吉原 泰介

株式会社ミガロ

RAD事業部 技術支援課 顧客サポート

Delphiテクニカルエッセンス Q&A集

略歴
1978年3月26日生
2001年龍谷大学法学部卒
2005年7月株式会社ミガロ入社
2005年7月システム事業部配属
2007年4月RAD事業部配属

現在の仕事内容
Delphi/400やJACi400の製品試験、
および月100件にのぼる問い合わせの
サポートやセミナー講師などを担当している。

はじめに

ミガロでは、月100件近く寄せられるミガロテクニカルサポートへの問い合わせや、個別サポートで得られたテクニックなどを「Delphiテクニカルエッセンス」というセッションの形で、各種セミナーで公開している。

それらのFAQトピックを、読んでわかる資料形式にまとめたものとして公開するのは、より多くの方にDelphiの技術の共有をしてもらいたいと考えたからである。

本稿の中では、具体的なソースが明示されている。ただし、あくまでそれは実現例の1つである。

もちろん、誰もがプログラムを最初に作るときには、既存のプログラムを真似して作成する。だが、もう一歩上達するためには、やはりその真似したプログラムを理解して自分で扱えるようになることが必要だろう。

FAQの回答では、仕組みやロジックをできるだけわかりやすい図形式等で説明しているので、理解に役立ててほしい。それらが、Delphiユーザーのスキルアップの手助けとなれば幸いである。

1.Excel出力 パフォーマンスの改善

Q. OLE を利用した Excel の出力処理で件数が多い場合、処理時間が長くて困っています。

A. Delphi で OLE を利用して Excel を出力する場合、処理量によってはかなりの時間がかかってしまう。

このとき実際に時間がかかっている箇所を特定すると、Delphi ⇔ Excel 間で発生する処理が問題であることがわかる。つまり、同じ内容を Excel に出力するにしても、Delphi ⇔ Excel 間の通信回数を減らす工夫をすることで、格段にパフォーマンス向上を図ることができる。

・処理時間のかかる例

まず、処理時間がかかってしまう例を示す。【図 1】

この場合、書き込むデータごとに、Excel の Cell へアクセスを行ってしまう。そのため、大量の書き込みを行うと Delphi ⇔ Excel 間の通信回数が多くなってしまい、パフォーマンスに問題が出てくる。

・通信回数を減らした例

次に、通信回数を減らす工夫を行った例を示す。【図 2】

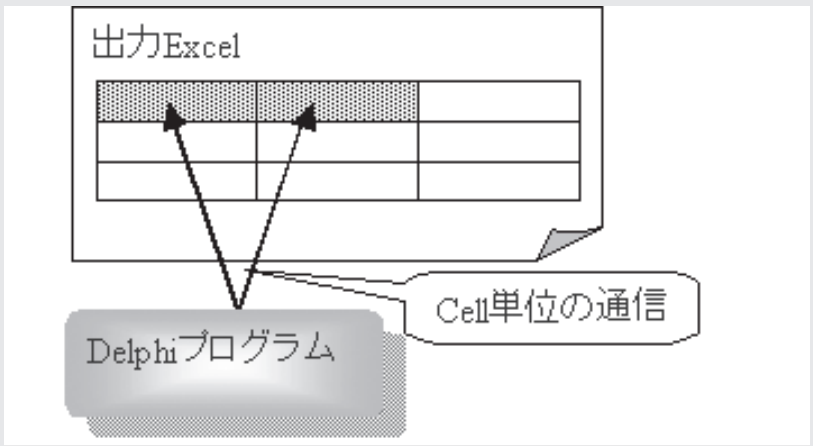
一括で Excel へ書き込むために、文字列を使用している。フィールドごとに #9、改行ごとに #13#10 のリテラルを挟んで編集することで、複数行の内容も文字列で格納することができる。

また、Excel の Cell 構成を考えて、2 次元配列で扱ってもよい。

この場合、書き込むデータが多くとも、その内容を一括で Excel へ書き込む。そのため、通信回数は 1 回（またはまとまった回数）しか行われないので、Delphi ⇔ Excel 間の通信回数を格段に減らすことができる。

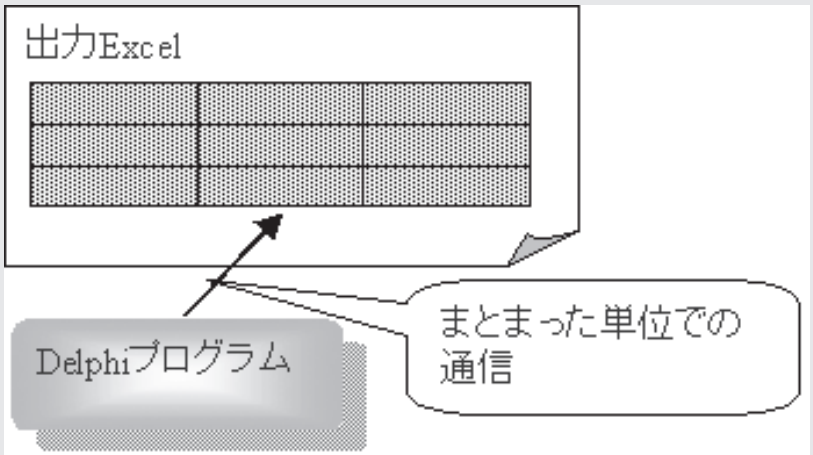
これらによって、Delphi ⇔ Excel 間で一番大きい処理時間を短縮できることになる。

