Foxbase+应用技术100例

李春葆 编

13

北京希望电脑公司

Foxbase+应用技术100例

李春葆　编

北京希望电脑公司

前　言

Foxbase+关系型数据库管理系统简单易学，使用灵活方便，适应能力强，是目前国内使用十分普及的系统，为了使更多的用户能够熟练掌握Foxbase+系统　快速高水平地研制出适用的管理信息系统软件．我们编写了《Foxbase+应用技术100例》一书，以满足广大用户需要。

本书收集了Foxbase+行家的宝贵经验，从适用性角度讨论了Foxbase+系统及其应用技术，重点放在扩充Foxbase+图形功能、菜单设计、Foxbase+与C、汇编语言接口等应用技术方面，同时也包含了Foxbase+的一般应用如宏代换技术、数据库操作和维护等。本书具有很强的实用性。

本书收集的应用例子中少数是讨论dBASEⅢ的，但它们也完全适用FoxBASE+，如FoxBASE+的.DBF文件与dBASEⅢ的.DBF文件结构完全兼容，且绝大部分命令和函数也相同，为此　我们也把这部分在dBASEⅢ上开发的例子也收集在本书中。

本书的出版得到希望电脑公司秦人华经理的大力支持，在此表示衷心的感谢。

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

Foxbase+应用技术100例

应用1：如何使用Foxbase+的陷井技术

我们在程序设计中经常需要设立中断陷阱，用以捕捉可能发生的非常事件或用户希望出现的事件，动态地控制程序流程。

陷阱是通过在程序开头设置陷阱语句来实现的。当程序运行时，FoxBASE⁺程序就去查找期望的中断事件发生了没有，如果发生了就暂时中断程序的正常运行，转到一个专门处理该事件的程序段中进行处理，待处理完毕后再返回到被打断的程序继续执行。那个专门处理预想事件的程序段叫作续元程序，是用户按照自己的意愿编写的。在程序设计中使用陷阱技术，可简化程序编制，并能设计出容错性极好的程序来。

FoxBASE⁺中陷阱设置语句主要有以下两种。

程序出错陷阱：

```
ON ERROR 〔命令〕
```

键盘陷阱：

```
ON ESCAPE〔命令〕
ON KEY〔〔命令〕
```

下面用例子说明这些命令的具体应用。

1. **出错陷阱**

在正常情况下，如果程序运行中发生错误，一般是自动停止程序的执行并在屏幕上显示错误信息。但是，若在程序开头设置了出错陷阱，则一旦捕捉到错误就自动转向该命令后的语句，这个语句可以是FoxBASE+中的任意语句，也可以是用户编的处理程序。用ERROR( )函数能够得到出错号，用MESSAGE ( ) 函数可以得到出错信息。

下面给出的处理程序是处理错误号为39和19的自动修改程序。39号错误为数值型字段溢出，19号错误为索引文件与数据库不匹配。

在主程序中可写如下语句：

```
ON ERROR DO MEND WITH ERROR( )    &&出错执行MEND程序。
```

那么可编写如下处理程序：

```
PARA ERRNO
DO CASE
   CASE  ERRNO=39
         E1=DBF( )
         E2=ALIAS( )
         E3=IIF(NDX( )==" ", " ", "INDEX  "+NDX ( ) )
         E4=RECN( )
         COPY STRC EXTE TO T
```

```
            USE T
            REPL FIELD-LEN WITH FIELD-LEN+1
            FOR FIELD-T YPE= "N" &&续上行
            CREA TT FROM T
            APPE FROM &E1
            USE
            ERASE &E1
            REMA TT.DBF TO &E1
            ERASE T.DBF
            USE &E1 ALIAS &E2&E3
            GO E4
            RETRY
        CASE ERRNO=19
            REINDEX
            RETRY
ENDCASE
```

2. 键盘陷阱

键盘陷阱有两条命令，一个是用ESC键来控制程序流向，另一个是用其它键来控制程序的流向。

ON ESCAPE〈命令〉常用来控制打印输出。打印输出时遇上打印机或其他故障，若按ESC键强行中断，则有可能导致数据库损失或数据丢失。最好的办法是按ESC键退出打印模块，返回菜单，排除故障后继续打印。

