第 3 章　关系数据库理论

关系数据模型以集合论中“关系”这个离散数学概念为基础，集合论中的“关系”描述的就是日常生活和科学技术等领域中各种各样的具体关系。关系是对客观世界中事物间相互联系的一种描述。例如，人与人之间有血缘关系、朋友关系；两个数之间有等于或不等于关系；电阻、电压与电流间存在欧姆定律等。宇宙万物之间存在着错综复杂的各种关系，各门学科提出了许多表述关系的数学模型，如人们熟知的函数、矩阵和图等。

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（具有某种联系的对象组合全体）来描述这种联系。这使得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。集合论中的“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。

3.1　关系模型概述

本节将从数据模型的组成要素：数据结构、数据操作和数据的完整性约束三方面详细介绍关系数据模型。

3.1.1　关系的数据结构

关系模型用关系结构来描述实体以及实体间的联系。

1. 关系结构

关系的概念建立在笛卡儿积的概念基础上。

笛卡儿积(Cartesian product)：给定一组域 $D_1,D_2,\cdots,D_n$，这 n 个域上的有序对的集合为这 n 个域的笛卡儿积，可以表示为

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

域(domain)：是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。这一组域中可以有相同的域。

每一个元素$(d_1,d_2,\cdots,d_n)$称为一个 n 元组(n-tuple)，简称**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(cardinality)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 为学生姓名的集合={张山，李斯，王武}

D_2 为性别的集合={男,女}

D_3 为年龄的集合={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)"和"(李斯,男,20)"等都是元组;"张山"和"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

姓　名	性　别	年　龄	姓　名	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(这里不考虑有重名的情况)。从笛卡儿积中取出那些有一定语义的元组构成一个集合,称之为关系,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
张山	男	19
李斯	女	20
王武	男	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称为在域 $D_1, D_2, \cdots, D_n$ 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 为导师姓名的集合={张明,李良}

D_2 为专业名称的集合={军事指挥学,软件工程}

D_3 为研究生的姓名集合={王敏,刘勇,李新}

$D_1 \times D_2 \times D_3$ 是一个三元组集合,元组个数(基数)为 2×2×3,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
张明	军事指挥学	王敏
张明	军事指挥学	李新
李良	软件工程	刘勇

由此可见，定义在笛卡儿积上的关系，作为关系模型的数据结构，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。关系模型的数据结构简单，只包含单一的数据结构——关系，并可用二维表来表示，这正是关系模型的优势所在。

2. 关系模式

概念模型中有关实体的描述反映在关系模式的定义上。实体的属性、域、实体型、实体集（值）分别用关系的属性、域、关系模式、关系实例来表示。

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，即关系由哪些属性构成、这些属性来自哪些域、属性与域的映像以及属性之间的数据依赖。

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 为关系 R 的属性集合，D 为属性组 U 中属性来自域的集合，Dom 为属性向域的映像的集合，F 为属性间数据依赖的集合。

下面对关系模式的这 5 个要素进行解释。

关系名：标识一个关系。因为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，所以关系的名称通常与对应实体的名称或者实体间联系的名称相一致。如表 3-2 中的“学生”关系，表 3-3 中的“指导”关系。

属性：对关系中元组分量的描述，用属性名表示，与定义关系的一组域对应。其语义就是关系所描述的实体的属性或者实体间联系的属性，所以在同一关系中属性名不能相同。如表 3-2 中的学生关系可用学生实体的姓名、性别和年龄等属性描述，表 3-3 中的指导关系可用学生实体和导师实体的姓名以及指导的专业名称等属性描述。

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，可能会出现导师和研究生的姓名相同的情况。属性的域可以相同，但属性名称不能相同。

在关系数据模型中，一般要求所有的属性域都是原子数据的集合，这种限制称为第一范式条件（范式的概念见 5.2 节）。因此，关系的属性只能是简单属性，也就是原子属性，不能是复合属性，属性值不能在系统里被划分为若干子部分；属性也不能是多值属性，不能在同一元组的同一属性上有多个值。

属性向域的映像：常常直接定义为属性的类型和长度。例如，可定义表 3-2 中学生关系的姓名属性为字符串类型，长度为 10 个字符；性别属性为由“男”和“女”两个汉字组成的集合类型；年龄属性为整数类型。

属性间的数据依赖：关系的属性与属性之间的一种约束关系，反映的是现实世界事物特征间的一种依赖关系，是数据内在的性质，是语义的体现。

例如,将表3-3中给出的导师姓名、专业名称和研究生姓名三个域的笛卡儿积上的指导关系,定义为表3-4中的“指导1”关系,该关系的语义是一个导师只能在一个方向上指导研究生,可同时指导多名学生。若要求一个导师只能在一个专业方向指导一名研究生,则导师实体与研究生实体的指导关系可定义为表3-4中的“指导2”关系,只能包含2个元组。这两个关系的区别就在于关系属性间约束不同,语义不同。

由此可见,属性间的数据依赖是关系模式的要素之一。有关数据依赖的内容将在第5章关系模式的规范化设计部分做进一步介绍。

表3-4 具有不同数据依赖的导师和研究生的指导关系

(a) 指导1

导师姓名	专业名称	学生姓名
张明	军事指挥学	王敏
张明	军事指挥学	李新
李良	软件工程	刘勇

(b) 指导2

导师姓名	专业名称	学生姓名
张明	军事指挥学	王敏
李良	软件工程	刘勇

根据讨论问题的关注点的不同,关系模式通常可以简化表示为$R(U)$或$R(U,F)$等形式,或直接表示为

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

其中,$A_1,A_2,\cdots,A_n$为属性名。

对于表3-2中的学生关系,其关系模式可表示为

学生(姓名,性别,年龄)

关系实例(relation instance):一个给定关系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)是关系的当前值,是动态的、随时间不断变化的,其变化通过关系的元组和属性值的改变表现出来。如表3-2、表3-3的内容分别是学生关系和指导关系的一个实例。