図1　OLEを使用したExcelへの書き出し




```
w hile not(EOF) do
begin
  //Cellごとに値を書き込みを行う。
  WorkSheet.Cells[iRow , 1].Value := FieldByName('TEST01').AsString;
  WorkSheet.Cells[iRow , 2].Value := FieldByName('TEST02').AsString;
  WorkSheet.Cells[iRow , 3].Value := FieldByName('TEST03').AsString;
  WorkSheet.Cells[iRow , 4].Value := FieldByName('TEST04').AsFloat;
   -
   -
   -
```

図2　改善イメージ




```
//クリップボードに書き込み内容を編集して格納
Clipboard.Clear;
Clipboard.AsText = strText;
  -
  -
  -
//クリップボードから貼り付け
WorkSheet.Cells[1, 1].Select;
WorkSheet.Paste;
```

2.DBGridのチェックボックス実装

Q. DBGrid で、行ごとにチェックを付けさせることは可能ですか？

A. DBGrid の標準機能では、チェックボックスを扱っていない。そのため、DrawColumnCell イベントなどに自身で描画処理を行う必要がある。

図は、チェックボックスを扱うフィールドが数値項目である場合の、チェックボックス描画の処理手順である。【図 3】

同じ仕組みで、チェックボックスを文字列フィールドで扱う場合は、ブランクと'1'などで代用することができる。

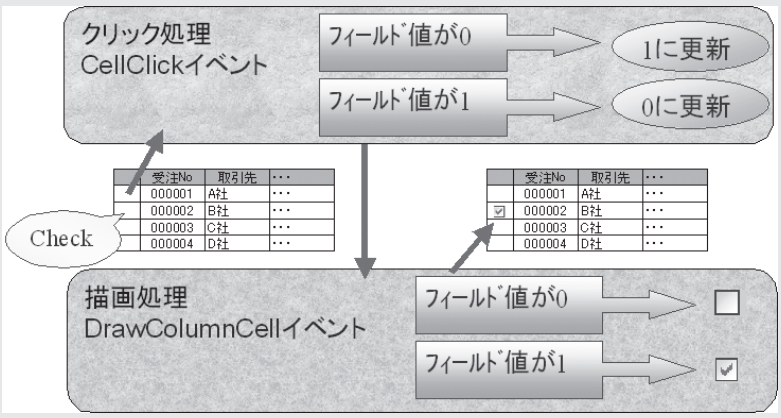
また、この仕組みを実現するためのDBGrid のクリックイベントと描画イベントのソース例を示す。【ソース 1】【ソース 2】

このソースでは、イベントの動作上、DBGrid の Options>dgEditing が False であることを前提としている。

ソース1 DBGrid上でのチェックボックスデータの操作例

```
//OnCellClick イベント
procedure TForm1.DBGrid1CellClick(Column: TColumn);
var
  SaveOptions:TDBGridOptions;
  AFieldName: String;
begin
  with DBGrid1 do
  begin
    if(Assigned(Column.Field)) then
    begin
      SaveOptions := Options;
      try
        if (not Column.ReadOnly) and (Column.Field.Tag = 9) and
          (DataSource.DataSet.Active) then
        begin
          Options := Options - [dgEditing];
          AFieldName := Column.FieldName;
          if (DataSource.DataSet.State = dsBrowse) then
            DataSource.DataSet.Edit;
          if (Column.Field.DataType = ftInteger) then
          begin
            if DataSource.DataSet.FieldByName(AFieldName).AsInteger = 1 then
              DataSource.DataSet.FieldByName(AFieldName).AsInteger := 0
            else
              DataSource.DataSet.FieldByName(AFieldName).AsInteger := 1;
          end;
          DataSource.DataSet.Post;
        end
        else
          Options := SaveOptions;
      except
        Options := SaveOptions;
        raise;
      end;
    end;
  end;
  inherited;
end;
```

