全国高职高专规划教材

数据库应用技术实用教程

黄崇本　主　编

马华林　程光华　副主编

科　学　出　版　社

北　京

内 容 简 介

本书共分 3 篇。第一篇为基础知识，内容包括：数据库技术概论、关系数据库、关系数据库语言 SQL 及数据库设计。第二篇为应用篇，内容包括：SQL Server 2000 概述、数据库的建立与维护、数据库的查询、T-SQL 语言、数据安全性与完整性、存储过程与触发器。第三篇为实训篇，内容包括：Access 2000 实训、SQL Server 2000 实训及 VB/SQL Server 2000 编程实训。

本书遵循理论必须够用、强调实践应用、好教好学好用的思路。一方面阐述了数据库的基本理论和方法，另一方面介绍了 SQL Server 2000 的各种功能和应用方法，同时安排了相应的实训及系统开发综合实训。

本书可作为高职高专计算机类学生的教材，也可供从事计算机信息处理工作的科技人员学习参考。

图书在版编目（CIP）数据

数据库应用技术实用教程/黄索本主编. —北京：科学出版社，2003.8
（全国高职高专规划教材）
ISBN 7-03-011985-1
I．数… II．黄… III．数据库系统—教材 IV. TP311.13
中国版本图书馆 CIP 数据核字（2003）第 065818 号

策划编辑：李振格／责任编辑：陈 钢
责任印制：吕春珉／封面设计：东方人华平面设计部

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com
印刷
科学出版社总发行 各地新华书店经销
*
2003 年 9 月第 一 版 开本：787×1092 1/16
2003 年 9 月第一次印刷 印张：17 1/2
印数：1—18 000 字数：394 000
定价：00.00 元
（如有印装质量问题，我社负责调换）

本书编写人员名单

主　编　黄崇本

副主编　马华林　程光华

撰稿人　章俊玲　陆世伟　汤化平

前　言

数据库技术是计算机科学技术中发展最快的应用之一，已经成为计算机信息系统与应用系统的核心技术，它与网络技术构成计算机应用的两个重要平台。数据库应用技术课程已成为高职高专院校计算机教学中的主干课程，是计算机应用专业的必修基础课程。

本着高职高专教学突出理论知识的应用和实践能力的培养，基础理论以必需、够用为度，专业教学加强针对性和实用性等原则，将本书的相关内容分为基础篇、应用篇和实训篇。书中既安排了目前使用非常广泛的桌面数据库管理系统 Access，让学生学会使用 Access 数据库的基本方法，也全面介绍了当前主流大型网络数据库系统 SQL Server 2000，同时还介绍了数据库应用系统开发的基本方法。

本书共 3 篇。第一篇为基础篇，共 4 章：第 1 章主要介绍数据库技术概论，内容包括数据管理技术的发展、数据库基本概念、数据库系统的体系结构、数据库保护、Access 数据库管理系统。第 2 章主要介绍关系数据库，内容包括关系的数学定义、关系数据库、关系运算、Access 数据库的建立。第 3 章主要介绍关系数据库语言 SQL，内容包括数据定义、数据查询、数据更新、数据控制及 Access 数据库操作。第 4 章主要介绍数据库设计，内容包括数据库设计概述、需求分析、概念结构设计、逻辑结构设计、物理设计及实施及 Access 数据库设计。第二篇为应用篇，共 6 章，内容包括 SQL Server 2000 概述、数据库的建立与维护、数据库的查询、T-SQL 语言、数据安全性与完整性、存储过程与触发器。第三篇为实训篇，共 3 章，内容包括 Access 2000 实训、SQL Server 2000 实训及 VB/SQL Server 2000 编程实训。

本书由黄崇本任主编，马华林、程光华任副主编，章俊玲、陆世伟、汤化平等参加编写工作。

由于编者水平有限，书中不足之处在所难免，希望读者批评指正。

编　者

2003 年 6 月

目　录

第一篇　基　础　篇

第二篇　应　用　篇

第一篇

基础篇

数据库技术是计算机科学的重要分支。数据库具有数据结构化、较低的冗余度、较高的程序与数据独立性、易于扩充和易于编制应用程序等优点，较大的信息系统都是建立在数据库设计之上的。数据库技术目前已成为最活跃、应用最广泛的计算机技术之一，几乎所有的信息系统中的数据存储都涉及到数据库。

从 20 世纪 80 年代至今，人们一直在探索新一代数据库系统的理论、技术和方法。本篇介绍了数据库系统的基础知识。包括数据库系统的发展过程、数据库系统的基本概念、数据模型、数据的安全性与完整性、关系数据库、关系数据库语言 SQL 和数据库设计等方面的内容。

第 1 章　数据库技术概论

本章要点

- 数据库系统的基本概念
- 数据库系统的模式和体系结构
- 数据库的安全性

本章难点

- 数据与信息的区别与联系
- Access 数据库管理系统

数据库是长期存储在计算机内有组织、可共享的数据集合。它不仅包括数据本身，而且包括相关数据之间的关系。数据库技术主要研究如何存储、使用和管理数据，即为计算机数据管理技术。在计算机的3大主要应用领域（科学计算、数据处理与实时控制）中，数据处理约占 70%，因此，数据库技术（数据管理技术）已成为计算机领域中最重要的技术之一，是软件学科的一个独立分支。数据库方法原本是针对事务处理中的大量数据管理，但是它的应用范围不断扩大，不仅应用于事务处理，而且应用到情报检索、人工智能、专家系统及计算机辅助设计等，涉及到非数据计算各方面的应用。应用范围的扩大又促进了数据库技术的深入发展。可以说，数据库系统已成为当代计算机系统的重要的组成部分。

1.1　数据与数据管理

1.1.1　数据与信息

人类的一切活动都离不开数据，离不开信息。数据和信息有时可以混用，例如，数据处理也称为信息处理。有时必须分清数据与信息，例如，不能把信息系统称为数据系统。

数据是指用符号记录下来的可以区别的信息，这里的“符号”不仅仅指数字、字母、文字和其他特殊符号，而且还包括图形、图像、声音等多媒体数据。数据的概念包含两个方面的含义：其一，数据内容是事物特性的反映或描述；其二，数据是符号的集合。

信息是关于现实世界中事物的存在方式或运动形态的反映，是人们进行各种活动所需的知识。它以数据的形式表示，即数据是信息的载体，但不是所有的数据都能表示信

息，信息是被人们消化了的数据。另一方面，信息是抽象的，不随数据设备所决定的数据形式而改变，而数据的表示方式具有可选择性。

1.1.2 数据处理与数据管理

数据处理是指将数据转换成信息的过程。广义地讲，它包括对数据的收集、存储、传播、检索、分类、加工或计算、打印各类报表或输出各种图形等一系列活动。狭义地讲，它是指对所输入的数据进行加工整理。在数据处理的一系列活动中，数据收集、存储、传播、检索和分类等操作是基本环节，这些基本环节统称为数据管理。

数据与信息之间的关系可以表示为：

信息 = 数据 + 数据处理

数据是原料，是输入，而信息是产出，是输出结果。当两个或两个以上的数据处理过程前后相继时，前一过程称为预处理。预处理的输出作为二次数据，成为后面处理过程的输入，即数据与信息的概念具有相对性，如图 1.1 所示。

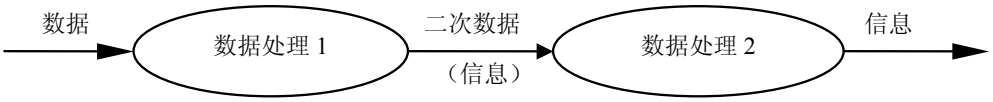

图 1.1 数据与信息的关系

1.2 数据管理技术的发展

计算机数据管理随着计算机硬件（主要是外存储器）技术、软件技术和计算机应用范围的发展而不断发展，大致经历了 3 个阶段：人工管理方式阶段、文件管理方式阶段和数据库管理方式阶段。

1.2.1 人工管理方式

20 世纪 50 年代中期以前，计算机主要用于科学计算。当时在硬件方面，外存储器只有卡片、纸带和磁带，没有像磁盘这样的可以随机访问、直接存取的外部存储设备。软件方面，没有专门管理数据的软件，数据由计算机或处理它的程序自行携带。数据处理方式基本是批处理。数据与应用程序之间的关系，如图 1.2 所示。

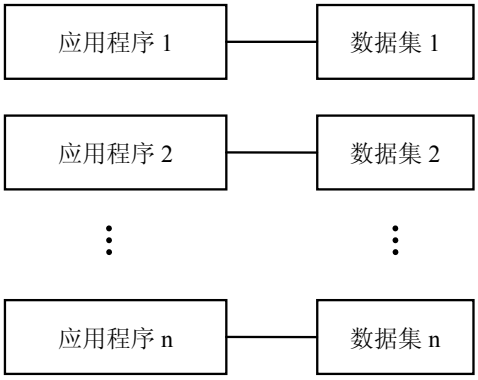

图 1.2 人工管理方式中程序与数据的关系

在这一阶段的特点是：

（1）数据和程序不具有独立性。一组数据对应一组程序，这就使得程序依赖于数据，如果数据的类型、格式或者数据量、存取方法、输入输出方式等改变了，程序必须做相应的修改。

（2）数据不能长期保存。由于数据是面向应用程序的，在一个程序中定义的数据，无法被其他程序利用，因此，程序与程序之间存在大量的重复数据。

（3）系统中没有对数据进行管理的软件。数据管理任务，包括存储结构、存取方法及输入输出方式等完全由程序设计人员自负其责，这就给应用程序设计人员增加了很大的负担。

1.2.2　文件管理方式

20 世纪 50 年代后期至 60 年代中后期，计算机开始大量地用于管理中的数据处理工作。大量的数据存储、检索和维护成为紧迫的需求。在硬件方面，可以直接存取的磁鼓和磁盘成为联机的主要外存。在软件方面，出现了高级语言和操作系统。操作系统中的文件系统是专门管理外存的数据管理软件。数据处理方式有批处理，也有联机实时处理。

在这一阶段，程序与数据有了一定的独立性，程序与数据分开存储，有了程序文件和数据文件的区别。数据文件可以长期保存在外存储器上多次存取，如进行查询、修改、插入及删除等操作。数据的存取以记录为基本单位，并有多种文件组织形式，如顺序文件、索引文件和随机文件等。

在文件系统的支持下，数据的逻辑结构与物理结构之间有一定的区别，逻辑结构与物理结构之间的转换由文件系统的存取方法来实现。数据与程序之间有设备独立性，程序只需用文件名访问数据，不必关心数据的物理位置。这样，程序员可以集中精力在数据处理的算法上，而不必考虑数据存储的具体细节。

数据的逻辑结构是指呈现在用户面前的数据结构。数据的物理结构是指数据在物理设备上的实际存储结构。例如，用户看到的记录是按照记录号顺序排列的，而实际上这些记录可能是分散存储在磁盘的不同扇区，用链接方式组织在一起的。用户不必关心记录在存储器上的地址和内、外存交换数据的过程。

该阶段数据与程序的关系如图 1.3 所示。

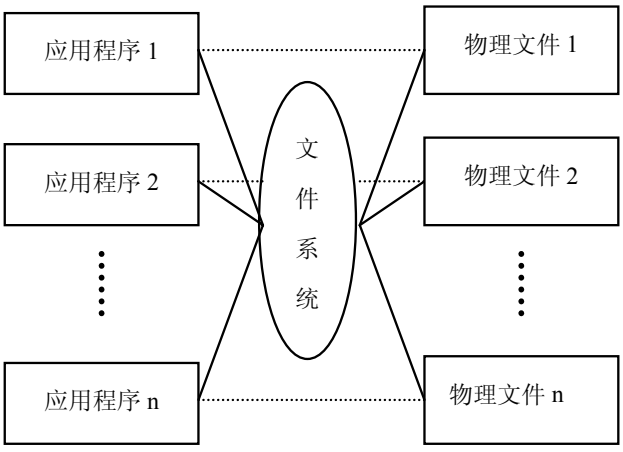

图 1.3　文件管理方式中程序与数据的关系

文件系统阶段对数据的管理虽然有了长足的进步，但一些根本性的问题仍然没有彻底解决，主要表现在以下 3 个方面：

（1）数据冗余大。数据冗余是指不必要的重复存储，同一数据项重复出现在多个文件中。在文件系统中，数据文件基本上与各自的应用程序相对应，数据不能与记录、数据项共享。这样不仅浪费存储空间，甚至会造成数据库的不一致性。

（2）缺乏数据独立性。文件系统中的数据文件是为某一特定的应用而设计的，数据与程序相互依赖，如果改变数据的逻辑结构或文件的组织方法，必须修改相应的应用程序，反之亦然。

