MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

No.1
2008年秋

株式会社ミガロ.

CONTENTS 目次

Migaro Technical Award 2008 お客様受賞論文／ミガロ テクニカルアワード

Migaro Technical Report 2008 SE 論文／ミガロ テクニカルレポート

Migaro Technical Award 2008

お客様受賞論文／ミガロ テクニカルアワード

最優秀賞

直感的に理解できるシステムを目指して
――情報の"見える化"の取り組み

石井 裕昭 様
豊鋼材工業株式会社
製造総括部　部長代行

豊鋼材工業株式会社
http://www.yutaka-steel.co.jp/

「鋼材のことならあらゆるニーズに応える企業」をモットーに、広島から沖縄までをカバーし、鋼材およびその他金属の加工、販売を行っている。伊藤忠丸紅鉄鋼・新日本製鐵系の会社である。

ソリューション導入の経緯と概要

豊鋼材工業株式会社の主体である溶断事業では、従来、生産に関するさまざまな情報が、紙の帳票による事後の実績入力や、あるいは担当者の経験等に基づく繁閑の判断と工程管理に依存していた。したがって営業部門、経営者、さらに製造部門内からも、工場の操業状況や受注製品の進捗・計画状況はブラックボックス化して、お客様対応のレスポンスやさまざまな判断に支障をきたしていた。

このような状況を打開するため、平成17年より工場内の無線LAN強化をはじめ、各設備へのWindows CE端末、無線ハンディターミナル、およびラベルプリンタの配備と帳票へのバーコード出力等の基盤整備を開始した。【図1-1】

Delphi/400は、この基盤整備でリアルタイムに収集される情報を定量的かつ直感的に利用する手段として導入した。本アプリケーションは、生産・出荷計画、進捗、履歴、山積み等の情報をさまざまな角度から提供する参照系と、生産指示や計画作成等の更新系の処理を備えている。

開発の独創性・創意工夫

新しいシステム基盤Delphi/400等の導入により、従来は参照できなかった情報がリアルタイムに抽出可能となった。しかし、膨大な情報の中から、ユーザーが着目すべき部分をわかりやすく、かつ直感的に把握・操作できるようにしなければ、使い勝手のよいシステムとは言えない。

そのため、今回のアプリケーションでは次のような工夫している。

① DBGrid表示データの中で着目すべき部分を着色表示、その閾値や色をユーザーによるカスタマイズ
② DBGridの着色（色相）情報をVB-Report色番号に変換し、Excelに出力（VB-Report）
③ DBGridの選択行全体を強調表示
④ Chartを使用したトレンド等の可視化と、DBGrid等との組み合わせによる詳細確認機能
⑤ DHTML形式の操作マニュアルによる、シミュレーション体験型ヘルプ機能
⑥ Webブラウザの活用
・GoogleMapによるお客様地図の表示（縮尺の異なる2段階表示）
・ネットワークカメラによる簡易Web会議
・ネットワークカメラによる工場の状況確認
・無料のWeb型リモートデスクトップソフトによる画面共有
⑦ BitBtn、SpeedBtn等への画像表示
⑧ DecisionGrid、DecisionGraphの活用

見える化への改善ポイント

アプリケーション開発の独創性・創意工夫について詳細を述べる。

図1-1　生産管理システム再構築イメージ

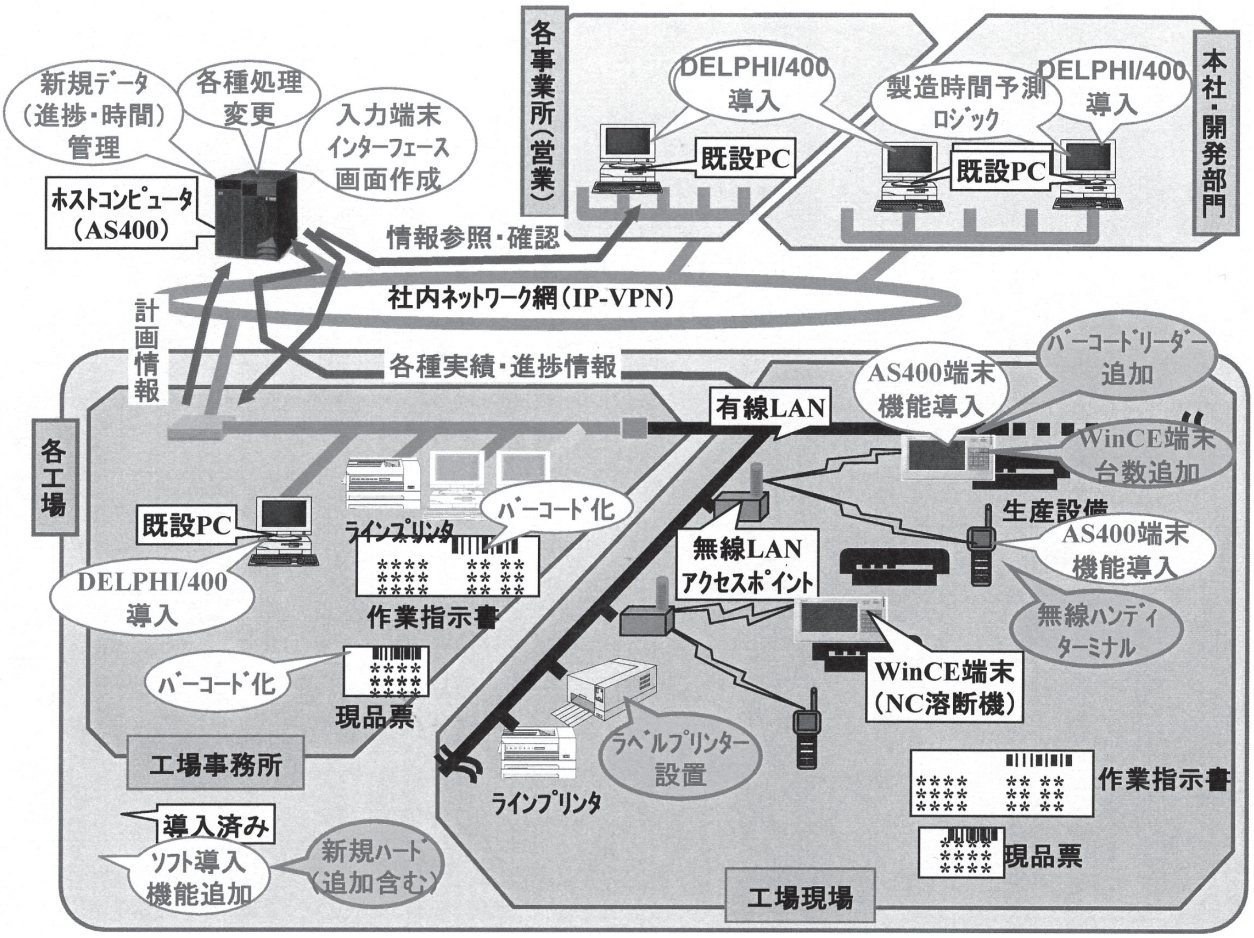


図2-1　汎用のGrid着色関数呼び出し例

```
procedure TForm1.PJJuSumGridDrawColumnCell(Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumn;
  State: TGridDrawState);
var
n:integer;
begin
    n:=(Sender as TDbGrid).Tag;

    with  (Sender as TDbGrid).Canvas do
    begin

     if GetGridColorForOP(n,DataCol)<>0 then
       Brush.Color:=GetGridColorForOP(n,DataCol);

     FillRect(Rect);
    end;
```

DBGridの配列番号=n、対象列番号=DataColで色を取得

① DBGrid の表示データ中で着目すべき行、セル、列あるいは特定のデータ内容などに応じて、傾向管理や識別容易化のために、OnDrawColumnCell イベントで着色判定用関数を呼び出している。

この関数は、呼び出し元の DBGrid の配列番号と項目を引数とし、DBGrid ごとの判定ロジックにより着色要否や色番号を返す。【図 2-1】【図 2-2】【図 2-3】

これによりユーザーは、着目すべき行、値の傾向等を直感的に把握可能になる。着色の判定の閾値と色には、ユーザーが入力した Edit の値や ColorBox で設定した色により変更可能なものもある。【図 2-4】

② VB-Report の機能で、Excel 出力する際に VB-Report3.0 の色定数を指定する必要がある。これは、Delphi 側の色を VB-Report3.0 の色定数に対応させる。【図 2-5】

DBGrid 各セルの着色用関数から返される色の色相範囲に対応して、VB-Report3.0 の色定数に変換する関数を作成した。【図 2-6】【図 2-7】

これにより、共通の着色ロジックでほぼ同等に着色された EXCEL の表を得ることが可能となった。

③通常 DBGrid では、選択行の左端の三角マークと選択セル色の反転で識別される。だが項目数が多い場合にはわかりにくい。

OnDrawColumnCell イベントで、TCustomDBGrid から派生させた TAccessDBGrid での選択行と現レコードのインデックスの一致判定により、着色を行うことが可能になった。【図 2-8】【図 2-9】

④ Query の動的 SQL で時系列に集約したデータを抽出し、ClientDataSet にキャッシュされた状態で、DBCrossTabSource 経由で DBChart 上の系列に接続している。

グループ化基準、集計項目等は、ユーザーの選択に応じて、DBCrossTabSource の GroupField、ValueField 等を動的に切り替える。ClientDataSet のキャッシュデータをもとにしているため、瞬時に角度を変えてのデータ分析が可能となった。

また、1 つの Query の SQL で抽出されたデータに対して、DataSetProvider 経由で複数の ClientDataSet を接続し、各 ClientDataSet の Filter プロパティを個々に変える。1 つの Query で、複数のグラフの一括表示を実現している。【図 2-10】【図 2-11】【図 2-12】【図 2-13】【図 2-14】

さらに、グラフ上のダブルクリックで共通の TeeCommander を呼び出し、ユーザーが独自にグラフの書式編集、印刷、コピー等をできるようにした。その際、呼び出し元のグラフに応じて表示位置を変更し、共有化を実現している。【図 2-15】

グラフで全体感を確認した後に、系列のデータの内容をドリルダウンしたい場面も多い。グラフ上の該当部分をクリックすることで、その要素情報をパラメータとして詳細を表示する機能を多く実装した。【図 2-16】【図 2-17】【図 2-18】

⑤ TENDA 殿のマニュアル作成ツール（DOJO）の機能の 1 つである、DHTML 形式の体験シミュレーション型マニュアルを各機能ページ別に作成し、社内の Web サーバーに保存した。

この該当 HTML を、各ページの Image のクリックイベントで、ShellExecute により呼び出せるようにした。【図 2-19】

実際に操作する感覚で機能・操作方法を習得できるため、学習効果が高く、マニュアル配布、教育等の負荷を軽減できる。音声解説も付加可能である。【図 2-20】【図 2-21】

なお、静的な HTML のマニュアル参照、PDF 形式のマニュアルのダウンロードも可能である。【図 2-22】【図 2-23】

⑥ Web ブラウザをコンポーネントとして組み込み、navigate メソッドで URL を指定するだけで容易に任意の Web ページを表示できる。この機能を応用して、お客様住所の GoogleMap 表示、ネットワークカメラ映像表示、リモートデスクトップ参照（操作）【図 2-24】を可能とした。

・GoogleMap

取得した Google Maps の APIkey を記述した雛形の HTML の住所とお客様名の部分のみを、データベースとの連動により動的に置き換える。【図 2-25】【図 2-26】

GoogleMap の各種コントロールの組み込みは、Google Maps API の解説資料を参照した。

なお、地図ソフトでは目的地近辺の詳細と全体感を確認することが多いため、縮尺の異なる 2 つの Map を 1 つの URL で表示可能とし、印刷も可能とした。【図 2-27】【図 2-28】

住所情報が GoogleMap での検索に適合しない場合（旧漢字等）は一部修正で、再試行可能とした。【図 2-29】

・ネットワークカメラ

映像は、設置したカメラに付与した IP アドレスを URL で指定することで、見ることができる（Web サーバー機能内蔵による）。

今回、各カメラの名称、IP アドレス、アクセス権、接続可能時間等の基準をデータベース管理している。これにより、ユーザーに応じて使用条件をコントロール可能にし、プライバシー、ネットワーク負荷等へ配慮した。また、アクセス中の対象者がわかるように、アクセス状況もデータベース管理している。【図 2-30】

Web 会議と工場状況確認は、ComboBox でカメラを切り替えることで汎用的に使える。【図 2-31】

なお、Web 会議の際には、簡易な IP 電話会議機器を併用する。

⑧ユーザーのフォームからの項目選択により、任意の分析を可能とした。ただし、レコード件数が多い場合は、次元数によりメモリの使用量が大きくなり、実用レベルでは使えなくなる。項目選択数上限を設定している。【図 2-32】【図 2-33】

教訓・知見と今後の予定

Web ブラウザで Web 画面をコンポーネントとして組み込めるということを、Delphi を使い始めて 2 年近く気づかなかった。同様に、有効な機能を持つコンポーネントで知らないものが多くあると思われる。

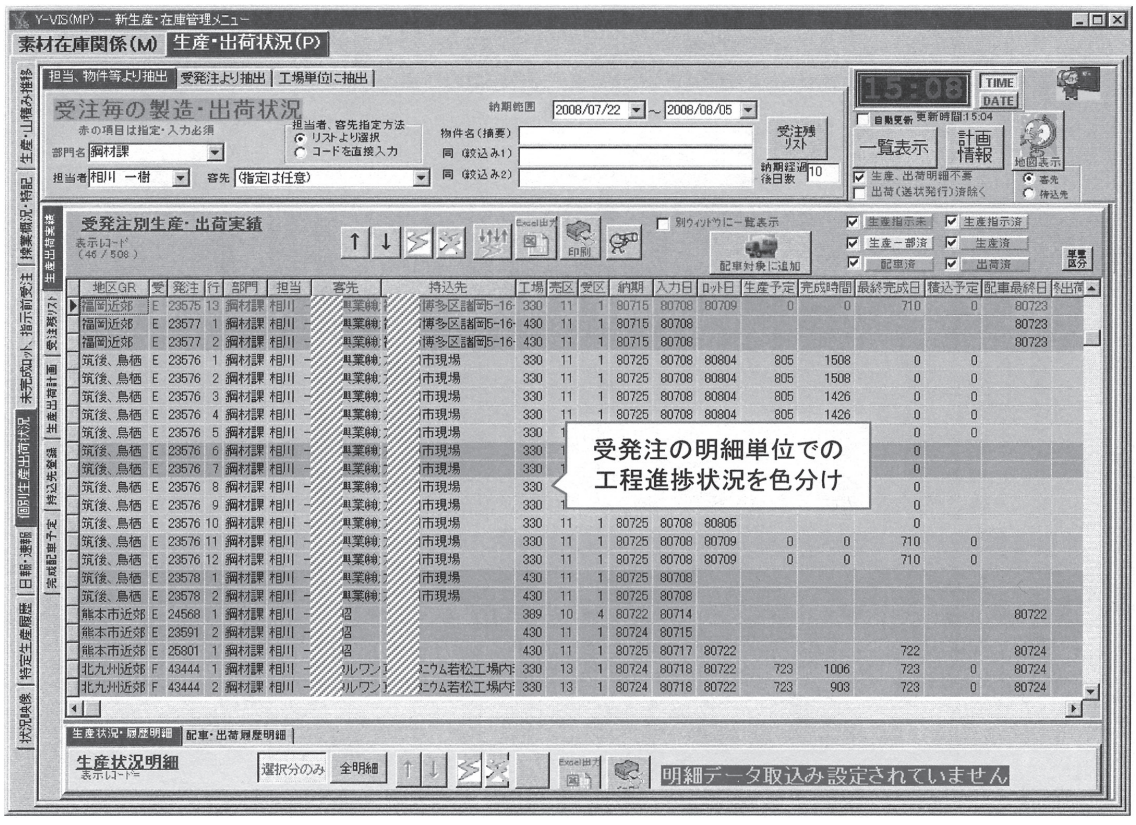
また、標準では付属していないが、有効利用できる可能性があるものも確認できた。

図2-2　DBGridの着色関数の例

```
//Gridの着色を設定する汎用関数
function TForm1.GetGridColorForOP(n:integer;j:integer): integer;
var
 todayNum,todayNum2:integer;
 selcolor:Tcolor;
 colornum:integer;

begin
   colornum:=0; //初期値
   selcolor:=ClWhite;//初期値

 case n of

36:
 begin
     if (Form1.mainCDS[n].FieldByName('SYU033T').Value=Form1.mainCDS[n].FieldByName('JU2013').Value) Or ((Form1.mainCDS[n].Fie
     begin
      selcolor := Form1.StateColorSelFin.Color;
     end
     else if  Form1.mainCDS[n].FieldByName('HAX008T').Value=Form1.mainCDS[n].FieldByName('JU2013').Value then   //全部配車
     begin
      selcolor := Form1.StateColorSelD.Color;
     end
     else if  Form1.mainCDS[n].FieldByName('LO2006F').Value=Form1.mainCDS[n].FieldByName('JU2013').Value then   //全部生産
     begin
      selcolor := Form1.StateColorSelP.Color;
     end
     else if (Form1.mainCDS[n].FieldByName('LO2006F').Value>0) And (Form1.mainCDS[n].FieldByName('LO2006F').Value<Form1.mainC[
     begin
      selcolor := Form1.StateColorSelSP.Color;
     end
     else if  Form1.mainCDS[n].FieldByName('LO2006T').Value=Form1.mainCDS[n].FieldByName('JU2013').Value then   //全部指示
     begin
      selcolor := Form1.StateColorSelL.Color;
     end
     else   //指示未
     begin
      selcolor := Form1.StateColorSelN.Color;
     end;

 end;  //n=36の場合ここまで

 if selcolor<>ClWhite then
 colornum:=selcolor
 else
 colornum:=0;

 result:=colornum ;

end;
```

呼び出したDBGrid毎に着色ロジックを記述

図2-3　DBGridの着色の例



これらについてStudyし、レベルアップを図りたい。

エンドユーザー評価

●印刷帳票からExcelに手入力したデータをもとに定期的に作成していた（または、できなかった）指標の長期・短期の最新トレンドグラフが、人の手を煩わすことなく確認できる。

資料作成、データ取り込み等の手間も大幅に軽減できた。

●工程進捗状況が、DBGridの色分けにより、データの値を見なくても確認できる。対応を迅速に行えるため非常に有効である。

M

図2-4　Grid着色の閾値、色を設定可能とした例

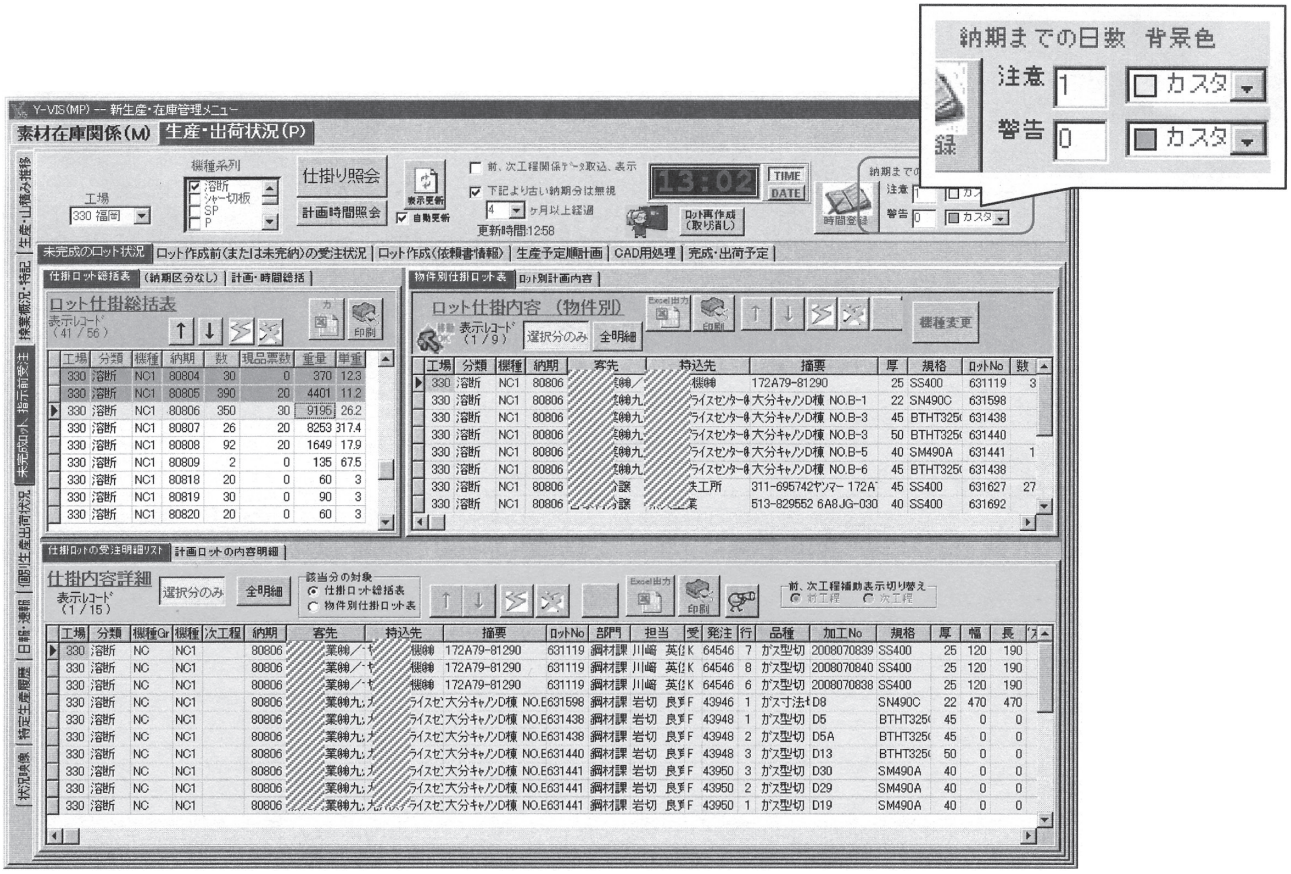


図2-5　VB-Report3.0の色番号例

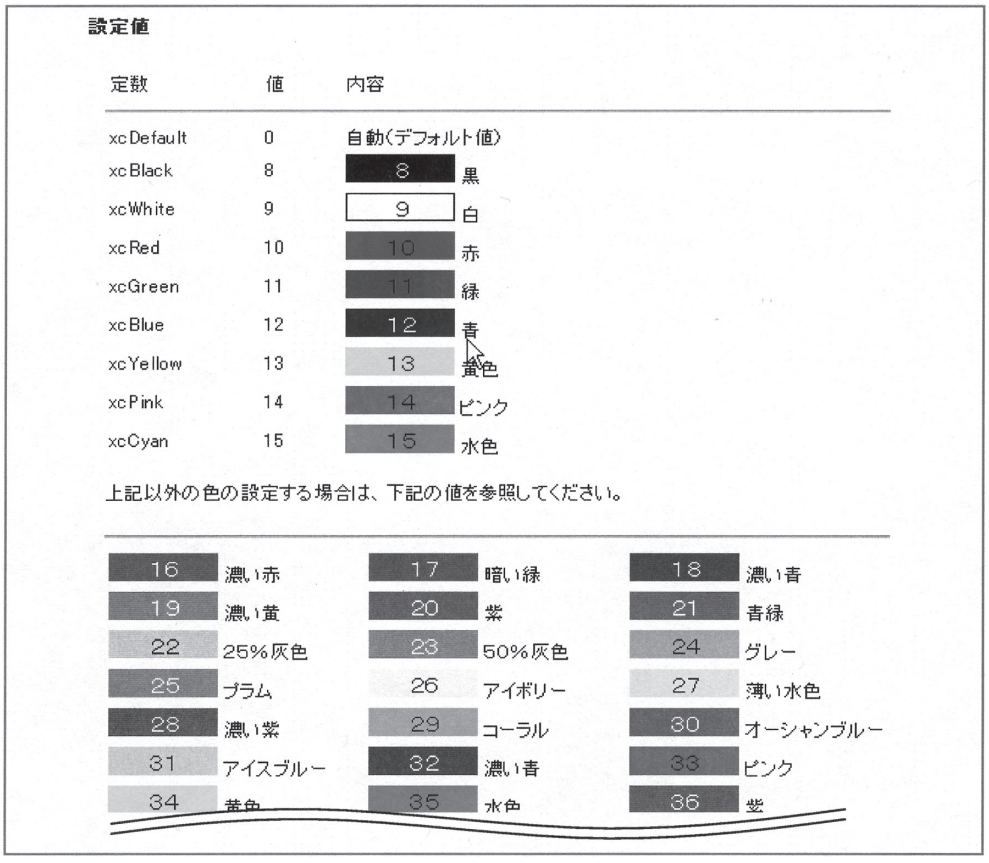


図2-6　EXCEL出力用の色番号取得用関数呼び出し(抜粋)

```
while not mainCDS[j].Eof do      //明細データ
begin
ii:=0;
 for i:=0 to mainCDS[j].FieldCount-1 do
 begin
  if mainCDS[j].Fields[i].Visible=true then
  begin
   XlsReport1.Pos[ii,n,ii,n].Value:=mainCDS[j].Fields[i].Value;
   XlsReport1.Pos[ii,n,ii,n].Attr.Box[0]:=lsNormal;

   if GetGridColorForOP(j,ii)<>0 then
   begin
    if EXCELcolorConv(GetGridColorForOP(j,ii))<>999 then
    XlsReport1.Pos[ii,n,ii,n].Attr.BackColor:=EXCELcolorConv(GetGridColorForOP(j,ii));
   end;
   ii:=ii+1;
  end;
 end;
n:=n+1;
mainCDS[j].next;
end;
```

対象DBGridの
各セルの色を設定

対象DBGridの列毎の色を
VB-Reportの色番号に変換

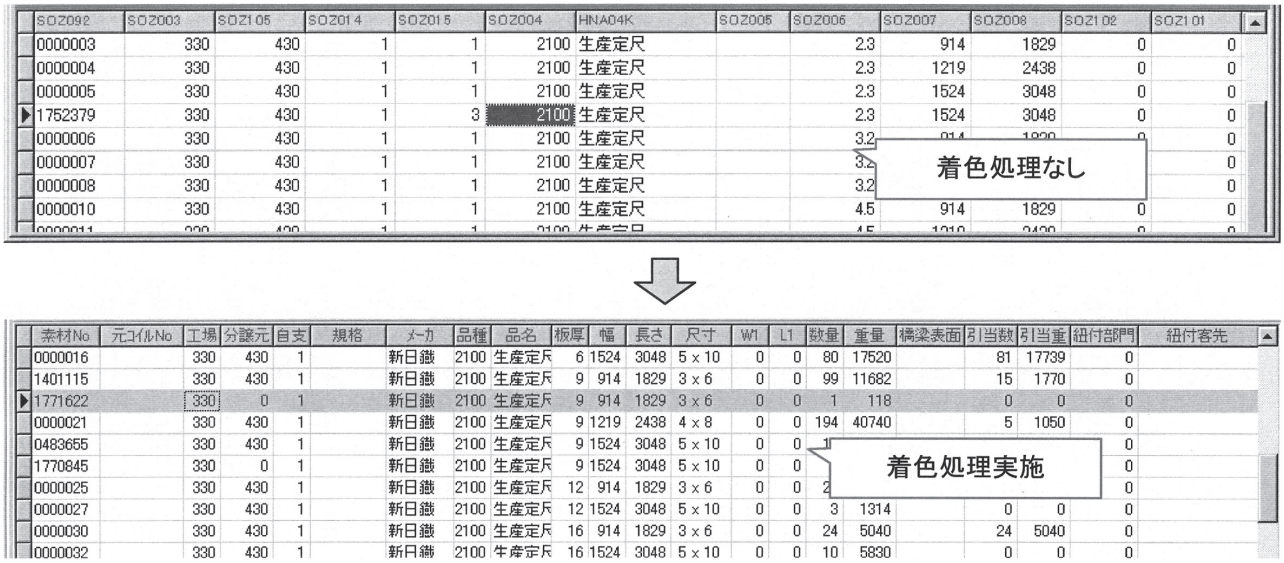
図2-7　同等の色相のVB-Report色番号に変換する関数

```
function TForm1.EXCELcolorConv(originColor: integer): integer;
var
  h: Real; // 色相 hue (0～360) →  この値で着色内容を設定
  s: Real; // 彩度 saturation
  v: Real; // 明度 value
  min: Integer;
  max: Integer;
  delta: Integer;
  IntColor: Integer;
  r,g,b:Byte;
begin
    IntColor:= ColorToRGB(originColor);
    r:= GetRValue(IntColor);
    g:= GetGValue(IntColor);
    b:= GetBValue(IntColor);

    case round(h) of
     0..30: result:=29;
     31..60: result:=47;
     61..90: result:=43;
     91..120: result:=42;
     121..150: result:=57;
     151..180: result:=41;
     181..210: result:=44;
     211..240: result:=31;
     241..270: result:=24;
     271..300: result:=46;
     301..330: result:=29;
     331..360: result:=45;
     else
      result:=999;
    end;

end;
```

ここにr,g,bからh,s,b(各色相、彩度、明度)
に変換する処理を記述

色相(h)の範囲に応じて
VB-Reportの色番号を割り当て

図2-8　DBGridの選択行着色例



SOZ092	SOZ003	SOZ105	SOZ014	SOZ015	SOZ004	HNA04K	SOZ005	SOZ006	SOZ007	SOZ008	SOZ102	SOZ101
0000003	330	430	1	1	2100	生産定尺		2.3	914	1829	0	0
0000004	330	430	1	1	2100	生産定尺		2.3	1219	2438	0	0
0000005	330	430	1	1	2100	生産定尺		2.3	1524	3048	0	0
1752379	330	430	1	3	2100	生産定尺		2.3	1524	3048	0	0
0000006	330	430	1	1	2100	生産定尺		3.2				0
0000007	330	430	1	1	2100	生産定尺		3.2				0
0000008	330	430	1	1	2100	生産定尺		3.2				0
0000010	330	430	1	1	2100	生産定尺		4.5	914	1829	0	0

素材No	元コイルNo	工場	分譲元	自支	規格	メーカ	品種	品名	板厚	幅	長さ	尺寸	W1	L1	数量	重量	橋梁表面	引当数	引当重	紐付部門	紐付客先
0000016		330	430	1		新日鐵	2100	生産定尺	6	1524	3048	5 x 10	0	0	80	17520		81	17739	0	
1401115		330	430	1		新日鐵	2100	生産定尺	9	914	1829	3 x 6	0	0	99	11682		15	1770	0	
1771622		330	0	1		新日鐵	2100	生産定尺	9	914	1829	3 x 6	0	0	1	118		0	0	0	
0000021		330	430	1		新日鐵	2100	生産定尺	9	1219	2438	4 x 8	0	0	194	40740		5	1050	0	
0483655		330	430	1		新日鐵	2100	生産定尺	9	1524	3048	5 x 10	0	0						0	
1770845		330	0	1		新日鐵	2100	生産定尺	9	1524	3048	5 x 10	0	0						0	
0000025		330	430	1		新日鐵	2100	生産定尺	12	914	1829	3 x 6	0	0	2					0	
0000027		330	430	1		新日鐵	2100	生産定尺	12	1524	3048	5 x 10	0	0	3	1314		0	0	0	
0000030		330	430	1		新日鐵	2100	生産定尺	16	914	1829	3 x 6	0	0	24	5040		24	5040	0	
0000032		330	430	1		新日鐵	2100	生産定尺	16	1524	3048	5 x 10	0	0	10	5830		0	0	0	

図2-9　選択行着色の処理コード部分抜粋

TAccessDBGridを宣言

```
type
  TAccessDBGrid = class(TCustomDBGrid);
```

（OnDrawColumnCell内に記述）

```
with TAccessDBGrid(Sender) do          //選択行を着色したい
 begin
 //描画行と現在行が一致するとき色を設定
 if (DataLink.ActiveRecord = (Row-1)) then   //ActiveRecordは描画している行（0から）,Row は選択行（1から）
   begin                                     //;対象列を指定
       if column.Field=mainDBGrd[n].SelectedField then
       mainDBGrd[n].Canvas.Brush.Color :=  clLtGray  ;

       mainDBGrd[n].Canvas.Font.Color  :=  clBlue;
       mainDBGrd[n].Canvas.FillRect(Rect);
   end;

 end;
mainDBGrd[n].DefaultDrawColumnCell(Rect,DataCol,Column,State);
```

図2-10　複数種グラフ連続作成のためのコンポーネント接続イメージ

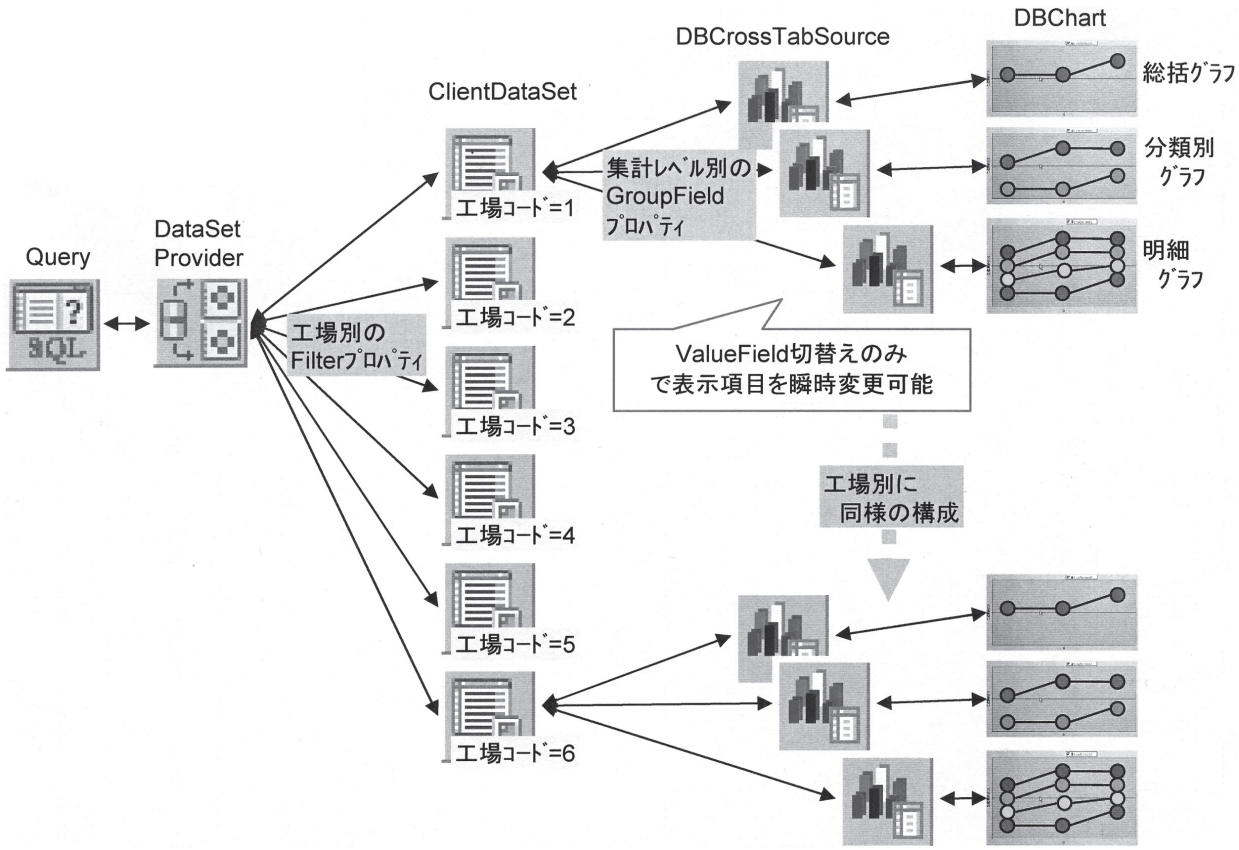


図2-11　複数種グラフ連続作成の基本処理部分

```
for i:=1 to High(DailyPTrendCDS) do    //工場別のClientDataSet毎に処理
begin
 DailyPTrendCDS[i].Filtered:=False;
 DailyPTrendCDS[i].Active:=False;

 DailyPTrendCDS[i].Active:=True;   // 各ClientDataSetをOPENしなおす、ﾌｨﾙﾀｰも再度有効化
 DailyPTrendCDS[i].Filtered:=True;

  for j:=1 to 3 do    //総括、ｸﾞﾙｰﾌﾟ別、明細の3ﾚﾍﾞﾙで作成
  begin
  DailyPTrendTabSrc[i,j].Active:=false;  //各DBCrossTabSource は工場、集計ﾚﾍﾞﾙ毎の2次元配列
  DailyPTrendTabSrc[i,j].Active:=true;   // 各DBCrossTabSourceをActiveにしなおす
   //ここにグラフ描画等に関する各種処理を記述
  end;
end;
```

図2-12　複数種グラフの表示項目連続切替えの処理例

```
for i:=1 to High(DailyPTrendCDS) do    //工場別のClientDataSet毎に処理
begin
 DailyPTrendCDS[i].Filtered:=False;

  for j:=1 to 3 do    //総括、ｸﾞﾙｰﾌﾟ別、明細の3ﾚﾍﾞﾙで作成
  begin
  DailyPTrendTabSrc[i,j].Active:=false;
  DailyPTrendTabSrc[i,j]. ValueField:=hyoujiItem ; // 予め設定した項目（ここではhyoujiItem）に変更
  DailyPTrendTabSrc[i,j].Active:=true;   // 各DBCrossTabSourceをActiveにしなおす
   //ここにグラフ描画等に関する各種処理を記述
  end;
end;
```

図2-13　グラフ一括表示の例

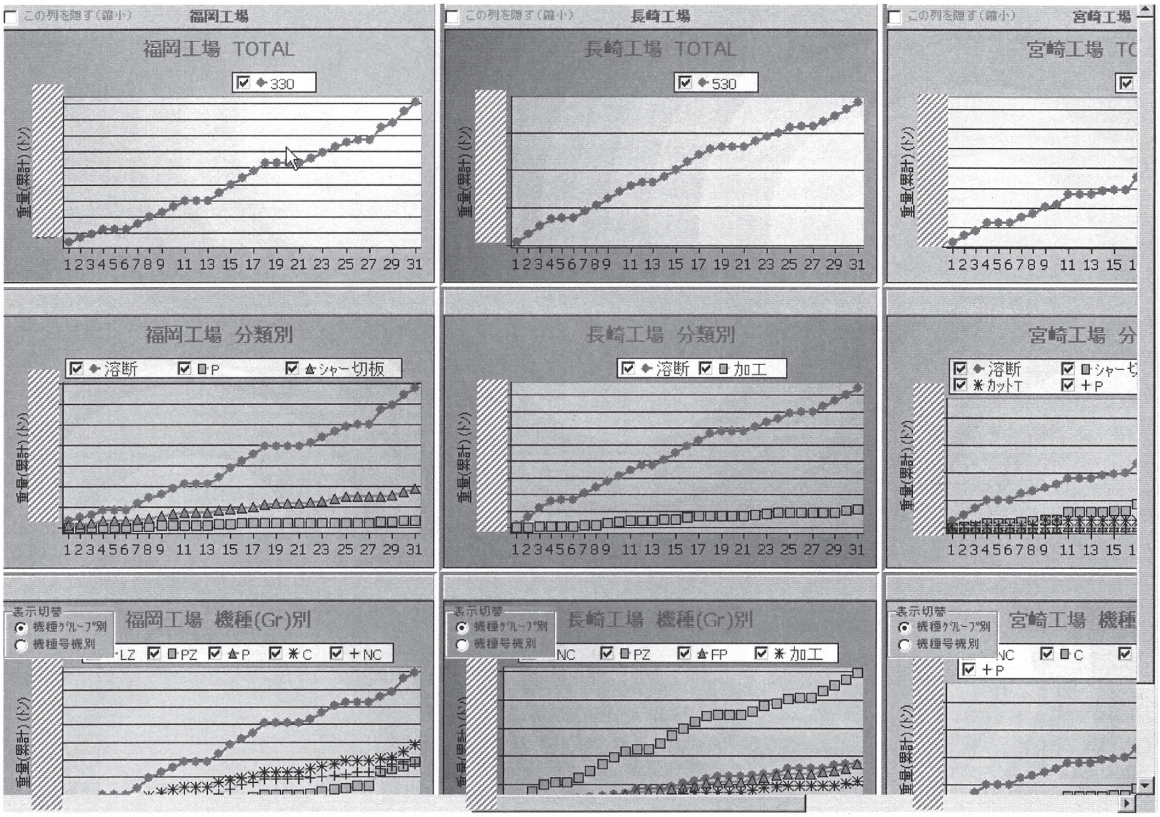


図2-14　グラフの表示項目一括変更の例

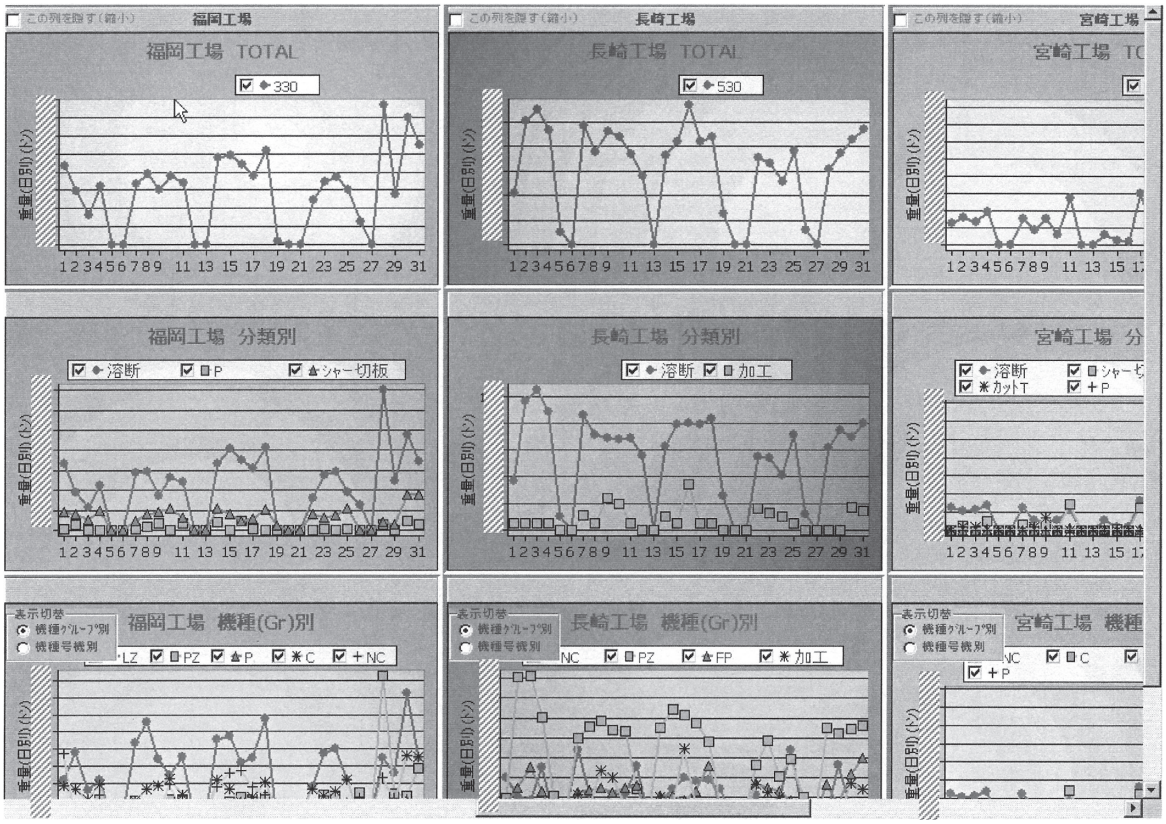


図2-15　TeeCommander呼び出しの例

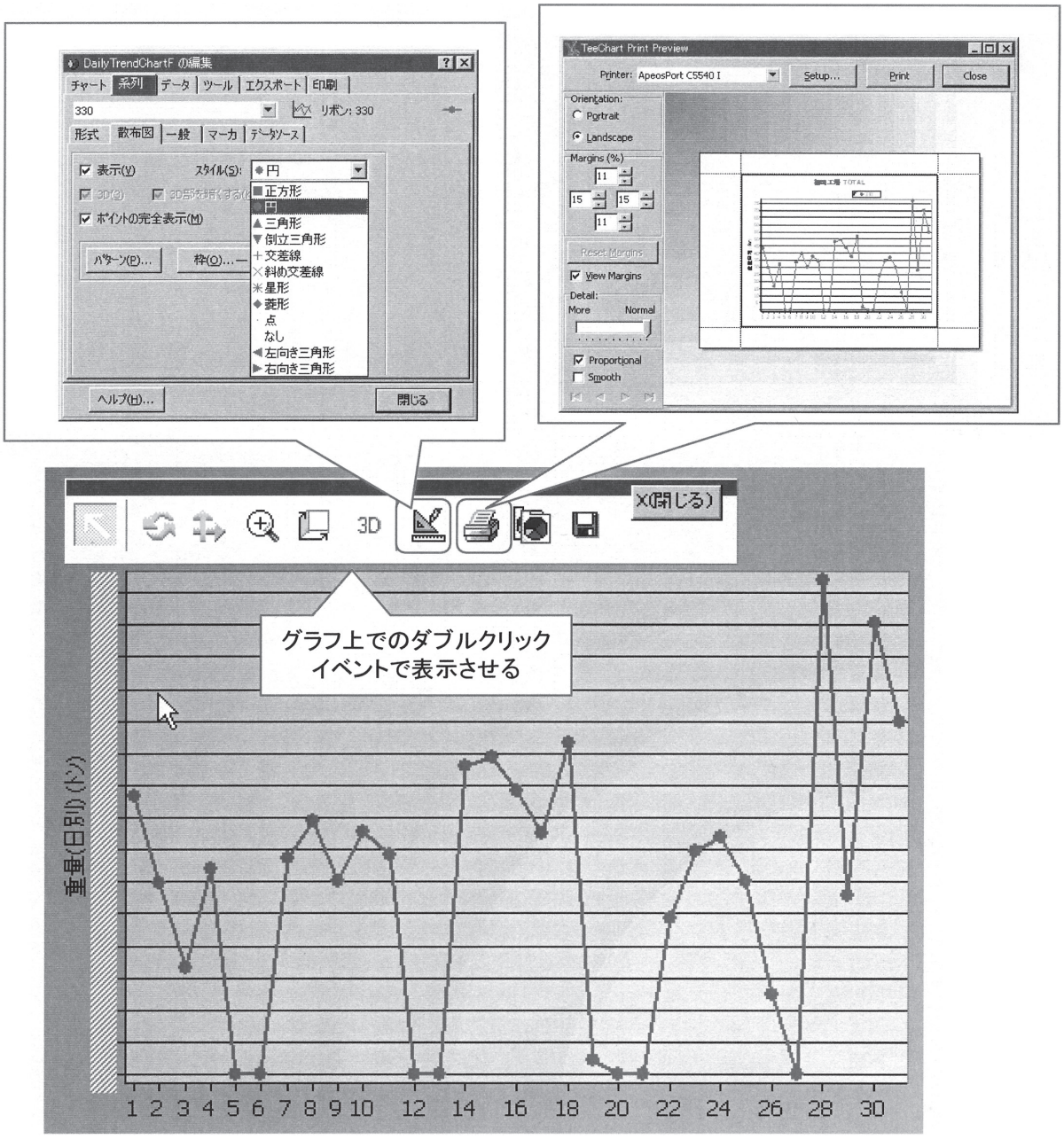
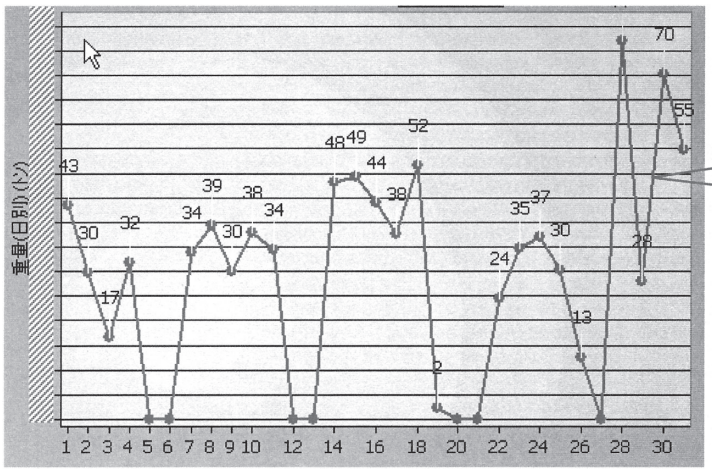