図3　実装イメージ



ソース2 DBGrid上でのチェックボックス描画例

```
procedure TForm1.DBGrid1DrawColumnCell(Sender: TObject; const Rect:
TRect;
  DataCol: Integer; Column: TColumn; State: TGridDrawState);
var
  AFieldName: string;
  AField: TField;
  MyRect: TRect;
const
  CBHeight=14;
begin
  MyRect := Rect;
  MyRect.Top := Trunc((MyRect.Bottom - MyRect.Top - CBHeight) / 2)
              + MyRect.Top;
  MyRect.Bottom := MyRect.Top + CBHeight;
  with DBGrid1 do
  begin
    if(Assigned(Fields[DataCol])) then
    begin
      if (Fields[DataCol].Tag = 9) then
      begin
        AFieldName := Columns[DataCol].FieldName;
        AField := DataSource.DataSet.FieldByName(AFieldName);
        Canvas.FillRect(Rect);
        if AField.Value <> Null then
        begin
          if (Fields[DataCol].DataType = ftInteger) then
          begin
            if (AField.AsInteger =1) then
            begin
                Windows.DrawFrameControl(Canvas.Handle, Myrect,
                  DFC_BUTTON, DFCS_BUTTONCHECK + DFCS_CHECKED);
            end
            else
            begin
                Windows.DrawFrameControl(Canvas.Handle, Myrect,
                  DFC_BUTTON, DFCS_BUTTONCHECK);
            end;
          end;
        end
        else
        begin
          Windows.DrawFrameControl(Canvas.Handle, Myrect,
            DFC_BUTTON, DFCS_BUTTONCHECK + DFCS_INACTIVE);
        end;
      end;
    end;
  end;
  inherited;
end;
```

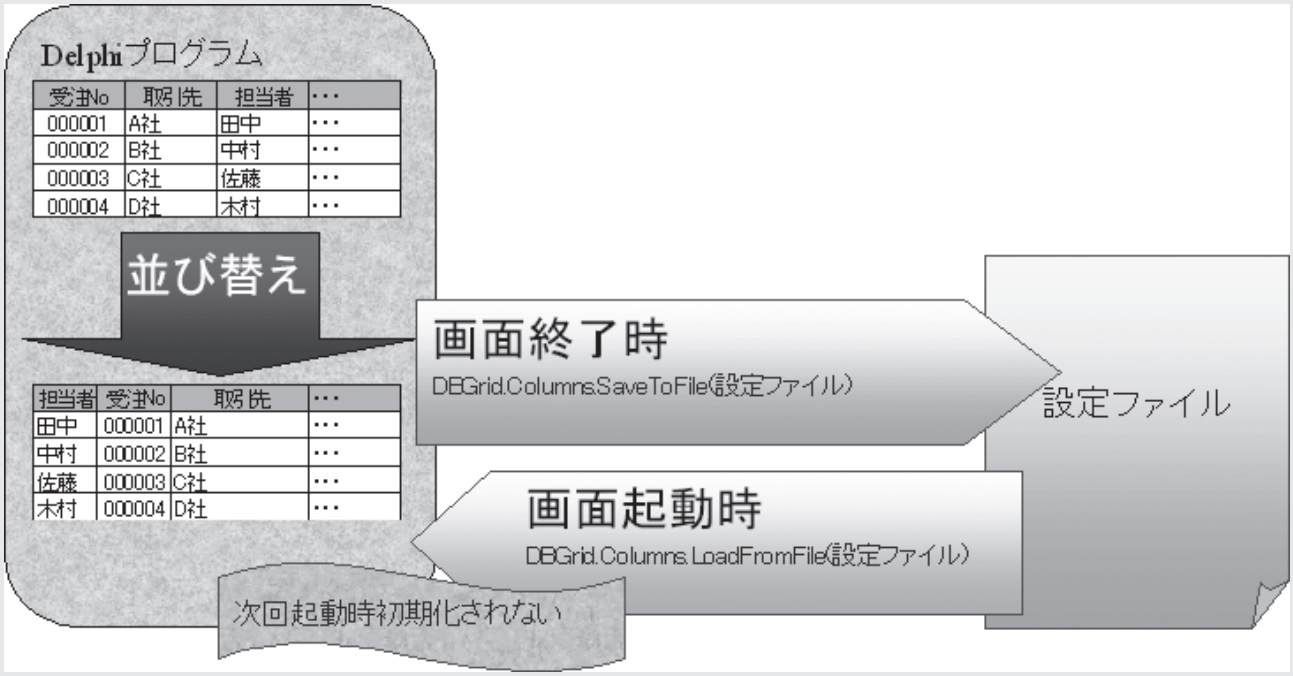
3.DBGridの表示状態の保存

Q. DBGrid のアプリケーション操作上で変更する表示状態を、ユーザーごとに持たせることはできますか？

A. DBGrid の Columns オブジェクトには、標準で SaveToFile や LoadFromFile というメソッドが用意されている。

これを利用することで、ユーザーが変更した表の実行状態を、テキストファイルに保存・読み込むことができる。【図 4】

図4　実装イメージ



4.DBGridの簡単ソートの実装

Q. DBGrid の明細に対して、マウスクリックで並び替えができますか？

A. DBGrid の OnTitleClick イベントで、表のタイトルをクリックした際の処理を制御することができる。

ClientDataSet 経由で DBGrid にデータを表示していることが前提である。ここで、ClientDataSet の index を操作することで、画面上からクリックするだけのソート機能を実装することができる。【図 5】

