第一章　数据库简介

1.1　概　述

本书介绍了一种非常高效的数据库管理系统软件，FoxPro for Windows，读者将会学到怎样使用它去开发用户自己的数据库管理系统，利用它去有效管理用户数据。但在讨论设计和开发这样的系统之前，读者有必要先学习一些数据库管理系统的基本概念和术语。这也就是本章的主题。

简而言之，数据库就是一个内部相关的数据元素的集合，其中元素按某种逻辑方式进行组织和安排，这样就能够很容易地访问有用信息。然而，可访问信息的类型以及用户检索数据库的速度在很大的程度上取决于用户存储和管理数据元素的方式。

用于组织数据元素的数据模型有很多种，但由于 FoxPro for Windows 使用的是关系模型，本章将只讨论这种模型的概念和结构。正如读者将在后面所见，这种结构的骨架是数据元素之间的相互关系，理解在数据库中如何定义这些关系对于用户系统的效能是至关重要的。本章中将通过例子介绍怎样设计一个具有正确结构的数据库。

1.2　数据元素

数据元素（data element）是数据库管理系统中信息的基本单元。它通常由字符组成，字符可以是字母（A－Z，a－z）、数字（1－9），或文本符号（逗号、句号、冒号等）。例如，一个雇员名 Smith 就是由一组字母构成，它就可以是数据库系统中的一个数据元素。这种数据元素的集合就构成了数据库的成员，例如，一个雇员的名和姓就组成了数据库的成员。可以给数据成员指定一个特殊的名字，比如数据字段、数据记录或数据文件。这些术语将在后面介绍。

1.3　数据库

如前所述，数据库（database）是数据元素按某种逻辑方式和结构组织起来的集合。其实际结构取决于用户组织元素的数据模型，大小由数据元素的数目决定。数据元素之间的关系影响了数据库的复杂程度。

一个简单的数据库通常具有很少数目的数据元素，可以按简单的结构将它们组织起来，例如，读者所有朋友的姓名和电话号码的集合就可以表示成一个小而完整的数据库，在其中存储的名、姓、地名代码、电话号码形成数据元素。

另外，一个包含很多数据元素的公司销售数据库将按稍复杂一些的结构进行组织。这样数据库的数据元素将和顾客、库存、订单和销售分支的情况有关。因此，和顾客有关的数据元素将存储在一组中，在此组中，和一个顾客相关的数据存储在单独的数据元素中；通常，这些项目包括名、地址、某一顾客的电话号码。只和货物相关的数据元素将存储在库存组中；每次完成一项交易时，有关销售和库存的项目还将被存储在其他数据元素组中。只要这些数据元素按一定逻辑方式组织和存储，用户就能够很容易地从这些组中取得所需的数据元素以生成库存清单或报表。

1.4 数据库管理系统

数据库管理系统（DBMS）提供了管理数据元素的系统化方法，以使提取和分配信息都可以轻易完成。除了提供信息之外，一个正确设计和结构化的数据库管理系统使用户能够有效管理数据元素，诸如向数据库中增加数据元素、修改和改变已有的数据元素或删除不再需要的数据元素。系统的速度应该是很快的，用户应该能够很快地寻找到数据元素；一旦找到了信息，用户可以按适当格式显示或提供结果。

尽管我们论及的系统可能很复杂以至需要计算机来处理，数据库管理系统有时并不是很复杂的，也并不一定要借助于计算机。一个数据库管理系统可以是人工系统，例如我们可以用一组卡片来存储和管理所有朋友的有关信息。这个卡片文件由一些每一个都记录着一组数据元素的卡片构成，这也可以称为数据库。

显然，本书所讨论的是使用 FoxPro for Windows 的计算机数据库管理系统。尽管用户是借助于计算机完成所有的数据管理功能，但遵循的规律和人工系统应该是一样的。如果读者已经对人工管理系统有一个较清晰的概念，那么就已具有一个好的基础来学习计算机化数据库管理系统的基本功能。

1.5 数据及信息

在数据库管理系统中，数据元素是信息系统的组成要素。这些数据元素本身通常并不提供任何有意义和有用的信息。必须把这些根据某一主题相关的数据元素连接起来才有意义。因此，数据和信息在本质上是两码事。

数据库管理系统的主要作用是组织数据元素，以使用户通过检索所有相关的数据元素来获取需要的信息。例如，若要存储一个朋友的地址，可以分为几个部分分别存储他（她）的街道名称，城市和国家，以及邮政区码。有关这位朋友的邮政地址可以通过从正确的数据库中检索数据元素来取得。再如，雇员名是一个由数个字母组成的数据元素。一个公司中所有雇员名组成了此公司人事数据库的一个子集。

