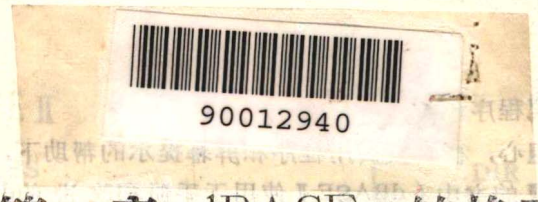
从dBASEⅡ到dBASEⅢ

（美）亚当·格林

贵州科学院情报图书室



第一章 dBASE Ⅱ转换到dBASE Ⅲ

程序的说明

设想在一分钟内存储1000个记录文件，或同时使用10个文件来工作。这是一种幻想吗？不是的。修改dBASE Ⅱ的时候已经到了，等待已经结束，人们提出了功能更强的dBASE Ⅲ。

自从1981年2月以来，我一直在使用dBASE Ⅱ并教授dBASE Ⅱ的课程。虽然我对此结果感到满意，但我从不隐瞒我的希望。我希望对原有dBASE Ⅱ的特性和功能作些修改。当Ashton-Tate的人们第一次告诉我有关dBASE Ⅲ的消息时，我很高兴，此后我了解到所有必须改进的dBASE Ⅱ的程序和一个专用的转换程序（被称为dBASE Convert）此专用程序可自动地做大量的转换工作。

写这部分有关转换问题的指导时，我作了几个假设。首先，我假设读者对dBASE Ⅱ程序语言很熟悉，此外也有一定数量的程序需要作转换。我将不对每一条新的命令作解释，而把重点放在变化上面，即dBASE Ⅱ已有的命令作了哪些变化。在讨论某个新特征时，将其与通常在dBASE Ⅱ中所用的同等功能作比较，得出它较之dBASE Ⅱ的优点。在某些情况下，我将简单地导出一条新的命令以供你参考，使你更好地掌握它的特征。对每一个变化，我将指出哪些变化可由转换程序来完成，而哪些变化将由你自己来完成。

在这篇文章结束后，将有一些详细关于使用转换实用程序（即dBASE Convert）的说明。如果你没有任何dBASE Ⅱ的程序文件需要转换，则请跳过这一部分，并参见下一部分，下一部分将告诉你怎样转换你的数据文件和索引文件。

在我们开始探索dBASE Ⅲ程序的奇妙之处时，让我们首先检查一下这些变化的类型和它们为什么要这样做的理由。有些变化仅仅是对dBASE Ⅱ功能和命令作简单的名称的变化。例如现在检查一个标了删除符的记录是根据VELETED（　），而不是象原来那样根据一个星号“*”。这种类型的改变提高了程序的可靠性。dBASE Convert将提供这种转变的能力。

另一类的变化是涉及到关于一个命令所产生的结果。最明显的例子是FIND命令，在dBSAE Ⅱ中，当FIND命令失败，即未找到一个关键字时，则记录数等于0，然而在dBASE Ⅲ中，当FIND命令失败时，记录数被设置到最后一个记录，且文件结束标志被置为真（即EOF（　）被置为T）。这样FIND命令与LOCATE命令是一致的。诸如此类的有关逻辑或假设方面的改变，dBASE Convert是不能完成的，必须手工做这种转换。

最后一种类型的变化是增强某些功能。最明显的是文件和变量的限制。一个单一的数据文件一次可允许多达128个字段，256个内存变量。当文件类型获得增强的功能后，dBASE Convert将使这种功能立即有效。

你也许要问，做这种转换是否值得。回答是肯定的。在Ashton—Tate的人们已认真的倾听了成千的dBASE Ⅱ使用者的意见和愿望。所有现在出版的dBASE Ⅲ是目前在任何计算

机上所使用的最成熟的最好的管理程序语言。

不必为转换所要做的工作而担心，在转换实用程序和屏幕提示的帮助下，我在一个星期内就完成了我两本有关dBASE Ⅱ的书中（dBASE Ⅱ使用于手册和改进的dBASE Ⅱ使用手册）的一百多个程序的转换，当然若要利用所有新的dBASE Ⅲ特性的优点，要花一定的时间。