この仕組みでは、DBGrid のタイトルをクリックするだけで、①で昇順・②で降順・③で解除、という操作を想定している。また、この仕組みはフィールド単位で Index 操作を行うため、複数のフィールドを昇順や降順で組み合わせて扱うこともできる。

この①～③の DBGrid の、タイトルクリックイベントのソース実装例を示す。【ソース 3】

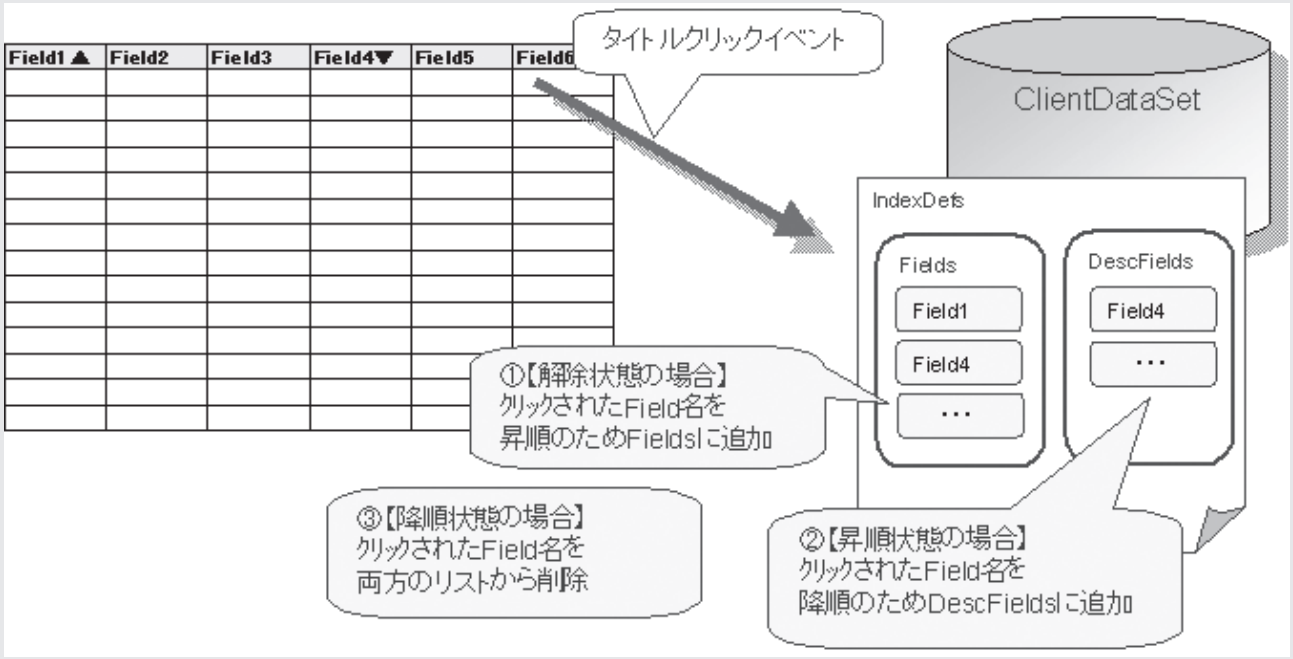
・Index の削除と再追加

ソースの最後に、Index の削除と再追加を行っている。これはパフォーマンス維持のためである。

操作した Index は通常、ClientDataSet を再度 Open した際に反映される。データ量によってはアクセスに時間がかかり、ソートでクリックするたびに時間がかかってしまうため、操作性が悪くなる。