（3）数据没有集中管理。数据文件均由相应的应用程序管理和维护，没有统一的管理机制。造成数据文件之间无法进行联系，不能反映现实世界事物之间的联系。

1.2.3 数据库管理方式

从 20 世纪 60 年代后期开始，计算机应用于管理的规模更加庞大，需要计算机管理的数据量急剧增长，并且对数据共享的需求日益增强。文件系统的数据管理方法已无法适应应用系统的需要。再加上大容量磁盘的出现，使计算机随机存取数据成为可能。为了解决数据的独立性问题，实现数据的统一管理，达到数据共享的目的，出现了数据库技术。

数据库是通用化的相关数据集合。它不仅包括数据本身，而且包括关于数据之间的联系。数据库中的数据不是面向某一特定的应用，而是面向多种应用，可以被多个用户、多个应用程序共享。其数据结构独立于使用数据的程序，对于数据的增加、删除、修改及检索等均由系统统一进行控制。

为数据库的建立、使用和维护而配置的软件称为数据库管理系统（DBMS）。它是在操作系统支持下运行的。目前较为流行的数据库管理系统包括：Oracle、Informix、SQL Server 及 PC 上的 DBMS（DBASE、FoxBASE、FoxPro、Access 等）。

现在，数据库已成为各类信息系统的核心基础。在数据库管理系统支持下的数据与程序关系，如图 1.4 所示。

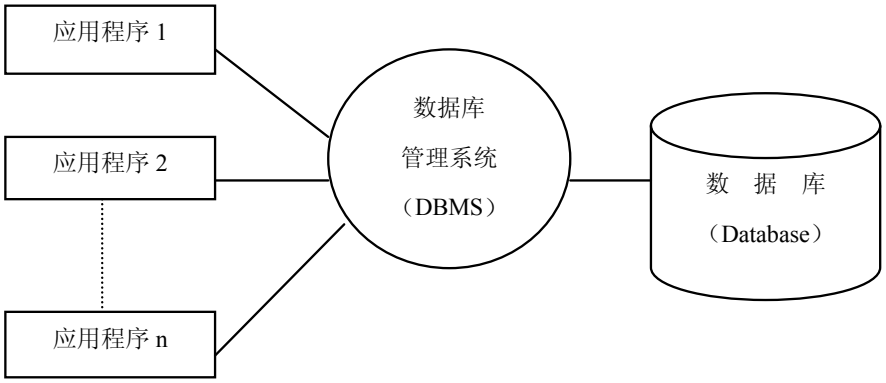

图 1.4　数据库系统数据与程序的关系

数据库系统的主要特点有：

（1）实现数据共享，减少数据冗余。数据库中存放通用化的综合数据，某一应用

通常仅使用总体数据的子集。

（2）数据高度结构化。数据库中的数据是有结构的，它是用某种数据模型表示出来的，这种结构既反映文件内数据之间的联系，也反映文件之间的联系。

（3）具有较高的数据独立性。在数据库系统中，DBMS 提供映像功能，确保应用程序对数据结构和存取方法有较高的独立性。数据的物理存储结构与用户看到的逻辑结构可以有很大的差别。用户只是用简单的逻辑结构来操作数据，无需考虑数据在存储器上的物理位置与结构。

（4）有统一的数据控制功能。数据库作为用户与应用程序的共享资源，对数据的存取往往是并发的，即多个用户同时使用同一个数据库。数据库管理系统必须提供并发控制功能、数据的安全性控制功能和数据完整性控制功能。

20 世纪 70 年代后期之前，数据库系统多数是集中式的，到了 80 年代中期，出现了分布式数据库系统。分布式数据库系统是数据库技术与计算机网络技术相结合的产物。分布式数据库是一个逻辑上统一、地域上分布的数据集合，是计算机网络环境中各个结点局部数据库的逻辑集合，同时受分布式数据库管理系统的控制和管理，如图 1.5 所示。

图 1.5　分布式数据库系统

分布式数据在逻辑上像一个集中式数据库系统，实际上数据存储在处于不同地点的计算机网络的各个结点上。每个结点有自己的局部数据库管理系统，它有很高的独立性。用户可以由分布式数据库管理系统(网络数据库管理系统)，通过网络通信相互传输数据。分布式数据库系统有高度透明性，每台计算机上的用户并不需要了解他所访问的数据究竟在什么地方，就像在使用集中式数据库一样。其主要优点有：

（1）局部自主。网络上每一个结点数据库系统都具有独立处理本地事务的能力，而且各局部结点之间也能够互相访问，有效地配合完成更复杂的事务。

（2）可靠性和可用性高。个别结点出问题，整个系统仍可用，比集中式系统可靠。

（3）效率和灵活性高。分布式系统分散了工作负荷，易于扩充。

1.3　数据库基本概念

1.3.1　数据库系统的组成

数据库是长期存储在计算机内、有组织的、可共享的数据集合。数据库本身不是独立存在的，它是组成数据库系统的一部分，在实际应用中，人们面对的是数据库系统（DBS），也称数据库应用系统（DBAS）。数据库系统是指具有管理和控制数据库功能

的计算机应用系统。例如，一个以数据库为基础的管理信息系统。数据库系统由 5 部分组成：硬件系统、数据库集合、数据库管理系统及相关软件、数据库管理员（DBA）和用户。

硬件系统是整个数据库系统的基础，需要有足够大的内存、足够大容量的磁盘等联机直接存取设备等。数据库集合是若干个设计合理、满足应用需求的数据库。数据管理系统是为数据库的建立、使用和维护而配置的软件，是数据库系统的核心组成部分。相关软件是支持软件，如操作系统等。数据库管理员是全面负责建立、维护和管理数据库系统的人员。用户是最终系统的使用和操作人员。

1.3.2 数据库管理系统

数据库管理系统作为数据库系统的核心软件，其主要目标是使数据成为方便用户使用的资源，易于为各种用户所共享，并增加数据的安全性、完整性和可用性。

在数据库系统中，数据是多个用户和应用程序的共享资源，已经从应用程序中完全独立出来，由 DBMS 来统一管理。DBMS 具有下列功能。

1. 数据库的定义功能

提供数据定义语言 DDL 或操作命令，用来对各级数据模式进行精确的描述。由此，系统必须包含 DDL 的编译或解释程序。这些数据模式并不是数据本身，而是具体 DBMS 所支持的数据模型结构。用 DDL 所做的定义将被系统保留在数据字典中，以便在进行数据操纵和控制时使用。用户可以查阅数据定义，以便共享数据库中的数据。

2. 数据操纵功能

为了对数据库中的数据进行追加、插入、修改、删除和检索等操作，DBMS 提供语句或命令，称为数据操作语言 DML。不同的 DBMS 语言的语法格式也不同，以其实现方法而言，可以分为两类：一类是 DML 可以独立交互式使用，不依赖于任何程序设计语言，称为自含或自主型语言。另一类是宿主型 DML，嵌入到宿主语言中使用。如嵌入到 C 程序设计语言中。在使用高级语言编写的应用程序中，需要调用数据库中的数据时，则要用宿主型 DML 语句来操纵数据。

3. 数据库运行控制功能

数据库中的数据是提供给多个用户共享的，用户对数据的存取可能是并发的，即多个用户同时使用一个数据库。DBMS 必须提供以下 3 方面的数据控制功能：并发控制功能。对多个用户并发操作加以控制和协调，避免产生一个用户要写某数据时，另一个用户要读该数据而产生的错误，或两个用户同时要对某数据进行写操作而出现错误等；数据的安全性控制。数据安全性控制是对数据库采用的一种保护措施，防止非授权用户存取造成数据泄密或破坏。例如，设置口令、确定用户访问权限等；数据的完整性控制。数据完整性是指数据的准确性和一致性。系统应采取一定的措施确保数据有效，与数据库的定义一致。

4. 数据字典

数据字典中存放数据库系统中有关数据的数据，称之为元数据。它提供对数据库数据描述的集中管理手段，对数据库的使用和操作都通过查阅数据字典来进行。内容包括对各级模式的描述，数据库的使用人员等信息。数据字典可以看作是数据库系统自身的小数据库，它既可以供数据库系统人员使用，又可以提供给一般用户使用。数据字典有以下两方面的作用：

（1）有利于数据库管理员掌握整个系统结构和系统运行情况。DBA 可以通过查阅数据字典来监视系统，如已经建立多少个数据库，各个数据库的结构如何，多少个用户在使用系统等，并协助用户完成其应用。

（2）方便用户使用系统。用户可以从数据字典中查对某个属性出现在哪个关系中等信息。

1.3.3　数据模型

数据库中的数据是有结构的，这种结构反映出事物和事物之间的联系。数据模型就是指对数据以及数据之间的联系的描述。任何一个数据库管理系统都是基于某种数据模型的，它不仅管理数据的值，而且要按照模型管理数据间的联系。一个具体数据模型应当反映所有数据之间的整体逻辑关系。

数据模型由 3 部分组成，即模型结构、数据操作和完整性规则。其中模型结构是数据模型最基本的部分，它将确定数据库的逻辑结构，是对系统静态特性的描述。数据操作提供了对数据库的操纵手段，主要有检索和更新两大类操作，它是对系统动态特性的描述。完整性规则是对数据库有效状态的约束。

数据库管理系统所支持的数据模型分为 4 种：层次模型、网状模型、关系模型及面向对象模型。层次模型与网状模型又统称为格式化模型，关系模型是当今最流行的数据库模型，面向对象模型则是面向对象技术与数据库技术相结合的产物，是目前研究与开发的一个热门方向。不同数据模型之间的根本区别在于数据之间联系的表示方式不同：关系模型是用“二维表”（或称为关系）来表示数据之间的联系；层次模型是用“树结构”来表示数据之间的联系；网状模型是用“图结构”来表示数据之间的联系。面向对象模型是用“对象、类及类层次”来表示数据、操作及相互关系。

1.4　数据库系统的体系结构

1.4.1　数据模式的概念

模式是数据库中全体数据的逻辑结构和特征的描述，它仅仅涉及模式的描述，不涉及具体的值。模式的一个具体值称为模式的一个实例（instance）。同一模式可以有很多实例。模式是相对稳定的，而实例是相对变动的，因为数据库中的数据是在不断更新的。模式反映数据的结构及其联系，而实例反映数据库某一时刻的状态。

1.4.2 数据库系统的三级体系结构

虽然实际的数据库管理系统产品种类很多，它们支持不同的数据模型，使用不同的数据库语言，建立在不同的操作系统之上，数据的存储结构也各不相同，但它们在体系结构上通常都具有相同的特征，即采用三级模式结构，并提供两级映像功能。

在数据库系统中，用户看到的数据与计算机中存储的数据是两回事，两者之间是有联系的，实际上它们之间已经过两次变换，即为两级映像。一次是系统为了减少冗余，实现数据共享，把所有用户的数据进行综合，抽象成一个统一的数据视图；第二次是为了提高存取效率，改善性能，把全局视图的数据按照物理组织的最优形式存放。

数据库系统的体系结构分成 3 级：内部级、概念级和外部级。这个三级结构也称数据库系统的“三级模式结构”。内部级又称“内模式”，概念级又称“模式”，外部级又称“外模式”，故数据库系统是由外模式、模式和内模式三级构成，如图 1.6 所示。

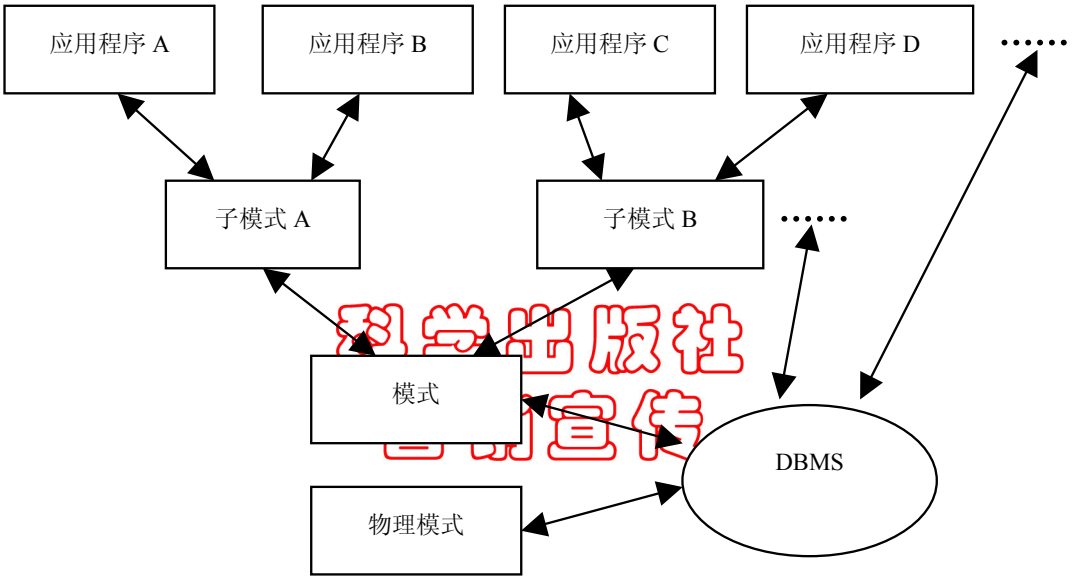

图 1.6 模式之间的关系

1. 外模式

用户使用的数据视图叫做外模式，外模式又称子模式或用户模式，它是用户与数据库系统的接口，是用户用到的那部分数据的逻辑结构和特征的描述，是数据库用户的数据视图，是一种局部的逻辑数据视图，表示用户所理解的实体、实体属性和实体关系。用户使用数据操纵语言（DML）对数据库进行操作，实际上是对外模式的外部记录进行操作。

