第 3 章　关系数据库理论

在日常生活和科学技术领域中，我们经常会碰到各种各样的具体“关系”。人与人之间有父子、兄弟、师生等关系；两数之间有大于、等于、小于关系；电学中有电压、电阻与电流间的关系。宇宙万物之间存在着错综复杂的关系，这种关系正是各门学科所关注的问题。关系概念是对事物间多值依赖的一种描述，大家熟知的函数是关系的特例。有许多表述关系的数学模型，如在高等代数中的矩阵、离散数学中的图。集合理论为描述这种关系提供了“关系”的概念。集合理论中的关系本身也是一个集合，以具有某种联系的对象组合——“序组”为其成员。

换言之，在离散结构的表示中，关系不是通过揭示其内涵来描述事物间联系的，而是通过列举其外延（具有那种联系的对象组合全体）来描述这种联系。这使关系的研究可以方便地使用集合论的概念、运算及研究方法和研究成果。

3.1　关系模型概述

本节将从数据模型的三要素：数据结构、数据操作和完整性约束这三个方面详细介绍关系数据模型。

3.1.1　关系的数据结构

1. 关系的定义

域（Domain）：一组具有相同数据类型的值的集合。

例如：自然数、整数、实数、长度小于 20B 的字符串集合、{0,1}等都可以是域。

笛卡儿积（Cartesian Product）：给定一组域 $D_1,D_2,\cdots,D_n$，它们之中可以有相同的域。$D_1,D_2,\cdots,D_n$ 这 n 个域的笛卡儿积可以表示为：

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

其中每一个元素 $(d_1,d_2,\cdots,d_n)$ 称为一个 n 元组（n-tuple），或简称为**元组（tuple）**。元素中的每一个值 d_i 称为一个元组**分量（Component）**。

若 $D_i(i=1,2,\cdots,n)$ 为有限集，假设其基数（cardinal number）为 $m_i(i=1,2,\cdots,n)$，则 $D_1 \times D_2 \times \cdots \times D_n$ 的基数 M 为：

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

域的笛卡儿积可以用二维表直观地表示。表中的每一行对应一个元组，表中每一列的取值来自一个域。

【例 3-1】 给定三个域：

D_1 为学生姓名集合＝｛张山，李斯，王武｝；

D_2 为性别集合＝｛男，女｝；

D_3 为年龄集合＝｛19，20｝。

则 D_1、D_2、D_3 的笛卡儿积是所有可能的(姓名，性别，年龄)元组集合：

$D_1 \times D_2 \times D_3$ ＝｛(张山，男，19)，(张山，男，20)，(张山，女，19)，(张山，女，20)，
(李斯，男，19)，(李斯，男，20)，(李斯，女，19)，(李斯，女，20)，
(王武，男，19)，(王武，男，20)，(王武，女，19)，(王武，女，20)｝

其中“(张山，男，19)”、“(李斯，男，20)”等都是元组。“张山”、“男”等都是元组的分量。

该笛卡儿积的基数为 3×2×2＝12，也就是说 $D_1 \times D_2 \times D_3$ 一共有 12 个元组。$D_1 \times D_2 \times D_3$ 可表示成二维表的形式，如表 3-1 所示。

表 3-1 $D_1 \times D_2 \times D_3$ 的二维表表示

学生	性别	年龄	学生	性别	年龄
张山	男	19	李斯	女	19
张山	男	20	李斯	女	20
张山	女	19	王武	男	19
张山	女	20	王武	男	20
李斯	男	19	王武	女	19
李斯	男	20	王武	女	20

由于一个学生只有一个姓名、性别和年龄，若用一个元组表示一个学生姓名、性别和年龄，则笛卡儿积中的许多元组是没有实际意义的(在这里不考虑有重名的情况)。从笛卡儿积中取出那些有一定含义的元组构成一个集合，我们称为关系，也即关系是笛卡儿积的某个有意义的子集。如表 3-2 所示，该二维表可表示域 D_1 中每个学生的基本情况。

表 3-2 学生关系的二维表表示

姓名	性别	年龄	姓名	性别	年龄
张山	男	19	王武	男	20
李斯	女	20			

至此，可以给出关系的定义。

关系(Relation)：$D_1 \times D_2 \times \cdots \times D_n$ 中某个有一定语义的子集叫做在域 D_1、D_2、…、D_n 上的关系，表示为 $R(D_1, D_2, \cdots, D_n)$。其中，R 为关系的名字，n 是关系的目或度(Degree)。