然而在你读到这里时，我又开始了新的愿望，希望能早日出现功能更强的dBASE Ⅳ，总而言之，我们不能埋怨微机的这种千变万化的进程。

第一节　数据库文件

1.1　放松对文件的限制

dBASE Ⅲ数据文件的限制已谨慎地扩展到使人们足以够用的数目。我认为每个人都对dBASE Ⅱ的两个数据库文件和32个字段的限止数感到头痛。在需要使用10个数据库文件，且每个文件包含128个字段和10亿个记录时，dBASE Ⅱ采用如此多的文件和字段将不得不写出非常复杂的程序

比较两个性能：

	dBASE Ⅱ	dBASE Ⅲ
每一记录的字段数	32	128
每一记录的字符数	100	4000
记录数	65,535	1,000,000,000
打开数据文件数	2	10
字段的数字精度	10	16

1.2　字段名

为了使字段名更易读，许多dBASE Ⅱ程序员使用冒号作为限止符。例如用AMOUNT"DUE的形式来表明两个字段名。dBASE Convert自动将两者之间的冒号改为底划线。即AMOUNT_UEA

dBASE Ⅱ	dBASE Ⅲ
可使用的字符 A—Z,0—9, ：	A—Z, 0—9, —

1.3　文件结构

因为增加了文件的限止数，内部文件的结构有所改变。若你试图在dBASE Ⅲ中使用一个dBASE Ⅱ的数据文件，则将得出错误的信息。

dBASE Ⅱ

```
.LIST FIKES
DATABASE FILES #RECD LAST UPDATE
NAMES  DBF      0008  01/01
JOBS   DBF      00006 01/01/08

USE Names
```

dBASE Ⅲ

```
.DIR
Database files#records lasT UpdaTe size
NAMES.  DBF dBASE 2.4 database
                                1024
JOBS.   DBF dBASE 2.4 database
                                1024

2048 bytes in 2 files
77824 bytes remainig on drive

USE Names
Not a dBASE database

?

USE Names
Do you want some help? (y/N)N
```

dBASE Convert有能力完全地将dBASE Ⅱ数据文件转换成dBASE Ⅲ的文件形式。

1.4 记录数

记录数是给某个函数新的名字。现用RECNO( )表示,而不用原来的符号"#"来表示。

dBASE Ⅱ

```
.USE Names
.DIsPLAy
00001 AL 100

.?#
    1
```

dBASE Ⅲ

```
USE Names
DISPLAy
Record*NAME ACCOUNT
    1 AL 100
? RECNO( )
    1
```

做此转换的关建部分是在条件中所用的是"*"的符号而不是'〈 〉'的符 号, dBASE Convert能很准确地理解上下文的关系，但你仍然必须要自己做一部分转换工作。

1.5 标注了删除符的记录

对已标了删除符的记录进行逻辑检查，已从检查"*"号转变为 查DELETED( )。这种转换由dBASE Convert自动完成。

dBASE Ⅱ

```
DELETE
0001 DELETION(S)
?*
T.
```

dBASE Ⅲ

```
.DELETE
1Record(s) deleted
.? DELETED( )
.T.
```

1.6 文件结束

dBASE Convert将转变文件结束名EOF为EOF( )这样使它与其它的系统变量和DATE( )和RECNO( )联系起来

dBASE Ⅱ

```
.GOTO BOTTOM
.SKIP
RECORD;00006
.? EOF
.T.
```

dBASE Ⅲ

```
.GOTO BOTTOM
.SKIp
Record No 6
.? EOF( )
.T.
```

1.7 在文件顶部SKIP的逆行

当给出命令SKIP—1时，记录指针将移向上一个记录，例如当前记录再向上移一个记录时，在dBASE Ⅱ中，当前记录数将成为0。这种状态是用于测试循环过程，即测试从BOTTOM到TOP的输入数据文章的过程，为了简化这个过程，dBASE Ⅲ继续使用了一个系统变量BOF( )，当指针在文件顶部执行SKIP—1命令时，BOF( )被置为真，（即BOF( )被置为T的状态）

dBASE Ⅱ

```
.GOTO TOp
.?#
        1
.SKIp-1
.?#
0


/BOF.PRG/
SET TALK OFF
USE Names
GOTO BOTTOM
DO WHILE#( )0
```

dBASE Ⅲ

```
GOTO TOp
.?RECNO( )
        1
.SKIp-1
?RECNO( )
        1
.?BOF( )
.T.
/BOF.PRG/
SET TALK OFF
USE Names
GOTO BOTTOM
DO WIHLE.NOT.BOF( )
```

```
    DISPLAY                          |    DISPLAY
    SKIP-1                           |    SKIp-1
ENDDO                                | ENDDO
```