外模式通常是模式的子集，一个数据库可以有多个外模式。由于它是各个用户的数据视图，如果不同的用户在应用需求、组织数据的方式及对数据保密的要求等方面存在差异，则其外模式描述就是不同的。另一方面，一个外模式也可以为某一用户的多个应用系统所使用，但一个应用程序只能使用一个外模式。

DBMS 提供子模式描述语言（子模式 DDL）来严格定义子模式。

2. 模式

模式又称逻辑模式或称概念模式，是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。它是数据库系统模式结构的中间层。它是全局的逻辑数据视图，是数据库管理员所看到的实体、实体属性和实体之间的关系。

模式实际上是数据库数据在逻辑级上的视图。一个数据库只有一个模式。在定义数据库各层次结构时，首先应定义模式。定义模式时不仅要定义数据的逻辑结构，例如数据记录由哪些数据项构成，数据项的名字、类型、取值范围等，而且要定义数据之间的联系以及定义与数据有关的安全性和完整性要求。

DBMS 提供模式描述语言（模式 DDL）来严格地定义模式。

3. 内模式

内模式又称存储模式，也称物理模式。它是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。一个数据库只有一个内模式。

内模式依赖于全局逻辑结构，其目的是将模式中定义的数据结构进行适当的组织和空间分配，以实现较好的时空运行效率。

DBMS 提供内模式描述语言（内模式 DDL 或者存储模式 DDL）来严格定义内模式。

1.4.3　数据库的两级映像与数据的独立性

DBMS 在三级模式之间提供了两级映像：外模式 / 模式映像及模式 / 内模式映像，这两层映像保证了数据库系统中的数据能够具有较高的逻辑独立性和物理独立性。

1. 外模式 / 模式映像

模式描述的是数据的全局逻辑结构，外模式描述的是数据的局部逻辑结构。对应于同一个模式可以有任意多个外模式。对于每一个外模式，数据库系统都有一个外模式 / 模式映像，它定义了该外模式与模式之间的对应关系。

如果模式改变（即整体逻辑结构要作修改），那么对各个外模式 / 模式的映像作相应改变，可以使外模式保持不变，可不必修改应用程序，这就保证了数据与程序的逻辑独立性，简称数据的逻辑独立性。外模式 / 模式映像放在外模式中描述。

2. 模式 / 内模式映像

模式 / 内模式映像是惟一的，它定义了数据库全局逻辑结构与存储结构之间的对应关系。如果数据库的存储结构（即内模式）要作修改，那么模式 / 内模式映像要作相应的修改，才可以使模式保持不变，也不必改变应用程序，这就保证了数据与程序的物理独立性，简称数据的物理独立性。

模式 / 内模式映像一般放在内模式中描述。

1.5　数据库保护

数据库保护是现代数据管理的重要课题，包括安全性保护、完整性保护、故障恢复

和并发控制。DBMS 中有专门程序用于此目的，不同 DBMS 控制能力有强有弱，采取的对策五花八门，这部分是系统中最为灵活也是较为复杂的部分。

1.5.1 安全性保护

数据库规模有大有小，但终归是综合了不同用户的数据，数据共享是数据库的一个重要特点，但是，这些数据不是什么人都可无限制使用的，军事部门有个保密问题，银行存款不能非法改动，等等，这意味着必须规定用户访问数据库的权限，所谓安全性保护，就是防止未被授权者非法存取数据。

许多系统采取各种措施层层设防，一般有以下几种：

（1）鉴定用户身份。用户在定义子模式时，DBMS 随之给他规定了一个用户身份。今后使用时系统首先鉴定其身份，符合身份者才有权使用机器，在一般多用户或微机局域网分布式数据库系统中常常采用这种办法。

（2）口令。口令就是暗号。采用暗号联络，最好设置一个机关，如事先约定个算法，系统与用户同时按规定算法作相同变换，两相对照，这比规定一个死的暗号要机密，当然若是时间函数就更好，因为暗号随时间而变，不知底细就很难识破。

（3）用密码记载数据。

（4）控制用户权限，可以分成若干等级，DBA 等级较高，比如他可读、可写、可建、可改，而其余用户或是只能读不能写，或根本无权操作数据库。

1.5.2 完整性保护

完整性保护是指数据正确性和一致性，DBMS 有完整性检查程序，在建立数据库时，把完整性作为模式的组成部分存入数据库。完整性保护可以通过对数据及数据之间的逻辑关系施加约束条件来实现。DBMS 按用户在模式中规定的约束条件对数据进行检查，以保证数据的正确性和一致性。约束条件可分为值的约束和结构的约束。

1. 值的约束和结构的约束

值的约束是指对数据的类型、范围和精度方面的限制，防止一些离奇古怪的数据装入数据库。例如，月份必须为 1～12 之间的整数，工资则要求准确到小数点后两位数。如果出现一年有 15 个月，一天工作了 28 小时等逻辑荒谬数据，DBMS 就拒绝接受。

结构的约束是指对数据之间联系方面的限制。例如，一个储户惟一对应一个账号，它能惟一决定了该储户的其他方面的数据。根据这个特点，账号取值不能为“空”值，并且它必须在数据库中是“惟一”的，以保证引用储户数据时，能根据账号取值“惟一”条件找到某储户的相关数据，如果违反了这种限制，就破坏了结构的规定。

以上所讲的属于静态约束，即它们都是在稳定状态下必须满足的约束条件。此外，还可以通过动态限制条件保证数据完整性。

2. 动态约束

动态约束指当数据从一种状态转变为另一种状态时，对新旧值间所规定的限制条件。例如，修改年龄时，新年龄不应小于旧年龄，以此保证数据的正确性。当然不是所

有的 DBMS 都能做到这一点。

1.5.3　其他数据的保护措施

1. 故障恢复

确保数据库免遭破坏是头等大事，是数据库运行的最基本要求，然而实际情况往往很难避免，如磁盘损坏、电脑病毒、操作失误等偶然因素使数据损坏或丢失。作为 DBMS 必须考虑这个问题，否则没有人敢用数据库。

目前有多种办法解决这个问题，最常用可靠的办法是随时或定期转存数据库（备份），时刻有一份或几份副本在手中。备份不仅要转存数据，还要转存记载数据库运行的动态信息，后者记录数据库在运行过程中，用户用了什么命令、做了什么操作造成了数据库的破坏，只有这样才能跟踪动态进程逐步回退到破坏前的状态，这叫故障恢复。

2. 并发控制

数据库多用户共享，当多个用户同时操作同一数据时，有可能造成数据不一致，因此并发操作时必须加以控制，这个概念很重要，特别是在多用户系统或网络数据库中，应用程序一定要考虑并发控制问题。

并发操作，两道处理的不合理的时差会造成数据修改的丢失。对此，我们可仿照操作系统中介绍的控制进程的同步思想，即当一个进程访问数据库时，就用一把锁，锁住别的进程的执行，直到本道作业完成并把数据写回数据库后再解锁，才允许其他作业的进行，这样就可以控制相关进程互斥地访问数据库，从而保证数据的完整性。

3. 死锁问题

在用户程序中使用锁，可能会出现一个死锁的问题，在操作系统中已经很好地解决了死锁这个问题，这里有必要重述一下死锁概念，因为在编应用程序中要考虑这个问题。

（1）死锁：设 S 是两个以上的处理程序组成的集合，若 S 中的每一个处理程序都在等待另一个当前已被 S 中的其他处理程序锁住的数据资源，会导致谁也进行不下去的情况，或者说是多个处理进程相互占用所需资源，导致相互等待，谁都不能运行下去的情况。这就产生了死锁问题。

（2）死锁的预防：死锁与资源分配和并发进程的速度有关。在数据库系统中，单用户方式不会发生死锁，因为单用户方式不存在资源共享问题，但是多用户方式却可能出现竞争共享资源而产生死锁现象，不过现代数据库管理系统中已加入了死锁检测，当发生死锁时 DBMS 会自动解锁。但编写应用程序时，采取某种预防死锁措施有利于程序的运行，下面介绍两种预防死锁的办法。

① 静态分配资源法。指多个处理程序的任意一个，开始处理之前，先申请所需的全部数据资源，仅当得到了所需的全部资源后才开始执行，这种方法之所以能预防死锁，是因为处理程序在执行过程中不必再申请资源，自然也就不会发生等待资源情况，而该处理程序的执行总是会结束的，一旦它结束后就使其他等待的处理程序得以执行，从而防止了死锁。静态分配方法简单，但资源利用率低，因为这种分配策略把资源预先全部

分配给某个处理程序，而这个处理程序却可能在占有资源情况下较长时间并不使用它，使其他急需使用该资源的处理程序不得不被迫处于等待状态。

② 按序分配资源法。指把系统中的全部数据资源编排一个顺序，并约定每个处理程序必须按这个顺序申请资源，这可以避免多个处理程序互相等待被另一程序占用的资源，打破循环等待资源的程序链，达到预防死锁的目的。按序分配资源，允许一个程序不必一开始就占用全部资源，仅在需要时才去申请，使其他处理程序有可能在它不用时得以利用，显然提高了资源利用率。这种分配策略，要求对资源的顺序做出适当安排，但因不同用户有不同的要求，一成不变的顺序很难兼顾所有的应用程序，这往往又造成另一些处理程序速度降低。

1.6 Access 数据库管理系统

Access 2000 是 Office 办公套件中一个极为重要的组成部分。目前是世界上最流行的桌面关系型数据库管理系统。不管是处理公司的客户订单数据、管理自己的个人通讯录，还是大量科研数据的记录和处理，人们都可以利用它来解决大量数据的管理工作。

Access 2000 与许多常用的数据库管理系统，如 FoxPro、Oracle、SQL Server 等一样，是一种关系数据库管理系统。它可以管理从简单的文本、数字字符到复杂的图片、动画或声音等各种类型的数据。在 Access 2000 中，可以构造应用程序来存储和归档数据，并可以使用多种方式进行数据的筛选、分类和查询，还可以通过显示在屏幕上的窗体来查询数据或生成报表将数据按一定的格式打印出来。

1.6.1 Access 的组成与特点

1. Access 2000 的组成

（1）数据库引擎：它是真正存储、排序和获取数据的软件，一般来说，数据库引擎是不可见的。

（2）数据库对象：Access 2000 是一种面向对象的开发环境，它的数据库窗口非常方便于用户访问各种对象。所谓对象就是提供一种特定的使用界面，用于查询、输入和提取数据库的有关信息。Access 2000 数据库最基本的构件是对象。一个数据库可以包含任意数量的对象，可以使用数据库窗口对各种对象进行处理，最常见的对象有表、窗体、查询及报表等。

（3）设计工具：Access 2000 包含一套设计工具，可用于创建对象。例如，利用查询设计器可以设计一个查询。

（4）编程工具：VBA。

2. Access 2000 的特点

（1）存储文件单一。一个 Access 2000 数据文件包含了该数据库中的全部数据表、查询以及其他与之相关的内容。文件单一便于文件的存储管理，也使得用户操纵数据库及编写应用程序更为方便。

（2）兼容多种数据格式。Access 2000 提供了与其他数据库管理软件的良好接口，能识别其他数据库管理系统生成的数据库文件，能直接导入如 Excel、Word 等编辑形成的数据表、文本文件和图形等多种内容，而且自身的数据库内容也可以方便地在这些软件中操作。

（3）具有 Web 页发布功能。Access 2000 增加了数据页功能，通过创建数据访问页，可以将数据库管理系统移植到浏览器中，从而实现将数据发布到 Internet 上，以及在 Internet 上管理和操作数据库的功能。

（4）操作使用方便。

1.6.2　Access 数据库的内部结构

Access 2000 所使用的对象包括表、查询、报表、窗体、模块和数据访问页。在一个数据库中，除数据访问页之外，其他的对象都存放在一个扩展名为 .mdb 的数据库文件中，而不像其他数据库那样分别存放在不同的文件中，这样就方便数据库文件的管理。

