数据库应用

dBASE Ⅲ模糊检索技术

一般情况下，对数据库进行检索时需要输入完整的定位条件，有时因输入字符较多容易出错，且效率低。例如：要检索题目为《模拟计算机和数字计算机及其系统》的刊物时这就要求用户一字不差的输入检索定位刊物名称，否则便会检索失败，有时因记不清楚或遗忘便无法进行检索。

为了解决上述dBASE Ⅲ检索数据库中存在的问题，就需要寻找新的检索方法，我们经过实践摸索找到了一种方法，即模糊检索方法。

所谓模糊检索，是指用户输入不完整或一些词语作为检索条件，然后检索出库中满足给定条件的一些记录，再从这些记录中得到所需找的记录。

实现模糊检索的指导思想是：分解检索长词组为短小词语，或者再用逻辑符(AND, OR)连接组合这些词语作为检索定位条件进行检索定位。通过 dBASE Ⅲ的字符运算函数 $ 来实现。

设A，B变量赋值如下：

```
A="abcdefgh"
B="cdef"
```

则字符运算函数 $ 运算结果：

```
.? A$B
.F.
.? B$A
.T.
```

字符运算函数 $ 首先测试前一变量的字符是否包含在后面变量的字符中，若包含(成立)运算结果得到逻辑真 (T)值，否则得到逻辑假(F)值。

为进一步说明问题，下面举例。

若有数据库名为 BKK. DBF，其结构如下：

字段名	类型	宽度
BKDH(刊物代号)	C	6
BKM(刊物名称)	C	20
BNJ(价　　格)	N	5.2
* * Total * *		32

若用户欲检索名称为：《无线电电子学与自动化》刊物，采用以下模糊检索程序，便能很快得到检索结果。这一小程序就是采用了模糊检索技术。

```
* *程序：JS88/7/20
set exact off
set talk off
use bkk.dbf
do while.t.
  clear
  store space(10)to bm
  @4,10 say"请输入模糊刊物名称："get bm
  read
  stor trim(bm)to bm
  locate for bm $ BKM
  stor 1 to n, h
  clear
```

```
do while.not. eof( )
  if n>8.and.h = 1
    stor 1 to n
    stor 40 to h
  endif
  if n>8.and.h=40
   wait" 看清后按一键继续…"
    stor 1 to h, n
    clear
  endif
  @n,h say"|"+BKDH+""+BKM+""
    +str(BNJ,5,2)
  stor n+1 to n
  continue
enddo
wait"查看完含有("+bm+")词语的刊物，按键!"
clear
stor space(2)to sf
@4,10 say"是否继续模糊检索?
    是(Y)/否(N)"get sf
read
if upper(sf) ="Y"
  loop
endif
exit
enddo
```

当你运行这一程序，屏幕上出现："请输入模糊刊物名:"这一提示时，你输入"电子"一词语，并按回车键后便会出现如下检索结果：（本文未给出BKK.DBF内容）

BKDH	BKM	BNJ
2-888	电子与电脑	2.76

2-889	电子技术应用	3.00
2-890	电子科学技术	3.30
2-708	无线电电子学与自动化	12.60
22-52	电子工艺技术与自动化	2.70

以上的例子很简单，旨在说明模糊检索技术的应用，还可以推广到多字段的复合条件模糊检索。

（舒　英）

提高统计速度的一种方法

微机关系数据库已广泛应用于各行各业的管理业务中，而且常常要用到统计操作，因此提高统计速度是个很重要的问题。

在人事档案、设备台账等管理业务中，常常要进行有m个横向栏目、n个纵向栏目的统计工作，dBASEⅢ仅给出了一个count命令，如果用该命令统计，就要重复多次。例如我厂人事档案管理系统中的工种与工资级别、学历对照表，横向栏目有14项，纵向栏目有121项，用count命令统计，就要重复14×121＝1694次。对于有3000条记录的数据库，采用这种方法做统计，需要30多小时。因此，必须另求新法。现向大家介绍一种方法：

在固有数据库中加上相应的标志位字段，该标志位字段和横向栏目一一对应，字段为数值型，通过程序实现统计。可以提高效率十几倍。该方法的基本思想在于，用减少访问记录的次数，来提高处理的速度。

