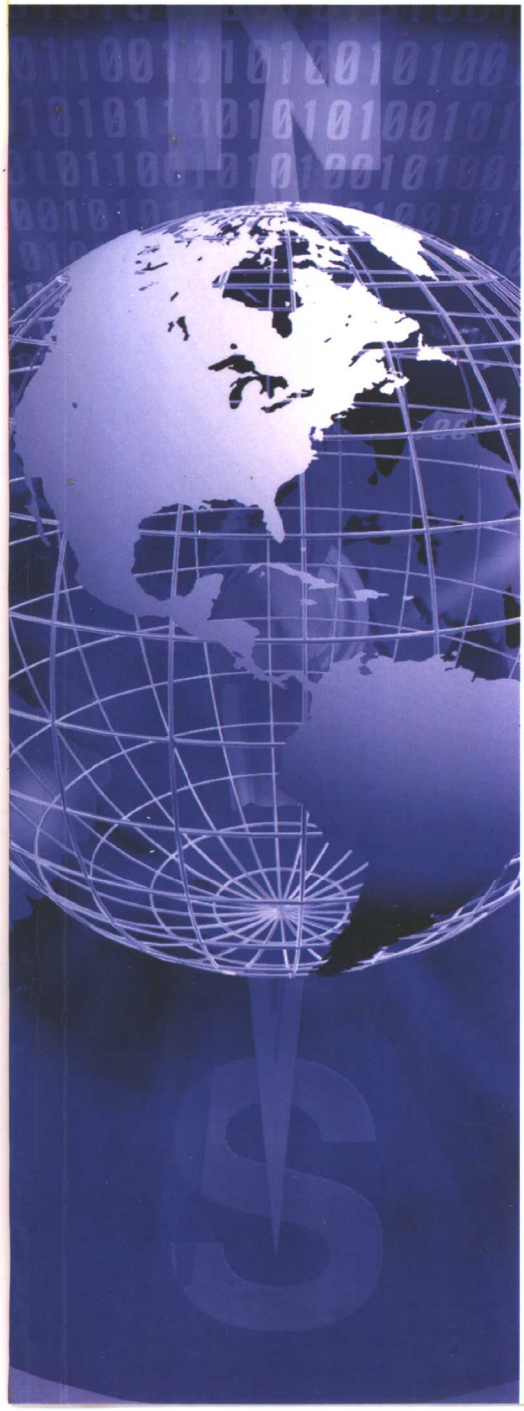
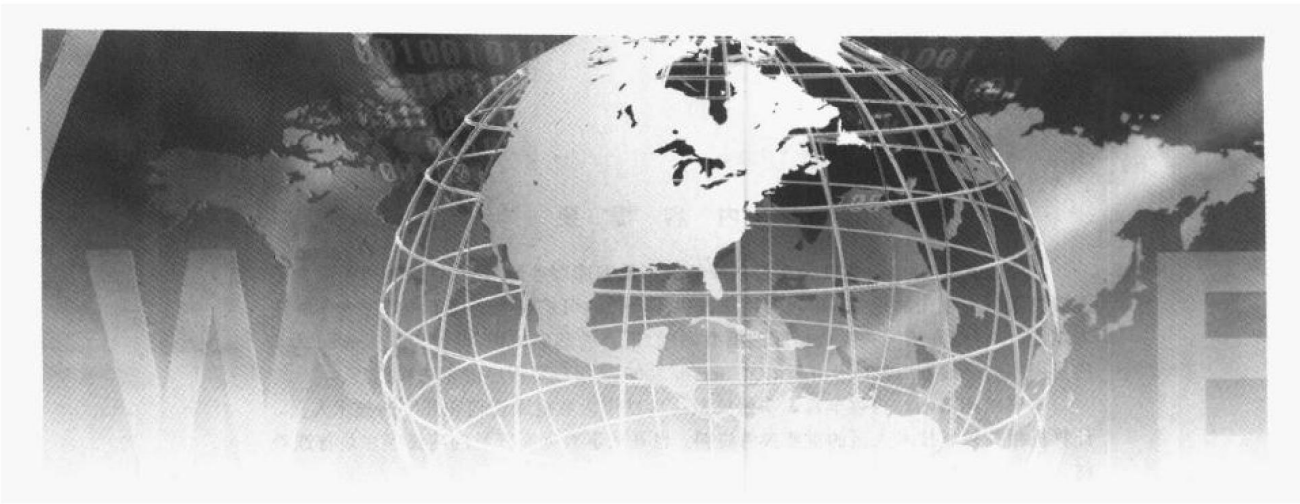
SQL Server 2000

