

第2章 关系模型与关系运算

第2章 关系模型与关系运算

2.1 关系模型与关系运算简述

--关系模型的提出与作用

--关系模型与关系运算概览

--本章的目标

2.2 关系与关系模型

2.3 关系代数运算

2.4 关系元组演算

2.5 关系域演算语言及QBE

2.6 小结

2.1 关系模型简述

—关系模型的提出与作用

- 关系模型最早由E. F. Codd在1970年提出
- 是从表(Table)及表的处理方式抽象出来的，是在对传统表及其操作严格定义基础上，引入集合理论与逻辑学理论提出的
- 是数据库的三大经典数据模型之一，也是现在大多数商品化数据库系统所仍然使用的数据模型
- 标准的数据库语言(SQL语言)是建立在关系模型基础之上的，数据库领域的众多理论也都是建立在关系模型基础之上的

2.1 关系模型简述

—关系模型与关系运算概览

- 形象地说，一个关系(relation)就是一个Table
- 关系模型就是处理Table的，它由三个部分组成：
(详细内容在后面讲述)
 - ❑ 描述DB各种数据的基本结构形式(采用Table描述)
 - ❑ 描述Table与Table之间所可能发生的各种操作
(采用关系运算等描述)
 - ❑ 描述这些操作所应遵循的约束条件(被称为完整性条件)

2.1 关系模型简述

—关系模型与关系运算概览

➤ 将要学习：Table如何描述，有哪些操作、结果是什么、有哪些约束等？

学生登记表

学号	姓名	班级	出生年月	家庭住址
11101	李德	2	1980.5	山东
11102	范燕美	2	1980.8	哈尔滨
11103	张靖	2	1981.3	北京
11104	许聪	2	1980.7	云南
11105	黄佩婷	2	1979.12	浙江

操作

有哪些？

学生成绩单

班级	姓名	语文	数学	英语
2	李德	75	86	71
2	范燕美	76	78	68
2	张靖	81	77	80
2	许聪	82	82	79
2	黄佩婷	80	79	82

结果

是什么？

2.1 关系模型简述

—关系模型与关系运算概览(续)

- 关系运算有**关系代数**和**关系演算**两种形式，关系演算又进一步区分：元组演算和域演算
- 关系代数示例：**基于集合的运算**

$$\pi_{\text{姓名,课程名}} \left(\sigma_{\text{课程号}=c2} (R \bowtie S) \right)$$

- 基于关系代数设计的数据库语言 (ISBL)：**用计算机可识别的符号表征关系代数的运算符号**

$$((R * S): \text{课程号} = c2) \% \text{姓名, 课程名}$$

$K:F$ 表选择运算, $K\%$ 表投影运算

2.1 关系模型简述

—关系模型与关系运算概览(续)

- 元组演算示例：基于逻辑的运算

$$\{t | (\exists u)(R(t) \wedge W(u) \wedge t[3] < u[1])\}$$

- 基于元组演算设计的数据库语言 (Ingres系统的 QUEL) : 用计算机可识别的符号表征元组演算的运算符号

range of t is R

range of u is W

retrieve t

where t.sage < u.sage

2.1 关系模型简述

—关系模型与关系运算概览(续)

➤ 域演算示例：基于示例的运算

$\{t_1 t_2 t_3 | S(t_1 t_2 t_3) \wedge R(t_1 t_2 t_3) \wedge t_1 < 20 \wedge t_2 > 30\}$

➤ 基于域演算设计的数据库语言示例：(QBE: Query By Example)

S	S#	Sname	Sage	Sclass	Sex
	<u>S1</u>	P. <u>X</u>			

C	C#	Cname	Cformat	Cteacher	Coffice
	<u>C1</u>	计算机原理			

SC	S#	C#	Semester	Scgrade
	<u>S1</u>	<u>C1</u>		P. <u>Y</u>

2.1 关系模型简述

—关系模型与关系运算概览(续)

- 关系代数与元组演算是标准数据库语言SQL语言的基础
- QBE语言由于其可视化效果较好，也很受使用者的欢迎
- SQL语言将在下一章学习
- 本章将要学习：关系代数和元组演算
- 本章还将学习QBE语言

2.1 关系模型简述

——本章目标

- 理解关系(relation)，理解如何用Relation对Table进行抽象或说严格定义
- 理解关系/表(relation/table)所具有的各种特性，理解关系模型
- 熟练掌握关系代数、元组演算和域演算(域演算以QBE为例来学习)，用这些关系运算来表达各种复杂的检索需求，以便于后续SQL语言的学习

第2章关系模型与关系运算

2.1 关系模型与关系运算简述

2.2 关系与关系模型

--关系

--关系模型

2.3 关系代数运算

2.4 关系元组演算

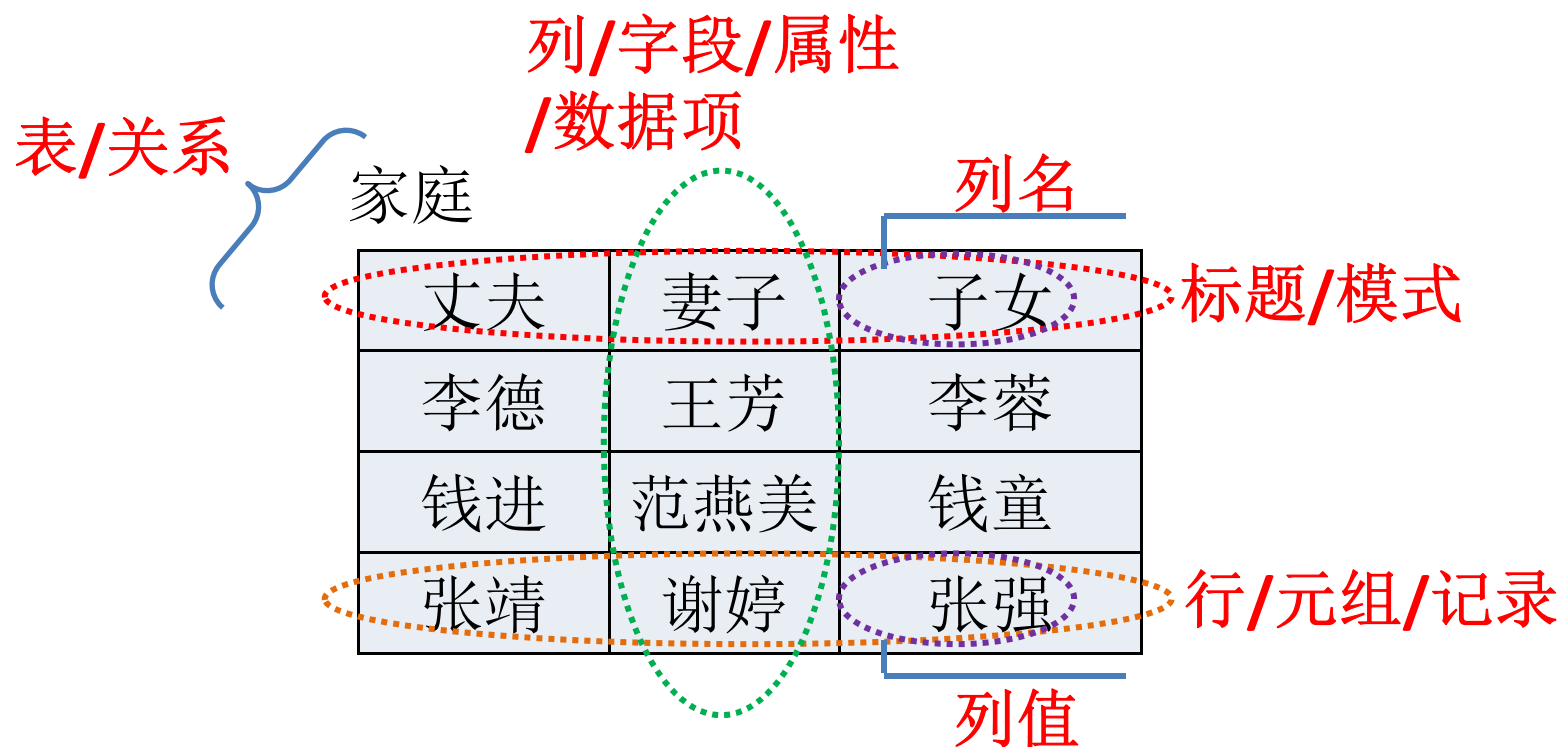
2.5 关系域演算语言及QBE

2.6 小结

2.2 关系与关系模型

--关系: Table的严格定义

➤ 如何严格地定义Table呢?



2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 首先定义“列”的取值范围“域(Domain)”

域(Domain)

- 一组值的集合，这组值具有相同的数据类型
 - ✓ 如整数的集合、字符串的集合、全体学生的集合
 - ✓ 再如，由8位数字组成的数字串的集合，由0到100组成的整数集合
- 集合中元素的个数称为域的基数(Cardinality)

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 首先定义“列”的取值范围“域(Domain)”

域(Domain)

家庭

丈夫	妻子	子女
李德	王芳	李蓉
钱进	范燕美	钱童

D3=儿童集合(CHILD)=(李蓉, 钱童)

D2=女人集合(WOMAN)=(王芳, 范燕美)

D1=男人集合(MAN)=(李德, 钱进)

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤再定义“元组”及所有可能组合成的元组:笛卡尔积
笛卡尔积(Cartesian Product)

□ 一组域D1, D2, ..., Dn的笛卡尔积为:

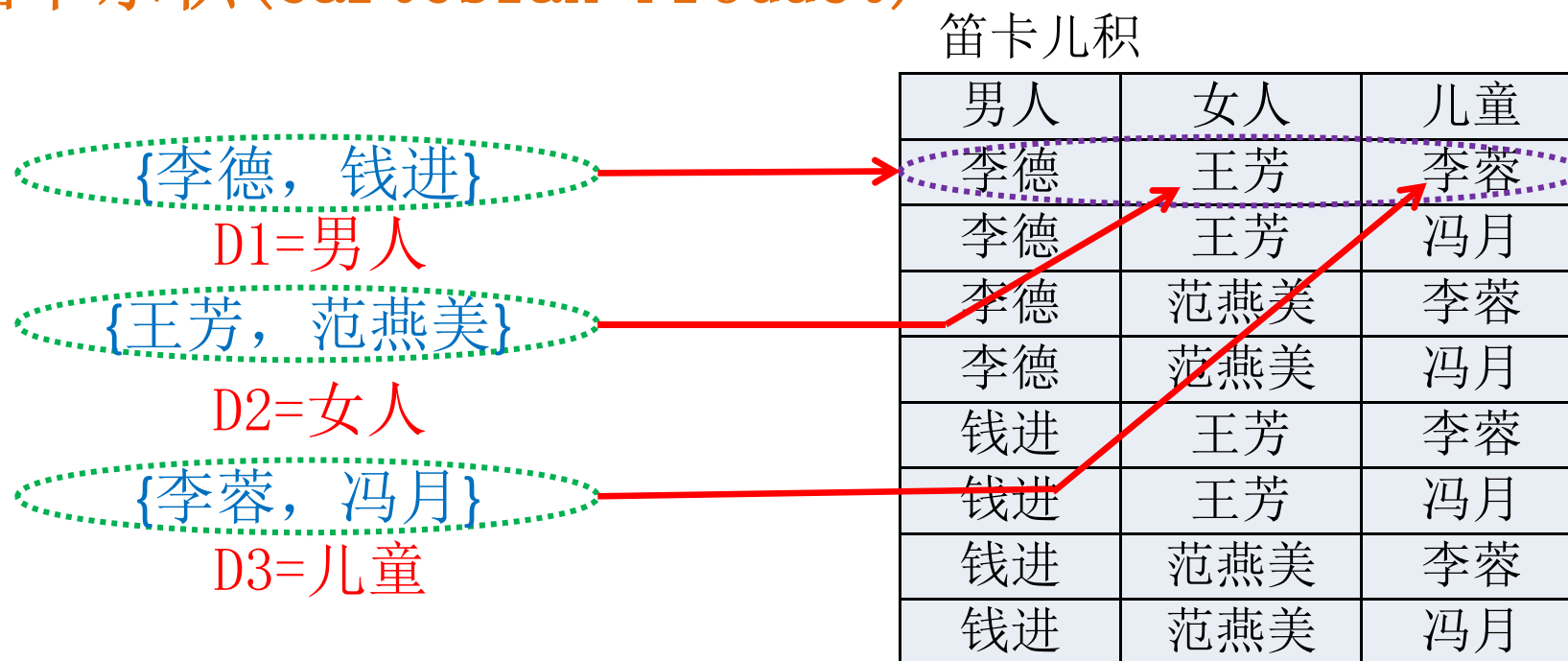
$$D1 \times D2 \times \dots \times Dn = \{(d1, d2, \dots, dn) \mid di \in Di, i=1, \dots, n\}$$

□ 笛卡尔积的每个元素(d1, d2, ..., dn)称作一个
n-元组 (n-tuple)

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤再定义“元组”及所有可能组合成的元组:笛卡尔积
笛卡尔积(Cartesian Product)



2.2 关系与关系模型

--关系: Table的严格定义(续)

➤再定义“元组”及所有可能组合成的元组:笛卡尔积
笛卡尔积(Cartesian Product)

- 元组 (d_1, d_2, \dots, d_n) 的每一个值 d_i 叫做一个分量(component)
- 元组 (d_1, d_2, \dots, d_n) 是从每一个域任取一个值所形成的一种组合, 笛卡尔积是所有这种可能组合的集合, 即: 笛卡尔积是由 n 个域形成的所有可能的 n -元组的集合
- 若 d_i 的基数为 m_i , 则笛卡尔积的基数, 即元组个数为 $m_1 \times m_2 \times \dots \times m_n$

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 再定义“元组”及所有可能组合成的元组:笛卡尔积
笛卡尔积(Cartesian Product)

笛卡儿积

男人	女人	儿童	→ 域名
李德	王芳	李蓉	
钱进	范燕美	钱童	
钱进	谢婷	张强	→ 域值
张靖	谢婷	冯月	

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 由于笛卡尔积中的所有元组并不都是有意义的, 因此...

关系(Relation)

- 一组域 D_1, D_2, \dots, D_n 的笛卡尔积的子集:
- 笛卡尔积中具有某一方面意义的那些元组被称作一个关系(Relation)
- 由于关系的不同列可能来自同一个域, 为区分, 需要为每一列起一个名字, 该名字即为属性名。不同列名的列值可以来自相同域。

2.2 关系与关系模型

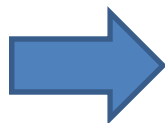
--关系: Table的严格定义(续)

- 由于笛卡尔积中的所有元组并不都是有意义的, 因此...

关系(Relation)

笛卡儿积

男人	女人	儿童
李德	王芳	李蓉
李德	王芳	冯月
李德	范燕美	李蓉
李德	范燕美	冯月
钱进	王芳	李蓉
钱进	王芳	冯月
钱进	范燕美	李蓉
钱进	范燕美	冯月



家庭

丈夫	妻子	子女
李德	王芳	李蓉
钱进	范燕美	钱童

列名(属性名)

列值: 来自域

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 由于笛卡尔积中的所有元组并不都是有意义的, 因此...

关系(Relation)

- ❑ 关系可用 $R(A_1:D_1, A_2:D_2, \dots, A_n:D_n)$ 表示, 可简记为 $R(A_1, A_2, \dots, A_n)$, 这种描述又被称为关系模式(Schema)或表标题(head)
- ❑ R 是关系的名字, A_i 是属性, D_i 是属性所对应的域, n 是关系的度或目(degree), 关系中元组的数目称为关系的基数(Cardinality)
- ❑ 关系模式中 $A_i (i=1, \dots, n)$ 必须是不同的, 而 $D_i (i=1, \dots, n)$ 是可以相同的

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 由于笛卡尔积中的所有元组并不都是有意义的, 因此...

关系(Relation)

▣ 例如下图的关系为一3目关系, 描述为
家庭(丈夫:男人, 妻子:女人, 子女:儿童)
或家庭(丈夫, 妻子, 子女)

家庭

丈夫	妻子	子女
李德	王芳	李蓉
钱进	范燕美	钱童

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 由于笛卡尔积中的所有元组并不都是有意义的, 因此...