Access 2000 中各个对象之间有着密切的关系。其中，表是数据库的核心与基础，它存放着数据库中的全部数据信息。报表、查询和窗体都是从数据表中获取数据信息的，以实现用户某一特定的需求，例如查询、统计计算、打印、编辑和修改等。窗体可以提供一种良好的用户操作界面，通过它可以直接或间接调用宏或模块，并执行查询、修改、打印、预览和计算等操作。

1.6.3　Access 2000 的功能

1. 组织数据

DBMS 最主要的作用就是组织、管理各种各样的数据。Access 2000 的表对象可用于组织数据的基本模块。组织数据就是按预先的设计建立各个表的结构，把各种类型的数据分别存放在不同的表中，并建立各表之间的联系，从而把相关数据组织在一起。

2. 建立查询

查询是操纵数据库的主要目的之一。查询对象用于建立查询的基本模块，通过创建查询来查找指定条件的数据，更新或删除记录，或对数据执行各种计算。

3. 设计窗体

窗体是用户和数据库应用程序之间的接口之一，在数据库系统中应用窗体可提供数据操作的安全性，并可以丰富用户操作界面。

4. 输出报表

Access 2000 中的报表对象是用于生成报表和打印报表的基本模块。报表可以用来分析数据或以特定方式打印数据。

5. 建立数据共享机制

Access 2000 提供了与其他应用程序的接口，即数据的导入和导出。通过这些功能，可以将其他系统的数据导入到 Access 2000 的数据库中，也可将 Access 2000 的数据导出到其他系统中。

6. 建立超链接

将一个字段的数据类型定义成超链接，并将 Internet 或局域网中的某个对象赋予这个超链接后，当用户在数据表或窗体中双击该超链接字段时，就可以启动浏览器，并进入该超链接所指的对象。

7. 建立应用系统

Access 2000 提供了宏和 VBA，可将各种数据库及其对象连接在一起，从而形成一个数据库应用系统。还提供了“切换面板管理器”，可以将已经建立的各种数据库对象连接在一起，形成所需要的应用系统。

1.6.4 Access 2000 的开发环境

作为 Microsoft Office 2000 的套件的成员，Access 2000 的使用界面与 Word、Excel 等的风格相同。在 Access 2000 中编辑数据库对象就像在 Word 里编辑文档、在 Excel 里编辑数据表一样方便。当然，由于各自设计目标有所侧重，其功能、界面和使用方法也会有所差别。

1. Access 2000 的主窗口

Access 2000 的主窗口如图 1.7 所示，其中包括标题栏、菜单栏、工具栏、状态栏以及编辑区等。

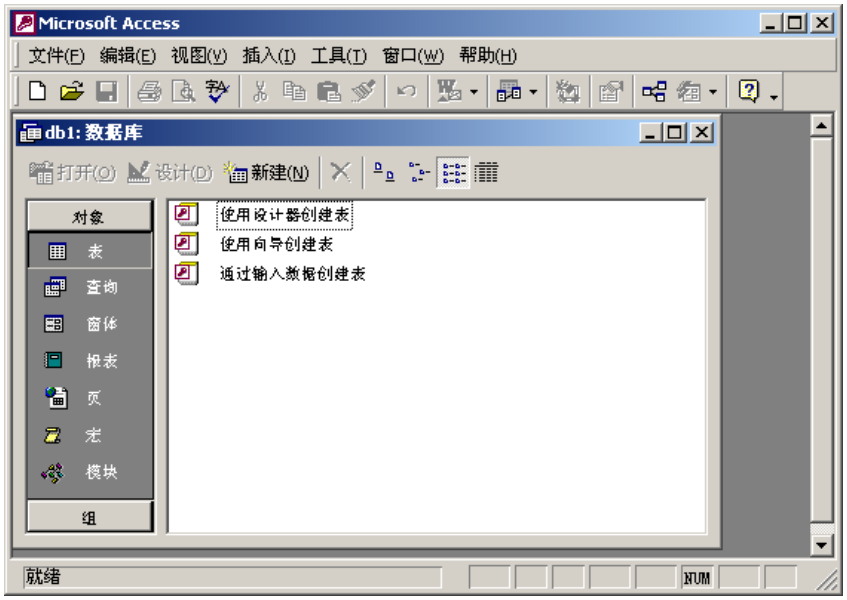

图 1.7 Access 2000 的主窗口与数据库窗口

2. 数据库窗口

打开一个数据库后，就会出现一个数据库窗口。如打开一个 db1 数据库，就出现如图 1.7 所示的“db1：数据库”窗口。所有的数据库操作都是围绕着数据库窗口进行的。数据库窗口由工具栏、对象栏和对象列表框 3 部分组成。

小　结

本章介绍了数据与信息的区别与联系，数据处理的基本概念。计算机数据管理的 3 种方式：人工管理方式、文件系统管理方式及数据库管理方式。数据库系统的基本概念：数据库、数据库管理系统、数据库系统及数据库系统的组成。数据模型（数据结构、数据操纵、数据完整性）：层次模型、网状模型及关系模型。数据模式：包括子模式、概念模式及物理模式。数据库系统的三级模式结构和两级映像的体系结构，它保证了数据库系统中能够具有较高的逻辑独立性和物理独立性。数据库保护包括数据安全性、完整性、并发控制和故障恢复等多方面。最后介绍了 Access 数据库管理系统。

习　题

1．试回答下列问题：

（1）什么是数据库？

（2）什么是数据库管理系统？

（3）什么是数据库系统？

2．计算机数据管理经历了哪几个阶段？

3．试述文件系统的缺点。

4．试述数据库系统的特点。

5．试述概念模型的作用。

6．解释下列概念：

内模式　模式　外模式　DDL　DML

7．试述数据库系统三级模式结构，这种结构的优点是什么？

8．试述数据库的物理独立性和逻辑独立性。

9．DBMS 由哪几部分组成？DBA 的职责是什么？

10．什么叫数据安全性及完整性？

11．何谓宿主语言，自含语言？

第 2 章　关系数据库

本章要点

- 关系的数学定义
- 关系数据库
- 关系数据运算
- Access 数据库的建立

本章难点

- 关系数据库的基本概念
- 关系数据运算

关系数据库用数学方法来处理数据库中的数据。关系模型是 1970 年由 E.F.Codd 提出的，30 年来，关系数据库系统的研究取得了辉煌的成就，使数据库系统的应用领域迅速扩大。

2.1　关系的数学定义

2.1.1　二维表与关系

如图 2.1 所示，这张二维表（或简称表）表示了学生类型的一些实体，每个二维表又称为关系。“学生信息表”是表名，在关系数据库中，表名对应于关系名。表头由一些反映实体属性的属性名组成，每个属性名各对应一列，它对应于数据库结构描述，定义

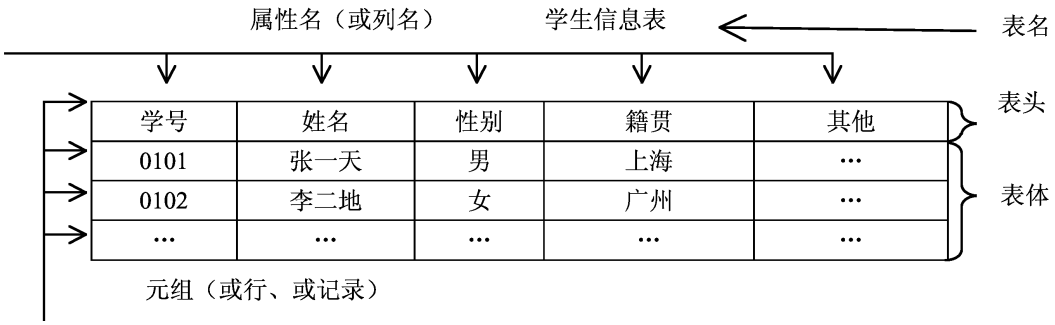

图 2.1　二维表及相关术语

了实体的型，也就是说规定了实体的值域。在表上属性名必须是惟一，不允许重名。表体是由一些行（元组或称记录）组成，它是数据库的内容及数据库操作对象。一个元组对应于文件系统中的一个记录，一个记录含有若干个域用以存储属性值。一个元组对应于一个学生实体。表体中的列反映实体的属性。

2.1.2　关系的定义

定义 2.1　域（Domain）是值（Value）的集合。

例如，整数、实数、{0，1，2，3}、{真、假}等等都可以是域。对于是什么值没有限定，值可以有结构，如{（杭州，0571）、（宁波，0574）}是一个域，其值为（杭州，0571）和（宁波，0574），其值是有结构的。

若域是一个有限集，其元数的个数叫做域的基数。例如域{1，2，3，4，5}的基数为5。

定义 2.2　给定一组域 D_1，D_2，…，D_n，这些域中有些域可以是相同的。D_1，D_2，…，D_n 的笛卡儿乘积为 $D_1\times D_2\times\cdots\times D_n=\{(d_1, d_2, \cdots, d_n) \mid d_i\in D_i, i=1, 2, \cdots, n\}$，其中每一个元素（$d_1$，$d_2$，…，$d_n$）叫做一个 n 元组，或简称元组，元组（$d_1$，$d_2$，…，$d_n$）中的值 d_i 叫做该元组的第 i 个分量。

若 D_i 为有限集且 D_i 的基数为 m_i（i=1，2，…，n），则 $D_1\times D_2\times\cdots\times D_n$ 的基数为 $m=m_1\times m_2\times\cdots\times m_n$。

定义 2.3　$D_1\times D_2\times\cdots\times D_n$ 的子集叫做域 D_1，D_2，…，D_n 上的关系，用 R（D_1，D_2，…，D_n）表示，这里的 R 表示关系的名字，n 是关系的度（或叫元数）。

一个 n 元关系 R（D_1，D_2，…，D_n）很容易表示成一个二维表。表名为 R，属性 D_1，D_2，…，D_n 作为表头，R 中的元数，即 $D_1\times D_2\times\cdots\times D_n$ 的子集作为表体。例如，D_1={A，B，C}，D_2={1，2}，D_3={True，False}，R（D_1，D_2，D_3）={（A，1，True），（B，2，False），（C，2，True）}可表示为表格，如图 2.2 所示。

根据关系的数学定义，关系是笛卡尔乘积的子集，但在关系数据库中，关系一般是指笛卡尔乘积的有意义子集。例如，父亲={王刚，李铁}，母亲={黄花，吕柳}，子女={王小刚，王大刚，李小英}，则家庭关系（父亲，母亲，子女）={（王刚，黄花，王大刚），（王刚，黄花，王小刚），（李铁，吕柳，李小英）}可表示为表格，如图 2.3 所示。

R

D_1	D_2	D_3
A	1	True
B	2	False
C	2	True

图 2.2　关系的表格表示

家庭关系

父亲	母亲	子女
王刚	黄花	王小刚
王刚	黄花	王大刚
李铁	吕柳	李小英

图 2.3　家庭关系的表格表示

2.1.3　关系的性质

在关系数据库中，要求关系中的每一个分量是不可再分的数据项，如图 2.4 所示的家庭情况表就不能成为关系数据库中的关系。关系数据库中的关系应具有以下性质：

家庭情况表

父亲	母亲	子　女	
		长子女	次子女
王刚	黄花	王大刚	王小刚
李铁	吕柳	李小英	

图 2.4　不是关系的表格

（1）列是同质的，即每一列中的分量均是同类型的数据，即均来自同一个域。

（2）不同的列可以出自同一个域，每一列称为一个属性，要给予不同的列不同的属性名。

（3）列的顺序是无所谓的，即列的次序可以变换。

（4）任意两个元组不能完全相同。

（5）行的顺序是无所谓的，即行的次序可以变换。

（6）每一分量必须是不可再分的数据项。

2.2　关系数据库

2.2.1　关系模型

关系模型由 3 部分组成：数据结构（即关系）、关系操作、关系完整性。下面将对这 3 部分进行讨论。

1. 单一的数据结构——关系

在关系模型中，无论是实体还是实体之间的联系均由单一的类型结构——关系来表示。在前面，已对关系进行了介绍，下面介绍关键字、关系模式和关系数据库等基本概念。

（1）关键字。关系中的某一属性（或属性组），若其值可以惟一标识一个元组，则称该属性为一个候选关键字，若一个关系中有多个候选关键字，则可以任选一个作为主关键字。主关键字中的属性称为主属性。

（2）关系模式。关系模式是指关系的描述，即对关系名、组成关系的各属性名、属性到域的映射、属性间的数据依赖关系等等。关系模式通常简记为 R（A_1，A_2，…，A_n），其中 R 是关系名，A_1，A_2，…，A_n 为属性名。属性到域的映射一般通过指定的类型和长度来说明。

