第一章　数据库管理系统概述

计算机的应用主要是数据处理。人们日常生活中接触到的大量数字、文字、图象、声音等，都可以以数据形式用计算机进行处理。数据处理的过程一般需经过原始数据的收集、编码转换（如汉字和字符等转换为合适的编码）、数据输入、数据处理和数据输出等五个步骤。

为了迅速、正确、有效地处理各种数据，就必须解决好数据管理的问题。数据管理技术的发展是与硬件、软件的发展紧密相连的，例如大量的数据必须存放在存储器中。因此，存储设备直接影响到数据管理技术的发展。1956年生产的第一台磁盘容量仅为5MB，而现在的大容量磁盘可达5000MB。随着光盘的广泛使用，存储容量还将进一步扩大，为数据库技术的发展提供良好的物质基础。

数据库是计算机软件的一个重要分支。虽然数据库技术的发展只有二十多年的历史，但无论从数据库技术的提高，还是从数据库应用的广泛性这两个侧面看，数据库发展的速度都是十分惊人的。

数据库管理技术的发展，大致经历过人工管理、文件系统和数据库三个阶段。

1. 人工管理阶段（五十年代中期以前）

当时的计算机主要用于科学计算，使用的外存储器是磁带、卡片和纸带等。当时还没有数据管理的专用软件，处理的数据一般不需要保存，数据与程序不独立，程序中要包括数据的存取方式；由于各应用程序处理的数据可能相同，因此，程序之间可能会有重复数据，数据的冗余较多，存取时也缺乏灵活性；数据存取一般不采用文件的方式。

2. 文件系统阶段（五十年代后期至六十年代中期）

计算机的外存已有磁盘等设备，软件已出现专门管理数据的文件系统。数据已可以长期保存在磁盘上，计算机的使用已从计算转向管理，对文件可以进行大量的查询、修改、插入等工作。程序可以通过文件名与数据发生联系，不必了解数据存储的物理结构。文件可以有索引文件、链接文件和直接存储文件等多种形式，但文件相互之间是独立的，缺乏联系，同样的数据可能在多个文件中重复存储，形成冗余，稍不注意，还会使同样的数据在不同的文件中变得不一致。

3. 数据库阶段（六十年代末开始）

随着数据量的不断增加，数据管理规模也相应地增大，加之硬件方面又产生了大容量的硬盘，使数据库管理技术应运而生。它采用了较复杂的数据结构，不仅能描述数据自身的特点，还能描述数据之间的联系。这种联系是通过存取数据的路径来实现的，通过存取路径表示数据间的联系是数据库与文件系统的根本区别。这些数据不再是面向特定的一个或几个应用程序，而是面向整个应用系统，不同的用户可以共同访问数据库中的每一项数据，实现了数据共享，减少了数据的冗余，避免了数据的不一致性。数据库管理系统还提供了较强的数据控制功能，如保障数据的完整性和安全性；在多用户操作时，避免不同用户程序互相干扰的并发控制；在数据库被破坏或数据不可靠时，管理系统能把数据恢复到以前的某个正确状态。

第一节　数据和信息

信息和数据是两个不同的概念。但它们之间常常容易混淆。数据是描述事物状态特性的数字，字符，以及所有输入到计算机中、被计算机加工处理的符号的集合。归纳起来，数据可分成两类，一类是数值化数据，如实验数据、统计数据等；另一类是非数值化的数据，如文字、声音等。无论是哪一类数据，计算机都是把它转化成二进制代码来处理的。信息是对人们有用的消息。数据是信息的载体，信息是对数据的解释。信息用二进制代码表示后，才能被计算机接受。这些二进制代码经过计算机处理后，得到的新数据又可以表示新的信息。

非数值数据是怎样转换成二进制代码供计算机处理的呢？计算机又是怎样把二进制代码数据转换成需要的图形、文字输出呢？下面我们就以汉字的输入输出为例加以说明。

输入一个汉字到计算机中并显示出来，必须经过以下三个步骤。

(1) 用一种编码输入汉字。按照“中华人民共和国国家标准信息交换用汉字编码字符集基本集 GB2312-80”的规定，该字符集共收集了汉字及各种图形符号 7445 个，其中汉字 6763 个，分为两级。第一级字符包括 3755 个；第二级包括 3008 个。字符集分成 94 个区，每个区有 94 个位置，区和位的取值范围都是 1～94。每个图形和符号都取不同的区号和位号，不存在重码，这种用十进制数表示的编码称为区位码。例如“中”的区位码是 5448。要记忆掌握 7000 多个编码十分困难，于是产生了其他的输入编码方法，如拼音码、五笔字型码、表形码、自然码等数百种编码方法。但无论用哪一种代码输入计算机，计算机接收的都是一连串的二进制代码。

