第8卷 第1期 华 侨 大 学 学 报 Vol.8 No.1
1987年2月 JOURNAL OF HUACHIAO UNIVERSITY Feb. 1987

中西文 dBASE Ⅱ 若干语句的正确使用

张 银 明

[计算机科学(电脑)系]

摘 要

本文根据作者应用 dBASE Ⅱ 的实践，分析了 TOTAL、DO WHILE 等语句存在的问题，提出了解决的方法，並对它们的使用作了更为确切的说明．文中还对数组在 dBASE 中的使用方法作了论述，这将给它的编程带来极大的方便。

中西文 dBASE Ⅱ 数据库管理系统是颇受用户欢迎的数据管理软件，又具汉字处理功能和全部提示信息汉字化而在国内得以迅速推广。它有许多特点，但又含有不少缺陷，其中有些问题已在 dBASE Ⅲ 中得到改善，可是有些问题依然存在。例如 dBASE Ⅲ 的主要技术指标得到全面增强；数据类型更丰富；命令功能加以扩充和增加新命令以及操作指示、屏幕格式与查询信息等更为丰富。但是对一些常用的主要命令，诸如 TOTAL、DO WHILE 等依然如故。根据我们在一个规模较大的企业综合管理辅助软件研制过程中的实践，发现其中一些命令和语句的使用说明不确切，甚至会导致程序系统的隐含错误，而且难以从程序逻辑上查出原因。为使该语言的使用者少走弯路，减少程序的调试时间，避免隐蔽的错误，以便缩短应用软件的研制周期。所以有必要指出这些命令或语句所存在的问题，提出正确的使用方法以及进行更为确切的说明。

其次，对 dBASE 中的二维数组的实现方法也作了探讨，並稍作介绍。尤其对 dBASE Ⅲ，它的内存变量已从 dBASE Ⅱ 的 64 个扩充到256个，所以数组的使用是可行的，而且将给编程带来极大的方便。

一、TOTAL 命令

TOTAL 命令*的一般格式为：

TOTAL ON 〈Key〉TO〈filename〉
[FIELDS〈field-LIST〉][FOR〈exp〉]

该命令在数据处理中是极为有用的，也是 dBASE Ⅱ 中功能较强的命令之一，在程序中

本文 1686 年 6 月 12 日收到。

• 文中所指命令，意即既可作为命令；也可作为程序中的语句使用。

可作为语句使用。但无论是IBM的原说明或至今出版的有关书籍中，对TOTAL命令的使用说明都是不够确切的，主要是对〈filename〉数据结构的说明隐含着错误。〈filename〉是存贮按关键字进行总计的数据库文件名，原说明中对其结构分为两种情况。

第一种情况是〈filename〉的数据结构在使用TOTAL之前没有另行建立。那么，它的结构将由字段（FIELDS）项所指出的字段名构成。若设有指出字段项，则将原数据库的结构复制给新的数据文件。

这个说明是不完整的。原因在于当我们设计一个数据库文件的结构时，每个字段的长度最极限的方法是按其最长的那个长度或位数来定义，而需要作为数据处理的数据库文件的记录，一般少者有数百条，多者有数千条至上万条。如我们研制的企业综合管理辅助软件，其库存机件有6000多件，材料库品种多达10000多种，其相应的数据库文件便有同样数目的记录数。这样，按某关键字进行累计时，和数的位数大多会超过原来定义的字段长度。这就使得在执行TOTAL命令或作为语句执行的过程中，引起“数值字段上溢”的错误，因而使总计的数据发生差错。所以，该使用说明在绝大多数情况下是行不通的。

第二种情况是〈filename〉的数据结构自行建立。对此，原说明认为“它的结构被完整地保存，且决定了那个字段将使用总计运算”或“循其结构以转移资料”。

此意是说，在使用TOTAL命令之前，须先行建立一个存贮求和数据结构。当然，其数值型字段的位数应比原字段的位数要多，以便容纳得下求和结果的数值。但实践证明，这样求得的总计将发生差错，更糟的是程序执行时並没有任何提示信息，程序也执行得很顺利，因为无论从逻辑上或语句的使用方法都是无可挑剔的，而且和数也求出来了，但有些结果则是错误的。这就造成了一种隐含的不易被觉察的错误。

