问题，需要每个系统管理员根据自己系统的配置灵活设置，以实现优化使用。

参考文献：

[1]Getting Started with Microsoft Windows, 1993, 6--96

[2] Microsoft Windows 3.1 User's Guide, 1993, 573--575

*[3]*张公忠、王钰，*Novell* 网组网原理与实用技术。北京：清华大学出版社，*1992.363--370*

FoxPro2.5 中实现特殊表达式的通用运算

罗　辉　(湖南省双峰工商银行)

艾工建　(湖南省娄底工商银行)

一、问题的提出

正如我们所熟知的，FoxPro2.5 与其它程序设计语言一样，具有相当强的表达式计算能力。它提供了许多的表达式计算函数和命令，因而能完成你所希望完成的几乎所有对数据库记录进行运算的需要；尤其它的宏代换命令(&)和名字表达式，更能使你编程完成某些通用性的运算要求。但是，这些都只是完成一些对数据库字段进行基本的加减乘除等四则混合运算任务，实际工作中我们却经常遇到一些涉及到统计计算的运算要求。比如，在某一表格中，经常要进行数据的检查，某一时期使用如下检查公式：

项间检查公式：

第 30 项＝第 1 项至第 12 项之和+第 14 项−第 15 项×100+第 16 项至第 29 项之和；

栏间检查公式：

第 6 栏＝(第 1 栏至第 3 栏之和−第 4 栏)×第 5 栏。

这些表达式不是简单的四则混合运算表达式，不能简单地进行一般的表达式运算。更甚，可能在另一个时期，由于情况发生了变化，表格的第 17 项将调到第 10 项前，相应的项间检查公式将发生变化，变成：

第 30 项＝第 1 项至第 13 项之和+第 15 项−第 16 项×100+第 17 项至第 29 项之和。

银行的诸多会计报表，就经常如此。对于这样的情况，我们当然可以一一编程完成所有的运算，但我们也应该看到，当这样的情况经常出现时，势必你的运算程序必须经常修改，以适应这样的变化。如果系统的操作者就是系统开发者那还好说，如果你的系统要向多个单位推广，靠修改程序来适应需要就行不通了。对这类特殊表达式，我们可以寻求一种通用的运算方式，将可任意设定的特殊运算表达式用一个公式库保存起来，运算时通过程序对公式库中的特殊表达式进行正确的识别，完成运算。本文主要介绍特殊表达式的识别和运算，这可借助于逆波兰表达式算法来实现。

二、逆波兰表达式转换和运算思想

通常，我们是将表达式用中置法来表示，二元运算符总是写在两个运算分量的中间。这种表示方法本身不能指出求值过程中的运算顺序，必须依赖关于运算符优先级与结合律等背景知识才能正确运算，而且还常常需要使用括号。程序实现起来相当复杂。因而实际运算中我们一般使用表达式后置法即转换为逆波兰表达式来完成计算，它可完全避免使用括号，也不再依赖优先级等概念，计算顺序也完全由表达式形式所确定。

譬如：表达式(a−b／c)＊d+e 用后置法表示为：abc／−d＊e+。对用逆波兰表达式表示的表达式求值按如下规则进行：自左向右扫描逆波兰表达式，每遇到一个 n+1 元组(opd1,opd2,...,opdn,opr)其中 opd1 至 opdn 为操作数，opr 为 n 元运算符，就计算一次 opr(opd1,opd2,...,opdn)的值，其结果取代原来表达式中 n+1 元组的位置，再从表达式开头重复上述过程，直到表达式中不再含有运算符为止。

为此，首先必须将通常用中置法表示的表达式转换为逆波兰表达式，然后再对逆波兰表达式自左至右地扫描运算，最终得出结果。

鉴于在 FoxPro2.5 环境的一般报表需要，我们只对扩充的二元运算符(和+、减−、乘×、除%、累加和-等)及左右园括号进行考虑。所有的运算符都可操作数据库记录，其中，运算符-是对连串记录字段进行累加和运算。对于其它统计运算如平均值等可仿照加入。

