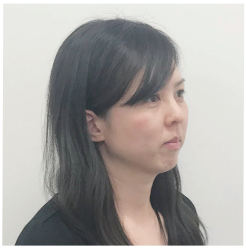
Delphi/400

Delphi/400アプリケーション向け開発支援ツールの作成方法

株式会社ミガロ.
システム事業部システム１課
都地　奈津美



略　歴

生年月日:1989年8月19日
最終学歴:2012年　関西学院大学　理工学部卒業
入社年月:2012年4月　株式会社ミガロ.入社
社内経歴:2012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受託開発とシステム保守を担当している。開発スキルの向上を目指し、日々精進している。

1.はじめに

アプリケーション開発や、関連するドキュメントの作成には、多大なリソースが必要とされる。リソース不足解消の手段として、開発支援ツールを導入している企業も多いだろう。開発支援ツールを導入することで、手軽にアプリケーション開発が可能となる。例えば、ボタンひとつで、データ取得ロジックの作成やドキュメントの作成が可能となるため、アプリケーション開発の効率化が期待される。また、プログラミングに関する知識や技術が不足しているユーザーでもアプリケーション開発に携われるため、リソース不足の解消にも繋がるだろう。

本稿では、Delphi/400アプリケーションに特化した開発支援ツールの作成例を以下の手順で紹介する。

【第2章】ExcelドキュメントからDDSソース＆オブジェクトを作成する方法
【第3章】DDSソース＆オブジェクトからExcelドキュメントを作成する方法
【第４章】ＤＤＳソース＆オブジェクトからＳＱＬ文やParamByNameメソッド、FieldByNameメソッドを作成する方法
【第5章】DDSソース＆オブジェクトからTFDMemTableのフィールド定義を作成する方法

IBMiの操作に不慣れな方や、Delphi/400の開発を始めたばかりの方でも、本サンプルを活用して頂けるよう、本稿の最後にダウンロードURLを記載している。本サンプルはダウンロード可能であるため、本稿へのソースの記載は一部抜粋とする。

なお、本サンプルのＤｅｌｐｈｉ/４００バージョンは11 Alexandriaを使用する。

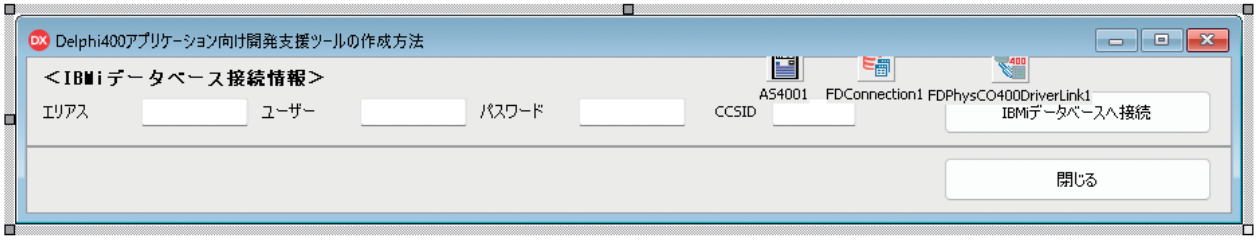
2.オブジェクトの自動生成（Excel→DDS変換）

本章では、Excelドキュメントで作成されたファイルレイアウト情報から、IBMiデータベース上にDDSソース並びにオブジェクトを自動生成する方法を紹介する。
IBMiデータベース上にDDSソースを使用してオブジェクトを作成する場合、①ライブラリを作成、②DDSソース格納用のオブジェクトを作成、③DDSソースを作成、④コンパイルの4ステップが必要となる。
本章で紹介する機能は、①Excelドキュメントでファイルレイアウトを作成、②Delphi/400アプリケーション上で必要情報を入力、③ボタンをクリックの3ステップで完了となる。ステップ数としては大きな違いはないが、IBMiの操作に不慣れなユーザーには、IBMi上でDDSソースを作成＆コンパイルをするより、Excelドキュメント＆Delphi/400アプリケーション上で必要な情報を入力すればオブジェクトが作成できる方が、作業の効率化に繋がるだろう。

2-1.IBMiデータベースへの接続

接続先情報を設定するためのTEdit、データベース接続で使用するTAS400、TFDConnection、TFDPhysCO400DriverLink、並びにTLabelやTButtonを画面に配置する【図1】。TFDConnectionのプロパティについては過去のミガロ.テクニカルレポート「FireDAC実践プログラミングテクニック」を参考に設定して頂きたい。
https://www.migaro.co.jp/tr/no11/tech/11_01_02.pdf

図 1　IBMi接続：コンポーネントの配置



「IBMiデータベースへ接続」ボタンのOnClick処理で、
IBMiデータベースへ接続する処理【ソース1】を記述する。

ソース 1

IBMiデータベース接続処理

```
// AS400の接続
with AS4001 do
begin
  Active := False;
  PLUAlias     := edtAlias.Text;  // エリアス
  Userid       := edtUser.Text;   // ユーザー
  PWD          := edtPass.Text;   // パスワード
  Active := True;

  // CCSID変更
  RemoteCmd('CHGJOB CCSID(' + edtCCSID.Text + ')');
end;

// FDConnectionの接続
with FDConnection1 do
begin
  Connected := False;
  Params.Values['Database']     := edtAlias.Text;  // エリアス
  Params.Values['User_Name']    := edtUser.Text;   // ユーザー
  Params.Values['Password']     := edtPass.Text;   // パスワード
  Connected := True;
end;
```

2-2.Excelフォーマットの出力

本章でのオブジェクト自動生成機能は、ファイル情報がExcelドキュメントに記述される想定とする。物理ファイル用のシート、論理ファイル用のシートを定義したExcelオブジェクトの雛形ファイル【図2】を事前に作成しておく。物理ファイルは物理ファイル単位、論理ファイルは基となる物理ファイル単位でシートが作成される想定とする。

図 2　ファイル自動生成：Excelフォーマットファイル

物理ファイル用　※1シート目の情報

ファイルID	ファイル名	レコード様式	I：ユニーク	備考

No	フィールドID	フィールド名	Key順	A：昇順 D：降順	属性	桁数	小数	備考
1								

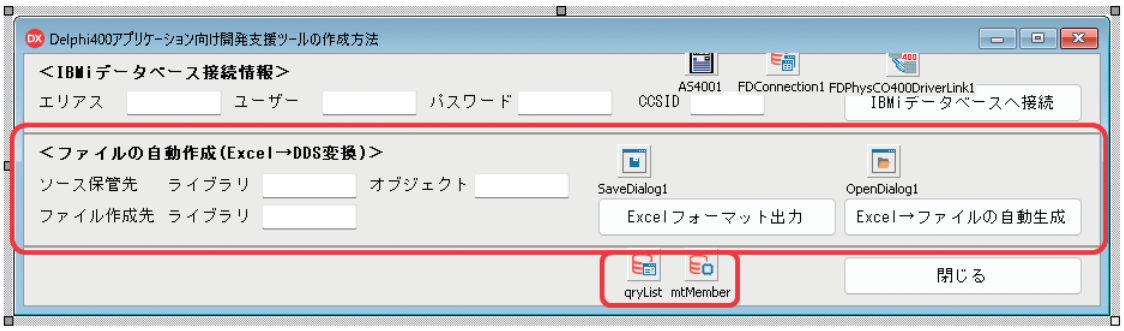
論理ファイル用　※2シート目の情報

物理ファイルID	物理ファイルレコード様式	備考

No	論理ファイルID	論理ファイル名	I：ユニーク	フィールドID	A：昇順 D：降順	S：選択 O：除外	AND/OR	EQ/NE/GT/GE/ LE/LT	条件	備考
1										

2-1.で作成したサンプルプログラムに、ソース保管先、オブジェクト作成先の情報を設定するためのTEdit、ファイル保存を行うためのTSaveDialog、ファイル選択を行うためのTOpenDialog、データベース接続で使用するTFDQueryとTFDMemTable、並びにTLabelやTButtonを画面に追加配置する【図3】。

図 3　ファイル自動生成:コンポーネントの配置



「Excelフォーマット出力」ボタンのOnClick処理で、【図2】の雛形ファイルをTSaveDialogで指定の保管先へ保存する処理を記述する【ソース2】。

ソース 2

btnExcelFormatClick（「Excelフォーマット出力」ボタン押下時処理）

```
// コピー元ファイル
sFrom := ExtractFilePath(Application.ExeName) + 'Template';
sFrom := IncludeTrailingPathDelimiter(sFrom)  + 'Template.xlsx';

// コピー先ファイル
sTo := SaveDialog1.FileName;

// フォーマットファイルを、保存ダイアログで指定した場所にコピー
bCopy := CopyFile(PChar(sFrom),   // コピー元
                  PChar(sTo),     // コピー先
                  False);         // True：同名ファイル存在時コピーしない、False：上書き

// エラー
if (not bCopy) then
begin
  ShowMessage('ファイルの保存に失敗しました。');
  Abort;
end;
```

2-3.オブジェクトの自動生成

「Excel→ファイルの自動生成」ボタンのOnClick処理で、オブジェクトの自動生成処理を記述する。処理の流れは、①ライブラリ作成、②DDSソース格納用のオブジェクト作成、③DDSソース作成、④コンパイルとなる。

①ライブラリ作成

ライブラリの作成には、IBMi上のコマンド「CRTLIB」を実行する必要がある。TAS400コンポーネントのRemoteCmdメソッドを利用し、コマンドを実行する。作成前にはコマンド「CHKOBJ」を実行し、ライブラリが存在しない場合のみ作成処理を行う。「CRTLIB」コマンドが実行できない場合、例外エラーが生成されるため、処理中断とする**【ソース3】**。

ソース 3

CHKOBJ:存在チェック

```
AS4001.RemoteCmd('CHKOBJ '
                 + 'OBJ(' + edtEtoD_SrcLib.Text + ') '    // ライブラリー名
                 + 'OBJTYPE(*LIB)');                      // オブジェクトタイプ：ライブラリ
```

CRTLIB:ライブラリの作成　※例外エラー時のみ実行

```
AS4001.RemoteCmd('CRTLIB '
                 + 'LIB(' + edtEtoD_SrcLib.Text + ') '    // ライブラリー名
                 + 'TEXT(''テストライブラリ'')' );        // テキスト
```

