DATA BASE APPLICATION METHOD AND CASE STUDY

数据库技术应用技巧和实例

下册

中国科学院成都计算机应用研究所情报室

数据库技术应用技巧

怎样提高数据库建库、检索、运行速度

怎样巧用数据库技术进行制表打印

数据库用户加密方法

其它数据库应用技巧

数据库技术应用的典型范例

提高具有汉字处理功能程序系统运行速度的编程技术

邮电部成都电缆厂计算机室　杨　厚　生

一、引　言

在微型计算机上，除了结构式高级程序语言PASCAL用得较多外，磁盘BASIC语言也是工程技术人员和软件工作者较普遍使用的语言之一。磁盘BASIC由于它在微机上不断的发展、成熟可靠，现在已形成了功能极强、调试十分方便的扩展型磁盘BASICA。在汉字磁盘操作系统CCDOS支持下，它还具有汉字功能，即C-BASICA。作为它强有力支持的伴侣——与之相匹配的编译系统BASIC Compiler 1.00版本的出现，基本上解决了源程序在解释方式下执行速度慢的不足。它使源程序编译后的目标程序（具有扩展名为EXE的文件）比源程序直接解释执行的速度提高五倍以上。近来，国外又推出了速度更快的所谓QUICK-BASIC的编译系统，提高了对源程序的编译速度和目标模块的优化程度，进一步提高了BASIC程序系统的运行速度。

遗憾的是，由于目前的BASIC编译系统不能有效的支持目标程序打印汉字的功能，以及它与在解释条件下进行汉字打印相比并没有显示出在打印速度方面有明显的优越性。从而使得它的应用受到了限制，甚至不如解释执行更好一些。例如，对长城0520C-H微机上配置的3070C打印机汉字打印支持模块（驱动程序）3070C.EXE来说，在解释条件下可支持它打印全部九种字型格式。而在编译后的目标程序下运行，却有五种字符不能打印。这至少是目前编译系统的缺欠。另外，汉字打印基本上是按图形方式进行，因此速度很慢。对目标程序和源程序均如此。甚至源程序的解释执行还能支持一种打印速度高一倍的P型汉字字型格式的打印（即如同一般西文字符的打印速度）。这样看来，在汉字打印输出方面，源程序的解释执行还优于目标程序。

但是，作为大量的、需要长时间进行的数学运算的主程序与输入输出联合在一块进行解释执行，这也是不可取的。因为它有两大缺点：其一是整个程序系统运行速度很慢。其二是程序长度（包括相当多的DATA数据区）和运算变量个数均受到在解释功能条件下内存容量小的限制。以目前较为流行的微机IBM-PC/XT和长城0520C-H来说，在CCDOS或CCBIOS（2.00版本）操作系统支持下的C-BASICA所开辟的可用内存空间不超过60KB（近似数）。这后一个缺点也是无法容忍的。对超过60KB的大程序，只能采用内存复盖技术来处理。即将大程序分成若干个小于60KB长度的小模块，然后依次用人工调入内存的解释空间内执行。操作繁杂，速度更慢。并且没有充分的利用60KB以外的更大的内存空间（如512KB）。但后面即将谈到内存复盖原理将在另一场合下应用是有益的。

为了克服以上两个缺点，笔者曾在IBM PC/XT微机上开发一项电缆模具设计计算工

程的应用软件中成功地克服了这个问题，收到了较好的效果。但笔者还认为，本文所论述的编程方法可作为在微机上开发工程计算软件的一种普遍设计技术。

二、基本思想方法

解决这个问题的基本思想是源程序与编译程序相结合；内存复盖技术与DOS批命令程序AUTOEXEC.BAT的用法相结合。具体来说，使用如下编程方法加以实现。

对前一缺点，我们将源程序分成三个模块。将具有人机对话和汉字输入的屏幕编辑形式之程序作为第一个模块——输入模块，将具有汉字处理并打印汉字的输出表格（包括大量运算的数据结果）之程序作为第三个模块—输出模块；将长时间进行运算之程序作为第二个模块—运算模块。显然第一、第三这两个模块由于具有人机对话的程序中断形式和外部设备—打印机的慢速打印而使这两个模块运行极慢。因此我们充分利用它处理汉字的功能及操作的方便性而保留它解释执行的功能。即这两个模块不进行编译，仍为解释执行模块。而第二个模块由于运算时间长，故需要高速地进行。因此应将其编译成可执行的目标模块。然后利用DOS操作系统中功能极强的批命令程序AUTOEXEC.BAT将第一、第二和第三个模块连接起来自动地执行。从而提高了整个程序系统的运行速度。