从表 3-2 中可以看到，关系模型的数据结构，即关系，可以表示一个学生实体的信息。数据模型的数据结构还应能描述实体以及实体之间的联系，那么关系模型如何表示实体以及实体之间的联系呢？

【例 3-2】 给出三个域：

D_1 为导师姓名集合＝{张明，李良}；

D_2 为专业名称集合＝{计算机应用技术，系统工程}；

D_3 为研究生姓名集合＝{王敏，刘勇，李新}

$D_1 \times D_2 \times D_3$ 是个三元组集合，元组个数（基数）为 $3 \times 2 \times 2$，是所有可能的（导师姓名，专业名称，学生姓名）的元组集合。

$D_1 \times D_2 \times D_3$ 中许多元组是没有意义的。因为在学校中，一名研究生只有一个导师，研究某一个专业方向（导师与研究生是一对多的联系）。

$D_1 \times D_2 \times D_3$ 的一个子集可表示导师与研究生的指导关系。这个关系可用表 3-3 所示的二维表来表示。

表 3-3 导师与研究生的指导关系

导师姓名	专业名称	学生姓名	导师姓名	专业名称	学生姓名
张明	计算机应用技术	王敏	李良	系统工程	刘勇
张明	计算机应用技术	李新			

从表 3-3 中可以看到，关系可以表示导师实体和学生实体之间的指导关系。由此可见，关系模型的数据结构非常简单，只包含单一的数据结构——关系。关系既可以表示概念模型中的实体，也可以用来描述实体间的各种联系。数据结构简单正是关系模型最大的优点。

2. 关系模型的相关概念

概念模型中实体的属性、域、实体型、实体集在关系模型中分别用关系的属性、域、关系模式、关系实例来表示。

属性（Attribute）：关系所对应的域命名为属性，属性用属性名表示。在同一关系中，属性名不能相同。如表 3-3 中的属性分别是导师姓名、专业名称和研究生姓名。

若关系对应一个实体，关系的属性就是所要描述的实体的属性，即实体所对应的事物对象的特征。如表 3-2 中，“学生”关系可用姓名、性别和年龄等属性描述。“图书”关系可用图书编号、书名、作者、出版社、价格等属性来描述。

域：属性的取值范围。不同的属性可以有相同的域。如表 3-3 中，“导师姓名”和“研究生姓名”这两个属性的域都可以是由若干字符组成的字符串的集合，但属性名称不能一样。

在关系数据模型中，一般要求所有的域都是原子数据的集合。这种限制被称为第一范式条件（参见第 5 章 5.2 节）。

关系模式（Relation Schema）：关系的描述称为关系模式。关系模式必须指出关系的结构，即它由哪些属性构成，这些属性来自哪些域，以及属性与域之间的映像。

关系模式可以形式化地表示为：

$$R(U, D, \mathrm{Dom}, F)$$

其中：R 为关系名，U 为组成关系 R 的属性集合，D 为属性组 U 中属性来自的域，Dom 为

属性向域的映像的集合，F 为属性间数据的依赖关系集合。

关系模式通常可以表示为：

$$R(D_1/A_1, D_2/A_2, \cdots, D_n/A_n)$$

或

$$R(A_1, A_2, \cdots, A_n)$$

$A_1, A_2, \cdots, A_n$ 为属性名，域名及属性向域的映像常常直接说明为属性的类型和长度。

对于表 3-2 中的关系可定义关系名为“学生”，则关系模式表示为：

学生(姓名，性别，年龄)

关系实例(Relation Instance)：一个给定关系的某一时刻的元组的集合，即当前关系的值。关系 R 的实例记为 $r(R)$。

关系模式是关系的型的描述，是静态的、稳定的。关系实例(值)是关系的“当前”元组的集合，是动态的、随时间不断变化的，其变化通过关系的元组的改变表现出来。如表 3-2、表 3-3 的内容就分别是两个关系的一个实例。

在实际使用中，人们常常把关系模式和关系实例都笼统地称为关系，这不难从上下文中加以区别。

候选键(Candidate Key，候选码，简称为键或码)：若关系中的某一属性或属性集能唯一标识一个元组，而其任意一个真子集无此性质，则称该属性或属性集为关系的候选键。也就是说，候选键是能唯一标识一个元组的最小属性集。

