灵活设计 FOXPRO 的有效性确定子句

罗 辉 (湖南省双峰工商银行)

摘要：本文对 *FOXPRO2.5* 的全屏幕设计命令@…*SAY*…*GET* 的有效性确定 *VALID* 子句的设计进行了分析，并设计了一种将全屏幕编辑域的有效性资源和在线式帮助信息统一管理的方法。

FOXPRO 数据库管理系统较之 FOXBASE+在功能上有了飞跃性的改进或增强。在软件设计的具体编程中，感触最深之一是 FOXPRO 的全屏幕设计命令@…SAY…GET 的功能强大的 WHEN 确认进入子句的 VALID 有效性确定子句。如果对它们运用得好，常常会收到意想不到而事半功倍的效果。本文在 FOXPRO2.5 环境下对@命令的有效性子句 VALID 的设计技巧作了一番较深入的探讨，所有程序都在华达汉字系统 HDDOS1.0 下运行通过。

一、命令@…SAY…GET 的使用格式

FOXPRO2.5 中全屏幕设计命令@…SAY…GET 的使用格式如下：

```
@<row, column>
 [SAY expr][PICTURE expr][FUNCTION list]
 [GET <memvar>    <field>]
 [FUNCTION <expC1>][PICTURE <expC2>]
 [FONT <expc3>[,<expN1>][STVLE <expC4>]
 [DEFAULT <expr1>][ENABLE    DISABLE]
 [MESSAGE <expC5>]
 [[OPEN]WINDOW <window name>]
 [RANGE [<expr2>][,<expr3>]]
 [SIZE <expN2>,<expN3>]
 [VALID          <expL1>   <expN1>[ERROR
<expC6>]]
 [WHEN <expL2>]
 [COLOR SCHEME <expN5>   COLOR <color
pair list>]
```

@命令把光标移到由 row 和 column 指定的屏幕位置上。@命令可以和一个或多个选择项一起使用。SAY 子句显示随后的表达式的值。GET 子句允许编辑内存变量或数据库子段。PICTURE 子句允许使用模型，它指定的数据显示和获取的方式与 SAY 或 GET 子句相对应。FUNCTION 子句使用功能码，它所完成的功能都可由 PICTURE 子句来实现。RANGE 子句和 GET 子句一起使用，以指定编辑域的范围。VALID 子句通过使用一个条件来确定 GET 接受的编辑域的有效性。如果没有 ERROR 子句，输入数据无效时系统显示一个固定的错误信息“Input Invalid”。COLOR 为@命令定义新的颜色对。COLOR SCHEME 是一个从 1 到 11 的数值，指示了基于相应的颜色组合的一种颜色。注意在 WINDOWS 版本的 FOXPRO 中，颜色命令及其选项仅改变用户自定义窗口和菜单中的颜色；WINDOWS 中所有其它部分的颜色必须通过 WINDOWS 的 CONTROL PANEL 中的 WINDOWS 颜色选项来改变。当进入 GET 中编辑域时，将同时在由 SET MESSAGE TO 命令指定的屏幕行显示 MESSAGE 子句中的提示信息。WHEN 是一个逻辑表达式，它确定是否进入该 GET 中编辑。DEFAULT 为 GET 提供一个缺省值。OPEN WINDOW 和备注字段一起使用，为该字段打开一个预定义窗口。DISABLE 用来禁止编辑一个 GET 字段，使光标跳过该字段。SIZE 被用来定义一个不同于缺省值的字段大小。FONT 和 STYLE 子句分别指定编辑域的字体和字型，它们只对 WINDOW 版本有效，DOS 版本将忽略之。

在用全屏幕设计命令@…SAY…GET 编辑一个字段或变量时，可通过 VALID 子句来限定该编辑域的取值范围。当编辑退出该编辑域时，系统将用 VALID 子

句中指定的逻辑表达式、数值表达式或用户自定义函数对输入值进行校验，以决定输入是否有效。

当你选择的是逻辑表达式 EXPL1 时，如果 EXPL1 为假，输入无效，系统将显示一串固定的或用户自定义的错误报警信息，同时将截留在原编辑域等待重新编辑。如 EXPR1 为真即输入有效后，才能退出该编辑域。

而当你选择的是数值表达式 EXPN1 时，如果 EXPN1 值为 0，表示输入无效，光标仍截留在原编辑域；当 EXPN1 的运算结果是一个正值或负值时，表示输入有效，该值指示 READ 语句要激活的下一个 GET 域的相对域数，其中正值表示向后继续，负值向前返回。

