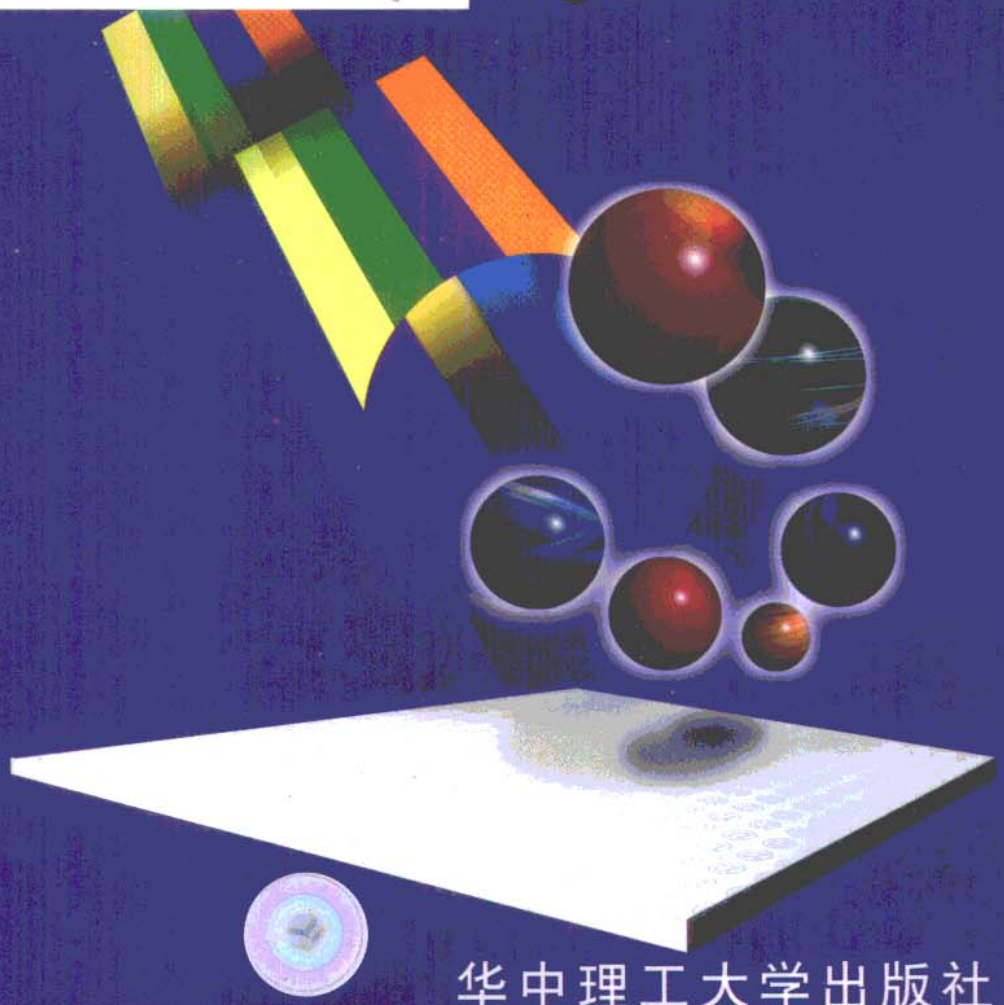
计算机实用技术培训·应用丛书

Delphi 5.0 实用培训教程

王宏　李冬　编著

抖斗书屋　审校



华中理工大学出版社

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY PRESS

E-mail: hustpp@wuhan.cngb.com

Delphi 5.0 实用培训教程

王 宏 李 冬 编著

抖斗书屋 审校

华中理工大学出版社

中国·武汉

图书在版编目(CIP)数据

Delphi 5.0 实用培训教程/王 宏　李 冬　编著
武汉:华中理工大学出版社，　2000 年 7 月
ISBN 7-5609-2240-6

Ⅰ.D…
Ⅱ.①王…　②李…
Ⅲ. 软件工具,Delphi 5.0-教程
Ⅳ.TP311.56

Delphi 5.0 实用培训教程　　王 宏　李 冬　编著

责任编辑:周筠　陈冰　　封面设计:潘　群

责任校对:蔡晓珊　　责任监印:熊庆瑜

出版发行:华中理工大学出版社　　武昌喻家山　邮编:430074　电话:(027)87545012

经销:新华书店湖北发行所

录排:华中理工大学惠友科技文印中心
印刷:华中理工大学出版社沔阳印刷厂

开本:787×1092　1/16　　印张:22.25　　字数:494 000
版次:2000 年 7 月第 1 版　　印次:2000 年 7 月第 1 次印刷　　印数:1—5 000
ISBN 7-5609-2240-6/TP・389　　定价:29.80 元