由于记录数不再为0，必须改变凡有此过程的一切程序的逻辑关系。即该由你自己来做变化到BOF( )的工作。

1.8 新的特性

有两个新的字段类型，它们相对dBASE Ⅱ数据文件来说是大大地增强了字段的功能。多达4000个字符的一个**摘**要字段转变dBASE Ⅲ成为一个合理长度的字处理器。另外一个是日期字段使日期进行运算，以及记录按年、月、日排序成为可能。进一步关于日期的讨论参见下面章节。

第二节 排 序

2.1 排序时间的进一步节省

你想了解dBASE Ⅲ排序一个文件的过程吗？要注意观看，否则会很快错过机会，我努力用眼睛观看此过程，但dBASE Ⅲ排序命令是惊人的快。原来排序1000个记录所需40分钟，而现在却不到1分钟。

2.2 对复合字段进行排序

dBASE Ⅲ可以在一遍扫描中对复合字段进行排序，这样比起原来对复合字段排序需要有主次之分和多次扫描才能完成来说简单多了。并节省了原dBASE Ⅱ所花的人力。

dBASE Ⅱ

```
.LIST STRUCTURE
STRUCTURE FOR FILE:
A:NAMES.DBF
NUMBER OF RECORDS:  00006
DATE OF LAST UPDATE:  01/01/80
PRIMARY USE DATABASE

FLD  NAME     TyPE  WIDTH  DEG
001  NAME     C     010
002  Account  N     004
```

dBASE Ⅲ

```
.LIST STRUCTURE
Structure for database:
A:NAMES dbf
Number of data records:  6
Date of last update:01/01/80

Field  Field name  TyPe  length  Dec
    1  NAME       C     10
    2  Account    N     4
     **Total**           15
```

```
.SORT ON Account TO TemP DESC
SORT COMPLETE
.USE Temp
.SORT ON Name TO MultiPle
SORT COMPLETE
.USE MultiPle
.LIST
00001  AL     100
00002  BoB    102
00003  JIM    104
00004  JoHN   105
00005  JoHN   103
00006  SALLy  101
```

```
.SoRT oN Name,Account10 To MultPle
100% Sorted 6 Records sorted

.USE Multiple
.LIST
Record#  NAME    ACCOUNT
1        AL          100
2        BoB         102
3        JIM         104
4        JoHN        105
5        JoHN        103
6        SALLy       101
```

2.3 排序时对文件的保护

大多数dBASE Ⅱ命令都允许复盖现已存在的文件，为了有利于保护数据文件在意外的情况下不被删除，dBASE Ⅲ将提示一个警告，询问你文件是否要复盖。

dBASE Ⅱ

```
.USE Names
.SORT ON Name TO TemP
SORT COMPLETE
.SORT ON Name TO TemP
SORT COMPLETE
```

dBASE Ⅲ

```
.USE Names
.SORT ON Name TO TemP
100% Sorted    6 records sorted
SORT ON Name TO TemP
Destination file already exists
Destroy it ? N
```

因为此信息将打断正常程序的执行，所以这种情况下必须人为地检查一下程序是否需要复盖，若要以dBASE Ⅱ的方式复盖文件（即可以不打断程序）则必须在程序开头加上SET SAFETY OFF的命令

dBASE Ⅱ

dBASE Ⅲ

```
.SET SAFETY oFF
.SORT ON Name TO TemP
    100% Sorted    6 records sored
```

第三节 检 索

3.1 FIND命令失效时，将不同地影响EOF( )和RECNO( )

dBASE Ⅱ中，当FIND命令不成功时（即没有找到当前所需记录数时），当前记录数等于0，然后程序可测试出# = 0。这种思路在dBASE Ⅲ做了变化，主要是为了使FIND命令与LOCATE命令联系起来，在dBASE Ⅲ中，当没有找到记录时，当前记录指针在文件的底部，并将EOF设置为真，所以程序的逻辑关系需要作某些改动，而dBASE COnvert不能自动做此改动。

