FoxBASE+索引功能的扩展

广州军区后勤部自动化工作站　　戴林清

摘要：索引是 *FoxBASE+*按特定顺序快速排列数据库记录的有效手段，然而其功能的局限性和实际应用中莫明其妙的失效常常令用户束手无策。本文介绍在多重索引下日期型字段和带负值数值型字段的正确索引方法，以及实现双向索引的途径和通用程序。

一、多重索引下负值字段索引偏差的纠正

对 FoxBase+数据库进行多重索引时，必须将数值型字段用 STR()函数转换成字符型，在取值不含负数的情况下，这种转换后所建索引的正确性是无须置疑的，但在取值含负数时，索引所表现的记录逻辑顺序有可能完全错误！设某数据库含有一数值型字段 V(N,6,2)，库中共有四条记录，有的记录在 V 上的取值为负数。打开该数据库并对其进行以下操作：

```
.LIST V
RECORD#          V
      1     -2.00
      2      5.00
      3    -12.00
      4     23.00
.INDEX ON STR(V,6,2) TO TEMP
.LIST V
RECORD#          V
      2      5.00
      1     -2.00
      4     23.00
      3    -12.00
```

由此可以看出，经过索引后的记录顺序并不象人们想象的那样，按字段 V 的取值从小到大依次排序。为什么会出现上述情况呢？这与 STR()函数的特性及某些字符的 ASCII 值有关。STR(F,M,N)将数值型值 F 转换成字符型值时，其结果是右对齐的，如果 F 的实际宽度达到不到 M，则转换后字符串前部(即左边)以空格补齐。由于负号"-"的 ASCII 码值为 45，而空格的 ASCII 码值为 32，按照字符串的大小比较规律，"-12.00"要比"23.00"大，故上例中字段 V 取值为-12.00 的记录理所当然地就排在字段 V 取值为 23.00 的记录后面了。同样的原因也使 V 值为-2.00 的记录排在了 V 值为 5.00 的记录后面。因此，负号的 ASCII 码值大于用来补足前部空位的空格符 ASCII 码值，是造成索引后记录不能正确排列的原因。

对症下药，纠正方法不外乎两种：一是不用空格符而用 ASCII 码值大于负号的字符来补足前部空位；二是使索引表达式中不出现负值。

使用第一种方法时，可用 ASCII 码值为 47 的字符"/"来填补前部空位(负号的 ASCII 码值为 45)。见以下例句，其中 LTRIM(STR(V,6,2))是将字段值转换成字符串后去掉前部这格，6-LEN(…)为前部空格数，REPL('/'…)则将前部空格替换成字符"/"：

```
.INDE ON REPL('/',6-LEN (LTRIM (STR(V,6,2,))))+
    LTRIM (STR(V,6,2)) TO TEMP
```

使用第二种方法时，可先将字段值加上取值上限值后再进行索引，为防止数值溢出，用 STR()函数转换时所取位长应比字段长大 1，见以下例句：.INDE ON STR (V+999.99,7,2) TO TEMP

经过上述两种方法的处理，应能保证负值字段在多重索引情况下应正确索引。相比之下，第二种方法更简练直观，建立索引的速度也来得快些。

二、多重索引下日期型字段的正确索引

同理，在复合关键字情况下对日期型字段进行索引时，必须用 DTOC()函数转换成字符型，如果不加任何措施，这种转换后所建索引有可能是不正确的！设某数

据库含有一个日期型字段 V(N,D,8),库中共有三条记录,打开该数据库并对其进行以下操作:

```
.LIST V
RECORD#      V
      1      02/10/89
      2      01/01/92
      3      10/01/87
.INDEX ON DTOC(V) TO TEMP
.LIST V
RECORD#      V
      2      01/01/92
      1      02/10/89
      3      10/01/87
```

可见索引后的记录顺序完全是错误的。原因归结于日期的格式和 DTOC()函数的特性。FoxBASE+允许用户根据习惯设置不同的日期格式,共有以下几种:

日期类型	日期格式
AMERICAN	MM/DD/YY
ANSI	YY.MM.DD
BRITISH	DD/MM/YY
ITALIAN	DD-MM-YY
GERMAN	DD.MM.YY

默认值一般为美国类型(即"月/日/年"),如前例中所见。DTOC()函数是按相应格式将日期型值转换成字符型值的,其结果使得上例中的记录排列是依次按月、日、年的优先级来进行的,而不是按正确的年、月、日顺序排列。

解决的方法是索引前先用 SET DATE 命令将日期设置为"YY.MM.DD"格式:

```
.SET DATE ANSI
.INDE ON DTOC(V) TO TEMP
```

三、双向索引功能的实现

FobBASE+提供的索引功能只能按关键字值从小到大的升序方式排列,而不能关键这段值从大到小的降序方式排列,这不能不说是美中不足。在实际应用中常常需要以降序方式排列数据库记录,比如考试成绩的名次表就是按分数从高到低的顺序排列的,人们通常是借助 SORT 命令来完成降序排列,但 SORT 是通过改变数据库记录的物理位置顺序来实现排序的,时间开销庞大,在数据库记录较多时完成排序所需时间将令人难以忍受。因此在命令响应速度要求较快、排列方式随机多变的环境中(如随机查询或随机打印程序中)用 SORT 命令来完成排序的做法是不明智的,必须另辟捷径。

众所周知,两数相减,当被减数固定不变时,减数越大则差越小,反之减数越小则差越大。此时如果将一组减数按降序方式排列,则其对应差一定是按升序方式排列的,反之亦然。将这一简单原理用于 FoxBASE+数据库索引,我们可以圆满地实现数据库的双向索引:需要按某字段降序排列时,以该字段的取值上限为固定被减数,以实际取值为减数,对差进行索引,此时记录是按差的升序方式排列,亦即按该字段的降序方式排列;需要按升序对某字段排列时,则以常规方式进行索引。

下面以具体实例来说明实现降序索引的方法。设某数据库包含三个字段:F1(N,3,1)、F2(D,8)、F3(C,3)。按这段 F1 降序方式建立索引:

```
INDE ON 9.9-F1 TO TEMP
```

按字段 F2 降序方式建立索引:

```
SET DATA ANSI
INDE ON CTOD('99.12.31')-F2 TO TEMP
```

按字段 F3 降序方式建立索引:

```
INDE ON CHR(255-ASC (SUBS(F3,1,1)))+ CHR(255-ASC
    (SUBS(F3,2,1))) + CHR (255- ASC
    (SUBS(F3,3,1))) TO TEMP
```

以 F3+F1+F2 为复合关键字、每个字段均按降序方式建立索引:

```
SET DATA ANSI
INDE ON CHR(255-ASC (SUBS(F3,1,1)))+ CHR(255-ASC
    (SUBS(F3,2,1))) + CHR (255- ASC (SUBS(F3,3,1))) +
    STR (9.9-F1,4,1)+ STR(CTOD('99.12.31')-F2,5) TO
TEMP
```

需要指出:字符型字段的最大值为长度与之相同、每位均为 CHR(255)的字符串,由于 FoxBASE+不具备两字符串按位相减的函数,自定义函数又不能出现在索引表达式中,故字符型字段需建立降序索引时只能按位形成索引表达式;两个日期型数量可以相减,其结果是两个日期间相差的天数,它是数字型值而不是人们想法中的日期型;

笔者设计了一个建立双向索引的通用程序，可以挂靠在任何程序的任意处，只要在调用前打开数据库并将其置为当前区，就能灵活地建立起双向索引并打开索引库，此后记录的逻辑顺序即为用户所需，运行效果相当满意。该程序从调用参数中获取关键字段及其升降序信息，自动构造索引表达式，如果表达式不超长则直接索引，否则给出错误信息退出运行。程序兼顾了日期型字段和带负值数值型字段的正确索引。调用参数写法规则如下：每个关键字段依次对应一个子项，关键字段多于一个时，子项间用逗号相间；每个子项以字段名+"/"+"A"(或"D")组成，其中升序时用 A，降序时用 D。以下是程序调用例句：

```
DO IDXPRG WITH "F1/D, F2/A, F3/D"
```

