第 2 章　关系数据库模型

【问题引出】在第 1 章介绍了数据库系统中的基本数据模型，不同的数据模型支持不同的数据库系统。由于层次模型和网状模型有其不可克服的缺点，而面向对象模型因比较复杂尚未得到普及应用，目前使用最广泛的是关系数据模型。那么，关系数据库与关系数据模型之间具有哪些关联？涉及哪些基本概念？关系数据模型具有哪些基本运算和操作？这就是本章所要讨论的问题。

【教学重点】关系数据库模型的组成、关系代数、关系演算、关系代数表达式的优化等。

【教学目标】了解关系数据模型的组成；理解关系、关系模型的概念并掌握模型的完整性约束；熟练掌握关系代数的各种运算；理解关系演算语言；了解优化规则及查询树优化方法。

§2.1　关系数据库模型的组成

一个完整的数据模型应能够准确地描述被建系统的静态特性、动态特性和完整性约束。在关系数据库中，相应的关系数据模型通常由关系数据结构、关系数据操作、关系数据的完整性规则三部分组成，并被称为组成关系数据模型的三要素。

2.1.1　关系数据结构

数据结构是指相互之间存在某种特定关系的数据元素的集合，是信息的一种组织方式，研究数据结构的目的是为了提高信息处理的效率。不同的数据模型，具有不同的数据结构表示方法。在关系模型中，无论是实体还是实体之间的联系，均采用单一的数据结构——关系来表示。关系数据结构是构成数据模型结构的主体，用于描述系统的静态特性，即描述数据库组成的对象本身的特征（如名字、类型、性质）及对象之间联系的关系。

1. 关系的基本元素

在关系模型中，实体和实体之间的联系是由单一的结构类型——关系表来表示的，关系模型的物理表示为二维表格，表的每一行表示一个元组，表的每一列对应一个域。每个关系有一个关系名和一个表头，表头称为关系框架，二维表的结构如图 2-1 所示。

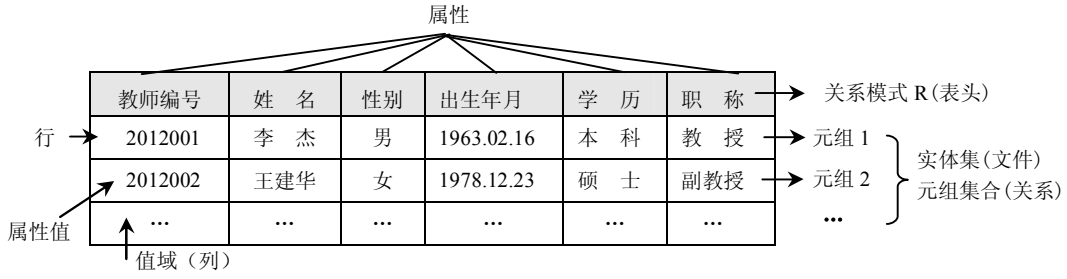


图 2-1　二维表格关系的相关概念描述

图 2-1 所示二维表是一个“教师信息”关系；该表头（框架）中包含 6 项属性；元组是关系中

的一行。下面进一步说明关系的相关概念，并给出关系中的相关定义。

（1）元组（Tuple）：是关系（表）中的一行数据，关系是元组的集合，元组是属性的集合。

（2）属性（Attribute）：二维表格中的每一列称为一个属性，属性也常称为字段，实体所具有的某一特性称为实体的属性。

（3）属性值（Attribute Value）：是指表中行和列的交汇处的元素，称为该行对应的元组在该列对应的属性上的取值，简称为属性值。属性值相当于记录中的一个数据项。

（4）值域（Domain）：是指属性的取值范围，通常简称为域，它是一组具有相同数据类型的值的集合。属性值总限定在某个值域内，例如“学号”的值域是字符串的某个子集，性别的值域为（男，女）。关系中的每个属性都必须有一个对应的值域，不同属性的值域可以相同。

要特别指出的是：关系模型对关系有一个最基本的限制要求，即关系中的每一个分量都是不可再分割的数据项，即不允许表中有表。具体说，如表 2-1 不能称为关系，是非关系表。

表 2-1　非关系表

工资号	姓　名	职　务	职　称	工　资		
				基本工资	奖　金	岗位津贴
…	…	…	…	…	…	…

2. 关系的定义

通常将一个没有重复行、重复列，并且每个行列的交叉格点只有一个基本数据的二维表看成一个关系。

【定义 2.1】$D_1\times D_2\times\cdots\times D_n$ 的子集叫作域 D_1，D_2，…，D_n 上的关系，表示为 R（D_1，D_2，…，D_n）。

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

关系与二维表格、传统的数据文件既有相似之处，又有区别。从严格意义上讲，关系是一种规范化了的二维表格（Table），其行（Row）称为元组（Tuple），其列（Column）表示属性。

（1）元组分量原子性：关系中的每一个属性值都是不可分解的数据项，并且对不同的属性要给予不同的属性名。

（2）元组的无序性：关系中不考虑元组之间的顺序，元组在关系中应是无序的，即没有行序。因为关系是元组的集合，按集合的定义，集合中的元素无序。

（3）元组唯一性。关系中的各个元组是不同的，即不允许有重复的元组。

（4）属性名唯一性：关系中的属性是不相同的，即不允许出现相同的属性名。

（5）属性的无序性：关系中属性也是无序的，即列的次序可以任意交换。

（6）分量值域同一性：关系中属性列中分量具有与该属性相同的值域（数据类型）。

3. 笛卡尔积（Cartesian Product）

为了便于实行对关系的运算，人们将关系定义为一系列域上的笛卡尔积的有限子集（关系的数学表示为笛卡尔积上有意义的子集）。这一定义与表的定义几乎是完全相符的。把关系看成是一个集合，就可以将一些直观的表格以及对表格的汇总和查询工作转换成数学的集合以及集合的运算问题。

【定义 2.2】给定一组域 D_1，D_2，…，D_n，其笛卡尔积为；

$$D_1\times D_2\times\cdots\times D_n=\{(d_1，d_2，\cdots，d_n)\mid d_i\in D_i，i=1，2，\cdots，n\}$$

在笛卡尔积中，涉及以下几个概念：

分量（Component）：笛卡尔积元素（d_1，d_2，…，d_n）中的每个值 d_i 称为一个分量。

基数（Cardinal Number）：用于描述任意集合所含元素数量多少。若 D_i（i=1，2，…，n）为有限集，其基数为 m_i（i=1，2，…，n），则 $D_1 \times D_2 \times \cdots \times D_n$ 的基数为 $M=\prod_{i=1}^{n} m_i$ 。

笛卡尔积是一个以元组为元素的集合，并且具有集合的性质。笛卡尔积中的每一个元素（d_1，d_2，…，d_n）称为一个 n 元组（n-tuplc），通常简称为元组。笛卡尔积可以表示为一张二维表，表中的每一行对应于笛卡尔积的每一个元组，表中的每一列对应于笛卡尔积的每一个域。

【实例 2-1】给定以下 3 个域：

D_1=姓名集合（name）={张三，李四}

D_2=性别集合（sex）={男，女}

D_3=专业集合（major）={计算机，自动化}

它们的笛卡尔积则为：

$D_1 \times D_2 \times D_2$={（张三，男，计算机），（张三，男，自动化），
（张三，女，计算机），（张三，女，自动化），
（李四，男，计算机），（李四，男，自动化），
（李四，女，计算机），（李四，女，自动化），

【提示】假定张三是男，李四是女，显然（张三，女，计算机），（张三，女，自动化），（李四，男，计算机），（李四，男，自动化），是没有意义的元组。这也正是需要通过触发器来解决的问题，触发器的具体内容在第 5 章中介绍。

4. 关系键与表之间的联系

由上述规范性限制可知，关系中不允许出现相同的元组，要求不同元组所具有的属性值不能都相同，例如描述读者情况的关系如表 2-2 所示。

表 2-2　读者情况关系

读者姓名	读者类别	限借数量	已借数量
张三	教师	10	7
李四	教师	10	8
赵建国	学生	5	4
钱学斌	学生	5	5

“读者类别”这一属性值就有很多元组是相同的，但由于各个元组并不是所有属性值都相同，所以各个元组还是互不相同的。当然，并不是任何属性值都可以相同。例如表 2-2 中，“读者姓名”这一属性值是不允许相同的，因为每个读者应该有不同的编号，即“读者姓名”可以唯一地标识不同的读者，“读者姓名”属性值相同的表示同一个读者，而“读者类别”、“限借数量”、“已借数量”属性值相同的则不一定是同一个读者。基于数据处理等原因，需要识别关系中的元组，因此需要考虑能够起标识作用的属性子集，于是引入了“键”的概念。下面，给出键的相关定义。

（1）键（Key）：在给定的关系中，需要用某个或某几个属性来唯一地标识一个元组，则称这样的属性或属性组为指定关系的键。

（2）超键（Super Key）：在一个关系中，若某一个属性或属性集合的值可以唯一地标识元组，

则称该属性或属性集合为该关系的超键。

例如表 2-2 中，“读者名称”+“读者类别”+“限借数量”+“已借数量”、“读者名称”+“读者类别”+“限借数量”、“读者名称”+“限借数量”、“读者名称”+“已借数量”等都是超键。

（3）候选键（Candidate Key）：如果一个属性或属性集合的值能唯一地标识一个关系的元组而又不含有多余的属性，则称该属性或属性集合为该关系的候选键。候选键可以有一个或多个，例如表 2-2 中，“读者姓名”即为候选键。

（4）主键（Primary Key）：若一个关系有多个候选键，则选定其中一个为主键。每一个关系都有且只能有一个主键。在关系模式中，常在主键下加下划线标出，包含在任一候选键中的属性称为主属性（Prime Attribute），不包含在任何候选键中的属性称为非键属性（Non-key Attribute）。

（5）全键（All Key）：在有些关系中，主键不是关系中的一个或部分属性集，而是由所有属性组成，这时主键也称为全键。

（6）外键（Foreign Key）：如果某个关系 R 中的属性或属性组 K 是另一关系 S 的主键，但不是本身的键，则称这个属性或属性组 K 为此关系 R 的外键。

（7）组合键（Composite Key）：由两个或两个以上属性组合而构成的键称为组合键。

关于键的应用实例，将在关系模型的完整性规则中详细介绍。

5. 关系模式（Relation Schema）

关系模式是对关系的描述，例如对于一个二维表格，表格的表头就是该表格所表示的关系的数据结构的描述。因此，每个关系表的表头所描述的数据结构称为一个关系模式。换句话说，关系模式是关系的框架，是关系的型，是记录格式或记录类型，而与具体的值无关。

【定义 2.3】对关系的结构及其特征的抽象描述称为关系模式。一个关系的完整模式可表示为：

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 U 中各属性来自的域集合；dom 为属性到域的映射集合，用来确定 U 中的每一个属性分别来自 D 中的哪一个域；F 为属性间的数据依赖集合，用来限定组成该关系的各元组必须满足的完整性约束条件，体现了关系的元组语义。

关系模式通常简记为 R（U）或 R（A_1，A_2，…，A_n），其中，A_1，A_2，…，A_n 为关系 R 的所有属性名。例如，表 2-2 中的读者情况关系可描述为：

读者情况关系（读者姓名，读者类别，限借数量，已借数量）

这就是读者情况关系的关系模式，该模式的一个具体取值就是表 2-2 中的读者情况关系。

关系是元组的集合，关系模式在某一时刻的状态或内容，即关系模式的值（Value），而关系模式则是对关系的抽象描述，即关系的型（Type）。因此，关系是动态的、会随时间不断变化的，而关系模式则是相对静态的、稳定的。

6. 关系数据库

在一个具体的应用领域中，所有实体以及实体之间的各种联系统一用关系表示，这些关系的集合就构成了一个关系数据库。以关系数据模型作为数据结构，一个关系就是一张表，图 2-1 和表 2-2 都是数据库中的关系。关系数据库是现代流行的数据库系统中应用最为普遍的一种，有着严格的数学基础，是最有效的数据组织方式之一。关系数据库具有如下特点：