高级开发指南



精英科技 / 编著

中国电力出版社
www.infopower.com.cn



SQL Server 2000

高级开发指南

精英科技／编著

中国电力出版社

内 容 提 要

SQL Server 2000 是微软公司最新版的大型数据库服务器，已在性能和可扩展性方面确立了世界领先的地位，可以为用户提供下一代的可扩展电子商务和数据仓库解决方案。书中详细介绍了关系数据库的模型、SQL 语言的基础知识、设计 SQL Server 2000 应用程序的一般方法，以及开发 SQL Server 2000 高级应用程序的方法和技巧。

本书结构清晰、内容丰富、举例典型，可以作为数据库管理和开发的技术人员、计算机相关领域技术人员的重要参考资料，也是从事数据库开发的高级程序员首选教材。

图书在版编目（CIP）数据

SQL Server 2000 高级开发指南/精英科技编. —北京：中国电力出版社，2002.6

ISBN 7-5083-1063-2

Ⅰ. S… Ⅱ. 精… Ⅲ. 关系数据库-数据库管理系统，SQL Server 2000 Ⅳ. TP311.138

中国版本图书馆 CIP 数据核字（2002）第 032899 号

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://www.infopower.com.cn）

汇鑫印务有限公司印刷

各地新华书店经售

*

2002 年 7 月第一版 2002 年 7 月北京第一次印刷

787 毫米×1092 毫米 16 开本 17 印张 407 千字

定价 **25.00** 元

前　言

SQL Server 2000 是微软公司精心打造、全力推出的最新版的大型数据库服务器，其强大的功能和简单的操作方法使得用户有物超所值的感觉。SQL Server 2000 已经在性能和可扩展性方面确立了世界领先的地位，是一套完全的数据库和数据分析解决方案，使用户可以快速创建下一代的可扩展电子商务和数据库系统。

程序员，尤其是数据库开发程序员，他们主要做的工作是使用诸如 DAO、ODBC 以及 OLE DB、ADO 这些 API 开发 SQL Server 数据库应用程序。他们对于 SQL Server 也需要掌握，但最主要的是要会一种以上编程语言和脚本语言，因此他们可以使用 Access，Visual Basic，Visual C++、PowerBuilder、InterDev 等作前端开发工具，底层用上述 DAO 等作为接口，为 SQL Server 定制客户端和服务器端应用产品。SQL Server 2000 新增了一些特性，它们要求程序员具有 ASP、XML、Web 集成的能力，也就是说现在已经从针对公司内部的局域网范围开发应用，向 Internet 开发应用扩展。他们要会 SQL Server 2000 高级数据库功能，也就是所谓的高级读者了。本书主要是针对这些高级读者的。

本书共分为 10 章。第 1 章讲述了关系数据库的模型。第 2、3 章讲述了 SQL 语言的基础知识。第 4 章讲述了设计 SQL Server 2000 应用程序的一般方法。第 5 章到第 10 章则分别讲述了开发 SQL Server 2000 高级应用程序的方法和技巧。

各章之间内容衔接紧密，互相呼应，每章最后都有小结，确保你对整章内容有总体把握。

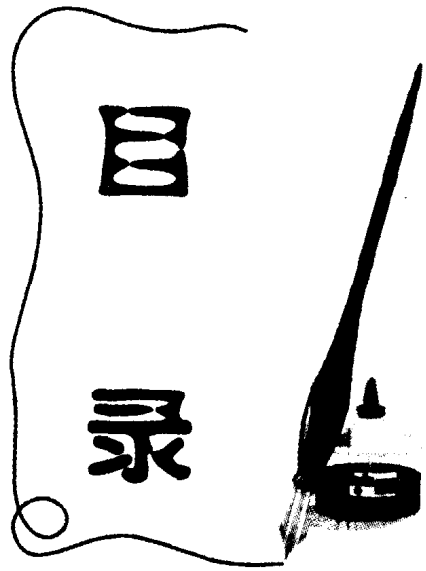
SQL Server 2000 是微软公司目前推出的最新的数据库服务器，其中增加了大量的新技术和新方法，关于这些内容的资料目前还是比较少的，加上写作本书时间仓促，因此书中难免有一些不妥之处，请读者谅解。