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,具体所指可从上下文中加以区别。可将“关系”的概念与程序设计语言中的“变量”概念做类比,“关系模式”和“关系实例”相当于变量的类型定义和变量的值。

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$(d_2,d_1,\cdots,d_n)$是不同的。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 $R(A_1,A_2,\cdots,A_i,A_j,\cdots,A_n)$ 与关系 $R(A_1,A_2,\cdots,A_j,A_i,\cdots,A_n)$ 的关系模式语义可相同，可表达同一关系。

归纳起来，关系数据模型中的关系具有如下一些性质。

(1) 元组个数有限性。

(2) 元组的唯一性，即关系中不能出现完全相同的两个元组。

(3) 元组的次序无关性，即元组的顺序可以交换。

(4) 属性名唯一性，即关系中不能有重名属性。

(5) 属性的次序无关性，即属性列的次序可以交换。

(6) 属性域的原子性，即属性值是不可分割的数据项。

(7) 属性域的同一性，即所有元组的同一属性值来自同一个域，是同一数据类型。

4. 关系与二维表

关系的逻辑数据结构是表结构，在关系模型中，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，每一列对应一个属性，每一行对应一个元组，所有行的集合构成了关系的实例。

关系描述的是实体及实体间的联系，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以二维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。

但关系和表实际上是不同的，注意加以区分，这有助于对关系的理解。关系和表的不同之处具体体现在以下几方面。

(1) 关系本质上是一个集合，是 n 个域上的一个 n 元组的集合，是“n 维的”；表是关系的二维呈现，是平面的。

(2) 关系是元组的集合，不是元组的列表，关系中元组的次序是任意的；表中各行从上到下是有序的。

(3) 关系中属性的次序是任意的；表中各列从左到右是有序的。

(4) 关系中不能有相同的元组；表中可能包含重复的行。

(5) 空关系相当于空集合，不含任何属性；空表可有列没有行。

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。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等问题。在实际的 DBMS 中，定义关系模式时，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 DBMS 的实现问题。

5. 关系数据库

在支持关系模型的 DBMS 中，数据库是由一个或多个关系组成的，数据库中各关系模式的集合称为关系数据库(relational database，RDB)的模式，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,描述所有实体以及实体之间联系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数据库管理系统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。

关系模型的完整性约束保证授权用户对数据库的操作不会破坏数据的完整性,即防止关系数据库中存在不符合语义的数据,也就是不正确的数据。

关系模型有三类完整性约束: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要求关系数据库必须满足的完整性约束条件,称为关系的两个不变性,一般由RDBMS默认提供支持。用户定义的完整性是应用领域要求关系数据库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

在关系模型中,用关系来描述实体,关系中每个元组表示的是每一个实体成员,每个实体用关键字属性来区分不同实体成员,例如学号是学生实体的关键字。对应实体的关键字,在关系中定义了候选键属性来唯一标识一个元组,实体完整性约束就是对关系的候选键中的属性进行约束。

首先介绍候选键及其相关概念。

候选键(candidate key):也称候选码,简称键或码。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)下面标出下画线,来表明它是键的组成部分。例如,表3-2中的学生关系可写成如下形式,其中"姓名"是候选键。

学生(姓名,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)。主键是数据库设计者选中用来在DBMS中区分一个关系中不同元组的候选键,主键的选择会影响某些实现问题,例如索引文件的建立(见第6章)。

包含候选键的属性集称为**超键**(super key)。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)。

【例 3-3】 在学生选课数据库中，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。数据库中各关系模式为

学生(学号，姓名，性别，出生时间，所在系)
课程(课程编号，课程名，先修课程号)
选课(学号，课程编号，成绩)

根据数据的语义，学生关系的候选键(主键)为“学号”；课程关系的候选键(主键)为“课程编号”；在“选课”关系中，由于一个学生可选多门课程，一门课程也会有多个学生选修，所以“学号”和“课程编号”两个属性的值才能唯一标识一个选课元组，需要共同构成关系的候选键(主键)。

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)。

实体完整性就是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。在关系数据库中使用空缺符 NULL 表示。

在例 3-3 中，“学生”关系的主属性“学号”和“课程”关系的主属性“课程编号”不能为空；选课(学号，课程编号，成绩)关系中主属性“学号”和“课程号”都不能为空。

在学生(姓名，性别，年龄)关系中，“姓名”属性是候选键，则“姓名”是主属性，不能为空。

这条约束规则体现了关系模型的键约束特性，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。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成员。

需要说明的是，实体完整性约束针对的是系统中定义的基本关系(存储的关系)，并不对查询的结果关系、外模式等进行约束。

2. 参照完整性

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，这样就存在描述联系的关系与描述实体的关系之间的参照，关系之间的参照一般通过外键来描述。

外键(foreign key)：也称外码。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 上的值来自关系 S 中某个元组的 K_s 上的值，则称该属性(集) F 为关系 R 的外键。

上述的关系 R 为**参照关系**(referencing relation)，又称引用关系，关系 S 为**被参照关系**(referenced relation)或**目标关系**(target relation)。

在例 3-3 中，若课程关系的某门课程的“先修课程号”只能对应课程关系的某门课程的“课程编号”，则“先修课程号”是课程关系的外键，被参照关系和参照关系是同一个关系。选课关系描述的是学生实体和课程实体之间的选课联系，选课关系中某个选课元组的“学号”和“课程编号”的值应分别对应学生关系和课程关系的某个元组的主键值，“学