1.6 组织数据

用户选择的组织数据元素的方式是数据库管理系统的主要问题。用户数据库管理系统

的效率和作用主要由数据库中存储和管理数据元素的结构所决定。一个数据库如果结构合理，在支持和管理数据库时能够节省很多力气和时间。

由于有用信息通常需要在数据库中连接数据元素，定义数据元素之间的关系就起到很重要的作用。因此，在学习怎样设计数据库管理系统之前，读者先要了解如何确立数据库中所有数据元素之间的关系。

1.6.1 数据关系

如前所述，数据元素是数据库的基本单位。它描述了数据实体在某一方面的性质。一个数据实体是一个单独对象，可以是一个人、表格、分支代理处或销售区。

观察数据库的结构可以看到，数据实体是构造结构的砖石。例如，要设计一个数据库系统管理销售机构的活动和功能，就必须包含有关销售员、销售办事处、销售区等内容的销售实体。

每一个数据实体都具有一组特性。例如，一个销售员具有以下特性：姓、名、生日、薪水、地址等。销售办事处则具有另外一组特性诸如地址和电话号码。通常的数据库设计允许用户按三种基本方式之一观察数据实体之间的连接：一对一关系，一对多关系，多对多关系。

1.6.2 一对一关系

一对一关系是两个数据实体之间最简单的连接方式。要理解这种关系，我们假定在销售组织采用以下规则：每一个销售员只分配到一个销售办事处；每一个销售办事处只分配一个人。

例如，有三个销售办事处：B1，B2 和 B3。有三个销售员分配到这些办事处中，它们是：Anderson，Bell 和 Carter。这样数据结构中就具有了两个数据实体：销售职员和办事处。它们之间的连接所处的一对一关系如图 1.1 所示。

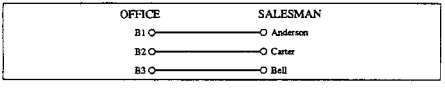

图 1.1 一对一关系

1.6.3 一对多关系

显然，一对一关系太简单，并不反映实际中销售员和办事处实体之间的真实连接。最大可能的情况下每个办事处包括多个销售员。因此，一对一关系不能充分描述两个数据实体之间的连接。表示销售员和办事处这两个实体之间关系的更实际的例子见图 1.2。

当然，通过转换一对多关系的两个实体的顺序，就形成了多对一的关系。因此，一对多关系和多对一关系在数据库设计中是相同的。

1.6.4 多对多关系

在数据库设计中，多对多关系是连接两个数据实体的最复杂的关系。一个数据实体中的每个对象被连接到另一数据实体的几个对象，反过来也是这样。要演示多对多的关系，假定有10个销售员（如图1.3中所示），其中每一个都被分配到六个销售区中的一个或多个。这些销售区分别命名为R1、R2、R3、R4、R5和R6。图1.3显示了连接销售员和销售区域的多对多关系。

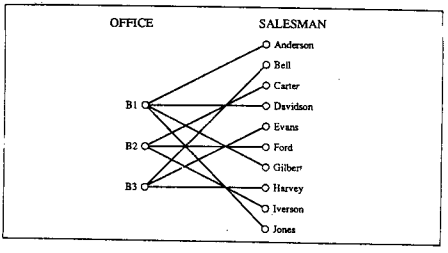

图1.2 一对多关系

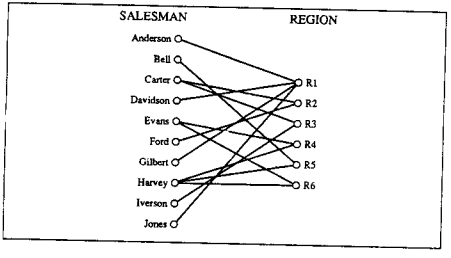

图1.3 多对多关系

在图 1.3 中可以看到，每个销售员可能被分配到多于一个的销售区域，每个销售区可能分配到多个销售员。因此，一个能够容纳多对多关系的数据结构应该能够处理一对多、多对一以及一对一的关系。但反过来就不是这样。若数据库设计只能处理一对一关系，就不能够管理一对多和多对多关系。用户在设计一个可用的数据库结构之前，必须理解数据元素之间的实质联系，以正确运用这些关系。

1.7 数据模型

有一些数据模型已被开发用于管理数据库。其中三种最常见的是分层、网络和关系数据模型。尽管本书的着重点是关系型数据模型，对其他模型的简要介绍将有助于读者理解关系型模型的用途和效能。此时，有关这些模型的知识对于设计数据库也能够起到帮助。

1.7.1 分层数据模型

分层数据模型按照树型结构来观察数据实体之间的关系。使用分层数据模型来观察前面例子中三个数据实体（Office、Salesman，Region）之间的关系，如图 1.4 所示。