对书中的内容有任何疑问或者建议，欢迎与我们联系。

我们的 Email 地址：yyvone@263.sina.com

作　者

2002 年 4 月



1 关系数据库模型

本书首先研究关系型数据库模型，因为它是 SQL Server 和所有其他关系型数据库管理系统（DBMS）的基础。另外一方面，由于本书重点不在于此，故本章只做较为详细的分析。

一个完整的数据库系统由数据库、数据库管理系统、数据库管理员和应用程序四部分组成，如图 1-1 所示。其中数据库管理系统（DBMS）是实现在数据库上进行各种操作的数据库管理软件，也是数据库系统的核心；各种应用程序是不同用户使用数据库的界面，它在 DBMS 的支持下为用户提供各类服务；而数据库管理员（DBA）是一个或一组人员，是完成数据库的规划、设计、维护和管理等工作的高级用户。

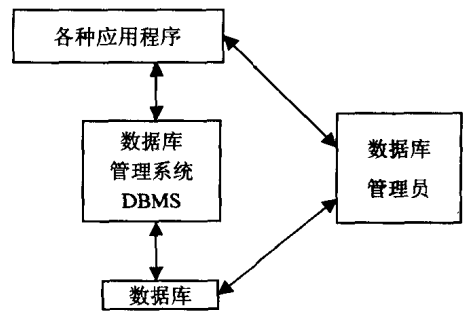


图 1-1 数据库系统的组成

在数据库系统中，要将数据按照一定的规则存储，就需要一定的数据模型，在数据模型的发展历史上，出现过许多数据模型，在今天，关系模型已经被普遍选择，其主要原因是：

（1）关系模型对数据及其联系方式的表示非常简洁，无论是数据还是数据之间的联系都用关系来表示。

（2）关系模型支持用高度非过程化的说明型语言表示数据的操作。

（3）同时，关系数据模型具有严格的理论基础——关系代数。

我们希望设计各种方便使用和管理的数据库，其前提是理解关系模型。进而就可以使用 SQL 阅读和更新自己工作组中的数据应用程序。

1.1 关系数据库模型

数据模型是数据库系统的数学形式框架，是用来描述数据的一组概念和定义，包括以下三方面的内容：

（1）数据的静态特征。包括对数据结构和数据间联系的描述；

（2）数据的动态特征。是一组定义在数据上的操作，包括操作的含义、操作符、运算规则及其语言等；

（3）数据的完整性约束。是一组规则，数据库中的数据必须满足这组规则。

而在今天，最具商业化、最被广泛运用的 DBMS 的基础是关系型数据模型。

关系模型是数据库设计和数据库实现的基础。包括三个主要部分——数据结构、完整性

和数据操作。对于其中的数据结构和完整性，关系模型的方法提供了一种设计和实现复杂数据库的良好的组织方式；而数据操作方面，关系型操作又为操纵数据提供了强有力的方式。

1.1.1 关系数据结构

关系型数据库模型中，数据结构部分是定义数据的形式。这种形式最基本的内容是关系的概念。

这里用图 1-2 这个例子来表示一种关系形式，它类似于通常所见到的表格。

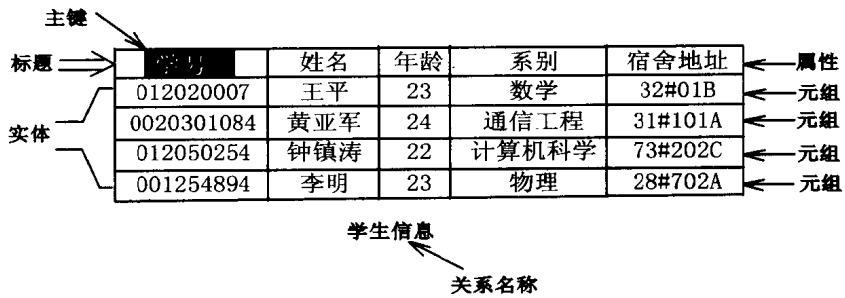


图 1-2 关系及其组成

关系可以看作由列和行组成。其中列在关系术语中称为属性，而行在关系术语中则被称为元组。每一列表示一种属性，每一行则代表该关系所表示的实体类型的特定实例。

总体上讲，关系由两个部分组成——关系标题和关系实体。

标题是一组属性和域对。对于图 1-2，“学生信息”关系中的标题是这些对的系列：{[学号]，[姓名]，[年龄]，[系别]，[宿舍地址]}。值得注意的是，这个系列中的元素没有特定的顺序限制，所以也可以说标题是下面的序列组合：{[年龄]，[学号]，[系别]，[宿舍地址]，[姓名]}。关系中，属性和域对的数量定义为度。图 1-2 中的关系是一个五度关系。