逆波兰表达式转换及运算过程，可方便地利用堆栈结构来实现。FoxPro2.5 没有提供堆栈概念及其操作命令或函数，我们可以借助系统的数组功能实现堆栈数组结构，提供堆栈的初始化 newstack、压栈 push、出栈 pop

等命令及栈空判别函数 stkempty()。

在逆波兰表达式中先出现的运算符先运算，因此在扫描普通表达式中读到的运算符并不能确定是否进行该运算，而要与后面的运算符作优先级比较。在扫描普通表达式当中暂时不能输出的运算符将保存到栈中。每扫描到一个运算符就与栈顶元素进行比较，若优先级高于栈顶元素则进栈，否则弹出栈顶元素，然后再与新的栈顶元素比较，依此类推。对括号则比较特殊：当读到"("显然应当入栈，但若栈顶元素为"("，那么读到任何运算符以及"("都应入栈，"("的运算优先级与它入栈的优先级不相同，见下表。

运算符	+	−	×	%	•	(
运算优先级	1	1	2	2	3	4
入栈优先级	1	1	2	2	3	0

")"没有优先级，一旦读到")"，栈一直退到第一个遇到的"("出栈为止，如果没有"("，则表达式语法有错。

为实现逆波兰表达式的转换和运算，本文程序也借助了两个堆栈：栈 1 和栈 2。栈 2 存放转换后的逆波兰表达式，栈 1 用于脱去普通表达式中的括号和确定运算符优先级。

转换完后，逆波兰表达式自左至右地在栈 2 中自底向顶地保存，其运算过程就是对栈 2 自底向顶地扫描该逆波兰表达式，每遇到一个操作数就压入栈 1 保存，每遇到一个运算符就从栈 1 中弹出栈顶的两个操作数进行运算，然后将结果压入栈 1 中；如此继续扫描直至处理完毕，则栈 1 顶部的值就是表达式运算的值。

三、通用运算程序设计实例

根据上面的思想，我们设计了一个对表格数据进行项间、栏间和点间检查的通用程序。为简便计，其通用性仅表现在对一具体数据库，你可任意改变其数据检查公式，也不需修改程序。对它稍加修改，即可实现对任意数据库检查的通用性。本文中仅列出通用项间检查的程序，栏间和点间检查及其检查公式的设置程序略。

程序中 data0101.dbf 是报表数据的数据库文件，其文件结构在 datast01.dbf 结构库中，结构库有五个字段：FIELD_NAME、FIELD_TYPE、FIELD_LEN、FIELD_DEC、FIELD_CNAM，分别对应 data0101.dbf 的字段名、字段类型、字段长度、小数点位置、字段汉字名。

项间检查公式用 calagsx.dbf 公式库储存，其结构是：xno(计算记录号)、formulax(检查计算公式)。如前面第一个项间检查公式对应于公式库中一条记录：

xno＝30、formulax＝R1−R12+R14−R15＊ 100+ R16−R29

检查公式中，对数据库记录数据的引用通过在记录号前加一个标记字符"R"来实现，除此之外，公式中只能出现 0-9、小数点、"+−＊／−"等符号，如果再出现其它字符则程序认为公式有误。程序也能对记录号的合法性进行一定的检查。