二、VALID 子句的一般设计

1.在程序中直接嵌入有效性确定条件

当程序简单、输入输出屏幕也不复杂且用于 VALID 子句有效性确定的条件也简单时，可简单地通过在程序中直接嵌入该 VALID 条件实现。如程序一。

＜程序一:EXAMPLE1.PRG＞

```
set talk off
clear
age=0
@ 1,1 say"请输入年龄:"get age valid ＜age＞ 18.and .age＜120);
    error'年龄超出范围!'
read
return
```

注意：当嵌入的 VALID 条件是算术表达时，输入无效时系统将只截留在原编辑域，并不显示 ERROR 子句中的错误信息。因此在可能时，最好使用逻辑表达式。

2.用自定义函数实现有效性确定

当 VALID 子句的有效性确定条件比较复杂时，直接嵌入@命令将难于实现或会降低程序的可读性，这时可通过一个自定义的有效性确定函数来取代直接嵌入。如程序二，它将完成程序一同样的工作

＜程序二:EXAMPLE2.PRG＞

```
set talk off
public errmsg
errmsg=space(0)
clear
age=0
@1,1 say "请输入年龄:"get age;
    valid V__age (age )error errmsg
read
return
FUNCTION V__age && "年龄"编辑域的有效性确定函数
parameter age
if (age＞30.and .age＜120)
    return.t.
else
    errmsg='年龄超出范围!'
    return.f.
endif
return
```

程序中如果没有 ERROR 错误信息子句，当输入数据无效时，系统将固定显示一个一般性的错误提示："Invalid Input"。你可以设计与上下文有关的错误信息提示，但这时错误信息不应在 VALID 函数中显示，因为这还不能抑制上述系统错误提示的显示，而必须使用 ERROR 子句显示该信息。

三、设计共用的 VALID 函数

如果对每一个编辑域都设计一个有效性确定的 VALID 函数，当编辑域较多时，类似的函数势必相应增多，增加了编程和测试工作量；同时使系统变得庞大，给阅读、修改和维护带来困难。鉴于数据有效性确定过程的相似性，可以建立一个存放编辑域的有效性表达式的数据库，让数据库来统一管理系统所有的有效性确定表达式和相应的出错信息等资源，一方面可以避免上述不便，另一方面也有助于系统的共用性，何乐而不为？

建立如下结构的有效性资源数据库 SVALID. DAF，其中保存有效性表达式的字段 VALIDEXPR 取 254 字节长，它一般可满足要求。其中的表达式必须是算术或逻辑表达式。

FieldName	Type	Width	Dec	
EDITFIELD	C	12		编辑域名
VALIDEXPR	C	254		该域的有效性确定表达式
ERRMSG	C	40		相应出错信息

实现的程序如程序三。此时 SVALID 库中的参考记录如下：

```
Record EDITFIELD VALIDEXPR   ERRMSG
1      XM                    (无有效性确定子句)
2      NL        iff <NL> 17. and .NL<70, 1,0> 年龄超出范围(算术表达式)
3      XL        XL='Q'. OR. XL='Z'. OR. XL='B'学历非法   (逻辑表达式)
```

注意:SVALID 数据库须在 EDITFIELD 字段建立索引(索引文件名是 SVALID),由于函数 VARREAD( )总是返回编辑域的大写,因此 EDITFIELD 中的编辑域名必须全部大写,否则可能会因搜索不到编辑域而总认为数据输入有效。

<程序三:EXAMPLE3.PRG>

```
set talk off
set message to 24
public errormsg
clear
errormsg=space(0)      &&编辑域初始化
xm=space(6)
nl=0
xl=spaoc(1)
select 25
use svalid index svalid
define window inwind from 8,1 to 20,78;
    shadow title"数据输入窗口"
activate window inwind
@2,1 say PADC("--职工档案--",WCOLS())  &&保持信息总在窗口行的中间
@3,4 to 12,74 Double
@ 5,10 say"姓 名 :"get xm valid F__VALID()error errormsg
@7,10 say "年龄:"get nl pict'99' valid F__VALID()error errormsg
@9.10 say "学历:"get x1 pict '1'valid F__VALID()error errormsg;
    message "B-本科 Z-专科 Q-其它"
read
clear windows
close data
return

function f__valid
select svalid
seek VARREAD()
if FOUND()
    vldexpr=svalid->validexpr
    if empty(vldexpr)
        return. t. &&如果没有有效性确定表达式
    endif
    do case
    case TYPE(vldexpr)='L'   &&是逻辑表达式时
        if (&vldexpr.)   &&进行有效性逻辑判断
            return.t.
        else
            errormsg=errmsg
            return.f.
        endif
    case TYPE(vldexpr)='N' &&是算术表达式时
        return EVALUATE (vldexpr) &&返回有效性运算结果
    endcase
else   &&有效性资源库中无此编辑域记录
    return.t.
endif
```