对后一缺点，我们用内存复盖技术、编译技术和批命令AUTOEXEC.BAT的用法三者结合起来解决。

不失一般性，我们假定源程序中的运算模块比第一和第三个模块大得多，并且程序长度超过解释条件下可用内存空间60KB（自然小于512KB）。这时，只要将第二个模块分成若干个小于60KB的子模块，然后在DOS系统下把它们再次合并成一个大模块。此时再将其进行编译，从而得到具有扩展名为EXE的可执行目标文件。最后用DOS的批命令AUTOEXEC.BAT将第一、第二和第三个模块连接起来自动地连续地执行。这样就提高了整个程序系统的运行速度。

三、具体作法

本文所论述的编程方法将结合微机IBM PC／XT或长城0520C-H(RAM均512KB)来加以具体说明。其作法步骤如下：

1. 建立解释执行的输入模块（第一模块）。取名TT1.BAS。

（1）建立具有人机对话的屏幕编辑方式汉字输入功能的输入程序。若以Ni表示程序行号，可以使用下列语句序列：

```
N1 LOCATE I1,J1 : INPUT "〈汉语变量名1〉" ; X1S
N2 LOCATE I2,J2 : INPUT "〈汉语变量名〉2" ; X2 S
. . . . . .
Nn LOCATE In, Jn : INPUT "〈汉语变量名n〉" ; Xn
. . . . . .
. . . . . 其他语句序列
. . . . . .
```

（2）紧接着建立（最好在硬盘上）随机文件，以便存入输入变量X1 S, ……, XnS。

. . .

倘若以Mi表示程序行号。写出以下程序块，从而构成完整的模块：

M_1 OPEN "C：AAA" AS#1 LEN-L_1

M_2 FIELD#1, R_1 AS Y1$, …, K_n AS YnS

M LSET Y1$ X1$:…:LSET Yn$ Xn$

M_4（若有数字变量，须写出转化为字符型语句）

M_5 PUT#1

. . . . . .

. . . . . 其他语句序列

. . . . . .

M_m SYSTEM

2. 建立解释执行的运算模块（假定其长度小于60KB），取名TT2.BAS。待调试成功后再用编译系统转化为可执行目标模块（第二模块）。取名TT2.EXE。

（1）建立访问随机文件"C：AAA"的源程序作为运算模块之开端，以便从输入模块中取数，提供运算模块的需要。

以Wi表示程序行号，写出以下程序块：

W_1 OPEN "C：AAA" AS#1 LEN=L_1

W_2 FIELD#1, K_1 AS Y1$, …, K_n AS Yn$

W GET#1

W_4（若有字符型变量，须写出转化为数字型语句）

. . . . . .

. . . . . 其他语句序列（运算部分）

. . . . . .

W_4以下的运算部分程序是运算模块的主部。并假定主部运算结果的数据保留在Z_1…，Z_r变量（工作单元）中。

（2）在运算模块主部末端紧接着建立随机文件"C：BBB"，以便保留运算后的输出结果。从而提供最后一个模块—第三模块打印汉字和数字的输出表格形式。以Vi表示程序行号，程序块格式如下：

V_1 OPEN "C：BBB" AS#1 LEN=L_2

V_2 FIELD#1, S_1 AS P1$, …, S AS Pr$

V_3 ISET P1$=Z1$:…:LSET Pr$=Zr$

V_4（若有数字变量，须写出转化为字符型语句）

V_5 PUT#1

. . . . . .

. . . . . 其他语句序列

. . . . . .