前　言

当今计算机技术最活跃的研究领域之一就是数据库编程技术，以关系数据库为代表的数据库产品已走向成熟。小型数据库（如 FoxPro，Access，Paradox 等）百花争艳，大型数据库（如 Oracle，Sybase，Ingres，Informix）分割天下。随着数据库技术在各个领域的广泛应用，用户对应用程序的数据库访问和管理功能也提出了更高的要求。

为了满足市场的需要，许多大型软件公司的程序开发工具包都加强了开发数据库应用程序的特性。而 Delphi 5.0 正是目前众多开发工具中性能卓越而又使用简便的一种。鉴于 Delphi 的各种优秀特性，越来越多的程序员已经开始转向使用 Delphi 开发程序。同时，由于 Delphi 方便易用，因而它也成为许多初学者喜爱的入门软件。总而言之，Delphi 正逐渐被越来越多的人所接受。

Inprise 公司的 Delphi 面向对象软件包从 3.0 版本开始就已经提供了强大的数据库开发能力。它的 BDE（Borland Database Engine）数据引擎包含了对大量的数据库驱动程序的支持，并且对不同的数据源，如 Paradox、dBASE、FoxPro、Access 和文本数据库等提供了一个一致的数据访问接口。多年的应用实践证明，基于 Windows 的 32 位 BDE 作为 Inprise 的核心数据库引擎和连接工具是非常优秀的。

在 Delphi 5.0 中，使用了 BDE 的最新版本 BDE 5.0，它不但继承了以前各种版本的优秀特性，而且还增加了对 Oracle 8.0 的全面支持，并支持 Microsoft 事务服务器。

Delphi 5.0 在数据库技术方面的另一重大突破是增添了对 ADO（ActiveX Data Objects）的支持。ADO 是由 Microsoft 公司开发的一种实现数据库一致性访问的技术。通过 ADO 技术，使用 Delphi 5.0 就能够开发出快速访问各种关系数据库、非关系数据库的应用程序。

本书重点介绍了 Delphi 5.0 在数据库应用程序开发方面的应用，书中由浅入深地介绍了 Delphi 5.0 中有关数据库应用程序开发的各种知识。其中既有对数据库组件的属性、方法的详细介绍，又有针对目前各种流行技术的讨论，包括对 BDE、ADO 技术的发展历程和技术原理的详细论述。

在介绍知识的同时，为了使读者能够在实践中掌握知识，本书在许多章节的最后都特意附加了一个应用程序的范例，作为对应章节内容的总结和概括。

真诚希望我们的劳动能够给读者带来收获，读者的收获是我们最大的成功。由于时间的限制，而且编者水平有限，错漏之处在所难免，恳请读者指正。

本书由中科辅龙计算机技术有限公司抖斗书屋策划，主要部分由王宏、李冬编写。另外参加编写的还有史惠康、郭美山、宋新波、郑红、刘小华、魏红、王艳燕。全书由杨桂莲、石利文统稿，徐平校排。

抖斗书屋坐落于中科院计算所院内，由中科辅龙计算机技术有限公司领导，是一家拥

有雄厚实力的计算机图书创作单位。在本书的编写过程中，书屋的全体员工都付出了大量劳动，借此机会对书屋全体人员的精诚团结与支持表示由衷的感谢！

本书的出版得到了华中理工大学出版社计算机编辑室的大力支持，在此表示由衷的感谢。

对本书内容有疑问的读者，可向抖斗书屋读者服务部提出咨询。

咨询电话：010-62565533 转 3301

E-mail：replybook @ 126.com

史惠康

2000 年 4 月于中科院计算所

第 1 章　数据库基础知识

1.1　数据库技术概述

当今，计算机已经被广泛应用于科技文化、组织管理等各个领域，以及国民经济的各行各业和日常生活的各个方面。在众多的应用中，计算机的作用已不仅仅是进行数值计算，而更多的是用于数据的加工和管理。为了有效地对这样一些数量庞大，而且结构比较复杂的数据进行处理，需要有专门的技术。数据库技术正是为了满足这种需要而发展起来的，由于数据库技术的不断普及和深入，它已经成为计算机应用中必须掌握的重要技术之一。

