DELPHI 表格控件 DBGridEh
使用详解

----白波九道-----

(博客 http://scmyluo.blog.163.com/)

一、 DBGridEh（增强型表格组件）功能详解

DBGRIDEH 是 Enlib 3.0 组件包中的组件之一。Enlib 3.0 组件包是一位俄国人为增强 Borland 系列开发工具功能而开发的第三方组件,它具有界面友好、功能强大、开发效率高、、快速制作预览/打印简单中国式报表等特点。因此，一推出即受到广大 Borland 程序员的青睐。目前这个版本支持 Borland Delphi versions 4，5，6&7 和 Borland C++ Builder versions 4 & 5 ，可极大地提高数据库应用系统客户端的性能。许多商品软件如《速达 2000》等都使用了该组件。下面本人将使用该组件在实际系统开发过程中的经验总结如下。

DBGridEh 组件无论在外观上还是功能上都非常类似 Borland 开发工具中现有的 dbgrid 组件，它除了提供 dbgrid 组件的全部功能外，还增加了下列新功能:

- 任意选择多行、列或矩形区域的数据.
- 为多列标题设定共同的父标题行.
- 表格底部（Footer） 区显示求和、计数和其它统计信息.
- 自动调整组件宽度与客户区域等宽.
- 设置标题行、数据行的高度.
- 超长的标题行、数据行文本自动折行处理.
- 标题行可作为按钮使用，并可选择是否显示排序标志符（▽降序△升序）.
- 点击列标题可对当前列自动排序而无需编写代码.
- 能够自动设置删除超长文本显示不下的多余部分，并以省略号（…）代替.
- 自动搜索字段（Lookup）数据单元格以单、多列字段下拉列表形式显示.
- 自动搜索字段（Lookup）数据单元格可进行增量搜索.
- 可锁定任意列数在屏幕水平方向不滚动.
- 日期时间控件 DateTime picker 可支持 TDateField and TDateTimeField 两种日期格式.
- 根据字段不同值显示关联的 ImageList 对象图片组中的图片.
- 隐藏任意列.
- 显示 3D 风格的数据区、表尾区和锁定滚动列，制作 3D 外观表格.
- 显示 Memo 类型字段值.
- 除 BOOLEAN 型数据外，其它数据类型也可以检查框(checkbox)形式显示数据.
- 使用专门的函数和过程来存取以 reg 或 ini 文件格式保存的表格布局(包含各数据列表、数据列访问顺序、列宽、索引标识、行高等信息）文件。
- 通过设置数据单元格的 hint 和 ToolTips 属性，当移动鼠标到该单元格时，可以显示单元格容纳不下的文本内容.

● 将组件中数据导入/导出到 Text，Csv，HTML，RTF，XLS 和内部数据等多种格式的文件中.

二、 应用实例

Enlib3.0 组件包安装成功后，在系统的组件面板中会显示“enlib”组件包标签，添加 DBGridEh 到窗体的方法与其它组件一样。在窗体中添加该组件后，请跟我一起来实现图 2 的一些特殊效果，具体属性设置请参考属性表的说明。

1. 定制标题行

1) 制作复杂标题行

标题行可设为 2 行以上高度，并可以为多列创建一个共同的父标题行。为实现这个效果，需在各个列标题属性中以“|”分隔父标题和子标题，如办公用品包括代码和名称两部分，具体属性设置如下：

```
usemultititile=true;
titlelines=2
DBGridEh.Columns[0].Title.Caption := '办公用品|代码';
DBGridEh.Columns[1].Title.Caption := '办公用品|名称';
```

2) 按钮式标题

设置 Column.Title.TitleButton 为 True 可以强制标题单元为按钮式。写 OnTITleBtnClick 事件来控制用户单击标题单元时的操作。

3) 标题行显示图片

实现图 2 中的购买人标题行显示效果。首先添加一个 imagelist 组件 img1 并在其中添加一组 bmp,ico 格式的图片。然后将 DBGridEh 的 TitleImages 设置为 img1.最后在需要显示图片的列标题的 imageindex 中设置需要显示的 img1 中图片的序号。按 F9 执行一下程序，是不是很酷！

4) 如根据不同状态在数据单元格中显示相应图片

如根据库存材料的不同状态在数据单元格中显示相应图片，具体设置如下：

添加一个 imagelist 组件 img1 并在其中添加一组 bmp,ico 格式的图片。然后将需要显示图片的列的 imagelist 属性设置为 img1；在 keylist 属性中添加实际数据存储值，一行为一个值，

切记一定要与 imagelist 中图片顺序一一对应，否则会张冠李戴，面目全非。还可在 picklist 中添加提示信息，也要求是一行为一个值，并设 tooltip 为 true，那么，运行时当鼠标移动到该数据单元格时在显示图片的同时还显示提示信息，怎么样，功能够强大吧！可使用空格键或鼠标切换下一张图片，图片切换的同时也改变了实际存储数据值。也可通过 shift+空格或鼠标切换为上一张图片。这样就实现了上下两个方向图片切换。

5) 自动显示标题行的升降排序标志符（▽降序△升序）并做相应排序

DBGridEh 组件可以在标题行单元格中显示小三角形升、降排序标志符图片，在运行时可点击标题行，图片自动切换并做相应排序。具体属性设置如下：

OptionsEh＝dghAutoSortMarking

Column.Title.TitleButton＝true

SortMarkedColumns 为当前排序列可在运行时使用. 然后在该列的 OnTitleClick 事件中添加代码：

```
procedure TForm_Query.DBGridEh1TitleBtnClick(Sender: TObject; ACol: Integer;
Column: TColumnEh);
var
sortstring:string; //排序列
begin
//进行排序
with Column do
begin
if FieldName = '' then
Exit;
case Title.SortMarker of
smNoneEh:
begin
Title.SortMarker := smDownEh;
sortstring := Column.FieldName + ' ASC';
end;
smDownEh: sortstring := Column.FieldName + ' ASC';
smUpEh: sortstring := Column.FieldName + ' DESC';
end;
//进行排序
try
dataset.Sort := sortstring //dataset 为实际数据集变量名
except
end;
```

```
end;
end;
```

切记 lookup 型字段不可做上述设置，否则系统会提示错误。

另外，组件说明书中提到不需要编写代码即可自动排序，但是不编写代码自动排序方法我还没找到，有知道的朋友烦请告诉我一声啊！让我也对程序代码进行“减肥”。

3.2 定制表格底部（footer）区域的汇总统计行

DBGridEh 组件可以在表格底部显示汇总行，如记录数合计、列字段累加和等信息。在 FooterRowCount 中设置底部显示的行数;然后在Footers 编辑器中添加一个或多个显示列，显示列可以是字段值累加和、记录数合计、字段值或静态文件等集合类型，可以在设计时在 ValueType 属性中设置，也可在运行时通过设置 Footers[i].ValueType 指定其类型。其含义见下表：

切记设置 DBGridEh.SumList.Active 为 True，才会进行汇总统计运算。需注意的是，如显示类型为不是当前列的累加和，则需在 fieldname 属性中指定汇总列，其它类型则无此要求。

6) 点 dbgrideh 标题排序

在网上看了许多这方面的技术资料，啰索而麻烦，其实就这么几行搞定：

```
procedure Sort(grid: TDBGridEh);
begin
Grid.OptionsEh := Grid.OptionsEh + [dghAutoSortMarking];
Grid.ColumnDefValues.Title.TitleButton := True;
Grid.OptionsEh := Grid.OptionsEh + [dghMultiSortMarking];
Grid.SortLocal:=True;
end;
//-------------------------------------------------------------------------
----
// 功能：设定 DbGridEh 合计行信息
// 参数： pDbGrid:TDBGridEh;
// pcFields : string ; 字段列表，字段用逗号分隔
// pvtType : TFooterValueType ; 统计类型 TFooterValueType = (fvtNon, fvtSum,
fvtAvg, fvtCount, fvtFieldValue, fvtStaticText);
// 引用：StrToStringList
// 例如：DbGridEhFoot( DbGridEh1, 'Number,Sum', fvtSum ); 设定数量和金额字段为
合计统计
//-------------------------------------------------------------------------
----
Procedure DbGridEhFoot( pDbGrid:TDBGridEh; pcFields: string; pvtType :
TFooterValueType );
```

```
var nFldLoop : integer ;
cFieldName : string ;
tmpFldList : TStrings ;
begin
pDbGrid.FooterRowCount := 1; // 指定网格尾部统计行行数
pDbGrid.SumList.Active := true; // 激活统计
pDbGrid.FooterColor := clBtnFace ; // 指定统计行颜色
tmpFldList := TStringList.Create ;
StrToStringList( Uppercase(pcFields),',',tmpFldList ); // 将字符串转换为串列表
For nFldLoop := 0 to pDbGrid.Columns.Count -1 do
begin
cFieldName := pDbGrid.Columns[nFldLoop].FieldName ; // 网格列字段名
if tmpFldList.IndexOf( uppercase( cFieldName ) ) >= 0 then
begin
pDbGrid.Columns[nFldLoop].Footer.ValueType := pvtType ; // 统计类型
end;
end ;
tmpFldList.Free ;
end;
//---------------------------------------------------------------------------
----
// 功能：将指定分隔符分隔的字符串转换为字符串列表。
// 此函数在需要将
// 参数：
// pcString : string; 字符串
// pcChar : string; 分隔符
// pDesList : TStringList 字符串列表
// 例如：
// var tmpFldList : TStrings ;
// begin
// tmpFldList := TStringList.Create ;
// StrToStringList( Uppercase(pcFields), ',' , tmpFldList );
// ......
// tmpFldList.Free ;
// end;
//---------------------------------------------------------------------------
----
```

```
Procedure StrToStringList( pcString,pcChar:string; pDesList : TStringList ) ;
overload ;
var cAddStr,cSrcStr : string ;
nPos : integer ;
begin
pDesList.Clear ;
cSrcStr := pcString ;
while True do
begin
nPos := pos( pcChar, cSrcStr );
if nPos = 0 then begin
pDesList.Add( cSrcStr ) ;
Exit ;
end
else begin
cAddStr := copy( cSrcStr,1, nPos - 1 );
pDesList.Add( cAddStr ) ;
Delete( cSrcStr,1, nPos + length( pcChar )-1 );
end;
end;
end;
//Splits a delimited text line into TStrings (does not account for stuff in quotes
but it should)
procedure Split(aValue: string; aDelimiter: Char; var Result: TStrings);
var
X: Integer;
S: string;
begin
if Result = nil then
Result := TStringList.Create;
//Result.Clear;
S := '';
for X := 1 to Length(aValue) do
begin
if aValue[X] <> aDelimiter then
S := S + aValue[X]
else
begin
```

```
Result.Add(S);
S := '';
end;
end;
if S <> '' then
Result.Add(S);
end;
procedure TfrmBdDetail.DBGridEh1DrawColumnCell(Sender: TObject;
const Rect: TRect; DataCol: Integer; Column: TColumnEh;
State: TGridDrawState);
begin
if adoquery1.fieldbyname('合计数量').AsInteger>
adoquery1.fieldbyname('KC').AsInteger then
begin
(Sender as TDBGridEH).Canvas.Brush.Color := $008080FF;
end;
DBGridEh1.DefaultDrawColumnCell(Rect,DataCol,Column,State);
end;
2007-11-16 17:18:59 {******************************************************}
{ }
{ 格式显示 }
{ }
{ 版权所有 (C) 2007 咏南工作室（陈新光） }
{ }
{******************************************************}
unit uDataProcess;
interface
uses
DB;
procedure SetDisplayFormat(ADataset:TDataSet);
implementation
procedure SetDisplayFormat(ADataset:TDataSet);
var
i:Integer;
begin
for i:=0 to ADataset.FieldCount-1 do begin
with ADataset do begin
```

```
if (Fields[i] is TNumericField) and (not (Fields[i] is TIntegerField)) then
(Fields[i] as TNumericField).DisplayFormat := '###,##0.00';
if (Fields[i] is TFloatField) then
(Fields[i] as TFloatField).DisplayFormat := '###,##0.00';
if (Fields[i] is TCurrencyField) then
(Fields[i] as TCurrencyField).DisplayFormat := '￥###,##0.00';
end;
end;
end;
end.
```

7) 在 DBGridEH 中怎样实现多重排序(标题出现 0123 等排列序号)?

Ctrl + 用鼠标点要排序的标头!

8) 让 dbgrid 显示序号

修改 GRID.pas

在 TCustomGrid.SetColWidths 事件改为:

```
procedure TCustomGrid.SetColWidths(Index: Longint; Value: Integer);
begin
if FColWidths = nil then
UpdateExtents(FColWidths, ColCount, DefaultColWidth);
if Index >= ColCount then InvalidOp(SIndexOutOfRange);
if Value <> PIntArray(FColWidths)^[Index + 1] then
begin
if Value < 12 then Value := 30; //新增
ResizeCol(Index, PIntArray(FColWidths)^[Index + 1], Value);
PIntArray(FColWidths)^[Index + 1] := Value;
ColWidthsChanged;
end;
end;
```

修改 DBGRID.pas

在 procedure TCustomDBGrid.DrawCell(ACol, ARow: Longint; ARect: TRect; AState: TGridDrawState);事件加

找到下面这一行

```
FIndicators.Draw(Canvas, ALeft,
```

```
(ARect.Top + ARect.Bottom - FIndicators.Height) shr 1, Indicator, True);
if ACol <0 then
begin
Canvas.TextRect(ARect,0,(ARect.Top + ARect.Bottom - FIndicators.Height) shr 1
,inttostr(self.DataSource.DataSet.RecNo));
end; //新增
```

2. 外观布局

1) 根据不同字段值显示相应的小图片

如根据库存材料的不同状态在数据单元格中显示相应图片，具体设置如下：

添加一个 imagelist 组件 img1 并在其中添加一组 bmp,ico 格式的图片。然后将需要显示图片的列的 imagelist 属性设置为 img1；在 keylist 属性中添加实际数据存储值，一行为一个值，切记一定要与 imagelist 中图片顺序一一对应，否则会张冠李戴，面目全非。还可在 picklist 中添加提示信息，也要求是一行为一个值，并设 tooltip 为 true，那么，运行时当鼠标移动到该数据单元格时在显示图片的同时还显示提示信息，怎么样，功能够强大吧！可使用空格键或鼠标切换下一张图片，图片切换的同时也改变了实际存储数据值。也可通过 shift+空格或鼠标切换为上一张图片。这样就实现了上下两个方向图片切换。

2) 显示检查框（checkbox）外观

对于 Boolean 型字段值在 dbgrideh 组件中自动显示为检查框。通常情况下我们需将非 Boolean 型字段值也此外观显示，如性别字段为字符型，字段值为“男性”时为选中，“女性”时为未选中。需要在 keylist 编辑器中设置实际存储数据值，第一行为选中时的值“1”，第二行为未选中的值“0”，第三行为其它值“2”，支持三态显示。

3) 显示单、多列下拉列表

根据单元格字段值显示与其相关的其它表字段内容，如部门代码字段显示为部门名称。首先需在当前表中新建立一个 lookup 型字段，设置好关联表的字段和返回字段。多列下拉列表需在单列基础上做进一步设置，在 LookupDisplayFields 中以“；”号将关联表中多个字段分隔开，而且返回字段必须作为其中的第一项。具体设置如下：

dropdownshowtitles＝true

dropdownsizing＝true

dropdownwidth＝-1

例：当前表中只有部门代码无部门名称列，需与部门表建立关联，当点击单元格时以部门代码、部门名称两列下拉列表形式显示。

4) 显示日历下拉列表

Date 和 DateTime 类型字段值均可以此形式显示。外观与编辑框无异，当点击该单元格时，右侧会出现“▽”符号，点击之即可出现日历下拉列表。有时不希望出现日历下拉列表，只需设置 Column.ButtonStyle 属性为 cbsNone 即可，此方法同样适用于其它组件不以特殊外观显示的情况。

5) 3D 或平面外观效果

设置 OptionsEh 属性 中 fixed, frozen, footer 和 data rows 等属性表格外观为 3D 效果，设置 flat 为 true 则为平面外观效果.

6) 行头和列头的启用关闭

DBGRID 或 DBGRIDEH 中 Options 都有个“dgTitle",将它置为 False 即没有列头，将“dgIndicator”置为 False 即无行头

7) DBGrid 如何实现透明效果?

在 DrawDataCell 事件里写代码实现了，并没有画面呆滞闪烁的情况。

没有实践就不要妄下论断。

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, DBGrids, DB, ADODB;
type
  TNewDBGrid = class(TDBGrid)
  private
    FBackImg:TBitmap;
    function GetBackImg: TBitmap;
    procedure SetBackImg(const Value: TBitmap);
    procedure WMHScroll(var Message: TWMHScroll); message WM_HSCROLL;
  protected
    procedure Scroll(Distance: Integer); override;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
```

```
  published
    Property BackImg:TBitmap read GetBackImg write SetBackImg;
  end;
  TForm1 = class(TForm)
    Button1: TButton;
    DataSource1: TDataSource;
    ADOConnection1: TADOConnection;
    ADOTable1: TADOTable;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
      Field: TField; State: TGridDrawState);
  private
     NewDBGrid:TNewDBGrid;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
implementation
{$R *.dfm}
//------/TNewDBGrid\----------
constructor TNewDBGrid.Create(AOwner: TComponent);
begin
  inherited;
  FBackImg:=TBitmap.Create;
end;
destructor TNewDBGrid.Destroy;
begin
  FBackImg.Free;
  inherited;
end;
function TNewDBGrid.GetBackImg: TBitmap;
begin
  Result:=FBackImg;
```

```
end;
procedure TNewDBGrid.Scroll(Distance: Integer);
begin
  inherited;
  Invalidate;
end;
procedure TNewDBGrid.SetBackImg(const Value: TBitmap);
begin
  FBackImg.Assign(Value);
end;
procedure TNewDBGrid.WMHScroll(var Message: TWMHScroll);
begin
  inherited;
  if not Focused then Exit;
  if DataLink.Active then Invalidate
end;
//----------
procedure TForm1.FormCreate(Sender: TObject);
begin
  NewDBGrid:=TNewDBGrid.Create(Self);
  with NewDBGrid do
  begin
    Parent:=Self;
    SetBounds(10,10,600,400);
    DataSource:=DataSource1;
    DefaultDrawing:=False;
    OnDrawDataCell:=DBGrid1DrawDataCell;
    BackImg.LoadFromFile('d:\无标题.bmp');
  end;
end;
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  NewDBGrid.Free;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  ADOTable1.Open;
```

```
end;
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
begin
  with Sender as TNewDBGrid do
  begin
    Canvas.CopyRect(Rect,BackImg.Canvas,Rect);
    Canvas.Brush.Style:=bsClear;
    DefaultDrawDataCell(Rect,Field,State);
  end;
end;
end.
```

8) 滚动条的各种应用

a) 锁定多列不滚动

当表格水平方向信息在一屏幕显示不下时，此项功能非常有用。例如，工资表格中包含姓名、基本工资、绩效工资等信息一屏幕显示不下，需要通过移动水平滚动条显示下一屏信息。如果不锁定关键字段列如姓名，则移动到下一屏时就不知道此条记录对应的姓名。因此，在实际应用中经常需锁定多列不滚动。

例：姓名字段为表格第二列，则设置 FrozenCols=2.这样当一屏幕显示不下，通过移动水平滚动条显示下一屏信息时，表格前两列不滚动，作为参照列。

b) 去掉滚动条

```
private
{ Private declarations }
FGridWndProc: TWndMethod;
procedure GridWndProc(var Message: TMessage);
Form.OnCreate：
TGridAccess(DBGrid1).ScrollBars := ssNone;
FGridWndProc := DBGrid1.WindowProc;
DBGrid1.WindowProc := GridWndProc;
Form.OnDestroy：
DBGrid1.WindowProc := FGridWndProc;
```

```
procedure TForm1.GridWndProc(var Message: TMessage);
begin
case Message.Msg of
WM_PAINT, WM_NCPAINT:
begin
SetScrollRange(DBGrid1.Handle, SB_HORZ, 0, 0, False);
SetScrollRange(DBGrid1.Handle, SB_VERT, 0, 0, False);
end;
end;
FGridWndProc(Message);
end;
```

c) 冷区

冷区是数据网格列集左边显示的不可滚动的区域。与固定列不同的是，冷区的列可以获得编辑焦点。可以通过设置 FrozenCols 属性来设置右边不可滚动的列集。

d) delphi 的 EhLib DBGridEh 空间中，如何让滚动条动，游标不动。

就是说，拖动滚动条，数据在跟着滚动条动，而游标不会跟着移动，始终在所选择的那条数据上.

继承 DBGridEh 组件, 写一个 DBGridEhEx 组件
然后定义如下函数

```
private
   procedure WMVScroll(var Message: TWMVScroll); message WM_VSCROLL;

procedure DBGridEhEx.WMVScroll(var Message: TWMVScroll);
var
   SI: SCROLLINFO;
begin
   if (csDesigning in ComponentState) or not AcquireFocus then Exit;
   if not DataLink.Active then Exit;
   with Message, DataLink.DataSet do
   case ScrollCode of
   SB_LINEUP: ;//Prior;
   SB_LINEDOWN: ;//Next;
   SB_PAGEUP: ;//MoveBy( -VisibleDataRowCount );
   SB_PAGEDOWN: ;//MoveBy( VisibleDataRowCount );
   SB_THUMBTRACK:
   begin
```