例のように、Index を一度削除して再度追加を行う手法であれば、ClientDataSet を Close せずにソート内容を画面に反映させることができる。

図5　実装イメージ



ソース3 DBGrid上でのクリックによるソート機能例

```
//OnTitleClick イベント
procedure Tfrm1.DBGrid1TitleClick(Column: TColumn);
var
  sFieldNM  :String;  // 退避フィールド名
begin
  with DBGrid1 do
  begin
    // 明細非表示時は処理無効
    if DataSource.DataSet.Active = False then Exit;

    // カラムのフィールド名を退避
    sFieldNM := Column.FieldName;

    with (DataSource.DataSet as TClientDataSet) do
    begin
      // インデックスフィールド作成
      if IndexDefs.Count = 0 then IndexDefs.Add('aIndex', '', []);

      // 明細の表題／並び替えの制御
      if AnsiPos(' ▲', Column.Title.Caption) <> 0 then
      begin
        // ----- 降順へ -----
        // 表題設定
        Column.Title.Caption := StringReplace(Column.Title.Caption, ' ▲', '', [rfReplaceAll]);
        Column.Title.Caption := Column.Title.Caption + ' ▼';

        // 降順フィールドの設定
        if IndexDefs[0].DescFields = '' then
          IndexDefs[0].DescFields := sFieldNM
        else
          IndexDefs[0].DescFields := IndexDefs[0].DescFields + ';' + sFieldNM;

        // インデックスフィールドのオプション初期化
        IndexDefs[0].Options := [];
      end
      else if AnsiPos(' ▼', Column.Title.Caption) <> 0 then
      begin
        // ----- 設定解除 -----
        // 表題設定
        Column.Title.Caption := StringReplace(Column.Title.Caption, ' ▼', '', [rfReplaceAll]);

        // 並び順の設定解除（降順フィールド）
        if AnsiPos(sFieldNM + ';', IndexDefs[0].DescFields) <> 0 then
          sFieldNM := sFieldNM + ';'
        else if AnsiPos(';' + sFieldNM, IndexDefs[0].DescFields) <> 0 then
          sFieldNM := ';' + sFieldNM;

        IndexDefs[0].DescFields := StringReplace(IndexDefs[0].DescFields,
                                                 sFieldNM, '', [rfReplaceAll]);
        // カラムのフィールド名を再取得
        sFieldNM := Column.FieldName;

        // 並び順の設定解除（昇順フィールド）
```

```
        if AnsiPos(sFieldNM + ';', IndexDefs[0].Fields) <> 0 then
          sFieldNM := sFieldNM + ';'
        else if AnsiPos(';' + sFieldNM, IndexDefs[0].Fields) <> 0 then
          sFieldNM := ';' + sFieldNM;

        IndexDefs[0].Fields := StringReplace(IndexDefs[0].Fields, sFieldNM, '  ',
                                                                    [rfReplaceAll]);
        // 並び替え完全解除の場合インデックス名クリア
        if IndexDefs[0].Fields = '' then IndexName := '';

        // インデックスフィールドのオプション初期化
        IndexDefs[0].Options := [];
      end
      else
      begin
        // ----- 昇順へ -----
        // 表題設定
        Column.Title.Caption := Column.Title.Caption + '▲';

        // 昇順フィールドの設定
        if IndexDefs[0].Fields = '' then
          IndexDefs[0].Fields := sFieldNM
        else
          IndexDefs[0].Fields := IndexDefs[0].Fields + ';' + sFieldNM;

        // インデックスフィールドのオプション初期化
        IndexDefs[0].Options := [];
          // インデックス名設定
        if IndexName = '' then IndexName := 'aIndex';
      end;

      // データを開いたままソートを適用させるため Index を削除して再設定
      DeleteIndex('aIndex');
      IndexName := 'aIndex';
      First;
    end;
  end;
end;
```

5.Editの右寄せ表示

Q. Edit コンポーネントで数値項目を表示する場合、右寄せで表示することはできますか？

A. Edit コンポーネントの標準プロパティには、Alignment プロパティが存在しない。そのため、右寄せや中央寄せといった操作を行うことができない。

対応方法はいくつかあるが、コンポーネント個別に対処を行うと、メンテナンスが手間となる。Edit コンポーネントを派生させて、新しいコンポーネントを作成する手法が一般的である。【ソース 4】

ソース4 右寄せコンポーネント実装例

```
unit REdit;

interface

uses
  SysUtils, Classes, Controls, StdCtrls, Windows;

type
  TREdit = class(TEdit)
private
  { Private 宣言 }
protected
  { Protected 宣言 }
  procedure CreateParams(var Params: TCreateParams); override;
public
  { Public 宣言 }
published
  { Published 宣言 }
end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents(' パレットページ名 ', [TREdit]);
end;

{ TREdit }
procedure TREdit.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  Params.Style := Params.Style or ES_RIGHT;
end;

end.
```

6.クライアント端末のIPアドレスの取得

Q. 実行プログラムが動作しているクライアント端末のIPアドレスを取得することはできますか？

A. 実行環境にもよるが、IPアドレスは、WinSock APIのGetHostNameおよびGetHostByNameを利用することで取得可能である。

IPアドレスを取得し、一般的な3桁区切りの'XXX.XXX.XXX.XXX'という形の文字列で、IPアドレスを返却する関数例を示す。【ソース5】

ソース5 IPアドレス取得関数例
usesにWinSockを追記

```
function GetIPAddress: String;
var
  wVerReq: Word;
  WSData: TWSAData;
  Buff: array[0..255] of Char;
  Host: PHostEnt;
  IP: PChar;
begin
  wVerReq := MakeWord(1, 1);
  if WSAStartup(wVerReq, WSData) = 0 then
  try
    if GetHostName(Buff, Length(Buff)) = 0 then
    begin
      Host := GetHostByName(@Buff);
      if Host <> nil then
      begin
        IP := Host^.h_addr_list^;
        Result := IntToStr(Integer(IP[0]))
          + '.' + IntToStr(Integer(IP[1]))
          + '.' + IntToStr(Integer(IP[2]))
          + '.' + IntToStr(Integer(IP[3]));
      end;
    end;
  finally
    WSACleanup;
  end;
end;
```