図2-16　系列要素クリックでのデータ表示例-1



要素クリックでプロットの値を
表示（再クリックで非表示）

図2-17　系列要素クリックでのデータ表示例-2

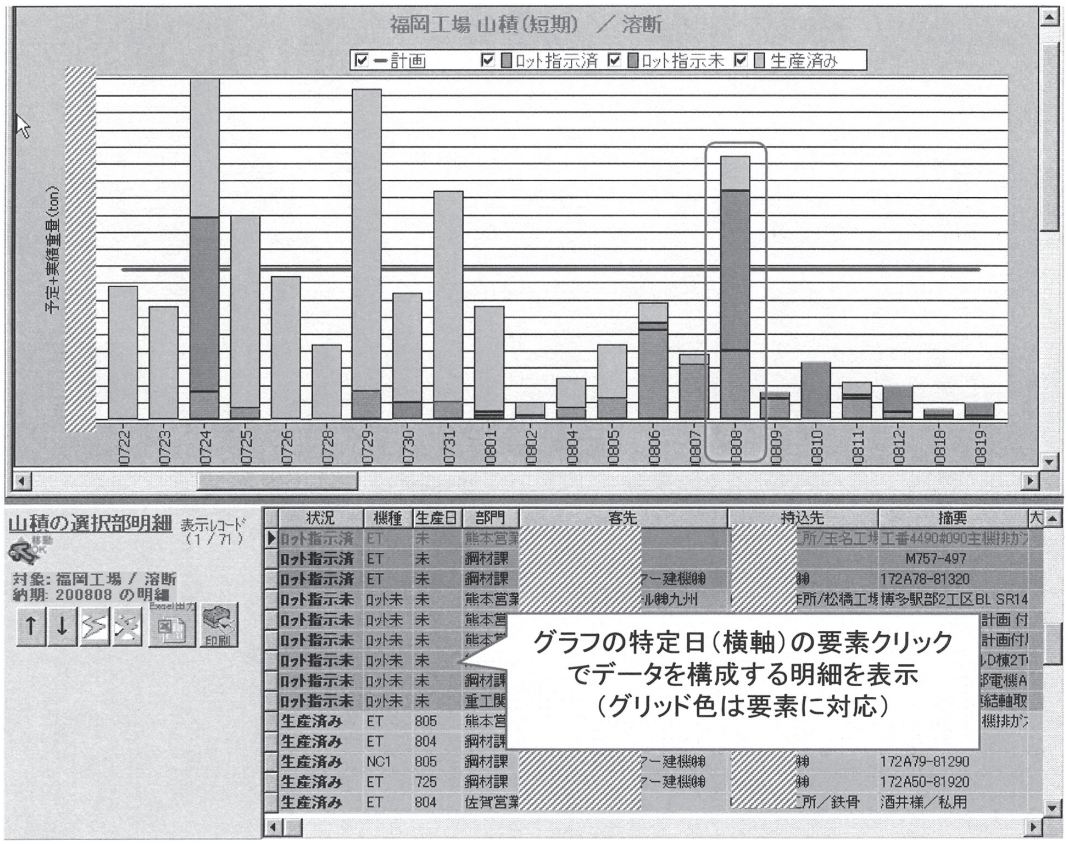


図2-18　系列要素クリックでのデータ表示例-3

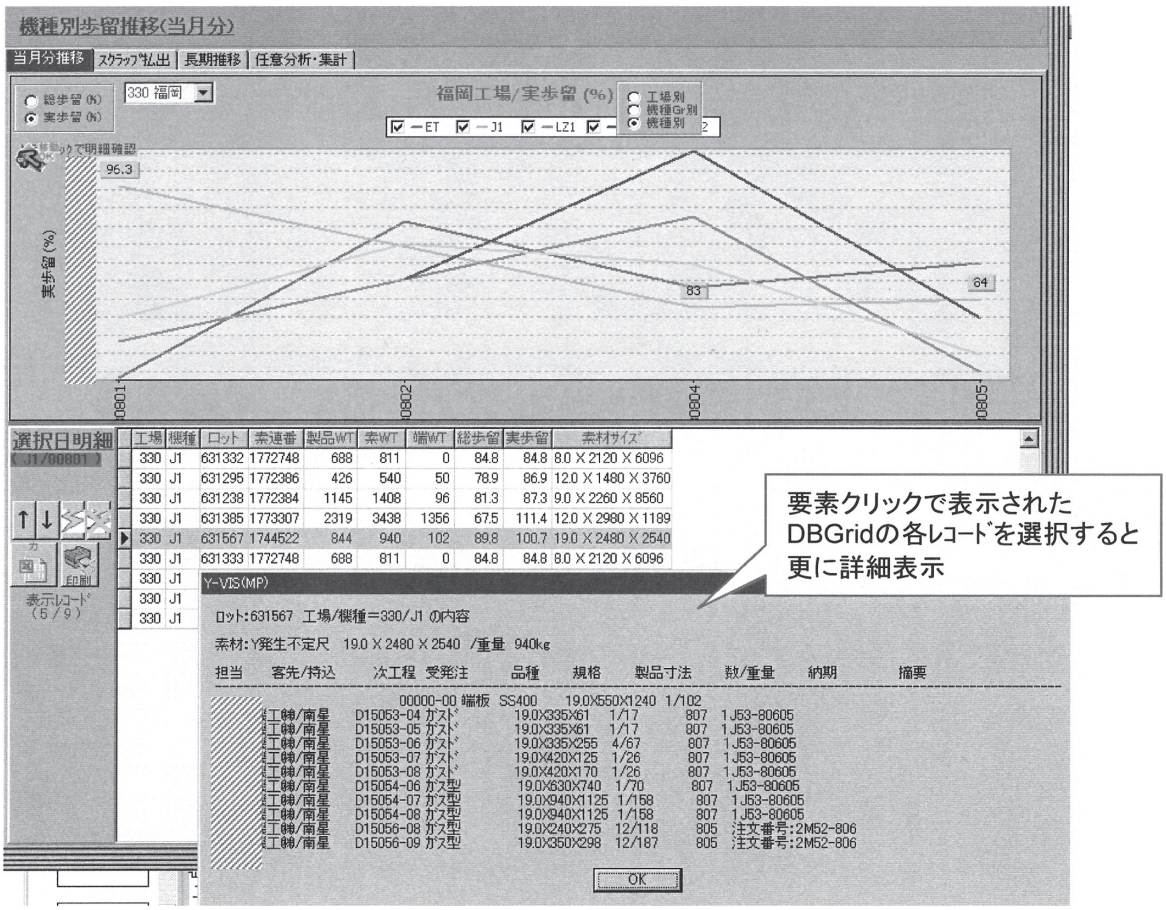


図2-19　マニュアル呼び出し方法例

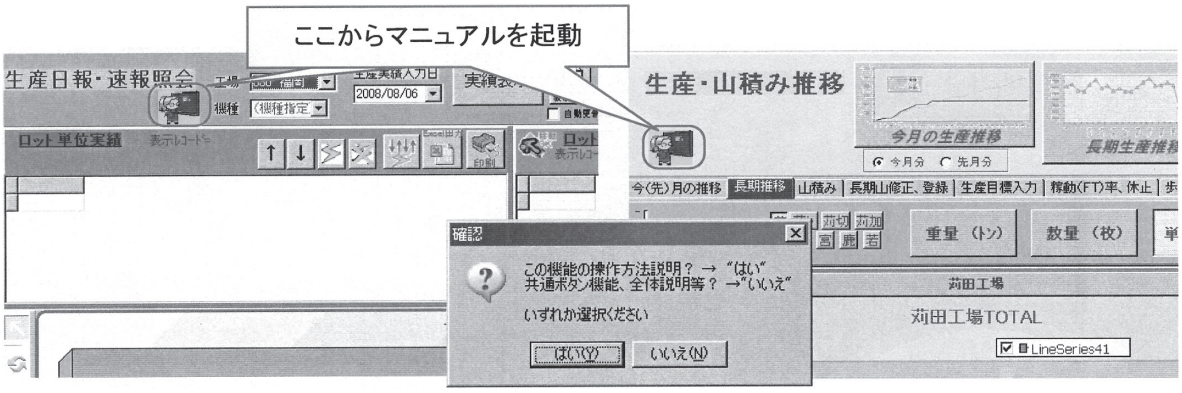


図2-20　DHTML形式のシミュレーション型マニュアル初期画面例

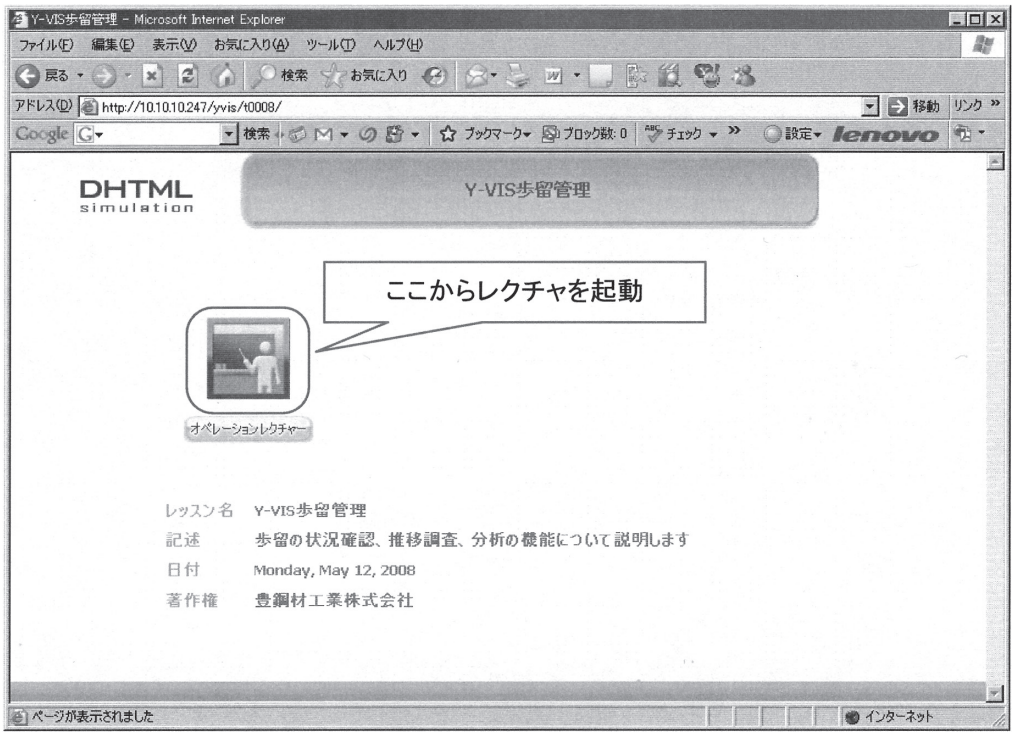


図2-21　DHTML形式のシミュレーション型マニュアル操作例

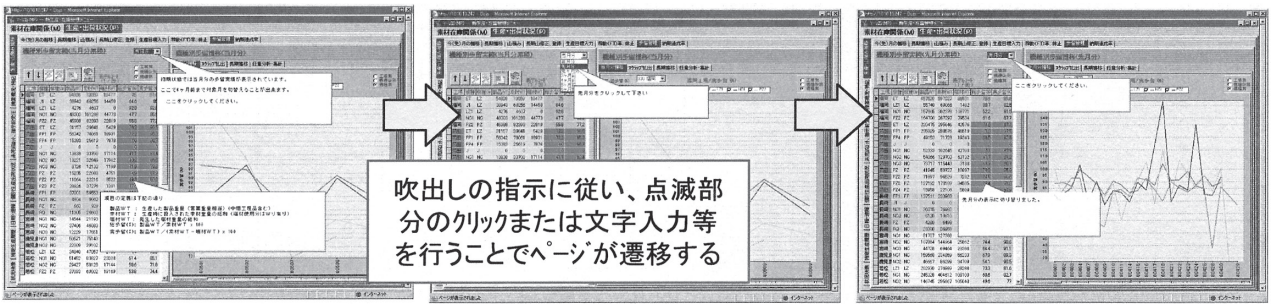


図2-22　静的HTML形式のマニュアル初期画面

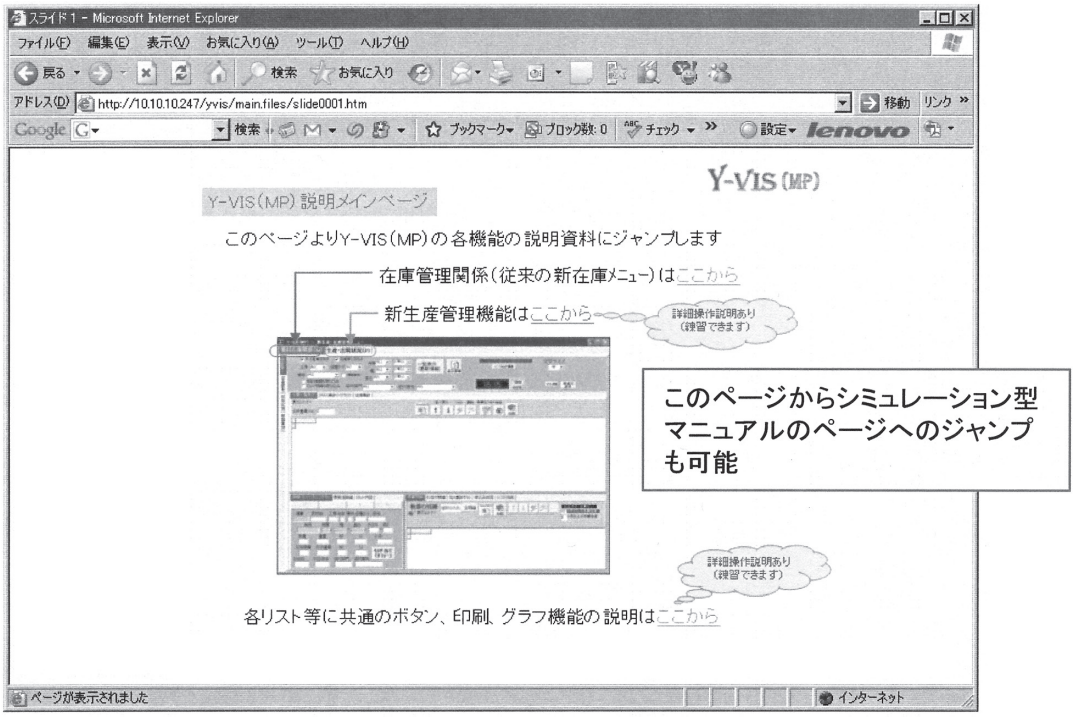


図2-23　共通機能の説明ページ初期画面

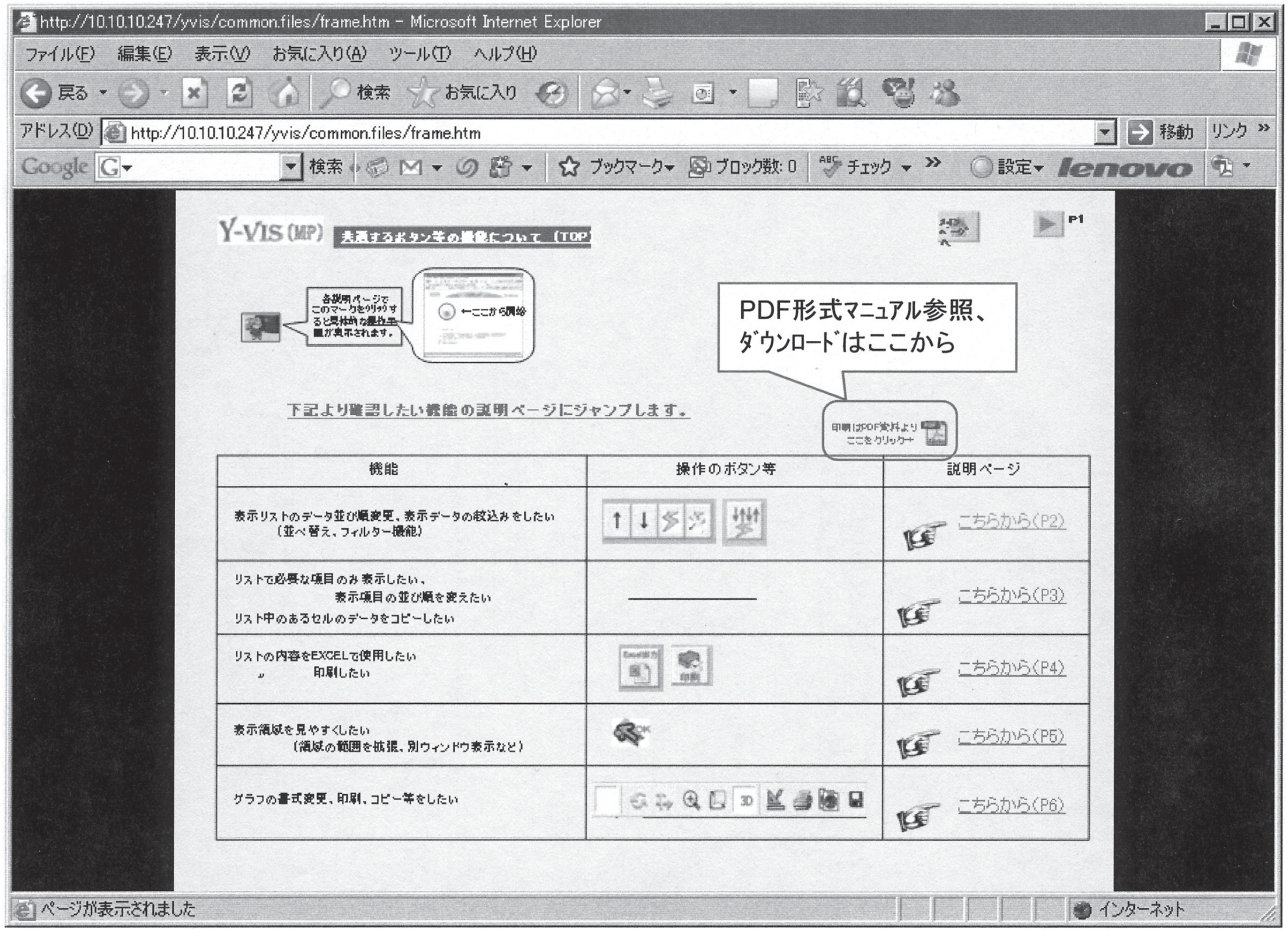


図2-24　リモートデスクトップ機能表示例

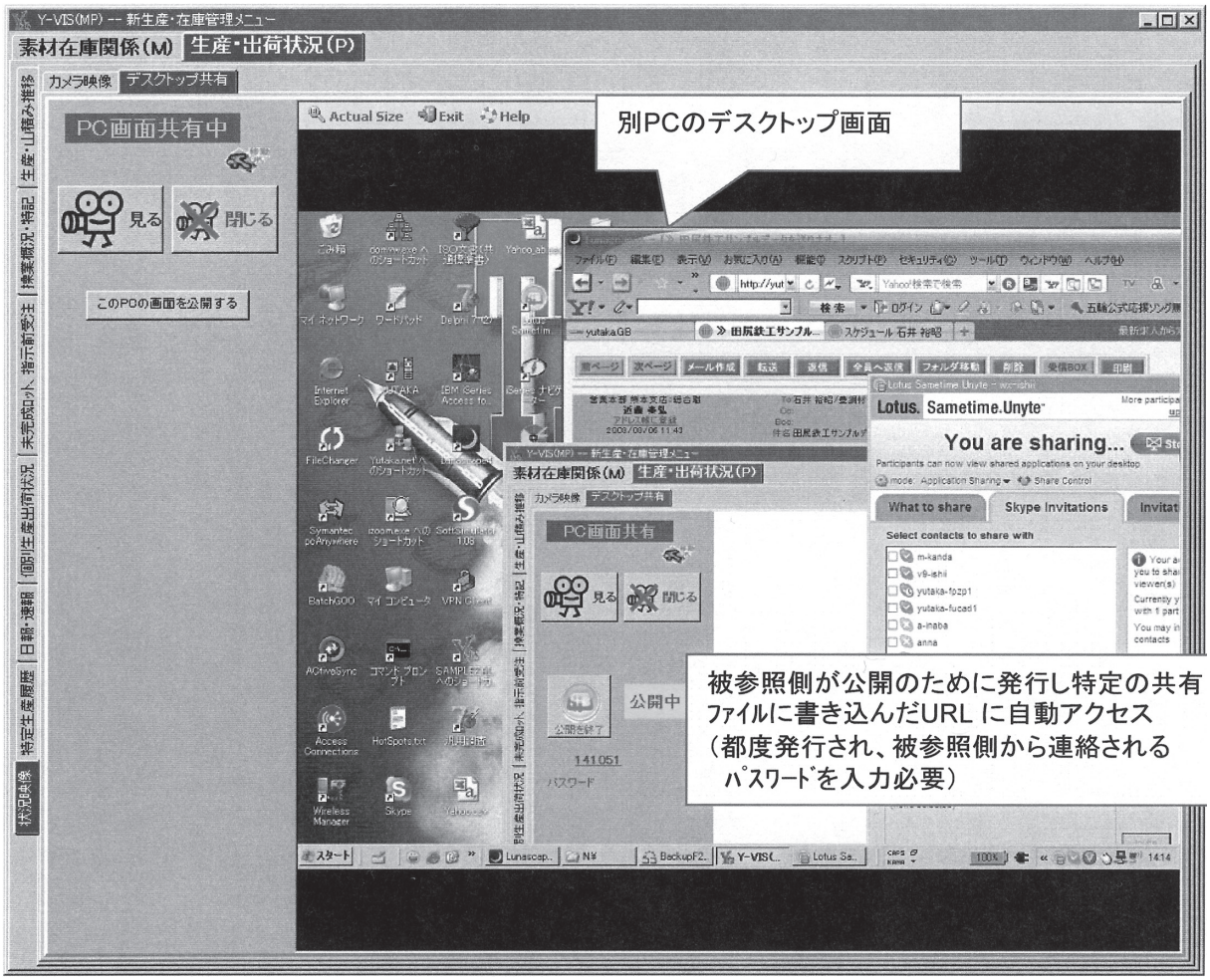


図2-25　お客様住所のGoogleMap表示処理フロー

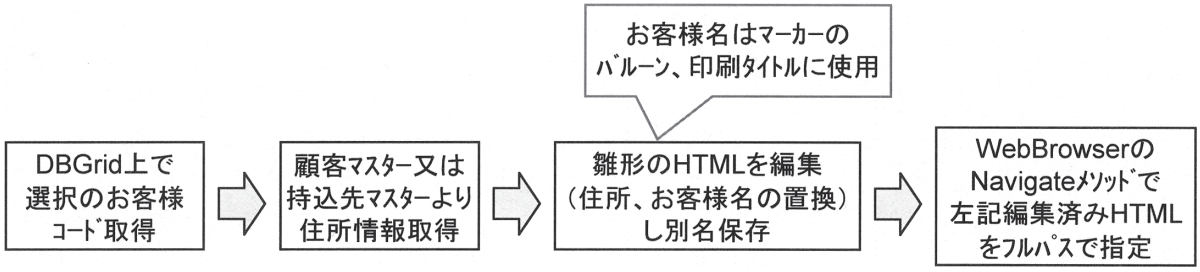


図2-26　お客様住所を表示するHTML記述部分（雛形）

```
if (geocoder) {
    geocoder.getLatLng(
      "KYAKUADD" ,
      function(point) {
        if (!point) {
          alert("KYAKUADD" + " では表示不能、下のﾎﾞｯｸｽの住所を少し短くして試して下さい(又は古い漢字修正)!!");
        } else {
          map.setCenter(point, 12);
          map2.setCenter(point, 15);
          var marker = new GMarker(point);
          var marker2 = new GMarker(point);
          map.addOverlay(marker);
          map2.addOverlay(marker2);
          marker.openInfoWindowHtml("KYAKUNAME");
          marker2.openInfoWindowHtml("KYAKUNAME");
        }
      }
    );
  }
```

KYAKUADD、KYAKUNAMEは
実際の住所、お客様名に置換される

図2-27　お客様住所のGoogleMap表示例

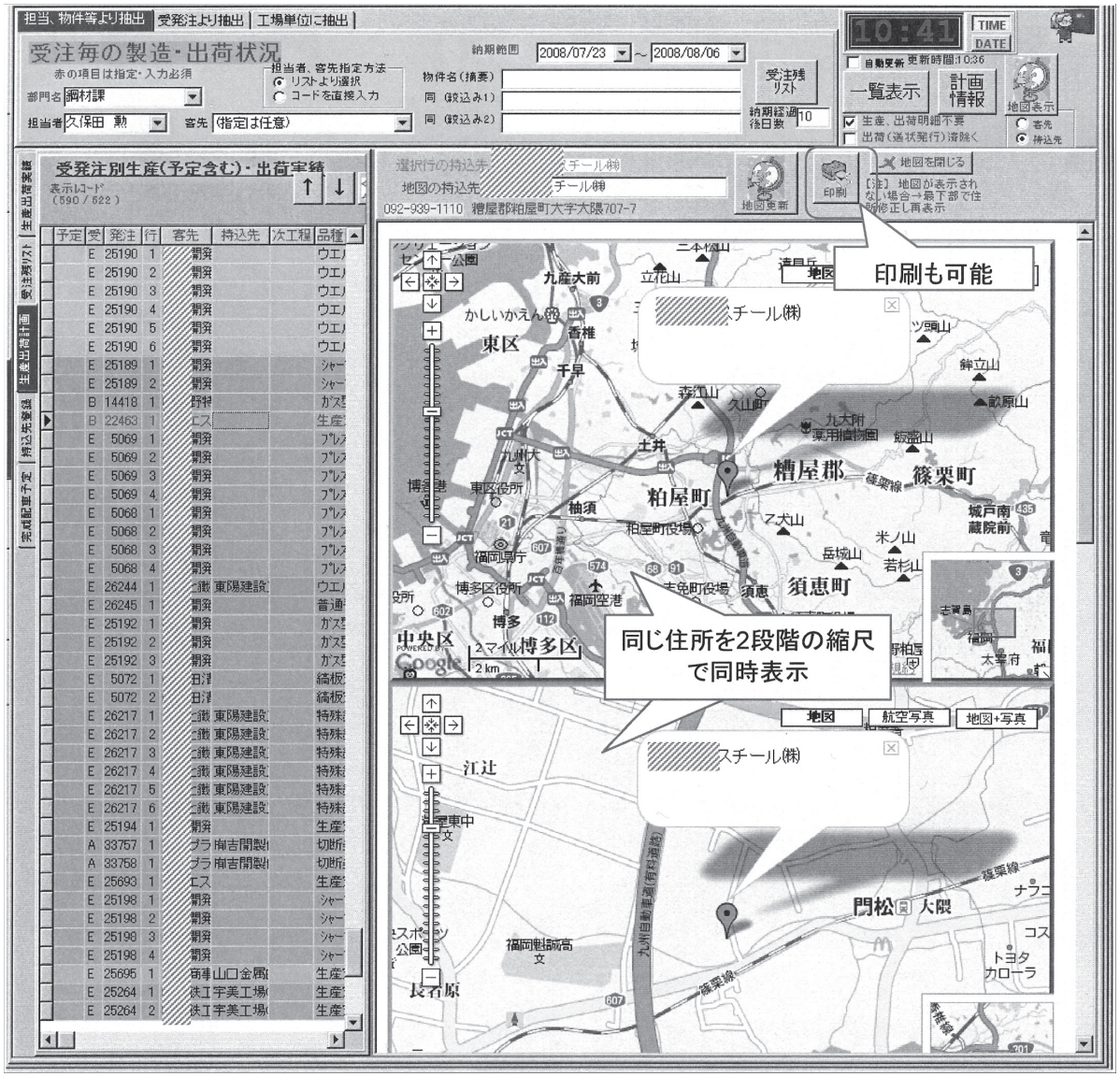


図2-28　WebBrowserの印刷処理記述

```
procedure TForm1.MapPrintBTNClick(Sender: TObject);
 var
  tmp:OleVariant;
begin
  Form1.WebBrowserMap.ExecWB(OLECMDID_PRINT, OLECMDEXECOPT_DODEFAULT ,tmp, tmp);
end;
```

図2-29　map表示不能時の住所内容編集、再試行手段

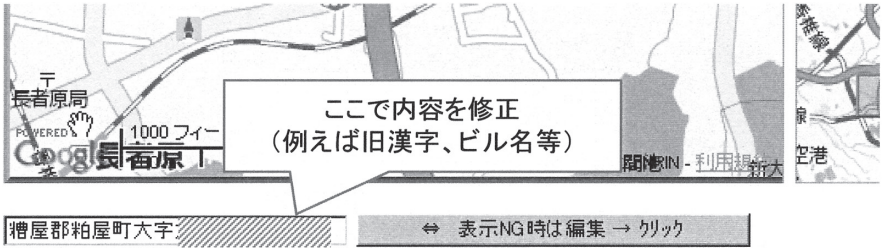


図2-30　ネットワークカメラ表示に関する管理イメージ

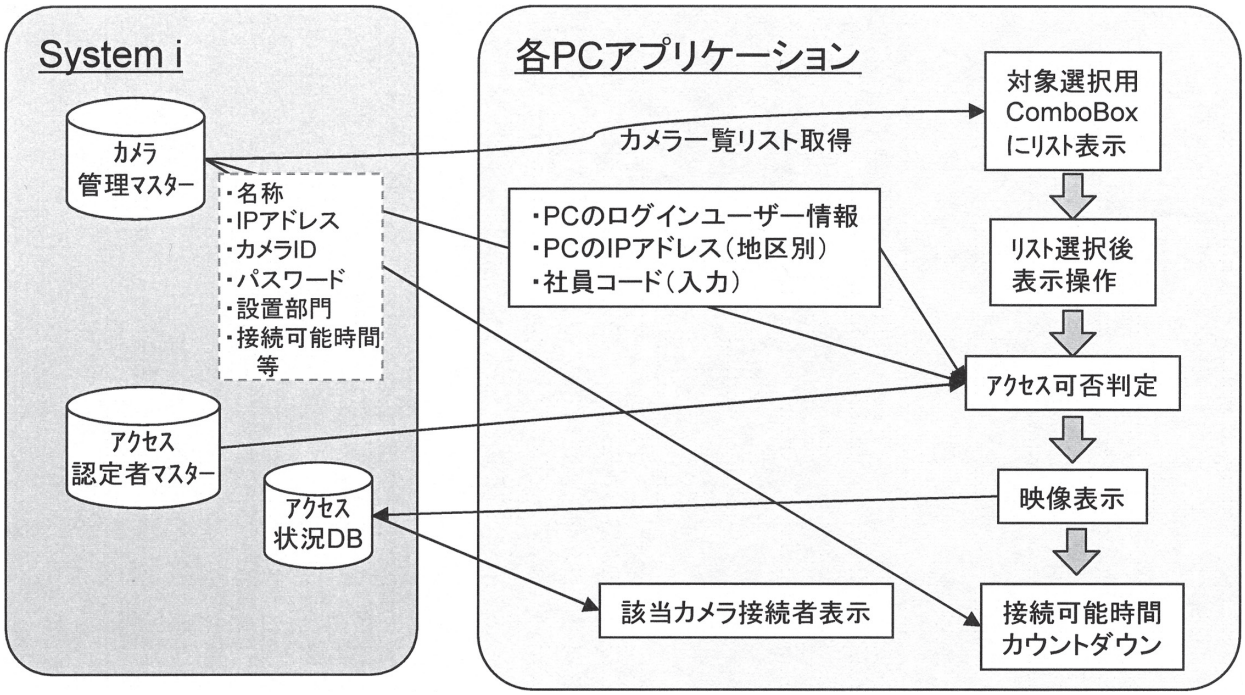


図2-31　ネットワークカメラ表示画面例

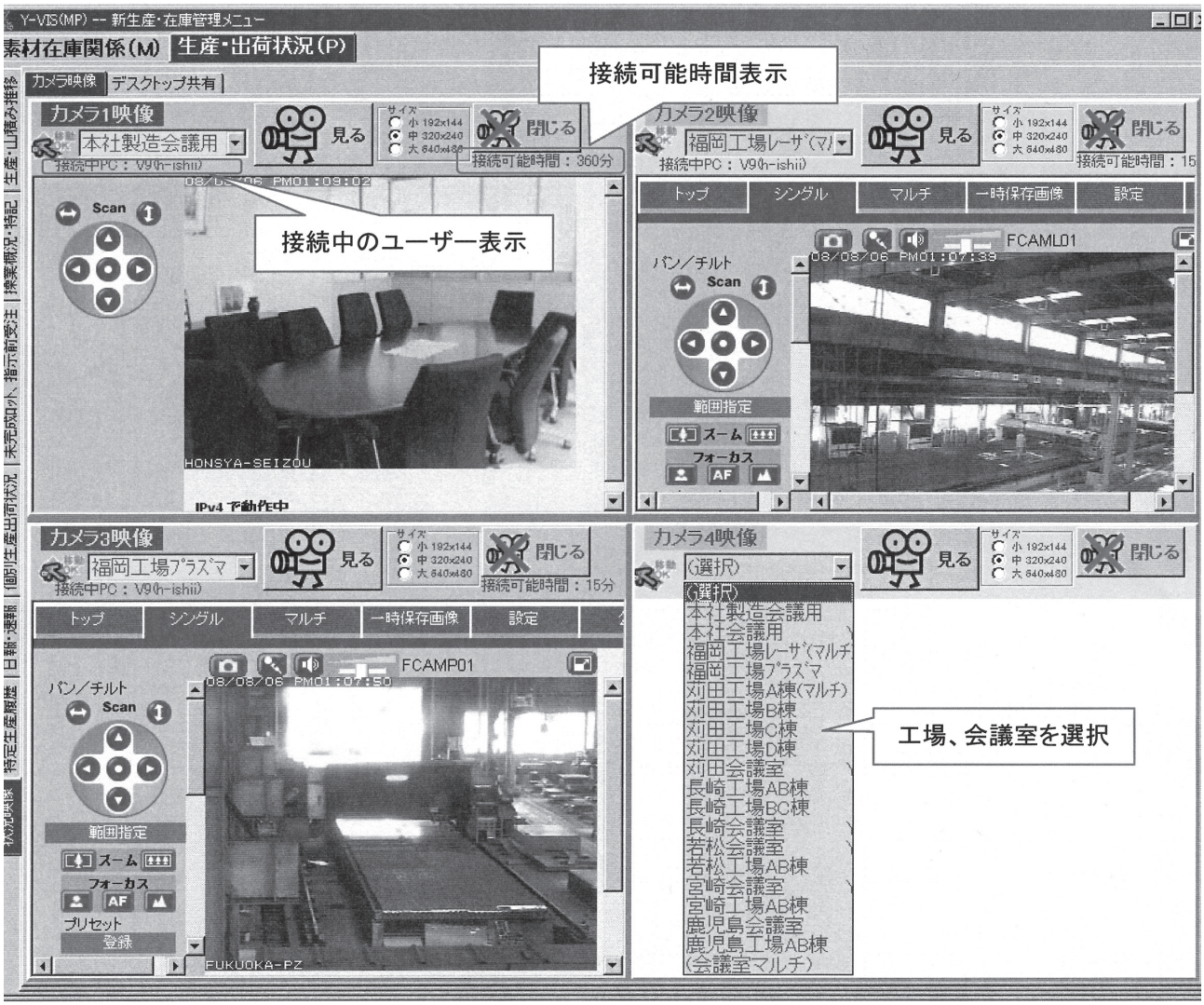


図2-32　DecisionGrid(Graph)の集計項目設定フォーム例

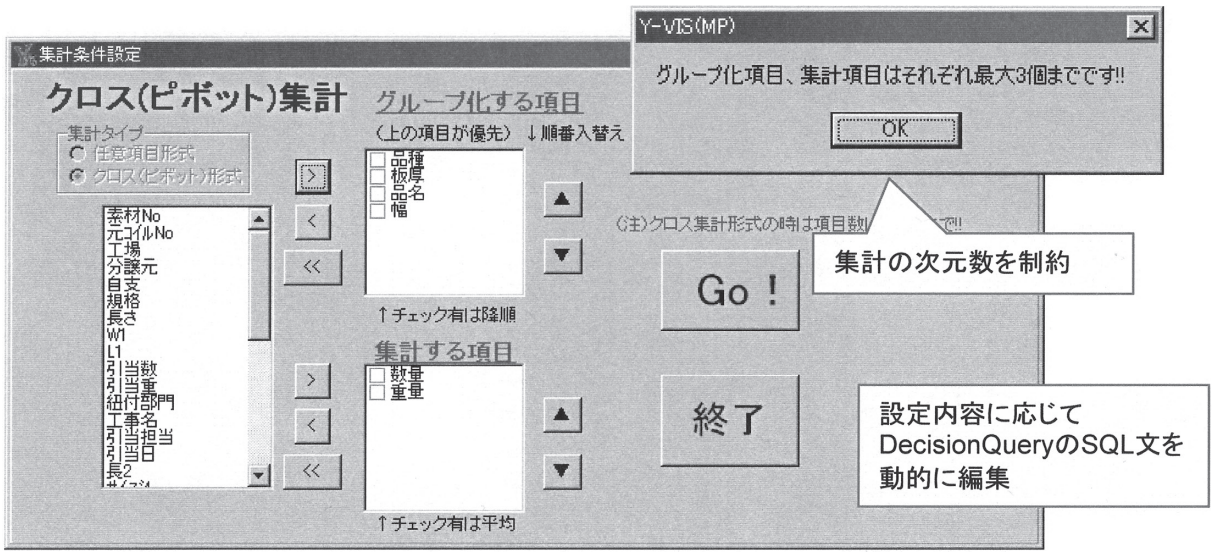
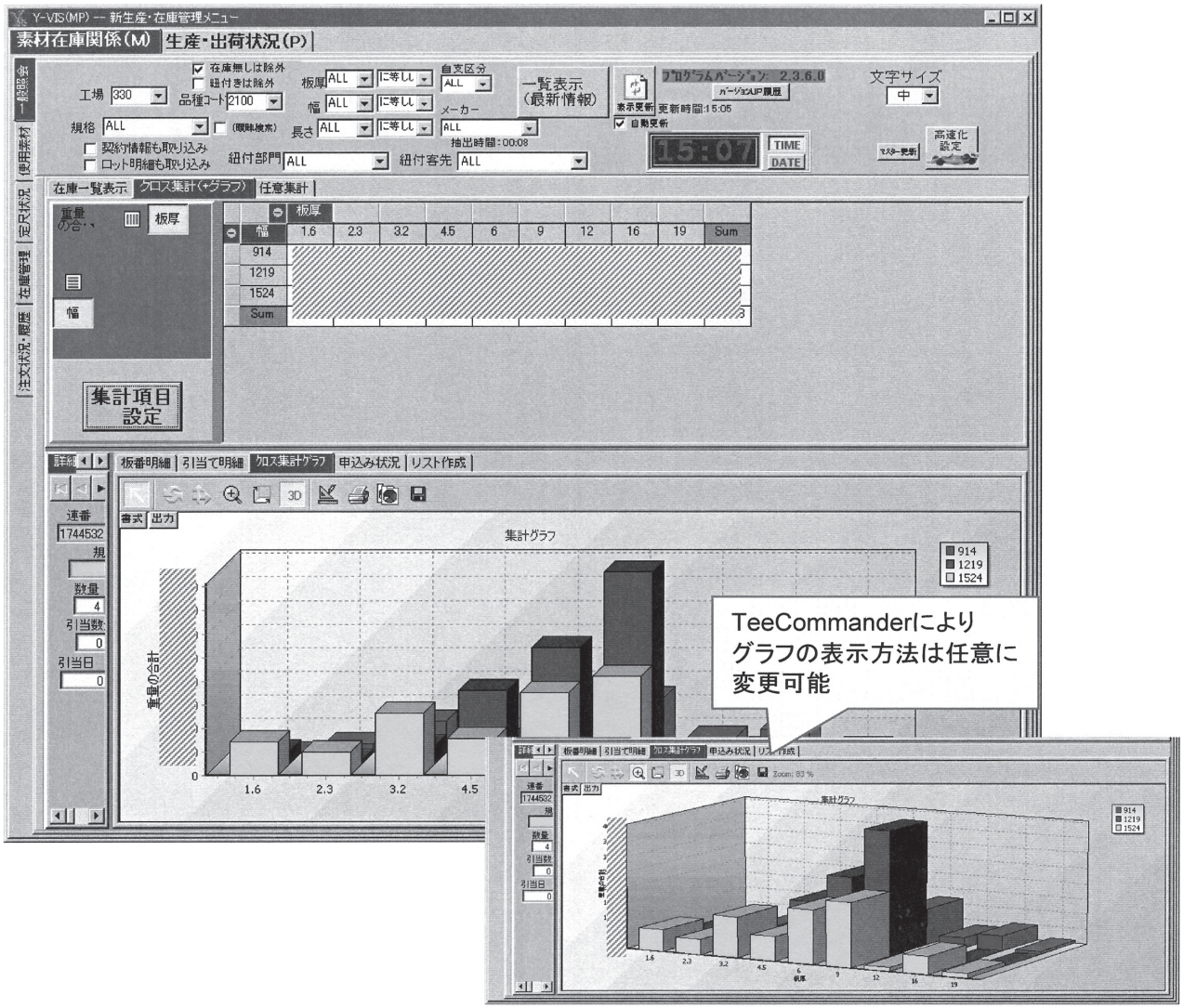


図2-33　DecisionGrid（Graph）の使用例



最優秀賞

RPGを知らなくてもここまでできる
――ユーザーサイドからのアプリケーション開発

石井 裕昭 様
豊鋼材工業株式会社
製造総括部　部長代行

豊鋼材工業株式会社
http://www.yutaka-steel.co.jp/

「鋼材のことならあらゆるニーズに応える企業」をモットーに、広島から沖縄までをカバーし、鋼材およびその他金属の加工、販売を行っている。伊藤忠丸紅鉄鋼・新日本製鐵系の会社である。

ソリューション導入の経緯と概要

豊鋼材工業株式会社の主体である溶断事業では、従来、生産に関するさまざまな情報が、紙の帳票による事後の実績入力や、あるいは担当者の経験等に基づく繁閑の判断と工程管理といったことに依存していた。したがって営業部門、経営者、さらに製造部門内からも、工場の操業状況や受注製品の進捗・計画状況はブラックボックス化して、お客様対応のレスポンスやさまざまな判断に支障をきたしていた。

このような状況を打開するため、平成17年より工場内の無線LAN強化をはじめ、各設備へのWindows CE端末、無線ハンディターミナル、およびラベルプリンタの配備と帳票へのバーコード出力等の基盤整備を開始した。【図1-1】

Delphi/400は、この基盤整備でリアルタイムに収集される情報を定量的かつ直感的に利用する手段として導入された。本アプリケーションは、生産・出荷計画、進捗、履歴、山積み等の情報をさまざまな角度から提供する参照系と、生産指示や計画作成等の更新系の処理を備えている。

開発の独創性・創意工夫

RPGを使用できるシステム室要員数が少なく、本アプリケーションで実現したい数多くの機能について、データ抽出の条件・項目等の仕様が事前に確定しにくいこともあり、開発はユーザーに近い立場の者が主担当で行った。

そのため、データ抽出はBDEのQueryで、SQL文を動的に生成して抽出する構成とした。だが、初めてのDelphi/400での開発ということもあって、当初は実用に堪えないものであった。

いかに一度に表示できる情報量が多くても、表示の速度が実用レベルになければ、また操作性等で5250での参照に対して大きな優位性がなければ、ユーザーには使われない。

そこで、次の対応を順次実行し、実用レベルに到達させた。

① Query → DatasetProvider → ClientDatasetの組み合わせで、抽出したデータに対して、ローカルでのフィルタリング、並べ替えのボタン、DBGridをマウス操作のみで実現するという機能を実装
② SQLのパフォーマンスについては、i-SeriesナビゲータのVisual Explainにより、推奨索引確認および索引自動生成での改善
③同じくVisual Explainにより、結合キーの型の妥当性確認と調整
④ SQL実行時のSQL文抽出機能とSQL動作検証用アプリにより、デバッグとチューニング
⑤関連データの一括抽出とマスターデータ参照による連携機能で、同時参照（マスター⇔詳細⇔最詳細）【図2-1】
⑥ SQL実行待機の体感時間の軽減（ProgressBar、AVI動画再生等）
⑦ QueryのOnCalcFieldsイベントでの処理により、SQLの速度低下の抑制

図1-1　生産管理システム再構築イメージ

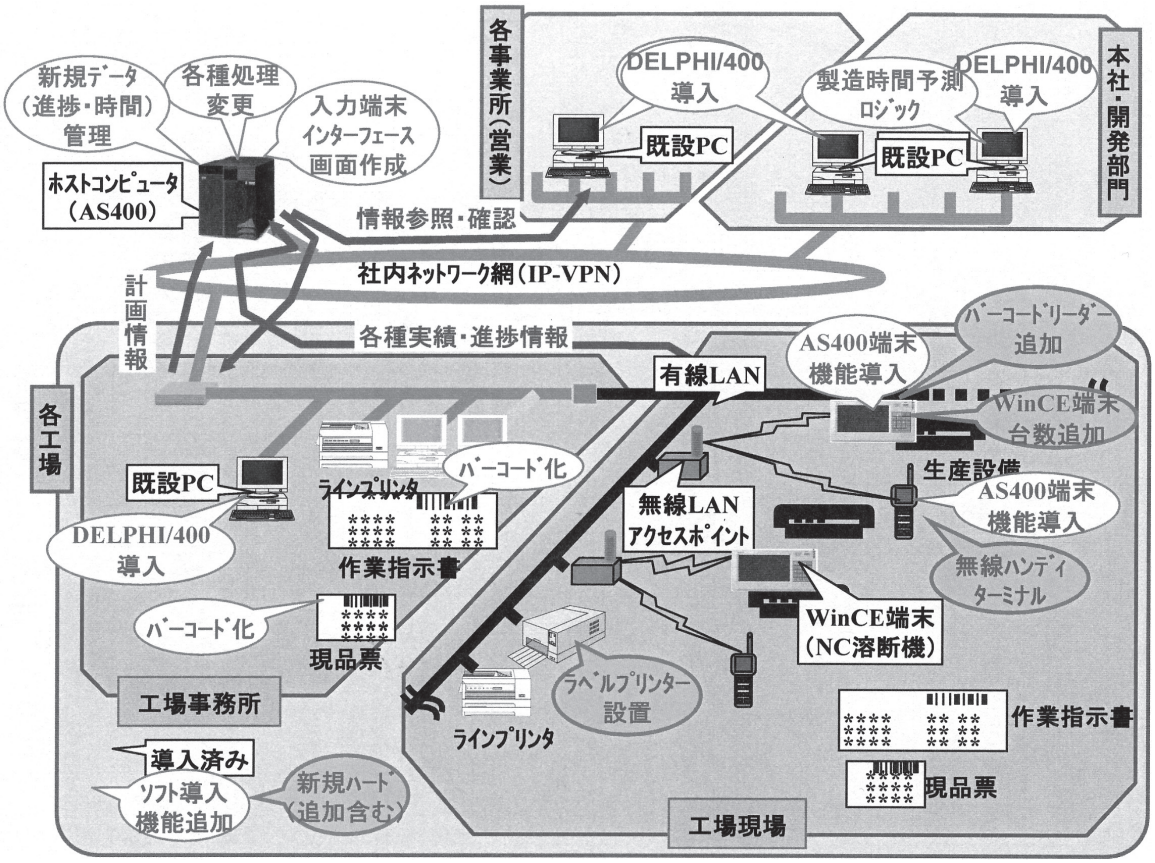
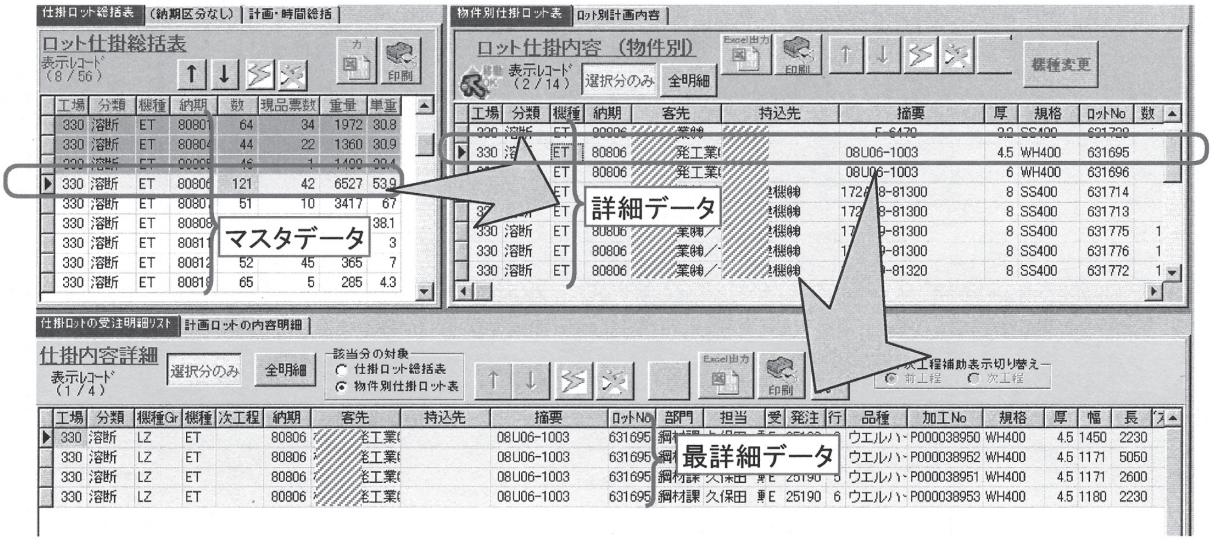


図2-1　マスタ⇔詳細⇔再詳細のデータ連携例



⑧動的に内容を変化させるcomboBox用のデータベースの、ローカル化と定期更新
⑨一定の未操作時間経過で、自動的に表示情報を更新
⑩Gridに表示される不要項目の非表示化､並び順変更の可能化｡その条件は､次回起動時にも保持（Iniファイルに配列情報を書き込み）

実用レベルに向けた改善ポイント

アプリケーション開発の独創性・創意工夫について詳細を述べる。

①毎回、抽出条件を設定しSQL処理を行うよりも、ある程度の範囲で抽出し、ローカルでキャッシュされたデータに対してフィルター、並べ替え、検索等を瞬時に実行可能にすることにした。これにより、思考の中断も少なくできる。そのために、ClientDatasetの使用を基本とした。

抽出した結果の並べ替えは、任意の項目の組み合わせで可能であり、優先順が明示できるようにしている。また、フィルター機能についても、複数条件の組み合わせが可能で、条件とフィルター中であることがわかるようにした。【図2-2】

なお、DBGridの各列のタイトル、巾等については、開発のメンテナンス性を考慮した。Queryの静的項目のみの設定で、その内容が自動的にClientDataSetに引き継がれるように、OnAfterOpenイベントに汎用処理を記述した。【図2-3】【図2-4】

またDBGrid、Query、ClientDataSet、フィルター用ボタン等、関連する一連のコンポーネントを配列化した｡とともに、Tagプロパティ等の活用により、呼び出し元の識別を汎用化し、コードの記述を抑えた。【図2-5】【図2-6】

②③IBMのIBM i講座「SQLパフォーマンス･チューニングの基礎」を受講し、IBM iに同梱されているi-Seriesナビゲータを開発用PCに導入した。【図2-7】

これにより､組み込むSQLのパフォーマンスについて検証を行うことができ、適正な索引の確認と索引の生成が一連の操作で可能となった。結合キーの型不一致のチェック等も可能であり、その際にはiSeries DB2のSQL解説書を参考に型の変換を行った。【図2-8-1】【図2-8-2】

i-Seriesナビゲータでは、GUIの画面で既存ライブラリに新規にテーブル、ビュー（論理ファイル）、索引等を簡単に作ることも可能であるため、情報システム室に依存せず、新たな管理項目を含むデータベースを作れるため有効である。【図2-9】【図2-10】

④開発時のテストにおいて、データが正しく抽出されない、またはエラーが発生する場合、実アプリケーションのコード修正を繰り返すとコンパイルの時間ロスが大きい。また、どのようなSQL文が実行されているかも、コードからは読み取りにくい。そのため、該当SQLの配列番号指定で実行されたSQL文を、テキストで読み出せる仕組みを作った。

また、任意のSQL文での抽出結果を確認するテスト専用のアプリケーションを、別途準備し、SQLのテストが簡単にできるようにした。【図2-11】【図2-12】【図2-13】

⑥1回の操作で複数のQueryをOPENする際には、その変わり目でProgress Barの Positionを変更し、SQL処理が進んでいることをユーザーに示す、ということを可能にした。

一方、単体の場合、個々の処理時間が長いと本当に処理が進んでいるか不安になり、体感時間も長くなる。そのため、動画を指定のPanel上で再生して、ユーザーの手待ち感の軽減を試みた。【図2-14】【図2-15】

⑦SQL文には文字数の制限もあり、CASE文等を多用するとレスポンスが低下するケースも多い。また、複雑な処理の記述が困難となるため、内容に応じてQueryのOnCalcFieldsイベントを用いた。

⑧抽出条件の設定では、ユーザーの好みに応じてコードのEdit入力､またはコードに対応する文字のComboBoxからの選択を行えるようにしている。

ComboBox使用の場合、選択肢を少なくし操作性を高めるために、例えば営業担当者のComboBoxの内容は、部門のComboBoxの選択結果に応じて動的に変更される。【図2-16】【図2-17】【図2-18】

その際、IBM iの情報を参照してComboBoxを設定すると、ネットワーク、SQLの待ち時間が多少発生する。そこで、ComboBoxで多用される社員マスター、客先マスター等の更新頻度が少ないデータベースは、定期的にユーザーのPCにParadox形式でダウンロードされる仕組みを作り、ComboBox操作時の待ちをなくした。【図2-19】【図2-20】【図2-21】

⑨同一の抽出条件であれば、設定に対応して起動中に一定時間アプリケーション操作がない場合は、ClientDataSetのRefreshメソッドを実行するようにした。

これは、ApplicationEventsのOnIdle、OnMessageイベントと、TimerのOnTimerイベントで実装できる。【図2-22】【図2-23】

⑩各機能で抽出される項目は、ある程度汎用性を持たせるために多くなり、ユーザーによって不要だったり、または優先度が異なる。横スクロールの手間、同時参照性により、並べ替え、一部項目の非表示化等のカスタマイズのニーズがあった。【図2-24】【図2-25】【図2-26】

教訓と今後の予定

参照系のアプリケーションについては、RPG等の知識がなくても十分構築できる。ただし､SQLの構文､JOINキーの型の整合性等により、パフォーマンスが大きく変化する。十分な検証が必要である。

関連する各種情報を同時参照可能とすることで、5250アプリケーション以上のTotalパフォーマンスが得られる。今後は､抽出に時間を要する機能について、パフォーマンスを改善する手段をさらに追求したい。

図2-2　フィルタ、並べ替えの状況

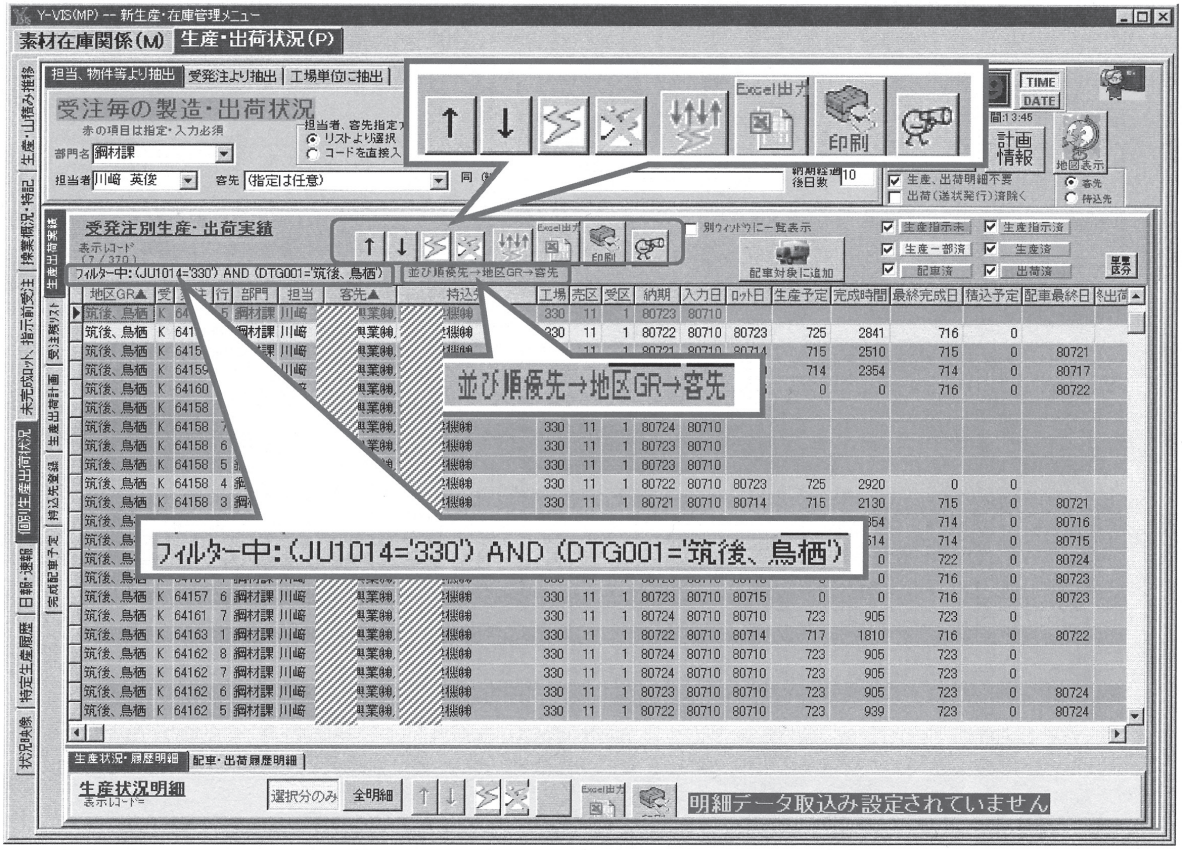
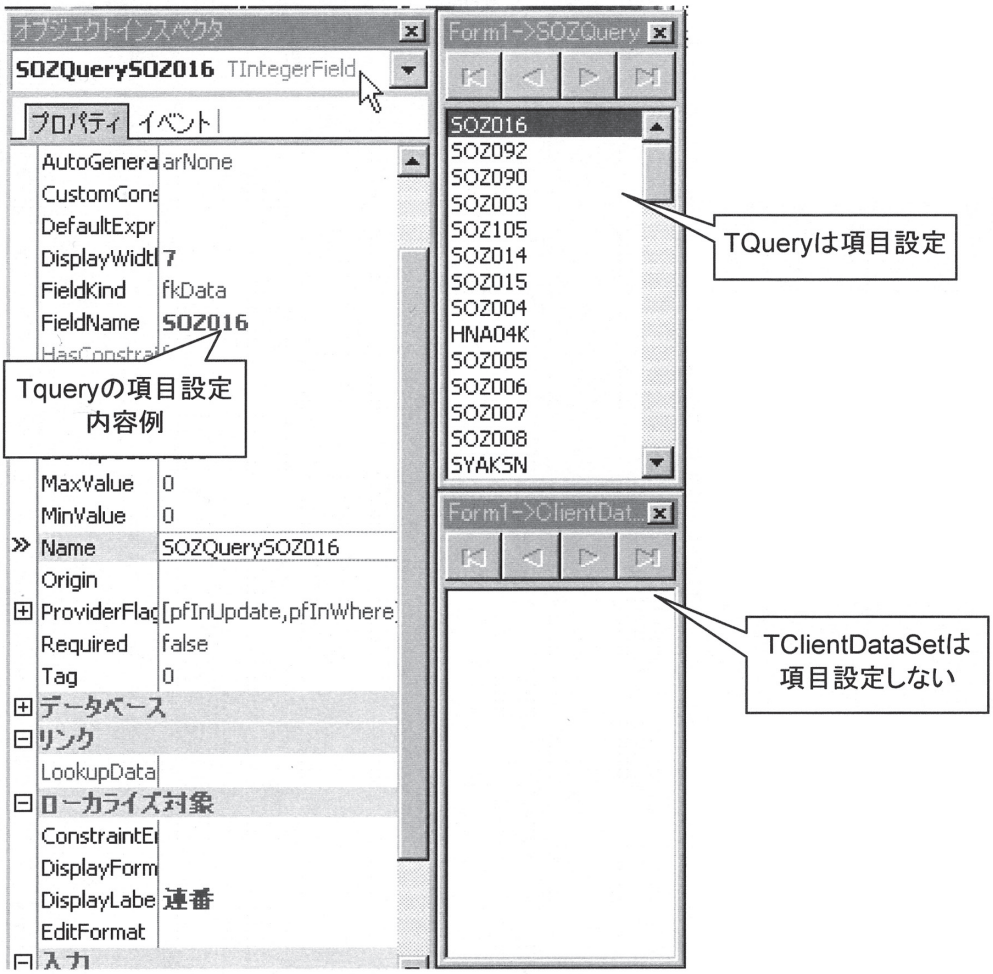


図2-3　Tqueryへの静的項目設定例



エンドユーザー評価

●従来からの使い慣れの点もあり、単純な照会については、レスポンスの点から5250画面が依然として使用される場合もある。だが、Delphi/400のアプリケーションでは、情報がより定量的に確認でき、かつ一度に多くの関連情報が参照できる。この部分に対する評価が高い。

●多目的なアプリケーションのため、使用者は役員～業務スタッフまでと幅広い。

M

図2-4　Tqueryへの静的項目設定内容をTClientDataSetに反映する処理例

```
n:=TClientDataSet(DataSet).Tag;

for i:=0 to mainQUERY[n].FieldCount-1 do
begin
  // TQueryの列名と同じ列がTClientDataSetに存在するか？
  if Assigned(Form1.mainCDS[n].FieldByName(Form1.mainQUERY[n].Fields[i].FieldName)) then
  begin
    // TClientDataSetの列に対してTQueryのDisplayLabelをコピー
    Form1.mainCDS[n].FieldByName(Form1.mainQUERY[n].Fields[i].FieldName).DisplayLabel :=Form1.mainQUERY[n].Fields[i].DisplayLabel;
    Form1.mainCDS[n].FieldByName(Form1.mainQUERY[n].Fields[i].FieldName).DisplayWidth :=Form1.mainQUERY[n].Fields[i].DisplayWidth;
    if (mainQUERY[n].Fields[i].DataType=ftFloat) Or (mainQUERY[n].Fields[i].DataType=ftInteger) then
     (Form1.mainCDS[n].FieldByName(Form1.mainQUERY[n].Fields[i].FieldName)As TNumericField).DisplayFormat
              :=(Form1.mainQUERY[n].Fields[i] As TNumericField).DisplayFormat;
    Form1.mainCDS[n].FieldByName(Form1.mainQUERY[n].Fields[i].FieldName).Visible :=Form1.mainQUERY[n].Fields[i].Visible;
    Form1.mainCDS[n].FieldByName(Form1.mainQUERY[n].Fields[i].FieldName).tag:=Form1.mainQUERY[n].Fields[i].tag;
  end;
end;
```

図2-5　機能毎の共通コンポーネントを配列化

```
mainQUERY: array[1..86] of TQuery;
mainCDS: array[1..86] of TClientDataSet;
mainSource:array[1..86] of TDataSource;
mainDBGrd:array[1..86] of TDBGrid;

OrderJunLBL: array[1..86] of TLabel;
EXCELoutBTN:array[1..86] of TBitBtn;
RecNoLBL:array[1..86] of TLabel;
FiltCondLBL:array[1..86] of TLabel;
```