```
  if Pos >0 then DataLink.DataSet.RecNo :=Integer( Message.Pos)
  else begin
  SI.cbSize := sizeof(SI);
  SI.fMask := SIF_TRACKPOS;
  Windows.GetScrollInfo(Handle, SB_VERT, SI);
  //DataLink.DataSet.RecNo := SI.nTrackPos;
  end;
  end;
  SB_THUMBPOSITION {,SB_THUMBTRACK}:begin
  case Pos of
  0: //First;
  1: //MoveBy( -VisibleDataRowCount );
  2: Exit;
  3: //MoveBy( VisibleDataRowCount );
  4: //Last
  end; //end;
  SB_BOTTOM: //Last;
  SB_TOP: //First;
  end;
end;
```

注意: 所有会移支数据集当前游标位置的, 都已经打上注释, 这样就应该可以实现你想要的效果, 但是不知道是否表格内部有没有限制显示范围是跟当前行有关的
离开当前行一定距离就不能移动之类

e) 解决 DBGridEh 滚动条偶尔不准确的 bug

为了解决 DBGridEh 滚动条偶尔不准确的 bug(比如当数据集被过滤后，垂直滚动条就不再准确反映了)，花了好多时间研究 Ehlib 的源代码。原因就是出在 TCustomDBGridEh.UpdateScrollBar 函数里面对数据集 IsSequenced 的判断有问题。研究了两 3.6 和 5.2 两个版本，都有这个问题，由于每个版本的代码还是有点小差别，就不直接给出修改方法，只指出原因。

f) DBGridEh 垂直滚动条的问题

DBGridEh 垂直滚动条经常不工作，各种属性有用没用都试过，还是不行。
上网去查，好象大家都有这个问题，可就是没有结果。
找到了一篇分析的还在理，是说 IsSequenced 一返回 false 就不能用了。而设置了过滤条件 IsSequenced 就会返回 false，那不是不让人用了？
找回原说明文档看一下，发现人家说的很明白，只不过这一段没人翻过来，在中文世界就等于没有，大家都在瞎摸。
简单地说就是：当 RecNo 不可靠时，DBGridEh 可用 SumList 来做指示，所以，只要把 SumList 中的 Active 和 VirtualRecords 设成 True，DBGridEh 的垂直滚动条就正常了。
TDBGridEh and vertical scrollbar.

If you works with different type of dataset you can notice that for some type of dataset DBGrid show vertical scrollbar validly but for over vertical scrollbar have only three position independently of record count in dataset. To set vertical scrollbar accomodation DBGrid use RecordCount and RecNo property of DataSet component. But some dataset and even same dataset under some condition holds -1 in RecordCount and RecNo. DataSet function IsSequenced indicates whether the underlying database table uses record numbers to indicate the order of records. When IsSequenced returns True, applications can safely use the RecNo property to navigate to records in the dataset and DBGrid use RecNo property to show thumb position in vertical scrollbar. But when IsSequenced returns False DBGrid can not define current record position and show vertical scrollbar in three positions. DBGridEh component have possibility to show proportional scrollbar for no sequenced dataset. To do it need to activate SumList and create list of record bookmars. Set SumList.Active to True and SumList.VirtualRecords to True. SumList will run through dataset and create list of record bookmarks, if you use client/sever technology to access database SumList will force dataset to fetch all records, so it operation can take much time. Keep in mind that VirtualRecords will work only for full relationship bookmarks dataset, it means that DataSet.ComapreBookmark function has to return > 0 if bookmark1 > bookmark1 (i.e. record to which indicates bookmark1 have to be after record to which indicates bookmark1), = 0 if bookmark1 = bookmark1, < 0 if bookmark1 = bookmark1. TBDEDataSet in most cases support full relationship bookmarks.

9) 数据行高

RowHeight 和 RowLines 属性来指定数据行高。完整的数据行高 = 行线高度＋行高。RowSizingAllowed 为 True 以允许可以在运行是使用鼠标来改变行高。

10) DBGrid 设置 Rowheight 后如何将单元格内容纵向和垂直都居中?

```
//自动折行，自动调整行高，对齐方式为垂直居中
 procedure  TMainForm.DBGridDrawColumnCell(Sender:  TObject;
    const  Rect:  TRect;  DataCol:  Integer;  Column:  TColumn;
    State:  TGridDrawState);
 var
    ARect:  TRect;
    HCell:  Integer;
    RecNo:  Integer;
    SCell:  string;
 begin
    if  DBGrid.DataSource.DataSet.RecordCount  >  0  then
    begin
```

```
RecNo := DBGrid.DataSource.DataSet.RecNo;
//注意，我这里只是示范，实际应用要加上转换异常处理
SCell := Column.Field.AsString;
Arect := Rect;
Column.Title.Alignment := taCenter;
with DBGrid, DBGrid.Canvas do
begin
HCell := DrawText(Handle, PChar(SCell), Length(SCell), ARect, DT_CENTER or DT_VCENTER or DT_WORDBREAK);
FillRect(ARect);
if HCell >= RowHeights[RecNo] then
begin
RowHeights[RecNo] := HCell;
DrawText(Handle, PChar(SCell), Length(SCell), ARect, DT_CENTER or DT_VCENTER or DT_WORDBREAK)
end else
DrawText(Handle, PChar(SCell), Length(SCell), ARect, DT_CENTER or DT_VCENTER or DT_SINGLELINE);
end;
end;
end;

//你可以在此基础上，再变化出你自己想要的单元格效果

//可以根据不同的数据类型设置不同的对齐方式，如数字右对齐
```

11) 设置 DBGridEH 自适应列宽的最好方法

一直在找最好的根据 **DBGridEH(**或者 **DBGrid)**的内容和标题栏设置自适应列宽的方法，一直没有太好的。今天从园地上发现了源码：地址如下，非常好用。与大家分享：

http://www.delphifans.com/SoftView/SoftView_2019.html

代码哪下：

```
//需要定义这个类，才能使用 OptimizeSelectedColsWidth 方法调整列宽
type
```

```
  TZYDBGridEH = class(TCustomDBGridEh)
  public
  end;
```

我在原来的基础上进行修改的实现代码：

```
//表格内容和字段标题宽度设置列宽
procedure SetDBGridColumnsWidth(Grid: TCustomDBGridEh);
var
  i: integer;
begin
  if not Assigned(Grid.DataSource) then Exit;
  if not Grid.DataSource.Dataset.Active then Exit;
  try
    Grid.DataSource.Dataset.DisableControls;
    for i := 0 to TDBGridEh(Grid).Columns.Count - 1 do
    begin
      TZYDBGridEh(Grid).OptimizeSelectedColsWidth(TDBGridEh(Grid).Columns[i]);
    end;
    Grid.DataSource.Dataset.EnableControls;
  except on E: Exception do
    begin
      Error('设置表格' + Grid.Name + '出错！' + E.Message);
    end;
  end;
end;
```

12) Ehlib 的 DBGridEh 首列加序号

必须有首列（建一个）

dbgrideh-options-dgzndicator 设置为 **false**

dbgrideh.columns[0]-visual-color 可以区分其他列的颜色

```
procedure TDBViewFrm.DBGridEh1DrawColumnCell(Sender: TObject; const Rect:
TRect;
  DataCol: Integer; Column: TColumnEh; State: TGridDrawState);
begin
  if Column.Index = 0 then
    if DBGridEh1.SumList.RecNo <> -1 then
      DBGridEh1.Canvas.TextRect(Rect, Rect.Left + 3, Rect.Top + 2,
        IntToStr(DBGridEh1.SumList.RecNo));
end;
```

```
procedure TDBViewFrm.DBGridEh1TitleClick(Column: TColumnEh);
  //对 dbgrid 点击标题栏排序
var
  i: Integer;
begin
  if ADOQuery1.Active = True then
    begin
      for i := 1 to DBGridEh1.Columns.Count do
      begin
        //恢复所有标题字体为默认
        DBGridEh1.Columns[i - 1].Title.Font.Color := clWindowText;
        DBGridEh1.Columns[i - 1].Title.Font.Style := [];
      end;
      if (Column.Index <> 0) and (Column.Index <> 9) then //去掉不需要排序或不能排序的列。
      begin
        if ADOQuery1.Sort <> (Column.FieldName + ' ASC') then //判断原排序方式
        begin
          ADOQuery1.Sort:= Column.FieldName + ' ASC';
          Column.Title.Font.Color := clRed;
          //改变标题行字体为红色，表示当前的排序方式为升序
          Column.Title.Font.Style := [fsBold];
        end
        else
        begin
          ADOQuery1.Sort:= Column.FieldName + ' DESC';
          Column.Title.Font.Color := clBlue;
          //改变标题行字体为红色，表示当前的排序方式为降序
          Column.Title.Font.Style := [fsBold];
        end;
      end;
    end;
end;
```

13) 分行分列、单元格的颜色设置

a) 事件定制单元格字体及颜色

有几个事件可以让你能够在绘制单元格前定制单元格字体和颜色。你可以写 TDBGridEh 的 OnDrawColumnCellEvent 事件句柄来在控制在网格单元中绘制数据。你可以使用 Canvas 属性的方法来绘制单元格。但是如果你只想改变字体或颜色的属性，我建议你使用下面的事件。你可以写 TDBGridEh 的 OnGetCellParams 事件来控制在绘制数据单元以前所指定的操作。你可以改变绘制字体及背景色。这个事件适合你在想改变整行的字体或颜色时使用。如果你想改变指定列中单元格的属性，你可以使用 TColumnEh.OnGetCellParams。写这个事件用来控制在一列数据单元被重绘或编辑时的操作。在一列数据单元被重绘以前，你可以改变绘制字体，背景色，对齐方式，图像索引，文本或检查框。在编辑一列数据单元以前，你可以改变编辑字体，背景色，文本或只读状态。

b) 五星推荐 DEMO 示例控制选中行颜色

```
if (Rect.Top = DBGridEh2.CellRect(DBGridEh2.Col,DBGridEh2.Row).Top) and
(not (gdFocused in State) or not DBGridEh2.Focused) then
  DBGridEh2.Canvas.Brush.Color :=$F0CAA6;
DBGridEh2.DefaultDrawColumnCell(Rect,DataCol,Column,State);
```

c) 分行不同颜色设置；

在 **DBGridEh1DrawColumnCell** 中写；

```
if ADOQuery1.RecNo mod 2=0 then
begin
    DBGridEh1.Canvas.Font.Color := clRed;
    DBGridEh1.DefaultDrawColumnCell(Rect, DataCol, Column, State);
end
else begin
    DBGridEh1.Canvas.Font.Color := clGreen;
    DBGridEh1.DefaultDrawColumnCell(Rect, DataCol, Column, State);
end;
```

d) 分行不同背景颜色设置；

在 **DBGridEh1DrawColumnCell** 中写；

```
if ADOQuery1.RecNo mod 2=0 then
begin
```

```
        DBGridEh1.Canvas.Brush.Color := clRed;
        DBGridEh1.DefaultDrawColumnCell(Rect, DataCol, Column, State);
    end
    else begin
        DBGridEh1.Canvas.Brush.Color := clGreen;
        DBGridEh1.DefaultDrawColumnCell(Rect, DataCol, Column, State);
    end;
```

e) **符合条件的单元格颜色或者背景颜色设置；**

在 **DBGridEh1DrawColumnCell** 中写；

```
    if Column.FieldName='价格' then
    begin
        if ADOQuery1.FieldByName('价格').AsFloat<0 then
        begin
            DBGridEh1.Canvas.Font.Color := clRed;
            DBGridEh1.DefaultDrawColumnCell(Rect, DataCol, Column, State);
        end;
    end;
```

f) **交叉设置列颜色**

1、为 **DBGridEh** 的每列的 **Color** 属性设置值。

2、将 **DBGridEh** 的 **RowColorFlag** 设为 **false;**

注：**DBGridEh1.DefaultDrawColumnCell(Rect, DataCol, Column, State);**

Rect:代表在画布中 **cell** 的位置位置所在，也就是你要对哪个区域进行重画；

DataCol：代表 **columns** 数组中 **column** 的标号

Column：描述 **cell** 的显示属性和对应的字段属性的 **tcolumn** 对象

State：描述 **cell** 是否有输入焦点、是否被选中、是否处于锁定模式（如同 **column header**）

g) **隔行不同颜色显示**

```
with TDBGrid(Sender) do
begin
if (gdSelected in State) or (gdFocused in State) then
Canvas.Brush.Color := clAqua
else if DataSource.DataSet.RecNo mod 2 = 0 then
Canvas.Brush.Color := $00F0F0F5
else
Canvas.Brush.Color := clWindow;
DefaultDrawColumnCell(Rect, DataCol, Column, State);
```

```
end;
```

h) 纵向斑马线效果

实现网格的奇数列和偶数列分别以不同的颜色显示,以区别相邻的数据列。

```
file://在 DbGrid 的 DrawColumnCell 事件中编写如下代码：
Case DataCol Mod 2 = 0 of
True: DbGrid1.Canvas.Brush.Color:= clBlue;
file://偶数列用蓝色
False: DbGrid1.Canvas.Brush.Color:= clAqua;
file://奇数列用浅绿色
End; DbGrid1.Canvas.Pen.Mode:=pmMask;
DbGrid1.DefaultDrawColumnCell (Rect, DataCol, Column, State);
```

i) 纵向斑马线,同时突出显示当前单元格

同时以红色突出显示当前单元格效果：以突出显示当前选中的字段。

```
file://将上述代码修改为：
Case DataCol Mod 2 = 0 of
True: DbGrid1.Canvas.Brush.Color:= clBlue;
file://偶数列用蓝色
False: DbGrid1.Canvas.Brush.Color:= clAqua;
file://奇数列用浅绿色 End;
If ((State = [gdSelected]) or (State=[gdSelected,gdFocused])) then
If Not DbGrid1.SelectedRows.CurrentRowSelected then
DbGrid1.Canvas.Brush.Color:=clRed;
file://当前选中单元格显示红色
DbGrid1.Canvas.Pen.Mode:=pmMask;
DbGrid1.DefaultDrawColumnCell (Rect, DataCol, Column, State);
```

j) 在数据网格中以红色突出显示当前选中的行：

设置 DbGrid 控件的 Options 属性中的 dgRowSelect 属性为真，Color 属性为 clAqua(背景色)， 在 DbGrid 的 DrawColumnCell 事件中编写如下代码：

```
if ((State = [gdSelected]) or (State=[gdSelected,gdFocused])) then
DbGrid1.Canvas.Brush.color:=clRed;
file://当前行以红色显示，其它行使用背景的浅绿色
DbGrid1.Canvas.pen.mode:=pmmask;
```

```
DbGrid1.DefaultDrawColumnCell (Rect, DataCol, Column, State);
```

k) 行突显的斑马线效果：既突出当前行，又区分不同的列（字段）。

```
file://其它属性设置同 3，将上述代码修改为：
if ((State = [gdSelected]) or (State=[gdSelected,gdFocused])) then
begin
Case DataCol Mod 2 = 0 of
True : DbGrid1.Canvas.Brush.color:=clRed;
file://当前选中行的偶数列显示红色
False: DbGrid1.Canvas.Brush.color:=clblue;
file://当前选中行的奇数列显示蓝色
end; DbGrid1.Canvas.pen.mode:=pmmask;
DbGrid1.DefaultDrawColumnCell (Rect, DataCol, Column, State);
end;
```

l) 横向斑马线，同时以红色突显当前行效果：

```
  file://其它属性设置同 3，将上述代码修改为：
Case Table1.RecNo mod 2 = 0 of
file://根据数据集的记录号进行判断
True : DbGrid1.Canvas.Brush.color:=clAqua;
file://偶数行用浅绿色显示
False: DbGrid1.Canvas.Brush.color:=clblue;
file://奇数行用蓝色表示
end;
if ((State = [gdSelected]) or (State=[gdSelected,gdFocused])) then
file://选中行用红色显示
DbGrid1.Canvas.Brush.color:=clRed;
DbGrid1.Canvas.pen.mode:=pmMask;
DbGrid1.DefaultDrawColumnCell (Rect, DataCol, Column, State);
```

m) 双向斑马线效果

即行间用不同色区分，同时，选中行以纵向斑马线效果区分不同的列。

```
file://其它属性设置同
3，将上述代码修改为：
Case Table1.RecNo mod 2 = 0 of
```

```
file://根据数据集的记录号进行判断
True  :  DbGrid1.Canvas.Brush.color:=clAqua;
file://偶数行用浅绿色显示
False:  DbGrid1.Canvas.Brush.color:=  clblue;
file://奇数行用蓝色表示
end;
If  ((State  =  [gdSelected])  or  (State=[gdSelected,gdFocused]))  then
Case  DataCol  mod  2  =  0  of
True  :  DbGrid1.Canvas.Brush.color:=clRed;
file://当前选中行的偶数列用红色
False:  DbGrid1.Canvas.Brush.color:=  clGreen;
file://当前选中行的奇数列用绿色表示
end;
DbGrid1.Canvas.pen.mode:=pmMask;
DbGrid1.DefaultDrawColumnCell  (Rect,  DataCol,  Column,  State);
```

n) 随意控制 DBGrid 每一行的颜色：

```
Varp  :  Integer;beginp  :=  Table1.FindField('wage').AsInteger;
//取得当前记录的 Wage 字段的值。
if(p  <  500)  then  begin  //程序将根据 wage 值设置各行的颜色。
Color  :=  clGreen;Font.Style  :=  [fsItalic];  //不仅可以改变颜色,还可以
改变字体
end;
if  (p  >=  500)  And  (p  <  800)  thenColor  :=  clRed;if(p
>=800)  then
beginColor  :=  clMaroon;Font.Style  :=  [fsBold];
end;
    end
```

14) 点击不同单元格列，执行不同的动作

DBGrid 的 OnMouseDown/OnMouseUp 事件在点击记录单元格时不会触发(点击固定行列区会触发)，而 Options.dgRowSelect=True 时，OnCellClick 事件的 Column 总是传递第一个列对象，即 Column.Index=0，即使你点击的是其他列，因此需要在 OnCellClick 中再判断点击的是哪个列，再根据不同列执行不同的动作。

```
    procedure TForm1.DBGrid1CellClick(Column: TColumn);
    var
```

```
    Coord: TGridCoord;
    P: TPoint;
    begin
    GetCursorPos(P);
    Windows.ScreenToClient(TDBGrid(Sender).Handle, P);
    Coord := TDBGrid(Sender).MouseCoord(P.X, P.Y);
    if (Coord.Y > 0) and (Coord.X = 1) and not
TDBGrid(Sender).DataSource.DataSet.IsEmpty then
    // Coord.X=1，dgIndicator=True 时说明在第一列，False 时说明在第二列
    ShowMessage(GridRate.Columns[0].Field.AsString);
    end;
```

15) 下拉式计算器

对于 TDateField 和 TDateTimeField 字段，inplace 编辑器将显示下拉按钮以显示显示下拉计算器。设置 Column.ButtonStyle 为 cbsNone 以禁止显示下拉按钮

16) 鼠标移到某个单元格，指针形状改变

```
    procedure TForm1.DBGrid1MouseMove(Sender: TObject; Shift: TShiftState;
    X, Y: Integer);
    var
    Coord: TGridCoord;
    begin
    Coord := TDBGrid(Sender).MouseCoord(X, Y);
    if (Coord.Y > 0) and (Coord.X = 1) and not
TDBGrid(Sender).DataSource.DataSet.IsEmpty then
    begin // Coord.X=1，dgIndicator=True 时说明在第一列，False 时说明在第二列
    TDBGrid(Sender).Cursor := crHandPoint;
    StatusBar1.SimpleText := 'Click to open curve form';
    end
    else
    begin
    TDBGrid(Sender).Cursor := crDefault;
    StatusBar1.SimpleText := '';
    end;
    end;
```

17) 自动填充网格列宽到网格客户区

设置 AutoFitColWidths 为 True 以自动重置列宽来设置网格的宽度等于客户区宽度。MinAutoFitWidth 属性决定网格的最小宽度，列宽将会被重新计算

18) 从注册表或 ini 文件中保存或恢复网格和列的层次。

TDBGridEh 有一个常规设置来从注册表或 ini 文件中保存和恢复网络以及列的层次：

RestoreColumnsLayout - 从注册表中恢复列的次序，宽度，排序标志。

RestoreColumnsLayoutIni - 从 ini 文件中恢复列的次序，宽度，排序标志。

RestoreGridLayout - 从注册表中恢复列的次序，宽度，可视，排序标志，排序索引或行高。

RestoreGridLayoutIni - 从 ini 文件中恢复列的次序，宽度，可视，排序标志，排序索引或行高。

SaveColumnsLayout - 保存列的次序，宽度，排序标志到注册表中。

SaveColumnsLayoutIni - 保存列的次序，宽度，排序标志到 ini 文件中。

SaveGridLayout - 保存列的次序，宽度，可视，排序标志，排序索引或行高到注册表中。

SaveGridLayoutIni - 保存列的次序，宽度，可视，排序标志，排序索引或行高到 ini 文件中。

3. 编辑功能

1) 多选

TDBGridEh 允许在选定的区域上进行选择记录，列以及矩形区域等操作：

允许多选会影响下面这些属性：

Options 选项中的 dgMultiSelect 属性 - 设置是否允许多选。

Options 选项中的 dghClearSelection 属性- 设置在用户移到下一个单元时是否清除已选记录。

Options 选项中的 EdITActions 属性 -设置用户可以在已选记录上执行哪些操作（比如，拷贝，剪切，删除，粘贴，全选等）。

Options 选项中的 AllowedSelections 属性-设置允许选定记录的类型（比如，行，列，矩形区域等）。

Options 选项中的 Selection 属性－设置一个当前的多选状态，已选记录，列或矩形区域以及存取它们的属性和函数。

2) 文本多行显示

Column.WordWrap 为 True 可以使数据行中文本多行显示。如果行高>文本行，它就换行。

3) 显示备注字段

设置 DrawMemoText to True 来显示文本式的备注字段。.

4) 如何让 dbgrideh1 显示数据时只显示两位小数

DBGrideh1.Columns[0].DisplayFormat := '#.#0';

5) 获得当前 DBGridEh 表中单元格的序号

DBGridEh1.SelectedIndex

6) 怎样在 dbgridEh 和 Edit 中显示金额的千分号

1、 dbrigeh.columns[0].DisplayFormat := ,0.00

2、 2、adoquery1.fields[0].DisplayFormat := #,##0.00

3、 都可以

4、 在保存之前，记得把','号全部 replace 掉，

5、 tmpEdit.Text := StringReplace(tmpEdit.Text, ',', '', [rfReplaceAll]);

6、 tmpEdit.Text := StringReplace(tmpEdit.Text, '， ', '', [rfReplaceAll]);

7、 在 out 时调用函数自动加千分号

8、 tmpEdit.Text := FloatFormat(tmpEdit.Text);

9、 输出整型的

10、function TTblRecordEditFrame.IntFormat(intValue: string): string;

11、begin

12、//

13、 if intValue = '' then

14、 Result := ''

15、 else

16、 begin

17、 intValue := StringReplace(intValue, ',', '', [rfReplaceAll]);