- 一张表中包含了零行或数行，表中的行没有什么特殊的顺序。
- 一张表中包含了一列或数列，表中的列没有什么特殊的顺序。
- 表中每一个列必须有一个列名，同一表中不能有同名列。
- 同一列的属性值全部来自同一个域。

- 表中不能有完全相同的两个行，即至少有一个列的值能区分不同的行。
- 表中每一行、列的交界处是数据项，每一个数据项一般会有一个值，且只能有一个值。

由此可以看出，关系模型是用二维表格表示实体集，外键表示实体间联系的数据模型。

7. 关系数据库模式

作为关系的集合，关系数据库也会随之而变化，人们通常用相对稳定的关系数据库的型来描述关系数据库，这就是关系数据库模式。关系的集合构成了关系数据库，对应的关系模式的集合也就构成了关系数据库模式。关系数据库模式在某一时刻的值就是这些关系模式在这一时刻对应的关系的结合，即关系数据库。因此，关系模型与关系模式的关系可表示为：

关系模型=关系模式+关系

其中：关系是由二维表的表体中各行（元组）组成的值集；关系模式由二维表的表头数据（又称列或属性）构成。

2.1.2　关系数据的完整性规则

关系数据的完整性规则是对关系的某种约束条件。关系模型中有三类完整性规则：实体完整性规则、参照完整性规则和用户定义完整性规则。其中，实体完整性规则和参照完整性规则是任何一种关系模型都必须满足的完整性约束条件，被称为关系模型的两个不变性，通常由 DBMS 自动支持。

1. 实体完整性规则（Entity Integrity）

现实世界中的一个实体集就是一个基本关系，如学生的集合是一个实体，对应学生关系。实体是可区分的，它们具有某种唯一性标识，在关系模型中用主关键字作为实体唯一性标识。主关键字的属性值不能取空值（即不知道或者无意义的值）。

【定义 2.4】如果属性 A 是基本关系 R 的主属性，则属性 A 不能取空值。实体完整性规则具有如下含义：

（1）实体完整性能够保证实体的唯一性：一个基本表通常对应现实世界的一个实体集，实体在现实世界中是可以区分的，它在关系中是以主键为标识，所以主键能保证实体的唯一性，主属性不能取空值。

（2）实体完整性能够保证实体的可区分性：规则要求实体的主属性不能取空值，说明关系中没有不可标识的实体，即实体的主属性不为空值就一定能保证实体是可区分的。

例如：学生关系（学号、姓名、性别、出生年月、系别、身份证号）中，“学号”为学生关系的主属性，是主键，则任一学生的学号不能为空。

2. 参照完整性规则（Referencing Integrity）

现实世界中，实体与实体之间往往存在着某种联系，在关系模型中实体与实体间的联系都是用关系来描述的，因此可能存在着关系与关系间的引用。

【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 的键。Ks 是基本关系 S 的主键。如果 F 与 Ks 相对应，则称 F 是 R 的外键（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 不一定是不同的关系。

【实例 2-2】学生、专业实体以及它们之间的“属于”联系可以用下面两个关系表示：

学生（学号，姓名，性别，出生年月，专业代码）

专业（专业代码，专业名称，专业带头人）

主键用下划线标明。学生和专业之间的“属于”联系表现为这两个关系之间的属性引用，即学生关系的“专业代码”属性引用了专业关系的主键“专业代码”，如图 2-2 所示。

学生关系

学　号	姓　名	性别	出生年月	专业代码
20121101	赵建国	男	02/05/1990	S1101
20121102	钱学斌	男	12/23/1989	S1102

参照关系

专业关系

专业代码	专业名称	专业带头人
S1101	计算机	李杰
S1102	自动化	杨波

被参照关系

图 2-2　学生关系和专业关系之间的属性引用

学生关系中的“专业代码”值必须是确实存在的某个专业的专业号，即专业关系中的某个“专业代码”值，专业关系中不存在的“专业代码”值是毫无意义的。因此，在这样的属性引用中，学生关系中“专业代码”属性的取值需参照专业关系中主键“专业代码”的取值。

除了取专业关系中的某个“专业代码”值外，学生关系的“专业代码”属性也可以取空值，表示该生尚未分配专业或不知道他的专业。

在学生关系中，“专业代码”属性虽然不是主键，但它却引用（或参照）了专业关系的主键，这样的属性引用不但可以表达学生和专业之间的“属于”联系，而且还使它的取值受到了一定的限制，这样的属性称为外键，定义如下：

【定义 2.6】若属性（或属性组）F 是基本关系 R 的外键，它与基本关系 S 的主键 Ks 相对应（基本关系 R 和 S 不一定是不同的关系），则对于 R 中的每个元组在 F 上的值必须取或取空值（F 的每个属性值均为空值），或等于 S 中某个元组的主键值。

有了外键定义，参照完整性就可以表达成：若属性（或属性组）F 是关系 R 的外键，引用（或参照）的是关系 S 的主键 Ks，则 R 中每个元组在 F 上的取值要么为空值，要么为 S 中某个元组的 Ks 值。和实体完整性不同，参照完整性约束的是外键的取值，用来保证外键对主键的正确引用，以体现客观对象之间的各种联系，如实例 2-2 中学生和专业之间的“属于”联系。

按照参照完整性，学生关系的“专业号”属性要么取空值，表示该生尚未分配专业；要么取专业关系中的某个“专业号”值，表示该生已属于某个确实存在的专业。

【实例 2-3】学生、课程实体以及它们之间的“选修”联系可以用下面 3 个关系表示：

学生（学号，姓名，性别，出生年月，专业代码）

课程（课程号，课程名，学时）

选修（学号，课程号，成绩）

选修关系用来表达学生和课程之间的“选修”联系，它和学生、课程关系之间都存在属性的引用，即“学号”属性引用了学生关系的主键“学号”，“课程号”属性引用了课程关系的主键“课程号”，如图 2-3 所示。因此，选修关系有两个外键：学号和课程号。

按照参照完整性，选修关系的“学号”属性要么取空值，要么取学生关系中的某个“学号”值，即某名学生的学号；选修关系的“课程号”属性要么取空值，要么取课程关系中的某个“课程号”值，即某门课程的课程号。但是，按照实体完整性，它们都不能取空值。

3. 用户定义完整性规则（User-defined Integrity）

实体完整性规则和参照完整性规则分别定义了对主键的约束和对外键的约束，是关系模型中最

基本的约束。此外，关系数据库系统根据现实世界中应用环境的不同，还需要一些特殊的约束条件。它是用户根据需要自己定义的，因而称为用户定义完整性规则。

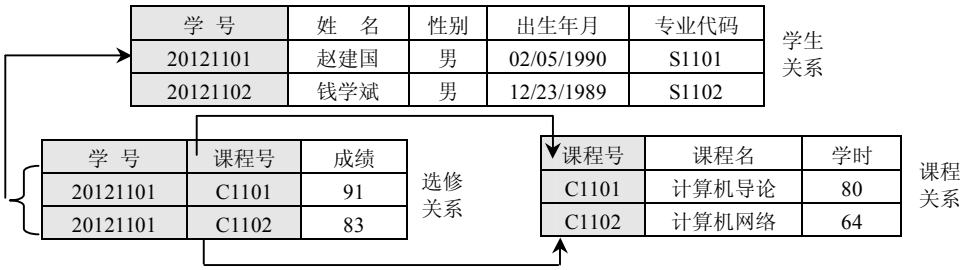


图 2-3　选修关系和学生、课程关系之间的属性引用示意图

【定义 2.7】针对某一具体应用环境，给出关系数据库的约束条件，这些约束条件就是反映某一应用所涉及的数据必须满足的语义要求。

例如：年龄属性，如果属于某一个学生主体，则可能要求年龄在 17 岁到 25 岁之间，而如果年龄属性属于某一个公司员工主体，则可能要求年龄在 18 岁到 55 岁之间。

用户定义完整性通常是定义对关系中除主键与外键属性之外的其它属性取值的约束，包括数据类型、精度、取值范围、是否允许空值等。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、系统的方法处理。

对于这类完整性，关系模型只提供定义和检验这类完整性的机制，以使用户能够满足自己的需求，而关系模型自身并不去定义任何这类完整性规则。

【提示】为了维护数据库中数据的完整性，在对关系数据库执行插入、删除和修改操作时，就要检查上述三类完整性规则。

2.1.3　关系数据操作

关系数据操作是指施加于数据模型中数据的运算和运算规则，用于描述系统的动态特性，反映事物的行为特征。为此，在数据模型中必须定义操作的含义、符号、规则以及实现操作的语言（包括数据定义、数据操纵和数据控制）。数据库主要有两类操作：查询和更新（插入、删除、修改）。查询操作是最基本、最重要的操作，它是更新操作的基础。

关系数据操作建立在关系的基础上，一般分为数据查询和数据操纵（更新）两大类。数据查询操作是对数据库进行各种检索；数据更新是对数据库进行插入、删除和修改等操作。

1. 数据查询（Data Query）

用户可以查询关系数据库中的数据，因而通常简称为关系查询，它包括一个关系内的查询和多个关系的查询。关系查询的基本单位是元组分量，查询的前提是关系中的检索或者定位。关系查询的表达能力很强，是关系操作中最主要的部分，包括选择（Select）、投影（Project）、连接（Join）、除（Divide）、并（Union）、差（Except）、交（Intersection）、笛卡尔积（Cartesian Product）等。其中，选择、投影、并、差、笛卡尔积是 5 种基本操作，其它操作可以在这 5 种基本操作的基础上导出。关系查询可分解为以下 3 种基本操作：

（1）关系属性指定：指定一个关系内的某些属性，用它确定关系这个二维表中的列。

（2）关系元组选择：用一个逻辑表达式给出关系中满足此表达式的元组，用它确定关系这个表的行。

用上述两种操作即可确定一张二维表内满足一定行、一定列要求的数据。

（3）两个关系合并：主要用于多个关系之间的查询，其基本步骤是先将两个关系合并为一个关系，由此将多个关系相继合并为一个关系。将多个关系合并为一个关系之后，再对合并后关系进行上述的两个定位操作。

2. 数据操纵（Data Manipulate）

数据操纵也称为数据更新（Data Change），分为数据删除、数据插入和数据修改 3 种基本操作。

（1）数据删除（Data Delete）：在进行数据删除操作时应注意以下两点：

① 数据删除的基本单位为元组，数据删除的功能是将指定关系内的指定元组删除。

② 数据删除是两个基本操作的组合，一个关系内的元组选择（横向定位）操作和关系中元组删除操作。

（2）数据插入（Data Insert）：在进行数据插入操作时应注意以下两点：

① 数据插入是针对一个关系而言，即在指定关系中插入一个或多个元组。

② 数据插入中不需要定位，仅需要对关系中的元组进行插入操作，即是说，插入只有一个基本动作：关系元组插入操作。

（3）数据修改（Data Update）：在进行数据修改操作时应注意以下两点：

① 数据修改是在一个关系中修改指定的元组与属性值。

② 数据修改可以分解为两个更为基本的操作：先删除需要修改的元组，再插入修改后的元组即可。

3. 空值处理

在关系操作中还有一个重要问题——空值处理。在关系元组的分量中允许出现空值（Null Value）以表示信息的空缺。在出现空值的元组分量中一般可用 NULL 表示。目前一般关系数据库系统中都支持空值处理，但是具有以下两个限制：

（1）主键中不允许出现空值：关系中主键不能为空值，因为主键是关系元组的标识，如果主键为空值，则失去了其标识的作用。

（2）定义有关空值的运算：在算术运算中如果出现空值，则结果也为空值。在比较运算中如果出现空值，则其结果为 F（假）。此外，在作统计时如果 SUM、AVG、MAX、MIN 中有空值输入时结果也为空值，而在作 COUNT 时如果有空值则其值为 0。