数据管理技术的发展是与计算机技术及其发展联系在一起的，它大致经历了人工管理、文件管理和数据库管理三个阶段。

数据库技术的根本目的就是解决数据的共享问题。当然，数据管理技术中由文件系统管理阶段过渡到数据库技术管理阶段，其前提是计算机硬件和软件技术有了进一步的发展，外存有了大容量的硬盘，硬件价格下降，软件价格上升，使编制系统软件和应用程序成本相对增加。因而使数据共享问题成为数据管理中一个亟待解决的关键问题。也正是这个问题的解决，使得数据库管理具有以下特点：

- 数据是结构化的，是面向系统的，数据的冗余度较小，节省了磁盘空间，而且也减少了数据的存取时间，提高了访问速度，避免了数据的不一致性，同时提高了可扩充性和数据应用的灵活性；
- 数据具有独立性；
- 保证数据的完整性、安全性和并发性。

数据库应用程序和数据库的关系可以用图 1-1 来表示。

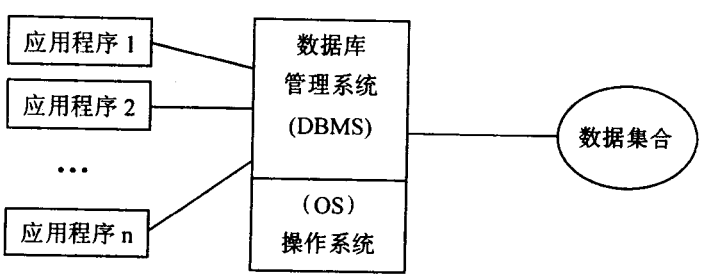


图 1-1　数据库应用程序和数据库的关系

数据库最为本质的特征就是实现数据的共享。为了实现数据的共享，保证数据的独立性、完整性和安全性，需要有一组软件来管理数据库中的数据，处理用户对数据库的访问，这组软件就是数据库管理系统 DBMS。数据库管理系统是数据库系统的核心，它与计算机

系统内其他的软件一样，也在操作系统的支持下工作，它与操作系统(OS)的关系十分密切，有时互相结合成为一个整体。操作系统、数据库管理系统以及应用程序在一定的硬件的支持下就构成了数据库系统。

数据库管理系统是数据库系统中实现各种数据管理功能的核心软件。它负责数据库中所有数据的存储、检索、修改以及安全保护等等，数据库内的所有活动都是在其控制下进行的。数据库管理系统依赖于操作系统的支持，但作为一个管理数据的独立的软件系统，较之计算机内其他的软件，它有着自己的一些特点。例如，它具有一套独立于操作系统的存取数据的命令，数据存储空间的分配由 DBMS 本身来完成，等等。一个数据库管理系统通常具有多种类型的操作系统的接口软件，因此，数据库管理系统作为一个子系统可以方便地安装到多种类型的计算机系统上。

数据库管理系统具有较强的对数据进行控制的能力，它包含了各种类型的系统程序，就大型的数据库系统而言，其复杂程度可能远远超过一个操作系统。一般来说，它具有以下一些功能：

- 定义数据库：包括全局逻辑数据结构的定义，局部逻辑结构的定义；
- 管理数据库：包括控制整个数据库系统的运行，数据存取、插入、删除和查询等操作，以及数据完整性和安全性控制，并发控制等等；
- 建立和维护数据库：包括数据库的建立、数据更新、数据库再组织、数据库的维护、数据库恢复以及性能监视等等；
- 数据通信：具备与操作系统的联机处理、分时系统及远程作业输入的相应接口。

一个数据库系统是由操作系统、数据库管理系统和应用程序在一定的硬件支持下构成的。也就是说，数据库系统不是数据库本身，也不仅指数据库管理系统，而是指计算机系统中引进数据库以后的整个系统。对于较大型的数据库系统来说，通常还应有数据库管理员 DBA(Database Administrator)。