由（1）和（2）两项形成了解释执行模块TT2.BAS。

（3）对TT2.BAS模块用BASIC编译系统（BASCOM. COM, BASCOM. LIB, BASRUN. EXE, BASRUN. LIB和LINK. EXE）将其编译成可执行的，快速运行的目标程序，即第二模块TT2. EXE。它的运行将与小空间60KB无关。

对于大于60KB的TT2.BAS，我们用内存复盖原理将其分成若干个子模块，

TT2(1).BAS, TT2(2).BAS, …, TT2(n).BAS

其中每个模块的长度均小于60KB，并在对每一模块TT2(i).BAS（1≤i≤n）的编程中要注意到模块间输入与输出之间数据交换（信息接口）的正确性，为以后子模块的连接作准备。同时要保证各子模块的相对独立性、完整性，并严格满足编译条件的要求。最后，将每个子模块依次调入60KB空间内解释执行，通过细致的调试来进一步保证各子模块功能的正确性。

为了进一步减少访问磁盘的次数，加快数据存取的速度，并充分利用512KB的内存空间，我们又需将这些调试好的子模块TT2(i).BAS重新按顺序合并成一个较大的模块——仍取名为TT2.BAS。这显然不能在BASICA开辟的空间内进行合并，而必须在CCDOS（或CCBIOS）操作系统下进行。即用COPY命令完成：

COPY TT2(1).BAS+……+TT2(n).BAS TT2.BAS从而得到

$$TT2.BAS=\begin{cases}TT2(1).BAS\\TT2(2).BAS\\\cdots\\TT(n).BAS\end{cases}$$

然后对大模块TT2.BAS进行编译，生成可执行的目标模块TT2.EXE。它的运行将不再受到小空间60KB的限制。

3. 用建立第一模块的方法建立起解释执行的具有汉字功能的输出打印模块，即第三模块TT3.BAS。若以Qi表示程序行号，则此程序块格式如下：

Q_1 OPEN "C：BBB" AS#1 LEN-L_2

Q_2 FIELD#1, S1 AS P1$, …, Sr AS Pr$

Q_3 GET#1

Q_4（若有字符型变量，须写出转化为数字型语句）

……其他语句序列

Q_t SYSTEM

4. 最后在DOS系统下用行编辑命令EDLIN将第一、第二、第三模块之名称（文件名）依次写入批命令AUTOEXEC.BAT中。考虑到所论程序的汉字功能AUTOEXEC.BAT的前半部执行命令应包括这样三个文件名：FILE1.EXE，CCCC.EXE和选定的汉字打印支持模块（如ALL9P.EXE）。其扩展名（EXE）亦可省略。