②DDSソース格納用のオブジェクト作成

オブジェクトの作成には、コマンド「CRTSRCPF」を実行する。作成前にはコマンド「CHKOBJ」を実行し、オブジェクトが存在しない場合のみ作成処理を行う。「CRTSRCPF」コマンドが例外エラーの場合、処理中断とする**【ソース4】**。オブジェクトのメンバー一覧(ファイル一覧)を、コマンド「DSPFD」を実行してQTEMPに出力し、TFDMemTableに内部保持しておく**【ソース5】**。

ソース 4

CHKOBJ:存在チェック

```
AS4001.RemoteCmd('CHKOBJ '
                 + 'OBJ(' + sSrcObj + ') '    // DDSソース保管先
                 + 'OBJTYPE(*FILE)');         // オブジェクトタイプ：ファイル
```

CRTSRCPF:オブジェクトの作成　※例外エラー時のみ実行

```
AS4001.RemoteCmd('CRTSRCPF '
                 + 'FILE(' + sSrcObj + ') '          // ライブラリ名/オブジェクト名
                 + 'IGCDTA(*YES) '                   // ユーザー指定のDBCSデータ
                 + 'TEXT(''DDSソース'') '            // テキスト
                 + 'CCSID(' + edtCCSID.Text + ')');  // 文字コード
```

ソース 5

オブジェクトのメンバー一覧取得

```
// メンバー一覧の取得
AS4001.RemoteCmd('DSPFD '
                + 'FILE(' + sSrcObj + ') '      // ライブラリ名/ファイル名
                + 'TYPE(*MBRLIST) '             // 情報のタイプ：メンバー一覧
                + 'OUTPUT(*OUTFILE) '           // 出力：ファイル
                + 'OUTFILE(QTEMP/MEMBER) '      // 出力ファイル
                + 'OUTMBR(*FIRST *REPLACE)');   // 出力メンバー：FIRST&置き換え

// メンバー一覧の読み込み
with qryList do
begin
  Close;
  SQL.Clear;
  SQL.Add('SELECT MLNAME');
  SQL.Add('FROM   QTEMP/MEMBER');
end;

// データ取得
qryList.Open;

try
  // MemTableに取得したデータを追加
  mtMember.Close;
  mtMember.AppendData(qryList, False);
finally
  qryList.Close;
end;
```

③DDSソース作成

(1)TOpenDialogで指定のExcel形式のファイル情報から、DDSソースを作成する。Excel形式のファイル情報の読み込みはOLEを利用し、シート数分、オブジェクト作成の処理を繰り返す。OLEの使用方法については、過去のミガロ.テクニカルレポート「OLEを利用したExcel出力のパフォーマンス向上手法」で記載があるため、本稿での説明は割愛させて頂く。

https://www.migaro.co.jp/tr/no11/tech/11_01_01.pdf

(2)②で取得したメンバー一覧にExcelドキュメント情報で指定のファイル名が存在しない場合、コマンド「ADDPFM」を実行してメンバーを追加する。メンバーが存在する場合、既に作成済みのDDSソースとなるため、次のシートの処理へ進む**【ソース6】**。

ソース 6

メンバー追加

```
// メンバーが存在しない場合、メンバーを追加
if (not mtMember.Locate('MLNAME', VarArrayOf([AFileID]), [])) then
begin
  AS4001.RemoteCmd('ADDPFM '
                 + 'FILE(' + sSrcObj + ') '          // ライブラリ名/オブジェクト名
                 + 'MBR(' + AFileID + ') '           // メンバー：ファイル名
                 + 'TEXT(''' + AFileName + ''') '    // テキスト
                 + 'SRCTYPE(' + AType + ')');        // ソース仕様タイプ
end
// メンバーが存在する場合、エラー
else
begin
  AErrList.Add('・【' + ASheetName + '】シート：既に作成済みのファイルです。' + sErrFile);
end;
```

エラーは内部保持し、まとめて
ShowMessageする

(3)Excel形式のファイル情報を基に、物理ファイル・論理ファイルそれぞれの形式に合った形のDDSソースを作成し、1ファイル単位でTStringListへ内部保持する。DDSソースの内部保持が完了すれば、(2)で追加したIBMiデータベース上のソースファイルへ内部保持値を追加する**【ソース7】**。

ソース 7

ソースファイルへ内部保持値を追加

```
// OVRDBF
AS4001.RemoteCmd('OVRDBF '
                 + 'FILE(' + AFileID + ') '     // 一時変更中のファイル名
                 + 'TOFILE(' + sSrcObj + ') '   // ライブラリ名/オブジェクト名
                 + 'MBR(' + AFileID + ') '      // メンバー：ファイル名
                 + 'OVRSCOPE(*JOB)');           // 有効範囲：ジョブ

try
  // DDSソースファイルに、内部保持したDDSソース情報を追加
  with qryList do
  begin
    for i := 0 to ADdsSource.Count - 1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('INSERT INTO ' + AFileID + '(SRCSEQ, SRCDAT, SRCDTA) VALUES (:SRCSEQ, :SRCDAT, :SRCDTA)');
      ParamByName('SRCSEQ').AsCurrency := i + 1;          // SEQ
      ParamByName('SRCDAT').AsInteger  := iDate;          // 日付
      ParamByName('SRCDTA').AsString   := ADdsSource[i];  // ソース
      ExecSQL;
    end;
  end;
finally
  // DLTOVR
  AS4001.RemoteCmd('DLTOVR FILE(' + AFileID + ') LVL(*JOB)');
end;
```

④コンパイル

DDSソース作成完了後、コンパイルを行う。物理ファイルはコマンド「CRTPF」、論理ファイルはコマンド「CRTLF」を実行し、画面項目「ファイル作成先」で指定のライブラリへDDSソースをコンパイルする**【ソース8】**。

ソース 8

コンパイル:物理ファイル　※内部保持したファイル数分コンパイルする

```
AS4001.RemoteCmd('CRTPF '
                 + 'FILE(' + edtEtoD_Lib.Text + '/' + sFilePF + ') '   // コンパイル先
                 + 'SRCFILE(' + sSrcObj + ') '                         // ソース保管場所
                 + 'SRCMBR(' + sFilePF + ') '                          // ファイルID
                 + 'MAXMBRS(*NOMAX) '                                  // メンバーの最大数
                 + 'SIZE(*NOMAX)');                                    // 初期レコード数
```

コンパイル:論理ファイル　※内部保持したファイル数分コンパイルする

```
AS4001.RemoteCmd('CRTLF '
                 + 'FILE(' + edtEtoD_Lib.Text + '/' + sFilePF + ') '   // コンパイル先
                 + 'SRCFILE(' + sSrcObj + ') '                         // ソース保管場所
                 + 'SRCMBR(' + sFilePF + ') '                          // ファイルID
                 + 'MAXMBRS(*NOMAX)');                                 // メンバーの最大数
```

以上で、Excelドキュメントに入力されたファイル情報から、DDSソース、オブジェクトを自動生成するプログラムは完成である。上記プログラムを実行し、自動生成されたDDSソース、オブジェクトを確認する。取り込むExcelドキュメントは、**【図4】**とし、Excelに記載の通り、IBMiデータベース上にDDSソース、オブジェクトが自動生成されていることが分かる**【図5～6】**。

図 4　ファイル自動生成:実行結果(ファイルレイアウト)

プログラムで読み込むExcelファイルのファイルレイアウト：物理ファイル　※1シート目の情報

ファイルID	ファイル名	レコード様式	ﾕﾆｰｸ	備考
TEST1	テストファイル１	RTEST1	1	

No.	フィールドID	フィールド名	Key順	A：昇順 D：降順	属性	桁数	小数	備考
1	DATA1_1	半角フィールド	1	A	A	10		
2	DATA1_2	全角フィールド	2	D	O	20		
3	DATA1_3	数値フィールド１			S	10	0	
4	DATA1_4	数値フィールド２			P	10	2	

プログラムで読み込むExcelファイルのファイルレイアウト：論理ファイル　※2シート目の情報

物理ファイルID	物理ファイルレコード様式	備考
TEST1	RTEST1	テストファイル１

No.	論理ファイルID	論理ファイル名	ﾕﾆｰｸ	フィールドID	A：昇順 D：降順	S：選択 O：除外	AND/OR	EQ/NE/GT/GE/ LE/LT	条件	備考
1	TEST1L01	テストファイル１　Ｌ０１	1	DATA1_3	A					
				DATA1_4	D					
				DATA1_1		S		EQ	'1'	データ１="1"

図 5　ファイル自動生成：実行結果（物理ファイル）

IBMiデータベース上に自動生成されたDDSソース

```
0001.00       A*************************************************************************  230831
0002.00       A*    テストファイル１    ／  TEST1                                        230831
0003.00       A*     CREATE : 2023/08/31   作成者 :  NTSUJI                             230831
0004.00       A*     UPDATE : YYYY/MM/DD   更新者 :                                     230831
0005.00       A*************************************************************************  230831
0006.00       A                                     UNIQUE                               230831
0007.00       A          R RTEST1                   TEXT(' テストファイル１ ')           230831
0008.00       A*                                                                        230831
0009.00       A            DATA1_1       10A       COLHDG(' 半角フィールド ')           230831
0010.00       A            DATA1_2       20O       COLHDG(' 全角フィールド ')           230831
0011.00       A            DATA1_3       10S 0     COLHDG(' 数値フィールド１ ')         230831
0012.00       A            DATA1_4       10P 2     COLHDG(' 数値フィールド２ ')         230831
0013.00       A*                                                                        230831
0014.00       A* キー情報                                                               230831
0015.00       A          K DATA1_1                                                      230831
0016.00       A          K DATA1_2                  DESCEND                             230831
```

IBMiデータベース上に自動生成されたファイル

物理ファイル	NTSUJILIB/TEST1	様式名	RTEST1	レコード長	46
様式記述	テストファイル１				

5= 詳細

選択	項目名	桁数	属性	キー順	開始	終了	テキスト記述／欄見出し
_	DATA1_1	10	A	1 ANN	1	10	半角フィールド
_	DATA1_2	20	O	2 DNN	11	30	全角フィールド
_	DATA1_3	10 0	S		31	40	数値フィールド１
_	DATA1_4	10 2	P		41	46	数値フィールド２

