
第3章 关系数据库系统

第3章 关系数据库系统

3.1 关系数据库系统概述

3.2 关系模型数学理论 — 关系代数

3.3 关系数据库语言SQL

3.1 关系数据库系统概述

□ 关系数据库系统的优点

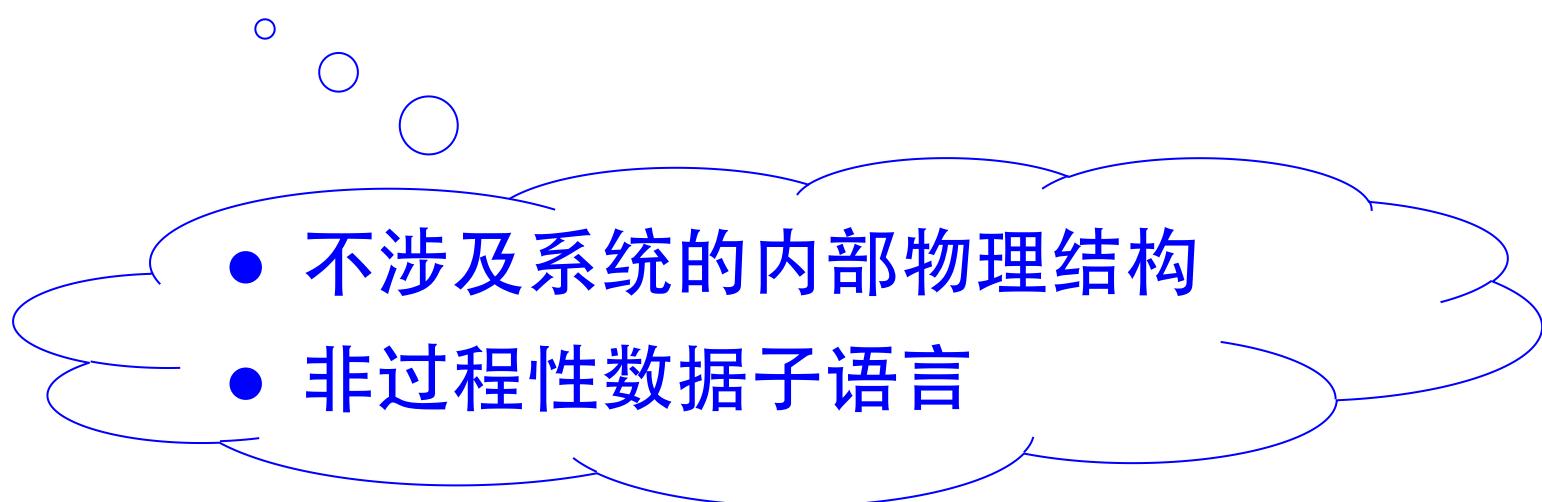
① 数据结构简单



3.1 关系数据库系统概述

□ 关系数据库系统的优点

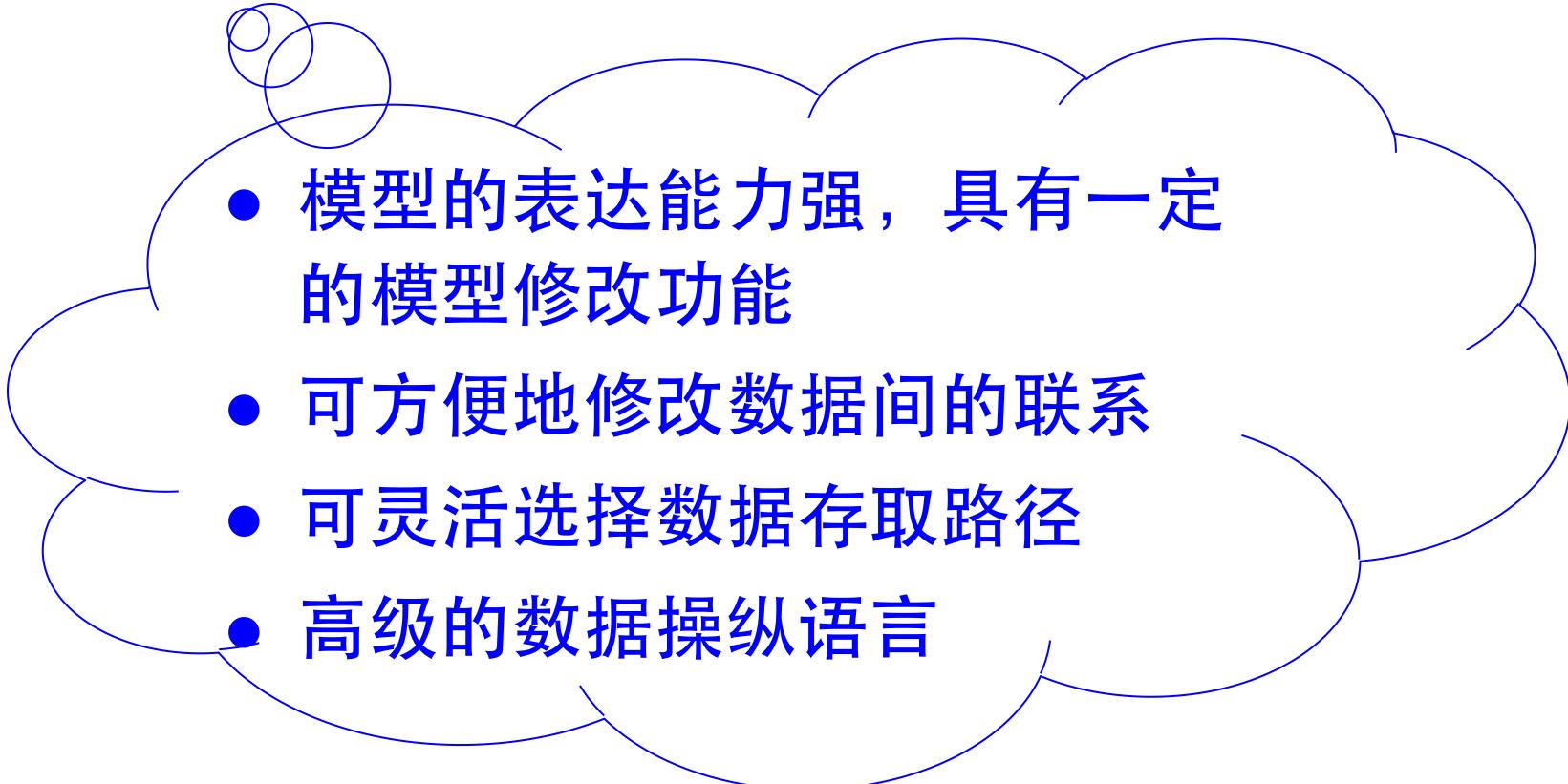
- ① 数据结构简单
- ② 使用方便



3.1 关系数据库系统概述

□ 关系数据库系统的优点

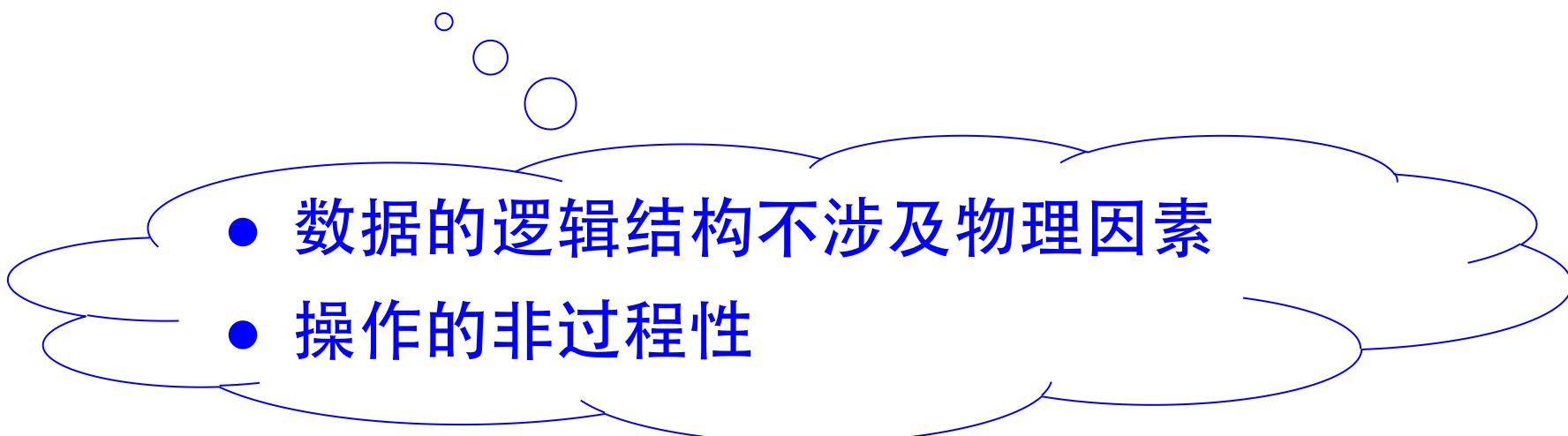
- ① 数据结构简单
- ② 使用方便
- ③ 功能强

- 
- 模型的表达能力强，具有一定
的模型修改功能
 - 可方便地修改数据间的联系
 - 可灵活选择数据存取路径
 - 高级的数据操纵语言

3.1 关系数据库系统概述

□ 关系数据库系统的优点

- ① 数据结构简单
- ② 使用方便
- ③ 功能强
- ④ 数据独立性高



3.1 关系数据库系统概述

□ 关系数据库系统的优点

- ① 数据结构简单
- ② 使用方便
- ③ 功能强
- ④ 数据独立性高
- ⑤ 理论基础深

3.1 关系数据库系统概述

□ 关系数据库系统的优点

- ① 数据结构简单
- ② 使用方便
- ③ 功能强
- ④ 数据独立性高
- ⑤ 理论基础深
- ⑥ 可移植性好

3.1 关系数据库系统概述

□ 关系数据库系统的优点

- ① 数据结构简单
- ② 使用方便
- ③ 功能强
- ④ 数据独立性高
- ⑤ 理论基础深
- ⑥ 可移植性好
- ⑦ 标准化程度高

- 数据子语言SQL的标准化
 - 1985年ANSI公布第一个SQL标准
 - 1989年ISO公布SQL标准：ISO/IEC 9075
 - 1992年ANSI和ISO批准新的SQL标准：SQL2
 - 1993年含有OO特性的SQL建议标准：SQL3
- SQL语言标准的制订是关系数据库系统领域的一次革命

3.1 关系数据库系统概述

□ 关系数据库系统的优点

- ① 数据结构简单
- ② 使用方便
- ③ 功能强
- ④ 数据独立性高
- ⑤ 理论基础深
- ⑥ 可移植性好
- ⑦ 标准化程度高
- ⑧ 分布式功能

- Client/Server (C/S)
- Browser/Server (B/S)
- 分布式数据库
-

3.1 关系数据库系统概述

□ 关系数据库系统的优点

- ① 数据结构简单
- ② 使用方便
- ③ 功能强
- ④ 数据独立性高
- ⑤ 理论基础深
- ⑥ 可移植性好
- ⑦ 标准化程度高
- ⑧ 分布式处理能力强
- ⑨ 开放性

通过提供各种网络环境下的数据访问接口来实现系统的开放性

单机环境	CLI (Call Language Interface)
	E-SQL (Embedded SQL)
网络环境	远程访问
	ODBC / JDBC
	CGI、ASP、JSP等 (Web)
	CORBA

3.1 关系数据库系统概述

□ 关系数据库系统的优点

- ① 数据结构简单
- ② 使用方便
- ③ 功能强
- ④ 数据独立性高
- ⑤ 理论基础深
- ⑥ 可移植性好
- ⑦ 标准化程度高
- ⑧ 分布式功能
- ⑨ 开放性
- ⑩ 其它方面的功能扩展

第3章 关系数据库系统

3.1 关系数据库系统概述

3.2 关系模型数学理论 — 关系代数

3.3 关系数据库语言SQL

3.2 关系模型数学理论 — 关系代数

3.2.0 关系模型

3.2.1 关系的表示

3.2.2 关系操纵的表示

3.2.3 关系模型与关系代数

3.2.4 关系代数中的扩充运算

3.2.5 关系代数实例

3.2.0 关系模型

3.2.0.1 关系数据结构

- 表结构
- 键
- 关系

3.2.0.2 关系操纵

- 查询， 插入， 删除， 修改
- 空值的处理

3.2.0.3 关系中的数据约束

3.2.0 关系模型

名词术语 (Terminology) 之间的对应关系

关系模型	关系数据库管理系统	文件系统
Relation	Table	File of Records
Attribute	Column	Field
Tuple	Row	Record
Schema	Table Heading	Type of Record

Table Heading: a set of named columns about a table

table name

CUSTOMERS

column name

<u>cid</u>	cname	city	discnt
c001	TipTop	Duluth	10.00
c002	Basics	Dallas	2.00
c003	Allied	Dallas	5.00
c004	ACME	Duluth	8.00
c006	ACME	Kyoto	0.00

table

column

table heading

row

3.2.0.1 关系数据结构

1. 表结构

□ 二维表（简称表）的组成

➤ 表框架 (Frame)

- 表框架由n个命名的‘属性’ (Attribute) 组成
 - n 被称为表的元数 (n元表)
 - 每个属性有一个取值范围，即值域 (Domain)

➤ 元组 (Tuple)

- 在表框架中可按行存放数据，其中的每行数据称为‘元组’
 - 一个元组由n个元组分量组成，每个元组分量都对应着表框架中的一个属性
- 一个表框架可存放m个元组，m被称为表的基数 (Cardinality)

3.2.0.1 关系数据结构

➤一个n元表框架及框架内的m个元组构成一个完整的二维表

【表1】一个有关学生 (S) 的二维表

sno	sn	sd	sa
98001	张曼英	CS	18
98002	丁一明	CS	20
98003	王爱国	CS	18
98004	李 强	CS	21

3.2.0.1 关系数据结构

□ 二维表的性质

- 1) 元组个数有限性
- 2) 元组的唯一性
- 3) 元组的次序无关性
- 4) 元组分量的原子性
- 5) 属性名唯一性
- 6) 属性的次序无关性
- 7) 分量值域同一性

sno	sn	sd	sa
98001	张曼英	CS	18
98002	丁一明	CS	20
98003	王爱国	CS	18
98004	李 强	CS	21

- 满足上述7个性质的二维表被称为 ‘关系’
(Relation)
- 以符合上述条件的二维表为基本数据结构所建立的模型称为 ‘关系模型’