可以编写如下程序段：

```
ON ESCAPE RETURN TO MASTER          && 按ESC键返
SET DEVI TO PRIN                    && 回主菜单
DO WHILE .NOT.EOF( )
      ……
      打印工资单程序
      ……
      SKIP
ENDDO
```

当使用ON KEY〈命令〉时，按下任一个非ESC键，系统总是在当前命令执行完后，才转向执行参数项所指定的命令。

若转移执行的命令是由DO驱动的命令文件，则该命令文件中必须包括删除键盘缓冲区的INKEY( )或READKEY( ) 命令。因为按下的任一键始终储存在键盘缓冲区里，系统会不断地重复执行命令，这样需要将按下的这一键从缓冲区中删除掉，不然就会形成一个死循环。

若转移执行的命令是一条完整的FoxBASE⁺语句，那么这条命令必须是可以接受字符的，如ON KEY WAIT TO〈内存变量〉命令。按下任一键并不会使程序暂停，只是将按下这一键的内容存入指定的内存变量中。

例如，在数据库系统运行过程中，想在程序状态下进行文件拷贝，可以通过按“C”键

来实现。

```
***MAIN.PRG***  主模块
ON KEY DO CB
@0, 5 SAY "按C键拷贝文件"
@2, 5 SAY "通用工资管理系统"
@4, 5 SAY "按ENTER键进入主菜单"
@6, 5 SAY "按SPACE键退出"
@8, 5 SAY "请稍候……"
N=1
DO WHILE N<100
N=N+1
ENDDO
    ⋮
    ⋮
***CB PRG***
WAIT "选择C键码" TO XZ"
IF UPPE(XZ)= "C"
CLEAR
ACCEPT "输入源文件名（路径名/文件名）" TO SOURCE
ACCEPT "输入目标文件名（路径名/文件名）" TO TARGET
COPY FIELS &SOURCE TO &TARGET
ENDIF
RETURN
```

需要说明的是，ON ESCAPE和ON KEY两条命令只有在程序执行过程中才能生效，在圆点提示符下不会发生转移执行命令。

本文所附程序段均在CCDOS 2.10下的FoxBASE 2.0中调试通过。

应用2：如何自动得到命令文件调用关系

一个用Foxbase语言写的管理信息系统通常由上百个至数百个prg文件组成。当要从该系统中隔离出一个子系统，或分析、修改现成的系统，或整理刚完工的系统时，都需要了解各prg文件间的调用关系。我们可以用人工的方法阅读每个prg文件以了解这种调用关系，但这样做的工作量很大。为此我们设计了一个自动得到命令文件间调用关系图的程序。

在这个程序中，我们针对以下调用关系进行了相应的处理：①标准树型、②共叶型（如几个文件调用一个公共过程）；③递归型；④连枝型（如一分系统引用另一分系统的功能）；⑤有飘叶的树型（如存在没有被引用的文件）；⑥混合型。该程序使用了三个数据库：

a) AAA.dbf：存放调用关系图，二个字段：

1. WJM，C型，14字节；2.YDBZ，C型，1字节

b) CCC.dbf：存放一命令文件的语句行，一个字段：YJH，C型，256字节

c) BBB.dbf：存放待分析系统的所有prg文件名，一个字段：BBBWJM，C型，14字节

程序工作的基本原理是：由用户输入最初调用的prg文件名，将此文件名填入AAA.dbf.YDBZ（阅读标志）置为“n”，以此为初始状态，程序自上而下检查AAA.dbf中有没有YDBZ为“n”的文件名。若没有则运行完毕；若有则将该prg文件用APPEND加入到CCC.dbf中，

每个语句行依次对应一个记录.这样我们便可象操作字符串一样处理语句行了。当发现“DO〈prg文件名〉”这种语句时，立即将该〈prg文件名〉填入AAA.dbf并进行测试。若AAA.dbf中没有该文件名，则将YDBZ置为“n”，以备递进地对其进行阅读分析。若有该文件名，则YDBZ置为“y”，表明无需阅读，避免②、③、④三种树型产生的重复工作或无限循环。当阅读完一个文件后，将其YDBZ置为“y”。运行完毕后，调用关系图存放在AAA.dbf中，其间不再有YDBZ为“n”的记录，YDBZ为“=”的文件是父节点，紧接其后至下一YDBZ为“=”的其它文件是子节点。子节点在其后又可能成为父节点。