図 6　ファイル自動生成：実行結果（論理ファイル）

IBMiデータベース上に自動生成されたDDSソース

```
0001.00       A*************************************************************************  230831
0002.00       A*    テストファイル１   ＬＯ１ (LF)   ／   TEST1                          230831
0003.00       A*     CREATE : 2023/08/31   作成者 :  NTSUJI                             230831
0004.00       A*     UPDATE : YYYY/MM/DD   更新者 :                                     230831
0005.00       A*************************************************************************  230831
0006.00       A                                     UNIQUE                               230831
0007.00       A          R RTEST1                   PFILE(TEST1)                         230831
0008.00       A                                     TEXT(' テストファイル１   ＬＯ１ ')   230831
0009.00       A*                                                                        230831
0010.00       A* キー情報                                                               230831
0011.00       A          K DATA1_3                                                      230831
0012.00       A          K DATA1_4                  DESCEND                             230831
0013.00       A*                                                                        230831
0014.00       A* セレクト&オミット情報                                                  230831
0015.00       A          S DATA1_1                  COMP(EQ '1')                        230831
```

IBMiデータベース上に自動生成されたファイル

論理ファイル	NTSUJILIB/TEST1LO1	様式名	RTEST1	レコード長	46
様式記述	テストファイル１				

5= 詳細

選択	項目名	桁数	属性	キー順	開始	終了	テキスト記述／欄見出し
_	DATA1_1	10	A		1	10	半角フィールド
_	DATA1_2	20	O		11	30	全角フィールド
_	DATA1_3	10 0	S	1 ASN	31	40	数値フィールド１
_	DATA1_4	10 2	P	2 DSN	41	46	数値フィールド２

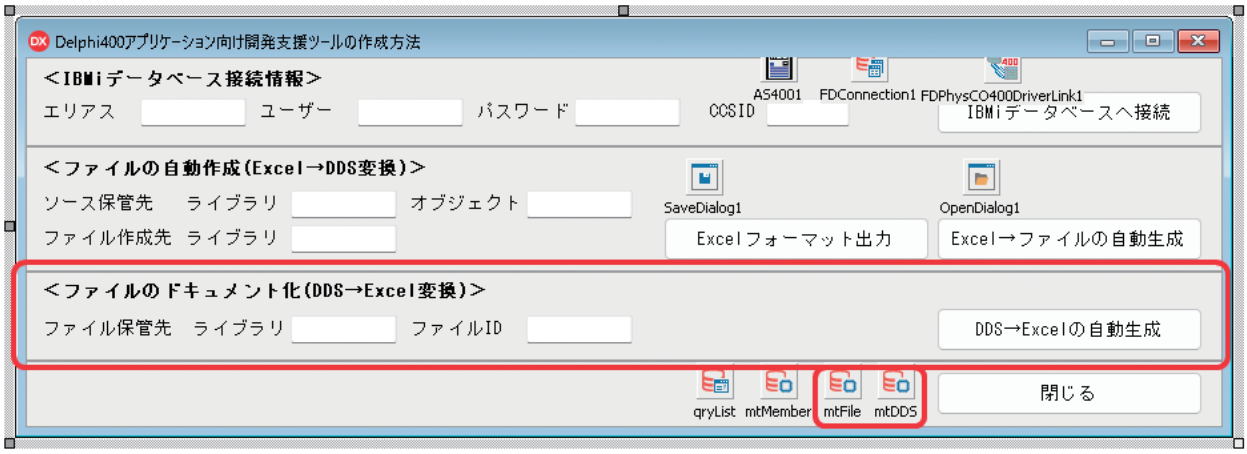
3.オブジェクトのドキュメント化(DDS→Excel変換)

本章では、IBMiデータベース上に作成されたDDSソース及びオブジェクトから、ファイルレイアウト(Excelドキュメント)を自動生成する方法を紹介する。
IBMiデータベース上のオブジェクトのドキュメント化が必要な場合、DDSソースを確認、もしくは、オブジェクトの情報を確認し、フィールド情報を資料に記載する必要がある。
本章で紹介する機能は、オブジェクト保管先の情報を入力すればDDS→Excelの自動変換ができるため、作業の効率化に繋がるだろう。

3-1.コンポーネントの配置

第2章で作成したサンプルプログラムに、ソース保管先・ファイル保管先の情報を設定するためのTEdit、データベース接続で使用するTFDMemTable、並びにTLabelやTButtonを画面に追加配置する【図7】。

図 7 ファイルドキュメント化:コンポーネントの配置



「DDS→Excelの自動生成」ボタンのOnClick処理で、ドキュメントの自動生成処理を記述する。自動生成するExcelドキュメントの雛形は、【図2】の雛形ファイルとする。

3-2.ファイル一覧の取得

コマンド「DSPOBJD」を実行し、画面項目「ファイル保管先」で指定のファイル一覧(ライブラリ内のオブジェクト一覧)および各オブジェクト情報(ファイルID、テキスト、属性、DDSソース保管先)をTFDMemTableに内部保持する【ソース9】。取得したファイル一覧の中に画面で指定のファイルIDが存在しない場合、処理中断とする。画面でファイルIDの指定がない場合、全ファイルを対象とするためチェック不要とする。

ソース 9

ファイル一覧の取得

```
// ファイル一覧の取得
AS4001.RemoteCmd('DSPOBJD '
                 + 'OBJ(' + edtDtoE_SrcLib.Text +'/*ALL) '  // DDSソース保管先
                 + 'OBJTYPE(*FILE) '                        // オブジェクトタイプ：ファイル
                 + 'OUTPUT(*OUTFILE) '                      // 出力：ファイル
                 + 'OUTFILE(QTEMP/FILELIST)');              // 出力ファイル

// ファイル一覧の読み込み
with qryList do
begin
  Close;
  SQL.Clear;
  SQL.Add('SELECT ODOBNM,');  // ファイルID
  SQL.Add(       'ODOBTX,');  // テキスト
  SQL.Add(       'ODOBAT,');  // 属性(PF/LF)
  SQL.Add(       'ODSRCL,');  // ソース・ファイル・ライブラリー
  SQL.Add(       'ODSRCF,');  // ソース・ファイル名
  SQL.Add(       'ODSRCM');   // ソース・ファイル・メンバー
  SQL.Add('FROM   QTEMP/FILELIST');
  SQL.Add('WHERE (ODOBAT = ''PF'' OR ODOBAT = ''LF'')');  // PF/LFのみを対象
  // ファイルID<>ブランクの場合、対象ファイルのみを取得
  if (edtDtoE_SrcFile.Text <> '') then
  begin
    SQL.Add('AND ODOBNM = :ODOBNM');
    ParamByName('ODOBNM').AsString := edtDtoE_SrcFile.Text;
  end;
end;

// データ取得
qryList.Open;

try
  // MemTableに取得したデータを追加
  mtFile.Filtered := False;
  mtFile.Close;
  mtFile.AppendData(qryList, False);
  mtFile.IndexFieldNames := 'ODOBNM';
finally
  qryList.Close;
end;
```

3-3.DDS→Excel変換

第2章の「Excelフォーマット出力」ボタンのOnClick処理を呼び出して雛形ファイルを保存し、DDS→Excel変換の処理を記述する。物理ファイルか論理ファイルかで処理のタイミングが少し違うが、処理の流れは、①DDSソース取得、②Excel雛形ファイルのシートコピー、③DDSソースの読み込み、④Excelドキュメント保存となる。

①DDSソース取得

コマンド「CHKOBJ」を実行してDDSソースが存在する場合、TFDQueryを利用してDDSソースを取得する**【ソース10】**。DDSソースの保管先は、3-2.で取得した値とし、ソースが存在しない場合、処理中断とする。取得したDDSソースはTFDMemTableに内部保持する。

ソース 10

CHKOBJ:DDSソース存在チェック

```
AS4001.RemoteCmd('CHKOBJ '
                 + 'OBJ(' + ASrcObj + ') '    // DDSソース保管先
                 + 'OBJTYPE(*FILE) '          // オブジェクトタイプ：ファイル
                 + 'MBR(' + ASrcMem + ')');   // メンバー：ファイル名
```

DDSソースの取得

```
// OVRDBF
AS4001.RemoteCmd('OVRDBF '
                 + 'FILE(' + AFileID + ') '    // 一時変更中のファイル名
                 + 'TOFILE(' + ASrcObj + ') '  // ライブラリ名/オブジェクト名
                 + 'MBR(' + AFileID + ') '     // メンバー：ファイル名
                 + 'OVRSCOPE(*JOB)');          // 有効範囲：ジョブ

try
  // DDSソースの読み込み
  with qryList do
  begin
    Close;
    SQL.Clear;
    SQL.Add('SELECT * FROM ' + AFileID);
  end;

  // データ取得
  qryList.Open;

  try
    // MemTableに取得したデータを追加
    mtDDS.Close;
    mtDDS.AppendData(qryList, False);
  finally
    qryList.Close;
  end;
finally
  // DLTOVR
  AS4001.RemoteCmd('DLTOVR FILE(' + AFileID + ') LVL(*JOB)');
end;
```

②Excel雛形ファイルのシートコピー
保存したExcelドキュメントに対してシートのコピーを行う
【ソース11】。

ソース 11

Excelファイルのシートのコピー

```
// シートをコピー
AWBook.Worksheets[iSheetNo].Copy(After := AWBook.Worksheets[AWBook.Sheets.Count]);
AWBook.ActiveSheet.Name := AFileID;  // シート名

// 対象シートを選択
ASheet := AWBook.Sheets[AFileID];
ASheet.Activate;

// 5～53行目を削除
ASheet.Range['A5', 'A53'].EntireRow.Delete;
```

③DDSソースの読み込み
①で内部保持したDDSソースを1行ずつ読み込み、ファイル情報とフィールド情報を内部保持する。

④Excelドキュメント保存
③で内部保持した情報をExcelドキュメントへセットする。不要になった雛形シート(1～2シート目)を削除後、上書き保存する。

以上で、DDSソース及びオブジェクトから、Excelドキュメントのファイルレイアウトを自動生成するプログラムは完成である。上記プログラムを実行し、自動生成されたExcelドキュメントを確認する。変換対象は第2章で作成したファイル**【図5～6】**とし、ファイル・フィールド情報通りにExcel変換されていることが分かる**【図8】**。