号”和“课程编号”应是选课关系的外键,选课关系是参照关系,学生关系和课程关系是被参照关系。

在实际应用中,外键的定义需要注意以下几点。

(1) 参照关系 R 和被参照关系 S 可以是同一个关系,即外键 F 参照本关系的主键,表明同一关系中不同元组之间的参照关系。

比如在例 3-3 中,课程关系中的“先修课程号”就是课程关系的外键,其对应的主键为本关系的主键“课程编号”。

(2) 外键与对应的主键必须定义在相同的值域上,即属性值的数据类型要完全一致。

例如在例 3-3 中,课程关系中的外键“先修课程号”与对应主键“课程编号”必须定义在相同的值域上。

(3) 当外键与相应的主键属于不同关系时,命名可以不同,但一般给它们取相同的名字。

例如在例 3-3 中,选课关系的外键“学号”和“课程编号”分别与对应的学生关系和课程关系的主键“学号”和“课程编号”同名。

参照完整性约束就是要求外键的取值遵循如下约束规则。

参照完整性约束规则:若属性(集)F 是关系 R 的外键,它与关系 S 的主键 K_s 对应,则 R 中元组在 F 上的取值只能有两种可能:

(1) 取空值。

(2) 等于 S 中某个元组的 K_s 值。

【例 3-4】 学生实体和专业实体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。

一个关系的外键 F 是否能为空值,应视具体问题而定。在例 3-3 中,选课关系中的“学号”和“课程编号”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关系和课程关系中某个元组的主键值。但由于“学号”和“课程编号”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,选课关系中的外键“学号”和“课程编号”只能取对应被参照关系中已经存在的某个元组的主键值。

这条约束规则的实质是不允许引用不存在的实体,在参照关系中出现的值也必须在被参照关系中出现。

3. 用户定义的完整性

关系数据库除了要满足实体完整性和参照完整性之外,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。用户定义的完整性主要有如下一些情况。

(1) 对属性域的约束。对每个属性的数据类型进行约束,使得该属性上的每个取值都只能是该类型。例如,“年龄只能取整数”及“姓名的字符串长度最大为20”等域约束条件。

(2) 对属性值的取值范围进行约束。例如,要求“学生考试成绩在0～100”和“在职职工的年龄不能大于60岁”等。

(3) 对同一关系中的元组进行约束。例如,要求不同元组的同一属性的值不能相同,即属性值具有唯一性,存在如“不允许出现两个不同的学生拥有相同的姓名”类似的约束。

(4) 对同一元组的各属性进行约束。例如,要求属性值间满足一定的依赖关系,存在如“职工工资与职工的工龄和职务满足一定的算术关系”等约束。

(5) 对数据库的各关系进行约束,即不同的关系中的元组有一定的约束。例如,要求“计算机系的学生必须选修数据库课程”等。

一般来说,一个自定义的完整性约束可以是关于数据库的任意谓词。但因检测任意谓词的代价太高,大多数DBMS允许用户定义只需极小开销就可以检测的完整性约束条件。

4. 完整性控制机制

为了保证数据库的完整性,DBMS必须提供定义、检查和控制数据完整性的机制(称为完整性子系统)。

完整性约束的定义通常作为数据库模式设计的一部分存入数据库中。在实际应用中,当在关系模式定义中定义了主键和外键后,由DBMS默认提供实体完整性和参考完整性;用户定义的一些关于属性、域的约束也可在关系模式定义时进行定义,其他一些对数据库中各关系的元组值,即数据库状态所施加的约束,可以通过定义触发器(见4.4节)和定义事务(见第8章)等来实现。

DBMS监视用户对数据库进行操作的整个过程,检查用户发出的操作请求,如果发现有违背了完整性约束的情况,则采取拒绝执行操作等动作来保护数据的完整性(见4.2节和4.4节)。

在早期的DBMS中,没有提供定义和检查数据库完整性的机制,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,若要实现参照完整性,每插入一条学生选课记录,必须在应用程序中写一段程序,来检查其中的“学号”和“课程编号”属性的值是否在学生和课程关系中出现。现在只需在关系模式定义中定义外键和对应的主键就可以了,由DBMS来实现参照完整性。

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)。

关系代数提供了对关系的基本运算和组合运算,能用对关系的代数运算来表达对关系的操作。关系代数的重要性体现在以下两个方面。

(1) 关系代数为关系模型的数据操作能力表达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数据操作语言表达能力的尺度。当一种关系数据操作语言至少拥有关系代数的作用,即该语言能够表达用关系代数可以表示的关系操作,则称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中,作为优化查询的基础(见第7章),用来实现从数据库中存取数据的基本操作。

与关系代数不同,关系演算是用关系操作得到的元组应满足的谓词条件来表达操作要求。根据谓词变元是关系中的元组还是属性,关系演算的表达形式分为元组关系演算和域关系演算。关系演算的重要性体现在其有坚实的数理逻辑基础,与关系代数在表达能力上是等价的,也可用来评估实际系统中查询语言能力的标准和基础。

此外,关系代数是一种过程化语言(procedural language),而关系演算是一种非过程化语言(nonprocedural language)。使用过程化语言,用户需要对数据库执行一系列操作以获得所需结果;使用非过程化语言,用户只需描述所需要的结果是什么,而不用给出获取结果的具体过程。

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(query language)。这些语言与具体的RDBMS中实现的操作语言并不完全相同,没有给出具体的语法要求。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。

下面是曾经出现的一些RDBMS实际操作语言。

(1) ISBL(information system base language)是IBM公司英格兰底特律科学中心在1976年研制的,用在一个实验系统PRTV(peterlee relational test vehicle)上。ISBL的每个查询语句都近似一个关系代数表达式。

(2) QUEL(query language)是美国加州大学伯克利分校研制的关系数据库系统INGRES使用的查询语言。QUEL参照E. F. Codd提出的ALPHA元组演算语言研制的,是一种基于元组关系演算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IBM公司高级研究实验室的M. M. Zloof提出的,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。

目前RDBMS实际使用的是一种结构化的查询语言(structured query language,SQL),SQL不仅具有丰富的查询功能,而且具有数据定义和控制功能。SQL吸纳了关系代数的概念和关系演算的逻辑思想,是一种非过程性语言,具有语言简洁,易学易用的特点,已成为关系数据库的标准语言。