关系的实体由一系列元组构成。图 1-2 中，标题下的每一行都代表一个元组，每一行代表一个学生的信息，类似地也可以说，每一个元组都表示“学生信息”实体类型的一个实例。正式地讲，元组是一组属性和值对。在“学生信息”关系中，[学号]：012020007 确定的元组实际上是下面一系列属性和值对{[学号：012020007]，[姓名：王平]，[年龄：23]，[系别：数学]，[宿舍地址：32#601B]}。一个元组中的一个值总是和相应的属性相匹配，导致了元组中的值是无序的。在一个关系中，元组的数目称为基数。例如，图 1-2 中的关系的基数为 4。

我们把最小属性的集合称为关系的关键字（Candidate Key）——其值能够惟一标识一个元组。图 1-2 中的关系，属性[学号]就能够作为这个关系的关键字，因为一个学号就惟一标识了一个元组。符合关键字条件的属性集合可能不止一个，理论上说它们都是关键字。但在实际系统中，我们只能选一个作为关键字，通常称为主键。在一个关系中没有特定元组顺序，因而元组就不能用行号来标识。那么如何引用一条特殊的元组呢？解决这个问题的方法是，关系中的每一个元组通过指定主键属性的值来引用。

通过名称引用属性和通过主键值引用元组，提供了一种数据模型，这种模型没有与数据组织相关的物理存储概念。对于层次、网络或者反转列表 DBMS 方法，不能使用这种语句。

因此，关系型数据模型是惟一完全实现物理数据独立性的一种方法。只要程序提供了带有主键值的实体和属性名称，关系型 DBMS 就能查到正确的相对应的数据，同时程序并不需要了解数据是如何存储的。只要 DBMS 跟踪数据的物理位置，数据就可以在程序执行时重新排列。那么对于关系型 DBMS，重要的是允许应用程序不使用物理参考来访问数据。

另外，数据结构的另外一个重要内容是域（Domain）的概念。域包含两部分信息：允许值的范围（有限或者无限）和值的含义（语义）。域不是关系的一部分，没有存储任何特殊的值。但是域定义了属性的数值范围。每一个属性只能定义在一个域上，而不同的属性可以被同时定义在同一个域上。表 1-1 举例列出了一些和基于这些域的定义的属性。

表 1-1　域和属性的例子

域 名 称	域 定 义	属 性 举 例
整　数	范围：{N|N∈[整数]}	一个月的天数，学生数
波　长	{电磁波、红外、可见、紫外、纶琴}	吸收光谱
流　量	范围：[0~100，000] 单位：吨	排水量，居民用水量
重　量	范围：[0~10，000，000，000] 单位：牛顿	汽车重量，集装箱重量

域是属性允许值的集合。由表 1-1 可以得到下面的结论。

（1）属性[一个月的天数]的值不能是小数或者字符值，因为它定义在了{整数}域上。

（2）域可以指定度量单位。这样，定义在具有相似度量单位上的域上的属性可以进行比较或者进行一定的运算，而定义在具有不同度量单位的域上的属性不能比较或者进行运算。

解决在不同单位间进行运算的方法是：利用一个映射函数。例如函数 Convert，该函数定义转变或者映射任何属性值为一个无单位值，下面的语句值得推荐："A（无单位）=Convert（B（有单位））+Convert（C（有单位））"。