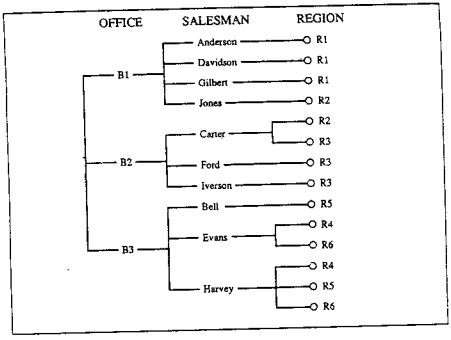

图 1.4 分层数据模型

从图中可以看到，模型假定了三级：主干和两级分支。第一级描述了 Office，其中每一

支都表示了一个实体对象并引向不同的子分支。每一个子分支又表示下一级数据实体中的一个对象。这样进行下去，每一个销售员子分支又导向另一层子分支，它表示的是 Region 对象。

可以看出，所有的数据元素严格按一种逻辑方式组织。例如：数据实体中的每个对象都和另一个数据实体中的一个或多个对象逻辑相关。读者能够找到每一个数据实体中的对象和它们之间的关系。但仔细研究一下这种模型可以发现，分层模型只允许一对一、一对多和多对一关系。要包含多对多关系，必须被分解为重复的一对多关系。

在图 1.3 中可以看到，多对多关系存在于销售员和区域数据实体中的对象之间。但由图 1.4 可见，这些多对多关系必须被定成重复的一对多关系。注意在树形结构中一个分支可以引向多个子分支，但所有的子分支通常引向一个上一层的分支。为符合这种结构，对于多对多关系必须重复子分支中的一些对象，以分解为多个一对多关系。

同时可以注意到，Region 中的一些对象出现在多个子分支中以描述和 Salesman 对象的关系，例如 R4 就在子分支中出现了两次，分别和 Salesman 中的两个对象相关——Evans 和 Harvey。对这种重复作出调整就能够开发出另一种数据模型——网络模型。

1.7.2 网络数据模型

网络数据模型可以视为分层模型的变体。如同分层模型，它也按照树型结构来组织数据元素；不同的是，网络模型能够在树形结构中定义多对多关系，而不用重复数据实体中的对象。图 1.5 中给出了相同的三个数据实体之间在网络模型下的关系。

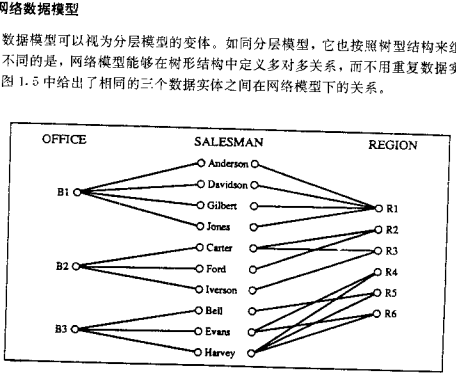

图 1.5 网络数据模型

注意数据模型与图 1.4 中所示的非常相似：它们都使用树形结构表示三个数据实体之间的关系。由于在办事处和销售员实体之间只存在一对多关系，分层和网络模型描述关系使用的方式是相同的。但如何表示销售员和销售区之间的多对多关系，两种模型则是不同

存储代码片断和定义接口对象及其属性。当我们存储一个新的屏幕，FoxPro 建立一个包含字段如 OBJTYPE、OBJCODE、NAME、VPOS、HPOS、COLORPAIR、VALID、FLOAT、CLOSE 等的. SCX 文件。每个屏幕对象都有自己的记录。

当我们产生代码时，FoxPro 运行 Screen Builder 的模板程序 GENSCRN. PRG。GENSCRN. PRG 利用 FoxPro 的文本连接功能，在 FoxPro 的命令下，将. SCX 和. SCT 文件中的数据和代码片断组合在一起。结果是 Screen Builder 中定义的属性和代码组合成一个 FoxPro 程序。

例如，我们可以应用 Screen Builder 建立一个名为 MAINSC 的屏幕，包括一个框和一个名为 mvar 的单个 GET。存储完定义以后，我们可以使用 USE 命令打开文件 MAINSC. SCX，并显示其内容。其中，我们发现 79 个字段和 3 个记录。第一个记录包含基本的屏幕设置数据。第二个记录包含框的定义，第三个包含 GET 的定义。下表仅显示了 79 个字段中的 10 个：

	记录 1	记录 2	记录 3
PLATFORM	DOS	DOS	DOS
UNIQUEID	_QCY1BSZ7V	QCY1BT1GW	_QCY1BT8UQ
TIMESTAMP	440382022	440382024	440382030
OBJTYPE	1	7	15
OBJCODE	63	4	1
NAME	memo	memo	Memo (contains mvar)
VPOS	0.000	0.000	8.000
HPOS	0.000	0.000	7.000
HEIGHT	25.000	7.000	1.000
WIDTH	80.000	8.000	10.000