我们将在3.2节介绍关系代数的操作,在3.3节介绍关系演算的表达,在第4章讨论SQL的功能。

3.2 关系代数

关系代数是一种过程化的查询语言,它用对关系的运算来表达关系操作。

一门代数总是由一些操作运算符和一些原子操作数组成的。例如,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运算结果上构造表达式，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，基于一组为数不多的以关系为操作对象的运算符，其原子操作数则为代表关系实例的关系名变量和具体元组组成的集合常量。

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。

传统的集合运算主要包括并、差、交和广义笛卡儿积。

专门的关系运算主要包括投影、选择、连接、除和重命名。

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。专门的关系运算不仅涉及元组，还涉及属性列，并需使用比较运算符和逻辑运算符来辅助完成。

每个运算符对一个或两个关系进行运算，产生的结果是另外一个关系，可以把多个关系代数运算组合成一个关系代数表达式(relational algebra expression)，如同将算术运算组合成算术表达式一样，来表达对数据库中的关系进行操作的过程。

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、交和广义笛卡儿积4种运算。其中，参与并、差和交运算的两个关系必须是相容的。

设两个关系R和S是相容的，即关系R和S具有相同的目(属性个数相同)，且相应的属性对应同一个域，则如下定义并、差和交运算。

1. 并运算

关系R与S的并(union)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，实现元组的插入操作。

2. 差运算

关系R与S的差(difference)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。

差运算可用来实现元组的删除操作。

3. 交运算

R和S的交(intersection)是一个与R、S相容的关系，其元组由既属于R又属于S的所有元组组成，表示为$R \cap S$。

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

关系的交运算可以用差运算来实现。

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

或

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

4. 广义笛卡儿积

关系的笛卡儿积运算可以将任意两个关系的信息组合在一起。关系 R 和 S 的**广义笛卡儿积**(extended Cartesian product,简称**笛卡儿积**、**叉积**或**积**)是一个有序对的集合,有序对的第一个元素是关系 R 中的任何一个元组,第二个元素是关系 S 中的任何一个元组,表示为 $R \times S$。

$$R \times S = \{\overset{\frown}{t_r t_s} \mid t_r \in R \wedge t_s \in S\}$$

设关系 R 和关系 S 分别是 m 目和 n 目关系,R 中有 k_1 个元组,S 中有 k_2 个元组,则 $R \times S$ 为一个 $m+n$ 目的新关系,共有 $k_1 \times k_2$ 个元组,且每个元组的前 m 个分量是关系 R 的一个元组,后 n 个分量是关系 S 的一个元组。$R \times S$ 的属性集是关系 R 和 S 的属性集的并集,如果 R 和 S 恰好有同名的属性,就需要把至少一个关系中相应的属性名更改为不同的名称。为了使含义清楚,如果属性 A 在关系 R 和 S 中均出现,则结果关系模式中分别用 $R.A$ 和 $S.A$ 表示来自 R 和 S 的属性。当某个关系如果需要与自身做笛卡儿积运算时怎么办呢?3.2.2 节将提供一种改名运算来解决这个问题。

广义笛卡儿积是连接操作的基础。

【例 3-5】 给定关系 R、S,$R \cup S$、$R \cap S$、$R - S$、$R \times S$ 的结果如图 3-1 所示。

关系R

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

关系S

A	B	C
d	a	f
b	g	a

$R \cap S$

A	B	C
d	a	f

$R \cup S$

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

R–S

A	B	C
a	b	c
c	b	d

$R \times S$

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

图 3-1 集合运算示例

3.2.2 专门的关系运算

专门的关系运算包括投影、选择、连接、除法和重命名运算。

1. 投影运算

投影(projection)运算是对一个关系的属性进行操作的一元运算。关系 R 上的投影运算是从 R 中选择若干属性列组成一个新的关系。

设关系 R 为 n 目关系,$A_{i1}, A_{i2}, \cdots, A_{im}$ 是关系 R 的属性 $A_1, A_2, \cdots, A_n$ 的一部分,则关系 R 在 $A_{i1}, A_{i2}, \cdots, A_{im}$ 上的投影是一个 m 目关系,其属性为 $A_{i1}, A_{i2}, \cdots, A_{im}$,表示为

$$\pi_{A_{i1}, A_{i2}, \cdots, A_{im}}(R)$$

或

$$\pi_{i1,i2,\cdots,im}(R)$$

投影运算符用小写希腊字母 π(pi)表示，在结果中出现的属性名作为 π 的下标，属性名间用逗号分隔，参与运算的关系作为 π 后括号内的参数。在投影运算中，也可以用属性的位置标记隐含地作为关系的属性名，位置标记不像属性名易于理解，本书后续基本不采用位置标记方法。

投影操作提取了原关系的某些属性，与原关系相比，投影后的新关系的属性列数目减少，可能会有重复元组被去除，则元组数也可能会减少。

通过投影运算，可以对关系内的任意属性的数据进行查询。

2. 选择运算

选择(selection)运算是对一个关系的元组进行操作的一元运算。关系 R 上的选择运算是从 R 中选择满足给定条件 F 的元组组成一个新的关系，这个新关系与 R 具有相同的关系模式，其值是 R 的一个子集，表示为 $\sigma_F(R)$。