域的概念允许系统检测无效的或者有问题的比较或者运算。值得注意的是，当前商业化的 DBMS 一般只是有限地支持域。

关于关系的另一点值得注意的内容是：一个数据库关系（不像关系的普通数学概念）对于某个属性不能有值集。元组中的每一个属性值必须是来自于基本域的单个元素（属性值不能是一个数值集）。为了更好地说明这一点，图 1-3 用表格形式列出了数学关系和数据库关系之间的一些区别。

学生姓名	所选课程
张浩	数学
凉天	物理
	政治
洪大	数据库
	C++语言
	大学英语

（a）数学关系

学生姓名	所选课程
张浩	数学
凉天	物理
凉天	政治
洪大	数据库
洪大	C++语言
洪大	大学英语

（b）数据库关系

图 1-3　数学关系和数据库关系的比较

1.1.2 关系数据的规范化——“范式”

考虑到关系中属性值的重复问题，应注意下面两点：

（1）在实际情况中，关系可能会出现表示信息的冗余。

（2）如果数据库文件或者表是直接与关系相对应而实现的，那么实际的冗余数据在修改时可能会产生不一致的情况。

这个时候就要求进行一定的“重新组织数据文件的过程”，这个过程称为数据的规范化。规范化的最终目的是把数据库中的数据简化为最简单的结构和最小的数据冗余，或者可以说是重新组织数据字段以达到以最有效而又灵活的方法来存储数据。形象地讲，规范过程就是把有冗余信息的关系分解成两个或者多个没有冗余的关系的过程。

在规范化的过程中有一些比较复杂的数学原理，也就是要经历一个“范式”的阶段。注意：“范式”专指一种设计概念。“范式”越到高阶，分解的限制性就越大。关系有5种主要的“范式”，每一种“范式”都描述了一种可能的冗余性。

第一“范式”就是关系自己。第一“范式”表示的数据关系不是理想模式，可能会出现信息的冗余。例如表1-2所示的就是一个典型的第一“范式”关系，每一个元组表示一个应聘人员信息。

表1-2 应聘表的第一“范式”

编　号	姓　名	应聘部门	联系电话
67	盛汝洋	市场部	13812345678
68	陈浩	市场部	13999999999
69	张杰	市场部	13622222222
70	陈浩	研究部	13999999999
71	张杰	研究部	13622222222

如果数据库文件或者表是直接与关系相对应实现的，那么在修改“陈浩”和“张杰”的信息时就会产生不一致的情况，于是就引出了第二“范式”。

第二“范式”要求清除关系中的重复数据。第二“范式”中需要避免的是重复表示的事实。在表1-2中，一个应聘人员提供惟一的联系电话，而一个应聘人员可以应聘不同的部门，于是就出现了数据重复，于是“应聘表”关系就分解成下面的两个关系，如表1-3、表1-4所示。

表1-3 分解为第二“范式”的应聘表

编　号	姓　名	应聘部门
67	盛汝洋	市场部
68	陈浩	市场部
69	张杰	市场部
70	陈浩	研究部
71	张杰	研究部

表 1-4　分解为第二“范式”的名单

姓　名	联 系 电 话
盛汝洋	13812345678
陈浩	13999999999
张杰	13622222222

第三“范式”是基于第二“范式”之上的，并且要求非主键属性只能依赖于主键。为了说明这一点，先来设计一个符合第二“范式”的关系，如表 1-5 所示。

表 1-5　应聘表

编　号	姓　名	性　别	应 聘 部 门
68	陈浩	男	市场部
69	孙美	女	市场部
70	陈浩	男	研究部
71	张杰	男	研究部

由于性别依赖于应聘人员自身，那么可以将此第二“范式”分解成下面的第三“范式”，如表 1-6、表 1-7 所示。

表 1-6　分解为第三“范式”的应聘表

编　号	姓　名	应 聘 部 门
68	陈浩	市场部
69	孙美	市场部
70	陈浩	研究部
71	张杰	研究部

表 1-7　分解为第三“范式”的性别表

姓　名	性　别
陈浩	男
孙美	女
张杰	男

分析一下就可以知道，在此第三“范式”中，姓名（代表一个应聘人员）惟一确定了性别，而不同的应聘人员可以有相同的性别。和第三“范式”中分解出来的“名单”关系有所不同的是：“名单”中不同的应聘人员不能有相同的联系电话。