程序清单如下，为简单起见，省略了参数正确性检测部分。

```
*通用双向索引程序 IDXPRG.PRG
PARA FNAME &&参数:字段名及升降序要求
SET SAFE OFF
SET EXAC ON &&比较为精确方式
SET DATE ANSI &&置日期形为"YY.MM.DD"
FILENAME=DBF() &&保存当前数据库文件名
COPY STRU TO TEMP1 EXTE &&生成结构库
USE TEMP1 &&打开结构库同时关闭原数据库
MEN="" &&记录索引表达式
LEN=0 &&记录索引表达式长度
MFN=LTRIM(TRIM(UPPER(FNAME))) &&规则参数值
DO WHIL MFN<>"" &&依次处理各关键字段
AI=AT(',',MFN) &&首个分隔符位置
IF MI<>0 &&子项分隔符存在时
MF=SUBS(MFN,1,MI-1) &&当前关键字段项
MFN=SUBS(MFN,MI+1) &&参数剩余部分
ELSE &&分隔符不存在时
MF=MFN &&最后一个关键字段项
MFN="" &&参数剩余部分为空
ENDI
FS=UPPER(RIGHT(MF,1)) &&升降符
MF=LEFT(MF,LEN(MF)-2) &&当前关键字段名
FLEN=LEN(MF) &&字段名长度
LOCA FOR TRIM(FIELD_NAME)=MF &&字段记录定位
ML=FIELD_LEN &&该关键字段宽度值
MD=FIELD_DEC &&该关键字段小数位
MEXP="" &&索引表达式中当前字段所构成部分
IF FIELD_TYPE='C'.AND.FS='D' &&字符字段降序
LEN=LEN+ML*(26+FLEN) &&索引长度递增
SJ=1 &&索引表达式依次加入每位上的部分
DO WHIL SJ<=ML
MEXP=MEXP+'+CHR(255-ASC(SUBS('+MF+',SJ,1)))+LTRIM(STR(SJ,2))+',1)))'
SJ=SJ+1 &&继续处理下一位
ENDD
ELSE
DO CASE
CASE FIELD_TYPE='C' &&此处肯定是升序情形
LEN=LEN+1+FLEN &&索引表达式长度递增
MEXP=MEXP+'+'+MF &&升序时直接构造本字段部分
CASE FIELD_TYPE='N'
ZMAX=REPL('9',ML) &&ZMAX为取值上限值式样
IF MD<>0 &&小数位不为0时
ZMAX=STUFF(ZMAX,ML-MD,1,'.') &&确定小数位置
ENDI
IF FS='D' &&降序索引时
LEN=LEN+12+ML+FLEN
MEXP='+STR('+ZMAX+'-'+MF+',7,'+STR(ML+1,2)+',2)+','+STR(MD,1)+1')' &&位长取ML+1防止数值溢出
ELSE
LEN=LEN+11+FLEN
MEXP='+STR('+ZMAX+'+'MF+',7,'+STR(ML+1,2)+',2)+','+STR(MD,1)+')' &&加上最大值防止出现负数
ENDI
CASE FIELDTYPE='D'
IF FS='D'
LEN=LEN+25+FLEN
MEXP='+STR(CTOD("99.12.31")-'+MF+',5)'
ELSE
LEN=LEN+7+FLEN
MEXP='+DTOC('+MF+')'
ENDI
ENDC
ENDI
IF LEN>236 &&以236为表达式超长标准
  24,26,SAY '表达式太长,无法建立索引'
USE &FILENAME &&重新打印原数据库
ERASE TEMP1.DBF &&删除临时文件
SS=INKEY(5) &&等待敲健,时间为5秒
RETU &&索引失败,返回调用处
ENDI
MEX=MEX+MEXP &&将本字段部分加入总表达式
ENDD
MEN=SUBS(MEX,2) &&去头部"+"得完整表达式
USE &FILENAME &&重新打开原库并覆盖结构库
INDE ON&MEX TO TEMP &&建立双向索引
ERASE TEMP1.DBF &&删除临时文件
RETU &&至此已按要求建立并打开索引文件TEMP
```