【A2】Delphiテクニカルセッション

Delphiで作るデータベースツール。その開発のポイントは‥

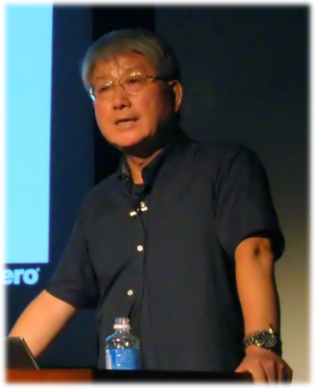
2014年11月11日

田中 芳起

Ver.1.0.0

自己紹介

名前：田中 芳起（たなか よしき）



- 中堅SIerでパッケージシステムの開発／プロジェクト管理／品質管理 等の仕事に従事
- 26th デブキャンプで「はじめてのFireDAC」の講師を担当
- Delphiとは 1.0US版 からの付き合い

- ホームページ ：http://www.avsoft.jp/
- ブログ ：http://avsoft.typepad.jp/blog/
- Facebook ：https://www.facebook.com/yoshiki.tanaka.942/
  ：https://www.facebook.com/VisualNavi

Agenda

➢ Introduction

- DB-Enginesによる調査結果
- Visual NAVI のご紹介
- おもな機能
- 動作環境
- デモ
- Oracle Databaseの歴史
- Oracle社が提供するミドルウェア
- Oracle接続の仕組み
- BDE(Borland Database Engine)の構造
- BDEの現状
- ネイティブ接続
- リポジトリを使う‥
- Frameを使う‥