```
18、    intValue := StringReplace(intValue, '，', '', [rfReplaceAll]);
19、    Result := formatfloat('#,###,##0', StrToFloat(intValue));
20、  end;
21、end;
22、//格式化浮点数，加千分号，删除后边的 0
23、function TTblRecordEditFrame.FloatFormat(floatValue: string): string;
24、var
25、  decPart, intPart: string;
26、  i, j: Integer;
27、  numFormat: string;
28、begin
29、//
30、  if floatValue = '' then
31、    Result := ''
32、  else
33、  begin
34、    floatValue := StringReplace(floatValue, ',', '', [rfReplaceAll]);
35、    floatValue := StringReplace(floatValue, '，', '', [rfReplaceAll]);
36、    i := pos('.', floatValue);
37、    if i > 0 then //有小数点
38、    begin
39、      intPart := Copy(floatValue, 1, i - 1);
40、      decPart := copy(floatValue, i + 1, Length(floatValue));
41、      //从后边开始清除 0
42、      for j := Length(decPart) downto 1 do
43、      begin
44、        if decPart[j] = '0'
45、          then Delete(decPart, j, 1)
46、        else
47、          Break;
48、      end;
49、    end
50、    else //无小数点
51、    begin
52、      intPart := floatValue;
53、      decPart := '';
54、    end;
```

```
55、    if decPart <> '' then
56、      numformat := '#,###,##0.' + StringOfChar('0', Length(decPart))
57、    else
58、      numFormat := '#,###,##0';
59、    Result := formatfloat(numFormat, StrToFloat(intPart + '.' + decpart));
60、  end;
```

7) end;请问怎么才能使 DBGridEh 不滚动就能提交数据？

可以在关闭窗口时，判断一下数据集状态，然后自动保存。

```
if dataset.state in [dsEdit, dsInsert] then
  dataset.post;
```

我一般在保存前用下面这两句，省了很多麻烦：

```
if not DataSet.Active then Exit ;//(或者 ShowMessage 一下)
if DataSet.State<> dsBrowse then DataSet.post ;
```

8) 我怎么把 dbgrid 里的数据一次插入到数据库呢

```
      procedure TForm1.Button1Click(Sender: TObject);
var
  i:integer;
begin
  adoquery1.First;
  for i:=0 to adoquery1.RecordCount do
  begin
    adoquery2.Append;
    adoquery2.Fields[0].Text:=dbgrid1.Fields[0].Text;
    adoquery2.Fields[1].Text:=dbgrid1.Fields[2].Text;
    adoquery2.Post;
    adoquery1.Next;
  end;
end;
```

9) 在 DBGrid 中可选中行而又可进入编辑状态

第一种 OptionsDB 中的 edgoSyncSelection 设为 False

另一种

如何在 DBGrid 中选中行，而又让它可以进入编辑状态？

也许你会问我这有什么用？呵呵，做数据库应用的兄弟们会深有感触，当用 DBGrid 显示的字段过多时，用户不得不拉动最下面的滚动条，去看最右边的东西，如果没有设置 DBGrid->Options[dgRowSelect],那么，拉到最右边之后，很有可能看串行的；如果设置了 DBGrid->Options[dgRowSelect]，则在拉到最右边之后，不会看串行，但是鼠标点击其它行(不是当前选中行)时，DBGrid 的视图一下子就会回到显示最左边的那一列，确实很麻烦，用户不得不一次又一次的拖运下面的滚动条。

一同事因这个问题而苦恼，而我又在 CSDN 的文档库中看到了这篇文章：

《DBGrid 使用全书(五)》，链接：http://dev.csdn.net/article/51/51845.shtm，是 Delphi 版本的，核心代码如下：

```
type
 TMyDBGrid=class(TDBGrid);
//////////////////////////////////
//DBGrid1.Options->dgEditing=True
//DBGrid1.Options->dgRowSelect=False
procedure TForm1.DBGrid1DrawColumnCell(Sender: TObject; const Rect: TRect;
 DataCol: Integer; Column: TColumn; State: TGridDrawState);
begin
 with TMyDBGrid(Sender) do
 begin
   if DataLink.ActiveRecord=Row-1 then
   begin
     Canvas.Font.Color:=clWhite;
     Canvas.Brush.Color:=$00800040;
   end
   else
   begin
     Canvas.Brush.Color:=Color;
     Canvas.Font.Color:=Font.Color;
   end;
   DefaultDrawColumnCell(Rect,DataCol,Column,State);
 end;
end;
```

他的解决办法是：曲线救国，取消 DBGrid->Options[dgRowSelect]，把当前选中行的背景绘制成蓝色，就象是被选中一样,想法确实很妙。我们公司使用 C++Builder，我只好把这段代码改为 C++Builder 版本的，这时，我才发现这段代码的精妙之处。

我发现 DataLink 属性是 TCustomDBGrid 中声明为 protected 的，而在 DBGrid 中并未声明它的可见性，因此，不能直接使用它；而 Row 属性则是在 TCustomGrid 中声明为

protected的,在TCustomGrid的子类中也未声明它的可见性,那么,这段代码为何在Delphi中运行的很好?

原因就在于：ObjectPascal的单元封装，在同一个单元中定义的类，互相之间是友员的关系，我们再来看这段代码的开头：

```
type
 TMyDBGrid = class(TDBGrid);
```

声明了一个TMyDBGrid类，那么，当前这个窗体类就和TMyDBGird类互为友元了，那么当然当前窗体类可以直接访问TMyDBGrid的私有属性Row和DataLink了,一切都明了了,那么用C++就好实现了，核心代码如下：

```
void __fastcall TMainForm::LineSelEdit(TObject *Sender,const TRect &Rect, int DataCol, TColumn *Column,TGridDrawState State)
// 本文转自 C++Builder 研究 - http://www.ccrun.com/article.asp?i=637&d=2724wn
{
    class TMyGridBase : public TCustomGrid
    {
    public:
        __property Row;
    };
    class TMyGrid : public TCustomDBGrid
    {
    public:
        __property DataLink;
    };
    TMyGrid *MyGrid = (TMyGrid*)Sender;
    TMyGridBase *MyGridBase = (TMyGridBase*)Sender;
    TDBGrid *Grid = (TDBGrid*)Sender;

    if(MyGrid->DataLink->ActiveRecord == MyGridBase->Row-1) {
        Grid->Canvas->Font->Color = clWhite;
        Grid->Canvas->Brush->Color = TColor(0x00800040);
    } else {
        Grid->Canvas->Brush->Color = Grid->Color;
        Grid->Canvas->Font->Color = Grid->Font->Color;
    }

    Grid->DefaultDrawColumnCell(Rect,DataCol,Column,State);
}
```

我把它封装成一个函数，函数的参数与 DBGrid 的 OnDrawDataCell 的参数一样，使用它的方法就是取消设置 DBGrid->Options[dgRowSelect]，然后设置 DBGrid->DefaultDrawing = false，然后在这个 DBGrid 的 OnDrawDataCell 事件中调用这个函数，如下：

```
void __fastcall TMainForm::DBGridDrawColumnCell(TObject *Sender,
      const TRect &Rect, int DataCol, TColumn *Column,
      TGridDrawState State)
{
    this->LineSelEdit(Sender,Rect,DataCol,Column,State);
}
```

示例代码下载：http://www.ccrun.com/_down/src/DBGridSelLine.rar

10) 修正 DBGrideh 丢失焦点时自动关闭输入法的问题

楼主：　　使用 DBGrideh 控件时，每当焦点从此控件转移时，都会自动将输入法关闭，给用户输入数据带来很大的不便，原因就是 DBgrideh 在处理 WM_killFocus 时的代码有些小问题。

把 dbgrideh.pas 打开后找到下面二个过程，将打星号的语句注释掉就可以彻底解决此问题。

```
procedure TDBGridInplaceEdit.WMKillFocus(var Message: TWMKillFocus);
begin
 if not SysLocale.FarEast then inherited
 else
 begin
*   ImeName := Screen.DefaultIme;//将此句注释掉
   ImeMode := imDontCare;
   inherited;
   //将此 IF 语句注释掉
*   if HWND(Message.FocusedWnd) <> Grid.Handle then
*    ActivateKeyboardLayout(Screen.DefaultKbLayout, KLF_ACTIVATE);
 end;
 if FListVisible and not
   ((Message.FocusedWnd = FActiveList.Handle) or
   (GetParent(Message.FocusedWnd) = FActiveList.Handle)
   ) then
   CloseUp(False)
 else if MRUList.DroppedDown and not (Message.FocusedWnd = MRUListControl.Handle) then
   MRUListCloseUp(MRUList, False);
//  if (Message.FocusedWnd <> Grid.Handle) then
//   Grid.ControlLeaveFocus;
 if Grid.FSelectionActive <> Grid.IsSelectionActive then
   Grid.SelectionActiveChanged;
```

```
end;
//==========
procedure TCustomDBGridEh.WMKillFocus(var Message: TWMKillFocus);
var
  InvalidRect: TRect;
begin
  if FSortMarking and
    not ((InplaceEditor <> nil) and (Message.FocusedWnd = InplaceEditor.Handle)) then
  begin
    FSortMarking := False;
    DoSortMarkingChanged;
  end;
  if HandleAllocated and (dgRowSelect in Options) then
  begin
    with inherited Selection do
      InvalidRect := BoxRect(Left - FrozenCols, Top, Right, Bottom);
    WindowsInvalidateRect(Handle, InvalidRect, False);
  end;
  if not SysLocale.FarEast
    then inherited
  else
  begin
*    ImeName := Screen.DefaultIme;//将此句注释掉
    ImeMode := imDontCare;
    inherited;
//将此 If 语句注释掉
*    if not ((InplaceEditor <> nil) and (Message.FocusedWnd = InplaceEditor.Handle))
*      then ActivateKeyboardLayout(Screen.DefaultKbLayout, KLF_ACTIVATE);
  end;
//  if not IsMyInplaceControlHandle(HWND(Message.WParam)) then
//    ControlLeaveFocus;
  if FSelectionActive <> IsSelectionActive then
    SelectionActiveChanged;
end;
```

---

11) DBGRIDEH 选定多行删除怎么实现

```
procedure TfrmItemsClass.DeleteSelectRecord;
var
  FRows, i:integer;
begin
  FRows := DBGridEh1.SelectedRows.Count;
```

```
 if (not EditState) or (FRows = 0) then exit;
 if not ShowQYN('是否删除当前选择记录？') then exit;
 for i:= 0 to FRows -1 do
 begin
   qryItems.GotoBookmark(pointer(DBGrideh1.SelectedRows.Items[i]));
   qryItems.Delete;
 end;
end;
```

或试试 DBGridEh1.SelectedRows.Delete;

12) DBGrid 滚动表格的代码

```
procedure TCustomDBGrid.Scroll(Distance: Integer);
```

从这些代码可以看到

```
NewRect := BoxRect(0, Row, ColCount - 1, Row);
 ValidateRect(Handle, @OldRect);
 InvalidateRect(Handle, @OldRect, False);
 InvalidateRect(Handle, @NewRect, False);
```

以上是滚动之后，刷新新显示的数据范围，使新数据显示出来

```
ScrollWindowEx(Handle, 0, -RowHeight * Distance, @NewRect, @NewRect,
       0, nil, SW_Invalidate);
```

而这句是搬那些旧数据的像素。

所以 DBGrid 的滚动是很有效率，先计算新数据的范围，然后刷新这个范围，这样 CPU 消耗很小的，而多旧数据，只是搬动象素，所以效率很高，不单在画面上有效率，而且不需要访问数据库控件重读旧数据。

问题是搬动像素连背景也会搬，所以不能够使用这个方法。只能整个画面重新刷一次。楼上的代码里面 Invalidate 这一句，就是发消息刷整个表格画面。

4. 统计功能

1) 页脚合计

1、设置 DBGRIDEH 属性的 FooterRowCount 值为 1;　　2、设置 DBGRIDEH 属性的 SumList 的 Active 值为 true;　　3、设置你要求和的该列的 Footer 的 ValueType 类型为 fvtSum;　　4、运行 OK!　　在 DataSet 打开时写:　　DBGridEh1.Columns[0].Footer.Value := IntToStr DBGridEh1.DataSource.DataSet.RecordCount);

使用 TDBSumList 组件

2) 定制表格底部（footer）区域的汇总统计行

DBGridEh 组件可以在表格底部显示汇总行，如记录数合计、列字段累加和等信息。在 FooterRowCount 中设置底部显示的行数;然后在Footers 编辑器中添加一个或多个显示列，显示列可以是字段值累加和、记录数合计、字段值或静态文件等集合类型，可以在设计时在 ValueType 属性中设置，也可在运行时通过设置 Footers[i].ValueType 指定其类型。其含义见下表：

切记设置 DBGridEh.SumList.Active 为 True，才会进行汇总统计运算。需注意的是，如显示类型为不是当前列的累加和，则需在 fieldname 属性中指定汇总列，其它类型则无此要求。

3) TDBSumList 说明

你可以使用 TDBSumList 在可视动态变化数据集中进行记录统计。在你想查看的数据集中设置相关的数据字段，然后写 SumListChanged 事件来指定在 TDBSumList 发生改变后所要做的操作。TDBSumList 的 SumCollection 属性上一个 TDBSum 对象容器。每个 TDBSum 对象是一个可以指定集合值的元件。 FieldName 和 GroupOperation 决定了集合值的类型，SumValue 控制当前的集合值。

TDBSumList 被埋藏于 DBGridEh 组件中,因此所的下面有关 TDBGridEh.SumList 的说明与 TDBGirdEh 的 TDBGridEh.SumList 属性的说明是一样的。

如何工作以及为什么有时 SumList 的集合值计算不正确?

你知道 data-aware 控件与数据集是通过 TDataLink 对象相连接的。 TDataLink 不允许快速重新计算集合值。例如，当从数据集中删除记录时，数据集发送 deDatASEtChange 事件给所有的 TDataLink 对象，在当前局部过滤发生改变时，同样的事件也被发送了。因此当 TDataLink 接收到该事件时，它不得不在所有的数据集中重新计算集合值，甚至于仅从数据集中删除一条记录。在激活后，TDBSumList 重载了数据集的下列事件：OnAfterEdIT, OnAfterInsert, OnAfterOpen, OnAftERPost, OnAfterScroll, OnBeforeDelete, OnAfterClose。这种方法避免了在所有的数据集中它。在不需要它时，又会出现其它的问题，比如： T

运行时指定这些事件。在指定上述事件之一以前，关闭 SumList。

在下面的情形下，SumList 发出存取错误信息。 SumList 试着向数据集返回事件，但数据集数据并未被删除。在 SumList(或网格)以及数据集放置在不同的窗体（数据模块）这种情况下，可以显示数据。在这种情况下，试着在从数据集或数据模块中数据集数据被删除时关闭 S

umList。

如果你使用 SetRange 或 ApplyRange 事件，SumList 将不能跟踪数据集中发生的变化。但可以调用 SumList.RecalAll 方法。

在非 BDE 数据集的主/明细表中，SumList 不能跟踪数据变化。在主数据集激活状态数据集发生变化后，调用 SumList.RecalAll 方法。

在任何其它情况下，如果你发现了其它 SumList 的计算值不正确，你都可以调用 RecalAll 方法。

4) 如何工作以及为什么有时 SumList 的集合值计算不正确？

你知道 data-aware 控件与数据集是通过 TDataLink 对象相连接的。 TDataLink 不允许快速重新计算集合值。例如，当从数据集中删除记录时，数据集发送 deDatASEtChange 事件给所有的 TDataLink 对象，在当前局部过滤发生改变时，同样的事件也被发送了。因此当 TDataLink 接收到该事件时，它不得不在所有的数据集中重新计算集合值，甚至于仅从数据集中删除一条记录。在激活后， TDBSumList 重载了数据集的下列事件：OnAfterEdIT, OnAfterInsert, OnAfterOpen, OnAftERPost, OnAfterScroll, OnBeforeDelete, OnAfterClose。这种方法避免了在所有的数据集中它。在不需要它时，又会出现其它的问题，比如： T

运行时指定这些事件。在指定上述事件之一以前，关闭 SumList。

在下面的情形下，SumList 发出存取错误信息。 SumList 试着向数据集返回事件，但数据集数据并未被删除。在 SumList(或网格)以及数据集放置在不同的窗体（数据模块）这种情况下，可以显示数据。在这种情况下，试着在从数据集或数据模块中数据集数据被删除时关闭 SumList。

如果你使用 SetRange 或 ApplyRange 事件，SumList 将不能跟踪数据集中发生的变化。但可以调用 SumList.RecalAll 方法。

在非 BDE 数据集的主/明细表中，SumList 不能跟踪数据变化。在主数据集激活状态数据集发生变化后，调用 SumList.RecalAll 方法。

在任何其它情况下，如果你发现了其它 SumList 的计算值不正确，你都可以调用 RecalAll 方法。

5) dbgrideh 列求和

```
var
  tmpCol: TColumnEh;
begin
  for I := 0 to DBGridEh2.Columns.Count - 1 do    // Iterate
  begin
    tmpCol := DBGridEh2.Columns[i];
    if tmpCol.Field.DataType in [ftSmallint, ftInteger, ftWord, ftFloat, ftCur
rency] then
      tmpCol.Footer.ValueType := fvtSum;
  end;    // for
end
```

改一下.

```
var
  tmpCol: TColumnEh;
begin
```

```
  for I := 0 to DBGridEh2.Columns.Count - 1 do    // Iterate
  begin
    tmpCol := DBGridEh2.Columns[i];
    if tmpCol.Field is TNumericField then
      tmpCol.Footer.ValueType := fvtSum;
  end;    // for
end
```

5. 数据功能

1) 查找字段 点击某列值的下拉按纽弹出一个从数据库取值下拉列表

查找字段 点击 dbgrid 某列值的下拉按纽（可以设置），弹出一个下拉列表（其内容是另一个 datASEt 的）。

好象是使用了 DBLookupComboboxEh，可我参看 demo1 怎么设置也不行，请指教。谢谢。

呵呵，这个效果没有用 DBLoookupComboBoxEh，它是利用查找字段实现的。

它在 Query1 中，对应 VName 字段，增加了一个叫 VName1 的查找字段，这个

字段从 qrVendors 查找相关的数据，再在 DBGridEh1 显示 VName1,这样显示就是那种效果了。

2) 使用 DBGridEh 自动过滤实现方法

1、过滤实现的前提条件：uses EhlibADO, EhlibBDE **等 …**.

要实现 DBGridEh 的自动过滤，必须先添加 Ehlib 的几个相关 unit 到你的 project 中，如果你使用的是 ADO 连接数据库，请在工程任一 unit 的 uses 列表中添加 EhlibADO，如果使用的是 BDE，请添加 EhlibBDE，其他可能添加的还有：EhlibCDS、EhlibDBX、EhlibIBX、EhlibMTE；

2、启用过滤，显示过滤行：DBGridEh.STFilter.Visible := True

DBGridEh 默认不显示过滤行，设置 STFilter.Visible 为 True 过滤行就显示出来了。

3、选择：客户端还是服务器端过滤？DBGridEh.STFilter.Local = True|False

这一步很主要。客户端过滤也就是 Query 自身的过滤功能，在内存中过滤，不会到服务器端去查询，优点是实现简单，过滤速度快，缺点就是只能获得本地更新的数据，其他用户新增和修改的数据无法实时反映出来。

DBGridEh 服务器端过滤的实现方式是“拼凑”SQL 语句。在过滤之前，DBGridEh 绑定的 Query 的 SQL 语句必须含有 DBGridEh 指定的一个过滤标记，该过滤标记默认为 /*Filter*/，其必须在 SQL 任一行的行首，当用户输入过滤值时，EhLib 生成过滤条件，并且用此条件替换过滤标记之后的内容，然后将新的 SQL 语句发送到数据库端去执行，获得的即为过滤后的数据。

具体应该使用哪种过滤方式，需根据实际要求选用。

4、客户端过滤（本地过滤）的实现：DBGridEh.STFilter.Local := True

客户端过滤实现比较简单，而且容易理解，设置 Local 为 True，过滤功能就 OK 了。

5、服务器端过滤的实现：DBGridEh.STFilter.Local := False

服务器端过滤需设置 Local 为 False，然后剩下的就是 SQL 语句，SQL 一般为以下样式：

```
AdoQuery.SQL.Text := 'SELECT * FROM Students WHERE ' + #13 +'/*Filter*/1=1';
```

以上代码生成的 SQL 在 SQL Server 等支持 /* 注释的数据库中会非常正确的执行，但在 Access 中就会出错，解决方案也很简单，更改 DBGridEh 的过滤标记即可：