図2-6　汎用処理での呼び出し元特定容易化例

```
for i:=1 to HighMainN do
  mainCDS[i].tag := i; //操作の呼び出し元の目印とする

for i:=1 to HighMainN do
  mainSource[i].tag := i; //操作の呼び出し元の目印とする

for i:=1 to HighMainN do
  mainDBGrd[i].tag := i; //操作の呼び出し元の目印とする

for i:=1 to HighMainN do
if EXCELoutBtn[i]<>Nil then
  EXCELoutBtn[i].tag := i; //操作の呼び出し元の目印とする
```

配列番号をTagﾌﾟﾛﾊﾟﾃｨに割り付け

（Formの OnCreate ｲﾍﾞﾝﾄで設定）

```
j:=(Sender as TBitBtn).Tag;

 if (mainCDS[j].Active=False) or (mainCDS[j].RecordCount=0) then
 begin
  showmessage('対象データを表示の上、実行しなさい!!');
  exit;
 end;
```

汎用化されたｴﾗｰ処理例

図2-7　iSeriesナビゲータの管理画面例（アクティブ・ジョブでDelphi/400使用状況確認）

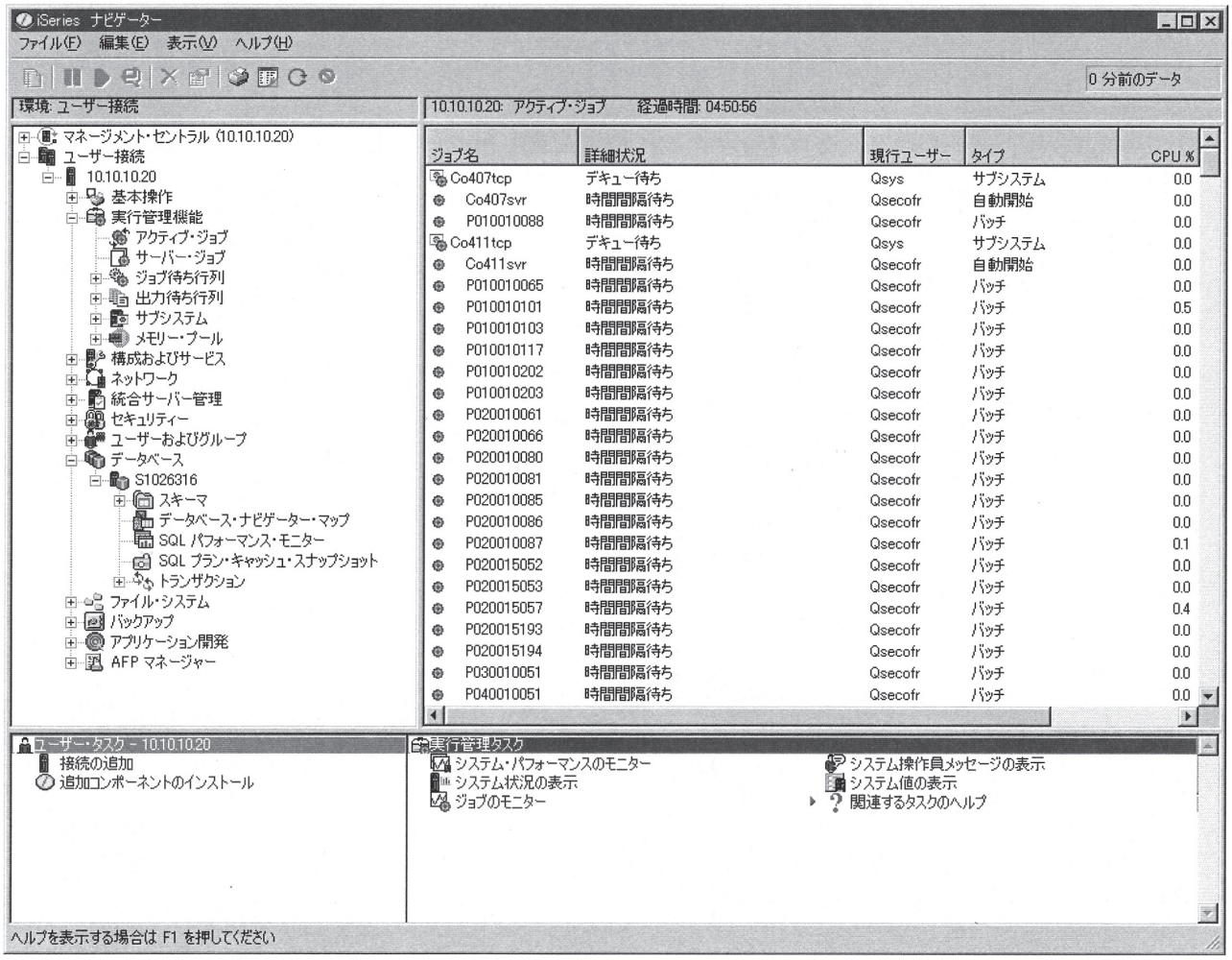


図2-8-1　iSeriesナビゲータでのSQLパフォーマンス検証例

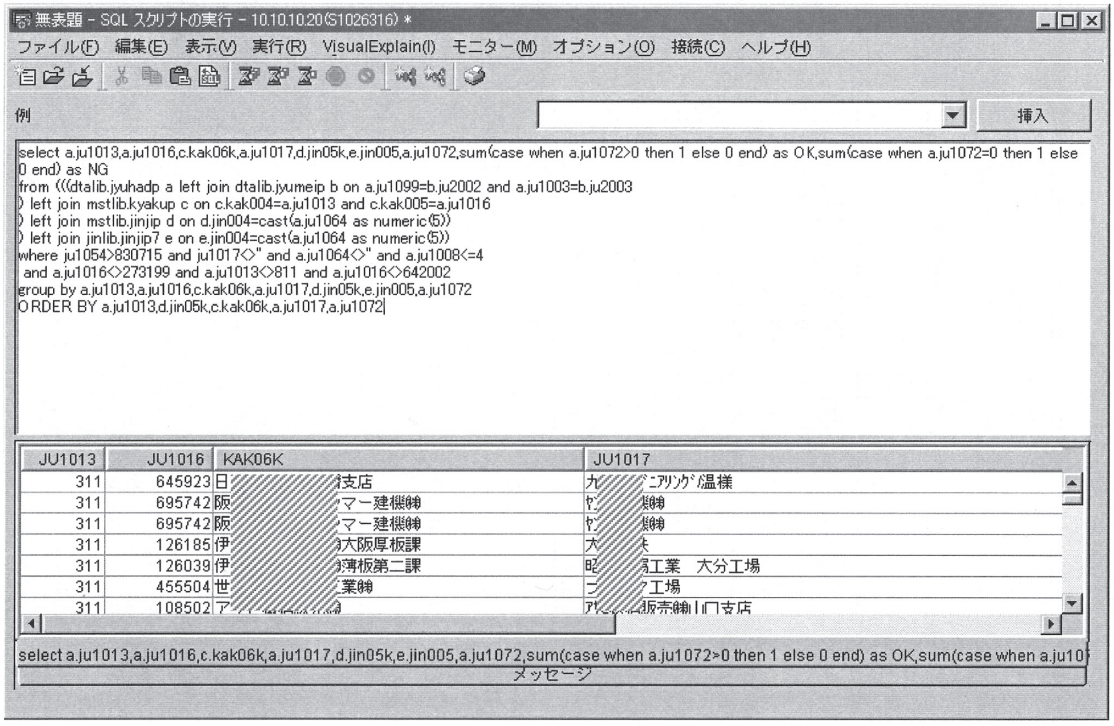


図2-8-2　iSeriesナビゲータ（VisualExplain）でのアクセスプラン、索引処理確認例

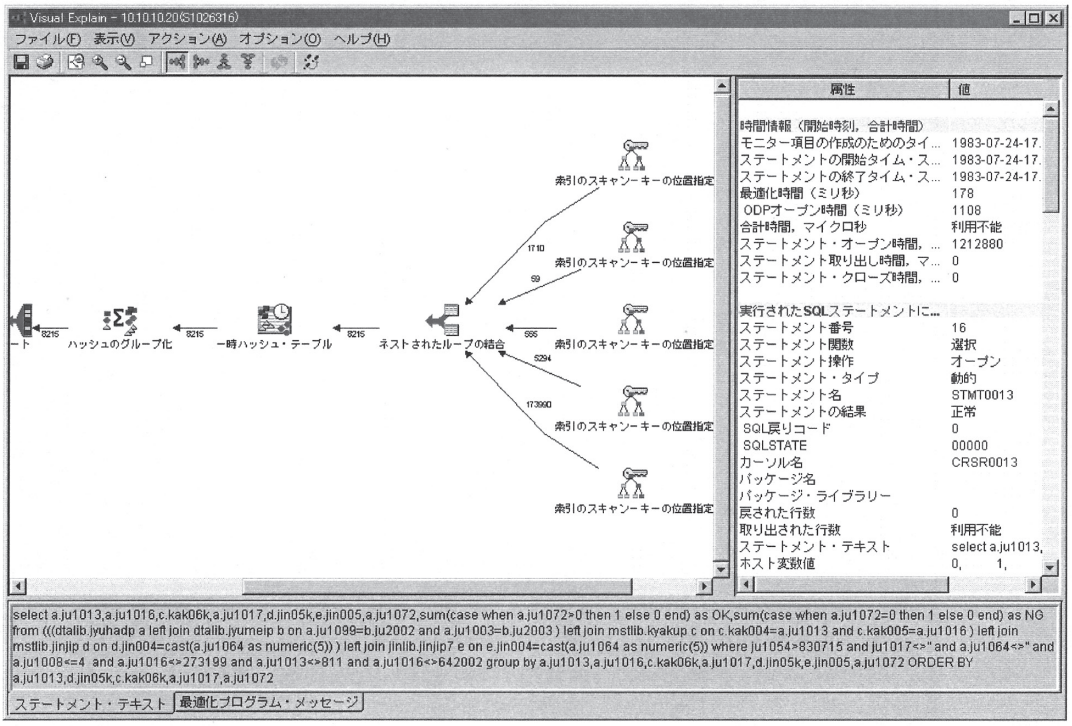


図2-9　iSeriesナビゲータによるGUIでのテーブル作成例

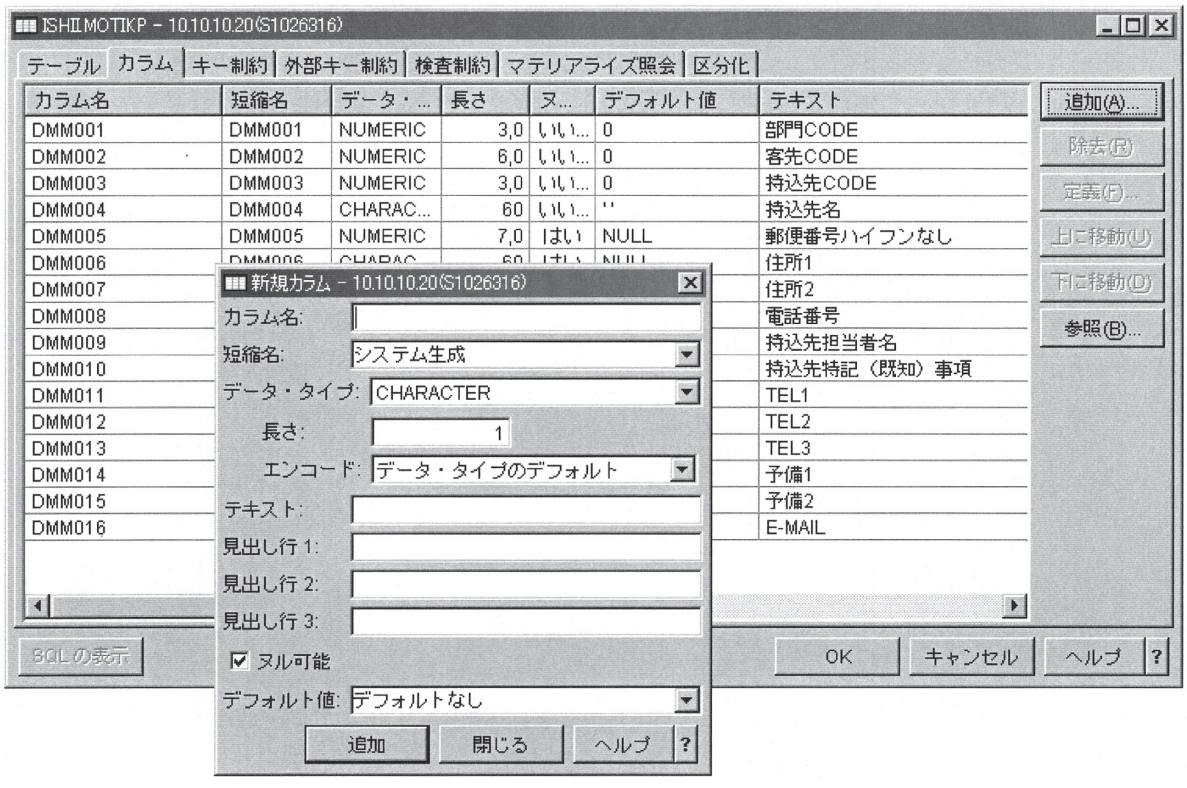


図2-10　iSeriesナビゲータによるGUIでの索引作成例

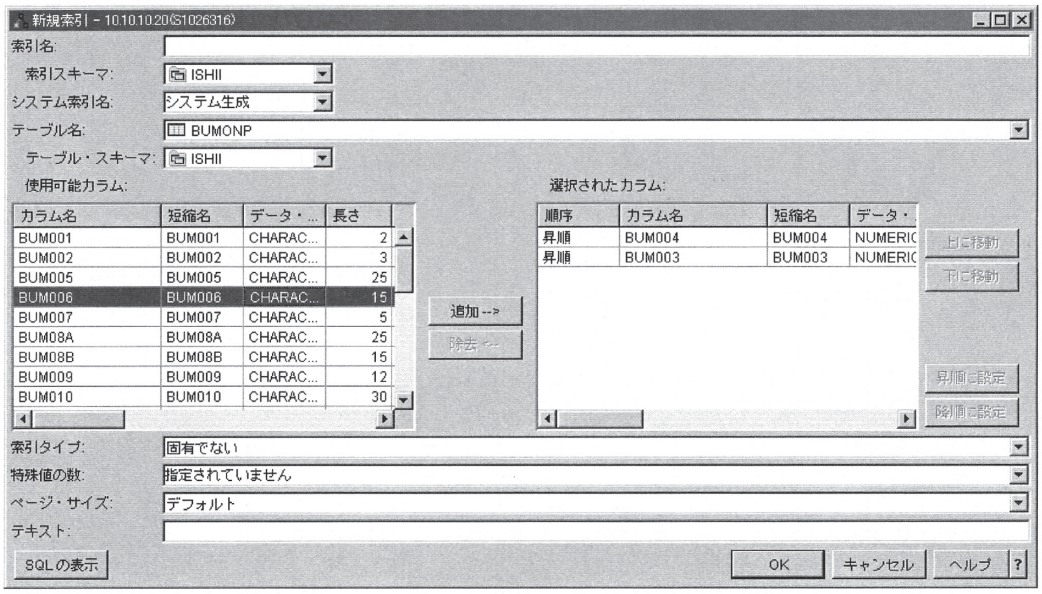


図2-11　Queryの配列番号でSQL文抽出

図2-12　抽出されたSQL文

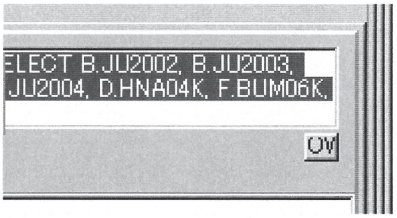


図2-13　単体テスト（SQL検証）用アプリケーション使用例

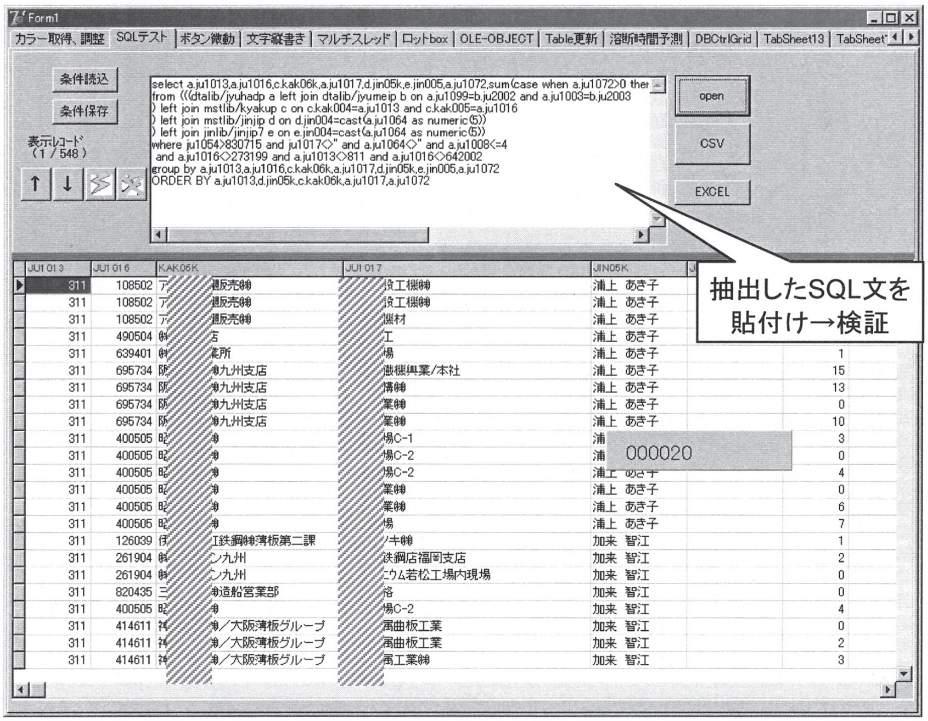


図2-14　SQL実行中の状況例-1

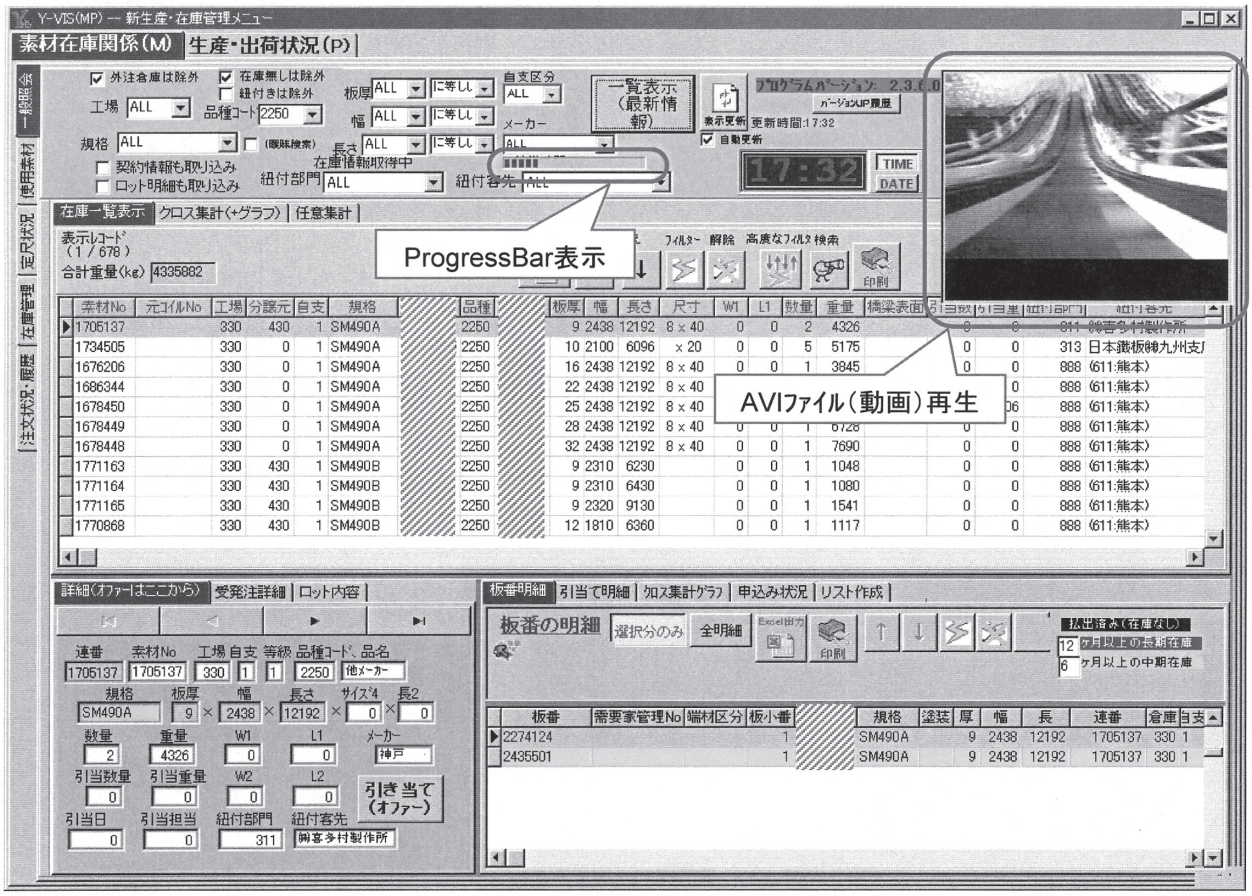


図2-15　SQL実行中の状況例-2

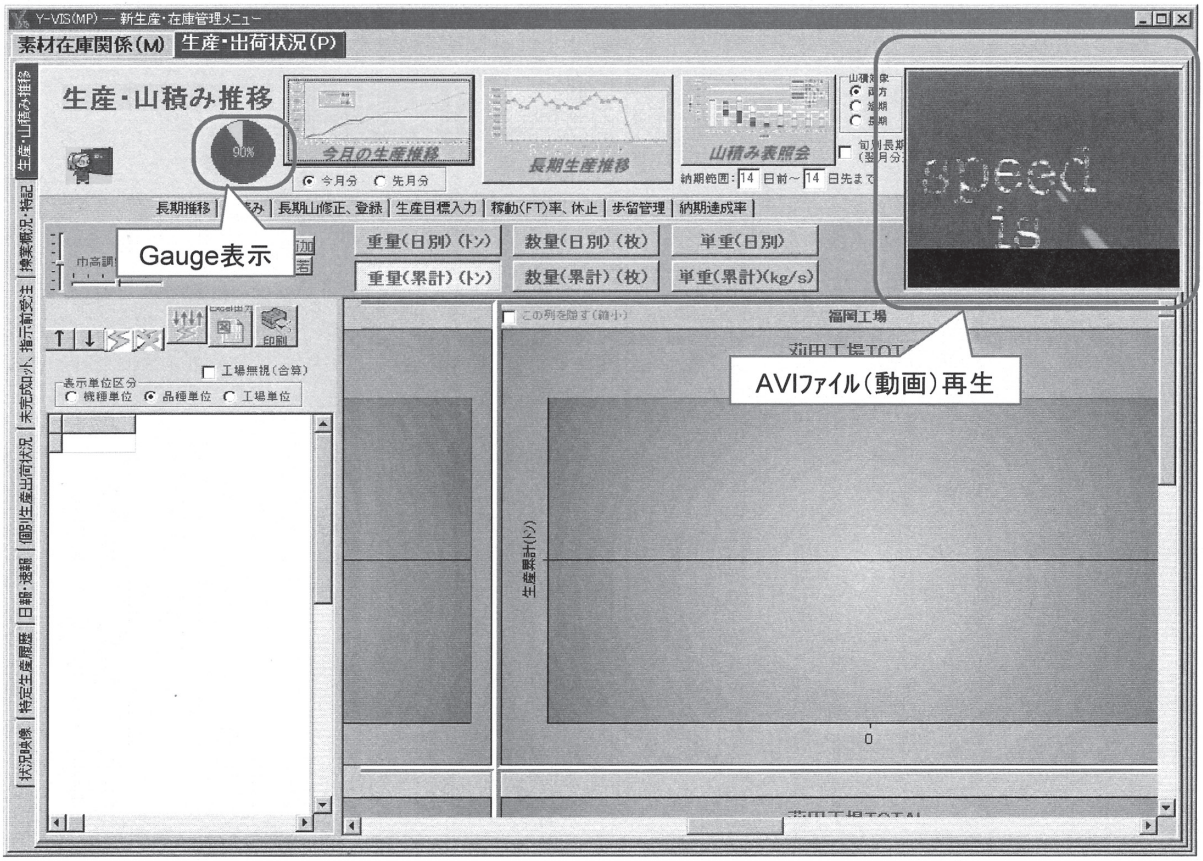


図2-16　ComboBoxのItem内容動的変更例-1

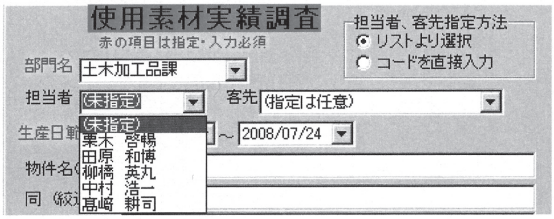


図2-17　ComboBoxのItem内容動的変更例-2

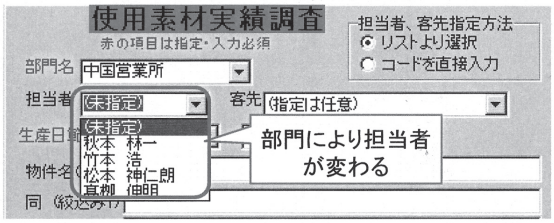


図2-18　ComboBoxのItem内容動的変更例-3

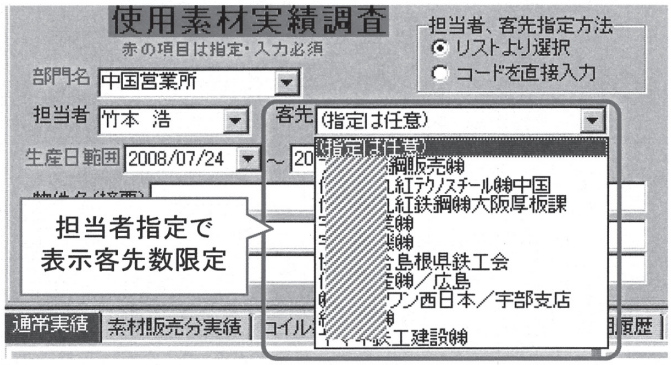


図2-19　各種マスターのローカルPC保存例

図2-20　各種マスターの定期更新例(起動時に実行)

図2-21　各種マスターの更新処理

```
for i:=1 to high(masterTB) do
begin
 BatchMoveMaster.Destination:=masterTB[i];
 BatchMoveMaster.Source:=masterOri[i];
 BatchMoveMaster.Execute;
end;
```

配列化したマスターを
BatchMoveで更新
（batCopyモード）

図2-22　一定時間 アプリケーション操作無しでの表示更新処理

```
procedure TForm1.ApplicationEventsRefIdle(Sender: TObject;
  var Done: Boolean);
begin
  RefreshTimer.Enabled:=true;
  Done:=true;
end;

procedure TForm1.ApplicationEventsRefMessage(var Msg: tagMSG;
  var Handled: Boolean);
begin
 case Msg.message of
    WM_MOUSEMOVE,WM_KEYDOWN,WM_NCLBUTTONDOWN,WM_NCRBUTTONDOWN
    begin
      RefreshTimer.Enabled:=False;
    end;
 end;
end;
```

アプリケーション操作が
無ければTimerを
起動

アプリケーション操作が
有ればTimer停止
→
Intervalのカウントは
一旦リセットされる

```
procedure TForm1.RefreshTimerTimer(Sender: TObject);
var
 t:integer;
begin
 //定時更新処理

 for t:=1 to high(AutoRefreshBTN) do
 begin
  if AutoRefreshYN[t].checked=true then
  AutoRefreshBTN[t].onClick(sender);
 end;

end;
```

Timer のInterval設定時
間に到達すれば
OnTimerイベントで
更新処理起動

図2-23　表示自動更新の設定例

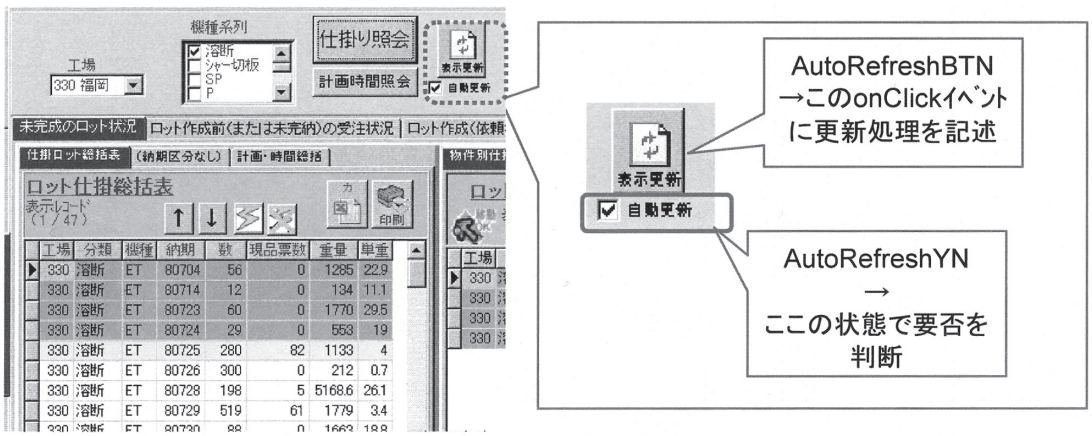


図2-24　項目の並び順カスタマイズの実現方法

```
procedure TForm1.ANYDBGridColumnMoved(Sender: TObject; FromIndex,
  ToIndex: Integer);   //GRIDの列並べ替えあれば並び順定義のStringList を書き直し
var
 n,c:integer;
begin
 n:=(Sender As TDBGrid).Tag;
 mainQUERYorder[n].Clear;  //Public変数の配列

 for c:=0 to mainDBGrd[n].Columns.Count-1 do
  mainQUERYorder[n].Add(mainDBGrd[n].Columns[c].FieldName); //並び順どおりの配列生成
end;
```

並べ替え時にOnColumnMovedイベントで並び順リスト(PublicのStringList)を変更

FormのCloseイベントにIniファイルに並び順等書き込み

```
for n:=1 to HighMainN do      //列の並び順,非表示項目 をPublic変数より書き込み
 begin
 Ini.WriteString('Form',IntToStr(n)+'ColumnORDER',mainQUERYorder[n].CommaText);
 Ini.WriteString('Form',IntToStr(n)+'ColumnNoVisible',mainQNoVisible[n].CommaText);
 end;
```

```
for i:=1 to HighMainN do   //各系列項目並び順のINIファイル定義を使用して表示順を入れ替え
begin
 mainQUERYorder[i]:=TStringList.Create;     //順番用の配列生成
 mainQUERYorder[i].CommaText:=Ini.ReadString('Form',IntToStr(i)+'ColumnORDER',''); //該当INIファイルより取込み
 if mainQUERYorder[i].CommaText<>'' then  //順番定義あれば以下処理
  begin
   for j:=0 to mainQUERYorder[i].Count-1 do
    begin
     for k:=0 to mainQUERY[i].FieldCount-1 do
      begin
       if mainQUERY[i].Fields[k].FieldName=mainQUERYorder[i].Strings[j] then
       begin
         mainQUERY[i].Fields[k].Index:=j;    //INIファイルの順番にIndexを振りなおし
         break;
       end;
      end;
    end;
  end;
end;
```

FormのCreateイベントにIniファイルより並び順を設定

図2-25　項目の並び順のIniファイル記述例

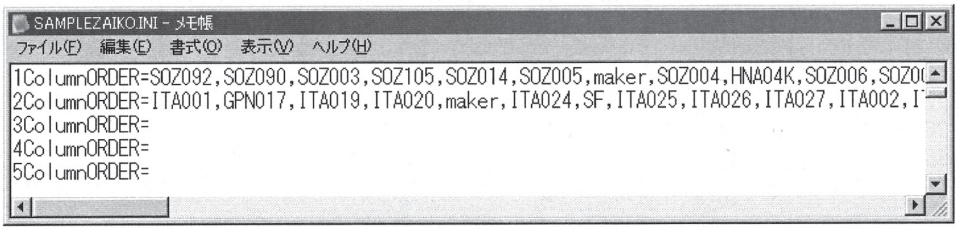
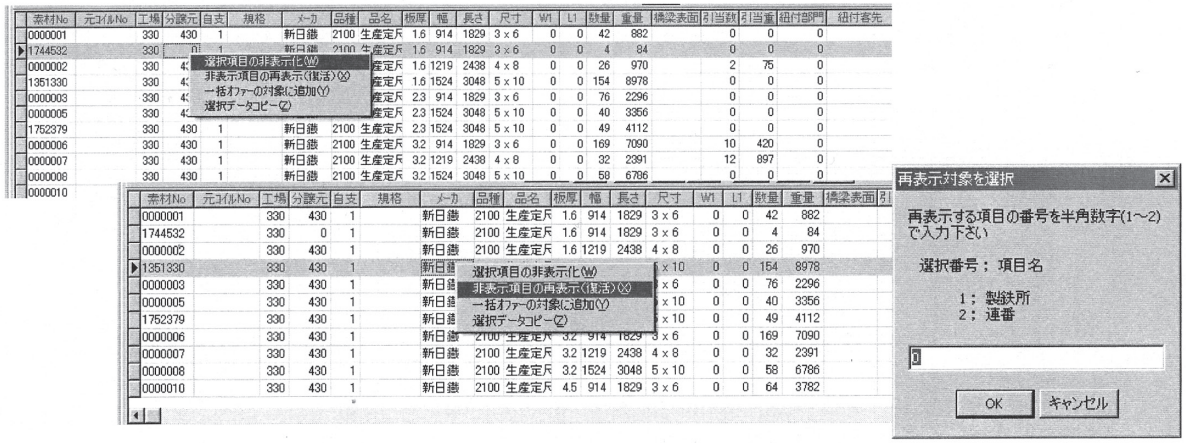
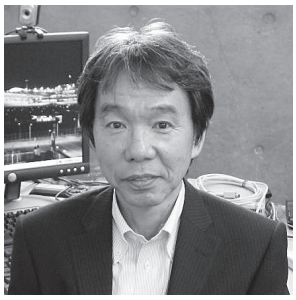


図2-26　項目の非表示化、再表示化操作例



運用部間にサプライズをもたらしたDelphi400

春木 治 様
株式会社ロゴスコーポレーション
管理本部 情報物流室　室長



株式会社ロゴスコーポレーション
http://www.logos-co.com/

スポーツ用品・アウトドア用品（キャンプ用品、登山用品、海水浴用品）の製造販売を行っている。

アプリケーションの開発経緯

以前より、運用部門から、基幹システムのIBM iに対してかなりの情報処理の拡張要求があり、またシステム部門では、グリーン画面での処理環境に限界を感じていた。

昨今のデータ処理におけるオープン化で、さまざまなシステム開発環境が存在している。

ロゴスコーポレーションでは、数年前IBM iをDB2/400マシンと位置付け、インターフェース、情報処理をピアJavaで構築したものの、エラーの解析、アプリケーションの拡張性、保守等々で実際の業務に堪えうるシステムを構築するに至らなかった。やはり、従来からのRPG技術者が自社開発で、いきなりピアJavaを使用した構築を手がけるには無理があったようだ。

しかし、単純にグリーン画面をブラウザすることだけでは、運用部門の情報処理拡張要求は満たされない。そこで、IBM iのデータベースとビジネスロジックのRPGプログラムを継承しながら、Javaのような自由度の高いインターフェースと情報処理の最適化ができる開発環境を求めた。いろいろな開発ツールを検討したが、なかなか結果を想定するのが困難であった。

その中で、Delphi/400のIBM iとの親和性（専用コンポーネントが豊富）と、ミガロのテクニカルサポート（導入前にかなりのFAQを投げた。）の対応が満足であったので、大きく開発環境がぶれることはないと判断し、導入に踏み切った。【図1】

アプリケーション開発の苦労と解決方法

アプリケーションの開発過程では、Delphi/400の書籍を買い集め、基礎的な文法をマスターし、またコンポーネント等々の意味合いや利用方法は都度、必要な時に書籍を調べるようにした。具体的利用方法は、インターネット上にあるサンプルが非常に役立ち、そのまま使える例も多々あった。

とはいえ、今後の継続的な開発を行うため、およびシステム部門のスキルアップ向上のため、構文の意味合いを理解し、Delphi/400をスタディしながらコーディングするように勤めた。Delphi/400の開発言語パスカルは、オブジェクト指向言語であり、昨今のオープン化言語に類似している。あまり違和感は感じなかった。

また、ミガロのサポートセンターにも多くのFAQを行い、敏速で丁寧な対応であったことがシステム開発に大きく貢献した。

アプリケーション開発詳細

1. 商品提案書の作成

システム開発の第一弾は、お客様向けの商品提案書の自動化であった。

この開発は、当社営業部門より4～5年前から依頼があったが、現行システム

図1　Delphiメニュー

図2　提案1

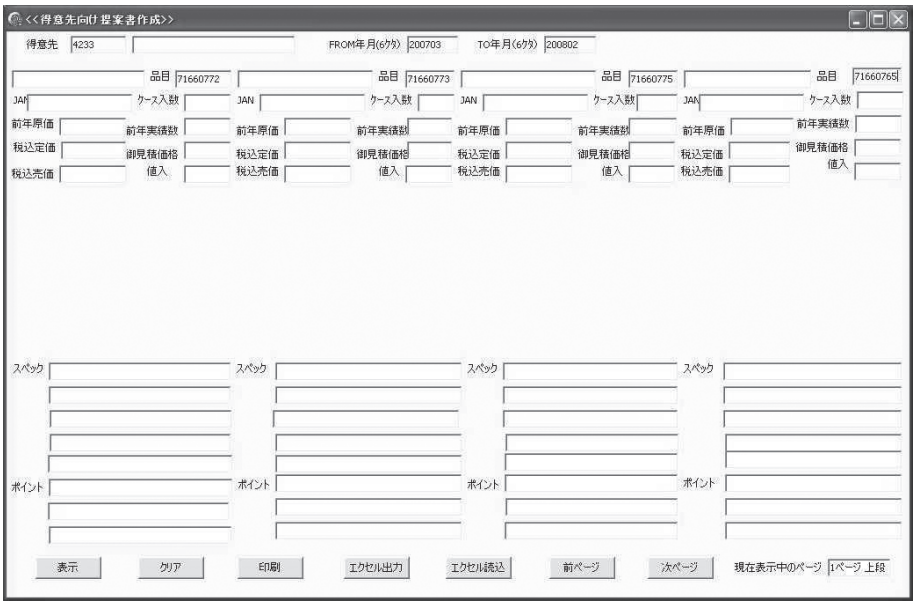
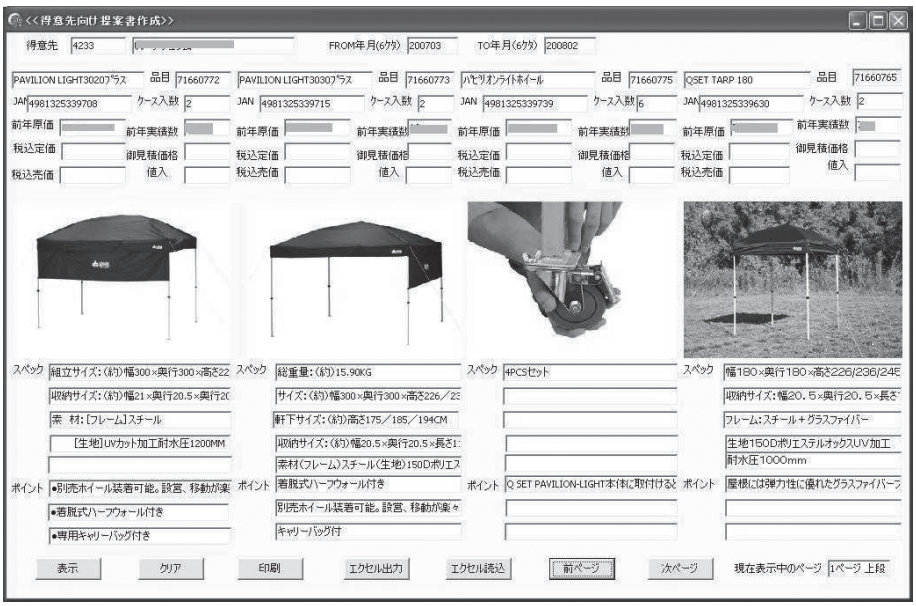


図3　提案2



では不可能であった。外部に製作依頼見積もしたが、当社の仕様が複雑でコストに見合った実用可能なアプリケーションに到達せず、かつ今後の開発を考えるとシステム部門に多少負荷はかかるが、やはり自社開発が好ましいと考えた。

Delphi/400 の導入成功の可否は、このアプリケーションの運用が可能になるか否かと言っても過言ではなかった。

具体的には、当社営業課員が各々手作りしていた得意先向の提案書を、品番の入力のみで 90％くらい作成し、残りはお客様にあわせて編集し、完成させるアプリケーションである。【図 2】【図 3】【図 4】

初期に完成したアプリケーションは、営業部門の実用性には程遠いプログラムであったが、Delphi/400 をマスターするという観点からはシステム部門としては得るものが多々あった。営業部門と度重なるミーティングを行い、実用に堪えうる、かつ画期的な処理も織り込んでアプリケーションを完成させた。

開発経緯

① Delphi 初期入力画面で、複数品番の入力
② IBM i から基幹情報を、ファイルサーバーから画像を、Delphi/400 画面に出力
③ Delphi/400 から Excel に、複数品番の情報（画像含む）の書き出し
④ Excel 上で、品番情報の編集 / 保存
⑤保存した Excel データは再度 Delphi/400 に読み込み、編集可能

Delphi/400 での入力画面は、最終成果物である商品提案書のイメージに近い状態で構成する必要があった。

そのため、グリッドコンポーネントを使用せず、フォームに複数品番の複数項目を配置する形式をとった。ワークファイルによるデータの入出力ができないので、パラメータによる IBM i のデータの入出力を行った。しかし、1 回の RPG の CALL によるパラメータには数の制限があったため、複数回のコールで対応した。

2. 売上予算実績照会

売上の予算 / 実績を、営業所別→担当者別→得意先別→商品別の構成にする。

大分類の営業所別から商品別にドリルダウンしながら、照会を可能にする。

開発経緯

1 つのフォームに 4 つの分類のパネルを配置し、パネルにグリッド表示とチャート表示を同期させる形式をとった。チャートに数値を表示するためにバルーンをつけたが、項目に複数の数値表示が必要なため、グリッド表示も並行表示した。

問題は、各パネルに配置したグリッドとチャートのスライダー BOX による同期が必要なことであった。【図 5】

3. 商品情報照会

商品コードの一覧から任意に商品を選択することにより、その商品の品番マスター情報、画像、倉庫別の在庫情報、発注残（国内、海外）、受注残、その商品の地域別の予算･実績情報､また倉庫コードを選ぶことによって、その倉庫の受払情報（時系列入出荷情報）などが一元表示される。【図 6】

開発経緯

開発で一番注意したことは、オペレーターが選択した品番のさまざまな情報について瞬時に運用判断ができるように情報の選択、配置、応答性を重視した。

グリーン画面では、品番に情報を一元的に把握するには、多岐にわたってメニューの切り替えが必要であった。一方 Delphi/400 では、1 つのフォームに複数のパネルを配置し、その上にグリッドを設定した。しかし、多くのパネルを配置する場合、PC の画面解像度によって、PC 画面にパネルが入りきれないケースがある。フォーム設計では、低い解像度にあわせるよう画面設計をした。

今回の開発で得たノウハウ・教訓・今後の予定 / 計画

RPG 技術者によるオープンシステムの開発は、ビジネスロジックをできる限り RPG で開発し、IBM i との通信および画面設計は Delphi/400 のコンポーネントをフル活用することであった。

イベントで発生するプロシージャの中身も、インターネット上のサンプルを多いに利用した（単にコピペをするのではなく、意味合いを理解する必要あり）。Excel を利用するアプリケーションでは､Excel の VBA の文法が大いに役立った。

今後の予定は、現行のグリーン画面を単にオープン系画面に切り替えるのではなく、業務に必要な情報処理環境に最適化したインターフェースを構築する。

具体的には、今回の品目情報照会のような、得意先コードを入力することにより得意先に関するさまざまな情報を一元的に表示し、即座に業務判断を可能にするといったアプリケーションを構築する。また、昨今増えている Web サービスをアプリケーションに組み込み、従来の処理とまったく違った処理環境を構築する。

最終的には、単独のアプリケーション開発から、サブシステム単位での照会系やエントリー系を組み合わせた Delphi システムを構築する。また、Web 化も検討する。

エンドユーザーの評価・評判・反応

1. 商品提案書の作成

当社営業部門では、新製品等のお客様への提案書を敏速に作成する必要があった。現状は、商品提案に個々の営業マンで微妙な違いがあったり、画像データの保存場所が統一化されていなかったために、商品提案書の作成に多大な工数が必要であった。

今回のシステム化により、商品のセールスポイントや画像データが一元管理され、統一フォーマットによる商品提案書の作成が自動化された。その結果、営業本来の業務に集中できるようになった。

2. 売上予算実績照会

売上等の管理者が、予算に対して大分類である営業所単位から最小単位である商品ごとの予算 / 実績の照会をドリルダウン形式で行え、一元表示が可能になった。

これより、的確に問題分析が可能になり、詳細に予算等の修正も敏速に行えるようになった。

図4　提案3

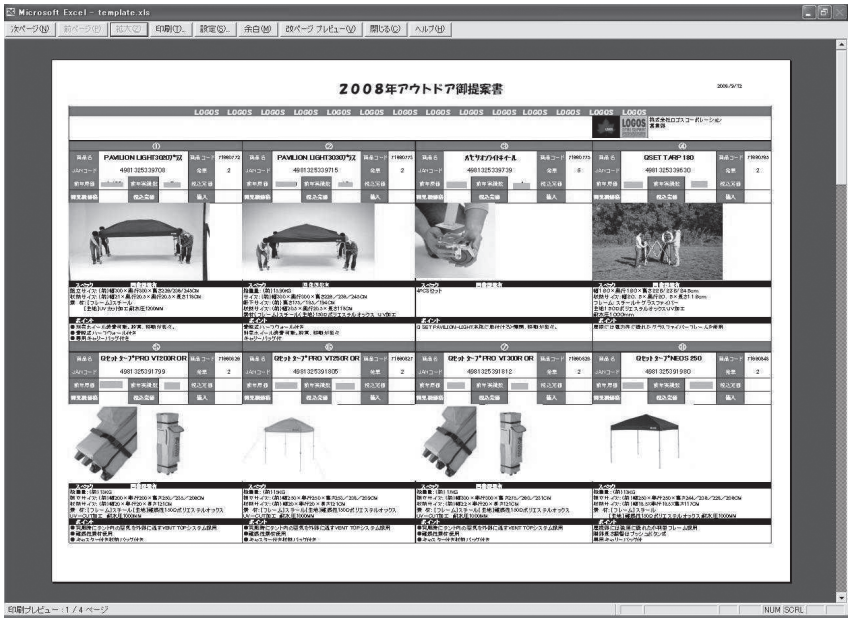


図5　予実照会

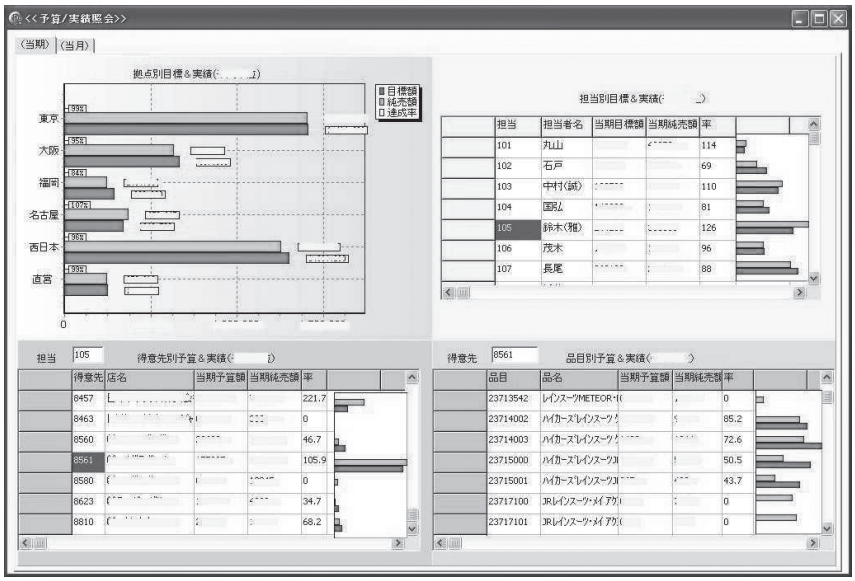
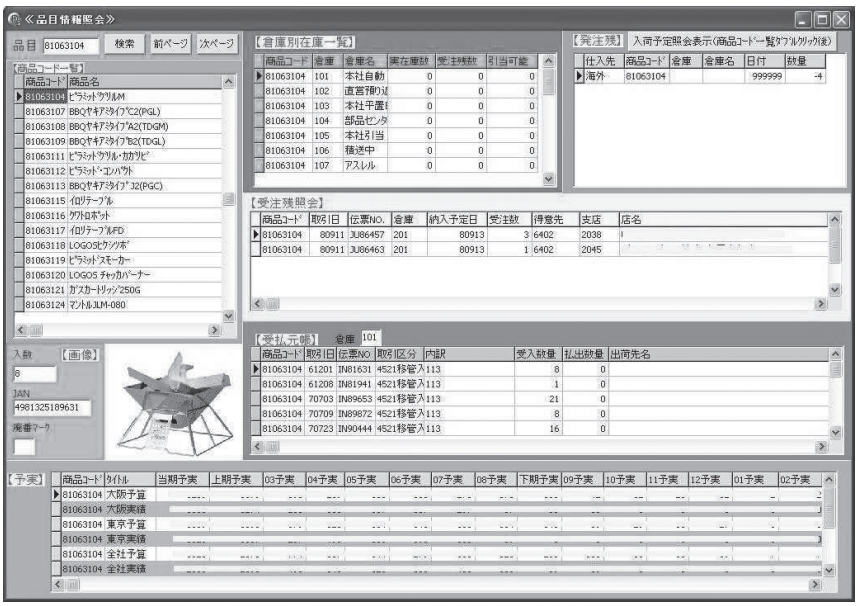


図6　品目情報照会



3. 商品情報照会

受注受付の窓口では、商品関連情報が一元表示されることによって、従来の画面メニューを何枚も切り替えることなく、お客様に対する応答を即座にすることが可能になった。受注受付後の修正等も少なくなった。

また商品デリバリー部門では、商品在庫の適正な配置が可能になり、不要な商品の倉庫間移動に伴う運賃 / 工数が削減された。

M

JACi400使用による Webアプリケーション開発工数削減

中富 俊典 様
日本梱包運輸倉庫株式会社
情報管理部　管理課

日本梱包運輸倉庫株式会社
http://www.nikkon.co.jp/

当社は1950年に設立された。自動車、住宅関連メーカー、プラント関連を中心に、調達・工場物流や国際物流などを一貫してサポートする複合物流業者である。国内事業拠点数80箇所、国内関連会社25社、海外関連会社12社を持ち、国内外を問わずグローバルに事業を展開している。

日本梱包運輸倉庫「WEB照会システム」

今回、開発対象となった「WEB照会システム」は、日本梱包運輸倉庫株式会社のお客様にWebアプリケーションにログインしていただくことで、お客様よりお預かりしている品物の在庫状況や出荷状況をリアルタイムに提供できるシステムである。【図1】

JACi400導入の背景と選定理由

当社では、自社システムとしてAS400を利用しており、基本的に自社開発を行っている。既存のWebアプリケーションツールとしては、さまざまな取引先向けに在庫照会、オーダーの入力を行うためのWebアプリケーションを複数構築していた。

このツールは、5250の画面からコンバージョンして、Web画面を生成するタイプのツールである。そこで開発するアプリケーションは、ツール独自のプログラムであり、開発に時間がかかるという問題点があった。また、メンテナンスも非常に効率が悪く、顧客要望に応えるのにも時間がかかっていた。

また、このシステムは2002年に導入され、Webサーバーが老朽化しており、これ以上そのサーバーで運用を行うことが難しい状況となっていた。

そこで、そのシステムを更新するためのツールとして、次のような理由から、AS400で開発ができるJACi400を採用した。

【JACi400導入の理由】
・開発から納入まで、短期間で行える。
・自社内開発対応で、Webアプリケーションが構築できる。
・HTMLに詳しくなくても、開発が可能。

開発作業について

開発作業は、JACi400Designerの設定およびRPGを筆者自身が担当し、HTMLは別の担当者へ依頼した。

1. HTMLの開発

設計時にHTMLのイメージを作成し、それをHTMLの開発者に依頼。HTMLは、基本的な知識で容易に作成することができた。

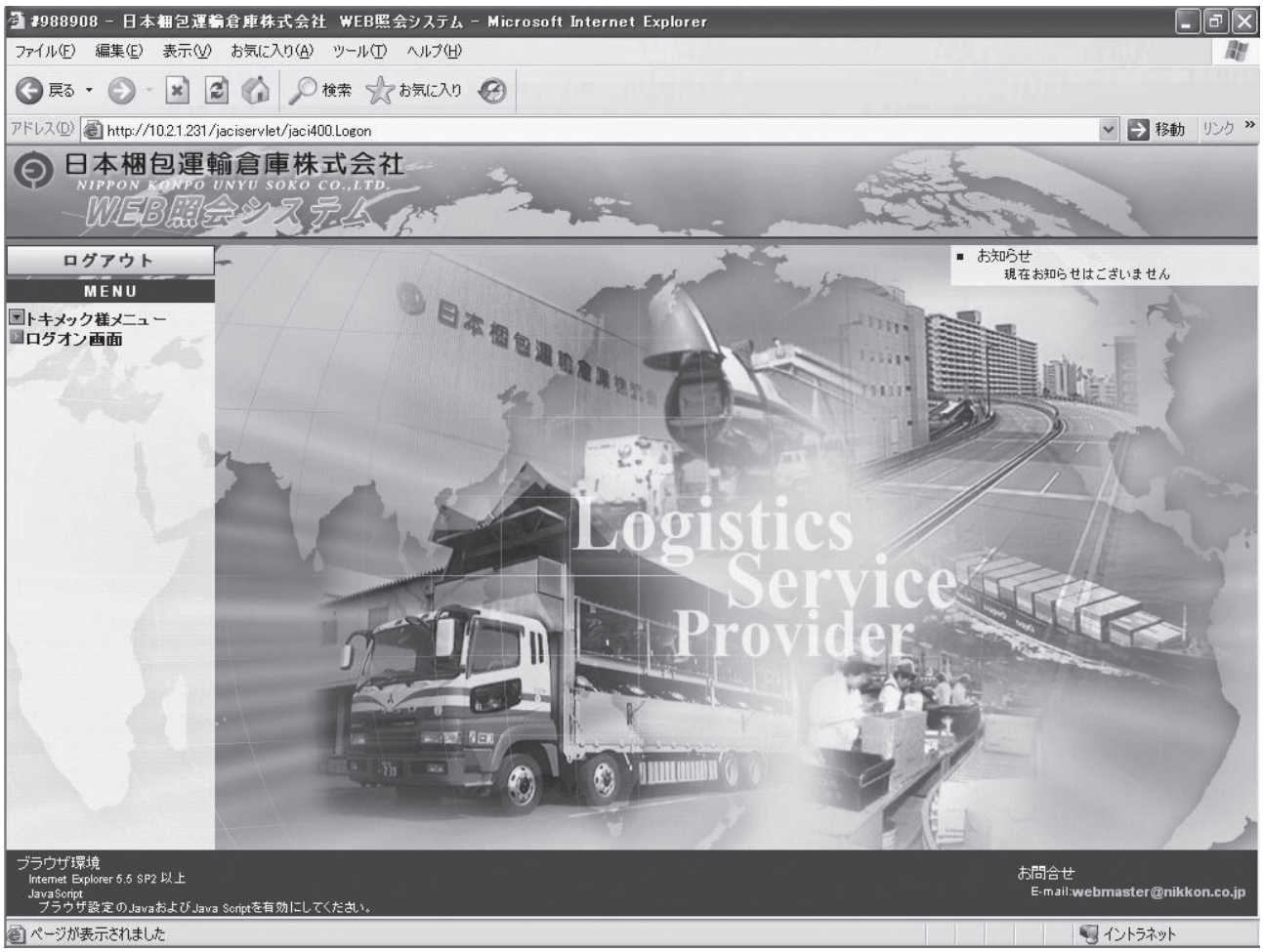
また、Designerで設定した内容が、HTMLの入力制御に反映されるので、HTMLの作成工数も削減することができた。