进入计算机中的数据可以代表各种各样的信息。为了对数据进行有效的管理，首先必须考察和分析反映客观事物的各种信息及其相互之间的联系，以便确切地用数据来描述它们。

客观存在的事物之间存在着各种各样的联系，在描述客观事物时除了描述其本身之外，还要描述相互之间的联系。

客观事物之间的联系有两个方面：一方面是实体内部的联系，在数据模型中表现为记录内部的联系；另一方面是实体与实体之间的联系，在数据模型中表现为记录与记录之间的联系，文件系统中的数据是面向应用的，仅考虑记录内部的联系，不考虑文件之间的联系。所以就整体而言，数据是无结构的。而数据库中的数据是结构化的，面向整体组织的数据，不仅要反映实体内部的联系，而且还要反映实体之间的联系。也就是说，除了考虑记录内部的联系之外，还必须考虑文件之间的联系。为了描述这些联系，就必须分析实体之间的各种关系，并且要对这些复杂关系进行归纳、抽象，以便建立起数据模型。

数据模型是对客观事物及其联系的数据描述，它反映了实体内部以及实体与实体之间的联系，是数据库设计的核心。

在数据库中，数据模型可以被分成如表 1-1 所示的三个层次。

表 1-1　数据库数据模型的三个层次

层　次	说　明
外层	反映局部的逻辑结构
概念层	反映全局的逻辑结构
内层	反映物理数据存储的模型

用树型结构来表示实体之间联系的模型叫层次模型。建立数据的层次模型需要满足下列两个条件：

- 有一个数据记录没有“父亲”，这个记录即是根节点；
- 其他数据记录有且仅有一个“父亲”。

网状数据模型是以记录为节点的网状结构，它的特点是：

- 可以有一个以上的节点无“父亲”；
- 至少有一个节点有多于一个的“父亲”。

由以上两个特点可以知道，网状模型可以描述数据之间的复杂关系。

关系模型是一个和格式化模型完全不同的数据模型。关系模型与层次模型、网状模型相比有着本质的差别，它用表格表示实体本身及其相互之间的联系。

在关系模型中，数据被看成一个二维表，每一个二维表称为一个关系。对于表示关系的二维表，其最为基本的要求就是表中记录的每一个数据项必须是不可分的数据项，即不允许表中再有表。

“关系”是关系模型中最基本的概念。关系是记录的集合，从集合的意义上讲，可以给出关系的严格的数学定义。

关系模型较之其他的模型来说，有着如下一些优点：

- 数据结构简单；
- 数据独立性很高；
- 可以直接处理多对多的联系；
- 有坚实的数学理论基础。

除了上述三种模型以外，还有近年发展起来的其他一些数据模型，如面向对象的数据模型以及关系和对象的混合模型等等。

1.2　关系数据库基础

1.2.1　关系数据库概述

现在大多数数据库系统都是基于关系数据库管理系统(Relation Database Management System，RDBMS)。SQL(Structured Query Language，结构化查询语言)是 RDBMS 的标准数据库语言。

关系模型最早是由 E.F.Codd 在 20 世纪 70 年代提出的。最早的 RDBMS 商业产品出现

在 20 世纪 80 年代初，如 Oracle。虽然今天基于新的数据模型的商业数据库管理系统已经出现，但是 RDBMS 仍然是数据库市场的主流，而且这种主流地位估计还将持续若干年。

关系数据库管理系统总的说来有以下优点：

- 面向集合的处理；
- 数据的逻辑独立性；
- 数据的自动导航。

关系模型是构成关系数据库管理系统的基础，一个关系模型主要由以下三部分组成：

- 数据结构；
- 完整性规则；
- 数据操作。

关系模型的数据结构很简单，即它是一个元组和列组成的二维表。一个表中的元组必须是互不相同的；列值必须具有原子性(即不能再分割)，数组或集合都不能作为列值。关系表中的元组和列都是无序的。

在关系模型中，表中的元组都是唯一的。即存在一个或多个列，其中每一个元组上的值都互不相同。因此可以用某一个或一组列来唯一标识一个元组，这样的列称为表的主码。关系模型对主码有如下的规则：即主码的任一属性不能为空。这就是所谓的实体完整性。这是因为主码通常是用来标识表中的元组的，带有空值的主码无法标识元组。

在实现关系数据库时，必须能够表示表之间的联系。为此，需要在一个表中含有另一个表的主码，这样的属性称为外码。外码列或者为空或者为它所参照的表的主码的某一个值，这就是关系模型的第二个完整性规则，称为参照完整性。