为处理飘叶型树，需进行以下步骤：先用DOS的管道命令得到所有PRG文件的列表，再转换到BBB.dbf中，然后与AAA.dbf比较，BBB.dbf有而AAA.dbf没有的PRG文件即为飘叶，因篇幅有限，程序中没有列出完成这个功能的代码。得到BBB.dbf的具体方法是：

!DIR *.prg>TT.TX USE BBB APPEND FROM TT.TXT SDF 附：程序清单：

```
set talk off
set safety off
set exact on
select 1
use aaa
zap
select 2
use ccc
zcdy=" "
@10, 10 say'请输入最初调用的文件;
            名（带扩展名）' get zcdy
read
sele 1
append blank
replace wjm with zcdy, ydbz with 'n'
**以下逐个分析aaa.dbf中没有分析过
** (即ydbz='n') 的PRG文件的调用关系
do whil .t.
  select 1
  locate for ydbz='n'
  if .not.eof ( )
    bhbh=recno ( )
    wjmn=wjm
    ? '正在分析'+wjmn
     jf file(wjmn)
     append blank
     **父节点标志 ydbz为'='
     repl wjm with wjmn, ydbz
       with '='
     select 2
     zap
     tt='append from'+wjmn+'sdf'
     &tt
     go top
     **去语句行左空，变字母为大写
     do whil .not. eof(    )
        replace yjh with ltrim(upper
         (yjh))
        skip
endd
go top
**定位在DO*.PRG处
locate for at('%DO','%'+yjh)< >0;
       .and.at('DO CASE', yjh)=0;
       .and.at('DO WHIL', yjh)=0
do while .not.eof( )
**求PRG文件名
   nn=at(' ', substr(yjh, 4))
   if at('.PRG', substr(yjh, 4))( )0
      nn=at('.PRG', substr(yih, 4))
   endi
? trim(wjmn)+'里含有语句'+yjh
ss=substr(yjh, 4, nn-1)+'.PRG'
select 1
locate for trim(wjm)=ss
*若调用关系库中有&ss则为2,3,4三种树
*故ydbz置为'n'以免死循环，否则置为 'y'
if .not. eof( )
   append blank
   replace wjm with ss, ydbz with'y'
else
   apoend blank
   replace wjm with ss,ydbz with 'n'
endif
select 2
continue
enddo
```

```
endif                                    else
**名为&wjmn的PRG文件的调用关系已处理完      exit
select 1                                 endif
go bhbh                                  endd
replace ydbz with 'y'                    return
```

应用3：如何实现数据库的关联

本文叙述了在编制数据库管理程序中，怎样充分应用SET RELA命令，以及使用该命令时应该注意的几个问题。

SET RELA命令的基本功能是：在某区（设为主区）将打开的数据库按关键字或表达式向其它区（设为从区）打开的数据库建立关联，以达到主区库记录指针移动时，从区库记录指针随之作相应移动的目的。

但是，如果不是按从区的数据库记录号设立关联，那么实现这一功能还须满足两个条件：一是从区数据库已按相同类型关键字或表达式建立了索引文件，二是索引文件已有效打开（在SET INDE TO命令的索引文件表列中排在第一位）。

另外，用SET RELA命令设置关联后，如果在主区移动记录指针而从区库中没有相关记录，则从区库的函数EOF（）值为真，从区库记录指针指向的记录号是BOOT的记录号加1，所有字段的数据值皆为空。

由此可见，①在用SET RELA建立主从区数据库关联的命令中，关键字或表达式只要与从区索引文件的类型一致，命令就可生效；②关联建立以后并不等于就有了主从区数据库记录的一一对应关系。

根据SET RELA命令的功能特性，与其它命令配合可以实现如下几项功能：

（设有数据库Q1.DBF含字段A1、A2；数据库Q2.DBF含字段X1、X2，并按与A1同类型的关键字建立了索引文件Q2.IND；数据库Q3.DBF含字段Y1、Y2，并按与A2同类型的关键字建立了索引文件Q3.IND。上列字段类型均为字符型。）