図 8　ファイルドキュメント化:実行結果

自動生成されたExcelファイルのファイルレイアウト：物理ファイル　※1シート目の情報

ファイルID	ファイル名	レコード様式	｜:ユニーク	備考
TEST1	テストファイル１	RTEST1	1	

No.	フィールドID	フィールド名	Key順	A：昇順 D：降順	属性	桁数	小数	備考
1	DATA1_1	半角フィールド	1	A	A	10		
2	DATA1_2	全角フィールド	2	D	O	20		
3	DATA1_3	数値フィールド１			S	10	0	
4	DATA1_4	数値フィールド２			P	10	2	

自動生成されたExcelファイルのファイルレイアウト：論理ファイル　※2シート目の情報

物理ファイルID	物理ファイルレコード様式	備考
TEST1	RTEST1	

No.	論理ファイルID	論理ファイル名	｜:ユニーク	フィールドID	A：昇順 D：降順	S：選択 O：除外	AND/OR	EQ/NE/GT/GE/ LE/LT	条件	備考
1	TEST1L01	テストファイル１　Ｌ０１	1	DATA1_3	A					
				DATA1_4	D					
				DATA1_1		S		EQ	'1'	

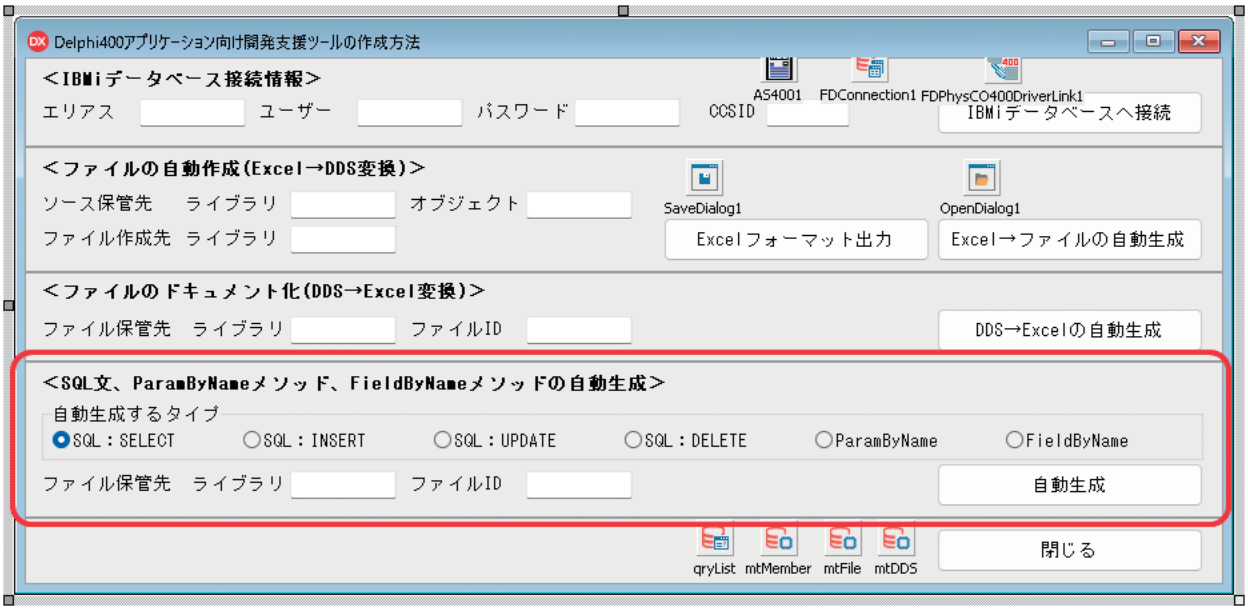
4.SQL文、ParamByName・FieldByNameメソッドの自動生成

本章では、IBMiデータベース上に作成されたDDSソース及びオブジェクトから、SQL文、ParamByName・FieldByNameメソッドを自動生成する方法を紹介する。SQL文の実行、Delphi/400でParamByName・FieldByNameメソッドを使用する場合、DDSソースを確認、もしくは、オブジェクトの情報を確認する必要がある。本章で紹介する機能は、オブジェクト保管先の情報を入力すれば自動生成できるため、作業の効率化に繋がるだろう。

4-1.コンポーネントの配置

第3章で作成したサンプルプログラムに、自動生成するタイプを選択するためのTRadioGroup、ファイル保管先の情報を設定するためのTEdit、並びにTLabelやTButtonを画面に追加配置する【図9】。

図 9　SQL文の自動生成:コンポーネントの配置

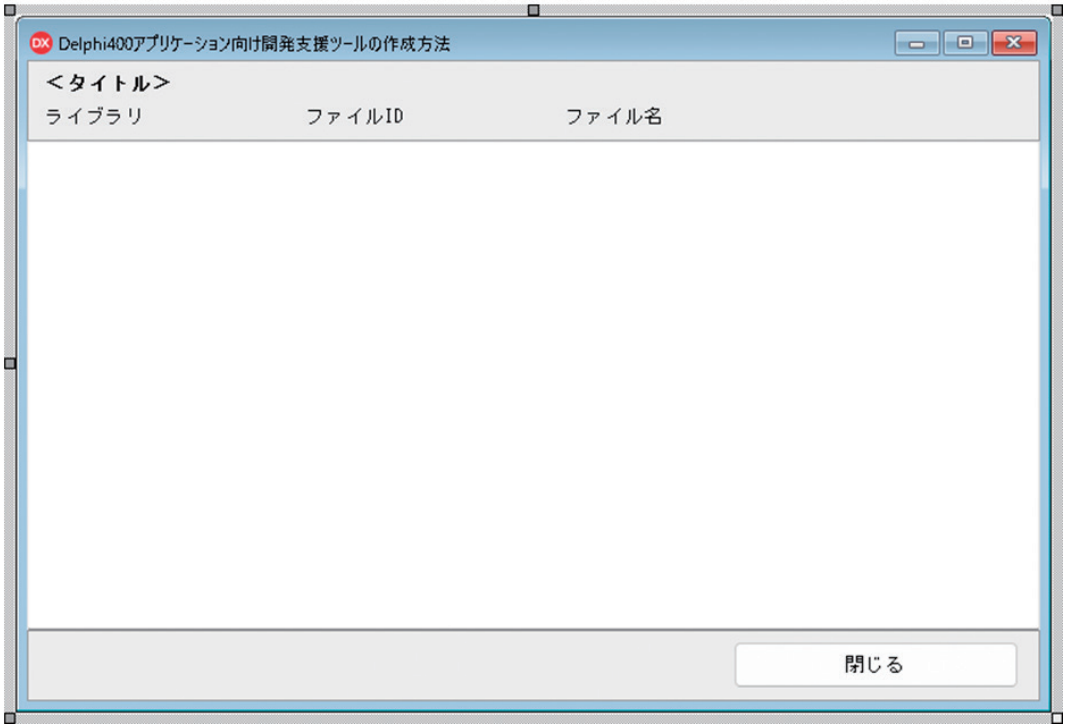


4-2.自動生成

「自動生成」ボタンのOnClick処理で、SQL文、ParamByName・FieldByNameメソッドを自動生成する処理を記述する。処理の流れは、①フィールド情報取得、②SQL文の自動生成、③ParamByName、FieldByNameメソッドの自動生成となる。

まず、自動生成した結果を表示するためのサブ画面を新規作成する。結果を表示するためのTMemo、並びにTLabelやTButtonを画面に配置する**【図10】**。

図 10　SQL文の自動生成：サブ画面の作成



①フィールド情報取得

コマンド「DSPFFD」を実行し、画面項目「ファイル保管先」で指定のファイル情報（ファイル名）、フィールド情報（フィールドID、フィールド名、桁数）をリストに内部保持、TFDQueryを利用してファイルを参照し、フィールドタイプをリストに内部保持する**【ソース12】**。

ソース 12

フィールド情報の取得

```
// コマンドを発行し、一時ファイルに結果出力
AS4001.RemoteCmd('DSPFFD '
                 + 'FILE(' + sLib + '/' + sFile + ') '  // ファイル
                 + 'OUTPUT(*OUTFILE) '                  // 出力：ファイル
                 + 'OUTFILE(QTEMP/' + sFile + ') '      // 出力ファイル
                 + 'OUTMBR(*FIRST *REPLACE)');          // 出力メンバー：FIRST&置き換え

// フィールド情報の読み込み
with qryList do
begin
```

```
  Close;
  SQL.Clear;
  SQL.Add('SELECT WHTEXT,');  // ファイル名
  SQL.Add(       'WHFLDI,');  // フィールドID
  SQL.Add(       'WHFTXT,');  // フィールド名
  SQL.Add(       'WHFLDB');   // 桁数
  SQL.Add('FROM   QTEMP/' + sFile);
  SQL.Add('ORDER  BY WHFOBO');
  Open;

  try
    AFileName := FieldByName('WHTEXT').AsString;  // ファイル名

    // リストに取得情報を保管
    while not Eof do
    begin
      AFieldIDList.Add(FieldByName('WHFLDI').AsString);    // フィールドID
      AFieldNameList.Add(FieldByName('WHFTXT').AsString);  // フィールド名
      AFieldLenList.Add(FieldByName('WHFLDB').AsString);   // 桁数
      Next;
    end;
  finally
    qryList.Close;
  end;
end;

// フィールドタイプの取得
with qryList do
begin
  Close;
  SQL.Clear;
  SQL.Add('SELECT * FROM ' + sLib + '/' + sFile);
  SQL.Add('FETCH FIRST 1 ROWS ONLY');
  Open;

  try
    // リストに取得情報を保管
    for i := 0 to FieldCount - 1 do
    begin
      case Fields[i].DataType of
        ftInteger : AFieldTypeList.Add('Integer');
        ftFloat   : AFieldTypeList.Add('Float');
        ftCurrency: AFieldTypeList.Add('Currency');
        else        AFieldTypeList.Add('String');
      end;
      Next;
    end;
  finally
    qryList.Close;
  end;
end;
```

②SQL文の自動生成

画面項目「自動生成するタイプ」で「SQL：SELECT」、「SQL：INSERT」、「SQL：UPDATE」、「SQL：DELETE」のいずれかを選択時、②で内部保持した情報を基にSQL文を自動生成する**【ソース13】**。WHERE句やSET句で使用する条件（右辺）、VALUES句で使用する値については、Delphi/400ソース上で使用することを想定しバインド変数とする。自動生成した内容は、①で作成したサブ画面に表示する。