关系(Relation)

- 关系模式 $R(A_1:D_1, A_2:D_2, \dots, A_n:D_n)$ 中属性向域的映象
在很多DBMS中一般直接说明为属性的类型、长度等

例如:

Student(S# char(8), Sname char(10), Ssex char(2), Sage integer,
D# char(2), Sclass char(6))

再如:

Course (C# char(3), Cname char(12), Chours integer,
Credit float(1), T# char(3))
SC(S# char(8), C# char(3), Grade float(1))

2.2 关系与关系模型

--关系: Table的严格定义(续)

➤ 关系模式与关系

- 同一关系模式下，可有很多的关系
- 关系模式是关系的结构，关系是关系模式在某一时刻的数据
- 关系模式是稳定的；而关系是某一时刻的值，是随时间可能变化的

Student(S# char(7), Sname char(10), Ssex char(2))

Student

S#	Sname	Ssex
1110101	张三	女
1110102	李四	男
1110201	王五	男

Student

S#	Sname	Ssex
1110101	张三	女
1110102	李四	男
1110201	王五	男
1110202	刘六	女

2.2 关系与关系模型

--关系(relation/table)的特性

- 列是同质：即每一列中的分量来自同一域，是同一类型的数据

Student

S#	Sname	Ssex	Sage	D#	Sclass
1110101	张三	女	21	03	11101
1110102	李四	男	20	03	11101
1110103	王五	男	19	软件	11101
1110201	刘六	大学	21	软件	11102
1110202	钱七	男	20	02	11102
1110203	赵八	高中	21	01	11102

2.2 关系与关系模型

--关系(relation/table)的特性

- 不同的列可来自同一个域，称其中的每一列为一个属性，不同的属性要给予不同的属性名。

例，假如：我们定义一个域为Person = 所有男人、女人和儿童的集合={李德，钱进，王芳，范燕美，李蓉，钱童，张峰}，则下述“家庭”关系的三个列将来自同一个域Person，因此需要不同的属性名“丈夫”“妻子”“子女”以示区分。

家庭

丈夫	妻子	子女
李德	王芳	李蓉
钱进	范燕美	钱童

2.2 关系与关系模型

--关系(relation/table)的特性(续)

- 列位置互换性: 区分哪一列是靠列名
- 行位置互换性: 区分哪一行是靠某一或某几列的值(关键字/键字/码字)
- 关系是以内容(名字或值)来区分的, 而不是属性在关系的位置来区分
- 如下面两个关系是完全相同的关系

家庭

丈夫	妻子	子女
李德	王芳	李蓉
钱进	范燕美	钱童

家庭

丈夫	妻子	子女
张强	孙巧	张建
吴鹏	刘淼	吴玉

2.2 关系与关系模型

--关系(relation/table)的特性(续)

- 任意两个元组不能完全相同。(集合的要求：集合内不能有相同的两个元素)
- 元组相同是指两个元组的每个分量都相同

Student

S#	Sname	Ssex	Sage	D#	Sclass
1110101	张三	女	21	03	11101
1110102	李四	男	20	03	11101
1110103	王五	男	19	02	11101
1110201	刘六	男	21	02	11102
1110102	李四	男	20	03	11101
1110203	赵八	女	21	01	11102

在同一个关系(表)中存在相同的元组,去掉其中的重复元组

2.2 关系与关系模型

--关系(relation/table)的特性(续)

➤ 属性不可再分特性: 又被称为关系第一范式

Student

S#	Sname	Ssex
1110101	张三	女
1110102	李四	男

符合第一范式(Table)

Student

S#	姓名		Ssex
	姓	名字	
1110101	李	四	男
1110102	王	五	男
1110103	刘	六	男

不符合第一范式
(Not Table)

2.2 关系与关系模型

--关系(relation/table)的特性(续)

➤ 属性不可再分特性: 又被称为关系第一范式(续)

家庭

丈夫	妻子	子女
李德	王芳	李蓉
钱进	范燕美	钱童

符合第一范式(Table)

家庭

丈夫	妻子	子女	
		第一	第二
李德	王芳	李刚	李蓉
钱进	范燕美	钱瑞	钱童

不符合第一范式
(Not Table)

2.2 关系与关系模型

--关系(relation/table)的特性(续)

➤ 属性不可再分特性: 又被称为关系第一范式(续)

下面的关系不符合第一范式(Table)

家庭

丈夫	妻子	子女	家庭住址
李德	王芳	李蓉	中国山东省威海市文化西路二号
钱进	范燕美	钱童	中国北京市王府井大街十号云丽弄三门

2.2 关系与关系模型

--关系(relation/table)上的一些重要概念

➤ 候选码(Candidate Key)/候选键

- 关系中的一个属性组，其值能唯一标识一个元组，若从该属性组中去掉任何一个属性，它就不具有这一性质了，这样的属性组称作候选码。
- 例如关系“学生(S#, Sname, Sage, Sclass)”中S#就是一个候选码，在学生关系中，任何两个元组的S#是一定不同的，而这两个元组的Sname, Sage, Sclass都可能相同(同名、同龄、同班)，所以S#是候选码。
- 再如关系“选课(S#, C#, Sname, Cname, Grade)”中(S#, C#)联合起来是一个候选码

2.2 关系与关系模型

--关系(relation/table)上的一些重要概念

➤ 候选码(Candidate Key)/候选键

▣ 有时，关系中有很多组候选码，

•例如：

学生(S#, Sname, Sage, Sclass, Saddress)

其中属性S#是候选码，属性组(Sname, Saddress)也是候选码(同名同地址的两个同学是不存在的)

•再如

Employee(EmpID, EmpName, Mobile)

每一雇员有唯一的EmpID，没有两个雇员有相同的手机号Mobile，则EmpID是候选码，Mobile也是候选码

2.2 关系与关系模型

--关系(relation/table)上的一些重要概念

➤ 主码(Primary Key)/主键

- ▣ 当有多个候选码时，可以选定一个作为主码。
- ▣ DBMS以主码为主要线索管理关系中的各个元组。
 - 例如可选定属性S#作为“学生”表的主码，也可以选定属性组(Sname, Saddress)作为“学生”表的主码。
 - 选定EmpID为Employee的主码。

2.2 关系与关系模型

--关系(relation/table)上的一些重要概念

➤ 主属性与非主属性

- ▣ 包含在任何一个候选码中的属性被称作主属性，如“选课”中的S#，D#。
- ▣ 而其他属性被称作非主属性，如“选课”中的Sname, Cname, Grade。

➤ 在最简单的情况下，候选码只包含一个属性

2.2 关系与关系模型

--关系(relation/table)上的一些重要概念

- 主属性与非主属性(续)
- 在最简单的情况下，候选码只包含一个属性
- 在最极端的情况下，关系的所有属性组是这个关系的候选码，称为全码(All-Key)。比如关系“教师授课”(T#, C#)中的候选码(T#, C#)就是全码。

教师授课表

T#	C#
几何	张三
物理	李四
代数	王五

2.2 关系与关系模型

--关系(relation/table)上的一些重要概念

➤ 外码(Foreign Key)/外键

- ▣ 关系R中的一个属性组，它不是R的候选码，但它与另一个关系S的候选码相对应，则称这个属性组为R的外码或外键。

• 例如“合同”关系中的**客户号**不是候选码，但却是外码。因它与“客户”关系中的候选码“**客户号**”相对应。

合同			外码	客户		
主码	合同号	合同名称	客户号	主码	客户号	联系人
	HIT001	电脑采购	ZQ		ZQ	志强电脑
	HIT002	外墙修补	JA		JA	吉安涂装
	HIT003	文件打印	ZQ		YY	用友印刷

2.2 关系与关系模型

--关系(relation/table) 的概念小结

- 域(Domain)
- 元组(Tuple)
- 笛卡尔积
- 关系(Relation)
- 关系模式(Relational Schema)
- 属性(Attribute)
- 关系的度(Degree)与关系的基数(Cardinality)
- 候选码、主码、主属性与非主属性、外码

2.2 关系与关系模型

——关系模型 注意区别：关系模式

➤ 关系模型 同一关系模式下，可有很多的关系

▣ DB数据的基本结构：Relation/Table

▣ DB数据的基本操作有：

\cup (并, UNION)、 \cap (交, INTERSECTION)、 \bowtie (连接, JOIN)、

$-$ (差, DIFFERENCE)、 \times (广义积, PRODUCT)、

σ (选择, SELECTION)、 π (投影, PROJECTION)、

\div (除, DIVISION) 等运算

▣ DB数据的结构与操作受三个完整性的约束：

实体完整性、参照完整性和用户自定义完整性

2.2 关系与关系模型

—关系模型中的完整性

- Relation/Table已在前面介绍过
- DB数据的基本操作对象是Relation/Table，操作的结果仍然是Relation/Table;
- 关系操作是集合操作，操作的对象及结果都是集合，是一次一集合(Set-at-a-time)的方式。而非关系型的数据操作方式是一次一记录(Record-at-a-time)，具体操作将在关系代数中介绍

下面将介绍：关系的完整性

2.2 关系与关系模型

—关系的实体完整性

➤ 实体完整性

- ❑ 关系的主码中的属性值不能为空值；
- ❑ 空值：不知道或无意义的值；
- ❑ 意义：关系中的元组对应到现实世界相互之间可区分的一个个个体，这些个体是通过主码来唯一标识的；若主码为空，则出现不可标识的个体，这是不容许的。

Student

主 码	S#	Sname	Ssex
	1110101	张三	女
		李四	男
	1110102	王五	男

2.2 关系与关系模型

—关系的实体完整性(续)

➤ 空值的含义

- ❑ 空值：不知道、不存在或无意义的值；
- ❑ 在进行关系操作时，有时关系中的某属性值在当前是填不上的，
 - 比如：档案中有“生日不详”、“下落不明”、“日程尚待公布”等，这时就需要空值来代表这种情况。关系模型中用‘?’表征

2.2 关系与关系模型

—关系的实体完整性(续)

➤ 空值的含义(续)

- ❑ 数据库中有了空值，会影响许多方面，如影响聚集函数运算的正确性，不能参与算术、比较或逻辑运算等
- 例如：“ $3 + ?$ ”结果是多少呢？ “ $3 * ?$ ”结果是多少呢？ “ $? \text{ and } (A=A)$ ”结果又是多少呢？
- 再例如，一个班有30名同学，如所有同学都有成绩，则可求出平均成绩；如果有一个同学没有成绩，怎样参与平均成绩的计算呢，是当作0，还是当作100呢？还是不考虑他呢？
- 因此有空值的时候是需要特殊处理的，要特别注意。

2.2 关系与关系模型

—关系的参照完整性

➤ 参照完整性

- ❑ 如果关系R1的外码Fk与关系R2的主码Pk相对应，则R1中的每一个元组的Fk值或者等于R2 中某个元组的Pk值，或者为空值
- ❑ 意义：如果关系R1的某个元组t1参照了关系R2的某个元组t2，则t2必须存在
 - 例如关系Student在“系别”上的取值有两种可能：
 - ✓ 空值，表示该学生尚未分到任何系中
 - ✓ 若非空值，则必须是Dept关系中某个元组的“系别”值，表示该学生不可能分到一个不存在的系中

R1--Student
外码Fk—系别

Student

学号	姓名	系别
101	张三	01
102	李四	06
103	王五	✓

R2--Dept
主码Pk—系别

Dept

系别	系名	人数
01	机械	900
02	船舶	850
03	软件	870

2.2 关系与关系模型

—关系的用户自定义完整性

➤ 用户自定义完整性

- ▣ 用户针对具体的应用环境定义的完整性约束条件

要求名字在5个汉字字符之内

性别只能是“男”或“女”

年龄在[12, 35]之间

颠覆完整性的示例

S#	Sname	Ssex	Sage
1110101	张三	女	21
1110102	李四	男	20
1110103	理查德·克莱德曼	男	19
1110201	刘六	0	21
1110202	李四	男	200

2.2 关系与关系模型

--DBMS对关系完整性的支持

- 实体完整性和参照完整性由DBMS系统自动支持
- DBMS系统通常提供了如下机制：它使用户可以自行定义有关的完整性约束条件，并能依据此，当有更新操作时，自动检验更新操作的正确性，即是否符合用户自定义的完整性

第2章关系模型与关系运算

2.1 关系模型与关系运算简述

2.2 关系与关系模型

2.3 关系代数运算

--关系代数概述

--关系代数的基本操作

--用关系代数表达检索请求的示例

2.4 关系元组演算

2.5 关系域演算语言QBE

2.3 关系代数运算

—关系代数概述

- 基于集合，提供了一系列的关系代数操作：并、差、笛卡尔积(广义积)、交、选择、投影、连接和关系除，是一种集合化的操作语言
- 关系代数操作以一个或多个关系为输入，结果是一个新的关系
- 用对关系的运算来表达查询，需要指明所用的操作，因此具有一定的过程性

$$\pi_{\text{姓名,课程名}}(\sigma_{\text{课程号}=c_2}(R \bowtie S))$$

- 是一种抽象的语言，是学习其他数据库语言，如SQL等的基础

2.3 关系代数运算

—关系代数基本操作概览

➤ 关系代数基本操作分为：集合操作和纯关系操作

(1) 集合操作

UNION(并)	R	S	$R \cup S$
INTERSECTION(交)	R	S	$R \cap S$
DIFFERENCE(差)	R	S	$R - S$
CARTESIAN PRODUCT(笛卡儿积)	R	S	$R \times S$

(2) 纯关系操作

PROJECT(投影)	R		$\pi_A(R)$
SELECT(选择)	R		$\sigma_{con}(R)$
JOIN(连接)	R	S	$R \bowtie S$
DIVISION(除)	R	S	$R \div S$

2.3 关系代数运算

—关系代数基本操作概览(续)

➤ 某些关系代数操作，如并、差、交等，需满足“并相容性”

➤ 并相容性

▣ 参与运算的两个关系及其相关属性之间有一定的对应性、可比性或意义关联性

▣ 定义：关系R与关系S存在相容性，当且仅当：

(1) 关系R和关系S的属性数目必须相同；

(2) 对于任意i，关系R的第i个属性的域必须和关系S的第i个属性的域相同

• 假设： $R(A_1, A_2, \dots, A_n), S(B_1, B_2, \dots, B_m)$

• R和S满足并相容性： $n = m$ 并且 $\text{Domain}(A_i) = \text{Domain}(B_i)$

2.3 关系代数运算

—关系代数基本操作概览(续)

➤ 并相容性的示例

STUDENT(SID char(10), Sname char(8), Age char(3))

PROFESSOR(PID char(10), Pname char(8), Age char(3))

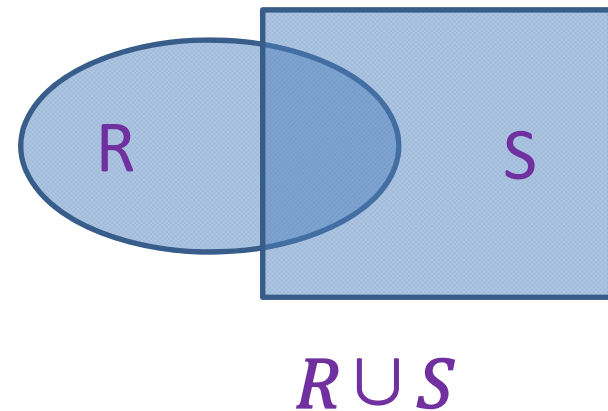
关系STUDENT与关系PROFESSOR是相容的，因为：

- (1) 关系R和关系S的属性数目都是3
- (2) 关系R的属性SID与关系S的属性PID的域都是char(10)
- (3) 关系R的属性Sname与关系S的属性Sname的域都是char(8)
- (4) 关系R的属性Age与关系S的属性Age的域都是char(3)

2.3 关系代数运算

—关系代数之集合操作：并(Union)

- **定义：**假设关系R和关系S是并相容的, 则关系R与关系S的并运算结果也是一个关系, 记作: $R \cup S$, 它由或者出现在关系R中, 或者出现在S中的元组构成。
- **数学描述：** $R \cup S = \{t | t \in R \vee t \in S\}$, 其中t是元组。
- 并运算是将两个关系的元组合并成一个关系, 在合并时去掉重复的元组。
- $R \cup S$ 与 $S \cup R$ 运算的结果是同一个关系。



2.3 关系代数运算

—关系代数之集合操作：并(Union) (续)

➤ 并操作的示例一(抽象的)

• 假设R与S是并相容的两个关系

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

RUS		
C1	C2	C3
a	b	c
a	d	g
f	b	e
a	b	e
h	d	g

2.3 关系代数运算

—关系代数之集合操作：并(Union) (续)

➤ 并操作的示例二(语义的)

- 查询参加体育队或者参加文艺队，或者体育队和文艺队都参加学生的信息($R \cup S$)

R(参加体育
队的学生)

姓名	性别
张三	女
李四	男
王五	男

S(参加文艺
队的学生)

姓名	性别
张三	女
刘六	男
钱七	男

$R \cup S$

姓名	性别
张三	女
李四	男
王五	男
刘六	男
钱七	男

2.3 关系代数运算

—关系代数之集合操作：并(Union) (续)

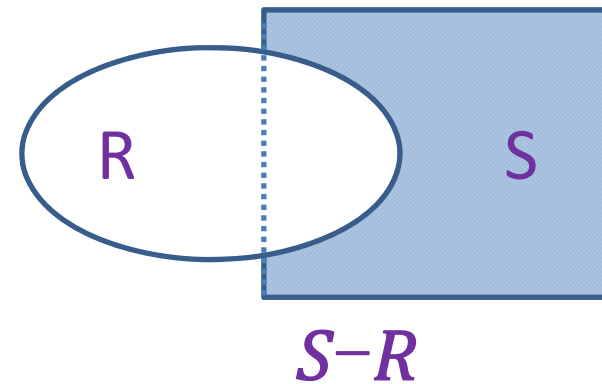
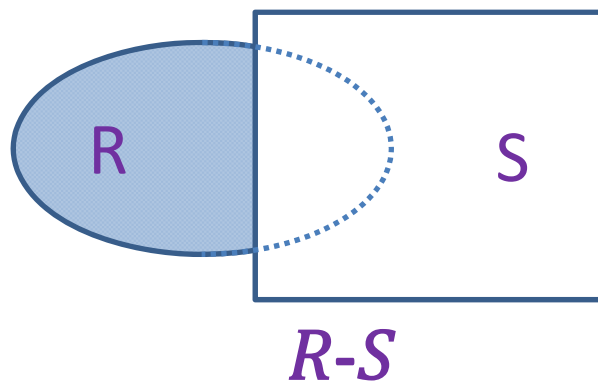
➤ 并操作的示例三(语义的)

- 若R为计算机学院的学生，S为材料学院的学生，
则 $R \cup S$ 为两院所有的学生
- 若R为学过数据库课程的学生，S为学过自控理论课程的学生，
则 $R \cup S$ 为学过两门课之一或者两门课都学过的所有学生
- 汉语中的“A或者B或者AB”通常意义是并运算的要求。首先要准确理解汉语的查询要求，然后再找到正确的操作