dBASE Ⅱ

```
.USE Names
.INDEX ON Name TO Names
.00006 RECORDS INDEXED
USE Names INDEX Names
.LIST
00001   AL      100
00003   BOB     102

00005   JIM     104
00004   JOHN    103
00006   JOHN    105
00002   SALLy   101
    .FIND BOB
    .DISPLAy
00003   BOB     102

.FIND FRED
NO FIND
  . ? #
    0
  .? EoF
  .F.
```

dBASE Ⅲ

```
.U.SE Names
.INDEX ON Name TO Names
6 Records indexed
.USE Names INDEX Names
.LIST
Record#  NAME   Account
   1     AL      100
   3     BOB     102
   5     JIM     104
   4     JOHN    103
   6     JOHN    109
   2     SALLy   101
   .FIND BOB
   .DISPLAy
  Record#  NAME   Account
     3     BOB     102
  .FIND FRED
    NO find
  ? RECNO( )
        2
  ? EOF( )
    .T.
```

3.2 用SEEK简化查找变量的过程

当FIND命令与一个内存变量在dBASE Ⅱ和dBASE Ⅲ中都作为宏变量来使用时：

dBASE Ⅱ

```
.STORE"BOB"TO Key
BOB
.FIND Key
NO FIND
.FIND & Key
.DISPLAY
00003 BOB   102
```

dBASE Ⅲ

```
.STORE"BOB" TO Key
  BOB
.FIND key
  NO FIND
.FIND & Key
.DISPLAy
Record# NAME ACCOUNT
  3   BOB   102
```

简化这项技术有一新的命令，此命令不需要转换。SEEK命令也用于检索，但假定它已被赋予一个变量，这种检索方法对于用数字作索引键时特别有效。

dBASE Ⅱ

```
  .INDEX ON ACCOUNT TO ACCOUNT
00006RECORDS INDEXED
.USE Names INDEX ACCOUNT
.FIND    101
  .DISPLAy
00002 SALLy 101
.STORE 101 TO key
    101
.FIND key
  NO FIND
.STORE STR(key,3)TO Newkey
    101
FIND &Newkey
DISPLAy
00002 SALLy 101
```

dBASE Ⅲ

```
.SEEK Key
DISPLAy
Record#NAME    ACCOUNT
    3 BOB              102
.INDEX ON ACCOUNT TO ACCOUNT
  6 Records indexed
.USE Names INDEX ACCOUNT
.FIND 101
DISPLAy
Record #NAME    ACCOUNT
    2  SALLy          101
.STORE 101 TO key
    101
.FIND key
  NO FIND


SEEK key
.DISPLAy
Record NAME    ACCOUNT
    2  SALLy          101
```

第四节　多个文件的联结

4.1　用SELECT　1…10代替　SELECT　主/次

这也是对dBASE Ⅱ改动较大之处，在dBASE Ⅲ中多达10个数据文件可以同时打开，而dBASE Ⅱ只能使用两个数据文件，但以我个人来说，难以想象在任何应用中使用到多于4个或5个数据文件，为了便于提及两个数据区域，PRIMARY和SECONDARY已用1，2，3等数字来代替，这种简单的名字改变由dBASE COnvert自动完成，但为了同时使用两个以上数据文件，应该重新写自己的程序，这个额外的程序设计是很值得的，因为当所有文件在内存中是打开状态时，复合文件程序执行起来明显的快得多。

dBASE Ⅱ

```
.SELECT PRIMARy
.USE Names
.DISPLAy
   00001   AL   100
SELECT SECONDARy
USEjobs
.INDEX ON ACCOUNT TO jobs
00006 RECORDS INDEXED
.USE Jobs INDEX jobs
.DISPLAy
00001 100 BUTCHER
```

dBASE Ⅲ

```
.SELECT  1
.USE Names
.DISPLAy
Record# NAME ACCOUNT
     1  AL              100
.SELECT  2
.USE jobs
.INDEX ON ACCOUNT TO jobs
   6 Records indexed
.USE jobs INDEX jobs
  .DISPLAY
.Record# ACCOUNT JOB 1 100
  BOTCHER
```

.2 在不同数据区域用于定义字段的文件名

在dBASE Ⅱ中，可以随便使用PRIMARY和SECONDARY中的字段，如果一个字段名在两者的内存区域中都被用到，则“P”或“S”作为此变量的前缀，以便有唯一的定义。