选择运算用小写希腊字母 σ(sigma)来表示，将选择条件写作 σ 的下标，参与运算的关系作为 σ 后括号内的参数。

选择条件 F 为一逻辑表达式，假设 t 是 R 中任意一个元组，把 t 代入逻辑表达式 F 中，如果 F 的结果为真，那么这个元组 t 就是 $\sigma_F(R)$ 中的一个元组，否则此元组不会在 $\sigma_F(R)$ 中出现。

$$\sigma_F(R)=\{t \mid t\in R \wedge F(t)=\text{TRUE}\}$$

逻辑表达式 F 由下面的规则组成。

(1) 由基本逻辑表达式 $a\theta b$ 组成，a、b 可为属性名或常量，但不能同时为常量。θ 为比较符<、>、=、≤、≥和≠。

注意：参与比较的每个属性必须是选择运算符的关系操作数里的一个属性，否则就是语法上的错误。

(2) 由基本逻辑表达式经逻辑运算 ¬(非)、∧(与)和∨(或)组成，称为复合逻辑表达式。

通过选择运算，可以对关系内的任意元组的数据进行查询。

关系 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

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

A	B	C
a	b	c
d	a	f

图 3-2 投影、选择运算示例

【例 3-6】 给出关系 R，$\pi_{C,A}(R)$、$\sigma_{A>'c'\vee C<'d'}(R)$ 的结果如图 3-2 所示。

3. 连接运算

通常情况下，涉及笛卡儿积的查询中会包含一个对笛卡儿积结果进行选择的运算，该选择运算大多数情况下会要求进行笛卡儿积运算的两个关系在某些属性上可以进行比较。θ-连接(join)运算是在 R 和 S 的广义笛卡儿积 $R\times S$ 中选取符合 $A\theta B$ 条件的元组，即选择在关系 R 中 A 属性组上的值与在关系 S 中 B 属性组上的值满足比较操作 θ(theta)的元组，表示为 $R\bowtie_{A\theta B}S$。

$$R\bowtie_{A\theta B}S=\{\widehat{t_r t_s} \mid t_r\in R \wedge t_s\in S \wedge t_r[A]\theta t_s[B]\}$$

其中，A 和 B 分别是 R 和 S 上属性个数相等且可比的属性组，θ 是比较运算符。$A\theta B$ 表

示$R.A_1\theta S.B_1 \wedge R.A_2\theta S.B_2 \wedge \cdots \wedge R.A_k\theta S.B_k$，$tr[A]$、$ts[B]$分别表示关系$R$、$S$的元组$t$在属性组$A$、$B$中$k$个属性上的值。

就像笛卡儿积操作一样，θ-连接的结果关系的属性集是关系R和S的属性集的并集，需要在重名的属性前面加上“$R.$”或“$S.$”。

θ-连接也可等价表示为

$$R \bowtie_{A\theta B} S \equiv \sigma_{R.A\theta S.B}(R \times S)$$

连接运算中有两种最为重要、最为常用的连接：**等值连接**和**自然连接**。

(1) 当θ为=时，θ-连接运算称为等值连接，表示为$R \bowtie_{A=B} S$。

$$R \bowtie_{A=B} S \equiv \sigma_{R.A=S.B}(R \times S)$$

$A=B$表示$R.A_1=S.B_1 \wedge R.A_2=S.B_2 \wedge \cdots \wedge R.A_k=S.B_k$。

(2) 自然连接是一种特殊的等值连接，它要求两个关系中进行比较的属性组必须是相同的，即A和B相同，并且在结果中把重复的属性列去掉，表示为$R \bowtie S$。

$$R \bowtie S \equiv \pi_{Z_1,Z_2,\cdots,Z_m}(R \bowtie_{A=A} S)$$

其中，$Z_1,Z_2,\cdots,Z_m$是从$R \bowtie_{A=A} S$中去掉重复属性后的诸属性。

一般的连接操作是从元组的角度进行运算，但自然连接还需要取消重复属性，所以它是同时从元组和属性列两个角度进行运算的。

在连接运算中，若用复合逻辑表达式F来代替基本逻辑表达式$A\theta B$，称为***F* 连接**。F连接运算也等价于在R和S的广义笛卡儿积$R \times S$中选取满足条件F的元组，表示为$R \bowtie_F S$。

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

【例 3-7】 给出关系R、S，则$R \bowtie_{D>E} S$、$R \bowtie_{D=E} S$以及$R \triangleright\triangleleft S$的结果如图 3-3所示。

关系R

A	B	D
1	2	4
2	4	6
1	1	7

关系S

D	E
7	5
6	7
8	4

$R \bowtie_{D>E} S$

$R.A$	$R.B$	$R.D$	$S.D$	$S.E$
2	4	6	7	5
2	4	6	8	4
1	1	7	7	5
1	1	7	8	4

$R \bowtie_{D=E} S$

$R.A$	$R.B$	$R.D$	$S.D$	$S.E$
1	2	4	8	4
1	1	7	6	7

$R \triangleright\triangleleft S$

A	B	D	E
2	4	6	7
1	1	7	5

图 3-3　连接运算示例

在两个关系R和S做$R \bowtie S$时，选择两个关系在相同属性上值相等的元组构成新的关系。R中的某些元组可能因在S中不存在相同属性上值相等的元组，从而使这些元组的信息不能保留在连接结果中。同样，S中的某些元组也可能被舍弃。如果关系的一个元组不能和自然连接的另外一个关系中的任何一个元组在相同属性上值相等而被舍弃，这个元组就称为**悬浮元组**(dangling tuple)。例如，对于例 3-7 中的$R \bowtie S$，R的第 1

个元组(1,2,4)、S 的第 3 个元组(8,4)就是悬浮元组。

在自然连接结果中,如果把舍弃的这些悬浮元组保留下来,并且在这些元组新增加的属性上赋空值 NULL,这种连接操作称为**外连接**(outer join)。如果在结果中只保留连接运算符左边关系中的悬浮元组,称为**左外连接**;如果在结果中保留运算符右边关系中的悬浮元组,称为**右外连接**;把两个关系中的悬浮元组都保留下来,称为**完全外连接**。

【例 3-8】 对例 3-7 中的关系 R 和 S,进行完全外连接、左外连接、右外连接运算,连接结果如图 3-4 所示。

左外连接

A	*B*	*D*	*E*
1	2	4	NULL
2	4	6	7
1	1	7	5

完全外连接

A	*B*	*D*	*E*
1	2	4	NULL
2	4	6	7
1	1	7	5
NULL	NULL	8	4

右外连接

A	*B*	*D*	*E*
2	4	6	7
1	1	7	5
NULL	NULL	8	4

图 3-4 外连接运算示例

4. 除运算

设有关系 $R(X,Y)$和 $S(Y)$,其中 X、Y 为属性组,$S(Y)\neq\varnothing$,则 R 除以 S 也是一个关系,称为 R 除以 S 的商,可记为 $R\div S$。