4. 关系数据查询语言

数据库的操作是通过语言来实现的。关系数据库抽象层次上的关系查询语言可分为三类：关系代数、关系演算和 SQL，它们都是非过程化的查询语言。其中，关系代数是用代数方式表达的关系查询语言；关系演算是用逻辑方式表达的关系查询语言，关系演算分为元组关系演算和域关系演算；SQL 是介于关系代数和关系演算之间、具有双重特点的结构化查询语言（Structured Query Language）。关系数据查询语言的分类如图 2-4 所示。

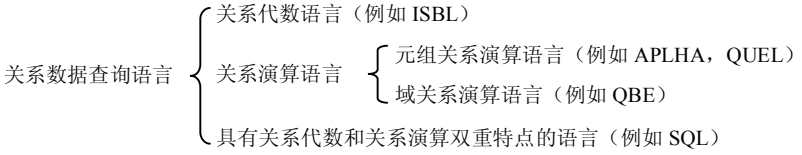


图 2-4　关系数据语言的种类

关系代数、元组关系演算和域关系演算 3 种查询语言在表达能力上是完全等价的，常用作评估实际系统中查询语言能力的标准或理论基础。而 SQL 是一种介于关系代数和关系演算之间的结构化非过程查询语言，不但具有丰富的查询功能，并且还具有数据定义、数据更新和数据控制等功能，是集数据查询、数据操纵和数据控制于一体的关系数据语言，是目前所有关系数据库都支持的标准语言（在第 4 章单独介绍 SQL）。

§2.2　关系代数

既然把二维表看成关系，那么就可以用关系代数（Relational Aalgebra）作为语言对关系进行操作。关系代数是以集合代数为基础发展起来的，是以关系为运算对象的一组运算的集合。关系代数每个运算都以一个或多个关系作为它的运算对象，并生成另一个关系作为该运算的结果。从数据操作的观点来看，关系代数是一种抽象的查询语言，通过对关系的运算来表达查询。

关系代数运算分为两类：传统的集合运算和专门的关系运算。传统的集合运算包括并、交、差、广义笛卡尔积。专门的关系运算包括选择、投影、连接、除。关系运算类型如表 2-3 所示。

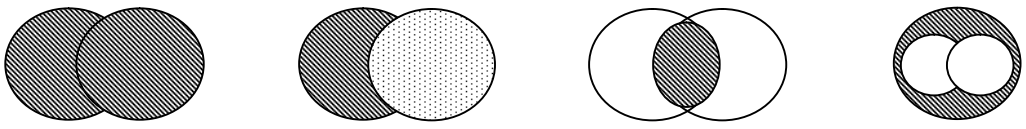
表 2-3　关系运算类型

传统的集合运算				专门的关系运算			
并	交	差	广义笛卡尔积	选择	投影	连接	除
∪	∩	－	×	δ	Π	∞	÷

由于任何一种运算都是将一定的运算符作用于一定的运算对象上，得到预期的运算结果，所以运算对象、运算符、运算结果是运算的三大要素。

2.2.1　传统的集合运算

传统的集合运算将关系看成元组的集合，其运算是从关系的“水平“方向（行的角度）来进行。集合运算包括并、差、交、广义笛卡尔积 4 种运算，关系的集合运算要求参加运算的关系必须是相容的关系。并、交、差、广义笛卡尔积 4 种运算通常也可以采用文氏图（Venn Diagram）来表示。文氏图是在集合论数学分支中，在不太严格的意义下用以表示集合的一种草图，用于展示在不同的事物集合之间的数学或逻辑联系。4 种运算的文氏图如图 2-5 所示。



（a）R∪S 运算　（b）R－S 运算　（c）R∩S 运算　（d）R×S 运算

图 2-5　传统的集合运算

1. 并（Union）运算

并运算是将两个相容的关系 R 和 S 中的所有元素合并，构成一个新的关系。

【定义 2.8】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 的元组组成，其结果关系仍为 n 目关系。记为：

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

其中，∪为二目运算，从“行”上取值。作用是在一个关系中插入一个数据集合，自动去掉相同元组，即在并的结果关系中，相同的元组只保留一个；等式右边大括号中的 t 是一个元组变量，表示结果集合由元组 t 构成；竖线“|”右边是对 t 的约束条件，或者说是对 t 的解释。以下其它运算的定义方式与此类似。

【实例 2-4】有两个结构完全相同的学生表 R 和学生表 S 分别存放两个班级的学生，若将学生表 R 的记录追加到表 S 中，则需要使用并运算 R∪S，其运算关系如图 2-6 所示。

2. 差（Difference）运算

差运算是将两个相容的关系 R 和 S，使其属于 R 但不属于 S 的元组构成一个新的关系。

【定义 2.9】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，其结果关系仍为 n 目关系。记为：

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

【实例 2-5】有两个结构完全相同的学生表 R 和学生表 S，R 是选修数据库课程的学生集合，S 是选修 VC++课程的学生集合，若查询选修了数据库但没有选修 VC++的学生，则需要使用差运算 R–S，其运算关系如图 2-7 所示。

表 R

学 号	姓 名
20121101	赵建国
20121102	钱学斌

表 S

学 号	姓 名
20121102	钱学斌
20121202	吴 微

表 R∪S

学 号	姓 名
20121101	赵建国
20121102	钱学斌
20121202	吴 微

图 2-6 集合运算 R∪S 示意图

表 R

课程名	姓 名
数据库	赵建国
数据库	钱学斌
数据库	周自强

表 S

课程名	姓 名
VC++	赵建国

表 R–S

课程名	姓 名
数据库	钱学斌
数据库	周自强

图 2-7 集合运算 R–S 示意图

3. 交（Intersection）运算

交运算是将两个相容的关系 R 和 S，使其属于 R 也属于 S 的元组构成一个新的关系。

【定义 2.10】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 的元组组成，其结果关系仍为 n 目关系。记为：

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

【实例 2-6】有两个结构完全相同的学生表 R 和学生表 S，R 是选修数据库课程的学生集合，S 是选修 VC++课程的学生集合，若查询选修了数据库并且选修 VC++的学生，则需要使用交运算 R∩S，其运算关系如图 2-8 所示。

表 R

课程名	姓 名
数据库	赵建国
数据库	周自强

表 S

课程名	姓 名
VC++	赵建国

表 R∩S

课程名	姓 名
VC++	赵建国

图 2-8 集合运算 R∩S 示意图

4. 广义笛卡尔积（Extended Cartesian Product）

对关系数据库的查询通常会涉及多个关系，例如，查询某个同学选修的各门课程成绩就涉及学生关系和选修关系，如何将这两个并不兼容的关系合并在一起呢？广义笛卡尔积就是用来合并两个关系的基本运算，这种基本运算即关系的乘法运算。

【定义 2.11】设 R 为 m 元关系，S 为 n 元关系，则 R 与 S 的广义笛卡尔积 R×S 是一个（m+n）

元关系，其中每个元组的前 m 个分量是 R 中的一个元组，后 n 个分量是 S 中的一个元组。若 R 有 K_1 个元组，S 有 K_2 个元组，则 R×S 有（$K_1 \times K_2$）个元组，即广义笛卡尔积为：

$$R \times S=\{(a_1,a_2,\cdots a_m,\ b_1,b_2,\cdots b_n) \mid (a_1,a_2,\cdots a_m) \in R \wedge (b_1,b_2,\cdots b_n) \in S\}$$

【实例 2-7】利用表 R 和表 S 所示数据做广义笛卡尔积，其结果如图 2-9 所示。

表 R

教师号	姓名	系部
T1101	张三	计算机
T1102	李四	自动化

表 S

教师号	姓名	职称
T1101	张三	教授
T1102	李四	副教授

表 R 与表 S 笛卡尔积

教师号	姓名	系部	教师号	姓名	职称
T1101	张三	计算机	T1101	张三	教授
T1101	张三	计算机	T1102	李四	副教授
T1102	李四	自动化	T1101	张三	教授
T1102	李四	自动化	T1102	李四	副教授

图 2-9　集合运算笛卡尔积示意图

2.2.2　专门的关系运算

关系运算是针对关系数据库数据进行的操作运算，但仅仅依靠传统的集合运算，还不能灵活地实现多样的查询操作。因此，E.F.Codd 又定义了一组专门的关系运算，包括选择、投影、连接和除法 4 种运算。

1. 选择（Select）运算

选择运算又称为限制（Restriction）运算，是指从一个关系 R 中选取满足给定条件的元组构成一个新的关系。换句话说，选择运算是根据给定的条件对关系进行水平分解。

【定义 2.12】在关系 R 中选择满足条件的元组组成一个新的关系，这个关系是关系 R 的一个子集。如果选择条件用 F 表示，则选择运算可记为：

$$\delta_F(R)=\{t \mid t \in R \wedge F(t)=\text{"真"}\}$$

其中：δ是选择运算符；R 是关系名；F 是限定选择条件，可以递归定义为：

① F 是一个逻辑表达式，取值为“真”或“假”。

② F 由逻辑运算符“∧”（and）、“∨”（or）、“¬”（not）连接各种算术表达式组成。

③ 算术表达式的基本形式为 xθy，θ={＞、≥、＜、≤、=、≠}。x、y 可以是属性名、常量或简单函数。

【实例 2-8】若从学生信息表 R 中选出性别为女的学生，则可以得到女生的学生信息表 S。选择运算结果如图 2-10 所示。

表 R

学　号	姓　名	性别	出生年月
20121101	赵建国	男	02/05/1990
20121102	钱学斌	男	12/23/1989
20121103	孙经文	女	01/12/1990
20121201	李建华	男	11/12/1989
20121202	周小燕	女	09/12/1990

表 S

学　号	姓　名	性别	出生年月
20121103	孙经文	女	01/12/1990
20121202	周小燕	女	09/12/1990

图 2-10　选择运算过程示意图

2. 投影（Projection）运算

投影运算是指从一个关系 R 中选取所需要的列构成一个新的关系，具体说，是对一个关系做垂直分解，消去关系中的某些列，删除重复元组，并重新排列次序。

【定义 2.13】设关系 R 为 r 目关系，其元组变量为 $t^r=(t_1, t_2, \cdots, t_r)$，关系 R 在其分量 A_{j1}，A_{j2}，…，A_{jk}（$k \leqslant r$，j_1，j_2，…，j_k 为 1 到 r 之间互不相同的整数）上的投影是一个 k 目关系，并定义为：

$$\Pi_{j1, j2, \cdots, jk}(R)=\{t|t=(t_{j1}, t_{j2}, \cdots, t_{jk}) \wedge (A_{j1}, A_{j2}, \cdots, A_{jk}) \in R\}$$

其中，Π为投影运算符。

投影运算是按照 j_1，j_2，…，j_k 的顺序（或按照属性名序列 A_{j1}，A_{j2}，…，A_{jk}），从关系中只取出列序号为 j_1，j_2，…，j_k（或按照属性名序列 A_{j1}，A_{j2}，…，A_{jk}）的 k 列，并除去结果中的重复元组，构成一个以 j_1，j_2，…，j_k 为顺序（或以 A_{j1}，A_{j2}，…，A_{jk} 为属性名序列）的 k 目关系。

【实例 2-9】从学生信息表 R 中抽出学号、姓名列，得到学生的花名册表 S。投影运算结果如图 2-11 所示。

表 R

学　号	姓　名	性别	出生年月	专业代码	班　级
20121101	赵建国	男	02/05/1990	S1101	201211
20121102	钱学斌	男	12/23/1989	S1101	201211
20121103	孙经文	女	01/12/1990	S1102	201211
20121201	李建华	男	11/12/1989	S1102	201212
…	…	…	…	…	