四、VALID 函数与在线式帮助统一设计

在进行全屏幕数据输入时,我们往往不能满足于输入出错时的错误信息提供的帮助,而希望提供在线式帮助信息。当然它可以如程序三中设计 ERROR 子句一样,在 SVALID 数据库中增添一个字符型的帮助信息字段 HELPMSG,对程序三略作修改,仿照错误信息的处理方式,利用 MESSAGE 子句即可提供及时响应的帮助信息。

然而这一方式也不尽如人意,一是它提供的帮助信息只有一个屏幕行,十分有限;二是我们并不总是需要及时响应的帮助信息,比如当操作者对该编辑域相当熟悉了时,反而很讨厌从一个编辑域跳到另一个编辑域时引

起帮助信息的屏幕行切换的闪动，这时，可能更希望系统提供另一种在线式帮助：仅仅在需要时，通过按某一个热键（如 F1 键），才激活帮助。一个替代的方法是借用系统的帮助机制，建立一个帮助数据库，该帮助数据库的前二个字段必须分别满足：第一个字段是字符类型，作为系统的帮助主题进行索引；第二个字段是备注类型，其内容是相应主题的帮助信息。而且顺序不能颠倒。在程序中通过 SET HELP TO 和 SET TOPIC TO 命令启用 FOXPRO 固有的 HELP 模式显示该库提供的应用系统帮助信息。但是，这一方式不能实现有效性确定资源与相应帮助信息统一用一个数据库管理，因为这一方法 SET HELP TO 命令指定的帮助库必须单独使用。

另一种解决方法是自制帮助系统。建立如下一个将有效性确定表达式和在线帮助信息合为一体的数据库，数据库结构如下：

FiledName	Type	Width	Dec
EDITFIELD	C	12	编辑域名(帮助主题)
HELPMSG	M	10	帮助信息
VALIDEXPR	C	254	该域的有效性确定表达式
ERRMSG	C	40	相应出错信息

利用 FOXPRO 的热键陷井激活系统的在线帮助机制，自编的在线式帮助子过程实现帮助。从程序四可略见端倪。

＜程序四:EXAMPLE 4.PRG＞

```
set talk off
public errormsg
clear
errormsg=space(0)    &&初始化
xm=space(6)
n1=0
x1=space(1)
select 25
use svalid index svalid
define window inwind from 3,1 to 16,78;
   shadow title"数据输入窗口"
activate window inwind
define windows helpwind from 18.1 to 24,78;
title"帮助窗口按 ESC 退出帮助"
on key =315 do deithelp &&设置 F1 键为热键
   @2,1 say PADC("--职工档案--",WCOLS())
   @3,5 to 12,74 Double
   @5,10 say "姓 名:" get xm valid F__VALID() erro
errormsg
   @7,10 say "年龄:" get n1 pict '99' valid F__VALID(
error errormsg
   @9,10 say "学历:" get x1 pict '!' valid F__VALID (
error errormsg
   read
   clear windows
   close data
   on key
   return
   function f__valid    &&有效性确定函数
   select svalid
   seek VARREAD()
   if FOUND()
   vldexpr=svalid->validexpr
   if empty(vldexpr)
      return.t.    &&如果没有有效性确定表达式
   endif
      do  case
      case TYPE (vldexpr)='L'
      if (&vldexp.)    &&进行有效性逻辑判断
        return.t.
      else
        errormsg=errmsg
        return.f.
      dndif
      case TYPE (vldexpr)='N'
      return EVALUATE(vldexpr)    &&返回有效性运算
结果
      endcase
    else    &&有效性资源库中无此编辑域记录
      return.t.
      endif
      procedure edithelp    &&帮助过程
      select svalid
      seek varread()
```

（下转第 64 页）

(上接第 40 页)

```
if found () &&如该编辑域有帮助则显示帮助信息
  activate windows helpwind
modify memo helpmsg noedit window helpwind
  deactivate window helpwind
else
  wait "对不起,本编辑域无帮助。"window
endif
```

五、结束语

FOXPRO 的全屏幕设计命令 …SAY…GET 的 VALID 子句的功能相当丰富,本文对它的编辑域从动性控制,在 VALID 函数中加进其它处理或控制等设计技巧未予讨论,读者可视具体需要自行设计之。总之,巧妙地运用 VALID 子句,将可以为你的系统设计提供强有力的帮助。

参考文献:

[1]《新一代 FOX 数据库及其实用工具》,海洋出版社,严晓舟等

[2]《FOXBASE+/ MAC 参考指南》,国防科大 MAC 发展中心