3.2.0.1 关系数据结构

□ Relational Rules

- Rule 1. First Normal Form Rule
 - Can't have multi-valued fields
- Rule 2. Access Rows by Content Only Rule
 - No order to the rows
 - No order to the columns
- Rule 3. The Unique Row Rule
 - Two rows can't be same in all attributes at once.
So that a relation is an unordered SET of tuples
 - But many products allow this for efficiency of load

3.2.0.1 关系数据结构

3. 键 (Key)

□ 在二维表中凡能唯一最小标识元组的属性集称为该表的‘键’，或称‘关键字’

➤ 候选键 (Candidate Key)

➤ 主键 (Primary key)

—在一张二维表的所有候选键中，被选中的一个候选键被称为该表的‘主键’（或称‘主关键字’）

□ 每一张二维表都至少存在一个‘键’

Keys & Superkeys

□ Key & Superkey

➤ Superkey

– is a set of columns that has the uniqueness property

➤ key

– is a minimal superkey:

- no subset of columns also has uniqueness property

Keys & Superkeys

□ Def. Table Key

- Given a table T, with $\text{Head}(T)=\{A_1, A_2, \dots, A_n\}$. A key for the table T, is a set of attributes, $K=\{A_{i1}, \dots, A_{ik}\}$, with two properties:
 - 1) If u, v are distinct tuples of T, then by designer intention $u[K] \neq v[K]$; that is ,there will always exist at least one column, A_{im} , in the set of columns K such that $u[A_{im}] \neq v[A_{im}]$
 - 2) No proper subset H of K has property 1

- Superkey: a set of columns that fulfills property 1 but not necessarily property 2

3.2.0.1 关系数据结构

□ 外键 (Foreign Key)

- 如果表A中的属性集F是表B的键，则称该属性集F为表A的‘外键’（或称‘外关键字’）
- 其中：
 - 表A被称为‘引用表’，表B被称为‘被引用表’
 - 表A和表B可以是同一张二维表

3.2.0.1 关系数据结构

【例】请判断下面的‘人事档案’表中有哪些候选键？

‘人事档案’ 表

A	B	C	D	E	F
138	徐英健	女	18	浙江	团员
139	赵文虎	男	23	江苏	党员
140	沈亦奇	男	20	上海	群众
141	王 宾	男	21	江苏	群众
142	李红梅	女	19	安徽	团员
143	王 宾	男	23	上海	党员

- 不能只根据一张表中已有的元组来判断其有哪些‘候选键’，必须根据表中属性的语义含义及其相互联系才能确定表中的‘键’的组成

3.2.0.1 关系数据结构

【例】‘学生’表

学号	姓名	性别	系	专业	入学时间	身份证号
----	----	----	---	----	------	------

两个候选键

【例】‘选课’表

学 号	课程号	成 绩
-----	-----	-----

两个属性联合构成一个候选键 两个外键

3.2.0.1 关系数据结构

学生			
学号	姓名	系别	年龄
S1001	张曼英	计算机	19
S1002	李 红	数学	20
S1003	丁一珉	中文	18
S1004	王爱民	计算机	20

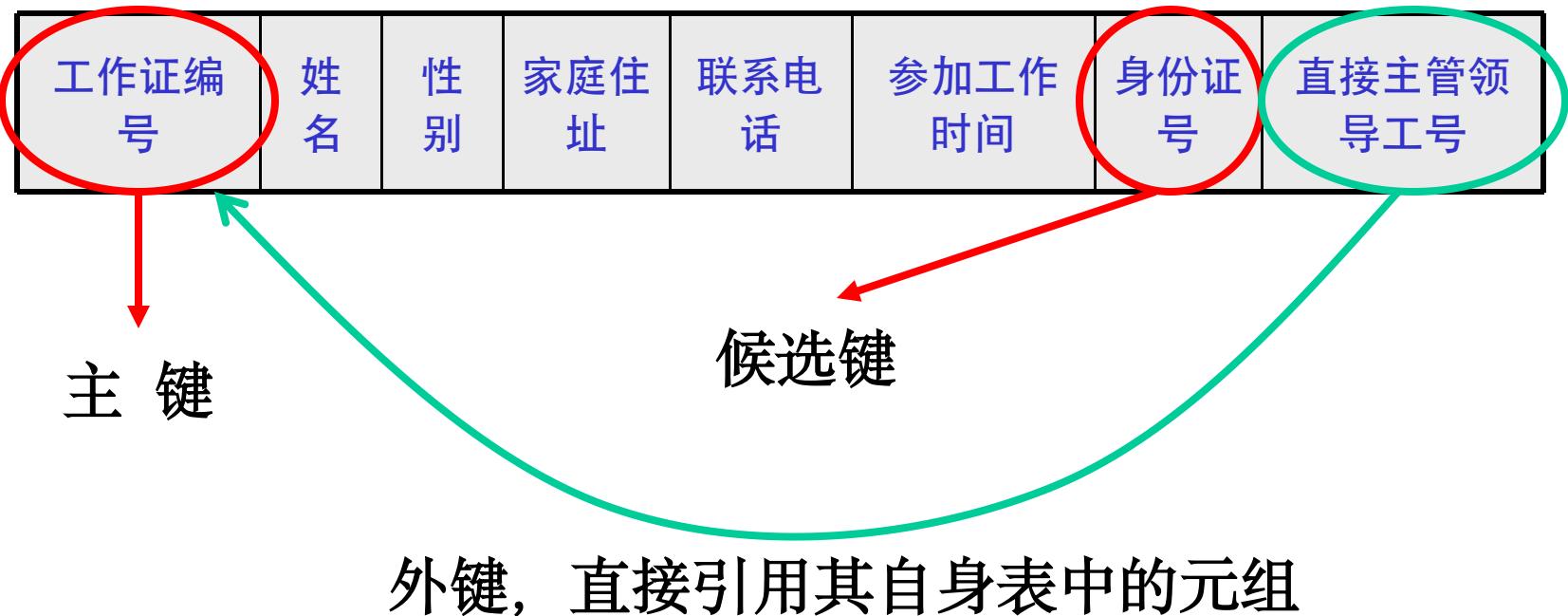
选课		
学号	课程编号	成绩
S1001	C101	85
S1001	C102	91
S1001	C103	95
S1003	C101	90
S1002	C101	76
S1003	C102	87
S1004	C101	88

课程		
课程编号	课程名	主讲教师
C101	C++	T02
C102	OS	T01
C103	DB	T02

外键与主键之间的引用关系

3.2.0.1 关系数据结构

【例】‘职工’表



3.2.0.1 关系数据结构

【思考题】在下列关系表中存在哪些键属性？

职工 (姓名, 工号, 出生日期, 家庭地址, 年薪,
管理员工号, 所在部门编号)

部门 (部门名称, 部门编号, 部门负责人的工号)

项目 (项目名称, 项目编号, 所在城市, 主管部门
编号)

工作 (职工工号, 项目编号, 工作时间)

家属 (职工工号, 家属的姓名, 家属的性别)

3.2.0.1 关系数据结构

□ 二维表的性质

- 1) 元组个数有限性
- 2) 元组的唯一性
- 3) 元组的次序无关性
- 4) 元组分量的原子性
- 5) 属性名唯一性
- 6) 属性的次序无关性
- 7) 分量值域同一性

sno	sn	sd	sa
98001	张曼英	CS	18
98002	丁一明	CS	20
98003	王爱国	CS	18
98004	李 强	CS	21

- 满足上述7个性质的二维表被称为 ‘关系’
(Relation)
- 以符合上述条件的二维表为基本数据结构所建立的模型称为 ‘关系模型’

3.2.0.2 关系操纵

□关系模型的数据操纵功能就是建立在关系上的数据操纵功能，包括：

- 查询 {
 - 单张表内的数据查询
 - 两张表之间的数据查询
 - 多张表上的数据查询
 - 增加
 - 删除
 - 修改
- { 纵向定位
横向定位

3.2.0.2 关系操纵

□ 数据查询

➤ 单张表内的数据查询操作

— 目标

- 在一张表内查询获得在指定行（元组）的指定列（属性）上的元组分量（属性值）

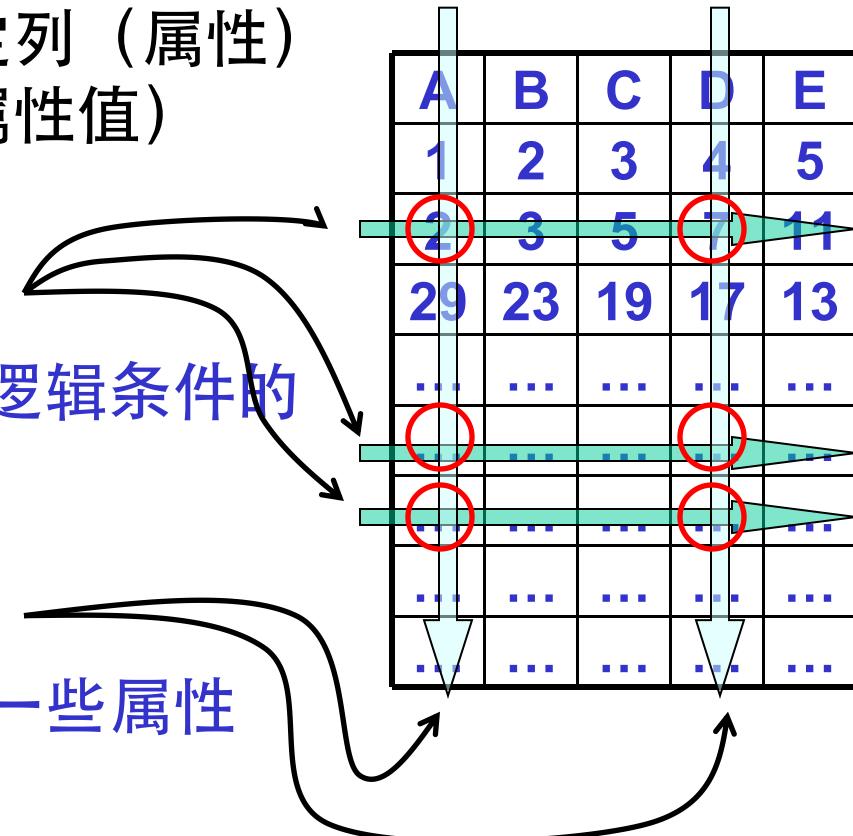
— 操作过程

- 行选择

» 选择满足某些逻辑条件的元组

- 列指定

» 指定关系中的一些属性

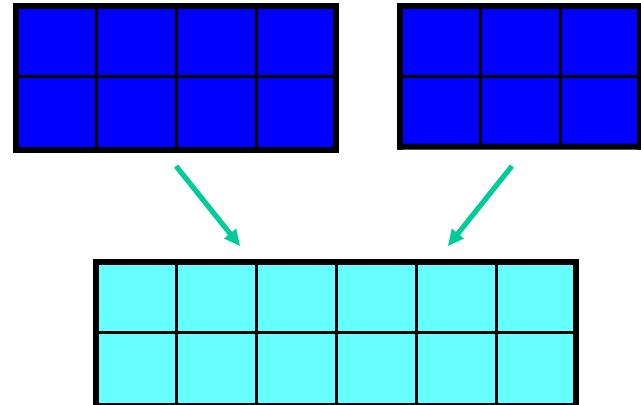


3.2.0.2 关系操纵

□ 数据查询 (cont.)

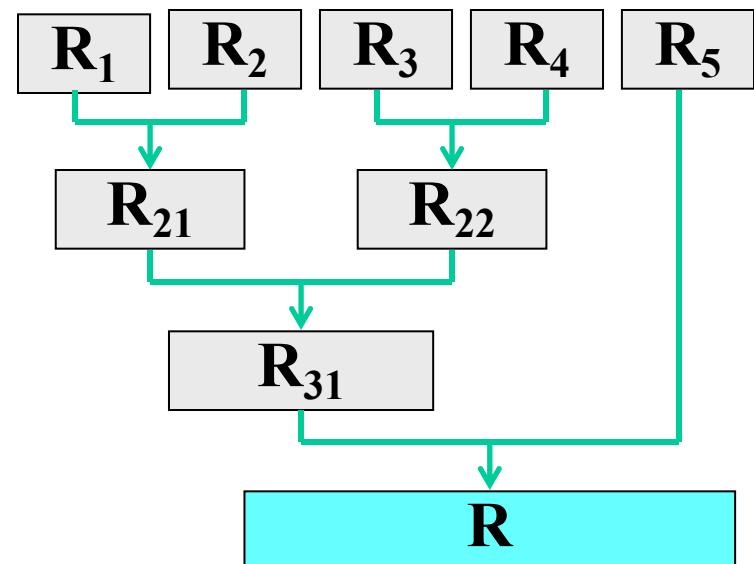
➤ 两张表之间的数据查询

- 先将两张表合并为一张表
- 再利用单张表内的数据查询操作进行查询



➤ 多张表上的数据查询

- 先将多张表合并为一张表
- 然后再利用单张表内的数据查询操作进行查询



3.2.0.2 关系操纵

□ 关系模型上的数据查询操作可以被分解为下面的三种基本操作方式：

1) 两个关系的合并

- 通过多个关系之间的两两合并，可以将其最终合并为一个关系

2) 单个关系内的元组选择

- 选择满足指定条件的元组

3) 单个关系内的属性指定

- 选择结果所需要的属性（值）

3.2.0.2 关系操纵

□ 数据删除

- 数据删除的基本单位是元组
- 其功能是将指定关系内满足给定逻辑条件的元组删除。其操作过程是：

1) 确定被删除的元组

- 一次删除操作只能删除一个关系内的元组
- 可采用单个关系内的元组选择操作来确定需要被删除的元组

2) 执行删除操作

- 删除选中的元组

3.2.0.2 关系操纵

□ 数据插入

- 在指定关系中插入一个或多个新的元组
- 一条数据插入操作只能向一个关系中增加新的元组

3.2.0.2 关系操纵

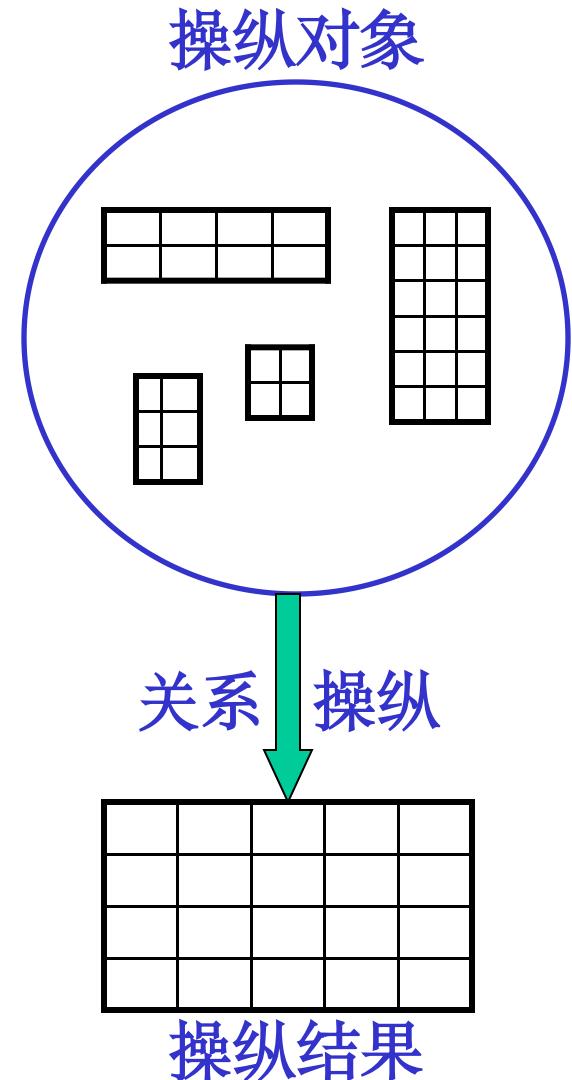
□ 数据修改

- 在一个关系内修改指定元组的某些列上的值（即修改某些元组分量的值）
- 数据修改操作不是一个基本操作，其功能可以由其它的数据操纵方式来实现
 - 1) 先删除需要修改的元组
 - 2) 然后插入修改后的新元组