③ParamByName、FieldByNameメソッドの自動生成

画面項目「自動生成するタイプ」で「ParamByName」、「FieldByName」のいずれかを選択時、②で内部保持した情報を基にParamByName、FieldByNameメソッドを自動生成する**【ソース14】**。自動生成した内容は、①で作成したサブ画面に表示する。

ソース 13

SQL文の自動生成

```
case iMode of
  1: slText.Add('SELECT');                          // SELECT
  2: slText.Add('INSERT INTO ' + sSQL_FileID + ' ('); // INSERT
  3: slText.Add('UPDATE '      + sSQL_FileID);       // UPDATE
  4: slText.Add('DELETE');                          // DELETE
end;

// UPDATE
if (iMode = 3) then
begin
  slText.Add('SET');
end;

// DELETE
if (iMode = 4) then
begin
  slText.Add('FROM ' + sSQL_FileID);
  slText.Add('');
end;

// フィールド情報(DELETE以外)
if (iMode <> 4) then
begin
  for i := 0 to slFieldID.Count - 1 do
  begin
    case iMode of
      1: sText := slFieldID[i];                       // SELECT
      2: sText := slFieldID[i];                       // INSERT
      3: sText := slFieldID[i] + ' = :' + slFieldID[i]; // UPDATE
    end;

    // 最終フィールドでない場合、カンマを付与
    if (i <> (slFieldID.Count - 1)) then
```

```
    begin
      sText := sText + ',';
    end;

    // 内部保持
    slText.Add(sText);

  end;
end;

case iMode of
  1: slText.Add(#13#10 + 'FROM ' + sSQL_FileID);  // SELECT
  2: slText.Add(#13#10 + ') VALUES (');           // INSERT
end;

// WHERE区を追加
case iMode of
  1: slText.Add(#13#10 + 'WHERE');  // SELECT
  3: slText.Add(#13#10 + 'WHERE');  // UPDATE
  4: slText.Add('WHERE');           // DELETE
end;

// フィールド情報
for i := 0 to slFieldID.Count - 1 do
begin
  case iMode of
    2:   sText := ':' + slFieldID[i];                    // INSERT
    else sText := slFieldID[i] + ' = :' + slFieldID[i];  // 上記以外
  end;

  // 最終フィールドでない場合、カンマ or " AND"を付与
  if (i <> (slFieldID.Count - 1)) then
  begin
    case iMode of
      2:   sText := sText + ',';     // INSERT
      else sText := sText + ' AND';  // 上記以外
    end;
  end;

  // 内部保持
  slText.Add(sText);
end;

// INSERTの場合、")"を追加
if (iMode = 2) then
begin
  slText.Add(')');
end;
```

ソース 14

ParamByName、FieldByNameの自動生成

```
for i := 0 to slFieldID.Count - 1 do
begin
  // フィールド情報
  case iMode of
    5: sText := 'ParamByName';
    6: sText := 'FieldByName';
  end;
  sText := sText + '('''' + slFieldID[i] + ''').As' + slFieldType[i] + ' := ';

  // フィールドタイプ
  // 文字
  if (slFieldType[i] = 'String') then
  begin
    sText := sText + '''';';
  end
  // 数値
  else
  begin
    sText := sText + '0;';
  end;

  // 内部保持
  slText.Add(sText);
end;
```

以上で、DDSソース及びオブジェクトから、SQL文、ParamByName・FieldByNameメソッドを自動生成するプログラムは完成である。上記プログラムを実行し、サブ画面を確認する。変換対象は第2章で作成したファイル【図5】とし、ファイル・フィールド情報通りの内容がサブ画面にセットされていることが分かる【図11～16】。

図 11　SQL文の自動生成:実行結果(SQL文:SELECT)

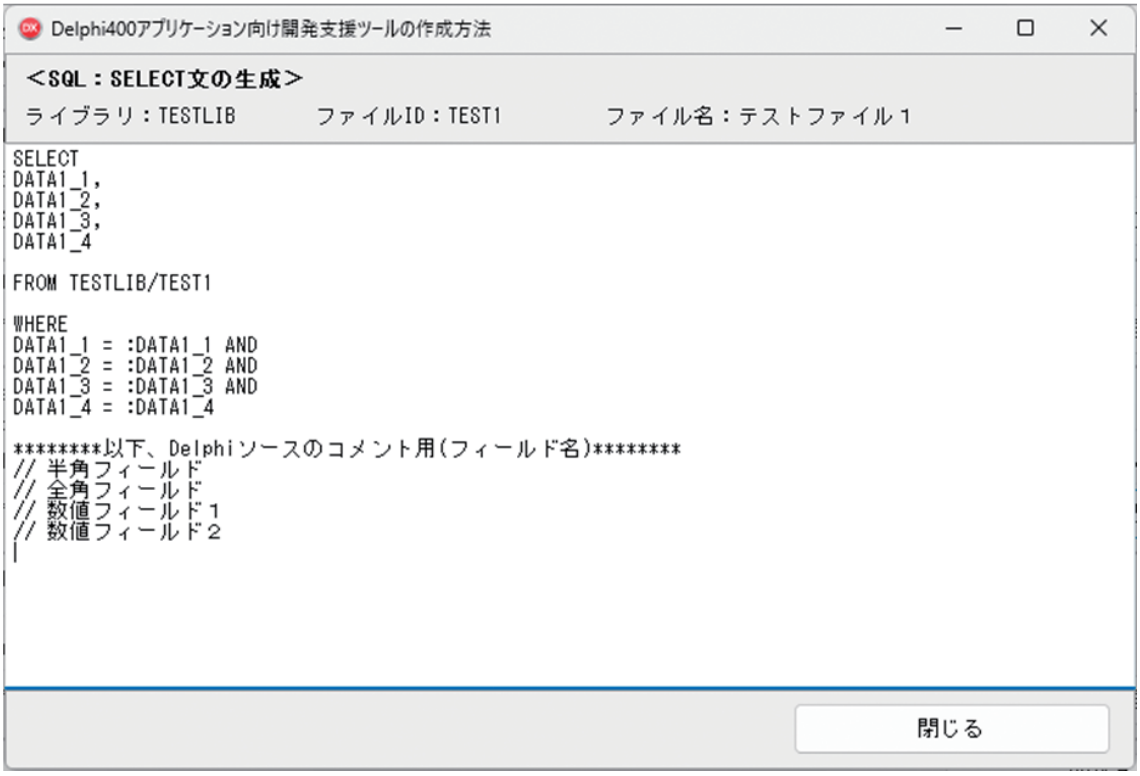


図 12　SQL文の自動生成:実行結果(SQL文:INSERT)

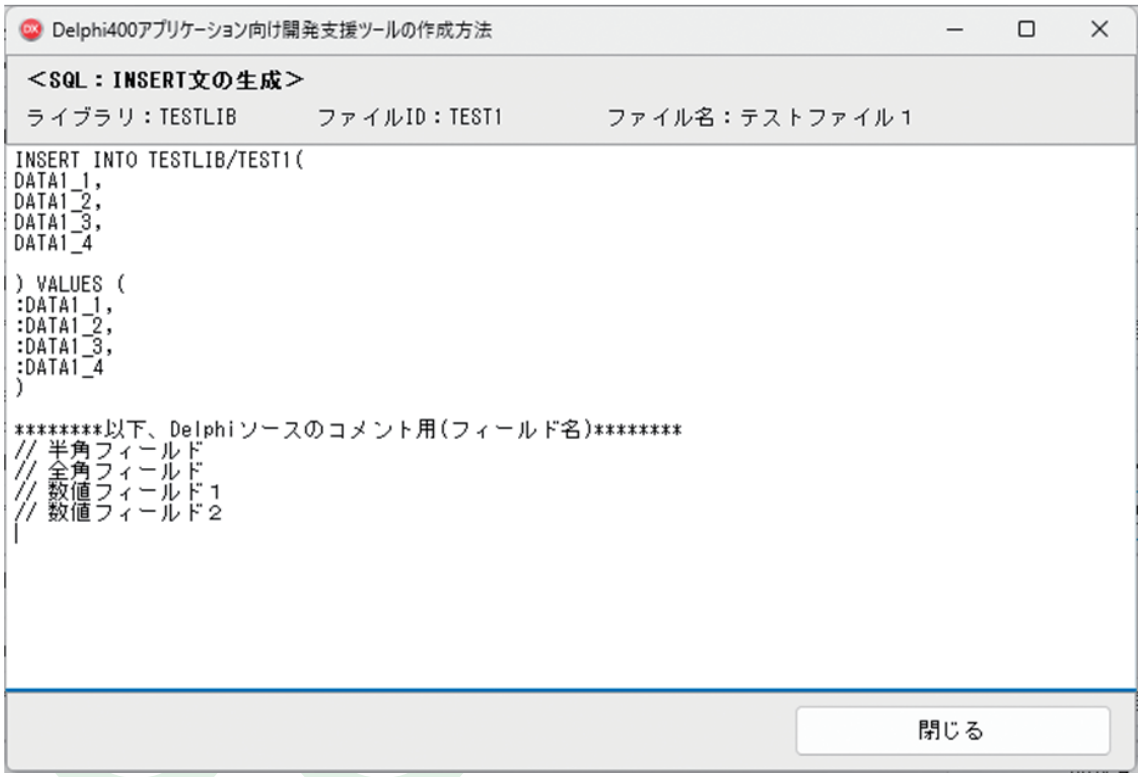


図 13 SQL文の自動生成:実行結果(SQL文:UPDATE)