一、与LIST、DISP命令配合，实现多区关联库字段数据的同步显示或打印。形如以下程序可在同一行上显示或打印数据。这里应注意的是，若各字段的类型不同，须用STR()、DTOC（）、VAL（）等函数进行类型一致的转换。

```
SELE 1
USE Q1
SELE 2
USE Q2 INDE Q2.IDX
SELE 3
USE Q3 INDE Q3.IDX
SELE 1
SET RELA TO A1 INTO B
LIST A1+A2+B->X1+B->X2 TO PRIN
SET RELA TO A2 INTO C
DISP ALL A1+A2+C->Y1+C->Y2 TO PRIN
```


此为试读，需要完整PDF请访问：www.ertongbook.com

二、与GET、REPL命令配合，实现向关联数据库中字段数据的同步写入或修改。如接着上面程序执行下列程序，就可实现在一个工作区内同时向多个工作区跟踪录入、替换数据的功能（主从区数据库已有一一对应的记录）。

```
X=1
Y=10
@X, Y GET A1
@X+1, Y GET A2
@X+2, Y GET C->Y1
@X+3, Y GET C->Y2
READ
SET RELA TO A1 INTO B
@X+4, Y GET B->X1
@X+5, Y GET B->X2
READ
? A1, A2,B->X2,C->Y2
repl B->X2 with A2,C->Y2 with A1
? A1, A2,B->X2,C->Y2
```

三、向多个从区建立关联，实现不用查找命令的多区查找功能。接着以上程序执行以下程序可同时显示2、3两区的有关记录：

```
GO TOP
DO WHIL .NOT.EOF( )
SET RELA TO A1 INTO B
SET RELA TO A2 INTO C
? A1,A2,B->X1,B->X2,C->Y1,C->Y2
SKIP
ENDD
```

四、向某个从区多次建立关联，实现不用查找命令的条件不同的多次查找功能。试执行以下程序：

```
GO TOP
DO WHIL .NOT.EOF( )
SET RELA TO A1 INTO B
? A1, A2,B->X1,B->X2
SET RELA TO A2 INTO B
? A1, A2, B->X1, B->X2
SKIP
ENDD
```

用SET RELA命令建立关联，有时并不能如愿以偿，这就要求我们注意以下几个问题：

①主区的关键字或表达式必须与从区有效打开的索引文件的关键字或表达式类型一致，否则将出现命令错误。②如果建立关联后到从区移动记录指针，主区的指针是不会跟着作相应移动的，而且在返回主区后如不作指针的重定位或隐含重定位动作的操作，从区指针也不会自动移到相关联的记录上。③用数字表达式建立关联时，若从区与主区相关联的关键字是库记录号，则从区不需建立、打开索引文件；若相关联的是数字型的字段或表达式，则从区必须有相应的索引文件有效打开，从区记录指针将按索引文件的关键字（而不是按记录号）

随主区作相关移动。④dBASEⅢ在同一时刻在某主区内只能向一个从区建立关联，但不等于在同一时刻只能建立一对关联。比如在1区向2区建立关联后再在3区向4区建立关联，两对关联都是成立的——FOXBASE也是如此。⑤FOXBASE在同一时刻在某主区可向多个从区建立关联（如上面第三点程序中的二个SET命令可合并成 SET RELA TO A1 INTO B, A2 INTO C），但应在关联使用完毕后及时将其断开，否则将影响程序的执行速度。⑥用SET RELA TO命令可以断开主区对从区的关联，但如果有USE或CLOS命令关闭主区或从区的任一数据库，这一关联自然断开；如果在主区或从区重新打开数据库（包括在当前区内对已打开的数据库再执行一次USE命令），关联也自然断开。

应用4：如何实现数据库关联操作

本应用程序实现数据输入时的库关联，所用库文件有LLK.DBF和ZLK.DBF，LLK.DBF的结构如下：

Field	Field Name	Type	Width	Dec
1	材料编号	Character	8	
2	材料名称	Character	12	
3	领料数	Numeric	12	2
4	金额	Numeric	12	2
** Total **			45	

zLK.DBF的结构如下：

Field	Field Name	Type	Width	Dec
1	材料编号	Character	8	
2	库存量	Numeric	12	2
** Total **			21	