第四“范式”的要求在第三“范式”的基础上更加细致。在一些关系中，存在这样的情况：有两个或者两个以上的属性，而且其中一个属性中的若干个值，它们由另一个属性的一个值决定。我们来看下面的一个关系，如表 1-8 所示。

表 1-8　应聘部门职位表

姓　名	应聘部门	应聘职位
陈浩	市场部	区域经理
陈浩	研究部	研究员
张杰	研究部	工作组长

应聘人员可以应聘不同的部门，同时也可以应聘一个部门的不同职位，于是出现了上面的情况。为了使关系更加直观，可以用下面的两个关系解决，如表 1-9、表 1-10 所示。

表 1-9　第四“范式”的应聘部门

姓　名	应聘部门
陈浩	市场部
陈浩	研究部
张杰	研究部

表 1-10　第四“范式”的应聘职位

姓　名	应聘职位
陈浩	区域经理
陈浩	研究员
张杰	工作组长

采用第五“范式”的目的是使符合第五“范式”关系的不能再分解成两个或者更多的关系，并且不丢失第一“范式”关系中的任何信息。通过合理地分解关系，能够清晰地表示关系。设计数据库时，能够准确地得到这些关系集是至关重要的。

总之，关系模型的数据结构和规范化为理论和实际相结合提供了非常好的体系，并且当查看这些由表表示的关系时，能够更好地理解和使用关系。前面说过，关系是 DBMS 的关键理论基础。数据结构概念的简洁性与关系的规则定义使得关系模型形成下面将要讲到的两个重要部分：数据完整性和数据操纵。

1.1.3　关系数据模型的完整性约束

在关系数据模型中，关系的元组除了要满足数据结构的定义外，还要受到数据有效性的约束。完整性的含义定义了确保所存储有效数据的机制。这些机制可能会限制某些属性的取值，也可能会制约不同关系或同一关系的属性的值之间的关系，并且要求每一个属性值是有效的，元组中的数值是惟一的，并且在这些元组中，彼此相关的关系有一致的数值。完整性约束是保证数据库的一致性和和正确性的重要手段。而实现完整性约束的重要工具是关系的关键字。一般来说，关系数据模型有三种类型的完整性约束。

（1）实体完整性约束（Entity Integrity Constraint）

这是一个很直观的概念。首先，主关键字的值不能重复，这就要求，每一个元组代表在

现实世界中客观存在的实体，在一个关系中，不能存在完全相同的两个元组；第二，关系中的主关键字不能为空。只要在关系模式中定义了主键，就可以实现这一约束。

（2）域完整性约束（Domain Integrity Constraint）

关系模式中属性的值应是域中的值。在关系模型中，一个普遍接受的变化是允许属性包含一个记号——NULL，它表示数据的丢失。应该注意的是，NULL 不是一个值，而只是一个占位符。

（3）参照完整性约束（Referential Integrity Constraint）

参照完整性要求存在于不同关系中但彼此相关的记录是由相应属性值明确关联的。也就是说，如果关系仅有外关键字 M 对应关系 P，那么 M 要么取 NULL 值，要么取 P 中存在的值，且 M 的某些属性值不与 P 中任何一个已有的元组的主键值相匹配。这是针对不同关系之间或同一关系中的不同元组之间的约束。

任何对关系型数据库的修改，关系型 DBMS 都可以自动强制这些约束，这些约束不仅保护了属性的值，同时也保护了元组的惟一性和相互关系。提供了这种完整性支持的 DBMS 大大地减少了应用程序设计人员的编程工作量。