3.2.0.2 关系操纵

□关系操纵小结

- 操纵对象是关系
- 操纵结果也构成一个关系
- 关系模型上的五种基本操纵功能
 - 元组选择
 - 属性指定
 - 两个关系的合并
 - 元组插入
 - 元组删除



3.2.0.2 关系操纵

□ 空值处理

- 与空值有关的数据完整性约束
 - 关系的主键中不允许出现空值
- 需要定义有关空值的运算
 - ① 在算术表达式中如出现空值，则其运算结果也为空值；
 - ② 在逻辑运算表达式中如出现空值，则其运算结果为逻辑假；
 - ③ 在统计计算中，对于‘空集’和集合中的‘空值’的处理方法

3.2.0.2 关系操纵

- 在作SUM, AVG, MAX, MIN或COUNT统计操作时，集合中的‘空值’元素是不统计在内的
- 在统计计算中，对‘空集’的处理方法如下：
 - ① 对空集作SUM、AVG、MAX或MIN统计操作时，其统计结果均为空值
 - ② 对空集作COUNT统计操作时，其统计结果为0
 - 注意 `count(<表达式>)` 和 `count(*)` 的区别

3.2.0.3 关系中的数据约束

□ 三类数据完整性约束

- 实体完整性约束
 - 主键中的属性不能有空值
- 参照完整性约束
 - 外键要么取空值，要么是被引用表中当前存在的某元组上的主键值
- 用户定义的完整性
 - 用户自己定义的属性取值约束

□关系代数(Relational algebra)

➤用代数的方法表示关系模型

- 关系的表示
- 关系操纵的表示
- 关系代数与关系模型
- 关系代数的扩充运算

3.2 关系模型数学理论 — 关系代数

3.2.0 关系模型

3.2.1 关系的表示

3.2.2 关系操纵的表示

3.2.3 关系模型与关系代数

3.2.4 关系代数中的扩充运算

3.2.5 关系代数实例

3.2.1 关系的表示

□关系是元组的集合

➤元组是元组分量的集合

- 一个n元关系中的元组是一个由n个元组分量按照属性的排列次序组织起来的n元有序组
- 因此，一个n元关系就是一个n元有序组的集合

3.2.1 关系的表示

□ 笛卡儿(乘)积

➤ 设存在 n 个集合 D_1, D_2, \dots, D_n , 它们的笛卡儿乘积是: $D_1 \times D_2 \times \dots \times D_n$

— 该笛卡儿乘积的结果是一个集合, 其中的每个元素都是一个具有如下形式的 n 元有序组:

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

➤ 设集合 D_i 的元素个数为 r_i ($i=1, 2, \dots, n$), 则它们的笛卡儿乘积的结果元素个数为:

$$r_1 \times r_2 \times \dots \times r_n$$

3.2.1 关系的表示

□关系R

- n元关系R是一个n元有序组的集合
- 设n元关系R的属性域分别是 D_1, D_2, \dots, D_n ，那么这n个域的笛卡儿乘积也是一个n元有序组的集合，并且与n元关系R存在如下联系：

$$R \subseteq D_1 \times D_2 \times \dots \times D_n$$

Example of Relation

- 设有一个由学号sno、姓名name和系别dept等三个属性所构成的学生关系S:

S	sno	name	dept
	1	张山	数学
	2	李英	数学
	3	王华	中文

- 在该关系中加入上述的三个学生元组，在关系代数中，上述的学生关系可以被表示为下面的集合： $S = \{(1, 张山, 数学), (2, 李英, 数学), (3, 王华, 中文)\}$

Example of Relation

S	sno	name	dept
	1	张山	数学
	2	李英	数学
	3	王华	中文

□ 假设只考虑上述的三个学生元组，那么：

$$\text{Domain}(sno) = \{1, 2, 3\}$$

$$\text{Domain}(name) = \{\text{张山}, \text{李英}, \text{王华}\}$$

$$\text{Domain}(dept) = \{\text{数学}, \text{中文}\}$$

□ 将上述的三个值域进行笛卡尔乘积运算可得到如下的关系表W：

W

1	张山	数学
1	李英	数学
1	王华	数学
2	张山	数学
2	李英	数学
2	王华	数学
3	张山	数学
3	李英	数学
3	王华	数学

1	张山	中文
1	李英	中文
1	王华	中文
2	张山	中文
2	李英	中文
2	王华	中文
3	张山	中文
3	李英	中文
3	王华	中文

□ 原关系S（绿色背景）显然只是笛卡尔乘积结果关系W的一个真子集

3.2 关系模型数学理论 — 关系代数

3.2.0 关系模型

3.2.1 关系的表示

3.2.2 关系操纵的表示

3.2.3 关系模型与关系代数

3.2.4 关系代数中的扩充运算

3.2.5 关系代数实例

3.2.2 关系操纵的表示

□ 在关系模型中有四种类型的数据操纵功能。这四种类型的数据操纵功能又可以被分解为关系模型上的五种基本操作

- 关系是元组的集合，在关系模型上进行数据操纵所获得的结果仍然构成一个关系。因此，我们可以将关系模型上的五种基本操作看成是关系上的五种基本运算
- 以关系做运算对象，利用关系上的五种基本运算可以构造出一个关系运算表达式。我们可以用这样的关系运算表达式来描述用户的数据操纵要求

3.2.2 关系操纵的表示

- 对于关系代数中的每一种运算符，我们都需要从以下两个方面去掌握：
 - ① 运算的执行条件
 - ② 运算的执行结果
 - 结果关系的关系模式
 - 结果关系中的元组

3.2.2 关系操纵的表示

□ Def. 2.6.1 Compatible Tables (相容表)

- Tables R and S are *compatible* if they have the same headings

- that is , if $\text{Head}(R)=\text{Head}(S)$, with attributes chosen from the same domains and with the same meanings

Compatible Table (example 1)

R1

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2

S1

A	B	C
a1	b1	c1
a1	b1	c2
a1	b2	c3
a3	b2	c3

□ Tables R1 and S1 are *compatible* because:

- with the same number of columns
- each pair of columns (one in table R1, and another in table S1) have:
 - the same domains (character), and
 - the same meanings, such as the same name of columns

Compatible Table (example 2)

R2

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2

S2

A	B	C
1	b1	c1
1	b1	c2
1	b2	c3
3	b2	c3

- Tables R2 isn't **compatible** with table S2 because:
 - The domain of column A in table S2 is integer, and the domain of column A in table R2 is character, and that
 - We can't find a pair of columns (X, A) (X is a column in table R2, and A is a column in table S2) with the same domain (integer)

Compatible Table (example 3)

R3

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2

S3

A	B	D
a1	b1	c1
a1	b1	c2
a1	b2	c3
a3	b2	c3

- Tables R3 isn't *compatible* with table S3 because:
 - We can't find a pair of columns (?, D) (? is a column in table R3, and D is a column in table S3) with the same meanings
 - In relational algebra, two columns are said to have the same meanings if they have the same name of column

Compatible Table (example 4)

R4

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2

S4

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d1
a1	b2	c3	d1
a3	b2	c3	d1

- Tables R4 isn't *compatible* with table S4 because:
 - The number of columns in table R4 isn't equal to the number of columns in table S4

Compatible Table (example 5)

R5

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2

S5

A	C	B
a1	c1	b1
a1	c2	b1
a1	c3	b2
a3	c3	b2

□ Tables R5 and S5 are *compatible* because:

- with the same headings of table
 - We can move the column B to the left of the column C in table S5 because No order to the columns

Compatible Table (example 5)

S5

A	C	B
a1	c1	b1
a1	c2	b1
a1	c3	b2
a3	c3	b2

S5

A	B	C
a1	b1	c1
a1	b1	c2
a1	b2	c3
a3	b2	c3

The diagram illustrates the transformation of table S5 into table R5. A grey curved arrow at the top of the S5 table indicates a row swap between the first and second rows. A large green arrow points from the transformed S5 table to the R5 table, indicating the compatibility of the two structures.

R5

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2

3.2.2 关系操纵的表示

□ 传统的集合运算

NAME	SYMBOL	FORM	EXAMPLE
UNION (并)	U	UNION	R U S
INTERSECTION (交)	∩	INTERSECT	R ∩ S
DIFFERENCE (差)	-	MINUS	R - S

□ Def Union, Intersection, Difference

➤ Let R and S be two compatible tables, where
 $\text{Head}(R) = \text{Head}(S)$

$-R \cup S$

- is a table with the same heading as R (or S)
- for each row t in R or in S, t in $R \cup S$

$-R \cap S$

- is a table with the same heading as R (or S)
- for each row t in R, if t appear in S, then t in $R \cap S$

$-R - S$

- is a table with the same heading as R (or S)
- for each row t in R, if t don't appear in S, then t in $R - S$

3.2.2 关系操纵的表示

	交换律	结合律
并运算	√	√
交运算	√	√
差运算	×	×

- ‘交’ 运算并不是一个基本运算，其功能可以由‘差’运算来实现。

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

- 有关‘交’运算和‘差’运算的一组等价变换公式

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

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

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

3.2.2 关系操纵的表示

□ 并运算：R ∪ S

➤ 条件

- 参与运算的两个关系必须是同类关系
- 同类关系：具有相同的属性个数，且对应列所表示的属性应具有相同的值域

➤ 结果

- 关系模式不变，由所有属于关系R或属于关系S的元组所组成的集合

R	A	B	C
a	b	c	
d	a	f	
c	b	d	

S	A	B	C
b	g	a	
d	a	f	

R ∪ S	A	B	C
a	b	c	
d	a	f	
c	b		
b	g	d	

3.2.2 关系操纵的表示

□ 差运算： $R - S$

➤ 条件

– 参与运算的两个关系必须是同类关系

➤ 结果

– 关系模式不变，由所有属于关系R但不属于关系S的元组所组成的集合

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

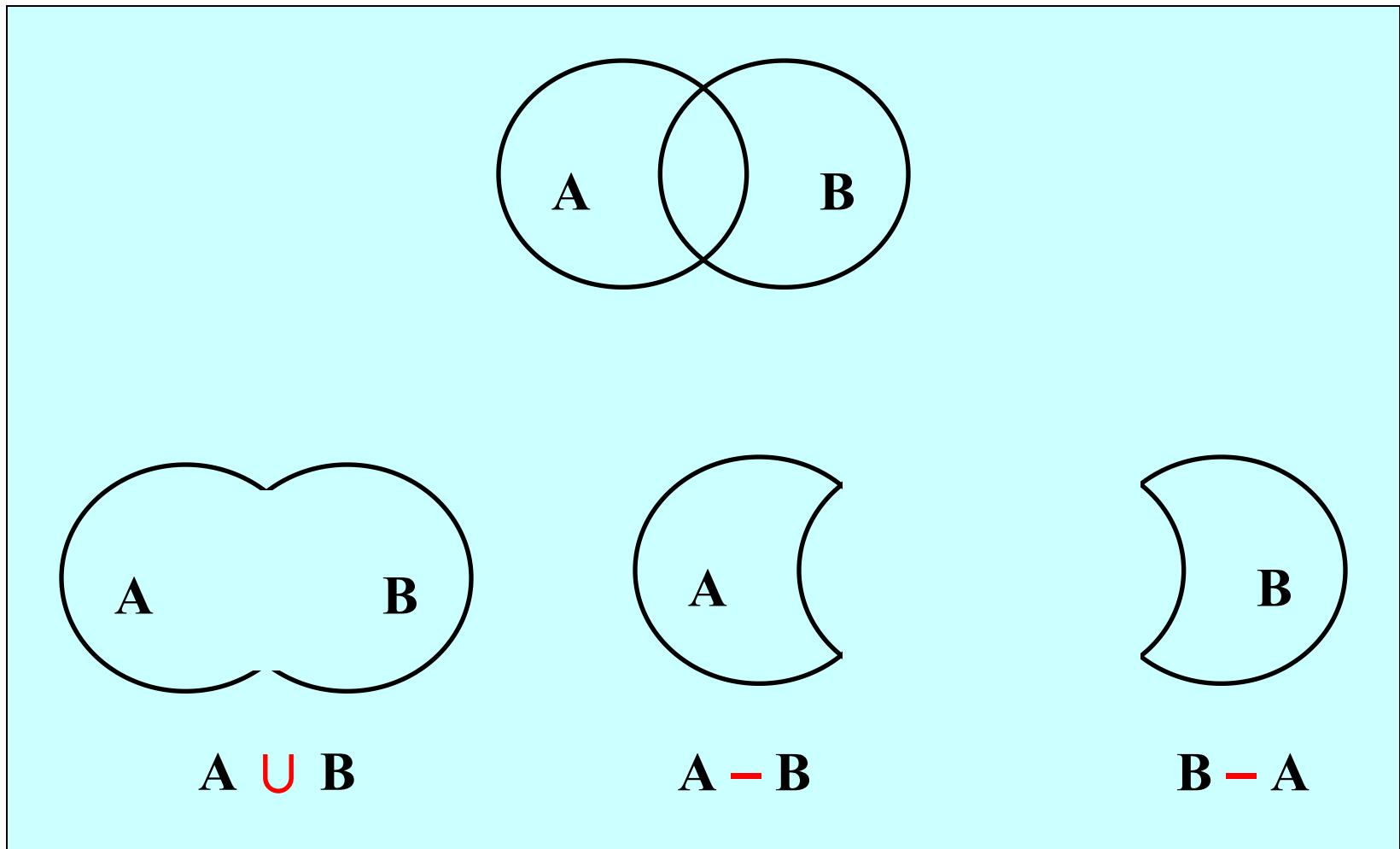
$R - S$	A	B	C
	a	b	c
	c	b	d

S	A	B	C
	b	g	a
	d	a	f

$S - R$	A	B	C
	b	g	a

3.2.2 关系操纵的表示

□ ‘并’ 运算与 ‘差’ 运算的示意图



3.2.2 关系操纵的表示

□‘并’ 运算满足交换律与结合律

$$R \cup S = S \cup R$$

$$(R \cup S) \cup T = R \cup (S \cup T)$$

□‘差’ 运算不满足交换律与结合律

$$R - S \neq S - R$$

$$(R - S) - T \neq R - (S - T)$$

3.2.2 关系操纵的表示

□ 投影 (Projection) 运算

- 略去关系中的某些列并重新安排剩余列的排列次序的运算
- 设关系R有n个属性 A_1, A_2, \dots, A_n , 在其中m个属性 B_1, B_2, \dots, B_m 上的投影运算可以表示为:

$\Pi_{B_1, B_2, \dots, B_m} (R)$

- 其中: $B_i \in \{ A_1, A_2, \dots, A_n \}$ ($i=1, 2, \dots, m$)

- 运算结果
 - 是一个由 B_1, B_2, \dots, B_m 所组成的m元关系
 - 关系R中的每个元组t在 B_1, B_2, \dots, B_m 这m个属性上的取值 t_1, t_2, \dots, t_m 构成结果关系中的一个元组

3.2.2 关系操纵的表示

【例】

R

A	B	C
a	b	c
d	a	f
c	b	d

$\Pi_{C,A}(R)$

C	A
c	a
f	d
d	c

$\Pi_B(R)$

B
b
a

□ 必须注意消除结果关系中可能出现的重复元组

3.2.2 关系操纵的表示

□选择运算: $\sigma_F(R)$

- 根据给定的条件F从关系R中选出符合条件的元组
- 结果
 - 结果关系的关系模式不变，由属于关系R且满足条件F的元组所组成
- 条件F的构造方式
 - 基本逻辑条件: $\alpha \theta \beta$
 - 复合逻辑条件: 由若干个基本逻辑条件经逻辑运算符组合而成
 - » 逻辑与(\wedge)，逻辑或(\vee)

3.2.2 关系操纵的表示

【例】

R		
A	B	C
a	b	c
d	a	f
c	b	d

$\sigma_{B='b'}(R)$

A	B	C
a	b	c
c	b	d

$\sigma_{A>'c' \vee C<'d'}(R)$

A	B	C
a	b	c
d	a	f

$\sigma_{B='b' \wedge C='c'}(R)$

A	B	C
a	b	c

3.2.2 关系操纵的表示

【例】查询‘人事档案’表E中‘年龄’小于或等于20岁的女职工

$\sigma_{\text{年龄} \leq 20 \wedge \text{性别} = \text{'女'}} (E)$

‘人事档案’ 表E

编号	姓名	性别	年龄	籍贯	政治面貌
138	徐英健	女	18	浙江	团员
139	赵文虎	男	23	江苏	党员
140	沈亦奇	男	20	上海	群众
141	王 宾	男	21	江苏	群众
142	李红梅	女	19	安徽	团员

3.2.2 关系操纵的表示

- 可以结合使用投影和选择运算来实现单张表中的数据查询操作: $\Pi_A (\sigma_F (R))$
 - 1) 先根据条件F对关系R进行选择运算
 - 选出符合条件的元组
 - 2) 再对选择运算的结果关系进行投影运算
 - 确定最终结果关系的关系模式
 - 根据选择运算的结果关系中的元组构造出最终结果关系中的元组
- 可以简写为: $\Pi_A \sigma_F (R)$
 - 在没有括号的情况下, 其运算顺序为: 从右向左

3.2.2 关系操纵的表示

【例】

R

A	B	C
a	b	c
d	a	f
c	b	d

$\sigma_{B='b'}(R)$

A	B	C
a	b	c
c	b	d

$\Pi_{A,C}(\sigma_{B='b'}(R))$

A	C
a	c
c	d

【注】两个运算的书写次序不能颠倒

3.2.2 关系操纵的表示

【例】查询‘人事档案’表E中‘年龄’小于或等于20岁的女职工的姓名和籍贯

$\Pi_{\text{姓名}, \text{籍贯}} (\sigma_{\text{年龄} \leq 20 \wedge \text{性别} = \text{'女'}} (\text{E}))$

‘人事档案’ 表E

编号	姓名	性别	年龄	籍贯	政治面貌
138	徐英健	女	18	浙江	团员
139	赵文虎	男	23	江苏	党员
140	沈亦奇	男	20	上海	群众
141	王 宾	男	21	江苏	群众
142	李红梅	女	19	安徽	团员

结果关系 T

姓名	籍贯
徐英健	浙江
李红梅	安徽

3.2.2 关系操纵的表示

□ 投影运算不满足交换律

$$\Pi_{A_set}(\Pi_{B_set}(R)) \neq \Pi_{B_set}(\Pi_{A_set}(R))$$

□ 选择运算满足交换律

$$\sigma_{F1}(\sigma_{F2}(R)) = \sigma_{F2}(\sigma_{F1}(R)) = \sigma_{F1 \wedge F2}(R)$$

□ 投影运算和选择运算不能相互交换

$$\Pi_A(\sigma_F(R)) \neq \sigma_F(\Pi_A(R))$$

- 如果右边的表达式是一个合法（即能够执行）的表达式，那么可以从右向左进行变换（即用左边的表达式去代替右边的表达式）
- 一般情况下，无法从左向右进行变换

3.2 关系模型数学理论 — 关系代数

3.2.0 关系模型

3.2.1 关系的表示

3.2.2 关系操纵的表示

3.2.3 关系模型与关系代数

3.2.4 关系代数中的扩充运算

3.2.5 关系代数实例

3.2.2 关系操纵的表示

□关系的笛卡儿乘积： $R \times S$

➤是两个关系的合并运算

➤设关系R和S分别有n和m个属性，即：

$R(A_1, A_2, \dots, A_n)$ 和 $S(B_1, B_2, \dots, B_m)$

—则它们的笛卡儿乘积 $T = R \times S$ 有(n+m)个属性，
即： $T(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$

➤若元组 $(a_1, a_2, \dots, a_n) \in R$, 元组 $(b_1, b_2, \dots, b_m) \in S$,
则将其合并所得到的新元组必属于这两个关系的笛
卡儿乘积，即：

$(a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m) \in T$

➤若关系R和S分别有p和q个元组，那么它们的笛卡
儿乘积中就含有 $(p \times q)$ 个元组

3.2.2 关系操纵的表示

【例】

R

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2

S

B	C	D
b1	c1	d1
b1	c1	d3
b2	c2	d2
b1	c2	d4

R × S

R.A	R.B	R.C	S.B	S.C	S.D
a1	b1	c1	b1	c1	d1
a1	b1	c1	b1	c1	d3
a1	b1	c1	b2	c2	d2
a1	b1	c1	b1	c2	d4
a1	b2	c3	b1	c1	d1
a1	b2	c3	b1	c1	d3
a1	b2	c3	b2	c2	d2
a1	b2	c3	b1	c2	d4
a2	b1	c2	b1	c1	d1
a2	b1	c2	b1	c1	d3
a2	b1	c2	b2	c2	d2
a2	b1	c2	b1	c2	d4

3.2.2 关系操纵的表示

□ 笛卡儿乘积满足交换律和结合律

$$\triangleright R \times S = S \times R$$

$$\triangleright (R \times S) \times T = R \times (S \times T)$$

□ 如果关系R和S中存在相同的属性名，则必须在结果关系中对其中的一个进行换名

$$\triangleright \text{换名: } \rho_{x(A_1, A_2, \dots, A_n)}(R)$$

3.2.2 关系操纵的表示

□ 关系数据库中的四种类型的操作

- 均可以用关系代数表达式来表示

1) 元组插入

- 设新增加的元组构成关系 R' ，则插入操作可表示为： $R \cup R'$

2) 元组删除

- 设欲删除的元组构成关系 R' （可以经查询操作获得），则删除操作可表示为： $R - R'$

3) 修改操作

- 设欲修改的元组构成关系 R' ，经修改后所生成的新元组构成关系 R'' ，则数据库的修改操作可表示为： $(R - R') \cup R''$

3.2.2 关系操纵的表示

4) 查询操作

- 单个关系：选择 + 投影
 - 例3-2 (属性指定) 查询所有学生的姓名与年龄
 - 例3-3 (简单条件的元组选择) 找出所有年龄大于20岁的学生元组
 - 例3-4 (复杂条件的元组选择) 找出年龄大于20岁且在数学系 (MA) 学习的学生元组
 - 例3-5 (元组选择+属性指定) 查出所有年龄大于20岁的学生的姓名
- 多个关系
 - 先用笛卡儿乘积将多个关系合并为一个关系，然后再执行单个关系上的数据查询操作

Example: The CAP Database

CUSTOMERS

<u>cid</u>	cname	city	disent
c001	TipTop	Duluth	10.00
c002	Basics	Dallas	12.00
c003	Allied	Dallas	8.00
c004	ACME	Duluth	8.00
c006	ACME	Kyoto	0.00

Example: The CAP Database

AGENTS

<u>aid</u>	aname	city	percent
a01	Smith	New York	6
a02	Jones	Newark	6
a03	Brown	Tokyo	7
a04	Gray	New York	6
a05	Otasi	Duluth	5
a06	Smith	Dallas	5

Example: The CAP Database

PRODUCTS

<u>pid</u>	<u>pname</u>	<u>city</u>	<u>quantity</u>	<u>price</u>
P01	comb	Dallas	111400	0.50
p02	brush	Newark	203000	0.50
p03	razor	Duluth	150600	1.00
p04	pen	Duluth	125300	1.00
p05	pencil	Dallas	221400	1.00
p06	folder	Dallas	123100	2.00
p07	case	Newark	100500	1.00

Example: The CAP Database

ORDERS

<u>ordno</u>	month	cid	aid	pid	qty	dollars
1011	jan	c001	a01	p01	1000	450.00
1012	jan	c001	a01	p01	1000	450.00
1019	feb	c001	a02	p02	400	180.00
1017	feb	c001	a06	p03	600	540.00
1018	feb	c001	a03	p04	600	540.00
1023	mar	c001	a04	p05	500	450.00
....

Example 1

□ 查询所有顾客(**CUSTOMERS**, 简写为**C**)的姓名

-令结果关系为**CN**, 则:

$$\mathbf{CN} := \prod_{\mathbf{cname}}(\mathbf{C})$$

C			
<u>cid</u>	cname	city	discnt
c001	TipTop	Duluth	10.00
c002	Basics	Dallas	12.00
c003	Allied	Dallas	8.00
c004	ACME	Duluth	8.00
c006	ACME	Kyoto	0.00

CN
cname
TipTop
Basics
Allied
ACME

Example 1 (cont.)

□ 查询所有顾客(**CUSTOMERS**, 简写为**C**)所居住的城市名称

C				$\Pi_{\text{city}(C)}$
<u>cid</u>	cname	city	disent	city
c001	TipTop	Duluth	10.00	Duluth
c002	Basics	Dallas	12.00	Dallas
c003	Allied	Dallas	8.00	Kyoto
c004	ACME	Duluth	8.00	
c006	ACME	Kyoto	0.00	

□ 需要在结果关系中剔除重复出现的 ‘Duluth’ 及 ‘Dallas’元组

Example 1 (cont.)

C	$\Pi_A(C)$	$\Pi_{B,A}(C)$	$\Pi_{B,C}(C)$																																				
<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>a1</td><td>b1</td><td>c1</td></tr><tr><td>a1</td><td>b1</td><td>c3</td></tr><tr><td>a2</td><td>b2</td><td>c2</td></tr><tr><td>a1</td><td>b2</td><td>c4</td></tr></tbody></table>	A	B	C	a1	b1	c1	a1	b1	c3	a2	b2	c2	a1	b2	c4	<table border="1"><thead><tr><th>A</th></tr></thead><tbody><tr><td>a1</td></tr><tr><td>a2</td></tr></tbody></table>	A	a1	a2	<table border="1"><thead><tr><th>B</th><th>A</th></tr></thead><tbody><tr><td>b1</td><td>a1</td></tr><tr><td>b2</td><td>a2</td></tr><tr><td>b2</td><td>a1</td></tr></tbody></table>	B	A	b1	a1	b2	a2	b2	a1	<table border="1"><thead><tr><th>B</th><th>C</th></tr></thead><tbody><tr><td>b1</td><td>c1</td></tr><tr><td>b1</td><td>c3</td></tr><tr><td>b2</td><td>c2</td></tr><tr><td>b2</td><td>c4</td></tr></tbody></table>	B	C	b1	c1	b1	c3	b2	c2	b2	c4
A	B	C																																					
a1	b1	c1																																					
a1	b1	c3																																					
a2	b2	c2																																					
a1	b2	c4																																					
A																																							
a1																																							
a2																																							
B	A																																						
b1	a1																																						
b2	a2																																						
b2	a1																																						
B	C																																						
b1	c1																																						
b1	c3																																						
b2	c2																																						
b2	c4																																						

剔除结果集中
重复出现的元
组 ‘a1’

剔除结果集中重
复出现的元组
(b1, a1)

Example 2

□ 查询所有居住在京都(Kyoto)的顾客

CUSTOMERS

<u>cid</u>	cname	city	discnt
c001	TipTop	Duluth	10.00
c002	Basics	Dallas	12.00
c003	Allied	Dallas	8.00
c004	ACME	Duluth	8.00
c006	ACME	Kyoto	0.00

T := $\sigma_{\text{city} = \text{'Kyoto'}}$ (CUSTOMERS)

<u>cid</u>	cname	city	discnt
c006	ACME	Kyoto	0.00

Example 2 (cont.)