R 能被 S 除(division)必须满足下面的前提条件。

(1) R 中的属性包含 S 中的所有属性。

(2) R 中有一些属性不出现在 S 中。

为了理解除运算,先介绍像集的概念。给定一个关系 $R(X,Y)$,X 和 Y 为属性组,若 $x\in\pi_X(R)$,R 中所有在属性组 X 上的分量等于 x 的元组在属性组 Y 上的分量的集合,称为 ***x* 在 *R* 中的像集**(images set),用 Y_x 表示。

$$Y_x=\{t\ [Y]\mid t\in R\wedge t[X]=x\}$$

【例 3-9】 对于关系 R(姓名,课程)中的元组在“姓名”(X 属性)上的一个值“张军”,其在“课程”(Y 属性)上的像集为{物理,数学},可表示张军选修的所有课程,如图 3-5 所示。

关系*R*

属性组*X*	属性组*Y*
姓名	课程
张军	物理
王红	数学
张军	数学

若*x*=张军
则Y_x

课程
物理
数学

Y_x表示张军选修的所有课程

图 3-5 像集概念示例

因此,对于关系 $R(X,Y)$和 $S(Y)$,$R\div S$ 得到一个新的关系,其属性由在 R 中而不在 S 中的属性 X 所组成,若 $x\in\pi_X(R)$,且 $S\subseteq Y_x$,Y_x 为 x 在 R 中的像集,则 $x\in R\div S$。

根据除运算的定义,对于给定的关系 R 和 S 的实例,要得到 $R\div S$ 的结果,可分如下

4步进行。

(1) 对关系 R 在属性组 X 上进行投影 $\pi_X(R)$,得到 x。

(2) 获取各 x 的像集 Y_x。

(3) 检查各个 x 在 Y 上的像集 Y_x 是否包含 S。

(4) 将满足条件的 x 放入结果集合。

【例 3-10】 给定关系 R 和 S,如图 3-6 所示。先对关系 R 中的元组在属性组 X 进行投影 $TL_x(R)$,得到3个 x 值,再分别计算其对应的像集 Y_{x1},Y_{x2},Y_{x3},判断每个像集是否包含关系 S 中的所有元组,将满足条件的像集 Y_{x2},Y_{x3} 对应的 x 放入 $R \div S$ 的结果中,即 $R \div S$ 的结果中包含 $x2$,$x3$。

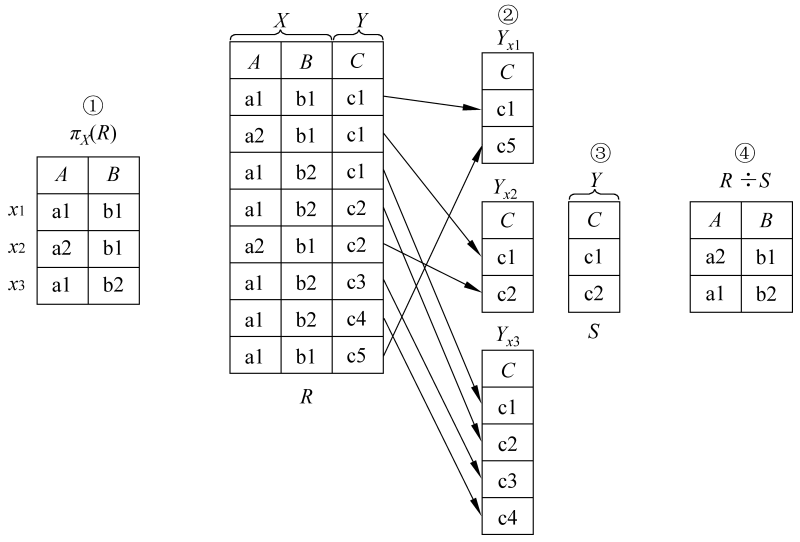

图 3-6 除法运算示例

【例 3-11】 给定选课关系 SC 和课程关系 C,如图 3-7 所示。查询选修所有课程的学生的学号。

关系SC

学号	课程号	成绩
1	C1	83
1	C5	83
5	C5	90
5	C1	92

关系C

课程号	课程名
C1	C语言
C5	数据库

图 3-7 例 3-11 关系实例

对如下三种表达,哪一种是正确的?

(1) SC$\div C$。

(2) SC$\div \pi_{课程号}(C)$。

(3) $\pi_{学号,课程号}(SC) \div \pi_{课程号}(C)$。

说明:

(1) 由于关系 C 中具有不包含在关系 SC 中的属性“课程名”,因而除运算的前提条件不满足,无法进行除运算。所以,解法(1)是错误的解法。

(2) 在进行除运算前,应对除关系 C 进行投影,去掉不包含在被除关系 SC 中的属性“课程名”,即计算 $\pi_{课程号}(C)$,再做除法运算。由于投影之后所得的关系只包含属性“课程号”,而“课程号”包含在被除关系 SC 中,所以满足除运算的条件,能够进行除运算。

但被除关系 SC 中不包含在除关系 C 中的属性是“学号”和“成绩”,所以除法运算的

结果中包含“学号”和“成绩”两个属性。除运算的结果是那些选修了课程表中全部课程(C1 和 C5)且成绩相同的学生的学号和成绩,运算结果为{(1,83)}。显然,解法(2)不满足查询需求。

(3) 要完成题目所要求的查询“选修所有课程的学生的学号”,在进行除运算前,应根据操作的要求准确地确定像集属性和结果属性。对除关系和被除关系进行投影,去掉不需要的属性,再做除法运算,即执行 $\pi_{学号,课程号}(SC) \div \pi_{课程号}(C)$ 运算,运算结果为{1,5}。所以,解法(3)是正确的解法。