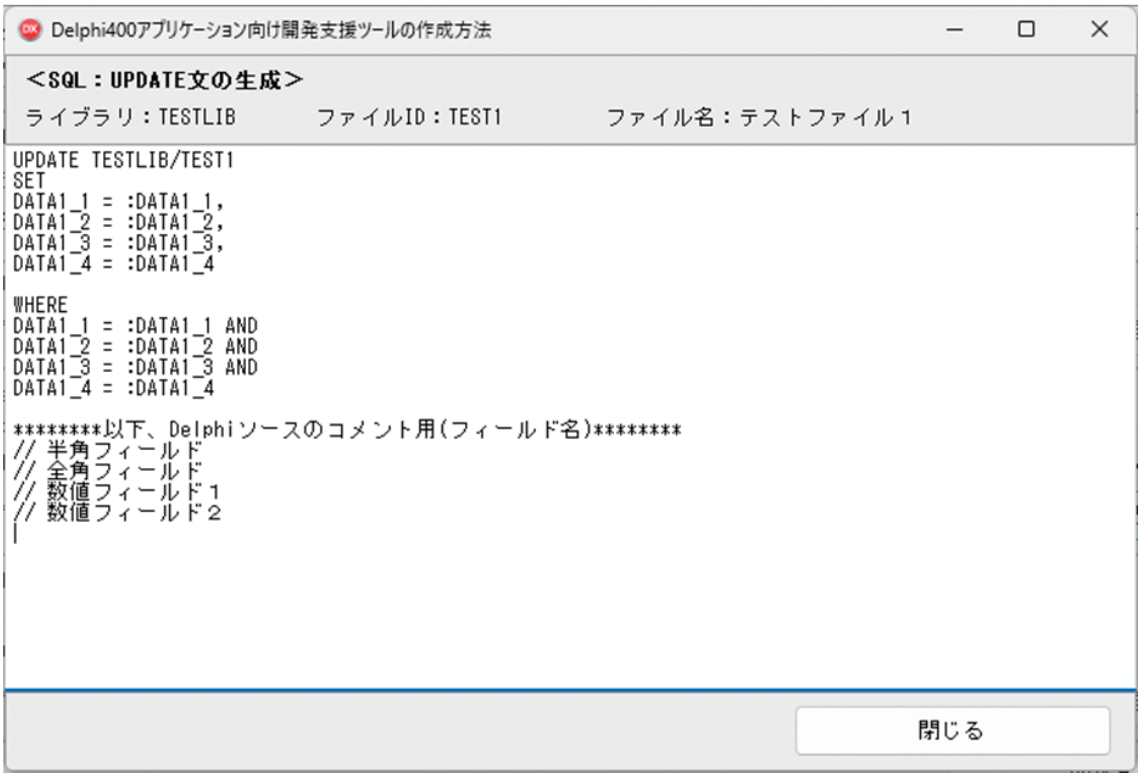


図 14 SQL文の自動生成:実行結果(SQL文:DELETE)

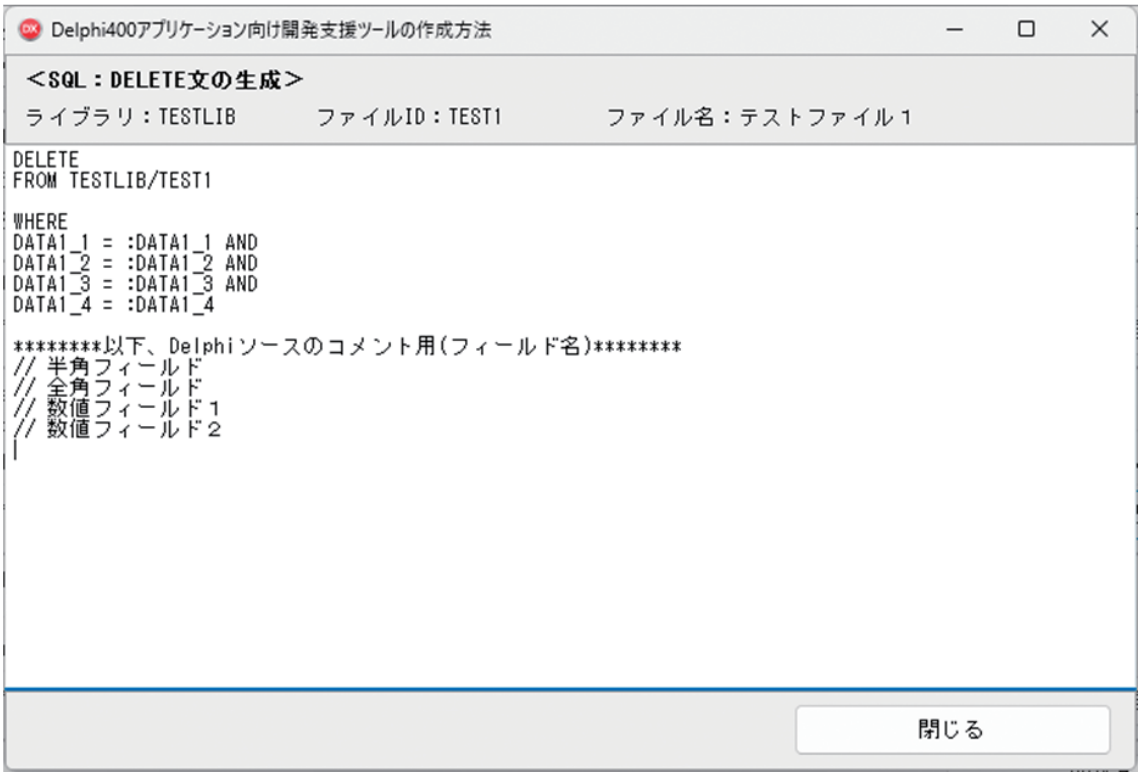


図 15　SQL文の自動生成:実行結果(ParamByName)

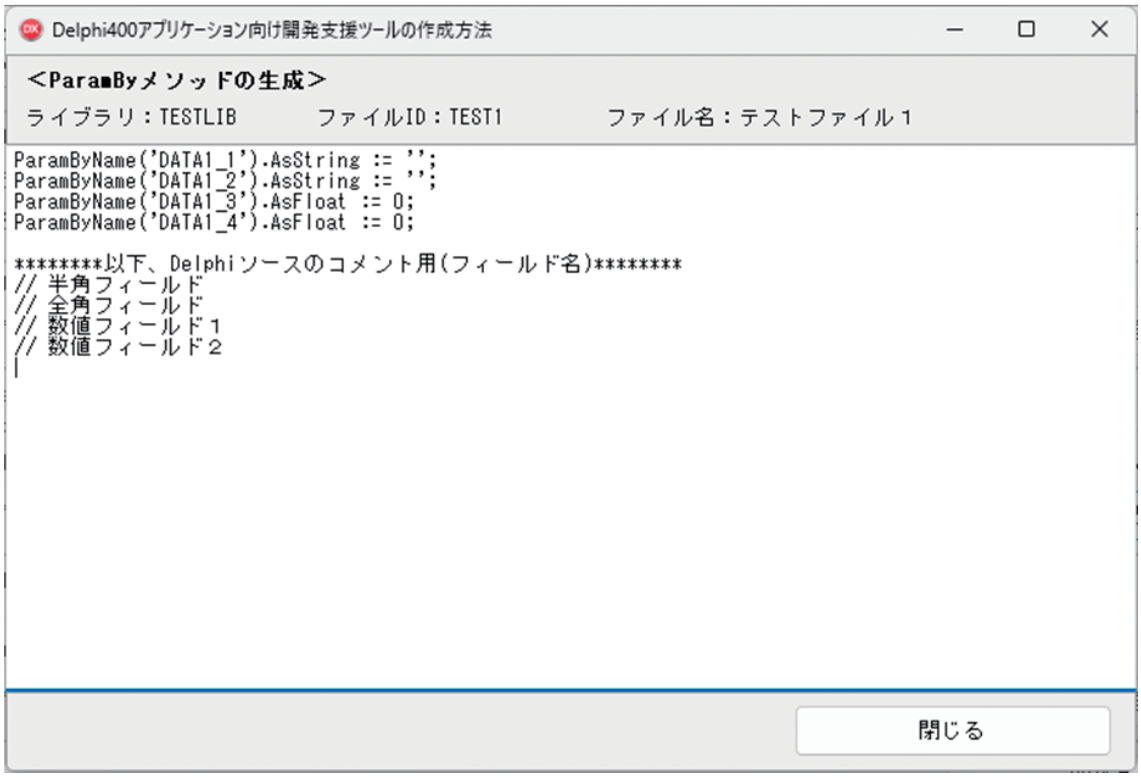
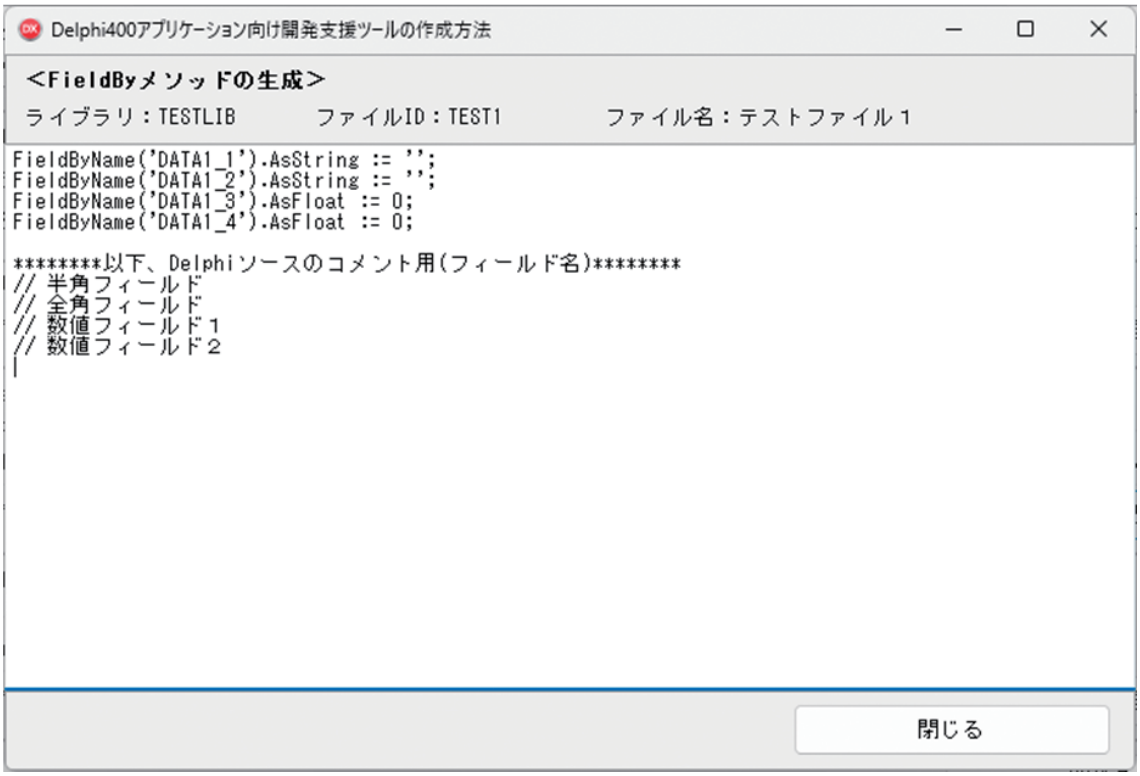