候选键可以保证关系实例上任何两个元组的值在候选键的属性(集)上取值不同。需要注意的是，构成候选键的属性(集)的值对于关系的所有实例都具有唯一性，而不是只针对某一个实例。

通常在关系模式中在构成候选键的属性(集)下面画上下划线，来表明它是键的组成部分。如表 3-2 所对应的“学生”关系可写成如下形式，其中“姓名”是候选键。

学生(<u>姓名</u>，性别，年龄)

每一个关系都至少存在一个候选键。若一个关系有多个候选键，可选择其中的一个作为**主键(Primary Key)**。在数据库应用系统中，主键的选择会影响某些实现问题，例如索引文件的建立(参见第 6 章)。

包含候选键的属性集称为**超键(Superkey)**。超键能唯一标识元组，但不具有最小化性质。

若关系只有一个候选键，且这个候选键包含了关系的所有属性，称该候选键为**全键(All-key)**。

主属性(Prime Attribute)：构成候选键的每个属性称为主属性。不包含在任何候选键中的属性称为**非主属性(Non-prime Attribute)**或**非码属性(Non-key Attribute)**。

外键(Foreign Key，外码)：若关系 R 的一个属性(集) F 与关系 S 的主键 K_s 对应，即关系 R 中的某个元组的 F 上的分量值也是关系 S 中某个元组的 K_s 上的分量值，则称该属性(集) F 为关系 R 的外键。

其中，关系 R 为参照关系(Referencing Relation，引用关系)，关系 S 为被参照关系

(Referenced Relation)或目标关系(Target Relation)。关系 R 和关系 S 可以是同一个关系。目标关系的主键 K_s 和参照关系 R 的外码 F 的命名可以不同,但必须定义在同一(或同一组)域上。

【例 3-3】 学生实体和课程实体分别用关系"学生"和"课程"来表示,它们之间的联系用关系"选课"来表示。

学生(学号,姓名,性别,出生时间,所在系)

课程(课程号,课程名,先修课程号)

选课(学号,课程号,成绩)

"学生"关系的候选键(主键)为"学号";"课程"关系的候选键(主键)为"课程号","课程"关系的"先修课程号"引用了"课程"关系的"课程号"属性,是"课程"关系的外键,被引用关系和引用关系是同一个关系。"选课"关系中的"学号"和"课程号"共同构成关系的候选键(主键),又分别是"选课"关系的外键,它们分别对应"学生"关系和"课程"关系的主键。

3. 关系的性质

在集合论中,关系可以是无限集合;而且,关系中每个元组是"序组",即元组中的分量有前后顺序,元组$(d_1,d_2,\cdots,d_n)$和$(d_2,d_1,\cdots,d_n)$不同。当关系作为关系数据模型的数据结构时,需要给予如下的限定和扩充:

(1) 限定关系数据模型中的关系必须是有限集合。无限关系在数据库系统中是没有意义的。

(2) 通过为关系的每个属性附加一个属性名来取消元组分量的有序性,即关系$R(A_1,\cdots,A_i,A_j,\cdots,A_n)$与关系$R(A_1,\cdots,A_j,A_i,\cdots,A_n)$为同一关系模式,其对应的元组$(d_1,d_2,\cdots,d_i,d_j,\cdots,d_n)$和$(d_1,d_2,\cdots,d_j,d_i,\cdots,d_n)$相同。

归纳起来,关系具有如下一些性质:

(1) 元组个数有限性。

(2) 属性名唯一性,即关系中不能有重名属性。

(3) 属性的次序无关性,即属性列的次序可以任意交换。

(4) 元组的唯一性,即关系中不能出现完全相同的两个元组。

(5) 元组的次序无关性,即元组的顺序可以任意交换。关系是元组的集合,而不是元组的列表。

(6) 元组分量的原子性,即每个分量都必须是不可分割的数据项。

(7) 分量值域同一性。每一属性列中的元组分量是同一数据类型,来自同一个域。

4. 关系与二维表

从用户的角度来看,关系模型的数据结构就是一张二维表,表中的每行对应一个元组,表中的每列对应一个取值域。

在数据库关系模型中,经常将关系与一张二维表等同起来。二维表的表头由各属性名构成,每一列表示一个属性,每一行表示一个元组,所有行的集合构成了关系 R 的

实例。

关系是一种抽象的对象，表则是一种具体的图形，关系这种抽象对象能在平面上以表的形式简单地表示出来，使得关系模型容易理解和使用，这是关系模型的一个巨大优势。

但表和关系实际上是不同的，注意加以区分，有助于对关系的理解。表和关系的不同之处具体体现在以下几方面：

(1) 表中各列从左到右是有序的，关系中属性的次序是任意的；

(2) 表中各行从上到下是有序的，关系中元组的次序是任意的；

(3) 表中可能包含重复的行，关系中不能有相同的元组；

(4) 表中至少含有一个列，但可存在不含任何属性的关系，相当于空集合；

(5) 表中允许包含空行(例在 SQL 中)，而关系中不允许；

(6) 表是“平面的”或是“二维的”，而关系却是“n 维的”，是 n 个域上的一个 n 元组的集合。

一般情况下，理论研究侧重关系的概念，而实际的 DBMS、数据库语言等更多地支持表的概念，而不是关系。

例如，在数据库语言和宿主语言中支持表的概念，会有“取出第 N 个元组的操作”，查询结果的呈现涉及元组的有序排列(游标、ORDER BY)等问题。在实际的数据库系统中，定义关系模型时，属性是没有顺序的，但定义后，在系统中就有了顺序。但这些问题不属于关系模型本身的问题，而是关系系统的实现问题。

5. 关系数据库(Relation DataBase，RDB)

在关系模型中，数据库是由一个或多个关系组成的。数据库的关系模式集合叫做关系数据库模式(relational database schema)，或者简称为数据库模式(database schema)，是对关系数据库的型的描述，包括若干域的定义以及在这些域上定义的若干关系模式。关系数据库的实例(值)是这些关系模式在某一时刻对应关系实例的集合。

在某一应用领域中，描述所有实体集及实体之间联系所形成的关系的集合就构成了一个关系数据库。

3.1.2 关系的完整性约束

关系模型的完整性约束是关系模型对于存储在数据库中的数据具有的约束能力，也就是关系的值随着时间变化应该满足的一些约束条件。这些约束条件实际上是现实世界对关系数据的语义要求。关系数据库中的任何关系在任何时刻都需要满足这些语义。

关系模型中有三类完整性约束：实体完整性、参照完整性和用户定义的完整性。实体完整性和参照完整性是关系模型必须满足的完整性约束条件，被称为关系的两个不变性，一般由关系型数据库管理系统(RDBMS)自动支持。用户定义的完整性是应用领域需要遵循的约束条件，体现了具体应用领域中的语义约束，一般在定义关系模式时由用户自己定义。

1. 实体完整性(Entity Integrity)

在关系模型中,实体用关系来描述,关系是元组的集合。为使候选键能唯一标识一个元组,需对构成候选键的每个主属性进行如下约束。

实体完整性约束规则:若属性 A 是关系 R 的主属性,则属性 A 的值不能为空值。

属性值为空的含义是该属性值"不知道"、"不清楚"、"不存在"或"无意义"等。在关系模型中使用空缺符 NULL 来表示。

例如,在例 3-3 中,"学生"关系的主属性"学号"、"课程"关系的主属性"课程号"不能为空;选课(学号,课程号,成绩)关系中主属性"学号"、"课程号"都不能为空。

这条约束规则的实质是体现了关系模型中的键约束特性,主属性为空,说明存在某个不可标识的元组,即存在不可区分的实体值。

对于实体完整性约束规则的使用做如下几点说明:

(1) 实体完整性是针对系统中定义的基本关系(存储的关系表)而言的,并不对查询的结果关系(临时表)、视图(参见第 4 章 4.5 节)等进行约束。

(2) 如果关系的候选键由若干属性组成,则所有构成候选键的属性即主属性都不能为空。

2. 参照完整性(Referential Integrity)

现实世界中实体之间往往存在某种联系,在关系模型中实体以及实体间的联系都是用关系来描述的。这样就自然存在着关系与关系之间的参照,关系之间的参照一般通过外键来描述,并遵循如下约束规则。

参照完整性约束规则:若属性(或属性集)F 是关系 R 的外键,它与关系 S 的主键 K_s 对应,则对于 R 中元组在 F 上的取值只能有两种可能:或者取空值,或者等于 S 中某个元组的 K_s 值。

【例 3-4】 若学生实体和专业实体可以用下面的关系来表示:

学生(学号,姓名,性别,专业号,出生时间)

专业(专业号,专业名)