2. RPGの開発

RPGは、JACi400 Designerでの設定項目より、サブルーチン化されたスケルトンプログラムを生成する。このサブルーチンの中身を記述するという手法だったため、記述パターンを理解するのに多少手間取った。だが、1本開発し理解した後は、スムーズにプログラミングできた。

Webサーバーとの通信部分のプログラムは、JACi400が自動で実行してくれてプログラミング不要である。なので、

図1


#988908 - 日本梱包運輸倉庫株式会社 WEB照会システム - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り
アドレス(D) http://10.2.1.231/jaciservlet/jaci400.Logon
移動 リンク
日本梱包運輸倉庫株式会社
NIPPON KONPO UNYU SOKO CO.,LTD.
WEB照会システム
ログアウト
MENU
トキメック様メニュー
ログオン画面
お知らせ
現在お知らせはございません
Logistics
Service
Provider
ブラウザ環境
Internet Explorer 5.5 SP2 以上
JavaScript
ブラウザ設定のJavaおよびJava Scriptを有効にしてください。
お問合せ
E-mail:webmaster@nikkon.co.jp
ページが表示されました
イントラネット

5250のプログラムを作るのと同様の考え方で構築することができた。

3. 開発上発生した問題とその解決方法

㈱ミガロのHPに、必要なマニュアルやサンプルプログラムがあるため、ほとんど困ることなく開発ができた。とはいえ、プログラミングの手法でいくつかの不明点が発生した。

●JavaScriptの使用手法

まず、JACi400上で、JavaScriptを使用する手法の例を挙げてみる。

現象：JACi400でIDを振った項目に、組み込んだJavaScriptが動作しない。
原因：一般に、ボタンのクリックはonclickイベントに記述する。だが、onclick時点では、JACi側のアクションが優先されてしまうため、動作しない。
対策：onmousedownを割り当てることによって解決した。

JACi400のHTMLに独自に組み込むJavaScriptは、onmousedownやonmouseupなどの発生が早いイベントに割り当ててやれば、動作させることができることがわかった。

●ファイルの組み込みパス

独自に外部ファイルを組み込んでいたため、次のような留意点も判明した。

①ファイルの組み込みパスは、WAS上のrootから、相対的なパスを指定する。
'../jaci400/html/該当フォルダ名/ファイル名'
②ファイルの組み込みパスは、'form'定義以降の行に組み込む。

●開発環境

WASに関わる部分で、HTMLのフィールド項目をASに反映させるのに、WASの再起動が必要であった。本番環境においては、日常業務の稼働中にWASを止めることはできない。そのため、常時開発できる開発環境を用意する必要があった。

㈱ミガロに相談して開発環境を別途構築し、開発環境で開発・テストを行い、その後本番環境に移行するようにした。

4 セキュリティについて

「SECUREMATRIX」というワンタイムパスワードの認証ツールを採用した。SECUREMATRIXのポータルサイトから、JACi400のアプリケーションを呼び出すことにより、強固なセキュリティを実現している。

2つのツール間でワンタイムパスワードを実現するための連携については、購入前に、SECUREMATRIXの販売元ネットマークスとミガロの協力を得た。連携手法の検討ならびにテストを実施し、実現方法を確認してあったため、導入後は問題なく稼働が実現した。

効果と今後

今回のWeb在庫照会のシステム開発の一番の成果は、開発工数の削減である。

約50画面の本システムで、以前のツールでは約12人月（3人×4ヶ月）だった開発が、JACi400では6人月（2人×3ヶ月）と半減した。

この要因として、JACi400は新たなスキルを習得することなく、汎用的な技術であるHTML、RPGで簡易に開発ができたことと、㈱ミガロのHPのTIPSやFAQ、テクニカルサポートを有効に活用したことが大きい。

本システムは、今秋を目処に随時展開中である。

また、昨今、Webよりオーダーを入力したいという案件・要望が増加している。取引先のニーズにすばやく対応することがビジネスの大きな鍵となっている現在、JACi400は、今後有力な開発ツールとなると予想される。

M

Delphi/400を利用したWeb受注システム

飯田 豊 様
東洋佐々木ガラス株式会社
経営管理部 情報管理課　課長

東洋佐々木ガラス株式会社
http://www.toyo.sasaki.co.jp/

資本金１億円・従業員数約 510 人で、拠点数は７箇所。その間をインターネット VPN で結んでおり、うち４箇所が営業拠点である。

当社は、2002 年に東洋ガラスのハウスウェア部門と佐々木硝子が統合した会社である。自社ブランド製品・オーダーメイド製品・輸出向け製品を取り扱っている。
製造方法として、大量かつ安定した品質のマシンメイド品から、職人が手造りする高級ハンドメイド品があり、双方の製造販売を行っている。【図 1】

アプリケーションの開発経緯

東洋佐々木ガラス株式会社の受注方式は、EDI が 5% で、残りの 95% が FAX によるものである。当社の顧客数は 2000 社を超え、製品点数も多いうえ、得意先数は増加傾向にあり、受注工数の増加が問題になっていた。

この処理を合理化するため、2005 年からシステム投資を行い、次のような取り組みを行ってきた。その延長線上に、今回の Web 受注システムがある。

①電子帳票の導入
（2006 年 4 月導入）

出荷処理後の伝票および請求書のファイリング工数削減のため、電子帳票を導入。

② FAX サーバーの導入
（2007 年 8 月導入）

FAX サーバーを導入することにより、受注 FAX をイメージデータとして蓄積し、受注処理はそのイメージデータを見ながら行うこととした。これにより、FAX 用紙のファイリング時間の削減、機器の保守費用、FAX 用紙代の削減が可能となった。【図 2】

受注担当は 2 画面にし、片方に FAX イメージを表示し、もう一方に IBM i の出荷指図画面が表示されるようにした。

受注 FAX をイメージデータとすることで、FAX 受信拠点と受注作業拠点が同一である必要がなくなり、受注入力工数の最適化が可能となった。さらに、出荷回答の FAX 返信を、システムが自動処理で行うことで、それに関わる作業時間が短縮された。

なお、FAX サーバー導入にあたっては、FAX 機器の老朽化が進み更新時期であったこともあり、上記のような改善が期待できたためにシステム導入を実施した。

③受注センターの作成
（2007 月 4 月実施）

システム導入のため受注業務に特化した部署を設置し、得意先との契約単価の整備・運賃契約の見直しなどを行い、より効率的に運用できるように業務改善を行った。【図 3】

アプリケーション開発で苦労した点

今回の Web 受注では、得意先に対するサービス向上として、在庫照会・製品画像およびスペックデータの提供・出荷状況報告などを実現した。

当社の合理化としては、得意先からの受注をデータ化し、入力工数の削減をテーマとしている。【図 4】【図 5】

開発にあたり、次の点を注意した。

1. セキュリティの確保

安全にデータを交換する方法として、SSL 通信でワンタイムパスワードが利用できることを前提とした。アプライアンス製品やキャリアが提供するサービスを比較したが、当社の要件に合致したの

図1　東洋佐々木ガラスの拠点

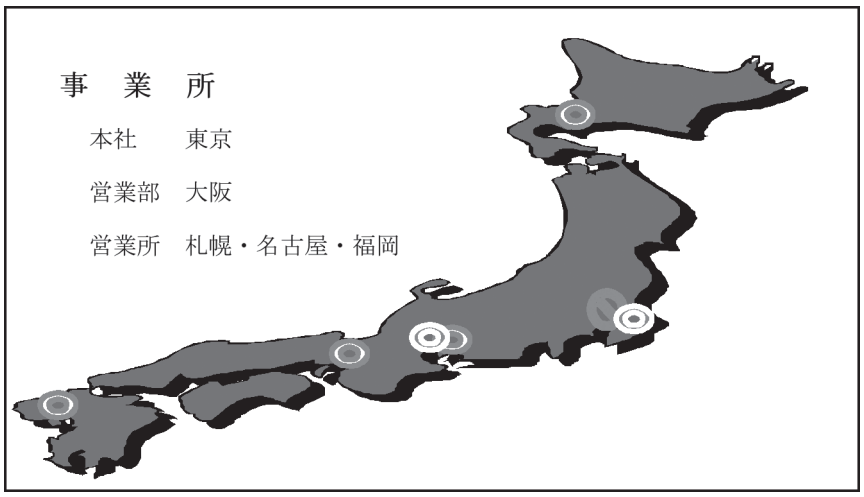


図2　FAXサーバー

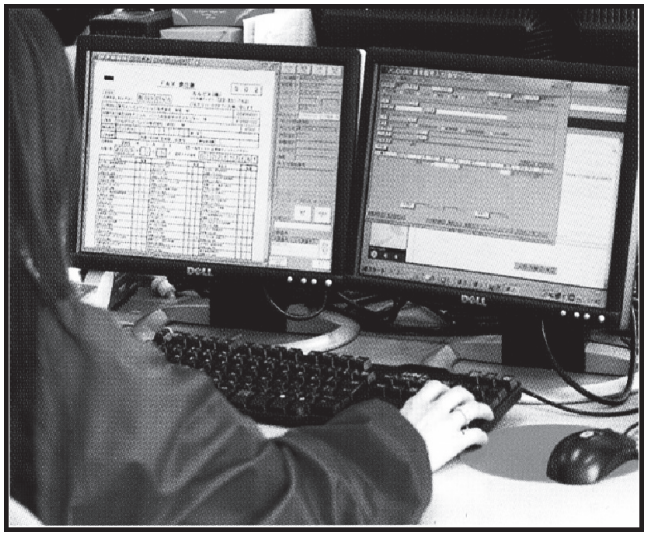
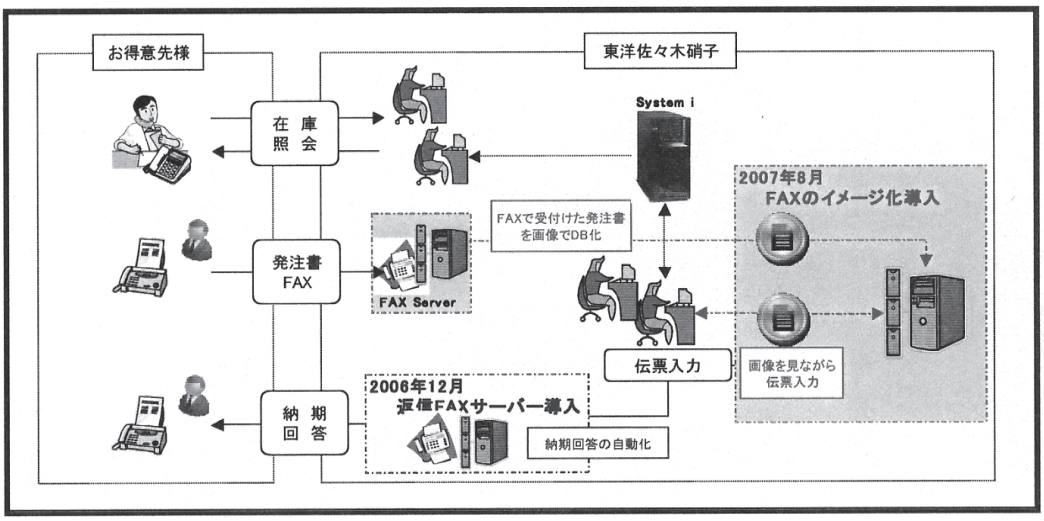


図3　Web受注導入前の処理フロー



が、e-JAN ネットの CACHATTO であった。

このソリューションは、前提条件を安価に利用できる点が決定理由の1つとなった。さらに、当社の別テーマである、営業マンが携帯電話（キャリアを指定しない）でノーツメール・スケジュールにアクセスし、照会・返信することが可能であり、モバイル PC 等からも安全に Web アプリが利用できるなど、複数課題の解決につながる点が決定理由となった。

2. 画面表示のレスポンス改善

開発当初は、在庫照会のレスポンスが悪かった。だが、在庫検索のロジックに Delphi/400 で用意されている IBM i のバッチジョブ起動を利用し、在庫検索処理を IBM i で行うことにより、画面応答がかなり改善された。【図 6】【図 7】

3. 受信データの安全かつ確実な取り込み

受注データを受信するにあたり、ウイルス混在の可能がある。そのため、サーバー内のフォルダに保存し、アンチウイルスソフトによる確認後、IBM i に取り込みを行うことで、データの安全を保っている。

また、受注データであるためデータ欠落の防止策として、まず、受信データをサーバーに確認用として保存し、次に IBM i に送信後、変換マスターにより各種変換を行い、受注ファイルに書き出し、さらにそのデータをサーバーに戻し、確認用のデータと付き合わせを行い、データ件数・内容を確認している。

一致した場合のみ、得意先に送信完了のメッセージを出す。不一致の場合は、当社システム担当に連絡をするようにメッセージを表示するようにしている

また、受注データのフォーマットは、得意先にあわせるようにしている。当社フォームにあわせてもらうと、得意先でのシステム変更・開発が必要となり、参加得意先が減ってしまう。そのため、開発工数の比較的少ない Delphi/400 でフォーマット変換を行うことを行って、当社側での負担も極力減らしている。【図 8】

4. 社内処理の効率化

受注をデータでもらっても、マスター整備に時間を要しては作業効率が上がらない。そのため、出荷処理を行うと同時にマスターメンテ用のデータを蓄積し、1日3回のバッチ処理で登録・変更が可能なシステムにした。これにより、新たに得意先が FAX から Web に変更しても、スムーズな対応が可能となった。【図 9】【図 10】【図 11】

また、FAX・Web の受注データを一覧表示し、そこからの出荷指図処理を可能とした。

また、得意先の発注方法も、FAX と Web の混在を可能とすることで、運用に柔軟性と安定感が出た。得意先の出荷状況照会も、FAX・Web データ混在可能な照会画面とした。【図 12】

システム稼働に向けて感じたこと、今後の予定

今回のシステムは、得意先にデータ提供することによるサービス向上と、得意先から発注データを送信してもらい、そのデータを活用することによる当社の作業効率を上げることを目的としている。

とはいえ、得意先に利用してもらわないと意味がない。そのため、システム担当者が営業と一緒に得意先を周り、システムの内容・目的・トブル対応を詳しく説明を行い、安心してもらえるようにしている。考え方に賛同してもらうことが最も重要と考える。

得意先と会話する中では、JAN コードによる在庫照会や納品明細・請求書データの提供など、さまざまな改善要望が出されている。より使いシステムにするために、機能追加を順次行う予定である。

得意先の感想と出荷担当の評価

●得意先の感想

現在までに参加の意思をいただいた得意先は 14 社で、うち 4 社が実稼働を行っている。残り 10 社も本稼働に向けて準備中である。最終的には、40 社程度まで増やしたいと考える。

説明後すぐ本稼働にならないのは、得意先の多くで発注事務処理の変更が必要であり、システム修正が必要な場合もあるので、得意先の対応をじっくり待つ必要があると考えている。

利用している得意先の感想は、準備した各機能の評価が高い。だが、画面レスポンスの改善要望が出ているので、検討が必要と考える。

本稼働の初期段階では、小さなトラブルもあったが、得意先の担当と顔見知りになっていた点が大きく、スムーズに対応できたと考える。

●当社出荷担当の評価

一方、社内では、入力工数が今までの 10 分の 1 程度に短縮されたとの報告がある。対象得意先を増やすような依頼も来ているので、社内処理はうまくできていると考える。

本稼働している 4 社の実績を考慮し、削減時間を計算すると、約 30 社で月当たり 250 時間程度の削減効果が期待できる。

M

図4　Web受注導入後の処理フロー

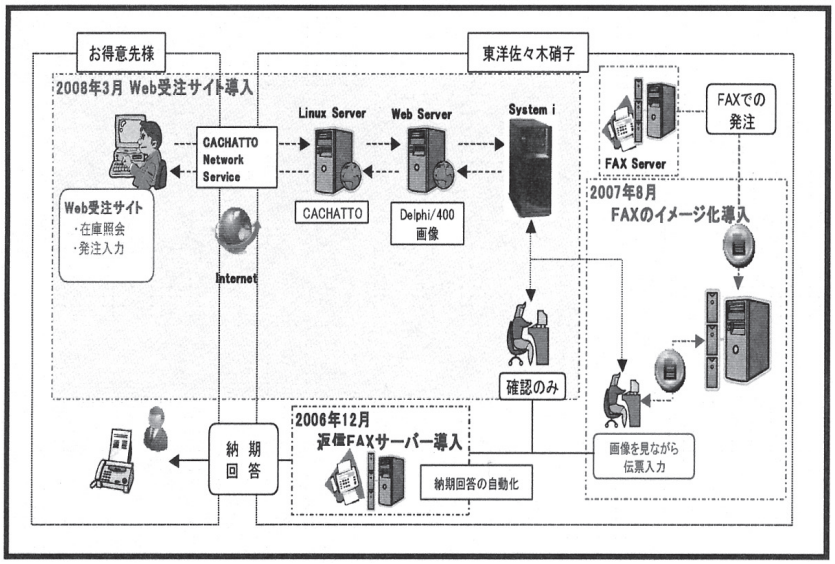


図5　Web受注導入後の運用フロー

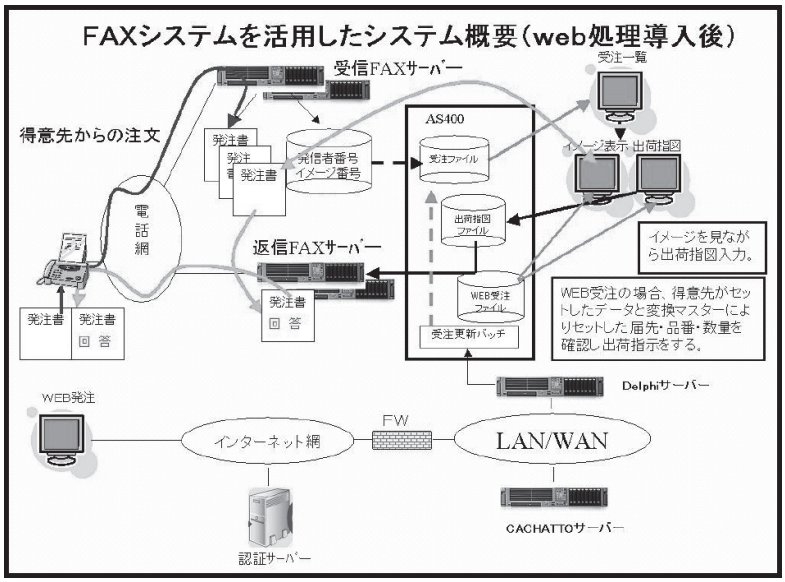


図6　得意先に提供するメニュー

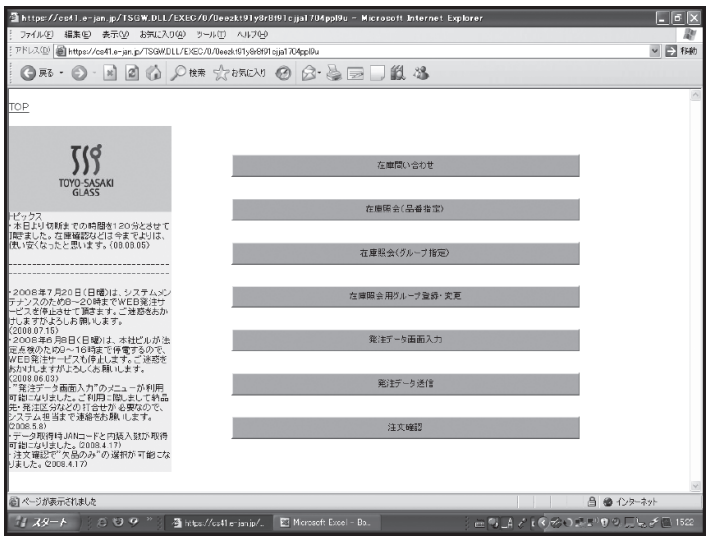


図7　在庫間照会および画像・スペックのデータ取得画面

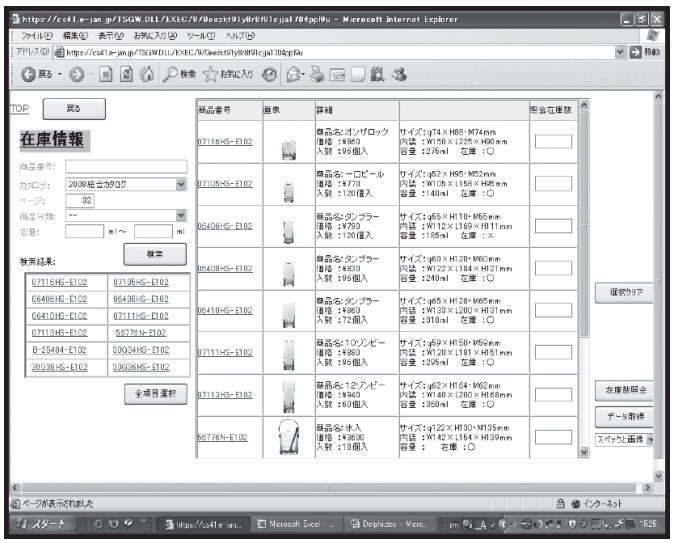


図8　得意先から発注データの送信用画面

図9　当社の出荷指図一覧①

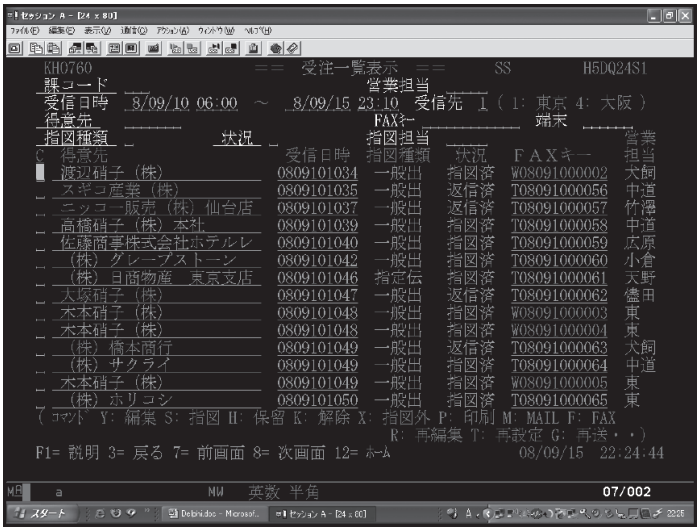


図10　当社で受注データを参照する画面

図11　当社の出荷指図一覧②

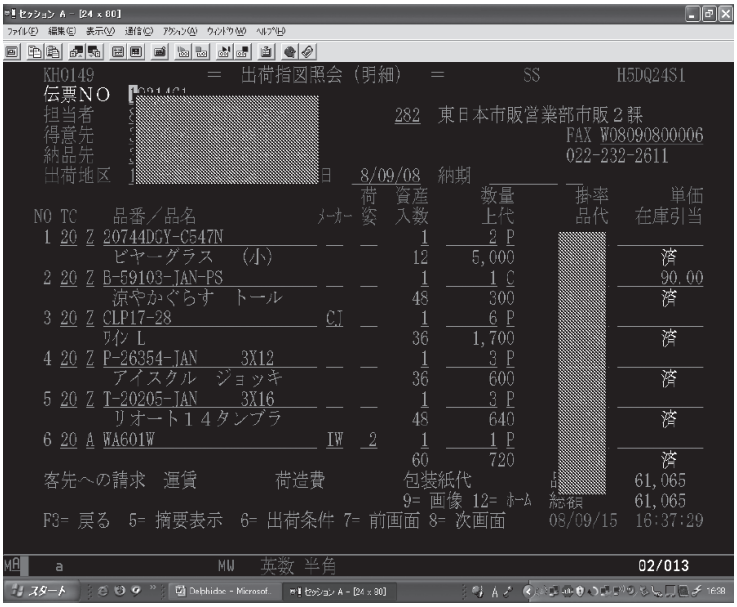


図12　得意先での出荷状況照会

優秀賞

Delphi/400による販売管理システム(FAINS)について

藤田 建作 様

株式会社船井総合研究所
システム室　主任　チームリーダー

株式会社船井総合研究所
http://www.funaisoken.co.jp

株式会社船井総合研究所は、企業経営のコンサルタントとして、売上や収益力のUPといった実務レベルのマーケティングから、中長期経営計画、新規ビジネス構築などの企業戦略の構築まで、総合的に支援している。

システム化の経緯と課題

株式会社船井総合研究所では、1998年より基幹の販売管理システムである「FAINS（Funai Advanced Intelligence System）」を、Delphi/400以外のツールで開発して稼働させてきた。その後数年たち、事務処理効率のさらなる向上や情報検索機能の充実を目的として、2004年にFAINSの再構築を行うことになった。

再構築を行うにあたり、次の3つの課題を検討する必要があった。

①事務処理の煩雑さ

第1の課題は、事務処理の煩雑さだ。これまでは事業部ごとに営業事務担当が配置され、売上入力、交通機関の切符手配、業務完了処理、顧客情報照会等を個別に行っていた。

業務フローが複雑でツールが汎用的でなかったため、業務の流れを把握し、ツールに慣れていないとスムーズな処理が行えなかった。

つまり、コンサルタントが直接、事務処理を行うことができなかったのである。

②顧客データの信頼性

次に顧客データの信頼性も課題であった。すでに取引のある顧客については一元管理が行えていたが、見込客の情報がチームごとに管理され、一元化されていなかった。

また、登録内容の変更依頼を受けても完全に対応できず、クレームが発生することもあった。

さらに、多くの市区町村合併が発生する中、顧客データの修正対応が十分に行えてなかった。

③会計処理

最後の課題は、プロジェクトが完了してから請求書が発行されるまでに、すごく時間がかかっていたことだ。交通費立替金請求等の処理も複雑なため、業務の見直しが必要であった。

アプリケーションの開発方針

先にあげた課題を解決するには、バラバラであったシステムを連携する必要があった。

今回のプロジェクトでは、3つのサブシステムを連携することにした。営業がメインで利用する「営業支援システム（FIELDS）」、スタッフが利用する「販売管理システム（新FAINS）」、経理部門が利用する「会計システム」の3つだ。

会計システムに関しては、早々にパッケージソフトを使用する方針が決定した。FAINSについては当初、従来のツールでのバージョンアップで対応しようとした。しかし、バージョンアップ費用が予想以上に高価で、費用対効果が少ないと感じたため、見直しが必要になった。

そこで、検討したのが、以前から出張精算システムや顧客情報を抽出する条件検索システムとして利用していた「Delphi/400」である。これらシステムは小さなものであったが、ユーザーから

の評判もよく、また AS/400 との相性もよいことがわかっていたため、今回は FAINS を Delphi/400 で全面再構築することにした。

アプリケーション開発の工夫と苦労

FAINS を構築するにあたり、次のような点を考慮した。

1. サブシステム間の連携

今回のプロジェクトでは、3つのサブシステムを連携させている。

営業支援システムは、PC Server をベースとする Web アプリーションのため、AS/400 とのデータ連携が必要になった。これに関しては、バッチ処理による FTP 転送を行うことで解決した。トランザクション系データは1時間ごとに、マスター系データは夜間バッチでそれぞれ転送している。

また、転送の際にはスムーズな連携を実現するために、やり取りするデータ量を最小限にする必要があった。これに関しては、転送対象の各ファイル上にデータ転送専用の更新情報を持つことで、必要最小限のデータのみが転送されるよう工夫を行った。

これらにより、コンサルタントが営業支援システムで登録した情報は、販売管理システムである FAINS にスムーズにデータが流れるようになった。

2. 住所情報の取り扱い

旧システムで管理が困難になっていた顧客データについては、メンテナンスを容易にするため、新 FAINS では住所マスターを持つこととした。

顧客マスター上は全国の住所を 10 桁のアドレスコードのみで保管するようにして、住所マスターを定期的に更新することで、最新の情報がいつでも参照できるようにした。

3. ログオン情報の使用

FAINS を使用するユーザーを限定する仕組みを検討するにあたっては、すでに導入していた Windows 上の Active Directory を使用することを検討した。

これは、ADSI という仕組みを利用することで、FAINS のログオン画面で入力されたユーザーとパスワードの情報で、Active Directory に問い合わせを行い、認証を行うということを可能にした。

プロジェクト運営で苦労したのは、やはり複数のサブシステムを並行して開発を進めるという点であった。

営業支援システムについては当初、Lotus Notes 上で稼働するツールを前提に設計したが、要求仕様を十分に満たせず、急きょ手組みによる Web 開発に変更したりもした。

そんな中、各開発パートナーの協力のもと、仕様変更に柔軟に対応しつつ、プロジェクトを推進した。その結果、当初の計画どおり約1年の開発期間を経て、2005年1月に無事システムをカットオーバーすることができた。

稼働後の効果

これまでは、営業事務が、業務完了処理や出張精算などの処理を行っていた。新システムでは、コンサルタントが直接システムに登録できるようになった。これにより、事務処理にかかわる人件費を大幅に圧縮できたことはとても大きな効果であった。

また、システムの連携もスムーズに行われるようになったため、コンサルタントが登録した情報は、即座に販売管理システムの FAINS にも反映できるようになった。これにより、請求書発行処理の早期化も実現できた。

さらに、見込み客情報も含めた顧客情報がシステムに登録されるようになり、住所情報の精度も向上したため、正確な顧客情報の一元管理が可能になった。また、顧客へ送付するダイレクトメールの返送率が下がったことも、大きな効果ではなかったかと思う。

現在の状況と今後の計画

システムの本格稼働から3年が経過し、現在は、保守フェーズとして日々発生する要望に関する対応を行っている。

プログラムの改修作業は、主に開発パートナーに委託しているが、社内に VPN 環境を構築することで、開発パートナーの即時対応が可能な環境が用意できた。昨今は内部統制に対応するためのルール化を行いながら、システムの安定稼働に努めている。

新システムが稼働して来年で丸5年となるため、いよいよ次期システムの検討を開始しようとしている。手薄となっている人事系システムとの連携や CRM の追加などを考慮しながら、現在のシステムが分散型になっている点を集約する方向で検討したいと考えている。

FAINS システムでの成功を、次期システムでも活かしていく予定である。

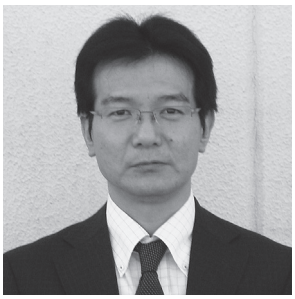
M

優秀賞

技研化成の新基幹システム再構築

藤田 健治 様

技研化成株式会社
管理部 情報システムグループ　グループ長



技研化成株式会社
http://www.gikenkasei.co.jp/

プラスチック容器の総合メーカーとして商社機能、ディーラー機能をあわせ持ちながら、幅広いサービスをお客様に提供している。
省資源化の代表である PSP をはじめとする軽量化容器に積極に取り組み、環境面にも配慮したサービスの提供を行っている。

2つの大きな狙い

技研化成株式会社は今回、2つの大きな目的によりシステム再構築を実施した。2大目的とは、①積水化成品工業グループのメリット享受と、②当社システムのスリム化である。特に後者は、導入前 1251 ジョブ→導入後 700 ジョブを実現した。

①積水化成品工業グループのメリット享受
- IBM インフラへの移行による、グループインフラの統一
- 情報の同質性によるメリット
- 連結経営の推進
- 効率化、スピードアップ

②システムのスリム化
- 開発生産性向上、開発コスト削減
- スピードアップ
- コスト削減
- 業務効率化
- 現場ニーズへの対応
- 顧客満足度の向上

アプリケーション開発の苦労点と解決方法

当社基幹システムは、EDI 受注による自動ピッキング・自動倉庫へのデータ連携・自動発注 FAX 連携など、多様なインフラとの連携、営業実績のすばやい把握、情報の迅速な共有化が必須であった。

また、24 時間 365 日の稼働が前提であり、停止のないシステム構築が必要であった。

解決戦略

①自動ピッキング処理の共通化。Delphi のメニューからの起動により、多種多様な顧客へのピッキング帳票出力を実現。
②関西流通センターとの連携は、入荷検品入力での FTP 転送を組み合わせ、リアルな連携を実現。
③自動発注 FAX は、Delphi/400 から FAX Press のデータ連携により、自動化を実現。
④営業実績の把握は、Delphi と VB Report の組み合わせにより、各営業担当による実績資料作成を実現。
⑤ Delphi の特徴であるタブ切替、画像の表示の機能を活用し、データを提供。

具体的な実現施策

①自動ピッキング処理の共通化

多種多様のピッキング条件、ピッキング帳票を分析分類し、条件をファイルにパッケージし、共通プログラムで処理することにより流れを統一した。とともに、Delphi も 1 画面の指示で、複数ピッキングの処理が可能となった。

この機能を構築したことにより、新規の顧客の発生に対してはファイル登録することですみ、プログラム作成が減少した。Delphi においても、メニューへの指示画面を登録するのみで対応できる。

自動ピッキングも、Delphi/400 で起動している CL を IBM i のバッチに投入することにより、同一プログラムを活用することができる。

【解決ポイント】
条件の統合と条件ファイルの組み合わせ、Delphiでの実装方法を確定するまでに、ユーザーからの情報収集に時間を要した。

②関西流通センターとの連携
Delphiの入荷検品入力とIBM iのFTP転送機能を活用し、入力終了と同時に自動倉庫にデータが連携され、自動倉庫への入荷動作がリアルに稼働する。
【解決ポイント】
自動倉庫へのデータ転送から動作するまでのタイミングについて、そのロス時間の調整にテストと時間を要した。

③自動発注FAX
FAX Press Serverに、Delphi/400の常駐JOBを待機させる。これにより、基幹システムの発注入力で、発注データを自動検知し自動FAX送付を可能とした。
【解決ポイント】
FAX Pressの特徴である既定フォルダにファイル（テキスト）を転送し、監視機能を活用するうえで、Delphiで、送付時にいかにデータをチェックし、連携の不具合が発生しないようにさせるか、ポイントはこの実現にあった。

④営業実績の把握
各営業担当は、Excel等のPC関連ツールの操作は長けている。そのため、違和感のないシステム構築を行う必要があり、DelphiとVB Reportの組み合わせを選択した。
これにより、帳票出力だけではなく、Excel保存が可能となり、営業担当者のスムーズな稼働が実現できた。
【解決ポイント】
営業担当が実績把握するために、日頃から使いなれたExcelと同等の操作性で、基幹システムからリアル実績を把握することがポイントであり、これを可能にした。

⑤Delphiのタブ切替、画像表示機能
基幹システムの全般の画面構築基準にタブ切替を採用し、これまでの複数画面遷移による情報確認のための煩雑なシステム操作から脱却することができた。
また品目については、画像コンポーネントを使用した。これにより、品目の問い合わせ時に製品が一目で判別でき、これまでのコード・品名の文字情報での確認から、視覚による確認が実現できた。
【解決ポイント】
タブ切替は、タブ内での情報分類の方向性の決定、検索対象のまとめ方で、必須の情報とサブ情報の切り分けを行えた。

獲得したノウハウ・ハウツウ・教訓

今回のシステム構築により獲得したノウハウ、ハウツウ、教訓を述べる。

【獲得Know How】
①量販店とのオンライン受注、支払い照合の機能、および銘柄のノウハウ（文字コード変換の特殊性）
②割戻し（契約された売上金額見合いの値引き）処理のノウハウ
③受注から出荷までのタイムラグを極小化するためのノウハウ（自動運転とプリンタ制御）
④DWH（データウェアハウス：DIAPLISM）へのデータ連携による、エンドユーザー活用
（売上DB、仕入DB、受払DB、日別得意先品目DB、日別品目倉庫DB、月別在庫DB、月別利益集計DB、月報DB、月別得意先品目DB）
⑤需要予測（過去3年の全社売上実績による売上予測数の算出、全社品目合計のバラ数量）のノウハウ
⑥自動倉庫（WMS）との、入荷・出荷・棚卸のデータ連携のノウハウ
⑦自動FAX連携（FAX Press + SVF）の連携ノウハウ
⑧電子ファイリング連携（e-image + SVF）の連携ノウハウ

【獲得How to】
・在庫更新ロジックの一元管理など、プログラムの構造化による生産性UP・品質確保
・各種伝票発行の、コントロールマスタのパラメータ制御による生産性UP

【教訓】
ユーザー電算室開発システムのきめ細やかさ、現場への適合性の高さは、想像以上であったこと。

開発経緯

1998.1　関西流通センター（新関西センター：自動倉庫）開設
関西流通センター向けシステム稼働
三菱電機製RX7000/α900の導入
受注・出荷・売上・発注・入荷・仕入れ・在庫管理システムの大幅見直し
AUTO-FAXシステム稼働開始
経理、給与システム稼働開始（ミロク情報サービス社製）
2000　システムの2000年対応
2003.1　FAXシステム拡張、電子帳票導入
AS400/800（オンライン受注）導入

＜新基幹システム再構築＞
2003.10　要件定義開始
2004.1　要件定義終了
2004.2　基本設計開始
2004.4　詳細設計開始
2004.8　詳細設計終了
2004.9　開発開始
2004.11　ステアリングコミッティにて2005.7本稼働予定
2005.11　本稼働

ユーザー評価と今後

エンドユーザーの評価と、今後の予定・計画を述べる。

【エンドユーザーの評価】
・処理画面のGUI化により、操作性の向上が実現できた。
・1画面で、あらゆる情報を見ることができるようになった。
・画像や写真が画面で確認でき、文字情報だけであったのが、直感的に判別できるようになった。
・ExcelやCSV出力が容易であり、情報分析がすばやくできるようになった。
・システムのスリム化により、業務処理の改善ができた。
・IBM iとの連携により、処理速度が速くなった。
・月次決算の短期化

・データ化推進により、ペーパーレス化が実現できた。
・エンドユーザーの任意出力が可能になり、省力化ができた。(従来情報システムでは帳票を一括出力し、各部署へ配布していた)。
・安定した処理運用の確立(止まらないシステム)。

【今後の予定・計画】

本社の販売管理の構築 / 運用が軌道にのった現在、関西工場での生産管理システムの統合を検討している。

関西工場では、自社製品の製造を行っており、現在は販売管理システムとは一部連携はしているが、単独でのシステム運用となっている。

製品の出来高や在庫の状況をリアルに把握することで、よりお客様のニーズにお応えできると考えている。また、過剰 / 不要な在庫の削減も期待できる効果として検討をすすめている。

M

技研化成株式会社
http://www.gikenkasei.co.jp/

本社:兵庫県尼崎市(東京営業所、名古屋営業所)
関西工場:兵庫県加西市
関西流通センター:兵庫県加西市

Migaro Technical Report 2008

SE 論文／ミガロ テクニカルレポート

畑中 侑

株式会社ミガロ

システム事業部 システム2課

はじめてのDelphi/400プログラミング

Delphi/400 を使用してアプリケーションを開発する場合
まず何をすればいいのだろうか。
Delphi/400 を使うからといって
何も特別なことをするわけではない。
要点は 2 つである。
IBM i への接続方法と既存資産の利用方法を押さえておけばいいのである。
ここでは、照会機能を持った簡単なアプリケーション開発を事例に挙げ
はじめての Delphi/400 アプリケーション開発を体感できるようにした。

略歴
1983 年 7 月 6 日 生
2006 年京都産業大学 法学部卒
2006 年株式会社ミガロ入社
2006 年 4 月システム事業部配属

現在の仕事内容
システム受託開発に携わって 3 年目。ミガロに入社し初めてプログラムを作成するも、今では担当顧客も持ち、小規模から中規模案件のリーダーや大規模案件のサブリーダーを務めるに至る。
Delphi/400 や RPG などのプログラム開発経験を積みながら、スキルを磨きつつ、お客様のご要望に耳を傾け、一歩一歩提案力をつけるため修行中の日々。

Delphi/400を使おう

新しくアプリケーションを開発するにあたって、まずは要件を整理する意味も含め必要な機能の洗い出し、それら機能ごとのつながりを画面遷移図としてまとめてみよう。

機能については IBM i のファイルに対して何を行うのかで、大きく 3 つに種類分けを行うことが可能である。

- **照会機能**：IBM i のファイルを、さまざまな抽出条件で参照する。
- **更新機能**：IBM i のファイルに新しいレコードを追加したり、すでにあるレコードを更新、または削除する。
- **支援機能**：上記の 2 機能のほか、検索や帳票出力、異なるアプリケーションの呼び出しや連携などを行う。

今回は「はじめての Delphi/400」ということで、IBM i 上にあるファイルを参照する照会機能を備えたアプリケーションの開発についてまとめることにする。

アプリケーションの全体像をイメージする

今回作るアプリケーションの構成は、全部で 3 画面となる。

まずは、アプリケーションの起動時に最初に表示させる画面として、サインオンダイアログを準備する。これは、どんなアプリケーションを作ろうと、IBM i のファイルを参照するのであれば、IBM i への接続が必要不可欠だからだ。今回のアプリケーションでは、ユーザーに ID とパスワードを入力してもらい、その入力値をもとに IBM i との接続を行うことにする。

無事に IBM i との接続ができれば、次画面として機能を統括するメニューを表示させ、このメニューからさまざまな機能へ展開できるようにしよう。単一の機能しかないアプリケーションであれば、メニューを作成しなくてもよいが、将来機能を増やすことも想定してメニュー画面を用意することにする。

そして、メニュー画面で「照会画面へ」を押下することで、メインとなる IBM i 上のファイルを照会する 3 画面目を表示させる。

上記の内容を、遷移図にまとめた。全体像がイメージできたので、さっそく開発に着手していこう。【図 1】

アプリケーションの基本部分を作る 1
～IBM i との接続（画面の設定）

Delphi/400 アプリケーションを作成するうえで、IBM i への接続（サインオン）が必要になる。データベースアプリケーションでは、データベースへサインオンするために、ユーザーとパスワードの情報が必要となる。もちろん Delphi/400 も例外ではない。

サインオンするためのユーザー / パスワードを設定する方法は、大きく分けて 2 つの手法がある。

①アプリケーション上で、接続情報を入力する（明示的サインオン）

図1　概要図

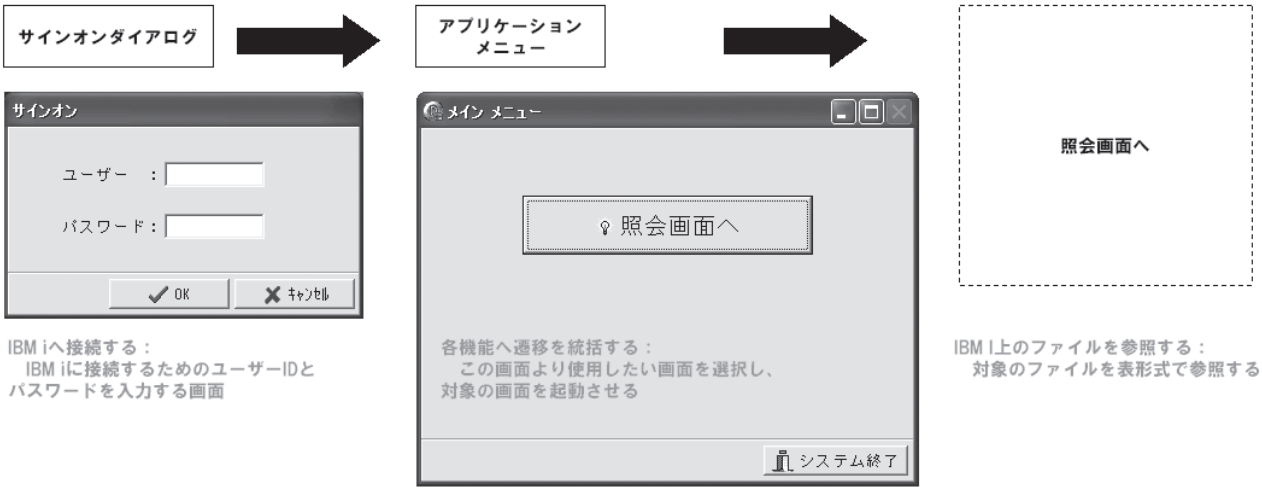
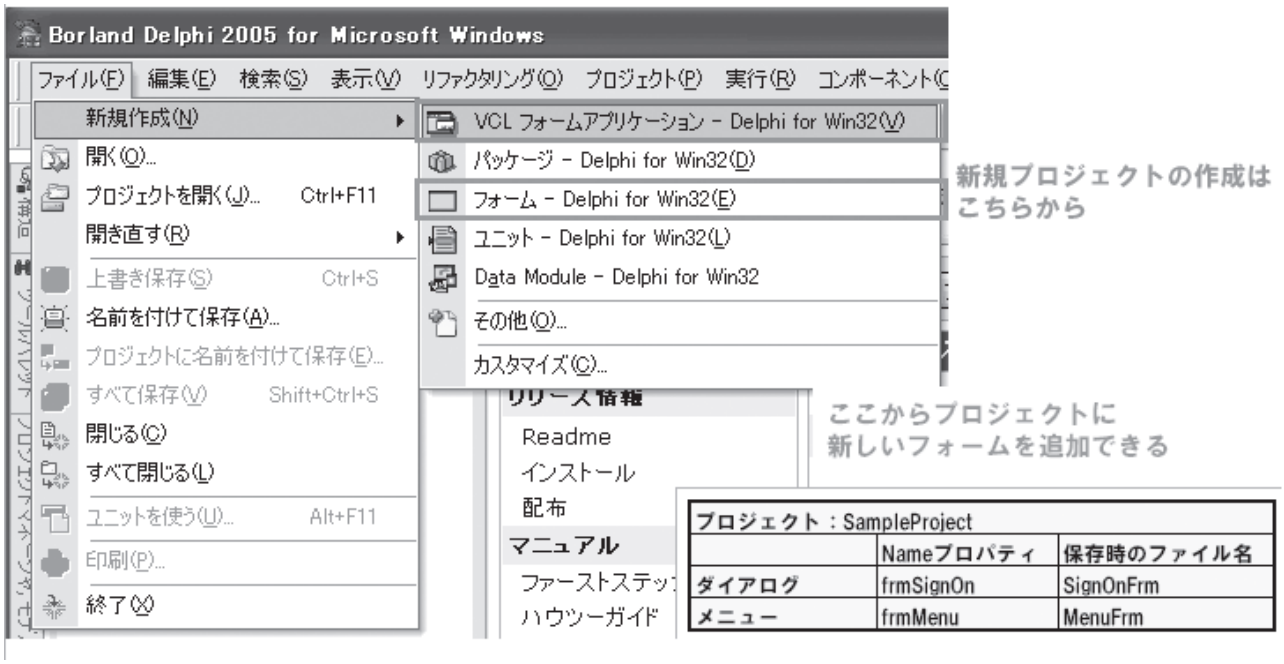
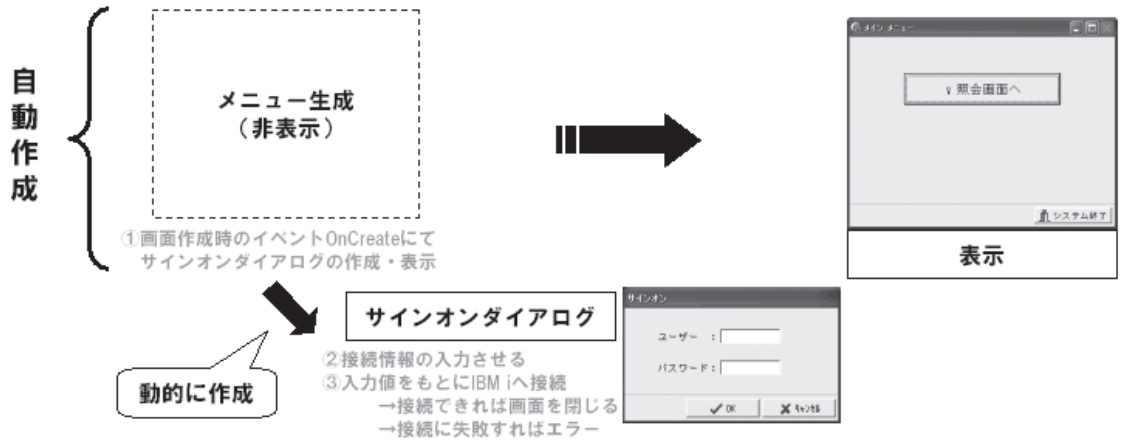


図2　プロジェクト追加



プロジェクト：SampleProject		
	Nameプロパティ	保存時のファイル名
ダイアログ	frmSignOn	SignOnFrm
メニュー	frmMenu	MenuFrm

図3　作成の流れ



サインオンダイアログ画面よりユーザー / パスワードを入力した内容で、接続を行う。

②プログラム上で、内部的に接続情報を設定する（暗黙のサインオン）

プログラム上で固定のユーザー / パスワードを設定しておくか、応用して外部ファイル（ini ファイル等）からユーザー / パスワードを読み込んで、接続を行う。

今回は①の手法を使って開発を行う。それでは、接続に成功した後にメニューを表示させる部分までを含め、作ってみよう。

プロジェクト追加

Delphi のメニューから［ファイル］［新規作成］［VCL フォームアプリケーション］を選択する。すると、画面 Unit1 を持つ新規プロジェクト Project1 が作成できる

まず、Unit1 の画面にあたる Form1 の Name プロパティを「frmSignOn」と変更し、続けて Delphi のメニューから［ファイル］の［名前を付けて保存］と、［プロジェクトに名前を付けて保存］より、Unit1 を「SignOnFrm」、Project1 を「SampleProject」と名付け、適当なフォルダに保存する。

保存ができたら、同様に、使用するメニューを同じプロジェクトに追加しておこう。【図 2】

なお、Form の Name プロパティと Unit のファイル名には、同じ名前を指定できないので注意が必要である。

作成の流れ

これで、接続までの流れで使用する画面が用意できた。

というわけだが、プロジェクトに追加されたフォームは、初期設定では起動時に自動作成されるようになっている。起動時に自動作成されるということは、その先に遷移しない可能性のある画面をも起動時に作成してしまい、無駄が発生するということである。

そこで Delphi のメニューから［プロジェクト］［オプション］を選択し、ツリーメニューのフォーム項目から自動作成の対象を必要最低限に設定しておくこと。

今回は、メニューを設定し、メニューの生成時にサインオンダイアログを表示し、入力された情報をもとに IBM i に接続、その後メニューを表示という流れにする。【図 3】【図 4】

また、メニューがサインオンダイアログを使用するには、参照関係を設定する必要がある。Delphi のメニューより［ファイル］［ユニットを使う］を選択し、サインオンダイアログを参照できるように設定する。【図 5】

コンポーネント

これで土台となる必要な画面が用意できた。ここからは、部品となるコンポーネントを、ツールパレットより各フォームに貼り付けていく。

【使用するコンポーネント】

● frmSignOn

・TPanel：ツールパレット（Standard）
コンテナ（土台）コンポーネント。
・TEdit：ツールパレット（Standard）
ユーザーに入力域を提供する、標準的なコンポーネント。
・TBitBtn：ツールパレット（Additional）
ビットマップを表示できるボタン。よく使用される動作がビットマップ付きで実装されている。

● frmMenu

・TAS400：ツールパレット（SCD400 Data）
IBM i とクライアントとの通信を管理するコンポーネント。コマンドを発行することも可能。
・TDataBase：ツールパレット（BDE）
BDE を使用したアプリケーションが、データベースに接続する。その管理を行うコンポーネント。
・TButton：ツールパレット（Standard）
標準的なプッシュボタン。

コンポーネントの貼り付けが終われば、それぞれのプロパティ値を設定していく。

まず、IBM i への接続の要となる AS400 コンポーネントであるが、特に変更するところはない。次にデータベースコンポーネントでは、今回はサインオンダイアログを作成するので、Login Prompt プロパティを False にしておく。【図 6-1】【図 6-2】

あとはビットボタンの Kind プロパティを変更する程度で、デフォルトの値を使う。（Kind プロパティを変更すると、それにあった外観と応答結果が自動で変更される点に留意してほしい。必要に応じてプロパティを再変更することを覚えておくといいだろう）。【図 6-2】

アプリケーションの基本部分を作る 2
～IBM i との接続（接続処理）

以上で一通りの準備が完了できた。いよいよ IBM i との接続処理をプログラミングしていく。

Delphi/400 を使った IBM i への接続は、AS400 コンポーネント、データベースコンポーネントのユーザー ID およびパスワードのプロパティにそれぞれの値をセットし、各コンポーネントを Open するだけのシンプルなプログラミングのみで可能となる。

また、AS400 コンポーネントではコマンドの発行ができるので、接続後には「ADDLIBLE」コマンドからライブラリー環境の設定を行うことも可能である。今回は初期値を「＊ LIBL」とし、「ADDLIBLE」コマンドより、対象のライブラリーをセッションのライブラリーリストに追加する。

これを踏まえて図 7 のように、アプリケーション起動時に自動生成されるメニューの OnCreate イベントで、サインオンダイアログを表示させて接続を行うようにする。ここでは、受け取った引数をもとに接続処理を行い、ライブラリー環境を設定する一連の処理を、関数［OpenDatabase］としてメニューに作成している。【図 7】【ソース 1】【ソース 2】

以上のプログラミングで、Delphi/400 を使って IBM i に接続を行い、アプリケーション終了時までその接続状態を持続する基本部分が完成した。接続後はもちろん、5250 画面の「WRKACTJOB」コマンドで確認できる。（このプログラムでは接続先を指定していないので、Delphi/400 Configuration の「AS/400List」にある最上部の IBM i へ接続される）。