2.3 关系代数运算

—关系代数之集合操作：差(Difference)

- 定义：假设关系R和关系S是并相容的，则关系R与关系S的差运算结果也是一个关系，记作： $R-S$ ，它由出现在关系R中但不出现在关系S中的元组构成。
- 数学描述： $R - S = \{t | t \in R \wedge t \notin S\}$ ，其中t是元组
- $R-S$ 与 $S-R$ 是不同的



2.3 关系代数运算

—关系代数之集合操作：差(Difference) (续)

➤ 差操作的示例一(抽象的)

• 假设R与S是并相容的两个关系

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

R-S		
D1	D2	D3
f	b	e

S-R		
E1	E2	E3
a	b	e
h	d	g

2.3 关系代数运算

--关系代数之集合操作：差(Difference) (续)

➤ 差操作的示例二(语义的)

- 查询只参加体育队而未参加文艺队的学生信息(R-S)
- 查询只参加文艺队而未参加体育队的学生信息(S-R)

R(参加体育
队的学生)

姓名	性别
张三	女
李四	男
王五	男

S(参加文艺
队的学生)

姓名	性别
张三	女
刘六	男
钱七	男

R-S

姓名	性别
李四	男
王五	男

S-R

姓名	性别
刘六	男
钱七	男

2.3 关系代数运算

--关系代数之集合操作：差(Difference) (续)

➤ 差操作的示例三(语义的)

- 若R为计算机学院的学生, S为四年级的学生,
则 $R-S$ 为是计算机学院但不是四年级的所有学生(计算机学院非四年级的学生),
 $S-R$ 为是四年级但不是计算机学院的所有学生(四年级非计算机学院的学生)。
- 若R为学过数据库课程的学生, S为学过自控理论课程的学生,
则 $R-S$ 为学过数据库课程但没学过自控理论课程的所有学生
- 汉语中的“是…但不含…”通常意义是差运算的要求。首先要准确理解汉语的查询要求, 然后再找到正确的操作

2.3 关系代数运算

—关系代数之集合操作：交(Intersection)

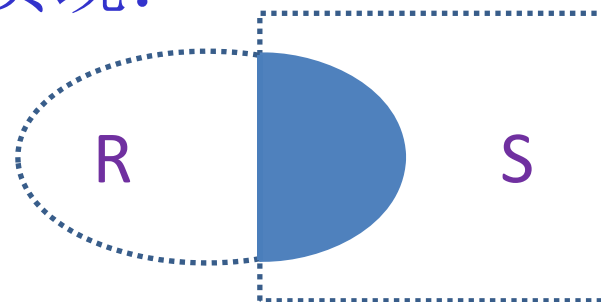
➤ **定义：**假设关系R和关系S是并相容的，则关系R与关系S的交运算结果也是一个关系，记作： $R \cap S$ ，它由同时出现在关系R和关系S中的元组构成。

➤ **数学描述：** $R \cap S = \{t | t \in R \wedge t \in S\}$ ，其中t是元组

➤ $R \cap S$ 和 $S \cap R$ 运算的结果是同一个关系

➤ 交运算可以通过差运算来实现：

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



$$R \cap S = S \cap R$$

2.3 关系代数运算

—关系代数之集合操作：交 (Intersection) (续)

➤ 交操作的示例一 (抽象的)

• 假设R与S是并相容的两个关系

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

$R \cap S$		
F1	F2	F3
a	b	c
a	d	g

2.3 关系代数运算

—关系代数之集合操作：交(Intersection) (续)

➤ 交操作的示例二(语义的)

- 查询既参加体育队又参加文艺队的学生信息 ($R \cap S$)

R(参加体育
队的学生)

姓名	性别
张三	女
李四	男
王五	男

S(参加文艺
队的学生)

姓名	性别
张三	女
刘六	男
钱七	男

$R \cap S = S \cap R$

姓名	性别
张三	女

2.3 关系代数运算

—关系代数之集合操作：交(Intersection) (续)

➤ 交操作的示例三(语义的)

- 若R为年龄小于20岁的学生, S为计算机学院的学生, 则 $R \cap S$ 为计算机学院并且年龄小于20岁的所有学生
- 若R为学过数据库课程的学生, S为学过自控理论课程的学生, 则 $R \cap S$ 为既学过数据库课程又学过自控理论课程的所有学生
- 汉语中的“既…又…”, “…, 并且…”通常意义是交运算的要求, 首先要准确理解汉语的查询要求, 然后再找到正确的操作

2.3 关系代数运算

--关系代数之集合操作：广义笛卡尔积
(Cartesian Product)

➤ 定义：关系 $R(\langle a_1, a_2, \dots, a_n \rangle)$ 与关系 $S(\langle b_1, b_2, \dots, b_m \rangle)$ 的广义笛卡尔积(简称广义积)运算结果也是一个关系，记作： $R \times S$ ，它由关系 R 中的元组与关系 S 的元组进行所有可能的拼接(或串接)构成。

➤ 数学描述：

$$R \times S = \left\{ \langle a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m \rangle \mid \langle a_1, a_2, \dots, a_n \rangle \in R \wedge \langle b_1, b_2, \dots, b_m \rangle \in S \right\}$$

2.3 关系代数运算

—关系代数之集合操作：

广义笛卡尔积(续)

➤ 广义积操作的示例一(抽象的)

• 关系R的元组数目是3, 度数是3;

关系S的元组数目是4, 度数是3;

则 $R \times S$ 的元组数目是12, 度数是6

2.3 关系代数运算

—关系代数之集合操作：
广义笛卡尔积(续)

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

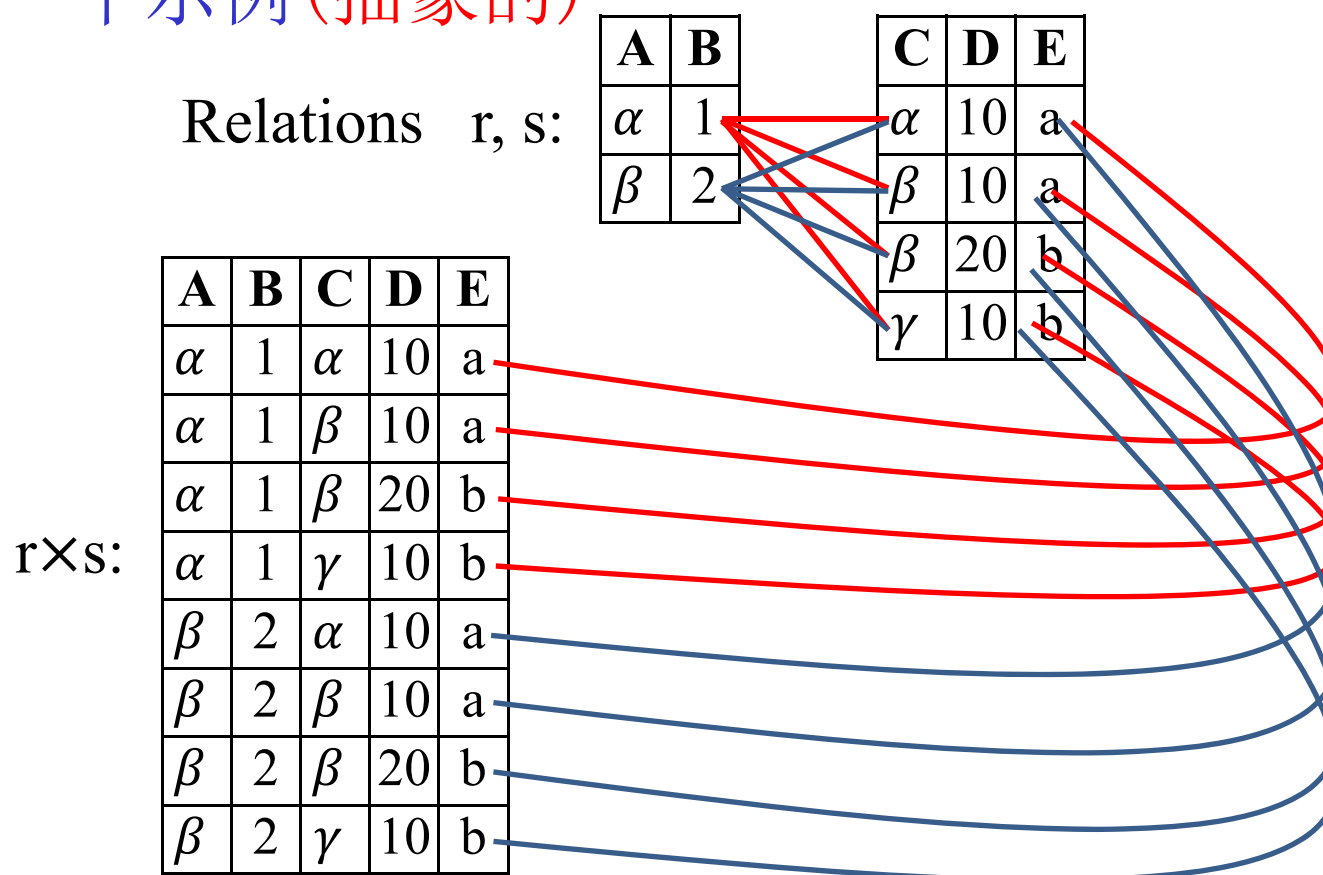
S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

R×S					
A1	A2	A3	B1	B2	B3
a	b	c	a	b	c
a	b	c	a	b	e
a	b	c	a	d	g
a	b	c	h	d	g
a	d	g	a	b	c
a	d	g	a	b	e
a	d	g	a	d	g
a	d	g	h	d	g
f	b	e	a	b	c
f	b	e	a	b	e
f	b	e	a	d	g
f	b	e	h	d	g

2.3 关系代数运算

--关系代数之集合操作：广义笛卡尔积(续)

➤ 再看一个示例(抽象的)



2.3 关系代数运算

—关系代数之集合操作：广义笛卡尔积(续)

➤ 广义积操作的示例之二(语义的)

- 当一个检索涉及到多个表时(如学生表和课程表)，便需要将这些表串接或拼接起来，然后才能检索，这时，就要使用广义笛卡尔积运算
- 是后面学习各种连接运算的基础

2.3 关系代数运算

—关系代数之集合操作：广义笛卡尔积(续)

➤ 广义积操作的示例之二(语义的)

学生表

学号	姓名	年龄	宿舍
01	李三	21	101
02	李四	21	102
03	李五	20	102

课程表

课程号	课程名	教师	学时
C1	几何	A	40
C2	物理	B	40
C3	代数	C	60

(所有学生)的(所有课程)

学号	姓名	年龄	住址	课程号	课程名	教师	学时
01	李三	21	101	C1	几何	A	40
01	李三	21	101	C2	物理	B	40
01	李三	21	101	C3	代数	C	60
02	李四	21	102	C1	几何	A	40
02	李四	21	102	C2	物理	B	40
02	李四	21	102	C3	代数	C	60
03	李五	20	102	C1	几何	A	40
03	李五	20	102	C2	物理	B	40
03	李五	20	102	C3	代数	C	60

2.3 关系代数运算

—关系代数之集合操作：广义笛卡尔积(续)

- $R \times S = S \times R$: $R \times S$ 为R中的每一个元组都和S中的所有元组进行串接。 $S \times R$ 为S中的每一个元组都和R中的所有元组进行串接。结果是相同的。
- 两个关系R和S，它们的属性个数分别为n和m(R是n度关系，S是m度关系)，则笛卡尔积 $R \times S$ 的属性个数为 $n + m$ ，即元组的前n个分量是R中元组的分量，后m个分量是S中元组的分量($R \times S$ 是 $n + m$ 度关系)。
- 两个关系R和S，它们的元组个数分别为x和y(关系R的基数x，S的基数y)，则笛卡尔积 $R \times S$ 的元组个数为 $x \times y$ ($R \times S$ 的基数是 $x \times y$)。

2.3 关系代数运算

—关系代数之纯关系操作：选择(Select)

➤ 定义：给定一个关系R, 同时给定一个选择的条件condition(简记con), 选择运算结果也是一个关系, 记作 $\sigma_{con}(R)$, 它从关系R中选择出满足给定条件condition的元组构成。

➤ 数学描述： $\sigma_{con}(R) = \{t | t \in R \wedge con(t) = '真'\}$

□ 设 $R(A_1, A_2, \dots, A_n)$, t是R的元组, t的分量记为 $t[A_i]$, 或简写为 A_i

□ 条件con由逻辑运算符连接算术表达式组成

□ 逻辑运算符： \wedge, \vee, \neg 或写为and, or, not

□ 算术表达式： $X \theta Y$, 其中X, Y是t的分量、常量或简单函数, θ 是比较运算符,
 $\theta \in \{>, \geq, <, \leq, =, \neq\}$

R		
A1	A2	A3

2.3 关系代数运算

—关系代数之纯关系操作：选择(Select) (续)

➤ 选择操作的示例一(抽象的)

R		
A1	A2	A3
a	a	10
a	d	-4
f	b	5

$\sigma_{A3>0}(R)$		
A1	A2	A3
a	a	10
f	b	5

$\sigma_{A1=a \vee A2="b"}(R)$		
A1	A2	A3
a	a	10
f	b	5

$\sigma_{A3>0 \wedge A1=A2}(R)$		
A1	A2	A3
a	a	10

2.3 关系代数运算

--关系代数之纯关系操作：选择(Select) (续)

➤ 选择操作的示例二(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03

查询所有男同学的信息 $\sigma_{Ssex=\text{“男”}}(R)$

S#	Sname	Ssex	Sage	D#
1110102	李四	男	20	02
1110103	王五	男	19	03

2.3 关系代数运算

--关系代数之纯关系操作：选择(Select) (续)

➤ 选择操作的示例二(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03

查询所有年龄小于20同学的信息 $\sigma_{Sage < 20}(R)$

S#	Sname	Ssex	Sage	D#
1110103	王五	男	19	03

2.3 关系代数运算

—关系代数之纯关系操作：选择(Select) (续)

➤ 选择操作的示例二(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03

查询所有1系或3系的同学信息 $\sigma_{D\#="01" \vee D\#="03"}(R)$

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110103	王五	男	19	03

2.3 关系代数运算

—关系代数之纯关系操作：选择(Select) (续)

➤ 选择操作的示例三(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03

查询所有年龄大于20的1系同学的信息 $\sigma_{Sage>20 \wedge D\#="01"}(R)$

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01

2.3 关系代数运算

--关系代数之纯关系操作：选择(Select) (续)

➤ 选择操作的示例三(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03

查询不在(年龄大于20的1系同学)要求之内的
所有其它同学的信息 $\sigma_{\neg(Sage>20 \wedge D\#="01")}(R)$

S#	Sname	Ssex	Sage	D#
1110102	李四	男	20	02
1110103	王五	男	19	03

2.3 关系代数运算

—关系代数之纯关系操作：选择(Select) (续)

- 选择操作的示例四 (语义的)
- 选择操作从给定的关系中选出满足条件的行，条件的书写很重要，尤其是当不同运算符在一起时，要注意运算符的优先次序，
- 优先次序自高至低为 {括弧； θ ； \neg ； \wedge ； \vee }
- 例如：

$\text{Sage} < 20 \vee \text{Sage} > 18 \wedge D\# = \text{"03"}$

与

$(\text{Sage} < 20 \vee \text{Sage} > 18) \wedge D\# = \text{"03"}$

2.3 关系代数运算

—关系代数之纯关系操作：投影(Project)

➤ 定义：给定一个关系R, 投影运算结果也是一个关系, 记作 $\Pi_A(R)$, 它从关系R中选出属性包含在A中的列构成。

➤ 数学描述：

$$\Pi_{A_{i1}, A_{i2}, \dots, A_{ik}}(R) = \{ \langle t[A_{i1}], t[A_{i2}], \dots, t[A_{ik}] \rangle \mid t \in R \}$$

□ 设 $R(A_1, A_2, \dots, A_n)$,

□ $\{A_{i1}, A_{i2}, \dots, A_{ik}\} \subseteq \{A_1, A_2, \dots, A_n\}$

□ $t[A_i]$ 表示元组t中相应于属性 A_i 的分量

□ 投影运算可以对原关系的列在投影后重新排列

➤ 投影操作从给定关系中选出某些列组成新的关系, 而选择操作是从给定关系中选出某些行组成新的关系。

R		
A1	A2	A3

2.3 关系代数运算

—关系代数之纯关系操作：投影(Project) (续)

➤ 投影操作的示例一(抽象的)

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

$\Pi_{A3}(R)$
A3
c
g
e

$\Pi_{A3, A1}(R)$	
A3	A1
c	a
g	a
e	f

2.3 关系代数运算

—关系代数之纯关系操作：投影(Project) (续)

➤ 如果投影后有重复元组，则应去掉

R		
A1	A2	A3
a	a	c
a	d	c
f	b	c

$\Pi_{A1, A3}(R)$	
A3	A1
a	c
f	c

2.3 关系代数运算

--关系代数之纯关系操作：投影(Project) (续)

➤ 投影操作的示例二(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03

$\Pi_{\text{Sname, Sage}}(R)$

查询所有学生的姓名和年龄

$\Pi_{\text{Sname, Sage}}(R)$

$\Pi_{\text{Sname, D\#}}(R)$

查询所有学生的姓名及其所在的系

$\Pi_{\text{Sname, D\#}}(R)$

2.3 关系代数运算

--关系代数之纯关系操作：投影(Project) (续)

➤ 投影操作的示例二(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03

查询所有学生的姓名和年龄

$\Pi_{Sname, Sage}(R)$

$\Pi_{Sname, Sage}(R)$	
Sname	Sage
张三	21
李四	20
王五	19

查询所有学生的姓名及其所在的系

$\Pi_{Sname, D#}(R)$

$\Pi_{Sname, D#}(R)$	
Sname	D#
张三	01
李四	02
王五	03

2.3 关系代数运算

--关系代数之纯关系操作：投影(Project) (续)