（3）关系数据库。关系数据库是指数据库结构的描述，它包括关系数据库名，若干属性的定义以及这些属性上的若干关系模式。

2. 关系操作

关系模型规定了关系操作的功能和特点，但不对 DBMS 语言的语法做具体规定。关系数据库语言的主要优点是其高度的非过程化，用户只需知道语句能做什么，而不必知

道怎么做。用户一般也不必求助于循环、分支等程序设计语句来完成数据操作。关系操作主要有：并、交、差、选择、投影和连接等，其中选择、投影及连接是最基本的关系操作。这些操作均对关系的内容或表体实施的，得到的结果仍为关系。

关系操作的特点是集合操作，即操作对象和结果都是集合。关系操作可以分为关系代数与关系演算两大类，关系演算又可以分为元组演算和域关系演算。

3. 关系模型的完整性

关系模型的 3 类完整性：实体完整性、参照完整性及用户定义的完整性。其中，实体完整性和参照完整性是任何关系模型都有必须满足的完整性约束条件，应该由关系数据库 DBMS 自动支持。而用户定义的完整性的支持是由 DBMS 提供完整性定义设施，可以随 DBMS 商品软件不同而有所变化。

（1）实体完整性是指若属性 A 是基本关系 R 的主属性中的属性，则属性 A 不能取空值。基本关系是指实际存在的表，它是实际存储数据的逻辑表示。而查询表与视图表则是导出表，是虚表。一个基本关系通常是对应于现实世界中的一个实体集，而实体是可以区分的，如果主属性可以取空，则实体就无法进行区分，所以主属性不能为空。

（2）参照完整性是指若基本关系 R 中含有另一个基本关系 S 的主关键字 Ks 所对应的属性组 F（F 称为 R 的外部关键字），则在关系 R 中的每个元组中的 F 上的值必须满足：

① 或是取空值（即 F 中的每个属性值均为空值）。

② 或等于 S 中某个元组的主关键字的值。

（3）用户定义的完整性是针对某一具体的数据库的约束条件。该条件是由现实世界中的应用环境决定的。它涉及到某一具体的应用中的数据所必须满足的语义要求。关系模型的 DBMS 应用提供定义和检验这类完整性条件的机制。

2.2.2　关系数据库描述

关系数据库描述，应理解为定义数据库的模式，它是由若干关系框架集合而成，根据关系数据模型的要求必须逐个对关系框架进行描述。

因关系从域出发定义，所以描述关系，首先对域进行描述，然后在域上定义各个关系模式。不同 RDBMS 的数据描述语言 DDL 不尽相同，采用的方式也不一样；一种采取问答式建立关系框架，另一种用专门的 DDL 语言写成关系模式，非问答式生成关系框架。

1. 问答式

问答式通过人机对话，由系统提问关系名，各个属性名及其类型和长度，对话完毕，关系框架随之建成，每个关系框架均通过问答，脱离应用程序单独建立，最终各个关系框架构成数据库概念模式，并被系统存储及管理，以备后用。

问答式使用简单，但人工干预多，速度较慢。

Access 数据库管理系统提供了问答式的关系数据库描述方式。

2. 语言描述式

语言描述式 DBMS，提供了专用的 DDL 语言用于定义数据库模式。

（1）域描述语句。域描述语句一般格式为：

```
DOMAIN domain_name DATATYPE（len） [CHECK]
```

其中，DATATYPE 为数据类型，CHECK 为约束。

（2）关系描述语句。关系描述语句格式如下：

```
RELATION relation_name (domain_name1,domain_name2,…)
            KEY=(domain_namei,domain_namej,…)
```

其中，KEY 子句定义关键字。

2.2.3 关系数据库操纵

关系数据语言可分为 3 类：数据描述语言（DDL），数据操纵语言（DML）和数据控制语言（DCL）。其中，DDL 负责数据库的描述，提供一种数据描述机制，用来描述数据库的特征或数据的逻辑结构。DML 负责数据库的操作，提供一种数据处理操作的机制。DCL 负责控制数据库的完整性和安全性，提供一种检验完整性和保证安全的机制。

DML 是用户经常使用的语言，包括 DBMS 的主要功能。DML 包括数据查询和数据的增、删、改等功能。其中查询的表达方式是 DML 的主要部分。关系数据库的 DML 按照查询方式可以分为两大类：

（1）用对关系的集合运算来表示查询方式，称为关系代数。

（2）用谓词演算来表达查询方式，称为关系演算；关系演算又可按谓词变元的基本对象是元组变量还是域又可分为元组关系演算和域关系演算两种。

关系代数和关系演算均是抽象的查询语言，是实际 DBMS 软件产品中实现的具体查询语言的理论基础。关系代数、元组关系演算及域关系演算 3 种语言在表达能力上是相互等价的。在实际的 DBMS 软件产品中查询语言可能是它们的综合。在关系数据库领域中广泛流行的结构化查询语言 SQL，就是介于关系代数和关系演算之间的一种语言，它不仅具有丰富的查询功能，而且还具有数据库定义和数据库控制功能。SQL 是集 DDL、DML、DCL 为一体的标准的关系数据库语言。

关系代数是以关系为运算对象的一组运算集合。它的运算可以分为两类：一类是传统的集合运算，包含并、交、差运算和笛卡尔积。这类运算将关系看成元组的集合，其运算是从关系的“水平”方向即行的角度进行的；另一类是针对数据库环境专门设计的关系运算，包括投影、选择、联接和除法等，这类运算不仅涉及行而且涉及列。下面对关系代数运算作进一步介绍。

2.3 关系运算

2.3.1 传统的集合运算

传统的集合运算是二目运算。设关系 R 和关系 S 具有相同的度，且相应的属性值取

自同一个域，则它们之间能进行并、交及差运算：

1. 并运算

两个关系 R 与 S 的并记为 R∪S，它是一个新的关系，由属于 R 或属于 S 的元组组成，可形式地定义为：

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

其中 t 是元组变量，表示关系中的元组。

2. 交运算

两个关系 R 与 S 的交记为 R∩S，它是由属于 R 且属于 S 的元组组成，可形式地定义为：

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

3. 差运算

两个关系 R 与 S 的并记为 R−S，它是由属于 R 但不属于 S 的元组组成，可形式地定义为：

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

图 2.5 给出了上述并、交与差运算的一个实例。

R

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

S

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

R∪S

A	B	C
A	c	e
A	d	f
B	d	e

R∩S

A	B	C
a	d	f
b	d	e

R−S

A	B	C
a	c	e

图 2.5　并、交、差运算的实例

4. 笛卡儿积

设 R 为 m 元关系，S 为 n 元关系，它们的笛卡儿积表示为 R×S，这个新关系具有 m+n 元，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，它可以用下列形式表示：

R

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

S

B	D
1	2
3	4

R×S

A	R.B	C	S.B	D
1	2	3	1	2
1	2	3	3	4
4	5	6	1	2
4	5	6	3	4
7	8	9	1	2
7	8	9	3	4

图 2.6　笛卡儿积实例

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

2.3.2 专门的关系运算

1. 选择运算

选择运算是从某个给定的关系中筛选出满足限定条件的元组子集，它是一元关系运算。可形式定义为：

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

其中，F 是筛选关系 R 中元组的限定条件的布尔表达式，它由逻辑运算符∧、∨、¬ 连接各算术表达式组成。

2. 投影运算

选择运算是从某个关系中选取一个“行”的子集，而投影运算实际上是生成一个关系的“列”的子集，它从给定的关系中保留指定的属性子集而删去其余属性。设某关系 R（X），X 是 R 的属性集，A 是 X 的一个子集，则 R 在 A 的投影可表示为：

$$\Pi_A(R)=\{t[A] \mid t\in R\}$$

其中，t[A]表示元组相应 A 属性中的分量。

3. 连接运算

连接运算是从两个给定的关系的笛卡尔积中选取满足一定条件的元组子集，可形式定义为：

$$R \bowtie S(A\,\theta\,B)=\{rs \mid r\in R\wedge s\in S\wedge r[A]\,\theta\,s[B]\}$$

其中，A 是关系 R 中的属性组，B 是关系 S 中的属性组，它们的度数相同且可以比较。θ 为算术比较运算符（即＜、＞、≤、≥、＝、≠）。

4. 自然连接运算

自然连接运算是连接运算中最有意义的一种运算，是连接运算的一种特例。它要求参与运算的两个关系在同名属性上具有相同的值，由于同名属性上的值相同，所以在产生的结果关系中同名属性也只出现一次，可形式定义为：

R

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

S

B	D
2	9
8	3

$R \bowtie S$（C≤D）

A	R.B	C	S.B	D
1	2	3	2	9
1	2	3	8	3
4	5	6	8	9
7	8	9	8	9

$\sigma_{A>1}(R)$

A	B	C
4	5	6
7	8	9

$\Pi_{A,B}(R)$

A	B
1	2
4	5
7	9

R×S

A	B	C	D
1	2	3	9
7	8	9	3

图 2.7 选择、投影、联接与自然联接的实例

$$R \times S = \{rs[-B] \mid r \in R \wedge s \in S \wedge r[A]=s[B]\}$$

其中，A 是关系 R 中的属性组，B 是关系 S 中的属性组，它们的度数相同且可以比较。

图 2.7 给出了选择、投影、连接与自然连接的一个实例。

5. 除法运算

一个 m 元关系 R 除以一外 n 元关系 S（其中 m>n，S 非空关系并且 R 中存在 n 个属性与 S 的 n 个属性定义在相同的域）所得到的结果是一个（m−n）元的新的关系，它表示满足以下条件的元组集合：

$$R \div S = \{t^{(m-n)} \mid \text{对任一 } t^{(n)} \in S \text{ 都有 } t^{(m-n)} \cdot t^{(n)} \in R\}$$

其中 $t^{(m-n)} \cdot t^{(n)}$ 表示一个（m−n）元的元组和一个 n 元的元组拼合成一个 m 元的新元组。图 2.8 给出了除法运算的实例。

S

S#	C#	G
101	C1	A
101	C2	B
101	C3	C
102	C2	C
102	C3	A
103	C1	C
104	C3	B
105	C2	B

R1

C#
C3

R2

C#
C1
C3

R3

C#	G
C2	B

S÷R1

S#	G
101	C
102	A
104	B

$\Pi_{S\#,\,C\#}(S) \div R2$

S#
101

S÷R3

S#
101
105

图 2.8　除法运算的实例

2.3.3　关系代数运算举例

在关系代数中，关系与关系可以经过有限次的代数运算，它的运算结果仍是一个关系。可以用关系代数表达式表达所需要进行的各种数据库查询和更新处理的需求。

设有学生—课程关系数据库，学生关系 S（学号 S#，姓名 SN，系名 SD，年龄 SA）、课程关系 C（课程号 C#，课程名 CN，任课教师 CT）和学生选课关系 SC（学号 S#，课程号 C#，成绩 G），如图 2.9 所示。

求计算机科学系 CS 的学生：

$$\sigma_{SD='CS'}(S)$$

检索学习课程 C2 的学生的学号与成绩：

$$\Pi_{S\#,\,G}(\sigma_{C\#='C2'}(SC))$$

求出学习全部课程的学生名单：

$$\Pi_{SN}(S \bowtie (\Pi_{S\#,\,C\#}(SC) \div \Pi_{C\#}(C)))$$

将新开课程记录（'C6','Q',R'）插入到关系 C 中：

$$C \cup \{('C6','Q','R')\}$$

将学号为 S3 学生的 C4 课程的成绩改为 A：

（SC−{('S3','C4','?')}∪{（'S3','C4','A'）}

求同时选修课程 C1 和 C2 的学生姓名：

$$C12=\Pi_{C\#}\ （\sigma_{C\#='C1'\ OR\ C\#='C2'}（C））$$

$$SC12=\Pi_{S\#,\ C\#}（SC）\div C12$$

$$S_{12}=\Pi_{SN}（S \bowtie SC12）$$

S

S#	SN	SD	SA
S1	A	CS	20
S2	B	CS	21
S3	C	MA	19
S4	D	CI	19
S5	E	MA	20
S6	F	CS	22

C

C#	CN	CT
C1	G	L
C2	H	M
C3	I	O
C4	J	P
C5	K	

SC

S#	C#	G
S1	C1	A
S1	C2	A
S1	C3	A
S1	C5	B
S2	C1	B
S2	C2	C
S2	C4	C
S3	C2	B
S3	C3	C
S4	C4	B
S4	C5	D
S5	C2	C
S5	C3	B
S5	C5	B
S6	C4	A
S6	C5	A

图 2.9 学生课程数据库

2.4 Access 数据库的建立

当我们想做一件事情的时候，一般都会先考虑一下，然后再去做。在建立一个新数据库的时候，也要想一想这个数据库是用来干什么的，它要存储那些数据信息，这些数据之间又有什么关系？一方面要知道哪些数据是必须的，是绝对不能缺少的，不然建立数据库获取信息的目的就没法达到了；另一方面也要知道那些数据是不必要，放在数据库当中只会增加数据库的容量，却并不起任何作用，所以要将这些冗余的数据剔除。这样建立起来的数据库才既能满足我们检索数据的需要，又能节省数据的存储空间。