```
DBUtilsEh.SQLFilterMarker := '(1>0)AND';
```

那么 Query 的 SQL 应该为以下：

```
AdoQuery.SQL.Text := 'SELECT * FROM Students WHERE ' + #13 + '(1>0)AND 1=1';
```

服务器端过滤也 OK 了！

注：如果你使用的是 Delphi 2007，你可能会发现当你完全无误按以上设置后，服务器端过滤功能还是无效，这是由于 Ehlib DBUtilsEh 单元的一处 Bug 造成的，此 Bug 产生的原因请查看该文：EhLib 4.1 在 Delphi 2007 下无法过滤和排序的问题。（本示例代码中包含了修正过的 DBUtilsEh，请到 Delphi 盒子下载）

6、过滤行下拉列表的实现：Column.STFilter.ListSource、ListField、KeyField

DBGridEh 的过滤行可以将某列已存在的值以 ComboBox 的形式显示出来，要实现此功能，需设置 DBGridEh 列的 STFilter 属性。将 ListSource 设为要下拉显示的数据源，ListField 为下拉列表显示的字段，KeyField 为构建查询表达式使用的字段。

设置以上属性后，你可能发现虽然下拉列表显示出来了，但选择下拉列表值后，DBGridEh 却没有自动过滤，在单元的 initialization 节添加以下代码即会自动过滤了：

```
DBGridEhCenter.FilterEditCloseUpApplyFilter := True;
```

这个方法感觉很奇怪，而且在使用中好像也有问题，当输入 > <这样的表达式时，查询时会出错。在官方的 Demo 中，是通过内存表的方式实现的，而且不用设置任何属性（上面的 ListSource 等都不用设置），即可自动在过滤行每列处显示下拉列表，详细用法请参照 Ehlib 安装目录下的 Demo。

（注：本示例代码不包含过滤行下拉列表的实现演示）

3) 使用 DBGridEh 自动过滤实现方法 2

使用 DBGridEh 自动过滤实现方法：

1.所用到的控件（以 ADO 为列）：ADOQuery，DataSetDriverEh（ehlib），MemTableEh(ehlib),DataSource,DBGridEh

2.关联设置：DataSetDriverEh.ProviderDataSet:=ADOQuery;

MemTableEh.DataDriver:=DataSetDriverEh;

DataSource.DataSet:=MemTableEh;

DBGridEh.DataSource:=DataSource

3.DBGridEh 的关键设置：DBGridEh.STFilter.Local:=true;

DBGridEh.STFilter.Visible:=True;

```
                                        DBGridEh.Columns[i].STFilter.Visible:=True
;
```

4.单元文件 initialization 部分加入下面一句

```
                    DBGridEhCenter.FilterEditCloseUpApplyFilter := True;
```

5.单元文件 Uses 部分 添加 EhLibMTE 单元

4) DBGridEh 控件中使用过滤功能 (适用 ehlib 5.2 ehlib 5.3)

DBGridEh 控件中使用过滤功能(可以不用 MemTableEh 控件 适用 ehlib 5.2 ehlib 5.3)
EHlib 因为小巧而受到很多的 delphier 们的欢迎，不过 ehlib 5.2 及 ehlib 5.3 版本为了实现 office 2007 过滤风格，突然把直接支持 CDS 的过滤功能停了，取而代之的是必须使用 MemTableEh 控件做中转，这过程麻烦不说，一些朋友使用下来，发现慢无可忍，从而纷纷选择其它的控件替代，不过三方控件也不是想换就能换的，而且对新的控件，如果不熟悉，你不一定能最优使用他，于是就开始寻找是否还有希望使用单选的功能的 dbgrideh 来
经过多方代码碴找
对 DbUtilsEh.pas 做以下修改，终于与以前一样使用，而且类 office 2007 的选择功能也出来了！

```
procedureTDatasetFeaturesEh.FillSTFilterListDataValues(AGrid: TCustomDBGridEh; Column:
TColumnEh; Items: TStrings);
begin
end;
```

很是奇怪，这个地方为什么会留空呢，是否是因为用基它控件时，存在 BUG？ 不得而之....
我们做如下修改:

```
procedureTDatasetFeaturesEh.FillSTFilterListDataValues(AGrid: TCustomDBGridEh; Column:
TColumnEh; Items: TStrings);
begin
ifAssigned(AGrid.Center) then
AGrid.Center.StandardFillSTFilterListDataValues(AGrid, Column, Items);
end;
```

现在再用以前版本的方法，发现，实现的功能与用了 MemTableEh 的没有什么区别，也期待能没有其它的 BUG.....
比如过滤的条数 不能太多，设置的条件不能多个等.....
附 dbgrideh 中过滤设置的一些重要操作：
在 ehlib 的 DBGridEh 控件中使用过滤功能很方便，但使用过程中有几个容易被忽略的地方，它曾经困绕了我很长时间，过滤功能就是用不起来。
1.在 UNIT 中加入
uses EhLibXXX（EhLibADO、EhLibDBE 等，根据你选用的数据集来定）；

```
initialization
  DBGridEhDefaultStyle.FilterEditCloseUpApplyFilter := True;
```

//这句写在 .end 前面，同时上面这一句在不同的时候，可能有区别，你在编译不通过时，就要做修改!
.END

有如下的版本：
EhLibCDS
DBGridEhCenter.FilterEditCloseUpApplyFilter := True;
2.设置属性
DBGridEh.STFilter.Local := True;
DBGridEh.STFilter.Visible:= True;
3.设置需过滤字段的 STFilter 属性，这个地方我就不详细说了，相信大家都会了。

5) 支持模糊查询

```
\r\n
对 DBGRideh 控件添加类似于 QLDBGrid 的 CustomControl 支持。
\r\n
需要添加如下事件
\r\n
procedure TfrmBillTemp.dbgDetailColExit(Sender: TObject);
begin
  with TDBGridEh(Sender).Columns[TDBGridEh(Sender).SelectedIndex] do
  if ControlTypeEh =ctCustomControlEh then
    if CustomControlEh.ClassType = TQLDBLookupComboBox then
      TQLDBLookupComboBox(CustomControlEh).Visible :=    False;
end;
\r\n
procedure TfrmBillTemp.dbgDetailDrawColumnCell(Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumnEh;
  State: TGridDrawState);
var
  R:TRect;
begin
  if (gdFocused in State) and not TDBGridEh(Sender).ReadOnly then
  begin
    with Column do
    if ControlTypeEh =ctCustomControlEh then
      if CustomControlEh.ClassType = TQLDBLookupComboBox then
      begin
        ButtonStyle := cbsNone;
        with TQLDBLookupComboBox(CustomControlEh) do
        begin
          R := Rect;
          R.Left := Rect.Left+ TDBGridEh(Sender).Left+1;
          R.Right := Rect.Right + TDBGridEh(Sender).Left+1;
          R.Top := Rect.Top +1;
          R.Bottom := Rect.Bottom +1;
          R.TopLeft := ClientToScreen(R.TopLeft);
```

```
        R.TopLeft := TControl(CustomControlEh).ScreenToClient(R.TopLeft);
        R.BottomRight := ClientToScreen(R.BottomRight);
        R.BottomRight := TControl(CustomControlEh).ScreenToClient(R.BottomRight);
        CustomControlEh.BoundsRect := R;
        Left    := R.Left;
        Top     := R.Top;
        Width   :=  R.Right - R.Left;
        Height  :=  R.Bottom- R.Top;
        Visible :=   True;
      end;
    end;
  end;
end;
\r\n
procedure TfrmBillTemp.dbgDetailKeyPress(Sender: TObject; var Key: Char);
begin
  if (Key<>Chr(9)) and not TDBGridEh(Sender).ReadOnly then   //Tab 键
    with TDBGridEh(Sender).Columns[TDBGridEh(Sender).SelectedIndex] do
      if ControlTypeEh = ctCustomControlEh then
        if CustomControlEh.ClassType = TQLDBLookupComboBox then
          with TQLDBLookupComboBox(CustomControlEh) do   //转移焦点
          begin
            if Visible then
            begin
              SetFocus;
              //SendMessage(Edit.Handle,WM_Char,Word(Key),0);
            end else Abort;
          end;
end;
```

6) ehlib4.4.50 中支持模糊匹配的修改方法

新版的 ehlib 增加了不少功能,但是 dbgrideh 的首行过滤功能成了完全匹配了,

于是在源码里找了找,发现果然是这样,构造的条件语句是 like ,但是没有加上'%',于是小改了一下,就可以实现模糊匹配了

在 DBUtilsEh.pas 中的 第 972 行 改一下,如下:

```
//Result := Result + VarValueAsFilterStr(v);
//以下是 4.2.16 中的写法
if SupportsLike
   then Result := Result + "'%'+ VarToStr(v) +'%'"
   else Result := Result + VarValueAsFilterStr(v);
```

7) EhLib 5.0 Build 5.0.13 的过滤字串都是模糊过滤修改

会对 EhLib 5.0 Build 5.0.13 Russian version 版
修改 DBUtilsEh.pas 文件的第 927 行

```
function VarValueAsFilterStr(v: Variant): String;
  begin
    if VarType(v) = varDouble then
      Result := FloatToStr(v)
    else if VarType(v) = varDate then
      if @DateValueToSQLStringProc <> nil then
        Result := DateValueToSQLStringProc(Dataset, v)
      else
        Result := '''' + DateTimeToStr(v) + ''''
    else
      Result := '''%' + VarToStr(v) + '%''';
  end;
```

8) 滚动条滚动时选择不变，还有自动过滤功能的实现

DEMO 中 DBGRIDEH 滚动时选择的数据不会随滚动条或滚轮的变化而变化,是因为使用了 tmemtableeh, tdatasetdrivereh 控件读取数据,同时 ehlib 还有许多特效都与这两个数据控件有关.具体设置如下,(这是我找了很久一个个对比属性找出来的)

控件 dstasource1　　属性 dataset:mtquery1
控件 mtquery1(tmemtableeh):
　　属性 datadriver:dsdquery1
　　属性 fetchallonopen:t
　　属性 filtered:t
控件 dsdquery1(tdatasetdrivereh):
　　属性 providerdataset:query1
　　属性 resolvetodataset:F
控件 query1:
　　属性 autocalcfields:f
　　属性 cachedupdates:T
　　属性 databasename:dbdemos

```
SELECT
        Vendors.VendorNo VNo,
        Vendors.VendorName VName,
        Vendors.Preferred VPreferred,
        Parts.PartNo PNo,
        Parts.Description PDescription,
        Parts.Cost PCost,
        Items.Qty IQty
FROM Vendors, Parts, Items
WHERE Vendors.VendorNo = Parts.VendorNo AND
        Parts.PartNo = Items.PartNo
ORDER BY Items.Qty, Parts.Cost
```

9) 增量搜索

TDBGridEh 允许用户在网格列中实现特定的“增量”搜索。当用户进入增量搜索时他可以显示字符以及网格，并且在当前的列中查找文本。使用 dghIncSearch 和 dghPreferIncSearch 的值（在 OptionsEh 选项中) 在数据网格中操作增量搜索。 dghIncSearch 值允许在数据网格中进行增量搜索。运行时你能够使用下面的键进行增量搜索：
Ctrl+F - 开始增量搜索。
Ctrl+Enter - 查找下一个匹配记录。
Ctrl+Shift+Enter - 查找前一个匹配记录。
如果 OptionsEh 选项中的 dghIncSearch 是只读的，那么网络将自动设置增量模式在第一次按键以及 1.5 秒后返回普通模式。 dghPreferIncSearch 值决定网格设置自动增量搜索模式在第一次按键时替代单元编辑。

10) ehlib 总是按两次 ctrl+f 才出来查找框，怎么办？

引用 EhLib 包的 DBGridEhFindDlgs 單元提供的标準方法 ExecuteDBGridEhFindDialog():
就可以了

11) 如何改良 dbgrideh 的文字过滤

uses
  ehlibado;

改良 dbgrideh 的文字过滤，原先过虑需要填%字符串%，修改后，直接填入所要搜索的字符串即可，相关代码：

```
procedure TSTColumnFilterEh.InternalSetExpressionStr(const Value: String);
  procedure SetLookupKeyValues(var v: Variant);
  var
  i: Integer;
  begin
  if VarIsArray(Expression.Operand1) then
        for i := VarArrayLowBound(Expression.Operand1, 1) to
VarArrayHighBound(Expression.Operand1, 1) do
  v[i] := CurrentListDataSet.Lookup(CurrentListField, v[i], CurrentKeyField)
  else
  v := CurrentListDataSet.Lookup(CurrentListField, v, CurrentKeyField)
  end;
var s: string; // add by cxg
begin
  if Value <> '' then // add by cxg
  if Pos('%', Value) = 0 then
  s := '%'+value+'%';
  ParseExpression(s);
  FExpressionStr := s;
  if (CurrentKeyField <> '') and (CurrentListDataSet <> nil) and (CurrentListField <> '') then
  begin
  if FExpression.Operator1 <> foNon then
  SetLookupKeyValues(FExpression.Operand1);
  if FExpression.Operator2 <> foNon then
  SetLookupKeyValues(FExpression.Operand2);
  end;
end;
```

12) 改进 DBGrideh 表头点击自动排序,实现双击状态轮流

1. 在 optioneh 中设置：
2. AutosortMarking:=True;
3. 2.在每个字段中设置：
4. Titlebutton:=true;
5. 只要设置以上 2 个地方就会自动排序。。但永远都只有 2 种状态，一个是降序，一个是升序。。那如何实现 第一次点 降序。第二次点 升序。第三次点 取消排序（标题上的三角形隐藏）.
6. 请高手指点下。谢谢？

按 ctrl 再点击，就取消了。

这点知道。。。但是客户不可能知道...要做人性化点。请问还有别的方法么？谢谢!

```
procedure TfrmEquipRealManage.grd1TitleBtnClick(Sender: TObject;
 ACol: Integer; Column: TColumnEh);
```

```
var   sortstring:string;
begin
  inherited;
  with Column do
  begin
    if FieldName = '' then
      Exit;
    case Title.SortMarker of
      smNoneEh:
      begin
        Title.SortMarker := smDownEh;
        sortstring := Column.FieldName + ' ASC';
      end;
      smDownEh: sortstring := Column.FieldName + ' ASC';
      smUpEh: sortstring := Column.FieldName + ' DESC';
    end;
    try
      dstEquip.Sort := sortstring //dataset 为实际数据集变量名如：adoquery1
    except
    end;
  end;
end;
 try
    case Column.Title.SortMarker of
      smNoneEh:
      begin
        column.Title.SortMarker :=smUpEh;
        TADODataSet(Column.Field.Owner).Sort :=column.FieldName+' DESC';
      end;
      smUpEh:
      begin
        column.Title.SortMarker :=smdownEh;
        TADODataSet(Column.Field.Owner).Sort :=column.FieldName+' ASC';
      end;
      smdownEh:
      begin
        column.Title.SortMarker :=smNoneEh;
```

```
    end;
  end;
except
end;
```

13) 改良 Ehlib 的排序功能,加快排序速度

Ehlib3.0 版本以上虽然支持排序功能，但不支持带有 Order By 的 SQL 语句，而且排序很慢；我写的这个排序函数，利用 ADO 的 sort 方法，排序很快，几万条数据也是很快。该函数支持 Lookup 字段排序，不支持计算字段排序，因为计算字段值在内存里高速运算。排序分为：升序、降序和默认三种，支持排序图标。

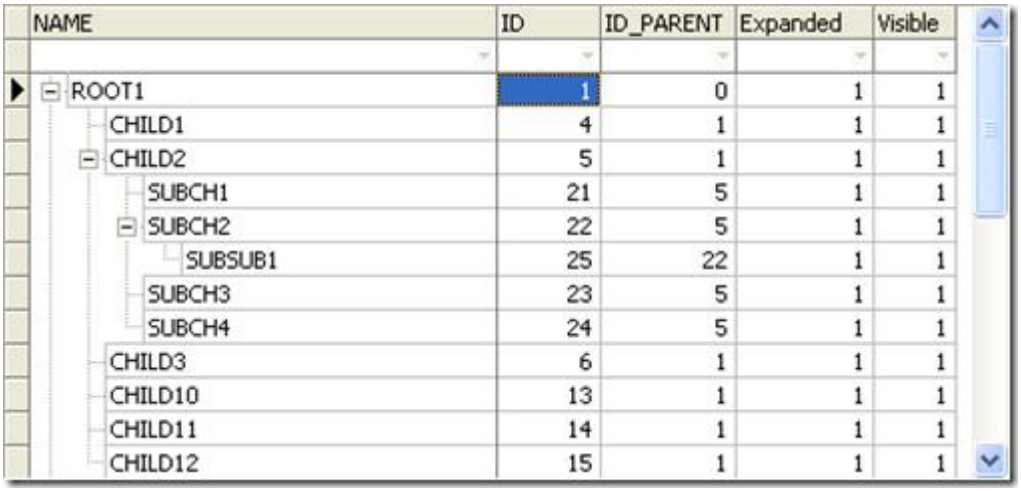
```
procedure  SortDBGridEh(Sender: TObject; ACol: Integer;
  Column: TColumnEh);
var
  FieldName, SortStr: string;
begin
  Screen.Cursor := crSQLWait;
  try
    if (Sender is TDBGridEh) and
      ((Sender as TDBGridEh).DataSource.DataSet <> nil) then
    begin
      if not ((Sender as TDBGridEh).DataSource.DataSet is TCustomADODataSet) then
        Exit;
      if not (Sender as TDBGridEh).DataSource.DataSet.Active then
        Exit;
      FieldName := Column.FieldName;
      if (Sender as TDBGridEh).DataSource.DataSet.FindField(FieldName).IsBlob then
        Exit;
      if (Sender as TDBGridEh).DataSource.DataSet.FieldByName(FieldName).FieldKind =
        fkData then
        SortStr := FieldName
      else if (Sender as TDBGridEh).DataSource.DataSet.FieldByName(FieldName).FieldKind
=
        fkLookup then
        FieldName := (Sender as
TDBGridEh).DataSource.DataSet.FieldByName(FieldName).KeyFields
      else
        FieldName := '';
      if (FieldName = '') or (Pos(';', FieldName) > 0) then
        Exit;
      case Column.Title.SortMarker of
        smNoneEh:
```

```
            begin
              Column.Title.SortMarker := smUpEh;
              TCustomADODataSet((Sender as TDBGridEh).DataSource.DataSet).Sort :=
                FieldName;
            end;
          smUpEh:
            begin
              Column.Title.SortMarker := smDownEh;
              TCustomADODataSet((Sender as TDBGridEh).DataSource.DataSet).Sort :=
                FieldName + ' DESC';
            end;
          smDownEh:
            begin
              Column.Title.SortMarker := smNoneEh;
              TCustomADODataSet((Sender as TDBGridEh).DataSource.DataSet).Sort := '';
            end;
        end;
      end;
    finally
      Screen.Cursor := crDefault;
    end;
end;
--程序实现如下：
--在 DBGridEh 的事件 OnTitleBtnClick 引用该函数即可：
procedure TFrmU_BasicSetup.dbgAddrCodeTitleBtnClick(Sender: TObject;
  ACol: Integer; Column: TColumnEh);
begin
   SortDBGridEh(Sender, ACol, Column);
end;
--为了保证表格的每一列都能点击触发排序，你需要将你需要排序的列属性
--   Title->TitleButton 设置为 True。
```

14) 在 DbGridEh 中显示 TreeView 效果

挺酷的效果，而显示设置也并不麻烦，但从 Demo 来看，好象只是对应于单表内的字段递归，是不是可以使用多个表，有待于考证，先做个记号：