➤ 投影与选择操作一起使用的示例(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03
1110104	刘六	女	21	03

查询所有在3系就读的且年龄
大于19的学生的学号和姓名

$\Pi_{S\#, Sname}(\sigma_{D\#="03" \wedge Sage > 19}(R))$

$\Pi_{Sname, Sage}(R)$

2.3 关系代数运算

--关系代数之纯关系操作：投影(Project) (续)

➤ 投影与选择操作一起使用的示例(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03
1110104	刘六	女	21	03

查询所有在3系就读的且年龄
大于19的学生的学号和姓名

$\Pi_{S\#,Sname}(\sigma_{D\#="03" \wedge Sage > 19}(R))$

$\Pi_{Sname, Sage}(R)$	
S#	Sname
1110104	刘六

2.3 关系代数运算

--关系代数之纯关系操作：投影(Project) (续)

➤ 投影与选择操作一起使用的示例(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03
1110104	刘六	女	21	03

查询所有在3系就读的
且男同学的学号和姓名

$\Pi_{S\#, Sname}(\sigma_{D\#="03" \wedge Ssex="男"}(R))$

$\Pi_{Sname, Sage}(R)$

2.3 关系代数运算

--关系代数之纯关系操作：投影(Project) (续)

➤ 投影与选择操作一起使用的示例(语义的)

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03
1110104	刘六	女	21	03

查询所有在3系就读的
且男同学的学号和姓名

$\Pi_{S\#,Sname}(\sigma_{D\#="03" \wedge Ssex="男"}(R))$

$\Pi_{Sname, Sage}(R)$	
S#	Sname
1110103	王五

2.3 关系代数运算

—关系代数之纯关系操作： θ -连接(θ -Join)

- 投影与选择操作只是对单个关系(表)进行操作, 而实际应用中往往涉及多个表之间的操作
- 比如: 查询数据结构成绩在80分及以上的学生姓名(涉及学生, 科目, 成绩), 这就需要 θ -连接操作

数据库系统

查询数据结构成绩在
80分及以上的学生姓名

学生表

S#	Sname	Ssex
1001	张三	女
1002	李四	男
1003	王五	男
1004	刘六	女

科目表

C#	Cname	Chours	Credit
001	数据库	40	4
002	数据结构	60	4
003	C语言	40	2
004	编译原理	60	4

成绩表

S#	C#	Score
1001	001	80
1001	002	85
1001	003	83
1001	004	88
1002	001	78
1002	002	80
1002	003	82
1002	004	85
1003	001	82
1003	002	68
1003	003	79
...

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join)(续)

➤ 定义：给定关系R和关系S，R与S的 θ 连接运算结果也是一个关系，记作 $R \bowtie_{A\theta B} S$ ，它由关系R和关系S的笛卡尔积中，选取R中属性A与S中属性B之间满足条件的元组构成。

➤ 数学描述： $R \bowtie_{A\theta B} S = \sigma_{t[A]\theta s[B]}(R \times S)$

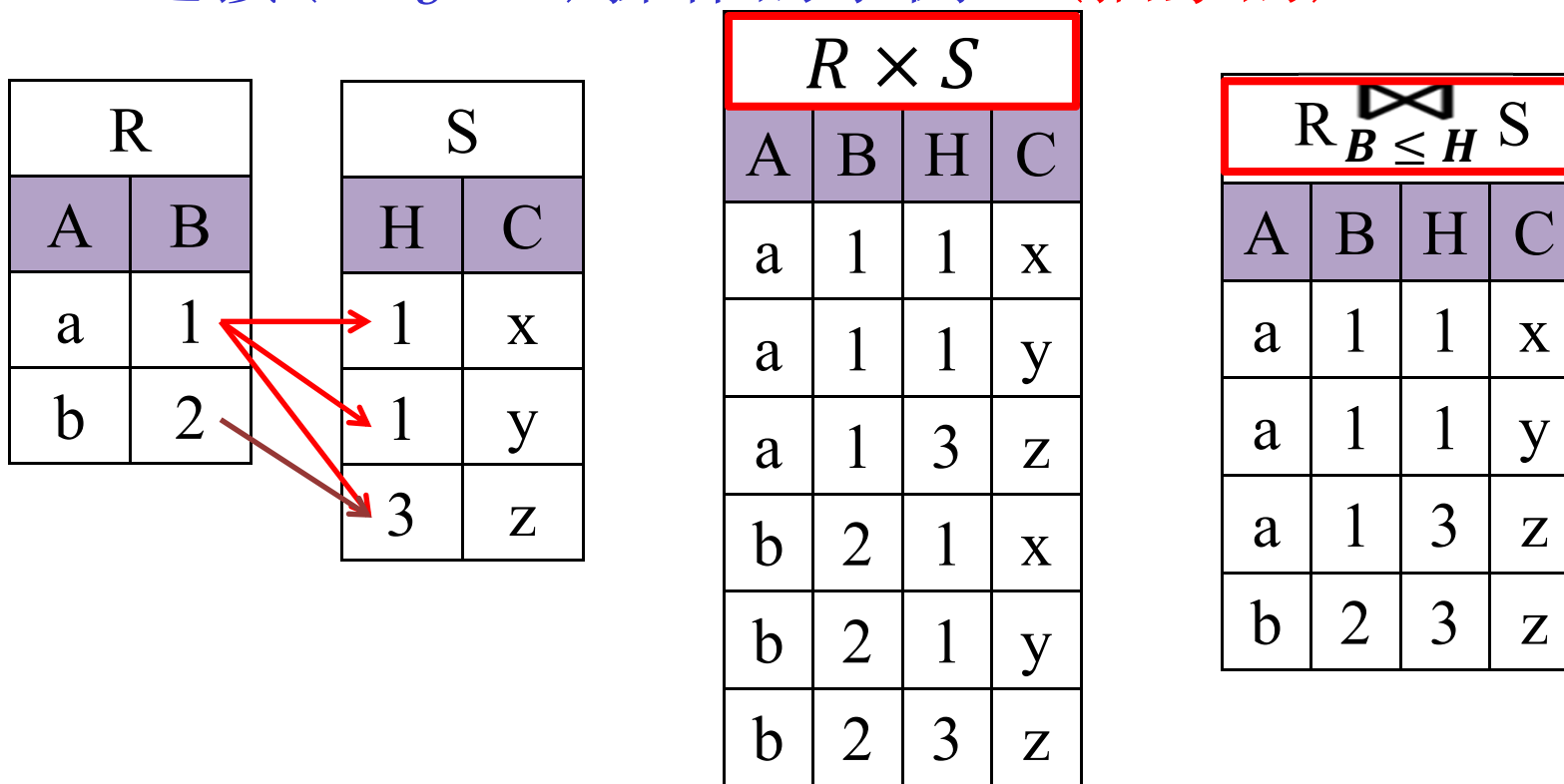
- 设 $R(A_1, A_2, \dots, A_n)$, $A \in \{A_1, A_2, \dots, A_n\}$
- $S(B_1, B_2, \dots, B_m)$, $B \in \{B_1, B_2, \dots, B_m\}$
- t 是关系R中的元组， s 是关系S中的元组
- 属性A和属性B具有可比性
- θ 是比较运算符， $\theta \in \{>, \geq, <, \leq, =, \neq\}$

➤ 在实际应用中， θ -连接操作经常与投影、选择操作一起使用

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接(θ -Join)操作的示例一(抽象的)



2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例二(语义的)

- 员工表Worker(W#, Wname, Wsex, Wage, Degree),
- 职位限定表Position(Type, Limited_Degree(LD))

员工表Worker

W#	Wname	Wsex	Wage	Degree
01	张三	男	35	1
02	李四	男	49	3
03	王五	女	45	2

职位限定表Position

Type	LD
组长	1
项目经理	2
部门经理	3

1:本科 2:硕士 3:博士

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例二(语义的) (续)

- 竞聘的岗位必须由不低于其最低学历要求的人员担任，找出所有员工的姓名及其可能竞聘职位的名称)


$\Pi_{\text{wname,type}}(\text{worker} \bowtie_{\text{Degree} \geq \text{LD}} \text{position})$

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例二(续)

第一步：对两个表进行广义笛卡尔积

(所有员工)的(所有职位)  广义笛卡尔积

W#	Wname	Wsex	Wage	Degree	Type	LD
01	张三	男	35	1	组长	1
01	张三	男	35	1	项目经理	2
01	张三	男	35	1	部门经理	3
02	李四	男	49	3	组长	1
02	李四	男	49	3	项目经理	2
02	李四	男	49	3	部门经理	3
03	王五	女	45	2	组长	1
03	王五	女	45	2	项目经理	2
03	王五	女	45	2	部门经理	3

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例二(续)

第二步：从广义笛卡尔积中选取符合条件的元组
(所有员工)的(所有职位) ← 广义笛卡尔积

W#	Wname	Wsex	Wage	Degree	Type	LD
01	张三	男	35	1	组长	1
01	张三	男	35	1	项目经理	2
01	张三	男	35	1	部门经理	3
02	李四	男	49	3	组长	1
02	李四	男	49	3	项目经理	2
02	李四	男	49	3	部门经理	3
03	王五	女	45	2	组长	1
03	王五	女	45	2	项目经理	2
03	王五	女	45	2	部门经理	3

Degree \geq LD

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例二(续)

第二步：从广义笛卡尔积中选取符合
($\text{degree} \geq \text{limited_degree}$)条件的元组

所有员工及其可以竞聘的职位

W#	Wname	Wsex	Wage	Degree	Type	LD
01	张三	男	35	1	组长	1
02	李四	男	49	3	组长	1
02	李四	男	49	3	项目经理	2
02	李四	男	49	3	部门经理	3
03	王五	女	45	2	组长	1
03	王五	女	45	2	项目经理	2

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例二(续)

第三步：在进行投影操作，得到最终的结果

$$\Pi_{\text{wname,type}}(\text{worker} \bowtie_{\text{Degree} \geq \text{LD}} \text{position})$$

最终想要得到的结果

Wname	Type
张三	组长
李四	组长
李四	项目经理
李四	部门经理
王五	组长
王五	项目经理

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例三 (续)

- 关系与自身的 θ -连接
- 查询1001号同学和1002号同学都学过的所有课程的课程号

$$\Pi_{sc.c\#}(\sigma_{sc.s\#="1001" \wedge sc1.s\#"1002"}(sc \bowtie \rho_{sc1}(sc)))$$

$sc.c\# = SC1.C\#$

- 注：上式 $\rho_{sc1}(sc)$ 表更名操作，即将表SC更名为SC1，当一个表需要和其自身进行连接运算时，通常要使用更名操作

数据库系统

SC

S#	C#	Score
1001	001	80
1001	002	85
1001	003	83
1002	001	78
1002	004	80
1002	005	82
1003	005	81
1003	006	68
1003	007	79
...

$SC \bowtie \rho_{SC1}(SC)$
 $SC.C\# = SC1.C\#$

SC. S#	SC. C#	SC. Score	SC1. S#	SC1. C#	SC1. Score
1001	001	80	1001	001	80
1001	001	80	1002	001	78
1002	001	78	1001	001	80
1002	001	78	1002	001	78
1002	005	82	1002	005	82
1002	005	82	1003	005	81
1003	005	81	1002	005	82
1003	005	81	1003	005	81
...

数据库系统

$$SC \quad \Pi_{sc.c\#}(\sigma_{sc.s\#="1001" \wedge sc1.s\#"1002"}(SC \bowtie \rho_{sc1}(sc)))$$

$SC.C\# = SC1.C\#$

S#	C#	Score
1001	001	80
1001	002	85
1001	003	83
1002	001	78
1002	004	80
1002	005	82
1003	005	81
1003	006	68
1003	007	79
...

SC. S#	SC. C#	SC. Score	SC1. S#	SC1. C#	SC1. Score
1001	001	80	1002	001	78
...

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

➤ θ -连接操作的示例三 (续)

- 关系与自身的 θ -连接
- 查询1001号同学和1002号同学都学过的所有课程的课程号

$$\Pi_{sc.c\#}(\sigma_{sc.s\#="1001" \wedge sc1.s\#"1002"}(sc \bowtie \rho_{sc1}(sc)))$$

$sc.c\# = sc1.c\#$

两个同学都学过的
所有课程的课程号：

SC.C#
001

2.3 关系代数运算

--关系代数之纯关系操作： θ -连接(θ -Join) (续)

- 虽然我们在讲解 θ -连接操作时，使用笛卡尔积然后再进行选择来得到 θ -连接结果。这主要是方便大家理解。但当引入连接操作后，DBMS可直接进行连接操作，而不必先形成笛卡尔积

2.3 关系代数运算

—关系代数之纯关系操作：等值连接(Equi-Join)

➤ **定义**：给定关系R和关系S, R与S的等值连接运算结果也是一个关系,记作 $R \bowtie_{A=B} S$ ，它由关系R和关系S的笛卡尔积中选取R中属性A与S中属性B上值相等的元组所构成。

➤ **数学描述**： $R \bowtie_{A=B} S = \sigma_{t[A]=s[B]}(R \times S)$

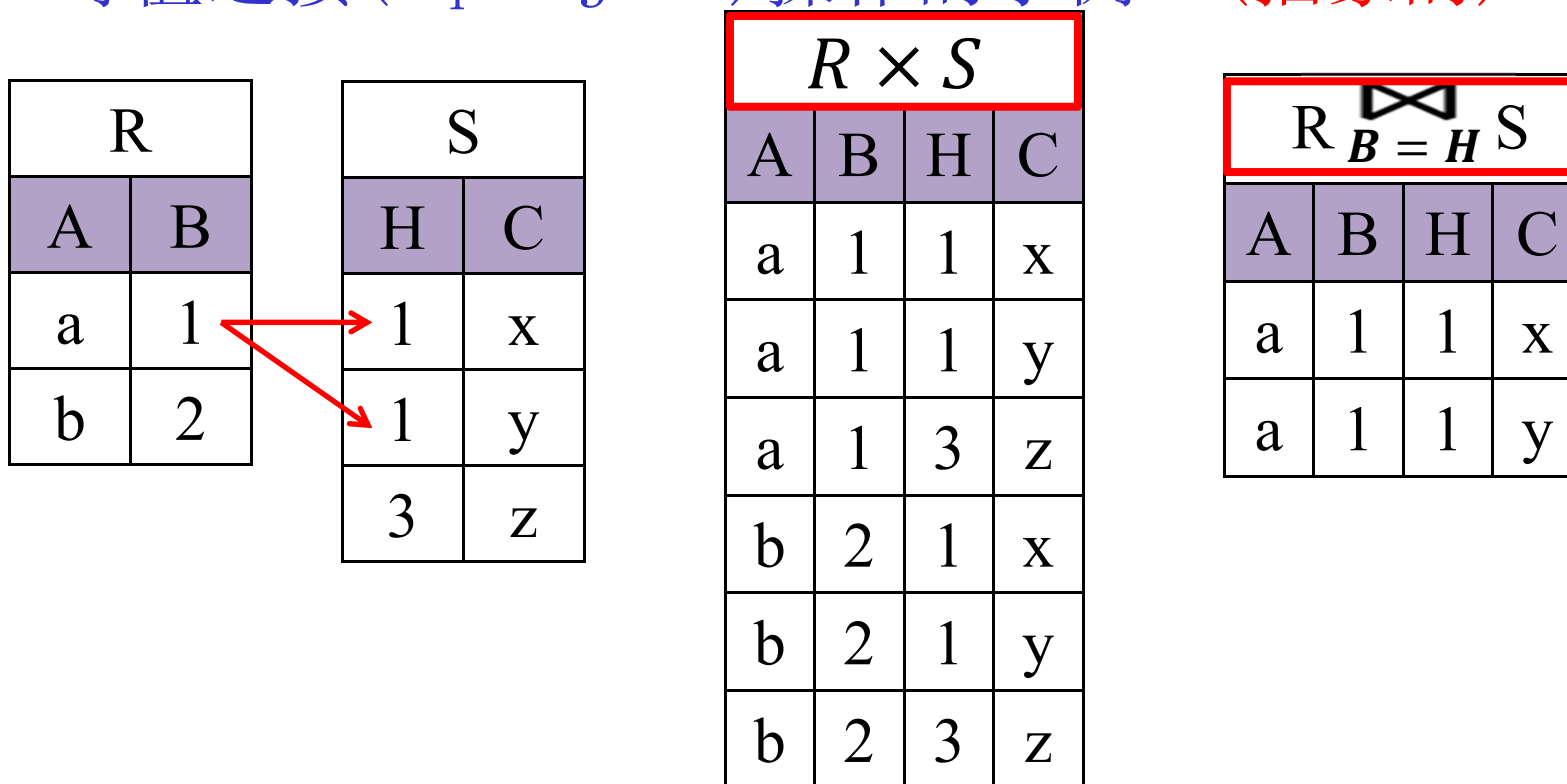
➤ 当 **θ -连接**中运算符为“=”时，就是等值连接，等值连接是 **θ -连接**的一个特例；

➤ 广义积的元组组合并不是都有意义的,另广义积的元组组合数目也非常庞大,因此采用 **θ -连接/等值连接**运算可大幅度降低中间结果的保存量,提高速度。

2.3 关系代数运算

—关系代数之纯关系操作：等值连接(Equi-Join)

➤ 等值连接(Equi-Join)操作的示例一(抽象的)



2.3 关系代数运算

—关系代数之纯关系操作：等值连接(Equi-Join)