当输入领料单（LLK.DBF）数据时，一旦输入材料编号便查找ZLK.DBF中该编号的材料的库存量，并显示在屏幕上，当输入领料数后，便计算出该材料的余额，如果余额小于零，便显示有关出错信息，否则便显示该次领料后该材料的库存量。

数据输入正确后，再把本次输入数据存放在LLK.DBF数据库中，程序清单如下：

```
set talk off
set stat off
set scor off
set colo to 7/1
clea
zlbh=space(8)
zlmc=space(12)
lls=0
je=0
select 1
use llt
seleck 2
use  zlk
select  z
@3, 25 say "输  入  领  料  单"
@4, 25 say "==========="
@6, 10 say "材料编号："
@6, 40 say "材料名称："
@8, 10 say "领 料 数："
@8, 40 say "金    额："
@10,10 say "领料前库存量："
@10,40 say "领料后库存量："
do whil .t.
   @6,20 get zlbh
   read
   locate for zlbh=材料编号
   if .not.  found( )
      @18,30 say "没有该编号的材料"
   else
      @10, 23 say  库存量
      exit
   endif
```

```
enddo
@6, 50 get zlmc
read
do whil .t.
    @8, 19 get lls pict '999999999.99'
    read
    if lls>库存量
        @18, 30 say "输入的领料数有误"
    else
        @10, 53 say 库存量-lls
        exit
    endif
enddo
@8, 49 get je pict '999999999.99'
read
select 1
appe blan
repl  材料编号  with zlbh,材料名称 with
zlmc
repl  领料数 with lls, 金额  with je
close all
return
```

应用5：如何在程序中检测错误

在应用程序设计好的检测错误功能可以提高整个程序的容错性，Foxbase+提供了on error语句来捕获所出的错误，如下程序具是这样的功能，当用户输入一个错误的库名时，use &fn出错，使b=1使程序进行相应的处理。

程序清单如下：

```
set  talk off
set  stat off
set  scor off
set  colo to 7/1
clea
on error b=1
fn=space(8)
@12, 30 say "输入数据库名："
do whil .t.
    b=0
    @12, 43 get fn
    read
    use &fn
    if b=1
        @20, 24 say "你给定的库文件不存在，请重新输入！"
    else
        exit
  endif
enddo
close all
return
```

应用6：如何将阿拉伯数字转换成英文数字

在中外合资、独资企业中，经常需要将阿拉伯数字转换成英文数字。本文介绍一个用FOXBASE编写的转换程序，可以将千亿以内的阿拉伯数字转换为英文数字，以满足一般

管理信息系统的需要。

转换原理

1．建立数据库SZ．DBF(见图一)。库中有最基本的基数词,这些基数词可以和“—”（十位和个位间所加连字符）、“，”（从后向前数，每三位加一个逗号）、and（百位和十位或个位间所加连词）、hundred（百）、thousand（千）、million（百万）、billion（美国用法：十亿）、milliad(英国用法：十亿)购成所有的其他基数词。注意：hundred thousand、million等词一般为单数形式。

2．本方法将程序ZH．PRG（见程序一）设计为一带参数的过程。需要转换时仅需将数字作为参数，调用本过程文件，即可在“NE”变量中得到英文数字。

3．将要转换的数字，首先取为字符串，然后采用逐位取字串的方法转为英文数字，这样可以避免转换中的误差。为了加快程序运行速度，程序在调用数据库sz．dbf中的基数词时，均使用快于FIND和SEEK命令的GO命令来快速定位。

4．英语中从后向前数，每三位加一个逗号，而每个逗号间的数除了最末一位的thousand、million、billion等不同外，其余的处理方法都比较类似，故在转换时，程序利用变量“NG”每处理三位阿拉伯数字循环一次，对于千亿以内的12位数，最多循环四次，加快了转换过程。变量“NC”用于确定第二次循环后应加的基数词是thousand，第三次循环后应加的基数词是million，第四次循环后应加的基数词是billion。为了符合英文习惯，程序利用stuff函数对连着的thousand、million、billion做了相应处理，并去掉了最末一位的“，”号，运行示例见图二。