统计可根据条件分为两类，第一类的统计条件关系式是一个等式，如工种为"计算机操作员"就是一个定值的关系式；

第2类是统计的条件关系式为不等式，如工龄取一个范围值，即从某一年到某一年。

进行第一类统计时，可以这样实现：首先根据横向栏目条件置标志位，如果条件为真，则相应标志位置"1"，否则置"0"，可以通过一系列的分支语句结合指针移动等命令来实现，然后使用数据库摘要语句按纵向栏目内容对标志位进行摘要，就可以得到所需的统计数据。这里需要将纵向栏目做为关键字对固有数据库进行索引。

进行第2类(m×n)统计时，可以采用以下几步来编程实现：第一根据横向栏目条件给标志位赋值，即当满足条件时标志位置"1"，否则置"0"；第二是根据一个纵向栏目条件产生一个小的中间数据库；第三是对中间数据库的标志位字段求和，并且将结果放到统计数据库中，然后再重复第二步，直至完成所有统计任务。

笔者采用本文所述的方法进行统计，所用时间仅为用count命令统计所用时间的1/14，但提高统计效率的代价是增加了磁盘空间的开销。

(张立华)

如何在程序运行中查询数据库结构

在dBASE程序执行中，有时需要在不中断程序执行的情况下查询当前库结构，此时如果忘记库中的字段名的写法，查询工作就无法进行。现提供一个程序(见程序1)，通过键入"|"符号，便能在第0行依次显示当前库的结构参数，显示完后仍回到原输入处，使用非常方便。

程序中首先判断是否有“|”符号，如有则利用 Copy 命令以当前库拷贝一新的结构库，并在第九区打开，逐字段显示当前库的结构参数，显示后将结构库删除。

使用时只需在主程序中(见 程 序 2) 调 用 ZD-HELP.PRG，便能在屏幕的零行开一窗口，查阅当前的库结构。

程序1

```
*  ZDHELP.PRG   *
*查询当前库的字段  *
PARA PBF
SET TALK OFF
IF AT('|', PBF) = 0
  RETU
ENDI
COPY TO ZDHELP STRU EXTE
SELE 9
USE ZDHELP
XS=' 字段名:        类型:        宽度:
      小数位:       〔任按一键继续〕'
DO WHIL.NOT.EOF( )
  @0,0 say XS
  @0,10 SAY FIELD-NAME
  @0,32  say FIELD-TYPE
  @0,43 say STR(FIELD-LEN,3)
  @0,56 say STR(FIELD-DEC,2)
  SKIP
WAIT''
ENDD
USE
ERAS ZDHELP.DBF
SELE1
```

```
RETU
```

程序2

```
ZDM=''
DO WHIL.T.
  @4,25 SAY'请输入字段名:'GET ZDM
  READ
  IF AT('|', ZDM)=0
    EXIT
  ENDI
  DO ZDHELP WITH ZDM
ENDD
```

（刘秋万）

如何提高在父表中删除或修改记录的效率

一般说来，在王安VS PACE数据库管理系统中，如果一个表是子表，尽管它的数据文件中的记录很多，我们要在其中删除或修改一条记录，CPU的响应时间是很快的，不存在什么问题。但若要在其父表中做这样的操作，CPU的响应时间就显得十分缓慢，不仅占用了大量的机时，而且影响了其他终端上用户的正常工作。如何提高这种操作的效率呢？尽管我们将子表中与父表有关系的字段全部定义为辅关键字，并且对程序也做了相应的优化，但其效果并不十分显著。究其原因，下面我们将结合一个实际的例子加以阐述，并介绍一种解决这个问题的行之有效的办法。

在一个财务管理系统中，科目代码表(下称DMB)是父表，除了其它几个与它有父子关系的，而记录不多的表外，

记帐凭证表(下称PZB)的数据文件中有一万条记录。如果每条记录里有10个科目代码字段与DMB里的科目代码字段是父子关系，那么这两个表之间就存在着20万个关系。由PACE数据库的有关定义可知：表间关系的完整性原则是在Runtime系统支持下自动实施的，它不仅保证了数据库中的数据是可靠有效的，而且控制着相关表中记录的修改和删除。因此，我们要在DMB中修改或删除一条记录时，Runtime系统就要根据上述原则，将PZB中与DMB有关系的20万个科目代码全部扫描一遍(即使PZB里没有一个科目代码与要修改或删除的记录中的科目代码相同)。假如按最好的情况(无其他用户使用终端)估算，即使Runtime系统每秒扫描100个科目代码，也需要花费1000秒(约16分钟)，而实际情况往往更糟。很显然，这是CPU处理速度不高所造成的。在无法提高机器本身性能的情况下，我们可按下列步骤去做，也不失为解决问题的一种办法。假定用户工作库的库名为CW，在REM-DAT卷上，现在我们要修改DMB中的科目代码。例如：将老的科目代码"145007001000"改为新的科目代码145006001000"。

