第一章 进入 FoxPro 世界

1.1 FoxPro 的特点

FoxPro 是 FOX SOFTWARE 公司自推出 FOXBASE+1.0、2.0、2.1 系列软件之后，推出的新一代关系数据库系统。由于该系统具有强大的对信息的分类、检索和处理等功能及友好的图形界面等诸多优点，因此又一次成为微机关系数据库中的佼佼者，为广大的新老用户带来了一次新的飞跃。

新的 FoxPro 2.1 版本具有下述的特点：

易于使用，初学者也能自如地使用它进行工作。

提供强有力的编程命令。

有良好的图形界面和窗口功能。

一个功能很强的内部文本编辑器。

与过去的 Foxbase 和 DBASE 完全兼容。

完善的编译系统，可以将源程序编译为可以在 DOS 环境下直接运行的“.EXE”文件。

较强的硬件适应性。

可重新定义的键盘宏。

内部的程序生成器。

此外，值得一提的是：

1. FoxPro 提供了对鼠标的支持。

2. FoxPro 对 Foxbase 命令作了大量的扩展。如：增加了 160 多条命令、142 个函数及 42 个系统内存变量。

3. 采用了一种被称为“RUSHMORE”(快速查找)的优化技术，可以使得在对较大的数据库执行查找时速度比普通的查找方法快几百倍，数据库越大，速度优势就明显。FoxPro 还采用了另一种被称为“Compact ”(紧缩)的索引方法，建立的索引文件长度较常规方法可缩短“20%”以上。这对大型的数据库文件来讲节省的磁盘空间是十分可观的。在 FoxPro 中，可以建立“Compound ”(复合)类型的文件，把对应数据库的所有索引文件都放在唯一的索引文件中，这样在修改数据库时，系统便可以自动修改与之相关联的索引文件。

4. FoxPro 提供了高级外部程序接口 API，可以方便地实现与“C”语言和汇编语言的连接，这对使用 FoxPro 编写大型的应用软件提供了十分便利的条件。

5. SQL(结构化查询语言)是日趋流行的一种标准关系数据库管理系统语言。FoxPro 的关系举例检索(RQBE)的功能就是通过 SQL 的 SELECT 命令处理的。

综上所述，由于 FoxPro 所具有的强大功能和特性，使它一问世就受到了广大用户和软件人员的高度重视，所以 FoxPro 具有十分美好的应用前景。

下表列出了一些在 DOS 和 Windows 环境下的 FoxPro 2.5 版的系统指标。

表 1.1 FoxPro 2.5 系统性能列表

	标准 FoxPro 2.5	增强 FoxPro 2.5	Windows FoxPro
每个数据库的最大记录数	十亿个	十亿个	十亿个
每条记录最大字符数	65000 个	65000 个	65000 个
每个数据库最大字段数	255	255	255
每个字段的最大字符数	254	254	254
最大工作区	25	255	255
IDX 索引表达式长度	100 各字符	100 各字符	100 各字符
CDX 索引表达式长度	240 各字符	240 各字符	240 各字符
可打开的索引文件数	只受可用内存和 files 数的限制		
数值计算的精度(小数位)	16	16	16
最大可用内存变量数	3600	65000	65000
数组的最大个数	3600	65000	65000
自定义函数的个数	不限制		
DO 命令的最多嵌套层数	32	32	32
READ 命令的最多嵌套层数	5	5	5
结构化命令的最多嵌套层数	64	64	64
可打开的最大窗口数	只受可用内存和 files 数的限制		
BROWSE 可打开的最大窗口数	25	255	255
每个键盘宏的最多击键次数	1024	1024	1024
最多可打开的文件数	99	只受 DOS 限制	

1.2 FoxPro 与 Foxbase+的兼容性

FoxPro 可以运行 Foxbase 的大多数源程序，而不需要作任何的改动，不过当你需要在 FoxPro 下运行 Foxbase 的程序时，最好使用下列命令来保证兼容性。

SET COMPATIBLE FOXPLUS

同时，FoxPro 还提供了一个特殊的系统配制文件 Config. FOX，它将保证 FoxPro 与 Foxbase 几乎百分之百地兼容，这个系统配制文件放在 FoxPro 2.5\goodies\misc 下，为了使用这一文件，你可以简单地将这个文件复制为 config. db，并放在当前工作目录下，也可以将 config. FOX 的内容放在 config. FP 文件中，为了实现兼容，还应在 config. FP 文件中包含如下语句：