表 S

学　号	姓　名
20121101	赵建国
20121102	钱学斌
20121103	孙经文
20121201	李建华
…	…

图 2-11　投影运算过程示意图

【提示】关系是一个二维表，对它的操作（运算）可以从水平（行）的角度进行，即选择运算；也可以从纵向（列）的角度进行，即投影运算。投影运算实质上是对关系按列进行垂直分割的运算，运算的结果是保留原关系中投影运算符下标所标注的那些列，消去原关系中投影运算符下标中没有标注的那些列，并去掉重复元组。

3. 连接（Join）运算

由于广义笛卡尔积会产生大量的无效元组，为了能实现关系的有效合并，元组之间应按一定的条件进行组合，这就是连接运算。连接运算是把两个关系中的元组按条件连接起来，形成一个新的关系。连接运算是笛卡尔积、选择和投影操作的组合。连接运算有多种类型，这里简要介绍最常用也是最重要的三种连接方法：条件连接、等值连接和自然连接。

（1）条件连接（Condition Join）：也称为 θ 连接，是从两个关系的广义笛卡尔积中选择属性间满足一定条件的元组构成的一个新关系，或者说，两个关系的元组只有在相应属性上的取值满足一定条件时才能组合成为新关系的元组，记作：

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

其中：A 和 B 分别为 R 和 S 中可比较的属性（组），θ 是比较运算符，如=、≠、≤、＞等。R 中元组 t_r 在属性（组）A 上的取值 $t_r[A]$和 S 中元组 t_s 在属性（组）B 上的取值 $t_s[B]$只有满足条件 $t_r[A]\theta t_s[B]$时才能组合成为 $R \underset{A\theta B}{\infty} S$ 中的元组$(t_r t_s)$。

（2）等值连接（Equivalence Join）：是指 θ 为等值比较谓词的连接运算，即 θ 为“=”的连接运算。具体说，是从两个关系（R 和 S）的笛卡尔积的运算结果中，选取 A、B 属性值相等的那些

元组构成新的关系操作。即有：

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

【实例 2-10】因某种情况，将原来的两个学习小组（R 和 S）合并为一个学习小组 W。其连接条件是表 R. 小组=表 S. 小组。等值连接结果如图 2-12 所示。

表 R

小组	学生
一组	赵建国
一组	钱学斌
二组	孙经文
二组	李建华

表 R.小组=
表 S.小组

表 S

小组	班主任
一组	刘老师
二组	黄老师

表 W

小组	学生	班主任
一组	赵建国	刘老师
一组	钱学斌	刘老师
二组	孙经文	黄老师
二组	李建华	黄老师

图 2-12　连接运算过程示意图

（3）自然连接（Natural Join）：是一种特殊的等值连接，是将两个基本关系在等值连接的基础上，消除冗余列（重复属性），形成新的关系操作。若 R 和 S 具有相同的属性（组）B，则它们的自然连接记作：

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

其中：$t_s[B]$表示在 t_s 中去掉和 t_r 重复的 B 分量。

【提示】自然连接与等值连接的区别主要体现在以下 3 个方面：

① 自然连接相等的属性必须是相同属性；等值连接相等的属性可以是相同或不同属性。

② 自然连接必须去掉重复的属性列；等值连接无此要求，并且等值连接不做投影运算。

③ 自然连接一般用于有公共属性的情况。如果 2 个关系没有公共属性，则自然连接就退化为广义笛卡尔乘积；如果是 2 个关系模式完全相同的关系进行自然连接运算，则变为交运算。

【实例 2-11】设有两个学生表，把两个表按属性名相同进行等值连接，对于每对相同的属性在结果中只保留一个。自然连接结果如图 2-13 所示。

表 R

教师号	姓　名	职　称
T1101	张　三	教　授
T1102	李　四	副教授
T1103	王　五	讲　师

表 S

教师号	姓　名	系　部
T1101	张　三	计算机
T1102	李　四	计算机
T1103	王　五	外　语

表 W

教师号	姓　名	职　称	系　部
T1101	张　三	教　授	计算机
T1102	李　四	副教授	计算机
T1103	王　五	讲　师	外　语

图 2-13　自然连接运算过程示意图

由以上两个实例看出：在连接运算中，按字段相等执行的连接称为等值连接；去掉重复值的等值连接称为自然连接。而自然连接是一种特别有用的连接。

【提示】基于自然连接运算的特殊的扩展方式：如果需要把不能连接的元组（即丢弃的元组）也保留到结果关系中，那么关系 R 中不能连接的元组在结果元组中的关系 S 的属性上可以全部置为空值 NULL，反之类似处理。这种连接就叫做外连接（Outer Join）。如果只把左关系中不能连接的元组保留到结果关系中，则称为左外连接（Left Outer Join 或 Left Join）；反之，如果只把右关系中不能连接的元组保留到结果关系中，则称为右外连接（Right Outer Join 或 Right Join）。

4. 除法（Division）运算

除运算也称为商（Quotient）运算，是基于选择、投影、连接，从关系的行方向和列方向进行

的运算。例如，若要查询选修了某些课程的学生，并且这些课程只是具有给定特征的一组不能明确列举出来的课程，仅用选择操作是远远不够的。因为选择条件无法明确指出具体的课程，这时就需要用除运算。因此，除运算在表达某种特殊类型的查询时是非常有效的。

【定义 2.14】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，但必须是出自相同的域集。R 与 S 的除运算得到一个新的关系 P(X)，P 是 R 中满足下列条件的元组在 X 属性列上的投影，元组在 X 上分量值 x 的像集 Y，包含 S 在 Y 上投影的集合。记作：

$$R \div Y=\{t_r[X] \mid t_r \in R \wedge \prod y(S) \in Y_x\}$$

其中 Y_x 为 x 在 R 中的像集，$x=t_r[X]$。

【实例 2-12】设有学生选修课程关系表 SC 和课程表 C，试找出选修了全部课程的学生的学号。对于这类问题可用除运算解决，即 SC÷C。除运算结果如图 2-14 所示。

表 SC

学号	课程号	学号	课程号
20121101	C1101	20121102	C1102
20121101	C1102	20121102	C1103
20121102	C1101	20121103	C1102

÷

表 C

课程号
C1101
C1101
C1102

表 SC÷C

学号
20121102

图 2-14　除运算过程示意图

以上介绍了传统的集合运算和专门的关系运算的定义，并通过图形解析方式描述了集合运算和关系运算的概念。怎样利用查询命令实现对关系数据库查询操作，将在下面讨论。

2.2.3　关系代数运算在关系数据库查询操作中的应用实例

在关系代数运算中，把 4 种基本代数运算经过有限次复合后形成的式子称为关系代数表达式（简称代数表达式）。这种表达式的运算结果仍是一个关系，可以用关系代数表示需要进行的各种数据库查询和更新处理的需求。查询语句的关系代数表达式的一般形式为：

$$\Pi\cdots(\delta\cdots(R\times S))\text{或}\quad \Pi\cdots(\delta\cdots(R\infty S))$$

上面的式子表示：首先取得查询涉及的关系，再执行笛卡尔积或自然连接操作得到一张中间表格，然后对该中间表格执行水平分割（选择操作）和垂直分割（投影操作），当查询涉及否定或全部包含值时，上述形式就不能表达了，就要用到差操作或除法操作。

下面根据图 1-10 所示的关系数据模型和表 2-4 中的代码表示，实现相关查询操作。

表 2-4　教学管理数据库中的 6 种关系模式

关系模式	关系模式的代码表示
学生关系模式（学号, 姓名, 性别, 出生年月, 专业代码, 班级）	S（Sno,Sname,Sex,Sbirthin,Scode, Class）
教师关系模式（教师号, 姓名, 性别, 年龄, 职称）	T（Tno,Tname,Tsex,Tage,Titleof）
课程关系模式（课程号, 课程名，学时）	C（Cno,Cname,Classh）
专业关系模式（专业代码, 专业名称，专业带头人）	SS（Scode,Sname,Slead）
学习关系模式（学号, 课程名, 成绩）	SC（Sno,Cname,Grade）
授课关系模式（教师号, 课程名）	TC（Tno,Cname）

【实例 2-13】查询学生数据库中所有的女学生。

问题解析：从学生数据库表中选择性别属性 Sex 之值为“女”的元组。查询命令为：

$$\delta_{Sex='女'}(S) \quad 或 \quad \delta_{3='女'}(S)$$

查询结果如图 2-15 所示。

【实例 2-14】查询全体教师的教师号、姓名、性别和职称。

问题解析：从教师数据库表中把各教师的教师号、姓名、性别和职称 4 列选出来。查询命令为：

$$\Pi_{Tno,\ Tname,\ Tsex,\ Titleof}(T) \quad 或 \quad \Pi_{1,\ 2,\ 3,\ 5}(T)$$

查询结果如图 2-16 所示。

学　号	姓　名	性别	出生年月	专业名称
20121103	孙经文	女	01/12/1990	S1101
…	…	…	…	…

图 2-15　选择查询

教师号	姓　名	性别	职　称
T1101	张　三	男	教　授
T1102	李　四	女	副教授
T1103	王　五	男	讲　师
T1104	赵　六	男	实验师

图 2-16　选择查询

【实例 2-15】查询专业代码为 S1101 的男生的学号和姓名。

问题解析：查询专业代码为 S1101 的男生就是从学生数据库表中选择出专业代码属性 Scode 为 S1101，且性别属性 Sex 之值为“男”的那些元组。取出其学号和姓名属性显然只要再对选出的那些元组在属性学号 Sno 和姓名 Sname 上进行投影即可。查询命令为：

$$\prod_{Sno,Sname}(\delta_{Sex='男'\wedge Scode='S1101'}(S)) \text{ 或 } \prod_{1,2}(\delta_{3='男'\wedge}\delta_{='S1101'}(S))$$

查询结果如图 2-17 所示。

【实例 2-16】查询选修了课程号为 S1101 或 S1102 的学生的学号和姓名。

问题解析：可以在选择运算的条件中用∨连接两个具有或关系的课程名；也可以分别以这两个课程名为条件，查询出满足条件的那些元组，再利用并运算将它们合并。查询命令为：

$$\prod_{Sno,Sname}(\delta_{Cno='S1101'\vee Cno='S1102'}(SC))$$

或

$$\prod_{Sno,Sname}(\delta_{Cno='S1101'}(SC)) \cup \prod_{Sno,Sname}(\delta_{Cno='S1102'}(SC))$$

查询结果如图 2-18 所示。

学　号	姓　名
20121101	赵建国
20121102	钱学斌

图 2-17　投影查询

学　号	姓　名
20121101	赵建国
20121102	钱学斌
20121103	孙经文

图 2-18　投影、并查询

【实例 2-17】查询选修了课程号为 S1102 和课程号为 S1103 的学生的学号和姓名。

问题解析：由于查询过程是按元组一行一行地检索，所以一个元组只能有一个课程号 Cno 属性。其查询方法有两种：一种方法是通过广义笛卡尔积运算 SC×SC，使同一个元组中具有两个课程号 Cno；另一种方法是分别求出“选修了以其课程号 Cno 表示的某门课程的学生，然后再通过具有相同学号的交操作，便可以找到同时选修了两门课程的学生。查询命令为：

$$\prod_{Sno,Sname}(\delta_{Cno='S1102'\wedge Cno='S1103'}(SC\times SC))$$

或

$$\prod_{Sno,Sname}(\delta_{Cno='S1102'}(SC)) \cap \prod_{Sno,Sname}(\delta_{Cno='S1103'}(SC))$$