了解了在建立一个数据库之前应该注意的问题后，现在就从最基本的新建一个空数据库开始，了解一下 Access 2000 数据库的结构。

2.4.1 建立数据库与表

1. 建立 Access 空数据库

在 Access 2000 中，新建一个空数据库很简单，只要用鼠标单击 Access 窗口左上角

数据库工具栏中的“新建”按钮，就会在屏幕上弹出一个“新建”对话框。

在这个对话框弹出以后，选择“常用”选项卡，并在“常用”选项卡上用鼠标左键双击“数据库”图标。屏幕上弹出“文件新建数据库”对话框，在“文件名”中给新建的数据库文件取名“db1”，此时，机器显示如图 1.7 所示的窗口，一个空数据库 db1 就建好了。

2. 建立数据表

现在简单介绍使用表设计器来创建一个表。在图 1.7 的数据库窗口中，在“对象”列表中选择“表”，然后双击“使用设计器创建表”选项，弹出“XS：表”对话框，如图 2.10 所示。

对话框分为两个部分，上半部分是表设计器，下半部分用来定义表中字段的属性。表的设计器其实就是一个数据表，只是在这个数据表中只有“字段名称”、“数据类型”和“说明”3 列，当我们要建立一个表的时候，只要在设计器“字段名称”列中输入表中需要字段的名称，并在“数据类型”列中定义那些字段的“数据类型”就可以了。设计器中的“说明”列中可以让表的制作人对那些字段进行说明，以便以后修改表时能知道当时为什么设计这些字段。

图 2.10　数据表的建立

3. 字段的数据类型

Access 2000 可使用的数据类型有：文本（Text）型、备注（Memo）型、数字（Number）型、时期及时间（Date/Time）型、货币（Currency）型、自动编号（AutoNumber）型、是/否（Yes/No）型、OLE 对象（Object）型、超链接（Hyperlink）型和查阅向导（Lookup Wizard）型。

4. 字段的属性

每一个字段都有一些用于自定义字段数据的保存、处理或显示的属性。

（1）字段大小：规定文本型或数字型的允许填充的长度。

（2）小数位数：指定小数型数的小数位数。

（3）格式：指定数据显示或打印的格式。

（4）输入法模式：对于大量输入中文字段，可以将其输入法模式设置为“输入法开启”，当光标移到段时，输入法窗口会自动打开。

（5）输入掩码：指定输入数据时的格式。

（6）标题：指定在数据表视图或窗体中显示该字段时所用的标题。

（7）默认值：添加新记录时，自动加入到字段中的值。

（8）有效性规则：用于限制输入数据的表达式，如：“<100”。

（9）有效性文本：设置在数据不符合有效性规则时所显示的出错提示信息。

（10）必填字段：指定该字段是否必须输入数据。

（11）允许空字符串：用于文本型字段，设置是否允许输入空字符串。

（12）索引：设置该字段是否进行索引以及索引的方式。

2.4.2 维护数据库与表

1. 表中的数据编辑

数据表中的数据编辑，主要有增、删和改，操作方法如同 Word。如输入数据到学生（XS）表中，如图 2.11 所示。

图 2.11 表中数据编辑

2. 表中记录的排序

选中某字段，单击工具条上的“升序”或“降序”按钮。

3. 表中记录的筛选

筛选记录是通过指定筛选条件来完成的，选中某一列，选择“记录”、“筛选”。

4. 创建表与表之间的关系

所谓创建关系，就是在表与表之间指定相关联的字段，以及关联的方式和属性。创

建了表与表之间的关系后，Access 2000 将实现以下功能：

（1）创建查询时自动设置表与表之间的关系。

（2）实施参照完整性，包括自动级联更新相关字段和自动级联删除相关记录。

（3）在数据表视图中显示子数据表。

可用菜单中“工具”下的“关系”或者工具条中关系按钮创建关系。

5. 修改表的设计

修改数据表的结构与创建数据表相似。

小　结

本章介绍了关系、关系数据模式等基本概念，从数学上可定义为笛卡儿积的有意义子集即为关系，关系数据库中的数据与联系均用关系进行描述，关系代数运算包括：传统集合运算（并、交、差等）和专门的关系运算（选择、投影、连接、自然连接、除法等）。

习　题

1．试述关系模型的 3 个组成部分。

2．描述并理解以下术语：域、笛卡儿积、关系、元组和属性。

3．设有关系 R（A，B，C）和 S（A，B，C），如图 2.12 所示，试计算 R 与 S 的并、交、差。

R

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

S

A	B	C
b	g	a
d	c	d

图 2.12　关系 R（A，B，C）和 S（A，B，C）

4. 设有关系 R 与 S，如图 2.13 所示，试计算 R 与 S 的笛卡儿积、R 与 S 的连接（B<C）、在 R 与 S 的笛卡儿积的基础上进行选择运算（A=C）。

R

A	B
a	b
c	b
d	e

S

C	D
c	b
a	e
d	b

图 2.13　关系 R 与 S

5．已知学生选课数据库 3 个关系：S（学生）、C（课程）、SC（学生选课）

S（S#，SN，SD，SA）（学号、姓名、系名、年龄）

C（C#，CN）（课号、课名）

SC（S#，C#，GS）（学号、课号、成绩）

试用关系代数表达式表示下列查询要求：

（1）取出所有学生的信息。

（2）取出选修课程号为 C2 的学生姓名及所在的系。

（3）取出同时选修课程号为 C1 和 C2 的学生姓名。

（4）取出选修“BASIC 语言”课程的学生姓名。

（5）取出年龄大于 20 岁的计算机系的学生姓名。

（6）取出不选“操作系统”课程的学生姓名和年龄。

（7）取出学号为 S1 的学生选修的课程号和课程名。

（8）取出全部学生都选修的课程号和课程名。

第 3 章　关系数据库语言 SQL

本章要点

- SQL 数据定义、数据操纵、数据控制及数据查询
- Access 数据查询

本章难点

- 数据操纵、数据控制及数据查询

SQL（结构化查询语言）是 Structured Query Language 的缩写。由于它具有功能丰富、使用灵活、语言简洁等特点，深受计算机用户的欢迎，许多数据库生产厂家推出各自支持 SQL 的软件。它是在 1974 年由 Boyce 和 Chamberlin 提出来的，1986 年被美国国家标准局（ANSI）的数据库委员会批准为关系数据库语言的美国标准，1989 年被国际标准化组织（ISO）定为国际标准，使 SQL 语言成为标准关系数据库语言，1990 年我国也颁布了《信息处理系统数据库语言 SQL》，并将其定为国家标准。

SQL 是一种介于关系代数和关系演算之间的一种结构化查询语言，它的主要功能包括数据定义、数据操纵及数据控制等方面，数据操纵又可以分为数据检索（查询）和数据更新两个方面。它是一种综合的、通用的、功能极强的关系数据库语言。

3.1　SQL 数据定义

3.1.1　SQL 数据库的体系结构

在具体介绍数据定义功能之前，首先需要了解 SQL 语言支持的关系数据库三级模式结构。SQL 语言可以对两种基本数据结构进行操作，一种是“表”，另一种是“视图（View）”。视图是由不同的数据库中满足一定条件约束的数据所组成，用户可以像基本表一样对视图进行操作。当对视图操作时，由系统转换成对基本表的操作。视图可以作为某个用户的专用数据部分，这样便于用户使用，提高了数据的独立性，有利于数据的安全保密。

SQL 语言支持关系数据库三级模式结构，如图 3.1 所示。用户可以用 SQL 语言对视图和基本表进行查询等操作，在用户观点里，视图和基本表都是关系。视图是从一个或几个基本表导出的表，它本身不独立存储在数据库中，即数据库中只存储视图的定义而

不存储对应的数据，因此视图是一个虚表。视图在概念上与基本表等同，用户可在视图上再定义新的视图。基本表是本身独立存在的表，一个（或多个）基本表对应一个存储文件，一个表可以带若干索引，索引也存放在存储文件中。存储文件的逻辑结构组成了关系数据库的内模式。

在 SQL 中，关系模式称为基本表，存储模式称为存储文件，子模式称为视图，元组称为行，属性称为列。

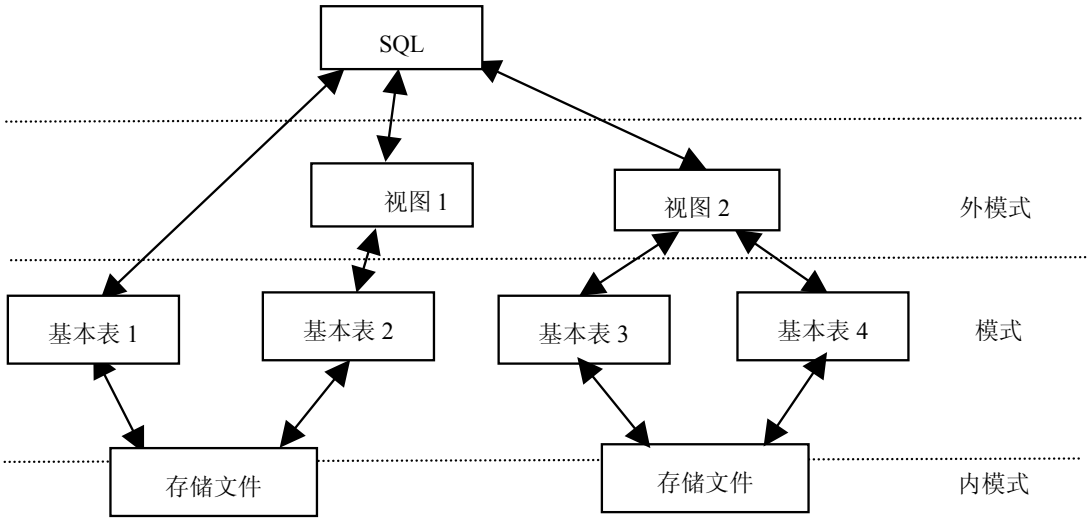

图 3.1　SQL 支持的数据库体系结构

3.1.2　基本表的定义与删改

SQL 的数据定义功能包括 3 部分：定义基本表、定义视图和定义索引。

我们以一个简单的图书管理关系数据模型为基础，通过示例来介绍 SQL 的使用方法。设图书管理关系数据模型包括以下 3 个关系模式：

图书（图书编号，书名，作者，出版社，单价）

读者（借书证号，姓名，性别，单位，职称，地址）

借阅（借书证号，图书编号，借阅日期，备注）

1. 定义基本表

定义一个基本表相当于建立一个新的关系模式，但尚未输入数据，只是一个空的关系框架。系统将一个基本表的数据描述存入数据字典中，供系统或用户查阅。定义基本表就是创建一个基本表，对表名（关系名）和它所包括的各个属性及其数据类型做出具体规定。不同的系统支持的数据类型有区别，但一般都支持如下数据类型：

char（n）　　字符串，长度为 n 个西文字符

decimal（x,y）　　十进制数，包括小数点及符号位共 x 位，其中 y 位小数

例 3.1　创建一个图书表。

```
CREATE  TABLE 图书（图书编号    CHAR（8） NOT NULL,
                    书名        CHAR（30）,
                    单价        DECIMAL（10，2））;
```

用 NOT NULL 指出该属性在输入数据时不允许有空值。在一般情况下不允许主关键字为空值，而其他属性可以暂时不填写，或是未知的值。

2. 修改基本表

修改基本表是指对已经定义的基本表增加、删除或修改某一列（属性）。

例 3.2　在图书表中增加作者和出版社两个列。

```
ALTER  TABLE  图书
        ADD（作者  CHAR（8），出版社 CHAR（20））;
```

新增加的属性处于表的最后一列。如果被修改的基本表原来已经有了数据，各个记录中新增加的属性全部是空值，有待以后用更新语句修改。在命令中可以用 ALTER COLUMN 修改某一列，DROP 删除某一列。

3. 删除基本表

删除基本表是把表的定义、表中的数据、相应的索引以及以该基本表为基础所建立的所有视图全部删除，并释放所占用的存储空间。

例 3.3　删除图书表。

```
DROP  TABLE  图书;
```

3.1.3　索引的建立与删除

为了提高数据的检索效率，对于一个基本表，可以根据应用环境的需要建立若干索引，以提供多种存取方式。通常，索引的建立和删除由 DBA 或表的主人（即建立表的人）负责。用户不必也不能存取数据时选择索引。存取路径的选择由系统自动进行。索引的描述存放在数据字典中。