这种方法对多至几个数据文件的复杂情况将是不适用的，当一文件在dBASE Ⅲ中被使用时，对不在当前数据区域的每一个字段必须以此文件名作为前缀，在这种情况下，此文件名被称为“假名”，为了增强假名之间的一致性，在SELECT一个数据区域时，文件名是一个易接受的数字的代用名，在假名后面做一个小的语法变动，即用“→”代替原来的“·”符号

dBASE Ⅱ

```
.? JOB
  BUTCHER
.? Name
  AL
```

dBASE Ⅲ

```
. ?JOB
  BUTCHER
?Name
  ?
?Name
Do you want some help(y/N)N
? Names→Name
  AL
```

```
.SELECT PRIMARY
.SKIP
RECORD:  00002
.DISPLAY
00002 SALLY 101
. ? ACCOUNT
  100
.SELECT SECONDARY
.?ACCOUNT
    100
.?P·ACCOUNT
    101
```

缺席的假名可用USE命令去改变

dBASE Ⅱ

```
SELECT Names
SKIp
Record no 2
.DISPLAY
Record#NAME ACCUNT 2 SALLY 101
.?Names→ACCOUNT
    100
.SELECT jobs
.? ACCOUNT
    100
? Names→ACCOUNT
    101
```

dBASE Ⅲ

```
USE jobs ALTAS Work
?Work→ job
BUTCHER
```

文件假名的这种提法也许在开始时感到有些不理解，但它是很有效的方法。使得程序不依赖于具体的数据区域。如果一个程序总是用缺席假名Jobs作为字假名，它将与Jobs文件 被使用的工作区域无关，可以说明程序现在“可重新安置”在任何数据区域

dBASE Convert能将SELECT PRIMARY转换为SELECT1和将SELECT SECONDARY转换为SELECT 2，但它不是足够灵活到为你提供合适的假名。

4.3 用SET RELATION TO简化连接过程

SET RELATION TO是一个完全新的dBASE Ⅲ命令，它对普 通的dBASE Ⅱ技术作了很大的改进。我们必须写出它与dBASE Ⅱ的功能的比较

dBASE Ⅱ

```
.SELECT PRIMARY
.USE Names
'STORE STR (ACCOUNT,4)To key

    101
.SELCT SECONDARY
```

dBASE Ⅲ

```
SELECT 2
USE jobs INDEX jobs
SELECT 1
.USE Names
SET RELATION To ACCOUNT INTO
    jobs

SELECT 2
```

```
.USE iobs  INDEX jobs
FIND & key
DISPLAY
0002 101 BAKER
```

```
DISPLAY
.Record#  ACCOUNT  JoB
    2      101      BAKER
```

在dBASE Ⅲ中，SET RELAION TO使两个文件的记录指针自动地联系起来。

dBASE Ⅱ

```
[LIST 2.PRG]
SET TALK OFF
SELECT PRIMARY
USE Names
SELECT SECNODARY
USE Jobs INDEX Jobs
SELECT PRIMARY
DO WHILE.NOT.EOF
    STORE STR (ACCOUNT,4)TO key
    SELECT SECONDARy
    FIND & key
    ?Account Name Job
    SELECT PRIMARY
ENDDO
.DO List 2
100 AL      BUTCHER
101 SALLY BAKER
102 BOB     HACKER
103 JOHN    LAWYER
104 JIM     WRITER
105  JOHN DOCTOR
```

dBASE Ⅲ

```
[LIST2. PRG]
SET TALK OFF
SELECT 1
USE Names
SELECT 2
USE Jobs INDEX Jobs
SELECT 1
SET RELATION TO ACCOUNT INTO
    Jobs
DO WHILE.NOT.EOF(  )
    ?Account,Name,Jobs→Job
    SKIP

ENDDO
·Do List 2
100  AL  BUTCHER
101  SALLY BEKER
102  BOB  HACKER
103  JOHN  LAWYER
104  JIM WRITER
105  JOHN DOCTOR
```