BRSTATUS=ON
MACKY=<expc>
NOTIFY=OFF

```
SCOREBOARD=ON
STATS=ON
```

1.2.1 FoxPro 2.5 和 Foxbase+的差别

尽管 FoxPro 和 Foxbase+尽可能地完全兼容，但当你想在 FoxPro 环境下运行 Foxbase+的应用程序时，必须考虑尽量避免使用如下范围内的命令，或使用时做一些修改。

1. 出错提示信息

FoxPro 可以检测到被 FOXBASE+所手略的错误。如果在你的 FOXBASE+程序中包含有对这些出错信息的处理，那在 FoxPro 环境下和 FOXBASE+下的处理是不同的，你应当对它们进行适当的修改。

2. 新增的函数和命令关键字

FoxPro 新增了许多的命令和函数，也就是增加了许多新的关键字。因此，在你的 FOXBASE+源程序中可能会出现和 FoxPro 关键字重名的自定义函数；在这种情况下 FoxPro 将按照自身的语法规则来解释并执行这些函数，而不是去执行用户所定义的功能。如果有这种情况出现，那系统将发生所答非所问的情况。要避免该类问题的出现。例如：EVALUATE()函数是 FoxPro 系统的函数，如你定义了自己的和它重名的函数，就要换一个名字。

3. SET COLOR TO 命令

在 FOXBASE+中执行不带任何参数的 SET COLOR TO 命令时，将会使屏幕恢复到缺省状态。而在 FoxPro 环境下执行不带任何参数的 SET COLOR TO 命令时，则不会引起屏幕的任何变化。

4. SET DEFAULT 命令

在 FOXBASE+环境下，在执行 SET DEFAULT 命令指定缺省驱动器和目录时，实际上只起了指定驱动器的作用。如下例：

```
FOXBASE+CSET DEFAULT TO C:\FOXBASE\USER
```

在该例中，实际上止指定了驱动器“C”。而在 FoxPro 环境下就可以精确地指向目录。

5. 空字符的使用

在 FOXBASE+环境下，用户想把一个字符串初始化为空字符串(不包含任何字符的字符串)，执行下面的两条将起到相同的作用。

```
FRISTSTR=CHR(0)或
FRISTSTR=""
```

而在 FoxPro 环境下，第一条命令将产生一个包含有一个“空”字符的字符串，第二条命令才真正的产生一个空字符串。

6. .VUE 文件

在 FOXBASE+环境下，“.VUE”文件包含有数据库、索引、格式文件及各种 SET 命令的 ON/OFF 状态；在 FoxPro 环境下，“.VUE”文件除了包含上述信息外，还需要加上缺省驱动器、路径、过程文件、时钟位置等等。

1.2.2 如何将 Foxbase+文件转换为 FoxPro 2.5 文件

对于大多数的 FOXBASE+文件，一般不需要用户作任何的修改，FoxPro 在运行时会自动的完成这一过程；只有两类文件需要由用户作适当的处理，分别是扩展名是：".NDX"的索引文件和".DBT"的备注型文件。

1. 扩展名为".NDX"的索引文件

由于采用了新的索引技术，FoxPro 对数据库所建立的索引文件格式与 DBASE、FOXBASE+的不同。如果你正在运行含有 DBASE 或 FOXBASE+的".NDX"索引文件的应用程序，你也不必担心 FoxPro 将自动完成对数据库的重新索引，并生成新的索引文件，同时显示如下信息。

DBASE INDEX —— REBUILDING

缺省时，这类新的文件将被自动加入".IDX"的扩展名。如果你想继续使用扩展名为".NDX"的索引文件，只要在 CONFIG.FP 文件中加入下列语句即可。

INDEX=NDX

2. 扩展名为".DBT"的索引文件

FoxPro 的备注型文件(.FPT 文件)与 FOXBASE+和 DBASE 系列的备注型文件(.DBT 文件)稍有不同，".DBT"文件只能存放 ASC 码的文本数据，而".FPT"文件中可以存放任意类型的数据。但 FoxPro 可以读写".DBT"文件而不需要将其转换为新的文件格式。