図4　自動作成

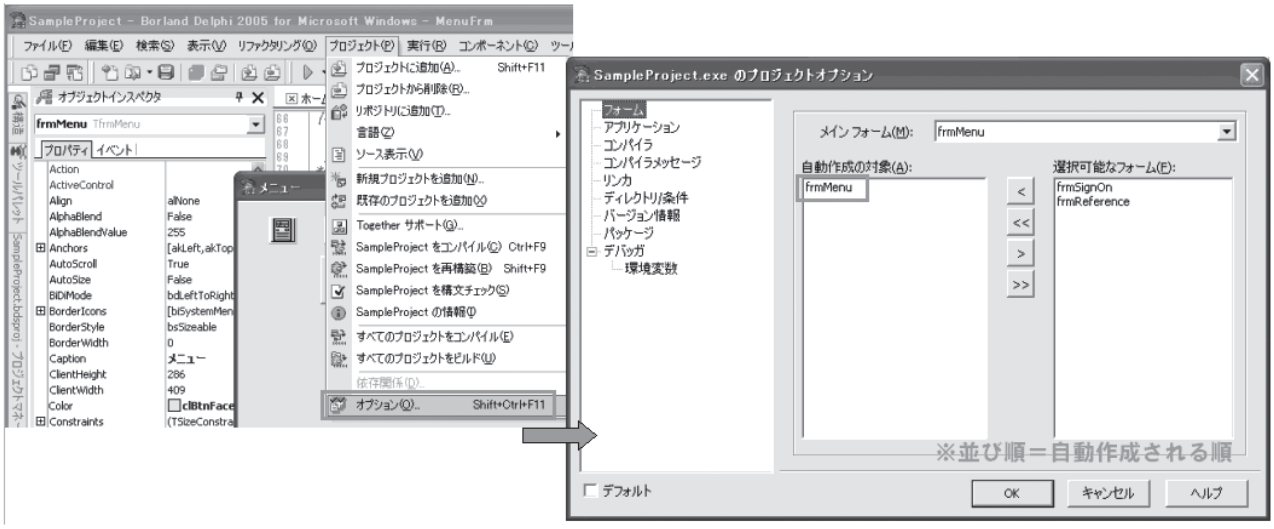


図5　ユニットの使用

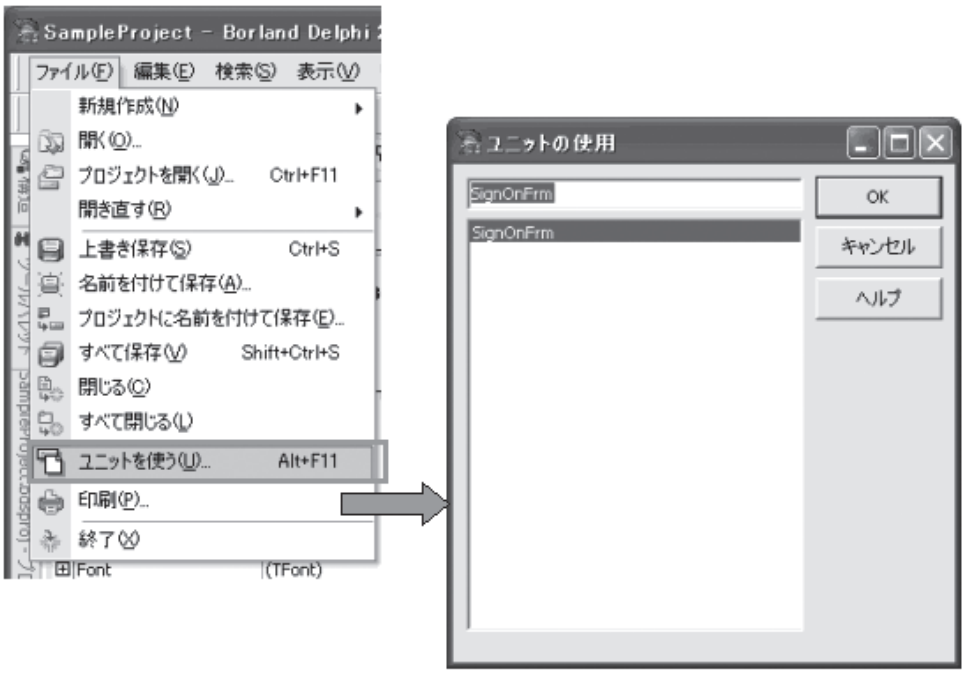
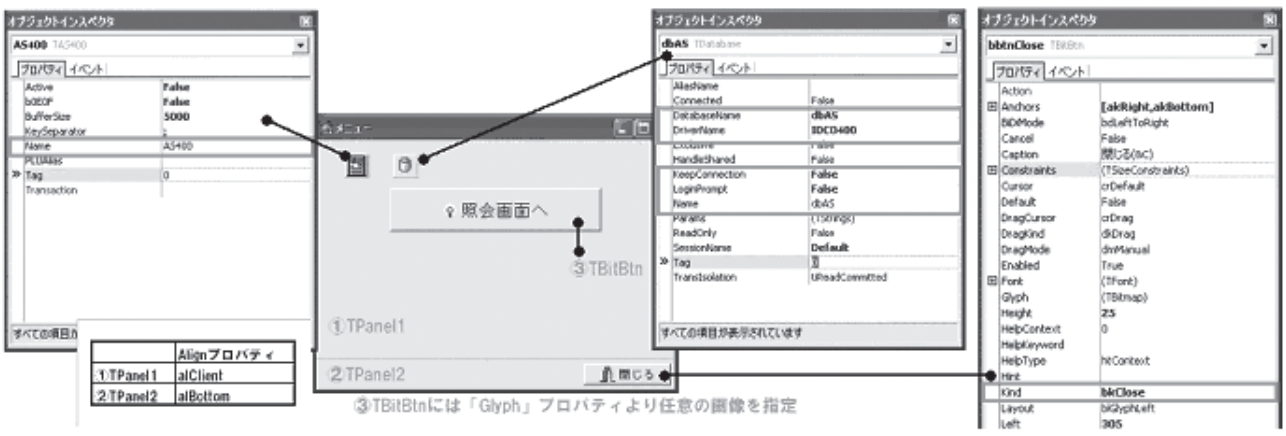


図6-1　プロパティ



メイン機能の実装 ～データ抽出処理

アプリケーションの基本部分が完成したら、いよいよメインの照会機能のプログラミングに取りかかろう。

Delphi/400 を利用した IBM i のファイル参照は、大きく分けて 2 種類の方法がある。

① SQL でデータを取得する方法
②ファイル単位で取得する方法

今回は、すでに IBM i 側に、RPG プログラムで作成された照会システムがあることを前提とする。CL プログラムや RPG プログラムとの連携で実現する②の手法を使って、開発を行い、IBM i 資産を利用する方法について紹介しよう。

既存プログラムの利用

すでに IBM i にある照会機能の GUI 化を Delphi/400 で行う場合、特に工数の削減が見込めるのが、既存プログラムを利用するという方法である。

図 8 を見てほしい。既存プログラムについて抽出結果をサブファイルに出力する代わりに、ワークファイルに出力するようにメンテナンスするだけで、既存の RPG/CL プログラムを流用することができる。

その際、ワークファイルのフィールド名を画面ファイルの項目と同じにすることで、変更を最小限に抑えることが可能である。

さらに、Delphi/400 プログラムから IBM i のプログラムを呼び出すために、CL プログラムを経由するというポイントも押さえていただきたい。CL において「MONMSG」を使用することで、呼び出す RPG プログラムの MSGW を回避することができるのである。

また、CL プログラムに、抽出条件のパラメータ以外にエラー区分のパラメータを持たせることで、対象データがない場合などのエラー発生を、Delphi/400 側に返すことが可能となる。【図 8-1】

今回は、図 8-2 のように、画面起動時に QTEMP 上にワークファイルを作る CL プログラムを実行し、条件入力後に検索ボタンを押下することで、メインの抽出処理 RPG プログラムを呼び出す CL プログラムを実行することとする。【図 8-2】【表】【ソース 3】【ソース 4】【ソース 5】

【使用するコンポーネント】

● frmReference（フォームを新規作成しメニューから呼び出す）

・TMaskEdit：ツールパレット（Additional）
入力された文字を指定した表示形式で検証し、表示する。

・TCall400：ツールパレット（SCD400 System）
IBM i 上のプログラムを呼び出す。パラメータはプログラムからセットすることも、実行後に返された値を参照することも可能。

・TTable：ツールパレット（BDE）
データベース上のファイルを保持・操作するデータセット。

・TDataSource：ツールパレット（Data Access）
データセットコンポーネントと、画面上にあるデータベース対応コンポーネントをつなぐ役割。

・TDBGrid：ツールパレット（Data Controls）
対象のデータセットより、レコードを表形式で表示し、操作する。

なお、照会画面はメニューを参照できるようユニット参照を行い、また、プロジェクトオプションで、自動作成の対象より除外すること。その他の、既出のコンポーネントについては割愛した。

プロパティ

プロパティは、図 9-1 と図 9-2 を参照してほしい。【図 9-1】【図 9-2】

ここでのポイントは、Call400 コンポーネントで、IBM i の CL プログラムへパラメータを渡すので、同数のパラメータ定義および各値の桁数と属性を指定する必要があるということだ。（ただし、属性については、プログラミング上は、全て文字型（String）でセットする点に注意）。【ソース 5】

ここで定義したパラメータに、プログラムから動的に値をセットし、CL プログラムを呼ぶのである。また、テーブルコンポーネントには、参照対象となるファイル名を設定する。そして、いずれのコンポーネントの AS400 プロパティにも、メニューに配置した AS400 コンポーネントを指定することをあわせて確認しておこう。

CLプログラム

さて、各コンポーネントの準備ができたので、プログラミングを行っていこう。

と、いっても、Delphi/400 から CL プログラムを実行し、対象データが更新されたワークファイルを参照するというのは、いたってシンプルなプログラミングで済む。

Call400 コンポーネントに設定した各パラメータに、ただ画面入力値をセットし、実行した後に、プロパティが正しくセットされたテーブルコンポーネントを開くだけで、画面に表示できるのである。【図 10】【ソース 6】

これで全てのプログラムが完成した。Delphi のメニューより［プロジェクト］［SampleProject をコンパイル］を選択し、エラーがないことを確認した後に、実行ファイル SampleProject.exe を実行してみよう。そして、ぜひとも IBM i 上にあるファイルを参照できていることを、ご自身の目で確かめていただきたい。

さいごに

以上で、照会機能を想定し、Delphi/400 を使用する一連のアプリケーション開発についての説明は終わりである。冒頭で述べたように「はじめての Delphi/400」と題したが、IBM i への接続、既存資産の利用方法が把握できていると、GUI 化は思うがままなのである。

とはいえ、アプリケーションからの SQL 実行により、ファイルの操作を簡単に行うことも可能である。もちろん Delphi/400 を使用した接続なので、早いレスポンスが期待できる。

この「はじめての Delphi/400」が、Delphi/400 ユーザーである開発者の増加の一助になれば幸いである。

M

図6-2　プロパティ

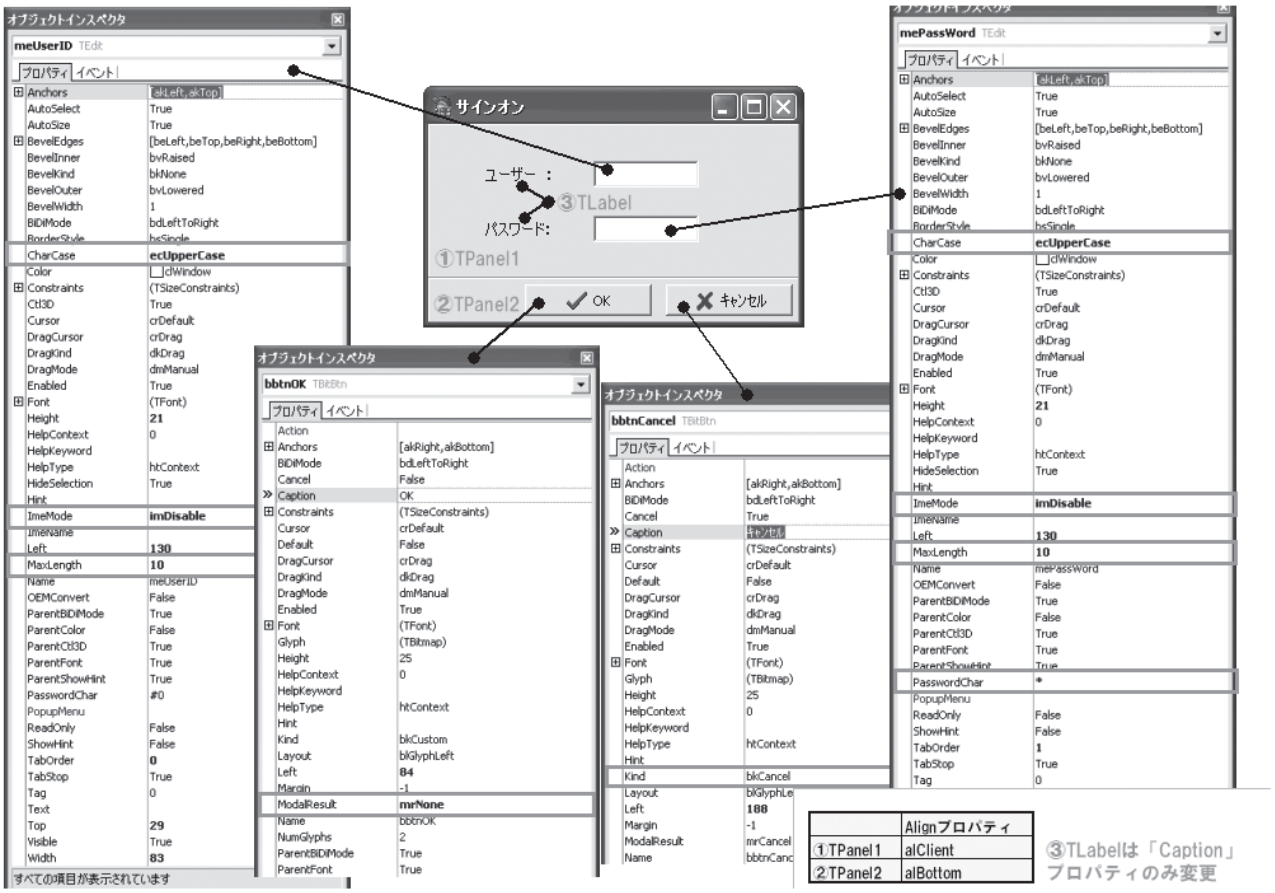


図7　各フォームのイベント

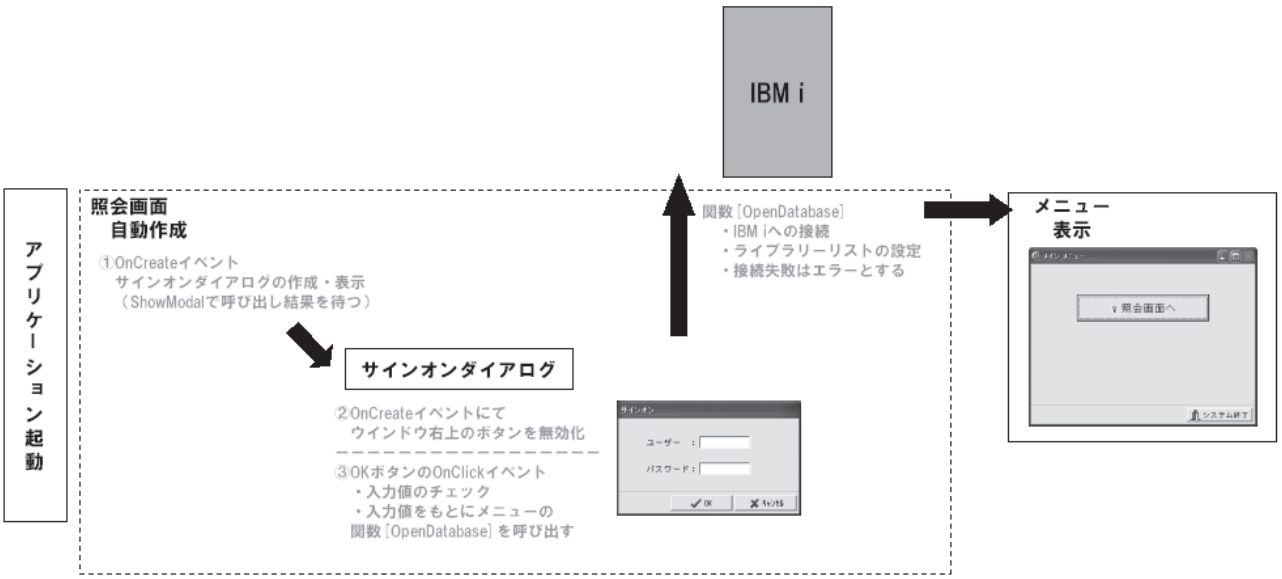


図8-1　既存資産の流用

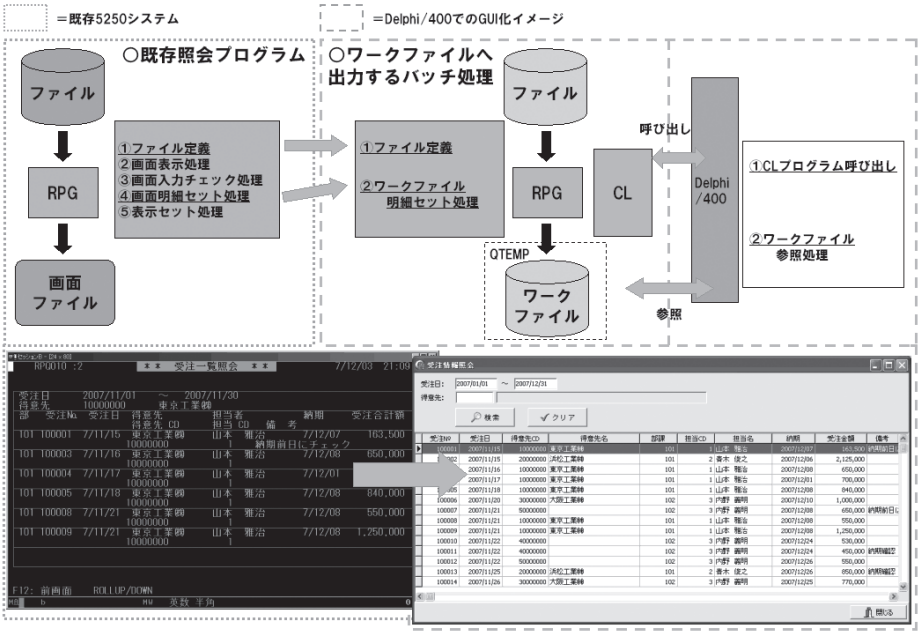


図8-2　既存資産の流用

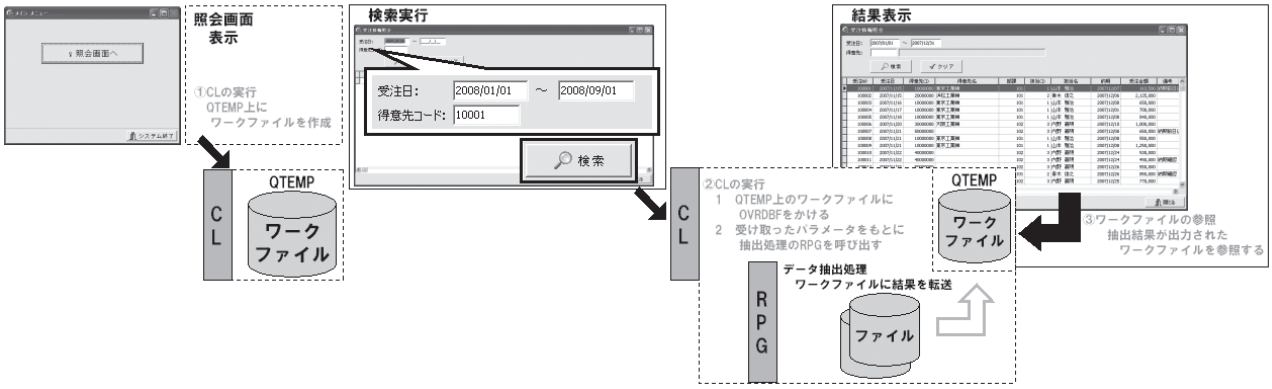


図9-1　プロパティ

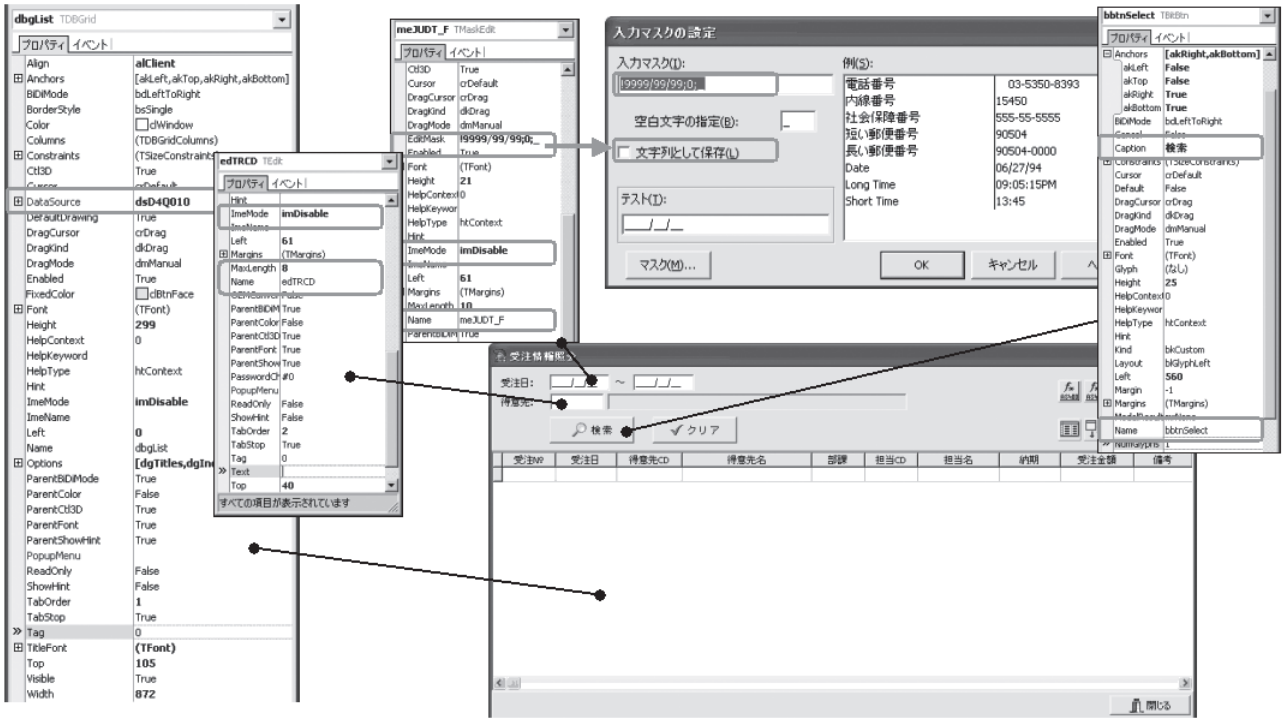


図9-2　プロパティ

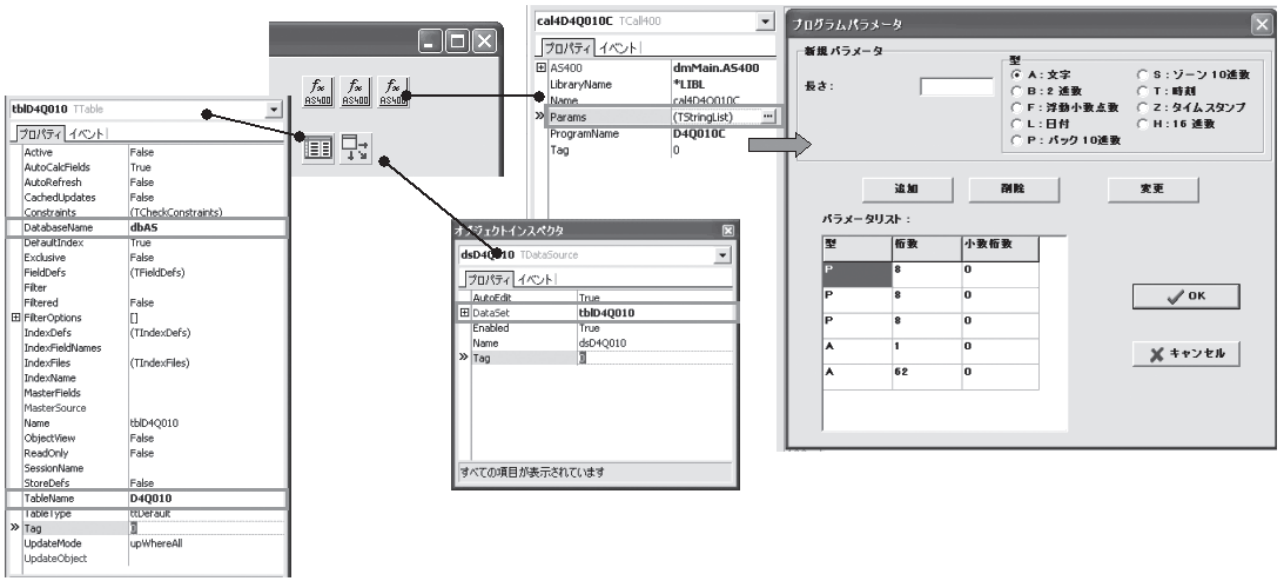


図10　照会の流れ

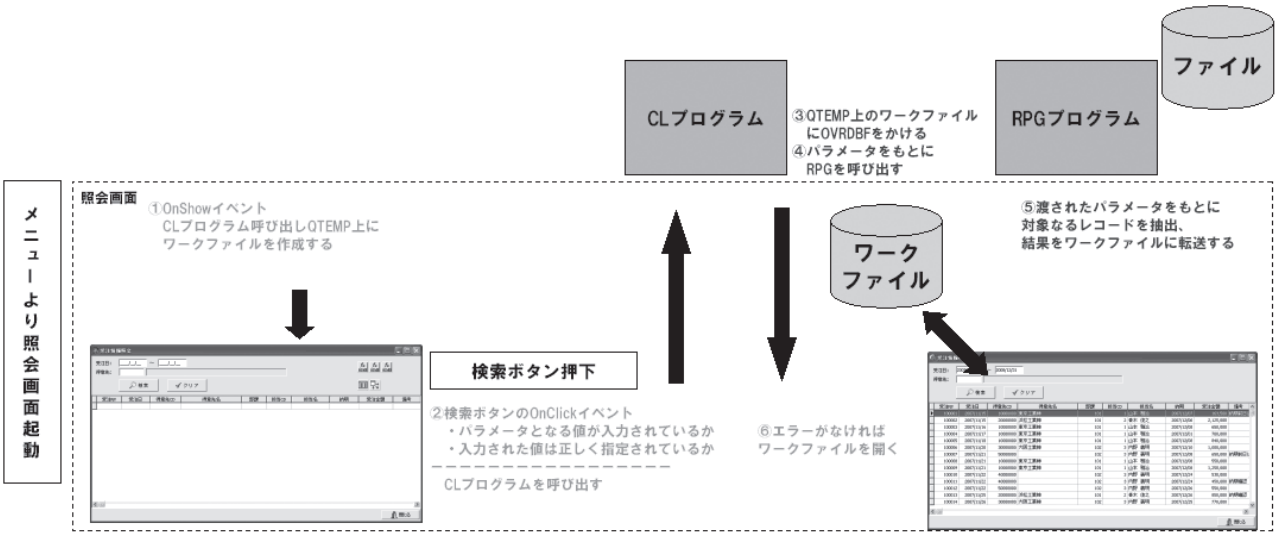


表　IBM iプログラムおよびファイルレイアウト

プログラム一覧		
	ID	名称
CL	D4Q010C	受注一覧照会（呼出）
	D4Q011C	受注一覧照会ＱＴＥＭＰ作成
RPG	D4Q010	受注一覧照会ワーク作成

ファイルレイアウト				
ファイル名：受注一覧ワーク　　ファイルID：D4Q010				
項目名	桁数		属性	テキスト記述／欄見出し
HWJGCD	4	0	S	部課
HWJUNO	6	0	S	受注No.
HWJUDT	8	0	S	受注日
HWTRRK	14		O	得意先名
HWTANM	16		O	営業担当者名
HWNOKI	8	0	S	納期日
HWJUGK	9	0	P	受注金額
HWTKCD	8	0	S	得意先 CD
HWTACD	5	0	S	担当者 CD
HWBIKO	32		O	備考

ソース1　サインオンダイアログ

```
{#############################################################################

  サインオンダイアログ

#############################################################################}
unit SignOnFrm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls;

type
  TfrmSignOn = class(TForm)
    pnlBottom: TPanel;
    bbtnOK: TBitBtn;
    bbtnCancel: TBitBtn;
    pnlMain: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    meUserID: TEdit;
    mePassWord: TEdit;
    procedure bbtnOKClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  end;

var
  frmSignOn: TfrmSignOn;

implementation

uses MenuFrm;

{$R *.dfm}

{*****************************************************************************
 目的：画面作成時
 引数：
 戻値：
*****************************************************************************}
procedure TfrmSignOn.FormCreate(Sender: TObject);
var
  SysMenu: HMENU;
begin
  //閉じるボタンを無効にする
  SysMenu := GetSystemMenu(Handle, False);
  EnableMenuItem(SysMenu, SC_CLOSE, MF_BYCOMMAND or MF_GRAYED);
  SysMenu := GetSystemMenu(Application.Handle, False);
  EnableMenuItem(SysMenu, SC_CLOSE, MF_BYCOMMAND or MF_GRAYED);
end;

{*****************************************************************************
 目的：ＯＫ処理
 引数：
 戻値：
*****************************************************************************}
procedure TfrmSignOn.bbtnOKClick(Sender: TObject);
begin
  inherited;
  // ユーザーID未入力チェック
  if meUserID.Text = '' then
  begin
    meUserID.SetFocus;
    raise Exception.Create('ユーザーIDを入力して下さい。');
  end;

  // パスワード未入力チェック
  if mePassWord.Text = '' then
  begin
    mePassWord.SetFocus;
    raise Exception.Create('パスワードを入力して下さい。');
  end;

  // ユーザーIDへフォーカス移動
  meUserID.SetFocus;

  // AS/400への接続処理
  frmMenu.OpenDatabase(meUserID.Text, mePassWord.Text);

  // 接続できれば終了
  ModalResult := mrOk;
end;

end.
```

【図Ⅶ】②
ウインドウ右上のボタンを無効化

【図Ⅶ】③
入力チェック

【図Ⅶ】③
メニュー【OpenDatabase】の
呼び出しにてIBM iへ接続

ソース2　メニュー

```
{#############################################################################

 メニュー

#############################################################################}
unit MenuFrm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, DB, DBTables, Scdconn;

type
  TfrmMenu = class(TForm)
    pnlBottom: TPanel;
    bbtnClose: TBitBtn;
    pnlMain: TPanel;
    BitBtn1: TBitBtn;
    AS400: TAS400;
    dbAS: TDatabase;
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private 宣言 }
  public
    { Public 宣言 }
    // AS/400への接続処理
    procedure OpenDatabase(AUserID, APassWord :String);
  end;

var
  frmMenu: TfrmMenu;

implementation

uses ReferenceFrm, SignOnFrm;

{$R *.dfm}

{*****************************************************************************
 目的：メニューフォーム作成時処理
 引数：
 戻値：
*****************************************************************************}
procedure TfrmMenu.FormCreate(Sender: TObject);
begin
    // サインオンダイアログを表示
  frmSignOn := TfrmSignOn.Create(Application);
  try
    // サインオンダイアログより接続結果を受け取り
    // 接続出来なかった場合、アプリケーションを終了する
    if frmSignOn.ShowModal <> mrOK then
    begin
      Application.ShowMainForm := False;
      Application.Terminate;
      Abort;
    end;
  finally
    frmSignOn.Release;
  end;
end;

{*****************************************************************************
 目的：メニューフォーム破棄時処理
 引数：
 戻値：
*****************************************************************************}
procedure TfrmMenu.FormDestroy(Sender: TObject);
begin
  // TDatabase切断処理
  dbAS.Connected := False;

  // TAS400切断処理
  AS400.Active := False;
end;

```

【図Ⅶ】
関数【OpenDatabase】宣言部分

【図Ⅶ】①
サインオンダイアログの表示

```
78
79  {*************************************************************************
80   目的：照会画面遷移ボタン押下時処理
81   引数：
82   戻値：
83  *************************************************************************}
84  procedure TfrmMenu.BitBtn1Click(Sender: TObject);
85  begin
86    inherited;
87    // 照会画面の呼び出し
88    frmReference := TfrmReference.Create(Self);
89    try
90      Self.Hide;
91      frmReference.ShowModal;
92    finally
93      frmReference.Release;
94      Self.Show;
95    end;
96
97  end;
98
99  {*************************************************************************
100  目的：IBM iへの接続処理
101  引数：
102  戻値：
103 *************************************************************************}
104 procedure TfrmMenu.OpenDatabase(AUserID, APassWord :String);
105 var
106   sMsg  :String;
107 begin
108   // 初期化処理
109   // TAS400の切断
110   if AS400.Active then AS400.Active := False;
111   // TDatabaseの切断
112   if dbAS.Connected then dbAS.Connected := False;
113
114   // TAS400の設定
115   with AS400 do
116   begin
117     // ユーザーID
118     Userid := AUserID;
119     // パスワード
120     PWD := APassWord;
121   end;
122
123   // TDatabaseの設定
124   with dbAS do
125   begin
126     // ライブラリー
127     Params.Values['LIBRARY NAME'] := '*LIBL';  {※「*LIBL」とすることで接続されたセッション
128     // ユーザーID                                  のライブラリーリストが扱えるようになる}
129     Params.Values['USER NAME']:= AUserID;
130     // パスワード
131     Params.Values['PASSWORD'] := APassWord;
132     // ログインプロンプトの表示
133     LoginPrompt := False;
134   end;
135
136   // データベース接続処理
137   try
138     AS400.Active := True;
139     dbAS.Connected := True;
140   except
141     // エラー発生時
142     sMsg := 'AS/400に接続することができませんでした。';
143     sMsg := sMsg + #13#10 + 'ユーザー名、パスワードを確認して下さい。';
144     raise Exception.Create(sMsg);
145   end;
146
147   //ライブラリーリストの追加（ここでは"TESTLIB"とし、QTEMPのうしろに追加）
148   try
149     AS400.RemoteCmd('ADDLIBLE LIB(TESTLIB) POSITION(*AFTER QTEMP)');
150   except
151     // エラー発生時
152     on E: Exception do
153     begin
154       raise Exception.Create(E.Message);
155     end;
156   end;
157 end;
158
159 end.
```

メニュー
「照会画面へ」ボタン押下時

【図VII】
関数【OpenDatabase】IBM iへの接続

ソース3　CL1

```
    /*****************************************************************
    /*  システム  :受注照会機能                                      *
    /* NAME       : 受注一覧照会 (呼出)                              *
    /*****************************************************************
            PGM         PARM(&PJUDF  +
                             &PJUDT  +
                             &PTKCD  +
                             &PPERR  +
                             &PPMSG  )
/* ＣＬ変数定義 */
            /* 受注日  FROM                (数値８桁)  */
            DCL         VAR(&PJUDF) TYPE(*DEC)  LEN(8 0)
            /* 受注日  TO                  (数値８桁)  */
            DCL         VAR(&PJUDT) TYPE(*DEC)  LEN(8 0)
            /* 得意先コード                (数値８桁)  */
            DCL         VAR(&PTKCD) TYPE(*DEC)  LEN(8 0)
            /* エラー                      (文字１桁)  */
            DCL         VAR(&PPERR) TYPE(*CHAR) LEN(1)
            /* メッセージ                  (文字６２桁)  */
            DCL         VAR(&PPMSG) TYPE(*CHAR) LEN(62)
   /*-------------*/
   /* QTEMP       */
   /*-------------*/
            CALL D4Q011C
   /*-----------------*/
   /*  更新処理       */
   /*-----------------*/
            OVRDBF      FILE(D4Q010) TOFILE(QTEMP/D4Q010) LVLCHK(*NO)
            MONMSG      MSGID(CPF0000)
            CALL        PGM(D4Q010)  PARM(&PJUDF  +
                                          &PJUDT  +
                                          &PTKCD  +
                                          &PPERR  +
                                          &PPMSG  )
            DLTOVR FILE(D4Q010)
            MONMSG      MSGID(CPF0000)
   /*-----------------*/
   /*  初期モード設定  */
   /*-----------------*/
            RETURN
            ENDPGM
```

ソース4　CL2

```
    /*****************************************************************
    /*  システム  : 受注照会機能                                     *
    /* NAME       : 受注一覧照会ＱＴＥＭＰ作成                      *
    /*****************************************************************
#START:     PGM
    /* 変数宣言 */
            DCL         VAR(&SRCLIB) TYPE(*CHAR) LEN(10)
    /* パラメーター */
            RTVDTAARA   DTAARA(D4DTAARA (1 10)) RTNVAR(&SRCLIB)
    /* オブジェクト確認 */
            CHKOBJ      OBJ(QTEMP/D4Q010) OBJTYPE(*FILE)
            MONMSG      MSGID(CPF9801) EXEC(GOTO CMDLBL(#CRTTBLA))
                GOTO        #END
    /* テーブル作成 */
#CRTTBLA:
            CRTPF       FILE(QTEMP/D4Q010) SRCFILE(&SRCLIB/QDDSSRC) +
                          OPTION(*NOSRC *NOLIST) SIZE(*NOMAX) +
                          LVLCHK(*NO)
    /* プログラムメッセージ・モニター */
            MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(#PGMERR))
    /* 正常終了 */
#END:
                CLRPFM      FILE(QTEMP/D4Q010)
                MONMSG      MSGID(CPF0000)
            RETURN
    /* エラー終了 */
#PGMERR:    ENDPGM
```

ソース5　RGB

```
********** データの始め **********
     H*------------------------------------------------------------*
     H*  PROGRAM 名     :   D4Q010
     H*  PROGRAM 見出   :   受注一覧照会ワーク作成
     H*          作成日 :
     H*          更新日 :
     H*------------------------------------------------------------*
     H           Y/                                    1
     F*------------------------------------------------------------*
     F*                    ＦＩＬＥ
     F*------------------------------------------------------------*
     F*<< 受注ＤＢ（得意先指定） >>
     FDTTHJL01IF  E           K        DISK
     F*<< 受注ＤＢ（受注日指定） >>
     FDTTHJL02IF  E           K        DISK
     F           HJR00                              KRENAMEHJR02
     F*<< 取引先Ｍ >>
     FDTMMCP  IF  E           K        DISK
     F*<< 社員Ｍ   >>
     FDTMMBP  IF  E           K        DISK
     F*<< メッセージＭ >>
     FDTMMJP  IF  E           K        DISK
     F* 受注一覧照会ワーク
     FD4Q010  O   E           K        DISK
     E*------------------------------------------------------------*
     E*                    配列
     E*------------------------------------------------------------*
     I*------------------------------------------------------------*
     I*                    ＤＳ
     I*------------------------------------------------------------*
     I* プログラムＩＤ
     I           SDS
     I                                      244 253 WSID
     I                                      254 263 WSUSR
     I                                     *PROGRAM D#PGID
     C*------------------------------------------------------------*
     C*                    ＰＬＩＳＴ
     C*------------------------------------------------------------*
     C           *ENTRY    PLIST
     C                     PARM           PPJUDF  80      * 受注日 FROM
     C                     PARM           PPJUDT  80      * 受注日 TO
     C                     PARM           PPTKCD  80      * 得意先コード
     C                     PARM           PPERR   1       * エラー
     C                     PARM           PPMSG  62       * メッセージ
     C*------------------------------------------------------------*
     C*                    ＫＬＩＳＴ
     C*------------------------------------------------------------*
     C* 受注ＤＢ
     C           KEYHJ1    KLIST
     C                     KFLD           HJTKCD          得意先コード
     C                     KFLD           HJJUDT          受注日
     C           KEYHJ2    KLIST
     C                     KFLD           HJTKCD          得意先コード
     C* 受注ＤＢ（受注日）
     C           KEYHJ3    KLIST
     C                     KFLD           HJJUDT          受注日
     C* 取引先Ｍ
     C           KEYMC1    KLIST
     C                     KFLD           MCTRCD
     C* 社員Ｍ
     C           KEYMB1    KLIST
     C                     KFLD           MBTACD          社員コード
     C*
     C*------------------------------------------------------------*
     C*                    ＭＡＩＮ　ＲＯＵＴＩＮＥ
     C*------------------------------------------------------------*
     C*<< 初期設定ルーチンにとぶ。 >>
     C                     EXSR SBINZ
     C*
     C                     EXSR SB0000
     C*
     C                     EXSR SBEND
     C*------------------------------------------------------------*
     C*          SBINZ     初期設定サブルーチン
     C*------------------------------------------------------------*
     C           SBINZ     BEGSR
```

```
C*============================================================*
C*          SBINZ     初期設定サブルーチン
C*============================================================*
C           SBINZ     BEGSR
C*<< 受注 DB>>
C           *LIKE     DEFN HJJUDT    WKJUDT           * 受注日
C           *LIKE     DEFN HJTKCD    WKTKCD           * 得意先
C                     MOVEL*BLANK    WKFST   1        * エラー
C                     MOVEL'0'       WKOK    1        * 処理 OK
C                     MOVEL*BLANK    PPMSG            * メッセージ
C*<<KEY 判断 >>
C                     MOVEL'0'       *IN70
C           PPTKCD    IFNE 0
C                     MOVEL'1'       *IN70
C                     ENDIF
C                     ENDSR
C*============================================================*
C*          SB0000    メイン処理
C*============================================================*
C           SB0000    BEGSR
C                     Z-ADDPPJUDF    HJJUDT             受注日
C*<< 得意先処理 >>
C           *IN70     IFEQ '1'
C                     Z-ADDPPTKCD    HJTKCD             得意先コード
C           KEYHJ1    SETLLHJR00
C                     ELSE
C*<< 受注日処理 >>
C           KEYHJ3    SETLLHJR02
C                     ENDIF
C                     MOVEL'0'       *IN90
C*<< メインＲＥＡＤ >>
C           *IN90     DOWEQ'0'
C           *IN70     IFEQ '1'
C           KEYHJ2    READEHJR00                  90
C                     ELSE
C                     READ HJR02                  90
C                     ENDIF
C*<< ファイルＥＯＦ >>
C           *IN90     IFEQ '1'
C                     LEAVE
C                     ENDIF
C*<< 受注日範囲を超えた場合終了 >>
C           HJJUDT    IFGT PPJUDT
C                     LEAVE
C                     ENDIF
C*<< ワークファイル更新 >>
C                     EXSR SB1000
C                     ENDDO
C*<< エラー処理 >>
C           WKFST     IFEQ *BLANK
C                     MOVEL'1'       PPERR
C                     MOVEL'E0010 '  MJMSGC
C                     EXSR SBMSG
C                     ENDIF
C                     ENDSR
C*============================================================*
C*          SB1000    ワークファイル更新
C*============================================================*
C           SB1000    BEGSR
C*
C                     MOVEL'1'       WKFST
C*
C                     CLEARHWR00
C                     Z-ADDHJJGCD    HWJGCD             部課 CD
C                     Z-ADDHJJUNO    HWJUNO             受注No.
C                     Z-ADDHJJUDT    HWJUDT             受注日
C                     Z-ADDHJTKCD    HWTKCD             得意先 CD
C                     Z-ADDHJTKCD    MCTRCD             得意先 CD
C           KEYMC1    CHAINMCR00                 91
C  N91                MOVELMCTRRK    HWTRRK             取引先略称
C                     Z-ADDHJTACD    HWTACD             営業担当者
C*<< 営業担当者名取得 >>
C                     Z-ADDHJTACD    MBTACD             営業担当者
C           KEYMB1    CHAINMBR00                 91
C  N91                MOVELMBTANM    HWTANM             社員名称
C                     Z-ADDHJNOKI    HWNOKI             納期日
C                     Z-ADDHJJUGK    HWJUGK             受注合計額
C                     MOVELHJBIKO    HWBIKO             備考
C*
C                     WRITEHWR00
C                     ENDSR
C*============================================================*
C*  SBMSG    メ ッ セ ー ジ 取 得                *
C*============================================================*
C           SBMSG     BEGSR
C*
C                     MOVEL*BLANK    PPMSG
C           MJMSGC    CHAINMJR00                 92
C  N92                MOVELMJMSGM    PPMSG
C*
C                     ENDSR
C*============================================================*
C*          SBEND     終了処理
C*============================================================*
C           SBEND     BEGSR
C*<< 終了処理を行なう。 >>
C                     SETON                      LR
C                     RETRN
C*
C                     ENDSR
********** データの終わり **********************************
```

ソース6　照会画面

```
{#############################################################################

 照会画面

#############################################################################}
unit ReferenceFrm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Scdcall, Grids, DBGrids, StdCtrls, Mask, Buttons, ExtCtrls, DB,
  DBTables;

type
  TfrmReference = class(TForm)
    Panel1: TPanel;
    Label2: TLabel;
    Label3: TLabel;
    Label1: TLabel;
    btnSelect: TBitBtn;
    meJUDT_F: TMaskEdit;
    meJUDT_T: TMaskEdit;
    edTRCD: TEdit;
    dbgList: TDBGrid;
    pnlBottom: TPanel;
    bbtnClose: TBitBtn;
    cal4D4Q010C: TCall400;
    cal4D4Q011C: TCall400;
    tbID4Q010: TTable;
    tbID4Q010HWJGCD: TIntegerField;
    tbID4Q010HWJUNO: TIntegerField;
    tbID4Q010HWJUDT: TIntegerField;
    tbID4Q010HWTRRK: TStringField;
    tbID4Q010HWTANM: TStringField;
    tbID4Q010HWNOKI: TIntegerField;
    tbID4Q010HWJUGK: TIntegerField;
    tbID4Q010HWTKCD: TIntegerField;
    tbID4Q010HWTACD: TIntegerField;
    tbID4Q010HWBIKO: TStringField;
    dsD4Q010: TDataSource;
    btnClear: TButton;
    procedure FormShow(Sender: TObject);
    procedure bbtnCloseClick(Sender: TObject);
    procedure btnClearClick(Sender: TObject);
    procedure btnSelectClick(Sender: TObject);
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  end;

var
  frmReference: TfrmReference;

implementation

uses MenuFrm;

{$R *.dfm}

{*****************************************************************************
 目的：画面表示時
 引数：
 戻値：
*****************************************************************************}
procedure TfrmReference.FormShow(Sender: TObject);
begin
  inherited;
  // QTEMPへワークファイルを作成
  cal4D4Q011C.Execute;

  // 初期フォーカスを受注日
  meJUDT_F.SetFocus;
end;

```

【図X】①
OnShowイベントで初期設定

```
{*****************************************************************************
 目的：検索処理
 引数：
 戻値：
*****************************************************************************}
procedure TfrmReference.btnSelectClick(Sender: TObject);
begin
  inherited;
  //----- 検索前チェック --------------------
  // 受注日(From)
  if meJUDT_F.Text = '' then
  begin
    meJUDT_F.SetFocus;
    raise Exception.Create('受注日(From)を指定して下さい。');
  end;

  // 受注日(To)
  if meJUDT_T.Text = '' then
  begin
    meJUDT_T.SetFocus;
    raise Exception.Create('受注日(To)を指定して下さい。');
  end;

  // 受注日範囲指定大小チェック
  if StrToIntDef(meJUDT_F.Text, 0) > StrToIntDef(meJUDT_T.Text, 0) then
  begin
    meJUDT_F.SetFocus;
    raise Exception.Create('受注日の範囲指定が不正です。');
  end;

  //----- 検索処理 --------------------
  // ワークファイルを閉じる
  if dsD4Q010.DataSet.Active then dsD4Q010.DataSet.Active := False;

  // データ抽出CLを実行
  with cal4D4Q010C do
  begin
    LibraryName := '*LIBL';      // ライブラリー
    Value[0] := meJUDT_F.Text;   // 受注日(From)
    Value[1] := meJUDT_T.Text;   // 受注日(To)
    Value[2] := edTRCD.Text;     // 得意先CD
    Value[3] := '';              // エラー
    Value[4] := '';              // メッセージ

    Execute;

    // エラー処理
    if Value[3] <> '' then
    begin
      meJUDT_F.SetFocus;
      raise Exception.Create(Value[4]);
    end;
  end;

  //----- 明細表示処理 --------------------
  // ワークファイルを開く
  dsD4Q010.DataSet.Active := True;

  // TDBGridへフォーカス移動
  dbgList.SetFocus;
end;

{*****************************************************************************
 目的：クリア処理
 引数：
 戻値：
*****************************************************************************}
procedure TfrmReference.btnClearClick(Sender: TObject);
begin
  inherited;
  // ワークファイルを閉じる
  if dsD4Q010.DataSet.Active then dsD4Q010.DataSet.Active := False;

  // 抽出条件クリア
  meJUDT_F.Clear; // 受注日(From)
  meJUDT_T.Clear; // 受注日(To)
  edTRCD.Clear;   // 得意先CD

  // 受注日(From)へフォーカス移動
  meJUDT_F.SetFocus;
end;

{*****************************************************************************
 目的：画面終了時
 引数：
 戻値：
*****************************************************************************}
procedure TfrmReference.bbtnCloseClick(Sender: TObject);
begin
  // ワークファイルを閉じる
  if tblD4Q010.Active then tblD4Q010.Active := False;

  // QTEMPへワークファイルをクリア
  cal4D4Q011C.Execute;

  // 画面を閉じる
  Self.Close;
end;

end.
```

【図X】②
入力値のチェック

【図X】②
CLプログラムの呼び出し

【図X】⑥
画面に表示する

照会画面
「条件クリア」ボタン押下時

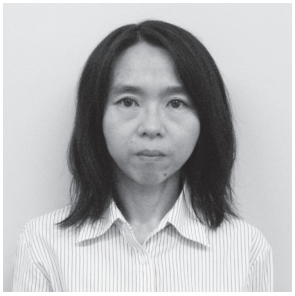
中嶋 祥子

株式会社ミガロ

RAD事業部 技術支援課

Delphi/400とExcelとの連携

本稿は
Delphi/400 アプリケーションと Excel をさらに使いこなすための
リファレンスである。
OLE オートメーションを使った Excel の基本的な出力方法から
Excel に対する各種テクニックをまとめた。



略歴
1968 年 2 月 23 日 生
1990 年奈良女子大学大学家政学卒
2002 年株式会社ミガロ入社
2002 年 11 月 RAD 事業部配属

現在の仕事内容
お客様からの Delphi/400 に関する技術的なご質問やお問い合わせに携わっている。また、メールマガジン「Migaro News」やホームページで、Tips など、開発に役立つ情報を担当していることもある。

1 Delphi/400アプリケーションとExcel

Delphi/400 でアプリケーションを作る場合、Excel の出力機能を利用しようとすると、帳票機能と同じぐらい組み込みを必要とされることが多い。

Excel を扱ううえで、専用のツール（例えばVB-Report3.0 など）を使わないケースでは、Office コンポーネントまたは OLE オートメーションを利用することになる。

Delphi/400 からの Excel の操作は、プログラムソースを見ると難しく感じるかもしれない。だが、よく使う Excel 操作は決まっているので、一度習得してしまえば非常に簡単に扱うことができる。

ここでは、Delphi/400 アプリケーションからの、OLE オートメーションによる Excel の基本的な出力方法や、Excel に対する各種テクニックを紹介しよう。

2 基本的なExcel出力

OLE オートメーションを使った基本的な Excel の出力は、手順として Excel と Book、Sheet を生成して、Cell に値を書き込んでいくという手順となる。

ソースコードを用意したので、参考にしてほしい。【図 1】

出力する Excel の全てを Delphi 側から操作するのは、プログラム的にもパフォーマンス的にも手間となる。そのため、Excel のテンプレートを用意すると、効率的な Excel 出力が行えるようになる。

具体的には、テンプレートとなる Excel をファイルコピーしたうえで、その Excel の中から必要な Cell 値の編集だけを行う。Cell の指定は、次のように指定する。

Cells［行番号 , 列番号］

テンプレートを利用する場合は、Excel 起動部分を、ソースのように処理することで指定の Excel を扱うことができる。【図 2】

3 Excel操作テクニック

ここから、Delphi/400 上で Excel を操作する際によく使われる動作を取り上げる。具体的なプログラムソースを交えて説明する。

①文字のフォント / フォントサイズを設定

フォントは、Font.Name で設定することができる。

フォントサイズは、Font.Size で設定することができる。【図 3】

②行の高さと列の幅を設定

行の高さは、RowHeight で設定することができる。

列の幅は、ColumnWidth で設定することができる。【図 4】

③オートフィットを設定

オートフィットは、AutoFit を使い、

図1

```
procedure TForm1.Button1Click(Sender: TObject);
var
  MsExcel      : Variant;
  MsApplication: Variant;
  WBook        : Variant;
  WSheet       : Variant;
begin
  //Excel起動
  MsExcel := CreateOleObject('Excel.Application');
  MsApplication := MsExcel.Application;
  MsApplication.Visible := True;
  WBook  := MsApplication.WorkBooks.Add ;
  WSheet :=WBook.ActiveSheet;

  //ExcelのCellに値を書き込む
  WSheet.Cells[1,1].Value := 'データ1';
  WSheet.Cells[1,2].Value := 'データ2';
  WSheet.Cells[1,3].Value := 'データ3';

  //保存の確認を行う
  WBook.Saved  :=  False;
  //Excel終了
  MsApplication.WorkBooks.Close;
  MsExcel.Quit;
end;
```

図2

```
  // テンプレートファイルよりコピーして作業ファイルを作成
  CopyFile(PChar('コピー元ファイル'), PChar('コピー先ファイル'), False);
  //Excel起動
  MsExcel := CreateOleObject('Excel.Application');
  MsApplication := MsExcel.Application;
  MsApplication.Visible := True;
  WBook  := MsApplication.Workbooks.Open('コピー先ファイル');
  WSheet :=WBook.ActiveSheet;
```

図3

```
WSheet.Cells.Font.Name := 'ＭＳ Ｐゴシック';  //フォント名を設定
WSheet.Cells.Font.Size := 12;                 //フォントサイズを設定
```

図4

```
WSheet.Rows[1].RowHeight := 50;              //行の高さを設定
WSheet.Columns[1].ColumnWidth := 50;         //列の幅を設定
```

図5

```
WSheet.Cells.Select;                              //セルを全選択
WSheet.Cells.EntireColumn.AutoFit;                //オートフィットを設定
```

セルの値で幅を設定できる。【図5】

④先頭のシートを選択

ワークシートは、WorkSheets［ワークシート番号］.Selectで選択できる。【図6】

⑤Book名を指定して保存

Bookの保存は、SaveAsで、ファイル名や詳細パラメータを指定することができる。【図7】

⑥範囲を指定

セルの範囲は、Rangeで指定することができる。【図8】

⑦罫線を出力

罫線は、Rangeで範囲を指定し、出力することができる。

Borders.LineStyleで、罫線のスタイルを設定できる。

Borders.Weightで、罫線の太さを設定できる。【図9】

⑧セルを結合

複数セルは、Rangeで指定して、MergeCellsで結合できる。【図10】

⑨シートを印刷

ワークシートは、PrintOutで印刷することができる。【図11】

⑩シートをプレビュー表示

⑪警告メッセージを制御

Excelの警告メッセージは、DisplayAlertで出力制御をすることができる。【図12】

4 Excelのマクロ記録機能を活用

「Excel操作テクニック」では、具体的に、Delphi/400でよく使うプログラム例を挙げてみた。だが、例にないExcelの動作を行いたいときはどうすればいいのだろうか。

Excelでは、マクロで動作を記録して、VBのソースとして調べることができる。VBのソースはもちろんDelphiのソースとは異なる。とはいえ、操作やプロパティは同じように扱えるため、十分にDelphiのプログラムの参考とすることができる。

Excelのマクロ記録

ここでは、前述の⑥の［範囲を指定］を例に見ていこう。

⑴最初に、Excelの［ツール］→［マクロ］から［新しいマクロの記録］を実行する。【図13】【図14】
⑵これで、マクロが記録状態となるので、調べたいExcelの操作を実際に行う。ここでは、A1からC5のセルを選択して、Ctrl + Cでコピーする。
⑶Excelの操作が終わったら、マクロの記録を終了する。【図15】

VBのソースで確認

では、マクロが記録されているので、VBのソースで、この記録されたExcelの操作を確認してみよう。

⑷確認するマクロを選択し、［編集］で開けば、マクロのソースを見ることができる。【図16】

ソースを見てみると、A1からC5のセルの範囲はRangeで指定されて、Selectで選択されていることがわかる。また、コピーは、Copyというメソッドが使用されているのがわかる。【図17】

Delphiのソースに組み替え

このソースを参考に、Delphiのソースに組み替えていく。すると、⑥の［範囲を指定］のようなプログラムを作成することができる。

Excelの操作は、基本的に何でもマクロに記録できる。そのため、参考となるようなソースが見つけられないときには、これらの調査を行って、独自にプログラムを作成することができる。少し応用的な内容ではあるが、Excelのプログラムで詰まった際にはぜひ試してみてほしい。

5 Excel2007への対応

Excel2007のPCで、xlsx形式を指定してファイル保存をしたいことがある。その場合、実行PCのExcelのバージョンがExcel2007かどうかを判断する必要がある。

