第2章 关系数据库

本章要点

本章介绍关系数据库的基本概念，基本概念围绕关系数据模型的三要素展开，利用集合代数、谓词演算等抽象的数学知识，深刻而透彻地介绍了关系数据结构，关系数据库操作及关系数据库完整性等的概念与知识。而抽象的关系代数与基于关系演算的ALPHA语言乃重中之重。

本章内容：

- 2.1 关系模型
- 2.2 关系数据结构及其形式化定义
- 2.3 关系的完整性
- 2.4 关系代数
- 2.5 关系演算
- 2.6 小结

2.1 关系模型

关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。

- 关系数据结构
- 关系操作集合
- 关系完整性

返回本章首页

1、关系模型的数据结构——关系

关系模型的数据结构：非常单一，在用户看来，关系模型中数据的逻辑结构是一张二维表。但关系模型的这种简单的数据结构能够表达丰富的语义，描述出现实世界的实体以及实体间的各种联系。

2、关系操作

1) 关系操作包括：选择（select）、投影(project)、连接(join)、除(divide)、并(union)、交(intersection)、差(difference)等查询(query)操作和增加(insert)、删除(delete)、修改(update)等更新操作两大部分。查询的表达能力是其中最主要的部分。

2) 关系操作表示方式：

   a) 代数方式→关系代数

   b) 逻辑方式→关系演算：元组关系演算和域关系演算

3) 关系数据语言可以分成三类：

   a) 关系代数：用对关系的集合运算表达查询要求，例如 ISBL。

   b) 关系演算：用谓词表达查询要求，可分为两类：①元组关系演算：谓词变元的基本对象是元组变量，例如 APLHA、QUEL；②域关系演算：谓词变元的基本对象是域变量，例如QBE。

   c) 关系数据语言，例如SQL。

BACK

3、关系的三类完整性约束

关系模型提供了丰富的完整性控制机制，允许定义三类完整性：实体完整性、参照完整性和用户自定义的完整性。其中实体完整性和参照完整性是关系模型必须满足的完整性约束条件，应该由关系系统自动支持。用户自定义的完整性是应用领域特殊要求而需要遵循的约束条件，体现了具体领域中的语义约束。

下面将从数据模型的三要素出发，逐步介绍关系模型的数据结构（包括关系的形式化定义及有关概念）、关系的三类完整性约束、关系代数与关系演算操作等。

BACK

2.2 关系数据结构及其形式化定义

在关系模型中，无论是实体还是实体之间的联系均由单一的结构类型即关系（二维表）来表示关系模型是建立在集合代数的基础上的，这里我们从集合论角度给出关系数据结构的形式化定义。

- **2.2.1 关系**
- **2.2.2 关系模式**
- **2.2.3 关系数据库**

返回本章首页

2.2.1 关系

1. 域（Domain）

定义2.1 域是一组具有相同数据类型的值的集合。又称为值域(用D表示)。域中所包含的值的个数称为域的基数（用m表示）。在关系中就是用域来表示属性的取值范围的。

例如，自然数、整数、实数、长度小于10字节的字符串集合、1-16之间的整数都是域，又如：

D_1={张三，李四}　　D_1的基数m_1为2

D_2={男，女}　　D_2的基数m_2为2

D_3={19，20，21}　　D_3的基数m_3为3

BACK

2. 笛卡尔积（Cartesian Product）

定义2.2 给定一组域D_1、D_2、...D_n(这些域中可以包含相同的元素，即可以完全不同，也可以部分或全部相同)，D_1、D_2、...D_n 的笛卡尔积为：$D_1 \times D_2 \times ... \times D_n = \{ (d_1, d_2, ..., d_n) \mid d_i \in D_i, i=1, 2, ..., n\}$

由定义可以看出，笛卡尔积也是一个集合。其中：

(1) 其中每一个元素（d_1，d_2，...，d_n）叫作一个n元组（n-tuple），或简称为元组（Tuple）。但元组不是d_i的集合，元组由d_i按序排列而成。

(2) 元素中的每一个值d_i叫作一个分量（Component）。分量来自相应的域（ $d_i \in D_i$）。

(3) 若D_i（i＝1，2，...，n）为有限集，其基数（Cardinal number）为m_i（i＝1，2，...，n），则$D_1 \times D_2 \times ... \times D_n$ 的基数为n个域的基数累乘之积，即$M= \prod_{i=1}^{n} m_i$

(4) 笛卡尔积可表示为一个二维表。表中的每行对应一个元组，表中的每列对应一个域。

BACK

例如D_1与D_2的笛卡尔积：

$D_1 \times D_2$={（张三，男），（张三，女），（李四，男），（李四，女）}

可以表示成二维表，如下表2.1所示：

表2.1笛卡尔积 $D_1 \times D_2$

姓名	性别
张三	男
张三	女
李四	男
李四	女

$D_1 \times D_2 \times D_3$={（张三，男，19），（张三，男，20），（张三，男，21），（张三，女，19），（张三，女，20），（张三，女，21），（李四，男，19），（李四，男，20），（李四，男，21），（李四，女，19），（李四，女，20），（李四，女，21）}

BACK

3. 关系（Relation）

定义2.3 $D_1 \times D_2 \times \ldots \times D_n$的任一子集叫作在域$D_1, D_2, \ldots, D_n$ 上的关系，用 $R(D_1, D_2, \ldots D_n)$表示。如上例中$D_1 \times D_2$ 笛卡尔积的子集可以构成关系T_1，如下表2.3所示：

表2.3 $D_1 \times D_2$笛卡尔积的子集（关系T1）

姓名	性别
张三	男
李四	女

R表示关系的名字，n是关系的目或度（Degree）。
当n=1时，称为单元关系。
当n=2时，称为二元关系。
……
当n=m时，称为m元关系。
关系中的每个元素是关系中的元组，通常用t表示。

BACK

1） **基本概念：**

关系：是笛卡尔积的子集，所以关系也是一个二维表。

元组：对应表的每行

域：　对应表的每列

属性（Attribute）：由于域可以相同，为了加以区分，必须对每列起一个唯一的名字。

候选码（Candidate key）：若关系中的某一属性组的值能唯一地标识一个元组，则称该属性组为候选码。关系至少含有一个候选码。

主码（Primary key）：若一个关系有多个候选码，则选定其中一个为主控使用者称为主码。

主属性（Prime attribute）：候选码中的诸属性。

非主属性（Non-key attribute）：不包含在任何侯选码中的属性。 在最简单的情况下，侯选码只包含一个属性。

全码（All-key）：关系模式的所有属性组是这个关系模式的侯选码，称为全码（All-key）。

BACK

2) 基本关系的性质：

① 列是同质的（Homogeneous），即每一列中的分量是同一类型的数据，来自同一个域。

② 不同的列可出自同一个域，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。

③ 列的顺序无所谓，即列的次序可以任意交换。

④ 任意两个元组不能完全相同。

⑤ 行的顺序无所谓，即行的次序可以任意交换。

⑥ 分量必须取原子值，即每一个分量都必须是不可分的数据项。

BACK

关系模型要求关系必须是规范化的，即要求关系模式必须满足一定的规范条件。这些规范条件中最基本的一条就是，关系的每一个分量必须是不可再分的数据项。规范化的关系称为范式。例如：

表2.4 课程关系C

课程名	学时	
	理论	实验
数据库	52	20
C语言	45	20
数据结构	55	30

表2.5 课程关系C

课程名	理论学时	实验学时
数据库	52	20
C语言	45	20
数据结构	55	30

如表2.4的关系就不规范，存在“表中有表”现象，可将它进行规范化为表2.5所示的关系。

BACK

2.2.2 关系模式

关系模式是一个5元组:

定义2.4 关系的描述称为关系模式（Relation Schema）。一个关系模式应当是一个五元组。它可以形式化地表示为：R（U, D, dom, F）。其中R为关系名，U为组成该关系的属性名集合，D为属性组U中属性所来自的域的集合，dom为属性向域的映象集合，F为属性间数据的依赖关系集合。

关系模式通常可以简记为：R(A_1, A_2, …, A_n)或R(U)。其中R为关系名，A_1, A_2, …, A_n为属性名。而域名及属性向域的映象常常直接说明为属性的类型、长度。

关系实际上就是关系模式在某一时刻的状态或内容。也就是说，关系模式是型，关系是它的值。关系模式是静态的、稳定的，而关系是动态的、随时间不断变化的，因为关系操作在不断地更新着数据库中的数据。把关系模式和关系统称为关系。

BACK

2.2.2 关系模式

关系模式的五元组可以如图2.1来说明，通过这五个方面，一个关系被充分地刻画、描述出来了。

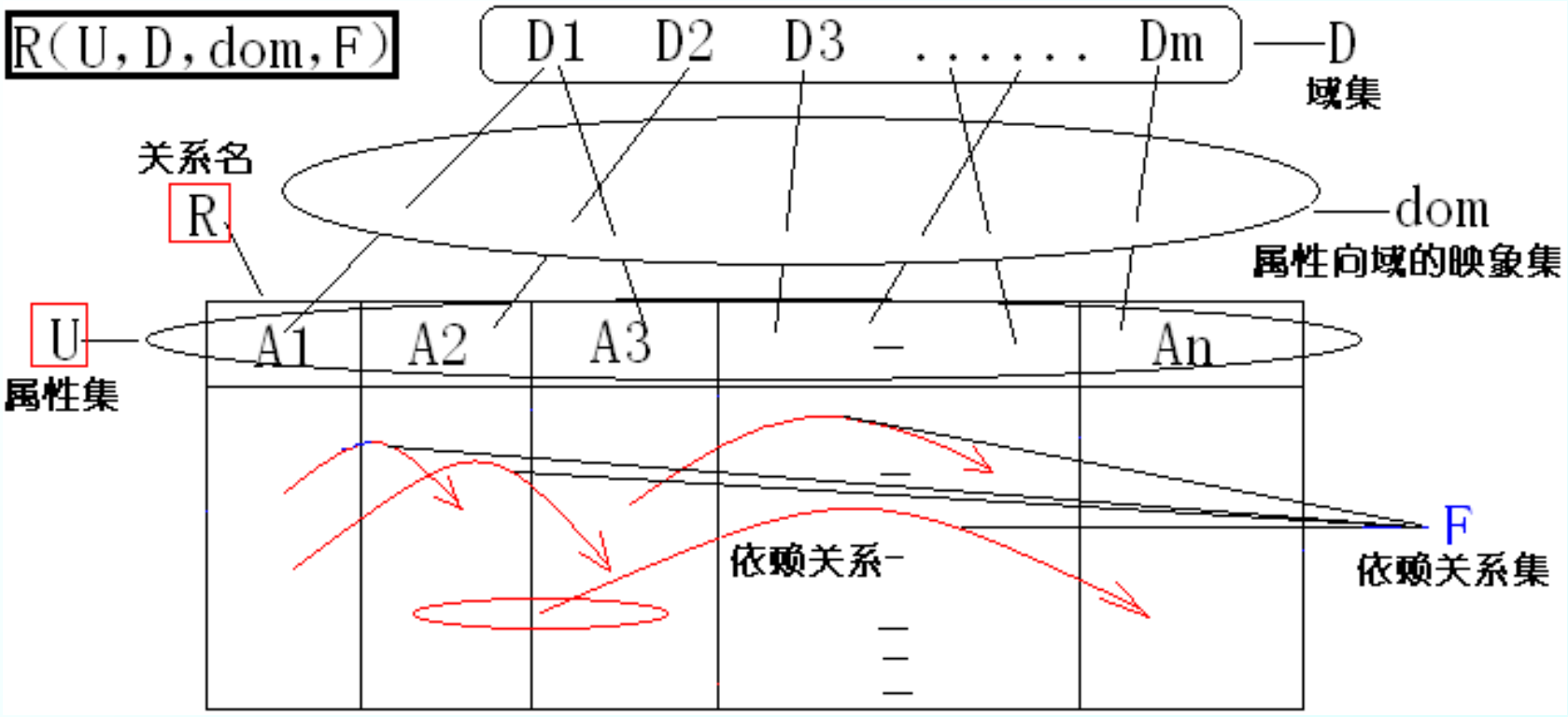


图2.1 关系模式的五元组示意图

BACK

2.2.3 关系数据库

在关系模型中，实体以及实体间的联系都是用关系来表示。例如学生实体、课程实体、学生与课程之间的多对多选课联系都可以分别用一个关系（或二维表）来表示。在一个给定的现实世界领域中，所有实体及实体之间的联系的关系的集合构成一个关系数据库。

关系数据库也有型和值之分。关系数据库的型也称为关系数据库模式，是对关系数据库的描述，是关系模式的集合。关系数据库的值也称为关系数据库，是关系的集合。关系数据库模式与关系数据库通常统称为关系数据库。

