FOXBASE+和 DBASEIII 的异同及其应用

江西拖拉机发动机厂　　黄焕如

摘要:*FOXBASE+是目前得到广泛应用的数据库管理软件，其中有不少应用软件是由 DBASEIII 过渡到 FOXBASE+的，虽然许多教科书声称 FOXBASE+T DBASEIII 有百分之百兼容，实际上两者之间仍然存在一些差别。本文以具体示例列举了一些鲜为人知的异同点及其应用，可供读者参考。*

DBASEIII, FOXBASE+都是目前 MIS 信息管理系统中应用最广泛的软件之一，由于 DBASEIII 绝大部分命令和函数都能准确无误地在 FOXBASE+下运行，某程序中断运行。笔者在实践中发现这些差异在 CHR()函数，SUBSTR()函数，备份文件或临时文件，同名字段的默任字段，数据库文件中的数据库头结束符等方面比较突出，如何克服和利用这些异同点是很重要的。

一、CHR( )函数

在实际工作中，经常会碰到字符串分段索引的问题。例如需要对数据库 LJ。DBF 中字段(BH, C, 8)分段排序，该字段前三位是量具类代码，第四位是分隔符"-"，后三位是同类量具号，假定数据库内原排列顺序和要求重新排列的结果如下：

原排列顺序		
RECORD#	BH	MC
1	013 -106	分厘卡
2	015 -9	卡尺
3	015 -1	卡尺
4	013 -12	分厘卡
5	016 -301	百分表
6	016 -3	百分表
7	015 -12	卡尺
8	016 bL23	百分表

重新排列的结果		
RECORD#	BH	MC
4	013 -12	分厘卡
1	013 -106	分厘卡
3	015 -1	卡尺
2	015 -9	卡尺
7	015 -12	卡尺
6	016 -3	百分表
8	016 -23	百分表
5	016 -301	百分表

一般在关键字中用 CHR()函数，可一次索引即可快速地完成。

```
USE LJ
INDE ON CHR ( SRBS( BH, 1 , 3 ) ) )+CHR
(VAL(SUBS(BH, 5 , 3 )))TO LJ
```

但是，FOXBASE+和 DBASEIII 在 INDEX 命令中关键字的使用有一点小区别，主要表现在 CHR()函数的取值范围。如果 CHR()函数的取值范围超过了 255 (上例中第五条记录 016 -301)，在 DBASEIII 下，如果设置 SET TALK ON，屏幕上出现如下提示：

* * * 执行错误，位于：CHR()：超出范围

8 个记录已索引

实际上并不影响索引的结果，而如果设置 SET TALK OFF(程序中通常设置)，则屏幕上没有任何揭示，正常运行，索引正确。

在 FOXBASE+下，无论如何设置 SET TALK，均中断运行并且屏幕上出现如下提示：

0 生成关键字

非法函数参数值，类型，或者个数

如果希望上述程序能在 DBASEIII 和 FOXBASE+下都能通过，可不使用字符转换，直接利用数字变换相加，也能一次完成这类复杂排序，由于索引关键字是数字累加值，执行速度相当快。其原理是将设备类代码转化成扩大 1000 倍(10 乘以同类设备序号位数)的数值，加上转换成数值的同类设备序号作为复合关键字索引。

INDE ON VAL (SUBS(BH,1,C)+'000')+VAL(SUBS(BH,5,3))TO<文件名>

二、SUBSTR( )函数

SUBSTR( )函数是字符串函数中常用的函数之一，其格式如下：SUBSTR(<字符串>,<起始位置>[,<字符个数>])。例如，下列程序从任意取出的汉字信息MIS(其范围为 1 -40 个汉字)显示在屏幕上指定位置(程序 1)。

```
*程序 1
*MIS 为任意取出的 1 -40 个汉字，如：
MIS="江拖发动机厂"

I=1
J=0
DO WHILE I<=80
@2,J SAY SUBS(MIS,I,2)
I=I+2
J=J+S
ENDDO
```

```
*程序 2
*MIS 为任意取出的 1 -40 个汉字，如：
MIS="江拖发动机厂"
LE=LEN(MIS)
I=1
J=0
DO WHILE I<=80。AND。LE>I
2,J SAY SUBS(MIS,I,2)
I=I+2
J=J+S
ENDDO
```

但是，FOXBASE+和 DBASEIII 在 SUBSTR()函数的<起始位置>的取值范围上有所区别。如果<起始位置>大于整个字符串的长度时，例如 MIS="江拖发动机厂"5 个汉字，远小于 40 个汉字，在 DBASEIII 下，如果设置 SET TALK ON，屏幕上出现如下提示：

