采用资源管理器的方式实现信息资源的管理

中国科学院空间科学与应用研究中心　宋友刚

Windows资源管理器具有直观、方便、层次结构明显的特点，我们在管理信息系统(MIS)的设计时能否也以同样的思路实现信息资源的管理，从而提供给用户更加直观方便的操作界面呢?本文就此提出了一种设计方法。其中最关键的数据库基表的设计，受到了C语言指针结构的启发。

问题的提出

Windows资源管理器具有直观、方便、层次结构明显的特点，那么在进行管理信息系统(MIS)设计的时候，是否也能够以这种思路进行设计，从而突破固有的表格、弹出式窗口的设计思路，充分利用其层次结构明显的特点，提供给用户更加直观、方便的界面呢?为此，我们利用PowerBuilder提供的TreeView、ListView控件，结合底层数据库的设计来实现此思路的设计。

设计方法

为了能够明确的解释此设计方法，以某一部门的人员信息管理为例。假设此部门的组织结构如图1所示:

由于PowerBuilder中的TreeView控件所提供的函数可以对其中的item进行添加和删除，但在应用程序关闭后，无法保存住信息。为此，我们需对此设计一个基表来记录TreeView中的层次结构以及所包含的数据，如图2所示:

此结构有些类似C语言中的指针: 对于一个item，“上一级代码”列中存放在TreeView中此item的上一级item的标识代码，“本级名称”列中存放此item的名称(Label)，“本级代码”列中存放着此item的标识代码，“本级类型”列中存放着此Item的标识类型(是级别则标识其层数; 是人员则置标识位0)，为了写程序的方便，我们在最后增

图 1　XX公司的人员组织结构

图 2　基表结构

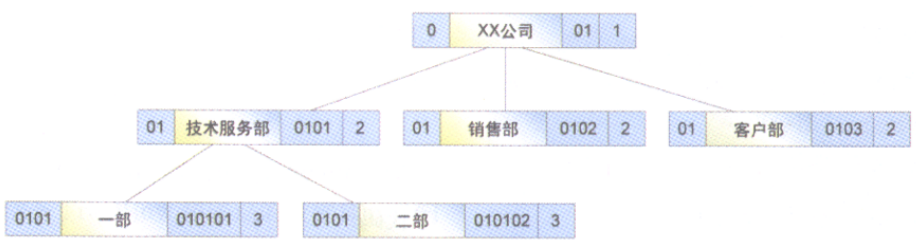

图 3　人员组织结构关系

加了两个列:“本级句柄信息”和“本级的实际代码”,分别存放此item在TreeView中的句柄值和其实际代码(即用户自己定制的编码)。举例来说,对于如图1中的XX公司,我们在基表t_rygx中的前四列中存放如下信息(图3),对于相应的后两列(本级句柄信息和本级实际代码),我们需要在程序运行时进行实时更新。

有了此基表,我们可进一步设计其界面,为其编写代码,如图4所示:

图 4 利用TreeView和ListView控件实现界面的设计

在其中设置了两个控件:TreeView(tv_1)和ListView(lv_1)。为了在应用程序打开时,将数据库中的信息检索到TreeView中,我们在此窗口的Open事件中调用如下窗口函数wf_gettvdata:

```
.....
// 向treeview加入第一层
DECLARE cur_first CURSOR FOR
SELECT   t_tygx.name,
         t_rygx.code,
         t_rygx.type
    FROM t_rygx
  WHERE t_rygx.last_id='0';
OPEN cur-first;
DO WHILE SQLCA.sqlcode <> 100
   FETCH cur_first into :ls_name,:ls_nextid,:ls_type;
   if SQLCA.sqlcode=100 then exit
   ll_picindex=1
   if ls_type='0' then ll_picindex=2
   ll_first=tv_1.insertitemlast(1,ls_name,ll_picindex)
   ls_first=string(ll_first)
   // 更新t_rygx的句柄信息列
   UPDATE t_rygx
      SET handle=:ls_first
   WHERE t_rygx.code=:ls_nextid;
   // 向treeview加入第二层
   DECLARE cur_second CURSOR FOR
      SELECT t_rygx.name,
             t_rygx.code,
             t_rygx.type
      FROM t_rygx
      WHERE t_rygx.last_id=:ls_nextid;
      OPEN cur_second;
      DO WHILE SQLCA.sqlcode <> 100
        FETCH cur_second into :ls_name,:ls_nextid,:
              ls_type;
              if SQLCA.sqlcode=100 then exit
              ll_picindex=1
              if ls_type='0' then ll_pecindex=2
              ll_second=tv_1.insertitemlast(ll_first,
              ls_name,ll_picindex)
              ls_second=string (ll_second)
              // 更新t_rygx的句柄信息列
              UPDATE t_rygx
              SET handle=:ls_second
              WHERE t_rygx.code=:ls_nextid;
              // 向treeview加入第三层
              .....
      LOOP
      CLOSE cur_second;
LOOP
CLOSE cur_first;
.....
```

在TreeView(tv_1)控件的IItemChanged事件中加入如下代码:

```
.....
DO WHILE ll_hd>0
    tv_1.getitem (ll_hd,treedata)
    listdata.data=treedata.data
    listdata.label=treedata.label
    listdata.pictureindex=treedata.pictureindex
    listdata.overlaypictureindex=treedata.selectedpic-
tureindex
    lv_1.additem(listdata)
    ll_hd=tv_1.finditem(Nexttreeitem!,ll_hd)
LOOP
```

在 ListView(lv_1)控件的 DoubleClicked 事件中加入如下代码

```
.....
//获取当前句柄信息
ll_tvi=tv_1.FindItem (CurrentTreeItem!,0)
ls_tvi-string(ll_tvi)
//获取本级代码、类型

SELECT    t_trgx.type,
          t_rygx.code
    INTO :ls_type,
          :ls_nextid
    FROM t_rygx
    WHERE t_rygx.handle=:ls_tvi;
if ls_type< > '0' then
    w_1.visible=false
    tab_1.visible=false
    lv_1.visible=true
else
    dw_1.visible=true
    tab_1.visible=true
    dw_1.retieve(ls_nextid)
    .....
end if
```

进一步的，结合 TreeView 和底层的数据库，我们可实现其中 item 的任意增加和删除，从而便实现了利用资源管理器的方式浏览整个公司的人员组织结构的设计思路。

剩下的问题是如何将右边人员的个人信息与左边的相应的 item 关联起来，如图 5 所示。

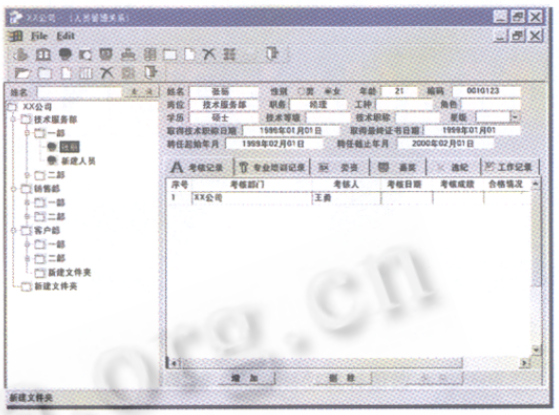

图 5　TreeView 与具体信息的关联

其实，我们通过在基表 t_rygx 的 type 列放置标识 0 来标识此记录为人员，并通过在新建人员记录时在相应人员信息基本数据表中放置相应的本级代码即可实现其关联。

结束语

此设计思路突破了传统的弹出式窗口设计，实现了对资源的更加直观、明晰、灵活的管理，同时简化了用户的操作使用步骤，更易于为用户所接受。■

参考文献

1 http://www.sybase.com/powersoft

2 《Oracle8 完全参考手册》（美）Koch Loney，K.著；梅刚等译。北京机械工业出版社，1998.8