查询结果如图 2-19 所示。

【实例 2-18】查询选修了课程号为 C1103 的学生的学号、姓名和考试成绩。

问题解析：学生的学号和姓名属性在学生关系 S 中，而考试成绩属性在学习关系 SC 中。显然，以学号 Sno 作为公共属性，将学生关系 S 与学习关系 SC 进行自然连接后，再对其进行以 Cno='C1103' 为条件的选择，便可以选出选修了课程号为 C1103 的那些学生的全部属性及其选修的课程号 C1103 和分数 Grade 属性组成的元组，然后再对其在学号 Sno、姓名 Sname 和分数 Grade 上投影，即为查询的结果。查询命令为：

$$\Pi_{Sno,Sname,Grade}(\delta_{Cno='C1103'}(S \infty SC))$$

查询结果如图 2-20 所示。

学　号	姓　名
20121102	钱学斌
20121103	孙经文

图 2-19　笛卡尔积、交查询结果

学　号	姓　名	分数
20121103	孙经文	88
…	…	…

图 2-20　自然连接、投影查询

【实例 2-19】查询计算机专业（专业代码为 S1101）学习计算机导论课程的学生的学号、姓名和成绩。

问题解析：由于查询条件之一需要用到课程名“计算机导论”。显然，以学号 Sno 作为公共属性，将学生关系 S 与学习关系 SC 进行自然连接后得到的元组中就包含课程号 Cno 属性了；然后，再以课程号 Cno 作为公共属性，将 S∞SC 与课程关系 C 进行自然连接后得到的元组中就包含“课程名”属性了。这时对所得到的元组就可以“Cname='计算机导论'”为条件进行选择操作了。其余思路与上题相似。查询命令为：

$$\Pi_{Sno,Sname,Grade}(\delta_{Cno='S1101' \wedge Cname='计算机导论'}(S \infty SC \infty C))$$

查询结果如图 2-21 所示。

【实例 2-20】查询没有学习课程号为 C1101 或课程号为 C1102 的学生的学号、姓名和班级。

问题解析：从已具有属性“学号，姓名，班级”的全部学生中去掉选修了“课程号为 C1101 或课程号为 C1102”的那些学生，剩余的显然就是没有选修“课程号为 C1101 或课程号为 C1102”的学生了。查询命令为：

$$\Pi_{Sno,Sname,Class}(S) - \Pi_{Sno,Sname,Class}(\delta_{Cno='C1101' \vee Cno='C1102'}(S \infty SC))$$

查询结果如图 2-22 所示。

学　号	姓　名	分数
20121101	赵建国	91
…	…	…

图 2-21　自然连接查询

学　号	姓　名	班级
20121103	孙经文	201211
…	…	…

图 2-22　差运算查询

【实例 2-21】查询学习了全部课程的学生的学号和姓名。

问题解析：可先通过对学习关系 SC 在“学号 Sno，课程号 Cno”属性上的投影运算，取出学生学习的全部课程；然后用课程关系中的课程号 Cno 与其进行除运算。根据除运算的性质，只有某个学生的学号 Sno 与所有的课程号 Cno 组成的元组都在 $\Pi_{Sno,Cno}(C)$ 中，该学生才是修完全部课程的学生。查询命令为：

$$\Pi_{Sno,Sname}(S \infty (\Pi_{Sno,Cno}(SC) \div \Pi_{Cno}(C)))$$

显然，在图 1-10 所示的大学教学管理数据库表中，没有学习了全部课程的学生。

【实例 2-22】为了进一步说明除运算，查询学习了图 2-23 所示全部课程的学生的学号。

问题解析：假设所开设的课程号均以 C 开头。对此，可以采用如图 2-23 所示步骤求解。

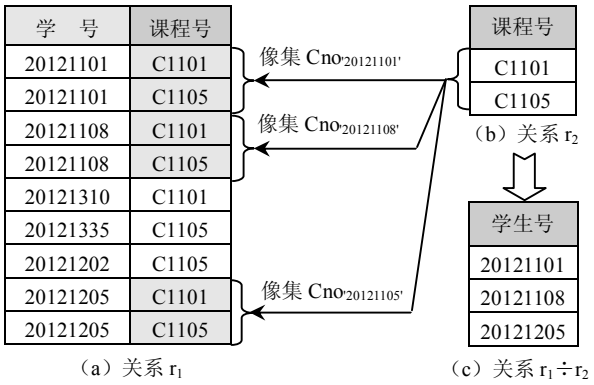
① 查询出修读过信息学院课程的所有学生，查询结果如图 2-23（a）所示（这里只保留了学号 Sno 和课程号 Cno 属性）。

$$r_1=\prod_{Sno,Cno}(\delta_{Cno\ LINK'C'\%}(Sourse))$$

② 查询开设的所有课程，查询结果如图 2-23（b）所示（这里只保留了课程号 Cno 属性）。

$$r_2=\prod_{Cno}(\delta_{Cno\ LINK'C'\%}(Course))$$

③ 比较图 2-23（a）和（b）可以发现：修读过所有课程的学生就是关系 r_1 中满足“Cno”列包含关系 r_2 的所有行的那些学生，就是 $r_1 \div r_2$ 的结果，如图 2-23（c）所示。



（a）关系 r_1　（b）关系 r_2　（c）关系 $r_1 \div r_2$

图 2-23　连接、除运算查询

【提示】在求 r_1，r_2 时使用了字符串匹配运算符“LINK”及其通配符“%”。

*§ 2.3　关系演算

关系演算是以数理逻辑中的谓词来表达查询的方式，把谓词演算（Predicate Calculus）推广到关系运算中就构成了关系演算（Relational Calculus）。因此，关系代数和关系演算是可以相互替代的。它们之间的基本区别是：关系代数提供了连接、并和投影等明确的集合操作符，并且这些集合操作符告诉系统如何从给定关系构造所要求的关系；而关系演算仅提供了一种描述来说明所要求的关系的定义。

关系代数中先做什么运算，后做什么运算是有顺序的，因而不是完全非过程化的。与关系代数不同，使用谓词演算只需要用谓词的形式给出查询结果应满足的条件，查询如何实现完全由系统自行解决，因而是高度非过程化的语言。

根据谓词变量不同，关系演算分为元组关系演算和域关系演算，二者间的主要区别是前者的变量为元组变量，而后者的变量为域变量。

2.3.1　元组关系演算

元组关系演算（Tuple Relational Calculus）是元组变量作为谓词变元的基本对象，是非过程化查询语言，它只描述所需信息，而不给出获得该信息的具体过程。在元组关系演算中，其元组关系演算表达式中的变量是以元组为单位的，其一般形式为：

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

其中，t 是元组变量，表示一个定长的元组；P(t)是元组关系演算公式，公式是由原子公式和运算符组成的。

1. 原子公式（Atom Formula）的三种形式

原子命题函数是公式，简称为原子公式，它有以下三种形式：

（1）R(t)：其中 R 是关系名，t 是元组变量，R(t)表示 t 是关系 R 中的一个元组。

（2）t[i]θs[j]：其中 t 和 s 是元组变量；θ 是算术比较运算符；t[i]和 s[j]分别是 t 的第 i 个分量和 s 的第 j 个分量；t[i]θs[j]表示元组 t 的第 i 个变量与元组 s 的第 j 个变量之间满足条件 θ。例如，t[3]＞s[4]表示元组 t 的第 3 个分量大于元组 s 的第 4 个分量。

（3）t[i]θc 或 cθt[i]：其中 c 是常量；t[i]θc 是元组 t 的第 i 个分量与常量 c 满足条件 θ。

例如，t[6]＜50 表示元组 t 的第 6 个分量小于 50。t[2]="数据库"表示元组 t 的第 2 个分量等于“数据库”。

2. 公式（Formulas）的递归定义

在定义关系演算操作时，要用到“自由元组变量”和“约束元组变量”的概念。若公式中的一个元组变量前有全称量词∀和存在量词∃符号，则称该变量为约束（Bound）变量，否则称为自由（Free）变量。公式递归定义如下：

（1）每个原子公式是一个公式，其中的元组变量是自由变量。

（2）如果 φ_1 和 φ_2 是公式，则¬ φ_1， $\varphi_1 \wedge \varphi_2$， $\varphi_1 \vee \varphi_2$ 也是公式，分别表示如下：

- 若 φ_1 为真，则¬ φ_1 为假。
- 如果 φ_1 和 φ_2 同时为真，则 $\varphi_1 \wedge \varphi_2$ 才为真，否则为假。
- 如果 φ_1 和 φ_2 中一个或同时为真，则 $\varphi_1 \vee \varphi_2$ 为真，仅当 φ_1 和 φ_2 同时为假时， $\varphi_1 \vee \varphi_2$ 才为假。

（3）若φ是公式，则∃t(φ)是公式。∃t(φ)表示若有一个 t 使φ为真，则∃t(φ)为真，否则∃t(φ)为假。

（4）若φ是公式，则∀t(φ)是公式。∀t(φ)表示对所有 t，都使φ为真，则∀t(φ)为真，否则∀t(φ)为假。

【提示】运算符的优先顺序为：算术比较运算符、∃、∀、¬、∧、∨。实现运算时可在公式中加括号，以改变上述优先顺序。

【实例 2-23】设有 R 关系（见表 2-5）和 S 关系（见表 2-6），写出元组演算表达式表示的关系。

表 2-5　关系 R

A	B	C
1	2	3
4	5	6
7	8	9

表 2-6　关系 S

A	B	C
1	2	3
3	4	6
5	6	9

（1）R1={t｜S(t)∧t[1]＞2}

（2）R2={t｜R(t)∧¬ S(t)}

（3）R3={t｜(∃u)(S(t)∧R(u)∧t(3)＜u[2])}

（4）R4={t｜(∀u)(R(t)∧S(u)∧t(3)＜u[1])}

（5）R5={t｜(∃u)(∃v)(R(t)∧S(v)∧u(1)＞v[2]∧t[1]=u[2]∧t[2]=v[3]∧t[3] =u[1])}

问题解析：关系 R1，R2，R3 的值分别如表 2-7、表 2-8、表 2-9 所示。

表 2-7　关系 R1

A	B	C
3	4	6
5	6	9

表 2-8　关系 R2

A	B	C
4	5	6
7	8	9

表 2-9　关系 R3

A	B	C
1	2	3
3	4	6

关系 R4，R5 的值分别如表 2-10、表 2-11 所示。

表 2-10　关系 R4

A	B	C
4	5	6
7	8	9

表 2-11　关系 R5

R.B	S.C	R.A
5	3	4
8	3	7
8	6	7
8	9	7

3. 关系代数的 5 种基本运算转换为元组演算表达式

关系代数表达式可以用元组演算表达式表示。由于任何一个关系代数表达式都可以用一种基本的关系运算组合表示，因此，只需给出 6 种基本的关系运算用元组演算表达式表示即可。

（1）并：$R \cup S=\{t \mid R(t) \vee S(t)\}$

（2）交：$R \cap S=\{t \mid R(t) \wedge S(t)\}$

（3）差：$R-S=\{t \mid R(t) \wedge \neg S(t)\}$

（4）笛卡尔积：假定关系 R 和 S 分别为 n 个属性和 m 个属性，则 R×S 后生成的新关系是 n+m 目关系，有 n+m 个属性。其元组演算表达式为：

$$R \times S=\{t^{(n+m)} \mid (\exists u^{(n)})(\exists v^{(n)})(R(u) \wedge S(v) \wedge t(1)=u[1] \wedge \cdots \wedge t[n]=u[n] \wedge t[n+1]=v[1] \wedge \cdots \wedge t[n+m]=v[m])\}$$

（5）选择：$\delta_F(R)=\{t \mid R(t) \wedge F'\}$

F′是 F 的等价表示形式。

（6）投影：$\Pi_{i1,i2,\cdots,ik}(R)=\{t^{(k)} \mid (\exists u)(R(u) \wedge t(1)=u[i_1] \wedge \cdots \wedge t[k]=u[i_k])\}$

【实例 2-24】设关系 R 和 S 都是具有两个属性列的关系，把关系代数表达式 $\Pi_{1,4}(\delta_{2=3}(R \times S))$ 转换成元组表达式。

问题解析：转换的过程需从里向外进行。

（1）$R \times S=\{t \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge t[1]=u[1] \wedge t[2]=u[2] \wedge t[3]=v[1] \wedge t[4]=v[2])\}$