ExcelApplicationのVersionで、Excelのバージョンを取得/判断することができる。

各Excelのバージョンは、以下の通りである。

・2007 - ‘12.0‘
・2003 - ‘11.0‘
・XP　- ‘10.0‘
・2000 - ‘ 9.0‘
・97　- ‘ 8.0‘または ‘8.0a‘

例えば、バージョンを判断してファイル名（拡張子）を設定する場合のソースを作成してみた。【図18】

応用して組み込めば、これまでの作成済のExcel処理も、バージョンに対応した汎用的な作りに組み替えることができるだろう。

6 まとめ

本稿では、基本的なExcelの操作から、応用的な調査方法まで説明してきた。これらを、DelphiでExcelを扱う際に、リファレンスとして活用してほしい。

具体的なコード例をコピーして使うのもよいし、また中身の仕組みがわかってくれば、自分で新しいExcel操作の調査もできるようになる。

DelphiからのExcelの操作はたくさんあるので、それらのテクニックをマスターして、ぜひ自分用のExcelリファレンスを作り上げてほしい。

M

図6

```
WBook.Worksheets[1].Select(True);            //WorkBookの先頭(1)Sheetを選択
```

図7

```
WBook.SaveAs('保存ファイル名.xls');
```

図8

```
WSheet.Range['A1','C5'].Copy;       //セルのA1からC5の範囲を選択してコピー
WSheet.Range['A6'].Select;          //セルのA6を選択
WSheet.Paste;                       //コピーした内容を貼り付け
```

図9

```
//セルのA1からC5の罫線スタイルを設定
WSheet.Range['A1','C5'].Borders[8].LineStyle := 1;
//セルのA1からC5の罫線の太さを設定
WSheet.Range['A1','C5'].Borders[8].Weight := 2;
```

図10

```
WSheet.Range['A1','C5'].MergeCells := true;    //セルのA1からC5を選択してセル結合
```

図11

```
WSheet.PrintOut ;       //WorkSheetを印刷
```

図12

```
MsApplication.DisplayAlerts := false;    //警告メッセージが出力されないよう設定
```

図13

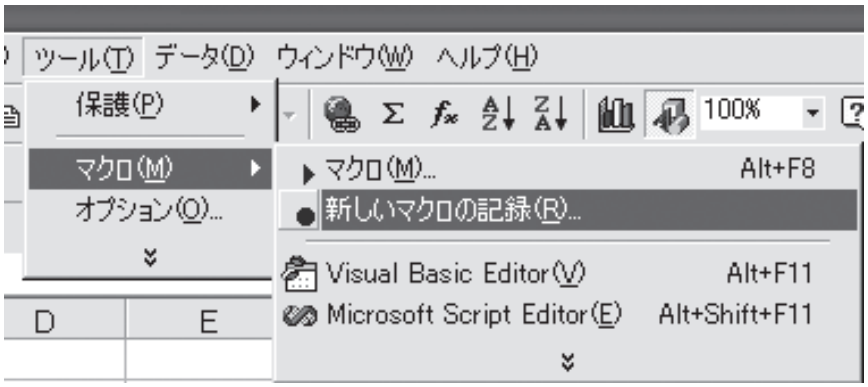


図14

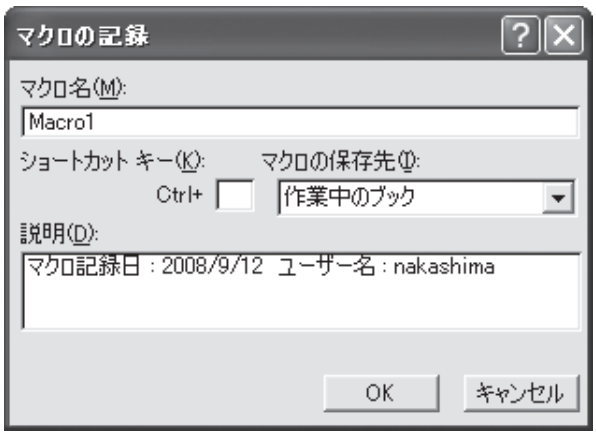


図15

図16

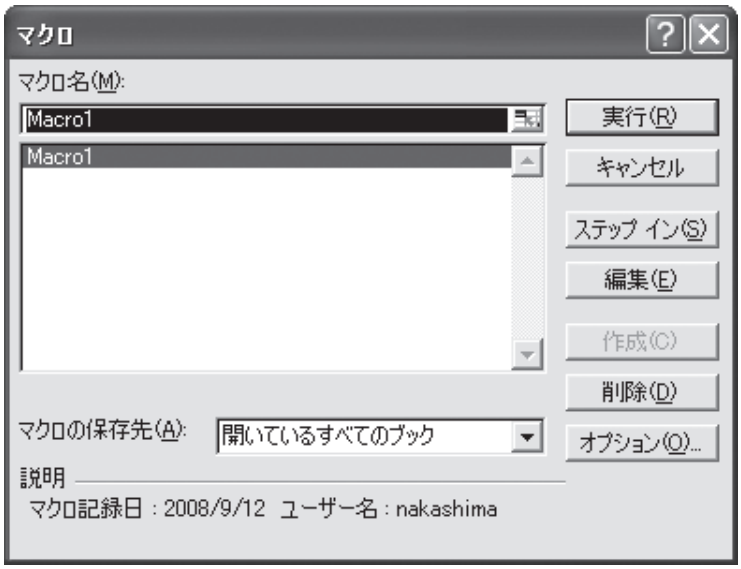


図17

```
' Macro1 Macro
' マクロ記録日 : 2008/9/12  ユーザー名 :nakashima
'

'
    Range("A1:C5").Select
    Selection.Copy
End Sub
```

セルのA1からC5を選択してコピー

図18

```
//バージョンを確認してファイル名(拡張子)を判断
if MsApplication.Version = '12.0' then
  WBook.SaveAs('C:¥Test.xlsx')
else
  WBook.SaveAs('C:¥Test.xls');
```

尾崎 浩司

株式会社ミガロ

システム事業部 システム2課

連携で広がるDelphi/400活用術

**開発の効率を上げるには
既存のいろいろな仕組みと「連携」することが最もよい手段だ。
ここでは、Delphi/400 と他の仕組みとを連携する手法を
具体的な手順を含め紹介する。**

略歴
1973 年 8 月 16 日 生
1996 年三重大学 工学部卒
1999 年 10 月株式会社ミガロ入社
1999 年 10 月システム事業部配属

現在の仕事内容
入社以来、主に Delphi/400 を利用した受託開発を担当している。

はじめに

Delphi/400 は、IBM i を使用するアプリケーションを開発するのに、最適な開発ツールである。と同時に、制約のない Windows アプリケーションを構築するのにも、最適な開発ツールである。

適用業務において可能性は無限に広がるが、1 から全てを構築するとなると工数がいくらあっても足りないであろう。開発の効率を上げるには、すでに用意されているいろいろな仕組みと「連携」を行うことが最適な手段だ。

本稿では、Delphi/400 と他の仕組みとを連携する手法を、具体的な手順を示しながら紹介していこう。これをきっかけに、皆様の開発のバリエーションが広がれば幸いである。

COMを使ったアプリケーション連携

「連携」というとまず思いつくのが、業務で最も使用する Excel との連携であろう。データベースから取得したデータを Excel に出力する、ソース 1 のようなアプリケーションを、私もたびたび開発している。【ソース 1】

COM

このような連携を実現するのが、「COM」と呼ばれる技術である。COM はコンポーネント・オブジェクト・モデルの略で、アプリケーションの持つ機能をオブジェクト化し、別のアプリケーションから容易に利用できるようにする技術である。

この COM という技術により、Delphi/400 からいろいろなアプリケーションの機能を使用することが可能になる。さらに、他のアプリケーションに機能を提供するプログラムを構築することも可能である。

例えば、IBM i から情報を取得するロジックを、Delphi/400 で開発し、COM とすることで、他のツールからの利用が可能になる。そうすることでエンドユーザーは、Excel や Access の VBA 等で、間接的に IBM i のデータを使用することも可能になるというわけである。

COMサーバーの開発

それでは、COM として情報を提供するサーバーはどのようにして開発するのか、見ていこう。

まず、新規プロジェクトを作成する。そして、Delphi のメニューから［ファイル］→［新規作成］→［その他］を選択し、選択カテゴリ「ActiveX」から「オートメーションオブジェクト」を選択する。【図 1a】

すると、ウィザードが始まるので、「CoClass 名」欄にクライアントがアクセスするためのクラス名を入力する。ここでは"AS400Infomation"としよう。【図 1b】

ウィザードが終了すると「タイプライブラリ」と呼ばれるインターフェースを定義する画面が表示される。【図 1c】

この画面で、外部に提供するプロパティあるいはメソッドを定義するわけだ。

"IAS400Infomation" をクリックする

図1a

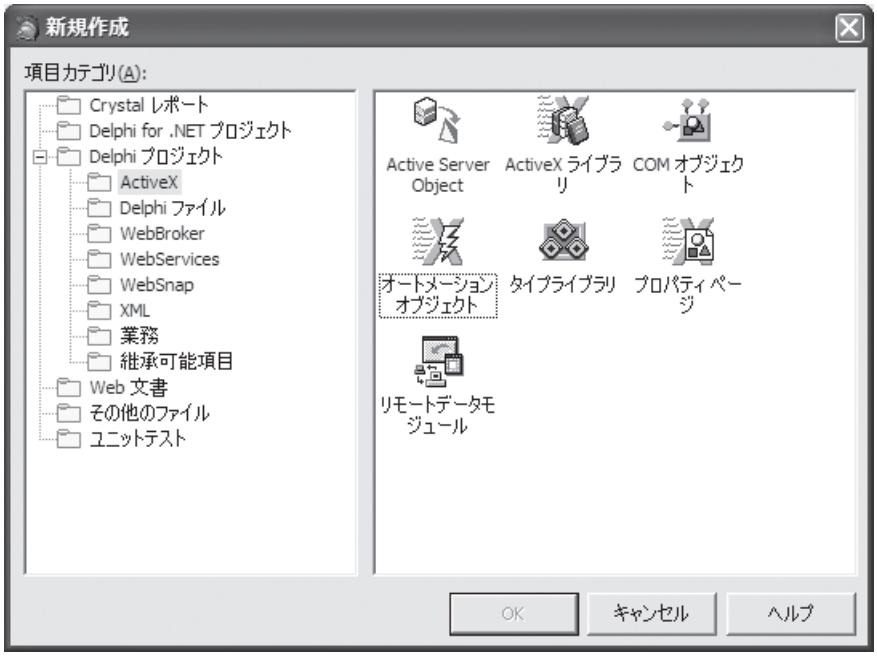

新規作成
項目カテゴリ(A):
Crystal レポート
Delphi for .NET プロジェクト
Delphi プロジェクト
ActiveX
Delphi ファイル
WebBroker
WebServices
WebSnap
XML
業務
継承可能項目
Web 文書
その他のファイル
ユニットテスト
Active Server Object
ActiveX ライブラリ
COM オブジェクト
オートメーション オブジェクト
タイプライブラリ
プロパティページ
リモートデータモジュール
OK
キャンセル
ヘルプ

図1b

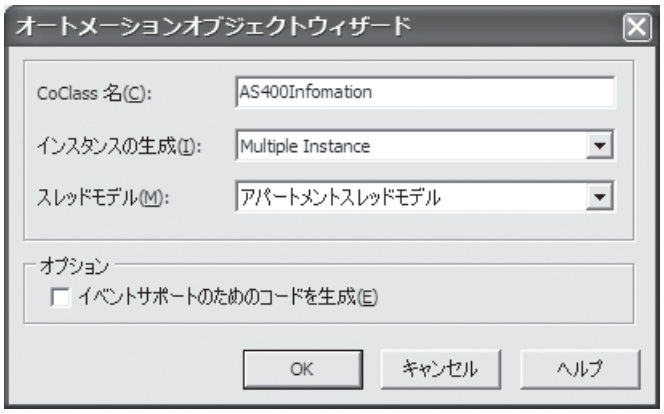
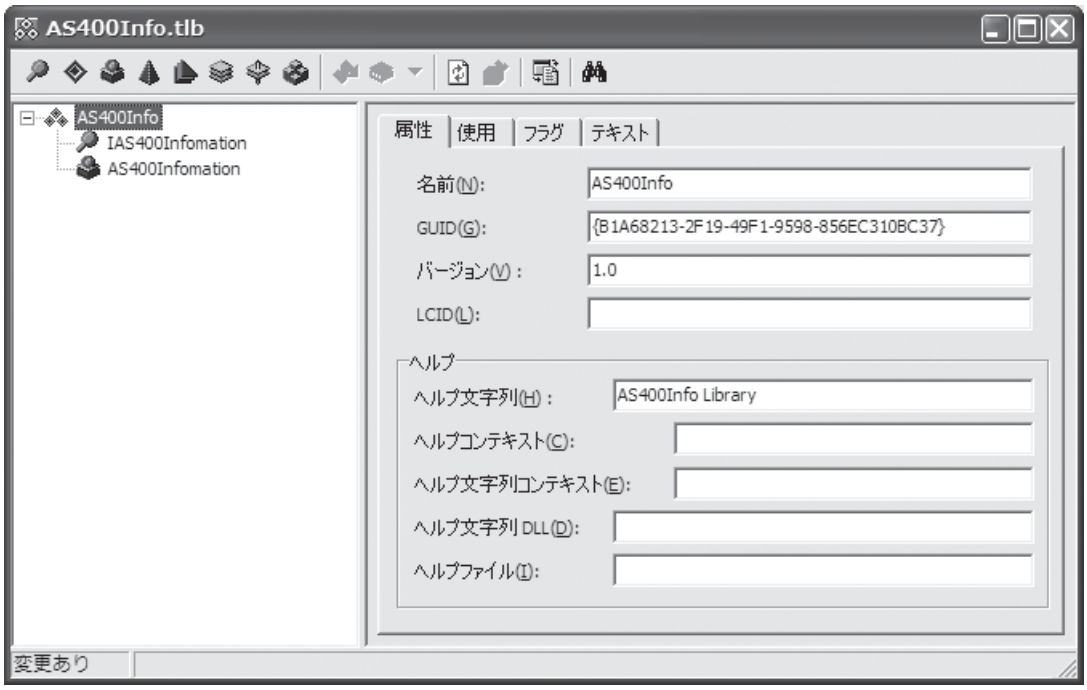

オートメーションオブジェクトウィザード
CoClass 名(C):
AS400Infomation
インスタンスの生成(I):
Multiple Instance
スレッドモデル(M):
アパートメントスレッドモデル
オプション
イベントサポートのためのコードを生成(E)
OK
キャンセル
ヘルプ

図1c


AS400Info.tlb
AS400Info
IAS400Infomation
AS400Infomation
属性
使用
フラグ
テキスト
名前(N):
AS400Info
GUID(G):
{B1A68213-2F19-49F1-9598-856EC310BC37}
バージョン(V):
1.0
LCID(L):
ヘルプ
ヘルプ文字列(H):
AS400Info Library
ヘルプコンテキスト(C):
ヘルプ文字列コンテキスト(E):
ヘルプ文字列 DLL(D):
ヘルプファイル(I):
変更あり

と、メソッドおよびプロパティの新規作成が選択可能になるので、必要に応じたインターフェース定義を行う。メソッドであれば名前を定義し、プロパティであれば名前と属性（タイプ）を定義する。

ここでは、メソッドとして"GetData"を定義し、［読み込み|書き込み］プロパティとして"CustNo"（タイプは数字属性［Long］）と［読み込み専用］プロパティとして"CustName"（タイプは文字属性［BSTR］）を定義する。【図1d】。

インターフェースの定義が終了したら、「ソースコード更新」というボタンを押してみよう。するとソース2のようなソースコードが自動作成される。あとは、そこに仕様に応じたユーザーコードを追加すればよい。【ソース2】

COMサーバーのサンプル

今回は、"CustNo"プロパティに得意先コードを指定し、GetDataメソッドを呼び出すことにより、データベースから得意先マスターを検索し、取得した得意先名を結果としてCustNameプロパティにセットするCOMサーバーを作成してみた。【ソース3】

このサンプルでは、新規プロジェクト作成時にあらかじめ用意されているフォーム（ここではfrmMainと命名）に、DataBaseコンポーネントとQueryコンポーネントを貼り付け、データベース接続定義ならびにデータ抽出用のSQLの定義を行っている。

なお、今回作成しているCOMオブジェクトは、データベースへのアクセス機能のみを持ったものとなる。そのため、画面（フォーム）は表示不要である。

このようなCOMオブジェクトを作成する場合、プロジェクトファイルのソースコードを表示し、"Application.ShowMainForm := False;"の1行を追加して完成となる。【ソース4】

COM登録

完成したプログラムを他のアプリケーションより利用可能にするためには、レジストリ登録が必要になる。

Delphi/400開発環境下でレジストリ登録を行う場合は、Delphiのメニューから［実行］→［実行時引数］を選択し、「パラメータ」欄に「/regserver」と入力し、プログラムを実行する。（反対にレジストリを解除する場合、「/unregserver」を指定して実行する）。【図2】

プログラムを実行するとどうなるか。実行後、画面は何も表示されず、そのままプログラムが終了するはずだ。ここでは、レジストリへの登録が行われるだけだからである。つまり、これでCOM登録が完了となり、他のアプリケーションから利用可能になる。

実際にこの作成したプログラムが動作するのは、COMを利用するクライアントがオブジェクトを生成したときとなる。

COMを使用するクライアント

では、COMを使用するクライアントはどうなるか。今回は、Excelから利用してみよう。

Excelを起動し、シートの中に得意先コードのセルと得意先名のセル、それから検索用のボタンを用意する。【図3】

そして、ボタンのクリックに対するイベント処理として、次のようなVBAコードを入力する。【ソース5】

完成したExcelを「マクロを有効」にして実行する。画面上で得意先コードを入力した後、検索ボタンを押すことで、データベースから検索された得意先名がセットされるだろう。【図4】

ここで、Excel上で作成したソース5のソースをよく見てみよう。言語の違いはあるが、ソース1で、Delphiから、Excelオブジェクトを生成して使用したのと似ていないか。同じように、Excelから、今回作成したCOMのオブジェクトを作成し、プロパティおよびメソッドを使用しているのがわかる。

今回は、単純なマスター検索を行う連携ではあるが、このCOMオブジェクトを使用するユーザーは、IBM iのデータ構造を意識せずにデータにアクセスできる。

つまり、IBM i上のデータをそのまま公開すると機密上よくない場合にも、このような仕組みを検討することにより、必要に応じた項目のみを公開できる。ユーザーが、IBM iのデータを、自由に安全に利用できるのではないだろうか。

それ以外にもいろいろな連携が実現できると思うので、ぜひ皆様も新たな連携にチャレンジしてほしい。

Webで提供される情報との連携

インターネットには多彩な情報が公開されている。ふだん皆様もいろいろ利用されているだろう。これらの情報とシステムとが連携すれば、便利になるのではと考えたことはないだろうか？

例えば、システムに登録されている得意先マスターの住所情報から、地図のWebサイトが表示できたり、出張精算画面で入力した駅名から路線検索ができたりするとたいへん便利と思われる。

Webサービス

世の中には一般に「Webサービス」と呼ばれる仕組みがあり、SOAPと呼ばれるプロトコルを使用すると完全な連携が可能だ。また、Delphi/400からWebサービスを使用する仕組みも用意されている。

しかし、これらを使いこなすには、SOAPやXML等の知識が必要になってくるため、少し敷居が高いのも事実である。ゆえにWebの連携は難しいと考えていないだろうか？　実は、そこまでのことをしなくても、ちょっとした連携であれば容易に実現可能である。

郵便番号検索Webアプリケーション

インターネットエクスプローラで、アドレス欄に以下のように入力してほしい。

http://search.post.japanpost.jp/cgi-zip/zipcode.php?zip=5560017

これは、日本郵便の郵便番号検索Webアプリケーションである。このURLから、郵便番号556-0017地域の所在地がわかる。【図5】

このように、一般的なWebアプリケーションは、Webブラウザから Webアプリケーションに問い合わせを行うことで動作する。

このときの問い合わせ方法には、GETメソッドとPOSTメソッドという2種類が存在するのだが、その中で

図1d

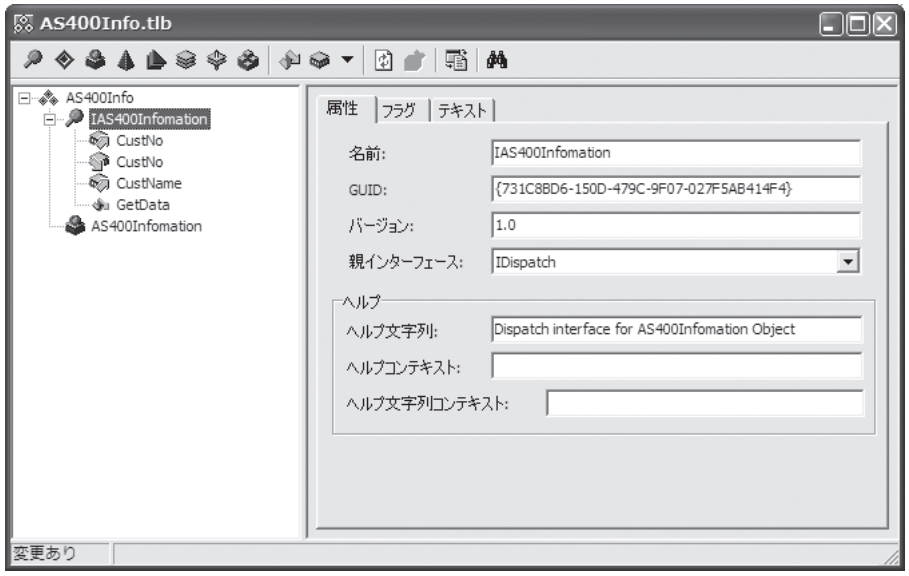

AS400Info.tlb
AS400Info
IAS400Infomation
CustNo
CustNo
CustName
GetData
AS400Infomation
属性
フラグ
テキスト
名前:
IAS400Infomation
GUID:
{731C8BD6-150D-479C-9F07-027F5AB414F4}
バージョン:
1.0
親インターフェース:
IDispatch
ヘルプ
ヘルプ文字列:
Dispatch interface for AS400Infomation Object
ヘルプコンテキスト:
ヘルプ文字列コンテキスト:
変更あり

図2

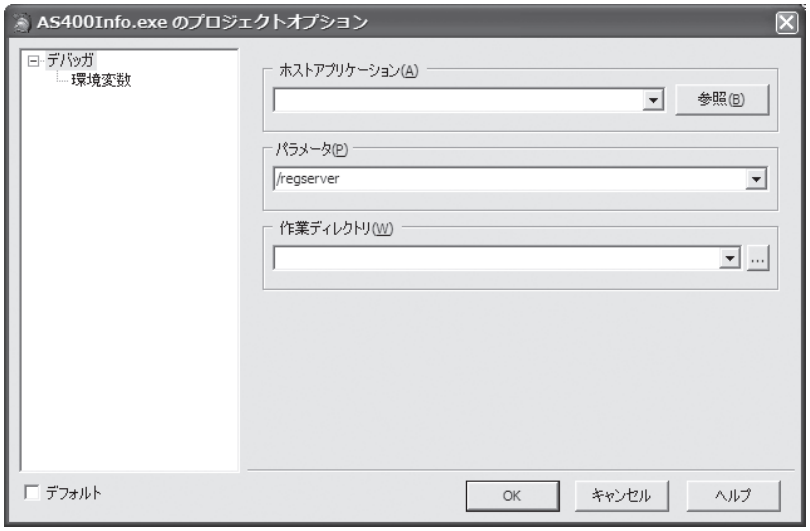
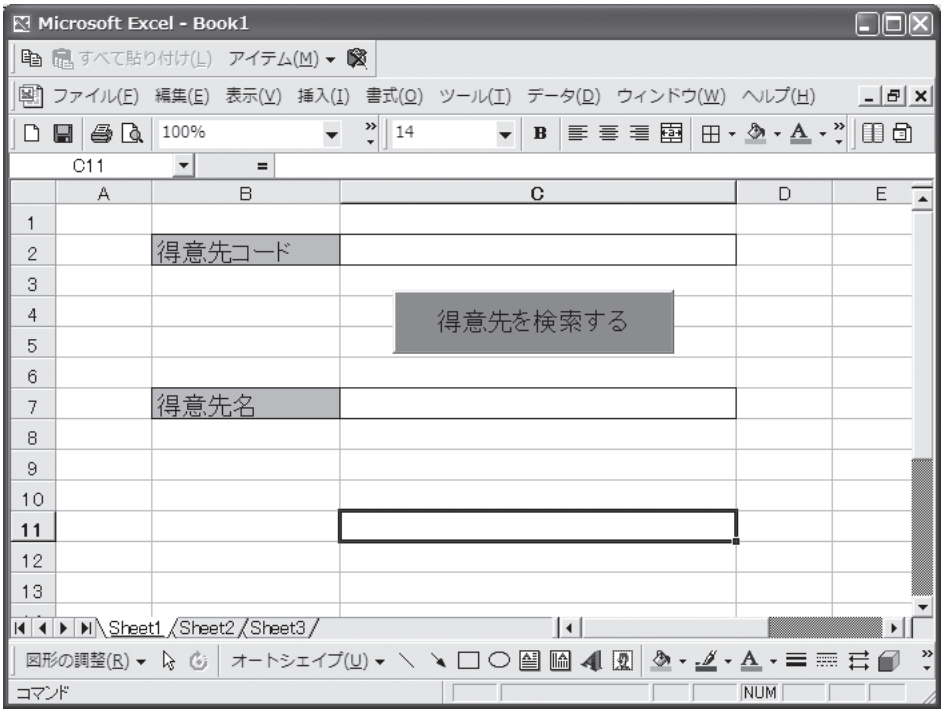

AS400Info.exe のプロジェクトオプション
デバッガ
環境変数
ホストアプリケーション(A)
参照(B)
パラメータ(P)
/regserver
作業ディレクトリ(W)
デフォルト
OK
キャンセル
ヘルプ

図3


Microsoft Excel - Book1
すべて貼り付け(L)
アイテム(M)
ファイル(F)
編集(E)
表示(V)
挿入(I)
書式(O)
ツール(T)
データ(D)
ウィンドウ(W)
ヘルプ(H)
100%
14
C11
得意先コード
得意先を検索する
得意先名
Sheet1
Sheet2
Sheet3
図形の調整(R)
オートシェイプ(U)
コマンド
NUM

GET メソッドは、URL の中に問い合わせを記載するのが特徴だ。先ほどの郵便番号検索の場合、"?zip=5560017" という部分が問い合わせクエリーとなる。

つまり、GET メソッドで問い合わせすることが可能な Web アプリケーションは、変数を含む URL を渡すだけで連携が可能になるのである。

連携プログラムの作成

それでは、簡単な連携プログラムを作成してみよう。

まず、新規プロジェクトを作成し、Button、Edit そして WebBrowser コンポーネントを貼り付ける。【図 6】

そして、ボタンコンポーネントの Click イベントに、処理を記述する。【ソース 6】

では、完成したプログラムを実行してみよう。先ほどのブラウザ画面からの表示結果と同じ内容が、Delphi のフォーム上に表示されることが確認できる。【図 7】

このように Web 情報への連携はとても簡単である。

Google検索

もう 1 つ見てみよう。先ほどと同じように、インターネットエクスプローラ上に下記アドレスを入力してほしい。

http://www.google.co.jp/search?num=30&q=%E3%83%9F%E3%82%AC%E3%83%AD

これは皆様おなじみの Google 検索である。検索キーワード " ミガロ " で、結果が 30 件表示されている。【図 8】

このサイトも先ほどと同じように、Get メソッドで問い合わせを行っている。"q=%E3%83…" のところを見てほしい。実は、この部分は「ミガロ」というキーワードで検索しなさいという問い合わせを表しているのだが、符号化されているのがわかるであろう。

つまり、Get メソッドで問い合わせする際には、通常 2 バイト文字等は利用できないのである。加えて、このように空白文字や特殊記号、日本語等の全角文字を符号化するルールを「URL エンコード」と呼んでいる。

では、Delphi/400 から使用する際に、URL エンコードはどうすればよいか？実は HTTPApp というユニットを uses 節に追加すると、HTTPEncode 関数が使用できるようになり、これを使うと容易に URL エンコードが可能である。

なお、URL エンコードの際には、対象の Web サイトが使用する文字コード体系によりさらに変換が必要な場合もある。先ほどの Google 検索サイトでは、UTF-8 という文字コード体系を使用しているので、このような場合、さらに AnsiToUtf8 関数を使うとよい。

Google検索を実現した Delphi/400連携プログラム

先ほど作成した日本郵便の検索プログラムを改良してみよう。ソース 7 は、Google 検索を実現した Delphi/400 連携プログラムとなる。【ソース 7】

完成したプログラムを実行すると、Delphi の画面で指定したキーワードをもとに Google 検索を行い、結果が画面に表示されていることがわかる。【図 9】

さいごに

このように、単純に Web サイトに対して問い合わせし、結果を画面に表示するだけの「連携」であった。とはいえ、皆様が開発するアプリケーションにおいて、入力した値がそのままパラメータとして利用できるようになれば、いろいろな呼び出しが可能になるだろう。

今回紹介した以外にも、Get メソッドを使用した検索可能なサイトが多数存在する。

例えば、http://ready.to/search/list/ というサイトでは、ブラウザから直接呼び出せるサイトが紹介されている。参考にしてはいかがだろうか。

アイデアしだいでは便利な連携が可能と思われるので、ぜひともいろいろチャレンジしていただきたい。

M

図4

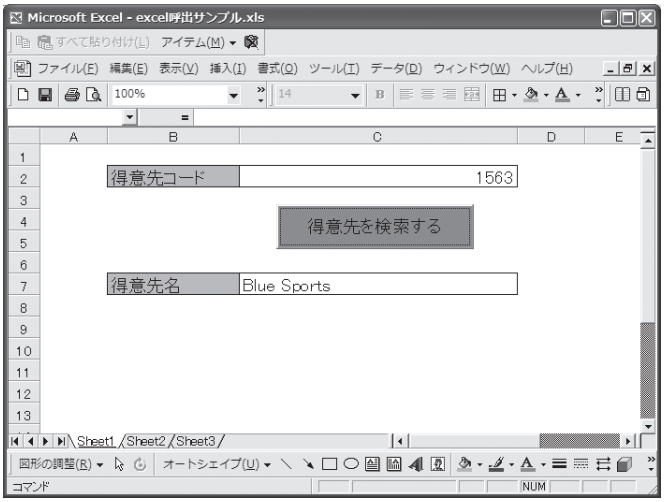

Microsoft Excel - excel呼出サンプル.xls
得意先コード
1563
得意先を検索する
得意先名
Blue Sports
Sheet1
Sheet2
Sheet3

図5


郵便番号 5560017 の検索結果 - 日本郵便 - Windows Internet Explorer
日本郵便
あたらしい ふつうをつくる。
郵便番号から検索
郵便番号 5560017 の検索結果
大阪府
マ行
556-0017
大阪府
大阪市浪速区
湊町
ミナトマチ

図6

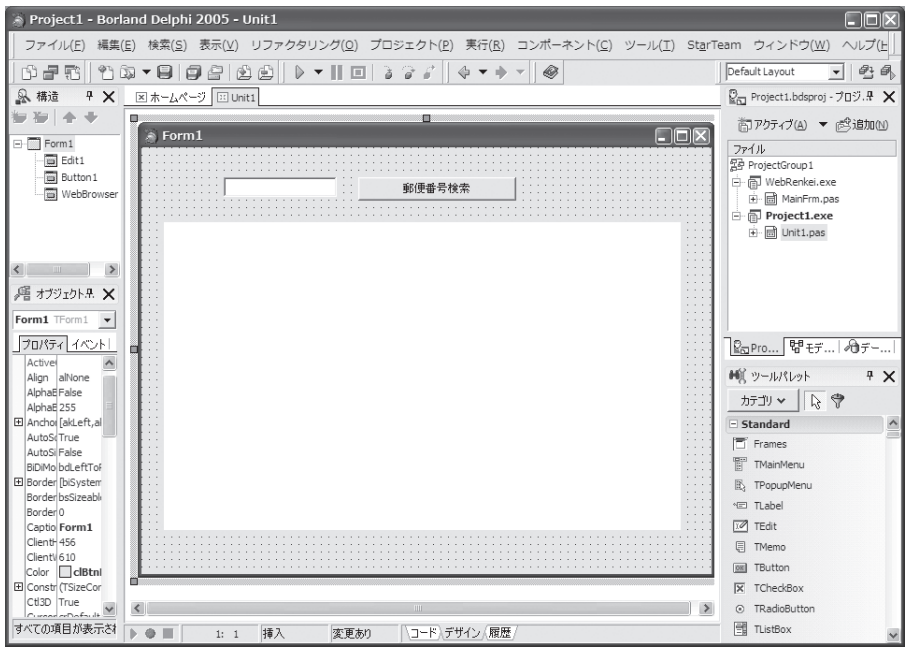

Project1 - Borland Delphi 2005 - Unit1
Form1
郵便番号検索

図7

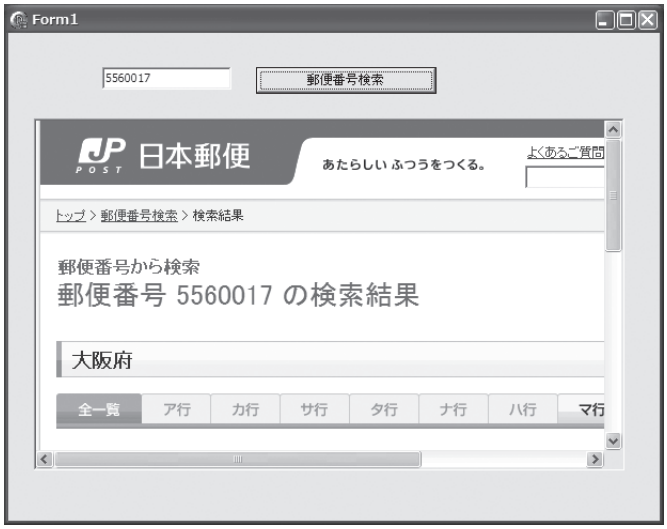
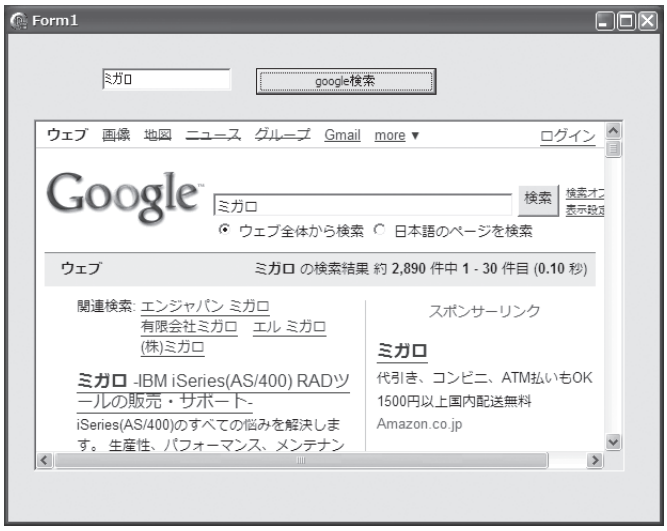

Form1
5560017
郵便番号検索
日本郵便
あたらしい ふつうをつくる。
よくあるご質問
トップ > 郵便番号検索 > 検索結果
郵便番号から検索
郵便番号 5560017 の検索結果
大阪府
全一覧
ア行
カ行
サ行
タ行
ナ行
ハ行
マ行

図8


ミガロ - Google 検索 - Windows Internet Explorer
http://www.google.co.jp/search?num=30&q=%E3%83%9F%E3%82%AC%E3%83%AD
Live Search
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
ミガロ - Google 検索
ページ(P) ツール(O)
ウェブ 画像 地図 ニュース グループ Gmail more ▼
ログイン
Google
ミガロ
検索
検索オプション
表示設定
ウェブ全体から検索 日本語のページを検索
ウェブ
ミガロ の検索結果 約 2,890 件中 1 - 30 件目 (0.20 秒)
関連検索: エンジャパン ミガロ 有限会社ミガロ エル ミガロ (株)ミガロ
ミガロ -IBM iSeries(AS/400) RADツールの販売・サポート-
iSeries(AS/400)のすべての悩みを解決します。 生産性、パフォーマンス、メンテナンス性に優れたDelphi/400、webアプリケーション開発の決定版JACi400でiSeriesの性能が120%up！
www.migaro.co.jp/ - 24k - キャッシュ - 関連ページ
Delphi/400 Version2007 発売開始
MIGARO＞導入事例＞事例一覧
noMAX特長
MKS特長
Tips
TecAward
migaro.co.jp からの検索結果 »
[PDF] 株式会社ミガロ 「第18期を迎えて」 社長 上甲 將隆
ファイルタイプ: PDF/Adobe Acrobat - HTMLバージョン
ミガロには「クレド」がある。会社の理念というよりは、皆で大切にすること、皆で伸ばしていきたい資質が書いてある ... さらに近年では、お客様の要望や問題が多様化しミガロ 単独ではなくパートナーやベンダーの方々とタッグを組んで提案する機会も増え ...
www.migaro.co.jp/contents/company/newspaper/images/MIGZINEVol6.pdf - 関連ページ
スポンサーリンク
ミガロ
代引き、コンビニ、ATM払いもOK
1500円以上国内配送無料
Amazon.co.jp
ページが表示されました
インターネット
100%

図9


Form1
ミガロ
google検索
ウェブ 画像 地図 ニュース グループ Gmail more ▼
ログイン
Google
ミガロ
検索
ウェブ全体から検索 日本語のページを検索
ウェブ
ミガロ の検索結果 約 2,890 件中 1 - 30 件目 (0.10 秒)
関連検索: エンジャパン ミガロ
有限会社ミガロ エル ミガロ
(株)ミガロ
ミガロ -IBM iSeries(AS/400) RADツールの販売・サポート-
iSeries(AS/400)のすべての悩みを解決します。 生産性、パフォーマンス、メンテナン
スポンサーリンク
ミガロ
代引き、コンビニ、ATM払いもOK
1500円以上国内配送無料
Amazon.co.jp

ソース1

```
uses ComObj;

procedure TForm1.Button1Click(Sender: TObject);
var
  MsExcel, MsApplication, WBook, WSheet: OleVariant;
  i: integer;
begin
  //Excel 起動
  MsExcel := CreateOleObject('Excel.Application');
  MsApplication := MsExcel.Application;
  MsApplication.Visible := True;
  WBook  := MsApplication.WorkBooks.Add;
  WSheet :=WBook.ActiveSheet;
  //Excel にタイトル出力
  WSheet.Cells[1,3].Value := ' 得意先マスター一覧表 ';
  WSheet.Cells[1,3].Font.Size := 15;
  WSheet.Cells[2,1].Value := ' 得意先コード ';
  WSheet.Cells[2,2].Value := ' 得意先名 ';
  WSheet.Cells[2,3].Value := ' 住所 ';
  WSheet.Cells[2,1].Font.Bold := 'True';
  WSheet.Cells[2,2].Font.Bold := 'True';
  WSheet.Cells[2,3].Font.Bold := 'True';
  //Excel にデータ出力
  with Table1 do
  begin
    Active := True;
    try
    i := 0;
    First;
    while not eof do
    begin
      WSheet.Cells[i+3,1].Value := FieldByName('CUSTNO').
      AsInteger;
      WSheet.Cells[i+3,2].Value :=
      FieldByName('COMPANY').AsString;
      WSheet.Cells[i+3,3].Value := FieldByName('ADDR1').
      AsString;
      Next;
      i := i + 1;
    end;
    finally
      Active := False;
    end;
  end;
end;
```

ソース2

```
unit Unit1;

{$WARN SYMBOL_PLATFORM OFF}

interface

uses
  ComObj, ActiveX, AS400Info_TLB, StdVcl;

type
  TAS400Infomation = class(TAutoObject,
  IAS400Infomation)
  protected
    function Get_CustName: WideString; safecall;
    function Get_CustNo: Integer; safecall;
    procedure GetData; safecall;
    procedure Set_CustNo(Value: Integer); safecall;

  end;

implementation

uses ComServ;

function TAS400Infomation.Get_CustName: WideString;
begin

end;

function TAS400Infomation.Get_CustNo: Integer;
begin

end;

procedure TAS400Infomation.GetData;
begin

end;

procedure TAS400Infomation.Set_CustNo(Value: Integer);
begin

end;

initialization
  TAutoObjectFactory.Create(ComServer,
  TAS400Infomation, Class_AS400Infomation,
    ciMultiInstance, tmApartment);
end.
```

ソース3

```
----- << ここまで省略 >> ------

type
 TAS400Infomation = class(TAutoObject,
 IAS400Infomation)
 private
  // 内部変数
  FCustNo: Integer;          // 得意先コード
  FCustName: String;          // 得意先名
 protected
  function Get_CustName: WideString; safecall;
  function Get_CustNo: Integer; safecall;
  procedure GetData; safecall;
  procedure Set_CustNo(Value: Integer); safecall;
 end;

implementation

uses ComServ, MainFrm, SysUtils, Dialogs, DBTables;

function TAS400Infomation.Get_CustName: WideString;
begin
 // 得意先名の内部値を渡す
 Result := FCustName;
end;

function TAS400Infomation.Get_CustNo: Integer;
begin
 // 得意先コードの内部値を渡す
 Result := FCustNo;
end;

procedure TAS400Infomation.GetData;
begin
 try
  // 得意先コードが指定されない場合、エラーとする
  if FCustNo = 0 then
   raise Exception.Create(' 得意先コードが指定されていま
   せん。');

  // クエリーを使用し、得意先コードをキーに得意先マス
  ターを検索し、
  // 得意先名を取得する
  with frmMain.Query1 do
  begin
   // クエリーを閉じる
   Active := False;

   // 得意先コードパラメータに得意先コード内部値を代入
   する
   ParamByName('CUSTNO').AsInteger := FCustNo;

   // クエリーを実行する
   Active := True;
   try
    First;

    // データが存在しない場合、エラーとする
    if Eof and Bof then
     raise Exception.Create(' 指定された得意先コードは
     存在しません。');

    // 取得結果を得意先名内部値に代入する
    FCustName := FieldByName('CUSTNM').AsString;
   finally
    // クエリーを閉じる
    Active := False;
   end;
  end;
 except
  // エラーメッセージの表示
  on E: Exception do
  begin
   MessageDlg(E.Message, mtError, [mbOK], 0);
  end;
 end;
end;

procedure TAS400Infomation.Set_CustNo(Value: Integer);
begin
 // パラメータを内部値にセットし、得意先名を初期化する
 FCustNo := Value;
 FCustName := '';
end;

----- << 以下省略 >> -----
```

ソース4

```
Application.Initialize;
Application.CreateForm(TfrmMain, frmMain);
Application.ShowMainForm := False;  // ( 追加 ) メイン
フォームを表示しない
Application.Run;
```

ソース5

```
Option Explicit


Private Sub CommandButton1_Click()

  Dim objAS400Info As Object   '---- 得意先情報取得オブ
  ジェクト
  Dim lngCustNo As Long        '---- 得意先コード

  ' 得意先情報取得オブジェクトの生成
  Set objAS400Info = CreateObject("AS400Info.
  AS400Infomation")

  ' 取引先コードのセット
  lngCustNo = ActiveSheet.Range("C2").Value
  objAS400Info.CustNo = lngCustNo

  ' データの取得 ( メソッドの実行）
  objAS400Info.GetData

  ' 取得した得意先名のセット
  ActiveSheet.Range("C7").Value = objAS400Info.
  CustName

  ' オブジェクトの解放
  Set objAS400Info = Nothing

End Sub
```

ソース6

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sURLText: String;
begin
  // URL 文の作成
  sURLText := 'http://search.post.japanpost.jp/cgi-zip/
  zipcode.php?zip=';
  sURLText := sURLText + Edit1.Text;

  // 問い合わせを実行し、結果を WebBrowser コンポーネン
  トに表示
  WebBrowser1.Navigate(sURLText);
end;
```

ソース7

```
uses HTTPApp;  // HTTPApp ユニットを追加する

procedure TForm1.Button1Click(Sender: TObject);
var
  sURLText: String;
begin
  // URL 文の作成
  sURLText := 'http://www.google.co.jp/
search?num=30&q=';
  sURLText := sURLText + HTTPEncode(AnsiToUtf8(Edit1.
  Text));