図 16　SQL文の自動生成:実行結果(FieldByName)



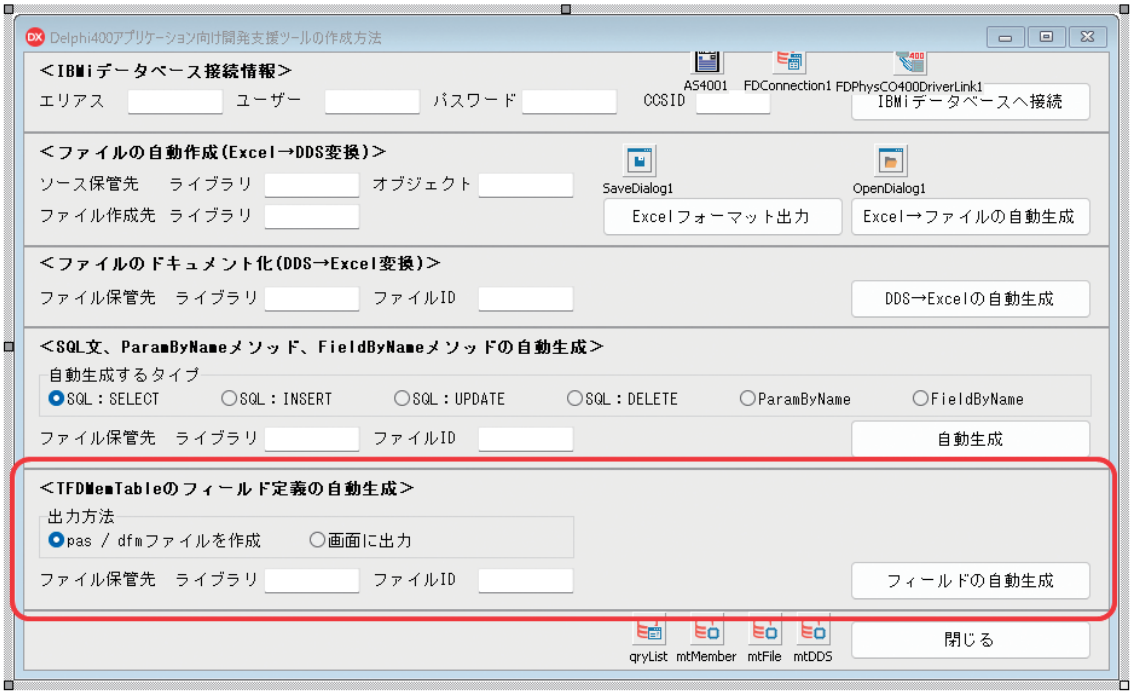
5.TFDMemTableのフィールド定義の自動生成

本章では、IBMiデータベース上に作成されたDDSソース及びオブジェクトから、TFDMemTableで使用するフィールド定義を自動生成する方法を紹介する。
TFDMemTableのフィールド定義の生成方法は、自動取込と手動作成の2パターンがある。自動取込の場合、①必要なフィールドを取得するSQL文を記述したTFDQueryを配置する。②TFDQueryをOpenし、フィールドエディタですべてのフィールドを取り込む。③TFDMemTable上で右クリック→メニューの「データセット割り当て」を選択→①のTFDQueryを選択する。④TFDMemTable上で右クリック→メニューの「フィールドエディタ」を選択→TFDQueryを選択する。⑤「フィールドエディタ」上で右クリック→メニューの「すべてのフィールドを追加」を選択し、フィールド定義を作成する。手動作成の場合、DDSソースを確認、もしくは、オブジェクトの情報を確認し、フィールド定義を作成する。
本章で紹介する機能は、オブジェクト保管先の情報を入力すれば自動生成できるため、作業の効率化に繋がるだろう。

5-1.コンポーネントの配置

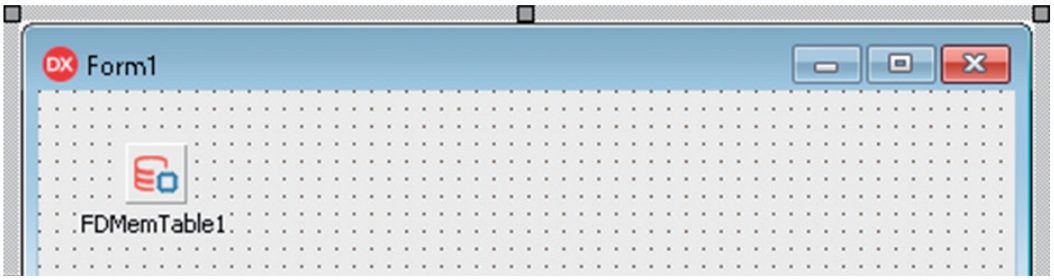
第4章で作成したサンプルプログラムに、出力方法を選択するためのTRadioGroup、ファイル保管先の情報を設定するためのTEdit、並びにTLabelやTButtonを画面に追加配置する【図17】。

図 17　フィールド定義自動生成:コンポーネントの配置



「フィールドの自動生成」ボタンのOnClick処理で、TFDMemTableのフィールド定義の自動生成処理を記述する。pas/dfmファイルを作成時用に、TFDMemTableのみを配置した雛形フォームを新規作成しておく【図18】。

図 18　フィールド定義自動生成：雛形となるDelphi/400画面



5-2.雛形フォームの保存

画面項目「出力方法」で「pas/dfmファイルを作成」を選択時、5-1.で作成した雛形フォームをTSaveDialogで指定の保管先へ保存する処理を記述する【ソース15】。

ソース 15

雛形フォームの保存

```
// 保存名の初期値
SaveDialog1.FileName   := 'Unit1';
SaveDialog1.DefaultExt := '.pas';
SaveDialog1.Filter     := 'Delphi ユニット (*.pas)|*.pas';

// 保存ダイアログの実行
if (not SaveDialog1.Execute) then
begin
  Abort;
end;
```

5-3.フィールド定義の生成

第4章の4-2.①【ソース12】の方法で、画面項目「ファイル保管先」で指定のファイルのフィールド情報をリストに内部保持する。内部保持した情報を基に、pasファイル用、dfmファイル及び画面出力用のTFDMemTableのフィールド定義を内部保持する処理を記述する【ソース16】。

ソース 16

フィールド定義を内部保持する処理

```
// dfmのスペース
case rgMemTableOut.ItemIndex of
  0:    sSpace := '    ';  // pas / dfmファイルを作成
  else sSpace := '';       // 画面に出力
end;

// フィールド定義の生成
for i := 0 to slFieldID.Count - 1 do
begin
  sName      := 'FDMemTable1' + slFieldID[i];    // 名前
  sFieldType := 'T' + slFieldType[i] + 'Field';  // フィールドタイプ

  // pasファイル
  slText_pas.Add('    ' + sName + ': ' + sFieldType + ';');

  // dfmファイル
  slText_dfm.Add(sSpace + 'object ' + sName + ': ' + sFieldType);          // 名前&フィールドタイプ
  slText_dfm.Add(sSpace + '  DisplayLabel = ''' + slFieldName[i] + '''');  // 表示ラベル
  slText_dfm.Add(sSpace + '  FieldName = ''' + slFieldID[i] + '''');       // フィールド名
  // 桁数(文字のみ)
  if (slFieldType[i] = 'String') then
  begin
    slText_dfm.Add(sSpace + '  Size = ' + slFieldLen[i] + '');
  end;
  slText_dfm.Add(sSpace + 'end');
end;
```

5-4.自動生成した内容を保存

画面項目「出力方法」で「pas/dfmファイルを作成」を選択時、5-2.で保存したpasファイル、dfmファイルに、5-3.で内部保持した情報を追記し、上書き保存する【ソース17】。画面項目「出力方法」で「画面に出力」を選択時、第4章の4-2.で作成したサブ画面【図10】に表示する。

ソース 17

フィールド定義を保存：pas/dfmファイルを作成

```
// コピー元ファイルの保管先
sFrom := ExtractFilePath(Application.ExeName) + 'Template';
sFrom := IncludeTrailingPathDelimiter(sFrom)  + 'Unit1.pas';

// 保存ファイル名（拡張子なし）
sSaveFileName := ExtractFileName(SaveDialog1.FileName);
sSaveFileName := StringReplace(sSaveFileName, '.pas', '', [rfReplaceAll]);

// 【pasファイル】
// コピー元ファイルの読み込み
slPas.LoadFromFile(sFrom);

// TFDMemTableの宣言部へ移動し、その後ろにフィールド情報を追加
iIndex := slPas.IndexOf('    FDMemTable1: TFDMemTable;');
for i := 0 to slText_pas.Count - 1 do
begin
  slPas.Insert(iIndex + i + 1, slText_pas[i]);
end;

// Unit1→ファイル保存ダイアログで指定の名前に変更
slPas.CommaText := StringReplace(slPas.CommaText, 'Unit1', sSaveFileName, [rfReplaceAll]);

// 保存
slPas.SaveToFile(SaveDialog1.FileName);

// 【dfmファイル】
// 拡張子を「dfm」に変更
SaveDialog1.FileName := System.SysUtils.ChangeFileExt(SaveDialog1.FileName, '.dfm');
sFrom                := System.SysUtils.ChangeFileExt(sFrom, '.dfm');

// コピー元ファイルの読み込み
slPas.LoadFromFile(sFrom);

// 末尾2行目にフィールド情報を追加
for i := 0 to slText_dfm.Count - 1 do
begin
  slPas.Insert(slPas.Count - 2, slText_dfm[i]);
end;

// 保存
slPas.SaveToFile(SaveDialog1.FileName);
```