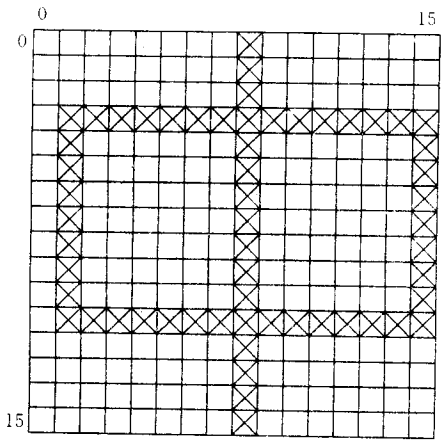
(2) 将各种输入编码转换成汉字的机内码。同一汉字，用不同的编码方法输入计算机，计算机接收的数据都不一样，例如用拼音方法输入一个“中”字的源代码为 ZHONG，用区位码时，源代码为 5448，使计算机无法统一处理这些数据。为此，必须将输入的汉字编码转化为统一的汉字机内码。这个工作是由汉字系统软件自身完成的。汉字的机内码与汉字的区位码之间存在着一一对应的关系，机内码由四位十六进制数组成，每个汉字的机内码是把它的区号和位号分别加上十六进制数 A0H 而得到的四位十六进制数，在计算机内部用两个字节的二进制代码来表示。(A0 后面的 H 表示这个数 A0 为十六进制数，它的值相当于十进制数 160。)

这样，在输入一段中文资料时，即使混杂使用了各种输入编码方法(如拼音、区位、五笔字型等)，得到的每一个汉字都对应着一个机内码。

(3) 两字节的汉字机内码并不是汉字本身的图形，要显示汉字，必须根据汉字的机内码，找出它对应的字型码(也称为字模码)，输出到指定的外部设备。目前使用的汉字字模方式，大多是以点阵方式来组成的。例如，通常在显示器上显示的汉字都采用 16×16 点阵方式，而用打印机打印出的汉字采用 24×24 点阵、32×32 点阵、48×48 点阵等方式。下面我们举例说明怎样利用二进制代码来显示汉字。

所谓点阵字型，就是把汉字图形置于网状方格上，每格对应于存储器的一个二进制位，有笔画的一格对应的二进制位取“1”，否则取“0”。例如采用 16×16 点阵的汉字显示方式，就是把一个方块分成 16 行、16 列，共有 256 个小格，每行、每列都是 16 格。每一小格对应一个二进制位，每一行对应十六个二进制位，即对应着两个字节的信息，一个汉字对应着 32 个

字节的信息。下面我们以“中”字为例说明 32 个字节中的信息是什么。



存储器中的信息

用二进制数据表示	用十六进制数表示
0000000010000000	0080
0000000010000000	0080
0000000010000000	0080
0111111111111111	7FFF
0100000010000001	4081
0100000010000001	4081
0100000010000001	4081
0100000010000001	4081
0100000010000001	4081
0100000010000001	4081
0111111111111111	7FFF
0000000010000000	0080
0000000010000000	0080
0000000010000000	0080
0000000010000000	0080

图 1-1　汉字点阵字形实例

综上所述，不难看出，信息和数据是两个不可分离又有区别的概念。信息是现实世界中的事物在人们头脑中的反映，通常人们可用文字和符号把它们记录下来。这些信息在计算机中又是以数据形式进行存取和处理的，处理结果又以文字、符号向人们提供各种有用的信息。

第二节　数 据 结 构

数据集合中的个体称为数据元素，通常还称作结点、记录等。一个数据元素常由一个或多个数据项组成，这些数据项也可称为域、字段等。例如，一个学校内教职工的基本情况构成一个数据集合，每个教职工的情况就是一个数据元素，一个数据元素又可分成姓名、性别、出生年月、专业、参加工作时间、职称等若干个数据项。

被计算机加工的数据元素不是孤立的，它们彼此之间存在着一种或多种特定的联系，这些联系在对数据存储和加工时都会被充分反映出来。因此，我们通常把相互之间存在一定联系的数据元素的集合称为数据结构。数据结构的概念大致包括三个内容，即数据的逻辑结构、存储结构和数据的运算。

数据的逻辑结构说明数据元素之间的逻辑关系，它只抽象地反映数据元素的结构，而不管这些数据元素在计算机中的具体存储方式。

数据的存储结构又称为数据的物理结构。数据的逻辑结构是直接面向用户的，对用户来说很直观，但这些数据元素在计算机中的存储方法却并不像逻辑结构一样，把逻辑结构映象到计算机中所得到的存储方式称为数据的存储结构。这种映象包括两方面的内容：一是数据元素各个数据项自身的取值，二是描述该数据元素与其他数据元素之间的联系（例如采用一个或多个指针表示结点之间的联系）。因此，一般情况下，用户看到的数据元素的逻辑结构与它们的物理结构是完全不同的。

数据的运算是定义在数据的逻辑结构上的，而具体的运算过程要在存储结构上实行。常用的运算有插入、删除、更新、排序、检索等。数据若采用不同的逻辑结构，运算的操作方法也不尽相同。