于是批命令AUTOEXEC.BAT应具有如下程序格式（存在软盘中）：

C：FILE1

C：CCCC

C：ALL9P（或其他汉字打印支持模块）——系统支持程序

BASICA TT1

TT2

BASICA TT3——应用程序

此时，软盘应具备以下文件：

COMMAND.COM

BASICA.COM
AUTOEXEC.BAT
BASRUN.EXE
TT1.BAS
TT2.EXE
TT3.BAS

硬盘中应有汉字系统文件：CCLIB，FILE1.EXE，CCCC.EXE和汉字打印模块（驱动程序）ALL9P.EXE（或其他打印支持模块）。

这是一种运行速度较快的程序系统。它除了按需要在输入输出因人机对话而中断外整个系统可自行启动并从头到尾自动地不停顿地运行，直到打印结果。

四、小　结

在开发微机应用软件的实践中，按上述编程方法所形成的程序系统来运行，效果较好。系统能快速运行体现了以下几个因素。

1．将长时间运行的主要运算模块进行编译，作为高速的目标程序来执行。

2．在汉字输出方面，适当地保留了解释执行程序，能有效的支持某种快速的打印输出（如GW0520C-II的P型字符打印）。

3．将若干小模块合并成大模块，减少了文件名，从而减少了程序运行时访问外存（软盘或硬盘）的次数。

4．保留运算中间结果和输出结果的方式尽量采用建立随机文件，而不采用顺序文件存取的方式。

5．尽量在硬盘上存放随机文件和系统支持程序。这样，数据存取速度较软盘为快。

6．所形成的执行文件通过功能极强的DOS批命令AUTOEXEC.BAT来连续地、自动地执行，从而尽量减少了程序中断和人工干预。

文献出处：《微小型计算机开发与应用》1987年2期　16～19页

一个实用的快速编辑程序

翟　明　德

下面的BASIC程序对于长的程序清单来说，是一个很有用的实用程序。它可以用一个字符串取代程序清单中所有需要替换的字符串。例如，将某一变量名PR改为PROGRAM，若用手工的方法来作这件事是很头痛的，因为只要漏掉其中一个PR变量未被修改，程序在下次运行时就可能不通过，而用此程序进行批量快速替换，既准确又迅速。有了这个程序，可以在设计原程序时用短的变量名，特别是对于次数频繁的变量名，这样可以缩短键盘输入

的时间。当程序通过后，为了使程序可读性好，可采用长的变量名或汉字变量名来替换。

使用本程序之前，要编辑的文件必须以ASCII码的格式存盘，切记！程序运行后，当按屏幕提示键入要编辑的文件名和替换与被替换的字符串后，自动产生一个后缀为"XRF"的文件，而原文件保留不变。当改变新文件的后缀并运行通过后，可用新文件代替原文件。

5，字符替换

```
10 CLEAR:SCREEN 0,0:WIDTH 80:KEY OFF:CLS
20 OPTION BASE 1:DIM P$(999)
30 INPUT "请输入文件名"; FILE$
40 OPEN FILE$ FOR INPUT AS #1
50 WHILE NOT EOF(1)
60 COUNT=COUNT+1
70 LINE INPUT #1,P$(COUNT)
80 WEND
90 CLOSE #1
100 FILE$=LEFT$(FILE$,INSTR(FILE$,"."))+"XRF"
110 INPUT "键入原字串"; SEARCH$
120 INPUT "替换成"; RP$
130 FOR I=1 TO COUNT
140 PTR=INSTR(P$(I), SEARCH$)
150 IF PTR=0 THEN 200
160 ZCS=RIGHT$(PS(I),LEN(P$(I))-PTR-LEN(SEARCH$)+1)
170 P$(I)=LEFT$(P$(I),PTR-1)+RP$+ZC$
180 PRINT P$(I)
190 GOTO 140
200 NEXT 1
210 OPEN FILE$ FOR OUTPUT AS #1
220 FOR I=1 TO COUNT
230 PRINT #1, P$(I)
240 NEXT I
250 CLOSE #1
260 END
```

文献出处：《计算机世界》1988年3月23日34版

用DBASE语言编制专业词组快速输入程序

河北医学院第四医院　刘保玄

目前，电脑被越来越多地应用于处理专项事务。例如，统计报表、仓库管理、病历检索等。这些应用都要输入大量的中文词汇。这些词汇往往数量不多，但专业性强，重复率高。以码代词的办法不是普遍可行。逐字输入中文速度太慢。而一般的通用词组输入法所含专业词组少，补充新的词组比较麻烦。另外，随着词组量的增多，用户要消耗较多时间和精力用于选词。

为了提高专业词组的输入速度，减轻操作人员的劳动强度，我们用DBASE语言编制了一套专业词组单键选择快速输入程序。在IBM—PC微型机上实现了多种专业词组的快速输入。

我们把专业词组的输入分为五类，各用一套相应的程序来实现。我们的基本思想就是把要输入的词组提示在屏幕上，并且标以单字代码。操作人员击代码键选词，该词即自动被输入。对需要不断补充新词的情况，可令一些代码后是空白，备需要时当场补入。

第一类：在程序里写上供用户选择的词组，提示到屏幕上，备选择输入：例如，性别、婚否等字段的输入就可这样处理。程序举例如下：

```
? '性别：男—回车键　女—字母键'
WAIT TO X
IF X=''
REPLACE性别WITH'男'
ELSE
REPLACE性别 WITH' 女'
ENDIF
? '性别：'+性别
```

第二类：把若干常用词组作为内存变量装入内存文件。使用时用RESTOR命令调入内存。可以自由组合来提示这些词组，以供选择输入。例如，把车间、科室的名称以及“车间”、“科室”等常用词作为内存变量就极为方便。这类文件的建立最好编一个小程序来实现，这样便于打印和修改。