有两种情况 FoxPro 会根据以存在的".DBT"文件来建立一个新的".FPT"文件。

(1) 根据一个已有的包含备注型字段且有备注型文件的数据库来建立一个新的数据库结构时，COPY TO 命令就是一例。

(2) 当你修改一个已有备注型文件的数据库结构时，在修改后的数据库存盘时".DBT"文件自动转换为".FPT"文件。

如想在 FoxPro 环境下为一含有备注型字段的数据库能在 FOXBASE+环境下使用则应执行如下命令：

COPY TO <Filename> TYPE FOXPLUS

1.3 Rushmore 技术介绍

Rushmore 技术是 FoxPro 2.5 采用的一种数据快速存取技术，它实现了对一系列记录的快速存取，其速读相当于对已索引的单个记录的存取速度。

用户通过使用 Rushmore 技术，可以使一些复杂的数据库操作比过去提高了成百上千倍。FoxPro 2.5 使得用户只需使用个人计算机便可以处理大的可包含上百万个记录的大型数据库，起动速度可与具有主机的数据库系统相媲美。

Rushmore 技术采用标准的 FoxPro "B"树索引，而不需要任何新型的文件或索引。Rushmore 技术可用于任何的 FoxPro 索引。如：".NDX"标准索引、".IDX"压缩索引或".CDX"复合索引。

Rushmore 技术不依赖于新的压缩索引文件格式。无论是.CDX 格式还是.IDX 格式的索

引，都采用一种压缩技术，它产生的索引文件只有旧格式索引文件长度的六分之一。压缩索引时可以以极高的速度运行，由于它只占用极小的物理空间，所以只须作少量的磁盘存取便可将大量的索引驻留在 FoxPro 的存储缓冲区。

尽管 Rushmore 技术得益于短的压缩索引文件，但它处理旧格式的索引文件时，效果也非常的显著。

当需要处理的数据库非常大时，内存配置低的微机系统不能够为 Rushmore 技术提供足够的内存时，会出现一些警告：

Not enough memory for optimization(没有足够的内存来实现优化)

程序还会继续运行下去，但不进行优化操作。虽然不会出现数据的丢失的情况，但在这种情况下，Rushmore 技术的优势已无法得到发挥。因此，如果用户在处理大型数据库(超过五十万条记录)时，建议用户使用 FoxPro 2.5 的增强型版本。

最简单的情况是：Rushmore 可加快对单个数据库进行操作的 FOR 子句的执行速度。采用 FOR 子句可根据以索引的数据库来限定符合条件的一组记录。当 SET FILTER 置为 ON。且对已索引过的数据库建立过虑条件后，Rushmore 技术还可加快表 1.2 中介绍的包含 FOR 子句的可执行优化的命令。

为了对多个数据库更好地发挥 Rushmore 技术的优势，用户应当使用 SQL SELECT 命令。FoxPro 的 SQL 工具在实现多个数据库的优化查询时。将 Rushmore 技术当作一个基本的工具，从而对已存在的索引实现 Rushmore 技术，甚至建立新的索引，以便加快查询速度。

1. 单个数据库的 Rushmore 技术

若处理的是单个数据库，用户可以在出现 FOR 子句的任何地方来充分发挥 Rushmore 技术的优势。Rushmore 技术的研制者使得它的速度与欲处理的记录数据成正比，即记录越多，使用 Rushmore 技术的优势越明显。

表 1.2 包含子句的可优化的命令

AVERAGE	COUNT	LIST	SORT
BROWSE	DELETE	LOCATE	SUM
CALCULATE	DISPLAY	RECALL	TOTAL
CHANGE	EDIT	REPLACE	LABEL
COPY TO	EXPORT	REPORT	SCAN

为了充分发挥 Rushmore 技术的优势，除了包含可优化的 FOR 子句，上表中的命令还必须含有一个带有 ALL 或 NEXT 的范围子句，若用户允许系统采用默认的范围 ALL，Rushmore 也可很好地发挥作用。

在进行优化时，Rushmore 可以使用任何已打开的索引文件，除了以过率的和唯一的索引。

在进行优化时，不必对数据库进行排序，如果用户建立了索引文件，那么这种排叙是自

动的。如果用户即想对一个大型数据库最大限度地发挥 Rushmore 技术的优势，同时又想要求其中记录按用户特殊的要求排序，那么最好使用命令 SET ORDER TO 关掉索引控制，然后执行排序命令 SORT。

2. 多重数据库的 Rushmore 技术

如果处理多余一个的数据库时，为了充分发挥 Rushmore 技术的优化作用，用户应使用 SQL SELECT 命令，SQL 在优化它的查询时，将 Rushmore 当作一个基本的工具。