第三节　数据模型

模型是对现实世界的一种抽象。在数据库技术中，必须应用数据模型对现实世界进行抽象。数据库把许多相互关联的数据汇集在一起，并以一定的数据模型编排、存放，以形成一个科学的数据集合。简单地说，数据模型是一种表示实体类型及实体间联系的模型。实体是指客观存在的事物，例如某学校有许多教师，每个教师担任若干门课程，每门课程有不少学生选修。因此，我们可以确定三个实体类型，即教师记录类型(TEACHER)，课程记录类型(COURSE)和学生记录类型(STUDENT)。每个实体可以通过它的若干个特性来描述，每一个特性称为属性，每一个属性有一个取值范围——值域。在数据库中，每一个实体称为一个记录，一个属性称为一个字段。假设教师记录类型有教师工号、姓名 、职称三个字段，课程记录类型有课程号、课程名两个字段，学生记录类型有学号、姓名、性别、年龄、家庭住址五个字段。这三个实体之间还存在着一定的联系，称为实体的联系。TEACHER 和 COURSE 之间的联系记为 T-C，COURSE 和 STUDENT 之间的联系记为 C-S。每个联系也有一定的属性，其属性必须包括关键字。所谓关键字就是指实体集中能唯一标识某个实体(即每个元素)的属性或属性集。如教师实体中的工号可作为关键字，姓名就不宜作为关键字。联系的属性除含有关键字以外，还可含有其他属性。不妨设 T-C 联系的属性包括教师工号、课程号、任课节数等三个字段，C-S 联系的属性包括学号、课程号、成绩等三个字段。由此可知，一个联系实际上也可看成是一个特殊的实体类型。下面我们用实体联系图(也称为 E-R 图)来描述上例，可使这种实体联系模型变得清晰、简洁。在实体联系图中，用方框表示实体，通常用菱形框表示实体间的联系，用椭圆框表示实体或联系的属性。

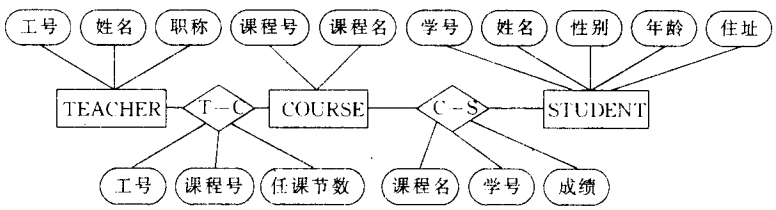


图 1-2　E-R 图实例

实体的联系有以下三种不同的类型。

(1) 1:1 联系。即实体集间的联系是一对一的联系，如一个学生只用一个学号，一个学号只对应一个学生。学生集合与学号集合之间的联系是 1:1 联系。

(2) 1:M 联系。即实体集间的一对多联系，如一个学生只属于一个班级，一个班级可拥有许多学生。班级和学生之间的联系是 1:M 联系。

(3) M:N 联系。即实体集间的多对多联系，如一个学生学习多门课程，一门课程由许多学生参加学习。学生集合与课程集合之间的联系是 M:N 联系。

在上例中，教师集合与课程集合之间的联系是 M:N 型的。同理，课程集合与学生集合

之间的联系也是 M:N 型的。

虽然实体联系模型已经是对现实世界的一种抽象，仍不能被现在的数据库管理系统立即接受。主要原因是它不能描述较详细的数据结构。一般来说，遇到一个实际问题，首先要把它变成实体联系模型，紧接着，还必须把实体联系模型变成某一种能为数据库管理系统接受的数据模型。数据模型的种类有层次模型、网状模型和关系模型三种。

(1) 层次模型。层次模型就像一棵倒置的树，是一种树型结构。例如，一个学校的组织机构就可用一个层次型数据模型来表示(见图 1-3)。

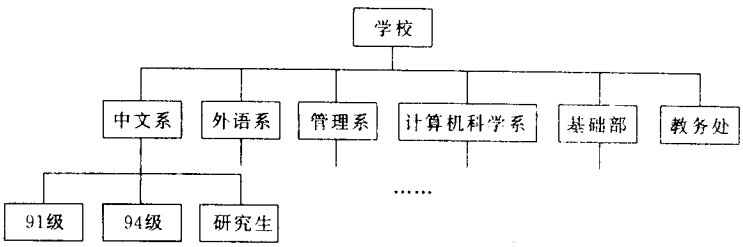


图 1-3　层次模型示意图

层次模型的特点是层次分明，结构清晰。但是，表示比较复杂的数据关系就要经过转换，故目前较少使用这类模型。

(2) 网状模型。网状模型用一般的图或网来表示数据之间的关系。如上面讲述的教师、课程、学生三个实体及其联系就可组织成如图 1-4 所示的网状数据模型。

网状模型的数据库管理系统一般都在大中型计算机上使用，效率较高，但编写应用程序较为复杂。层次模型还可看成网状模型的特例。

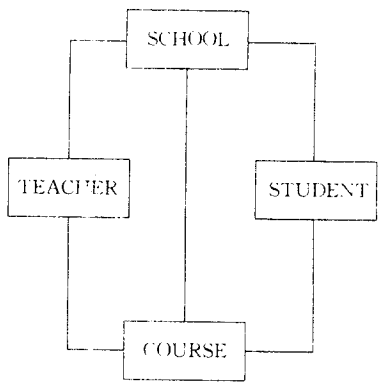


图 1-4　网状模型示意图

(3) 关系模型。关系模型是一种比较简单的数据模型，用户易懂，很容易和他们的日常生活经验、处理数据方法联系起来，所以被数据库管理系统广泛采用。关系模型的主要特征是用表格形式来表示实体及实体间的联系。一个关系模型往往由若干个关系组成，一个关系实际上是一张表格。在上面我们介绍过的 E-R 联系图中，每一个实体和联系都可以分别用一张表格来表示，关系名就可用实体或联系的名称来表示。如表 1-1 所示。

表 1-1　关系模型

TEACHER 关系

工号	姓　名	职称
B001	王文杰	一级
D002	宋国华	高级
I003	胡月英	二级
H001	黄兴生	高级
D011	张　萍	一级
A002	李晓蕾	高级
.....		