从例 3-11 可看到,除运算可实现查询一个关系的属性值集合是否包含另一个关系实例的代数操作,通常需要对数据库中参与运算的关系的属性进行预处理,处理掉可能影响结果的多余的属性,使被除关系 R 的属性包含除数关系 S 的所有属性,商中属性是代表查询结果信息的最小属性组。

除运算不是基本运算,它可以由基本运算进行如下运算推导出来。

$$R \div S = \pi_X(R) - \pi_X[(\pi_X(R) \times S) - R]$$

其中,X 是在 R 中而不在 S 中的属性(组)。

前面介绍的 8 种关系代数运算,其中并、差、广义笛卡儿积、投影和选择 5 种运算是基本运算;而其他 3 种即交、连接和除运算,均可以用前 5 种基本运算来表达,这称为组合运算。

5. 重命名运算

当一个数据库查询涉及对一个关系的不同元组的属性进行 θ 比较运算时,需要对一个关系进行关系的自身 θ 连接操作(如例 3-12 中的查询(8)),对来自 2 个同名关系的不同元组进行 θ 比较运算。为了标识同一关系中的不同元组,需要将自身连接关系中至少一个关系改名,即对关系进行重命名(rename)运算。

重命名运算符用小写希腊字母 ρ(rho)表示,若将关系 R 重命名为 S,并将 R 中的属性按从左到右的顺序重命名为 $A_1, A_2, \cdots, A_n$,则重命名运算表示为

$$\rho_{S(A_1,A_2,\cdots,A_n)}(R)$$

如果只是想把关系的名字改变为 S,并不改变其中的属性名,简单地使用 $\rho_S(R)$ 即可。

重命名运算还可用来解决含有相同属性的两个关系的笛卡儿乘积或连接操作的属性命名问题,也可用来给一个代数表达式的结果命名为一个新关系。

许多文献还介绍一些扩展的关系代数运算,如广义投影、聚集运算等,本书不再做介绍,这些操作在后续 SQL 的学习中会很容易理解和实现。

3.2.3 用关系代数运算实现数据库操作

数据库描述的是整个应用领域的实体对象及实体间的联系,因此一个关系数据库会包含多个关系,用户对一个数据库的操作会涉及多个关系中的元组或属性。若用关系代数操作语言来表达用户对数据库的操作,将涉及对数据库中多个关系的一系列代数操作,某个代数操作的结果关系可作为操作数参与另一个代数运算,运算过程可用一个关系代数表达式来表达,表达式的运算结果就是对数据库的操作结果。

1. 关系代数表达式

关系代数基本表达式是关系代数的原子操作数,包括如下两方面。

(1) 数据库中的一个关系,用代表关系实例的关系名表示。比如例3-11中的选课关系SC、课程关系C等。

(2) 一个常数关系,用在{ }内列出的元组集合来表示。比如{(张山,男,19),(王武,男,20),(李斯,女,20)}等。

设E_1和E_2是关系代数基本表达式,则进行以下基本运算的结果都是关系代数表达式。

- $E_1 \cup E_2$;
- $E_1 - E_2$;
- $E_1 \times E_2$;
- $\sigma_F(E_1)$,其中F是E_1的属性上的逻辑表达式;
- $\pi_S(E_1)$,其中S是E_1中某些属性的列表;
- $\rho_X(E_1)$,其中X是E_1结果的新名字。

以上关系代数表达式进行有限次代数运算构成新的关系代数表达式。

由关系代数的基本运算足以表达通常的数据库操作,但若局限于基本运算,某些常用操作表达出来会很冗长。所以,关系代数表达式中也使用以下组合运算,它们不能增强关系代数的表达能力,却可以简化一些数据库操作的关系代数表达式。

- $E_1 \cap E_2$;
- $E_1 \bowtie E_2$,$E_1 \bowtie_{A\theta B} E_2$,$E_1 \bowtie_F E_2$;
- $E_1 \div E_2$。

2. 用关系代数表达式对数据库查询

下面以对数据库的查询操作为例,用关系代数表达式来对数据库进行查询。

为正确书写满足查询语义的关系代数表达式,可遵循如下步骤。

(1) 确定查询目标,即结果关系来自哪些关系中的属性。

(2) 明确查询条件,即决定结果关系中元组来源的因素。

(3) 寻找从条件到目标的查找路径,明确在查询过程中需要涉及哪些关系,这些关系又是如何进行连接的。

(4) 根据步骤(2)明确的查询条件进行元组的选择,把选择出来的元组集合作为新关系,参与下一步操作。

(5) 根据步骤(3)的分析结果进行关系的连接。对有些只涉及单个关系的查询,可忽略此步骤。

(6) 确定结果属性,即根据步骤(1)确定的查询目标执行投影操作。

步骤(1)~(3)可以看作一个分析查询语义的过程,步骤(4)~(6)是书写表达式的过程。有时也可以先做步骤(5),再做步骤(4),即先进行关系的连接,再从连接结果中选择满足条件的元组。

【例3-12】 有一个描述学生及其选修课程的关系数据库,由学生关系S、课程关系C和选课关系SC三个关系组成,其关系模式如下。

S(学号,姓名,性别,出生时间,专业)

C(课程号,课程名,先修课程号)

SC(学号,课程号,成绩)

用关系代数表达式对数据库进行如下查询。

(1) 查询2000年元旦(含)以后出生的学生姓名。

$$\pi_{\text{姓名}}(\sigma_{\text{出生时间}\geqslant\text{'2000-01-01'}}(S))$$

(2) 查询选修了课程号为C2的学生学号。

$$\pi_{\text{学号}}(\sigma_{\text{课程号}=\text{'C2'}}(\text{SC}))$$

(3) 查询选修了课程名为“数据库”且成绩大于90的所有学生姓名。

$$\pi_{\text{姓名}}(\sigma_{\text{课程名}=\text{'数据库'}}(C) \bowtie (\sigma_{\text{成绩}>90}(\text{SC})) \bowtie S)$$

或

$$\pi_{\text{姓名}}(\sigma_{\text{课程名}=\text{'数据库'}\wedge\text{成绩}>90}(C \bowtie \text{SC} \bowtie S))$$

请思考,这两种表达哪个更好?判断的依据是什么?