用户一旦选择了 SELECT，就没有必要牢记在通常情况下的那些规则了。SQL 会自动决定在什么时候需要对一个查询进行优化，并自动完成优化查询，用户甚至不需要打开数据库或相应的索引文件，如果 SQL 觉得需要索引文件，它会为自己建立临时的索引文件。

3. 基本的可优化的表达式

Rushmore 技术产生作用与否，取决于在一个 FOR 子句中是否出现一个基本可优化的表达式。一个基本可优化的表达式可以形成一个完整的表达式，或只作为表达式的一部分。

一个基本可优化的表达是的形式如下：

<index expression><relational operator><constant expression>或

<<constant expression><relational operator><index expression>

(<索引关键字表达式><关系运算符><常量表达式>)或

(<常量表达式><关系运算符><索引关键字表达式>)

在一个基本可优化的表达式中：

. <索引关键字表达式>必须与建立索引文件时的索引关键字表达式精确匹配，而且索引关键字不能包含别名。

. <关系运算符>必须是下列运算符之一：<、>、=、<=、>=、<>、#、!。

. <常量表达式>可以是任何表达式，可包括内存变量和其他无关数据库的字段变量。

4. 组合的可优化的表达式

Rushmore 处理技术取决于 FOR 子句表达式，通过一个简单的或复杂的 FOR 子句表达式，如果 FOR 数子句是一个可优化的表达式，数据处理速度可以得到加强。基本的可优化的表达式可以通过逻辑运算符 AND、OR 或 NOT 组合成一个复杂的 FOR 子句表达式。

通过可优化的基本表达式组合成的表达式是完全可优化的。若其中有一个或多个基本表达式不可优化，则组合成的表达式也许是部分可优化的或不可优化的。

5. Rushmore 技术不可使用的情况

在少数情况下，Rushmore 技术可能达不到对数据进行加速处理的目的。一旦它不能对一个可优化的 FOR 子句进行优化处理，则 Rushmore 技术对于该命令失败。

对于在命令中含有 WHILE 子句，则 Rushmore 技术对于该条命令失败。

在 FoxPro 的标准版本下，当所有打开的数据库中记录总数超过五十万个，FoxPro 也需要关闭 Rushmore 技术，然而增强型版本的 FoxPro 可以加快命令的执行速度，无论是记录超过五十万条，还是少于五十万条。

在内存较小的情况下，Rushmore 技术不能优化数据处理命令。

1.4 运行 FoxPro 2.5 所需要的软硬件环境

单用户的 FoxPro 2.5 for DOS 由两种版本可供用户选择，标准型版本和增强型版本。增强型版本是一个真正的 32 位产品，它可以充分使用 80386 或 80486 微处理器的特性，并可使用所有的扩展内存，这意味着文件的运行速度会更快一些。

标准版本的 FoxPro 只要求 8088/8086 或 80286 的微处理器。

表 1.3 FoxPro 2.5 系统运行环境

	标准版本	增强版本
处理器	8088 或更高	386sx 或更高
RAM	640K(建议使用 2MB)	3MB
鼠标	建议使用	建议使用
DOS	DOS 3.1 或更高	DOS 3.1 或更

当 FoxPro 被正确地使用在合适的硬件平台上，它可提供一种非常快的速度。不过，作为一个复杂的软件产品，它需要扩展内存、EMS 内存、XMS 内存、高端内存、DOS 文件柄和缓充器、局域网操作系统等诸多因素的相互作用。因此，欲使 FoxPro2.5 能够高效地运行，就需要对你的计算机系统进行合理的配置，建立最有效的 FoxPro 的系统运行环境 CONFIG.FP。系统配置的越合理，FoxPro 运行的速度就越快。而这一要求的关键在于对基本内存的分配上，基本内存越大系统运行效路就越高。哪怕基本内存仅仅增加 5K 或 10K，也将大大地改善 FoxPro 的整体性能。

在 DOS 环境下，可通过 CONFIG.SYS 文件来完成对 FoxPro 环境的自动设置。DOS 的系统配置文件 CONFIG.SYS，在 DOS 启动时将根据文件内容配置微机的资源，其中有两条对 FoxPro 系统较为重要：

```
FILES= ??
BUFFERS= ??
```

关于 CONFIG.SYS 文件的各项参数的使用，请参考相应 DOS 版本的使用手册。在 FoxPro 环境下，上述两个参数在一般情况下较为理想的参数是：

```
FILES=35
BUFFERS=40
```