T-C 关系

工号	课程号	任课节数
B001	02	10
D002	03	10
I003	01	14
H001	05	15
D011	03	10
A002	09	16
.....		

COURSE 关系

课程号	课程名
02	高一政治
03	高一语文
01	高一数学
05	高一物理
03	高一化学
09	高一英语
……	

C-E 关系

课程号	学号	成绩
01	91001	82
02	91001	84
03	91001	95
……	……	……
01	91002	78
……	……	……
01	91003	65
……	……	……

STUDENT 关系

学 号	姓 名	性 别	年 龄	家庭住址
91001	李 明	男	14	南京路 10 号
91002	王 丽	女	14	福建路 40 号
91003	马 虹	女	13	淮海路 99 号
91004	陈 刚	男	16	兰绫 103 号
91005	张 萍	女	14	花园路 48 号
……	……	……	……	……

关系模型和层次模型、网状模型的最大差别在于关系模型是用表格数据而不是用指针来实现数据之间的联系。用户容易看懂，只需用简单的查询语句就可对用关系模型建立的数据库进行操作。关系模型本身并不涉及存储结构、访问数据和技术方面的问题，因为这些问题已交给数据库管理系统去做了。正因为如此，用关系模型来设计数据库管理系统就比较复杂，效率也较低，但这些工作只需少数设计人员去完成就可以了，而且由于计算机运算速度的加快，效率问题也变得不突出了。

关系的每一行称为一个记录，也可称为一个数据元素或一个元组，它的每一列称为一个字段(FIELD)，也可叫作数据项、属性等。每一个字段都有一个取值范围，称为字段值(或称属性值)。每一列的标题称为字段名，每个记录由若干个字段(即若干个数据项)组成，一个关系由若干个记录组成。一个关系的所有字段放在一起，写成关系名(字段名 1，字段名 2，……字段名 N)的形式，称为一个关系式。

例如图 1-2 中的五个关系写成如下关系模式：

TEACHER(工号，姓名，职称)

T-C(工号，课程号，任课节数)

COURSE(课程号，课程名)

C-S(课程号，学号，成绩)

STUDENT(学号，姓名，性别，年龄，住址)

每一个关系必须具备下列基本性质：

(1) 任意两个记录不能完全相同，即没有重复记录；

(2) 行的顺序可以任意，即可任意变换两个记录的次序；

(3) 列的顺序可以任意，即可任意交换两列的次序，但交换时必须把字段名连同字段值一起交换；

(4) 每一列中的字段值必须是同类型的数据，而且每个字段应是最基本的、不可分裂的数据项，即表中不允许再有表。

综上所述，不难看出，把一个实体联系模型转化为关系数据模型比转化为其他类型的数据模型简单，只要把实体及联系分别转化为关系模式就行了。在若干个关系模式中，通过关键字的帮助，就能查询到用户需要的各种信息。

第四节　数据库和数据库管理系统

在介绍具体的数据库管理系统之前，我们必须搞清一些基本概念之间的区别。

数据库系统(Data Base System，简称 DBS)是采用了数据库技术的计算机系统。它的含义不仅是指一个供用户存取数据的数据库，也不仅是指一组对数据库进行管理的软件，数据库系统是包括存储介质、处理对象和管理系统的集合体。它通常是由数据库、硬件、软件、数据库管理员四部分组成，其中数据库管理员必须是十分熟悉企事业单位全部数据的性质和用途，而且十分了解系统性能的、控制数据整体结构的人员，包括用户、应用程序员、系统分析员等不同级别的开发、管理、使用人员。

数据库(Data Base，简称 DB)是以一定方式组织的、相互关联的、有结构的数据构成的集合，通常由两大部分组成：一部分是实际应用所需要的工作数据的集合，称为物理数据库；另一部分是描述各级数据结构的描述数据库，通常由一个数据库字典系统管理。

数据库管理系统(Data Base Management System 简称 DBMS)是对数据进行管理的软件系统，是数据库系统的核心组成部分。其功能强弱是衡量数据库系统性能优劣的主要因素。与文件系统不同，在数据库系统中用户不能直接和存储的数据资源打交道，对数据库进行的各种操作，包括查询数据、更新数据以及各种控制，都是通过数据库管理系统实现的。数据库管理系统实际上起着一种管理作用。数据库系统的数据有较大的独立性，数据与程序相对独立，一组数据能为多个用户共享，完全改变了文件系统中数据从属于应用程序的做法。同时，由于数据库管理系统对数据的存储、管理、操作和控制提供了统一的有效手段，使用户编写应用程序变得十分简单，从而大大方便了用户。

根据所采用的数据模型的不同，数据库管理系统也可以分为层次型、网状型和关系型等几种。关系数据库管理系统具有数据结构比较简单；可以直接处理 M:N 联系；数据的独立性程度高等优点，所以被大、中、小型计算机和微机广泛应用。目前用得较多的有 ORACLE、DBASE、FOXBASE、FOXPRO 等多种关系型数据库管理系统。