7.VB-Report Ver3.0での効率的なExcelフォーマット

Q. VB-Report Ver3.0 を使用して、帳票・Excel の出力をしています。パフォーマンスを見直すにはどうしたらよいでしょうか？

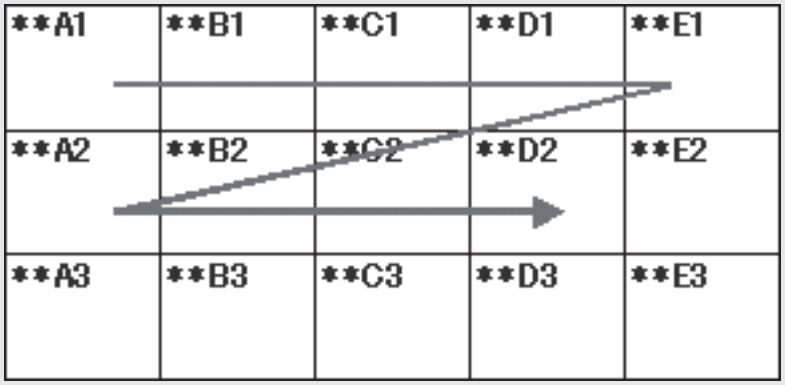
A. VB-Report Ver3.0 で Excel を扱う場合、出力パフォーマンスにはフォーマットとなる Excel 自体が大きく影響する。

以下の点をふまえてフォーマットを作成すると、同じ Delphi プログラムからの Excel 出力も効率よく行うことができる。

① Cell の属性や設定は、フォーマット側で事前に設定する。
②フォーマット内の情報を少なくする。
Cell 数、書式、属性、結合、罫線、網掛等
③可能であれば、固定（A1 参照や座標）形式で指定する。
※変数指定の場合、変数の位置検索が行われる。そのため、通常の固定指定よりもパフォーマンスが落ちる。
④位置指定で変数を使用する場合、変数の位置検索にあわせて先頭行（あるいは前処理での Cell）から、小さい行、小さい列の順に処理を行う。【図 6】

特に②については、Excel で帳票フォーマットを作る際に、レイアウトを細かく設定するために、必要以上に細かく Cell を分割していることが多く、これらの Cell 情報が多ければそのぶん、処理に時間がかかってしまう。

図6　VB-Report Ver3.0での位置検索順



8.TreeViewでの動的メニュー制御

Q. メニュー内容を、DB上のファイルで動的に構成できないでしょうか？

A. メニューは一般的にボタンを配置した画面が多いが、動的にメニューを作成するのであれば、TreeViewを使用する。エクスプローラーのような形で、見やすいメニューアイテムを動的に制御することができる。【図7】

・TreeView コンポーネント

TreeView コンポーネントでは、アイテム（ノード）に親子関係を持たせることができる。コードの体系によって親子関係をブレイクするロジックにしておけば、データのコードによって、動的なアイテム（ノード）を構築することができる。

この仕組みで、メニューを管理するファイルからメニューアイテムを読み込み、TreeViewへ動的にアイテム（ノード）を追加していく関数例を示す。【ソース6】

アイテム（ノード）を追加する際には、AddChildObjectというメソッドを使用して、パラメータに親となるアイテム（ノード）、表示名、付加情報（ここでは起動プログラム名）を受け渡す。最上位のアイテム（ノード）の場合、親は存在しないので、nilをパラメータに設定する。

・アイコンと階層

また、TreeViewは、Imagesプロパティで TImageListを設定して、表示アイコンを持つことができる。事前にアイコンを登録したり、実際にメニューから起動される実行ファイルで ExtractIconを使用して、動的にアイコンを取得・設定することもできる。