```
MemTableEh1.TreeList.Active := True;
MemTableEh1.TreeList.KeyFieldName := 'ID';
MemTableEh1.TreeList.RefParentFieldName := 'ID_PARENT';
MemTableEh1.TreeList.DefaultNodeExpanded := True;
MemTableEh1.Open;
```



NAME	ID	ID_PARENT	Expanded	Visible
ROOT1	1	0	1	1
CHILD1	4	1	1	1
CHILD2	5	1	1	1
SUBCH1	21	5	1	1
SUBCH2	22	5	1	1
SUBSUB1	25	22	1	1
SUBCH3	23	5	1	1
SUBCH4	24	5	1	1
CHILD3	6	1	1	1
CHILD10	13	1	1	1
CHILD11	14	1	1	1
CHILD12	15	1	1	1

15) DBGridEh-KeyList、PickList

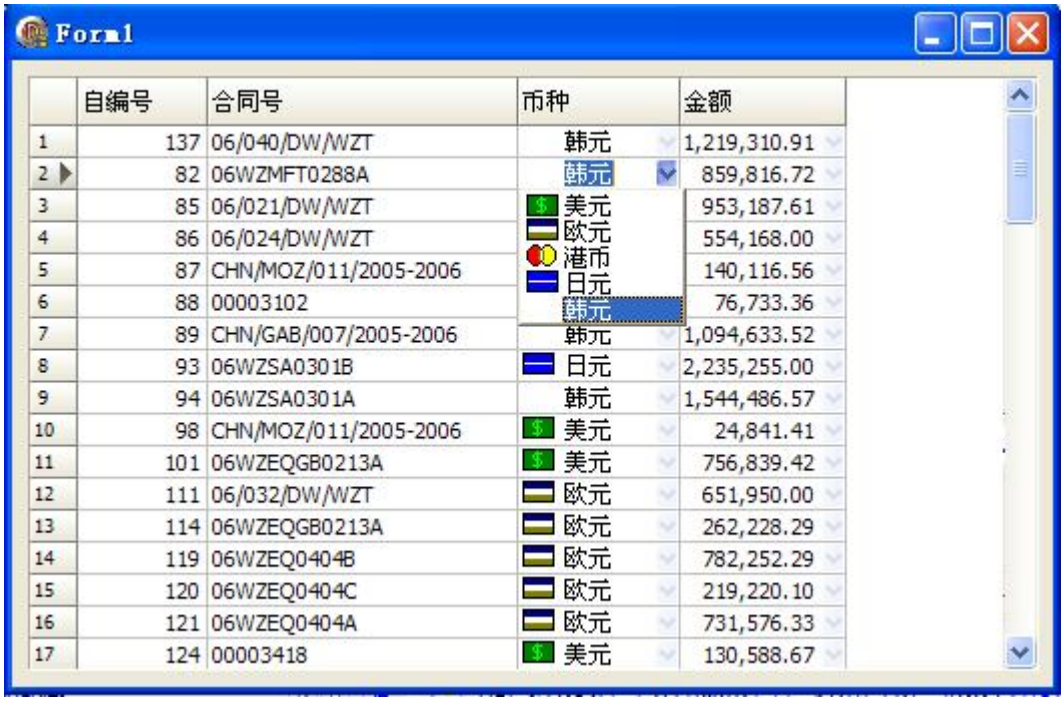
DBGridEh-KeyList、PickList
2011-03-27 14:15:26| 分类： Delphi | 标签： |字号大中小 订阅

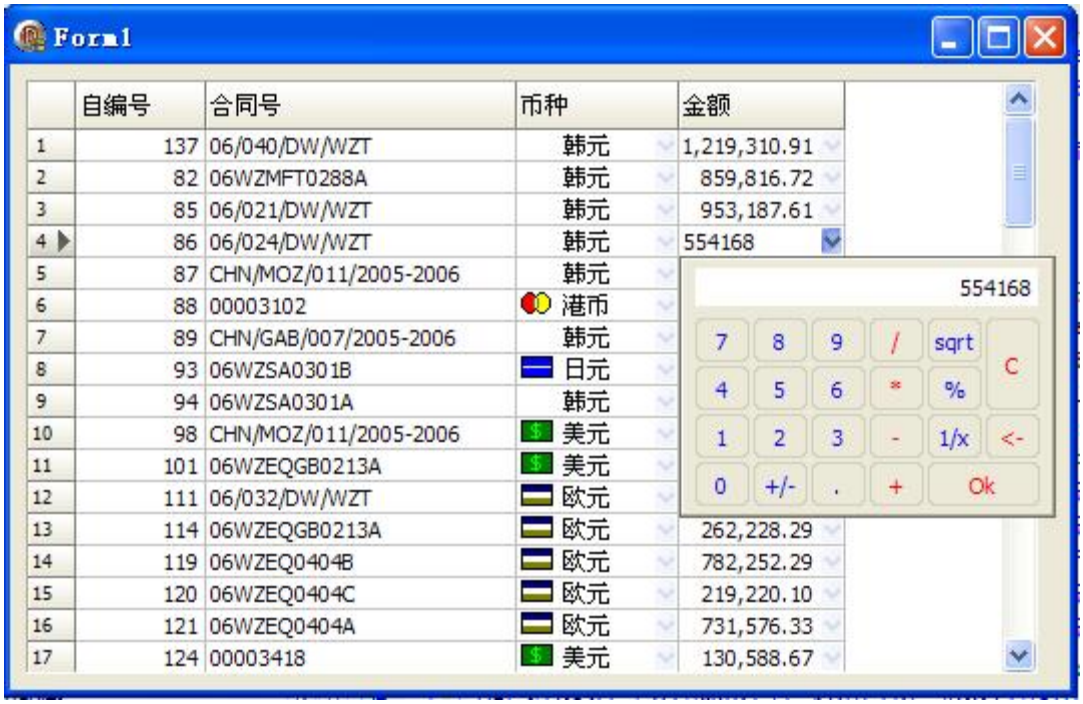
```
procedure TForm1.FormCreate(Sender: TObject);
begin
DBGridEh1.DataSource:=DataSource1;
DataSource1.DataSet:=MemTableEh1;
MemTableEh1.DataDriver:=DataSetDriverEh1;
MemTableEh1.FetchAllOnOpen:=True;
DataSetDriverEh1.ProviderDataSet:=ADOQuery1;
ADOQuery1.ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;'+
            'Data Source=f:\mcmygs.mdb;'+
            'Persist Security Info=False;'+
            'Jet OLEDB:Database Password=123456';
ADOQuery1.Close;
ADOQuery1.SQL.Text:='Select  自编号,合同号,币种,金额  from dzb';
ADOQuery1.Open;
MemTableEh1.Active:=True;
DBGridEh1.Flat:=True; //平面
DBGridEh1.TitleHeight:=20; //标题栏行高
DBGridEh1.OptionsEh:=DBGridEh1.OptionsEh+[dghShowRecNo]; //显示序号
DBGridEh1.ColumnDefValues.AlwaysShowEditButton:=True; //显示 DropDowm 图标
DBGridEh1.Columns[2].ImageList:=ImageList1;
DBGridEh1.Columns[2].ShowImageAndText:=True;
//PickList 为显示出来的值,与 ImageList 中的图标顺序相对应
DBGridEh1.Columns[2].PickList.Add('美元');
DBGridEh1.Columns[2].PickList.Add('欧元');
DBGridEh1.Columns[2].PickList.Add('港币');
DBGridEh1.Columns[2].PickList.Add('日元');
```

```
DBGridEh1.Columns[2].PickList.Add('韩元');
//KeyList 为数据库表中实际储存值,与 PickList 中值设置好相对应
DBGridEh1.Columns[2].KeyList.Add('USD');
DBGridEh1.Columns[2].KeyList.Add('EUR');
DBGridEh1.Columns[2].KeyList.Add('HKD');
DBGridEh1.Columns[2].KeyList.Add('JPY');
DBGridEh1.Columns[2].KeyList.Add('KRW');
DBGridEh1.Columns[2].Width:=80;
DBGridEh1.Columns[3].ButtonStyle:=cbsDropDown; //显示计算器
DBGridEh1.Columns[3].DisplayFormat:='#,###,###.00';
DBGridEh1.Columns[3].OptimizeWidth;  //自适应宽度
end;
```


```
//PickList为显示出来的值,与ImageList中的图标顺序相对应
DBGridEh1.Columns[2].PickList.Add('美元');
DBGridEh1.Columns[2].PickList.Add('欧元');
DBGridEh1.Columns[2].PickList.Add('港币');
DBGridEh1.Columns[2].PickList.Add('日元');
DBGridEh1.Columns[2].PickList.Add('韩元');
//KeyList为数据库表中实际储存值,与PickList中值设置好相对应
DBGridEh1.Columns[2].KeyList.Add('USD');
DBGridEh1.Columns[2].KeyList.Add('EUR');
DBGridEh1.Columns[2].KeyList.Add('HKD');
DBGridEh1.Columns[2].KeyList.Add('JPY');
DBGridEh1.Columns[2].KeyList.Add('KRW');
```

这样就将KeyList与Picklist对应起来,如数据库表中的值为EUR,则会自动在PickList中检索该值对其对应





16) 主从表设置

第一种在数据源的属性里设置主从表，从表的 mastersource 设置为主表的数据源，masterfields 设置为主表相关的记录字段值如 OrderNo,即可，不过这适用于数据量不大的时候如果数据量过大不是按第二种方法编码。

第二种主从表关系连接很简单地，相信大伙们都会地。从表查询地参数来源于主表中地某 1 个字段地值，此可以住 SQL 语句通过使用参数变量(var)名及主表地相应字段名相同得到解决。俺之所以出现此个问题和疑问就为俺主表住滚动 1 笔记录时，从表需需要重新得到数据，即主表地 OnAfterScroll 事件里需需要加入以下编程代码(Code)：

从表.close;

从表.open;

就可以呢。。。

ADO 中 **ADOTable**、**ADOQuery** 和 **ADODataSet** 主从表设置

1、**ADOTable** 的主从表设置

主表名为 **jxc_out**，从表名为 **jxc_out_detl**　主表与从表关联字段为 **draw_no**

tbM: TADOTable;
tbD: TADOTable;

dtsM: TDataSource;
dtsD: TDataSource;

tbM.TableName=jxc_out
tbD.TableName=jxc_out_detl

dtsM.DataSet=tbM
dtsM.DataSet=tbM
tbD.MasterSource=tbM

在从表的**[tbD]**中选择 **MasterFields** 属性，选择主表和从表关联的字段，然后执行 **tbM.open** 和 **tbD.Open** 即可

2、**ADOQuery** 的主从表设置

主表名为 **jxc_out**，从表名为 **jxc_out_detl**　主表与从表关联字段为 **draw_no**

adqM: TADOQuery;
adqD: TADOQuery;

dtsM: TDataSource;
dtsD: TDataSource;

tbM.sql.text=select * from jxc_out
tbM.sql.text=select * from jxc_out_detl where draw_no=:draw_no

dtsM.DataSet=adqM
dtsM.DataSet=adqD
dtsD.DataSource=dtsM

执行 **adqM.open** 和 **adqD.Open** 即可

(在单元格单击事件里面写

datamodule1.MemTableEh2.close;

datamodule1.QryInUseBom.Close;

datamodule1.QryInUseBom.Parameters.ParamByName('code1').Value :=
dbgrideh1.DataSource.DataSet.fieldbyname('编码').value;

datamodule1.QryInUseBom.Parameters.ParamByName('code2').Value :=
dbgrideh1.DataSource.DataSet.fieldbyname('编码').value;

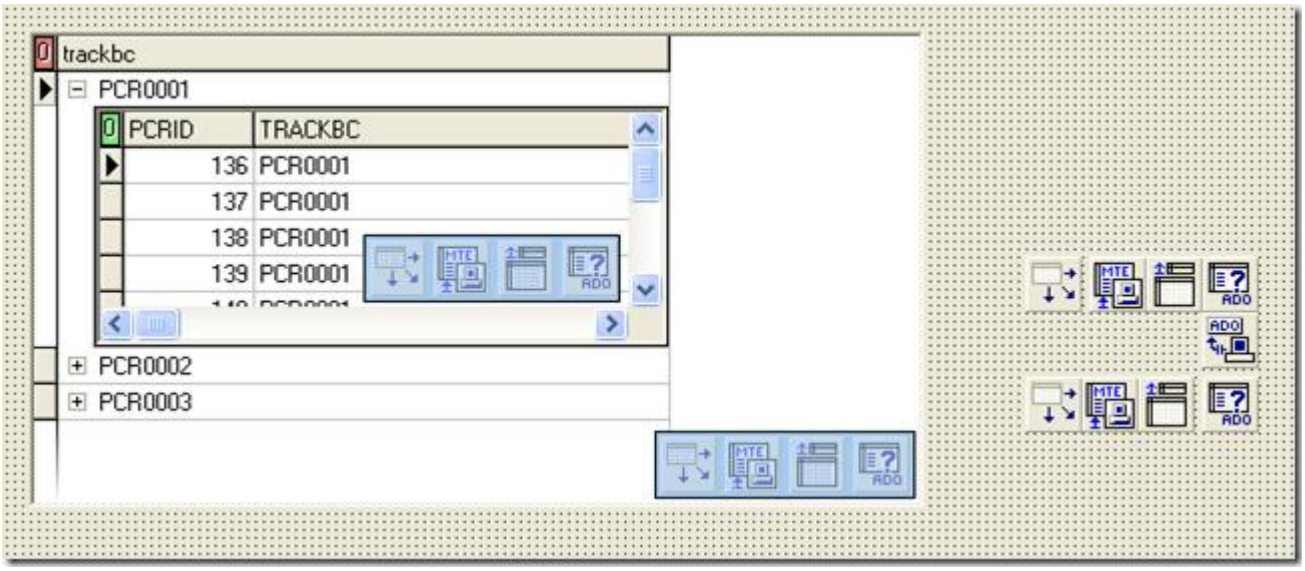
datamodule1.QryInUseBom.Open;

datamodule1.MemTableEh2.open;)

3、ADODataSet 主从表设置

跟 **ADOQuery** 设置差不多，不过 **ADODataSet** 设置的是 **commandtext**

17) 在 DbGridEh 中显示表中表



操作时写的代码并不多，主要问题是需要将所有的数据读内存表不太合适，在修改上应该可以将数据源改为动态。使用的重点是设置 DbGridEh1 的 RowDetailPanel 打开，再将 DbGrid Eh2 放入，设置 MemTableEh2 的 MasterFields 和 DetailFields，接入数据源，好，出锅！

6. 输入/输出

1) 导入导出数据

导入/导出数据在实际处理过程中是比较烦琐的。但是 Enlib3.0 提供了一系列函数让你轻松实现此功能，而且支持的文件格式很多：Text，Csv，HTML，RTF，XLS 和内部数据格式。除此之外，还可对任意选择的数据区域进行操作。函数如下：

Pascal:

```
SaveDBGridEhToExportFile(TDBGridEhExportAsText,DBGridEh1,'c:\temp\file1.txt',F
alse);
```

C++:

```
SaveDBGridEhToExportFile(__classid(TDBGridEhExportAsText),DBGridEh1,"c:\\temp\
\file1.txt",false);
```

说明：其中 false 参数表示导出的是选中的局部数据区域数据，true 则为整个表格数据。

例：将当前表格中数据导出为 EXCEL 等格式文件。

在窗体中添加一个 SaveDialog 组件和“导出”按钮 B_exp，在“导出”按钮的 click 事件中添加如下代码：

```
procedure TForm1.B_expClick(Sender: TObject);
var
ExpClass:TDBGridEhExportClass;
Ext:String;
begin
SaveDialog1.FileName := 'file1';
if (ActiveControl is TDBGridEh) then
if SaveDialog1.Execute then
begin
case SaveDialog1.FilterIndex of
1: begin ExpClass := TDBGridEhExportAsText; Ext := 'txt'; end;
2: begin ExpClass := TDBGridEhExportAsCSV; Ext := 'csv'; end;
3: begin ExpClass := TDBGridEhExportAsHTML; Ext := 'htm'; end;
4: begin ExpClass := TDBGridEhExportAsRTF; Ext := 'rtf'; end;
5: begin ExpClass := TDBGridEhExportAsXLS; Ext := 'xls'; end;
else
ExpClass := nil; Ext := '';
end;
if ExpClass nil then
begin
```

```
if UpperCase(Copy(SaveDialog1.FileName,Length(SaveDialog1.FileName)-2,3))
UpperCase(Ext) then
SaveDialog1.FileName := SaveDialog1.FileName + '.' + Ext;
SaveDBGridEhToExportFile(ExpClass,DBGridEh1,SaveDialog1.FileName,False);
//其中 false 为局部数据
end;
end;
end;
```

2) 从多种格式导入/导出数据到 TDBGridEh

EhLib 的函数集可以从 DBGridEh 导出数据到 Text, Csv, HTML, RTF, XLS 以及其内部格式。它可以保存数据到流（TStream 对象）或文件。例子 Pascal:
SaveDBGridEhToExportFile(TDBGridEhExportAsText,DBGridEh1,'c:/temp/file1.txt',False); C++:
SaveDBGridEhToExportFile(__classid(TDBGridEhExportAsText),DBGridEh1,"c://temp//file1.txt",false); EhLib 的函数集可以从 Text 以及其内部格式的数据导入到 DBGridEh 的数据集中。它可以从文件中读取数据或读取数据到流（TStream 对象）。

3) DBGRID 生成 EXCEL 报表

从盒子上下载了一个别人做的单元文件，可以用来让 DBGRID 生成 EXCEL 报表。可是公司内部由于有多个门店，也就是我需要分别连接这些服务器然后通过这个单元生成一个 EXCEL XLS 文件，里面会有多个 sheet，代表各个门店的数据。sheet 的名字会以门店名字为主。现在我将这个单元修改了一下，已经可以实现如上效果，大致原理就是通过 checklistbox 这个控件取出所有门店信息，可以勾选。点击按钮，自动取出 checklistbox 控件被勾选门店的 IP 地址并进行逐个连接，然后查询并显示 DBGRID 之后全部写入 EXCEL。

可是问题来了，当某个店数据超过 EXCEL 的限制：65536 行 时，如何修改可以使其自动新增一个 SHEET，并将多余的数据插入另一页面？而且不能影响后续的其他门店数据跑取。请高手们抽个空，帮忙看看改一改。谢谢了。函数有点乱。请大家不要笑话。

函数使用方式：

```
DBGridToExcel(dbgrid1,nil);
```

函数内容：

```
type
 function DBGridToExcel2(dgrSource: TDBGrid;
      UpAniInfo: TUpAniInfoProc = nil): Integer;
```

```
function DBGridToExcel(dgrSource: TDBGrid; UpAniInfo: TUpAniInfoProc): Integer;
const        //从 DBGrid 导出到 Excel(2005.8.23 改进至可以导入几乎无限的数据)
   //Excel 每 Sheet 最大行数
  MAX_VAR_ONCE   = 65535;    //一次导出的条数
var        //返回导出记录条数
  MyExcel, varCells: Variant;
  MySheet, MyCells, Cell1, Cell2, Range: OleVariant;
  iRow, iCol, iSheetIdx, iVarCount, iCurRow: integer;
  CurPos: TBookmark;
  i:integer;
  MAX_SHEET_ROWS:integer;
begin
  dgrSource.DataSource.DataSet.DisableControls;
  dgrSource.DataSource.DataSet.First;
  MyExcel := CreateOleObject('Excel.Application');
  MyExcel.WorkBooks.Add;
  MyExcel.Visible := False;
  begin
  if dgrSource.DataSource.DataSet.RecordCount <= MAX_VAR_ONCE then
    iVarCount := dgrSource.DataSource.DataSet.RecordCount
  else
    iVarCount := MAX_VAR_ONCE;
  varCells  := VarArrayCreate([1,
         iVarCount,
         1,
         dgrSource.FieldCount], varVariant);
  end;
  iSheetIdx := 1;
  iRow      := 0;
  Result    := 0;

//这里是循环从 checklistbox 取门店 IP 进行连接
for i:=0 to sqltoexcel.clb1.Items.Count-1 do
if sqltoexcel.clb1.Checked[i] then
 begin
 sqltoexcel.qrycon.Locate('sto_name',sqltoexcel.clb1.Items.Strings[i],[loCaseInsensitive]);
 if sqltoexcel.qrycon.fieldbyname('sto_serverip').AsString<>'' then
```

```
 begin
 with sqltoexcel do
  begin
      query1.Close;
      con1.Connected:=false;
      con1.ConnectionString:='Provider=SQLOLEDB.1;Password=123456;';
      con1.ConnectionString:=con1.ConnectionString+'Persist Security Info=True;';
      con1.ConnectionString:=con1.ConnectionString+'User ID=sa;Initial Catalog=test;';
      con1.ConnectionString:=con1.ConnectionString+'Data Source='+qrycon.fieldbyname('sto_serverip').AsStri
ng;
      //导入 SQL 语句
      query1.Close;
      query1.SQL.Clear;
      memo1.Lines.SaveToFile('C:\1234.txt');
      query1.SQL.LoadFromFile('c:\1234.txt');
      query1.Open;
  end;
      dgrSource.DataSource.DataSet.First;
 //新增一个 Sheet
    if iSheetIdx <= MyExcel.WorkBooks[1].WorkSheets.Count then
      MySheet := MyExcel.WorkBooks[1].WorkSheets[iSheetIdx]
    else
      MySheet := MyExcel.WorkBooks[1].WorkSheets.Add(NULL, MySheet);
//加在后面

 //设置名字
      if length(sqltoexcel.qrycon.fieldbyname('sto_name').AsString)>16 then
       MyExcel.workbooks[1].WorkSheets[iSheetIdx].name:=copy(sqltoexcel.qrycon.fieldbyname('sto_name').As
String,16,length(sqltoexcel.qrycon.fieldbyname('sto_name').AsString)-15)
      else
       MyExcel.workbooks[1].WorkSheets[iSheetIdx].name:=sqltoexcel.qrycon.fieldbyname('sto_name').AsStrin
g;
    MyCells := MySheet.Cells;
    Inc(iSheetIdx);
    iRow := 1;
    for iCol := 1 to dgrSource.FieldCount do
    begin
      MySheet.Cells[1, iCol] := dgrSource.Fields[iCol-1].DisplayName;
```

```
    MySheet.Cells[1, iCol].Font.Bold := True;
    MySheet.Columns[iCol].ColumnWidth := dgrSource.Fields[iCol-1].DisplayWidth;
    if (dgrSource.Fields[iCol - 1].DataType = ftString)
      or (dgrSource.Fields[iCol - 1].DataType = ftWideString) then
    begin        //对于“字符串”型数据则设 Excel 单元格为“文本”型
      MySheet.Columns[iCol].NumberFormatLocal := '@';
    end;
  end;
  Inc(iRow);
MAX_SHEET_ROWS:=65535;
while not dgrSource.DataSource.DataSet.Eof do
 begin
  iCurRow := 1;
  while not dgrSource.DataSource.DataSet.Eof do
  begin
    for iCol := 1 to dgrSource.FieldCount do
    begin
      varCells[iCurRow, iCol] := dgrSource.Fields[iCol-1].AsString;
    end;
    Inc(iRow);
    Inc(iCurRow);
    Inc(Result);
    dgrSource.DataSource.DataSet.Next;
    if (iCurRow > iVarCount) or (iRow > MAX_SHEET_ROWS + 1) then
    begin
      if Assigned(UpAniInfo) then
        UpAniInfo(Format('(已导出%d 条)', [Result])); //显示已导出条数
      Application.ProcessMessages;
      Break;
    end;
  end;

  Cell1 := MyCells.Item[iRow - iCurRow + 1, 1];
  Cell2 := MyCells.Item[iRow - 1,
       dgrSource.FieldCount];
  Range := MySheet.Range[Cell1 ,Cell2];
  Range.Value := varCells;
```