对于 data0101.dbf 中记录数据，当其项间检查公式发生变化时，只需将公式的变化简单地反应到公式库中，程序不需作任何修改即可完成其公式的运算。

程序中，xjjs()函数完成逆波兰表达式的转换和计算，其中的计算具体调用 nbljs()函数完成。

```
SET TALK OFF
IF USED('datast01')
    SELECT datast01
ELSE
    SELECT 0
    USE datast01
ENDIF
IF USED('data0101')
    SELECT data0101
ELSE
    SELECT 0
    USE data0101
ENDIF
IF USED('calagsx')
    SELECT calagsx
ELSE
    SELECT 0
    USE calagsx
ENDIF
GO TOP
DEFINE WINDOW datack AT 0.000, 0.000 SIZE 12.100,62.500 ;
    TITLE "数据检查窗口" FONT "宋体", 12 ;
    FLOAT CLOSE nominimize DOUBLE COLOR RGB(0,0,0,128,0,128)
MOVE WINDOW datack CENTER
ACTIVATE WINDOW datack NOSHOW
@ 4.200,18.000 GET m.choice4 ;
    PICTURE "  ＊VN 项间纵向检查;＼＼栏间横向检查;＼＼点间检查" ;
    SIZE 1.550,24.000,0.500 DEFAULT 1 FONT "宋体", 12
VALID_qz30zo9h7()
ACTIVATE WINDOW datack
READ CYCLE DEACTIVATE_qz30zocna()
```

```
RELEASE WINDOW datack
FUNCTION_qz30zo9h7 && m.choice4 变量的 VALID 子句
SELECT calagsx
SCAN      &&公式记录循环
    gs=ALLTRIM(UPPER(formulax))
    SELECT datast01
    GO TOP
    SCAN FOR fldslt=2     &&字段循环
      IF field_type#'N'     &&跳过非计算字段
        LOOP
      ENDIF
      jsfld=field_name
      jslen=field_len
      jsdec=field_dec
      jscname=field_cname
      SELECT data0101
      GO calagsx.xno        &&跳到要计算的记录上
      fldvalue=&jsfld.      &&字段值
      jsvalue=xjjs(calagsx.formulax)  &&计算值
      IF fldvalue=jsvalue                &&核对相符
        WAIT WINDOW ;
          jscname+STR(calagsx.xno,4)+'号 记 录 检 查  OK!'
NOWAIT
      ELSE                               &&核对不相符
        WAIT WINDOW jscname+STR(calagsx.xno,4)+'号 记
录! 字段值:';
          +STR(fldvalue,jslen,jsdec)+'计算值:'+;
          STR(jsvalue,jslen,jsdec)+'差额:'+;
          STR(fldvalue-jsvalue,jslen,jsdec)
      ENDIF
      SELECT datast01
    ENDSCAN
ENDSCAN
WAIT WINDOW '检查结束! ' NOWAIT
FUNCTION xjjs    &&逆波兰表达式转换函数
PARA gsexpc      && gsexpc—检查公式的表达式
SELECT data0101
stlen=200    &&设堆栈容量为二百个元素,可按实际需要调整
DIMENSION stk(stlen)
DO newstack
stb=''
stl=.F.
stc=0
STD=0
gsexpc=STRTRAN(ALLTRIM(gsexpc),' ',',1)
gsexpc=UPPER(gsexpc)
DO WHILE LEN(gsexpc)#0       &&公式循环
    stb=SUBSTR(gsexpc,1,1)
    gsexpc=RIGHT(gsexpc,LEN(gsexpc)-1)
    IF stb='R'          &&开始是记录号标记
      tmp=SUBSTR(gsexpc,1,1)
      IF .NOT. BETWEEN(tmp,'0','9')
        WAIT WINDOW '公式缺乏记录号! '
        RETURN
      ENDIF
      DO WHILE BETWEEN(tmp,'0','9')      &&读出记录号
        stb=stb+tmp
        gsexpc=RIGHT(gsexpc,LEN(gsexpc)-1)
        tmp=SUBSTR(gsexpc,1,1)
      ENDDO
      IF RECCOUNT()< VAL(RIGHT(stb,LEN(stb)-1))   &&记
录号越界
        WAIT WINDOW '记录号越界! '
        RETURN
      ENDIF
ELSE
      IF BETWEEN(stb,'0','9')      &&开始是数字
        tmp=SUBSTR(gsexpc,1,1)
        DO WHILE BETWEEN(tmp,'0','9') .or. tmp='.'  &&
读出完整的数字
          stb=stb+tmp
          gsexpc=RIGHT(gsexpc,LEN(gsexpc)-1)
          tmp=SUBSTR(gsexpc,1,1)
        ENDDO
      ELSE
        IF .NOT. stb
'()+- * / -' 不是记录号和操作数也不是操作符
          WAIT WINDOW '公式有非法符号! 请核查。'
          CLEAR GETS
          HIDE WINDOWS datack
          RETURN MASTER        &&返回主画面
        ENDIF
      ENDIF
ENDIF
DO yxj          &&如为运算符,取优先级,否则压入栈 2
stelm=stb
IF stl                &&输入是操作数
    logical=.T.  &&对栈 2 操作
    DO PUSH
ELSE                  &&输入是操作符
    logical=.F.   &&对栈 1 操作
    IF stb='('        &&输入是左括号
      DO PUSH
      STD=stc     &&存栈 2 顶部元素优先级
    ELSE
      IF stb=')'    &&输入是右括号时
        DO TOP
        DO WHILE stelm#'('
        DO POP        &&弹出栈 1 顶部元素
        logical=.T.
        DO PUSH       &&存入栈 2
        logical=.F.
        DO TOP
      ENDDO
      DO POP          &&脱栈 1 中开括号
```

```
      STD=stc
    ELSE                &&既不是左也不是右括号时
      IF stc>STD        &&输入项优先级较高时
        DO PUSH
        STD=stc
      ELSE              &&输入项优先级较低时
        sth=stb     &&存当前输入项
        stg=stc
        DO WHILE stg<=STD .AND. .NOT. stkempty (top2,
top1)
          DO TOP
          DO POP
          logical=.T.
          DO PUSH
          logical=.F.
          IF .NOT. stkempty(top2,top1)
            DO TOP
            stb=stelm
            DO yxj
            STD=stc
          ENDIF
        ENDDO
        stelm=sth       &&恢复当前输入项
        DO PUSH
      ENDIF
    ENDIF
  ENDIF
ENDIF
ENDDO
logical=.F.     &&栈1中运算符后进先出压入栈2
DO WHILE .NOT. stkempty(top2,top1)
    DO TOP
    DO POP
    logical=.T.
    DO PUSH
    logical=.F.
ENDDO
RETURN  nbljs()     &&转换结束
PROCEDURE yxj       &&确定优先级过程
stl=.F.             &&输入项为运算符
DO CASE
CASE stb='('
    stc=0
CASE stb=')'
    stl=.F.
CASE stb='+'.OR. stb='-'
    stc=1
CASE stb='*'.OR. stb='/'
    stc=2
CASE stb='-'&&记录连接符
    stc=3
OTHERWISE
    stl=.T.         &&输入项为操作数或记录号
ENDCASE
RETURN
```