数据库管理系统是一组软件的集成，它可用于定义数据库，帮助用户访问和控制数据库，保证数据的独立性、完整性和安全性。它处于用户和物理数据库之间，把数据库的物理细节屏蔽起来，提供给用户一种友善的界面。它的主要功能包括以下五个方面。

(1) 数据库的定义功能。DBMS 提供了必要的数据定义语言 DDL(Data Definition Language)，用于定义数据库的结构、定义数据完整性的约束条件、保密限制的约束条件等。

(2) 数据库的操纵功能。DBMS 还提供了一种数据操作语言 DML(Data Manipulation Language)，用于对数据库进行各种操作。它的语句分成两部分，一部分是查询语句，用于用户的各种检索操作，这部分的理论基础较复杂；另一部分是非查询语句，主要用于数据库记录的插入、修改和删除等操作。

(3) 运行控制功能。这是 DBMS 的核心功能，主要作用是包括下列四个部分：即数据的安全控制；数据的完整性控制；多用户环境下的并发控制；在数据库遇到被破坏的可能时，及时保护，恢复数据库。

数据的安全性往往是和保密性紧密联系的。安全性主要是指在数据库资源共享时，要审查用户的使用权限，以免数据的泄密或任意更改，甚至破坏。例如一个市的教育系统有一个数据库系统，计财处可以查询各级教师的姓名、工资等情况；人事处也可以查询教师姓名、任教学科、职称、工资等基本情况，并且在工资调整时，还可以修改教师的职称、工资情况，但这个权限就不属于计财处。因此 DBMS 必须严格控制每一用户的存取权，即对用户进行合法性检查，检查的方法可采用向机器提供口令或密码等方式，由系统进行验证。

数据的完整性是指运行过程中，DBMS 能对数据进行完整性检查。在修改数据时，有时会因误操作或系统故障等原因，引起非法更新，此时 DBMS 会采取恰当的动作（如拒绝操作、报告错误信息等）来处理这些问题。DBMS 还能根据约束条件进行管理，如对数据取值的类型、范围、精度等数据值的约束条件；对数据之间联系的约束条件。DBMS 还提供了定义和检查完整性约束条件的功能，以此来保证运行过程中数据的正确性和一致性。

并发操作是指多个用户并发地对同一数据库进行操作。此时，数据的完整性可能会遭到破坏，也可能会产生不正确的数据（称为脏数据）。例如，人事处和计财处同时用教师基本情况数据库中的某一个教师的数据，人事处需修改教师的职称，计财处需修改他的工资总额，各部门修改完后相继写回数据库，结果得到的却是脏数据。因为在职称和工资两个字段中，只是改写了一个字段。这种情况就是由于并发操作引起的。并发控制就是要正确地调度并发操作，这是 DBMS 控制功能的一个重要方面。

DBMS 对数据库的另一个控制功能是当数据库被破坏时，及时地把数据库恢复到以前的某一个正确状态。在对数据库进行操作时，可能因电源故障、硬件错误、软件错误、操作错误和其他意外而破坏数据库。此时，DBMS 就能使正在执行的应用程序退回，或是利用自身的恢复子系统，把数据库恢复到某一正常状态。

(4) 数据库的建立和维护功能。这些功能可由用户自己编写的应用程序去完成。

(5) 含有数据字典（Data Dictionary 简称为 DD）。DD 是 DBMS 的重要组成部分，在进行数据库的每一种操作时，系统都要通过查阅数据字典才能进行。

第五节　关系型数据库管理软件 FOXBASE 简介

一、FOXBASE 简介

FOXBASE PLUS（通常也称为 FOXBASE 或 FOXBASE +，本书中简称为 FOXBASE）是美国 Fox Software 公司 1987 年推出的关系数据库管理系统。它在 286、386 及其兼容机等个人微机上，在 PC-DOS 等操作系统支持下可作为单用户管理系统使用，在 VAX 等中、小型机上，在 UNIX、XENIX 等操作系统支持下可作为多用户管理系统，还可以在网络软件支持下工作。它与目前国内外较为流行的 DBASEⅢ、DBASEⅢPLUS 等关系数据库管理系统完全兼容，也就是说在 DBASE 数据库管理系统下编写的程序无需改动就可在 FOXBASE 系统下运行。FOXBASE 不仅包括了 DBASEⅢ的命令和功能，而且还有两个明显的优点：首先是 FOXBASE

的运行速度比 DBASEⅢ快 6～7 倍，其次是比 DBASEⅢ增加了几十条命令和函数，使其在性能和功能上得到了扩充和发展。这些功能包括内存变量数组、自定义函数、数据编辑中的检验功能、多个数据工作区的同时操作等。

二、FOXBASE 的系统运行环境

硬件配置：IBM/AT 以上档次的微机（包括兼容机）、中、小型计算机都可使用 FOXBASE 数据库管理系统，对单用户系统要求内存空间在 375KB 以上；对多用户系统，要求内存在 1.5MB 以上。每个单用户系统，至少应带有两个软盘驱动器或一个硬盘驱动器加一个软盘驱动器。此外，还应配一台能打印汉字的宽行打印机。