显然，第二种情况的出错如不详细核查是难于发现的，因而比第一种更具危害性。

为说明问题，下面以配件库数据文件的部分字段（数据库中称为属性域）为例。

假定机床配件数据库为PJK.DBF，其数据结构说明如表1所示。

表1 数据库文件 PJK.DBF 结构

名 称	机 型	图 号	最高库存量	最低库存量	库存量	库存金额	代销库存1	代销库存2
字段名	JC	TH	ZGKCL	ZDKCL	KCL	KCJG	DSKC1	DSKC2
类 型	C	C	N	N	N	N	N	N
字段长	8	12	4	3	4	9,2	4	4

该配件库有50多种机型的6000多个图号，相应的有同样数目的记录，亦即，每类机型含有少则数个多至数百个的记录。为了以机型为别统计库存情况和资金，建立了以JC为关键字的索引文件PJKJC.NDX。仅对三种机型的1000多个记录进行统计而不另建新的数据结构，在执行

```
USE PJK INDEX PJKJC
TOTAL ON JC TO PJKHJ FIELDS JC ZGKCL,
        ZDKCL KCL KCJG DSKC1 DSKC2
```

的过程中，便产生“数值字段上溢”的提示信息。根据第一种使用说明，存放机型统计结果

的数据库文件PJKHJ的结构将由FIELDS字段项的结构决定。所以，其字段长同PJK中的数值字段长相同，但容纳不下和数的数值而导至溢出。

为解决求和的数值上溢问题，依照第二种使用说明，事先生成了PJKHJ的数据结构，每个数值型字段都增加到足够存放和数的长度，具体结构如表2所示。

表 2　数据库文件 PJKHJ 的结构

字段名	JC	TH	ZGKCL	ZDKCL	KCL	KCJG	DSKC 1	DSKC 2
类　型	C	C	N	N	N	N	N	N
长　度	8	12	6	5	6	11,2	6	6

这样一来，重新执行相应的程序时，既无上溢的提示信息，也无其它的出错或提示。TOTAL语句也执行得很顺利，同时给出统计的结果並存于PJKHJ之中。具体数据由表3给出。

表 3　机型统计数据

机　型	名　称	最高库存量	最低库存量	库存量	库存金额	代销库存1	代销库存2
C615	齿　轮	1634	444	2043	45218.50	38	198
C616	拨动齿轮	9728	2711	8076	65039.30	124	98
C616CQ	齿　条	60	30	76	39.00	24	98
合　计		11422	3185	10195	110296.80	186	394

按配件库当时的实际存贮数据。C616的代销库存2应为0，C616CQ的库存量、库存金额、代销库存1及2也为0，但却得到98、76、39.00、24、98等值。经分析可以看出，当前一行的统计数值不为0，而下一行的和数本该为0时，经TOTAL命令的统计后，它把前一行整数的低两位数抄给它。这说明，当用户为存贮统计结果的数据而建立新的数据结构后，其数值型字段的长度超过原来字段的扩大位数部分，统计的结果数据可能产生错误。这也说明第二种使用方法也是行不通的。

为解决上述存在的缺陷，作者经过探索和实践而取得了解决的方法。其正确的过程如下：

第一步，按统计的需要建立一个新的数据库文件的数据结构，其数值型的字段长度须足够存贮统计后的结果数据；第二步，把被求统计的数据库文件的所有记录复制给新定义的数据库文件；第三步，对新的数据库文件建立以求统计字段名为关键字的索引，並对该关键字使用统计的TOTAL命令（或语句）；最后，删除中间文件的全部记录，也就是新定义其结构的数据库文件的所有记录，以回收存贮空间，並保留其结构以供后用。而对存贮统计结果的数据库文件可按处理的需要决定其取舍。

这个方法的过程如图1所示。

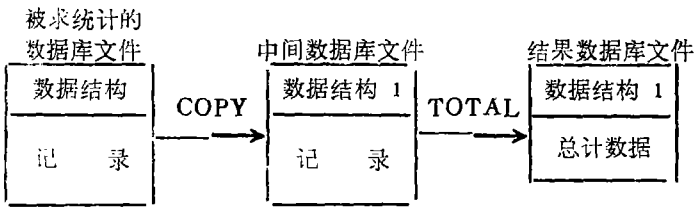

图 1 使用 TOTAL 的正确过程

假如已建立了中间数据库文件 PJKJC.DBF 的结构，对 PJK 按机型求总计的结果存于 PJKHJ 之中，那么，其过程的部分程序可描述为：

```
SELE PRIM
USE PJK
COPY TO PJKJC
USE PJKJC
  INDEX ON JC TO PJKJCNDX
USE PJKJC INDEX PJKJCNDX
  TOTAL ON JC TO PJKHJ FIELDS JC ZGKCL,
            ZDKCL KCL KCJG DSKC1 DSKC2
        〈输出打印及中间处理〉
SELE PRIM
 USE PJKJC
    DELE ALL
    PACK
 DELE FILE PJKJCNDX.NDX
 USE
RETURN
```