□ 查询存放于达拉斯(Dallas)且单价(price)超过\$0.50的商品

<u>pid</u>	<u>pname</u>	<u>city</u>	<u>quantity</u>	<u>price</u>
P01	comb	Dallas	111400	0.50
p02	brush	Newark	203000	0.50
p03	razor	Duluth	150600	1.00
p04	pen	Duluth	125300	1.00
p05	pencil	Dallas	221400	1.00
p06	folder	Dallas	123100	2.00
p07	case	Newark	100500	1.00

T := $\sigma_{\text{city} = \text{'Dallas'}} \wedge \text{price} > 0.50$ (Products)

Example 3 复杂查询

□各个关系的关系名可简写如下：

CUSTOMERS → C

AGENTS → A

PRODUCTS → P

ORDERS → O

Example 3 (cont.)

- 查询所有折扣(discnt)低于10%的顾客所在城市以及佣金(percent)低于6%的代理商所在城市的城市名称

$$\Pi_{\text{city}} (\sigma_{\text{discnt} < 10} (\text{C})) \cup \Pi_{\text{city}} (\sigma_{\text{percent} < 6} (\text{A}))$$

- 查询位于同一个城市的顾客(customers)和代理商(agents)，给出顾客的姓名(cname)、代理商的名称(aname)以及他们所在城市的名称(city)

$$\Pi_{\text{C.cname}, \text{C.city}, \text{A.aname}} (\sigma_{\text{C.city} = \text{A.city}} (\text{C} \times \text{A}))$$

Example 3 (cont.)

□ 查询姓名为‘Allied’的顾客的订购信息(orders)，
列出其每一条订单记录所购买商品的商品编号(pid)、
订购月份(month)以及订购数量(qty)

$$\Pi_{O.pid, O.month, O.qty} (\sigma_{C.cid=O.cid \wedge C cname='Allied'} (C \times O))$$

或

$$\Pi_{O.pid, O.month, O.qty} (\sigma_{C.cid=O.cid} (\Pi_{cid} (\sigma_{C cname='Allied'} (C)) \times O))$$

Example 3 (cont.)

□ 查询顾客、代理商以及所订购的商品都位于同一个城市的订单的编号(ordno)

$\Pi_{ordno} ($

$\sigma_{C.city=A.city \wedge P.city=A.city \wedge C.cid=O.cid \wedge A.aid=O.aid \wedge P.pid = O.pid} ($

$\Pi_{cid,city}(C) \times \Pi_{aid,city}(A) \times \Pi_{pid,city}(P) \times O$

)

)

Example 3 (cont.)

□ 在当前的所有顾客中, 查询享受最大折扣(discnt)的顾客的编号(cid)

1) 查询所有顾客的编号, 其结果构成关系R₁

$$R_1 := \Pi_{cid} (C)$$

2) 查询折扣并非最大的顾客 (其折扣低于其他某个顾客的折扣, 或至少存在一个顾客X, 而X的折扣高于当前顾客的折扣) 的编号, 查询结果构成关系R₂

令 S := C, 则:

$$R_2 := \Pi_{C.cid} (\sigma_{C.discnt < S.discnt} (C \times S))$$

3) 利用减法 (difference) 运算获得享受最大折扣顾客的编号

$$T := R_1 - R_2$$

Example 3 (cont.)

□ (合并上述的三个公式) 在当前的所有顾客中, 查询享受最大折扣(discnt)的顾客的编号(cid) (令S := C)

$$\Pi_{cid} (C) - \Pi_{C.cid} (\sigma_{C.discnt < S.discnt} (C \times S))$$

□ 思考题

- 查询具有最小折扣的顾客的编号
- 查询折扣并非最大(小)的顾客的编号
- 查询其折扣为第二大(小)的顾客的编号
- 查询具有最大折扣的顾客的姓名(cname)

回顾

	符号	交换律	结合律
并运算	$R \cup S$	√	√
差运算	$R - S$	✗	✗
投影运算	$\prod_{B1, B2, \dots, Bm} (R)$	✗	投影运算和选择 运算不能互换
选择运算	$\sigma_F(R)$	√	
笛卡儿乘积	$R \times S$	√	√

3.2 关系模型数学理论 — 关系代数

3.2.0 关系模型

3.2.1 关系的表示

3.2.2 关系操纵的表示

3.2.3 关系模型与关系代数

3.2.4 关系代数中的扩充运算

3.2.5 关系代数实例

3.2.4 关系代数中的扩充运算

□关系代数的扩充运算

- 交 (intersection) 运算
- 除 (division) 运算
- 联接 (join) 运算
 - 自然联接 (natural join) 运算

3.2.4 关系代数中的扩充运算

□‘交’ 运算： $R \cap S$

➤ 条件

– 同类关系

➤ 结果

– 关系模式不变，由所有既属于关系R也属于关系S的元组所组成的集合

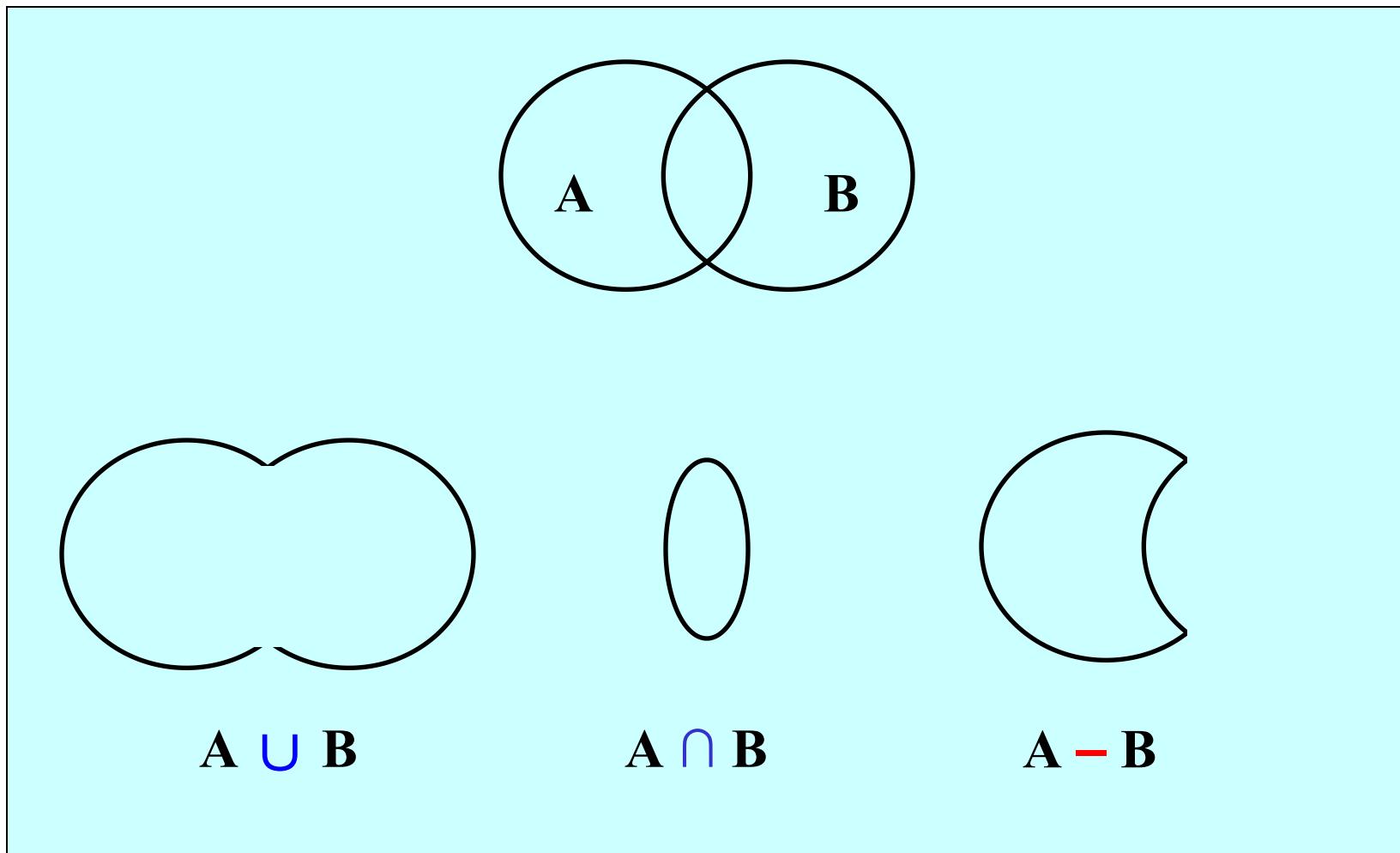
R	A	B	C
	a	b	c
d	a	f	
c	b	d	

S	A	B	C
	b	g	a
d	a	f	

$R \cap S$	A	B	C
	d	a	f

3.2.4 关系代数中的扩充运算

□ ‘并’、‘差’、‘交’ 运算的示意图



3.2.4 关系代数中的扩充运算

□ ‘交’ 运算满足交换律与结合律

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

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

□ ‘交’ 运算并不是一个基本运算，其功能可以由‘差’运算来实现

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

□ 有关‘交’运算和‘差’运算的一组等价变换公式

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

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

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

3.2.4 关系代数中的扩充运算

【例】并，交，差的例子

R	A	B	C
a1	b1	c1	
a1	b2	c3	
a2	b1	c2	

S	A	B	C
a1	b1	c1	
a1	b1	c2	
a1	b2	c3	
a3	b2	c3	

$R \cup S$

A	B	C
a1	b1	c1
a1	b2	c3
a2	b1	c2
a1	b1	c2
a3	b2	c3

$R \cap S$

A	B	C
a1	b1	c1
a1	b2	c3

具有蓝色背景的元组只出现在关系R中，具有黄色背景的元组只出现在关系S中

$R - S$

A	B	C
a2	b1	c2

$S - R$

A	B	C
a1	b1	c2
a3	b2	c3

3.2.4 关系代数中的扩充运算

STUDENT(S)

FN	LN
Susan	Yao
Amy	Ford
Jimmy	Wang
Ramesh	Shah
John	Ford

INSTRUCTOR(I)

FN	LN
Ramesh	Shah
Francis	Johnson
Susan	Yao
John	Smith

S ∪ I

FN	LN
Susan	Yao
Amy	Ford
Jimmy	Wang
Ramesh	Shah
John	Ford
Francis	Johnson
John	Smith

S ∩ I

FN	LN
Susan	Yao
Ramesh	Shah

S - I

FN	LN
Amy	Ford
Jimmy	Wang
John	Ford

I - S

FN	LN
Francis	Johnson
John	Smith

3.2.4 关系代数中的扩充运算

□除运算： $R \div S$

➤ 设关系R和S的关系模式(即它们的属性集)分别是
 $\text{Head}(R)$ 和 $\text{Head}(S)$

➤ 运算条件： $\text{Head}(S) \subset \text{Head}(R)$

— 可以假设为：

- $\text{Head}(R) = \{ A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m \}$
- $\text{Head}(S) = \{ B_1, B_2, \dots, B_m \}$

— 其中：

- S 被称为 ‘除数关系’
- R 被称为 ‘被除数关系’
- 结果关系被称为 ‘商’

除运算 (cont.)

➤ 结果关系 (令 $T = R \div S$)

– 关系模式

- $\text{Head}(T) = \text{Head}(R) - \text{Head}(S) = \{ A_1, A_2, \dots, A_n \}$

– 结果元组

- 关系 S 中的 **所有** 元组在关系 R 中所对应的 **同一个值**
- 设 x 是结果关系 T 中的一个元组 ($x \in T$), 则对于关系 S 中的每一个元组 y 必有:
$$(x, y) \in R$$
- 由所有符合上述条件的元组 x 构成结果关系的元组集合

Result of $T=R \div S$

1. Assume a row x is in T , then:

for each row y in S {

we can find a row z in R , and {

$x(A_i) = z(A_i)$ for $1 \leq i \leq n$

$y(B_j) = z(B_j)$ for $1 \leq j \leq m$

}

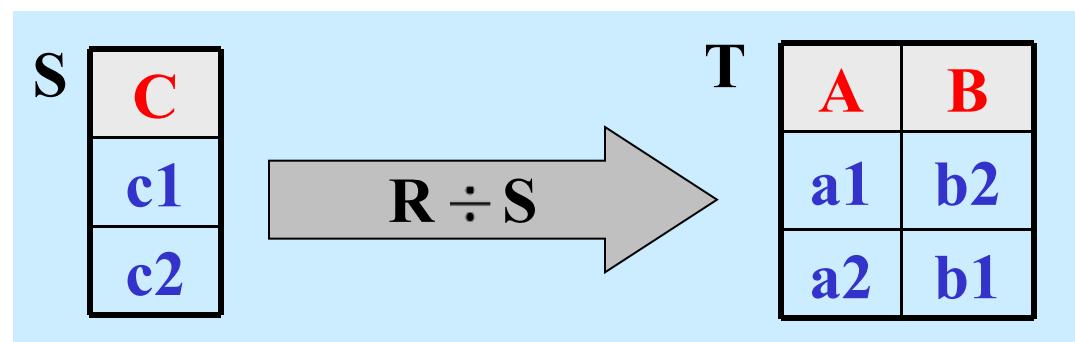
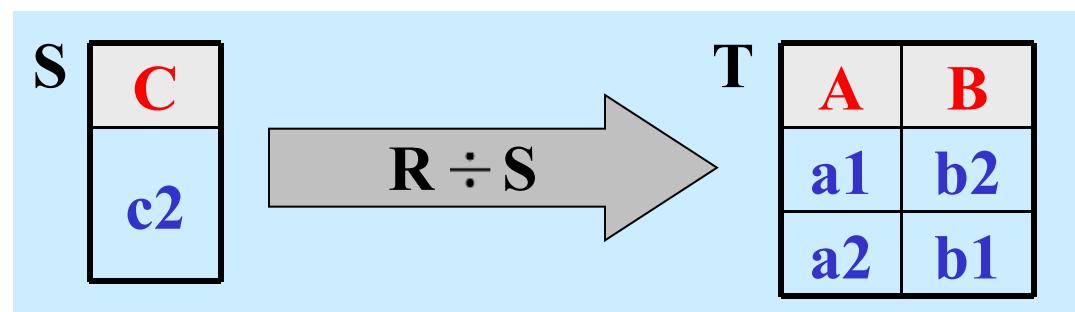
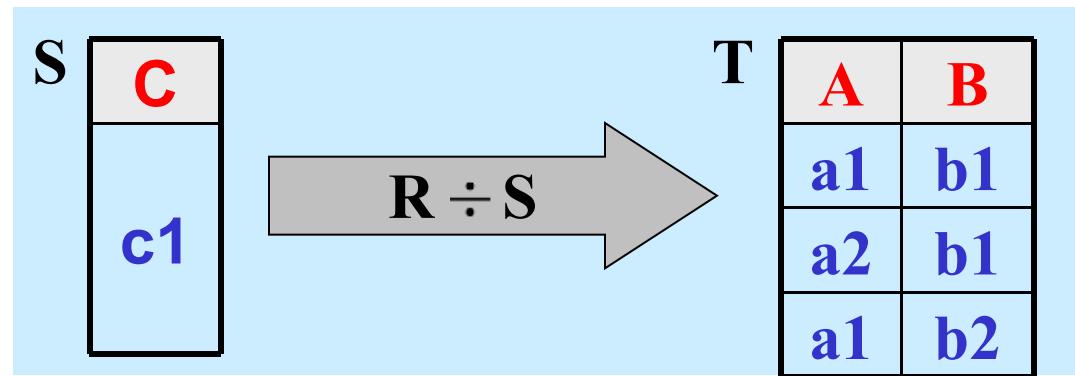
}