我们通过检验记录 2 框的定义来说明代码的生成。注意小写字母 memo 标记表示备注字段为空。OBJTYPE 字段为 7，表示是框；OBJCODE 为 4，表明为单线边界。VPOS 和 HPOS 都为 0，表示框起始于交点(0,0)。同样，HEIGHT 是 7.000，WIDTH 为 8.000。PLATFORM 字段表示对象是用 FoxPro for MS-DOS 建立的，UNIQUEID 和 TIMESTAMP 字段被 FoxPro 的转换器用以帮助管理跨平台开发。

当我们为这个屏幕生成代码时，一个 GENSCRN. PRG 中名为 GENBOXES 的过程处理 OBJTYDE、OBJCODE、VPOS、HPOS、HEIGHT 和 WIDTH 字段并生成画该框的代码。

过程 GENBOXES 中，下面这段程序说明了 FoxPro 的文本合并命令\和\\的用法。每一行中，由\或\\开始，FoxPro 求（（ ））内的表达式的值，然后把它们和其他字符文本一起放到输出文件中去。命令\对输出内容增加一个回车和换行指令，命令\\继续在输出的同一行中增加内容，没有回车和换行。

```
\@( (Vpos)), ( (Hpos)), ( (m.bottom)), ( (m.right))
DO CASE
CASE objcode = c_sgbox
  m.thisbox = c_single
  \\BOX " ( (m.thisbox)) ( (Fillchar))"
CASE objcode = c_sgboxd
```

分别称为 Office 和 Region，如图 1.7 所示。

Table: Salesman

Salesman ID	Last Name	First Name	Hire Date	Salary
SD	Anderson	Doris	07/01/86	2,800
S1	Bell	George	10/12/88	2,400
S2	Carter	Jack	05/14/87	2,550
S3	Davidson	Edward	06/04/90	1,500
S4	Evans	Henry	03/08/88	2,000
S5	Ford	Ida	11/22/87	2,600
S6	Gilbert	Fred	04/15/87	2,300
S7	Harvey	Candy	12/01/89	2,450
S8	Iverson	Albert	10/25/88	2,200
S9	Jones	Betty	09/26/89	2,500

图 1.6　使用关系模型的数据表

Table: Office

Office ID	Address	City	State	Zip	Phone #
B1	100 Park Avenue	New York	NY	10016	800-123-5555
B2	200 Lake Drive	Chicago	IL	60607	800-234-5555
B3	500 Century Blvd.	Los Angeles	CA	94005	800-456-5555

Table: Region

Region ID	Region	Manager
R1	Northeast	Alice F. Gibson
R2	Southeast	Bob L. Major
R3	Northcentral	John K. Freed
R4	Southcentral	Cathy M. Wilson
R5	Northwest	Chris C. Hall
R6	Southwest	Helen T. Taylor

图 1.7　Office 和 Region 表

从图 1.7 中可以看出，Office 和 Region 表与 Salesman 表的格式相同，它们都被分隔为数行和数列。在 Office 表中，描述办事处特征的数据元素——标识号、地址以及电话号码被存储在列中；每一行都存储了和某一办事处有关的数据元素。在 Region 表中，有关区域的特征存储在列中，例如 Manager 列就存储了区域经理名。

在关系数据模型中，数据库通常包含多于一个的数据表，这些数据表常常称为数据库文件。表的每一行称作数据记录，它包含一组数据字段，每个字段对应于数据表的一列。因

此，关系型数据库中的数据元素被组织为数据库文件（表）。每个数据库文件包括许多记录（行），每个记录又含有一些字段（列）。

使用这些关系型数据库的术语，我们可以按如下方式描述包括 Salesman、Office 和 Region 的数据库：

表名：Salesman
记录数：10
数据字段：Salesman ID
Last Name
First Name
Hire Date
Salary

表名：Office ID
记录数：3
数据字段：ID #
Address
City
State
Zip
Phone #

表名：Region ID
记录表：6
数据字段：ID #
Region
Manager

图 1.8 中给出了关系表的一个例子，名为 Assignment，它的结构与上面三个数据表是很相似的。差别在于 Salesman 表中的每一行描述了给定销售员的特征，而 Assignment 表中的每一行指定了销售区和销售区之间的联系。

关系表在决定关系型数据库的用途和功效中起到很重要的作用。正确定制了结构后，用户能够处理数据实体之间存在的任何关系。下一章中还将详细讨论关系数据表。

1.8 设计关系型数据库

数据库的用途和效能主要取决于如何建立数据表以描述数据实体及其关系。一个正确定制了结构的数据表使用户能够通过检索相关的数据元素来很轻易地得到所需信息。结构若定制不合适，管理数据库就会变得很困难，甚至不可能。所以，花费一些时间去认真计划怎样建立数据表并建立连接是很必要的。