使用这种方法所得到的统计结果，其数据是正确的。

据上所述，TOTAL 命令中对〈filename〉结构的可行的确切说明应该是：

如果按〈key〉求总计的数值长度不超过被求和数据库文件的对应字段长度时，则〈filename〉的文件结构可不必另行建立，系统将按 FIELDS 所指定的字段名组成新数据库文件的结构，或把数据结构复制给新文件。

若〈filename〉的数据结构需事先建立，且字段长度同被求和数据库文件的对应字段长度不一致时，必须先把后者的记录复制到一个同〈filename〉的结构相同的中间数据库文件，尔后按要求对该中间文件使用 TOTAL 命令或语句。

诚然，这给 TOTAL 命令的使用找到了正确的途径。而且，一般的程序中使用它来编程是较为方便的。

但笔者认为这样编程的程序执行起来的效率是很低的。为使 TOTAL 命令能执行得正确，要额外花上 COPY 及其它的相应处理时间，对一个数千个记录的文件复制一次也是颇费时间的，何况 TOTAL 语句执行时也是极慢的。为此，我们在研制的课题中，凡原来使用

TOTAL 语句的地方都用自编的统计程序加以替代，这样反而使执行速度有了可观的提高。由于使用 TOTAL 语句的条件是事先须按要求统计的字段名作为关键字建立索引或进行 SORT。既如此，按该字段名求统计的程序也是容易编制的。

所以，本人认为使用 dBASE Ⅱ 编制应用软件时，尽量少用 TOTAL 命令作为语句。如要使用，请参照上述提供的方法，方能得到正确的结果。

二、DO WHILE 语句

dBASE Ⅱ 中的 DO WHILE 是同 ENDDO 联用而组成循环语句的，其形式为：

```
DO WHILE〈exp〉
〈循环体〉
ENDDO
```

任何高级语言中，循环语句都是极为重要的和不可缺少的，dBASE Ⅱ 也不例外。

对循环语句通常的说明是：当条件成立时重复执行循环体的处理，而一旦条件不满足，便结束循环。在循环结束后，除与控制条件有关的变量可能改变外，其他内存变量及有关参数应保持循环结束时的状态。

然而，我们在 dBASE Ⅱ 的 DO WHILE 语句的使用中则发生了例外的情况。

例如，在机床配件中，有些不同的机型、图号是可以通用的。因处理需要而使用拉链的方法。在互为通用的配件中，有一种定为主件，它的相应记录存贮必要的数据，而其他定为附件，不存数据且其库存量以（-1）作为标志。拉链采用单向闭环方式，则从任一附件开始，通过链可找到其主件，也可查到同主件通用的所有附件。该配件库文件记录的有关部分的形式如下：

〈记录号〉	〈部件号〉(BJH)	〈机型〉(JC)	〈图号〉(TH)	〈库存量〉(KCL)	〈通用件链〉(TYJL)
00001	0001	C615	2014	100	
00002	0002	C615	2016	150	0003
00003	0003	C615-1	2016-C	-1	0005
00004	0004	C615-2	3010	200	
00005	0005	Q615-3	2016-Q	-1	0002

该数据库文件仍命名为 PJK.DBF，相应的字段名为 BJH，JC，TH，KCL，TYJL。使用部件号作为链指针，按部件号（BJH）建立的索引文件为 PJKBJH.NDX，按 JC+TH 建立的索引文件为 PJKJT.NDX。按这种结构模型存贮配件信息时，对任一张进库单的处理都要考虑到找主件的问题，因而其处理过程为：

```
    .  .  .
〈取得进库单的记录〉
STORE JC+TH TO JT
SELE SECO
USE PJK INDEX PJKJC PJKBJH
```

```
FIND &JT
  IF KCL = -1
    STORE "'" +TYJL+ "'" TO MBJH
    SET INDEX TO PJKBJH
      DO WHILE KCL = -1
        FIND &MBJH
        STORE "'" +TYJL+ "'" TO MBJH
      ENDDO
    ENDIF
REPL KCL WITH KCL+P.SL
```