若属性"专业号"是"学生"关系的外键,又是"专业"关系的主键,则"学生"关系中每个元组的"专业号"属性值只能是下面两种情况:

(1) 空值,表示尚未给学生分配专业;

(2) 非空值,这时元组在"专业号"属性上的元组分量值必须是"专业"关系中某个元组的"专业号"值,表示该学生只能就读某个存在的专业。

这条约束规则的实质是不允许引用不存在的实体,在某个关系中出现的值也必须在另外一个相关的关系中出现。

对于参照完整性规则的使用做如下几点说明:

(1) 关系 R 和 S 可以是同一关系,表明同一关系中不同元组之间的参照关系;

例如,在例 3-3 中,在"课程(课程号,课程名,先修课程号)"关系中,如果规定每门课程的直接先修课程只能是所开设课程中的一门,那么"先修课程号"就是"课程"关系中的

外键，其对应的主键为本关系的主键“课程号”。

（2）外键并不一定要与相应的主键同名，如例 3-3 中的“课程号”和“先修课程号”，但必须定义在相同的值域上。在实际应用中，为便于识别，当外键与相应的主键属于不同关系时，往往给它们取相同的名字。

（3）若外键 F 为一属性集且为空值，则 F 中的每个属性值均为空值。F 是否能为空值，应视具体问题而定。

例如，在例 3-3 中，“选课”关系中的“学号”和“课程号”分别是该关系的外键，按照参照完整性约束规则，属性值可以为空值或被参照关系“学生”和“课程”关系中某个元组的主键值。但由于“学号”和“课程号”又分别是“选课”关系的主属性，按照实体完整性约束规则，它们均不能取空值，只能取相应被参照关系中已经存在的某个元组的主键值。

3. 用户定义的完整性（User-defined Integrity）

任何 RDBMS 都应该支持实体完整性和参照完整性，这是关系模型所要求的。除此之外，对用户定义的针对某一具体关系数据库提出的约束条件，RDBMS 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。

不同的关系数据库根据其应用环境的不同，往往还需要一些特殊的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求。

比如，每个属性都有一个类型约束，使得该属性的每个取值都只能是该类型，存在着“只能取整数”、“字符串长度最大为 30”等域约束条件。还可能对属性值的取值范围进行约束，如“学生考试成绩在 0～100 之间”、“在职职工的年龄不能大于 60 岁”等都是针对具体关系提出的约束条件。

关系的完整性约束是为了防止关系数据库中存在不符合语义的数据，也就是防止数据库中存在不正确的数据。目前商用的 RDBMS 都支持完整性控制，DBMS 中定义和检查完整性约束条件并对违反完整性约束条件的操作进行处理的机制称为完整性子系统。

在早期的 RDBMS 中，没有提供定义和检验这些完整性的机制，因此需要应用开发人员在应用系统的程序中进行完整性检查。

例如，对于例 3-3 中的“选课”关系，每插入一条记录必须在应用程序中写一段程序来检查其中的“学号”、“课程号”是否与“学生”、“课程”关系的相应属性一致。

3.1.3 关系操作

关系模型给出了关系操作的能力说明，早期的关系操作能力通常用代数方式或逻辑方式来表示，分别称为关系代数（Relational Algebra）和关系演算（Relational Calculus）。

关系代数是用对关系的运算来表达查询要求的。用户可以指定基本的检索请求，检索的结果是一个新的关系，这个新关系可能由一个或多个关系所构成。可以使用同样的代数操作进一步操纵这些新关系。关系代数操作的一个序列构成一个关系代数表达式，其结果还是一个关系，它表示一个数据库查询（或检索请求）的结果。

关系代数的重要性体现在以下几个方面：

(1) 它为关系模型操作提供了一个形式化的基础，因此经常被用作衡量另一种关系模型语言表达能力的尺度。当一种语言至少拥有代数的作用，即它的表达式允许通过代数的形式来定义每一个关系时，我们就说该语言是关系完备的。

(2) 关系代数被用在关系数据库管理系统(RDBMS)中，作为实现和优化查询的基础(参见第7章)。

(3) 面向RDBMS的SQL标准查询语言中结合了关系代数中的一些概念。

与关系代数不同，关系演算为关系查询提供了一个更高级的描述性表示法。关系演算是用查询得到的元组应满足的谓词条件来表达查询要求。关系演算表达式创建了一个新关系，这个新关系以变量形式指定。而变量的取值范围为数据库关系中的元组(元组演算)或属性(域演算)。在演算表达式中，对指定如何检索查询结果的操作没有次序上的要求，演算表达式只指定了结果中应当包含什么信息。关系演算的重要性体现在其有坚实的数理逻辑基础，同时面向RDBMS的SQL标准查询语言也以元组关系演算作为其部分基础。