Table: Assignment

Region ID	Salesman ID
R1	S0
R1	S3
R1	S6
R1	S9
R2	S2
R2	S5
R3	S2
R3	S8
R4	S4
R4	S7
R5	S1
R5	S7
R6	S4
R6	S7

图 1.8　关系表

设计关系型数据库时，首先认真研究一下问题的实质以及数据元素的类型。预料一下可能需要的信息的类型有助于决定存储在数据库中的相关数据元素。例如，我们需要一个关于给定销售区中雇员薪水情况的详细报表。对此，可能需要描述薪水及所属区域的数据元素。再如，要生成一个库存清单，就必须有描述每一笔交易的数据元素，诸如订单号、数量等。

决定了所需的数据元素之后，下一步是按某种逻辑方式组织它们。为此，尽管使与一个实体相关的元素分组到一个数据表中。例如，在 Salesman 表中包括所有描述销售员特征的数据元素。每一种特征用数据字段中的数据元素进行描述。

建立数据表用于存储和数据实体相关的数据元素之后，应该建立关系表以提供不同数据表中数据元素之间的连接。关系数据表的数目取决于数据表中关系的复杂程度。通常，用户不必建立关系表以处理一对一或一对多的关系；只有处理多对多关系时才需要这样做。后面还将详细讨论这个问题。

1.8.1　关系数据库的特征

设计数据库可能是一个复杂的过程，特别是在包括大量多对多关系的数据表时。但许多很有用途的数据库只包括少量的数据表，处理连接的数据表也很少。读者若能理解在小型关系数据库中所应用的规则，对大型复杂数据库的处理仅仅是一个积累经验的事情。

无论关系型数据库的大小和复杂程度如何，都必须具有一组特征。若要避免多余的数据元素并正确连接用户的数据元素，就必须实现这些特征。这些必需的特征可归纳如下：

· 所有的数据元素必须被组织到数据表中。

- 单个数据表中的元素必须具有一对一关系。
- 数据表中不能有重复的行。
- 数据表中不能有重复的列。
- 每个数据单元中只能存储一个数据元素。

在表中组织数据

关系型数据库的第一个必要的特征是和特殊数据实体相关的数据元素必须被组织到表中，如同前面的例子所示。数据表是关系型数据库的基本单位，每个表可以包括任意数目的行（记录）和列。行、列的数目是无关紧要的。

无重复的行

在结构正确的关系型数据表中，不能具有数据元素相同的两行。重复行不仅浪费宝贵的存储空间，而且降低了处理速度。更重要的是，重复行会造成错误（不准确的行数）。例如，若在 Salesman 表中具有两行和多行，其中的数据元素都和同一销售员有关，在计算行数以统计销售员人数或统计工资总额时，就会出现错误。

使用单值数据单元

在数据表中，行和列的交汇处常常称为数据单元。关系型数据库的另一个基本特征是在一个数据单元中只能存储一个数据元素。例如，我们在 Salesman 表中增加 Phone No 列（见图 1.9），表中，行和列交汇处的数据单元只能存储表示电话号码的一个值。

Table: Salesman

Salesman ID	Last Name	First Name	Phone No	Hire Date	Salary
S0	Anderson	Doris	123-4567	07/01/86	2,800
S1	Bell	George	234-3456	10/12/85	2 400
S2	Carter	Jack	456-9023	05/14/87	2,550
S3	Davidson	Edward	234-5645	06/04/90	1,500
S4	Evans	Henry	635-2345	03/08/88	2,000
S5	Ford	Ida	345-2345	11/22/87	2,600
S6	Gilbert	Fred	214-4576	04/15/87	2,300
S7	Harvey	Candy	555-3323	12/01/89	2,450
S8	Iverson	Albert	121-3333	10/25/88	2,200
S9	Jones	Betty	342-4567	09/26/89	2,500

图 1.9　在 Salesman 表中增加 Phone No 列

但若一个销售员具有多个电话号码时，比如 Anderson 具有两个电话号码：123-4567 和 987-6543，就出现了问题。无法在同一个数据单元中存储它们，必须把它们存储在两个不同的单元中。对这个问题的解决可以有多种方法，其中之一是另外增加一列，如图 1.10 所示。

仔细研究一下修改后的 Salesman 表可以注意到，在新列中出现了不希望的特征。除非

大多数销售员都具有两个电话号码，表将不具有许多空单元。这将导致两个主要问题：由于每个单元在磁盘上都将占用一定数据的存储空间，空单元将造成存储上的浪费。另外，要根据电话号码查找销售员时就必须查寻两列，这样也就降低了处理速度。