➤ 等值连接(Equi-Join)操作的示例二(语义的)

- 员工表

Worker(W#, Wname, Wsex, Wage, Honor_type(Ht)),

- 获奖类别表Honor(Type, Title)

员工表Worker

W#	Wname	Wsex	Wage	Ht
01	张三	男	35	1
02	李四	女	49	3
03	王五	女	45	2

获奖类别表Honor

Type	Title
1	全国劳模
2	“五一”奖章获得者
3	“三八”妇女红旗手

2.3 关系代数运算

--关系代数之纯关系操作：等值连接(Equi-Join)

- 找出所有获奖员工姓名、年龄及其获奖的名称

$\Pi_{\text{wname, Wage, Title}}(\text{Worker} \bowtie_{\text{Ht=Type}} \text{Honor})$

(所有员工)的(所有获奖) ← 广义笛卡儿积

W#	Wname	Wsex	Wage	Ht	Type	Title
01	张三	男	35	1	1	全国劳模
01	张三	男	35	1	2	“五一”奖章获得者
01	张三	男	35	1	3	“三八”妇女红旗手
02	李四	女	49	3	1	全国劳模
02	李四	女	49	3	2	“五一”奖章获得者
02	李四	女	49	3	3	“三八”妇女红旗手
03	王五	女	45	2	1	全国劳模
03	王五	女	45	2	2	“五一”奖章获得者
03	王五	女	45	2	3	“三八”妇女红旗手

第一步：
对两个表
进行广义
笛卡尔积

2.3 关系代数运算

--关系代数之纯关系操作：等值连接(Equi-Join)

- 找出所有获奖员工姓名、年龄及其获奖的名称

$\Pi_{\text{wname, Wage, Title}}(\text{Worker} \bowtie_{\text{Ht=Type}} \text{Honor})$

(所有员工)的(所有获奖) \leftarrow 广义笛卡儿积

W#	Wname	Wsex	Wage	Ht	Type	Title
01	张三	男	35	1	1	全国劳模
01	张三	男	35	1	2	“五一”奖章获得者
01	张三	男	35	1	3	“三八”妇女红旗手
02	李四	女	49	3	1	全国劳模
02	李四	女	49	3	2	“五一”奖章获得者
02	李四	女	49	3	3	“三八”妇女红旗手
03	王五	女	45	2	1	全国劳模
03	王五	女	45	2	2	“五一”奖章获得者
03	王五	女	45	2	3	“三八”妇女红旗手

Ht=Type

第二步：
从广义笛
卡尔积中
选取出符
合条件的
元组

2.3 关系代数运算

--关系代数之纯关系操作：等值连接(Equi-Join)

- 找出所有获奖员工姓名、年龄及其获奖的名称

$\Pi_{\text{wname, Wage, Title}}(\text{Worker} \bowtie_{\text{Ht=Type}} \text{Honor})$

第二步：从广义笛卡尔积中选取
取出符合(Ht=Type)条件的元组

所有员工及其所获奖的信息

W#	Wname	Wsex	Wage	Ht	Type	Title
01	张三	男	35	1	1	全国劳模
02	李四	女	49	3	3	“三八” 妇女红旗手
03	王五	女	45	2	2	“五一” 奖章获得者

2.3 关系代数运算

—关系代数之纯关系操作：等值连接(Equi-Join)

- 找出所有获奖员工姓名、年龄及其获奖的名称

$\Pi_{\text{wname, Wage, Title}}(\text{Worker} \bowtie_{\text{Ht=Type}} \text{Honor})$

第三步：在(Wname, Wage, Title)上
进行投影运算，得到最终结果

所有获奖员工
姓名、年龄及
其获奖的名称：

Wname	Wage	Title
张三	35	全国劳模
李四	49	“三八” 妇女红旗手
王五	45	“五一” 奖章获得者

2.3 关系代数运算

--关系代数之纯关系操作：自然连接(Natural-Join)

➤ 定义：给定关系R和关系S, R与S的自然连接运算结果也是一个关系，记作 $R \bowtie S$, 它由关系R和关系S的笛卡尔积中选取相同属性组B上值相等的元组所构成。

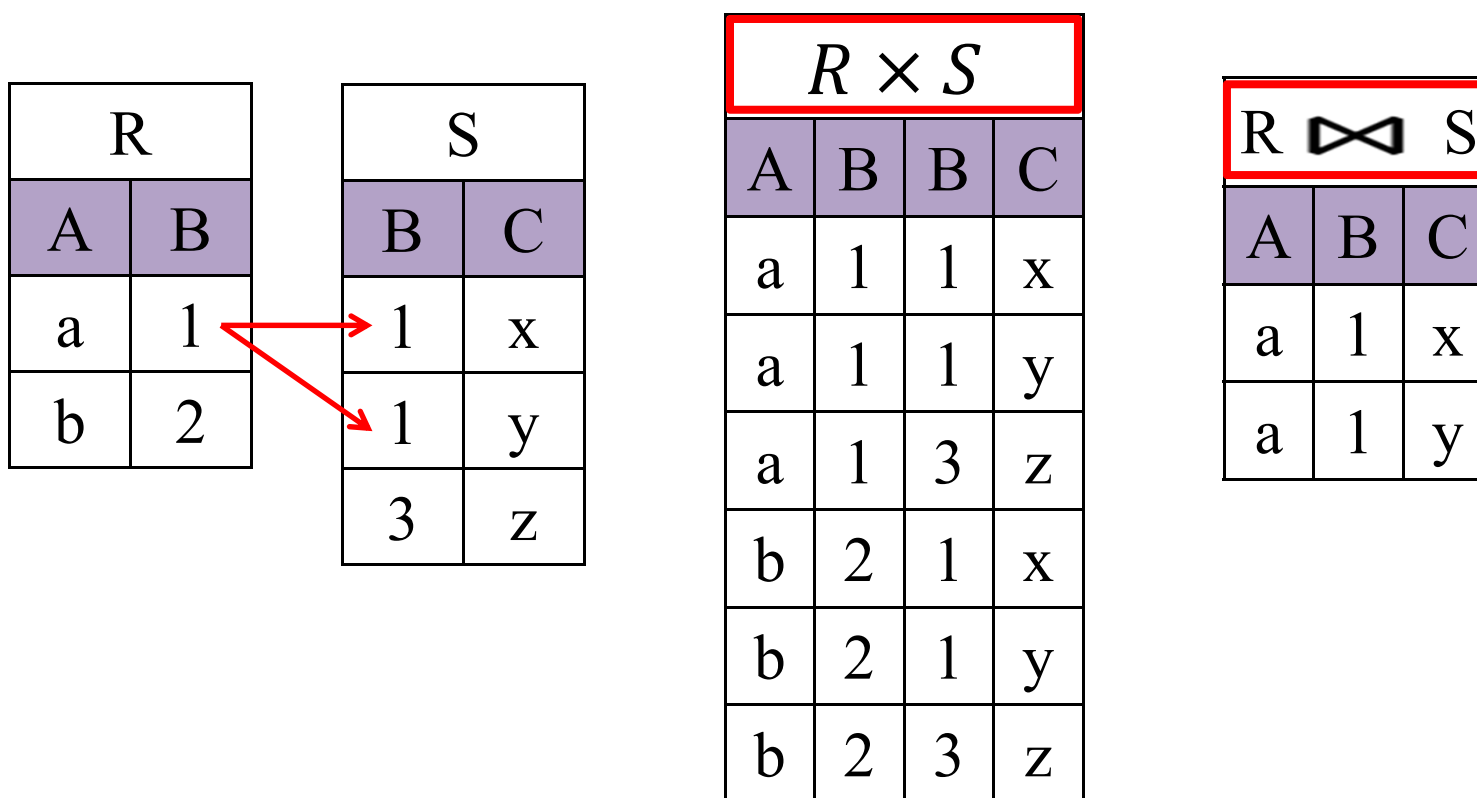
➤ 数学描述： $R \bowtie S = \sigma_{t[B]=s[B]}(R \times S)$

- ❑ 自然连接是一种特殊的等值连接
- ❑ 要求关系R和关系S必须有相同的属性组B(如R, S共有一个属性B1, 则B是B1; 如R, S共有一组属性B1, B2, ..., Bn, 则B是这些共有的所有属性)
- ❑ R, S属性相同, 值必须相等才能连接, 即 $R.B1=S.B1$ and $R.B2=S.B2 \cdots$ and $R.Bn=S.Bn$ 才能连接
- ❑ 要在结果中去掉重复的属性列(因结果中 $R.Bi$ 始终是等于 $S.Bi$ 所以可只保留一列即可)

2.3 关系代数运算

—关系代数之纯关系操作：自然连接 (Natural-Join)

➤ 自然连接 (Natural-Join) 操作的示例一 (抽象的)



2.3 关系代数运算

—关系代数之纯关系操作：自然连接(Natural-Join)

➤ 自然连接(Natural-Join)操作的示例二(语义的)

- 学生选课表SC(S#, C#, Score)
- 课程表Course(C#, Cname, Chours, Credit)

学生选课表SC

S#	C#	Score
1001	001	80
1002	002	85
1003	003	83

课程表Course

C#	Cname	Chours	Credit
001	数据库	40	4
002	数据结构	60	4
003	C语言	40	2

2.3 关系代数运算

--关系代数之纯关系操作：自然连接(Natural-Join)

➤ 自然连接(Natural-Join)操作的示例二(语义的)

• 查询所有学生选课的成绩(包括学号, 课程名称, 成绩)

$\Pi_{S\#,Cname,Score}(SC \bowtie Course)$

广义笛卡尔积

S#	C#	Score	C#	Cname	Chours	Credit
1001	001	80	001	数据库	40	4
1001	001	80	002	数据结构	60	4
1001	001	80	003	C语言	40	2
1002	002	85	001	数据库	40	4
1002	002	85	002	数据结构	60	4
1002	002	85	003	C语言	40	2
1003	003	83	001	数据库	40	4
1003	003	83	002	数据结构	60	4
1003	003	83	003	C语言	40	2

第一步：
对两个表
进行广义
笛卡尔积

2.3 关系代数运算

--关系代数之纯关系操作：自然连接(Natural-Join)

➤ 自然连接(Natural-Join)操作的示例二(语义的)

• 查询所有学生选课的成绩(包括学号, 课程名称, 成绩)

$\Pi_{S\#,Cname,Score}(SC \bowtie Course)$

广义笛卡尔积

S#	C#	Score	C#	Cname	Chours	Credit
1001	001	80	001	数据库	40	4
1001	001	80	002	数据结构	60	4
1001	001	80	003	C语言	40	2
1002	002	85	001	数据库	40	4
1002	002	85	002	数据结构	60	4
1002	002	85	003	C语言	40	2
1003	003	83	001	数据库	40	4
1003	003	83	002	数据结构	60	4
1003	003	83	003	C语言	40	2

第二步:

从广义笛卡尔积中选取
在相同列
(C#)上值相
同的元组

2.3 关系代数运算

--关系代数之纯关系操作：自然连接(Natural-Join)

➤ 自然连接(Natural-Join)操作的示例二(语义的)

• 查询所有学生选课的成绩(包括学号, 课程名称, 成绩)

$\Pi_{S\#,Cname,Score}(SC \bowtie Course)$

第二步：从广义笛卡尔积中选取在相同列(C#)上值相同的元组

S#	C#	Score	C#	Cname	Chours	Credit
1001	001	80	001	数据库	40	4
1002	002	85	002	数据结构	60	4
1003	003	83	003	C语言	40	2

第三步：去掉重复的列

S#	C#	Score	Cname	Chours	Credit
1001	001	80	数据库	40	4
1002	002	85	数据结构	60	4
1003	003	83	C语言	40	2

2.3 关系代数运算

--关系代数之纯关系操作：自然连接(Natural-Join)

➤ 自然连接(Natural-Join)操作的示例二(语义的)

- 查询所有学生选课的成绩(包括学号, 课程名称, 成绩)

$\Pi_{S\#,Cname,Score}(SC \bowtie Course)$

第四步：在(S#, Cname, Score)上进行投影操作，得到最终结果：

S#	Score	Cname
1001	80	数据库
1002	85	数据结构
1003	83	C语言

提问：如果查询所有学生选课的成绩(包括学生姓名，课程名称，成绩)

$\Pi_{Sname,Cname,Score}(SC \bowtie Course \bowtie Student)$

2.3 关系代数运算

—组合操作示例

➤ 多种操作结合的示例 (语义的)

- 查询学习课程号为002的学生学号和成绩

$$\Pi_{S\#,Score}(\sigma_{C\#="002"}(SC))$$

- 查询学习课程号为001的学生学号、姓名

$$\Pi_{S\#,Sname}(\sigma_{C\#="001"}(Student \bowtie SC))$$

- 查询学习课程名称为数据结构的学生学号、姓名和这门课程的成绩

$$\Pi_{S\#,Sname,Score}(\sigma_{Cname="数据结构"}(Student \bowtie SC \bowtie Course))$$

2.3 关系代数运算

—组合操作示例(续)

➤ 多种操作结合的示例(语义的)

- 查询学习课程号为001或002的学生的学号

$$\Pi_{S\#}(\sigma_{C\#="001" \vee C\#="002"}(SC))$$

- 查询至少学习课程号为001和002的学生的学号

□ 是否可写成如下形式呢？

~~$$\Pi_{S\#}(\sigma_{C\#="001" \wedge C\#="002"}(SC))$$~~

$$\Pi_{SC.S\#}(\sigma_{SC.C\#="001" \vee SC1.C\#="002"}(SC \bowtie_{SC.S\#=SC1.S\#} \rho_{SC1}(SC)))$$

2.3 关系代数运算

—组合操作示例(续)

➤ 多种操作结合的示例(语义的)

- 查询至少学习课程号为001和002的学生的学号

$$\Pi_{SC.S\#}(\sigma_{SC.C\#="001" \vee SC1.C\#="002"}(SC \bowtie_{SC.S\#=SC1.S\#} \rho_{SC1}(SC)))$$

- ▣ 请问：上式使用的是等值连接，换成自然连接，写成如下形式是否正确？

$$\Pi_{SC.S\#}(\sigma_{SC.S\#=SC1.S\# \wedge SC.C\#="001" \wedge SC1.C\#="002"}(SC \bowtie \rho_{SC1}(SC)))$$

2.3 关系代数运算

—组合操作示例(续)

➤ 多种操作结合的示例(语义的)

- 前例我们也可以采用交运算来实现

$$\Pi_{S\#}(\sigma_{C\#="001"}(SC) \cap \Pi_{S\#}(\sigma_{C\#="002"}(SC))$$

- 再举一个例子：查询不学习课程号为002的学生姓名和年龄。

- ▣ 同学给出了如下的查询表达式，这些表达式的结果是什么？正确吗？

$$\Pi_{Sname,Sage}(\sigma_{C\#<>"002"}(Student \bowtie SC))$$

$$\Pi_{Sname,Sage}(Student - (\sigma_{C\#="002"}(Student \bowtie SC))$$

$$\Pi_{Sname,Sage}(Student) - \Pi_{Sname,Sage}(\sigma_{C\#="002"}(Student \bowtie SC))$$

2.3 关系代数运算

—组合操作示例(续)

➤ 书写关系代数表达式的基本思路

- 检索是否涉及多个表，如不涉及，则可直接采用并、差、交、选择与投影，只要注意条件书写正确与否即可
- 如涉及多个表，则检查
 - ▣ 能否使用自然连接，将多个表连接起来(多数情况是这样的)
 - ▣ 如不能，能否使用等值或不等值连接(θ -连接)
 - ▣ 还不能，则使用广义笛卡尔积，注意相关条件的书写
- 连接完后，可以继续使用选择、投影等运算，即所谓数据库的“**选投联**”操作

$$\Pi_{\text{Sname,Sage}}(\sigma_{C\#="002"}(S \bowtie SC))$$

2.3 关系代数运算

--关系代数之纯关系操作：除(Division)

- 下面介绍关系的除法，除法运算经常用于求解“查询… 全部的…”问题
- **前提条件：**给定关系 $R(A_1, A_2, \dots, A_n)$ 为 n 度关系和关系 $S(B_1, B_2, \dots, B_m)$ 为 m 度关系，如果可以进行关系 R 与关系 S 的除运算，**当且仅当：**属性集 $\{B_1, B_2, \dots, B_m\}$ 是属性集 $\{A_1, A_2, \dots, A_n\}$ 的**真子集**，即 $m < n$ 。
- **定义：**关系 R 和关系 S 的除运算结果也是一个关系，记作 $R \div S$

2.3 关系代数运算

--关系代数之纯关系操作：除(Division) (续)

- 我们先看 $R \div S$ 结果的属性应有哪些，然后再看 $R \div S$ 的元组怎样形成。
- 设属性集 $\{C1, C2, \dots, Ck\} = \{A1, A2, \dots, An\} - \{B1, B2, \dots, Bm\}$ ，则有 $k=n-m$ 则 $R \div S$ 结果关系是一 k 度($n-m$ 度)关系，由 $\{C1, C2, \dots, Ck\}$ 属性构成

R		
A1	A2	A3
a	b	c
a	b	c
a	e	c
a	e	f

S	
A2	A3
b	c

$R \div S$
A1
a
b

2.3 关系代数运算

--关系代数之纯关系操作：除(Division) (续)

- 再设关系 $R(\langle a1, \dots, an \rangle)$ 和关系 $S(\langle b1, \dots, bm \rangle)$, 那么 $R \div S$ 结果关系为元组 $\langle c1, \dots, ck \rangle$ 的集合, 元组 $\langle c1, \dots, ck \rangle$ 满足下述条件: 它与 S 中每一个元组 $\langle b1, \dots, bm \rangle$ 组合形成的一个新元组都是 R 中的某一个元组 $\langle a1, \dots, an \rangle$