1.5 汉字环境 UCDOS 3.0 简介

软件汉化过去一直是困扰用户的一大难题。但这随着各类能直接支持西文软件的新一代汉字 DOS 系统的不断推出，这一难题已成为历史。下面要向大家介绍的是希望电脑公司于一九九三年十一月推出的新一代汉字系统 UCDOS 3.0 版。

UCDOS 3.0 系统的特点如下：

支持直接写屏，英文制表符自动识别

. 绝大部分西文软件毋需汉化即可支持汉字，充分保持原版西文软件的面貌，如原版 FoxPro、Peshell、Borland 系列、Quick 系列、Norton 系列等软件均可直接显示和输入汉字

. 采用高效先进的智能处理技术，能正确识别英文制表符，使之与汉字共存于同一屏幕，而不引起神突国内唯一真正可实现零内存的汉字系统

• 386 以上微机，只要有一定的扩充内存，系统在启动时就可自动将所有程序和数据放入扩充内存，不占用任何 DOS 基本内存，不受 DOS 版本限制，在 DOS 5.0 下，最多可为用户保留 637K 内存空间可直接利用 WPS 进行文字处理

. UCDOS 支持 WPS 的运行，并使之可在 DOS 5.0 和网络环境中运用自如，共享 UCDOS 提供的全部矢量字库

. 模拟显示和打印速度可提高 2—3 倍

真正实现网络共享

. 将网络版系统安装于服务器后，各工作站(包括无盘工作站)即可使用 UCDOS，工作站数目不受限制

. 各工作站均可拥有与单机相同的功能，如直接写屏、共享打印字库、特殊显示等

. 彻底解决网络中远程终端间的通讯问题，通讯数据可确保万无一失

. 显示字库可存放于服务器上，为各站点保留更多的低端内存，保证仅有 640K 内存的无盘工作站有更多的内存资源

强大的打印功能

. 国内唯一将点阵字库和矢量字库有机结合的汉字系统，保证了低点阵汉字的质量

. 支持大部分针打、喷打和激打，在任何软件中均可直接打印 2048×2048 点阵以内的任意点阵汉字

. 独特的打印字库还原技术，还原速度可与硬件媲美，使打印速度得到极大的提高，甚至超过硬字库的打印速度

. 支持 26 种矢量字库，基本系统提供宋、仿、黑、楷四种，用户可选配其它矢量字库

特殊显示功能

. 可在屏幕上显示任何点阵的汉字，大小仅受屏幕尺寸限制

. 提供丰富的作图功能，可利用控制字符在各种显示模式下，实现点、线、圆、椭圆、扇形、矩形及图形填充等多种功能

. 提供完善的音乐功能，利用控制字符可实现对简谱文件的后台演奏

注：以上所有特殊功能都可以被编程语言(汇编、C/C++、FoxBASE、BASIC 等)调用，彻底支持 DOS 5.0、DOS 6.0 和 DRDOS 6.0。

新一代的汉字输入方法

. 系统自带一经特别设计的"普通汉字输入法",该方法以词组输入为主,采用两位编码,拆分容易,平均码长短,动态重码率低,普通人只需稍加学习即可实现快速输入

. 独创"记忆词组"汉字输入领域的新概念,成功地解决了局部词组和专业性词汇输入困难的问题。任何人一经使用,便会爱不释手

. 同时提供区位、全拼词组、简拼词组、双拼词组、预选字等多种输入方法,并提供外挂输入法接口

. 支持屏幕取词、动态存词、系统级的宏定义、整字处理等功能

系统装载实现智能化,硬件适应性强

. 显示字库可选择多种驻留方式,并能自动按系统当前配置择优选取

. 自动识别各种显示设备,包括 HGC、EGA、VGA、CEGA、CVGA 等

在汉字环境下,彻底支持鼠标功能等等。

1.6 FoxPro 2.5 命令的基本约定

FoxPro 语言由两部分组成:命令和函数。一条命令执行一个操作,而一个函数则返回一个值。例如:CREATE 命令用于建立一个数据库,函数 DATE()返回当前的系统日期供你使用。函数包含一对括号以便同命令区分开。

命令和函数可以组合在一起组成一条 FoxPro 语句。函数不能单独使用,它们必须同命令一起使用。

例如,问号(?)是一条把输出发送到屏幕的命令,函数 TIME() 执行返回系统时间的功能。它们结合起来就可以完成向屏幕输出系统时间的功能。

```
? TIME()
```