关系模型内容简单，但功能强大。这是因为在其中定义了很多的操作，这些操作包括选择、投影、连接和并：

- 选择(Selection)操作能够返回一个表的元组的一个子集。它用于表的横向元组的选择；
- 投影(Projection)操作能够返回表的列的一个子集。它用于对表的纵向列的投影；
- 连接(Join)操作用于从多个表中取得结果。虽然连接操作的种类很多，但最常用的是我们所说的自然连接。自然连接是指将一个表的属性与另一个表中的属性进行匹配连接。两个表通常含有一个同名的列。按同名列自然连接后的结果集中不包括不匹配的元组；
- 并(Union)操作可以垂直地把不同的表中的元组组合成一个结果。并操作的两个表，其列必须是可相容的。这就是说，两表对应的列类型相同。这一点是很显然的，不同类型的列值是不能拼接在一起的。

1.2.2 关系数据库设计

关系模型的目标之一就是要让数据库能够被用户和程序员双方都理解。要达到这一点并建立稳定的系统，就要认真对待数据库的设计。大致说来，一个数据库的设计可以分为以下三个主要步骤：

- 定义实体；
- 定义联系；

● 定义属性。

按照以上次序进行数据库设计是非常重要的，否则在设计时难免会漏掉一些实体或者联系。

注意：在数据库设计中，对表和实体的命名一般采用单数名词。

当为一个大型复杂数据库系统设计数据库时，要按照系统的主题组织数据库的设计。在商业数据库系统中，这样的主题有客户、产品、业务等。如果围绕这些主题来组织数据库，就可以实现最大可能的数据共享。

1. 定义实体

设计数据库的第一步就是定义实体．实体实际上对应了数据库中的表。所谓实体就是一个人、一个事物或数据库中要存储的一个对象。在商业应用系统中，实体包括雇员、产品、合同和订货单等等。作为实体，必须能够与其他实体相区别。对不能明确区分的对象，应不按实体对待。

注意：实体一旦选定，就要为其选择主码，主码必须是唯一的且不允许为空值。

如果某一个实体多次出现，最好选择由系统自动生成的值为其主码。

某些实体的存在要依赖其他的实体，具有这种依赖性的实体被称为依赖实体。被依赖的实体称为父实体，依赖实体的主码总是由多个列组成，其中某一个列一定是父实体的主码。

在确定一个实体为依赖实体之前必须检查其是否具备以下条件：

● 必须有一个父实体；
● 最多只有一个父实体；
● 其父实体总是同一个．

2. 定义联系

实体确定之后，就可以着手定义实体之间存在的联系，这种联系可以先用相当机械的方式进行，然后再检查数据库中每对实体之间是否都有联系。

实体之间的联系有以下三种类型：

● 一对一；
● 一对多；
● 多对多。

其中一对一和一对多的联系可以通过在某一实体中引入另一实体的主码来实现．多对多的联系则需要一个专门的表来实现。

当某一个实体最多只能与另外一个实体相联系时，则称这种联系是一对一的。要建立一对一联系，只需将一个实体的主码列放入另一个实体中即可．

一对多联系是一种最为常见的联系．它是指一实体 A 最多只能与另外一个实体 B 相关，而实体 B 可以与实体 A 等多个实体相关。

所谓多对多联系，是指一实体 A 可以与另一个实体 B 的多个实体相关，反之亦然．多

对多联系的处理方法与上述两种情况不同，多对多联系需要新增一个表，这个新增的表是由各个实体的主码构成的。

3. 定义属性

数据库设计的目的是为了减少数据的冗余度并便于维护，为了实现这一目的，在数据库中引入规范化的概念。所谓规范化是一种基于数学理论的数据库处理方法，通过它可以产生理想的数据库。规范化规则有助于确定属性的定义。

规范化主要基于范式的概念。当一个表满足某种条件时，即称该表是某一范式的。数据库中一共定义了五种范式，这里着重介绍前三种范式。

当一个表的任何一列均只取原子数据时，则称该表是第一范式的，这也是为什么数据库管理系统都不支持数组、表列和嵌套关系的原因。

第二范式用于有多列主码的表。若一个表满足第一范式的要求并且所有非主码列的值都依赖所有的主码列，则称这样的表符合第二范式。

如果一个表已经满足第二范式的要求并且所有列依赖于主码或者候选码，则称该表是第三范式的。

在设计数据库时，通常会在常规意义上把数据库作规范化处理。虽然规范化显得有些复杂，但它能帮助人们对表选择正确的列。其规范化的要求也是很自然的事。

在知道如何确定表的属性之后，实现设计思想之前还需做以下事情：

- 为每一个列选择数据类型；
- 为每一列选择长度、精度或标度；
- 确定列是否允许空值；
- 确定列的默认值；
- 确定当用户没有给列赋值时，是否由数据库管理系统为之设置默认值；
- 确定列的值是否是唯一值；
- 确定列所要满足的约束条件。