TreeView コンポーネントは、階層別に折りたたみができるので、視覚的にわかりやすい画面を作ることができる。

例えば、勘定科目のような照会画面などに活用することで、使いやすいユーザーインターフェースを提供することができるだろう。【図8】

図7　実装イメージ

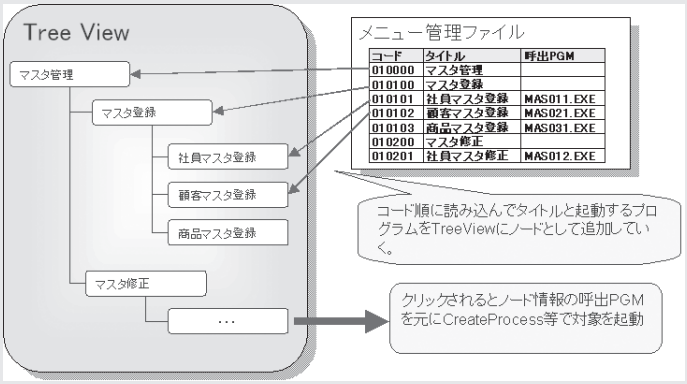
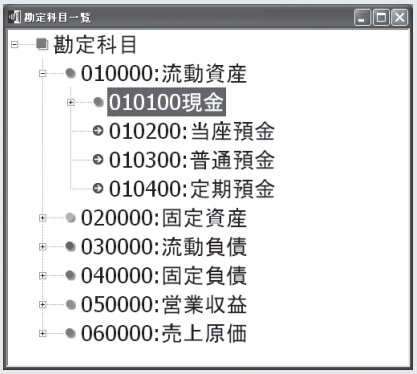


図8　勘定科目一覧の例



ソース6 メニューアイテム動的作成関数例

```
// 〜〜〜〜〜〜
type
  PTVRec = ^TTVRec;
  TTVRec = record
    EXENM: string;
end;
// 〜〜〜〜〜〜

// メニューアイテム動的作成関数
procedure TfrmQ3.LoadMenu;
var
  TopNode : TTreeNode;          // 追加されたトップノード
  ChdNode1: TTreeNode;          // 追加された子ノード
  ChdNode2: TTreeNode;          // 追加された孫ノード
  TVRecPtr: PTVRec;             // ノード情報
  BrkCd1  : string;             // 大区分
  BrkCd2  : string;             // 中区分
  Title   : string;             // タイトル
  MenuCD  : string;             // メニューコード
begin
  // 既存のノードを削除
  TreeView1.Items.Clear;

 // 初期設定
  TopNode := nil;
  BrkCd1  := '';
  BrkCd2  := '';
```

```
  // ファイルより追加
  with  Datamodule1.CdsMenu  do
  begin
    First;
    while  not  eof  do
    begin
      // データをセット
      New(TVRecPtr);

      // プログラムファイルの設定
        MenuCD              :=FieldByName('CODE').AsString;
      Title                 :=FieldByName('TITLE').AsString;
      TVRecPtr^.EXENM:=PathName +
                              FieldByName('EXE').AsString +
                              '.EXE';
// トップノード
      if  Copy(MenuCD,1,2) <> BrkCd1  then
      begin
        TopNode := TreeView1.Items.AddChildObject(nil,Title,TVRecPtr);
      end
      else
      begin
        if  Copy(MenuCD,3,2) <> BrkCd2  then
        // 子ノード
        begin
          // ノード追加
          ChdNode1 := TreeView1.Items.AddChildObject(TopNode,Title,TVRecPtr);
        end
        else
        // 孫ノード
        begin
          // 中分類 '00' のとき、親ノードがトップノード
          if  Copy(MenuCD,3,2) = '00'  then
            ChdNode1 := TopNode;

          ChdNode2 := TreeView1.Items.AddChildObject(ChdNode1,Title,TVRecPtr);
        end;
      end;

      // 大中分類退避
      BrkCd1 :=  Copy(MenuCD,1,2);
      BrkCd2 :=  Copy(MenuCD,3,2);

      // 次データへ
      Next;
    end;
    First;
  end;

  // 全ノードを展開
  TreeView1.FullExpand;

end;
```