Table: Salesman

Salesman ID	Last Name	First Name	Phone #1	Phone #2	Hire Date	Salary
S0	Anderson	Doris	123-4567	987-6543	07/01/86	2,800
S1	Bell	George	234-3456		10/12/88	2,400
S2	Carter	Jack	456-9023		05/14/87	2,550
S3	Davidson	Edward	234-5645		06/04/90	1,500
S4	Evans	Henry	635-2345		03/08/88	2,000
S5	Ford	Ida	345-2345		11/22/87	2,600
S6	Gilbert	Fred	234-4576		04/15/87	2,300
S7	Harvey	Candy	555-2323		12/01/89	2,450
S8	Iverson	Albert	123-3333		10/25/88	2,200
S9	Jones	Betty	342-4567		09/26/89	2,500

图 1.10　在 Salesman 表中增加更一列

考虑另一种解决方案：使用两行来存储有关 Doris Anderson 的信息，如图 1.11 所示。从图中可以看到 Anderson 被赋给了两个标识号：S0 和 S1。这是很必要的，因为 Salesman 表在用作主表连接到另一个表上时，每一行必须具有不同的标识号码。但分配给 Anderson 两个不同的标识号码，实质上也就是当作两个销售员来处理。很显然，这将带来很多问题。

Table: Salesman (Revised)

Salesman ID	Last Name	First Name	Phone No	Hire Date	Salary
S0	Anderson	Doris	123-4567	07/01/86	2,800
S1	Anderson	Doris	987-6543	07/01/86	2,800
S2	Bell	George	234-3456	10/12/88	2,400
S3	Carter	Jack	456-9023	05/14/87	2,550
S4	Davidson	Edward	234-5645	06/04/90	1,500
S5	Evans	Henry	635-2345	03/08/88	2,000
S6	Ford	Ida	345-2345	11/22/87	2,600
S7	Gilbert	Fred	234-4576	04/15/87	2,300
S8	Harvey	Candy	555-2323	12/01/89	2,450
S9	Iverson	Albert	123-3333	10/25/88	2,200
S10	Jones	Betty	342-4567	09/26/89	2,500

图 1.11　在 Salesman 表中增加为一行

正确的解决方法是，把电话号码从 Salesman 表中移走并存储到另一个表中。作为示例，我们把 Salesman 表分成两个表：Salesman 和 Phone，如图 1.12 所示。在需要时可以将它们连

接到一起。

Table: Salesman

Salesman ID	Last Name	First Name	Hire Date	Salary
S0	Anderson	Doris	07/01/86	2,800
S1	Bell	George	10/12/88	2,400
S2	Carter	Jack	05/14/87	2,550
S3	Davidson	Edward	06/04/90	1,500
S4	Evans	Henry	03/08/88	2,000
S5	Ford	Ida	11/22/87	2,600
S6	Gilbert	Fred	04/15/87	2,300
S7	Harvey	Candy	12/31/69	2,450
S8	Iverson	Albert	10/25/88	2,200
S9	Jones	Betty	09/26/89	2,500

Table: Phone:

Salesman ID	Phone No
S0	123-4567
S0	987-6543
S1	234-3456
S2	456-9023
S3	234-5645
S4	635-2345
S5	345-2345
S6	234-4576
S7	555-2323
S8	123-3333
S9	342-4567

图 1.12　把 Salesman 分开成两个表

分开 Salesman 表之后，可以把第一个表作为主表，第二个表在需要时连接到主表上。有关销售员 Anderson 的所有数据元素，除了电话号码之外，都可以在主 Salesman 表中找到。要查询电话号码，必须先在主表中找到标识号码 (S0) 然后进入 Phone 表中检索所有和标识号码相关的电话号码。但若想检索和某一销售员相关的所有数据元素，可以使用标识号码作为连接键连接两个表。

1.8.2　其他所期望的特征

前面我们提及了设计一个关系数据库时所必须遵循的一些特征，可归纳如下：

- 预先计划好所需要的数据。
- 预测所需信息，只存储相关数据。

- 在列中逻辑分组数据元素。
- 存储所有的数据元素和关系到表中。
- 无重复的行。
- 在一个数据单元中只能存储一个数据。
- 尽管避免空数据单元。

除了这些之外，用户还应尽量使数据库具备以下特征：

- 使数据库结构灵活，易于改变。
- 尽可能地减少冗余的数据。
- 使表保持逻辑索引。
- 尽可能地使数据表简明。

使数据库结构灵活

在确定数据表结构时，用户应尽可能地在同一个表中包括与一个数据实体相关的所有列。这些列应被逻辑分开，以适应将来的变动需要。也就是说，有关每个数据实体的所有信息都应能从数据表中得到。若用户的数据表结构灵活，能适应所有的数据操作和查询的需要，在以后希望得到一个新报表或不同类型的信息时，就不再需要重新建立结构。重新建立结构不仅费时，而且常常改变了许多相关的数据对象。