(其中, $a1, \dots, an, b1, \dots, bm, c1, \dots, ck$ 分别是属性 $A1, \dots, An, B1, \dots, Bm, C1, \dots, Ck$ 的值)

- 数学描述:

$$R \div S = \{t | t \in \prod_{R-S}(R) \wedge \forall u \in S (tu \in R)\}$$

$R \div S$: 在 R 中, 找出与 S 中所有的元组有关系的 R 元组

2.3 关系代数运算

—关系代数之纯关系操作：除(Division) (续)

➤ 除(Division)操作的示例一(抽象的)

R		
A1	A2	A3
a	b	c
d	b	c
a	e	c
a	e	f
d	b	f
a	e	g
a	e	h
a	b	l

S	$R \div S$
A3	A1 A2
c	a b
	d b
	a e

(1)

S	$R \div S$
A3	A1 A2
c	d b
f	a e

(2)

2.3 关系代数运算

—关系代数之纯关系操作：除(Division) (续)

➤ 除(Division)操作的示例一(抽象的)

R		
A1	A2	A3
a	b	c
d	b	c
a	e	c
a	e	f
d	b	f
a	e	g
a	e	h
a	b	l

S	$R \div S$
A3	A1 A2
c	a e
f	
g	
h	

(3)

S	$R \div S$
A2 A3	A1
b c	a
	d

(4)

2.3 关系代数运算

--关系代数之纯关系操作：除(Division)

➤ 除(Division)操作的示例二(语义的)

- 查询选修了全部课程的学生的学号

$$\Pi_{s\#,c\#}(SC) \div \Pi_{c\#}(Course)$$

学生选课表SC

S#	C#	Score
1001	001	80
1001	002	85
1001	003	83
1002	001	78
1002	002	83
1003	001	87
1003	002	83
1004	001	79

课程表Course

C#	Cname	Chours	Credit
001	数据库	40	4
002	数据结构	60	4
003	C语言	40	2

选修了全部课程
的学生的学号

S#
1001

2.3 关系代数运算

--关系代数之纯关系操作：除(Division)

➤ 除(Division)操作的示例三(语义的)

- 查询选修了学号1001学生所学全部课程的同学的姓名

$$\Pi_{\text{Sname}} \left(s \bowtie \left(\Pi_{\text{S\#,C\#}}(sc) \div \Pi_{\text{C\#}}(\sigma_{\text{S\#}="1001"}(sc)) \right) \right)$$

▣ 请问下述写法与上有何不同？结果是否一样

$$\Pi_{\text{Sname}} \left(s \bowtie \left(sc \div \Pi_{\text{C\#}}(\sigma_{\text{S\#}="1001"}(sc)) \right) \right)$$

2.3 关系代数运算

--关系代数之纯关系操作：外连接(Outer-Join)

➤ 下面介绍关系的外连接操作

➤ 外连接问题的提出：先看例子

- Teacher (T#, Tname, Salary),
- Teach (T#, C#),
- Course (C#, Cname)

Teacher

T#	Tname	Salary
001	赵三	1200
002	赵四	1400
003	赵五	1000
004	赵六	1100

Teach

T#	C#
001	001
002	002
004	002

Course

C#	Cname
001	物理
002	数学
003	化学

数据库系统

- Teacher(T#,Tname,Salary),Teach(T#,C#),Course(C#,Cname)

T#	Tname	Salary	C#	Cname
001	赵三	1200	001	物理
002	赵四	1400	002	数学
004	赵六	1100	003	数学

- 请列出所有老师的有关信息,包括姓名,工资,所教课程等
 $\pi_{T\#,Tname,Salary,C\#,Cname}(Teacher \bowtie Teach \bowtie Course)$
- 按上式连接的结果,003号教师的姓名和工资信息丢失了
- 因为在Teach表中没有和003号教师相匹配的元组,元组003号教师(又称为失配元组)不能和其他表的元组形成连接元组,信息因而丢失。
- 怎样保证使003号教师信息仍旧出现在结果关系中呢?
----这就需要外连接----

2.3 关系代数运算

—关系代数之纯关系操作：外连接(Outer-Join)

- 定义：两个关系R与S进行连接时, 如果关系R(或S)中的元组在S(或R)中找不到相匹配的元组, 则为了避免该元组信息丢失, 从而将该元组与S(或R)中假定存在的全为空值的元组形成连接, 放置在结果关系中, 这种连接称之为外连接(Outer Join)。

R和S的外连接(city值相等)

R

S#	City
S7	威海
S8	烟台
S9	青岛

S

P#	City
P7	威海
P8	北京
P9	上海

S#	R.City	P#	S.City
S7	威海	P7	威海
?	?	P8	北京
?	?	P9	上海
S8	烟台	?	?
S9	青岛	?	?

2.3 关系代数运算

--关系代数之纯关系操作：外连接(Outer-Join)

➤ 外连接 = 自然连接(或 θ 连接)+失配的元组(与全空元组形成的连接)

➤ 外连接的形式：左外连接、右外连接、全外连接

▣ 左外连接 = 自然连接(或 θ 连接)+左侧表中失配的元组

▣ 右外连接 = 自然连接(或 θ 连接)+右侧表中失配的元组

▣ 全外连接 = 自然连接(或 θ 连接)+两侧表中失配的元组

➤ 左外连接(Left Outer Join)记为： $R \bowtie\!\!\!\lrcorner S$

➤ 右外连接(Right Outer Join)记为： $R \bowtie\!\!\!\rceil S$

➤ 全外连接(Full Outer Join)记为： $R \bowtie\!\!\!\times\!\!\!\rceil S$

2.3 关系代数运算

--关系代数之纯关系操作：外连接(Outer-Join)

➤ 外连接操作示例

R

S#	City
S7	威海
S8	烟台
S9	青岛

S

P#	City
P7	威海
P8	北京
P9	上海

R和S的左外连接(city值相等)

S#	R.City	P#	S.City
S7	威海	P7	威海
S8	烟台	?	?
S9	青岛	?	?

R和S的右外连接(city值相等)

S#	R.City	P#	S.City
S7	威海	P7	威海
?	?	P8	北京
?	?	P9	上海

2.3 关系代数运算

--关系代数之纯关系操作：外连接(Outer-Join)

► 外连接操作示例

R

S#	City
S7	威海
S8	烟台
S9	青岛

S

P#	City
P7	威海
P8	北京
P9	上海

R和S的全外连接(city值相等)

S#	R.City	P#	S.City
S7	威海	P7	威海
?	?	P8	北京
?	?	P9	上海
S8	烟台	?	?
S9	青岛	?	?

R和S的等连接(city值相等)

S#	R.City	P#	S.City
S7	威海	P7	威海

2.3 关系代数运算

--关系代数之纯关系操作：外连接(Outer-Join)

➤ 外连接操作示例

- 前面问题例子的解决方案：查询所有老师的信息

$\pi_{T\#,Tname,Salary,C\#,Cname}(Teacher \bowtie Teach \bowtie Course)$

Teacher

T#	Tname	Salary
001	赵三	1200
002	赵四	1400
003	赵五	1000
004	赵六	1100

Teach

T#	C#
001	001
002	002
004	002

Course

C#	Cname
001	物理
002	数学
003	化学

=

T#	Tname	Salary	C#	Cname
001	赵三	1200	001	物理
002	赵四	1400	002	数学
003	赵五	1000	null	null
004	赵六	1100	003	数学

2.3 关系代数运算

--关系代数之纯关系操作：外连接(Outer-Join)

➤ 外连接操作示例

- 查询所有课程的信息

$\pi_{T\#,Tname,Salary,C\#,Cname}(Teacher \bowtie_{\square} Teach \bowtie_{\square} Course)$

Teacher

T#	Tname	Salary
001	赵三	1200
002	赵四	1400
003	赵五	1000
004	赵六	1100

Teach

T#	C#
001	001
002	002
004	002

Course

C#	Cname
001	物理
002	数学
003	化学

=

T#	Tname	Salary	C#	Cname
001	赵三	1200	001	物理
002	赵四	1400	002	数学
004	赵六	1100	003	数学
null	null	null	003	化学

2.3 关系代数运算

--关系代数之纯关系操作：外连接(Outer-Join)

➤ 外连接操作示例

- 查询所有老师和所有课程的信息

$\pi_{T\#,Tname,Salary,C\#,Cname}(Teacher \bowtie Teach \bowtie Course)$

Teacher

T#	Tname	Salary
001	赵三	1200
002	赵四	1400
003	赵五	1000
004	赵六	1100

\bowtie

Teach

T#	C#
001	001
002	002
004	002

\bowtie

Course

C#	Cname
001	物理
002	数学
003	化学

=

T#	Tname	Salary	C#	Cname
001	赵三	1200	001	物理
002	赵四	1400	002	数学
003	赵五	1000	002	数学
004	赵六	1100	null	null
null	null	null	003	化学

第2章关系模型与关系运算

2.1 关系模型与关系运算简述

2.2 关系与关系模型

2.3 关系代数运算

2.4 关系元组演算

- 关系演算概述

- 关系元组演算

- 元组演算公式

- 元组演算符的优先次序

- 元组演算符的等价性

- 用元组演算公式实现关系代数公式

2.5 关系域演算及QBE

2.6 小结

2.4 关系元组演算

—关系演算概述

➤ 我们在前面已经见过关系演算的基本形式:

- 如, 并运算定义中: $R \cup S = \{r \mid r \in R \vee r \in S\}$
- 再如, 差运算定义中: $R - S = \{r \mid r \in R \wedge r \notin S\}$
- 又如, 交运算定义中: $R \cap S = \{r \mid r \in R \wedge r \in S\}$

以上花括号内的演算公式既为关系演算公式

2.4 关系元组演算

—关系演算概述(续)

- 关系演算是以数理逻辑中的谓词演算为基础的
- 关系演算是描述关系运算的另一种思维方式
- SQL语言是继承了关系代数和关系演算各自的优点所形成的
- 按照谓词变量的不同，可分为关系元组演算和关系域演算形式：
 - ❑ 关系元组演算是以元组变量作为谓词变量的基本对象
 - ❑ 关系域演算是以域变量作为谓词变量的基本对象，域演算将在下一小节介绍

2.4 关系元组演算

—关系元组演算

➤ 关系元组演算的基本形式:

$$\{t \mid P(t)\}$$

上式表示：所有使谓词P为真的元组t的集合

- t是元组变量
- $t \in r$ 表示元组t在关系r中
- $t[A]$ 表示元组t的分量，即t在属性A上的值
- P是与谓词逻辑相似的公式， $P(t)$ 表示以元组t为变量的公式

2.4 关系元组演算

--关系元组演算(续)

➤ $P(t)$ 可以是如下三种形式之一的原子公式:

□ $t \in R$

- t 是关系 R 中的一个元组
- 例如: $\{t \mid t \in \text{Student}\}$

□ $s[A] \theta c$

- 元组分量 $s[A]$ 与常量 c 之间满足比较关系 θ
- θ : 比较运算符 $<, \leq, =, <>, >, \geq$
- 例如: $\{t \mid t \in R \wedge t[\text{Sage}] \leq 19 \wedge t[\text{Sname}] = \text{'张三'}\}$

□ $s[A] \theta u[B]$

- $s[A]$ 与 $u[B]$ 为元组分量, A 和 B 分别是某些关系的属性, 他们之间满足比较关系 θ
- 例如: $\{t \mid t \in \text{Student} \wedge \exists (u \in \text{Student}) (t[\text{Sage}] > u[\text{Sage}])\}$
“检索出年龄不是最小的所有同学”

2.4 关系元组演算

--关系元组演算（续）

➤ $P(t)$ 可以由公式加运算符 \wedge (与)、 \vee (或)、 \neg (非)递归构成

- 如果 F 是一个公式，则 $\neg F$ 也是公式
- 如果 F_1 、 F_2 是公式，则 $F_1 \wedge F_2$ ， $F_1 \vee F_2$ 也是公式

F1	F2	F1 \wedge F2
真/True	真/True	
真/True	假/False	
假/False	真/True	
假/False	假/False	

F1	F2	F1 \vee F2
真/True	真/True	
真/True	假/False	
假/False	真/True	
假/False	假/False	

F1	$\neg F_1$
真/True	
假/False	

2.4 关系元组演算

--关系元组演算（续）

➤ $P(t)$ 可以由公式加运算符 \wedge (与)、 \vee (或)、 \neg (非)递归构成

- 如果 F 是一个公式，则 $\neg F$ 也是公式
- 如果 F_1 、 F_2 是公式，则 $F_1 \wedge F_2$ ， $F_1 \vee F_2$ 也是公式

F1	F2	F1 \wedge F2
真/True	真/True	真/True
真/True	假/False	假/False
假/False	真/True	假/False
假/False	假/False	假/False

F1	F2	F1 \vee F2
真/True	真/True	真/True
真/True	假/False	真/True
假/False	真/True	真/True
假/False	假/False	假/False

F1	$\neg F_1$
真/True	假/False
假/False	真/True

2.4 关系元组演算

--关系元组演算（续）

➤ $P(t)$ 可以由公式加运算符 \wedge (与)、 \vee (或)、 \neg (非) 递归构成

▣ 例如：检索出年龄小于20岁并且是男同学的所有学生

$\{ t \mid t \in \text{Student} \wedge t[\text{Sage}] < 20 \wedge t[\text{Ssex}] = \text{'男'} \}$

▣ 再例如：检索出年龄小于20岁或者03系的所有男学生。

$\{ t \mid t \in \text{Student} \wedge (t[\text{Sage}] < 20 \vee t[\text{D\#}] = \text{'03'}) \wedge t[\text{Ssex}] = \text{'男'} \}$

在元组演算公式构造过程中，如果需要，可以使用括号，通过括号改变运算的优先次序，即：括号内的运算优先计算。

▣ 再例如：检索出不是03系的所有学生

$\{ t \mid t \in \text{Student} \wedge \neg (t[\text{D\#}] = \text{'03'}) \}$

▣ 再例如：检索不是(小于20岁的男同学)的所有同学

$\{ t \mid t \in \text{Student} \wedge \neg (t[\text{Sage}] < 20 \wedge t[\text{Ssex}] = \text{'男'}) \}$

2.4 关系元组演算

--关系元组演算（续）

- 构造 $P(t)$ 还有两个运算符： \exists （存在）、 \forall （任意）
 - ▣ 如果 F 是一个公式，则 $\exists(t \in r)(F(t))$ 也是公式
 - ▣ 如果 F 是一个公式，则 $\forall(t \in r)(F(t))$ 也是公式
- 运算符 \exists 和 \forall ，又称为量词，前者称“存在量词”，后者称“全称量词”
- 而被 \exists 或 \forall 限定的元组变量 t ，或者说，元组变量 t 前有存在量词或全称量词，则该变量被称为“约束变量”，否则被称为“自由变量”。

$t \in r$	$F(t)$	$\exists(t \in r)(F(t))$
对 r 中的每一个 t 进行 $F(t)$ 的检验	所有 t 都使 $F(t)=\text{假}$	假/False
	有一个 t 使 $F(t)=\text{真}$	真/True

$t \in r$	$F(t)$	$\forall(t \in r)(F(t))$
对 r 中的每一个 t 进行 $F(t)$ 的检验	所有 t 都使 $F(t)=\text{真}$	真/True
	有一个 t 使 $F(t)=\text{假}$	假/False

2.4 关系元组演算

—关系元组演算（续）

➤ 构造 $P(t)$ 还有两个运算符： \exists （存在）、 \forall （任意）

▣ 例如：“检索出年龄不是最小的所有同学”

$\{t \mid t \in \text{Student} \wedge \exists(u \in \text{Student}) (t[\text{Sage}] > u[\text{Sage}])\}$

请大家写一下，在关系代数中，如何表达上面的查询需求？

▣ 再例如：检索出课程都及格的所有同学

$\{t \mid t \in \text{Student} \wedge \forall(u \in \text{SC} \wedge t[\text{S\#}] = u[\text{S\#}]) (u[\text{Score}] \geq 60)\}$

用关系代数，如何书写上面例子？

▣ 请注意上式写成下面的公式会表达什么意思？

$\{t \mid t \in \text{Student} \wedge \forall(u \in \text{SC}) (t[\text{S\#}] = u[\text{S\#}] \wedge u[\text{Score}] \geq 60)\}$

2.4 关系元组演算

--关系元组演算（续）

➤ 我们再举几个例子

- 例如：检索计算机系的所有同学

$\{t \mid t \in \text{Student} \wedge \exists(u \in \text{Dept}) (t[\text{D\#}] = u[\text{D\#}] \wedge u[\text{Dname}] = \text{'计算机'})\}$

- 例如：检索出比张三年龄小的所有同学

$\{t \mid t \in \text{Student} \wedge \exists(w \in \text{Student})(w[\text{Sname}] = \text{'张三'} \wedge t[\text{Sage}] < w[\text{Sage}])\}$

2.4 关系元组演算

--关系元组演算（续）

➤ 我们再举几个例子(续)

- 例如：检索学过所有课程的同学

$\{t \mid t \in \text{Student} \wedge \forall (u \in \text{Course})(\exists (s \in \text{SC})(s[S\#]=t[S\#] \wedge u[C\#]=s[C\#]))\}$

- 例如：检索所有同学所有课程全都及格的系

$\{t \mid t \in \text{Dept} \wedge \forall (s \in \text{Student} \wedge s[D\#]=t[D\#])(\forall (u \in \text{SC} \wedge s[S\#]=u[S\#]) (u[\text{Score}] \geq 60))\}$