  // 問い合わせを実行し、結果を WebBrowser コンポーネン
  トに表示
  WebBrowser1.Navigate(sURLText);
end;
```

吉原 泰介

株式会社ミガロ

RAD事業部 技術支援課

フォーム継承による効率向上開発手法

フォーム継承は
開発効率やメンテナンス性をよくするためのプログラム側の手法である。
それによって、アプリケーションの動作がよくなるわけではない。
フォーム継承をどのように行えば効率のいい開発を行えるのか、
それを考察する。

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。

- 開発効率、メンテナンス性がよい開発手法
- フォーム継承のポイント
- アプリケーション内の継承構成
- フォーム継承によるメリット

1 開発効率、メンテナンス性がよい開発手法

システム構築から運用までを行う場合、開発者はプロジェクトに費やす工数において、以下の 2 点を考慮する必要がある。

●開発効率
どれだけ工数をかけずに、開発ができるか
●メンテナンス性
どれだけ工数をかけずに、変更や修正が容易にできるか

では、プログラムの開発手法にはどういった形があるだろうか。

1 つの画面の単純なアプリケーションを作成するとき、ほとんどの開発者は 1 つのフォームにプログラムを組み込んでアプリケーションを作成する。これを「開発手法 A」とする。

それに対して、1 度作成した画面を流用して開発できるように、フォーム継承を行ってアプリケーションを作成するやり方を「開発手法 B」とする。

【開発方法】
A：プログラムが全て 1 つのフォームの中に集約されたもの
B：プログラムがフォーム継承を行って作成されたもの

この 2 つの手法でそれぞれ開発をした場合、どういった違いがあるのかを考察してみよう。

【照会プログラムを 1 画面開発した場合】
●開発効率
A：継承を考慮しないで単純に開発できるので、開発工数がかからない。
B：継承を考えながら開発するぶん、A より開発工数がかかる。
●メンテナンス性
A：メンテナンス対象は 1 つのフォームだけなので、変更が容易。
B：継承を考慮しての変更となるため、A よりスキルが必要。

ここで重要なのが、A と B の手法で作成された 2 つのアプリケーションに、動作上の違いはないということである。フォーム継承は、開発効率やメンテナンス性をよくするためのプログラム側の手法であり、それによってアプリケーションの動作がよくなるわけではない。

また、この結果を見ると、フォーム継承を行わないほうが優れているようにように見える。

では、次のケースはどうだろうか。

【同じような照会プログラムを 10 画面開発した場合】
●開発効率
A：単純に 1 画面 × 10 の開発工数がかかる。
B：フォームが継承された部分は再利用できるため、各画面の固有部分だけの開発工数で済む。【図 1】
●メンテナンス性
A：共通の変更が発生すると、10 画面全てに同じ対応を行う必要がある。
B：共通の変更が発生しても、継承元

図1

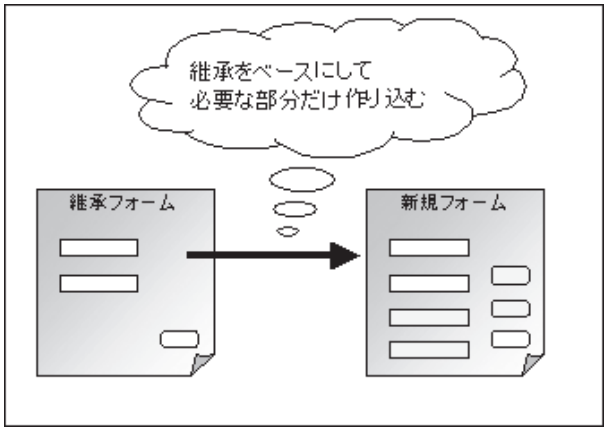

継承をベースにして
必要な部分だけ作り込む
継承フォーム
新規フォーム

図2

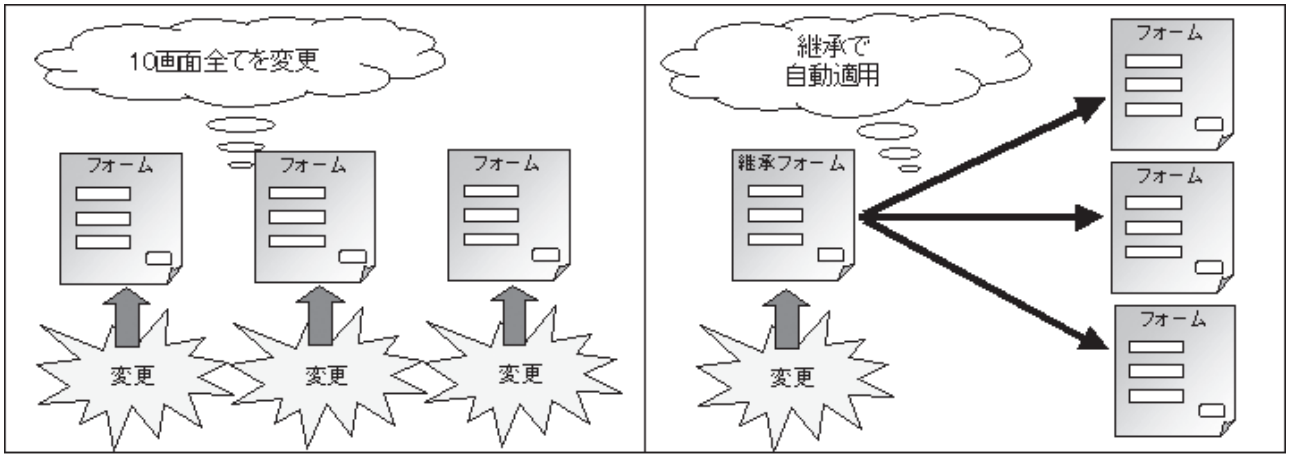
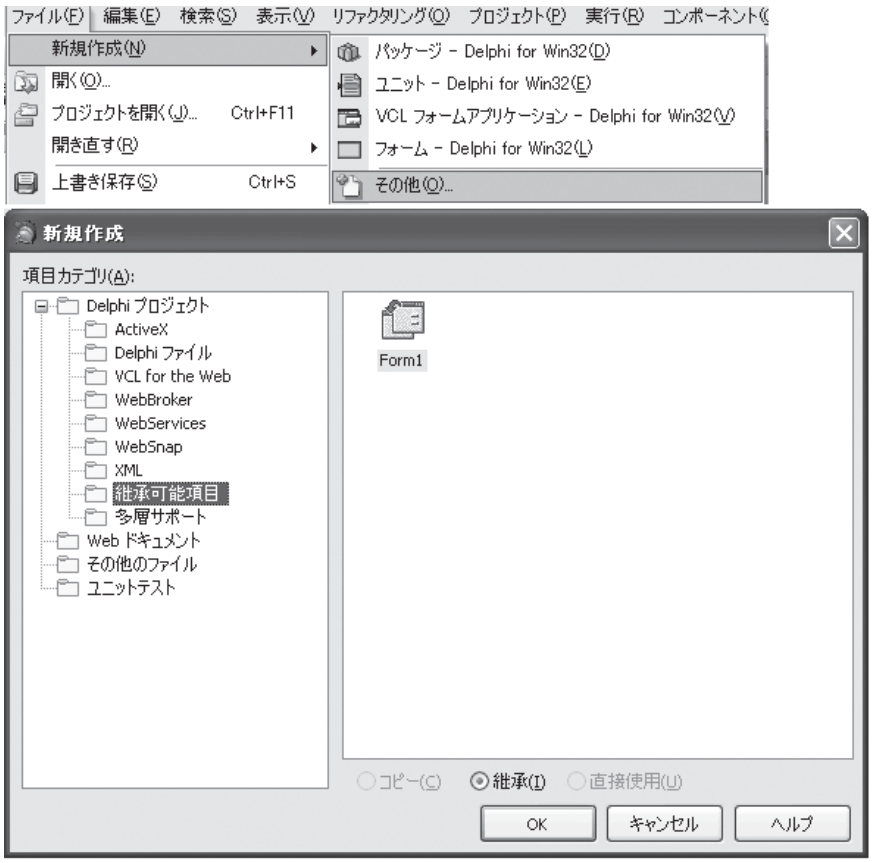

10画面全てを変更
フォーム
フォーム
フォーム
変更
変更
変更
継承で
自動適用
継承フォーム
フォーム
フォーム
フォーム
変更

図3


ファイル(F) 編集(E) 検索(S) 表示(V) リファクタリング(O) プロジェクト(P) 実行(R) コンポーネント(
新規作成(N)
開く(O)...
プロジェクトを開く(J)... Ctrl+F11
開き直す(R)
上書き保存(S) Ctrl+S
パッケージ - Delphi for Win32(D)
ユニット - Delphi for Win32(E)
VCL フォームアプリケーション - Delphi for Win32(V)
フォーム - Delphi for Win32(L)
その他(O)...
新規作成
項目カテゴリ(A):
Delphi プロジェクト
ActiveX
Delphi ファイル
VCL for the Web
WebBroker
WebServices
WebSnap
XML
継承可能項目
多層サポート
Web ドキュメント
その他のファイル
ユニットテスト
Form1
コピー(C)
継承(I)
直接使用(U)
OK
キャンセル
ヘルプ

フォームを変更すれば10画面全てに全て適用される。【図2】

Aは工数がかかるだけでなく、プログラムの開発/変更量が多い。ということは、それだけ人為的なミスが発生する可能性も高く、プログラム試験の工数もそれに伴って増大する。

それに対して、Bはプログラムの開発/変更量が少なく、また、継承化された部分は基本的に全て同じ動作となる。そのため、試験の確認パターンも必要最低限で済む、という違いがある。

ただし、上記2例からわかるように、継承化は必ずしも有効な開発手法ではあるとは限らない。継承の理解が誤っているために、逆に開発効率やメンテナンス性が悪くなってしまったという話もよく耳にする。

では、フォーム継承はどのように行えば効率がよい開発が行えるのか、これを順を追ってみていこう。

2 フォーム継承のポイント

フォーム継承がどういった特性を持つのかを把握しておかないと、煩雑なプログラムになり、後々のメンテナンスで逆に手間がかかってしまう。

では、フォーム継承を作成していくポイントをまとめてみる。

2-1. フォーム継承とは

継承とは、オブジェクト指向プログラミングにおいて、すでに定義されているオブジェクトをもとに、拡張や変更を加えた新しいオブジェクトを定義することである。

例えば、Excelで帳票を作成する場合、同じ帳票を印刷するたびに新規のExcelから作り込むことはあまりしないだろう。必要と思われる帳票フォーマットに対して、定型となるExcelをあらかじめ用意しておき、そのExcelに必要な内容だけ入力して印刷すれば手間もかからず、効率よく帳票が作成できる。

これと同じように、フォームの定型があらかじめ用意されていれば、そのフォームに対して必要な箇所だけ手を加えればよいので、便利ではないだろうか。これが、フォーム継承である。

継承フォームの新規作成についても押さえておこう。

Delphiのメニューから［ファイル］→［新規作成］→［その他］から、同じプロジェクト内のフォームを選択して、新規フォームを作成する。これにより、選択したフォームを継承した新しいフォームを作成することができる。【図3】

2-2. コンポーネントの配置

では、継承元フォームを作成するうえで、定型となるフォームにはどういったコンポーネントを配置するのが望ましいかを考えてみる。

仮に、継承元フォームに、照会画面で使う全ての画面項目を配置したとする。そうすると、全く同じ照会画面を作成する場合にはほとんど継承だけで動作することになり、一見非常に便利に思える。

しかし、少し違う照会画面になった場合、必ず不要になるコンポーネントがあることに気づくだろう。【図4】

・不要なコンポーネント

この不要なコンポーネントが多いと、その不要なコンポーネントを非表示にしたり、動作しないように制御したりすることで、逆に開発に手間がかかってしまう。また、不要なコンポーネントが存在すると、その画面の動作はそのぶんパフォーマンスが悪くなる。フォームの継承を行ううえで失敗するのは、このケースであることが非常に多い。

継承は便利である一方、継承で配置したコンポーネントは、継承先フォームでは削除することはできない。継承先フォームへの制約となることも意識する必要がある。

つまり、継承元フォームに配置するものは、その継承を利用して作成する画面に共通して必ず存在するコンポーネントだけにすることが望ましい。

例えば、検索を行う照会画面で項目の要/不要を選別してみる。【図5】

●**検索項目**：画面によって異なるので、不要になる可能性がある。
●**検索ボタン**：検索画面には必ず存在するので、必要である。

継承元フォームを作成するうえで最も重要なポイントは、継承画面に本当に必要な項目だけを過不足なく洗い出すことである。これは、上流工程の画面設計で、共通の画面レイアウトを考える際に意識しておくと、継承化の時にも格段に楽になる。

2-3. コンポーネントとコーディング

では、そうしたことを概念として理解しながらも、なぜ余計な画面項目を継承元フォームに配置して、失敗するケースが多いのだろうか。

その理由の多くは、処理プログラムを書くためである。

例えば、［クリア］ボタンを押すと、画面項目のEditコンポーネントの表示内容をクリアする機能を実装したとする。

その場合、開発者は、ソース1のようにコーディングしたい。【ソース1】

しかし、画面上にEdit1コンポーネントがないと、ソース1のようなコードは記述できないために、しかたなく継承元フォームに不要なEdit1・・・を配置してしまう。

では、検索項目をクリアする処理は、個別の画面フォームでコーディングするべきだろうか？ しかし、同じ処理を個別の画面にコーディングするのであれば、継承を行う意味が薄れてしまう。

・動的なコーディング

このような場合は、画面固有のコンポーネント名に依存しない、動的なコーディングを行う必要がある。

例えば、継承元フォームで、ソース2のようにコーディングする。画面のコンポーネント名がわからなくても、全てのEditコンポーネントを動的にクリアすることができる。【ソース2】

なお、これを応用すれば、Formの代わりにPanelコンポーネントのようなコンテナコンポーネントをあてはめて、コンテナコンポーネント上だけの制御も行える。

このように継承元フォームでは、できるだけ画面固有のコンポーネント名に依存しないコーディングが有効である。これによって、無駄なコンポーネントを配置する必要がなくなり、継承元フォーム

図4

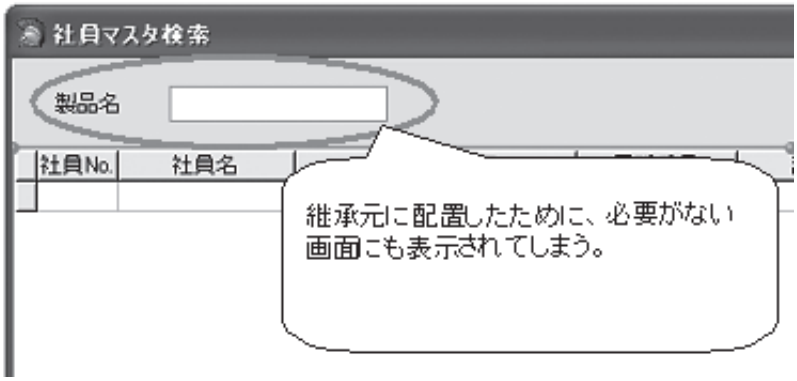

社員マスタ検索
製品名
社員No.
社員名
継承元に配置したために、必要がない
画面にも表示されてしまう。

図5

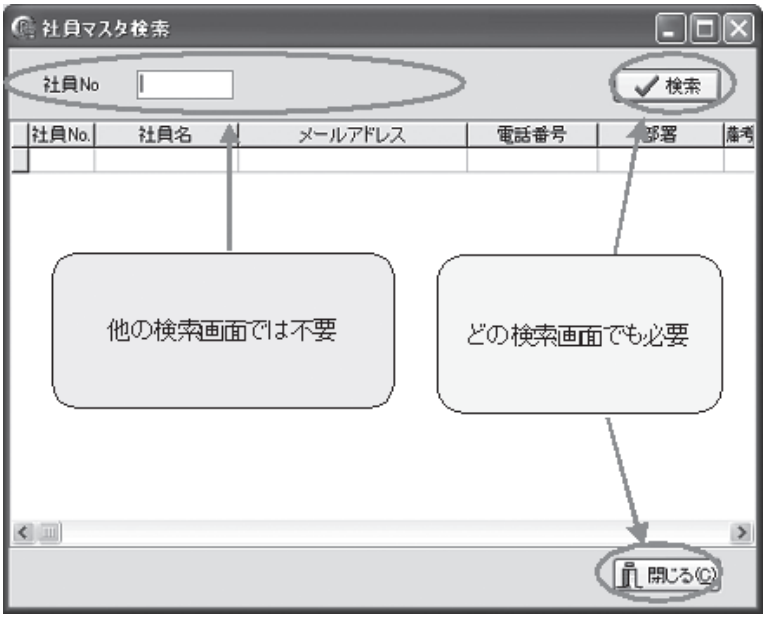
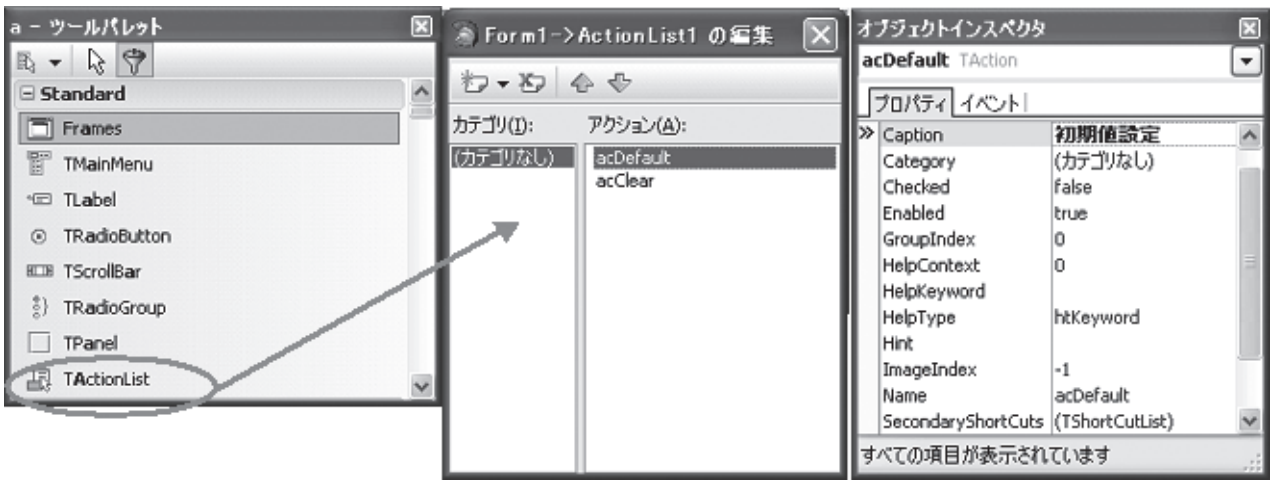

社員マスタ検索
社員No
検索
社員No.
社員名
メールアドレス
電話番号
部署
備考
他の検索画面では不要
どの検索画面でも必要
閉じる(C)

図6


a - ツールパレット
Standard
Frames
TMainMenu
TLabel
TRadioButton
TScrollBar
TRadioGroup
TPanel
TActionList
Form1->ActionList1 の編集
カテゴリ(I):
アクション(A):
(カテゴリなし)
acDefault
acClear
オブジェクトインスペクタ
acDefault TAction
プロパティ
イベント
Caption
初期値設定
Category
(カテゴリなし)
Checked
false
Enabled
true
GroupIndex
0
HelpContext
0
HelpKeyword
HelpType
htKeyword
Hint
ImageIndex
-1
Name
acDefault
SecondaryShortCuts
(TShortCutList)
すべての項目が表示されています

での活用範囲をさらに広げることができる。

2-4. イベント制御

継承元フォームに配置するコンポーネント同様、どのイベントをどこまで継承元で制御するかも重要なポイントである。

例えば、［閉じる］ボタンを押下したときにフォームを閉じるといったイベントは、継承元フォームで記述するということは想像しやすいだろう。また、検索や更新などの画面固有の処理は、継承先フォームに処理を記述すればよい。これらの切り分けは比較的簡単に行える。

では、画面起動時の処理を行うイベントは、継承元フォームと継承先フォームのどちらで処理を記述するべきだろうか。画面起動時に行う処理を考えてみよう。

【画面起動時の処理】
①画面項目をクリアする。
②画面項目に初期値をセットする。

①は、画面全体に対しての処理なので、前述の 2-3 のように継承元フォームで処理できそうである。一方、②の初期値は、画面ごとに内容やセットする項目が異なるため、継承先フォームで処理することになるだろう。

なお、こういった場合、イベントの処理内容はどちらかにまとめて記述する必要がある、というようなことはない。

①のイベント処理は、継承元フォームに記述しておくことができる。
②の内容は、継承先フォームで同イベントに記述する。

ここで気をつけなければならないのは、同じイベントが、それぞれどのタイミングで実行されるかということである。【ソース 3】【ソース 4】

同じイベントで処理を記述した場合、継承元フォームのイベント内容は inherited の箇所で実行される。

これは、デバッグ実行時における F7 キーでのトレースや ShowMessage を組み込んで、実行される実際の順番を確認すると理解しやすい。

つまり、こういったイベント内の内容を切り分けることによっても、継承先フォームでの開発負担を軽減することができる。

2-5 アクションコンポーネントの利用

別の視点からイベントを見てみようか。前述の①と②の処理順は決まっているので、この処理順を、継承元フォームで制御することはできないだろうか。

そのような場合に便利に使えるのが、TActionList というコンポーネントである。TActionList では、TAction というアクションコンポーネントを持つことができる。この TAction コンポーネントであらかじめ想定される動作を用意しておけば、継承元フォームで、継承先のイベントの制御に利用することができる。【図 6】

例えば、②の初期値設定処理を想定して、アクションとして acDefault という TAction コンポーネントを継承元フォームに配置しておく。

このアクションを利用して、前述の画面起動時の処理を変えてみよう。【ソース 5】【ソース 6】

一見、同じようなプログラムに見えるが、処理①②の順番を制御しているのは、継承元フォームである。

これにより、フォーム継承を利用して個別画面（継承先フォーム）を作成する開発者は、inherited などの継承を考慮する必要がなくなる。acDefaultExecute イベントに初期値設定の処理を記述しておけば、継承元フォームが自動的に適切な箇所でその処理を呼び出してくれる。

例えば、acDefault は初期値設定処理、acSearch は検索処理など、アクション名を決定して継承元フォームに配置しておく。そうすれば、各開発者はそのアクションのロジック作成のみに専念することができ、また、システム全体としても動作の統一を行うことが容易になる。

3 アプリケーション内の継承構成

2 で、継承元フォームの作成のポイントをまとめた。では、この継承元フォームは一体どの程度用意しておけばよいのだろうか。

フォームの継承は何階層でも多重継承が可能なので、汎用性を考えると細分化された継承フォームを用意する必要がある。

例えば、メニュー画面、検索画面、更新画面から構成されるアプリケーションの構成を考えてみよう。

まず単純に、3 つの画面フォームのパターンが考えられる。【図 7】

このうち、アプリケーション共通の画面デザイン（タイトルラベルやセッション表示など）を別々に作成する必要はないので、画面デザインだけの共通フォームを用意する。【図 8】

また、同じように検索処理画面においても、一覧検索画面や詳細情報検索画面などレイアウトが異なる可能性があるので、検索ボタンを配置した検索の共通画面を用意する。同様に、更新画面も画面のパターンを用意する。【図 9】

このように、継承フォームを細分化すれば、より多くの画面に対応することができる。その際、構築するシステムの規模や複雑さ、画面パターンなどによって、その細分化のレベルは異なる。

また、新しいパターンの画面が追加された場合は、それに対応する継承フォームを追加していけばよい。そのため、後から継承フォームが追加できるように細分化を考えておく必要がある。

ただし、継承フォームの細分化によって汎用的で便利になる一方、継承フォーム自体の管理が手間になってしまっては意味がない。その見極めには十分に配慮する必要がある。

4 フォーム継承によるメリット

フォーム継承を利用するときには、さまざまな開発ポイントがある。

システム構築当初に、これらをしっかり検討してフォームの継承を利用すれば、システム構築では、以下のような利点が得られる。

●開発効率
① 1 画面あたりの開発工数が少なくなる。
同系統の画面作成に、同じ工数をかける必要がなくなる。
②開発者に求められるプログラムの敷居を下げることができる。
難易度の高い共通ロジックを継承元で

図7


メニュー画面
検索画面
更新画面

図8

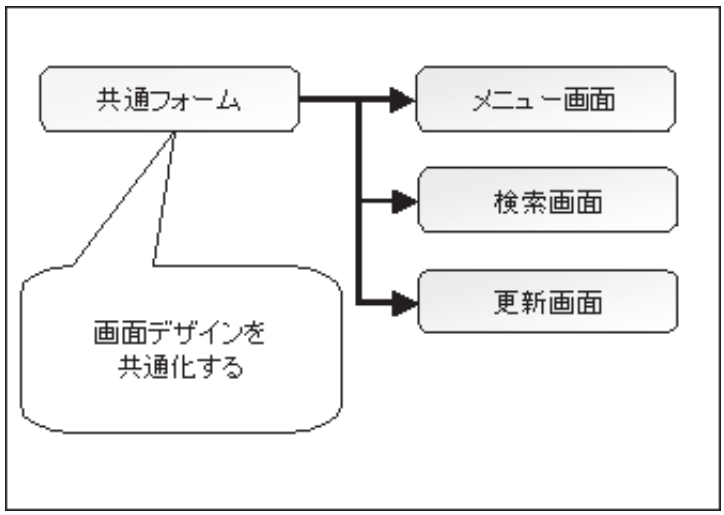
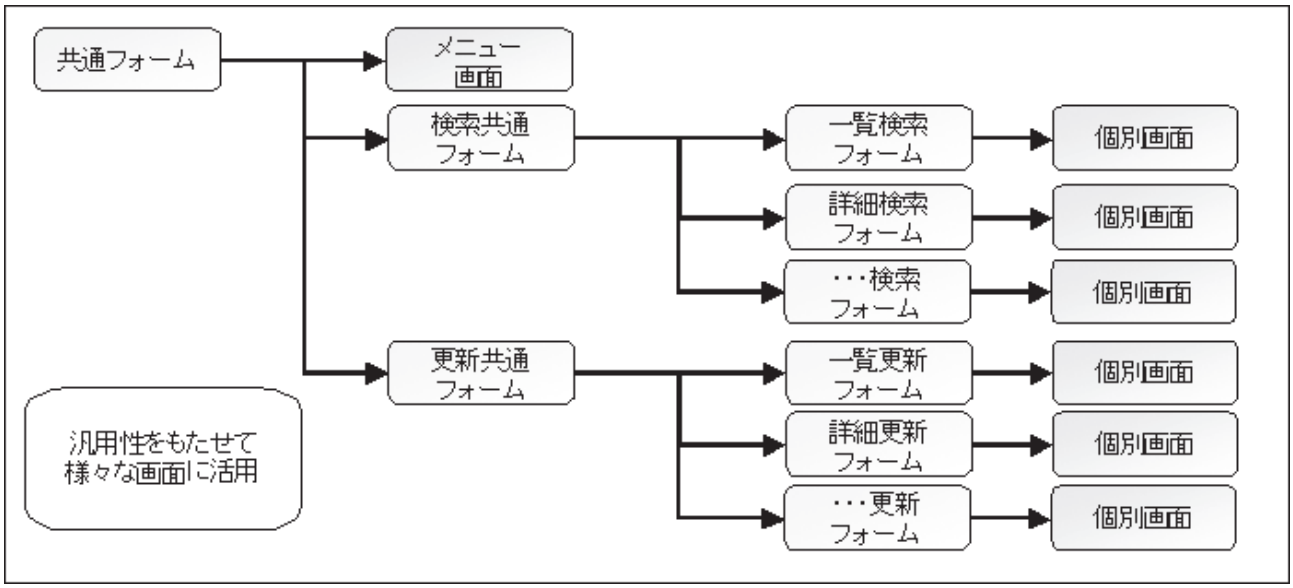

共通フォーム
メニュー画面
検索画面
更新画面
画面デザインを
共通化する

図9


共通フォーム
メニュー
画面
検索共通
フォーム
一覧検索
フォーム
個別画面
詳細検索
フォーム
個別画面
・・・検索
フォーム
個別画面
更新共通
フォーム
一覧更新
フォーム
個別画面
詳細更新
フォーム
個別画面
・・・更新
フォーム
個別画面
汎用性をもたせて
様々な画面に活用

行うことで、各画面開発で必要とされるプログラムの難易度が下がる。

③画面動作の統一を開発者依存ではなく、継承上で管理できる。
開発者によるプログラムのばらつきを継承元で統一する。

●メンテナンス性

④システムが統一された作りになることで、管理が容易になる。
システムの画面や動作が統一されることで、画面ごとの動作やレイアウトの違いなどの煩雑な問題を解消できる。

⑤画面共通の変更が継承元の変更工数だけで対応できる。
画面共通の変更が継承元一箇所で行えるため、迅速かつ安全なメンテナンスが行える。

これらは、システムの規模や開発人員が多いほど､効果をあげることができる。また、洗練された継承の仕組みは、新たに開発する別のシステムにも流用して生かしていくことができるだろう。

M

ソース1

```
// 項目のクリア
Edit1.Clear；
Edit2.Clear；
・・・
```

ソース2 クリア処理

```
// フォーム全てのコンポーネントを処理
for i := 0 to Form1.ControlCount - 1 do
begin
  //Edit であれば Edit として扱う
  if (Form1.Controls[i] is TEdit) then
    TEdit(Form1.Controls[i]).Clear;
end;
```

ソース3 継承元フォーム

```
// 継承元フォームの FormCreate イベント
procedure TFormBase.FormCreate(Sender: TObject);
begin
  // 処理①
end;
```

ソース4 継承先フォーム

```
// 継承先フォーム画面の FormCreate イベント
procedure TForm1.FormCreate(Sender: TObject);
begin
  inherited; // ここでイベントが継承されて①が実行される
  // 処理②
end;
```

ソース5 継承元フォーム

```
// 継承元フォームの FormCreate イベント
procedure TFormBase.FormCreate(Sender: TObject);
begin
  // 処理①
  acDefault.Execute; // ここで処理②の内容が実行される。
end;
```

ソース6 継承先フォーム

```
// 継承先フォーム画面の acDefault の Execute イベント
procedure TForm1.acDefaultExecute(Sender: TObject);
begin
  inherited;
  // 処理②
end;
```

鶴巣 博行

株式会社ミガロ

RAD事業部 技術支援課

APIを利用した出力待ち行列情報の取得方法

IBM i のプログラミングはそこそこできるが
Delphi/400 はまだ使いこなせていないという人のために
このレポートを通じ、出力待ち行列処理の情報を画面上に表示し
表示した情報に対して処理を実行する簡単な方法を解説する。

略歴
1968 年 3 月 29 日 生
1990 年摂南大学 経営情報学部卒
2002 年株式会社ミガロ入社
2002 年 4 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
現場（営業グループとシステム事業部）が活動しやすいような支援と、ミガロおよびミガロ製品の積極的なアピール活動を志している。

はじめに

Delphi/400 をご使用のユーザー様でアプリケーションの GUI 化はできているが、出力待ち行列処理等の管理作業のため、なかなか 5250 エミュレータを捨てきれないという声をよく聞く。

例えば､出力待ち行列の処理であれば、Delphi/400 には、SPOOLLIST400 や SPOOL400 という出力待ち行列処理を扱うための便利なコンポーネントが用意されている。それらのコンポーネントを駆使することで、出力待ち行列処理の制御を行うことができる。

Delphi/400 では、IBM i 上のデータを GUI 画面に表示させることが簡単にできる。また、GUI 画面上のデータをパラメータとし、IBM i のプログラムを呼び出すことも、そんなに難しいことではない。ゆえに、出力待ち行列処理の情報をデータベース化できさえすれば、簡単にその情報を GUI 画面上に表示することが可能だ。

また、出力待ち行列処理画面のオプションで行われる処理（例えば、3= 保留 4= 削除 5= 表示 6= 解放）は、結局のところ、選択されたファイルに対してコマンドを発行（保留：HLDSPLF 削 除：DLTSPLF 表 示：DSPSPLF 解放：RLSSPLF）しているだけである。なので、コマンド発行に必要な情報を付加し、例えば CALL400 コンポーネントを使用してプログラムで処理するか、もしくは AS400 コンポーネントで直接コマンドを発行してやれば、処理を行うことができる。これは、出力待ち行列処理画面のオプションで行われる処理と同じ処理である。

データベース化する方法

まずは、出力待ち行列処理の情報をデータベース化する方法である。

コマンドで実現できないかといろいろ調べたのだが、直接データベース化できそうなコマンドは見つけられなかった。

そこで、コマンドとプログラムとの組み合わせで、何とかデータベース化する方法を思いついた。だか、やり方が強引であることと取得できる情報量が少ないことから、あまり実用性がないと判断した。【図 1】

IBM アンサーラインや Web などを利用し、他に実現方法がないか調査したところ、API を使用し、出力待ち行列処理が取得できることがわかった。

APIとは

API（アプリケーション・プログラミング・インタフェース、Application Programming Interface）について、IT 用語辞典（e-Words）の説明をそのまま引用する。

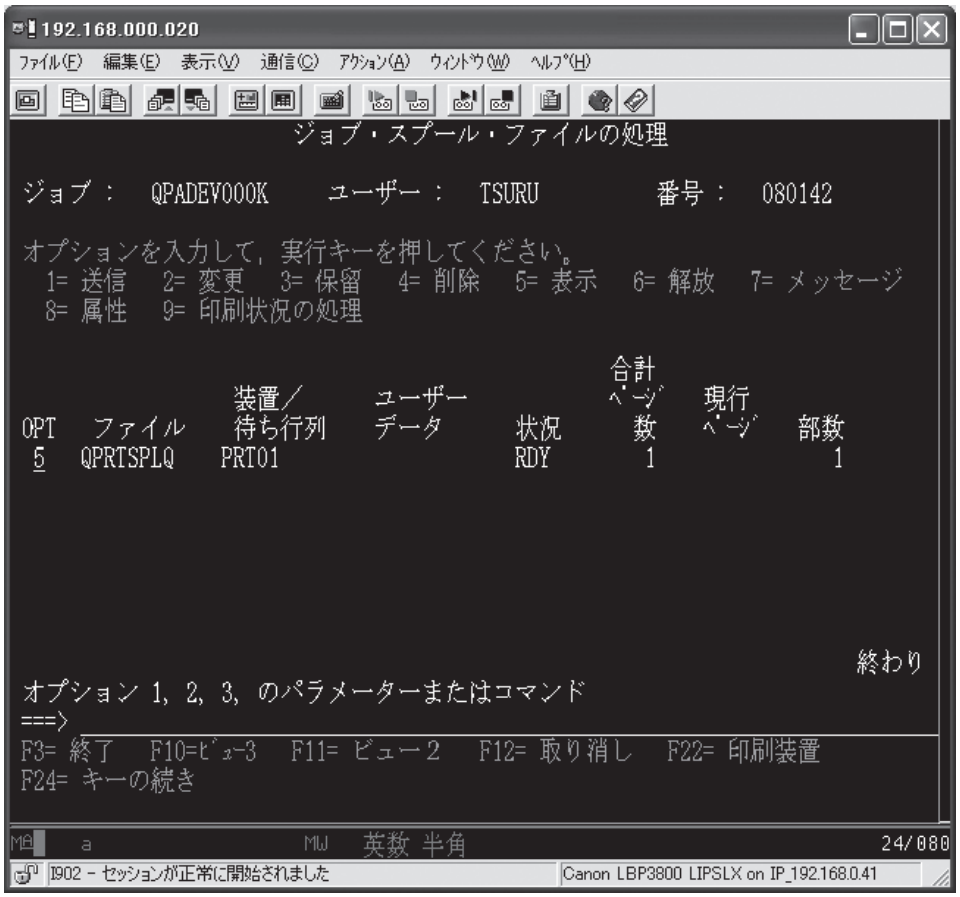
「あるプラットフォーム（OS やミドルウェア）向けのソフトウェアを開発する際に使用できる命令や関数の集合のこと。また、それらを利用するためのプログラム上の手続きを定めた規約の集合。

個々のソフトウェアの開発者がソフトウェアの持つすべての機能をプログラミングするのは困難で無駄が多いため、多

図1　コマンドとプログラムのを組合わせた出力待ち行列のデータベース化

手順１．WRKOUTQ コマンドでデータベース化したい OUTQ の一覧を印刷する。

WRKOUTQ OUTQ(QGPL/QPRINT) OUTPUT(*PRINT)



くのソフトウェアが共通して利用する機能は、OSやミドルウェアなどの形でまとめて提供されている。

個々の開発者は規約に従ってその機能を「呼び出す」だけで、自分でプログラミングすることなくその機能を利用したソフトウェアを作成することができる」

OSのコマンドでサポートされない機能をサポートする、ということが主要な目的のようだ。IBM iのOSで用意されているAPIは、主にオブジェクトの属性取り出しやオブジェクトの一覧処理だが、その他にも数字の編集などがある。(APIを使いこなしているわけではないので、もっと他にもたくさんあるかもしれない)。

難点は、わかりやすいマニュアルがない点だが、使用方法はパラメータで要求情報をAPIプログラムに渡し、結果がパラメータで返ってくるといったものなので、各々のパラメータの意味がわかれば使用できる。

コーディング方法

まず、APIを使用して取得できる情報のDDSを作成する。データベース名はUSRSPLPとし、用意するフィールドは［図2］のとおりである。【図2】

あとは、このデータベースにデータを落とすプログラムを作成する。ソースコードは［図3］のようになる。これを参考にしながら説明を読んでほしい。【図3】

①ユーザー・スペースの作成

ユーザー・スペースとは、APIで使用する新しいオブジェクト（オブジェクトタイプは＊USRSPC）で、一覧形式で予め何レコードの結果が出るかわからないものを検索する場合に使用される。

といっても、難しい作業をする必要はなく、ユーザー・スペース作成用のAPI（QUSCRTUS）が用意されている。なので、そのAPIを規定のパラメータに値をセットし、呼び出せば作成してくれる。

今回は、ライブラリー：QTEMPに、ユーザー・スペース：SPLINFを、ユーザー・スペース初期サイズ：1024で作成するようにしている。

ユーザー・スペースは「総称見出し」と「リストセクション」から構成される。(自分なりにわかりやすく表現すると、ヘッダーと明細の関係みたいな感じではないかと思う)。

②ユーザー・スペースに出力待ち行列のデータを展開

次に、①で用意したユーザー・スペースに、出力待ち行列の情報を展開する。

使用するAPIはQUSLSPLである。QUSLSPLに、以下のパラメータをセットし実行する。

・ユーザー・スペース（20桁）
・フォーマット形式（8桁）
・出力待ち行列のユーザー（10桁）
・OUTQ（20桁）
・フォームタイプ（10桁）
・ユーザーデータ（10桁）

今回はすべてを対象にしているが、このパラメータのセットの仕方により、取得したい出力待ち行列を制御することが可能である。

例えば、待ち行列QPRINT分だけを取得したい場合は、パラメータのOUTQに“QRINT △△△△△ QGPL △△△△△△”（三角はスペース）と指定する。

データの展開が失敗した場合は（ソース上では標識95がONになる）、処理が中断される。

③総称見出しからデータを取り出す

次に、②で取得した情報から総称見出しの情報を取得する。(取得したい項目は、リスト・データ・セクションのオフセットと、リスト・データ・セクションのサイズと、リストセクションに書き出された数)。

使用するAPIはQUSRTVUSである。QUSRTVUSに、以下のパラメータをセットし実行する。

・ユーザー・スペース（20桁）
・開始位置（4桁）
・取得数（4桁）
・結果を入れるフィールド

総称見出しのレイアウトは［図4］のとおりである。レイアウトを見るとオフセット10進数が0から始まっている。ただし、QUSRTVUS APIの開始位置を決定するためには、オフセット値に1を加算しなければならないので注意しよう。

例えば、レイアウト中の情報の状況を使用したい場合、開始位置は104（オフセット10進数:103 + 1）となる。【図4】

今回の場合、開始位置を125（リスト・データ・セクションのオフセット）から16バイト分（各項目のサイズまで）の情報を取得する。取得した情報は、I仕様書のDSで分解する。

総称見出しからのデータ取り出しが失敗した場合は（ソース上では標識95がONになる）、処理が中断する。また、取得したリスト項目の数(フィールド名:NOENTH）が0の場合は、データがないということなので、この場合も処理を中断する。

④リストセクション読み込みの初期化

⑤で使用するAPIのパラメータ値の初期設定を行う。

③で取得したリスト･データ･セクションのオフセットに、1を足し、パラメータ値にセットする。また、同じく③で取得したリスト・データ・セクションのサイズも、パラメータ値にセットする。

⑤リストセクションの各項目の取り出し

②で取得した出力待ち行列の情報を、リストセクションから取得する。

使用するAPIは③で使用したのと同じQUSRTVUSである。パラメータSF0100に結果がセットされる。取得できる情報は、以下のものである。

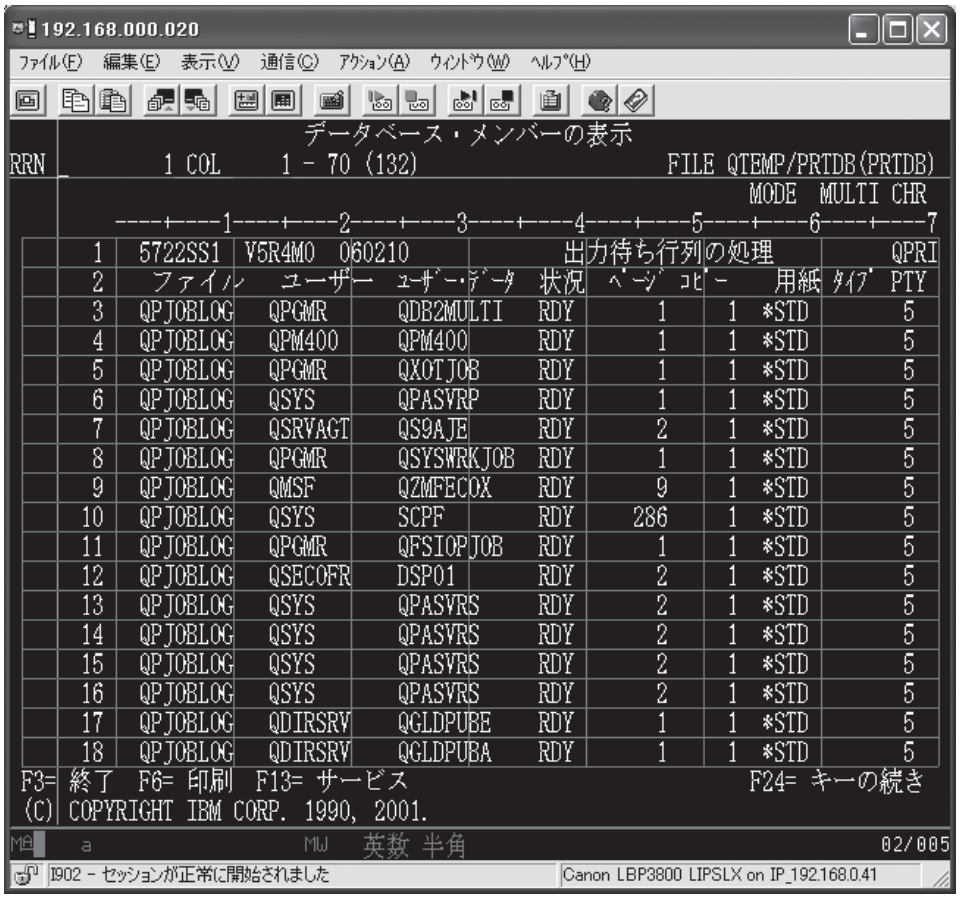
・ユーザー名（USRNML）
・OUTQとライブラリー（OUTQL）
・フォームタイプ名（FRMTYL）
・ユーザーデータ（USRDTL）
・内部JOB確認者（JOBID）
・内部スプール・ファイル確認者（PLFID）

次に、QUSRTVUSで取得した内部JOB確認者と内部スプール・ファイル確認者をもとに、追加の属性を取得する。(サブルーチン @ATR)

使用するAPIはQUSRSPLAである。QUSRSPLAに、以下のパラメータを

手順２．印刷したスプールファイルを CPYSPLF でデータベース化する。

ｉ．データベースを作成する。
CRTPF FILE(QTEMP/PRTDB) RCDLEN(132) IGCDTA(*YES)
ⅱ．CPYSPLF コマンド実行
CPYSPLF FILE(QPRTSPLQ) TOFILE(QTEMP/PRTDB)



手順３．プログラムでデータ化した一覧を項目毎に区切り DDS をきったデータベースに移行する。

ｉ．１レコードずつ手順２で作成したファイルを読み込む。
ⅱ．３桁目が英字（A ～ Z）のレコードを対象とする。
ⅲ．３～１２桁目をファイル名、１４～２３桁目までをユーザー名といった手順で各フィールド毎に切り分け、フィールドにセットする。

セットし実行する。

・結果を受け取るフィールド
・結果を受け取るフィールドのサイズ（4桁）
・形式名（8桁）
・修飾ジョブ名（26桁）
・内部ジョブ識別コード（16桁）
・内部スプール･ファイル識別コード（16桁）
・スプール・ファイル名（10桁）
・スプール・ファイル番号（4桁）

取得できる情報は、USRSPLPのレイアウト［図2］の情報になる。

⑥項目の書き出し

⑤で取得した情報を、データベースUSRSPLPの各フィールドにセットし、レコードを書き出す。

コーディング方法を①～⑥に述べた。⑤と⑥の作業を、③で取得したリスト項目の数（フィールド名：NOENTH）分繰り返すことで、出力待ち行列の情報をデータベース化できる。

あとは、データベース化した情報を、Delphi/400を使用して、GUI画面上にグリッド等を利用し表示する。出力待ち行列を照会するだけであれば、これで完成だ。

出力待ち行列への処理実行

出力待ち行列に対して処理を実行させたい場合は、処理ごとのボタン（保留、削除、表示、解放）を用意する。

各ボタンから呼び出されるプログラムは、共通でも、ボタンごとにCall400コンポーネントでプログラムを呼び出しても、AS400コンポーネントでリモートコマンドを発行してもかまわない。

今回は、共通のCLPを使用する例をあげる。【図5】

CLP実行時パラメータを以下のようにする。

・処理区分（1桁）
　保留：3　削除：4　表示：5　送信：6
・スプール・ファイル名（10桁）
・JOB名（10桁）
・ユーザー名（10桁）
・ジョブ番号（6桁）
・SPLF番号（6桁）
・JOBシステム名（8桁）

各ボタン押下時、押されたボタンの処理区分（保留：3　削除：4　表示：5　送信：6）をセットし、グリッドで選択されているスプール・ファイルからパラメータに必要な情報をセットし、CLPを呼び出すロジックを組み込めば、プログラムが完成する。IBM iの出力待ち行列の処理と、同等の仕組みを持ったプログラムである。

まとめ

ここまで述べてきたように、出力待ち行列の処理であれば、

・出力待ち行列表示用画面（Delphi/400）
・出力待ち行列データ作成プログラム（RPG）
・出力待ち行列処理用のプログラム（CLP）

を組み合わせれば、簡単に作成することができる。

APIを使用する部分は若干難しいかもしれないが、APIの使用ルールは形式が定まっているので、そのとおり作成すれば問題ない。

IBM iには、今回取り上げた出力待ち行列処理以外にもいろいろな管理画面がある。結局のところ、どこからかデータを取得し、画面に表示し、何らかの処理を実行する場合、そのデータ値をパラメータとしコマンドを発行しているだけである。なので、データの取得方法さえわかれば、同様の処理をGUI化することはそう難しくないと思う。

今後も便利な情報を提供するために、Delphi/400をより有効に使用できる手段を見つけたいと思っている。

M

現在の仕事内容（詳細）
セミナーやフェアの開催と集客、お客様への製品説明や提案といった営業推進活動、および、製品とお客様からの問い合わせへの対応などの顧客サポート活動を行っている。また、製品出荷やメンテナンス更新管理、売上管理などの営業事務管理と、ホームページ更新やメールマガジンといった宣伝活動も行っている。

図2　APIを使用して取得できる出力待ち行列情報

```
A*----------------------------------------------------------------*
A*  FILE ID.           :  USRSPLP                                  *
A*  DESCRIPTION        :  スプールファイル属性                     *
A*  FORMAT ID.         :  USRSPLR                                  *
A*                                                                 *
A*----------------------------------------------------------------*
A          R USRSPLR
A            SPL001        4B 0       COLHDG('返り値バイト数')
A            SPL002        4B 0       COLHDG('利用可能バイト数')
A            SPL003        16A        COLHDG('内部 JOB 確認者')
A            SPL004        16A        COLHDG('内部 SPLF 確認者')
A            SPL005        10A        COLHDG('ジョブ名')
A            SPL006        10A        COLHDG('ユーザー名')
A            SPL007        6A         COLHDG('ジョブ番号')
A            SPL008        10A        COLHDG('SPLF 名')
A            SPL009        4B 0       COLHDG('SPLF 番号')
A            SPL010        10A        COLHDG('形式タイプル名')
A            SPL011        10A        COLHDG('ユーザー指定データ')
A            SPL012        10A        COLHDG('状態ファイル名')
A            SPL013        10A        COLHDG('利用可能ファイル')
A            SPL014        10A        COLHDG('書込前 HOLD FILE')
A            SPL015        10A        COLHDG('記述後ファイル保存')
A            SPL016        4B 0       COLHDG('合計ページイル保存')
A            SPL017        4B 0       COLHDG('書き込み中ページ')
A            SPL018        4B 0       COLHDG('開始ページ')
A            SPL019        4B 0       COLHDG('終了ページ')
A            SPL020        4B 0       COLHDG('最終印刷ページ')
A            SPL021        4B 0       COLHDG('再印刷')
A            SPL022        4B 0       COLHDG('トータルコピー数')
A            SPL023        4B 0       COLHDG('生成用コピー残')
A            SPL024        4B 0       COLHDG('行／インチ')
A            SPL025        4B 0       COLHDG('文字／インチ')
A            SPL026        2A         COLHDG('優先出力')
A            SPL027        10A        COLHDG('出力キュー名名')
A            SPL028        10A        COLHDG('出力 Q ライブラリ名')
A            SPL029        7A         COLHDG('日付作成ファイル')
A            SPL030        6A         COLHDG('時間作成ファイル')
A            SPL031        10A        COLHDG('装置ファイル名')
A            SPL032        10A        COLHDG('装置ファイル LIB')
A            SPL033        10A        COLHDG('FILE を開いた PGM')
A            SPL034        10A        COLHDG('')
A            SPL035        15A        COLHDG('経理基準')
A            SPL036        30A        COLHDG('テキスト印刷')
A            SPL037        4B 0       COLHDG('レコード長')
A            SPL038        4B 0       COLHDG('最大レコード数')
A            SPL039        10A        COLHDG('装置タイプ')
A            SPL040        10A        COLHDG('プリンタ装置タイプ')
A            SPL041        12A        COLHDG('ドキュメント名')
A            SPL042        64A        COLHDG('フォルダ名ル名')
A            SPL043        8A         COLHDG('SYSTEM36 手続き名')
A            SPL044        10A        COLHDG('印刷精度')
A            SPL045        1A         COLHDG('印刷不可能文字変換')
A            SPL046        1A         COLHDG('文字置換')
A            SPL047        4B 0       COLHDG('ページ長')
A            SPL048        4B 0       COLHDG('ページ幅')
A            SPL049        4B 0       COLHDG('分割数')
A            SPL050        4B 0       COLHDG('オーバフロー行番号')
A            SPL051        10A        COLHDG('DBCS データル名')
A            SPL052        10A        COLHDG('DBCS 拡張文字')
A            SPL053        10A        COLHDG('DBCS 変更間隔')
A            SPL054        10A        COLHDG('DBCS 文字回転')
A            SPL055        4B 0       COLHDG('DBCS 文字／インチ')
A            SPL056        10A        COLHDG('GRAPHIC 文字 SET')
A            SPL057        10A        COLHDG('コードページ')
A            SPL058        10A        COLHDG('定義名形式')
A            SPL059        10A        COLHDG('ライブラリ定義形式')
A            SPL060        4B 0       COLHDG('ソース引出ル名')
A            SPL061        10A        COLHDG('プリンタフォント')
A            SPL062        6A         COLHDG('SYS36 SPLF 確認者')
A            SPL063        4B 0       COLHDG('ページ回転')
A            SPL064        4B 0       COLHDG('正当化')
A            SPL065        10A        COLHDG('両面印刷')
A            SPL066        10A        COLHDG('重複レコード')
A            SPL067        10A        COLHDG('制御文字')
A            SPL068        10A        COLHDG('整列形式')
A            SPL069        10A        COLHDG('印刷品質')
A            SPL070        10A        COLHDG('形式材料')
A            SPL071        71A        COLHDG('ボリューム（配列）')
A            SPL072        17A        COLHDG('FILE ラベル確認者')
A            SPL073        10A        COLHDG('交換タイプ')
A            SPL074        10A        COLHDG('文字コード')
A            SPL075        4B 0       COLHDG('合計レコード数')
A            SPL076        4B 0       COLHDG('倍角文字')
A            SPL077        10A        COLHDG('前面オーバーレイ')
A            SPL078        10A        COLHDG('ライブラリー')
A            SPL079        15P 5      COLHDG('下方向オフセット')
A            SPL080        15P 5      COLHDG('横方向オフセット')
A            SPL081        10A        COLHDG('背面オーバーレイ')
A            SPL082        10A        COLHDG('ライブラリー')
A            SPL083        15P 5      COLHDG('下方向オフセット')
A            SPL084        15P 5      COLHDG('横方向オフセット')
A            SPL085        10A        COLHDG('ユニット寸法')
A            SPL086        10A        COLHDG('ページ定義名')
A            SPL087        10A        COLHDG('ページ定義 LIB 名')
A            SPL088        10A        COLHDG('ライン間隔')
A            SPL089        15P 5      COLHDG('ポイントサイズ')
A            SPL090        15P 5      COLHDG('下へのマージン')
A            SPL091        15P 5      COLHDG('横へのマージン')
A            SPL092        15P 5      COLHDG('下へのマージン')
A            SPL093        15P 5      COLHDG('横へのマージン')
A            SPL094        15P 5      COLHDG('ページ長')
A            SPL095        15P 5      COLHDG('ページ幅')
A            SPL096        10A        COLHDG('測定方法イル名')
A            SPL097        1A         COLHDG('高度関数印刷資源')
A            SPL098        10A        COLHDG('文字セット名')
A            SPL099        10A        COLHDG('文字セット LIB 名')
A            SPL100        10A        COLHDG('コードページ名')
A            SPL101        10A        COLHDG('コードページ LIB')
A            SPL102        10A        COLHDG('コードフォント名')
A            SPL103        10A        COLHDG('コードフォント LIB')
A            SPL104        10A        COLHDG('DBCS コードフォント')
A            SPL105        10A        COLHDG('DBCS コード FONT LIB')
A            SPL106        10A        COLHDG('ユーザー定義 FILE')
A            SPL107        10A        COLHDG('縮小出力')
A            SPL108        1A         COLHDG('固定バック')
A            SPL109        4B 0       COLHDG('BIN 出力 ル名')
A            SPL110        4B 0       COLHDG('CCSID ァイル名')
A            SPL111        100A       COLHDG('ユーザ定義テキスト')
A            SPL112        8A         COLHDG('ファイル作成 SYSTEM')
A            SPL113        8A         COLHDG('ファイル作成 ID')
A            SPL114        10A        COLHDG('ファイル作成者')
A            SPL115        2A         COLHDG('保存')
A            SPL116        4B 0       COLHDG('ユーザ定義 OPT')
A            SPL117        4B 0       COLHDG('戻値ユーザ定義 OPT')
A            SPL118        4B 0       COLHDG('ユーザ定義 ENTRY')
A            SPL119        255A       COLHDG('ユーザ定義データ')
A            SPL120        10A        COLHDG('ユーザ定義 OBJ')
A            SPL121        10A        COLHDG('ユーザ定義 OBJ LIB')
A            SPL122        10A        COLHDG('ユーザー OBJ TYPE')
A            SPL123        3A         COLHDG('保存')
A            SPL124        15P 5      COLHDG('文字セットサイ')
A            SPL125        15P 5      COLHDG('コードフォント SIZE')
A            SPL126        15P 5      COLHDG('DBCSCODE FONTSIZE')
A            SPL127        4B 0       COLHDG('補助記憶装置プール')
A            SPL128        4B 0       COLHDG('SPLF サイズ')
A            SPL129        4B 0       COLHDG('SPLF サイズ乗数')
A            SPL130        4B 0       COLHDG('INTERNET 印刷')
A            SPL131        1A         COLHDG('SPLF生成セキュリティ')
A            SPL132        1A         COLHDG('SPLF 生成認証方法')
A            SPL133        7A         COLHDG('SPLF 処理開始者日付')
A            SPL134        6A         COLHDG('SPLF 処理開始者時間')
A            SPL135        7A         COLHDG('SPLF 処理完成者日付')
A            SPL136        6A         COLHDG('SPLF 処理完成者時間')
A            SPL137        8A         COLHDG('ジョブシステム名')
```

図3　APIを使用して出力待ち行列情報を取得するプログラム

```
H           Y
F*************************************************************
F*DEFINE
F*************************************************************
FUSRSPLP O  E           K        DISK
F*
IUSRSPC     DS
I                                        1  10 USNAME
I                                       11  20 USLIB
IGENHED     DS
I                                    B   1   4ODTLOFS
I                                    B   5   8OLSTSIZ
I                                    B   9  12ONOENTH
I                                    B  13  16ODTLSIZ
I           DS
I                                    B   1   4OUSSIZE
I                                    B   5   8OLENDTA
I                                    B   9  12OSTRPOS
I           DS
I                                    B   1   4ORCVLEN
I                                    B   5   8OSPLF#
ISF0100     DS
I                                        1  10 USRNML
I                                       11  30 OUTQL
I                                       11  20 OUTQNL
I                                       21  30 OUTQLL
I                                       31  40 FRMTYL
I                                       41  50 USRDTL
I                                       51  66 JOBID
I                                       67  82 SPLFID
ISA0100     DS
I                                    B   1   4OBYTESR
I                                    B   5   8OBYTESA
I                                        9  24 LJOBID
I                                       25  40 LSPLID
I                                       41  50 JOBNAM
I                                       51  60 USRNAM
I                                       61  66 JOBNUM
I                                       67  76 FILNAM
I                                    B  77  80OFILNUM
I                                       81  90 FRMTYP
I                                       91 100 USRDTA
I                                      101 110 STAT
I                                      111 120 FILAVA
I                                      121 130 HOLDF
I                                      131 140 SAVF
I                                    B 141 144OTOTALP
I                                    B 145 148OBEGWRT
I                                    B 149 152OSTART
I                                    B 153 156OEND
I                                    B 157 160OLASTPG
I                                    B 161 164ORESTAT
I                                    B 165 168OCOPYS
I                                    B 169 172OPRODUC
I                                    B 173 176OLINPER
I                                    B 177 180OCHRPER
I                                      181 182 PRIORT
I                                      183 192 OUTQNM
I                                      193 202 OUTQLB
I                                      204 209 DATFOP
I                                      210 215 TIME
I                                      216 225 DVFINM
I                                      226 235 DVFLNM
I                                      236 245 POFINM
I                                      246 255 POFLNM
I                                      256 270 ACCCD
I                                      271 300 TEXT
I                                    B 301 304ORECLEN
I                                    B 305 308OMAXREC
I                                      309 318 DEVTYP
I                                      319 328 PRDVTP
I                                      329 340 DOCNAM
I                                      341 404 FLDNAM
I                                      405 412 S36PNM
I                                      413 422 PRFIDE
I                                      423 423 REUNCH
I                                      424 424 REMCHR
I                                    B 425 428OPAGLEN
I                                    B 429 432OPAGWID
I                                    B 433 436ONMSEP
I                                    B 437 44000VFLN
I                                      441 450 DBCSDA
I                                      451 460 DBCSEX
I                                      461 470 DBCSSO
I                                      471 480 DBCSCR
I                                    B 481 484ODBCSCI
I                                      485 494 GRCHS
I                                      495 504 CODPAG
I                                      505 514 FOMDEF
I                                      515 524 FOMDEL
I                                    B 525 528OSOUDRA
I                                      529 538 PRFONT
I                                      539 544 S36SPL
I                                    B 545 548OPAGROT
I                                    B 549 552OJUST
I                                      553 562 PRTOBT
I                                      563 572 FOLREC
I                                      573 582 CNTCHR
I                                      583 592 ALIFOR
I                                      593 602 PRTQUA
I                                      603 612 FOMFED
I                                      613 683 VOL
I                                      684 700 FILLAB
I                                      701 710 EXCHTP
I                                      711 720 CHRCOD
I                                    B 721 724OTOLREC
I                                    B 725 728OMULTUP
I                                      729 738 FROVRN
I                                      739 748 FROVRL
I                                    P 749 756SFROVRO
I                                    P 757 764SFROVRA
I                                      765 774 BAOVRN
I                                      775 784 BAOVRL
I                                    P 785 792SBAOVRO
I                                    P 793 800SBAOVRA
I                                      801 810 UNTOME
I                                      811 820 PAGDEF
I                                      821 830 PAGDEL
I                                      831 840 LINSPC
I                                    P 841 848SPITSIZ
I                                    P 849 856SFRTMAG
I                                    P 857 864SFRTMAA
I                                    P 865 872SBACMAG
I                                    P 873 880SBACMAA
I                                    P 881 888SLEGOPG
I                                    P 889 896SWIDTH
I                                      897 906 MESMET
I                                      907 907 ADVFUN
I                                      908 917 CHRSET
I                                      918 927 CHRSEL
I                                      928 937 CODPAN
I                                      938 947 CODPAL
I                                      948 957 CODFNT
I                                      958 967 CODFNN
I                                      968 977 DBCSNM
I                                      978 987 DBCSLI
I                                      988 997 USRDEF
I                                     9981007 REDOUT
I                                    10081008 CONST
I                                    B100910120BIN
I                                    B101310160CCSID
I                                     10171116 USRTXT
I                                     11171124 SYSCRT
I                                     11251132 IDCRT
I                                     11331142 USRFIL
I                                     11431144 RESERV
```

```
I                                 B1145114800FFSET
I                                 B1149115200PT
I                                 B1153115600LEN0PT
I                                  11571411 DEFDAT
I                                  14121421 OBJNAM
I                                  14221431 OBJLNM
I                                  14321441 OBJTYP
I                                  14421444 RES
I                                 P1445145250NTSI
I                                 P1453146050FNTPOS
I                                 P1461146850BCSSZ
I                                 B1469147200AUXST0
I                                 B1473147600SPLFSI
I                                 B1477148000SPLFSM
I                                 B1481148400INTPRT
I                                  14851485 SECMET
I                                  14861486 AUTMET
I                                  14871493 DATWTR
I                                  14941499 TIMWTR
I                                  15001506 DCOMP
I                                  15071512 TCOMP
I                                  15131520 JOBSYS
I***
I             'MATSUP    QUSRSYS   'C         OUTQ#
C*****************************************************
C*            M A I N - R O U T I N E               :
C*****************************************************
C*                                                  :
C* ①ユーザー・スペースの作成                        :
C*                                                  :
C                     Z-ADD1024     USSIZE          :
C                     MOVEL'SPLINF' USNAME          :
C                     MOVEL'QTEMP'  USLIB           :
C*                                                  :
C                     CALL 'QUSCRTUS'           95  :
C                     PARM          USRSPC          :
C                     PARM 'SPL'    EXTATR 10       :
C                     PARM          USSIZE          :
C                     PARM X'00'    USINIT  1       :
C                     PARM '*ALL'   USAUTH 10       :
C                     PARM          USTEXT 50       :
C*                                                  :
C* ②ユーザー・スペースにデータを展開                :
C*                                                  :
C                     CALL 'QUSLSPL'            95  :
C                     PARM          USRSPC          :
C                     PARM 'SPLF0100'FMTRCD  8      :
C                     PARM 'USER    'USER   10      :
C                     PARM '*ALL'   OUTQ   20       :
C                     PARM '*ALL'   FRMTYP 10       :
C                     PARM '*ALL'   USRDTA 10       :
C*                                                  :
C           *IN95     CABEQ*ON      $END            :
C*                                                  :
C* ③総称見出しからデータを取り出す                  :
C*                                                  :
C                     CALL 'QUSRTVUS'           95  :
C                     PARM          USRSPC          :
C                     PARM 125      STRPOS          :
C                     PARM 16       LENDTA          :
C                     PARM          GENHED          :
C*                                                  :
C           *IN95     CABEQ*ON      $END            :
C           NOENTH    CABEQ0        $END            :
C*                                                  :
C* ④リストセクション読み込みの初期化                :
C*                                                  :
C                     Z-ADD1        W1CNT   50      :
C*                                                  :
C                     Z-ADDDTL0FS   STRPOS          :
C                     ADD  1        STRPOS          :
C                     Z-ADDDTLSIZ   LENDTA          :
C*                                                  :
C* ⑤リストセクションの各項目取り出し                :
C*                                                  :
B001 C           W1CNT     DOWLENOENTH              :
 001 C*                                             :
 001 C                     CALL 'QUSRTVUS'           95  :
 001 C                     PARM          USRSPC          :
 001 C                     PARM          STRPOS          :
 001 C                     PARM          LENDTA          :
 001 C                     PARM          SF0100          :
 001 C*                                             :
 001 C                     EXSR @ATR                :
     C* ⑥項目の書き出し                                :
     C*           OUTQNM    IFEQ 'FAXQ'
     C                     EXSR @WRITE
     C*                    ENDIF
 001 C*                                             :
 001 C                     ADD  1        W1CNT           :
 001 C                     ADD  DTLSIZ   STRPOS          :
 001 C*                                             :
E001 C                     ENDDO                    :
     C*                                             :
     C           $END      TAG                      :
     C*                                             :
     C                     MOVE *ON      *INLR           :
     C                     RETRN                    :
     C*****************************************************
     C*            S U B  - R O U T I N E
     C*****************************************************
     C*---------------------------------------------------*
     C           @ATR      BEGSR
     C*---------------------------------------------------*
     C                     Z-ADD1520     RCVLEN          :
     C                     MOVE *BLANK   JOBINF 26       :
     C                     MOVE *BLANK   SPLFNM 10       :
     C                     Z-ADD0        SPLF#           :
     C*                                             :
     C                     CALL 'QUSRSPLA'           95  :
     C                     PARM          SA0100          :
     C                     PARM          RCVLEN          :
     C                     PARM 'SPLA0100'FMTATR  8      :
     C                     PARM '*INT'   JOBINF          :
     C                     PARM JOBID    LJOBID          ::
     C                     PARM SPLFID   LSPLID          :
     C                     PARM '*INT'   SPLFNM          :
     C                     PARM          SPLF#           :
     C*
     C* ここで、スプールの各属性が取り込まれる。
     C*                                             :
     C                     ENDSR                    :
     C*---------------------------------------------------*
     C           @WRITE    BEGSR
     C*---------------------------------------------------*
     C                     MOVELBYTESR   SPL001
     C                     MOVELBYTESA   SPL002
     C                     MOVELLJOBID   SPL003
     C                     MOVELLSPLID   SPL004
     C                     MOVELJOBNAM   SPL005
     C                     MOVELUSRNAM   SPL006
     C                     MOVELJOBNUM   SPL007
     C                     MOVELFILNAM   SPL008
     C                     MOVELFILNUM   SPL009
     C                     MOVELFRMTYP   SPL010
     C                     MOVELUSRDTA   SPL011
     C                     MOVELSTAT     SPL012
     C                     MOVELFILAVA   SPL013
     C                     MOVELHOLDF    SPL014
     C                     MOVELSAVF     SPL015
     C                     MOVELTOTALP   SPL016
     C                     MOVELBEGWRT   SPL017
     C                     MOVELSTART    SPL018
     C                     MOVELEND      SPL019
     C                     MOVELLASTPG   SPL020
     C                     MOVELRESTAT   SPL021
     C                     MOVELCOPYS    SPL022
     C                     MOVELPRODUC   SPL023
     C                     MOVELLINPER   SPL024
     C                     MOVELCHRPER   SPL025
```