例如，建立一个名叫KEM.PRG的程序文件。内容如下：

```
CLEAR
SRORE'内科'TO K1
STORE'外科'TO K2
    :
STORE'科别'TO K
SAVE TO KEM
RETURN
```

它运行的结果，产生KEM.MEM内存变量文件。利用变量文件不仅可以念快词组的输入，而且使程序的编制大为简化，例如，往一条新记录里输入“科别”这一字段。其程序如下：

```
RESTORE FROM KEM
USE〈文件名〉
APPEND BLANK
? '请选择科别：1－&k1 2－&k2
……A－&KA'
WAIT TO X
REPLACE &K WITH K&X
? '&K'+&K
```

第三类：把可能被选的词组作为数据项装入词组库文件。一般在副区里调用，为主区输入词组服务。这类词组库文件的字段名最好采用统一格式，并且不与主区字段重名。例如取名为：Z1、Z2……ZU、ZV。

对于可选词不超过32条的字段，例如，民族、职业、学历等字段，每项内容的可选词组占一条记录。主区该输什么字段，副区就定位在对应的记录上。屏幕按某一固定格式提示该记录里的词组，供选择输入。

库序举例如下：

```
副区调词组文件，选主区，  )
STORE'民族'TO M1
STORE'职业'TO M2
STORE'学历'TO M3
STORE' 1'TO Y
DO WHILE Y 〈 )'4'
ERASE
@0，0 SAY'请选'+M&V
@1，0 SAY' 1－'+Z1+' 2－'+Z2+' 3－'+Z3+' 4－'+Z4
@2，0 SAY' 5－'+Z5+' 6－'+Z6+' 7－'+Z7+' 8－'+Z8
@3，0 SAY'9－'+Z9+' 0－'+Z0+'A－'+ZA+'B－' +ZB
@4，0 SAY'C－' +ZC +'D－'+ZD+'E－'+ZE+'F－' + ZF
@5，0 SAY'G－'+ZG +'H－'+ZH+'I－' +ZI +'J－' +ZJ
@6，0 SAY'K－'+ZK +'L－' +ZL +'M－'+ZM+'N－'+ZN
@7，0 SAY'O－'+ZO +'P－' +ZP +'Q－' +ZQ+'R－'+ZR
@8，0 SAY'S－'+ZS +'T－' +ZT +'U－' +ZU+'V－'+ZV
WAIT TO X
STORE; (X) TO X
IF X=' '. OR.X='W'. OR.X='X'.OR.X='Y'.OR. X='Z'
LOOP
ENDIF
REPLACE &M&V WAIT Z&X
```

```
STORE STR(VAL(Y)+1, 1)TO Y
SELECT SECONDARY
SKIP
SELECT PRIMARY
ENDDO
```

以上两类词组的输入，每词一键。

第四类：原理与第三类相同。适用于可选词几十条到上百条，并且以顺序查找方式选词的情况。这些词组按序装在数条记录之内。

程序举例如下：

```
(主、副区打开文件，定位记录，选副区)
STORE' ' TO X
DO WHILE X=' '.OR.X='Z'.OR.X='W'.OR.X='X'.OR.X='Y'
ERASE
@0, 0 SAY '请选择。(打回车接下页，打Z键返前页)'
@1, 0 SAY '1-'+Z1+'2-'+Z2+……
@2, 0 SAY '5-'+Z5+'6-'+Z6+……
   :
@8, 0 SAY 'S-'+ZS +'T-'+ZT+……
WAIT TO X
IF X=' '
SKIP
ENDIF
IF X='Z'
SKIP-1
ENDIF
ENDDO
SELECT PRIMARY
REPLACE(字段名)WITH Z&X
```

第五类：用于词汇量大，需要按树型结构查找的情况。例如，可把词组分为32类，每类可装32条词组。第一条记录装类别提示。第一类词在第二条记录里，第二类词在第三条记录里，依次类推。