1.2 关系数据模型的数据操作

关系数据模型定义了关系上的一组操作。这些操作的表示方法通常有两种：

（1）关系代数的代数符号。通过特殊的运算符号来表示关系上的操作。

（2）关系演算的逻辑符号。也就是用目标元组必须满足的逻辑公式来表示关系上的操作。

在这里，关系演算就不做介绍了，只介绍关系代数。

关系代数运算包括四个标准的集合运算和四个针对数据库的关系运算。集合运算包括集合的并（∪）、交（∩）、差（-）、笛卡尔积（×）等；关系运算包括选择（σ）、投影（Π）、连接（∞）、除（÷）运算等。关系运算的对象是关系，运算结果仍然是关系，结果也可以再次作为运算对象进行各种运算，由此可以构成各种复杂的关系代数表达式，并可以表示各类十分复杂的查询。在数据库上做的各种各样的查询一般都可以用一个关系代数的表达式来表示。

1．集合运算

集合运算中，并、交、差运算要求两个关系有相同的属性，也就是要求这些关系必须是兼容合并的。而笛卡尔积运算不要求两个关系具有相同的属性，如果其中一个关系有 M 种属性，另一关系有 N 种属性，那么笛卡尔积运算结果就会有 M+N 种属性。并、交和笛卡尔积是无顺序运算，运算因子的顺序无关紧要。比如："关系 M 并 N" 等同于 "关系 N 并 M"。而差运算是有顺序的。

并：关系 M 和 N 的并为 M∪N，结果由在 M 中、或者在 N 中的所有元组组成。

交：关系 M 和 N 的交为 M∩N，结果是由同时在 M 和 N 中的元组组成。

差：关系 M 和 N 的差为 M-N，结果是由在 M 中而不在 N 中的元组组成。

笛卡尔积：如果关系 M 和 N 的关系模式为：M(M1，M2，M3，…，Mm)，N(N1，N2，N3，…，Nn)，那么关系 M 和 N 的笛卡尔积 P=M×N 也是一个关系，P 的关系模式为 (M1，M2，M3，…，Mm，N1，N2，N3，…，Nn)。M 和 N 的笛卡尔积的关系模式的属性集包括了 M 和 N 的所有属性。

四种运算中，关系的交不是基本运算，它可以由关系的差表示，即 M∩N=M－(M-N)。其余的三个运算是关系代数的基本运算。

如下面的两个关系：

M

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

N

A	B	C
b	g	a
d	a	f

那么这两个关系的四个集合运算可分别表示如下：

M∩N

A	B	C
d	a	f

M-N

A	B	C
a	b	C
C	b	d

M∪N

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

M×N

M。A	M。B	M。C	N。A	N。B	N。C
a	b	c	b	g	a
d	a	f	b	g	a
c	b	d	b	g	a
a	b	c	d	a	f
d	a	f	d	a	f
c	b	d	d	a	f

2．关系运算

关系运算包括四种运算。

（1）选择运算

选择运算是在一个关系 M 上生成新的关系，新的关系是 M 的子集，新关系中的元组要满足选择的条件，并且新关系和 M 具有相同的关系模式。具体来讲，选择运算是按照给定的条件在表（关系）上选择出符合条件要求的行（元组）。选择运算可以表示为：

$\sigma_S(M) = \{\sigma_S(t) \mid t \in M\}$，

其中，$\sigma_s(t) = \begin{cases} t & t\text{满足条件}S \\ \Phi & \text{不满足} \end{cases}$

t 为 M 中的元组，而 S 就是条件表达式，它和一般的程序设计语言中的条件表达式相似，只是运算对象限制为常量和关系模式的属性名。 在选择运算进行过程中，关系 M 的每个元组都被用来判断是否符合关系 S，对条件 S 中每个关系 M 的属性名，用该元组的对应分量代替，如果条件表达式的计算结果为真，那么该元组就被包括在选择的结果中；如果计算结果为假，此元组就不在选择结果中。

S 是一个公式，它有常量和关系模式的属性名、算术比较运算符（<，=，>，≤，≥，≠）和逻辑运算符（∧（与），∨（或），¬（非））等组成。

（2）投影运算

投影运算是在关系上选择多个属性组成新的关系，并可以重新排列属性。没有被选择的属性被剔除后，新的关系中有些元组可能变得相同，投影运算就可以删除这些重复的元组。具体来讲，投影在一个表上选择某些列的数据，结果得出的关系的关系模式的属性减少了。作用在关系 M 上的投影运算可以表示为：

$\prod_p(M) = \{\prod_p(t) \mid t \in M\}$

M(M1，M2，…，MM)，假设 P={Mi1，Mi2，…，Mik}(k≤M)，且 P⊆{M1，M2，…，MM}，t=(t1，t2，…，tM)，t 是 M 的一个元组，那么：

