIGHT OUTER JOIN**或**RIGHT JOIN**)
  - 只保留右边关系S中的悬浮元组



# 连接 (续)

下图是例2.8中关系R和关系S的外连接

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2
a2	b4	12	NULL
NULL	b5	NULL	2

# 连接 (续)

图(b)是例2.8中关系R和关系S的左外连接,图(c)是右外连接

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2
a2	b4	12	NULL

图(b)

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2
NULL	b5	NULL	2

图(c)

## 4. 除运算 (Division)

给定关系  $R (X, Y)$  和  $S (Y, Z)$ , 其中  $X, Y, Z$  为属性组。  
 $R$  中的  $Y$  与  $S$  中的  $Y$  可以有不同的属性名, 但必须出自相同的域集。

$R$  与  $S$  的除运算得到一个新的关系  $P(X)$ ,

$P$  是  $R$  中满足下列条件的元组在  $X$  属性列上的投影:

元组在  $X$  上分量值  $x$  的象集  $Y_x$  包含  $S$  在  $Y$  上投影的集合, 记作:

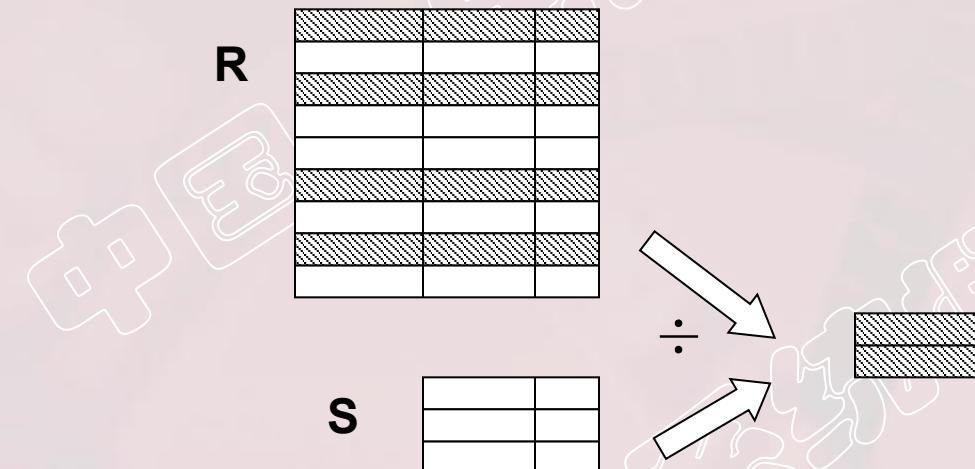
$$R \div S = \{t_r[X] \mid t_r \in R \wedge \pi_Y(S) \subseteq Y_x\}$$

$Y_x$ :  $x$  在  $R$  中的象集,  $x = t_r[X]$



# 除运算 (续)

◆ 除操作是同时从行和列角度进行运算



# 除运算 (续)

[例2.9]设关系R、S分别为下图的(a)和(b),  $RS$ 的结果为图(c)

R

A	B	C
a1	b1	c2
a2	b3	c7
a3	b4	c6
a1	b2	c3
a4	b6	c6
a2	b2	c3
a1	b2	c1

S

B	C	D
b1	c2	d1
b2	c1	d1
b2	c3	d2

$R \div S$

A
a1

# 除运算 (续)

❖ 在关系R中，A可以取四个值{a1, a2, a3, a4}

$a_1$ 的象集为  $\{(b_1, c_2), (b_2, c_3), (b_2, c_1)\}$

$a_2$ 的象集为  $\{(b_3, c_7), (b_2, c_3)\}$

$a_3$ 的象集为  $\{(b_4, c_6)\}$

$a_4$ 的象集为  $\{(b_6, c_6)\}$

❖ S在(B, C)上的投影为

$\{(b1, c2), (b2, c1), (b2, c3)\}$

❖ 只有 $a_1$ 的象集包含了S在(B, C)属性组上的投影

所以  $R \div S = \{a_1\}$



# 综合举例

以学生-课程数据库为例

**[例2.10]** 查询至少选修1号课程和3号课程的学生号码。

首先建立一个临时关系K:

Cno
1
3

然后求:  $\pi_{Sno, Cno}(SC) \div K$

# 综合举例（续）

❖ [例2.10]续

$\pi_{Sno, Cno}(SC)$

201215121象集{1, 2, 3}

201215122象集{2, 3}

$K=\{1, 3\}$

于是：

$\pi_{Sno, Cno}(SC) \div K=\{201215121\}$

Sno	Cno
201215121	1
201215121	2
201215121	3
201215122	2
201215122	3



# 综合举例（续）

[例2.11] 查询选修了2号课程的学生的学号。

$$\pi_{Sno}(\sigma_{Cno='2'}(SC)) = \{201215121, 201215122\}$$

[例2.12] 查询至少选修了一门其直接先行课为5号课程的学生姓名

$$\pi_{Sname}(\sigma_{Cpno='5'}(Course) \bowtie SC \bowtie \pi_{Sno, Sname}(Student))$$

或

$$\pi_{Sname}(\pi_{Sno}(\sigma_{Cpno='5'}(Course) \bowtie SC) \bowtie \pi_{Sno, Sname}(Student))$$

[例2.13] 查询选修了全部课程的学生号码和姓名。

$$\pi_{Sno, Cno}(SC) \div \pi_{Cno}(Course) \bowtie \pi_{Sno, Sname}(Student)$$


# 小结

## ❖ 关系代数运算

### ■ 关系代数运算

- 并、差、交、笛卡尔积、投影、选择、连接、除

### ■ 基本运算

- 并、差、笛卡尔积、投影、选择

### ■ 交、连接、除

- 可以用5种基本运算来表达

- 引进它们并不增加语言的能力，但可以简化表达



# 小结（续）

## ❖ 关系代数表达式

- 关系代数运算经有限次复合后形成的式子

## ❖ 典型关系代数语言

- **ISBL (Information System Base Language)**

- 由 **IBM United Kingdom** 研究中心研制
- 用于 **PRTV (Peterlee Relational Test Vehicle)** 实验系统



# 第二章 关系数据库

2.1 关系模型概述

2.2 关系数据结构

2.3 关系的完整性

2.4 关系代数

2.5 \*关系演算

2.6 小结



## 2.6 小结

- ❖ 关系数据库系统是目前使用最广泛的数据库系统
- ❖ 关系数据库系统与非关系数据库系统的区别：
  - 关系系统只有“表”这一种数据结构
  - 非关系数据库系统还有其他数据结构，以及对这些数据结构的操作



# 小结（续）

## ❖ 关系数据结构

### ■ 关系

- 域
- 笛卡尔积
- 关系
  - 关系, 属性, 元组
  - 候选码, 主码, 主属性
  - 基本关系的性质

### ■ 关系模式

### ■ 关系数据库

### ■ 关系模型的存储结构



# 小结（续）

## ❖ 关系操作

### ■ 查询

- 选择、投影、连接、除、并、交、差

### ■ 数据更新

- 插入、删除、修改



# 小结（续）

## ❖ 关系的完整性约束

- 实体完整性
- 参照完整性
  - 外码
- 用户定义的完整性



# 小结（续）

## ❖ 关系数据语言

### ■ 关系代数语言

### ■ 关系演算语言

- 元组关系演算语言 **ALPHA**

- 域关系演算语言 **QBE**