1. 建立索引

例 3.4　为基本表图书建立一个按图书编号升序的索引，名为 TSNO。

```
CREATE  INDEX  TSNO
         ON  图书（图书编号  ASC）;
```

2. 删除索引

例 3.5　删除索引 TSNO。

```
DROP INDEX  TSNO;
```

3.1.4　视图的定义与删除

数据库系统中的基本表包含多个用户共享的数据，某一个具体应用可能只使用其中一部分数据，基本表的格式也可能不直接满足用户要求。可以从一个或几个基本表以及已有的视图中导出适合具体应用的视图。用户对视图的查询与基本表一样。从用户观点来看，基本表和视图都是关系。但由于视图是虚表，它并不对应一个存储的数据文件，因此通过视图对数据的修改要受到一定的限制。

建立视图有两个作用：可简化查询命令；可限制某些用户的查询范围。即视图本身是对若干数据表的一种检索，并能将视图以外的数据屏蔽起来，起到对数据的安全和保密作用。

1. 定义视图

例 3.6 建立信息工程系的读者视图，名称为 XX_READER。

```
CREATE VIEW  XX_READER（姓名,性别,职称,家庭地址）
AS
SELECT  姓名,性别,职称,地址
FROM   读者
WHERE  单位="信息工程系";
```

如果所建视图的属性与子查询的 SELECT 子句相同，可以省略不写。建立视图的 SELECT 子句中不能含有 ORDER BY、COMPUTE 等子句，不能含有 INTO 等关键字，即视图中的 SELECT 子句受到一定的限制。关于 SELECT 子句下一节将作较为详细的介绍。

例 3.7 建立各个单位当前借阅图书情况的简单统计视图，名称为 TJ_READER。

```
CREATE  VIEW TJ_READER（单位,借书人数,借阅人次）
AS
SELECT 单位,COUNT（DISTINT 借书证号）,COUNT（图书编号）
FROM 借阅,读者
WHERE 读者.借书证号=借阅.借书证号
GROUP BY 单位;
```


在本例中，视图是基本表数据的动态窗口。对这个视图的直接查询便得到各单位当前借阅图书的情况统计。每次查询这一视图结果是变化的，而不是定义视图当时的静态拷贝。

2. 删除视图

例 3.8 删除视图 XX_READER。

```
DROP VIEW XX_READER;
```

取消视图后，其定义以及在它基础上所建立的其他视图将自动删除。

3.2 SQL 数据查询

SQL 的查询语句也称 SELECT 命令，其基本形式是 SELECT-FROM-WHERE 查询块。多个查询块可以逐层嵌套执行。SELECT 命令是 SQL 结构化查询语言最具特色的核心语句，使用方便，查询速度块。

3.2.1 简单查询

这里所指的简单查询是指只涉及一个关系的查询。

例 3.9 查找所有读者的全部情况。

```
SELECT *
FROM 读者;
```

SELECT 子句中的星号*表示选择了关系的全部属性，由于查询中无条件限制，所以省略了 WHERE 子句，但要注意在语句块的末尾必须加分号。

例 3.10　列出图书馆中所有藏书的书名和出版社。

```
SELECT  书名,出版社
FROM  图书;
```

例 3.11　查找读者黄刚所在的单位。

```
SELECT  姓名,单位
FROM 读者
WHERE  姓名="黄刚";
```

例 3.12　查找价格在 10 元和 20 元之间的图书，结果按单价升序排列。

```
SELECT  书名,作者,单价,出版社
FROM  图书
WHERE  单价  BETWEEN  10  AND 20
ORDER  BY  出版社,单价  ASC;
```

WHERE 单价 BETWEEN 10 AND 20 子句相当于 WHERE 单价>=10 AND 单价<=20。用 ASC 表示升序，也可以省略，降序用 DESC。SQL 允许多重排序，ORDER BY 后面按次序给出主排序关键字和次排序关键字。输出结果先按主排序关键字的值排序，在主关键字值相等的情况下，再按次关键字的值排序。

例 3.13　查找清华大学出版社和北京大学出版社的所有图书及作者。

```
SELECT  书名,作者,出版社
FROM  图书
WHERE  出版社 IN （"清华大学出版社","北京大学出版社"）;
```

谓词 IN 表示属于，即什么在某集合中。它可以用一个或几个 OR 来代替，如：WHERE 出版社 IN （"清华大学出版社"，"北京大学出版社"）可以表示成 WHERE 出版社="清华大学出版社" OR 出版社="北京大学出版社"。

例 3.14　查找书名以“数据库”开头的所有图书及作者。

```
SELECT 书名,作者
FROM 图书
WHERE 书名 LIKE "数据库%";
```

谓词 LIKE 后面必须是字符串常量，其中可以使用两个通配符。下划线_代表任意一个字符，百分号%代表任意多个字符。例如：

WHERE 书名 LIKE "%数据库%"：表示包含"数据库"的书名。

WHERE 书名 LIKE "%数据库"：表示以"数据库"结尾的书名。

WHERE 作者 LIKE "%强_"：表示作者姓名至少 4 个字符（两个汉字）且倒数第 2 个汉字必须是“强”字。

3.2.2　连接查询

如果查询某查询要涉及两个或多个关系，往往要进行连接运算。由于 SQL 是高度非

过程化的，用户只要在 FROM 子句中指出关系名称，在 WHERE 子句写明连接条件即可，连接运算由系统去完成并实现优化。

例 3.15 查找所有借阅了图书的读者姓名及所在单位。

```
SELECT DIST 姓名,单位
FROM 读者,借阅
WHERE 读者.借书证号=借阅.借书证号;
```

注意 如果不同关系中有相同的属性名，为了避免混淆，应当在前面冠以关系名，并用.分开，用 DIST 表示无论一位读者借几本书，在输出结果中只出现一次。

例 3.16 找出李晶所借的所有图书的书名及借阅日期。

```
SELECT "李晶所借的图书：",书名，借阅日期
FROM 图书,借阅,读者
WHERE 读者.借书证号=借阅.借书证号
                AND 借阅.图书编号＝图书.图书编号
                AND 姓名="李晶";
```

该查询涉及到 3 个关系之间的自然连接，用户只需用外关键字指出连接条件。SELECT 子句中允许有字符串常量，例中“李晶所借的图书：”是提醒用户，使查询结果易于阅读。

例 3.17 查找价格在 20 元以上已借出的图书，结果按单价降序排列。

```
SELECT *
FROM 图书,借阅
WHERE 图书.图书编号=借阅.图书编号 AND 单价＞=20
ORDER BY 单价 DESC
```


这里 SELECT *代表图书和借阅两个关系连接后的所有属性。

3.2.3 嵌套查询

嵌套查询是指在 SELECT-FROM-WHERE 查询块内部再嵌入另一个查询块，称为子查询，并允许多层嵌套。由于 ORDER 子句是对最终查询结果的输出排序提出要求，因此它不能出现在子查询当中。

例 3.18 找出借阅了“C 语言”一书的读者姓名及所在单位。

```
SELECT 姓名,单位
FROM 读者
WHERE 借书证号 IN（SELECT 借书证号
                    FROM 借阅
                    WHERE 图书编号 IN （SELECT 图书编号
                                        FROM 图书
                                        WHERE 书名="C 语言"））;
```

在执行嵌套查询时，每一个内层子查询是在上一级外层处理之前完成的，即外层用到内层的查询结果。从形式上看是自下向上进行处理的。从这个规律出发，按照手工查询的思路来组织嵌套查询就轻而易举了。

在嵌套查询中最常用的是谓词是 IN，由于查询的外层用到内层的查询结果，用户事先并不知道内层结果，这里的 IN 就不能用一系列 OR 来代替。另外，许多嵌套查询可以

用连接查询完成。但并非所有的嵌套查询都能用连接查询替代，有时结合使用更显得简洁、方便。

例 3.19　找出读者的姓名、所在单位，他们与“王明”在同一天借了书。

```
SELECT 姓名,单位,借阅日期
FROM 读者,借阅
WHERE 借阅.借书证号=读者.借书证号 AND 借阅日期 IN
      (SELECT 借书日期
       FROM 借阅,读者
       WHERE 借阅.借书证号=读者.借书证号 AND 姓名="王明");
```

3.2.4　使用库函数查询

SQL 提供的常用的统计函数称为库函数。这些库函数使检索功能进一步增强。它们的自变量是表达式的值，是按列计算的，最简单的表达式是属性，也就是列。

SQL 的库函数有：

（1）计数函数 COUNT（*） 计算元组的个数；COUNT（属性名）对列的值计算个数。

（2）求和函数 SUM 对某一列的值求和（属性必须是数值类型）。

（3）计算平均值 AVG 对某一列值计算平均值（属性必须是数值类型）。

（4）求最大（小）值 MAX（MIN） 找出一列值中的最大（小）值。

例 3.20　求藏书总册数。

```
SELECT "藏书总册数：",COUNT（*）
FROM 图书;
```

例 3.21　求计算机科学系当前借阅了图书的读者人数。

```
SELECT"借书人数：",COUNT（DISTINCT 借书证号）
FROM 借阅
WHERE 借书证号 IN（SELECT 借书证号
                   FROM 读者
                   WHERE 单位="计算机科学系"）;
```

如果询询的是该系当前借阅图书的总人次，则应省略 DISTINCT。

例 3.22　求科学出版社图书的最高价格、最低价格和平均价格。

```
SELECT MAX（单价）,MIN（单价）,AVG（单价）
FROM 图书
WHERE 出版社="科学出版社";
```

例 3.23　求出各个出版杜图书的最高价格、最低价格和平均价格。

```
SELECT 出版社,MAX（单价）,MIN（单价）,AVG（单价）
FROM 图书
GROUP BY 出版社;
```

其中 GROUP BY 的作用是按属性的取值对元组分组，然后对每一组分别使用库函数。在此例中，有几个出版社就分几个组，按组分别计算最高价格、最低价格和平均价格。

例 3.24　找出藏书中各个出版社的册数、价值总额，并按总价降序，总价相同者按

册数降序排列。

```
SELECT 出版社,"册数：",COUNT（*）,"总价：",SUM（单价）
FROM 图书
GROUP BY 出版社
ORDER BY SUM（单价）,COUNT（*）DESC;
```

例 3.25 找出当前至少借阅了 5 本图书的读者及所在单位。

```
SELECT 姓名,单位
FROM 读者
WHERE 借书证号 IN (SELECT 借书证号
                   FROM 借阅
                   GROUP BY 借书证号
                   HAVING COUNT（*）>=5);
```

例中的 HAVING 子句通常跟随在 GROUP BY 之后，其作用是限定检索条件。条件中必须包含库函数，否则条件可直接放到 WHERE 子句里。

例 3.26 找出没有借阅任何图书的读者及所在单位。

```
SELECT 姓名,单位
FROM 读者
WHERE NOT EXISTS（SELECT *
                  FROM 借阅
                  WHERE 借阅.借书证号=读者.借书证号）;
```

其中 EXISTS 表示存在，如果子查询结果非空，则满足条件；NOT EXISTS 正好相反，表示不存在，如果子查询结果为空，则满足条件。

本例中的查询称为相关子查询。子查询的查询条件依赖于外层的某个值（读者.借书证号），这里的子查询不能只处理一次，要反复求值以供外层查询使用。

在 WHERE 子句中，逻辑非 NOT 可以放在任何查询条件之前，也可以用在条件表达式之前。例如 NOT LIKE、NOT IN、NOT BETWEEN、NOT 单价>15 等。

SQL 查询功能很强，表现方式也很灵活，本例也可写为如下形式：

```
SELECT 姓名,单位
FROM 读者
WHERE 借书证号 NOT IN（SELECT 借书证号
                        FROM 借阅
                        WHERE 借阅.借书证号=读者.借书证号）;
```

在这种写法中，子查询只需处理一次。

3.2.5 集合运算查询

关系是元组的集合，可以进行传统的集合运算。集合运算包括：并 UNION、差 MINUS、交 INTERSECTION。求一个 SELECT 子查询的结果与另一个 SELECT 子查询结果的并、差、交，集合运算是以整个元组为单位的运算。因此，这些子查询目标的结构与类型必须互相匹配。集合运算结果将去掉重复元组。

例 3.27 有一个校友通信录关系，包含姓名、职称和单位属性，相应的数据定义与读者关系一致。求校友与读者中具有教授、副教授职称人员的并集。

```
SELECT 姓名,职称,单位
FROM 读者
WHERE 职称 IN（"教授”,"副教授"）
    UNION
SELECT 姓名,职称,单位
FROM 校友
WHERE 职称 IN（"教授","副教授"）;
```

SQL 语言中的 SELECT 句型灵活多样，所表达的语义可以从简单到复杂。通过上述例子可知 SELECT 语句的一般语法如下：

```
SELECT 查询目标
FROM 关系
[WHERE <条件表达式>]
[GROUP BY 分组属性名 [HAVING 组选择条件表达式]]
[ORDER BY 排序属性[DESC],…]
```