1.6.1 命令和函数

FoxPro 命令和函数由一个或多个成分组成。这些组成决定了如何使用一个命令或函数。

让我们看一个典型的 FoxPro 命令的语法。下面以 REPLACE 命令来举例说明,该命令用于对数据库记录的字段进行更新。

```
REPLACE <field> WITH <expr1> [ADDITIVE]
[, <field2> WITH <expr2> [ADDITIVE]]...
[<scope>] [FOR <exeL1>] [WHILE <expL2>]
[NOOPTIMIZE]
```

表 1.4 FoxPro 命令结构举例

组成部分类型	以 REPLACE 举例说明
关键字	REPLACE,WITH,ADDITIVE,FOR,WHILE,NOOPTIMIZE
表达式	expr1,expr2,expL1,expL2
名 称	field1,field2

每一个 FoxPro 的命令或函数都至少包含一个关键字。它们采用大写形式,FoxPro 采用关键字来标识一条命令或函数。这些特殊字符被 FoxPro 保留位内部使用,因此由被称为保留字。

在命令 REPLACE 中关键字有 REPLACE(FoxPro 用它来标识一条命令),WITH、ADDITIVE、FOR 和 WHILE。关键字 WITH、ADDITIVE、FOR、WHILE 被用于字句中。

关键字也可简写成前四个字符。

< > 角括号及所括起来的小写字符表示这是需要由用户提供的信息。该信息可以是文件名、表达式和内存变量。

( ) 在所有的 FoxPro 函数中必须包含小括号。小括号以其自身的形式出现在函数的语法说明中。

[ ] 方括号及所扩起来的内容表示 FoxPro 命令或函数的任选部分。在实际输入过程中可不输入它们。例如,REPLACE 命令提供了大量的任选项,你可以选择其中的一项或多项,子句 ADDITIVE,<scope>,FOR<expL1>,WHILE<expr2> 和 NOOPTIMIZE 都被放在方括号中以表示它们是可选项。

| 一个竖杠符号用于分隔任选项,这些并置的任选项一次只能选择其中的一项。

... 一个省略符号表明一条命令或函数的一部分可以以类似的格式继续。

1.6.2 表达式

一个 FoxPro 表达式由这些部分组成:数据库字段、函数、内存变量、数组元素、常量和操作符。FoxPro 系统的表达式分成四类:自符表达式、数字表达式、日期表达式、逻辑表达式。

在 FoxPro 命令和函数中,表达式以下表方式给出。

表 1.5 表达式类型列表

表达式	表达式类型
<expC>	字符表达式
<expN>	数字表达式
<expD>	日期表达式
<expL>	逻辑表达式
<expr>	字符、数字、日期或逻辑表达式
<expr list>	以逗号分割表达式

当命令或函数中相同类型的表达式出现一次以上时，则在表达式之后加上一个数字以表明它们在命令或函数中的位置。例如，函数 CHRTRAN()的语法为：

CHRTRAN (<expC1>,<expC2>,<expC3>)

这表明 CHRTRAN()函数需要三个字符表达式参数：<expC1>,<expC2>,<expC3>

由数据库字段、函数、内存变量和数组元素组合成的表达式各成份应当具有相同的数据类型。若它们的类型不一致，则 FoxPro 产生一条"操作符/操作类型不匹配"(Operator / operand type misnatch)的错误信息。

1. 字符表达式

字符表达式组成如下：

. 字符类型的数据库字段

. 返回字符值的函数

. 包含字符数据的内存变量和数组元素

. 字符常量或称为字符串

一个字符串是由单引号或双引号括起来的一串字符。例如："hello "或 ' hello'。引号必须匹配，也就是说，一单引号开始的字符串不能以双引号结束。

你可以在一种类型的引号中括起来的文字串中嵌入另外一种引号。例如 " Don't touch!"是一个合法的字符串。下表列出字符表达式的所有操作符。

表 1.6 字符表达式操作符

操作符	操作
+	连接字符串，将两个字符串连接在一起
−	连接字符串，删去第一个字符串和第二个字符串的尾逢空格
$	字符表达式比较

空串可以使用一对二者之间没有空格的引号来表示如："" 或 ''。如果 CHAR()函数的参

数为'0',即 CHAR(0),则它返回为空串。

注意:在所有的 FoxPro 资料中,将会经常提到空串。空串是一个长度为零的字符串,它不包含任何字符。

2. 数字表达式

数字表达组成如下:

. 数字类型的数据库字段

. 返回数字值的函数

. 包含数字数据的内存变量和数组元素

. 数字常量

数字常量是由数字组成的变量。下表列出了数字表达式的所有操作符。

表 1.7 数字表达式操作符列表

操作符	操作
()	扩号用与改变运算顺序
**|^	幂
*,/	乘、除
%	取模(余数)
+,-	加、减

3. 日期表达式

日期表达式组成如下:

. 日期类型的数据库字段

. 返回日期值的函数

. 包含日期数据的内存变量和数组元素

. 日期常量

日期常量为数字:1、2、3、1981 等等。日期常量表示天数,对于一个给定的日期,你可以减去或加上一个日期常量(天数)。

可以用花括号括起来以指定日期。如命令:

STORE {12/22/85} TO F_DAY

建立一个日期型的内存变量 F_DAY 来保存日期"12/22/85"。

4. 逻辑表达式

逻辑表达式的值是下列二者之一"真"或"假",在 FoxPro 系统中分别用".T."和".F."表示真和假。

逻辑表达式组成如下:

. 逻辑类型的数据库字段
. 返回逻辑值的函数
. 包含逻辑数据的内存变量和数组元素
. 由专门的关系操作符分割开的其它表达式(字符、数字、或日期)

下表分别列出了逻辑表达式的操作符和关系操作符。

表 1.8 逻辑表达式操作符列表

逻辑表达式的操作符(按优先顺序)	
操作符	操作
()	括号用于改变运算次序
! |NOT	逻辑非
AND	逻辑与
OR	逻辑或

表 1.9 逻辑表达式关系操作列表

逻辑表达式的关系操作符	
操作符	操作
<	小于
>	大于
=	等于
<>|#|! =	不等于
<=	小于等于
>=	大于等于
==	字符串模等比较(当 EXTRACT 为 ON 时尾部空格有意义)

5. 记录范围:<scope>、FOR 和 WHILE

当命令对数据库中的记录进行操作时,你可以特别指定该命令所起作用的范围。使用<scope>、FOR 和 WHILE 子句确定记录的范围。

范围:当包含一个范围时,命令在数据库中特定的记录范围内起作用。你可以使用下列子句确定记录的范围。

ALL

命令对数据库中的所有记录起作用。

NEXT <expN>

命令在某范围内执行。该命令起始于当前记录并包含以后的 N 个记录,N 是数字表达式 expN 的值。例如:NEXT 1 在当前的记录上操作。

RECORD <expN>

命令对数据库中指定的第<n>个记录进行操作，N 是数字表达式 expN 的值。

REST

命令在某范围内执行，该命令起始于当前记录并持续到文件的最后一个记录。

FOR <expL>

一个命令的作用范围也可以使用 FOR 子句和 WHILE 子句来确定。

当使用子句时，命令对每一个满足指定逻辑条件的记录产生影响。

WHILE <expL>

从另一角度看，WHILE 表达式命令仅在逻辑表达式为真时，才对每一数据库记录进行处理。如果表达式的值为假，则不管后面的记录如何，都中止命令的执行。这一表达使经常与数据库字段一其使用，并根据 WHILE 表达式中使用的一个或多各字段对数据库进行排序和索引。

范围、FOR 表达式和 WHILE 表达式均可在同一 FoxPro 命令中使用。如果在命令中同时指定了 FOR 和 WHILE，则 WHILE 表达式具有较高的表达式。

1.6.3 数据工作区

FoxPro 允许在 25 个工作区内打开并操作数据库文件。可以用字母 A 至 J 表示前十个工作区或使用数字"1 — 25"来表示工作区，当然也可使用数据库文件的别名来区分。

为了在非当前的其它工作区将某个数据库文件打开，你必须制定或选择工作区。一旦库文件被打开，它就获得一个能够表示它的别名。除非你另外指定，否则系统将自动地为它分配一个别名，该别名和你的数据库文件名相同。