例如，若要在数据表中存储客户时，就需同时考虑如何确定表的结构以适应以后使用的需要。需要考虑以下问题：名和姓作为一个数据元素存储在一列中，还是作为两个数据对象分别存在两列呢？中间名怎样存储？是作为姓的一部分还是存在单独一列中？这些问题的回答取决于数据元素在应用程序中如何使用。若计划用名来寻找数据记录，就需要把名存储在单独一列中。若不考虑这些将来的使用要求，过后将是很被动的。

避免冗余数据

一个设计良好的数据库不应使单个数据元素多次出现。从前面的例子中可以知道，冗余数据会浪费宝贵的磁盘空间，并造成数据操作的错误。因此，在设计结构时应尽可能避免冗余数据。

这一规则的例子是数据操作将用在连接键时。要连接两个数据表，行必须共用相同的标识号码；因此，这种类型的数据重复是必须的。读者可回到图 1.12，注意在两个表中都出现了 Salesman ID。当需要把两个表连接到一起时，这些标识号被用作连接键。

逻辑索引数据

索引操作用来将数据记录按预定顺序排列。用户把记录加入到数据表中时，它们通常按输入的顺序排列。但这种顺序不利于寻找某个数据记录。为此，可按某种顺序来重新安排数据记录。

FoxPro 能够很容易地使用一个或多个列做为关键字段索引记录。数据库可按升序或降序排序。用户甚至可以使用包含键值组合的复合索引键。以后还将详细介绍索引操作。目前只需记住索引的概念。

使数据表简明

在大多数数据库管理程序中，用户需要一组数据表以存储所有的数据元素。理论上讲，应尽量减少列数以使数据表简单。表的列数越少，检索信息的速度越快，而且越易于管理。

数据表的列数应由数据元素的实际需要来决定。如前所述，尽量把和一个数据实体相关的列分组存入同一数据表中。但是，设计良好的数据库并不需要把一个数据实体的所有数据元素都存储到同一个表中。一些情况下，描述一个数据实体的特征所需的列很多，用户可以把它们放入多个表中。这就需要在数据表的数目和表的大小之间作出均衡。但要记住，行数对处理时间的影响要比列数大得多。

1.8.3 使数据相关

如上所述，数据元素之间的关系是关系型数据库的基础。它们是数据元素之间的连接，一旦用户掌握了怎样连接表，就能够创造出灵活的数据表结构。例如，当表的列数很大时，可以把它分开成几个表。然后，在需要利用分散在不同表中数据元素的关系时，可以通过提供必要的信息来连接它们。另外，当表中存在多对多关系时，也需要建立关系表来连接它们。

用户建立数据表的结构以适应关系的方式取决于数据元素间存在的关系的类型。这些类型即是前面所讲述的一对一、一对多和多对多关系。每一种类型都需要不同的结构。

处理一对一关系

最简单的一种关系是一对一关系。这种关系将一个数据对象和另一个独特的数据对象相关，每个对象可能包含一个或多个数据元素。例如，我们假定每个销售员只有一个雇佣日期，因此，销售员的名和雇佣日期间具有一对一关系。我们还假定一个销售员只分配到一个销售办事处中，那么在销售员和办事处之间也具有了一对一关系。

当在数据对象间具有一对一关系时，就能够在同一表中包括它们，在图 1.1 中我们已假定了销售员和销售办事处之间的一对一关系。为适应这种一对一关系。可以把和这些销售员及办事处有关的数据元素存储在同一表中，如图 1.13 所示。

Table: Salesman (including data for sales offices)

Last Name	First Name	Hire Date	Salary	Address	City	ST	Zip
Anderson	Doris	07/01/86	2,800	100 Park Avenue	New York	NY	10016
Bell	George	10-12-88	2,400	200 Lake Drive	Chicago	IL	60607
Carter	Jack	05-14-87	2,550	500 Century Blvd.	Los Angeles	CA	94005

图 1.13 在同一个表中处理一对一关系

注意在此图中每一行包括了有关一个销售员及办事处的数据元素。只有在销售员及办

事处之间存在一对一关系时才可以这样做。若在这些数据对象间存在其他类型的数据关系，就要相应地改变结构。

处理一对多关系

数据对象之间存在一对多关系时，若用户想避免重复数据，就不能在同一个表中存储所有相关的数据元素。若分配多于一个的销售员到给定办事处中，显然在一个表中必然重复有关办事处的数据。例如，若分配四个销售员——Anderson、Davidson、Gilbert 和 Jones 到 NewYork 办事处（B1），Salesman 表将如图 1.14 所示。

Table: Salesman (including data for sales offices)