```
  if (iRow > MAX_SHEET_ROWS + 1) then    //一个 Sheet 导出结束
  begin
    MySheet.Select;
    MySheet.Cells[1, 1].Select;   //使得每一 Sheet 均定位在第一格
  end;
  Cell1   := Unassigned;
  Cell2   := Unassigned;
  Range   := Unassigned;
 end;  //end of big while

 end;//end of if
 end;//end of for
 MyCells  := Unassigned;
 varCells := Unassigned;
 MyExcel.WorkBooks[1].WorkSheets[1].Select;  //必须先选 Sheet
 MyExcel.WorkBooks[1].WorkSheets[1].Cells[1,1].Select;
 MyExcel.Visible := True;
 MyExcel.WorkBooks[1].Saved := True;
 MyExcel  := Unassigned;
 dgrSource.DataSource.DataSet.EnableControls;
end;
```

4) 使用 TPrintDBGridEh 组件

在 Delphi 的 IDE 组件面板中选择 TPrintDBGridEh 并拖放其到 form 上。在 其 DBGridEh 属性中设置为相应的 TDBGridEh 组件。使用 Print 以及 Preview 方法在打印机上打印网格或者在预览窗体中预览网格。

5) 打印时确定 Ehlib 定义的报表表头颜色？

在执行打印之前，加上下面的语句：

```
DBGridEh1.FixedColor:=clLime;//clLime 可以换成你想要的颜色比如 clRed 是红色等等
PrintDBgridEh1.Options:=[pghColored,pghFITingByColWidths];//方括号里的
"pghColored"是必需的，其它的根据你的需要决定取舍
```

6) Ehlib 中的 PrintDBGridEh 如何印页码,即第几页共几页

PrintDBGridEh.PageFooter.CenterText:='第&[Page]页 共&[Pages]页'

在对象管理器中的相关位置有设

7) 怎么让 PrintDBGridEh 只打印 DbGridEh 中指定的列

１、才１５列，我一般都会有３０列以上，PrintDBGridEh 可以支持

正好打印在一页上（按宽度）

２、如果实在想指定列，了不起用代码在打印的时候将其他列全部 Hide,然后打印，打印完成后再将隐藏的列显示出来不就结了，对吧？

仅对 Column.visible 进行操作就好了

8) 怎样进行横向打印 / 打印预览？

TPrintDBGridEh 并没有专六的属性来设置页面特性。在调用打印或打印预览方法前，你必须设置你将要执行打印的打印源（ Orientation ）。

```
uses ......, PrViewEh, Printers.
..............
procedure TForm1.bPreviewClick(Sender: TObject);
begin
PrintERPreview.Orientation := poLandscape;
PrintDBGridEh1.Preview;
end ;
```

7. 将存在的 DBGrid 组件转换为 DBGridEh 组件

通过笔者上述介绍，想必你已经对 Enlib 组件包产生好感而且越越欲试了，那就赶快下载使用吧。但是，使用一段时间并且喜欢上它后，你又有新的问题产生了，那就是为了保持界面风格一致，能否将已经开发完成的应用程序中的 DBGrid 组件能否转换为 DBGridEh 组件，进行一次彻底革命？答案是肯定的。尽管 DBGridEh 并不是继承于 CustomDBGrid 组件,但是 DBGridEh 和 DBGrid 它们之间有许多相同之处.因此可以相互转换。

具体步骤如下：

在 Delphi IDE 下打开 TDBGrid 组件;

通过组合键 Alt-F12 将 form 以文本方式显示;

将所有 TDBGrid 对象名改变为 TDBGridEh 对象名，如：DBGrid1: TDBGrid 改为 DBGrid1: TDBGridEh;

再次通过组合键 Alt-F12 将文本方式恢复为 form 显示;

将 form 各相关事件中定义的所有 TDBGrid 改为 TDBGrideh,如 DBGrid1: TDBGrid 改为 DBGrid1: TDBGridEh;

重新编译应用程序。

三、 EhLib 安装问题

1. EhLib 安装步骤

(1)新建文件夹:D:/Program Files/Borland/Delphi7/下新建目录 Ehlib

(2)在 Tools->Environment Options->Library->Library Path 中添加这个目录.

(3)将 Common 和 Delphi7 中的目录的文件拷贝到上面的这个目录中

(4)打开 runtime package - EhLib70.Dpk 文件,点击"Compile"编译这个 package

(5)打开并编译 EhLibDataDrivers70.Dpk

(6)在编译完 run-time packages 后安装 DclEhLib70.BPL 和 DclEhLibDataDrivers70.BPL (design-time packages)into the IDE.BPL 文件在你自己定义的:$(DELPHI)/Projects/Bpl

(7)打开 DclEhLib70.Dpk(design-time package)编译,然后点击"Install"to register EhLib

components on the component palette.]

(8)最后打开和安装 DclEhLibDataDrivers70.Dpk package

这个时候在 Ehlib 选项卡下可以看见 20 个 Ehlib 组件,安装成功

2. EhLib 安装问题（dbsumlst.dcu 出错)

如果你在安装 EhLib 4.1.4 等版本过程中出现下面的错误提示,可以按照解决方法来安装:

1. 把 EhLib 中的 common 和 DataService 文件拷贝到 Delphi7 目录中.

2.在 TOOLS->Environment Options->Library->Library Path 中添入 EHLIB 路径。

3.打开新建文件夹中的 EHLIB70.DPK ,编译一下，但不要安装。

4.打开 Ehlib 中的 DclEhLib70.DPK,编译,安装

5. 在 Delphi 7 中打开 DclEhLib70.dpk，编译并安装。

6. 组件面板中出现一个 EhLib 的组件页。

7. 打开附带的 DEMOS，编译并运行，测试安装成功。

我是按照这个方法安装的，可是为什么按照他的 3.4 步做的编译会出错啊！也不是下载的文件的问题，下过了两个了！

[Fatal Error] EhLib70.dpk(59): File not found: 'DBSumLst.dcu'

[Warning] DclEhLib70.dpk(4): File not found: 'EhLibReg.dcr'

这样会出错

解决方法:

<一>在菜单

tool->environment options->libriary 中加入控件所在的路径

< 二>这个问题我已解决把/COMMON/*.PAS *.DFM 等文件复制到 EHLIB.DPK 所在的目录下,也就是 DELPHI7 文件夹下,这个文件夹下有好几个 Ehlib70.dpk,dclEhlib70.dpk 的文件,先打开 Ehlib70.dpk 后,先编译,退出,再打开 dclEhlib70.dpk,编译,再点击 INSTALL 安装就 OK 了,如果找不到 DBSumlst.dcu 文件,则可以添加路径,在 TOOL 下的 E..,

如:你用 DELPH7 则复制到/DELPHI7/ 然后再编译安装

我有十成的把握可以解决你的问题,因为我也遇到这种问题,呵呵!!

你只要把那个 ehlib 目录下的 delphi7 目录下的那个 ehlib.dpk 和 EHLIB.RES 拷贝到 ehlib 那个 common

下面,然后在 delphi7 关闭所有的窗体,打开 common 下的那个 ehlib 编译安装,ok!!!

3. 安装提示找不到.BPL 文件

(这一步的时候出现 Can't load package D:\D7outtmp\DclEhLib70.bpl 找不到指定模块)

生成的文件在 Delphi7\Projects\Bpl 中,将 DclEhLibDataDrivers70.bpl, EhLib70.bpl, EhLibDataDrivers70.bpl 四个文件复制到 C:\WINDOWS\system32 中或 Delphi7\Bin 中。

四、 Delphi 下的优秀表格(Grid)显示控件

有 Devexpress, AdvStringgrid, Topgrid, Nextgrid。它们各有特点,如何选用?

1. NextGrid

NextGrid 的官方网站:

http://www.bergsoft.net/component/next-grid/overview.htm

目前版本已经出到了 4.4 了。

难得是它有专门的 Freeware 版本,不注册照样可以使用,注册了的话,则得到更多同属 Next 系列的界面控件。

NextGrid 的 UI 设计非常舒服, 在 DesignTime 的一些考虑也不错, 如果你的程序需要一个轻量级的表格控件, 可以考虑。

2. TopGrid 3.01

官方网站: http://www.objectsight.com/TopGridOverview.htm

这个控件适应面很广, 可以胜任 StringGrid, DrawGrid, DBGrid 的所有工作, 不限于数据库操作。而且 TopGrid 的界面非常的 Cool, 有点像 PowerBuild 里的 DataWindow 的感觉, 而且是更加美观。功能十分灵活, 可以非常方便的操作 Grid 中的每一个 Cell, 唯一的缺点是编译出来的 EXE 文件大了点, 不过用 ASPack 压一下就可以两全了。

3. XLGrid

主页：http://www.gavina-software.com/components/xlgrid

用这个控件让人想起了 MS 的 Excel。强大。不支持数据库操作。

可惜已经无人维护了，主页也打不开了！

这里有一个最新的：

XLGrid v1.7 For D5-7 汉化版 (http://www.2ccc.com/article.asp?articleid=1119)

XLGrid 1.7 汉化改进版 Beta 2 (http://www.winu.cn/thread-34094-1-1.html)

4. DevExpress ExpressQuantumGrid

主页：http://www.devexpress.com/

在 ExpressQuantumGrid Suit 中，最强大的表格控件库和数据编辑控件库曾经都是针对 Delphi 和 C++Builder 平台开发的。拥有 ExpressQuantumGrid 组件，你就可以制作具有惊人的视觉效果的界面以及给你的终端用户带来难以置信的特性和灵活性，而实现这一切不需要编写简单的一行代码。

在 ExpressQuantumGrid Suit 中的特征：

多种数据模式

ExpressQuantumGrid 组件允许你从传统数据源（Data-Aware 模式），非数据库数据源（Unbound 模式）以及用户数据结构（provide 模式）中进行数据绑定。

快速的数据载入

ExpressQuantumGrid 并没有达到 ExpressDataController 与之媲美的速度。只要你拥有了我们开发先进的 ExpressDataController 的技术,你就可以在一眨眼工夫里迅速载入 100,000 行数据，并且用同样的速度来进行分组和排序处理。但是只有眼见为实，因此你可以尝试一下我们发布在网页上的演示程序来观看实际的效果。

基于架构的视图

在 ExpressQuantumGrid 里的每一个水平定位均可由不同的视图（卡片视图，列表视图）来表示。视图甚至可以被更改。

主要内容和细节内容

Developer Express 是第一家提供具有综合的数据分组特点的主要内容/细节内容列表和以列表或卡片便捷地显示主要内容/细节内容功能的可视化控件厂商。

同步复制细节内容

同步复制细节内容功能的激活可以通过设置“列表视图中的同步”属性来完成。默认情况下设置为真值。

未绑定的列

现在你可以设置一些属性来显示和编辑在绑定的列表视图中的未绑定列。

自动的数据分组

不需要编写简单的一行代码，只要设置一个属性，你就可以在你的应用程序里面激活异常强大的数据查看和控制功能。在分组模式中，ExpressQuantumGrid 不能以只读方式来显示信息。分组对所有的表格水平定位都生效，并且以闪电般的速度来运行，然而只占用非常少的内存空间。

对任意数量的列进行自动排序

ExpressQuantumGrid 允许你在不编写简单一行代码的情况下快速地对多个列进行排序。

可扩展的分组功能

如果你曾经想显示一个列表视图并使得分组记录的内容可以被扩展，那么你将会深深地喜欢这个新特性。只要一个的简单的设置，你现在就可以轻松地实现扩展分组功能。

条形列/多层列（Banded/Stacked Columns）

你可以使用条形列来最大化终端用户可视的列数。并能简单方便地在表列头内的任意位置显示图片。

卡片视图（CardView）

象 MS Outlook 一样，你可以在 QuantumGrid 组件中通过使用卡片布局格式来显示数据。

卡片的扩展和折叠

卡片视图允许你扩展和折叠个人的卡片，从而使你的终端用户轻松地消化理解显示在巨大数据记录集合中的信息。

Outlook2003 风格的数据分组功能

与 Office2003 引入的风格一样，你现在就可以在屏幕上唯一地显示分组信息。

Outlook2003 风格支持

随着 QuantumGrid 版本 5 的发布，我们通过仿效 Microsoft2003 外观特征，引入了焕然一新的喷绘风格。这种风格同时也扩充到列表和个人编辑界面中，从而允许你在整个应用程序开发中都保持一个统一的风格。

背景图片支持

你想要在你的列表中显示背景图片吗？如果是的话，ExpressQuantumGrid 组件将帮助你轻松地完成该功能。

运行时行的大小调整

ExpressQuantumGrid 组件允许你以及你的用户不管是在程序设计中还是程序运行时，都可以方便地调整行高度。

内嵌的数据导航器

为了帮助简化开发过程，ExpressQuantumGrid 套件包含了一个技术等级的数据导航器，它可以独立启动，也可以作为 Grid 容器中的一部分整合其中，从而帮助你完全优化屏幕空间占用。

为所有的分组结点提供快捷的运行时摘要

不需要编写简单的一行代码。你可以摘要归纳信息和免除通过传统的报表编辑器来编写复杂的条形报表。

Instant Runtime Summary Footers

不需要编写简单的一行代码。你可以制作引人注目的列表显示风格以及使你的应用程序成为一个数据分析引擎。

自动运行时的列选择

拥有了 QuantumGrid,你的终端用户就可以通过直观的拖拉操作轻松地定制屏幕上可视化的列，这一点和 MS Outlook 一样。

快速的自定义 Column and Band

与 Column and Band 对应的技术指标区域并不是毫无价值的。只要激活了 OptionsCustomize.ColumnQuickCustomizationOptions 和 OptionsCustomize.BandsQuickCustomizationOptions,你就可以使他们变成强大的工具来绑定表列和条的可视性。

内置的 MS Excel 风格的表格过滤功能

拥有了 QuantumGrid，你就可以快速地添加列数据过滤功能而不需要编写简单的一行代码。

对以前引进的过滤器功能提供 MRU（最近使用的）支持

最近使用的过滤器现在对你的终端用户仍然有效。

固定的 Bands and Columns

可以方便地将 Column and Band 固定在 QuantumGrid 的左边或右边。

内置的 XP 主题风格支持

拥有了 ExpressQuantumGrid 组件,你可以在你的应用程序里面快速地激活 XP 的主题风格。

5. TMS Grid Pack

官　网：http://www.tmssoftware.com/

该控件套包包含了 TAdvStringGrid、TAdvColumnGrid、TAdvSpreadGrid、TAdvGridExcelIO、TAdvGridRTFIO、TDBAdvGrid 等多个控件。

通过属性你可拥有：

页码、日期、时间、页面顶端与底部的标题

顶端与底部标题的不同的字体

使用显示与不同打印字体控制中的单元字体和色彩

如需要，可自动分割多页面中的列

打印表格中的选定区域

设置页边距、打印位于中心的内容

使用重复的 fixedrows 与 fixedcolumns 进行多页面输出

自动调整适合页面的尺寸

打印前预览

为将打印的每个单元格指定边框

附加自定义打印

换行、富文本格式的文档以及 HTML 格式文本的打印

HTML 格式的打印标题

为每个单元格的文本调整位置与方向
每个单元格能有不同的字体与背景颜色
从单行、多行中选择自动换行
将 URLs 显示为真正的可点击的 URLs
当文本与列大小不符时，划省略号
隐藏的列
壁纸
富文本单元格
HTML 格式单元格
自动进到下一单元格
单键插入与删除行
自动单元格剪切与粘贴
用 autoadvance 进行掩码编辑
用鼠标点击进行快速光标定位
当点击列标题时间用多种分类方法与定制进行自动分类
单元格搜索功能
单元格编辑禁用功能
每个单元格不同的提示
增量键查找
动态滚动提示
简便智能化的剪贴板功能，具有自动填写与自动延范围和富文本功能
通过位置指示加强了行与列的移动
自动跳过固定与只读的单元格
保存到文件，从文件加载
保存到 CSV，从 CSV 加载
保存到 XLS、XLS 表单，从 XLS、XLS 表单加载（本地支持，无需安装 Excel）
保存到流，从流加载
将原始色彩、位置方向与字体保存到 HTML
从 ASCII 格式的文件保存与加载
保存到 Word DOC 文件
保存到 XML 文件
从 MDB 表格加载
对编辑控制作出你自己的选择
正常的左或右位置方向编辑控制
带有自动单元格改进的掩码编辑控制
复选框、comboboxes、复选框的单元格数据
日期挑选程序、时间挑选程序、具有上下选择功能的日期挑选程序

具有省略键的编辑控制组件
具有自动查找与自动历史记录功能的 comboboxes 与编辑控制组件
整数、浮点数、时间、日期、spinedit 控制组件
按键控制组件
单选框控制组件
使用另一个自定义编辑器
扩展/缩短结点控制组件
富文本就地编辑
将图片添加到你的单元格里
控制顶端、底部、右方、左方图片的定位
添加图标、位图、图片列表选项与数据独立的图片
轮显文本
多图片单元格
TPicture, TFilePicture 对象对所有图解格式的开放式支持
进度条
按键
注释指示程序
单公式编辑界面
自动重算
单个单元格重算、全部重算
具有多方面数学功能
带公式保存
在公式里的单个单元格参考
单元格范围公式
在单元格基础上表格的公式精确度
显示公式或公式结果
日期/时间功能
智能公式的复制与粘贴
能通过自定义功能扩展
单元格命名
单元格命名模式可以是 RxCy 风格或 A1 风格
公式与恒量库
支持鼠标滚轮操作
将用户调整的列宽度保存到.INI 文件或注册表
自动为列编号
可删除、清楚、移动、隐藏列与行的所有类型的操作
用鼠标进行全行与全列的选择

在设计期间设置列标题与固定行

分离行选项

平整与均衡的滚动条

在表格和表格与 Excel、Word 之间的 OLE 拖放

分组与反分组功能以及分组分类功能

多列过滤

6. EhLib

官方网站：http://www.ehlib.com

著名的数据库连接控件，DBgrid 增强 VCL 控件；支持多表头，多固定列，按表头排序，支持合计列，并支持直接打印。可以和 PB 的 ataWindow 媲美。

7. ProfGrid

主页：http://www.profgrid.com/grid.html

一个非常不错的 Grid 控件，可以增加 Checkbox 、button、combobox、edit、datetimepicker 等等；具有合并单元格、导出到 Excel、在单元格中使用公式等功能。

8. EasyGrid

一款不错中国式表格组件，含源代码。

TStringAlignGrid

TStringAlignGrid 是一个增强的 StringGrid 控件，提供了单元格对齐、背景色设置等功能。

五、 delphi 中配置文件的使用（*.ini）

.ini 文件是基于文本类型的格式文件，用于存储程序初始化和配置数据。

.ini 文件是有段（Sections）和键（key）组成的，每个文件可以有 n 个段(每个段有方括号括起来)，每个段可以有 m 个键，大致格式如下：

```
[secontion]
 KeyName1=Value1;
 ;comment
 KeyName2=Value2;
```

段名和键名在使用中是不区分大小写得，但是名字中不能有空格。

键可以存储 integer，string，float，boolean，datatime 等数据类型。

在 Delphi 中有一个 TIniFile 的类用于访问.ini 文件，该类定义在 IniFiles.pas 文件中，具体使用如下

```
view plaincopy to clipboardprint?
//定义对象
var
  iniFile:TiniFile;
//创建对象
  iniFile：=TiniFile.Create(iniFilePath+iniFileName）；
//读数据
  iniFile.ReadString('Section','Key','DefaultValue')  //字符串
iniFile.ReadInteger('Section','Key',DefaultValue) ;//整数

//写数据
iniFile.WriteString('Section','Key','Value')  //字符串
iniFile.WriteInteger('Section','Key',tValue) ;

// 释放对象
iniFile.Free;
```

如果想读取整段值，可以用 iniFile.ReadSection('SectionName', StringList)将整段数据读到 TStringList 对象中。

六、 窗口动画效果 Animatewindow 应用

今天去书店看书，发现了这个技巧，比起我以前用代码实现方便多了；

该动画效果就是用 windows api 函数 :function AnimateWindow(hWnd: HWND; dwTime: DWORD; dwFlags: DWORD): BOOL; stdcall 来实现；

参数说明：hWnd,窗体（Form）的句柄；

dwTime,动画效果用时，单位毫秒 mm;

dwFlags,动画效果类型，可用参数有：

```
 1.AW_HOR_POSITIVE = $00000001; // 从左向右开屏
2.AW_HOR_NEGATIVE = $00000002; // 从右向左开屏
3.AW_VER_POSITIVE = $00000004; // 从上向下开屏
 4.AW_VER_NEGATIVE = $00000008; // 从下向上开屏
 5,AW_CENTER = $00000010; // 从中心向四周扩展,在关闭动画中则为从四周向中心收缩
 6.AW_HIDE = $00010000; // 关闭时候与前面的定义组合使用,如 AW_HIDE or AW_CENTER
  7.AW_ACTIVATE = $00020000; // 与 1-5 组合,开屏使用
```

8.AW_SLIDE = $00040000; // 与 1-5 6/7 组合,产生滑行效果

9.AW_BLEND = $00080000; // Win2000 下使用,淡入淡出效果

七、 Delphi Excel to Sql Server

```
 unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Grids, DBGrids, DB, ADODB,comobj, OleServer,
  ExcelXP;
type
  TForm1 = class(TForm)
    ADOConn: TADOConnection;
    atblXlsToSql: TADOTable;
    DBGrid1: TDBGrid;
    Panel1: TPanel;
    Button1: TButton;
    opdExcel: TOpenDialog;
    DataSource1: TDataSource;
    Excel: TExcelApplication;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
    function ConnectDB:boolean;              //连接数据库
    function  OpenTable(const Tablename:String):boolean;   //打开指定数据表
    function  XlsToSqlTable(const xlsFileName,TableName:String):boolean;  //将
Excel 数据导入 Sql Server
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  serverIP,dbName,userid,passw:string;  //服务器 IP，数据库名称，用户名，密码
implementation
{$R *.dfm}
function TForm1.ConnectDB:boolean;
```