2. T contains the largest possible set of rows x

除运算的例子

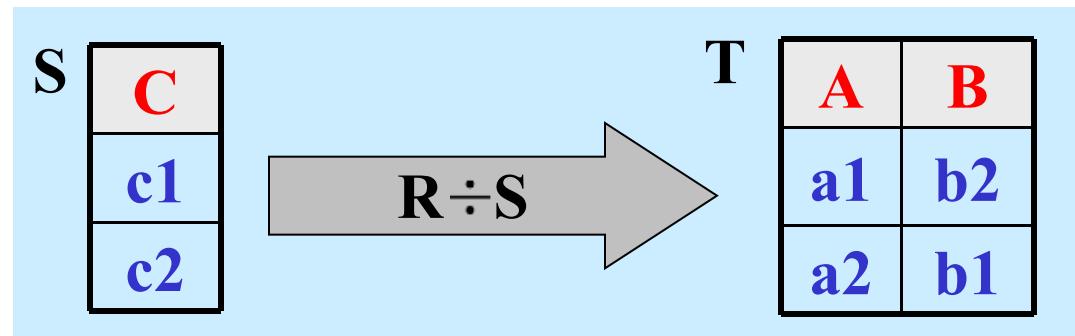
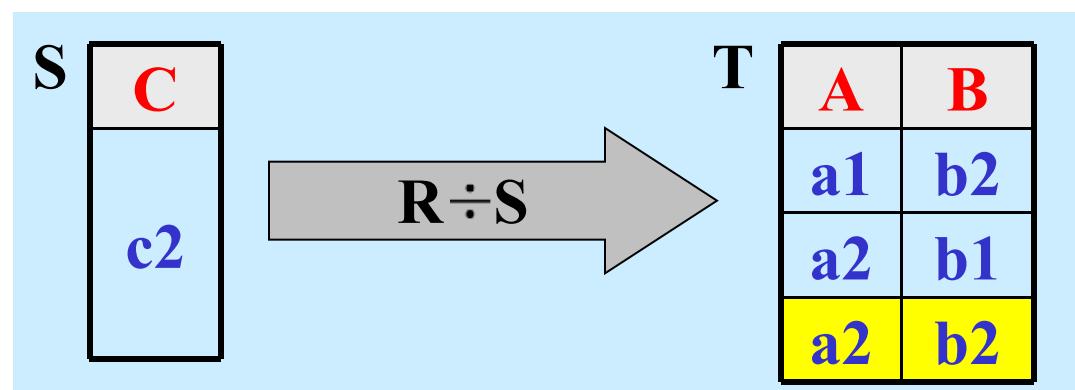
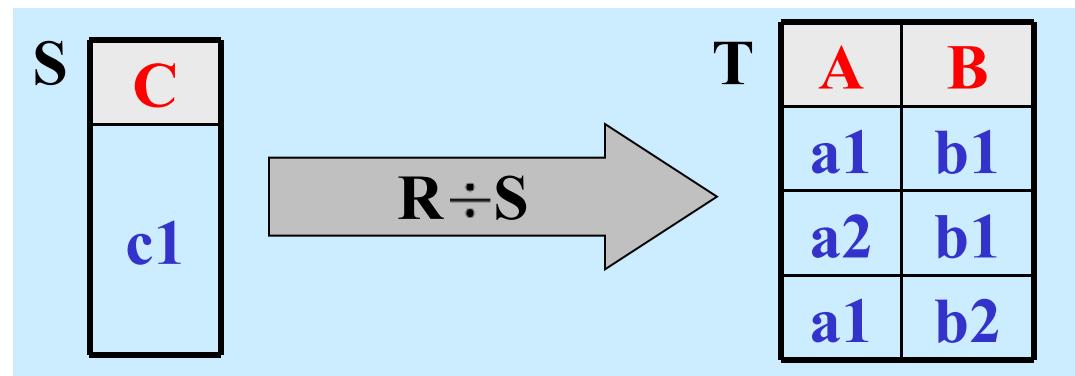
R

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c1
a1	b2	c2
a2	b1	c2
a1	b2	c3
a1	b2	c4
a1	b1	c5



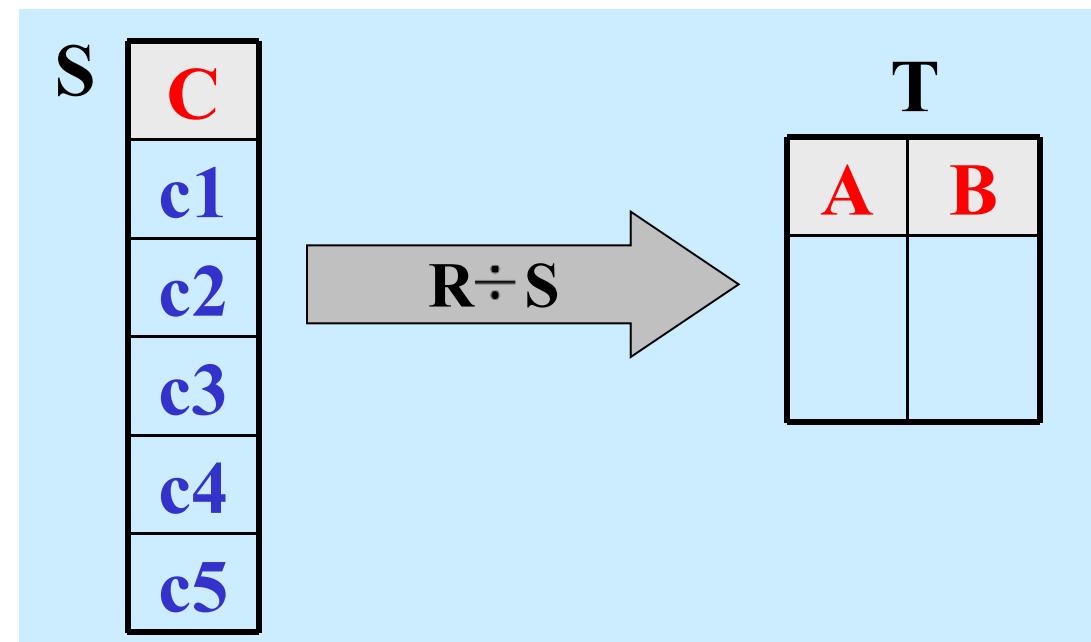
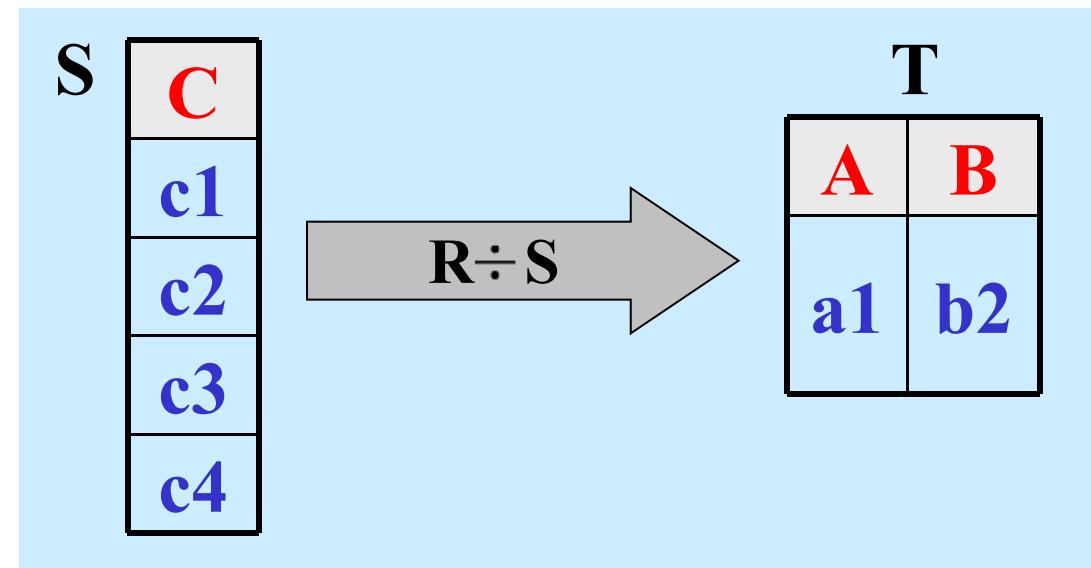
除运算的例子 (cont.)

R		
A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c1
a1	b2	c2
a2	b1	c2
a1	b2	c3
a1	b2	c4
a1	b1	c5
a2	b2	c2



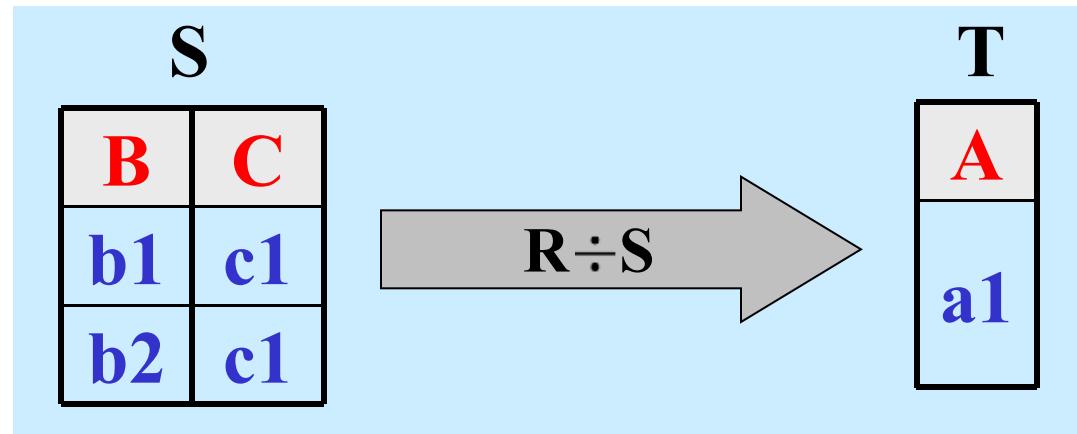
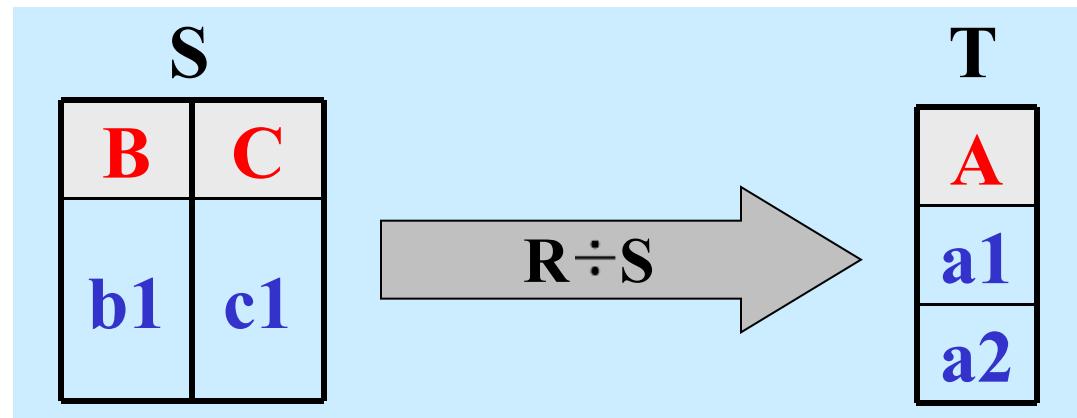
除运算的例子 (cont.)

R		
A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c1
a1	b2	c2
a2	b1	c2
a1	b2	c3
a1	b2	c4
a1	b1	c5



除运算的例子 (cont.)