缺省的别名

如果你在工作区'A'使用以下命令打开数据库文件 CUSTOMER. DBF

SELECT A

USE CUSTOMER

系统将自动为数据库赋一别名为 CUSTOMER，该别名随后可以 在命令和函数中用于表示数据库。

用户指定的别名

在打开一个数据库时，你可以为数据库赋一别名。你可以用下面的命令打开 CUSTOMER. DBF 并为其赋一别名 PEOPLE

SELECT A

USE CUSTOMER ALIAS PEOPLE

随后就可以使用别名 PEOPLE 引用该数据库。

数据工作区的使用

在数据库被打开之前，你可以使用工作区符或数字引用个工作区。

例：

SELECT A

或

SELECT 1

之后你就可以选择相应的工作区来使用该工作区下的数据库文件。

1.6.4 使用的缩写

下表列出了包含 FoxPro 命令和函数在说明中采用的术语和缩写。

表 1.10 FoxPro 术语和缩写一览表

简写	含义
alias	别名
array	数组名
boder string	边界定义串
column	屏幕或窗口列
command	命令
delimiter	分割符
dir	目录
drive	磁盘驱动器
fcode	函数代码
key	索引表达式
list	表达式、字段…的集合
macro name	键盘宏定义名称
memo field	备注字段
pad name	条行菜单中的项目
parm list	参数表
path	路径
procedure name	过程文件
row	屏幕或窗口行
scope	数据库记录范围
topic	题目
var	内存变量、数组元素
work aera	数据工作区

第二章 FoxPro 程序设计

Foxpro 提供的编程功能是十分完善的，Foxpro 的命令格式类似于英语的日常用语。易读性好，它的每一基本指令又可派生出多条命令。整个命令系统提供了处理大型、复杂数据库系统的能力，利用这些命令你可以开发出大型的数据库管理信息系统。

2.1 结构化程序设计

FoxPro 提供了先进的结构化程序设计功能。整个命令语言采用了类似于 PASCAL 语言的方式。体现了结构化程序设计的特点。第一，控制流基于顺序、选择、重复三种方式，不使用 GOTO 语句。第二，大程序要分解成小的模块。结构化程序设计可以使用由上至下、逐步细化的方式进行编程。用这种方法编制的程序易于阅读、易于检验其正确性、便于用户维护等等。

2.1.1 顺序设计

顺序程序设计就是根据事物的处理顺序和要求，将相应的指令按照它们所完成的功能有机的结合起来的一个指令序列；这些指令的执行是按它们的排列顺序一条接一条的来执行。

如下例：该程序完成将一个数据库打开并显示该数据库的数据结构。

```
SET TALK OFF                    &&  关闭系统对话
USE PZK0                        &&  打开数据库"PZK0"
CLEAR                           &&  清除屏幕
DISPLAY STRUCTURE               &&  显示数据库结构
WAIT "请键入任意一键返回"       &&  等待用户响应
CLEAR                           &&  清除屏幕
SET TALK ON                     &&  打开系统对话
RETURN                          &&  返回调用程序
```

从上例可以看出顺序程序设计不需要特殊的编成技巧，只要将所需完成的工作一步一步地分析清，再将之翻译成相应的 FoxPro 命令即可。

2.1.2 分支设计

最简单的程序只是一列命令从头执行到尾。但世界上的事物不可能只是一个简单的顺

序过程;大多数的时候你需要根据不同的情况采取不同的处理方法。这就需要程序本身具备有控制程序流向的能力,这也是结构程序设计的三大特征之一。FoxPro 系统为我们提供了专门用于选择和判断的命令序列。

一、简单判断语句(IF … ENDIF)

格式:IF (条件表达式)

·

语句序列

·

ENDIF

其中(条件表达式)可以是各种表达式的组合,但表达式的返回值必须是逻辑值;即真".T."或假".F."。当条件表达式的值为真时,执行在 IF 和 ENDIF 之间的命令行序列;当表达式的返回值为假时,则跳过在 IF 和 ENFIF 之间的指令序列而去直接执行在 ENDIF 语句后面的命令序列。

下面是一个带简单判断的范例。

```
SET TALK OFF
USE customer
GETEXPR '请输入查询的条件' TO temp;
        TYPE 'L' DEFAULT 'COMPANY = ""'
LOCATE FOR &temp    && 设定要查询的条件
IF FOUND()    && 判断是否找到
        ? 'Company:' + company    && 如果找到则先时客户信息
ENDIF
USE
SET TALK ON
```

二、选择判断语句(IF — ELSE — ENDIF)

如果有时需要判断的问题比较复杂。如上例,我们需要对问题的处理需要进一步的处理,当未找到时如何处理?那简单的判断语句就显得力量比较单薄,同时使用起来也不是很方便。FoxPro 系统提供了另一种处理这种问题的语句。选择判断语句。其格式如下:

```
IF(条件表达式)
    命令序列一
ELSE
```