程序举例如下：

```
(打开主、副区文件。主区定位在要输入的记录上。选副区，定位于第一条记录。)
STORE T TO H2
DO WHILE H2
ERASE
@0, 0 SAY'请选择(打回车键返类别提示)'
@1, 0 SAY'1-'+Z1+'2-'+Z2+……
@2, 0 SAY'5-'+Z5+'6-'+Z6+……,
```

```
@8，0 SAY'S—'+ZS+'T—'+ZT+……
WAIT TO X
STORE! (X) TO X
IF X=' '
GO 1
LOOP
ENDIF
IF #=1
DO CASE
CASE X='0'
GO 1 1
CASE VAL(X)>0
GO &X+1
CASE@(X, 'ABCDEFGHIJKLMNOPQRSTUV')>0
GO@(X, 'ABCDEFGHIJKLMNOPQRSTUV')+11
OTHERWISE
GO 1
ENDCASE
ELSE
STORE F TO H2
ENDIF
ENDDO
SELECT PRIMARY
REPLACE(字段名) WITH Z&X
```

词组库文件里的词不必在建库时全都装入，留出空白待工作中补充可能更好。已输入的词有些也需要随时修改。可用下述方法实现。

首先在第一提示行的“请选择……”一句里加上“打W键补充改写”七个字。再在“WAIT TO X”一句之后，插入下面一段程序：

```
STORE! (X)TO X
IF X='W'
STORE ' ' TO Y
@0，0 SAY'请选改写词代号:
'GET Y
READ
@0，0SAY'请补充或修改:
'GET Z&Y
READ
LOOP
```

```
ENDIF
```

这种方式对于第三、四、五类词组的补充、修改都适用。

另外，为改正选词击键失误所造成的错误，可设一个外层循环程序，以便重输。

程序举例如下：

```
STORE T TO H1
DO WHILE H1
(选词程序)
(显示所选的词)
?'认可—回车键  重输—字母'
WAIT TO X
IF X=' '
STORE F TO H1
ENDIF
ENDDO
```

总之，灵活运用这些方法，可以有效地解决多种专业词组的快速输入问题。它的程序简单，运行速度快，补充词组和纠错都方便。如果程序严密，一般按键错误都能处理，可使工作的全过程都在屏幕提示下进行。不懂计算机语言，甚至不会输入汉字的人也能上机工作。

使用这种方法的另一个好处是保证输入内容规范化，因此便于检索。

以上所述，以DBASE—Ⅱ为程序语言，其程序思想也适用于DBASE—Ⅲ等其它程序语言。

另外，词组代码用两位或三位均可，只是击键次数多了一些。

文献出处：《计算机应用研究》1986年3期　44～48页

一个实用的提高dBASE程序可读性的程序

上海打字机厂　　胡宗唐

在编制dBASE源程序时，用户希望除了程序的运行速度快以外，还希望程序的可读性好。怎样提高程序的可读性呢？笔者认为，一是程序的思路要清楚，另外程序的书写格式最好能写成锯齿形的结构。即每条语句的起始位置要根据循环条件选择等，按嵌套层次对齐。这种书写格式使人能清楚地看出程序所执行的内容，若程序有错也很容易发现——程序可读性好。同时在编制程序时，有人用大写字母书写，有人用小写字母书写，当一个交流程序由几个人共同编制时，阅读起来很凌乱，对以后的软件维护工作很不利。

针对上述问题，笔者在IBM—PC机上用BASIC语言编制了一个处理程序。运行此程序，只要输入要转换的程序名，就能将按任意格式书写的源程序全部自动转换成用大写字母书写的，而且是按锯齿形结构排列的源程序。同时，当被处理的源程序中DOWHILE…

ENDDO、IF…ELSE…ENDIF、DOCASE…ENDCASE及TEXT …ENDTEXT 嵌套语句出现语法上的错误时，该程序将会提示是哪一类错误及在哪一条语句上出错。