BACK

2.3 关系的完整性

关系模型的完整性规则是对关系的某种约束条件。关系模型中可以有三类完整性约束：实体完整性、参照完整性和用户定义的完整性。

- 实体完整性
- 参照完整性
- 自定义完整性

返回本章首页

一、实体完整性（Entity Integrity）

规则2.1 实体完整性规则：若属性组（或属性）K是基本关系R的主码（或称主关键字），则所有元组**K的取值唯一，并且K中属性不能全部或部分取空值**。

例如：在课程关系T中，若“课程名”属性为主码，则“课程名”属性不能取空值，并且课程名要唯一。

对于实体完整性规则说明如下：

1） 实体完整性规则是针对基本关系而言的。一个基本表通常对应现实世界的一个实体集。

2） 关系模型中以主码作为其唯一性标识，主码中属性即主属性不能取空值。

二、参照完整性（Referential integrity）

定义2.5 设F是基本关系R的一个或一组属性，但不是关系R的码，如果F与基本关系S的主码Ks相对应，则称F是基本关系R的外码（Foreign key），并称基本关系R为参照关系（Referencing relation），基本关系S为被参照关系（Referenced relation）或目标关系（Target relation）。关系R和S可能是相同的关系，即自身参照。

目标关系S的主码K_s和参照关系的外码F必须定义在同一个（或一组）域上。参照完整性规则就是定义外码与主码之间的引用规则。

[例1] 学生实体和专业实体可以用下面的关系表示，其中主码用下划线标识：

学生（学号，姓名，性别，年龄，系别号）

系别（系别号，系名）

BACK

［例 2］学生，课程，学生与课程之间的多对多联系可以如下三个关系表示：

学生（学号，姓名，性别，年龄，系别号）

课程（课程号，课程名，课时）

选修（学号，课程号，成绩）

例1中，学生关系的“系别号”与系别关系的“系别号”相对应，因此，“系别号”属性是学生关系的外码，是系别关系的主码。这里系别关系是被参照关系，学生关系为参照关系。例2中，选修关系的“学号”属性与学生关系的“学号”属性相对应，“课程号”属性与课程关系的“课程号”属性相对应，因此“学号”和“课程号”属性是选修关系的外码，这里学生关系和课程关系均为被参照关系，选修关系为参照关系。

BACK

规则2.2 参照完整性规则：若属性（或属性组）F是基本关系R的外码，它与基本关系S的主码Ks相对应（基本关系R和S可能是相同的关系），则对于R中每个元组在F上的值必须为：**或者取空值（F的每个属性值均为空值）；或者等于S中某个元组的主码值。**

例如，例1中学生关系中的每个元组的“系别号”属性只能取下面两类值：空值，表示尚未给该学生分配系别非空值，这是该值必须是系别关系中某个元组的“系别号”的值，表示该学生不可能分配到一个不存在的系中。即被参照关系“系别”中一定存在一个元组，它的主码值等于该参照关系“学生”中的外码值。

对于例2，按照参照完整性规则，“学号”和“课程号”属性也可以取两类值：空值或目标关系中已经存在的值。但由于“学号”和“课程号”是选修关系中的主属性，按照实体完整性规则，它们均不能取空值。所以选修关系中的“学号”和“课程号”属性实际上只能取相应被参照关系中已经存在的主码值。

BACK

三、用户定义完整性（User-defined integrity）

实体完整性和参照性适用于任何关系数据库系统。除此之外，不同的关系数据库系统根据其应用环境的不同，往往还需要能制定一些特殊的约束条件。用户定义的完整性就是针对某一具体应用的关系数据库所制定约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求。关系模型应提供定义和检验这类完整性的机制，以便用统一的系统的方法处理它们，而不要由应用程序承担这一功能。

BACK

关系完整性约束示意图

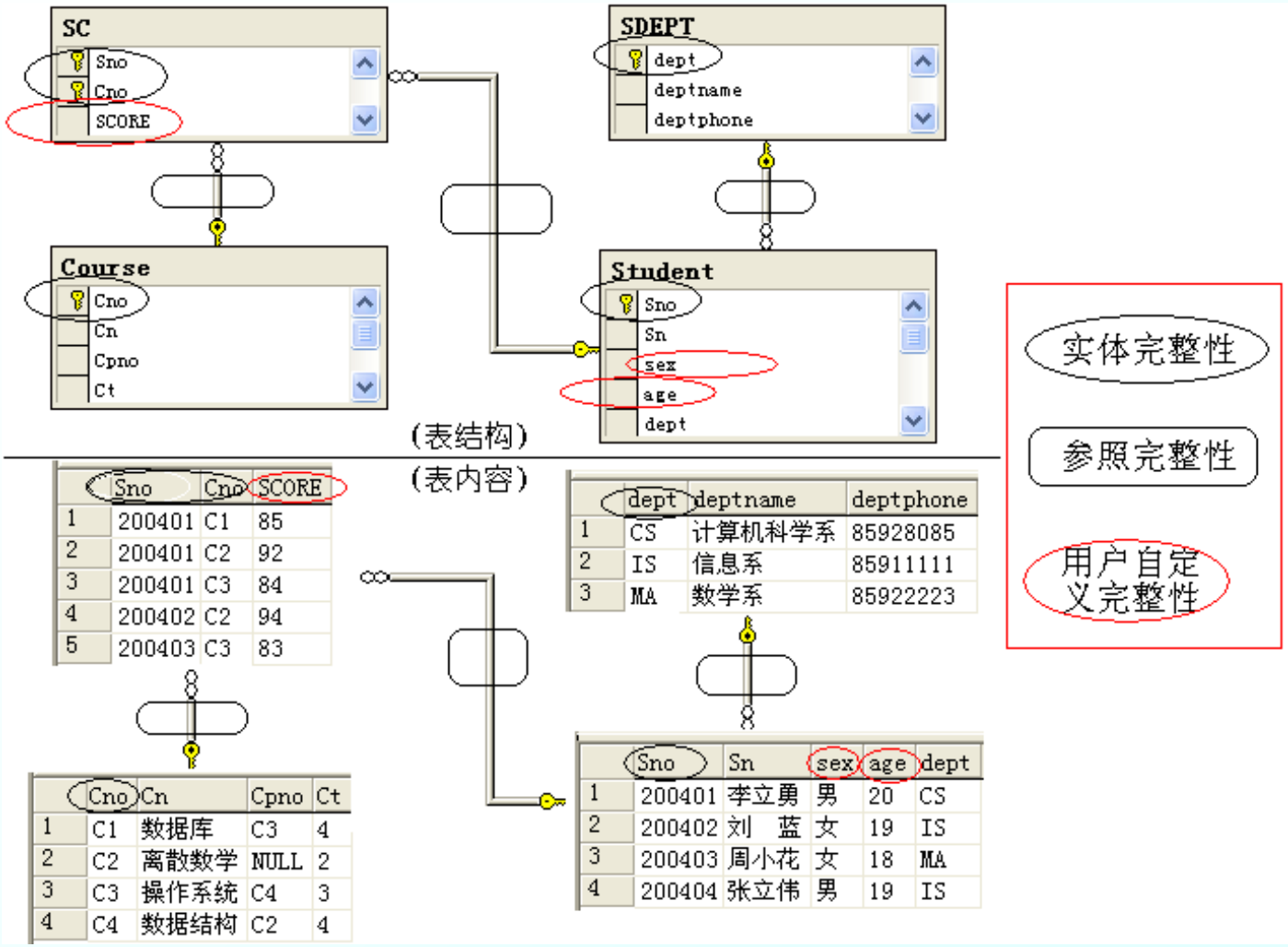


图2.2 关系完整性约束示意图

BACK

2.4 关系代数

关系代数是一种抽象的查询语言，用对关系的运算来表达关系操作，关系代数是研究关系数据操作语言的一种较好的数学工具。

- 集合运算
- 关系运算

返回本章首页

2.4 关系代数

关系代数以一个或两个关系为输入（或称为操作对象），产生一个新的关系作为其操作结果。即其运算对象是关系，运算结果亦为关系。关系代数用到的运算符包括四类：集合运算符、专门的关系运算符、算术比较符和逻辑运算符，如表2.6所示，各运算操作示意图如图2.3所示。

表2.6关系代数的运算符

运算符		含义	运算符		含义
集合运算符	∪ ∩ -	并 交 差	比较运算符	＞ ≥ ＜ ≤ ＝ ≠	大于 大于等于 小于 小于等于 等于 不等于
专门的关系运算符	× σ ∏ ÷ ∞	广义笛卡尔积 选取 投影 除 连接	逻辑运算符	∧ ∨ ¬	与 或 非

返回本章首页

2.4 关系代数

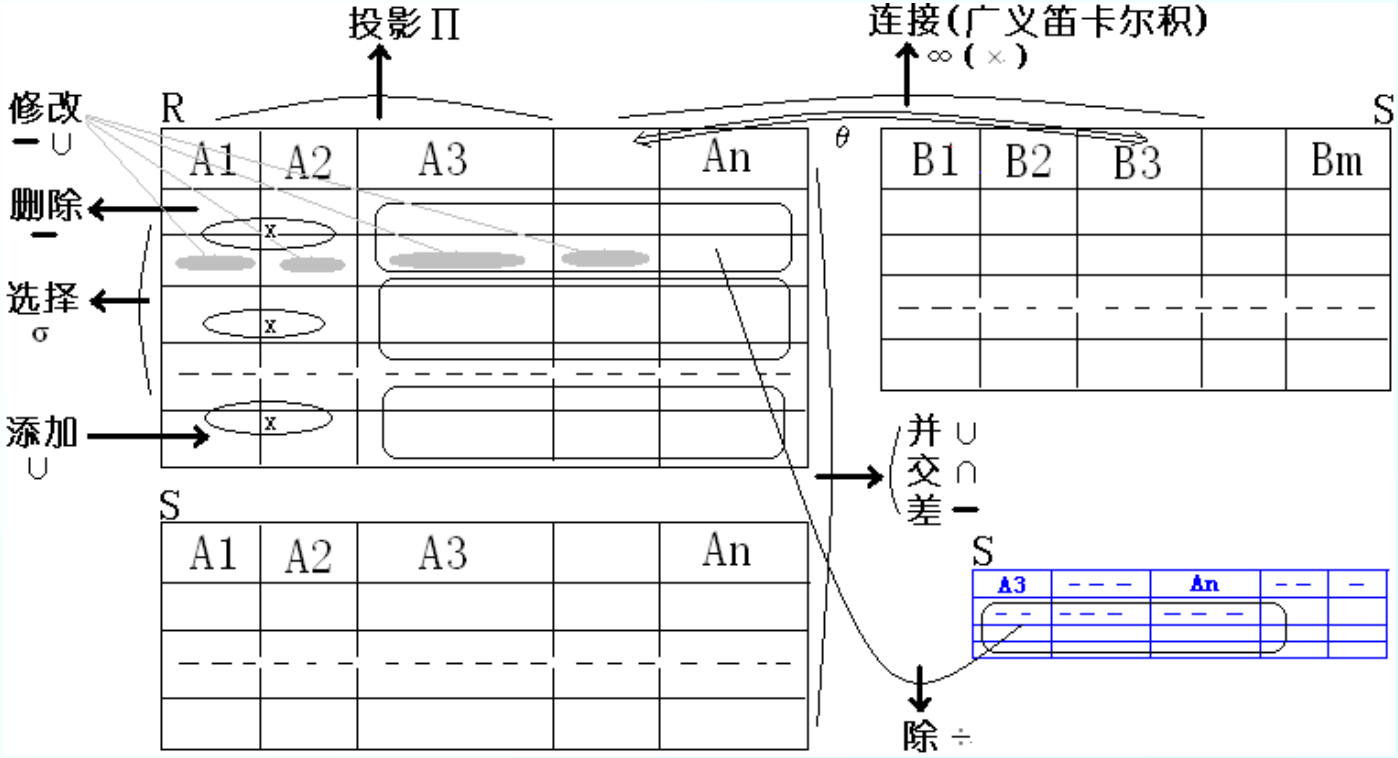


图2.3 关系代数运算操作示意图

返回本章首页

2.4.1 传统的集合运算

传统的集合运算是二目运算，包括并、交、差、广义笛卡尔积四种运算，其关系操作示意如图2.4所示。

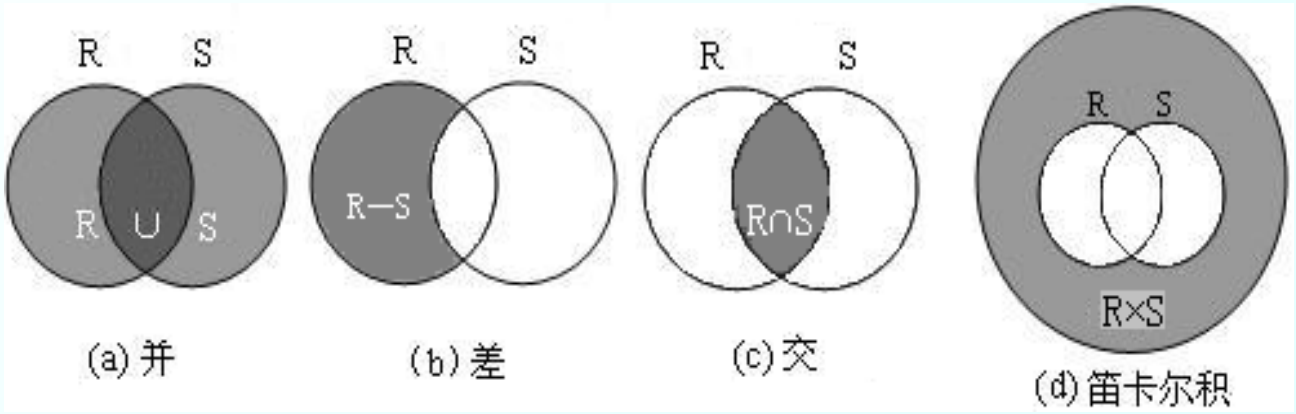


图2.4 传统集合运算关系操作示意图

BACK

2.4.1 传统的集合运算

1. 并（Union）

设关系R和关系S具有相同的目n（即两个关系都有n个属性），且相应的属性取自同一个域，则关系R与关系S的并由属于R或属于S的所有元组组成。记作：$R \cup S=\{t \mid t \in R \vee t \in S\}$，其结果关系仍为n目关系，由属于R或属于S的元组组成。