2.4 关系元组演算

—关系元组演算（续）

➤ $\{t \mid P(t)\}$ 中公式 $P(t)$ 的构造小结

- 三种形式的原子公式是公式

(1) $s \in R$ (2) $s[A] \theta c$ (3) $s[A] \theta u[B]$

- 如果 P 是公式，那么 $\neg P$ 也是公式
- 如果 P_1, P_2 是公式，则 $P_1 \wedge P_2, P_1 \vee P_2$ 也是公式
- 如果 $P(t)$ 是公式， R 是关系，则 $\exists (t \in R)(P(t))$ 和 $\forall (t \in R)(P(t))$ 也是公式
- 需要时可加括弧
- 上述运算符的优先次序自高至低为：括弧； θ ； \exists ； \forall ； \neg ； \wedge ； \vee ；
- 公式只限于以上形式

2.4 关系元组演算

一元组演算符的优先次序

➤ $P(t)$ 运算符优先次序为:

括弧; θ ; \exists ; \forall ; \neg ; \wedge ; \vee ;

示例, 请注意下述语句的结果差异

$\{ t \mid t \in \text{Student} \wedge (t[\text{Sage}] < 20 \vee t[\text{D\#}] = '03' \wedge t[\text{Ssex}] = \text{'男'}) \}$

$\{ t \mid t \in \text{Student} \wedge ((t[\text{Sage}] < 20 \vee t[\text{D\#}] = '03') \wedge t[\text{Ssex}] = \text{'男'}) \}$

$\{ t \mid t \in \text{Student} \wedge \neg (t[\text{Sage}] < 20 \wedge t[\text{Ssex}] = \text{'男'}) \}$

$\{ t \mid t \in \text{Student} \wedge \forall (u \in \text{SC}) (t[\text{S\#}] = u[\text{S\#}] \wedge u[\text{Score}] \geq 60) \}$

2.4 关系元组演算

一元组演算公式的等价性

➤ P(t)公式,如谓词演算一样,也有一系列演算的等价性

▣ 例如: θ 与 \neg 的等价性(符号 \Leftrightarrow 表示等价于)

$$\neg(\alpha = \beta) \Leftrightarrow \alpha <> \beta$$

$$\neg(\alpha > \beta) \Leftrightarrow \alpha \leq \beta \Leftrightarrow \alpha < \beta \vee \alpha = \beta$$

$$\neg(\alpha < \beta) \Leftrightarrow \alpha \geq \beta \Leftrightarrow \alpha > \beta \vee \alpha = \beta$$

$$\neg(\alpha \geq \beta) \Leftrightarrow \alpha < \beta$$

$$\neg(\alpha \leq \beta) \Leftrightarrow \alpha > \beta$$

$$\neg(\alpha <> \beta) \Leftrightarrow \alpha = \beta$$

▣ 等价性示例:

$\{ t \mid t \in \text{Student} \wedge \neg (t[\text{D\#}] = \text{'03'}) \}$ 等价于

$\{ t \mid t \in \text{Student} \wedge t[\text{D\#}] <> \text{'03'} \}$

2.4 关系元组演算

一元组演算公式的等价性(续)

➤ $P(t)$ 公式,如谓词演算一样,也有一系列演算的等价性

▣ 再例如 \wedge 、 \vee 与 \neg 运算之间的等价性)

$$P1 \wedge P2 \Leftrightarrow \neg(\neg P1 \vee \neg P2)$$

n 个否定的或操作的再否定,便是 n 个肯定的与操作

$$P1 \vee P2 \Leftrightarrow \neg(\neg P1 \wedge \neg P2)$$

n 个否定的与操作的再否定,便是 n 个肯定的或操作

▣ 等价性示例:

“或者学过001课程或者学过002课程”

$$\{u \mid u \in SC \wedge (u[C\#] = '001' \vee u[C\#] = '002')\}$$

等价于“去掉既未学过001课程又未学过002课程的所有人”

2.4 关系元组演算

一元组演算公式的等价性(续)

➤ $P(t)$ 公式,如谓词演算一样,也有一系列演算的等价

▣ 等价性再示例:

“既年龄小于20岁又是男性的所有学生”

$\{s \mid s \in \text{Student} \wedge (s[\text{Sage}] < 20 \wedge s[\text{Ssex}] = \text{'男'})\}$

等价于“去掉: 或者年龄大于等于20岁, 或者不是男性的所有人”

$\{s \mid s \in \text{Student} \wedge \neg (s[\text{Sage}] \geq 20 \vee s[\text{Ssex}] \neq \text{'男'})\}$

2.4 关系元组演算

一元组演算公式的等价性(续)

➤ $P(t)$ 公式, 如谓词演算一样, 也有一系列演算的等价

▣ 还例如 \exists 、 \forall 与 \neg 运算之间的等价性

$$\forall(t \in R)(P(t)) \Leftrightarrow \neg(\exists(t \in R)(\neg P(t)))$$

$$\exists(t \in R)(P(t)) \Leftrightarrow \neg(\forall(t \in R)(\neg P(t)))$$

▣ 等价性示例

“既学过001课程又学过002课程的学生”

$$\{t \mid t \in \text{Student} \wedge \exists(s \in \text{SC} \wedge u \in \text{SC} \wedge s[S\#] = t[S\#] \wedge u[S\#] = t[S\#]) (s[C\#] = '001' \wedge u[C\#] = '002'))\}$$

等价于“去掉：或者未学过001课程，或者未学过002课程的所有人”

$$\{t \mid t \in \text{Student} \wedge \neg(\forall(s \in \text{SC} \wedge s[S\#] = t[S\#])(s[C\#] \neq '001') \vee \forall(s \in \text{SC} \wedge s[S\#] = t[S\#])(s[C\#] \neq '002')))\}$$

2.4 关系元组演算

——用元组演算公式实现关系代数

➤ 关系代数有五种基本操作：并、差、广义积、选择、投影操作，还有：交、 θ -连接操作。

□ 并运算： $R \cup S = \{ t \mid t \in R \vee t \in S \}$

□ 差运算： $R - S = \{ t \mid t \in R \wedge t \notin S \}$

□ 交运算： $R \cap S = \{ t \mid t \in R \wedge t \in S \}$

□ 广义笛卡尔积

$R(A) \times S(B) = \{ t \mid \exists (u \in R) \exists (s \in S) (t[A] = u[A] \wedge t[B] = s[B]) \}$

□ 选择运算： $\sigma_{con}(R) = \{ t \mid t \in R \wedge F(con) \}$

□ 投影运算： $\pi_A(R) = \{ t[A] \mid t \in R \}$

2.4 关系元组演算

一元组演算公式与关系代数对比应用的例子

➤ 已知:

- ▣ 学生关系: Student(S#, Sname, Sage, Ssex, Sclass)
- ▣ 课程关系: Course(C#, Cname, Chours, Credit, Tname)
- ▣ 选课关系: SC(S#, C#, Score)

➤ 求学过李明老师讲授所有课程的学生姓名(全都学过)

✓ $\pi_{Sname}((Student \bowtie Course \bowtie SC) \div \sigma_{Tname='李明'}(C))$

✓ $\{ t[Sname] \mid t \in Student \wedge \forall (u \in Course \wedge u[Tname]='李明') \}$
 $(\exists (w \in SC)(w[S\#]=t[S\#] \wedge w[C\#]=u[C\#])) \}$

2.4 关系元组演算

一元组演算公式与关系代数对比应用的例子(续)

➤ 已知:

- ▣ 学生关系: Student(S#, Sname, Sage, Ssex, Sclass)
- ▣ 课程关系: Course(C#, Cname, Chours, Credit, Tname)
- ▣ 选课关系: SC(S#, C#, Score)

➤ 求没学过李明老师讲授任一门课程的学生姓名(全没学过)

✓ $\pi_{Sname}(\text{Student}) - \pi_{Sname}(\sigma_{Tname='李明'}(\text{Student} \bowtie \text{Course} \bowtie \text{SC}))$

✓ $\{ t[Sname] \mid t \in \text{Student} \wedge \forall (u \in \text{Course} \wedge u[Tname]='李明') (\neg \exists (w \in \text{SC})(w[S\#]=t[S\#] \wedge w[C\#]=u[C\#])) \}$

2.4 关系元组演算

一元组演算公式与关系代数对比应用的例子(续)

➤ 已知:

- ▣ 学生关系: Student(S#, Sname, Sage, Ssex, Sclass)
- ▣ 课程关系: Course(C#, Cname, Chours, Credit, Tname)
- ▣ 选课关系: SC(S#, C#, Score)

➤ 求至少学过一门李明老师讲授课程的学生姓名(至少学过一门)

✓ $\pi_{Sname}(\sigma_{Tname='李明'}(Student \bowtie Course \bowtie SC))$

✓ $\{ t[Sname] \mid t \in Student \wedge \exists(u \in Course) \exists(w \in SC) (u[Tname]='李明' \wedge w[S\#]=t[S\#] \wedge w[C\#]=u[C\#]) \}$

2.4 关系元组演算

一元组演算公式与关系代数对比应用的例子(续)

➤ 已知:

- ▣ 学生关系: Student(S#, Sname, Sage, Ssex, Sclass)
- ▣ 课程关系: Course(C#, Cname, Chours, Credit, Tname)
- ▣ 选课关系: SC(S#, C#, Score)

➤ 求至少有一门李明老师讲授课程没有学过的学生姓名(至少有一门没学过)

- ✓ $\pi_{Sname}(\text{Student}) - \pi_{Sname}((\text{Student} \bowtie \text{Course} \bowtie \text{SC}) \div \sigma_{Tname='李明'}(C))$
- ✓ $\{ t[Sname] \mid t \in \text{Student} \wedge \exists (u \in \text{Course} \wedge u[Tname]='李明') (\neg \exists (w \in \text{SC}) \wedge w[S\#]=t[S\#] \wedge w[C\#]=u[C\#]) \}$

第2章 关系模型与关系运算

2.1 关系模型与关系运算简述

2.2 关系与关系模型

2.3 关系代数运算

2.4 关系元组演算

2.5 关系域演算及QBE

- 关系域演算简要介绍

- 关系域演算与关系元组演算比较

- QBE的基本形式

- 用QBE实现关系代数公式

2.6 小结

2.5 关系域演算及QBE

—关系域演算简要介绍

➤ 关系域演算的基本形式

$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$

其中, x_i 代表域变量或常量, P 为以 x_i 为变量的公式. P 的构造如下:

▣ 三种形式的原子公式是公式

✓ $\langle x_1, x_2, \dots, x_n \rangle \in R$

x_i 代表域变量或常量, 表示由域变量构成的 $\langle x_1, x_2, \dots, x_n \rangle$ 是属于关系 R 的

✓ $x \theta c$

域变量 x 与常量 c 之间满足比较关系

θ : 比较运算符 $<, \leq, =, <>, >, \geq$

✓ $x \theta y$

域变量 x 与域变量 y 之间满足比较关系 θ

2.5 关系域演算及QBE

—关系域演算简要介绍(续)

➤ 关系域演算的基本形式

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

其中, x_i 代表域变量或常量, P 为以 x_i 为变量的公式. P 的构造如下:

- 如果 P 是公式, 那么 $\neg P$ 也是公式
- 如果 P_1, P_2 是公式, 则 $P_1 \wedge P_2, P_1 \vee P_2$ 也是公式
- 如果 P 是公式, x 是域变量, 则 $\exists(x)(P(x))$ 和 $\forall(x)(P(x))$ 也是公式
- 需要时可加括弧
- 上述运算符的优先次序自高至低为: 括弧; \neg ; \exists ; \forall ; \wedge ; \vee ;
- 公式只限于以上形式

2.5 关系域演算及QBE

—关系域演算简要介绍(续)

- 例如：检索出不是03系的所有学生

$\{ \langle a,b,c,d,e,f \rangle \mid \langle a,b,c,d,e,f \rangle \in \text{Student} \wedge e \neq '03' \}$

- 再例如：检索不是(小于20岁的男同学)的所有同学的姓名

$\{ \langle b \rangle \mid \exists a,c,d,e,f (\langle a,b,c,d,e,f \rangle \in \text{Student} \wedge \neg (d < 20 \wedge c = \text{'男'})) \}$

- 再例如：检索成绩不及格的同学姓名、课程及其成绩

$\{ \langle b,h,m \rangle \mid \exists a,c,d,e,f,g,I,j,k (\langle a,b,c,d,e,f \rangle \in \text{Student} \wedge (g,h,I,j,k) \in \text{Course} \wedge (a,g,m) \in \text{SC} \wedge m < 60) \}$

2.5 关系域演算及QBE

—关系域演算与关系演算的比较

- 元组演算的基本形式: $\{ t | P(t) \}$
- 域演算的基本形式: $\{ \langle x_1, x_2, \dots, x_n \rangle | P(x_1, x_2, \dots, x_n) \}$
- 元组演算是以元组为变量, 以元组为基本处理单位, 先找到元组, 然后再找到元组分量, 进行谓词判断;
- 域演算是以域变量为基本处理单位, 先有域变量, 然后再判断由这些域变量组成的元组是否存在或是否满足谓词判断。
- 公式的构造过程是相似的
- 元组演算和域演算可以等价互换。
- 下面以QBE为例, 介绍域演算的应用

2.5 关系域演算及QBE

——域演算语言QBE

- QBE: Query By Example (实例查询语言)
- 1975年由M. M. Zloof提出, 1978年在IBM370上实现
- 特点: 操作独特, 基于屏幕表格的查询语言, 不用书写复杂的公式, 只需将条件填在表格中即可
- 是一种高度非过程化的基于屏幕表格的查询语言
- 特别适合于终端用户的使用 (输入和输出)

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P.X	Female	<17		

2.5 关系域演算及QBE

--QBE与SQL的区别

➤ QBE与SQL的最大区别:

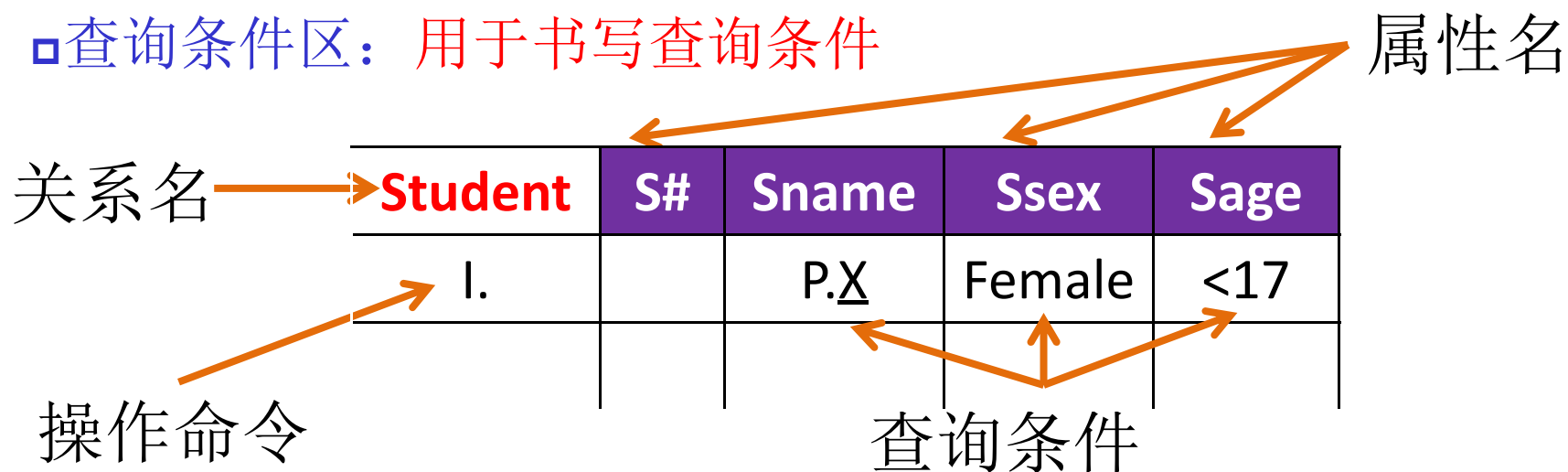
- ▣ QBE具有图形用户界面，允许用户通过在屏幕上创建示例表来编写查询。
- ▣ QBE特别适合于不太复杂、可用几个表描述的查询。

2.5 关系域演算及QBE

—域演算语言QBE的基本形式

➤ QBE操作框架由四个部分构成

- ▣ 关系名区：用于书写欲待查询的关系名
- ▣ 属性名区：用于显示对应关系名区关系的所有属性名
- ▣ 操作命令区：用于书写查询操作的命令
- ▣ 查询条件区：用于书写查询条件



2.5 关系域演算及QBE

--域演算语言QBE的基本形式（续）

➤ QBE操作命令

□Print 或P. --显示输出操作

□Delete或D. --删除操作

□Insert或I. --插入操作

□Update或U. --更新操作

□示例，如下图表示，向Student表中插入两个元组

<1001, 张三, 男, 20, 03, 1101>

<1002, 李四, 女, 21, 03, 1101>

Student	S#	Sname	Ssex	Sage	D#	Sclass
I.	1001	张三	男	20	03	1101
I.	1002	李四	女	21	03	1101

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE操作命令

- ▣ 再示例，如下图表示，将Student表中如下的两个元组删除掉

<1001, 王五, 男, 20, 03, 1101>

<1002, 刘六, 女, 21, 03, 1101>

Student	S#	Sname	Ssex	Sage	D#	Sclass
D.	1001	王五	男	20	03	1101
D.	1002	刘六	女	21	03	1101

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件—简单条件

- ▣ QBE查询条件可以直接在查询条件区中书写，形式为 θ 参量
- ▣ θ 可以是 $<$, $>$, $>=$, $<=$, $=$, $<>$; 如省略 θ ，则默认为 $=$
- ▣ θ 参量中的参量可以是常量，直接书写;
- ▣ 例如：找出年龄小于17岁的所有同学