在dBASE Ⅲ中仍可用旧的方法（即dBASE Ⅱ所用的方法），但作出这种新技术的变换是值得的。

4.4 删除了SET LINKAGE ON的命令

现在数据区域已多于2，SET LINKAGE ON命令已没什么意义，将它改为SET RELATION TO，然后加上适当的数据区域号或段名。

第五节　内存变量

5.1　对内存限制的扩充

与数据文件扩充一样，内存变量的限制已扩充到对一般的程序或程序员可能的需求以外。

	dBASE Ⅱ	dBASE Ⅲ
最大内存变量数	64	256
最大内存变量字节数	1536	6000

5.2　新的内存文件结构

dBASE　Convert能迅速地将dBASE Ⅱ中的内存文件转换为dBASE Ⅲ内存文件的格式。

5.3　变量

在命令文件和内存文件中，内存变量的冒号由dBASE Convert转换为底划线。

dBASE Ⅱ	dBASE Ⅲ
Acceptable characters A-Z,0-9:	A-Z,0-9,-

5.4　ACCEPT、INPUT或WAIT中不出现冒号

很巧，这也是我个人的希望之一，在dBASE Ⅲ中ACCEPT,INPUT和WAIT语法在提词后都不加冒号，并不需要对这种变化做什么工作，但可以在提词后加一个问号，若还认为糊涂的话，可以加冒号，但最好别这么做。

dBASE Ⅱ	dBASE Ⅲ
.ACCEPT"Name" To Name NAME? MAX	.ACCEPT"Name?"To Name NAME?MAX .ACCEPT"NAME:"To Name NAME:MAX

5.5　在ACCEPT或WAIT下回车产生一个空

在dBASE Ⅱ的ACCEPT或WAIT语句中出现回车时，产生一个空格，众所周知，在dBAES Ⅱ中，学过对此状态的测试。

dBASE Ⅱ

```
[ACCEPT.PRG]
SET TALK OFF
ACCEPT"WELL?"TO Answer
IF Answer="Y"
  ?"yes"
ENDIF
IF Answer="  "
  f"CARRIAGE RETURN"
ENDIF
.DO Accept
WELL?:Y
.DO Accept
WELL?:↵
CARRIAGE RETURN
```

dBASE Ⅲ

```
[ACCEPT.PRG]
SET TALK OFF
ACCEPT""WELL?"TO Answer
IF Answer="Y"
  ?"yes
ENDIF
IF Answer="  "
?"CARRIAGE RETURN"
ENDIF
.DO Accept
WELL¿:Y
.DO Accept
WELL? ↵
```

在dBASE Ⅲ中测试ANSWER=" " 不起作用，这是因为一个回车将产生一个新类型的变量，被称为"空"，它甚至比空格还要小，可认为没有任何数据。一个空变量可由它的长度确定，空变量的长度为零。

dBASE Ⅱ

```
.ACCEPT"WELL? "TO Test
WELL? :↵
.? LEN(Test)
 1
```

dBASE Ⅲ

```
.ACCEPT"WELL? " TO Test
WELL? :↵
.?  LEN(Test)
0
```

空变量最特殊的性质是任何量都与它相等。

dBASE Ⅱ

dBASE Ⅲ

```
.Test="ABC"
.F.
.? "ABC"=Test
.T.
[NEWWAY.PRG]
SET TALK OFF
ACCEPT"WELL? "TO Answer
IF Answer="Y"
  ? "yes"
```

```
ENDIF
IF Answer=" "
  ? "CARRIAGE RETURN1"
ENDIF
IF" "=Answer
  ? "CARRIAGE RETURN2"
ENDIF
.DO Newway
WELL? Y
YES
CARRIAGE BETURN1
.DO Newway
EWLL? ↲
CARRIAGE RETURN1
CARRIAGE RETURN2
```

在NEWWAY程序中" "=Answer是对空变量另一种可能的测试方法，这种改变有很大的辨别能力，用dBASE Ⅱ的方法ANSWER=" "是没有办法区别空格和回车的。在dBASE Ⅲ中，回车就产生一个空变量。在很长的运行过程中，由于有了这种改变，dBASE Ⅲ程序的执行要精确得多。