关系的并操作对应于关系的插入或添加记录的操作，俗称“+”操作，是关系代数的基本操作。

R

X	Y	Z
x_1	y_1	z_1
x_1	y_3	z_2
x_2	y_3	z_3

S

X	Y	Z
x_1	y_1	z_2
x_1	y_3	z_2
x_2	y_3	z_3

RUS

X	Y	Z
x_1	y_1	z_1
x_1	y_3	z_2
x_1	y_1	z_2
x_2	y_3	z_3

BACK

2. 差（Difference）

设关系R和关系S具有相同的目n，且相应的属性取自同一个域，则关系R与关系S的差由属于R而不属于S的所有元组组成。记作：

$R-S=\{t \mid t \in R \wedge t \notin S\}$ 其结果关系仍为n目关系，由属于R而不属于S的所有元组组成。

关系的差操作对应于关系的删除记录的操作，俗称“-”操作，是关系代数的基本操作。

R

X	Y	Z
x_1	y_1	z_1
x_1	y_3	z_2
x_2	y_3	z_3

S

X	Y	Z
x_1	y_1	z_2
x_1	y_3	z_2
x_2	y_3	z_3

R-S

X	Y	Z
x_1	y_1	z_1

BACK

3. 交（Intersection）

设关系R和关系S具有相同的目n，且相应的属性取自同一个域，则关系R与关系S的交由既属于R又属于S的所有元组组成。记作：$R \cap S=\{t \mid t \in R \wedge t \in S\}$ 其结果关系仍为n目关系，由既属于R又属于S的元组组成。关系的交可以用差来表示，即$R \cap S=R-(R-S)$。

关系的交操作对应于寻找两关系共有记录的操作，是一种关系查询操作。关系的交操作能用差操作来代替，为此不是关系代数的基本操作。

R

X	Y	Z
x_1	y_1	z_1
x_1	y_3	z_2
x_2	y_3	z_3

S

X	Y	Z
x_1	y_1	z_2
x_1	y_3	z_2
x_2	y_3	z_3

$R \cap S$

X	Y	Z
x_1	y_3	z_2
x_2	y_3	z_3

BACK

4. 广义笛卡尔积（Extended Cartesian Product）

两个分别为n目和m目的关系R和S的广义笛卡尔积是一个(n+m)列的元组的集合。元组的前n列是关系R的一个元组，后m列是关系S的一个元组。若R有k1个元组，S有k2个元组，则关系R和关系S的广义笛卡尔积有k1×k2个元组。记作：$R \times S=\{t_r \frown t_s \mid t \in R \wedge t \in S\}$

关系的广义笛卡尔积操作对应于两个关系记录横向合并的操作，俗称“×”操作，是关系代数的基本操作，关系的广义笛卡尔积是多个关系相关联操作的最基本操作。

BACK

R

X	Y	Z
x_1	y_1	z_1
x_1	y_3	z_2
x_2	y_3	z_3

S

X	Y	Z
x_1	y_1	z_2
x_1	y_3	z_2
x_2	y_3	z_3

R×S

X	Y	Z	X	Y	Z
x_1	y_1	z_1	x_1	y_1	z_2
x_1	y_1	z_1	x_1	y_3	z_2
x_1	y_1	z_1	x_2	y_3	z_3
x_1	y_3	z_2	x_1	y_1	z_2
x_1	y_3	z_2	x_1	y_3	z_2
x_1	y_3	z_2	x_2	y_3	z_3
x_2	y_3	z_3	x_1	y_1	z_2
x_2	y_3	z_3	x_1	y_3	z_2
x_2	y_3	z_3	x_2	y_3	z_3

BACK

2.4.2 专门的关系运算

基本概念：

1） 分量：设关系模式为R(A_1，A_2，…，A_n)。它的一个关系设为R。t∈R表示t是R的一个元组。t[Ai]则表示元组t中相应于属性Ai的一个分量。

2） 属性列或域列：若A={A_{i1}，A_{i2}，…，A_{ik}}，其中A_{i1}，A_{i2}，…，A_{ik}是A_1，A_2，…，A_n中的一部分，则A称为属性列或域列。t[A]=(t[A_{i1}]，t[A_{i2}]，…，t[A_{ik}])表示元组t在属性列A上诸分量的集合。$\bar{A}$则表示{A_1，A_2，…，A_n}中去掉{A_{i1}，A_{i2}，…，A_{ik} }后剩余的属性组。

3） 元组的连接：R为n目关系，S为m目关系。t∈R，t∈S，$t_r \frown t_s$称为元组的连接（Concatenation）。它是一个(n+m)列的元组，前n个分量为R中的一个n元组，后m个分量为S中的一个m元组。

BACK

2.4.2 专门的关系运算

分量、属性列和元组连接示意如图2.6所示。

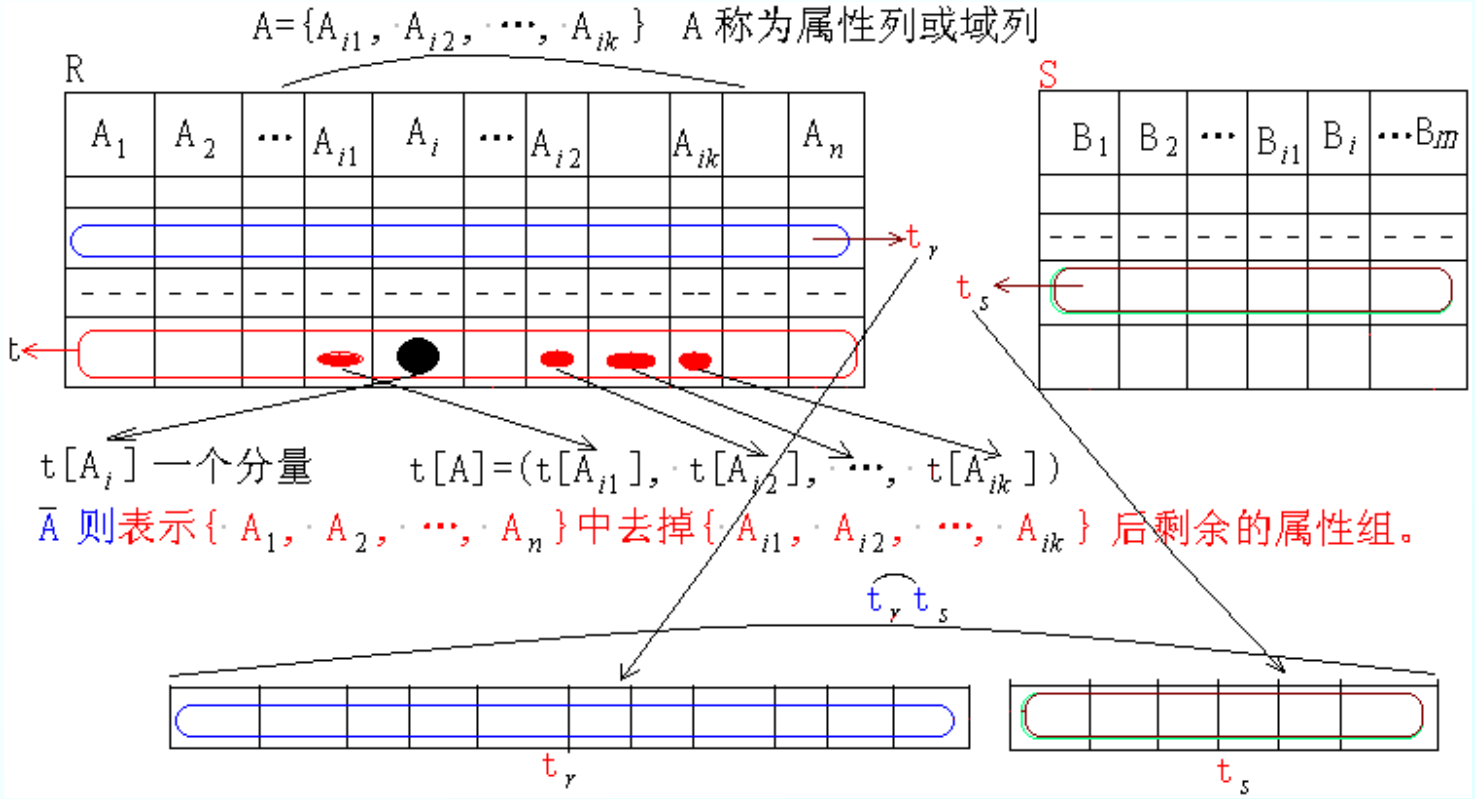


图2.6 分量、属性列和元组连接示意图

2.4.2 专门的关系运算

基本概念：

4）象集：给定一个关系R(X，Z)，X和Z为属性组。我们定义，当t[X]=x时，x在R中的象集（Images Set）为：

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

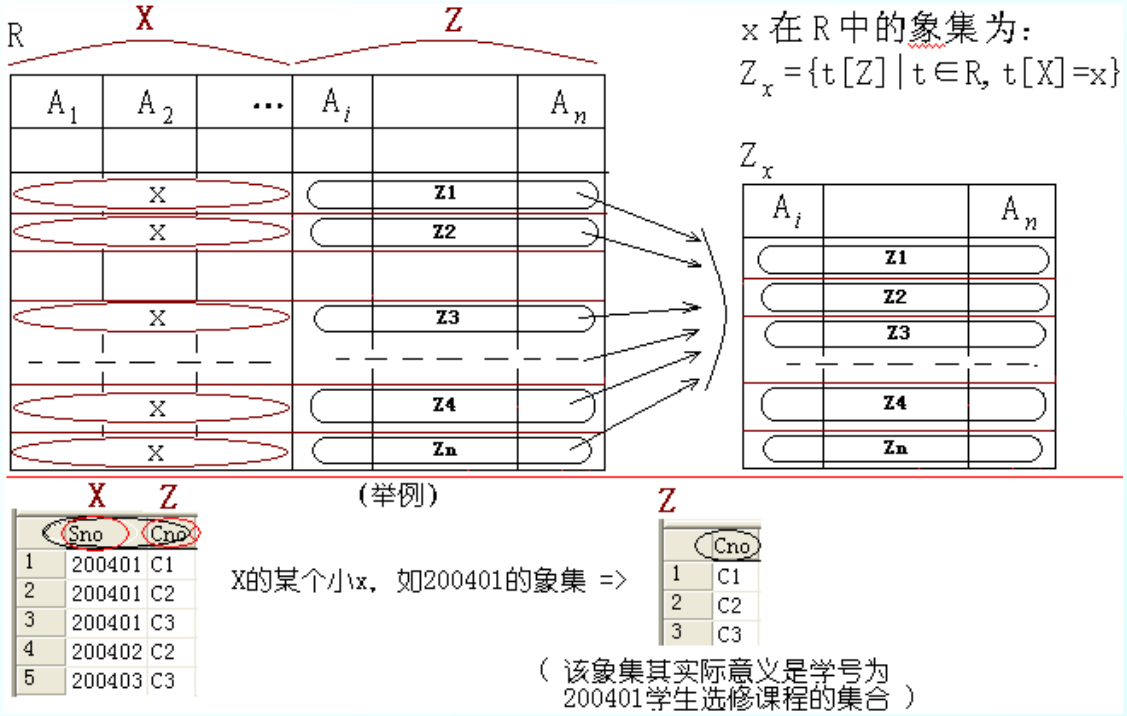


图2.7 象集示意图及举例说明

BACK

例如，参见图2.8 学生-课程关系数据库中的选课关系SC中，设X={SNO}，Z={CNO，SCORE}，令X的一个取值’200401’为小x，则Z_x= {CNO，SCORE}$_{sno}$