(4) 查询至少选修了学号为S5的学生所选修的一门课程的学生的姓名。

$$\pi_{\text{姓名}}(\pi_{\text{学号}}(\pi_{\text{课程号}}(\sigma_{\text{学号}=\text{'S5'}}(\text{SC})) \bowtie \text{SC}) \bowtie S)$$

题解:首先从选课关系SC得到学号为S5的学生所选修的课程号,然后与选课关系SC作自然连接,去筛选具有相同课程号的选课元组,得到所有学生(包括S5)选修了这些课程的选课元组,最后与学生关系S作自然连接,得到学生的姓名。

(5) 查询被所有学生都选修了的课程名。

$$\pi_{\text{课程名}}((\pi_{\text{学号,课程号}}(\text{SC}) \div \pi_{\text{学号}}(S)) \bowtie C)$$

(6) 查询没选修“数据库”课程的学生的学号。

$$\pi_{\text{学号}}(S) - \pi_{\text{学号}}(\pi_{\text{课程号}}(\sigma_{\text{课程名}=\text{'数据库'}}(C)) \bowtie \text{SC})$$

问:此查询是否可如下表达?

$$\pi_{\text{学号}}(\pi_{\text{课程号}}(\sigma_{\text{课程名}\neq\text{'数据库'}}(C)) \bowtie \text{SC})$$

(7) 检索选修了“张山”同学所选修的所有课程的学生姓名。

$$\pi_{\text{姓名}}(S \bowtie (\pi_{\text{学号,课程号}}(\text{SC}) \div \pi_{\text{课程号}}(\pi_{\text{学号}}(\sigma_{\text{姓名}=\text{'张山'}}(S)) \bowtie \text{SC})))$$

题解:首先从学生关系S得到“张山”的学号,然后与选课关系SC作自然连接,去筛选该同学的所有选课元组,再去除选课关系SC,得到满足条件的学生学号,最后与学生关系S作自然连接,得到学生的姓名。

(8) 检索至少选修两门课程的学生学号。

$$\pi_{\text{学号}}(\sigma_{C\neq\neq\text{课程号}}(\rho_{\text{SG}(S\neq,C\neq,G)}(\text{SC}) \bowtie_{S\neq=\text{学号}} \text{SC}))$$

题解:首先将选课关系SC重命名为SG($S\neq$,$C\neq$,G),然后与选课关系SC作自身连接,并选择两个关系中学号相同而课程号不同的元组,即得到选课关系SC中学号相同而课程号不同的不同元组,最后投影得到学生的学号。

3.3 关系演算

将数理逻辑中的谓词演算推广到关系运算中,用谓词演算来表达关系的操作,即关系演算。关系演算是用关系操作的结果应满足的谓词条件来表达关系操作要求的,而不是

像关系代数那样用操作符计算结果。

如何表达关系满足的谓词条件呢？需要说明关系的另外一种表示方式。

3.3.1 关系演算中关系的表示

关系是一个集合，集合主要有两种表示方法：列举法和描述法。列举法是列举出集合中的元素，这种方法比较适用于有限集合。描述法用集合中的元素所要满足的特性来表示集合，比如$\{x \mid x>3, x \in \mathbf{N}\}$表示的是所有大于3的整数集合。可以用集合描述法建立谓词与关系间的联系。

在关系演算中，关系用谓词(predicate)表示，关系R可以看成满足一定谓词条件的元组或属性域的集合，可表示为

$$\{u \mid R(u)\}$$

其中，u可为元组变量或域变量，$R(u)$是一个谓词。

谓词$R(u)$相当于一个返回逻辑值的函数名，如果R是一个包含n个属性的关系，若$(a_1, a_2, \cdots, a_n)$是R的元组，则$R(a_1, a_2, \cdots, a_n)$的值为TRUE，否则为FALSE。

在$\{u \mid R(u)\}$中，若用R表示对关系操作的结果关系，则关系的操作要用关系的谓词演算来表达，称$\{u \mid R(u)\}$为关系演算表达式，u为演算变量，$R(u)$为是演算公式。当u为元组时，所进行的关系演算为元组关系演算；当u为域变量时，所进行的关系演算为域关系演算。

3.3.2 元组关系演算

1. 元组演算公式

在元组关系演算表达式中，元组演算公式由原子公式组成。

1) 原子公式

原子公式有下面三种形式。

(1) $R(t)$：其中，R是关系名称，t是元组变量，$R(t)$表示t是R中的元组。关系R就可以直接表示为$\{t \mid R(t)\}$。

(2) $t[i] \theta u[j]$：其中，t和u是元组变量，θ是比较运算符，$t[i] \theta u[j]$表示“元组t的第i个分量与元组u的第j个分量满足比较关系θ”，例如$t[2]<u[3]$。

(3) $t[i] \theta C$：其中，C是常量，$t[i] \theta C$表示“元组t的第i个分量与常量C满足比较关系θ”，例如$t[2]=3$。

2) 元组演算公式

元组演算公式的递归定义如下。

(1) 原子公式是公式。

(2) 设$\varphi_1(t_1)$和$\varphi_2(t_2)$是公式，则$\neg \varphi_1(t_1)$，$\varphi_1(t_1) \wedge \varphi_2(t_2)$，$\varphi_1(t_1) \vee \varphi_2(t_2)$，$\varphi_1(t_1) \rightarrow \varphi_2(t_2)$也是公式。

(3) 设$\varphi(t)$是公式，t是$\varphi(t)$中的元组变量，则$(\exists t)\varphi(t)$，$(\forall t)\varphi(t)$也是公式。

(4) 有限次使用上述规则得到的式子都是公式。

其中，$\exists$是存在量词符号，$(\exists t)\varphi(t)$表示“若有一个t使φ为真，则$(\exists t)\varphi(t)$为真，否