1. 首先进入应用系统的父表DMB中，增加新的科目代码"145006001000"。然后再进入子表PZB中，将所有记录中凡是涉及老的科目代码全部改成新的科目代码。退出应用系统。

2. 进入系统的REMOAT卷上的@CW####A(数据文件)库中，将该库中的数据文件PZB改名为PZBBY。退出系统。

3. 进入CW库的字典中，不做任何修改字典的操作，只要按PF16键退出，将字典重新编译一遍。此时，记帐凭

证表又将产生一个新的，但其中记录为0的数据文件 PZB。

4. 重新进入应用系统的DMB中，删除老的科目代码"145007001000"和其它已经废弃的科目代码。由于与DMB相关的PZB中的记录已经为0，因此，这样的操作可以十分迅速的完成。

5. 退出应用系统，再进入REMDAT卷上的@CW####A库中，将PZB改名为PZBBB(不要将其删除，以留做备用)。最后再将PZBBY改名为PZB。

其实，我们可以模拟上述的第二步和第五步，用过程语言编一个小程序，来完成将数据文件重新命名的操作。另外，有必要重申的是：如果子表PZB中的老科目代码没有全部改成新的科目代码，而父表DMB中的老科目代码已被删除了，这样就会造成子表中某些记录的科目代码成为"孤独"的科目代码。用户在使用这些记录时，将会产生系统出错的信息。

上述问题，是笔者在王安VS-65小型机，PACE数据2.0版本上遇到的。解决的办法是否适合其它VS系列小型机，在此就不深究了，仅供同行参考。

（曲 衷）

提高dBASEⅢ分类统计速度的几种方法

在数据处理中，常要对源数据库进行分类统计操作。所谓分类统计，是指按关键字字段(1个或数个)的取值(分类值)进行分类来对其他数值字段求和。例如对本单位的个人(以工号标识)进行实绩统计就是典型的例子。笔者在编写dBASEⅢ应用程序的实践中发现，针对不同的分类条件，灵

活运用有关命令，可以在不影响数据准确性的前提下成倍提高处理速度。下面是几种常见情况下的改进方法。

1．分类值可以预见时，用UPDATE命令分类更新求和要比常规的索引(排序)和汇总统计方法快2倍左右。其方法是利用预先已创建好的目标库及其索引文件(其关键字字段值的集合包含源库对应关键字字段值的集合)，通过UPDATE命令从设在另一工作区的源库进行更新求和。程序段的通用格式为：

```
SELE 2
USE 源库名
SELE 1
USE 目的库名INDEX索引文件名1 ALIAS别名
REPLALL 求和字段名1 WITH[，……]
GO TOP
UPDATE ON分类关键字字段名FROM别名REPL求和字段名1 WITH求和字段名1＋别名→求和字段名1[，……]
```

利用这种方法，在IBM—PC/XT(640KRAM)上进行过测试，一个有19个字段、1138条记录的源库，按机车车号对油耗、作业时间和行走公里3个字段进行统计，只用了1分26秒，而用TOTAL方法则用了3分12秒，速度提高了2倍多。

2．分类关键字为多个字段时，直接利用索引(排序)和汇总命令分类统计往往会出现差错，所以也采用了多重判别、多重循环、逐条记录处理的方式，这使时间开销过大而影响系统运行效率。通过摸索采用集成关键字字段后，问题迎刃而解。所谓集成关键字字段，就是在源库增加一个宽度为各关键字字段宽度之和的字段。在处理时，以此集成关键字段(实际上也是原来数个关键字段)索引(排序)汇总求和就可以

了。一个实际应用例子是对13个字段的源库，按6个关键字字段分类统计，在记录数为377条的情况下，对2个字段分类统计，原方法用了6分40秒，改用集成关键字字段（宽度为46个字节）后只用了2分07秒（包括集成关键字字段替换时间）。这种方法的通用格式是（如果用SORT命令，则其后要打开排序库）：

```
USE源库
REPL ALL集成字段名WITH主关键字字段名+第1关键字字段名[+……]
INDEXON集成字段名TO索引文件名
TOTAL ON集成字段名TO目的库名
```