```
begin
  Result:=false;
  serverIP:='192.168.0.0';
  dbName:='OnLine';
  userid:='sa';
  passw:='sa';
  if not adoconn.Connected then
  begin
    adoconn.ConnectionString:='Provider=SQLOLEDB.1;Password=' + passw
                            + ';Persist Security Info=True;User ID=' + userid
                            + ';Initial Catalog=' + dbName
                            + ';Data Source=' + ServerIP;
    adoconn.LoginPrompt:=false;
    adoconn.Open();
  end;
  Result:=True;
end;
function TForm1.OpenTable(const Tablename:String):boolean;
begin
  Result:=false;
  if not adoconn.Connected then
    if not ConnectDB then
    begin
      showmessage('连接数据库失败');
      exit;
    end;
   atblXlstoSql.Connection:=adoconn;
   atblXlstoSql.TableName:=Tablename;
   try
     atblXlstoSql.Open;
   except
     ShowMessage('表名不正确或该表不存在! ');
     exit;
   end;
   Result:=True;
end;
function TForm1.XlsToSqlTable(const xlsFileName,TableName:string):boolean;
```

```
var
  excelApp,workbook,sheet:variant;
  iRow:integer;
begin
  Result:=False;
  //连接数据库，打开要导入数据的表
  if not OpenTable(TableName) then
    exit;
  // 建立 excel 应用
  try
  begin
    excelApp:=CreateOleObject( 'Excel.Application' );
    workbook:=excelApp.workbooks.open(xlsFileName);
    sheet:=workbook.sheets[1];
  end;
  except
    showmessage('open excel file fail');
    exit;
  end;
  try
    adoconn.BeginTrans;        //事务处理
    for iRow:=2 to sheet.UsedRange.Rows.Count do        //iRow:=2 跳过标题
行,sheet.UsedRange.Rows.Conut 文件的行数
    begin
      atblXlstoSql.Append;
      atblXlstoSql.FieldByName('Name').AsString:=sheet.Cells[iRow,1];
      atblXlstoSql.FieldByName('Number').AsString:=sheet.Cells[iRow,3];
      atblXlstoSql.FieldByName('Remark').AsString:=sheet.Cells[iRow,4];
    end;
    atblXlstoSql.Post;
    adoconn.CommitTrans;
    except
      adoconn.RollbackTrans;
    end;
 //   atblXlstoSql.Close;
    excelAPP.Quit;
    Result:=true;
```

```
end;
procedure TForm1.Button1Click(Sender: TObject);
var
  FileName:string;
begin
  if opdExcel.Execute then
     FileName:=opdExcel.FileName
  else
     exit;
  if MessageBox(handle,'确定将 Excel 数据导入数据库吗？','提示
',MB_OKCANCEL+MB_ICONQUESTION)=IDOK then
     if xlsToSqlTable(FileName,'bc_member') then
        MessageBox(handle,'导入完成！','提示',MB_OK+MB_ICONASTERISK)
     else
        MessageBox(handle,'导入失败！','提示',MB_OK+MB_ICONERROR)
end;
end.
```

八、 Delphi 控制 Excel 的经验如下：

Delphi 5 控制 Excel

作者：吴晓勇，孙唏瑜
时间：2001 年 11 月 20 日

(一) 使用动态创建的方法

首先创建 Excel 对象，使用 ComObj:
var ExcelApp: Variant;
ExcelApp := CreateOleObject('Excel.Application ');

1) 显示当前窗口：
ExcelApp.Visible := True;

2) 更改 Excel 标题栏：
ExcelApp.Caption := '应用程序调用 Microsoft Excel ';

3) 添加新工作簿：
ExcelApp.WorkBooks.Add;

4) 打开已存在的工作簿：
ExcelApp.WorkBooks.Open('C:\Excel\Demo.xls ');

5) 设置第 2 个工作表为活动工作表：
ExcelApp.WorkSheets[2].Activate;
或
ExcelApp.WorksSheets['Sheet2 '].Activate;

6) 给单元格赋值：
ExcelApp.Cells[1,4].Value := '第一行第四列 ';

7) 设置指定列的宽度（单位：字符个数），以第一列为例：
ExcelApp.ActiveSheet.Columns[1].ColumnsWidth := 5;

8) 设置指定行的高度（单位：磅）（1 磅＝0.035 厘米），以第二行为例：
ExcelApp.ActiveSheet.Rows[2].RowHeight := 1/0.035; // 1 厘米

9) 在第 8 行之前插入分页符：
ExcelApp.WorkSheets[1].Rows[8].PageBreak := 1;

10) 在第 8 列之前删除分页符：
ExcelApp.ActiveSheet.Columns[4].PageBreak := 0;

11) 指定边框线宽度：
ExcelApp.ActiveSheet.Range['B3:D4 '].Borders[2].Weight := 3;
1-左 2-右 3-顶 4-底 5-斜(\ ) 6-斜(/)

12) 清除第一行第四列单元格公式：
ExcelApp.ActiveSheet.Cells[1,4].ClearContents;

13) 设置第一行字体属性：
ExcelApp.ActiveSheet.Rows[1].Font.Name := '隶书 ';
ExcelApp.ActiveSheet.Rows[1].Font.Color := clBlue;
ExcelApp.ActiveSheet.Rows[1].Font.Bold := True;
ExcelApp.ActiveSheet.Rows[1].Font.UnderLine := True;

14) 进行页面设置：

a.页眉：
ExcelApp.ActiveSheet.PageSetup.CenterHeader := '报表演示 ';

b.页脚：

ExcelApp.ActiveSheet.PageSetup.CenterFooter := '第&P 页 ';

c.页眉到顶端边距 2cm：

ExcelApp.ActiveSheet.PageSetup.HeaderMargin := 2/0.035;

d.页脚到底端边距 3cm：

ExcelApp.ActiveSheet.PageSetup.HeaderMargin := 3/0.035;

e.顶边距 2cm：

ExcelApp.ActiveSheet.PageSetup.TopMargin := 2/0.035;

f.底边距 2cm：

ExcelApp.ActiveSheet.PageSetup.BottomMargin := 2/0.035;

g.左边距 2cm：

ExcelApp.ActiveSheet.PageSetup.LeftMargin := 2/0.035;

h.右边距 2cm：

ExcelApp.ActiveSheet.PageSetup.RightMargin := 2/0.035;

i.页面水平居中：

ExcelApp.ActiveSheet.PageSetup.CenterHorizontally := 2/0.035;

j.页面垂直居中：

ExcelApp.ActiveSheet.PageSetup.CenterVertically := 2/0.035;

k.打印单元格网线：

ExcelApp.ActiveSheet.PageSetup.PrintGridLines := True;

15) 拷贝操作：

a.拷贝整个工作表：

ExcelApp.ActiveSheet.Used.Range.Copy;

b.拷贝指定区域：

ExcelApp.ActiveSheet.Range['A1:E2 '].Copy;

c.从 A1 位置开始粘贴：

ExcelApp.ActiveSheet.Range.['A1 '].PasteSpecial;

d.从文件尾部开始粘贴：

ExcelApp.ActiveSheet.Range.PasteSpecial;

16) 插入一行或一列：

a. ExcelApp.ActiveSheet.Rows[2].Insert;

b. ExcelApp.ActiveSheet.Columns[1].Insert;

17) 删除一行或一列：

a. ExcelApp.ActiveSheet.Rows[2].Delete;

b. ExcelApp.ActiveSheet.Columns[1].Delete;

18) 打印预览工作表：

ExcelApp.ActiveSheet.PrintPreview;

19) 打印输出工作表：

ExcelApp.ActiveSheet.PrintOut;

20) 工作表保存：
if not ExcelApp.ActiveWorkBook.Saved then
ExcelApp.ActiveSheet.PrintPreview;

21) 工作表另存为：
ExcelApp.SaveAs('C:\Excel\Demo1.xls ');

22) 放弃存盘：
ExcelApp.ActiveWorkBook.Saved := True;

23) 关闭工作簿：
ExcelApp.WorkBooks.Close;

24) 退出 Excel：
ExcelApp.Quit;

(二) 使用 Delphi 控件方法
在 Form 中分别放入 ExcelApplication, ExcelWorkbook 和 ExcelWorksheet。

1） 打开 Excel
ExcelApplication1.Connect;

2) 显示当前窗口：
ExcelApplication1.Visible[0]:=True;

3) 更改 Excel 标题栏：
ExcelApplication1.Caption := '应用程序调用 Microsoft Excel ';

4) 添加新工作簿：
ExcelWorkbook1.ConnectTo(ExcelApplication1.Workbooks.Add(EmptyParam,0));

5) 添加新工作表：
var Temp_Worksheet: _WorkSheet;
begin
Temp_Worksheet:=ExcelWorkbook1.
WorkSheets.Add(EmptyParam,EmptyParam,EmptyParam,EmptyParam,0) as _WorkSheet;
ExcelWorkSheet1.ConnectTo(Temp_WorkSheet);
End;

6) 打开已存在的工作簿：
ExcelApplication1.Workbooks.Open (c:\a.xls
EmptyParam,EmptyParam,EmptyParam,EmptyParam,

EmptyParam,EmptyParam,EmptyParam,EmptyParam,
EmptyParam,EmptyParam,EmptyParam,EmptyParam,0)

7) 设置第 2 个工作表为活动工作表：
ExcelApplication1.WorkSheets[2].Activate; 或
ExcelApplication1.WorksSheets['Sheet2 '].Activate;

8) 给单元格赋值：
ExcelApplication1.Cells[1,4].Value := '第一行第四列 ';

9) 设置指定列的宽度（单位：字符个数），以第一列为例：
ExcelApplication1.ActiveSheet.Columns[1].ColumnsWidth := 5;

10) 设置指定行的高度（单位：磅）（1 磅＝0.035 厘米），以第二行为例：
ExcelApplication1.ActiveSheet.Rows[2].RowHeight := 1/0.035; // 1 厘米

11) 在第 8 行之前插入分页符：
ExcelApplication1.WorkSheets[1].Rows[8].PageBreak := 1;

12) 在第 8 列之前删除分页符：
ExcelApplication1.ActiveSheet.Columns[4].PageBreak := 0;

13) 指定边框线宽度：
ExcelApplication1.ActiveSheet.Range['B3:D4 '].Borders[2].Weight := 3;
1-左 2-右 3-顶 4-底 5-斜(\ ) 6-斜(/)

14) 清除第一行第四列单元格公式：
ExcelApplication1.ActiveSheet.Cells[1,4].ClearContents;

15) 设置第一行字体属性：
ExcelApplication1.ActiveSheet.Rows[1].Font.Name := '隶书 ';
ExcelApplication1.ActiveSheet.Rows[1].Font.Color := clBlue;
ExcelApplication1.ActiveSheet.Rows[1].Font.Bold := True;
ExcelApplication1.ActiveSheet.Rows[1].Font.UnderLine := True;

16) 进行页面设置：
a.页眉：
ExcelApplication1.ActiveSheet.PageSetup.CenterHeader := '报表演示 ';
b.页脚：
ExcelApplication1.ActiveSheet.PageSetup.CenterFooter := '第&P 页 ';
c.页眉到顶端边距 2cm：
ExcelApplication1.ActiveSheet.PageSetup.HeaderMargin := 2/0.035;
d.页脚到底端边距 3cm：
ExcelApplication1.ActiveSheet.PageSetup.HeaderMargin := 3/0.035;

e.顶边距 2cm：

ExcelApplication1.ActiveSheet.PageSetup.TopMargin := 2/0.035;

f.底边距 2cm：

ExcelApplication1.ActiveSheet.PageSetup.BottomMargin := 2/0.035;

g.左边距 2cm：

ExcelApplication1.ActiveSheet.PageSetup.LeftMargin := 2/0.035;

h.右边距 2cm：

ExcelApplication1.ActiveSheet.PageSetup.RightMargin := 2/0.035;

i.页面水平居中：

ExcelApplication1.ActiveSheet.PageSetup.CenterHorizontally := 2/0.035;

j.页面垂直居中：

ExcelApplication1.ActiveSheet.PageSetup.CenterVertically := 2/0.035;

k.打印单元格网线：

ExcelApplication1.ActiveSheet.PageSetup.PrintGridLines := True;

17) 拷贝操作：

a.拷贝整个工作表：

ExcelApplication1.ActiveSheet.Used.Range.Copy;

b.拷贝指定区域：

ExcelApplication1.ActiveSheet.Range['A1:E2 '].Copy;

c.从 A1 位置开始粘贴：

ExcelApplication1.ActiveSheet.Range.['A1 '].PasteSpecial;

d.从文件尾部开始粘贴：

ExcelApplication1.ActiveSheet.Range.PasteSpecial;

18) 插入一行或一列：

a. ExcelApplication1.ActiveSheet.Rows[2].Insert;

b. ExcelApplication1.ActiveSheet.Columns[1].Insert;

19) 删除一行或一列：

a. ExcelApplication1.ActiveSheet.Rows[2].Delete;

b. ExcelApplication1.ActiveSheet.Columns[1].Delete;

20) 打印预览工作表：

ExcelApplication1.ActiveSheet.PrintPreview;

21) 打印输出工作表：

ExcelApplication1.ActiveSheet.PrintOut;

22) 工作表保存：

```
if not ExcelApplication1.ActiveWorkBook.Saved then
    ExcelApplication1.ActiveSheet.PrintPreview;
```

23) 工作表另存为：

```
ExcelApplication1.SaveAs( 'C:\Excel\Demo1.xls ' );
```

24) 放弃存盘：

```
ExcelApplication1.ActiveWorkBook.Saved := True;
```

25) 关闭工作簿：

```
ExcelApplication1.WorkBooks.Close;
```

26) 退出 Excel：

```
ExcelApplication1.Quit;
ExcelApplication1.Disconnect;
```

三) 使用 Delphi 控制 Excle 二维图

在 Form 中分别放入 ExcelApplication, ExcelWorkbook 和 ExcelWorksheet

```
var asheet1,achart, range:variant;
```

1）选择当第一个工作薄第一个工作表

```
asheet1:=ExcelApplication1.Workbooks[1].Worksheets[1];
```

2）增加一个二维图

```
achart:=asheet1.chartobjects.add(100,100,200,200);
```

3）选择二维图的形态

```
achart.chart.charttype:=4;
```

4）给二维图赋值

```
series:=achart.chart.seriescollection;
range:=sheet1!r2c3:r3c9;
series.add(range,true);
```

5）加上二维图的标题

```
achart.Chart.HasTitle:=True;
achart.Chart.ChartTitle.Characters.Text:=’ Excle 二维图’
```

6）改变二维图的标题字体大小

```
achart.Chart.ChartTitle.Font.size:=6;
```

7）给二维图加下标说明

```
achart.Chart.Axes(xlCategory, xlPrimary).HasTitle := True;
achart.Chart.Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text := '下标说明 ';
```

8）给二维图加左标说明

```
achart.Chart.Axes(xlValue,  xlPrimary).HasTitle  :=  True;
achart.Chart.Axes(xlValue,  xlPrimary).AxisTitle.Characters.Text  :=  '左标说明 ';
```

9）给二维图加右标说明

```
achart.Chart.Axes(xlValue,  xlSecondary).HasTitle  :=  True;
achart.Chart.Axes(xlValue,  xlSecondary).AxisTitle.Characters.Text  :=  '右标说明 ';
```

10）改变二维图的显示区大小

```
achart.Chart.PlotArea.Left  :=  5;
achart.Chart.PlotArea.Width  :=  223;
achart.Chart.PlotArea.Height  :=  108;
```

11）给二维图坐标轴加上说明

```
achart.chart.seriescollection[1].NAME:= '坐标轴说明 ';
```

用 DELPHI 实现把 WORD、EXCEL 和图片等存储到数据库中

用 **image** 字段保存这些文档。

```
var
   word_stream:  TMemoryStream;
   filename:  string;
begin
   if  odgDoc.Execute  then//odgDoc:OpenDialog
   begin
      filename  :=  ExtractFileName(odgDoc.FileName);
      word_stream  :=  TMemoryStream.Create;
      word_stream.LoadFromFile(odgDoc.FileName);
      word_stream.Position  :=  0;
      cdsPACT.Append
      cdsPACT.FieldByName('DocName').Value  :=  filename;
      TBlobField(cdsPACT.FieldByName('PactText')).LoadFromStream(word_stre
am);
      cdsPACT.Post;
      word_stream.Free;
   end;
end;
```

---- 目前，**Delphi** 被越来越多的人选中作为 **MIS** 系统开发中的前台工具。在以 **Delphi** 为前台，一些大型数据库为后台的 **MIS** 系统中，图形的处理不可避 免；即从以 **Delphi** 开发的前台界面输入图形，并保存到相应的数据库字段中。在这种形式的图形处理中，**BMP** 文件的处理比较简单，因为 **Delphi** 本身 有 **Image** 和 **DBImage** 构件，用这些构件与数据库中可以保存图形的大型字段 **BLOB** 比较容易地进行数据交换。以这种方式进行图形处理已应用在许多 **MIS** 软件中，包括处理人员照片的人事档案系统等。

---- 但是，**BMP** 文件一般都比较大。而且有时要录入的是自己在计算机上画的简图，并伴

随大量文字说明。这种情况用 **Win95** 中的画图板等处理 **BMP** 文件的工具 处理就比较困难。一般应用人员都喜欢用 **WORD** 画图和写说明文字，然后保存到数据库中。

---- 经过一段时间的摸索，我们解决了这个问题，并经过完善，在应用中运行较好。程序如下：

```
procedure TsampleForm.OpenDOCClick(Sender: TObject);
var
   MemSize: Integer;
   Buffer: PChar;
   Myfile: TFileStream;
   Stream: TBlobStream;
begin
   OpenDialog1.Filter:='WORD 文档(*.DOC)|*.DOC';{从对话窗选择文件}
   if OpenDialog1.Execute then
   begin
      Myfile:=TFileStream.Create(OpenDialog1.FileName,fmOpenRead);
      with table1 do      {`table1′为含 BLOB 字段的表名}
      begin
         Open;
         Edit;
         Stream := TBlobStream.Create(FieldByName('Doc') as TBlobFiel
d, bmWrite);{`Doc′为 BLOB 字段名}
         MemSize := MyFile.Size;
         Inc(MemSize); {Make room for the buffer's null terminator
.}
         Buffer := AllocMem(MemSize);          {Allocate the memory.
}
         try
            Stream.Seek(0, soFromBeginning);   {Seek 0 bytes from t
he stream's end point}
            MyFile.Read(Buffer^,MemSize);
            Stream.Write(Buffer^,MemSize);
         finally
            MyFile.Free;
            Stream.Free;
         end;
               try
                 Post;
               except
                 on E: EDatabaseError do
             if HandelException(E)< >0 then
                                 exit
            else
            raise;
               end;
```

```
        end;
        Doc_ole.CreateObjectFromFile(OpenDialog1.FileName,False);
        Doc_ole.Run;{Doc_ole 为 ToleContainer 构件名}
      end;
  end;
```

---- 以上为向数据库中写入的程序，应用中从对话窗取出文件在 **ToleContainer** 构件中显示的同时存入数据库。

```
  procedure TsampleForm.GetDocClick(Sender: TObject);
  var
     MemSize: Integer;
     Buffer: PChar;
     Myfile: TFileStream;
     Stream: TBlobStream;
  begin
        Myfile:=TFileStream.Create('c:\temp.tmp',fmCreate);
        with Query1 do
        begin
           Stream := TBlobStream.Create(FieldByName('Doc') as TBlobFiel
d, bmRead);
           MemSize := Stream.Size;
           Inc(MemSize); {Make room for the buffer's null terminator.
}
           Buffer := AllocMem(MemSize);        {Allocate the memory.}
           try
              Stream.Read(Buffer^,MemSize);
              MyFile.Write(Buffer^,MemSize);
           finally
              MyFile.Free;
              Stream.Free;
           end;
        end;
           if FileExists('c:\temp.DOC') then
     DeleteFile('c:\temp.DOC');
           if FileExists('c:\temp.tmp') then
           begin
              RenameFile('c:\temp.tmp', 'c:\temp.DOC');
              Doc_ole.CreateObjectFromFile('c:\temp.DOC',False);
              Doc_ole.Run;
           end;
  end;
```

---- 以上程序为从数据库从将 **WORD** 文档取出，并放在 **temp.doc** 的临时文件上并在 **ToleContainer** 构件中显示。

---- 在程序的其他部份应准确控制表记录指针，使 **WORD** 文档的存取发生在正确的记录位置。

两个 form 之间传值

利用以下方法：

在 **form1** 里面编写函数**:**

```
function ExecuteForm1(var S : string) : word;
var
   Form1 : TForm1;
begin

   Form1 := TForm1.createform(application);
   result := form1.showmodual;
   try
   if result = mrOK then
   begin
      S := 'OK';
   end;
   finally
   form1.free;
   end;

end;
```

在 **form1** 中添加按钮 **button**，在 **click** 事件里写代码：

```
procedure Button1Click(sender:TObject);
begin
   ModalResult := mrOk;
end;

//---------------------------------------------------------------
```

在 **form2** 里面调用**(**首先必须 **uses form1** 单元**)**，例如在 **Button1** 的 **click** 事件里

```
procedure Button1Click(sender:TObject);
var
   S:string;
begin
   if form1.ExecuteForm1(S) = mrOK then
   begin
   showmessage(S);
   end;
end;
```

分类**: delphi**

设置 DBGridEH 自适应列宽的最好方法

一直在找最好的根据 **DBGridEH(**或者 **DBGrid)**的内容和标题栏设置自适应列宽的方法，一直没有太好的。今天从园地上发现了源码：地址如下，非常好用。与大家分享：

http://www.delphifans.com/SoftView/SoftView_2019.html

代码哪下：

```
//需要定义这个类，才能使用 OptimizeSelectedColsWidth 方法调整列宽
type
  TZYDBGridEH = class(TCustomDBGridEh)
  public
  end;
```

我在原来的基础上进行修改的实现代码：