（2）$\delta_{2=3}(R \times S)$，只要在上述公式后面加上“$\wedge t[2]=t[3]$”，即

$\{t \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge t[1]=u[1] \wedge t[2]=u[2] \wedge t[3]=v[1] \wedge t[4]=v[2] \wedge t[2]=t[3])\}$

（3）对于 $\Pi_{1,4}(\delta_{2=3}(R \times S))$，可得到下面的元组表达式：

$\{w \mid (\exists t)(\exists u)(\exists v)(R(u) \wedge S(v) \wedge t[1]=u[1] \wedge t[2]=u[2] \wedge t[3]=v[1] \wedge t[4]=v[2] \wedge t[2]=t[3] \wedge w[1]=t[1] \wedge w[2]=t[4])\}$

（4）再对上式化简，去掉元组变量 t，可得下式：

$$\{w \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge u[2]=v[1] \wedge w[1]=u[1] \wedge w[2]=v[1])\}$$

【实例 2-25】设有如下四个关系：

教师关系　T(Tno，Tname，Title)

课程关系　C(Cno，Cname，Tno)

学生关系　S(Sno，Sname，Age，Sex)

选课关系　SC(Sno，Cno，Score)

要求用元组关系演算表达式实现下列每个查询语句。

（1）检索学习课程号为 C2 课程的学生学号与成绩，则为：

$$\{t \mid (\exists u)(SC(u) \wedge u[2]= \text{'C2'} \wedge t[1]=u[1] \wedge t[2]=u[3])\}$$

（2）检索学习课程号为 C2 课程的学生学号与姓名，则为：

$$\{t \mid (\exists u)(\exists v)(S(u) \wedge SC(v) \wedge v[2]= \text{'C2'} \wedge u[1]=v[1] \wedge t[1]=u[1] \wedge t[2]=u[2])\}$$

（3）检索至少选修赵老师所授课程中一门课程的学生的学号与姓名，则为：

$$\{t \mid (\exists u)(\exists v)(\exists w)(\exists x)(S(u) \wedge SC(v) \wedge C(w) \wedge T(x) \wedge u[1]=v[1] \wedge v[2]=w[1] \wedge w[3] =x[1] \wedge x[2]= \text{'赵'} \wedge t[1]=u[1] \wedge t[2]=u[4])\}$$

（4）检索选修课程号为 C2 或 C4 的学生学号，则为：

$$\{t \mid (\exists u)(SC(u) \wedge u[2]= \text{'C2'} \vee u[2]= \text{'C4'}) \wedge t[1]=u[1]\}$$

（5）检索至少选修课程号为 C2 和 C4 的学生学号，则为：

$$\{t \mid (\exists u)(\exists v)(SC(u) \wedge SC(v) \wedge u[2]= \text{'C2'} \wedge u[2]= \text{'C4'} \wedge u[1]=v[1] \wedge t[1]=u[1])\}$$

（6）检索学习全部课程的学生姓名，则为：

$$\{t \mid (\exists u)(\forall v)(\exists w)(S(u) \wedge C(v) \wedge SC(w) \wedge u[1]=w[1] \wedge v[1]=w[2] \wedge t[1]=u[2])\}$$

【提示】关于元组关系演算语言和域关系演算语言的概念，将在配套的辅导书中介绍。

2.3.2　域关系演算

域关系演算（Domain Relational Calculus）是以元组变量的分量（域变量）作为谓词变元的基本对象。域演算表达式的定义类似于元组演算表达式的定义，所不同的是公式中的元组变量由域变量替代。域变量是表示域的变量，关系的属性名可以视为域变量。域演算表达式的一般形式为：

$$\{t_1,t_2,\cdots,t_k \mid P(t_1,t_2,\cdots,t_k)\}$$

其中，$t_1,t_2,\cdots,t_k$ 是域变量，即元组分量的变量，其变化范围是某个值域；$P(t_1,t_2,\cdots,t_k)$是由原子公式和运算符组成的公式。该公式的含义是使 $P(t_1,t_2,\cdots,t_k)$为真的那些域变量 $t_1,t_2,\cdots,t_k$ 组成的元组的集合。

1. 原子公式

原子命题函数是公式，简称为原子公式。有以下三种形式：

（1）$R(t_1,\cdots,t_i,\cdots,t_k)$：R 是关系名，$t_i$ 是元组变量 t 的第 i 个分量，$R(t_1,\cdots,t_i,\cdots,t_k)$表示以 $t_1,\cdots,t_i,\cdots,t_k$ 为分量的元组在关系 R 中。

（2）$t_i\theta C$ 或 $C\theta t_i$：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。

（3）$t_i\theta u_j$：t_i，u_j 是两个域变量，t_i 是元组变量 t 的第 i 个分量，u_j 是元组变量 u 的第 j 个分量，它们之间满足 θ 运算。

例如，$t_1 > u_4$ 表示元组 t 的第 1 个分量大于元组变量 u 的第 4 个分量。

2. 公式定义

若公式中的一个元组变量前有全称量词 ∀ 和存在量词 ∃ 符号，则称该变量为约束变量，否则

称为自由变量。公式可递归定义如下：

（1）原子公式是公式。

（2）如果φ_1和φ_2是公式，则$\neg\ \varphi_1$，$\varphi_1 \wedge \varphi_2$，$\varphi_1 \vee \varphi_2$也是公式。分别表示如下：

- 若φ_1为真，则$\neg\ \varphi_1$为假。
- 如果φ_1和φ_2同时为真，则$\varphi_1 \wedge \varphi_2$才为真，否则为假。
- 如果φ_1和φ_2中一个或同时为真，则$\varphi_1 \vee \varphi_2$为真，仅当φ_1和φ_2同时为假时，$\varphi_1 \vee \varphi_2$才为假。

（3）若φ是公式，则$\exists\, t_i(\varphi)$是公式。$\exists\, t_i(\varphi)$表示若有一个t_i使φ为真，则$\exists\, t_i(\varphi)$为真，否则$\exists\, t_i(\varphi)$为假。

（4）若$\varphi\ (t_1,\cdots,t_i,\cdots,t_k)$是公式，则$t_i(\varphi)$是公式。$t_i(\varphi)$表示对所有$t_i$使$\varphi(t_1,\cdots,t_i,\cdots,t_k)$为真，则$\forall\ t_i(\varphi)$为真，否则$\forall\ t_i(\varphi)$为假。

【实例 2-26】设有 R 关系（见表 2-12）和 S 关系（见表 2-13），写出 R1、R2 域表达式的值。

表 2-12　关系 R

A	B	C
1	2	3
4	5	6
7	8	9

表 2-13　关系 S

A	B	C
1	2	3
3	4	6
5	6	9

（1）R1={x,y,z｜R(xyz)∧x＜5∧y＞3}

（2）R2={x,y,z｜R(xyz)∨(S(xyz)∧y=4)}

问题解析：关系 R1，R2 的值分别如表 2-14 和表 2-15 所示。

表 2-14　关系 R1

x	y	z
4	5	6

表 2-15　关系 R2

x	y	z
1	2	3
4	5	6
7	8	9
3	4	6

【实例 2-27】设有关系 Student(Sno，Sname，Sex，Age，Dept，Addr)，写出下列域关系演算表达式。

（1）查询计算机系（CS）的全体学生，则为：

{Sno,Sname,Sex,Age,Dept,Addr｜Student(Sno,Sname,Sex,Age,Dept,Addr)∧Dept='CS'}

（2）查询不到 18 岁的女学生，则为：

{Sno,Sname,Sex,Age,Dept,Addr｜Student(Sno,Sname,Sex,Age,Dept,Addr)∧Sex='女'∧Age＜18}

（3）查询名字叫“赵建国”或“钱学斌”的学生的学号和所在系，则为：

{Sno,Dept｜∃Sno,Sname,Sex,Age,Dept,Addr(Student(Sno,Sname,Sex,Age,Dept,Addr)∧Sname='赵建国'∨Sname='钱学斌')}

【提示】上面介绍了关系代数、元组演算和域演算的概念和实例。虽然它们是三种不同的表示关系的方法，但可以证明，经安全约束后的三种关系运算的表达能力是等价的。

*§2.4 关系代数表达式的查询优化

上面讨论的关系代数和关系演算，其实质都是讨论关系代数表达式中的数据查询问题。数据查询是数据库系统中最基本、最常用和最复杂的数据操作。在数据库系统中，用户的查询是通过相应的查询语句提交给 DBMS 执行的。DBMS 在对一个关系表达式进行查询操作时，需要指出若干关系的操作步骤。那么，系统应该以什么样的顺序操作，才能做到既省时，又省空间，而且查询效率也比较高呢？这就是查询优化要研究的问题。虽然影响 DBMS 性能的因素很多，但一个数据库应用系统的查询性能直接影响到系统的推广和应用。因此，数据库系统性能和查询优化成为数据库应用领域备受关注的热点问题。

2.4.1 问题的提出

对于同一个查询要求，通常可以对应于多个不同形式但相互等价的关系代数表达式。相同的查询要求和结果存在着不同的实现策略，系统在执行这些查询策略时所付出的开销通常有很大的差别。在关系代数运算中，笛卡尔积和连接运算是最费时间的。若关系 R 有 m 个元组，关系 S 有 n 个元组，那么 R×S 就有 m×n 个元组。显然，当关系很大时，R 和 S 本身就要占较大的外存空间，由于内存的容量是有限的，只能把 R 和 S 的一部分元组读进内存，如何有效地执行笛卡尔积操作，花费较小的时间和空间，就有一个查询优化的策略问题。

【实例 2-28】设关系 R 和 S 都是二元关系，属性名分别为 A、B 和 C、D。设有一个查询可用关系代数表达式表示：

$$E_1=\prod_A(\delta_{B=C\wedge D='C2'}(S\times SC))$$

也可以把选择条件 D='C2'移到笛卡尔积中的关系 S 前面：

$$E_2=\prod_A(\delta_{B=C}(R\times\delta_{D='C2'}(S)))$$

还可以把选择条件 B=C 与笛卡尔积结合成等值连接形式：

$$E_3=\prod_A(R\underset{B=C}{\infty}\delta_{D='C2'}(S))$$

这 3 个关系代数表达式是等价的，但执行的效率却不大一样。显然，求 E_1、E_2、E_3 的大部分时间是花在连接操作上的。

对于 E_1，先做笛卡尔积，把 R 的每个元组与 S 的每个元组连接起来。在外存储器中，每个关系以文件形式存储。设关系 R 和 S 的元组个数都是 10000，外存的每个物理存储块可存放 5 个元组，那么关系 R 有 2000 块，S 也有 2000 块。而内存只给这个操作 100 块的内存空间。此时，执行笛卡尔积操作较好的方法是先让 R 的第一组 99 块数据装入内存，然后关系 S 逐块转入内存去做元组的连接；再把关系 R 的第二组 99 块数据装入内存，然后关系 S 逐块转入内存去做元组的连接，……直到 R 的所有数据都完成连接。

这样关系 R 每块只进内存一次，装入块数是 2000；而关系 S 的每块需要进内存（2000/99）次；装入内存的块数是（2000/99）×2000，因而执行 R×S 的总装入块数是：2000+（2000/99）×2000≈42400（块），若每秒装入内存 20 块，则需要约 35min，这里还没有考虑连接后产生的元组写入外存的时间。

对于 E_2 和 E_3，由于先做选择，所以速度快。设 S 中 D='99'的元组只有几个，因此关系的每块只需进内存一次。则关系 R 和 S 的总装入块数为 4000，约 3min，相当于求 E_1 花费时间的 1/10。

如果对关系 R 和 S 在属性 B、C、D 上建立索引，那么花费时间还要少得多。这种差别的原因是计算 E_1 时 S 的每个元组进内存多次，而计算 E_2 和 E_3 时，S 的每个元组只进内存一次。在计算 E_3 时又把笛卡尔积和选择操作合并成等值连接操作。

