Delphi控件的使用经验计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/622/2021_2022_Delphi_E6_8E_A7_E4_c97_622344.htm ---- 一.Delphi中树型控件的使用技巧 ---- 我们都知道，开发者主要用Delphi来开发数据库管理软件，正因如此，树型控件的使用最好与数据库联系起来。Delphi提供了一个树型控件TTreeView，可以用来描述复杂的层次关系。 ---- 1.树节点信息的存储和加载 ---- 常用的方法是用树控件的 LoadFromFile和SavetoFile方法，来实现树控件和文件之间的交互；或用Assign方法实现树控件和DBMemo，也就是和数据库间的交互。该方法的优点是编程相对简单，缺点是树控件的实际节点数可能会很大，对于\"大树\"，每次加载和存储的数据量会加大，将降低速度，增大系统开销，造成数据冗余。另一种方法，就是只在树上产生\"看得见\"的节点，没有专门记录全部树节点结构的文件或数据库字段，而将树节点结构分散在数据库的每一个记录中。 ---- 具体方法是：创建一个数据库，字段根据实际业务而定，其中必然有一个字段的信息将在树型控件的节点上显示，另外还要一个字段来保存节点的惟一标识号，该标识号由长度相等的两部分组成，前段表示当前节点的父节点号，后段表示当前节点的节点号，此标识号相当于一个\"链表\"，记录了树上节点的结构。该方法的优点：用户操作\"大树\"时，一般不会展开所有的节点，而只用到有限的一部分，同时只能从树根一层一层地展开，该法只在树上产生\"看得见\"的节点，所以，存储和加载\"大树\"的速度快，数据量小，系统开销和数据冗余较小。缺点：编程较复杂，但可以结合该方法编成一个新的

树控件，将大大提高编程效率。值得注意的是，ID号必须惟一，所以在编程中如何合理产生ID尤为重要。 ---- 2.数据库结构示例 ---- 创建一个数据库，为简化程序，我只创建两个数据库字段，定义如下： 字段名 类型 长度 Text C 10 LongID C 6 ---- LongID字段实际上由两段组成，每一段3位，LongID只能表示1000条记录。将LongID定义为索引字段，存为c: esttree ree.dbf。编辑该DBF文件，新建一条记录，Text字段设为TOP，LongID字段设为\"000\"(3个\"0\"前为三个空格)。 ---- 3.创建演示程序 ---- 在Form1上放置TreeView1、Table1、PopupMenu1、Edit1、Edit2。TreeView1的PopupMenu属性设为 PopupMenu1；Table1的DataBaseName属性设为c: esttree，TableName属性设为tree.dbf，IndexFieldNames属性设为LongID；为PopupMenu1加选单项 Add1和Del1，Caption分别为Add和Del；Edit1用来输入新节点的Text属性值，Edit2用来输入新节点的3位ID号。存为c: esttree reeunit.pas和c: esttree esttree.dpr。在treeunit.pas的Type关键字后加入一行：Pstr:^string.{Pstr为字符串指针} 为Form1的OnCreate事件添加代码： procedure TForm1.FormCreate(Sender: TObject). var p:Pstr.Node:TTreeNode. begin with Table1,Treeview1 do begin open. first. new(p).{为指针p分配内存} p^:=FieldByName(' LongID ').AsString. Node:=Items.AddChildObject(nil,FieldByName (' Text ').AsString,p). if HasSubInDbf(Node) then Items .AddChildObject(Node, ' ' ,nil).{有子节点则加一个空子节点} end. end. ---- HasSubInDbf为自定义函数，自变量为Node，检查节点Node有无子节点，有则返回True，反之返回False，并

在TForm1的类定义里加入原型声明(其它自定义函数的原型也在TForm1的类定义里声明，不另作解释)，函数代码如下：
function TForm1.HasSubInDbf(Node:TTreeNode):Boolean. begin with Table1 do begin Table1.FindNearest([copy(Pstr(Node.Data)^,4,3)＋＇000＇]). result:=copy(FieldByName(＇LongID＇). AsString,1,3)=copy(Pstr(Node.Data)^,4,3). {如数据库里当前记录的LongID字段内容的前3位和 节点Node的Data的后3位相同，则Node应该有子节点} end. end. 为TreeView1控件的OnDeletion事件添加代码，需要指出的是， 不仅调用Delete方法可以触发OnDeletion事件，而且当树控件本身被释放前， 也触发OnDeletion事件，所以，在此处加入dispose(node.data)会很\"安全\"： procedure TForm1.TreeView1Deletion (Sender: TObject. Node: TTreeNode). begin Dispose(Node.Data).{释放节点数据内存} end. 为Add1选单项的OnClick事件添加代码如下： procedure TForm1.Add1Click(Sender: TObject). var p:pstr.Tmpstr:string.i:integer. begin try StrToInt(Edit2.Text). Tmpstr:=Edit2.Text.{注：在实用中，必须用更好的方法来产生ID} except. ShowMessage(＇重新输入Edit2的内容＇). abort. end. with TreeView1 do begin new(p). p^:=copy(Pstr(Selected.Data)^,4,3)＋TmpStr. Items.AddChildObject(Selected,Edit1.Text,p). end. with Table1 do{在数据库里添加记录} begin Append. FieldByName(＇Text＇).AsString:=Edit1.text. FieldByName(＇LongID＇).AsString:=p^. Post. end. TmpStr:=inttostr(strtoint(TmpStr)

＋1). for i:=length(TmpStr) to 2 do TmpStr:=＇0＇＋TmpStr. Edit2.Text:=TmpStr. end. 为Del1菜单项的OnClick事件添加代码如下： procedure TForm1.Del1Click(Sender: TObject). var DelList:TStringList.LongID,NSubLongID:string. begin DelList:=TStringList.create. DelList.Sorted:=True. DelList.Add(Pstr(TreeView1.Selected.Data)^). while DelList.Count&gt.0 do begin LongID:=DelList.Strings[0]. DelList.Delete(0). Table1.SetKey. Table1.FieldByName(＇LongID＇).AsString:=LongID. if Table1.GotoKey then Table1.Delete. if HasSubInDbf(TreeView1.Selected) then