$\prod_p(t) = \prod_{Mi1Mi2\ Milk}(t) = (ti1, ti2, \cdots, tik)$

注意：投影运算作用到一个关系上依然得到一个关系，作用到一个元组上则得到一个元组。

（3）连接运算

连接运算等价于两个关系进行乘积（笛卡尔积）后，再从中选择符合条件的一些元组。连接运算不是关系代数的基本运算，但是对于两个关系的关联，它起着非常大的作用，所以也把它列为关系运算的一员。一般来讲，连接运算有四种。

1）条件连接：关系 M 中的某个属性 A 和关系 N 中的某个属性 B，规定它们之间的关系是：AθB（其中θ 取自≤、≥、<、>、=和≠等关系运算符）。那么关系 M 和 N 在条件 AθB 下的连接可以表示为：

$M_{A\theta B}^{\infty}N = \sigma_{A\theta B}(M \times N)$

结果是从两个关系的笛卡尔积中选取属性值满足条件 AθB 的元组组成的关系，结果关系的关系模式和笛卡尔积 M×N 的模式相同。

2）自然连接：自然连接属于条件连接，不过其连接条件被默认为在两个关系中的同名属性的值相等。计算关系 M 和 N 的自然连接时，首先计算笛卡尔积 M×N，然后用同名属性的值相等的条件来选择笛卡尔积 M×N 中的元组，当然，最后应当去掉结果中重复的属性。自然连接可以表示为：

$M \infty N = \{m \infty n \mid m \in M, n \in N\}$

可以设 M(A1，A2，…，Aj，B1，B2，…，Bk)，N(B1，B2，…，Bk，C1，C2，…，Cp)，则公共属性集为(B1，B2，…，Bk)，元组 m∈M，n∈N，m=(a1，a2，…，aj，b1，b2，…，bk)，n=(b1，b2，…，bk，c1，c2，…，cp)，于是元组的自然连接表示为：

$$m \infty n = \begin{cases} (a1\ a2 \cdots aj\ b1\ b2 \cdots bk\ c1\ c2 \cdots cp) & \text{如果符合 M.bi = N.bi，(i = 1~k)} \\ \Phi & \text{不符合} \end{cases}$$

特别地，如果 M 和 N 没有公共属性，自然连接就没有实际意义，如果 M 和 N 的关系模式完全相同，那么 M∞N=M∩N。

3）半连接：半连接就是关系 M 和 N 的自然连接在 M 的属性集上的投影。明确地说，它相当于根据同名属性的值相等的条件在关系 N 中找匹配（同名属性值相等）的元组，找到了就保留，同时对 M 的元组进行选择。注意：当关系 M 和 N 的关系模式相同时，半连接运算类似于集合的差运算。

4）外连接：外连接和自然连接的区别在于：保留被运算量的关系中在另一关系中找不到匹配的元组的元组，其空缺的另一关系的属性的值用空值 NULL 填充。一般来讲，外连接有三种：

- 左外连接：表示为 M*∞N，在结果中，保留关系 M 中的所有元组；
- 右外连接：表示为 M∞*N，在结果中，保留关系 N 中的所有元组；
- 全部连接：表示为 M*∞*N，在结果中，保留关系 M 和 N 中的所有元组。

（4）商运算。

设有两个元组 M 和 N，如果可以做 M÷N 商运算，那么就必须要求 M 关系模式中的属性数目多于 N 关系模式中的属性数目。M÷N 也是一个关系，它的任意一个元组和 N 中的任意一个元组结合在一起，必须构成 M 中的某一元组。就是说，关系 M 和 N 的商是关系 M 在不属于关系 N 的属性上的投影。可以用其他的关系运算来表示商运算，设 X 为属于 M 而不属于 N 的属性集合，那么商运算表示为：

$M \div N = \prod_X (M) - \prod_X ((\prod_X (M) \times N) - M)$

如果 T=M÷N，那么 $T \times N \subseteq M$

比如有下面两个关系：

M

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

N

C	D
c	d
e	f

那么 M÷N 为：

A	B
a	b
e	d