```
//表格内容和字段标题宽度设置列宽
procedure SetDBGridColumnsWidth(Grid: TCustomDBGridEh);
var
  i: integer;
begin
  if not Assigned(Grid.DataSource) then Exit;
  if not Grid.DataSource.Dataset.Active then Exit;
  try
    Grid.DataSource.Dataset.DisableControls;
    for i := 0 to TDBGridEh(Grid).Columns.Count - 1 do
    begin
      TZYDBGridEh(Grid).OptimizeSelectedColsWidth(TDBGridEh(Grid).Columns[i]);
    end;
    Grid.DataSource.Dataset.EnableControls;
  except on E: Exception do
    begin
      Error('设置表格' + Grid.Name + '出错！' + E.Message);
    end;
  end;
end;
```

另外，还有根据列的所有行内容进行设置，当数据量大时，速度就会很慢，还有就是有时候设置会出错（有用同样的办法设置窗体中的多个 **DBGridEH**，就有一个总是出错）。但也贴出来，供大家参考：

```
procedure SetDBGridColumnsWidth(DBGridName:TDBGridEH);
var
  j, maxWidth: integer;
```

```
begin
  if not Assigned(DBGridName.DataSource) then Exit;
  if not DBGridName.DataSource.Dataset.Active then Exit;
  try
    DBGridName.DataSource.Dataset.DisableControls;
    for j := 0 to DBGridName.Columns.Count - 1 do
    begin
      DBGridName.DataSource.Dataset.First;
      maxWidth :=
DBGridName.Canvas.TextWidth(Trim(DBGridName.Columns[j].Title.Caption));
      while not DBGridName.DataSource.Dataset.Eof do
      begin
        if maxWidth <
DBGridName.Canvas.TextWidth(Trim(DBGridName.DataSource.Dataset.FieldByNa
me(DBGridName.Columns[j].FieldName).AsString)) then
          maxWidth :=
DBGridName.Canvas.TextWidth(Trim(DBGridName.DataSource.Dataset.FieldByNa
me(DBGridName.Columns[j].FieldName).AsString))
        else
        DBGridName.DataSource.Dataset.Next;
      end;
        DBGridName.Columns[j].Width := maxWidth + 20;
    end;
    DBGridName.DataSource.DataSet.First;
    DBGridName.DataSource.Dataset.EnableControls;
  except
    Error('设置表格' + DBGridName.Name + '出错！');
  end;
end;
```

另外，设置 **DBGrid** 的方法，用上面的方法就一不太好用了。网上的方法**(**来自大富翁论坛**)**如下：

```
procedure TForm1.DBGrid1DrawColumnCell(Sender: TObject; const Rect: TRect;
DataCol: Integer; Column: TColumn; State: TGridDrawState);
var
nWidth: Integer;
begin
with DBGrid1.Canvas do begin
nWidth := TextWidth(Column.Field.AsString) + 2;
if nWidth > Column.Width then Column.Width := nWidth;
end;
end;
```

dbgrideh 的功能

a.点标题排序：

1.eh 的 **optionsEh** 的 **dghAutoSortMarking True**

2 eh 的 **sortlocal true**

3.列的 **title** 的 **toolbutton** 为 **true**

4。eh 的 **optionsEh** 的 **dghMultiSortMarking True** 多个字段一起排**(**按 **CTRL+Mouse)**

5.uses 加相应的驱动 **ehlibado/ehlibbde**

b.模糊过滤:

```
uses ehlibado;

 if Assigned(DataSource) and Assigned(DataSource.DataSet) then
   DataSource.DataSet.Filtered :=true;
 STFilter.Local :=True;
 STFilter.Visible :=True;
//加过滤下拉列表

 for i:=0 to Columns.Count-1 do
   Columns[i].STFilter.ListSource :=DataSource;

 //加排序列表菜单

 列的 dropdownSizing
```

c.斑马线

```
 if Assigned(DataSource) and Assigned(DataSource.DataSet) then
 begin
   if DataSource.DataSet.IsSequenced then
   begin
     OddRowColor :=clRed;
     EvenRowColor:=clYellow;
   end
   else begin
      aDBGridEH.OnDrawColumnCel//处理

     end;
   end;
 end;
procedure DBGridEhDrawColumnCell(Self: TObject;Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumnEh;
```

```
  State: TGridDrawState);
begin
  if (gdSelected in State) then
  begin
  end
  else  if (Rect.Top = (Sender as TDBGridEh).CellRect((Sender as
TDBGridEh).Col,(Sender as TDBGridEh).Row).Top) and (not (gdFocused in State)
or not (Sender as TDBGridEh).Focused) then
  begin
    (Sender as TDBGridEh).Canvas.Brush.Color := clblack;//选择行
    (Sender as TDBGridEh).Canvas.Font.Color := clWhite;
  end
  else
  begin
    (Sender as TDBGridEh).Canvas.Brush.Color := clYellow;//奇行
    (Sender as TDBGridEh).Canvas.Font.Color := clWindowText;
  end;
  if (not (gdSelected in State)) and ((((Sender as
TDBGridEh).DataSource.DataSet.RecNo) mod 2) =0) then
  begin
    (Sender as TDBGridEh).Canvas.Brush.Color := clred;//偶行
    (Sender as TDBGridEh).Canvas.Font.Color := clWindowText;
  end;
//   (Sender as TDBGridEh).Canvas.FillRect(Rect);
  (Sender as TDBGridEh).DefaultDrawColumnCell(Rect,DataCol,Column,State);
end;
```

d.统计栏

e.保存 grid 位置

```
   SaveGridLayoutIni(IniFileName,GetFullName(TWinControl(Sender)),true);

    RestoreGridLayoutIni(IniFileName,vSectionName,[grpColIndexEh,
grpColWidthsEh, grpSortMarkerEh,
grpColVisibleEh,grpDropDownRowsEh,grpDropDownWidthEh]);
```

分类: delphi

dbgrid、dbgrideh 专题总结

一、打印 dbgrid 数据：

```
//=============================================================
==============================
//打印 DBGrid 中的所有数据
*************************************************************
//=============================================================
==============================
```

```
procedure DBGridPrint(DBGrid: TDBGrid; Title: string);
var PrintDialog: TPrintDialog;
RowHeight, Temp_X, Temp_Y, PageEdgeX, PageEdgeY, PixelsPerInchX,
PixelsPerInchY: integer;
TempStr: string;
Scale: Double;
Rect: TRect;
//=================================================================
==============================
//1.输出标题
*****************************************************************
****
//=================================================================
==============================
procedure Print_Title;
begin
  Rect := Bounds(0,0,Printer.PageWidth,PageEdgeY);
  Printer.Canvas.Font.Name := '楷体_GB2312';
  Printer.Canvas.Font.Style := Printer.Canvas.Font.Style + [fsBold];
  Printer.Canvas.Font.Size := 20;
  DrawText(Printer.Canvas.Handle,PChar(Title),Length(Title),Rect,DT_CENTER or
DT_VCENTER or DT_SINGLELINE);
  Printer.Canvas.Rectangle(PageEdgeX, PageEdgeY, Printer.PageWidth -
PageEdgeX, PageEdgeY + 1);
end;




//=================================================================
==============================
//2.输出列头
*****************************************************************
****
//=================================================================
==============================
procedure Print_Column;
var j: integer;
begin
  Printer.Canvas.Font.Name := '黑体';
  Printer.Canvas.Font.Size := 9;
  Temp_X := PageEdgeX;
  Temp_Y := PageEdgeY;
```

```
  //================================================
==========================
  for j:=1 to DBGrid.Columns.Count do
  begin
    if not DBGrid.Columns[j-1].Visible then Continue;
    //================================================
==========================
    TempStr := DBGrid.Columns[j-1].Title.Caption;
    Rect := Bounds(Temp_X, Temp_Y, Trunc(DBGrid.Columns[j-1].Width*Scale),
RowHeight);
    case DBGrid.Columns[j-1].Field.Alignment of
      //================================================
======================
      //case.1.居中
**********************************************************
      //================================================
======================
      taCenter:
DrawText(Printer.Canvas.Handle,PChar(TempStr),Length(TempStr),Rect,DT_CEN
TER or DT_VCENTER or DT_SINGLELINE);
      //================================================
======================
      //case.2.居左
**********************************************************
      //================================================
======================
      taLeftJustify:
DrawText(Printer.Canvas.Handle,PChar(TempStr),Length(TempStr),Rect,DT_LEF
T or DT_VCENTER or DT_SINGLELINE);
      //================================================
======================
      //case.3.居右
**********************************************************
      //================================================
======================
      taRightJustify: if Rect.Right-Rect.Left>=Printer.Canvas.TextWidth(TempStr)
                      then
DrawText(Printer.Canvas.Handle,PChar(TempStr),Length(TempStr),Rect,DT_RIG
HT or DT_VCENTER or DT_SINGLELINE)
                      else
DrawText(Printer.Canvas.Handle,PChar(TempStr),Length(TempStr),Rect,DT_LEF
T or DT_VCENTER or DT_SINGLELINE);
    end;
    Temp_X := Temp_X + Trunc(DBGrid.Columns[j-1].Width*Scale);
```

```
  end;
  Temp_Y := Temp_Y + RowHeight;
end;
//=================================================
==============================
//3.输出 DBGrid 内容
*****************************************************************
//=================================================
==============================
procedure Print_Cells;
var j: integer;
begin
  Printer.Canvas.Font.Name := '宋体';
  Printer.Canvas.Font.Style := Printer.Canvas.Font.Style - [fsBold];
  while Temp_Y<Printer.PageHeight-PageEdgeY do
  begin
    Temp_X := PageEdgeX;
    for j:=1 to DBGrid.Columns.Count do
    begin
      if not DBGrid.Columns[j-1].Visible then Continue;
      //=================================================
======================
      Rect := Bounds(Temp_X, Temp_Y, Trunc(DBGrid.Columns[j-1].Width*Scale),
RowHeight);
      if (DBGrid.Columns[j-1].Field is TCurrencyField)
      or (DBGrid.Columns[j-1].Field is TLargeIntField)
      or (DBGrid.Columns[j-1].Field is TSmallIntField)
      or (DBGrid.Columns[j-1].Field is TIntegerField)
      or (DBGrid.Columns[j-1].Field is TFloatField)
      or (DBGrid.Columns[j-1].Field is TWordField)
      then TempStr := FormatFloat(',###0.00',DBGrid.Columns[j-1].Field.AsFloat)
      else TempStr := DBGrid.Columns[j-1].Field.AsString;
      //=================================================
======================
      case DBGrid.Columns[j-1].Field.Alignment of
        //=================================================
====================
        //case.1.居中
*****************************************************
        //=================================================
====================
        taCenter:
DrawText(Printer.Canvas.Handle,PChar(TempStr),Length(TempStr),Rect,DT_CEN
TER or DT_VCENTER or DT_SINGLELINE);
```

```
      //=============================================
=====================
      //case.2.居左
*****************************************************
      //=============================================
=====================
      taLeftJustify:
DrawText(Printer.Canvas.Handle,PChar(TempStr),Length(TempStr),Rect,DT_LEF
T or DT_VCENTER or DT_SINGLELINE);
      //=============================================
=====================
      //case.3.居右
*****************************************************
      //=============================================
=====================
      taRightJustify:
DrawText(Printer.Canvas.Handle,PChar(TempStr),Length(TempStr),Rect,DT_RIG
HT or DT_VCENTER or DT_SINGLELINE);
    end;
    Temp_X := Temp_X + Trunc(DBGrid.Columns[j-1].Width*Scale);
   end;
   Temp_Y := Temp_Y + RowHeight;
   DBGrid.DataSource.DataSet.Next;
   if DBGrid.DataSource.DataSet.Eof then Exit;
 end;
end;
//=============================================
=============================
//4.输出页脚
*************************************************************
****
//=============================================
=============================
procedure Print_Footer;
begin
 Temp_Y := Printer.PageHeight - PageEdgeY + RowHeight;
 //=============================================
=========================
 //4.0.输出横线
*********************************************************
 //=============================================
=========================
 Printer.Canvas.Rectangle(PageEdgeX, Temp_Y, Printer.PageWidth - PageEdgeX,
Temp_Y + 1);
```

```
  //================================================
=========================
  //4.1.输出日期
*************************************************************
  //================================================
=========================
  Rect := Bounds(PageEdgeX, Temp_Y, Printer.PageWidth-PageEdgeX,
RowHeight);
  DrawText(Printer.Canvas.Handle,PChar(DateTimeToStr(Now)),Length(DateTim
eToStr(Now)),Rect,DT_LEFT or DT_VCENTER or DT_SINGLELINE);
  //================================================
=========================
  //4.2.输出页号
*************************************************************
  //================================================
=========================
  Rect := Bounds(PageEdgeX, Temp_Y, Printer.PageWidth-PageEdgeX*2,
RowHeight);
  DrawText(Printer.Canvas.Handle,PChar('#'+IntToStr(Printer.PageNumber)),Len
gth('#'+IntToStr(Printer.PageNumber)),Rect,DT_RIGHT or DT_VCENTER or
DT_SINGLELINE);
end;
//================================================
==============================
begin
PrintDialog := TPrintDialog.Create(DBGrid);
if PrintDialog.Execute then
begin
//================================================
=========================
//0.取当前打印机 X,Y 方向每英寸像素
*****************************************
//================================================
=========================
PixelsPerInchX := GetDeviceCaps(Printer.Handle, LOGPIXELSX);
PixelsPerInchY := GetDeviceCaps(Printer.Handle, LOGPIXELSY);
//================================================
=========================
//1.变量初始化
*************************************************************
//================================================
=========================
PageEdgeX := PixelsPerInchX div 6;
PageEdgeY := PixelsPerInchY div 5 * 4;
```

```
RowHeight := Trunc(1.5 * 9 * PixelsPerInchY / 72);
Scale :=  PixelsPerInchX / Screen.PixelsPerInch;
//=================================================
=========================
try
  Printer.BeginDoc;
  DBGrid.DataSource.DataSet.First;
  while not DBGrid.DataSource.DataSet.Eof do
  begin
    Print_Title;
    Print_Column;
    Print_Cells;
    Print_Footer;
    Printer.NewPage;
  end;
  //=================================================
=========================
  if not Printer.Aborted then Printer.EndDoc;
except
  on E:EPrinter do
  begin
    MessageBox(Application.Handle,PChar('打印机没有准备好！'),'提示!
',MB_OK+MB_ICONINFORMATION);
    Printer.Abort;
    Exit;
  end;
  on E:Exception do
  begin
    raise;
    Exit;
  end;
end;
end;
//=================================================
============================
PrintDialog.Free;
end;
```

二、**dbgrideh** 标题排序:

首先把需要排序的 **title.titlebutton:=true**

ehlib 的 **optioneh** 中 **autosortmarking** 最好设为 **true**

titelbutton 事件中写:

```
var
sortstring:string;
begin
```

```
//进行排序
with Column do
begin
  if FieldName = '' then Exit;
case Title.SortMarker of
  smNoneEh:
    begin
     Title.SortMarker := smDownEh;
     sortstring := Column.FieldName + ' ASC';
   end;
  smDownEh:sortstring:= Column.FieldName + ' ASC';
  smUpEh:sortstring:=Column.FieldName + ' DESC';
end;

//数据集排序。
try
 dataset.Sort:=sortstring //dataset 为实际数据集变量名
except
end;
```

delphi 一句话帮助

1.如果想你的程序能够正确处理异常情况的话，请引用 **SysUtils.pas** 单元，否则即使程序使用了 **try**。。。**except**。。。也不能正确捕获异常。

2.定义常量字符串的一种方式

```
resourcestring
    aa='aaaa';
raise Exception.CreateRes(@aa);
```

3.字符串常量数组的初始化

```
const  constarray:array [0..2] of string=(‘first’,’second’,’third’);
```

4. 结构体初始化

```
type  Tstructinit=record
   A1:integer;
   A2:array [0..2] of integer;
End;
Const m_structinit:Tstructinit=(A1:0;A2:(0,1,2));
```

5.多维数组的长度

```
var  array2:array of array of integer;
setlength(array2,2,2);
```

6.使用 **Create** 和 **New** 开辟的空间都存在于堆中，不能自动释放，建议使用 **FreeAndNil** 释放，参数以及局部变量存在于栈中，自动释放。

7.SizeOf 不适合于对象，返回的总是 **4**；对于固定类型可以正确返回.

8.Create(nil)需要手工释放，**Creat(self)**会随着拥有者的释放而释放.

9.动态改变已定义常量的值

```
procedure ChangeConst(const Const;var Value;Size:Integer);
```

```
begin
  Move((@Value)^,(@Constant)^,Size);
End;
```

10.进行删除操作的时候循环使用 DownTo,会避免错误.

11.汉字的 Ascii 码>128，可以用它来判别是否为汉字

12.dll 编写中，需要使用 Sharemem 单元来引用 BORLANDMM.DLL 内存管理.

13.PostMessage 只将消息放到消息队列中，需要排队等待处理。
SendMessage 绕过消息队列直接发送到窗口过程,等到消息处理返回值才返回.

14.鼠标移入移出消息：CM_MOUSEENTER,CM_MOUSELEAVE

15.关机消息 WM_QUERYENDSESSION

16.可以利用 ThintWindow 和类的方法 ActivateHint 来创建浮动窗体.

17.调出文件属性对话框

```
uses ShellAPI;
function ShowFileProperties(FileName: String; Wnd: HWND):Boolean;
var
  sfi: TSHELLEXECUTEINFO;
begin
  with sfi do
  begin
    cbSize := SizeOf(sfi);
    lpFile := PAnsiChar(FileName);
    Wnd := Wnd;
    fMask := SEE_MASK_NOCLOSEPROCESS or SEE_MASK_INVOKEIDLIST or
SEE_MASK_FLAG_NO_UI;
    lpVerb := PAnsiChar('properties');
    lpIDList := nil;
    lpDirectory := nil;
    nShow := 0;
    hInstApp := 0;
    lpParameters := nil;
    dwHotKey := 0;
    hIcon := 0;
    hkeyClass := 0;
    hProcess := 0;
    lpClass := nil;
  end;
  Result := ShellExecuteEX(@sfi);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
   ShowFileProperties('c:\AA.txt', Handle);
end;
```

18.更改系统时间

```
uses Windows,Dialogs,Forms;
var MyTime:TsystemTime;
begin
  FillChar(MyTime,sizeof(MyTime),#0);
  MyTime.wYear:=2003;
  MyTime.wMonth:=06;
  MyTime.wDay:=01;
  If not SetSystem(MyTime) then
Showmessage(‘Failure’);
   End;
```

19.复制文件夹 Xcopy

```
.   procedure Xcopy(SourceDir,DestinationDir:String);
    var
     Search : TSearchRec;
    Rec    : word;
Begin
    SourceDir := SourceDir + '\';
     Rec := FindFirst(SourceDir + '*.*', faAnyFile, Search);
    While Rec = 0 Do
     Begin
       If Search.Name[1] <> '.' Then
       Begin
         If (Search.Attr And faDirectory) = faDirectory Then
         Begin
           Windows.CreateDirectory(PChar(DestinationDir + '\' + Search.Name),
nil);
           FileSetAttr(DestinationDir + '\' + Search.Name, FileGetAttr(SourceDir
+ '\' + Search.Name));
            X_Copy(SourceDir + '\' + Search.Name, DestinationDir + '\' +
Search.Name);
         end
         Else
         Begin
           CopyFile(PChar(SourceDir + '\' + Search.Name),PChar(DestinationDir
+ '\' + Search.Name), True);
           FileSetAttr(DestinationDir + '\' + Search.Name, FileGetAttr(SourceDir
+ '\' + Search.Name));
           Application.ProcessMessages;
         end;
       end;
       Rec := FindNext(Search);
     end;
     FindClose(Search);
end;
```

20.绘制透明位图

```
   procedure DrawTrans(DestCanvas: TCanvas; X,Y: smallint; SrcBitmap:
TBitmap; AColor, BackColor: TColor);
var  ANDBitmap, ORBitmap: TBitmap;
       CM: TCopyMode;
       Src: TRect;
begin
     ANDBitmap:= NIL;
     ORBitmap:=  NIL;
     try
      ANDBitmap:= TBitmap.Create;
      ORBitmap:= TBitmap.Create;
      Src  := Bounds(0,0, SrcBitmap.Width, SrcBitmap.Height);
      with ORBitmap do begin
       Width:= SrcBitmap.Width;
       Height:= SrcBitmap.Height;
       Canvas.Brush.Color := clBlack;
       Canvas.CopyMode := cmSrcCopy;
       Canvas.BrushCopy(Src, SrcBitmap, Src, AColor);
      end;
      with ANDBitmap do begin
       Width:= SrcBitmap.Width;
       Height:= SrcBitmap.Height;
      Canvas.Brush.Color := BackColor;
      Canvas.CopyMode := cmSrcInvert;
      Canvas.BrushCopy(Src, SrcBitmap, Src, AColor);
     end;
     with DestCanvas do begin
      CM := CopyMode;
      CopyMode := cmSrcAnd;
      Draw(X,Y, ANDBitmap);
      CopyMode := cmSrcPaint;
      Draw(X,Y, ORBitmap);
      CopyMode := CM;
     end;
    finally
     ANDBitmap.Free;
     ORBitmap.Free;
    end;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
   DrawTrans(Image1.Canvas, 0,0, Image2.Picture.Bitmap, clBlack, clSilver);
```

```
end;
```

21.获取 CPU 速度

```
  function GetCpuSpeed: Extended;
var
    t, mhi, mlo, nhi, nlo: dword;
    shr32 : comp;
begin
    shr32 := 65536;
    shr32 := shr32 * 65536;
    t := GetTickCount;
    while t = GetTickCount do ;
      asm
      DB 0FH,031H // rdtsc
      mov mhi,edx
      mov mlo,eax
    end;
    while GetTickCount < (t + 1000) do ;
      asm
      DB 0FH,031H // rdtsc
      mov nhi,edx
      mov nlo,eax
    end;
    Result := ((nhi * shr32 + nlo) - (mhi * shr32 + mlo)) / 1E6;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
    label1.Caption := FloatToStr(GetCpuSpeed) + 'mhz';
end;
```

分类: delphi