* * *执行错误，位于：SUBSTR()：起点超出范围

实际上并不影响显示的结果，而如果设置 SET TALK OFF(程序中通常设置)，则屏幕上没有任何提示，正常运行，显示正确。

在 FOXBASE+下，无论如何设置 SET TALK，均中断运行并且屏幕上出现如下提示：

超出串范围

如果希望上述程序能在 DBASEIII 和 FOXBASE+下都能通过，可在程序中加上对<起始位置>取值范围的限定，如程序 2 所示。

三、BAK 文件和 SSS 文件

一般程序员编写程序常用 WORKSTAR，EDLIN 或者 DBASEIII(FOXBASE+)下的 MODIFY COMMAND 命令，而这些软件在编辑程序或文本时，如果存盘将生成一个以 BAK 为扩展名的备份文件。

在 DBASEIII 下，当备份文件发生变化，例如由于某种原因变成了只读文件时，原文件编辑重新存盘后，将生成一个临时文件。而且这个临时文件是一个不断更新的副本，原文件却保持原样不变，而在 FOXBASE+下却不受任何备份或临时文件的属性影响。

因此，在 DBASEIII 下，可利用改变备份和临时文件的属性来保护程序不被修改。其主要原理是：由于在编辑文件存盘时，系统将原文件写入 BAK 为扩展名的备份文件，新文件替换了原文件。如果设法将原文件的 BAK 文件改成只读属性，备份文件无法写入，只好将新文件写入以 SSS 为扩展名的临时文件，原文件依然如旧，达到了保护文件不被修改的目的。具体方法如下：

方法一：制造一个原文件的备份文件，设置为只读属性。当编辑原文件并存盘时，新文件将摆在临时文件内，而原文件始终是原样。

例如：假设原文件名为 ABC-PRG

C>COPY CON ABC。BAK(回车、^Z、回车两字节的空文件)

C>CM! ABC。BAK(利用 2 -13 系统的 CM 命令设置为只读文件，当然也可用其他软件或方法设置属性。)

这样无论是使用 WS、EDLIN，还是 DBASE 下的 MODI COMM 命令，编辑 ABC。PRG 文件，新的副本均在 ABC-SSS 临时文件内，ABC-PRG 却原封不动。

方法二：制造原文件的备份和临时文件各一个，设置

为隐藏属性。使编辑软件无法修改该文件。

这样当使用 WS 软件时，能调入并可编辑，一但存盘则退出并显示信息：

* * * FATAL ERROR F27: DISK DIRECTORY FULL

当使用 EDLIN 软件时，不能调入并退出，显示信息：

NO ROOM IN DIRECTORY FOR FILE

当使用 DBASEIII 下 MODI COMM 命令时，不能打开该文件，并提示"文件是不可访问的"。

上述方法可引用到其它应用软件中需要利用备份或临时文件的地方。例如：DBASEIII 中 SORT 命令的使用。该命令将建立一个以某一字段为关键字的重新排序文件，在完成排序的过程中，其先后建立两个临时文件（新名-W44、新名-T44），在排序完成之后改名为 DBF 文件然后删除。因此可以信效以上办法，制造这两个临时文件并设置为只读属性（实际上可仅设置其中-T44即可），使排序工会我法完成。

在 DBASEIII 下执行 SORT 命令无效，并提示"文件是不可访问的"或"文件是不可索引的"。

四、同名字段的默认选择

无论是 DBASEIII 还是 FOXBASE+的数据库都不能含有相同的字段名，但是在某些特殊的应用中，需要在同一数据库中设置相同名字的字段，而同名字段的默认选择是不相同的。为了容易说明，首先建立一简单数据库 LS。DBF，并输入几条记录：

.DISPLAY STRUCTURE

数据库结构：E: LS.DBF .USE LS

数据记录数： 3 .LEST

最新更改日期 ：09/26/92

字段	字段名	类型	宽度	小数
1	JS1	数值	6	2
2	JG	数值	6	2
3	JS3	数值	6	2
* * 总和 * *			19	

记录号#	JS1	JG	JS3
1	1.11	0.00	3.33
2	2.22	0.00	4.44
3	3.33	0.00	5.55

然后利用 DEBUG 或 PCTOOLS 等工具软件，将字段名 JS3 改成 JS1，这样该数据库具有两个相同的字段名 JS1。

.! DEBUG LS。DBF

.E162 31

.Q

对于相同字段名，DBASEIII 和 FOXBASE+是如何处理默认的字段，以 REPLACE 命令为例，略见一斑。

FOXBASE+:

.USE LS

.REPL ALL JG WITH JS1

.LIST

记录号#	JS1	JG	JS1
1	1.11	3.33	3.33
2	2.22	4.44	4.44
3	3.33	5.55	5.55

DBASEIII

.USE LS

.REPL ALL JG WITH JS1

.LIST

记录号:	JS1	JG	JS1
1	1.11	1.11	3.33
2	2.22	2.22	4.44
3	3.33	3.33	5.55

读者不难看出，FOXBASE+取靠最后过的同名字段为其默认字段，实际参与计算或替换，而 DBASEIII 取靠最前边的同名字段为其默认字段，忽略了其后面的字段。在实际应用中，如果限定某应用软件仅能在 DBASEIII 或者仅能在 FOXBASE+下使用，利用上述差异是很容易实现，例如：

```
USE LS
REPLJG WITH JS1
IF JG = 3.33
  ?"FOXBASE+"&&或作一步处理
ELSE
  ?"DBASEIII"&&或作进一步处理
ENDIF
```

五、数据库文件头结束符

FOXBASE+和 DBASEIII 数据库文件的内部结构存在着一个小小的差异，对于数据库头结束符，前者为 ODH，后者为 ODH 和 OOH。

假设在 DBSEIII 下任意建立一个数据库 LS。

DBF,利用 DEBUG 查看数据库头以及数据库头的结束符如下:

```
C>DEBUG LS.DBF
-D100 11F
43E7:0100  03 50 01 01 2B 00 00 00-E2 05 5A 01 00 00 00 00   .P..+...b.Z.....
43E7:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-D6D0 6EF
43E7:06D0  08 02 00 00 01 00 00 00-00 00 00 00 00 00 00 00   ................
43E7:06E0  0D 00 20 39 32 2E 32 20-20 39 39 28 B9 A4 29 BD   .. 92.2  99(工)金
```

不难看出,偏移地址 06E0 处 0 DH、00H 就是数据库头的结束符。如果在 FOXBASE+下将该数据库调入内存,执行 MODI STRU 命令,不作任何修改按回车键存盘,退到 DOS 下利用 DIR 命令查该数据库文件的字节数,将发现该文件比原文件少了一个字节。然后再使用 DEBUG 查看如下:

```
-D100 11F
09D8:0100  03 50 01 01 2B 00 00 00-E1 05 5A 01 00 00 00 00   .P..+...a.Z.....
09D8:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-D6D0 6EF
09D8:06D0  08 02 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
09D8:06E0  0D 20 39 32 2E 32 20 20-39 39 28 B9 A4 29 BD F0   . 92.2  99(工)金
```

可以看出,偏移地址 06E0 处 ODH 就是数据库头的结束符,而 00H 被省略了,同时也修改了文件头的第 9、10 字节(数据库头长度),由原来的 E2H 、05H(1506 字节)修改成 E1H、05H(1505 字节)。正因为这两者同时被修改,无论是 FOXBASE+,还是 DBASEIII 的数据库都是互相兼容的,这也是一般用户不容易发现这微小差异的原因。

值得注意的是,虽然这点小差异对于数据库在 FOXBASE+或 DBASEIII 下运行绝对不会发生任何问题,但是在某些利用高级语言处理数据库的程序中,如果没有结合数据库头参数来为实际数据定位,这两种数据库在使用上就不兼容 ,甚至会发生严重的数据错位后果。

因此,笔者利用 C 语言编制了一个小程序,用来鉴别数据库究竟是在 DBASEIII 下建立的,还是 FOXBASE+的数据库。

```
/ * DBFJB,C 数据库鉴别 * /
/ * 调用: 1.C > DBFJB < 文件名 > * /
/ *       2..RUN DBFJB < 文件名 > * /
/ * TURBO C VER 2 ,00 * /
#INCLUDE"STDIO, H"
MAIN(ARGC, ARGV)
INT ARGC; CHAR * ARGV[ ];
{
FILE * FP; INT I = 0 ;
LONG F[10], S1 ;
IF(ARGC < 2 )
{PRINTF(格式: DBFJB < 文件名 > / N"); EXIT(1 ); }
IF ((FP = FOPEN(ARGV[1 ],"RB")) = = 0 )
{PRINTF("%S, ARGV[1 ]);
PRINTF("文件不存在! / N");
EXIT(1 ); }
WHILE(I < 10)
{F[I] = FGETC(FP); I++; }
IF(F[0 ]! = OX03&&F[0 ]! = 0 X83)
{PRINTF("非数据库! / N");
EXIT(1 ); }
S1 = F[8 ]+F[9 ] * 256 -1 ; / * 数据库头长度 * / FSEEK(FP,
S1 ,0 );
IF 法(FGETC(FP) = = 0 )
PRINTF("该数据库是 DBASEIII 下建立的! / N);
EXSE
PRINTR("该数据库是 FOXBASE+下建立的! / N);
FCLOSE(FP); / * 关闭文件 * /
}
```