程序清单如下：

```
10 CLS:ON ERROR GOTO 390
20 INPUT "INPUT FILENAME" ; DDASE$
30 CLS
40 OPEN DDASE$ FOR INPUT AS #1
50 OPEN "$$$" FOR OUTPUT AS #2
60 S=0:CASE=0:FCASE=0:TEXT=0:L=1:PRINT "LINE"
70 WHILE NOT EOF(1)
80 LINE INPUT #1,A$:N=LEN(A$)
90 D$=A$:C$=" "
100 FOR I=1 TO N
110 N=ASC(MID$(A$,I,1))
120 IF N>96 AND N<123 THEN C$=C$+CMR$ (N-32) ELSE C$=C$
    +CHR$(N)
130 NEXT I
140 FOR I=1 TO N:IF MID$(C$,I,1)= " " THEN 160
150 C$=MID$(C$,I,255):GOTO 180
160 NEXT I
170 C$= " "
180 IF LEFT$(C$,8)= "DO WHILE" OR LEFT$(C$,2)= "IF"
    THEN B$=SPACE$(S*3+CASE*3)+C$:S=S+1:GOTO 250
190 IF LEFT$(C$,5)="ENDDO"OR LEFT$(C$,5)="ENDIF" THEN
    S=S-1: GOTO 240
200 IF LEFT$(C$,4)="ELSE" THEN B$=SPACE$(S*3-3+CASE*3)
    +C$:GOTO 250
210 IF LEFT$(C$,4)="CASE"AND FCASE=1 THEN B$=SPASE$(S*
    3+CASE*3-5)+C$:GOTO 250
220 IF (LEFT$(C$,4)="CASE" OR LEFT$(C$,5)= "OTHER") AND
    FCASE=O THEN B$=SPACE$(S*3+CASE*3-5)+C$:GOTO 250
230 IF LEFT$(C$,7)="ENDCASE" THEN S=S-1:CASE=CASE-1
240 B$=SPACE$(S*3+CASE*3)+C$
250 FCASE=O
260 IF LEFT$(C$,7)="ENDTEXT"THEN IF TEXT=O THEN PRNT
    "ENDTEXT Without TEXT":GOTO 380 ELSE TEXT= 0 :GOTO
    330
270 IF TEXT=1 THEN B$=D$
280 IF LEFT$(C$,4)="TEXT"THEN TEXT=1:GOTO 300
```

```
290 IF LEFT$(C$,7)="DO CASE" THEN CASE=CASE+1 : FCASE=1:
    S=S+1
300 LOCATE 1, 6 : PRINT L : L=L+1
310 PRINT #2, B$
320 WEND
330 CLOSE
340 IFS>O AND CASE=O THEN"DO WHILE Without ENDDO OR IF
    Without ENDIF" : GOTO 380
350 IF CASE>O THEN PRINT"DO CASE Without ENDCASE" :GOTO
    380
360 IF TEXT=1 THEN PRINT "TEXT Without ENDTEXT" : GOTO 380
370 KILL DDASE$ : NAME"$$$" AS DDASE$
380 END
390 PRINT "ENDDO Without DO WHILE OR ENDIF Without IF OR
    ENDCASE Without DO CASE" : END
400 RESUME
```

文献出处：《计算机世界》1987年7月23日

在DBASE Ⅲ 中加快数据统计的一种方法

福州市经济委员会计算站　　陈仪昌

应用DBASE Ⅲ关系数据库进行数据统计用count、sum计数和求和命令时，通常是打开数据库后，寻找数据指针从数据库中的第一条记录开始一直寻找最后一条记录，对符合条件的记录进行计数并求和。当计算完符合一种条件的数据后，再进行另一个条件数据计算时，寻找数据指针又要从头开始一直寻找到最后一条记录，包括寻找上一次已经统计了的数据，检验它们是否满足条件，从而使统计时间显得太长。

我们利用记录删除命令DELETE〔<作用域>〕〔FOR/WHILE<条件>〕可以解决这一问题，首先将数据库中所有不符合统计条件的记录进行删除，即在数据记录中做上删除标记（•），然后再统计符合条件的记录。在设置有SET DEL ETED ON命令的程序中，count、sum命令就忽略了所有标有删除记号（•）的记录，从而使数据库中的记录数量相对减少，达到缩短统计时间的目的。在统计结束时，为了使数据库中的记录得以恢复，我们用RECALL ALL命令将所有被标有删除记号（•）的记录恢复正常。为了避免程序在统计过程中非正常中断，我们在程序中将SET ESCAPE设置在OFF状态，使键盘上ESC键无效，从而保护了数据的完整。程序例子如下：

```
SET ESCAPE OFF
```