其中：SELECT 子句的查询目标可以用以下格式：

```
[DIST] *|表名.*|COUNT(*)|表达式[,表达式]…
```

表达式可以是由属性、库函数和常量用算术运算符组成的公式。

WHERE 子句的条件可以使用下列运算符：

算术比较运算符：=、>、<、>=、<=、!=、BETWEEN

逻辑运算符：AND、OR、NOT

集合运算符：IN、NOT IN

存在量词：EXISTS（SELECT 子查询）

集合运算符：并 UNION、交 INTERSECTION、差 MINUS

通配符：LIKE_、LIKE%

3.3　SQL 数据更新

3.3.1　插入数据

向基本表中插入数据的命令有两种格式，一种是向具体元组插入常量数据；另一种是把从子查询的结果输入到另一个关系中去。前者进行单元组（一行）插入，后者一次可插入多个元组。新增元组各个列（属性）的值必须符合定义。增加一个完整元组，并且属性顺序与定义一致，可在基本表名称后面省略属性名称。若插入一个元组的若干字段（属性）值，其他字段值暂时为空，此时，基本表名称后面的属性名称必须指明。

例 3.28　向图书基本表中新加一个元组。

```
INSERT INTO 图书
VALUES（"TP31/138","计算机基础","杨计算","高等教育出版社",18.00）;
```

例 3.29　向图书基本表中插入一个元组的部分字段。

```
INSERT INTO 图书（图书编号，书名，单价）
VALUES（"TP31/138","计算机基础",18.00）;
```

例 3.30 建立一个各单位借阅图书情况统计基本表，名称为 DWTJ，每隔一段时间，如一个月，向此基本表里追加一次数据。

```
CREATE TABLE DWTJ（单位     CHAR（20），
                  借书人数 INT,
                  借书人次 INT）;
INSERT  INTO  DWTJ （单位,借书人数,借书人次）
SELECT 单位,COUNT（DISTINCT 借书证号）,COUNT（借书证号）
FROM 借阅,读者
WHERE 读者.借书证号=借阅.借书证号
GROUP BY 单位;
```

3.3.2 修改数据

在修改命令中可以用 WHERE 子句限定条件，对满足条件的元组进行更新修改。若不写条件，则对所有元组更新。

例 3.31 将“图书编号”为“TP31/138”的图书填上作者和出版社。

```
UPDATE 图书
SET 作者＝"王为民",出版社="电子工业出版社"
WHERE 图书编号="TP31/138";
```

例 3.32 将所有图书的单价下调 5%。

```
UPDATE 图书
SET 单价＝单价*0.95;
```


例 3.33 把借书证号“7902070”改为“7912070”。

```
UPDATE 读者
SET 借书证号="7912070"
WHERE 借书证号="7902070";
UPDATE 借阅
SET 借书证号="7912070";
```

在修改同名属性时应当特别注意保持数据的一致性。第一个更新命令执行之后，数据库处于不一致状态，因为读者中的借书证号已有变动，而借阅中的借书证号未动。只有当下一个更新的命令执行过之后，数据库才又达到一个新的一致状态。

3.3.3 删除数据

删除命令比较简单，删除单位是元组，不是元组的部分属性。一次可以删除一个、几个元组，甚至将整个表删成空表，只保留表的结构定义。删除同名属性时也要注意保持数据的一致性。

例 3.34 删除借书证号“9011100”所有借阅登记。

```
DELETE
FROM 借阅
WHERE 借书证号="9011100";
```

例 3.35 删除借书证号以 90 开头的所有读者登记和借阅登记。

```
DELETE
```

```
FROM 读者
WHERE 借书证号="90%";
DELETE
FROM 借阅
WHERE 借阅证号="90%";
```

3.4　SQL 数据控制

数据控制是指通过对数据库各种权限的授予或回收来管理数据库系统。每个用户在系统登录时都要输入用户名称和口令，通过了系统的合法性检查之后才能使用系统。用户有定义基本表的权限，有使用自己所定义的基本表的所有权限。数据库管理员对数据库的所有资源拥有所有权限，包括数据定义、数据查询、数据更新和数据控制。

数据库管理员和基本表的定义者有权对基本表的各种权限授予别人，授权者也可以回收权限。非基本表的建立者必须经过授权才能使用不是自己定义的表，这些权限包括对基本表的修改（ALTER）、插入（INSERT）、删除（DELETE）、更新（UPDATE）、建立索引（INDEX）、查询（SELECT）和 ALL（所有权限）。

3.4.1　授权

授权语句的格式如下：

```
GRANT 权限表 ON TABLE 表名 TO 用户名表[WITH GRANT OPTION]
```


该语句把对指定表的某些权限授予若干用户，若选用 WITH GRANT OPTION 短语时，被授权的用户有权将获得的权限再授予其他用户，被授权的对象可以是 PUBLIC（所有用户），或具体的用户名。

例 3.36　将查询和更新各单位借阅图书情况统计基本表 DWTJ 的权限授予所有用户。

```
GRANT SELECT，UPDATE
ON TABLE DWTJ
TO PUBLIC;
```

注意　系统为了防止滥用权限，WITH GRANT OPTION 短语不能与 PUBLIC 同时使用。

3.4.2　回收权限

例 3.37　回收用户 LIMING 和 WANGWEI 对基本表 DWTJ 的更新权限。

```
REVOKE UPDATE
ON TABLE DWTJ
FROM LIMING,WANGWEI;
```

授予和回收权限是有层次的。设用户 A 把自己所建立的表“通信录”的所有权限授予了用户 B 并用 WITH GRANT OPTION 允许他再向别人授权。用户 B 又将对“通信录”的 UPDATE 和 SELECT 权授予用户 C。后来，用户 A 回收了用户 B 的 UPDATE 权限，用户 C 的这种权限自然也被取消了。

3.5 Access 数据库查询

建立数据库的目的是为了查询和使用数据库中的数据，查询用于按用户的需求和商业规则从表中提取数据。可将查询看成动态的数据集合。

3.5.1 用界面方式创建查询

在数据库窗口中，对象列中选择“查询”，双击“在设计视图中创建查询”，添加数据表，选择要显示的字段，设置选择条件，如图 3.2 所示。

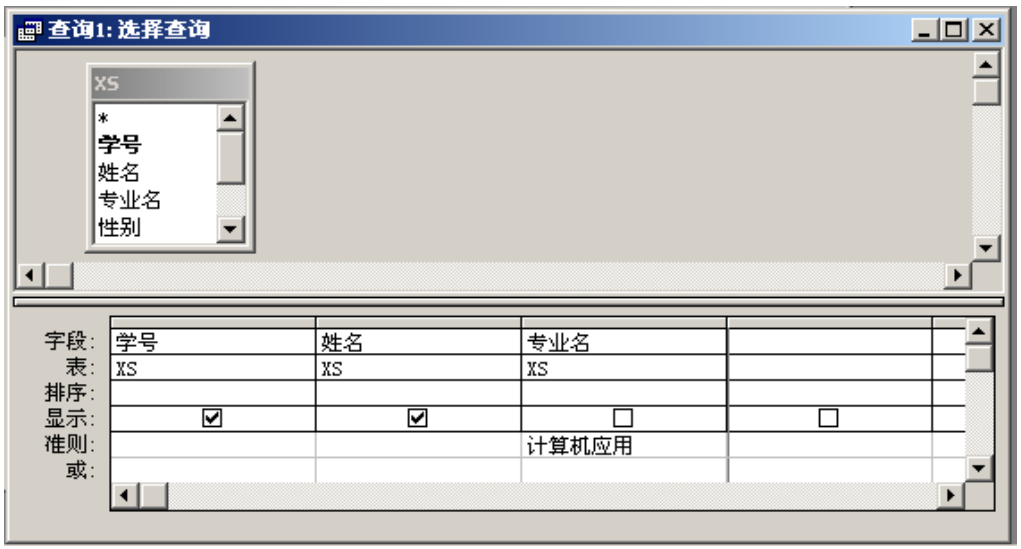

图 3.2 查询设计界面

设计完查询后，可以保存，以后就可以运行。运行结果如图 3.3 所示。

图 3.3 查询运行结果

3.5.2 使用 SQL 命令方式进行查询

在数据库窗口中，对象列中选择“查询”，双击“在设计视图中创建查询”，不需要添加数据表，在菜单“视图”中选择“SQL 视图”，出现如图 3.4 所示的界面。

输入 SQL 命令：

```
select 学号,姓名
from xs
```

```
where 专业名='计算机应用'
```

执行 SQL 命令，结果如图 3.3 所示。

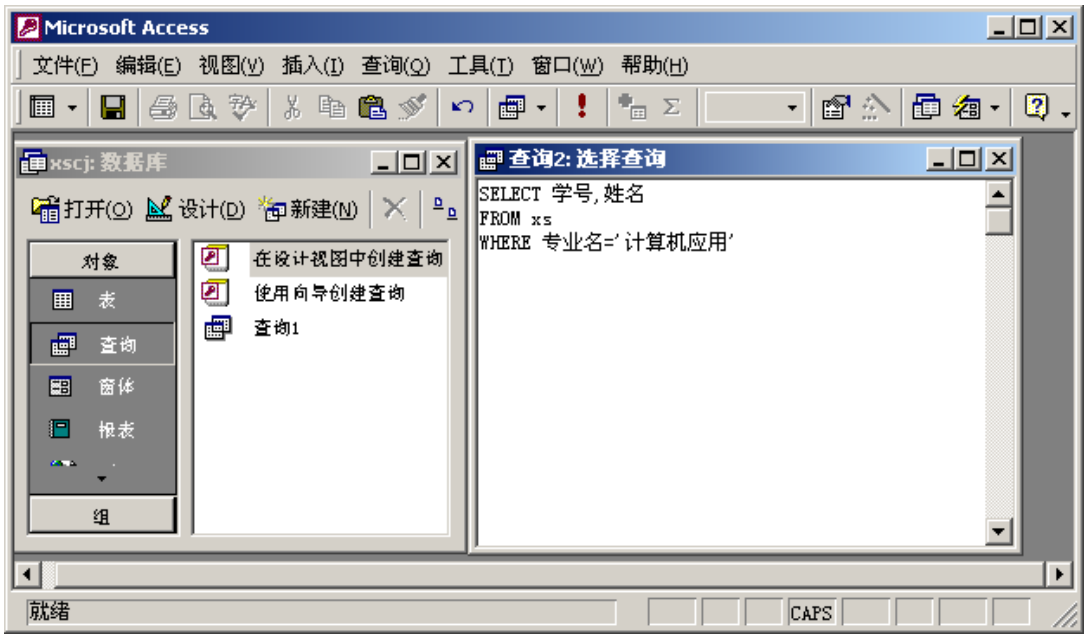

图 3.4　SQL 命令查询界面

小　结

本章介绍了关系数据库的标准语言和结构化查询语言 SQL。它具有语言一体化、高度非过程化、语言简洁、易于应用的特点。以一个图书管理关系数据模型为基础，通过示例讲解 SQL 的使用方法，SQL 的查询功能十分灵活，用 SELECT-FROM-WHERE 可以实现简单查询、连接查询、嵌套查询、使用库函数查询，并且能够进行集合运算。SQL 同样可以定义关系数据模式并录入数据，从而建立数据库。提供查询、更新、维护、数据库安全控制等一系列操作，即 SQL 能够实现数据库系统的全部活动。其全部功能概括如下：

数据定义　CREATE，DROP
数据查询　SELECT-FROM-WHERE
数据操纵　INSERT，DELETE，UPDATE
数据控制　GRANT，REVOKE

另外，SQL 支持关系数据库的三级模式结构，它的体系结构体现了数据库的三级结构和两级独立的特点。

习　题

1．SQL 语言包括哪几部分功能？

2．设有学生选修课程数据库：

S（S#，SNAME，AGE，SEX，DEPARTMENT，ADDRESS）
SC（S#，C#，GRADE）
C（C#，CNAME，TEACHER）

试用 SQL 语言查询下列问题。

（1）李老师所教的课程号、课程名称。

（2）年龄大于 23 岁的女学生的学号和姓名。

（3）“李小波”所选修的全部课程名称。

（4）所有成绩都在 80 分以上的学生姓名及所在系。

（5）没有选修“操作系统”课的学生姓名。

（6）英语成绩比数学成绩好的学生。

（7）至少选修两门以上课程的学生姓名、性别。

（8）选修了李老师所讲课程的学生人数。

（9）没有选修李老师所讲课程的学生。

（10）“操作系统”课程得最高分的学生的姓名、性别、所在系。

3．建立第 2 题中的数据表，数据类型和长度根据需要自行确定。