= {CNO，SCORE}$_{'200401'}$

={t[CNO，SCORE]|t∈SC,t[SNO]=’200401’}

={(’C1’,85)，(’C2’,92)，(’C3’,84)}

实际上对关系SC来说，某学号（代表某小x）学生的象集即是该学生所有选课课程号与成绩组合的集合。

BACK

2.4.2 专门的关系运算

在给出专门的关系运算的定义前，请先预览各操作的示意图（图2.8）：

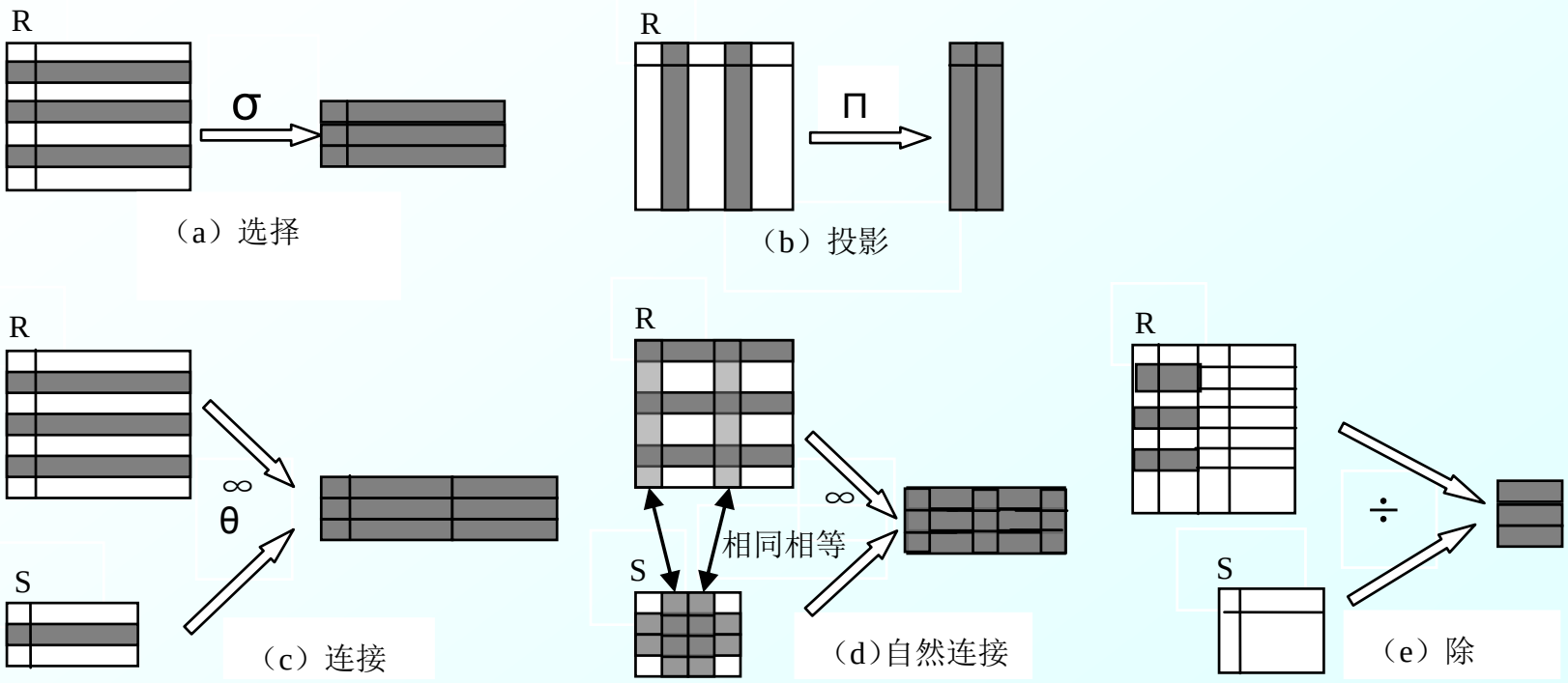


图2.8 专门的关系运算操作示意图

1. 选择（Selection）

定义：选择又称为限制（Restriction）。它是在关系R中选择满足给定条件的诸元组，记作：$\sigma_F(R) = \{t \mid t \in R \wedge F(t)=\text{“真”}\}$其中F表示选择条件，它是一个逻辑表达式，取逻辑值“真”或“假”。逻辑表达式F的基本形式为：X θ Y[φ X θ Y...]

其中θ表示比较运算符，它可以是＞、≥、＜、≤、＝或≠。X1、Y1等是属性名或常量或简单函数。属性名也可以用它的序号来代替（如1，2，...）。φ表示逻辑运算符，它可以是¬、∧或∨。[ ]表示任选项，即[ ]中的部分可以要也可以不要，...表示上述格式可以重复下去。

因此选择运算实际上是从关系R中选取使逻辑表达式F为真的元组。这是从行的角度进行的运算。关系的选择操作对应于关系记录的选取操作（横向选择），是关系查询操作的重要成员之一，是关系代数的基本操作。

例如：有一个学生-课程关系数据库如图2.9包括学生关系S(说明：CS表示计算机系、IS表示信息系、MA表示数学系)、课程关系Course和选修关系SC。

S

学号 SNO	姓名 SN	性别 SEX	年龄 AGE	系别 DEPT
200401	李立勇	男	20	CS
200402	刘　蓝	女	19	IS
200403	周小花	女	18	MA
200405	张立伟	男	19	IS

创建命令

BACK

C

课程号 CNO	课程名 CN	先行课 CPNO	学分 CT
C1	数据库	C2	4
C2	离散数学		2
C3	操作系统	C4	3
C4	数据结构	C2	4

SC

学号 SNO	课程号 CNO	成绩 SCORE
200401	C1	85
200401	C2	92
200401	C3	84
200402	C2	94
200403	C3	83

图2.9 学生-课程关系数据库

[例2.3] 查询计算机科学系（CS系）全体学生。

$\sigma_{DEPT='CS'}(S)$ 或 $\sigma_{5='CS'}(S)$

[例2.4] 查询年龄大于19岁的学生。

$\sigma_{AGE>19}(S)$

BACK

2. 投影（Projection）

定义：关系R上的投影是从R中选择出若干属性列组成新的关系。记作：$\Pi_A(R) = \{ t[A] \mid t \in R \}$。其中A为R中的属性列。

[例2.5] 查询选修关系SC在学号和课程号两个属性上的投影。

$\Pi_{SNO,CNO}(SC)$ 或 $\Pi_{1,2}(SC)$

[例2.6] 查询学生关系S中都有哪些系，即是学生关系S在所系属性上的投影操作。

$\Pi_{DEPT}(S)$

BACK

3. 连接（Join）

1）定义：连接也称为θ连接。它是从两个关系的笛卡尔积中选取属性间满足一定条件的元组。

记作：$R \underset{A\theta B}{\infty} S = \{ t_r \frown t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A]\ \theta\ t_s[B] \}$

其中A和B分别为R和S上度数相等且可比的属性组。θ是比较运算符。连接运算从R和S的笛卡尔积R×S中选取（R关系）在A属性组上的值与（S关系）在B属性组上值满足比较关系θ的元组。

为此：$R \underset{A\theta B}{\infty} S = \sigma_{A\theta B}(R \times S)$

2） 分类：等值连接（equijoin）和自然连接（Natural join）。

θ为”＝”的连接运算称为等值连接。它是从关系R与S的笛卡尔积中选取A、B属性值相等的那些元组。等值连接表示为：

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

为此：$R \underset{A\theta B}{\infty} S = \sigma_{A=B}(R \times S))$

BACK

θ为”＝”的连接运算称为等值连接。它是从关系R与S的笛卡尔积中选取A、B属性值相等的那些元组。等值连接表示为：

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

为此：$R \underset{A\theta B}{\infty} S = \sigma_{A=B}(R \times S))$

自然连接（Natural join）是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组，并且要在结果中把重复的属性去掉。即若R和S具有相同的属性组B，则自然连接可记作：

$$R \infty S = \{ t_r \frown t_s[\bar{B}] \mid t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B] \}$$

为此：$R \infty S = \Pi_{\bar{B}}(\sigma_{R.B=S.B}(R \times S))$

BACK

[例5] 设图2.10(a)和图2.10(b)分别为关系R和关系S，图2.10(c)为$R \underset{C<E}{\infty} S$的结果，图2.10(d)为等值连接$R \underset{R.B=S.B}{\infty} S$的结果，图2.10(e)为自然连接$R \infty S$的结果。

图2.10 连接运算举例

R

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

(a)

S

B	E
b1	3
b2	7
b3	10
b3	2
b5	2

(b)

BACK

$R \infty S$
$C<E$

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

(c)

$R \infty S$
R.B=S.B

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

(d)

BACK

R∞S

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

(e)

BACK

4. 除（Division）

定义：给定关系R(X，Y)和S(Y，Z)，其中X，Y，Z为属性组。R中的Y与S中的Y可以有不同的属性名，但必须出自相同的域。R与S的除运算得到一个新的关系P(X)，P是R中满足下列条件的元组在X属性列上的投影：元组在X上分量值x的象集Y_x包含S在Y上投影的集合。记作：

$R \div S = \{t_r[X] \mid t_r \in R \wedge Y_x \supseteq \Pi_Y(S)\}$，其中$Y_x$为x在R中的象集，$x=t_r[X]$。

关系的除操作，也是一种由关系代数基本操作复合而成的查询操作，显然它不是关系代数的基本操作。关系的除操作能用其它基本操作表示为：

$$R \div S = \Pi_X(R) - \Pi_X(\Pi_X(R) \times \Pi_Y(S) - R)$$

说明：除操作是同时从行和列角度进行运算。除操作适合于包含“对于所有的/全部的”语句的查询操作。

BACK

[例6]：

R

A	B	C
a1	b1	c2
a2	b3	c5
a3	b4	c4
a1	b2	c3
a4	b6	c4
a2	b2	c3
a1	b2	c1

S

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

R÷ S

A
a1

图2.11 除运算举例

在关系R中，A可以取四个值{a1, a2, a3, a4}。

a1的象集为{(b1,c2), (b2,c3), (b2,c1)}

a2的象集为{(b3,c5), (b2,c3)}

a3的象集为{(b4,c4)}

a4的象集为{(b6,c4)}

S在(B,C)上的投影为{(b1,c2), (b2,c3), (b2,c1)}

显然只有a1的象集（B,C）a1包含S在（B,C）属性组上的投影，所以R÷S={a1}

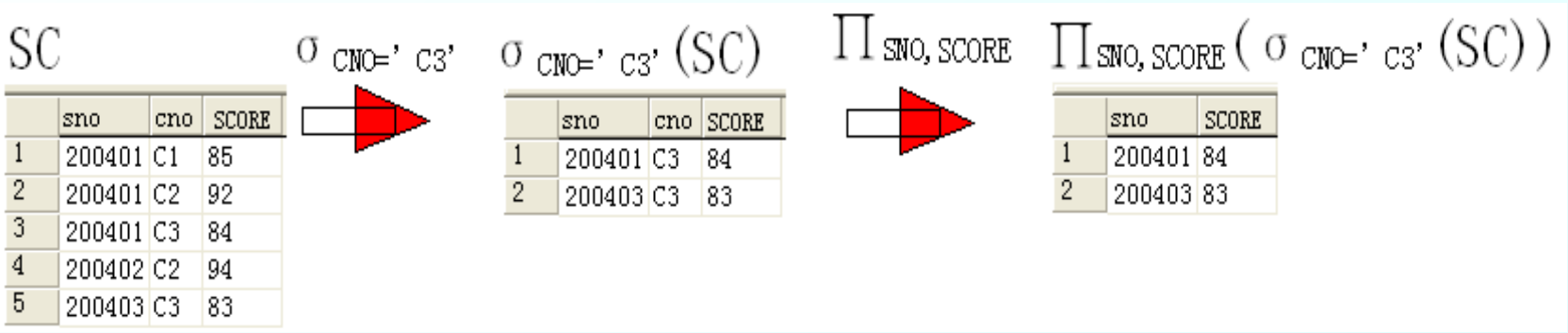
BACK

5．关系代数操作表达举例

设教学数据库中有三个关系，学生关系：S(SNO，SN，AGE，SEX)、学习关系：SC(SNO，CNO，SCORE)、课程关系：C(CNO，CN，TEACHER)

（1）检索学习课程号为C3的学生学号和成绩

$\Pi_{SNO,SCORE}(\sigma_{CNO='C3'}(SC))$



	sno	cno	SCORE
1	200401	C1	85
2	200401	C2	92
3	200401	C3	84
4	200402	C2	94
5	200403	C3	83

	sno	cno	SCORE
1	200401	C3	84
2	200403	C3	83

	sno	SCORE
1	200401	84
2	200403	83

图2.12 例2.9(1)关系代数表达式运算过程

BACK

5．关系代数操作表达举例

设教学数据库中有三个关系，学生关系：S(SNO，SN，AGE，SEX)、学习关系：SC（SNO，CNO，SCORE)、课程关系：C（CNO，CN，TEACHER)

（2）检索学习课程号为C3的学生学号和姓名

$\Pi_{SNO,SN}(\sigma_{CNO='C3'}(S\infty SC))$

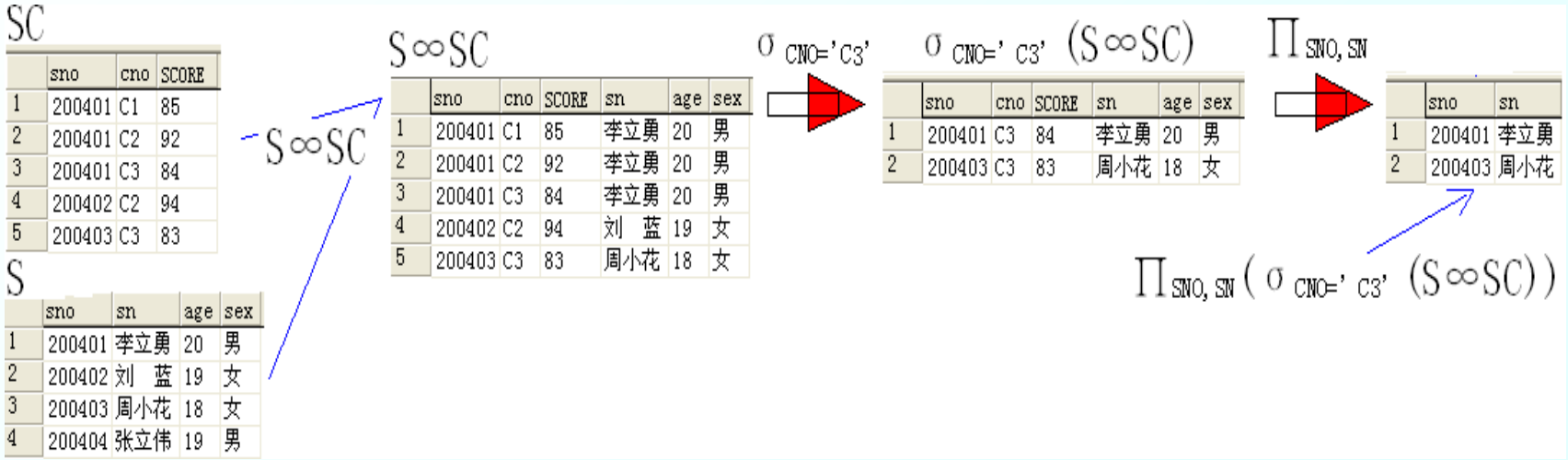


图2.13 例2.9(2)关系代数表达式运算过程

5．关系代数操作表达举例

设教学数据库中有三个关系，学生关系：S(SNO，SN，AGE，SEX)、学习关系：SC (SNO，CNO，SCORE)、课程关系：C (CNO，CN，TEACHER)