```
C                   MOVELPRIORT     SPL026
C                   MOVELOUTQNM     SPL027
C                   MOVELOUTQLB     SPL028
C                   MOVELDATFOP     SPL029
C                   MOVELTIME       SPL030
C                   MOVELDVFINM     SPL031
C                   MOVELDVFLNM     SPL032
C                   MOVELPOFINM     SPL033
C                   MOVELPOFLNM     SPL034
C                   MOVELACCCD      SPL035
C                   MOVELTEXT       SPL036
C                   MOVELRECLEN     SPL037
C                   MOVELMAXREC     SPL038
C                   MOVELDEVTYP     SPL039
C                   MOVELPRDVTP     SPL040
C                   MOVELDOCNAM     SPL041
C                   MOVELFLDNAM     SPL042
C                   MOVELS36PNM     SPL043
C                   MOVELPRFIDE     SPL044
C                   MOVELREUNCH     SPL045
C                   MOVELREMCHR     SPL046
C                   MOVELPAGLEN     SPL047
C                   MOVELPAGWID     SPL048
C                   MOVELNMSEP      SPL049
C                   MOVELOVFLN      SPL050
C                   MOVELDBCSDA     SPL051
C                   MOVELDBCSEX     SPL052
C                   MOVELDBCSSO     SPL053
C                   MOVELDBCSCR     SPL054
C                   MOVELDBCSCI     SPL055
C                   MOVELGRCHS      SPL056
C                   MOVELCODPAG     SPL057
C                   MOVELFOMDEF     SPL058
C                   MOVELFOMDEL     SPL059
C                   MOVELSOUDRA     SPL060
C                   MOVELPRFONT     SPL061
C                   MOVELS36SPL     SPL062
C                   MOVELPAGROT     SPL063
C                   MOVELJUST       SPL064
C                   MOVELPRTOBT     SPL065
C                   MOVELFOLREC     SPL066
C                   MOVELCNTCHR     SPL067
C                   MOVELALIFOR     SPL068
C                   MOVELPRTQUA     SPL069
C                   MOVELFOMFED     SPL070
C                   MOVELVOL        SPL071
C                   MOVELFILLAB     SPL072
C                   MOVELEXCHTP     SPL073
C                   MOVELCHRCOD     SPL074
C                   MOVELTOLREC     SPL075
C                   MOVELMULTUP     SPL076
C                   MOVELFROVRN     SPL077
C                   MOVELFROVRL     SPL078
C                   Z-ADDFROVRO     SPL079
C                   Z-ADDFROVRA     SPL080
C                   MOVELBAOVRN     SPL081
C                   MOVELBAOVRL     SPL082
C                   Z-ADDBAOVRO     SPL083
C                   Z-ADDBAOVRA     SPL084
C                   MOVELUNTOME     SPL085
C                   MOVELPAGDEF     SPL086
C                   MOVELPAGDEL     SPL087
C                   MOVELLINSPC     SPL088
C                   Z-ADDPITSIZ     SPL089
C                   Z-ADDFRTMAG     SPL090
C                   Z-ADDFRTMAA     SPL091
C                   Z-ADDBACMAG     SPL092
C                   Z-ADDBACMAA     SPL093
C                   Z-ADDLEGOPG     SPL094
C                   Z-ADDWIDTH      SPL095
C                   MOVELMESMET     SPL096
C                   MOVELADVFUN     SPL097
C                   MOVELCHRSET     SPL098
C                   MOVELCHRSEL     SPL099
C                   MOVELCODPAN     SPL100
C                   MOVELCODPAL     SPL101
C                   MOVELCODFNT     SPL102
C                   MOVELCODFNN     SPL103
C                   MOVELDBCSNM     SPL104
C                   MOVELDBCSLI     SPL105
C                   MOVELUSRDEF     SPL106
C                   MOVELREDOUT     SPL107
C                   MOVELCONST      SPL108
C                   MOVELBIN        SPL109
C                   MOVELCCSID      SPL110
C                   MOVELUSRTXT     SPL111
C                   MOVELSYSCRT     SPL112
C                   MOVELIDCRT      SPL113
C                   MOVELUSRFIL     SPL114
C                   MOVELRESERV     SPL115
C                   MOVELOFFSET     SPL116
C                   MOVELOPT        SPL117
C                   MOVELLENOPT     SPL118
C                   MOVELDEFDAT     SPL119
C                   MOVELOBJNAM     SPL120
C                   MOVELOBJLNM     SPL121
C                   MOVELOBJTYP     SPL122
C                   MOVELRES        SPL123
C                   Z-ADDPONTSI     SPL124
C                   Z-ADDFNTPOS     SPL125
C                   Z-ADDDBCSSZ     SPL126
C                   MOVELAUXSTO     SPL127
C                   MOVELSPLFSI     SPL128
C                   MOVELSPLFSM     SPL129
C                   MOVELINTPRT     SPL130
C                   MOVELSECMET     SPL131
C                   MOVELAUTMET     SPL132
C                   MOVELDATWTR     SPL133
C                   MOVELTIMWTR     SPL134
C                   MOVELDCOMP      SPL135
C                   MOVELTCOMP      SPL136
C                   MOVELJOBSYS     SPL137
C                   WRITEUSRSPLR
C*
C                   ENDSR                   :
```

図4　総称見出しのレイアウト

```
  0 CHAR(64)  ユーザー域
 64 BINARY(4) 総称見出しのサイズ
 68 CHAR(4)   構造のリリースおよびレベル
 72 CHAR(8)   形式名
 80 CHAR(10)  使用された API
 90 CHAR(13)  作成された日時
103 CHAR(1)   情報の状況
104 BINARY(4) 使用されるユーザー空間のサイズ
108 BINARY(4) 入力パラメーター・セクションへのオフセット
112 BINARY(4) 入力パラメーター・セクションのサイズ
116 BINARY(4) 見出しセクションのオフセット
120 BINARY(4) 見出しセクションのサイズ
124 BINARY(4) リスト・データ・セクションのオフセット
128 BINARY(4) リスト・データ・セクションのサイズ
132 BINARY(4) リスト項目の数
136 BINARY(4) 各項目のサイズ
140 BINARY(4) リスト項目のデータの CCSID
144 CHAR(2)   国別 ID
147 CHAR(3)   言語 ID
149 CHAR(1)   サブセットされたリスト標識
150 CHAR(42)  予約済み
```

図5　出力待ち行列処理実行プログラム

```
PGM         PARM(       +
                &PRMKBN +
                &PRMSPF +
                &PRMJOB +
                &PRMUSR +
                &PRMJNO +
                &PRMSFN )

              /*  変数定義 */
            DCL  VAR(&PRMKBN) TYPE(*CHAR) LEN(1)   /*  処理区分         */
              /* ３＝保留　４＝削除　５＝表示　６＝送信 */
            DCL  VAR(&PRMSPF) TYPE(*CHAR) LEN(10)  /*  ＳＰＬＦ名       */
            DCL  VAR(&PRMJOB) TYPE(*CHAR) LEN(10)  /*  ＪＯＢ名         */
            DCL  VAR(&PRMUSR) TYPE(*CHAR) LEN(10)  /*  ユーザー名       */
            DCL  VAR(&PRMJNO) TYPE(*CHAR) LEN(6)   /*  番号             */
            DCL  VAR(&PRMSFN) TYPE(*DEC)  LEN(6 0) /*  ＳＰＬＦ番号     */
            DCL  VAR(&PRMJBN) TYPE(*CHAR) LEN(8)   /*  ＪＯＢシステム名 */

            /* 保留 */
                 IF         COND(&PRMKBN *EQ '3')  +
                    THEN(DO)
                     HLDSPLF FILE(&PRMSPF) +
                     JOB(&PRMJNO/&PRMUSR/&PRMJOB) SPLNBR(&PRMSFN)
                     MONMSG MSGID(CPF0000)
                    ENDDO

            /* 削除 */
                 IF         COND(&PRMKBN *EQ '4')  +
                    THEN(DO)
                     DLTSPLF FILE(&PRMSPF) +
                     JOB(&PRMJNO/&PRMUSR/&PRMJOB) SPLNBR(&PRMSFN)
                     MONMSG MSGID(CPF0000)
                    ENDDO

            /* 表示 */
                 IF         COND(&PRMKBN *EQ '5')  +
                    THEN(DO)
                   CPYSPLF    FILE(&PRMSPF) TOFILE(*LIBL/WFR10HPF) +
                                JOB(&PRMJNO/&PRMUSR/&PRMJOB) +
                                SPLNBR(&PRMSFN) TOMBR(&PRMMBR)
                     MONMSG MSGID(CPF0000)
                    ENDDO

            /* 送信 */
                 IF         COND(&PRMKBN *EQ '6')  +
                    THEN(DO)
                     RLSSPLF FILE(&PRMSPF) +
                     JOB(&PRMJNO/&PRMUSR/&PRMJOB) SPLNBR(&PRMSFN)
                     MONMSG MSGID(CPF0000)
                    ENDDO

ENDPGM
```

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】

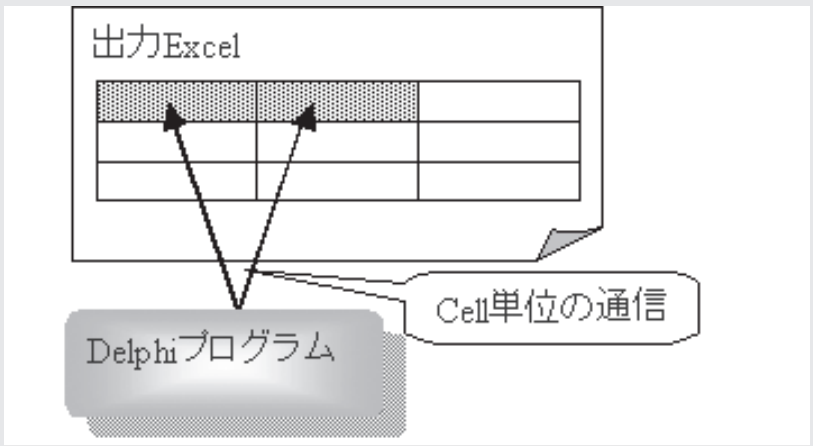
一括で Excel へ書き込むために、文字列を使用している。フィールドごとに #9、改行ごとに #13#10 のリテラルを挟んで編集することで、複数行の内容も文字列で格納することができる。

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。

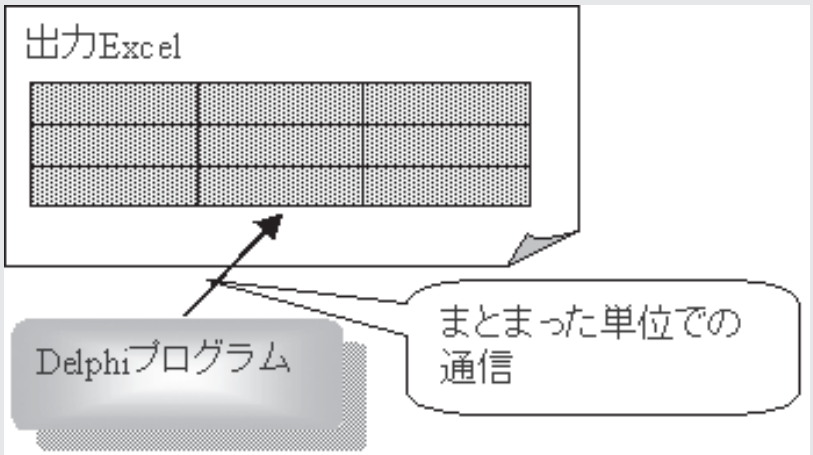
これらによって、Delphi ⇔ Excel 間で一番大きい処理時間を短縮できることになる。

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。

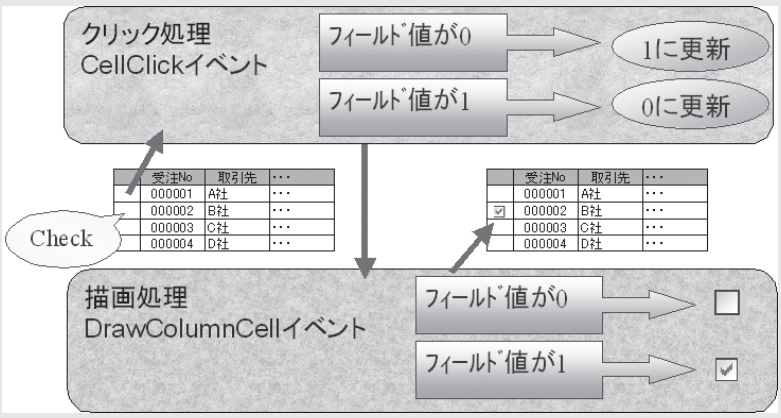
また、この仕組みを実現するためのDBGrid のクリックイベントと描画イベントのソース例を示す。【ソース 1】【ソース 2】

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;
```

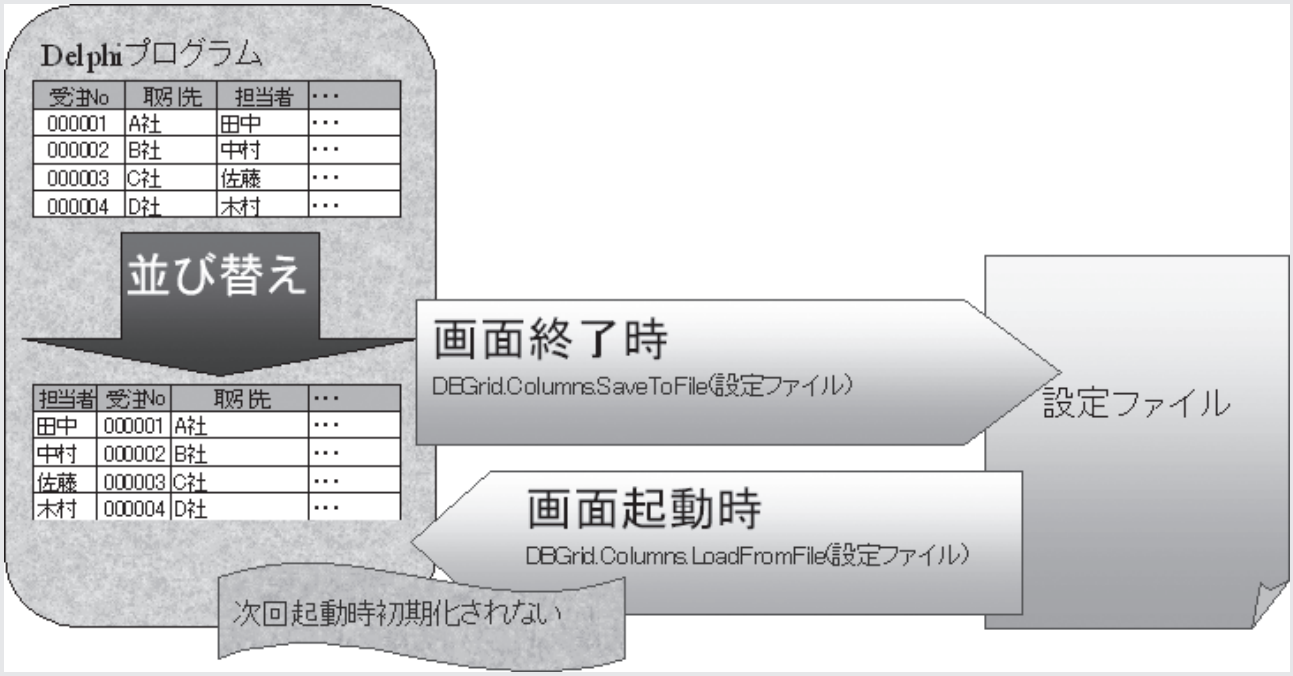
3.DBGridの表示状態の保存

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】

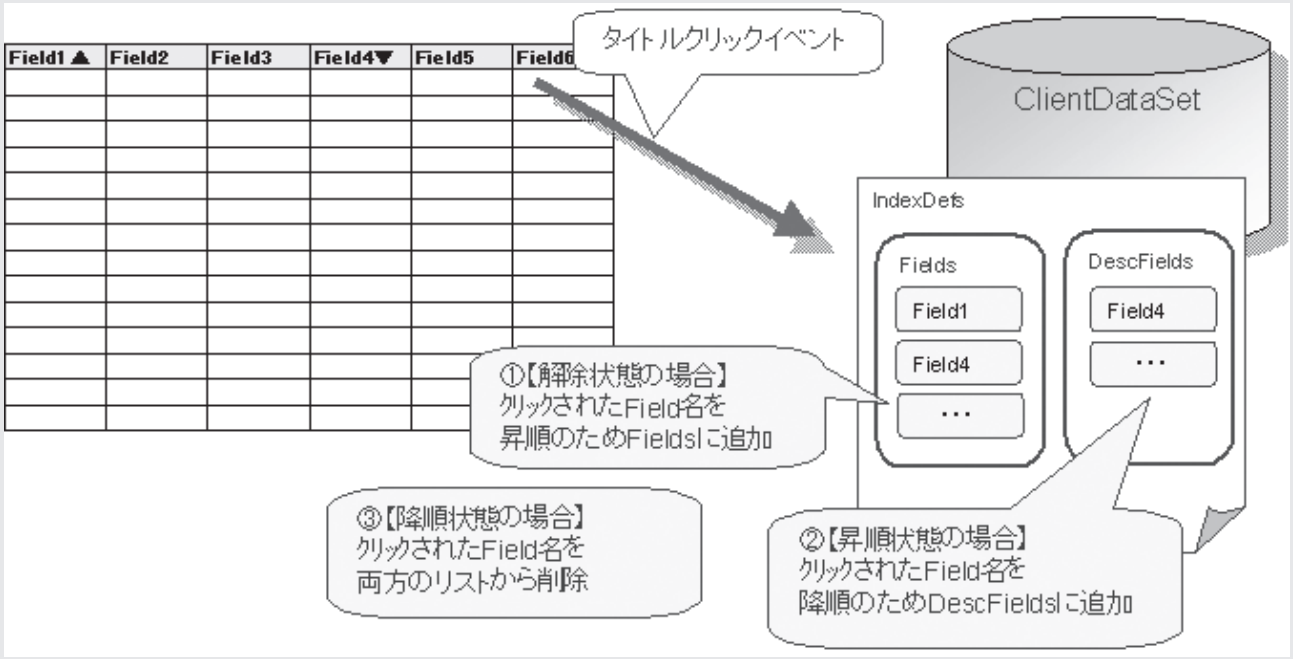
・Index の削除と再追加

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？

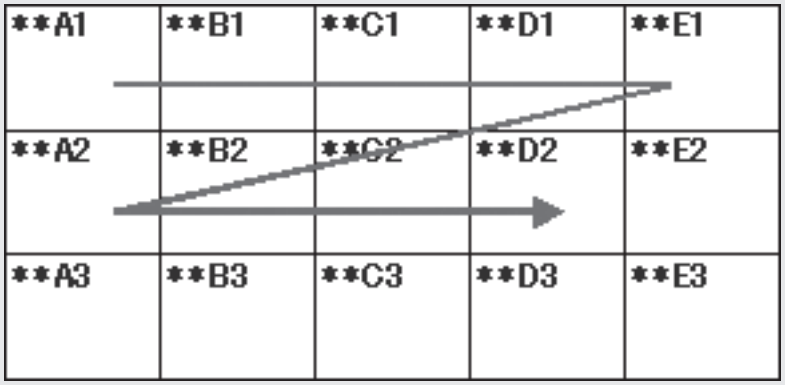
A. VB-Report Ver3.0 で Excel を扱う場合、出力パフォーマンスにはフォーマットとなる Excel 自体が大きく影響する。

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

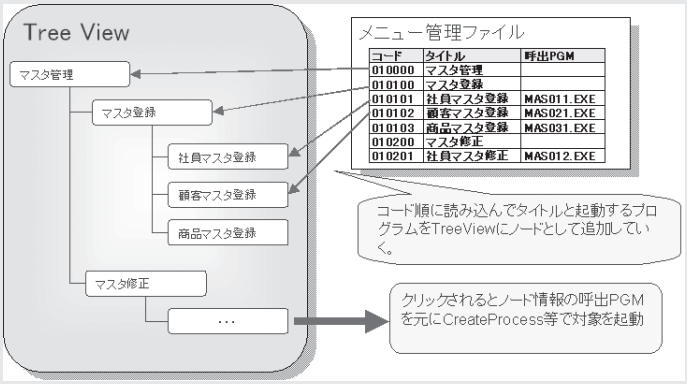
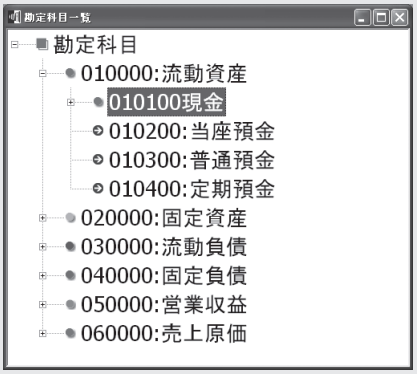


図8　勘定科目一覧の例



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。

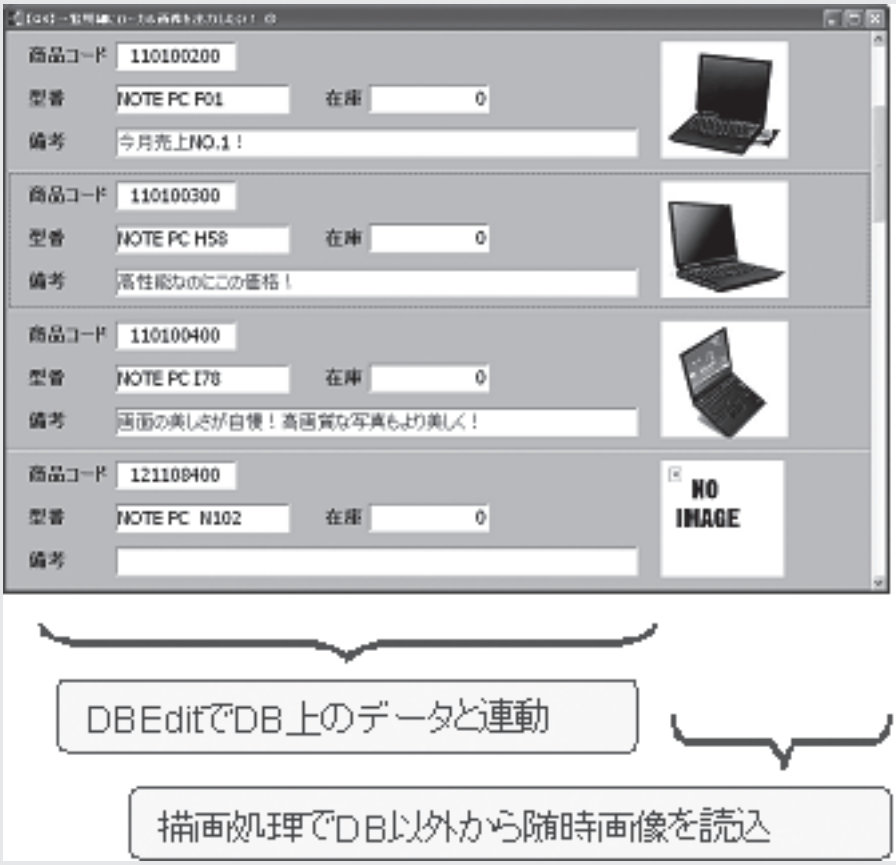
DBCtrlGrid上で一覧明細として表示したい場合は、描画を行うOnPaintPanelイベントを利用することで実現が可能である。【図9】

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

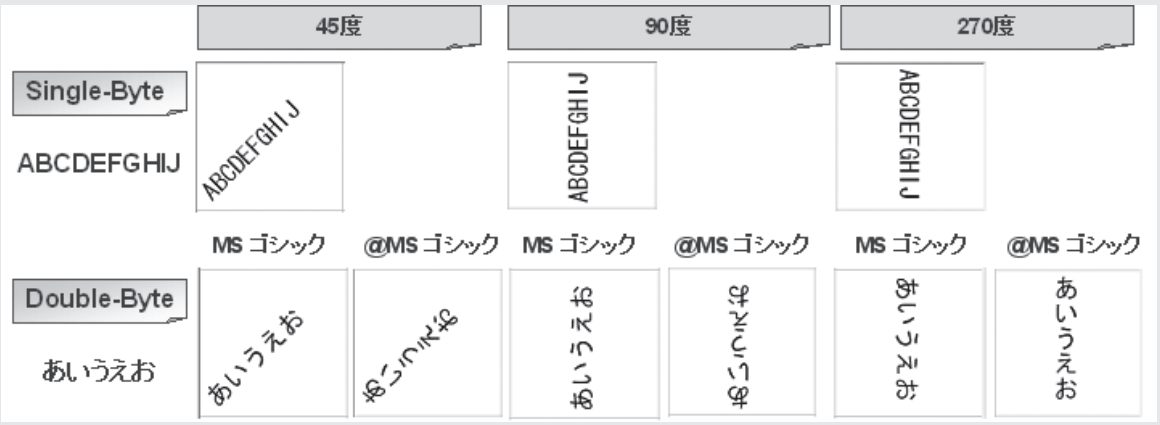
注意点としては、使用するフォントに縦書き用のフォントを使用した場合は、Single-Byte 文字と Double-Byte 文字とで回転方向が異なる点である。

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



松尾 悦郎

株式会社ミガロ

システム事業部 システム2課

JACi400を使って RPGでWeb画面を制御する方法

**RPG 開発者のために
RPG プログラムで、Web 画面を制御する方法を紹介する。
ラジオボタンについてと
画面遷移時の値の受け渡し方法を説明したい。**

略歴
1979 年 6 月 16 日 生
2002 年広島大学 理学部卒
2006 年株式会社ミガロ入社
2006 年 6 月システム事業部配属

現在の仕事内容
SE。主に JACi400 を使った Web アプリケーションの開発を担当しており、システムの要件定義から納品・フォローまで行っている。

はじめに

昨今の世界的なインターネットの普及により、アプリケーションの Web 化はますます進んでいる。IBM i で稼働するアプリケーションも例外ではなく、Web 化のニーズが高まってきている。

これまで RPG プログラムをメインに、IBM i 上で稼働するアプリケーションの開発を行ってきた私にとっても、Web 化は避けては通れないものとなってきた。Java などの RPG 以外の言語を習得する必要性を感じているが、敷居が高くなかなか思うようにはいかず、少なからず歯がゆい思いもしていた。

JACi400

そんなときに出会ったのが、RPG プログラムで、Web アプリケーションを開発できる「JACi400」である。

Web アプリケーションを開発するには、Java などの言語を覚えないといけないと思っていたが、なんと JACi400 では、これまで私が身につけてきた知識やノウハウをそのまま活かして、RPG で簡単に Web アプリケーションを開発できるのである。

しかも、HTML 画面との連携は、予め作成される RPG ソースに記述される。私たちは、ただデータの抽出やデータベースの更新を RPG ソースに記述すればよいし、DSPF を制御するように、HTML 画面も RPG プログラムで制御できる。従来の開発と同じように開発が可能なのである。

しかし、画面が DSPF から HTML になることで、画面で使用する部品が違うため、若干制御の方法を覚える必要がある。例えば、コンボボックスやラジオボタンがそうだ。これらは、いずれも Web 画面では必須の部品と言っても過言ではないだろう。

また、画面の遷移も異なってくる。こちらは、例えば、DSPF であれば 1 つのプログラムに複数の画面を持てるが、HTML だと 1 つしか持てない、といったことである。

今回は、RPG 開発者のために、RPG プログラムで Web 画面を制御する方法のうち、ラジオボタンと、画面遷移時の値の受け渡し方法を紹介したいと思う。

ラジオボタン

ラジオボタンは、Web 画面ではよく使用される部品である。いくつかの選択肢から 1 つを選ぶものであるが、これを RPG プログラムで制御できるのだ。

①画面で選択したものを、RPG プログラムで受け取る方法

画面で選択した値を RPG に受け取るにはどうしたらいいのだろうか。

まずは、HTML にラジオボタンを設定し、それぞれの項目に対して id 属性と、value 属性を定義する。id 属性は全て同じ値を指定するが（異なる値を指定すると正しく動作しないので注意）、value 属性はそれぞれに異なる値を指定する必要がある。

ここでは、東京・名古屋・大阪をラジオボタンで選択するものを作ろう。【図 1】

図1

図2

```
<td bgcolor="#ffffff" align="center">
 <div class="BLU11">
  <p><br>
   ++ ラジオボタングループ ++
    <br>
    <br>
  </p>
 </div>
    <p>
     <label>
     <input name="RBG01" type="radio" value="R1" checked id="RBG01">東京</label>
     <label>
     <input type="radio" name="RBG01" value="R2" id="RBG01">名古屋</label>
     <label>
     <input type="radio" name="RBG01" value="R3" id="RBG01">大阪</label>
    </p>
```

IRBG01には、画面で選択された値のHTMLで指定した"value値"がセットされます。
東京＝R1
名古屋＝R2
大阪＝R3

図3

ラジオボタンで選択された値により、処理をコーディングするRPGは以下のようになります。

```
0092.00 C050  * <YOURCODE>
0093.00 --->  * CHECK ACTION CODE JCACTN HERE AND PROCESS.
0094.00       C           JCACTN    IFEQ 'ED'
0095.00       C                     GOTO ENDPGM
0096.00       C                     ENDIF
0097.00       C**
0098.00       C           JCACTN    IFEQ 'ET'
0098.01       C*+ ラジオボタン
0098.02       C*++ 東京が選択されたときの処理
0099.00       C           IRBG01    IFEQ 'R1'
0100.00       C                     ENDIF
0100.01       C*++ 名古屋が選択されたときの処理
0100.02       C           IRBG01    IFEQ 'R2'
0100.03       C                     ENDIF
0100.04       C*++ 大阪が選択されたときの処理
0100.05       C           IRBG01    IFEQ 'R3'
0100.06       C                     ENDIF
0100.07       C*+
0100.08       C                     ENDIF
0101.00       C**
0102.00       C                     GOTO T100
0103.00 --->  * YOUR CODE
0104.00 C050  * </YOURCODE>
```

id 属性は全て“RBG01”と指定する。value 属性は、東京“R1”、名古屋“R2”、大阪“R3”とそれぞれ指定する。【図 2】

これで、画面からいずれかの都市を選択すると、id = RBG01 に HTML で指定したそれぞれの value 値がセットされるようになる。例えば、画面で東京を選択した場合は、“R1”が RPG プログラムに渡される。

HTML の設定が正しくでき、さらに画面で実行処理を行うと、JACi400 が RPG プログラムを呼び出し、RPG プログラムで id = RBG01 の値を受け取ることができるのだ。あとは、それぞれの値別に、処理を記述すればよい。【図 3】

②ラジオボタンで選択した値を保持する方法

HTML を触ったことがある人ならば、ラジオボタンの初期値は、HTML ソースで“checked”を指定した項目になることを知っているだろう。そのため、画面で値を選択して実行したあと、次に画面表示するときには選択した値を保持せずに、常に初期値として指定した項目に戻ってしまう。

これでは都合が悪い。照会画面では自分が指定した内容は保持していたいし、前の画面に戻ったときも選択した値を保持していたいものだ。

ではどうするか。実はこれも RPG プログラムで対応できるのだ。選択時に受け取った値を、そのまま次の画面にも送ってあげればよい。

画面に送るための手順

それでは、画面に送るための手順を順に説明しよう。

・JACi400Designer での設定

最初に、JACi400Designer で設定を行う。

ラジオボタン id = RBG01 の設定で、Usage 項目を“Both”にし、Additional 項目にラジオボタンの項目数・value 属性の長さ・画面に表示する記述の長さを指定する。これで、ラジオボタンが入出力フィールドとして定義される。

ここでは、ラジオボタンの項目数を“3”・value 属性の長さを“2”・画面に表示する記述の長さを“10”に設定している。

・生成される DDS

JACi400Designer で各フィールドを設定したあと、JACi400 で DDS を作成する。

今回の例では、図 4 のような、INPUT と OUTPUT タイプの DDS が作成される。INPUT タイプは、画面から選択された値を受け取るためのものである。OUTPUT タイプのフィールド O01002 は、ラジオボタンの項目数を指定する。フィールド ORBBG01 には、選択フラグ・value 属性の値・画面に表示する記述内容を、ラジオボタンの項目数だけ指定する。【図 4】

・RPG プログラムのコーディング

DDS が作成されると、いよいよ RPG プログラムのコーディングをしなければならない。

まずは各定義を行う。ラジオボタン情報用の配列を宣言し、同じくラジオボタン情報用の DATA STRUCTURE を定義する。配列のレコード長は 39 バイトである。これは、選択フラグ 1 バイト・value 属性の値 2 バイト・画面に表示する記述内容の 10 バイトをあわせた 13 バイトに対して、ラジオボタンの項目数 3 つ分の合計値である。

また、DATA STRUCTURE は、各項目別に選択フラグ・value 値・表示する記述内容を定義しておく。そして、配列の要素も“1R1 東京 0R2 名古屋 0R3 大阪”と内部で定義しておく。

データセットの処理

以上の定義ができたら、次にデータセットの処理を行う。

まず、配列の要素“1R1 東京 0R2 名古屋 0R3 大阪”を DATA STRUCTURE にセットし、ラジオボタンの項目数“3”をフィールド O01002 にセットする。

このまま処理を終了すると、画面にはラジオボタンが表示され、“東京”にチェックが入った状態になる。しかしこれだと、常に画面のラジオボタンは東京が選択された状態になってしまう。

では、名古屋や大阪を選択状態にするにはどうしたらよいか、それは、画面で選択状態にする項目の選択フラグに“1”を、その他には“0”をセットすればよい。つまり、名古屋であれば、DATA STRUCTURE で定義した RD21 に“1”をセットし、その他の RD11 と RD31 に“0”をセットするのである。同様に、大阪を選択状態にするには、RD31 に“1”をセットすればよい。

ここまでできたら、最後に DATA STRUCTURE の値をフィールド ORBG01 に転送して RPG から画面にデータを送信して完了となる。【図 5】

これで、ラジオボタンで選択した値を、次回画面表示時に保持することができる。

実は、これと同じ方法でコンボボックスも制御でき、とても簡単である。ぜひ覚えていただきたい。

第2画面で選択した値を、第1画面に戻す

ここから画面遷移の方法を紹介しよう。

RPG 開発者にとっては、Web アプリケーションの画面遷移も、どのようにすればよいのかピンと来ないのではないだろうか。なぜなら、Web アプリケーションでは多くの場合、画面処理ごとに通信は切断されるからである。操作するユーザーからの入力を待ち続けているわけではない。

一方、従来の 5250 では、RPG プログラムが次の処理を待機している。それゆえ、制御方法が違い、どのようにしたらよいかわからないのである。

JACi400の場合

JACi400 の場合は、メニューから画面が呼び出されると、JACI400 サブシステム配下にジョブが作成される。HTML を出力しても、CGI は終了するが、RPG プログラムは待機した状態になるのである。つまり、従来どおりの手法で開発ができるのだ。

また、アプリケーション構成は、HTML 画面 1 つに対して、RPG プログラム 1 つである。したがって、ある画面から別の画面に遷移するときは、もとの RPG プログラムから別の画面用の RPG プログラムを実行することになる。画面間でデータを受け渡す場合には、何らかの手段で受け渡ししなければならない。

とはいえ、これまでの 5250 での開発と比べても、特別な手法ではないので安心していただきたい。

図4

```
JCTST301I
       ********** データの始め **********
0001.00       * HTML FILE: JCTST3.HTML
0002.00      A          R JCTST3
0003.00      A            IRBG01     00002
       ********** データの終わり **********
```

```
JCTST301O
       ********** データの始め **********
0001.00       * HTML FILE: JCTST3.HTML
0002.00      A          R JCTST3
0003.00      A            O01002         4S 0       TEXT('NUMBER OF VALUES ORBG
0004.00      A            ORBG01     00039
       ********** データの終わり **********
```

図5

```
Radioボタン情報用の配列を宣言
0018.00 E010  * <YOURCODE>
0019.00 --->  * YOUR ARRAYS
0019.01      E                    @RD     1   1 39
0020.00 E010  * </YOURCODE>

Radioボタン情報用のDATA STRUCTURES
0025.00       * DATA STRUCTURES
0026.00 I010  * <YOURCODE>
0027.00 --->  * YOUR INPUT SPECIFICATIONS
0027.01      I            DS
0027.02      I                                        1  39 RD
0027.03      I                                        1   1 RD11
0027.04      I                                        2   3 RD12
0027.05      I                                        4  13 RD13
0027.06      I                                       14  14 RD21
0027.07      I                                       15  16 RD22
0027.08      I                                       17  26 RD23
0027.09      I                                       27  27 RD31
0027.10      I                                       28  29 RD32
0027.11      I                                       30  39 RD33

0093.00 C050  * <YOURCODE>
0094.00 --->  * CHECK ACTION CODE JCACTN HERE AND PROCESS.
0095.00      C           JCACTN    IFEQ 'ED'
0096.00      C                     GOTO ENDPGM
0097.00      C                     ENDIF
0098.00      C**
0099.00      C           JCACTN    IFEQ 'ET'
0099.01      C*+  ラジオボタンワークリセット
0099.02      C                     CLEARRD
0099.03      C*+  ラジオボタン項目とアイテム数セット
0099.04      C                     MOVEL@RD,1     RD
0099.05      C                     Z-ADD3         O01002
0099.07      C*+  ラジオボタン
0099.08      C*++  東京が選択されたときの処理
0099.09      C           IRBG01    IFEQ 'R1'
0099.10      C*++ ラジオボタン選択フラグに１えおセット
0099.11      C                     MOVEL'1'       RD11
0099.12      C                     ENDIF
0099.13      C*++  名古屋が選択されたときの処理
0099.14      C           IRBG01    IFEQ 'R2'
0099.15      C*++ ラジオボタン選択フラグに１えおセット
0099.16      C                     MOVEL'1'       RD21
0099.18      C                     ENDIF
0099.19      C*++  大阪が選択されたときの処理
0099.20      C           IRBG01    IFEQ 'R3'
0099.21      C*++ ラジオボタン選択フラグに１えおセット
0099.22      C                     MOVEL'1'       RD31
0099.24      C                     ENDIF
0099.25      C*++  ラジオボタン情報を画面上フィールドに返す
0099.26      C                     MOVELRD        ORBG01
0099.27      C*+
0101.00      C                     ENDIF
0102.00      C**
0103.00      C                     GOTO T100
0104.00 --->  * YOUR CODE
0105.00 C050  * </YOURCODE>

0245.00 C510  * <YOURCODE>
0246.00       *---------------------------------------------------------------
0247.00       * YOUR INITIALIZATION
0248.00       *---------------------------------------------------------------
0249.00      C           YRINIT    BEGSR
0249.01      C                     Z-ADD3         O01002
0249.02      C                     MOVEL@RD,1     ORBG01
0250.00      C                     ENDSR

@RD配列要素
0285.00 **
0286.00 1R1 東京     0R2 名古屋   0R3 大阪
```

手順③の構成に合わせ
DATA STRUCTURESを定義

① 選択フラグ

② Radio値・・・
HTML上の
Value値を
セット

③ Radio記述

画面遷移時の値の受け渡し方法

では、ここで第1画面から第2画面を呼び出し、第2画面の値を第1画面に返し、その値を第1画面で表示する例を紹介しよう。

・処理の流れ

まず処理の流れだが、第1画面と第2画面のRPGプログラムが別々であるため、第1画面から第2画面を呼び出す。呼び出された第2画面は、名称の一覧を表示する画面である。その一覧の中でいずれか1つにチェックを入れ実行すると、第2画面を終了し、第1画面の名称欄に第2画面で選択した名称をセットするものである。

これら全てを、RPGプログラムで処理を行う。

・RPGプログラムのコーディング

冒頭にも述べ、何度も繰り返すが、JACi400は画面1つに対してRPGプログラム1つである。

ここでは、第1画面のRPGプログラムを“JCTST1A”とし、第2画面のRPGプログラムを“JCTST1B”とする。

まず、第1画面のJCTST1Aでは、第2画面のJCTST1Bを、CALL命令で呼び出すようにする。このときJCTST1Bからの戻り値として、名称をパラメータ設定しておく。【図6】

次に、第2画面のJCTST1Bだが、選択した名称を戻りパラメータにセットして､プログラムを終了するだけでよい。

あとは、第1画面のプログラムJCTST1Aに制御が戻り、名称をパラメータで受け取って、その名称をフィールドにセットする。これで、第2画面で選択した名称が、第1画面で表示されるのである。

このように、第1画面から第2画面に遷移するときは、第1画面からCALL命令を実行し、値を引き継ぐときは、パラメータを設定して受け渡しを行えばよいのである。

いかがだろうか。この内容も、RPGプログラムを開発したことのある人にとっては、それほど難しいことではないはずだ。

最後に

今回いくつかRPGのコーディング例を紹介した。どうであっただろうか。どれも難しい内容ではないので、RPGプログラムの開発経験がある人ならば、すぐに理解できる内容であったと思う。

私がそうであったように、これまでWebアプリケーション開発に壁を感じていたRPGプログラムの開発者にとっては、少しはWebアプリケーションを身近に感じられるようになったのではないだろうか。

それでも、今回紹介した内容は、JACi400を使ってRPGプログラムでできる項目の一部に過ぎない。まずは、実際にJACi400を使って、その性能を実感していただきたいと思う。そして、これまでに培った知識とノウハウを十分に活用できるJACi400で、押し寄せてくるWeb化の波に対応してほしい。

M

現在の仕事内容（詳細）
SE。主にJACi400を使ったWebアプリケーションの開発を担当しており、システムの要件定義から納品・フォローまで行っている。
以前はRPGプログラムの開発に携わっていたこともあり、JACi400の開発フェーズでは、RPGプログラム開発の管理を行っている。また、HAツールである*noMAXの技術サポートも担当している。

図6

JCTST1A

```
0098.00       * GET YOUR DATA
0099.00      C                    EXSR GTDATA
0100.00 C060  * <YOURCODE>
0101.00 --->  * YOUR CODE
0102.00      C*++ ENTER PUSH
0103.00      C           JCACTN   IFEQ '1 '
0104.00      C                    CALL 'JCTST1B'
0105.00      C                    PARM           PRMOUT 32･･･①
0106.00      C                    MOVELPRMOUT    OEDT01   ･･･④
0107.00      C                    MOVELPRMOUT    IEDT01
0108.00      C                    GOTO T200
0109.00      C                    ENDIF
```

```
0018.00 E010  * <YOURCODE>
0019.00 --->  * YOUR ARRAYS
0020.00      E                   @CK         5  1          ･･･選択
0021.00      E                   @BA         5 10          ･･･コード
0022.00      E                   @BB         5 32          ･･･名称
0023.00 E010  * </YOURCODE>


0030.00 I010  * <YOURCODE>
0031.00 --->  * YOUR INPUT SPECIFICATIONS
0032.00      I            DS
0033.00      I                                            1   5 @CK
0034.00      I                                            1   1 CHK01
0035.00      I                                            2   2 CHK02
0036.00      I                                            3   3 CHK03
0037.00      I                                            4   4 CHK04
0038.00      I                                            5   5 CHK05
0039.00      I            DS
0040.00      I                                            1  50 @BA
0041.00      I                                            1  10 LBA01
0042.00      I                                           11  20 LBA02
0043.00      I                                           21  30 LBA03
0044.00      I                                           31  40 LBA04
0045.00      I                                           41  50 LBA05
0046.00      I            DS
0047.00      I                                            1 160 @BB
0048.00      I                                            1  32 LBB01
0049.00      I                                           33  64 LBB02
0050.00      I                                           65  96 LBB03
0051.00      I                                           97 128 LBB04
0052.00      I                                          129 160 LBB05
0053.00 I010  * </YOURCODE>

0089.00 C010  * <YOURCODE>
0090.00 --->  * YOUR CODE
0091.00      C           *ENTRY   PLIST
0092.00      C                    PARM           PRMOUT 32･･･②
0093.00 C010  * </YOURCODE>
                                   ↑
                                   ↓
0308.00      C           GTDATA   BEGSR
0326.00      C*+                                          ･･･③
0327.00      C           1        DO   5         J       20
0328.00      C           @CK,J    IFEQ '1'
0329.00      C                    MOVEL@BB,J     PRMOUT
0330.00      C                    SETON                    10
0331.00      C                    ENDIF
0332.00      C                    ENDDO
0333.00      C                    ENDSR
```

画面上で選択にチェックが入れられた場合、チェックされたところによりCHK01～05のいずれかに"1"が戻ってくる。

画面表示時、プログラム内でコードを画面のアウトプット用フィールドにセットする。

画面表示時、プログラム内でコードと対となる名称を画面のアウトプット用フィールドにセットする。

福井 和彦

株式会社ミガロ

システム事業部 システム1課

あなたはブラインドタッチができますか？

略歴
1972 年 3 月 20 日 生
1994 年大阪電気通信大学 工学部卒
2001 年 4 月株式会社ミガロ入社
2001 年 4 月システム事業部配属

現在の仕事内容
主に Delphi/400 を使用したシステムの
受託開発を担当しており、
要件確認から納品・フォローに至るまで、
システム開発全般に携わっている。
また、Delphi/400 の導入支援や
セミナーの講師も行っている。

突然ですが、皆さんは「ブラインドタッチ」はできますか？

このようなところで堂々と言えることではありませんが、しかも、私はこの業界に入って 14 年以上になるのですが、現在に至るまで「ブラインドタッチ」をマスターしようと練習をしてきませんでした。だからという訳でもないでしょうが、私はキーボード入力があまり得意ではありません。そんな私だからこそ思いついたであろう今回の話題をここから述べます。私と同じような経験をされた方には、共感してもらえる何かがあるのではないかと思います。

最近ではプログラムを書く機会が減りましたが、プログラムを頻繁に書いていた頃は「いかにキーボード入力が少なく、プログラムが書けるか」ということを考え、「コピー＆ペースト」や「文字の置き換え」等いろいろな方法を駆使していました。もちろん、プログラムを作成する時間の割合は、大半が考えている時間であり、書いている時間の占める割合はそんなに多くはないと思います。しかし、このプログラムを書いていく中で一番面倒だったのが項目転送の文です。

例えば Delphi で、ファイルから画面の項目に値を転送する場合、次のように記述します。

```
"edtWKTOCD.Text := Table1.FieldByName
('WKTOCD').AsString;"
```

このような転送文を画面の項目の数だけ書かなければならず、更新がある場合には、画面からファイルへの逆方向の転送文も必要になってきます。これがけっこう面倒な作業なのです。どちらかと言えば単純作業で、転送の数が多くなればなるほど時間もかかり、間違える可能性も高くなってきます。ブラインドタッチに自信のある方でも、この転送文を 1 文字 1 文字キーボード入力で書く方は少ないと思います。

今回の話題は、「Delphi/400 の転送文を書く」ということにスポットを当て、キーボード入力の回数を少なくかつより早く正確に作れるか、ということを考えて、創意工夫をしてきたある開発者の話です。

ただし、これから出てくる各手法については、弊社のコーディングルールを前提にしているところが多々ありますので、あらかじめご了承いただきたいと思います。

では、興味をもたれた方はこの先を読み進めていただければ幸いです。

先程の転送文を書く場合、次に書く方法を使われている方が多いのではないかと思います。

転送文の書き方

① 1 行目の転送文を書く
② 1 行目の文を必要な転送項目分コピーする
③ 2 行目以降の、左辺のコンポーネント名、右辺の "FieldByName" のフィールド名と "As ～ " の型指定を変更する

私の場合、③のときにエミュレーター画面を開き、"DSPFMT" コマンドを使ってファイルレイアウトを表示します。次に、表示したファイルレイアウトの項目名をコピーして、Delphi 画面に切り替えコンポーネント名とフィールド名にそれぞれ貼り付けます。こうすることで、キーボードの入力が「Ctrl + "C"」「Ctrl + "V"」「Alt + Tab」「Shift + 矢印キー」に絞られます。ここで使用するアルファベットは "C" と "V" だけなので、ブラインドタッチが苦手な私でも容易に操作をすることができるのです。キーボード操作の流れは以下のようになります。

キーボード操作の流れ

① エミュレーター画面のコピーする項目名を「Shift ＋矢印キー」で範囲選択する
②「Ctrl ＋ "C"」で項目名コピーする
③「Alt ＋ Tab」で Delphi 画面に切り替える
④ 貼付先を「Shift ＋矢印キー」で範囲選択する（"FieldByName" のフィールド名はマウスでダブルクリックしても楽に選択できる）
⑤「Ctrl ＋ "V"」で貼り付ける
⑥「Alt ＋ Tab」でエミュレーター画面に切り替える

⑦ ①から繰り返す

また、1行目に入力する転送文を、次のようにフィールド名にあたる部分を除いて書きます。

"edt.Text := Table1.FieldByName('').AsString;"

そして必要な行数分をコピーしておくと、④で「Shift＋矢印キー」を使用することなく、"edt" の後ろと "'" の間にフォーカスを移動して貼り付けるだけなので、さらにキーボードの入力回数を減らすことができます。

最初はキーボードの操作もぎこちなくペースが上がりにくいのですが、一度リズムを掴むと、「トン、トン、トン、トン…」とリズムよく書いていくことができます。(このときばかりは誰にも声をかけられたくないものです。また、メールソフトを開いたまま作業をしていると、メール着信のダイアログが突然開き、せっかく掴んだリズムが崩れてしまうので、突然ダイアログが開くソフトを私は極力閉じるようにしています)。

これはDelphi/400で開発を始めて、キーボード入力の回数をいかに少なく、早く、正確に作れるかという方法を、私なりに考えた最初の形でした。

ただ、この方法では頻繁に画面の切り替えが発生し、しかも項目の1つ1つをコピーして貼り付けていかなければならず、転送項目が多くなれば、どんなにリズムよく書けたとしても時間がかかってしまいます。もっと一度にコピーして貼り付けることができる方法はないのかと、試行錯誤するようになっていました。そんなとき、私は "Excel" の能力に気付き、転送文を書くのに劇的なスピードを手に入れることになりました。

その当時、私の中で "Excel" の存在は表計算ソフトであり、ドキュメントツールでしかありませんでした。当然、プログラムを書くことに結び付けようなどと考えもしませんでした。そんなときに、以前から知ってはいましたが「Excel はけっこう文字列操作が得意」ということに改めて気付いたのです。今となってはきっかけが何だったのかは思い出すことはできませんが、"TRIM" 関数と "&"（文字結合）という機能が組み合わさったときに、"ピン" ときたのです。

その方法が次の方法になります。

Excelを使った転送文の書き方

① Excel を開く
② エミュレーター画面を開き、"DSPFMT" コマンドでファイルレイアウトを表示する
③ 表示したファイルレイアウトの項目を全て選択しコピーする
④ Excel に切り替え、"A1" セルから貼り付ける
⑤ エミュレーター画面に切り替え、テキスト記述 / 欄見出しも全て選択しコピーする
⑥ Excel に切り替え、"B1" セルから貼り付ける
⑦ ファイルレイアウトが次画面以降もある場合は、③〜⑥を繰り返す（Excel では、A 列 B 列ともに前回の続きから貼り付ける）
⑧ "C1" セルに "=TRIM(A1)" と書き、D 列も含め A 列の最終項目の行まで一度にコピーする
⑨ "E1" セルに次の式を記述する
="edt"&C1&".Text := Table1.FieldByName('"&C1&"').AsString; //"&D1
⑩ "E1" セルの内容を最終項目の行まで、一度にコピーする

実際に見ていただくと理解していただきやすいのですが、このとき E 列には転送文が全項目分書かれており、しかも各項目の見出しがコメントとして付いています。あとは E 列の内容を Delphi に貼り付ければ、転送文ができあがることになります。そして、この考え方で F 列に次のように書けば、画面からファイルへの転送文も書くことができます。

="Table1.FieldByName('"&C1&"').AsString := edt"&C1&".Text; //"&D1

この Excel ファイルを雛形として持っていれば、エミュレーター画面から項目名とテキスト記述 / 欄見出しをコピーするだけで、いつでもあっという間に転送文を書くことができます。この方法は、最初に書いた方法と比べ、確実にかつ劇的にスピードが速くなるのです。数十行の転送文であれば「ファイル→画面」「画面→ファイル」の両方が、1分とかからず書くことができます。

いかがでしょうか？ここまで読んでいただいた開発者の方で、この感動に共感していただける方はどれだけおられるでしょうか。私は、この方法に気付き結果を見たとき、心の中でガッツポーズをしていました。私の中では「キーボード入力の回数を少なくかつ早く正確に転送文を書く」ということにおいては、さまざまな手間を考えても、この方法が完成形ではないかと考えていました。

このあと、転送文はほぼこの方法を使って書いていました。しかし、しばらくすると、私はプログラムを書く機会が減っていきました。後輩も増え、私の業務は SE 職が中心になり、プログラムをあまり書かなくなっていったのです。

そんなある日、一人の後輩社員が Delphi の画面で転送文を書いているところを見ました。あまり他人がプログラムを書いているところを見る機会はないのですが、たまたま転送文を書いているところだったので、どうやって書くのか興味を感じ観察したのです。

そのとき彼が書いていた方法は、私が最初に書いた、1行目をコピーしていく方法でした。しかもコピーした後に、各項目を1行1行キーボード入力で書き換えていたのです。「それでは遅すぎる！」と、私はさっそく Excel を使った方法を彼に伝授しました。すると、先程まで苦労していた転送文をあっという間に書くことができ、彼もかなり感動していました。そしてそのとき、彼が言った一言が次のステップへの引き金となったのです。

「AS/400 から直接ファイル情報を持ってきて編集できたら、コピーの手間がなくなりますよね」

確かにその通り！ そして、それを可能にするツールが身近に存在していたのです。「Delphi/400」ならそれができるのです。

その後、社内でツール化することとなり「Delphi/400 のプログラムを書くためのプログラムを、Delphi/400 を使って開発する」という、少し不思議なプロジェクトがスタートすることとなりました。そして、とうとう今年の春に、ミガロ開発支援ツールの第1弾として華々しく社内デビューを飾りました。

この開発支援ツールは、ライブラリー名とファイル名を入力するだけで、一瞬にして転送文を画面に表示してくれるという優れもので、あとは画面の内容をコピーして Delphi へ貼り付ければいいだけです。も

ちろん「ファイル→画面」「画面→ファイル」の両方に対応しているので、特に入力系のプログラムで転送文が多い場合、工数削減につながるのではないかと思います。今後どこかの機会で、この開発支援ツール第1弾を皆様に公開させていただければと思っています。

「Delphi/400の転送文を書く」ことに対して、キーボード入力の回数を少なくしかつ早く正確に作れる方法を考えてきた開発者の話はこれで終わりです。「転送文を書く」という話題でここまで話を膨らませることができるとは、正直思っていませんでした。お楽しみいただけましたでしょうか。

この本を読まれている方の中には、プログラム開発に携わられている方も多くいらっしゃると思います。そして、開発する際にはいろいろな創意工夫をされてきたのではないでしょうか。今度は、ぜひ皆様の創意工夫を教えていただければ嬉しいです。

最後までお付き合いいただきまして、ありがとうございました。

MIGARO TECHNICAL REPORT

Migaro Technical Report
No.1 2008 秋
ミガロ テクニカルレポート

2008 年 11 月 1 日 初版発行

◆発行
株式会社ミガロ
〒 556-0017
大阪市浪速区湊町 2-1-57 難波サンケイビル 13F
TEL：06（6631)8601 FAX：06（6631)8603
http://www.migaro.co.jp/

◆発行人
上甲 將隆

◆編集協力
アイマガジン株式会社

◆デザインフォーマット
近江デザイン事務所

©Migaro Technical Report2008
本誌コンテンツの無断転載を禁じます
本誌に記載されている会社名、製品名、サービスなどは一般に各社の商標または
登録商標です。本誌では、TM、® マークは明記していません。

MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

株式会社ミガロ.

http://www.migaro.co.jp/

本社
〒556-0017
大阪市浪速区湊町2-1-57
難波サンケイビル 13F
TEL:06(6631)8601
FAX:06(6631)8603

東京営業所
〒106-0041
東京都港区麻布台1-4-3
エグゼクティブタワー麻布台 11F
TEL:03(5573)8601
FAX:03(5573)8602