1.3 结构化查询语言(SQL)

SQL 是关系数据库存取的工业标准语言。它在数据库行业中应用相当普遍，以至于支持其它模型的数据库也使用 SQL。各个数据库开发商对 SQL 又作了极大的扩充，使其远远超出了数据库语言的范围。经扩展的 SQL 甚至可以作为一种成熟的程序设计语言。

1.3.1 SQL 的数据类型

SQL 中定义了三级一致性：即最低限度(Minimum)、核心(Core)和扩展(Extended)。这些一致性级别用于数据类型。

最低限度一致性级别定义了如下的字符数据类型：

- CHAR(n)——固定长度为 n 的字符串；
- VARCHAR(n)——可变长度字符串，最大长度为 n；

- LONG VARCHAR——可变长度字符串。

核心一致性级别定义了如下的数字数据类型：

- DECIMAL(p,s)或 NUMERIC(p,s)——精度为 p，标度为 s 的精确数值。
- SMALLINT——两字节整数；
- INTEGER——四字节整数；
- REAL——四字节浮点数；
- FLOAT 或 DOUBLE PRECISION ——八字节浮点数。

扩展一致性级别定义了以下附加的数据类型：

- BIT——单个二进制位；
- TINTINT——一字节整数；
- BIGINT——八字节整数；
- BINARY(n)——固定长度的二进制数，长度为 n；
- VARBINARY(n)——可变长度的二进制数，最大长度为 n；
- LONG VARBINARY——可变长度的二进制数；
- DATE——日期；
- TIME——时间；
- TIMESTAMP——时间戳(日期和时间)。

数据库还可以定义其它数据类型(如货币类型)．如果数据库允许用户自定义数据类型，则可以在数据定义语句中使用这些已定义的数据类型．

1.3.2　SQL 语句

SQL 包含了一套用于操纵数据库的语句集合．这些语句根据其实现的功能的不同，可以大致划分为四大类：

- 数据定义语言；
- 数据控制语言；
- 数据操纵语言；
- 数据查询语言．

数据定义语言(Data Definition Language，DDL)是一种定义对象的语言或其子集．SQL 使用以下三个基本命令来定义 SQL 对象：

- CREATE
- ALTER
- DROP

数据控制语言(Data Control Language，DCL)用于定义数据库用户的权限．SQL 使用下面两条基本语句来实现权限管理：

- GRANT
- REVOKE

数据操纵语言(Data Manipulate Language，DML)是一组操纵表中数据的语句．SQL 使用下面三条基本语句来实现数据操纵：

- DELETE
- INSERT
- UPDATE

数据查询语言(Data Select Language，DSL)是一组用于查询数据库中的数据的语句。SQL使用 SELECT 语句来实现数据查询。

1. 数据定义语言

数据定义语言是一种定义对象的语言或者语言的子集。SQL 使用以下三种命令来定义SQL 对象：

- CREATE
- ALTER
- DROP

这些命令可以用来创建、定义、修改和删除表、索引及视图。

CREATE 语句用于创建一个新表，其使用格式如下：

```
CREATE TABLE table_reference
    (column_definition [, column_definition,...] [, primary_key_constraint])
```

例如，可以通过如下语句来定义一个表。

```
CREATE TABLE TEST
(
    ID INTEGER,
    NAME CHAR(20),
    GENDER CHAR(1),
    AGE INTEGER,
    INCOME NUMBER(6,2)
)
```

上面的例子只是 CREATE 语句的最为一般的用法。实际上，CREATE 语句还可以有很多的选项，这些选项包括定义数据表的外部约束和内部约束等。注意在创建数据表时，数据的一致性和可移植性之间要有一个权衡。在共享的数据库环境中，确保所有的应用程序都提供合适的默认值是非常困难的。在这些权衡中，出于数据库管理的考虑，一般总是偏向于数据库的完整性。考虑上述因素后，上面的例子可以改写成下面的语句。

```
CREATE TABLE TEST
(
    ID INTEGER    NOT NULL REFERENCE TEMP STU_ID UNIQUE,
    NAME CHAR(20)    NOT NULL,
    GENDER CHAR(1)    NOT NULL,
    AGE INTEGER,
    INCOME NUMBER(6,2)
)
```