（1）检索学习课程号为C3的学生学号和成绩

$\Pi_{SNO,SCORE}(\sigma_{CNO='C3'}(SC))$

（2）检索学习课程号为C3的学生学号和姓名

$\Pi_{SNO,SN}(\sigma_{CNO='C3'}(S\infty SC))$

（3）检索学习课程名为MATHS的学生学号和姓名

$\Pi_{SNO,SN}(\sigma_{CN='MATHS'}(S\infty SC\infty C))$

（4）检索学习课程号为C1或C3的学生学号

$\Pi_{SNO}(\sigma_{CNO='C1'V'CNO='C3'}(SC))$

（5）检索不学习课程号为C2的学生的姓名和年龄

$\Pi_{SN,AGE}(S)-\Pi_{SN,AGE}(\sigma_{CNO='C2'}(S\infty SC))$

BACK

5．关系代数操作表达举例

设教学数据库中有三个关系，学生关系：S(SNO，SN，AGE，SEX)、学习关系：SC (SNO，CNO，SCORE)、课程关系：C (CNO，CN，TEACHER)

（6）检索学习全部课程的学生姓名

$\Pi_{SN}(S \infty (\Pi_{SNO,CNO}(SC) \div \Pi_{CNO}(C)))$

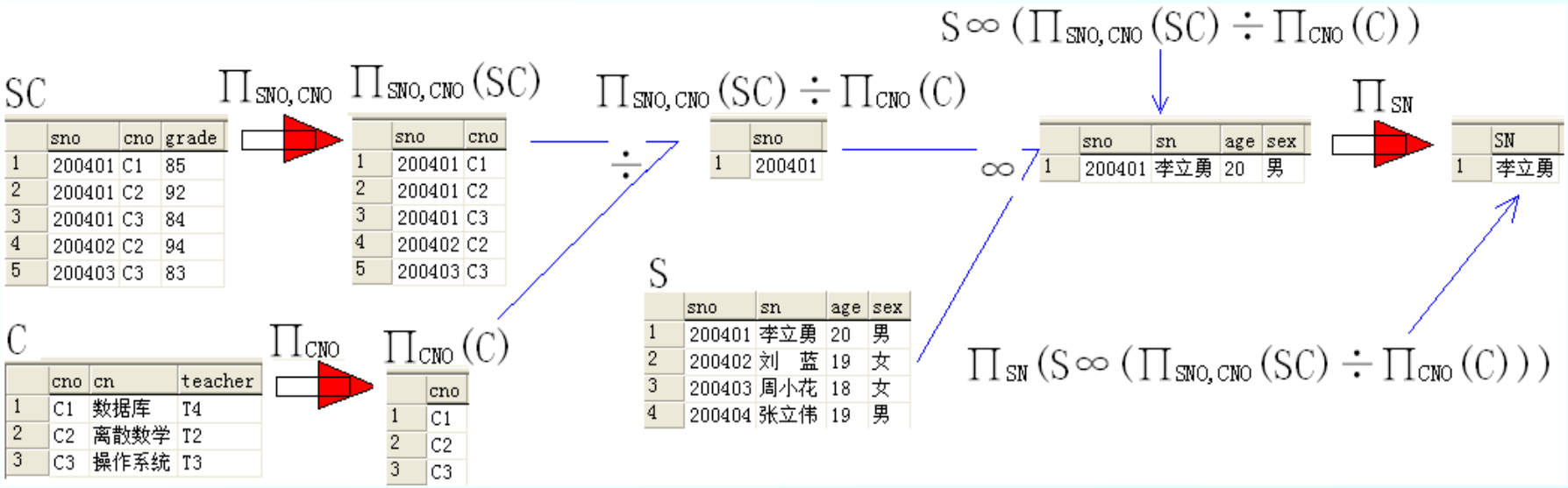


图2.14 例2.9(6)关系代数表达式运算过程

（7）检索所学课程包括200401所学课程的学生学号

$\Pi_{SNO,CNO}(SC) \div \Pi_{CNO}(\sigma_{SNO='200401'}(SC))$

BACK

2.5 关系演算

关系演算是以数理逻辑中的谓词演算为基础的。按谓词变元的不同，关系演算可分为元组关系演算和域关系演算。

2.5.1 抽象的元组关系演算*

关系R可用谓词R(t)表示，其中t为变元。关系R与谓词R(t)间的关系如下：

$$R(t)=\begin{cases}\text{True}\ (\text{当}t\text{在}R\text{内}) \\ \text{False}\ (\text{当}t\text{不在}R\text{内})\end{cases}$$

为此，关系R = {t | R(t) }，其中t是变元或变量。

一般地可令R = {t | Φ(t) }，t是变元或变量。当谓词Φ(t)（Φ读作“fai”）以元组（与表中的行对应）为度量时，称为元组关系演算（Tuple Relational Calculus）；当谓词以域（与表中的列对应）为变量时，称为域关系演算（Domain Relational Calculus）（抽象的域关系演算类似于抽象的元组关系演算，不再多叙述）。

返回本章首页

2.5.1 抽象的元组关系演算*

在元组关系演算中，把{t | Φ(t)}称为一个元组关系演算表达式，把Φ(t)称为一个公式，t为Φ中唯一的自由元组变量。

如下递归地定义元组关系演算公式。

（1）原子命题公式是公式，称为原子公式，它有下面3种形式。

①R(t)。R是关系名，t是元组变量。

②t[i] θ C或C θ t[i]。t[i]表示元组变量t的第i个分量，C是常量，θ为算术比较运算符。

③t[i] θ u[j]。t、u是两个元组变量。

（2）设Φ1、Φ2是公式，则¬φ1、Φ1∧Φ2、Φ1∨Φ2、Φ1→Φ2也都是公式。

（3）设Φ是公式，t是Φ中的某个元组变量，那么(∀t)(Φ)、(∃t)(Φ)都是公式。

∀为全称量词，含义是“对所有的…”；∃为存在量词，含义是“至少有一个…”。受量词约束的变量称为约束变量，不受量词约束的变量称为自由变量。

（4）在元组演算的公式中，各种运算符的运算优先次序为：

①算术比较运算符最高；②量词次之，且按∃∃、∀ 的先后次序进行；③逻辑运算符优先级最低，且按¬、∧、∨、→的先后次序进行；④括号中的运算优先。

（5）元组演算的所有公式按（1）、（2）、（3）、（4）所确定的规则经有限次复合求得，不再存在其他形式。

返回本章首页

2.5.1 抽象的元组关系演算*

为了证明元组关系演算的完备性，只要证明关系代数的5种基本运算均可等价地用元组演算表达式表示即可，所谓等价是指等价双方运算表达式的结果关系相同。

设R、S为两个关系，它们的谓词分别为：R(t)、S(t)，则：

（1）R ∪S 可等价地表示为{t｜R(t)∨S (t)}

（2）R －S 等价于 {t｜R(t)∧S(t)}

（3）R ×S等价于 {t ｜(∃*u*)(∃ *v*)(R (u)∧S (v)∧t [1]= u [1]∧…∧t [k1]= u [k1]∧t[k1+1]= v[1]∧…∧t[k1+k2]= v[k2])}。式中，R、S依次为k1、k2元关系，u、v表示R、S的元组变量。

（4）πi1,i2,…, in(R)等价于{t｜(∃u)(R(u)∧t[1]=u[i1]...∧t[n]= u[in]}

（5）σF(R)等价于{t｜R(t)∧F'}

其中F'为F在谓词演算中的表示形式，即用t[i]代替F中t的第i个分量即为F'。

返回本章首页

2.5.1 抽象的元组关系演算*

关系代数的5种基本运算可等价地用元组关系演算表达式表示。因此，元组关系演算体系是完备的,是能够实现关系代数所能表达的所有操作的，是能用来表示对关系的各种操作的。

如此，元组关系演算对关系的操作，就转化为求出这样的满足操作要求的φ(t)谓词公式了。如下基于元组关系演算语言的ALPHA的操作表达中就蕴含着这样的φ(t)谓词公式，请能体会之。

在关系演算公式表达时，还经常要用到如下3类等价的转换规则：

（1）$\varphi 1 \wedge \varphi 2 \equiv \neg\neg(\varphi 1 \wedge \varphi 2) \equiv \neg(\neg\varphi 1 \vee \neg\varphi 2)$；

$\varphi 1 \vee \varphi 2 \equiv \neg\neg(\varphi 1 \vee \varphi 2) \equiv \neg(\neg\varphi 1 \wedge \neg\varphi 2)$。

（2）$(\forall t)(\varphi(t)) \equiv \neg(\exists t)(\neg\varphi(t))$；$(\exists t)(\varphi(t)) \equiv \neg(\forall t)(\neg\varphi(t))$。

（3）$\varphi 1 \rightarrow \varphi 2 \equiv (\neg\varphi 1) \vee \varphi 2$

返回本章首页

2.5 关系演算

抽象的关系演算对应的语言有：

- 元组关系演算语言ALPHA
- 域关系演算语言QBE*

返回本章首页

2.5.2　元组关系演算语言ALPHA

元组关系演算是以元组变量作为谓词变元的基本关系演算表达形式。一种典型的元组关系演算语言是ALPHA语言，ALPHA语言主要有GET、PUT、HOLD、UPDATE、DELETE、DROP六条语句，语句的基本格式是：

操作语句　工作空间名（表达式）：操作条件

其中表达式用于指定语句的操作对象，它可以是关系名或属性名，一条语句可以同时操作多个关系或多个属性。操作条件是一个逻辑表达式，用于将操作对象限定在满足条件的元组中，操作条件可以为空。除此之外，还可以在基本格式的基础上加上排序要求，定额要求等。

BACK

检索操作用GET语句实现。学习操作表达前说明几点：

（1）操作表达前，根据查询条件与要查询信息等，同样要先确定本查询涉及到哪几个表？

（2）ALPHA语言的查询操作与关系代数操作表达思路完全不同，表达中要有谓词判定、量词作用的操作表达理念。如下表达举例中部分给出的图示，直观地说明了其操作办法与操作思路。思考时画出相关各关系表能便于直观分析，利用操作表达。

（3）ALPHA语言的查询操作表达也是不唯一的，很值得推敲的。

BACK

一、检索操作

检索操作用GET语句实现。

(1) 简单检索（即不带条件的检索）

[例11] 查询所有被选修课程的课程号码

GET W(SC.CNO)

[例12] 查询所有学生的信息

GET W(S)