软件配置：单用户系统应使用 PC-DOS、CCDOS2.0 以上版本的磁盘操作系统，多用户系统可使用 3.1 以上版本的 XENIX 操作系统。

三、FOXBASE 的系统文件

常用的中西文 FOXBASE 数据库管理系统 2.0 版是由西文版 FOXBASE 的多用户版本 MFOXBASE2.0 汉化开发而成的，系统文件主要包括以下内容：

MFOXPLUS.EXE	执行程序
MFOXPLUS.OVL	覆盖程序
FOXPCOMP.EXE	过程组合程序
FOXPHELP.HLP	帮助文件
FOXBIND.EXE	准编译程序
EUROPEAN.MEM	欧洲文字排序基准文件

执行程序（MFOXPLUS.EXE）和覆盖程序（MFOXPLUS.OVL）是 FOXBASE 的主要部分。启动 FOXBASE 后，执行程序常驻内存（约占 242KB），覆盖程序只有一部分装入内存，在需要时，可用覆盖方式再调入另一部分。其余四个文件是辅助部分。帮助文件向用户提供了语法和用法的说明；过程组合程序可把若干个命令文件组合成一个命令文件（也称为过程文件），每个命令文件分别调试成功后组成一个过程文件，它不仅能完成各个命令文件的功能，而且能提高运行效率，也比管理多个文件方便得多；准编译程序用于编译用户的应用程序，产生后缀名为.FOX 的文件，有利于提高程序的装入速度和运行速度，并可先检查出程序中的语法错误，还可为用户应用程序保密；欧洲文字排序基准文件用于解决法、德、意等欧洲各国字母排序问题。本书因限于篇幅，对此不作详细介绍。

本书前面七章介绍的命令都以 FOXBASE2.0 版为准，在第八章的部分实例中使用了 FOXBASE2.1 版的命令。

四、FOXBASE 的主要技术指标

数据库文件的记录个数——最多 10 亿个
数据库文件能容纳的字节数——最多 20 亿个
一个记录的字段数——最多 128 个
一个记录的字符数——最多 4000 个字节
数值型字段宽度——最多 19 个字节

字符型字段宽度——最多 254 个字节
日期型字段宽度——8 个字节
逻辑型字段宽度——1 个字节
备注型字段宽度——最多 64K 字节
数字计算中的有效位数——16 位
可用内存变量的个数——默认为 254 个,最多为 3600 个
每个数组中元素的个数——最多为 3600 个
每个命令行的长度——最多为 254 个字符
可同时打开数据库文件的个数——最多为 10 个
可同时打开索引文件的个数——对于每个数据库文件可同时打开 7 个索引文件,一共打开的索引文件不超过 21 个
可同时打开的文件个数——默认为 16 个,最多 48 个
每个过程文件中包含的子过程个数——最多 128 个

五、系统的安装、启动和退出

如果用户希望在硬盘上用 FOXBASE,则可在硬盘上建立一个子目录,并给它一个名字(例如 FOX)具体方法如下(假定在硬盘 C 上建立子目录):

C: > MD FOX〈回车〉

C: > CD FOX〈回车〉

此时,把含有 FOXBASE 系统文件的软盘放入 A 驱动器(若用两张盘存放系统文件的可一个个磁盘相继插入 A 驱动器)并输入如下命令:

C: > COPY A: * . *〈回车〉

(若有两张系统盘,则需执行两次)

C: > CD \〈回车〉

为了使今后用户无论在哪个驱动器或子目录上都可使用 FOXBASE,用户可在硬盘中的 AUTOEXEC.BAT 文件中增加一条命令,即 PATH C:FOX。

在 CCDOS 操作系统支持下,如果 FOXBASE 已在当前工作盘上,或者 FOXBASE 虽在硬盘 C 中,但已设置过路径,则只需在当前工作盘上键入如下命令:

MFOXPLUS〈回车〉

系统启动后,屏幕上即出现圆点提示符".",表示已进入人机交互的状态,等待用户键入命令,对数据库进行有关的操作。

在圆点提示符状态下,键入 QUIT 命令即可实现系统的退出。

QUIT〈回车〉

执行 QUIT 命令时,系统首先把缓冲区中的数据嵌入相应的文件中去,然后关闭所有文件,并退出 FOXBASE,回到 DOS 操作系统中去。这种退出方式称为正常退出。如果因突然掉电或死机等原因而导致非正常退出,则缓冲区中数据将丢失,不仅会导致当前正在处理的数据无效,而且可能破坏数据库文件。因此,用户使用数据库时必须及时存盘并有备份。

习　题

1.数据库管理技术的发展经历了哪三个阶段?
2.关系模型必须具备哪些基本性质?
3.数据库系统和数据库管理系统有哪些区别?
4.练习进入和退出 FOXBASE 系统的方法。

第二章　FOXBASE 的基本概念

FOXBASE 系统是一种数据库管理系统，同其他任何一种高级语言系统一样，也有常量、变量、函数、表达式等基本概念。

第一节　数 据 类 型

在 FOXBASE 系统中，用户使用的数据共有以下五种类型。

字符型数据：用 C(Character) 表示，由西文字符、汉字、数字及其他符号组成，最大长度为 254。

数值型数据：一种可以进行算术运算的数据，用 N(Numeric) 表示，由数字、小数点、正负号组成。数值型数据可用数据中含有 E 的科学计数法表示，E 后跟一整数，表示 10 的幂，例如 2.34E5、2E－3 分别表示 234000、0.002。