Last Name	First Name	Hire Date	Salary	Address	City	ST	Zip
Anderson	Doris	07/01/86	2,800	100 Park Avenue	New York	NY	10016
Davidson	Edward	06/04/90	1,500	100 Park Avenue	New York	NY	10016
Gilbert	Fred	04/15/87	2,300	100 Park Avenue	New York	NY	10016
Jones	Betty	09/26/89	2,500	100 Park Avenue	New York	NY	10016
Bell	George	10/12/88	2,450	200 Lake Drive	Chicago	IL	60607
Carter	Jack	05/14/87	2,550	500 Century Blvd.	Los Angeles	CA	94005

图 1.14 在同一个表中处理一对多关系

Table: Salesman (including data for sales offices)

Last Name	First Name	Hire Date	Salary	Address	City	ST	Zip
Anderson	Doris	07/01/86	2,800	100 Park Avenue	New York	NY	10016
Davidson	Edward	06/04/90	1,500	100 Park Avenue	New York	NY	10016
Gilbert	Fred	04/15/87	2,300	100 Park Avenue	New York	NY	10016
Jones	Betty	09/26/89	2,500	100 Park Avenue	New York	NY	10016
Bell	George	10/12/88	2,400	200 Lake Drive	Chicago	IL	60607
Evans	Henry	03/08/88	2,000	200 Lake Drive	Chicago	IL	60607
Harvey	Sandy	12/01/89	2,450	200 Lake Drive	Chicago	IL	60607
Carter	Jack	05/14/87	2,550	500 Century Blvd.	Los Angeles	CA	94005
Ford	Ida	11/22/87	2,600	500 Century Blvd.	Los Angeles	CA	94005
Iverson	Albert	10/25/88	2,250	500 Century Blvd.	Los Angeles	CA	94005

图 1.15 Salesman 表中所有的一对多关系

从表中可以看出，冗余数据出现在表的第二到第四行。因为前四个销售员都属于 New York 办事处，前四行中有关办事处的数据元素应该是相同的。若同一办事处中包括数个销售员，冗余数据的问题就会更加严重。而且，若试图在表中包含所有的一对多关系，如图 1.12 所示，重复数据的问题就会变得很明显，参见图 1.15。这些重复的数据元素浪费存储

空间，破坏了数据库信息的准确性，应当尽量避免。

减少冗余的一种方法是在设计数据表的结构时，使有关给定办事处的数据元素只出现在数据表的一行中。为此，应存储所有和销售办事处有关的数据元素在一个表中，使有关销售员的数据元素存储在另一个表中。为提供两个表之间的连接，在 Salesman 表中增加一列以存储销售办事处的标识号码。图 1.16 中给出了数据库的新结构。

Table: Salesman (after adding the Office_ID column)

Salesman ID	Last Name	First Name	Hire Date	Salary	Office ID
50	Anderson	Doris	07/01/86	2,800	B1
51	Bell	George	10/12/88	2,400	B3
52	Carter	Jack	05/14/87	2,550	B2
53	Davidson	Edward	06/04/90	1,500	B1
54	Evans	Henry	03/08/88	2,000	B3
55	Ford	Ida	11/22/87	2,600	B2
56	Gilbert	Fred	04/15/87	2,300	B3
57	Harvey	Candy	12/01/89	2,450	B3
58	Iverson	Albert	10/25/88	2,200	B2
59	Jones	Betty	09/26/89	2,500	B1

Table: Office

Office ID	Address	City	State	Zip	Phone #
B1	100 Park Avenue	New York	NY	10016	800-123-5555
B2	200 Lake Drive	Chicago	IL	60607	800-234-5555
B3	500 Century Blvd.	Los Angeles	CA	94035	800-456-5555

图 1.16　使 Office 表和 Salesman 表相关

注意在 Salesman 和 Office 表中都出现了 Office ID 列。它提供了两个数据表之间的连接。因此，当需要属于某一销售员的给定办事处的信息时，首先是进入 Salesman 表通过名或标识号寻找销售员所在的一行，从 Office ID 列检索到办事处号码，然后进入 Office 表以得到有关办事处的信息。所有的任务都可以通过连接两个表中的数据元素来完成。在本例中，Salesman 表作为主表，Office 表作为连接表。

连接键

Office ID 列用作连接两个表的连接键，在 Salesman 中称为外部键，因为它向存储在另一个数据表中的数据元素提供了连接。

注意，一个设计优秀的关系型数据库中所有的行应该是根据某一列或一些列中的数值索引。本例中 Salesman ID 列可用作此目的。因为每个销售员号码都是独一无二的，它提供了一个快速而又准确的参考，指向和给定销售员相关的单个行。这样的行常常称为主表中的主键。与此类似，Office 表中的 Office ID 列也被称为主键，它给表中行提供了独特的标识