(2) 限定的检索（即带条件的检索），由冒号后面的逻辑表达式给出查询条件。

[例13] 查询计算机系(CS)中年龄小于22岁的学生的学号和姓名

GET W(S.SNO,S.SN):S.DEPT='CS'∧ S.AGE<22

BACK

[例13] 查询计算机系(CS)中年龄小于22岁的学生的学号和姓名

GET W(S.SNO, S.SN): S.DEPT='CS' ∧ S.AGE<22

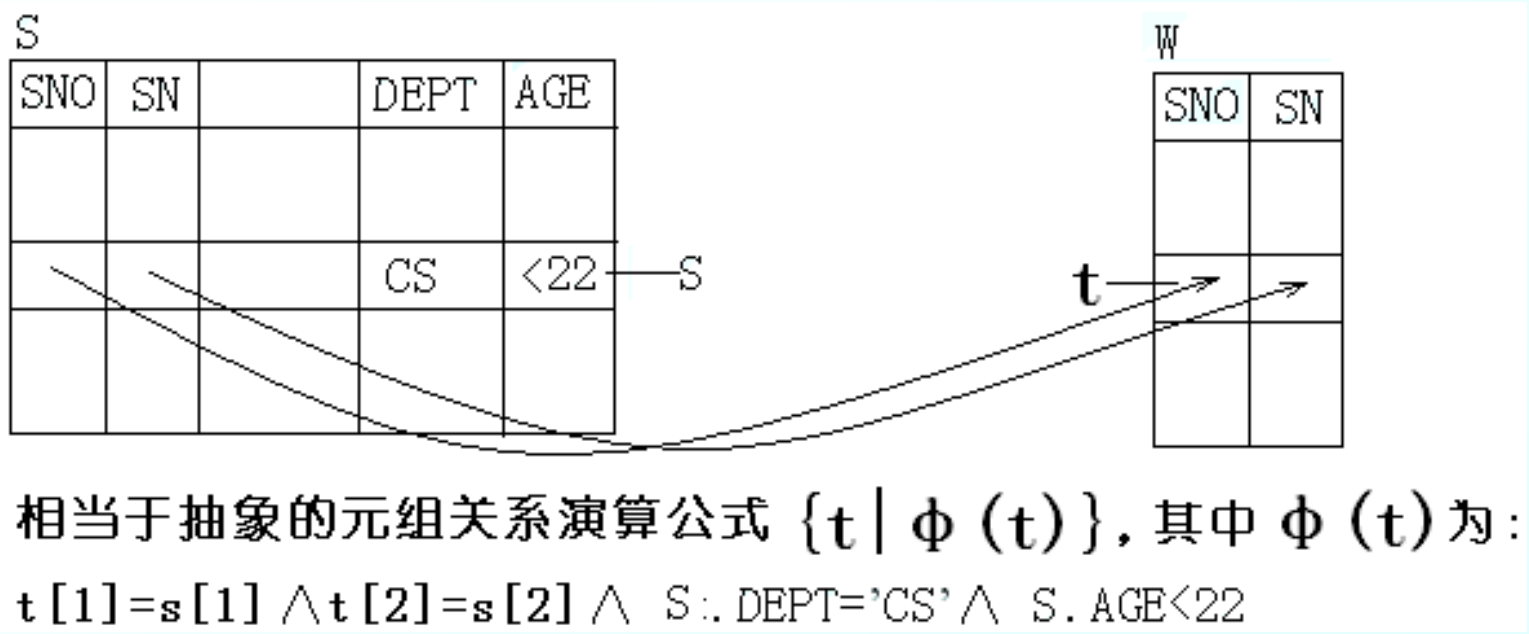


图2.15 例2.13关系演算表达式操作示意图

BACK

(3)带排序的检索

[例14] 查询信息系(IS)学生的学号、年龄，并按年龄降序排序

GET W(S.SNO, S.AGE): S.DEPT='IS' DOWN S.AGE

DOWN代表降序排序，后面紧跟排序的属性名。当升序排列时使用UP。

(4)带定额的检索

[例15] 取出一个信息系学生的姓名

GET W(1)(S.SN): S.DEPT='IS'

所谓带定额的检索是指规定了检索出的元组的个数，方法是在W后的括号中加上定额数量。排序和定额可以一起使用。

[例16] 查询信息系年龄最大的三个学生的学号及其年龄，并按年龄降序排列。

GET W(3)(S.SNO, S.AGE): S.DEPT='IS' DOWN S.AGE

BACK

(5) 用元组变量的检索

因为元组变量是在某一关系范围内变化的，所以元组变量又称为范围变量（Range Variable）。元组变量主要有两方面的用途：

①简化关系名。在处理实际问题时，如果关系的名字很长，使用起来就会感到不方便，这时我们可以设一个较短名字的元组变量来简化关系名。

②操作条件中使用量词时必须用元组变量。

元组变量是动态或逻辑的概念，一个关系可以设多个元组变量，每个元组变量独立地代表该关系中的任一元组。

[例17] 查询信息系学生的名字。

```
RANGE Student X
GET W(X.SN): X.DEPT='IS'
```

这里元组变量X的作用是简化关系名Student。

BACK

(6) 用存在量词的检索

[例18] 查询选修C2号课程的学生名字。

RANGE SC X

GET W(S.SN): ∃X(X.SNO=S.SNO ∧ X.CNO='C2')

[例19] 查询选修了直接先行课为C2号课程的课程的学生学号。

RANGE C CX

GET W(SC.SNO): ∃CX(CX.CNO=SC.CNO ∧ CX.CPNO='C2')

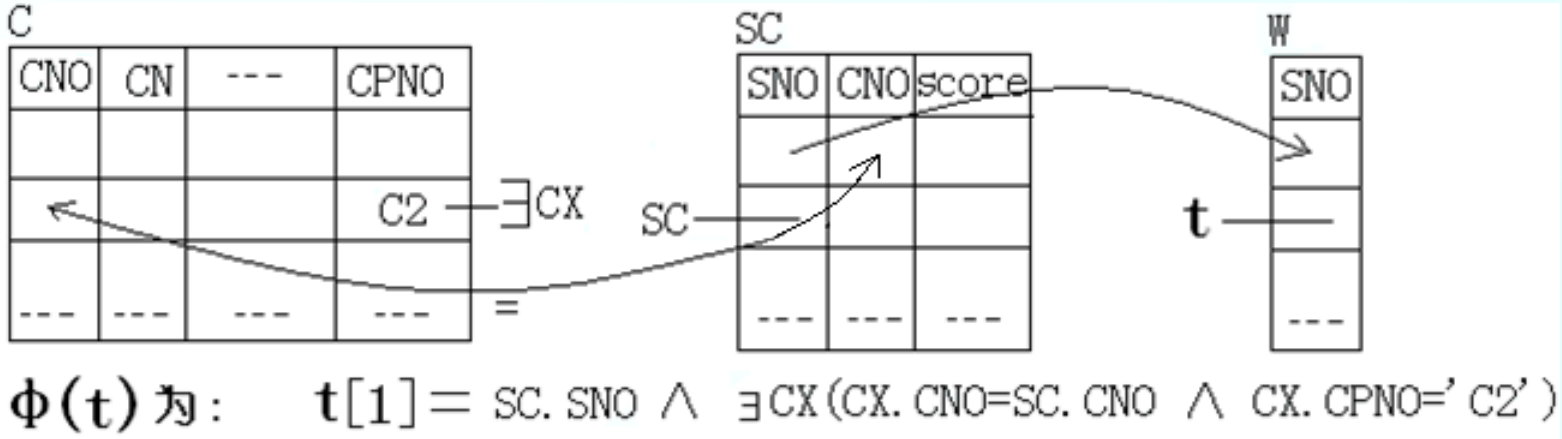


图2.16 例2.19关系演算表达式操作示意图

图2.16示意：从选修表当前记录中取学号，条件是存在一门课CX，其直接先修课为C2，该课程正为该学号学生所选。

BACK

[例20] 查询至少选修一门其先行课为C2号课程的学生名字。

RANGE C CX

SC SCX

GET W(S.SN): ∃SCX(SCX.SNO=S.SNO ∧

∃CX(CX.CNO=SCX.CNO ∧ CX.CPNO='C2'))

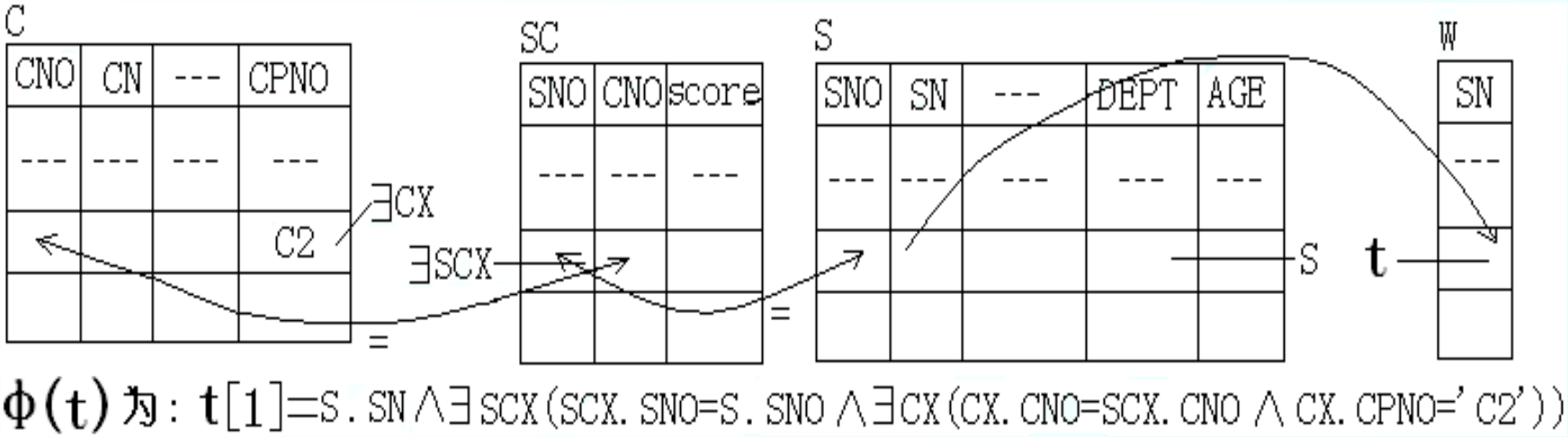


图2.17 例2.20关系演算表达式操作示意图

图2.17示意：从学生关系中当前记录取姓名，条件是该生存在选修关系SCX，还存在某课程CX，其先修课为C2，课程CX正是SCX所含的课程。

本例中的元组关系演算公式可以变换为前束范式（Prenex normal form）的形式：

GET W(S.SN): ∃SCX ∃CX(SCX.SNO=S.SNO∧

CX.CNO=SCX.CNO∧CX.CPNO='C2')

BACK

(7) 带有多个关系的表达式的检索

上面所举的各个例子中，虽然查询时可能会涉及多个关系，即公式中可能涉及多个关系，但查询结果都在一个关系中，即查询结果表达式中只有一个关系。实际上表达式中是可以有多个关系的。

[例21] 查询成绩为90分以上的学生名字与课程名字。

RANGE SC SCX

GET W(S.SNO, C.CN): ∃SCX(SCX.SCORE≥90 ∧ SCX.SNO=S.SNO ∧ C.CNO=SCX.CNO)

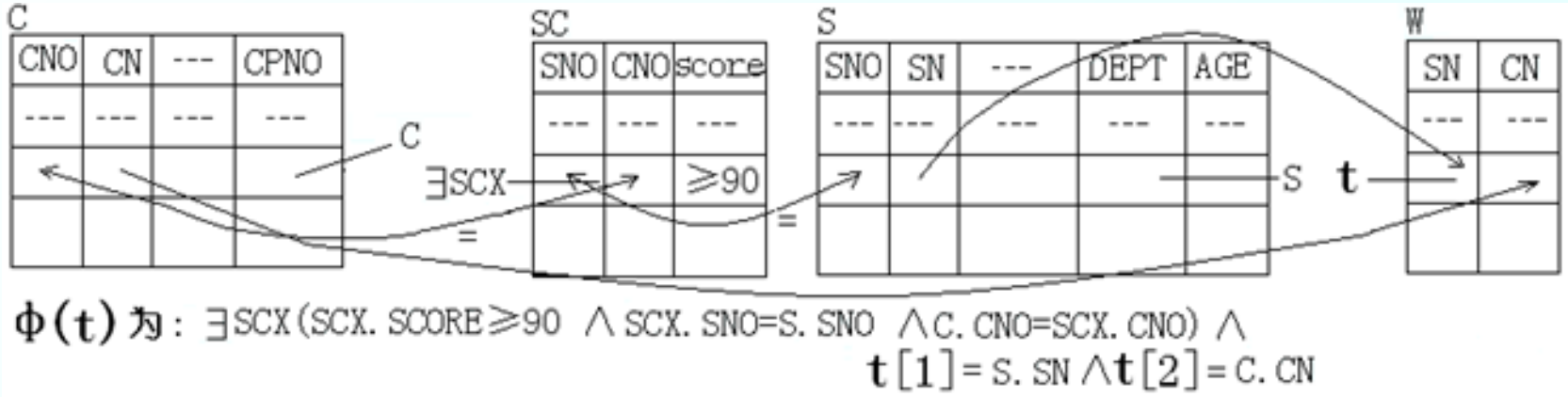


图2.18 例2.21关系演算表达式操作示意图

图2.18示意：分别从学生表S和课程表C的当前记录中取学生姓名和课程名，条件是有选修关系元组SCX存在，SCX是该学生的选修关系，并选修了该课程，成绩为≥90分。

本查询所要求的结果学生名字和课程名字分别在Student和Course两个关系中。

BACK

(8) 用全称量词的检索

[例22] 查询不选C1号课程的学生名字

RANGE SC SCX

GET W(S.SN): $\forall$SCX (SCX.SNO≠S.SNO ∨ SCX.CNO≠'C1')

本例实际上也可以用存在量词来表示：

RANGE SC SCX

GET W(S.SN):¬ $\exists$SCX (SCX.SNO=S.SNO ∧ SCX.CNO='C1')