现假设有张进库单，其机型为C615-1，图号为2016-C，而其数量SL=50。按程序的原意，因该配件为附件，所以经DO WHILE KCL=-1 循环之后，理应找到对应的主件00002号记录，並使它的库存量修改成150+50=200。可实际执行的结果则将00001号记录的库存量改为150。在对其它的进库单为附件的处理过程，其结果也总是修改00001号记录的库存量。

查找原因，在循环体内对寻找主件过程进行的跟踪表明，程序执行过程是正确的。而在循环结束后的REPLACE语句前，由PRINT # 语句测试的结果总是为0。这说明系统在处理循环结束的过程中把记录号函数#的状态作了错误的清除而造成的。但决不是所有循环结束时都发生这种问题，如在DO WHILE〈exc〉中的〈exd〉不用关系表达式而改用逻辑表达式来控制循环，也就不会产生类似的错误。因而，上面的循环程序部分可改为：

```
STORE T TO LOG
DO WHILE LOG
  FIND &MBJH
    IF KCL = -1
      STORE "'" +TYJL+ "'" TO MBJH
    ELSE
      STORE F TO LOG
    ENDIF
ENDDO
```

这样，便可顺利地实现预期的目的，亦即记录指针不再因循环结束而被改变。例如上例中进库单将正确地把00002记录的库存量改成为200。这就说明用关系表达式控制循环可能出错。

据此，对DO WHILE〈exp〉的使用要进一步说明的是：如果循环使用关系表达式作为控制条件，而在循环结束后未能达到预期的目的时，可把控制条件改为逻辑表达式，且在循环体内由关系表达式改变逻辑表达式的取值，以达到同样的控制效果。

三、删除记录的 DELETE 命令

删除记录的DELETE命令的一般形式为：

DELETE [SCOPE][FOR〈EXP〉]

其中 SCOPE 有四种选择：RECORD〈N〉，NEXT〈N〉，ALL 及省略。

选择中的 N 虽有显式之意，但没有明确的说明。实际上，N 只能是数值常数，不能为变量或表达式。当需要使用变量时，则必须由宏代换进行转换。

在综合管理的辅助软件及其它数据处理中，一种处理通常要涉及几个数据库文件。因此，有时要删除一个记录，得先按一定条件在某个数据库文件中找到该记录，但它的删除与否尚不能直接由 FOR〈exp〉决定，又须经过同其它数据库文件的记录进行分析核准后，如符合要删除的条件，方可使用 DELETE 命令。据此，当查到记录时就得把记录号保存下来，经判断后决定可删除时再使用。然而不能使用下面的语句形式：

```
STORE # TO RN
• • •
DELETE RECORD RN
```

而应该是

```
STORE STR(#, 5, 0)TO VN
• • •
DELETE RECORD &VN
```

的语句形式才能正确地删除所指定的记录。

四、INDEX 与 FIND 命令

数据处理要进行多样化的处理，诸如多个数据库文件记录的综合修改和查询、单个库文件的组合查询、系统维护及日常的事务处理，这些都要求进行快速的搜索。为此，必须建立多种索引。这同 INDEX 与 FIND 命令有关，但原说明也没有交代清楚，尤其是索引文件同记录增加的关系与实际功能不相符。

1. INDEX 命令

该命令的形式为：

INDEX ON〈key-exp〉TO〈indexfilename〉

原说明指出：〈key-exp〉可以是一个字段变量，也可以是几个字段变量相加。对于数字型字段变量必须使用 STR( )函数，先转换成字符型。

这说明的前面部分无疑是正确的，但后一句不确切。实际上可直接以数字型字段变量作为〈key-exp〉建立索引。但不同类型的字段变量不能建立组合索引，须转换成同一类型的变量（一般都转换成字符串变量）。而且可以由字段变量中的某几个字符组成关键字建立索引。当然，在搜索时的类型和形式必须同建立索引时相匹配。

例如，进库单（JKD）的文件结构为：内外标志（RWBZ, C, 1）+序号（XH, C, 3）+发票号（XPH, C, 6）+日期（RQ, C, 4）+合同编号（HTBH, C, 12）+机型（JC, C,8）+图号（TH, C, 12）+数量（SL, N, 4）+单价（DJ, N, 7, 2）。那么，对它建立的部分索引可以是：

```
USE JKD
```

```
INDEX ON XH TO JKDXH
INDEX ON JC+TH+RWBZ TO JKDJTR
INDEX ON SL TO JKDSL
INDEX ON $(RQ, 3, 2)TO JKDYF
```