以上で、DDSソース及びオブジェクトから、フィールド定義を自動生成するプログラムは完成である。上記プログラムを実行し、生成されたpas/dfmファイル及びサブ画面を確認する。変換対象は第2章で作成したファイル【図5】とし、ファイル・フィールド情報通りの内容がファイル保存、もしくはサブ画面にセットされていることが分かる【図19〜23】。

図 19　フィールド定義自動生成：実行結果（pasファイル）

```
unit Unit1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, FireDAC.Stan.Intf, FireDAC.Stan.Option,
  FireDAC.Stan.Param, FireDAC.Stan.Error, FireDAC.DatS, FireDAC.Phys.Intf,
  FireDAC.DApt.Intf, Data.DB, FireDAC.Comp.DataSet, FireDAC.Comp.Client;

type
  TForm1 = class(TForm)
    FDMemTable1: TFDMemTable;
    FDMemTable1DATA1_1: TStringField;
    FDMemTable1DATA1_2: TStringField;
    FDMemTable1DATA1_3: TFloatField;
    FDMemTable1DATA1_4: TFloatField;
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

自動生成されたフィールド情報

図 20　フィールド定義自動生成：実行結果（dfmファイル）

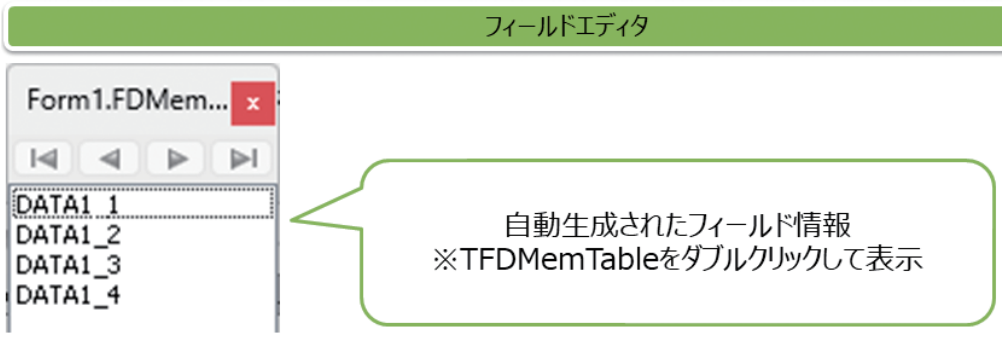
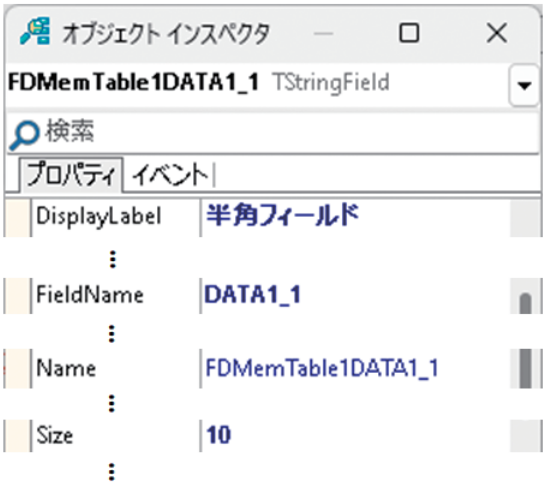


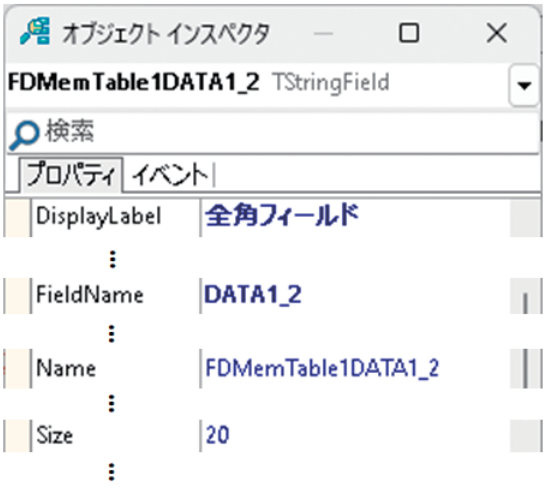
図 21　フィールド定義自動生成：実行結果（dfmファイル）

オブジェクトインスペクタ　※サンプルプログラムから設定したプロパティのみピックアップ

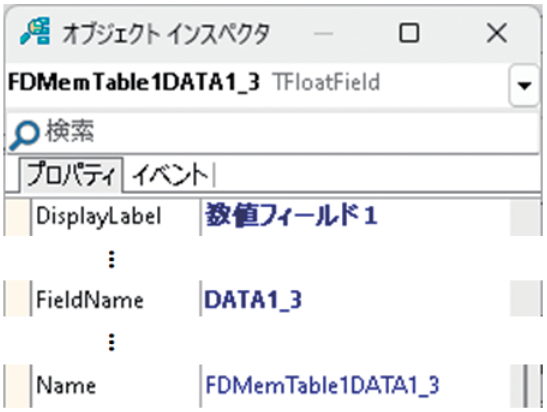
フィールド：DATA1_1



フィールド：DATA1_2



フィールド：DATA1_3



フィールド：DATA1_4

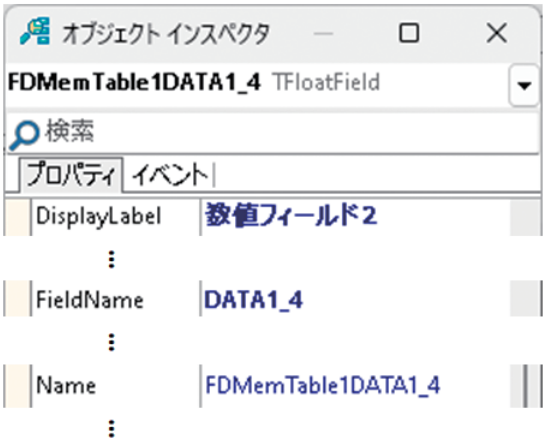


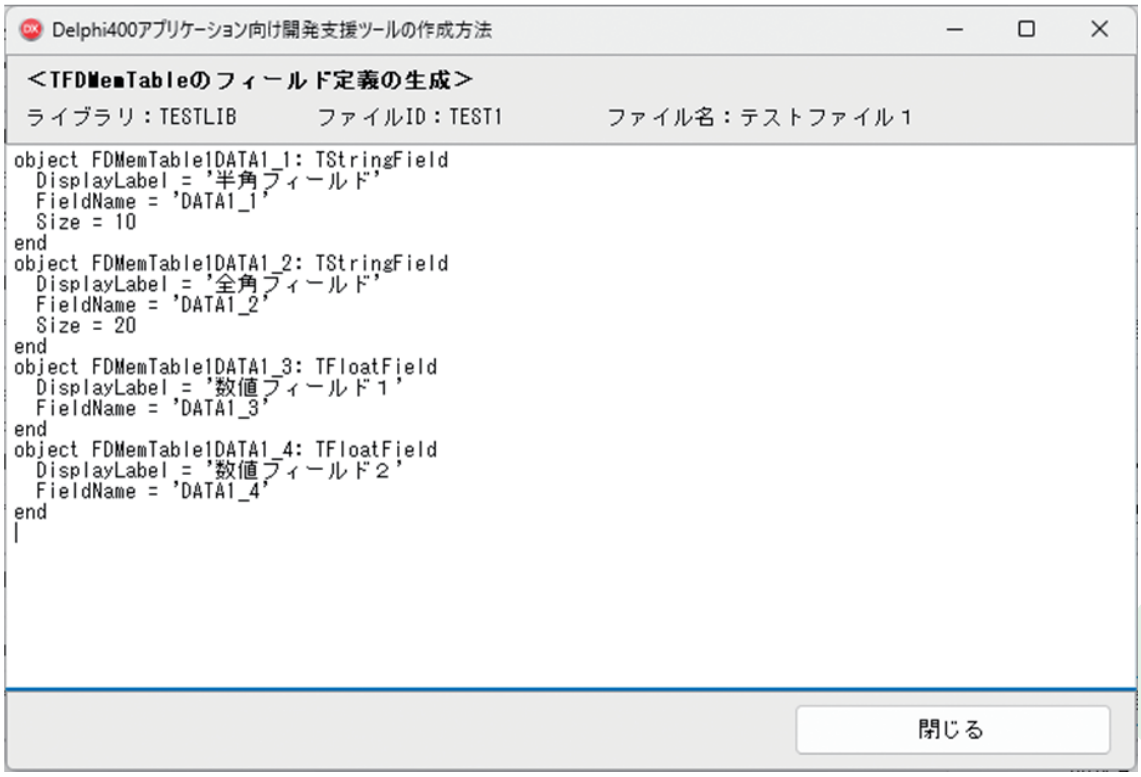
図 22 フィールド定義自動生成：実行結果（dfmファイル）

フィールドエディタ

```
object Form1: TForm1
  Left = 0
  Top = 0
      :
    object FDMemTable1DATA1_1: TStringField
      DisplayLabel = #21322#35282#12501#12451#12540#12523#12489
      FieldName = 'DATA1_1'
      Size = 10
    end
    object FDMemTable1DATA1_2: TStringField
      DisplayLabel = #20840#35282#12501#12451#12540#12523#12489
      FieldName = 'DATA1_2'
    end
    object FDMemTable1DATA1_3: TFloatField
      DisplayLabel = #25968#20516#12501#12451#12540#12523#12489#65297
      FieldName = 'DATA1_3'
    end
    object FDMemTable1DATA1_4: TFloatField
      DisplayLabel = #25968#20516#12501#12451#12540#12523#12489#65298
      FieldName = 'DATA1_4'
    end
  end
end
```

自動生成されたフィールド情報

図 23 フィールド定義自動生成：実行結果（画面出力）



6.さいごに

本稿では、Delphi/400アプリケーション向けの開発支援ツールの作成例を紹介した。今回紹介した内容を参考に、必要な機能を追加するなど各自に合ったものにカスタマイズして頂くことも可能である。本稿を参考に、アプリケーション開発の効率化に役立てて頂ければ幸いである。

なお、今回紹介した開発支援ツールのソース一式を以下よりダウンロード可能なので、是非活用して頂きたい。

https://www.migaro.co.jp/d4sample/ntsuji2023.zip

（※ダウンロードには、Delphi/400メンテナンスページへのログインユーザー・パスワードが必要）