以上程序在SUN386、HP386、GW386主机，AR3240打印机上实现，操作系统为DOS 3.3，汉字系统为CCDOS2.13F。

程　　序

```
C>TYPE ZH.PRG
*阿拉伯数字转换为英文数字
para n
publ ne
set talk off
use sz
ng=str(n, 12)  &&取字符串
ne=""
szl=""
do whil val (ng) >0
szs=right(ng, 3)
ng=iif(len(ng)>=3, subs(ng, 1, len(ng)-3), len(ng))
nc=len(trim(str(n, 12)))-len(ng)
szt=subs(szs, 2, 2)
do case
  case val(szt)=0
    szl=""
  case val(szt)>=1.and.val(szt)<10
    go val(szt)
    szl=trim(gw)
```

```
    case val(szt)=10
      go 1
      szl=trim(sw1)
    case val(szt)>10.and.val(szt)<20
      go val(szt)-10
      szl=trim(sw)
  case val(szt)>=20
    if val(subs(szt, 2, 1))#0
      go val(subs(szt, 2, 1))
      szl=trim(gw)
      go val(subs(szt, 1, 1))
      szl=trim(sw1)+'-'+szl
  else
    go val(subs(szt, 1, 1))
    szl=trim(sw1)+szl
    endi
  endcase
  if val(szs)>=100
    go val(subs(szs, 1, 1))
     szl=iif(val(subs(szs,2,2))#0,trim(gw)+"hundred and "+szl,trim(gw)+"hundred"+szl)
  endi
    ne=iif(nc=0.or.nc=3.or.nc=6, iif(nc=6,szl+" thousand, "+ne,szl), ;
      iif(nc=9,szl+" million, "+ne,szl+" billion, "+ne))          &&续上行
enddo
ne=iif(at("million, thousand", ne)#0,stuff(ne,at("million, thousand",ne)+8,11,""), ne)
ne=iif(at("billion, million", ne)#0,stuff(ne,at("billion, million", ne)+8, 10, ""), ne)
ne=iif(rightr(tim(ne), 1) =", ", subs (ne, 1, len (trim (ne))-1), ne)
use
? ne
retu
```

图一

数据库文件名：C:\FOX\SZ.DBF
文件记录总数：　　9
最后更新日期：11/03/91

序号	字段名	类型	宽度	小数
1	GW	字符	8	
2	SW	字符	10	
3	SW1	字符	10	
**	合计 **		29	

记录号	GW	SW	SW1
1	one	eleven	ten
2	two	tweven	twenty
3	three	thirteen	thirty
4	four	fourteen	forty
5	five	fifteen	fifty

```
6  six        sixteen  sixty
7  seven      seventeen  seventy
8  eight      eighteen   eighty
9  nine       nineteen   ninety
```

图二：

阿拉伯数字	英文数字
100	one hundred
9743	nine thousand, seven hundred and fifty-three
57453	fifty-seven thousand, four hundred and fifty-three
100000	one hundred thousand
733021	seven hundred and sixty-eight thousand, nine hundred and twenty-one
500000000	five hundred million
50000000000	fifty billion
500000000001	five hundred billion,one

应用7：如何将小写金额自动转换成大写金额

我们在用Foxbase+进行财务管理时，通常需要将小写金额转换成大写金额。由于Foxbase＋没有提供直接转换的函数，因此，过去都用手工来输入大写金额的，这给使用者带来了诸多不便。这里，笔者用Foxbase＋设计了一个小程序，解决了将小写金额自动转换成大写金额这一问题。经试验，准确无误，效果很好，给使用者带来方便。程序清单如下：

```
TYPE DX. PRG
SET TALK OFF
CLEAR
INPUT "小写金额：" TO SJ
SJ=TRIM (STR (SJ, 13) )
DO WHILE ASC (SJ) =32
   SJ=SUBS (SJ, 2)
ENDDO
Z1= "分角元拾百千万拾百千亿拾百"
Z2= "零壹贰叁肆伍陆柒捌玖"
L=LEN (SJ)
K=0
DXJE= "整"
DO WHILE L>0
   K=K+1
   M=K*2-1
   B1=SUBS (SJ, L, 1)
   N=VAL (B1)*2+1
   B2=SUBS (Z2, N, 2)
   B3=SUBS (Z1, M, 2)
   DXJE=B2+B3+DXJE
   L=L-1
```


此为试读，需要完整PDF请访问：www.ertongbook.com