Student	S#	Sname	Ssex	Sage	D#	Sclass
P.				<17		

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件—不同属性上的‘与’条件

▣ QBE不同属性上的与条件可以写在同一行中

▣ 例如：找出年龄小于17岁的所有男同学

Student	S#	Sname	Ssex	Sage	D#	Sclass
P.			男	<17		

▣ 同一行中各个条件之间是“ \wedge ”关系(与)，如上图，可写为域演算公式：

$\{ \langle a, b, c, d, e, f \rangle \mid \langle a, b, c, d, e, f \rangle \in \text{Student} \wedge c = \text{'男'} \wedge d < 17 \}$

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件—示例元素与投影

- ▣ 条件 θ 参量中的参量也可以是域变量，用任何一个值（不必是结果中的值）带有下列划线表示，被称为示例元素。示例元素下划线上面的值不起作用，被当作域变量名称来对待，只用于占位或是连接条件。不带下划线的则是构成实际条件一部分的值。

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件—示例元素与投影（续）

▣例如：找出年龄小于17岁的所有男同学的姓名

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P. <u>X</u>	男	<17		

▣当不是显示所有内容时，可在条件区对应要显示的列下面书写显示输出命令(即投影运算)，如上例输出姓名P. X，你可以任意写一个名字，如P. 张三都是一样的，他们是示例元素不是条件

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件—用示例元素实现‘与’和‘或’运算

▣当书写 \vee 条件(或运算)时,可以采用在多行书写,然后在打印命令后使用不同的示例元素来表征。

▣如下图,一行写为P.X,一行使用P.Y。

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P. <u>X</u>		<17		
		P. <u>Y</u>		>20		

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件—用示例元素实现‘与’和‘或’运算

- ▣ 如果一批 \wedge 条件分多行书写，则相互存在 \wedge 关系的行要采用相同的示例元素。
- ▣ 如下，则表示找出年龄大于17并且年龄小于等于20的同学姓名。

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P. <u>X</u>		>17		
		P. <u>X</u>		<=20		

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件——相当于括号的条件表示

▣ 也可以将 \wedge 、 \vee 和 \neg 条件写在操作命令区。

▣ 如下所示。

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P. <u>X</u>		<17		
\vee		P. <u>Y</u>		>20		

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件——相当于括号的条件表示（续）

- ▣ 当 \wedge 、 \vee 和 \neg 运算符写在操作区时，是对整行条件而言，相当于将该行条件放在括号中一样

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P. <u>X</u>	男	<17		
\vee		P. <u>Y</u>	女	>20		

如上例，表达的是：（年龄<17 并且是男的）**或者**（年龄>20 并且是女的）

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询条件—用示例元素实现多个表的连接

□当检索涉及多个表时，可利用同一连接条件使用相同的示例元素，来实现多个表的连接。

□例如：李明老师教过的所有学生

Student	S#	Sname	Ssex	Sage	D#	Sclass
P.	X					

SC	S#	C#	Score
	X	Y	

Course	C#	Cname	Chours	Credit	Tname
	Y				李明

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：查询计算机系年龄大于19岁的男同学的姓名

Student	S#	Sname	Ssex	Sage	D#	Sclass

Dept	D#	Dname	Dean

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：查询计算机系年龄大于19岁的男同学的姓名

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P.张三		>19	X	
		P.张三	男		X	

Dept	D#	Dname	Dean
	X	计算机	

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：查询计算机系或者年龄大于19岁或者是男同学的姓名

Student	S#	Sname	Ssex	Sage	D#	Sclass

Dept	D#	Dname	Dean

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：查询计算机系或者年龄大于19岁或者是男同学的姓名

Student	S#	Sname	Ssex	Sage	D#	Sclass
		P.张三		>19	X	
		P.李四	男		Y	

Dept	D#	Dname	Dean
	X	计算机	
	Y	计算机	

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

- ▣例如：查询既选修了001号课程又选修了002号课程的学生
的学号

SC	S#	C#	Score

- ▣再例如：找出比男同学张三年龄大的所有男同学的姓名

Student	S#	Sname	Ssex	Sage	D#	Sclass

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

- ▣ 例如：查询既选修了001号课程又选修了002号课程的学生
的学号

SC	S#	C#	Score
	P.X	001	
	P.X	002	

- ▣ 再例如：找出比男同学张三年龄大的所有男同学的姓名

Student	S#	Sname	Ssex	Sage	D#	Sclass
		张三	男	X		
		P.李四	男	>X		

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：将张三同学的年龄更新为19岁

Student	S#	Sname	Ssex	Sage	D#	Sclass

▣再例如：将每位同学的年龄增加1岁

Student	S#	Sname	Ssex	Sage	D#	Sclass

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：将张三同学的年龄更新为19岁

Student	S#	Sname	Ssex	Sage	D#	Sclass
		张三		U.19		

▣再例如：将每位同学的年龄增加1岁

Student	S#	Sname	Ssex	Sage	D#	Sclass
	<u>0001</u>			<u>19</u>		
U.	<u>0001</u>			<u>19+1</u>		

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：找出成绩不及格同学的姓名及不及格的课程和分数

Student	S#	Sname	Ssex	Sage	D#	Sclass

SC	S#	C#	Score

Course	C#	Cname	Chours	Credit	Tname

2.5 关系域演算及QBE

—域演算语言QBE的基本形式（续）

➤ QBE的查询示例

▣例如：找出成绩不及格同学的姓名及不及格的课程和分数

Student	S#	Sname	Ssex	Sage	D#	Sclass
	<u>X</u>	P. <u>张三</u>				

SC	S#	C#	Score
	<u>X</u>	<u>Y</u>	< 60
	<u>X</u>	<u>Y</u>	P. <u>50</u>

Course	C#	Cname	Chours	Credit	Tname
	<u>Y</u>	P. <u>数据库</u>			

2.5 关系域演算及QBE

—用QBE实现关系代数

➤ 关系代数的并、差、乘积、选择和投影运算可以用QBE来实现

□ $T = R \cup S$

R	A1	A2	...	Ak
	<u>RA1</u>	<u>RA2</u>	...	<u>RAk</u>

S	A1	A2	...	Ak
	<u>SA1</u>	<u>SA2</u>	...	<u>SAk</u>

T	A1	A2	...	Ak
I.	<u>RA1</u>	<u>RA2</u>	...	<u>RAk</u>
I.	<u>SA1</u>	<u>SA2</u>	...	<u>SAk</u>

2.5 关系域演算及QBE

—用QBE实现关系代数

➤ 关系代数的并、差、乘积、选择和投影运算可以用QBE来实现

□ $T = R - S$

R	A1	A2	...	Ak
	<u>RA1</u>	<u>RA2</u>	...	<u>RAk</u>

S	A1	A2	...	Ak
	<u>SA1</u>	<u>SA2</u>	...	<u>SAk</u>

T	A1	A2	...	Ak
I.	<u>RA1</u>	<u>RA2</u>	...	<u>RAk</u>
D.	<u>SA1</u>	<u>SA2</u>	...	<u>SAk</u>

2.5 关系域演算及QBE

—用QBE实现关系代数

➤ 关系代数的并、差、乘积、选择和投影运算可以用QBE来实现

□ $T = R \times S$

R	A1	A2	...	An
	<u>RA1</u>	<u>RA2</u>	...	<u>RAn</u>

S	B1	B2	...	Bm
	<u>SB1</u>	<u>SB2</u>	...	<u>SBm</u>

T	A1	A2	...	An	B1	B2	...	Bm
I.	<u>RA1</u>	<u>RA2</u>	...	<u>RAn</u>	<u>SB1</u>	<u>SB2</u>	...	<u>SBm</u>

2.5 关系域演算及QBE

—用QBE实现关系代数

➤ 关系代数的并、差、乘积、选择和投影运算可以用QBE来实现

□ $T = \pi_{i1,i2,...im}(R)$

□ 其中 RA_{ij} 是某一个 RA_k

R	A1	A2	...	An
	<u>RA1</u>	<u>RA2</u>	...	<u>RA_n</u>

T	T1	T2	...	Tm
I.	<u>RA_{i1}</u>	<u>RA_{i2}</u>	...	<u>RA_{im}</u>

2.5 关系域演算及QBE

—用QBE实现关系代数

➤ 关系代数的并、差、乘积、选择和投影运算可以用QBE来实现

□ $T = \sigma_F(R)$

□ 其中 *Condition Box* 中表示选择运算中的 F 公式(详细转换略)

R	A1	A2	...	An
	<u>RA1</u>	<u>RA2</u>	...	<u>RA_n</u>

T	A1	A2	...	A _m
I.	<u>RA1</u>	<u>RA2</u>	...	<u>RA_m</u>

Condition Box

2.6 小结

——本章我们学习了以下一些概念

➤ 关系

- ❑ 域、笛卡尔积、关系；基数与目(度)数
- ❑ 属性/字段/列/数据项、元组/记录/行、关系模式、关系/Table
- ❑ 候选码/候选键、主码/主键、主属性和非主属性、外码/外键
- ❑ 关系的特性

➤ 关系模型

- ❑ 基本数据结构：关系
- ❑ 基本操作：并、差、广义积、选择、投影
- ❑ 完整性约束：实体完整性, 参照完整性, 用户自定义完整性

2.6 小结

——本章我们学习了以下一些概念

➤ 关系运算

▣ 关系代数

- 并、差、广义积、选择、投影;
- 交、除、 θ -连接、等值连接、自然连接、外连接
- 用关系代数表达日常检索需求是按集合思维方式进行

▣ 关系演算

- 关系元组演算 $\{ t \mid P(t) \}$
 - ✓ 以元组为变量, 由 $\in, \theta, \wedge, \vee, \neg, \exists, \forall$, 括号等算式构成 $P(t)$
- 关系域演算 $\{ xyz \mid P(xyz) \}$
 - ✓ 以域为变量, 由 $\in, \theta, \wedge, \vee, \neg, \exists, \forall$, 括号等算式构成 $P(x)$
 - ✓ QBE: Query By Example
- 用关系演算表达日常检索需求是按逻辑思维方式进行

2.6 小结

——本章我们学习了以下一些概念

- 三种关系运算之间是具有等价性的
- 三种关系运算都可说是非过程性的，但相比之下：域演算的非过程性最好，元组演算次之，关系代数最差
- 三种关系运算虽是抽象的，但他们是衡量数据库语言完备性的基础算
 - ▣ 一个数据库语言能够等价地实现这三种关系运算的操作，则说该语言是完备的

2.6 小结

——本章我们学习了以下一些概念

- 目前多数数据库语言都能够实现这三种运算的操作，在此基础上还增加了许多其他的操作，如赋值操作、聚集操作等。
- QBE是一种方便终端用户使用的域演算语言，是通过表格操作来表达查询需求的一种语言

下一章的学习内容

基于关系代数和关系演算而提出的SQL语言目前已成为数据库语言的标准，我们将在下一章进行介绍。

第3章SQL语言：标准数据库语言的语法
及其交互式应用训练

--SQL的各种操作语句：重点是SQL的查询与统计语句，(子模式)视图的应用语句；

数据库系统

数据库系统

□ 例如：“检索出年龄不是最小的所有同学

$\{ t \mid t \in \text{Student} \wedge \exists (u \in \text{Student}) (t[\text{Sage}] > u[\text{Sage}]) \}$

R(学生表)

	S#	Sname	Ssex	Sage	D#
	1110101	张三	女	21	01
t →	1110102	李四	男	20	02
	1110103	王五	男	19	03
u →	1110104	刘六	女	19	03

[返回](#)

数据库系统

□ 再例如：检索出课程都及格的所有同学

$\{ t \mid t \in \text{Student} \wedge \forall (u \in \text{SC} \wedge t[\text{S\#}] = u[\text{S\#}])(u[\text{Score}] \geq 60) \}$

Student

t →

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

SC

u →

S#	C#	Score
1001	001	80
1001	002	86
1001	003	87
1002	001	79
1002	002	87
1002	003	56
1003	001	77
1003	003	89
1004	002	78

[返回](#)

数据库系统

□ 请注意上式写成下面的公式会表达什么意思？

$\{ t \mid t \in \text{Student} \wedge \forall (u \in \text{SC}) (t[\text{S\#}] = u[\text{S\#}] \wedge u[\text{Score}] \geq 60) \}$

Student

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

t →

SC

S#	C#	Score
1001	001	80
1001	002	86
1001	003	87
1002	001	79
1002	002	87
1002	003	56
1003	001	77
1003	003	89
1004	002	78

u →

[返回](#)

数据库系统

□ 例如：检索计算机系的所有同学

$\{ t \mid t \in \text{Student} \wedge \exists (u \in \text{Dept}) (t[D\#] = u[D\#] \wedge u[Dname] = \text{'计算机'}) \}$

Student

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

t →

Dept

D#	Dname
01	机电
02	能源
03	计算机

← u

返回

数据库系统

□ 例如：检索出比张三年龄小的所有同学

$\{ t \mid t \in \text{Student} \wedge \exists (w \in \text{Student})(w[\text{Sname}] = \text{'张三'} \wedge t[\text{Sage}] < w[\text{Sage}]) \}$

Student

	S#	Sname	Ssex	Sage	D#	Sclass
w →	1001	张三	女	21	01	1101
	1002	李四	男	20	02	1101
t →	1003	王五	男	19	03	1102
	1004	刘六	女	21	03	1103

[返回](#)

数据库系统

□ 例如：检索学过所有课程的同学

$\{ t \mid t \in \text{Student} \wedge \forall (u \in \text{Course})(\exists (s \in \text{SC})(s[S\#]=t[S\#] \wedge u[C\#]=s[C\#])) \}$

Student

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

Course

C#	Cname	Chours	Credit
1001	化学	40	4
1002	数学	40	4
1003	物理	60	6

SC

S#	C#	Score
1001	001	80
1001	002	86
1001	003	87
1002	002	79
1002	003	87
1003	001	77
1003	003	89
1004	002	78

返回

数据库系统

□ 例如：检索所有同学所有课程全都及格的系

$\{t \mid t \in \text{Dept} \wedge \forall (s \in \text{Student} \wedge s[D\#]=t[D\#]) (\forall (u \in \text{SC} \wedge s[S\#]=u[S\#]) (u[\text{Score}] \geq 60))\}$

Student

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

s →

Dept

D#	Dname
1001	化学
1002	数学
1003	物理

t →

SC

S#	C#e	Score
1001	001	80
1001	002	86
1001	003	87
1002	002	79
1002	003	87
1003	001	77
1003	003	89
1004	002	78

u →

[返回](#)

数据库系统

$\{ t \mid t \in R \wedge t[\text{Sage}] \leq 19 \wedge t[\text{Sname}] = \text{'张三'} \}$

R(学生表)

S#	Sname	Ssex	Sage	D#
1110101	张三	女	21	01
1110102	李四	男	20	02
1110103	王五	男	19	03
1110104	刘六	女	21	03

t



[返回](#)

数据库系统

$\{ t \mid t \in \text{Student} \wedge \exists (u \in \text{Student}) (t[\text{Sage}] > u[\text{Sage}]) \}$

“检索出年龄不是最小的所有同学”

R(学生表)

	S#	Sname	Ssex	Sage	D#
	1110101	张三	女	21	01
t →	1110102	李四	男	20	02
	1110103	王五	男	19	03
u →	1110104	刘六	女	21	03

[返回](#)

数据库系统

检索出不是03系的所有学生

$\{ \langle a,b,c,d,e,f \rangle \mid \langle a,b,c,d,e,f \rangle \in \text{Student} \wedge e \neq '03' \}$

Student

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

a **b** **c** **d** **e** **f**

Course

C#	Cname	Chours	Credit
1001	化学	40	4
1002	数学	40	4
1003	物理	60	6

g **h** **i** **j**

SC

S#	C#e	Score
1001	001	80
1001	002	86
1001	003	87
1002	002	79
1002	003	87
1003	001	77
1003	003	89
1004	002	78

a **g** **m**

[返回](#)

数据库系统

检索不是(小于20岁的男同学)的所有同学的姓名

$\{ \langle b \rangle \mid \exists a, c, d, e, f (\langle a, b, c, d, e, f \rangle \in \text{Student} \wedge \neg (d < 20 \wedge c = \text{'男'})) \}$

Student

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

a b c d e f

Course

C#	Cname	Chours	Credit
1001	化学	40	4
1002	数学	40	4
1003	物理	60	6

g h i j

SC

S#	C#e	Score
1001	001	80
1001	002	86
1001	003	87
1002	002	79
1002	003	87
1003	001	77
1003	003	89
1004	002	78

a g m

[返回](#)

数据库系统

检索成绩不及格的同学姓名、课程及其
 $\{ \langle b, h, m \rangle \mid \exists a, c, d, e, f, g, i, j, k (\langle a, b, c, d, e, f \rangle \in \text{Student} \wedge (g, h, i, j, k) \in \text{Course} \wedge (a, g, m) \in \text{SC} \wedge m < 60) \}$

Student

S#	Sname	Ssex	Sage	D#	Sclass
1001	张三	女	21	01	1101
1002	李四	男	20	02	1101
1003	王五	男	19	03	1102
1004	刘六	女	21	03	1103

a

b

c

d

e

f

Course

C#	Cname	Chours	Credit
1001	化学	40	4
1002	数学	40	4
1003	物理	60	6

g

h

i

j

SC

S#	C#e	Score
1001	001	80
1001	002	86
1001	003	87
1002	002	79
1002	003	87
1003	001	77
1003	003	89
1004	002	78

a

g

m

返回