9.一覧明細での画像表示

Q. ファイルサーバやローカルにある画像ファイルを、一覧検索の明細に表示できますか？

A. DBのファイルレコード上にない画像を表示する場合、TImageを使用することになる。

DBCtrlGrid上で一覧明細として表示したい場合は、描画を行うOnPaintPanelイベントを利用することで実現が可能である。【図9】

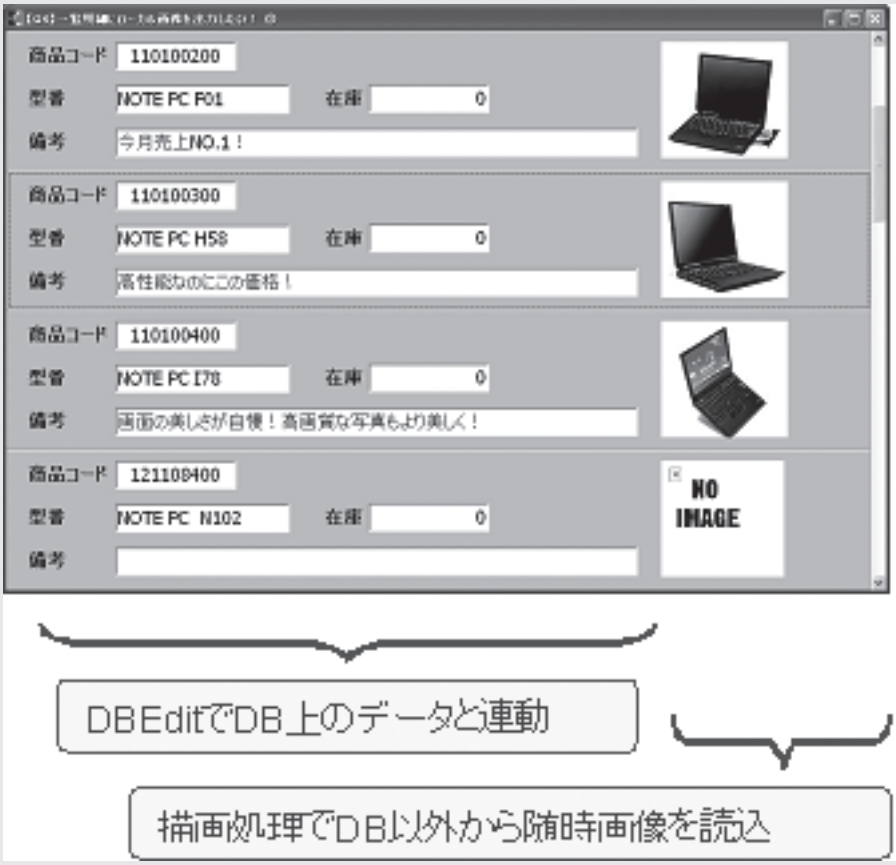
OnPaintPanelイベントでの画像読み込みの例を示す。【ソース7】

ソース7 OnPaintPanelイベントでの画像読み込み例

```
procedure Tfrm1.DBCtrlGrid1PaintPanel(DBCtrlGrid: TDBCtrlGrid; Index:
Integer);
begin
  with DBCtrlGrid.DataSource.DataSet do
  begin
    if not(Active) then Exit;

    // 画像出力
    try
      // 画像を読み込み (bmp 画像 )
      Image1.Picture.LoadFromFile(‘フィールド内容で画像を指定’);
    except
        try
      Image1.Picture.LoadFromFile(‘対象画像ファイルがない場合の画像 ');
      except
      end;
    end;
  end;
end;
```

図9　実装イメージ



10.StringGridでの文字列縦表示

Q. StringGrid で表示される文字列を縦表示したいのですが、可能でしょうか？

A. StringGrid の標準出力では、横表示となってしまう。そのため、描画イベントの OnDrawCell イベントで対象のフォントを回転させて、TextOut や Drawtext で出力させることで縦描写ができる。

OnDrawCel イベントでの描画のソース例を示す。【ソース 8】

この仕組みでは、回転したフォントを作成して、Canvas 上のフォントと入れ替えを行っている。LFont の ifEscapement で回転の角度を設定することができる。

例えば、90 度であれば 900、270 度であれば 2700 と設定する。

・注意点

注意点としては、使用するフォントに縦書き用のフォントを使用した場合は、Single-Byte 文字と Double-Byte 文字とで回転方向が異なる点である。

フォントの角度ごとの回転例を示すので参考にしてほしい。【図 10】

ソース8 文字列回転描画例

```
//OnDrawCell イベント
procedure TfrmQ5.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
var
  LFont: TLogFont;
  NFont, OFont: HFont;
begin
  with StringGrid1 do
  begin
    Canvas.Font.Size := 20;
    //Font を取得し、回転させます。
    GetObject(Canvas.Font.Handle, SizeOf(LFont), @LFont);
    with LFont do
    begin
      lfEscapement      := 900; // 通常：0、左：900、右：2700
      lfCharSet         := DEFAULT_CHARSET;
      lfOutPrecision    := OUT_DEFAULT_PRECIS;
      lfClipPrecision   := CLIP_DEFAULT_PRECIS;
      lfQuality         := DEFAULT_QUALITY;
      lfPitchAndFamily:= DEFAULT_PITCH;
        lfFaceName      := '@ＭＳ ゴシック';
    end;
     // 描画領域が持っている Font と入れ替えます。
    NFont := CreateFontIndirect(LFont);
    OFont := SelectObject(Canvas.Handle, NFont);
     // 描画領域の内容を FillRect でクリアします。
    Canvas.FillRect(Rect);
    //Font が変更された状態で描画領域へ文字列を書込みます。
    Canvas.TextOut(CellRect(ACol, ARow).Left + 9,
                   CellRect(ACol, ARow).top  + 85,
                   Cells[ACol, ARow]);
    // 入れ替えた Font を元に戻し、新たに設定した Font 情報を
    // 開放します。
    NFont := SelectObject(Canvas.Handle, OFont);
    DeleteObject(NFont);
  end;
end;
```

図10　フォントの角度ごとの回転例