需要说明的是，集成关键字字段一般设置为字符型类型，在替换时，对不是字符型的关键字字段要用函数方法转换成字符型。此外，如果集成关键字字段值可以事先预见，则按上述第1种方式，用UPDATE方式处理，效果会更好。

3．分类统计记录数的简便方法：记录数有时隐含着统计数据，例如在运输统计中，一条实绩记录往往代表1车次，因此分类统计记录数也是编程常会遇到的问题。以往采用带筛选条件的COUNT命令进行统计，由于在执行COUNT命令时，要对源库记录全部扫描，结果，记录越多、分类值（即筛选类别）越多，所耗时间越长，令人焦急，后改用索引（排序）及汇总命令处理，情况大有好转。具体方法是：先找一个处理时用不着的数值字段（如果没有就事先设置一个），将段值全赋值为1，然后按关键字字段排序索引。再用汇总命令TOTAL求和，一次可得到全部分类的统计数。在实际中改用这种方法后，处理时间仅为原来方法的1/3～1/5。一般而言，这种场合下，用UPDATE命令处理也是可行的。

从上述几个简单例子可以看出，灵活运用dBASE Ⅲ的各

种统计求和命令，将会带来大幅度提高运行效率的好处。这不但有赖于对dBASE Ⅲ系统的深入了解，在实践中不断探索总结也是很重要的。

（季启发）

dBASE Ⅲ与BASIC互用探讨

用dBASE Ⅲ语言来编写各种事务性管理系统已是司空见惯的事情了，现在几乎所有的事务管理系统都用它来编制。但如何科学地利用各种高级语言之优点参与dBASE Ⅲ的编写工作却是鲜为人知的。用BASIC，FORTRAN，PASCAL及COBOL等高级语言来参与dBASE Ⅲ的编程工作，不但可以提高程序的运算和处理能力，而且可以增强管理系统的灵活性。本文将重点阐述dBASE Ⅲ与BASIC语言的互用问题，而对其它语言不作叙述。现将对dBASE Ⅲ与BASIC语言的互用性进行探讨。

一、用批处理方法实现从BASIC到dBASE Ⅲ的转换

假定运行BASIC文件ZK.BAS后就必须转入运行dBASE Ⅲ中的ZKM.PRG命令文件，这时我们可以先在BASICA(或EDLIN)状态下，在ZK.BAS文件中加入如下语句：

CLS：LOCATE5，6：PRINT“请插入dBASE软盘，”：LOCATE6，18：PRINT“后按RETURN键”：SYSTEM

然后建一个批处理文件ZK.BAT，建法如下：

```
C>COPY CON ZK.BAT
   ECHO OFF
   IF EXIST ZK.BAS BASICA ZK
   ECHO OFF
   IF EXIST ZKM.PRG DBASE ZKM
   ECHO ON
   ^Z(或按F6键)
```

这样每次运行这个系统时，只要在C>状态下敲一下ZK后敲回车键，便可以实现从BASICA状态到dBASEⅢ状态的转变过程。

二、dBASEⅢ状态下运行BASIC程序

只要在dBASEⅢ命令文件中加入如下命令则可：

RUN BASIC文件名. 后缀(若后缀为.BAS，则可省略后缀)。但BASIC文件必须有"SYSTEM"这个语句，才能自动返回dBASE原来所在之状态。

三、数据库互换

在dBASEⅢ中具体可用这个命令：

COPY TO〈系统数据格式文件名〉[范围][FOR〈逻辑表达式〉[FIELD〈数据项名〉][SDF/DELIMITED[WITH〈界限符〉]]