逻辑型数据：用来表示逻辑判断的结果，用 L(Logical) 表示。它只有两个值：真(.T.)和假(.F.)。

日期型数据：以特定格式表示日期的一种数据，用 D(Date) 表示。系统默认的日期型数据格式为：月/日/年，共占 8 个字符位置，其中年、月、日各占两位(通常用格式 MM/DD/YY 表示)，用来处理本世纪日期，例如：09/12/92 表示 1992 年 9 月 12 日。若需用到跨世纪日期，应用四位数表示年份，如 10/01/2010，此时 SET CENTURY ON/OFF 应设置为 ON(该命令默认设置为 OFF)。

记忆型数据：用 M(Memo)表示，是一种只能在数据库文件中使用的特殊的字符型数据。

第二节　常量、变量

一、常量

常量就是在程序运行过程中不变的量，也叫常数。常量类型有数值型、字符型、逻辑型、日期型四种。

数值型常量：如 3.14、－25、3.34E7 等。

字符型常量：用定界符括起来的字符串，由中英文字符、数字、空格及其他专用字符组成，定界符为' '、" "、[ ]，如 "姓　名"，"ABCD"、'12345'、[07－012D]等。如果一种定界符已成为字符串的组成部分，则应选用另一种定界符来标识字符串，如 "He said: 'I am a student' "、'初二[一]班'是合法的字符串，而"He said:"I am a student" "、[初二[一]班]是非法的。

逻辑型常量：只有“真”或“假”两个逻辑值，通常用“.Y.”或“.T.”表示逻辑真，“.N.”或“.F.”表示逻辑假。

日期型常量：如 02/10/93 表示 1993 年 2 月 10 号。

二、变量

变量就是在程序运行过程中其值可能发生变化的量。FOXBASE 中提供了内存变量和字段变量两种变量类型。

1.字段变量

字段变量是指数据库结构中已定义的任一字段,是构成数据库文件的基本数据单元。字段在不同时刻取值不同。数据库文件有一指针,指针指向的记录定义为当前记录,字段变量的值就是当前记录中对应字段的值;因为指针是可以移动的,因此字段的取值也随指针的移动而改变,所以字段也是变量。有关字段变量的详细概念将在第三章讨论。

2.内存变量

数学中常用 X、Y 等来表示未知量,FOXBASE 中也可用类似于 X、Y 的变量来存放常数和一些临时性中间结果,这些变量称为内存变量。

为了使用内存变量,必须给每个变量取名。变量取名规则为:

(1) 变量名必须以字母、汉字、下划线及数字符组成,且不能由数字开头。

(2) 变量名长度不超过 10 个字符(一个汉字相当于两个字符)。

如:X、AB、姓名、Y1、T11_1 等为合法变量名,而姓　名、11Y、N $ 、BADCOMMAND11 等为非法变量名。

内存变量是独立于数据库的变量。它是一种临时性的工作单元,需要时可以随时定义,不用时可以释放掉。系统通过内存变量名访问内存变量。若内存变量名与当前工作的数据库文件字段变量名相同,字段变量名优先于内存变量名,取字段变量的现值。为了避免这种情况,可在内存变量名前加"M->"以示区别。

内存变量的类型可为:数值型(N)、字符型(C)、逻辑型(L)、日期型(D),内存变量的命名、数据类型和宽度的确定是在内存变量赋值时一起实现的。

第三节　FOXBASE 简单命令介绍

为了方便此后例题的介绍,本节先简要介绍 FOXBASE 中几种常用的简单命令。

一、命令行尾注释命令

格式:&& [〈注释信息〉]

功能:用来给某条命令加注释。

说明:

(1) 命令中,方括号[ ]中的内容是任选项,可有可无,视用户需要而定;尖括号〈 〉中的内容是必选项,其内容由用户自行输入。输入时,方括号及尖括号不必输入。

(2)〈注释信息〉是由任何字符组成的字符串。

(3) 该命令可以作为独立行出现,也可以出现在一条命令的尾部。

二、给内存变量赋值命令

格式:〈内存变量名〉=〈表达式〉

功能:先计算〈表达式〉的值,然后将〈表达式〉的值及类型赋给〈内存变量〉。

若执行此赋值命令前,该内存变量已存在,执行此命令后,〈表达式〉的值将覆盖〈内存变量〉中原来的内容;若内存变量不存在,执行此命令后,将产生新的内存变量。

例:

```
. A = 19 * 2 + 3          && 此赋值语句确定了内存变量 A 的类型为数值型,值为 41
. Y1 = 'ABCD'
. A = '01 - 1263'         && 变量 A 的类型变为字符型,值为"01 - 1263"
. 年龄 = 20
. Y1 = .T.
. 姓名 = '张三'
```

三、屏幕显示输出命令

格式:? [〈表达式表〉]