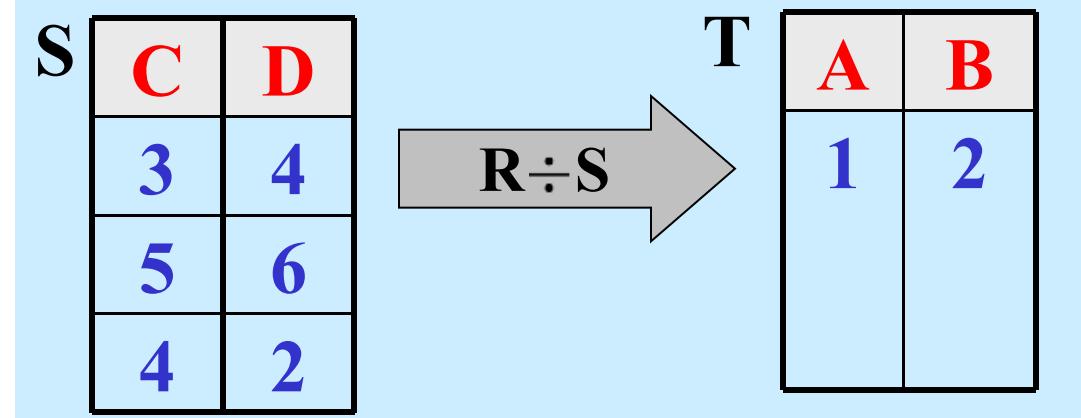
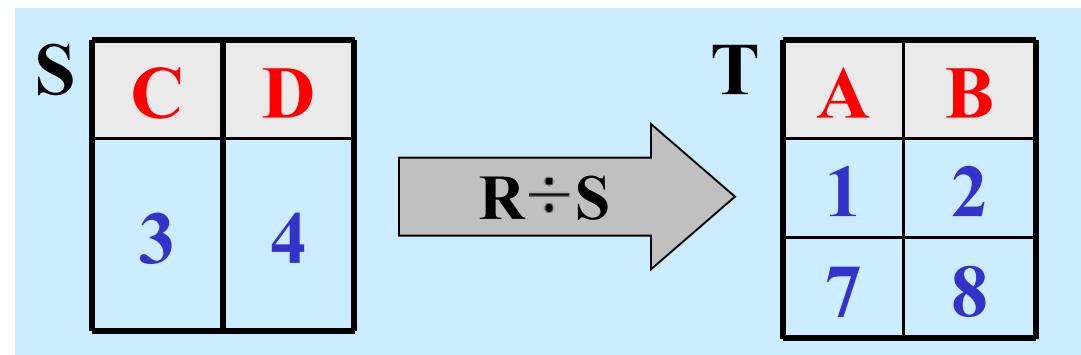
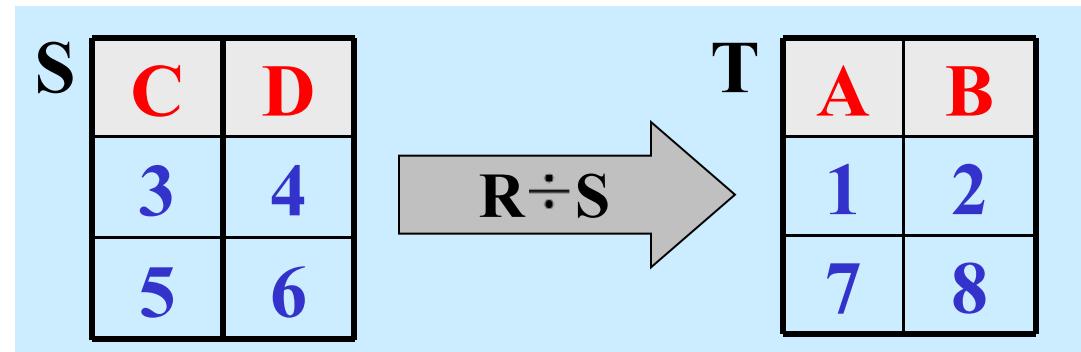
R	A	B	C
a1	b1	c1	
a2	b1	c1	
a1	b2	c1	
a1	b2	c2	
a2	b1	c2	
a1	b2	c3	
a1	b2	c4	
a1	b1	c5	



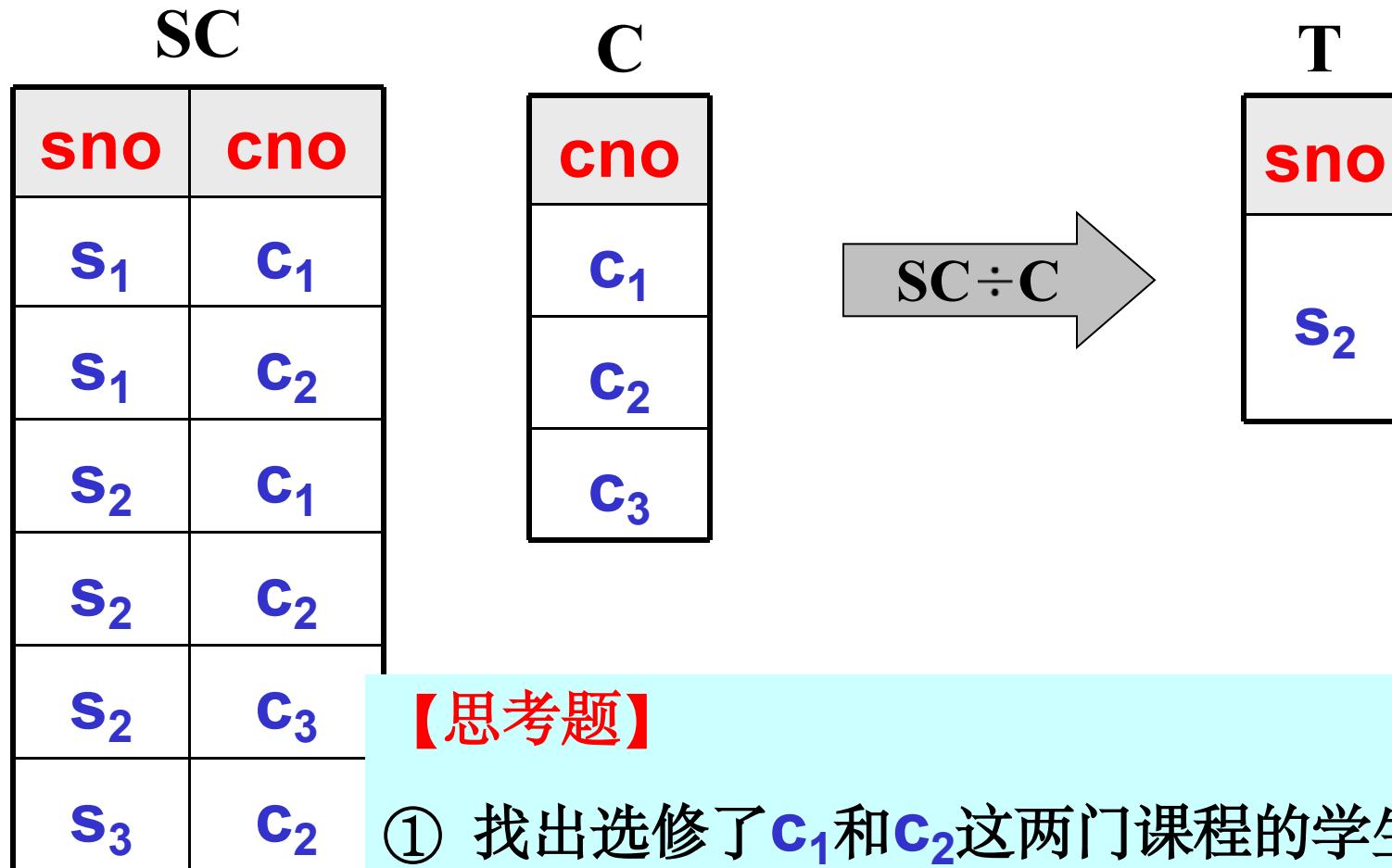
【例3-7】

R

A	B	C	D
1	2	3	4
7	8	5	6
7	8	3	4
1	2	5	6
1	2	4	2



【例3-8】找出修读关系C中的所有课程的学生的学号



【思考题】

- ① 找出选修了**C₁**和**C₂**这两门课程的学生的学号？
- ② 找出选修了**C₂**这门课程的学生的学号？

【例】

SC

sno	cno	G
S ₁	C ₁	80
S ₁	C ₂	85
S ₂	C ₁	90
S ₂	C ₂	70
S ₂	C ₃	85
S ₃	C ₂	85

C

cno
C ₁
C ₂
C ₃

【思考题】在选课关系SC中增加一个属性成绩 G,
然后执行 $SC \div C$, 那么:

- ① 结果关系的关系模式是什么?
- ② 结果关系中的元组又有哪些?

3.2.4 关系代数中的扩充运算

□‘除’ 运算与 ‘笛卡儿乘积’ 的关系

➤如果 $R = T \times S$, 那么有:

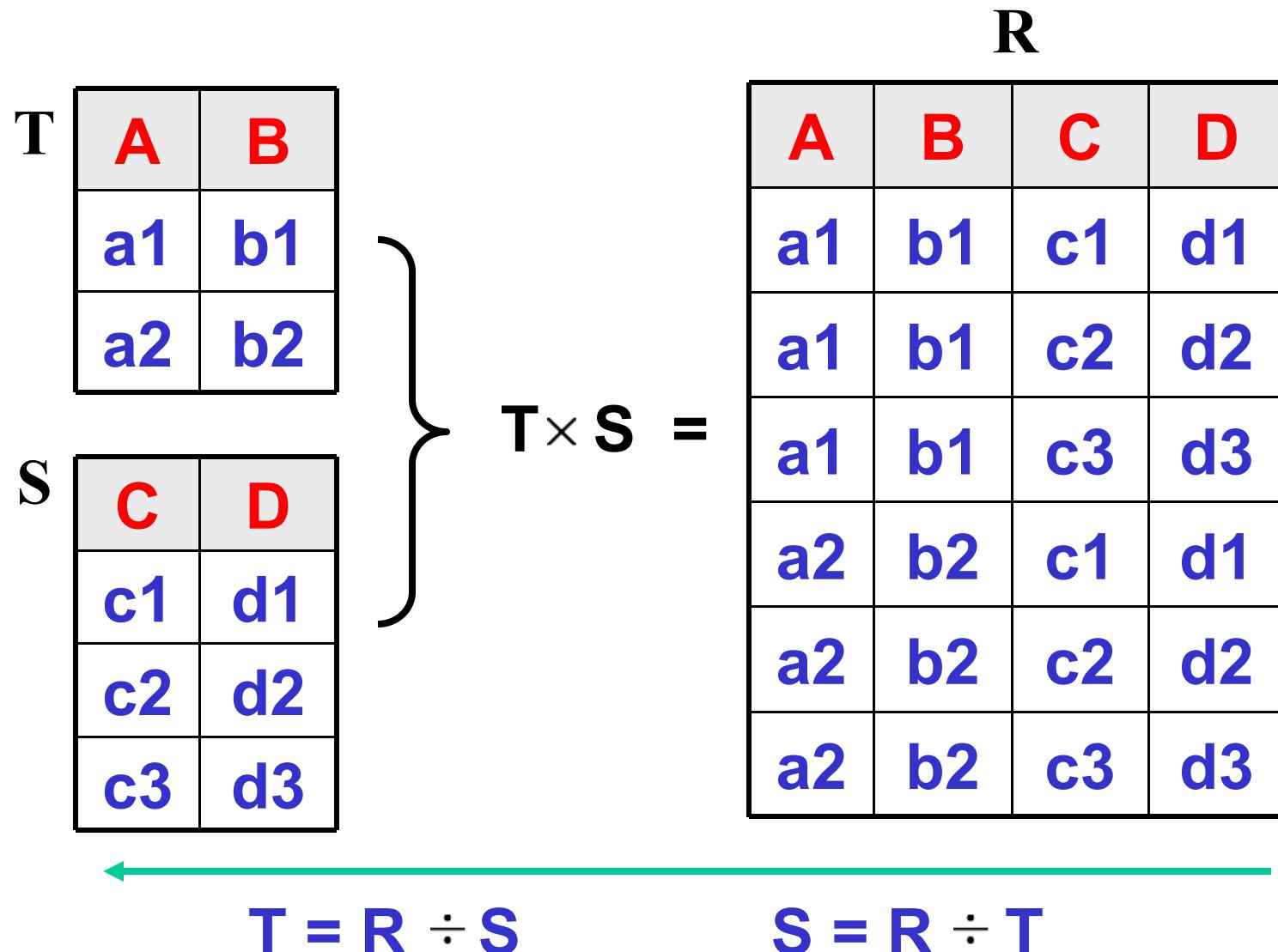
$$T = R \div S$$

$$S = R \div T$$

➤如果 $T = R \div S$, 那么有:

$$T \times S \subseteq R$$

【例】除运算与笛卡儿乘积的关系



【例】除运算与笛卡儿乘积的关系

R			
A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a1	b1	c3	d3
a1	b2	c1	d1
a2	b2	c1	d1
a2	b2	c2	d2
a2	b2	c3	d3
a2	b1	c2	d2

S	
C	D
c1	d1
c2	d2
c3	d3

$T = R \div S$

R' := T \times S			
A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a1	b1	c3	d3
a2	b2	c1	d1
a2	b2	c2	d2
a2	b2	c3	d3

3.2.4 关系代数中的扩充运算

□‘除’ 运算与基本关系运算的关系

IF

$$\text{Head}(R) = \{ A_1 \dots A_n \ B_1 \dots B_m \}$$

$$\text{Head}(S) = \{ B_1 \dots B_m \}$$

THEN

$$R \div S = \Pi_{A_1 \dots A_n}(R) - \Pi_{A_1 \dots A_n}((\Pi_{A_1 \dots A_n}(R) \times S) - R)$$

‘除’ 运算的推导过程

1) $T_{max} := \Pi_{A1...An}(R)$

// T_{max} 是最大可能的结果元组集合

2) $R_{max} := T_{max} \times S$

// R_{max} 与关系R是同类关系

3) $T_1 := R_{max} - R$

4) $T_2 := \Pi_{A1...An}(T_1)$

// T_2 是关系 T_{max} 中不满足除运算的结果要求的那些元组，即：对于关系 T_2 中的任一个元组 q ，至少能在关系 S 中找到一个元组 s ，使得由元组 q 和 s 所构成的元组 (q,s) 不在关系 R 中出现

5) $R \div S := T_{max} - T_2$

【例】‘除’ 运算的推导过程

SC	C	T_{\max}	SC_{\max}	$\Pi_{sno}(SC_{\max} - SC)$
sno S_1 S_1 S_2 S_2 S_2 S_3	cno C_1 C_2 C_1 C_2 C_3	sno S_1 S_2 S_3	sno S_1 S_1 S_2 S_2 S_2 S_3	sno S_1 S_1 S_3