在上述语句中定义了 ID 域不能为空，且其值应在 TEMP 表的 STU_ID 域中，同时这

个域的值在整个表中必须是唯一的。这样，当向表 TEST 中插入数据时，如果不满足上述约束条件，则数据库管理系统将拒绝执行这样的语句。

要删除一个表可以通过 DROP 语句来实现，其使用格式如下：

```
DROP TABLE table_reference
```

例如，要删除上例中定义的表 TEST，可以通过如下语句来实现。

```
DROP TABLE TEST
```

部分数据库管理系统还支持一种被称为级联删除的 DROP 语句，所谓级联删除是指所删除表中的完整性约束涉及此表，则其他的表和完整性约束也同时被删除。例如，下面的语句在删除表 TEST 的同时，还会删除表 TEMP。

```
DROP TABLE TEST CASCADE
```

ALTER 语句用于对表进行修改，其使用格式如下所示：

```
ALTER TABLE table_reference
    DROP [COLUMN] column_reference | ADD [COLUMN] column_reference
    [,reference DROP [COLUMN] column_reference | ADD [COLUMN] column_reference...]
```

例如，希望向表 TEST 中增加一个名为 ADDRESS 的列，可以通过下面的语句来实现。

```
ALTER TABLE TEST
    ADD COLUMN
    ADDRESS CHAR(20)
```

要删除刚才所增加的列，可以通过下面的语句来实现。

```
ALTER TABLE TEST
    DROP COLUMN ADDRESS
```

2. 数据控制语言

数据控制语言是一种用于定义数据库用户权限的 SQL 子集。它主要包括下面两种命令。

- GRANT
- REVOKE

其中，GRANT 语句用于对用户授权，而 REVOKE 语句则用于回收权限。例如，下面的语句用于给当前的所有用户授予对表 TEST 的查询操作。

```
GRANT SELECT ON TEST TO PUBLIC
```

其中，PUBLIC 表示当前数据库中所有被定义过的用户。

在数据库中可以授予的权限如表 1-2 所示。

表 1-2　数据库中的权限

权限名	说　明
SELECT	查询权限
INSERT	插入数据的权限
DELETE	对数据进行删除的权限
UPDATE	对数据库进行更新的权限
REFERENCE	允许用户涉及在一完整性约束中指定的一列

回收授予所有用户的查询权限，可以通过如下语句来实现。

```
REVOKE SELECT ON TEST FROM PUBLIC
```

3. 数据操纵语言

SQL 中的数据操纵语言是一组操纵表中数据的语句。它包括以下三种语句：

- INSERT
- DELETE
- UPDATE

其中，INSERT 语句用于向表中插入数据，其使用格式如下所示：

```
INSERT INTO table_reference
    [(columns_list)]
    VALUES (update_atoms)
```

例如向表 TEST 中插入一条记录可以通过如下语句来实现。

```
INSERT INTO TEST
    VALUES(1,'John','F',20,2000.40)
```

上述语句中没有指定列名，因此必须给出所有列的值，如果指定特定的列，则只需插入相应的列即可。例如，下面的语句只向表 TEST 中插入 ID、NAME 和 GENDER 三个字段的值。

```
INSERT INTO TEST
    (ID,NAME,GENDER)
    VALUES(1,'John','F')
```

在插入数据时，还可以用参数代表要插入的数据，这时相应字段的位置上应设置为“？”号，例如下面的语句要求应用程序指定参数才能完成数据的插入。

```
INSERT INTO TEST
    VALUES(1,'John','F',20,? .? )
```

DELETE 语句用于从表中删除数据，其使用格式如下：

```
DELETE FROM table_reference
    [WHERE predicates]
```

例如，删除表 TEST 中的所有记录，则可以通过如下语句来实现。

```
DELETE FROM TEST
```

可以设置一定的条件，使得 DELETE 语句仅删除符合指定条件的记录。例如，下面的 DELETE 语句删除满足 ID 为 5 的记录。

```
DELETE FROM TEST WHERE ID=5
```

UPDATE 语句用于改变数据库中某条记录的值，其使用格式如下所示：

```
UPDATE table_reference
    SET column_ref = update_atom [, column_ref = update_atom...]
    [WHERE predicates]
```