5.6 类型的改进

对于dBASE Ⅱ，当内存变量改变其类型，而没有改变名字时，总存在一些问题。现在dBASE Ⅲ中能得到解决。

dBASE Ⅱ

```
.STORE 1 TO X
  1
.STORE STR(X,I)TOX
  0
.LIST MEMORY
X    (C)     0
**TOTAL**01 VARIABLES USED
00002 BYTES USED


.STORE"1"TO X
  1
```

dBASE Ⅲ

```
.STORE I To X
  1
.STORE STR(X,1)To X
  1
.LIST MEMORY
X    C 1
1 Variables defined.    3  bytes used

255 Variables available,   5997 byty
available
.STORE"1" To X
  1
```

```
.STORE VAL(X) TO X
  0
.LIST MEMORY
X [N]  0
**TOTAL** 01 VARIABLES USED
000007 BYTES USED
```

```
.STORE VAL (X) TO  X
  1
.LIST MEMORY
X N 1 〔1,00000000〕
1 variables defined,9 bytes used.
255 variables available,5991
bytes available
```

5.7 全局变量的丢失

在dBASE Ⅱ中所有内存变量对任何命令文件来说是可用的。这种“全局”形式的变量有一种基本回收方法。在同一个应用程序中、许多程序员都用“全局”变量时，在程序换块之间会出现相互影响的问题。用英语表达的意思是每个人都想使用称为Name和Date的变量。

dBASE Ⅲ采用“范围规则”和“参数传递”的方法解决此问题。这两个技术提供了许多优点，但也需要大量应用系统的支持。

范围意味着内存变量在生成它们的程序执行完毕后被释放。举例来说，程序Master生成一个称为Master的变量，这时又有程序Sub开始执行。此时变量Master已经存在，并可由程序Sub改变。在Sub程序中，又产生了一个变量Sub ’当程序Sub执行完毕返回程序 Master，重新开始执行时，内存中已不存在变量Sub。并且，变量Master将保持被Sub程序改变过的值。

dBASE Ⅱ

```
〔MASTER. PRG〕
SET TALK OFF
? “NOW I’ M IN MASTER”
STORE “ I WAS CREATED IN MAS-
TER ” TO Master
LIST MEMORY
DO Sub
?
? “NOW I’M BACK IN MASTER”
LIST MEMORY
〔SUB.PRG〕
?
? “NOW I’M IN SUB”
STORE“I’VE BEEN CHANGED IN
SUB” TO Master
STORE “I WAS CREATED IN
SUB” TO Sub
```

dBASE Ⅲ

```
〔MASTER〕
SET TALK OFF
? “NOW I’M IN MASTER”
STORE“I WAS CREATED
IN MASTER”TO Master
LIST MEMORY
DO Sub
?
? “NOW I’ M BACK IN MASTER”
LIST MEMORY
〔SUB PRG〕
?
? “NOW I’M IN SUB”
STORE “I’VE BEEN CHANGED
IN SUB” TO Master
STORE “I WAS CREATED IN
SUB”TO Sub
```

```
LIST MEMORY
.DO Master
NOW I'M IN MASTER
MASTER (C) I WAS CREATED
IN MASTER
**TOTAL** 01 VARIABLES
USED 00024 BYTES


NOW I'M IN SUB
MASTER (C) I' VE BEEN
CHANGED IN SUB
SUB (C) I WAS CREATED IN SUB
**TOTAL** 02 VARIABLES USED
 00046 BYTES USED


NOW I'M BACK IN MASTER

MASTER (C) I'VE BEEN CHANGED
 IN SUB
 SUB (C) I WAS CREATED IN SUB
 **TOTAL **02 VARIABLES USED
 00046 BYTES USED
```

```
LIST MEMORY
.DO Master
NOW I'M IN MASTER
MASTER C I Was CREATED
IN MASTER
1 variables defined,
25  bytes used
255 variables available,
5975 bytes available
NOW I'M IN SUB
MASTER C I' VE BEEN
CHANGED IN SUB
C I WAS CREATED IN SUB


2 Variables defined.
48 bytes used
254 variables available
5952 bytes available


NOW I'M BACK IN MASTER
MASTER C I'VE BEEN
CHANGED IN SUB
1Variables defined,26
26 bytes used
255 variables available,
5974 bytes available
```

dBASE Convert明显地不能帮助进行这个变化，如果不想采用这项新技术，只需在第一个程序运行中产生出整个系统所需的所有变量，其效果与dBASE Ⅱ全局变量相同。

参数传递的意义是一个程序可以将信息传递到另一程序，而不需要使用相同的变量名。许多小块的程序可分别地写出，然后装配在一起成为一个完整的系统。

dBASE Ⅱ

dBASE Ⅲ

```
[PARAMS.PRG]
PARAMETERS Name
?  Name
.DO Params WITH "JOE"
```