没有出现在SC中的元组，其sno不符合‘除’运算对结果关系中元组的要求

结果关系T

sno S_2

【例】‘除’ 运算的推导过程

SC

sno	cno	G
S_1	C_1	80
S_1	C_2	85
S_2	C_1	90
S_2	C_2	70
S_2	C_3	85
S_3	C_2	85

C

cno
C_1
C_2
C_3

T_{max}

sno	G
S_1	80
S_1	85
S_2	90
S_2	70
S_2	85
S_3	85

3.2.4 关系代数中的扩充运算

□ 联接 (join) 运算: $R \bowtie_F S$

➤ 又称 θ -联接运算, 可以将关系 R 和关系 S 根据联接条件 F 合并为一个关系

□ 设结果关系为 T , 则关系 T 与关系 R 和 S 的关系是:

➤ $\text{Head}(T) = \text{Head}(R) \cup_{\text{all}} \text{Head}(S)$

– 不必消除它们之间的同名属性, 但在结果关系中必须对同名属性进行换名

➤ 从关系 R 和 S 中分别任取一个元组 r 和 s , 如果元组 r 和元组 s 之间满足联接条件 F , 那么可以由 r 和 s 合并构成结果关系 T 中的一个元组, 即: $(r,s) \in T$

3.2.4 关系代数中的扩充运算

□ 联接条件F的构造方式: $R \bowtie_F S$

➤ 基本条件: $i \theta j$

— 其中:

- i是关系R中的属性, j是关系S中的属性
 - » 也可以用属性在关系中的位置代替属性名
 - θ 是比较运算符
 - » 大于联接, 小于联接
 - » 等值联接
- 由若干个基本条件经逻辑运算符 \wedge 和 \vee 连接而成的复杂条件

【例3-9】

R

	A	B	C	D
1	2	3	4	
3	2	1	8	
7	3	2	1	

S

	E	F
1		8
7		9
5		2

R  S
D>E

A	B	C	D	E	F
1	2	3	4	1	8
3	2	1	8	1	8
3	2	1	8	7	9
3	2	1	8	5	2

R  S
D=E

A	B	C	D	E	F
7	3	2	1	1	8

3.2.4 关系代数中的扩充运算

□ 联接运算的推导公式

$$R \bowtie_F S = \sigma_F(R \times S)$$

□ 联接运算与笛卡儿乘积运算的关系

– 设 $T_1 = R \times S$, $T_2 = R \bowtie_F S$, 则:

$\text{Head}(T_2) = \text{Head}(T_1)$, 且 $T_2 \subseteq T_1$

3.2.4 关系代数中的扩充运算

□ 自然联接 (natural join) 运算： $R \bowtie S$

➤ 功能：根据两个关系中的同名属性（公共属性）
进行等值联接

➤ 运算条件
— 关系 R 和 关系 S 有公共属性：

$$\text{Head}(R) \cap \text{Head}(S) \neq \emptyset$$

3.2.4 关系代数中的扩充运算

□ 自然联接 (natural join) 运算: $R \bowtie S$

➤ 运算结果

- 结果关系的属性集合为 $\text{Head}(R) \cup \text{Head}(S)$
- 结果关系中的元组:
 - 从关系R和关系S中分别任取一个元组 r 和 s ,
如果 元组 r 和元组 s 在它们的同名属性上的
取值都相等, 那么可以由 r 和 s 合并构成结果
关系中的一个元组
 - 同名属性上的取值在结果关系中只保留一份

3.2.4 关系代数中的扩充运算

□ 自然联接运算的推导公式

- 设有关系R和S， R有属性A₁, A₂, …, A_n, S有属性B₁, B₂, …, B_m, 它们之间的公共属性为A₁, A₂, …, A_j与 B₁, B₂, …, B_j
- 关系R与S的自然联接运算的结果关系的推导公式是：

$$R \bowtie S = \Pi_{A_1, A_2, \dots, A_n, B_{j+1}, \dots, B_m} (\sigma_{A_1=B_1 \wedge A_2=B_2 \wedge \dots \wedge A_j=B_j} (R \times S))$$

【例3-10】自然联接运算的例子

R

A	B	C	D
1	2	3	4
1	5	8	3
2	4	2	6
1	1	4	7

S

D	E
5	1
6	4
7	3
6	8

R \bowtie S

A	B	C	D	E
2	4	2	6	4
2	4	2	6	8
1	1	4	7	3

【例】自然联接运算的例子

R

A	B	C
1	2	3
2	4	6
3	1	4

S

B	C	D
2	3	4
2	3	5
1	4	2
3	5	1

$R \bowtie S$

A	B	C	D
1	2	3	4
1	2	3	5
3	1	4	2

其它的一些关系运算

□ 自然联接 $R \bowtie S$

□ 其它的一些联接运算

➤ 外联接 (outer join)

$$R \Join S$$

➤ 左外联接 (left outer join)

$$R \leftarrow \Join S$$

➤ 右外联接 (right outer join)

$$R \rightarrow \Join S$$

可以通过结果
来重构原来的关系 R 和 / 或 S

【例】外联接、左外联接、右外联接的例子

R	A	B
a1	b1	
a2	b2	
a3	b5	

S	B	C
b1	c1	
b2	c2	
b3	c3	
b4	c4	

$R \bowtie S$

A	B	C
a1	b1	c1
a2	b2	c2

$R \triangleright\!\!\! \bowtie \triangleleft S$

A	B	C
a1	b1	c1
a2	b2	c2
a3	b5	null
null	b3	c3
null	b4	c4

$R \triangleright\!\!\! \bowtie S$

A	B	C
a1	b1	c1
a2	b2	c2
a3	b5	null

$R \triangleleft\!\!\! \bowtie S$

A	B	C
a1	b1	c1
a2	b2	c2
null	b3	c3
null	b4	c4

关系代数小结

□ 关系的表示

- 元组的集合

□ 关系操纵的表示

- 集合上的运算
- 五种基本运算
 - 并 \cup , 差 –
 - 投影 Π , 选择 σ , 笛卡儿乘积 \times

- 三种扩充运算

- 交 \cap

- 除 \div

- θ -联接  , 自然联接  , 外联接 

3.2 关系模型数学理论 — 关系代数

3.2.0 关系模型

3.2.1 关系的表示

3.2.2 关系操纵的表示

3.2.3 关系模型与关系代数

3.2.4 关系代数中的扩充运算

3.2.5 关系代数实例

3.2.5 关系代数实例

□ 求解过程

- 1) 确定查询目标（结果关系中的属性）
- 2) 明确查询条件
- 3) 选择从条件到目标的查找路径，并据此确定操作对象，即：
 - 在操作过程中需要使用到那些关系？
 - 这些关系又是如何被联接成一个关系的？

3.2.5 关系代数实例

□ 求解过程 (cont.)

4) 关系的合并

- 根据步骤 3) 的分析结果进行关系的联接

5) 元组的选择

- 根据步骤 2) 的分析结果(查询条件)进行元组的选择

6) 属性的指定

- 根据步骤 1) 的分析结果执行投影操作

3.2.5 关系代数实例

□ 例子数据库的关系模式

- 学生关系： **S(sno, sn, sd, sa)**
- 课程关系： **C(cno, cn, pno)**
- 选课关系： **SC(sno, cno, G)**

学生 S (sno, sn, sd, sa)

课程 C (cno, cn, pno)

选课SC (sno, cno, g)

【例3.11】检索学生所有情况

S

【例3.12】检索学生年龄大于等于20岁的学生姓名

$\Pi_{sn} (\sigma_{sa \geq 20} (S))$

【例3.13】检索预修课号为C2的课程的课程号

$\Pi_{cno} (\sigma_{pno='C2'} (C))$

学生 S (sno, sn, sd, sa)

课程 C (cno, cn, pno)

选课SC (sno, cno, g)

【例3.14】检索课程号为C，且成绩为A的所有学生姓名

$\Pi_{sn} (\sigma_{cno='C'} \wedge g='A' (S \bowtie SC))$

【例3.15】检索s1所修读的所有课程名及其预修课号

$\Pi_{cn, pno} (\sigma_{sno='S1'} (C \bowtie SC))$

【例3.16】检索年龄为23岁的学生所修读的课程名

$\Pi_{cn} (\sigma_{sa=23} (S \bowtie SC \bowtie C))$

学生 S (sno, sn, sd, sa)

课程 C (cno, cn, pno)

选课SC (sno, cno, g)

【例3.17】检索至少修读为S₅所修读的一门课的学生姓名

【分析】

1) 结果元组需要满足的条件?

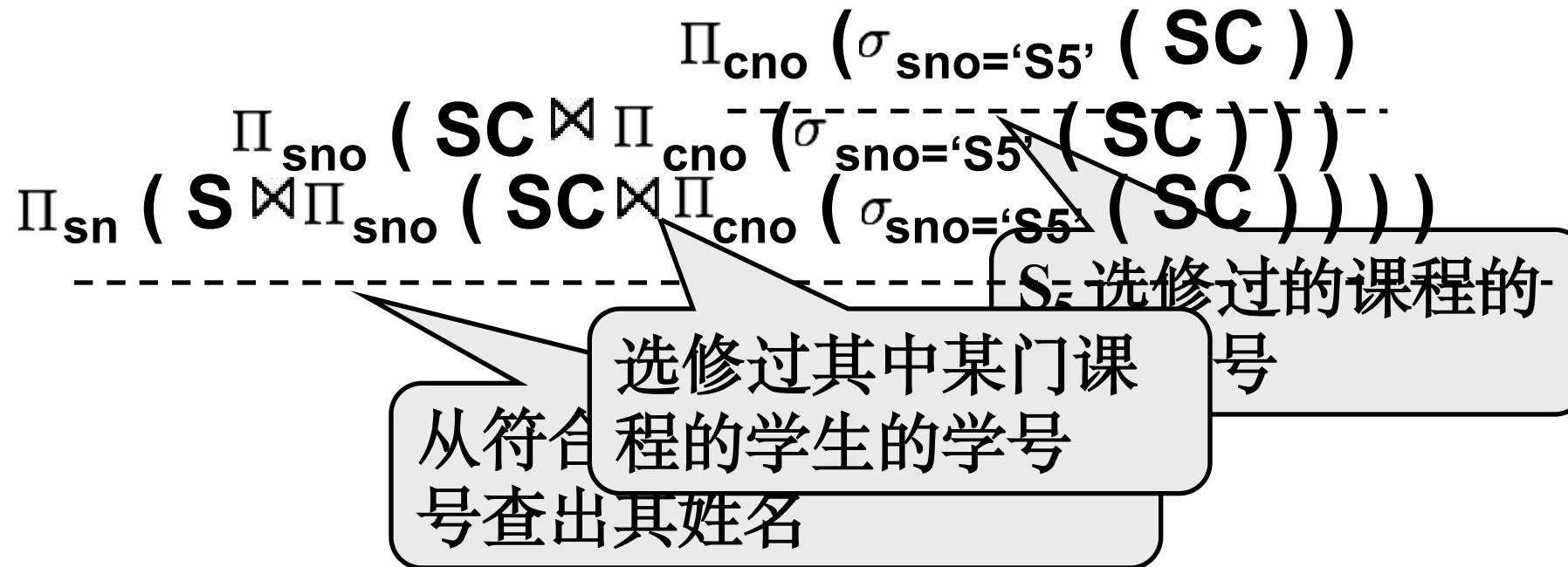
修读了S₅所修读过的某一门课程

2) S₅修读了哪些课程?

3) 如何从S₅修读过的课程查出满足要求的学生元组?

$\Pi_{sn} (S \bowtie \Pi_{sno} (SC \bowtie \Pi_{cno} (\sigma_{sno='S5'} (SC))))$

3.2.5 关系代数实例



学生 S (sno, sn, sd, sa)

课程 C (cno, cn, pno)

选课SC (sno, cno, g)

【例3.18】检索修读S4所修读的所有课程的学生姓名

【分析】

1) 结果元组需要满足的条件?

修读了S4所修读过的所有课程

2) S4修读了哪些课程?

3) 如何根据一组课程 (S4所修读过的课程) 查出
修读了其中的所有课程的学生元组?

$\Pi_{sn} (S \bowtie (\Pi_{sno,cno}(SC) \div \Pi_{cno} (\sigma_{sno='S4'}(SC))))$



【例3.18】几种错误的表示方式

- 1) $\Pi_{sn} (S \bowtie \Pi_{sno} (SC \bowtie \Pi_{cno} (\sigma_{sno='S4'} (SC))))$
- 2) $\Pi_{sn} (S \bowtie (SC \div \Pi_{cno} (\sigma_{sno='S4'} (SC))))$
- 3) $\Pi_{sn} (S \bowtie (\Pi_{sno,cno} (SC) \div \sigma_{sno='S4'} (SC)))$
- 4) $\Pi_{sn} (S \bowtie (SC \div \sigma_{sno='S4'} (SC)))$
- 5) $\Pi_{sn} ((S \bowtie SC) \div \Pi_{cno} (\sigma_{sno='S4'} (SC)))$
- 6) $\Pi_{sn,cno} (S \bowtie SC) \div \Pi_{cno} (\sigma_{sno='S4'} (SC))$

学生 S (sno, sn, sd, sa)

课程 C (cno, cn, pno)

选课SC (sno, cno, g)

【例3.19】检索修读所有课程的学生学号

$\Pi_{sno,cno} (SC) \div \Pi_{cno} (C)$

【思考题】检索所有学生都选修过的课程的编号和名称

【例3.20】检索不修读任何课程的学生学号

$\Pi_{sno} (S) - \Pi_{sno} (SC)$

【思考题】

- 1) 检索所有计算机系学生都选修过的课程的课程号和课程名

- 2) 检索所有选修《数据库》的学生都选修过的课程的编号

3.2.5 关系代数实例

【例3.21】在关系C中增添一门新课程

$$C \cup \{ (C_{13}, ML, C_3) \}$$

【例3.22】学号为S₁₇的学生因故退学,请在S及SC中将其除名

$$S - (\sigma_{s\#=‘s17’}(S))$$

$$SC - (\sigma_{s\#=‘s17’}(SC))$$

【例3.23】将关系S中学生S₆的年龄改为22岁

【例3.24】将关系S中的年龄均增加1岁

$$S (sno, sn, sd, sa + 1)$$

关系代数运算 - 小结

□ 关系代数

➤ 基本运算

— 并，差，笛卡儿乘积，投影，选择

➤ 扩充运算

— 交，除，联接与自然联接

➤ 其它运算

— 外联接，左外联接，右外联接

□ 关系代数应用的例子

□ 总结