但不能建立下面的索引：

```
INDEX ON JC+TH+SL TO JKDJTS
```

而可以改为：

```
INDEX ON JC+TH+STR(SL, 4, 0, )TO IKDJTS.
```

2．**FIND 命令**

该命令的形式为：

```
FIND 〈Key-String〉OR '〈Key-String〉'
```

对该命令要补充的说明是对几个字段变量组合所建立的索引，在使用 FIND 命令时，可以由字段变量中自前至后的顺序组合作为〈key-string〉。

比如，对上述的索引文件 JKDJTR.NDX，可使用 JC，JC+TH，JC+TH+RWBZ等作为〈key-string〉，但不能使用 TH，RWBZ，TH+RWBZ 或 JC+RWBZ 作为〈key-string〉进行搜索。

至于建立索引与记录搜索时的类型、形式的匹配，或者相一致则是必须满足的。

3．**已建立索引文件的数据库文件修改**

对此，原说明指出："在打开数据库文件及其索引文件（最多为7个）后，当使用：APPEND、EDIT、REPLACE、BROWSE、DELETE、READ、PACK 命令时，中西文 dBASE 系统会自动修改这些索引文件，保证了数据库一致性"。这个说明比较含糊。实际的功能也並不尽然。

根据了解到的几个应用系统的情况，较为合适的说明应区别加以阐明才好，亦即：

在打开数据库文件及其索引文件的情况下：

(1) 使用 EDIT、BROWSE、DELETE 以及对非索引关键字的 REPLACE 修改，系统将自动修改相应的索引文件。但 DELETE+PACK 则不能保证；

(2) 当使用 APPEND BLANK+READ+REPLACE 的语句格式或 APPEND+GET+READ 的语句格式增加记录时，新增加的记录並不会反映到索引文件中去，必须重新建立所有的相应索引；

(3) 只有在成批处理时，使用

```
APPEND FROM 〈filename〉
```

语句，而不使用 REPLACE 语句，方能把新增加的记录添加到索引文件而真正保证了数据库的一致性。

五、数组问题

在 dBASE Ⅱ 中，虽因限于内存变量数目而使数组的使用受到限制，但並非不能应用二维或高维数组。实际上，数组元素的下标完全可由宏代换的使用来实现。

假定内存变量许可，那么矩阵相乘

$$\sum_{j=1}^{5} Aij \times Bj = Ci \qquad i = 1,\ 2,\ 3,\ 4,\ 5$$

则可由下面的程序段实现：

```
STORE 1 TO I1
DO WHILE I1<=5
  STORE STR(I1, 1, 0)TO I
  STORE 1 TO J1
  STORE 0 TO C&I
    DO WHILE J1<=5
      STORE STR(J1, 1, 0)TO J
      STORE C&I+A&I&J*B&J TO C&I
      STORE J1+1 TO J1
    ENDDO
  STORE I1+1 TO I1
ENDDO
```

从上面程序段中可以看到，在定义一个数组元素的下标时，从数值型变量径 STR( ) 函数转换成字符型变量，再由宏代换使之成为下标，而经数值型变量改变下标的取值。

另一个方法是由字符型变量经宏代换定义为下标，但下标值的改变要经过 STR( ) 与 VAL( ) 函数的转换。例如，要给内存变量 Ai (i=11, 12, …, 40) 置 0。其程序段可写成：

```
STORE '11' TO I
DO WHILE I<= '40'
  STORE 0 TO A&I
  STORE STR(VAL(I)+1, 2, 0)TO I
ENDDO
```

作为程序设计人员很清楚数组的使用同循环语句的结合给编程所带来的方便和好处。有的书籍中断言 dBASE Ⅱ 中不能使用数组是不确切的。事实上，在我们研究的课题中不少地方使用数组的方法。当然，对超过内存变量规定数目的数组处理，可以边处理边释放数组元素的途径给予解决。而在 dBASE Ⅲ 中，则更有条件应用数组。

参　考　文　献

〔1〕王电，汉字 dBASE Ⅱ 实用技术，中国计算机用户协会总会，计算机应用通讯编辑部，（1984）。
〔2〕中西文 dBASE，上海微电脑厂，（1985）。
〔3〕欧阳枫，dBASE Ⅲ 与汉字 dBASE Ⅲ 简介，计算机世界月刊，9（85）。

The Proper Use of Some Statements in Chinese and western Languages dBASE II

Zhang Yinming

Abstract

On the basis of the author's practice in the application of dBASE II, this paper analyzes the problems existed in the statements of TOTAL, DO WHILE, etc. and presents the method for their solution and gives more precise direction for their application. It discusses also the use of array which is helpful to the programming of dBASE.