本章所讨论的关系代数、元组关系演算和域关系演算均是抽象的查询语言，与具体的RDBMS中实现的实际语言并不完全一样，不对RDBMS语言给出具体的语法要求，即不同的RDBMS可以定义和开发不同的语言来实现这些操作。但关系代数和关系演算能用作评估实际系统中查询语言能力的标准和基础。实际的查询语言除了提供关系代数语言和关系演算语言所表达的功能外，还提供许多附加的功能。

曾经出现的一些RDBMS实际查询语言有：

(1) ISBL(Information System Base Language)是IBM公司英格兰底特律科学中心在1976年研制出来的，用在一个实验系统PRTV(Peterlee Relational Test Vehicle)上。ISBL语言的每个查询语句都近似一个关系代数表达式。

(2) QUEL(Query Language)是美国伯克利加州大学研制的关系数据库系统INGRES使用的查询语言。QUEL语言参照E. F. Codd提出的APLHA元组演算语言研制出来的，是一种基于元组关系演算的并具有完善的数据定义、检索、更新等功能的数据库语言。

(3) QBE(Query By Example)是M. M. Zloof提出的，在约克镇IBM高级研究实验室为图形终端用户设计的一种域演算语言，1978年在IBM 370上实现。QBE属于人机交互语言，使用方便。其思想已渗入到许多DBMS中。

关系模型与其他数据模型相比，最具特色的是关系数据操作语言。关系操作语言灵活方便，表达能力和功能都非常强大。

目前使用的是一种结构化的SQL查询语言，不仅具有丰富的查询功能，而且具有数据定义和控制功能。它具有语言简洁，易学易用的特点，是关系数据库的标准语言和主流语言。

我们将在第3.2节讨论关系代数的操作，在第3.3节讨论关系演算的内容，在第4章讨论SQL语言的功能。

3.2 关系代数

关系代数是一种抽象的查询语言，它用对关系的运算来表达查询要求。一门代数总是由一些操作运算符和一些原子操作数组成。比如，算术代数中的原子操作数是像常量5和变量 x 这样的操作数，而加(+)、减(−)、乘(×)、除(÷)是其中的操作运算符。任何一门代数都允许把运算符用在原子操作数或者是其他代数表达式上构造表达式，括号一般被用来组合操作数和运算符。关系代数也是一门代数，它基于一组为数不多的以关系为操作对象的运算符。每个运算符对一个或多个关系进行运算，产生的结果是另外一个关系，数据查询就是一个包含这些运算符的表达式。表达式的结果是关系，也就是这个查询的答案。

关系代数的原子操作数包括代表关系的变量和代表关系实例的常量。

关系代数的运算符可分为两类：传统的集合运算和专门的关系运算。传统的集合运算将关系看成元组的集合，其运算是从关系的“水平”方向即元组的角度来进行的。而专门的关系运算不仅涉及元组，而且涉及属性列。在关系代数表达式中还使用比较运算符和逻辑运算符来辅助专门的关系运算。

传统的集合运算主要包括并、差、交、广义笛卡儿积。

专门的关系运算主要包括投影、选择、连接、除。

3.2.1 传统的集合运算

传统的集合运算是二目运算，主要包括并、差、交、广义笛卡儿积4种运算。

设两个关系 R 和 S 具有相同的类型，或 R 和 S 是相容关系，即关系 R 和 S 具有相同的目，且相应的属性取自同一个域，则定义并、差、交和广义笛卡儿积运算如下：

1. 并(Union)运算

关系 R 与 S 的并是一个与 R、S 相容的关系，且其元组由属于 R 或 S 的元组组成，表示为 $R \cup S$。

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

并运算可用于实现两个关系的合并，建立多关系间的查询和定位，实现元组的插入操作。

2. 差(Difference)运算

关系 R 与 S 的差是一个与 R、S 相容的关系，且其元组由属于 R 但不属于 S 的元组组成，表示为 $R-S$。

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

注意：$S-R$ 与 $R-S$ 是不同的，$S-R$ 表示由只在 S 中出现而不在 R 中出现的元组构成的关系。