dBASEⅢ与BASIC互用，将对事务管理系统的编写与应用增加很多灵活性，同时还能起到dBASEⅢ及BASICA自身所不能起到的作用。

（杨沐章）

用BASIC调用 dBASEⅢ的数据文件

要用其它语言调用dBASEⅢ的数据文件，必须先了解它的结构。本文先介绍了解dBASEⅢ数据文件结构的一种方法，再介绍调用它的方法。使用的是IBM-PC机。

一、dBASEⅢ数据文件的结构

dBASEⅢ的数据文件由说明和记录内容两部分组成。

1. 说明部分

最前32个字符为总的说明。其中2-4字符为生成或修改文件的年月日，5-7字符为记录总个数，11-12两字符为记录长度。

其后每32个字符说明一个字段。其中1—10字符为字段名称，12字符为字段类型，17字符为字段长度，18字符为小数位数。

说明部分以ASCII码为13和0的字符结束。

2. 记录内容

记录内容按记录的先后，字段的先后依次陆续存放。各字段之间无分隔符，两记录之间，有空格作为分隔符。

最后以ASCII码为26的字符表示文件结束。

运行下列程序即可打印出dBASEⅢ数据文件的说明部分。

打印dBASEⅢ数据文件说明部分的程序：

```
100 INPUT "NAME OF FILE"; NA$
110 NA$ = NA$ + ".DBF": A$ = "####"
```

```
120 OPEN "R",#1,NA$,1:FIELD #1,1 AS G$
130 FOR I=1 TO 16
140 LPRINT USING A$;I;
150 NEXT I
160 J=1:GET #1,1
170 WHILE G$<>CHR$(32)
180 IF J MOD 16=1 THEN LPRINT
190 IF J MOD 32=1 THEN LPRINT
200 LPRINT USING A$;ASC(G$);
210 J=J+1:GET #1,J
220 WEND:LPRINT:LPRINT
230 END
```

二、用BASIC调用dBASEⅢ的数据文件

下列程序是一个通用程序，仅需用户提供文件名。运行时，首先从文件的说明部分取出必要的数据；形成记录长度，各字段的长度等；最后逐个记录地加工，把各个字段分离后输出。程序结构清晰，没有使用GOTO语句。所用变量的意义如下。

L　记录长度

G　每个记录的字段个数

SUM　说明部分长度

A(I)　第I个字段的长度

B(I)　第I个字段第一字符的序数

Q$　记录的前一部分

H$　记录的后一部分

参照此程序，并注意各类数据的存储方式，不难对dBASEⅢ数据文件实现随机存取。

显示dBASEⅢ数据文件字段的程序：

```
100 OPTION BASE 1:DIM A(128),B(128)
110 INPUT "NAME OF FILE";NA$
120 NA$=NA$+".DBF"
130 OPEN "R",#1,NA$,32:FIELD #1,32 AS G$
140 GET #1,1:L=ASC(MID$(G$,12,1))
150 L=L*256+ASC(MID$(G$,11,1))
160 I=1:S$="1":B(1)=2
170 WHILE S$<>CHR$(13)
180     I=I+1:GET #1,I:S$=MID$(G$,1,1)
190     A(I-1)=ASC(MID$(G$,17,1))
200     B(I)=A(I-1)+B(I-1)
210 WEND:CLOSE
220 SUM=32*(I-1)+2:G=I-2
230 HEAD=SUM MOD L:START=INT(SUM/L)
240 OPEN "R",#1,NA$,L
250 FIELD #1,HEAD AS H$,L-HEAD AS Q$
260 GET #1,1+START:R#=1
270 WHILE S$<>CHR$(26)
280     U$=Q$:R#=R#+1:GET #1,R#+START
290     W$=U$+H$:PRINT R#-1,
300     FOR J=1 TO G
310       PRINT MID$(W$,B(J),A(J)),
320     NEXT J
330     PRINT:S$=MID$(Q$,1,1)
340 WEND
350 END
```

（杨先桂）

dBASEⅢ向SC3传送数据的一种简单方法

一、问 题 提 出

dBASEⅢ在数据管理方面的能力是毋庸置疑的。然而，它没有照顾到用图形反映数据的功能，这一点恰是电子报表软件SC3的长处。若用SC3的图形功能来弥补这一缺点，关键在于如何把dBASEⅢ中的数据自动地传送给SC3。

本文介绍一种不需人工干预，通过一简单程序自动实现dBASEⅢ向SC3传送数据的方法。

二、方 法 讨 论

SC3有一条斜杠命令：/X(execute)可以接受外部数据。本命令可以执行除了数据行编辑功能以外的其它SC3操作，包括光标移动和数据输入。因此，我们可以用dBASEⅢ产生一个SC3可执行的宏命令文件，此文件包括生成工作卡的所有操作，这样就可使dBASEⅢ向SC3传送数据的整个过程全自动地完成。

利用SET ALTE TO〔文件名〕命令配合其它命令就可以生成包括光标控制等信息在内的、完全符合SC3宏命令文件格式的文件。

为了生成此宏命令文件，首先，通过COPY STRU EXTE TO STRU.DBF命令把待传送的数据库结构信息存入STRU.DBF中备用。然后，用“? CHR(26)”产生光标定向（向下）控制码；用“?”命令输出光标定位控制码和字段名信息。