例如，下面的语句更新表 TEST 中 ID 为 5 的记录。

```
UPDATE TEST
```

```
SET AGE = 25
WHERE (ID = 5)
```

4. 数据查询语言

SELECT 命令用于对数据库中的数据进行查询并把结果返回给用户，它是具有许多选项的功能强大的语句，利用它可以完成关系数据模型中描述的关系操作。SELECT 语句的使用格式如下：

```
SELECT [DISTINCT] * | column_list
    FROM table_reference
    [WHERE predicates]
    [ORDER BY order_list]
    [GROUP BY group_list]
    [HAVING having_condition]
```

例如，下面的语句用于查询表 TEST 中的所有数据。

```
SELECT * FROM TEST
```

如果希望查询的结果中只显示某几个指定的列，则可以通过如下的语句来实现。

```
SELECT ID,NAME,GENDER FROM TEST
```

上述两个语句都将返回表 TEST 中的所有记录，通过 WHERE 子句可以过滤返回的数据。例如，查询表 TEST 中所有 AGE 字段的值为大于 20 的记录，可以通过如下的语句来实现。

```
SELECT * FROM TEST WHERE AGE>20
```

上面的语句使用了“>”运算符，在 SQL 中支持如表 1-3 所示的各种运算符。

表 1-3　SQL 支持的运算符

运算符	说　明
>	大于
<	小于
=	等于
>=	大于等于
<=	小于等于
<>	不等于
+	加
–	减
*	乘
/	除

可以在一个 WHERE 子句中同时指定多个条件，这时可以通过一些逻辑运算符把这些条件组织起来。SQL 支持的逻辑运算符如下所示：

- AND：两个条件同时为真时总条件才为真；

- OR：两个条件有一个为真时总条件就为真；
- NOT：条件为假时结果为真。

例如，查询表 TEST 中的 AGE<=20 且 ID>5 的记录，可以通过如下语句来实现。

```
SELECT * FROM TEST WHERE AGE<=20 AND ID>5
```

判断某一列是否为 NULL，可以通过 IS 子句来实现。例如，可以通过如下语句在表 TEST 中查询 INCOMING 列为 NULL 的所有记录。

```
SELECT * FROM TEST WHERE INCOMING IS NULL
```

可以通过如下语句在表 TEST 中查询 INCOMING 列不为 NULL 的所有记录。

```
SELECT * FROM TEST WHERE INCOMING IS NOT NULL
```

使用 LIKE 子句，可以实现字符串的模式匹配。在匹配模式中，字符“%”匹配 0 或者多个字符，“_”恰好匹配一个字符。

例如，下面的语句在表 TEST 中查询 NAME 的最后三个字母为“ONG”的记录。

```
SELECT * FROM TEST WHERE NAME LIKE '%ONG'
```

使用 DISTINCT 可以在一个结果集中取消重复行。关于 DISTINCT 子句的使用可以参考下面的例子。

```
SELECT DISTINCT AGE FROM TEST
```

查询所给出的结果集中的记录，一般是按照其在数据表中给出的先后顺序，通过 ORDER BY 子句可以实现对记录的排序。例如，下面的例子是结果集中的记录按照 NAME 的字母顺序的逆序排列(降序 DESC)。

```
SELECT * FROM TEST ORDER BY NAME DESC
```

有时需要查询某一个字段的值在某个集合内的记录，这时可以通过子句 IN 来实现。例如，下面的语句查询表 TEST 中 AGE 字段的值为 20 或者 21 的记录。

```
SELECT * FROM TEST WHERE AGE IN (20,21)
```

有时需要找出表中的列值在某一个范围内的所有记录，这时可以通过 BETWEEN 子句来实现。例如，执行下面的语句可实现查询表 TEST 中 AGE 字段的值在 20 到 40 之间的所有记录。

```
SELECT * FROM TEST BETWEEN AGE IN (20,40)
```

5. SQL 中的函数

SQL 中还定义了许多函数，这些函数可以分成以下几大类：

- 字符串函数；
- 数字函数；
- 时间和日期函数；
- 系统函数；
- 数据类型转换函数；
- 集函数。

例如，执行下面的语句返回 NAME 字段的长度。

```
SELECT NAME, LENGTH(NAME) FROM TEST
```

其他函数的使用与 LENGTH 函数的使用类似，这里要重点介绍的是应用非常广泛的