由此例可以看出，如何安排选择、投影和连接的顺序是个很重要的问题。

2.4.2　关系代数表达式的等价变换规则

关系代数表达式的优化是按照一定的规则，改变代数表达式中操作的次序和组合，使查询执行更高效。代数优化策略就是通过对关系代数表达式的等价变换来提高查询效率的。

所谓关系代数表达式等价，是指用相同的关系代替两个表达式中相应的关系所得到的结果是相同的。例如，两个关系代数表达式 E_1 和 E_2 是等价的，可记为 $E_1 \equiv E_2$。而所谓“结果相同”，是指两个相应的关系表具有相同的属性集合和相同的元组集合，但元组中属性顺序可以不一致。查询优化的关键是选择合理的等价表达方式，因此需要一套完整的表达式等价变换规则。下面介绍关系代数中常用的等价变换规则。

1. 连接、笛卡尔积的结合律

设 E_1、E_2、E_3 是关系代数表达式，F_1 和 F_2 是连接运算的条件，F_1 只涉及 E_1 和 E_2 的属性，F_2 只涉及 E_2 和 E_3 的属性，则有

（1）笛卡尔积结合律：$(E_1 \times E_2) \times E_3 \equiv E_1 \times (E_2 \times E_3)$

（2）条件连接结合律：$(E_1 \underset{F1}{\infty} E_2) \underset{F2}{\infty} E_3 \equiv E_1 \underset{F1}{\infty} (E_2 \underset{F2}{\infty} E_3)$

（3）自然连接结合律：$(E_1 \infty E_2) \infty E_3 \equiv E_1 \infty (E_2 \infty E_3)$

2. 连接、笛卡尔积交换律

设 E_1 和 E_2 是关系代数表达式，F 是连接运算的条件，则有

（1）笛卡尔积交换律：$E_1 \times E_2 \equiv E_2 \times E_1$

（2）条件连接交换律：$E_1 \underset{F}{\infty} E_2 \equiv E_2 \underset{F}{\infty} E_1$

（3）自然连接交换律：$E_1 \infty E_2 \equiv E_2 \infty E_1$

3. 投影的串接定律

$$\Pi_{A_1,A_2,\cdots,A_n}(\Pi_{B_1,B_2,\cdots,B_m}(E)) \equiv \Pi_{A_1,A_2,\cdots,A_n}(E)$$

其中，E 是关系代数表达式，A_i（i=1,2,⋯,n），B_j（j=1,2,⋯,m）是属性名且$\{A_1,A_2,\cdots,A_n\}$构成$\{B_1,B_2,\cdots,B_m\}$的子集。

4. 选择的串接定律

$$\delta_{F_1}(\delta_{F_2}(E)) \equiv \delta_{F_1 \wedge F_2}(E)$$

其中，E 是关系代数表达式，F_1、F_2 是选择条件。选择的串接律说明选择条件可以合并，这样一次就可检查全部条件。

5. 选择与投影操作的交换律

$$\delta_F \Pi_{A_1,A_2,\cdots,A_n}(E) \equiv \Pi_{A_1,A_2,\cdots,A_n}(\delta_F(E))$$

其中，选择条件 F 只涉及属性 $A_1,A_2,\cdots,A_n$。若 F 中有 $A_1,A_2,\cdots,A_n$ 的属性 $B_1,B_2,\cdots,B_m$，则有更一般的规则：

$$\Pi_{A_1,A_2,\cdots,A_n}(\delta_F(E)) \equiv \Pi_{A_1,A_2,\cdots,A_n}(\delta_F(\Pi_{A_1,A_2,\cdots,A_n,B_1,B_2,\cdots,B_m}(E)))$$

6. 选择与笛卡尔积的交换律

如果 F 中涉及的属性都是 E_1 中的属性，则

$$\delta_F(E_1 \times E_2) \equiv \delta_F(E1) \times E_2$$

如果 $F=F_1 \wedge F_2$，并且 F_1 只涉及 E_1 中的属性，F_2 只涉及 E_1 和 E_2 中的属性，则由上面的等价变换规则 1、4、6 可推出：

$$\delta_F(E_1 \times E_2) \equiv \delta_{F1}(E_1) \times (E_2)$$

若 F_1 只涉及 E_1 中的属性，F_2 只涉及 E_1 和 E_2 两者的属性，则仍有

$$\delta_F(E_1 \times E_2) \equiv \delta_{F2}(\delta_{F1}(E_1) \times (E_2))$$

它使部分选择在笛卡尔积前先做。

7. 选择与并的分配律

设 $E=E_1 \cup E_2$，E_1、E_2 有相同的属性名，则

$$\delta_F(E_1 \cup E_2) \equiv \delta_F(E_1) \cup \delta_F(E_2)$$

8. 选择与差运算的分配律

若 E_1 和 E_2 有相同的属性名，则

$$\delta_F(E_1 - E_2) \equiv \delta_F(E_1) - \delta_F(E_2)$$

9. 选择对自然连接的分配律

$$\delta_F(E_1 \infty E_2) \equiv \delta_F(E_1) \infty \delta_F(E_2)$$

F 只涉及 E_1 和 E_2 的公共属性。

10. 投影与笛卡尔积的分配律

设 E_1 和 E_2 是两个关系表达式，$A_1,A_2,\cdots,A_n$ 是 E_1 的属性，$B_1,B_2,\cdots,B_m$ 是 E_2 的属性，则

$$\Pi_{A_1,A_2,\cdots,A_n,B_1,B_2,\cdots,B_m}(E_1 \times E_2) \equiv \Pi_{A_1,A_2,\cdots,A_n}(E_1) \times \Pi_{B_1,B_2,\cdots,B_m}(E_2)$$

11. 投影与并的分配律

设 E_1 和 E_2 有相同的属性名，则

$$\Pi_{A_1,A_2,\cdots,A_n}(E_1 \cup E_2) \equiv \Pi_{A_1,A_2,\cdots,A_n}(E_1) \cup \Pi_{A_1,A_2,\cdots,A_n}(E_2)$$

12. 选择与连接操作的结合律

根据 F 连接的定义可得

$$\delta_F(E_1 \times E_2) \equiv E_1 \underset{F}{\infty} E_2$$

13. 并和交的交换率

$$E_1 \cup E_2 \equiv E_2 \cup E_1$$

$$E_1 \cap E_2 \equiv E_2 \cap E_1$$

14. 并和交的结合率

$$(E_1 \cup E_2) \cup E_3 \equiv E_1 \cap (E_2 \cup E_3)$$

$$(E_1 \cup E_2) \cap E_3 \equiv E_1 \cap (E_2 \cap E_3)$$

上面列出了常用的 14 条等价交换规则，其它变换规则可以查阅相关文献。

2.4.3 查询优化的一般策略

查询优化的总目标是选择有效的策略，求得给定的关系表达式的值。这个有效的策略就是逻辑层优化的一般策略和物理层优化的一般策略。

1. 逻辑层优化的一般策略

逻辑层优化一般不涉及具体的数据库物理结构，也不考虑系统提供的空间容量等其它物理因素，其主要目标是对用户提出的查询需求，选择一个较优的实现查询的逻辑方案。

关于查询的逻辑层优化包括两大内容：一是查询表达式的优化，二是操作步骤的划分。在这两方面的一些主要策略有：

（1）选择运算应尽可能先做，这在优化策略中是最重要、最基本的一条。因为选择运算一般能使计算的中间结果大大变小，并且使执行时间降低几个数量级，所以应该优先采用这一策略。

（2）如果在查询表达式中，某一子表达式的形式为一个笛卡尔积运算后紧接着执行某些选择运算，则将这两个运算合并为一个连接运算，可以使运行时间大为减少。

（3）表达式中的投影运算一般应尽可能早地执行。与选择运算一样，投影运算提前也可以减少中间运算结果的规模，因而也能提高查询效率。但应当注意的是，这种提前一般是需要谨慎处理的，因为可能有某些属性虽然在最后结果中不需要保留，但在执行指定的关系运算中却不可缺少。在这种情况下，为了尽可能早地执行投影运算，不得不把原投影运算分成两次或多次进行。

（4）如果在一个表达式中有某个子表达式重复出现，则应先将该子表达式算出的结果保存起来，以免重复计算。这种情况在利用视图进行查询时经常出现。

（5）如有若干投影和选择运算，并且它们都对同一个关系进行操作，则可以在扫描此关系的同时完成所有这些运算，以避免重复扫描关系。

（6）把投影运算同其前或其后的二元运算结合起来，没有必要为了去掉某些字段而扫描一遍关系。

【实例 2-29】设有关系代数表达式$\prod_A(\delta_{B=C\wedge D=15}(AB\times CD))$，根据策略（1），我们可以将原式修改为：

$$\prod_A(\delta_{B=C}(AB\times\delta_{D=15}(CD))) \qquad (2\text{-}1)$$

根据策略（2），我们又可进一步将前式优化为：

$$\prod_A(AB\underset{B=C}{\infty}\times\delta_{D=15}(CD)) \qquad (2\text{-}2)$$

但在运用策略（3）时，则不能简单地将投影运算提前。因为如果将关系 AB 中的 B 属性删去，则连接运算将无法进行，从而无法保证结果的等价性。正确的处理应将式（2-1）优化为式（2-2），最后按策略（5）和策略（6）进行运算。这一表达式求解过程应分为两个操作步骤：

第一步：计算$\prod_C(\delta_{D=15}(CD))$，并将中间结果送入外存；

第二步：将中间结果与关系 AB 做连接计算，并将结果投影后输出。

2. *物理层优化的一般策略*

物理层优化是在逻辑方案确定以后，如何实现逻辑方案，即如何实现每一个操作步骤的方式与存取路径的选择问题。这一层次的优化问题与数据库的物理组织密切相关。查询处理的物理层优化，主要包括各种关系代数操作的具体实现算法和索引选择等方面的考虑。

（1）关系代数操作的实现算法的研究，目前主要集中在笛卡尔积和连接运算上，因为这两种运算在多关系查询的场合是必不可少的，而且比较费时。在目前的 DBMS 中，实现联接操作和笛卡尔积的基本算法有块嵌套算法和排序合并算法两种，运用这些算法能使运行效率得到进一步的改善。

（2）在算法实现过程中，为进一步改善查询效率，一般要考虑索引、数据的存储分布等存取路径，这就要求用优化器去查找数据字典，获得当前数据库状态的信息。在执行连接前对关系进行适当的预处理，主要是先在连接的属性上建立索引和对关系进行排序，然后执行连接。由于在索引方式下算法的开销直接与关系的元组数、关系中索引属性值的分布有关，因此是否应该利用索引，常常需要结合具体的查询，对各种算法的开销进行比较后才能确定。这使得开销的估算成为许多 RDBMS 优化处理程序中的一个重要组成部分。

2.4.4 查询优化的具体实现

对于一个关系代数表达式，通过等价变换规则可以获得多个等价的表达式。表达式优化实际上就是优化操作顺序，因为不同的操作顺序会有不同的执行效率，所以确立关系代数表达式的选取规则，即从多个等价的表达式中选取查询效率高的表达式，是实现表达式优化的基本策略。

1. 优化原则

关系代数表达式优化的基本原则是减少查询处理的中间结果的大小，以此提高查询效率，缩短执行时间。尽管不同的关系数据库管理系统在解决查询优化时所采用的优化算法不尽相同，但一般都遵循下列启发式规则。