BACK

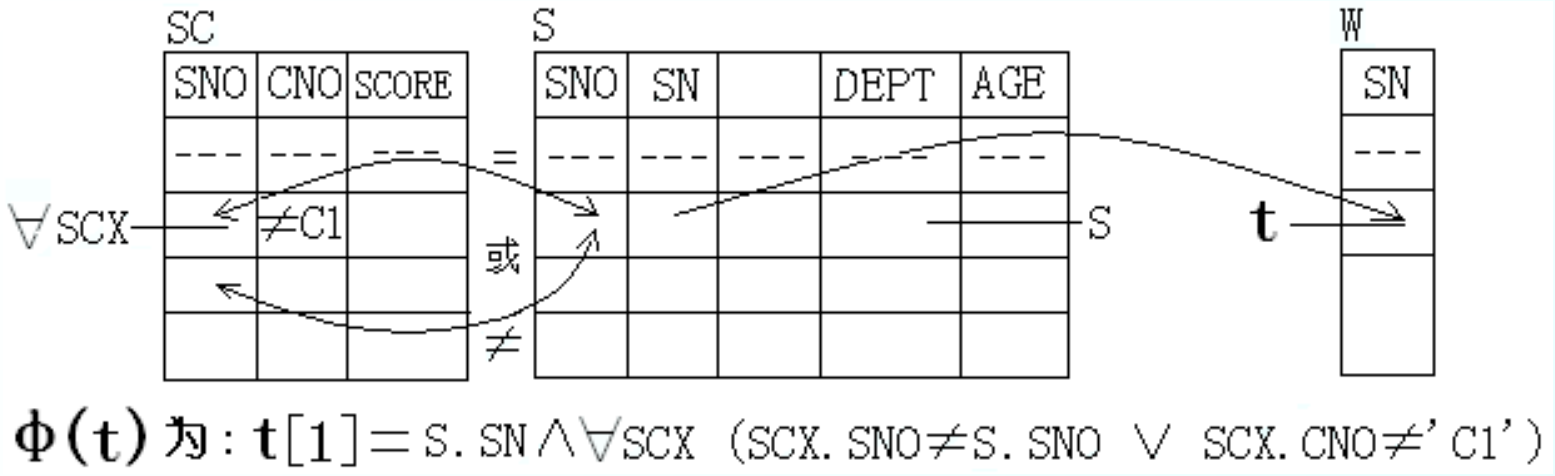


图2.19 例2.22关系演算表达式操作示意图之一

图2.19示意：从学生表S的当前记录中取姓名，条件是对任意的选修元组SCX都满足：该选修元组不是当前被检索学生的选修或是该学生的选修但课程号不是C1。

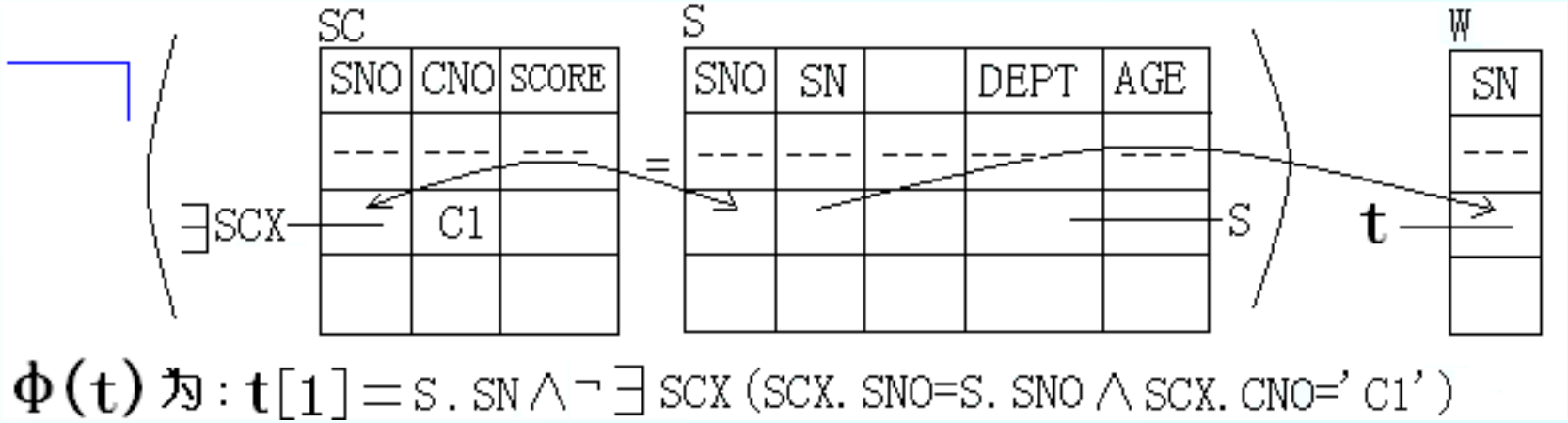


图2.20 例2.22关系演算表达式操作示意图之二

图2.20示意：从学生表S的当前记录中取姓名，条件是该学生不存在对C1课程的选修元组SCX。

BACK

(9) 用两种量词的检索

[例23] 查询选修了全部课程的学生姓名。

RANGE C CX

SC SCX

GET W(S.SN): ∀CX ∃SCX(SCX.SNO=S.SNO∧SCX.CNO=CX.CNO)

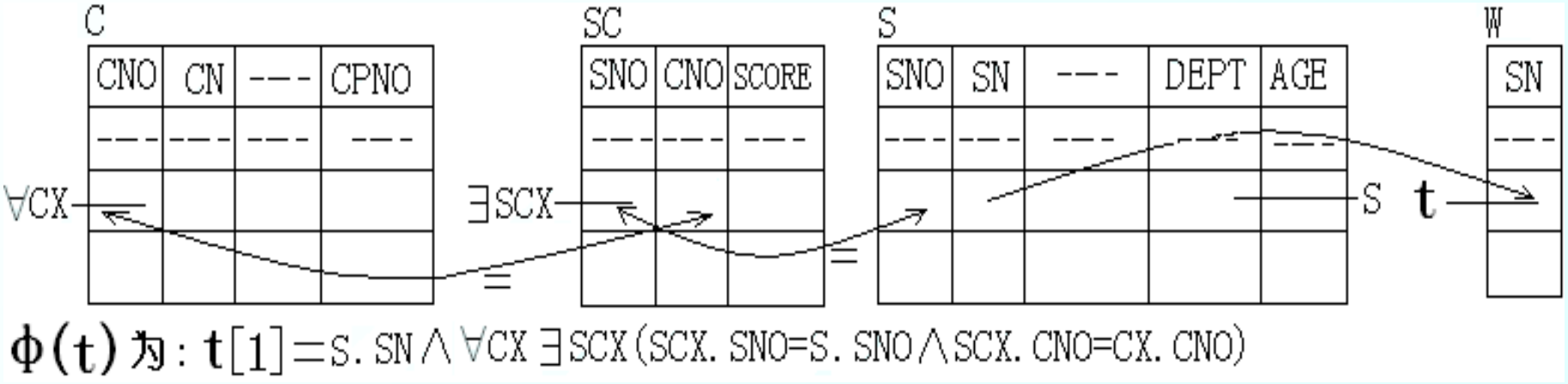


图2.21 例2.23关系演算表达式操作示意图

图2.21示意：从学生表S中取学生姓名SN，条件是对任意的课程CX，该学生都有选课关系SCX存在，并选了CX这门课。

BACK

(10) 用蕴函（Implication）的检索

[例24] 查询最少选修了学号为200402的学生所选课程的学生学号。

本例题的求解思路是，对Course中的所有课程，依次检查每一门课程，看200402学生是否选修了该课程，如果选修了，则再看某一个学生是否也选修了该门课。如果对于200402所选的每门课程该学生都选修了，则该学生为满足要求的学生。把所有这样的学生全都找出来即完成了本题。

```
RANGE C CX
      SC SCX
      SC SCY
GET W(S.SNO): ∀CX( ∃SCX(SCX.SNO='200402' ∧ SCX.CNO=CX.CNO)
→ ∃SCY(SCY.SNO=S.SNO∧SCY.CNO=CX.CNO))
```


BACK

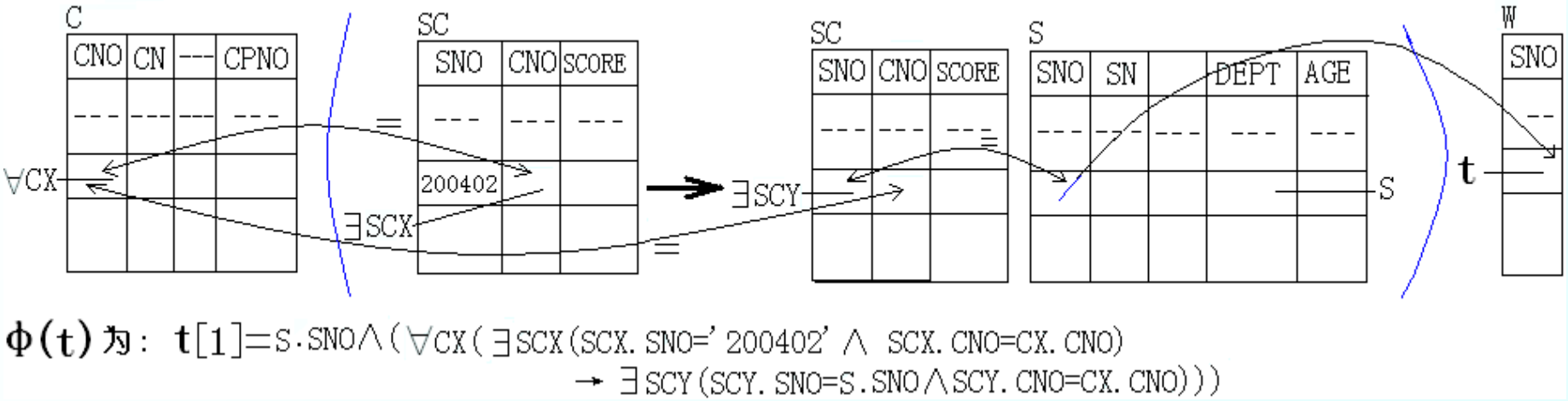


图2.22 例2.24关系演算表达式操作示意图

图2.22示意：从学生表S的当前记录取学号，条件是对任意的课程CX都有：如果存在有200402学生的选修元组SCX，其选修的课程是CX，则当前被检索学生必存在选修元组SCY，也选修了课程CX。

BACK

(11) 集函数

用户在使用查询语言时，经常要作一些简单的计算，例如要求符合某一查询要求的元组数，求某个关系中所有元组在某属性上的值的总和或平均值等。为了方便用户，关系数据语言中建立了有关这类运算的标准函数库供用户选用。这类函数通常称为集函数（Aggregation function）或内部函数（Build-in function）。关系演算中提供了COUNT，TOTAL，MAX，MIN，AVG等集函数，其含义如表2.7所示。

表2.7 关系演算中的集函数

函数名	功能
COUNT	对元组计数
TOTAL	求总和
MAX	求最大值
MIN	求最小值
AVG	求平均值

[例25] 查询学生所在系的数目。

GET W(COUNT(S.DEPT))

COUNT函数在计数时会自动排除重复的DEPT值

[例26] 查询信息系学生的平均年龄。

GET W(AVG(S.AGE)): S.DEPT='IS'

BACK

二、更新操作

1．修改操作

修改操作用UPDATE语句实现。其步骤是：

首先用HOLD语句将要修改的元组从数据库中读到工作空间中；然后用宿主语言修改工作空间中元组的属性；最后用UPDATE语句将修改后的元组送回数据库中。

[例27] 把200407学生从计算机科学系转到信息系。

```
HOLD W(S.SNO, S.DEPT): S.SNO='200407'
  （从S关系中读出200407学生的数据）
MOVE 'IS' TO W.DEPT  （用宿主语言进行修改）
UPDATE W             （把修改后的元组送回S关系）
```

说明： 1）该例中用HOLD语句读200407的数据，不用GET语句。如果修改操作涉及两个关系，就要执行两次HOLD－MOVE－UPDATE操作序列。

2）修改主码的操作是不允许的，即不能用UPDATE语句将学号200401改为200402。如果需要修改关系中某个元组的主码值，只能先删除该元组，然后把具有新主码值的元组插入到关系中。

BACK

2. 插入操作

插入操作用PUT语句实现。其步骤是：首先用宿主语言在工作空间中建立新元组；然后用PUT语句把该元组存入指定的关系中。

[例28] 学校新开设了一门2学分的课程“计算机组织与结构”，其课程号为C8，直接先行课为C4号课程。插入该课程元组。

```
MOVE ’C8’ TO W.CNO
MOVE ’计算机组织与结构’ TO W.CN
MOVE ’C4’ TO W.Cpno
MOVE ’2’ TO W.CT
PUT W(C)          （把W中的元组插入指定关系C中）
```

PUT语句只对一个关系操作，也就是说表达式必须为单个关系名。如果插入操作涉及多个关系，必须执行多次PUT操作。

BACK

3. 删除操作

删除操作用DELETE语句实现。其步骤为：用HOLD语句把要删除的元组从数据库中读到工作空间中；再用DELETE语句删除该元组。

[例29] 200410学生因故退学，删除该学生元组。

```
HOLD W(S): S.SNO='200410'
DELETE W
```

[例30] 将学号200401改为200410。

```
HOLD W(S): S.SNO='200401'
DELETE W
MOVE '200410' TO W.SNO
MOVE '李立勇' TO W.SN
MOVE '男' TO W.SEX
MOVE '20' TO W.AGE
MOVE 'CS' TO W.DEPT
PUT W(S)
```

修改主码的操作，一般要分解为先删除、再插入的方法完成操作

BACK

[例31] 删除全部学生。

HOLD W(S)

DELETE W

由于SC关系与S关系之间具有参照关系，为了保证参照完整性，删除S关系中全部元组的操作可能会遭到拒绝（因为SC关系要参照S关系的）或将导致DBMS自动执行删除SC关系中全部元组的操作，如下：

HOLD W(SC)