功能:在屏幕上输出一个或多个表达式的值。

例:

```
. A = 19 * 2 + 3
. ? A
    41                    && 41 为屏幕输出内容
. Y1 = 'ABCD'
. ? Y1
ABCD
. ? 12 + 4,2 * 4 - 9      && 同时输出两个表达式的值,表达式之间用",”分隔
 16    - 1
```

第四节　函　　数

FOXBASE 系统提供了近 80 个各种类型的标准函数,函数中的自变量称为参数;当用户使用时,只需提供合适的参数就可调用函数。本章只介绍一些常用的函数。

一、日期型函数

日期型函数是指函数的返回值是日期型数据,可以参加日期运算。

1.系统日期函数

格式:DATE()

功能:给出 FOXBASE 系统日期,系统默认日期格式为:"月/日/年"。

例:

```
. ? DATE()
05/13/94
```

2.字符转化为日期函数

格式:CTOD(〈字符型表达式〉)

功能:将字符串转化成日期。当字符串格式和日期型数据格式一致时函数值为相应日期,否则函数值为空日期。

例:

```
. ? CTOD('01/01/94')
01/01/94
. ? CTOD('34/32/98')          && 月、日的值超出范围,函数值为空日期
  / /
. ? CTOD('ASSDSD')            && 字符数据格式不对,函数值为空日期
  / /
```

二、数值型函数

数值型函数是指函数的返回值为数值型数据,可以参加数值运算。

1. 取整函数

格式:INT(〈数值型表达式〉)

功能:取〈数值型表达式〉值的整数部分。

例:

```
. ? INT(2.24)
    2
. X = 6.7
. ? INT(X)
        6
. ? INT(X + 5.6)
       12
```

2. 四舍五入函数

格式:ROUND(〈数值型表达式 1〉,〈数值型表达式 2〉)

功能:对〈数值型表达式 1〉的值进行四舍五入。四舍五入位的位置由〈数值型表达式 2〉的值确定。设表达式 2 值的整数部分为 N,当 N > = 0 时,表达式 1 在小数点后第 N + 1 位进行四舍五入;若 N < 0,则在表达式 1 的整数部分从右向左第 N 位进行四舍五入,整数部分末尾含 N 个 0。四舍五入后小数部分位数不变。

例:

```
. ? ROUND(12.3456,3)
 12.3460
. X = 1234.667
. ? ROUND(X,0)
     1235.000
. ? ROUND(X, - 1)
     1230.000
```

3. 取绝对值函数

格式:ABS(〈数值型表达式〉)

功能:取〈数值型表达式〉值的绝对值。

例:

```
. X = 5
. ? ABS(X)
          5
. ? ABS(X - 6)
          1
```

4. 平方根函数

格式:SQRT(〈数值型表达式〉)

功能:求〈数值型表达式〉值的平方根(要求〈数值型表达式〉的值大于或等于 0)。该函数值至少保留两位小数;若〈数值型表达式〉值的小数位数超过两位,函数值的小数位数和〈数值型表达式〉值的小数位数相同。

例:

```
. ? SQRT(49)
 7.00
. ? SQRT(49.000000)
 7.000000
```

5. 指数函数

格式:EXP(〈数值型表达式〉)

功能:求以自然数 e 为底,〈数值型表达式〉值为指数的函数值。

例:

```
. Y = 2.34
. ? EXP(Y)
          10.38
. Y = 2.3423
. ? EXP(Y)
          10.4051
```

6. 自然对数函数

格式:LOG(〈数值型表达式〉)

功能:取〈数值型表达式〉的自然对数。要求〈数值型表达式〉的值大于 0。该函数是 EXP 函数的逆运算。

例:

```
. ? LOG(1.23345)
0.20982
```

7. 较大值函数

格式:MAX(〈数值型表达式 1〉,〈数值型表达式 2〉)

功能:取表达式 1 和表达式 2 中的较大者。

例:

```
. X = 10
```

. Y = 20
. ? MAX(X,Y)
20
8. 较小值函数
格式:MIN(〈数值型表达式 1〉,〈数值型表达式 2〉)
功能:取表达式 1 和表达式 2 中的较小者。
例:
. X = 10
. Y = 20
. ? MIN(X,Y)
10
9. ASCII 码值函数
格式:ASC(〈字符型表达式〉)
功能:取〈字符型表达式〉值中左边首字符的 ASCII 码值。
例:
. X = 'ABCD'
. ? ASC(X)
65 && 字符"A"的 ASCII 码值为 65
. Y = 'aBCD'
. ? ASC(Y)
97 && 字符"a"的 ASCII 码值为 97
10.字符型数据转化为数值型数据函数
格式:VAL(〈字符型表达式〉)
功能:将〈字符型表达式〉的值转化为数值型数据。系统默认转换后所得数值型数据最多保留两位小数。若〈字符型表达式〉的小数位数多于 2 位,在第 3 位上四舍五入。
说明:
(1) 组成〈字符型表达式〉的合法字符为正负号、小数点、数字符号及 E。
(2) 若表达式第一个字符为非法字符,则函数值为 0。
(3) 若表达式中包含非法字符,则函数值为〈字符型表达式〉中从左至右第一个非法字符左边的子字符串的值。
例:
. X = '453'
. ? VAL(X)
453.00
. X = '1234.546'
. ? VAL(X)
1234.55
. Y = 'A234'
. ? VAL(Y)