（1）将选择操作尽可能提前执行：对于有选择运算的表达式尽可能提前执行选择操作，既可减少读取外存的次数，又可缩短查询执行的时间，以最大程度减少中间结果的大小。

（2）将一连串的投影和选择操作合并进行：对于有若干投影和选择运算，并且它们都对同一个关系操作，则可在扫描此关系的同时完成所有的运算，以避免重复扫描关系。

（3）将投影同其前或其后的双目运算结合起来：不必为去掉某些字段而扫描一遍关系。

（4）将某些选择同要执行的笛卡尔积结合起来成为一个连接运算：两个关系的笛卡尔积的结果会得到一个很大的关系，加入选择转换为连接操作，以节省时间和空间上的开销。

（5）存储公共子表达式：如存在公共子表达式，则先计算一次公共子表达式并把结果写入中间文件，以避免重复计算。

2. 优化算法

查询优化是从查询的多个执行策略中进行合理选择的过程，查询优化是建立在对关系代数表达式的优化基础上的，而关系代数表达式的优化是由 DBMS 的 DML 编译器完成的。因此，查询优化的基本前提就是需要将关系代数表达式转换为某种内部表示。对一个关系代数表达式进行词法分析和语法分析后将其转换为某种内部表示，通常采用的内部表示是语法树。其实现的过程是先对一个关系代数表达式进行语法分析，将分析结果用树的形式表达出来。语法树具有如下特征：

- 树中的叶结点表示关系；
- 树中的非叶结点表示操作。

有了语法树之后，再使用关系表达式的等价变换公式对语法树进行优化变换，将原始语法树变换为标准语法树（优化语法树）。按照语法树的特征和查询优化的规则，语法树变换的基本思想是尽量使选择运算和投影运算靠近语法树的叶端。也就是说，使选择运算和投影运算得以先执行，从而减少开销。

利用等价变换规则和优化策略对关系代数表达式进行优化的步骤如下：

（1）将关系代数表达式转化为关系代数语法树。

（2）将关系代数语法树转换成优化树。

（3）将优化树转换成优化后的关系代数表达式。

（4）执行查询前对关系代数表达式中的连接关系进行适当的预处理。

【实例 2-30】设教学管理数据库中有学生关系模式、学生学习关系模式和课程关系模式：

S(Sno，Sname，Age，Sex)

SC(Sno，Cno，Grade)

C(Cno，Cname，Credit)

现有一个查询语句：检索选修了“数据库技术及应用”课程且成绩大于 90 分的所有学生的学

号和姓名。该查询语句的关系代数表达式可以表示如下：

$$\prod_{Sno,Sname}(\delta_{Cname='数据库技术及应用'\wedge Grade>90}(\delta_{S.Sno=SC.Sno\wedge SC.Cno=C.Cno}(S\times SC\times C)))$$

下面按查询优化的步骤进行优化。

（1）将关系代数表达式转化为关系代数语法树，如图 2-24 所示。

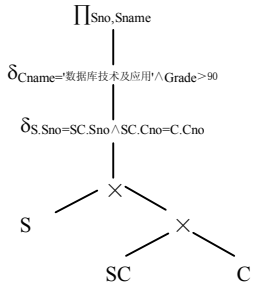


图 2-24　关系代数语法树

（2）将关系代数语法树转换成优化树，如图 2-25 所示。

（3）将优化树转换成优化后的关系代数表达式为：

$$\prod_{Sno,Sname}(S\infty(\delta_{Grade>90}(SC)\infty\delta_{Cname='数据库技术及应用'}(C)))$$

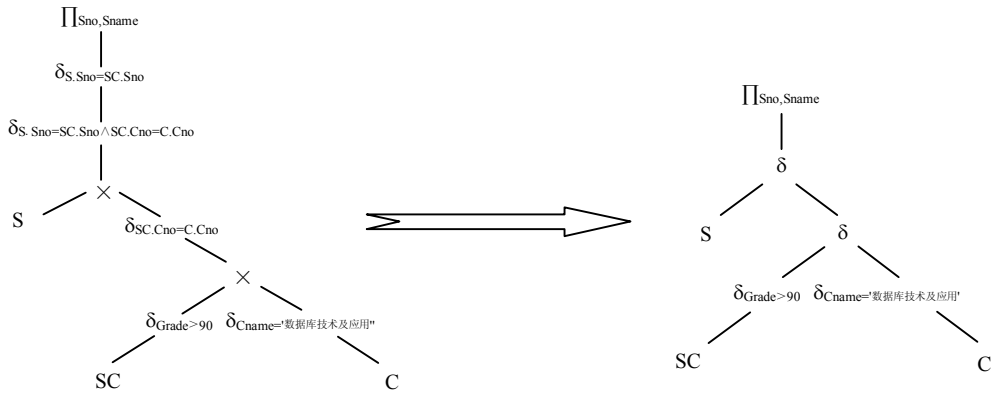


图 2-25　关系代数语法树

（4）执行查询前对关系代数表达式中的连接关系进行适当的预处理。

若 SC 和 C 关系中记录较多，则先对$\delta_{Grade>90}(SC)$和$\delta_{Cname='数据库技术及应用'}(C)$关系按 Cno 进行索引或排序将有助于减少两者连接的时间。同样，若 S 和$(\delta_{Grade>90}(SC)\infty\delta_{Cname='数据库技术及应用'}(C)$关系中记录较多，则先对这两个关系按 Sno 进行索引或排序也会有助于减少两者连接的时间。

【提示】查询优化的优点不仅在于用户不必考虑如何最好地表达查询以获得较好的效率，而且在于系统可以比用户程序的“优化”做得更好。因为优化器可以从数据字典中得到许多有用的信息，如当前的数据情况，而用户程序则得不到。优化器可以对各种策略进行比较，用户程序也做不到。

本章小结

1. 关系数据库系统是目前使用最广泛的数据库系统，因而也是本书的重点。在数据库发展的历史上，最重要的成果就是关系模型。关系理论的确立标志着关系数据库系统的基础研究已经接近顶峰，关系数据库系统已经占据了数据库系统的市场。

2. 关系运算理论是关系数据库查询语言的理论基础，只有掌握了关系运算理论，才能深刻理解查询语言的本质和熟练使用查询语言。将关系定义为元组的集合，但关系又有其特殊的性质。关系模型必须遵循实体完整性、引用完整性和用户定义完整性的规则。

3. 关系代数的五个基本操作以及四个组合操作是本章的重点，要求能进行两方面的运用：一是计算关系代数表达式的值；二是根据查询语句写出关系代数表达式的表示形式。

4. 关系演算是一种基于谓词演算的与关系代数等价的关系运算，要求了解关系演算表达式的语义，并能计算其值。

5. 查询优化是指系统对关系代数表达式要进行优化组合，这部分内容要求学生理解关系代数表达式的若干变换规则和优化的一般策略，并掌握关系代数表达式查询优化的步骤，同时希望读者在学习和实际运用中加以练习和体会。

习题二

一、选择题

1．数据的完整性是指数据的正确性、有效性和（　　）

A．可维护性　　B．独立性　　C．安全性　　D．相容性

2．关系中的“主关键字”不允许取空值是指（　　）约束规则。

A．实体完整性　　B．引用完整性　　C．用户定义完整性　　D．数据完整性

3．在关系模型中可以有 3 类完整性约束条件，任何关系必须满足其中的（　　）条件。

A．参照完整性、用户自定义完整性　　B．数据完整性、实体完整性

C．实体完整性、参照完整性　　D．动态完整性、实体完整性

4．下列对于关系的叙述中，（　　）是不正确的。

A．关系中的每个属性是不可分解的　　B．在关系中元组的顺序是无关紧要的

C．任意的一个二维表都是一个关系　　D．每一个关系只有一种记录类型

5．设关系 R 和 S 的元素分别是 3 和 4，关系 T 是 R 与 S 的笛卡尔积，即 T=R×S，则关系 T 的元素是（　　）。

A．7　　B．9　　C．12　　D．16

6．数据库的完整性是指数据库的正确性和相容性。下列叙述不是 DBMS 的完整性控制机制的是（　　）。

A．提供定义完整性约束

B．检查用户发出的操作请求是否违背了完整性约束条件

C．系统提供一定的方式让用户标识自己的名字或身份，用户进入系统时，由系统核对用户提供的身份标识

D．如果发现用户的操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性

7．对关系模型叙述错误的是（　　）。

A．建立在严格的数学理论、集合论和谓词演算公式基础之上

B．微机 DBMS 绝大部分采取关系数据模型

C．用二维表表示关系模型是其一大特点

D．不具有连接操作的 DBMS 也可以是关系数据库管理系统

8．有两个关系 R 和 S，分别包含 15 个和 10 个元组，则在 R∪S，R-S，R∩S 中，不可能出现的元组数目情况是（　　）。

A．15，5，10　　B．18，7，7　　C．21，11，4　　D．25，15，0

9．在下面两个关系中，职工号和部门号分别为职工关系和部门关系的主关键字。

职工（职工号，职工名，部门号，职务，工资）；

部门（部门号，部门名，部门人数，工资总额）。

在这两个关系的属性中，只有一个属性是外关键字。它是（　　）。

A.“职工”关系中的“职工号”　　B.“职工”关系中的“部门号”

C.“部门”关系中的“部门号”　　D.“部门”关系中的“部门名”

10．下列关系运算中，（　　）不要求关系 R 与关系 S 具有相同的属性个数。

A．R×S　　B．R∪S　　C．R∩S　　D．R－S

二、填空题

1．在关系模型中，实体和实体之间的联系是由单一的结构类型——关系表来表示的，关系模型的物理表示为_____。

2．数据库完整性的实现应该包括两个方面：一是系统要提供定义完整性约束条件的功能；二是提供_____的方法。

3．关系的数据操纵语言按照表达式查询方式可以分为两大类：关系代数和_____。

4．在关系模型中，若属性 A 是关系 R 的主关键字，则在 R 的任何元组中，属性 A 的取值都不允许为空，这种约束称为_____。

5．关系是一种规范化了的二维表格，其行称为_____，其列表示_____。

6．两个关系没有公共属性时，其自然连接操作表现为_____操作。

7．在域关系演算中，域变量的变化范围是_____。

8．根据关系模型的完整性规则，一个关系中的主键不能有_____个。

9．设关系 R 有 K_1 个元组，关系 S 有 K_2 个元组，则关系 R 和 S 的自然连接后的结果关系的元组数目是_____个。

10．设关系 R 有 K_1 个元组，关系 S 有 K_2 个元组。则关系 R 和 S 的笛卡尔积有_____个元组。

三、问答题

1．关系模型由哪几部分组成？

2．笛卡尔积、等值连接、自然连接三者之间有什么区别？

3．为什么关系中的元组没有先后顺序？

4．关系代数运算与关系演算运算有什么区别？

5．关系查询语言根据其理论基础的不同分为哪两类？

6．关系演算有哪两种？

7．为什么对关系代数表达式进行优化？

8．简述查询优化的优化策略。

9．关系数据语言有哪些类型和特点？

10．在参照完整性中，为什么外键属性的值也可以为空？什么情况下才可以为空？

四、应用题

1．设有教学管理数据库的三个关系模式：S(Sno，Sname，Age，Sex)、SC(Sno，Cno，Grade)、C(Cno，Cname，Teacher)，用关系代数表达式表示下列查询语句：

（1）查询“张三”老师所授课程的课程号、课程名。

（2）查询年龄大于 22 岁的男学生的学号与姓名。

（3）查询学号为“20121101”的学生所学课程的课程名与任课教师名。

（4）查询至少选修“李四”老师所授课程中一门课的女学生的姓名。

（5）查询“钱”同学不学的课程的课程号。

（6）查询全部学生都选修的课程的课程号与课程名。

（7）查询选修课程包含“张三”老师所授全部课程的学生的学号。

2．在教学管理数据库中，查询女同学选修课程的课程名和任课教师名，并且要求：

（1）写出该查询的关系代数表达式。

（2）画出该查询初始的关系代数表达式的语法树。

（3）使用优化算法，对语法树进行优化，并画出优化后的语法树。

（4）写出查询优化的关系代数表达式。

3．设教学管理数据库中有两个关系模式；Student(Sno，Sname，Age，Sex)；SC(Sno，Cno，Grade)。查询选修了课程 Cno='C110'的学生姓名，要求给出查询优化语法树。