DELETE W

一般可先删除SC中的元组，再删除S表中的元组。

BACK

2.5.3 域关系演算语言QBE*

关系演算的另一种方式是域关系演算。域关系演算以元组变量的分量即域变量作为谓词变元的基本关系演算表达形式。1975年由M.M.Zloof提出的QBE就是一个很有特色的域关系演算语言，该语言于1978年在IBM 370上得以实现。QBE也指此关系数据库管理系统。

QBE是Query By Example（即通过例子进行查询）的简称，其最突出的优点是它的操作方式。它是一种高度非过程化的基于屏幕表格的查询语言，用户通过终端屏幕编辑程序以填写表格的方式构造查询要求，而查询结果也是以表格形式显示，因此非常直观，易学易用。

QBE中用示例元素来表示查询结果可能的情况，示例元素实质上就是域变量。QBE操作框架如图2.23所示：

关系名	属性名	属性名	属性名
操作命令	元组属性值或查询条件		

图2.23 QBE操作框架

BACK

一、检索操作

1. 简单查询

[例32] 求信息系全体学生的姓名。(以学生-课程关系数据库为例，来说明QBE的用法。)操作步骤为：

(1) 用户提出要求

(2) 屏幕显示空白表格

(3) 用户在最左边一栏输入关系名S

S					

BACK

(4) 系统显示该关系的属性名；

Student	SNO	SN	SEX	AGE	DEPT

(5) 用户在上面构造查询条件；

Student	SNO	SN	SEX	AGE	DEPT
		P.T			IS

◆ 这里T是示例元素，即域变量。QBE要求示例元素下面要加下划线。IS是查询条件，不用加下划线。P.是操作符，表示打印（Print），实际上是显示。

◆ 查询条件中可以使用比较运算符>，≥，<，≤，＝，≠，=可以省略。

◆ 示例元素是这个域中可能的一个值，它不必是查询结果中的元素。

BACK

对于例1，可如下构造查询要求：

S	SNO	SN	SEX	AGE	DEPT
		P.李立勇			IS

这里的查询条件是Sdept='IS',其中“=”被省略。 屏幕显示查询结果

S	SNO	SN	SEX	AGE	DEPT
		李立勇 张立伟			IS

根据用户的查询要求，求出了信息系的学生姓名。

[例33] 查询全体学生的全部数据。

S	SNO	SN	SEX	AGE	DEPT
	P.200401	P.李立勇	P.男	P.20	P.CS

BACK

显示全部数据也可以简单地把P.操作符作用在关系名上，因此本查询也可以简单表示如下：

S	SNO	SN	SEX	AGE	DEPT
P.					

2.条件查询

[例34] 求年龄大于19岁的学生的学号。

S	SNO	SN	SEX	AGE	DEPT
	P. 200401			>19	

[例35] 求计算机系年龄大于19岁的学生的学号。

本查询条件是DEPT='CS'和AGE>19两个条件的“与”。在QBE中，表示两个条件的“与”有两种方法：

(1)把两个条件写在同一行上。

BACK

S	SNO	SN	SEX	AGE	DEPT
	P.200401			>19	CS

(2)把两个条件写在不同行上，但使用相同的示例元素值。

S	SNO	SN	SEX	AGE	DEPT
	P.200401 P.200401			>19	 CS

[例36] 查询计算机系或者年龄大于19岁的学生的学号。

本查询条件是DEPT='CS'和AGE>19两个条件的“或”。在QBE中，把两个条件写在不同的行上，并且使用不用的示例元素值，即表示条件的“或”。

S	SNO	SN	SEX	AGE	DEPT
	P.200401 P.200402			>19	 CS

BACK

[例37] 查询既选修了C1号课程又选修了C2号课程的学生的学号。
本查询条件是在一个属性中的“与”关系，它只能用“与”条件的第（2）种方法表示，即写两行，但示例元素相同。

SC	SNO	CNO	SCORE
	P. 200401 P. 200402	C1 C2	

[例38] 查询选修了C1号课程的学生姓名。
本查询涉及到两个关系：SC和Student。在QBE中实现这种查询的方法是通过相同的连接属性值把多个关系连接起来。

S	SNO	SN	SEX	AGE	DEPT
	P. 200401	P. 李立勇			

SC	SNO	CNO	SCORE
	P. 200401	C1	

这里的示例元素SNO是连接属性，其值在两个表中要相同。

BACK

[例39] 查询未选修C1号课程的学生姓名。

S	SNO	SN	SEX	AGE	DEPT
	P. 200401	P. 李立勇			

SC	SNO	CNO	SCORE
¬	P. 200401	C1	

[例40] 查询有两个人以上选修的课程号。

本查询就是在一个表内连接。这个查询就是要显示这样的课程C1，它不仅被200401选修，而且也被另一个学生（¬200401）选修了。

SC	SNO	CNO	SCORE
	P. 200401 ¬ P. 200401	P. C1 C1	

BACK

3. 集函数

为了方便用户，QBE提供了一些集函数，主要包括CNT、SUM、MAX、MIN、AVG等，其含义如表2.8所示：

表2.8 QBE中的集函数

函数名	功能
CNT	对元组计数
SUM	求总和
MAX	求最大值
MIN	求最小值
AVG	求平均值

[例41] 查询信息系学生的平均年龄。

S	SNO	SN	SEX	AGE	DEPT
				P.AVG.ALL	IS

BACK

4. 对查询结果排序

对查询结果按某个属性值的升序排序，只需在相应列中填入”AO.”，按降序排序则填”DO.”。如果按多列排序，用“AO（i）”或“DO（i）。表示，其中，i为排序的优先级，i值越小，优先级越高。

[例42] 查全体男生的姓名，要求如果按所在系升序排序，对相同系的学生按年龄降序排序。

S	SNO	SN	SEX	AGE	DEPT
		P.李立勇	男	DO(2)	AO(1)

BACK

二、更新操作

1. 修改操作

修改操作符为“U.”。在QBE中，关系的主码不允许修改，如果需要修改某个主码，只能先删除该元组，然后再插入新的主码元素。

[例43] 把200401学生的年龄改为18岁。

(1) 将操作符“U.”放在值上。

S	SNO	SN	SEX	AGE	DEPT

(2) 将操作符“U”放在关系上。

S	SNO	SN	SEX	AGE	DEPT
U.	200401			18	

这里，码200401标明要修改的元组，“U.”标明所在的行是修改后的新值。

BACK

[例44] 把200401学生的年龄增加1岁。

这个修改操作涉及表达式，所以只能将操作符“U.”放在关系上。

S	SNO	SN	SEX	AGE	DEPT
U.	200401 200401			20 20+1	

2. 插入操作

插入操作符为“I.”。新插入的元组必须具有码值，其他属性值可以为空。

[例45] 把信息系的女生200428，姓名张三，年龄17岁存入数据库中。

S	SNO	SN	SEX	AGE	DEPT
I.	200428	张三	女	17	IS

BACK

3. 删除操作

删除操作符为“D.”。

[例46] 删除学生200409。

Student	SNO	SN	SEX	AGE	DEPT
D.	200409				

由于SC关系与Student关系之间具有参照关系，为保证完整性，删除200409后，通常还应删除200409学生选修的全部课程。

SC	SNO	CNO	SCORE
D.	200409		

BACK

2.6 小结

关系数据库系统是本书的重点。这是因为关系数据库系统是目前使用最广泛的数据库系统。20世纪70年代以后开发的数据库管理系统产品几乎都是基于关系的。更进一步，数据库领域近40年来的研究工作也主要是关系的。在数据库发展的历史上，最重要的成就是创立了关系模型，并广泛应用关系数据库系统。

关系数据库系统与非关系数据库系统的区别是，关系系统只有“表”这一种数据结构；而非关系数据库系统还有其他数据结构，对这些数据结构有其他复杂而不规则的操作。

本章系统讲解了关系数据库的重要概念，包括关系模型的数据结构、关系的完整性以及关系操作。介绍了用代数方式来表达的关系语言即关系代数、基于元组关系演算的ALPHA语言和基于域关系演算的QBE。本章抽象的关系操作表达，为进一步学习下一章关系数据库国际标准语言SQL打好了坚实的基础。

返回本章首页

习 题

一、单项选择题

1、设关系 R 和 S 的属性个数分别为 r 和 s，则(R×S)操作结果的属性个数为( )

A. r+s　　B. r-s　　C. r×s　　D. max(r,s)

2、在基本的关系中，下列说法正确的是( )

A. 行列顺序有关　　B. 属性名允许重名

C. 任意两个元组不允许重复　　D. 列是非同质的

3、有关系 R 和 S，R∩S 的运算等价于( )

A.S-(R-S)　　B.R-(R-S)　　C.(R-S)∪S　　D.R∪(R-S)

4、设关系 R(A，B，C)和 S(A，D)，与自然连接 R ∞ S 等价的关系代数表达式是( )

A. $\sigma_{R.A=S.A}(R\times S)$　　B. $R_{1=1}S$

C. $\Pi_{B,C,S.A,D}(\sigma_{R.A=S.A}(R\times S))$　　D. $\Pi_{R.A,B,C}(R\times S)$

5、五种基本关系代数运算是( )

A. ∪、-、×、π 和 σ　　B. ∪、-、　、∏和 σ

C. ∪、∩、×、π 和 σ　　D. ∪、∩、　、π 和 σ

返回本章首页

习 题

6、关系代数中的θ联接操作由( )操作组合而成。

A. σ 和 π　　B. σ 和×　　C. π 、 σ 和×　　D. π 和×

7、在关系数据模型中，把( )称为关系模式。

A.记录　　B.记录类型　　C.元组　　D.元组集

8、对一个关系做投影操作后，新关系的基数个数( )原来关系的基数个数。

A.小于　　B.小于或等于　C.等于　　D.大于

9、有关系：R(A，B，C)主键=A,S(D，A)主键=D，外键=A，参照 R 的属性 A，系 R 和 S 的元组如下：指出关系 S 中违反关系完整性规则的元组是( )。

R:

A	B	C
1	2	3
2	1	3

S:

D	A
1	2
2	null
3	3
4	1

A. (1,2)　　B. (2,null)　　C. (3,3)　　D. (4,1)

10、关系运算中花费时间可能最长的运算是( )。

A.投影　　B.选择　　C.广义笛卡儿积　　D.并

返回本章首页

习 题

二、填　空

1、关系中主码的取值必须惟一且非空，这条规则是_____完整性规则。

2、关系代数中专门的关系运算包括：选择、投影、连接和除法，主要实现____类操作。

3、关系数据库的关系演算语言是以_____为基础的 DML 语言。

4、关系数据库中，关系称为_____，元组亦称为____，属性亦称为____。

5、数据库描述语言的作用是_____。

6、一个关系模式可以形式化地表示为_____。

7、关系数据库操作的特点是_____式操作。

8.数据库的所有关系模式的集合构成_____，所有的关系集合构成_____。

9、在关系数据模型中，两个关系 R1 与 R2 之间存在 1：m 的联系，可以通过在一个关系 R2 中的_____在相关联的另一个关系 R1 中检索相对应的记录。

10、将两个关系中满足一定条件的元组连接到一起构成新表的操作称为_____操作。

返回本章首页

习 题

三、简答、计算或查询

1、试述关系模型的三要素内容。

2、试述关系数据库语言的特点和分类。

3、定义并理解下列概念，说明它们间的联系与区别：

（1）域、笛卡尔积、关系、元组、属性

（2）主码、候选码、外码

（3）关系模式、关系、关系数据库

4、关系数据库的完整性规则有哪些？试举例说明。

5、关系运算有哪两大类，试说明每种运算的操作含义。

6、关系代数的基本运算有哪些？请用基本运算表示非基本运算。

7、举例说明等值连接与自然连接的区别与联系。

返回本章首页

习 题

8、设有关系 R、S(如下表所示)，计算：

R

A	B	C
3	6	7
4	5	7
6	2	3
5	4	3

S

C	D	E
3	4	5
6	2	3

(1) R1 =R　　S　　(2) R2=$R_{2<2}$ S

(3) R3 = $\sigma_{B=D}(R \times S)$

9、设有学生-课程关系数据库，它由三个关系组成，它们的模式是：学生 S（学号 S#，姓名 SN，所在系 SD，年龄 SA）、课程 C（课程号 C#，课程名 CN，先修课号 PC#）、SC（学号 S#，课程号 C#，成绩 G）。

请用关系代数与 ALPHA 语言分别写出下列查询：

返回本章首页

习 题

(1) 检索学生的所有情况。

(2) 检索学生年龄大于等于 20 岁的学生姓名。

(3) 检索先修课号为 C2 的课程号。

(4) 检索课程号 C1 的成绩为 A 的所有学生姓名。

(5) 检索 S1 修读的所有课程名及先修课号。

(6) 检索年龄为 23 岁的学生所修读的课程名。

(7) 检索至少修读为 S5 修读的一门课的学生的姓名。

(8) 检索修读 S4 所修读的所有课程的学生的姓名。

(9) 检索选修所有课程的学生的学号。

(10) 检索不选修任何课程的学生的学号。

(11) 在关系 C 中增添一门新课

(12) 学号为 S17 的学生因故退学请在 S 与 SC 中将其除名。

(13) 将关系 S 中学生 S6 的年龄改为 22 岁（只需 ALPHA 操作）。

(14) 将关系 S 中学生的年龄均增加 1 岁（只需 ALPHA 操作）。

返回本章首页