提高 FoxPRO 软件运行速度的若干措施

杜学绘 张 斌 (郑州电子技术学院)

摘要:在信息管理系统(*MIS*)应用开发中,提高应用软件运行速度至关重要,本文介绍在 *NOVELL* 网络环境下提高 *FoxPRO* 应用软件运行速度的几种措施。

FoxPRO 网络数据库在过去几年有了长足的发展,用其编制的应用软件越来越多,也越来越趋大型化。由客户、服务器方式构成的数据库网络在当前十分流行,客户、服务器结构把 DBMS 和应用分开,使网络中某些计算机专门用于执行 DBMS 的功能,这些计算机称作数据库服务器,而其他结点上的计算机安装 DBMS 的外用开发工具(如 FoxPRO 等),支持用户的应用,这些计算机称为客户机。从而构成了客户/服务器方式的数据库网络。

客户/服务器结构的优点在于将微机工作站和小型机、大型机上的 DBMS 的优势有机结合起来,把文件服务器的观点向前推进了一大步,客户机承担应用开发工具和支持应用程序运行的任务,同时它还可以通过网络获得服务器的服务,使用服务器的共享资源,这种把处理任务一分为二的做法,克服了文件服务器的弱点,使用户接口、事务处理、数据管理等项工作统一规划,保障了应用的相对独立性,提高了用户对基础硬件、软件更新换代的自由度。

因此,在客户/服务器方式下开发 FoxPRO 网络数据库必然成为大势所趋,为了满足 FoxPRO 网络数据库大型化趋势对软件的运行速度的要求,需要采用一系列措施,现介绍如下。

一、系统配置

1.硬件环境

一台 486/33 微机作服务器,5 台 386 微机作客户机,NOVELL 网络(NETWARE3.31), DOS6.0。其结构如图所示:

2.内存管理