Agenda

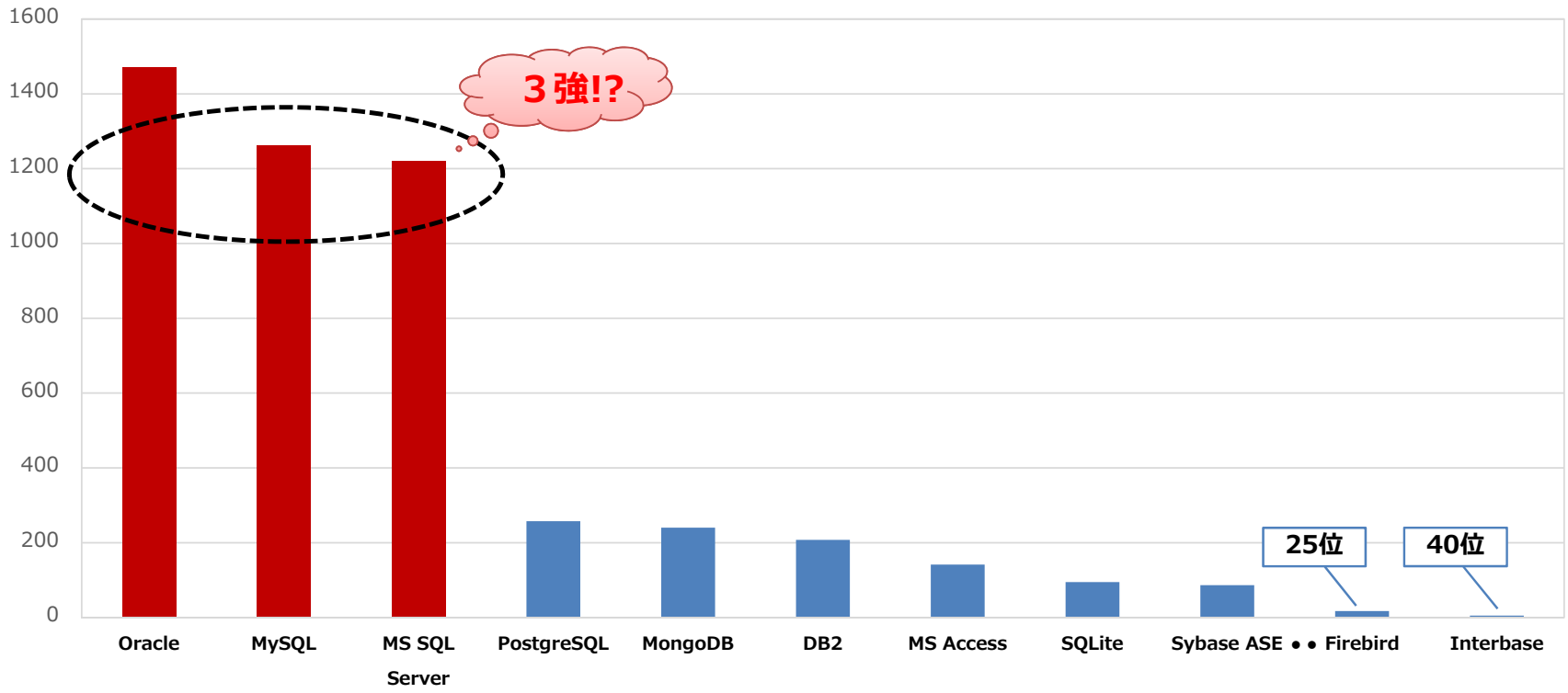
➢ The Next Steps

- 今後の進化
- マルチデータベースのサポート
- ソフトウェア構造の根本的な見直し
- 無謀にも‥ VCLからFireMonkey(FMX)フォームを呼び出す
- DLL内でMDI子フォームを生成する
- FireMonkeyの状態保存
- FireMonkeyでクリップボードを扱う
- コア技術は、コールバック関数！
- コールバック関数の体系
- コールバック関数の登録
- コールバック関数の呼出
- SQLの実行
- Visual NAVI のSQLエディタを起動・実行する
- パイプを使ってメッセージを横取りする！
- デモ

Introduction

DB-Enginesによる調査結果

データベースソフトウェアの普及度や人気を、インターネット上の求人情報や職務経歴上での経験、および検索エンジンやSNSでの情報量を元に毎月作成し公開されている。
http://db-engines.com/en/



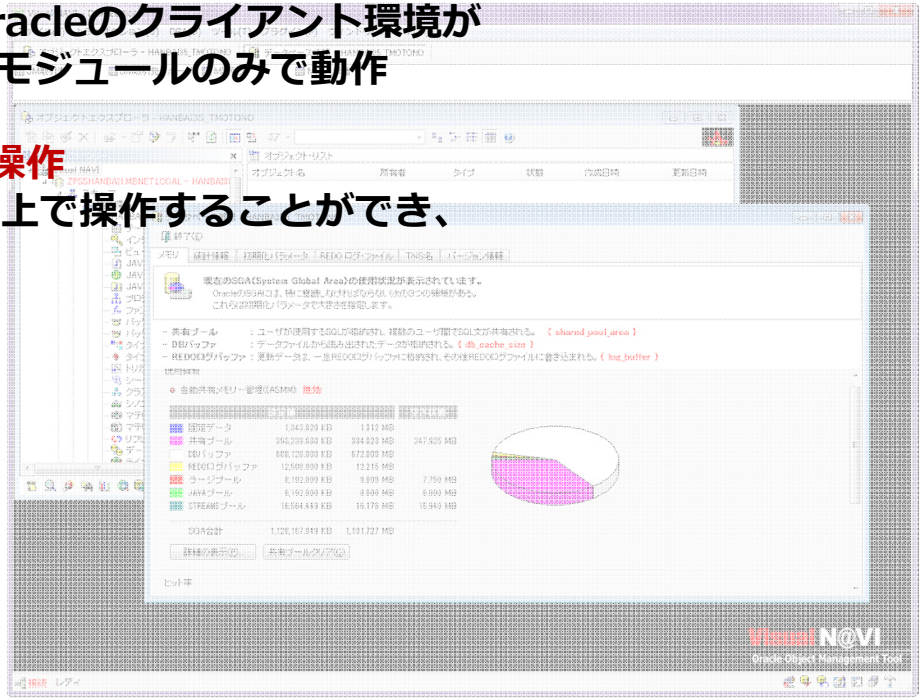
Visual NAVI のご紹介

コンセプトは **Easy to use ！**（使いやすさ）

- **Oracle専用の統合型開発支援ツール**
  アプリケーション開発に必要な機能やデータベース管理に必要な機能を統合

- **ネイティブ接続**
  Oracleとの接続は OCI (Oracle Call Interface) を使用。
  接続用のミドルウェアは一切必要とせず、Oracleのクライアント環境が
  インストールされている端末であれば、実行モジュールのみで動作

- **GUI (Graphical User Interface) による操作**
  オブジェクトの作成、SQLの実行などをGUI上で操作することができ、
  開発効率が大幅に向上



おもな機能

- **DBA向け機能**
  各種データベース情報の表示
  表領域、ユーザー等の新規作成／類似作成／変更／削除

- **高機能なSQL実行機能**
  SQL実行(DDL、DML)
  SQL文の解析､エラー個所(行、位置)の表示
  バインド変数を使用したSQLの実行
  DBMS_OUTPUTパッケージを使用したデバッグ機能
  実行計画のグラフィカル表示
  SQLのバッチ実行

- **スキーマ・オブジェクトの管理機能**
  GUIによるオブジェクトの新規作成／類似作成／変更／削除
  オブジェクトの定義情報、ソース・データをリバース表示
  オブジェクト一覧表、テーブル・ビューの仕様書出力

- **ストアドプログラムの作成・編集・実行**
  専用エディタによるストアドプログラムの作成／コンパイル／実行

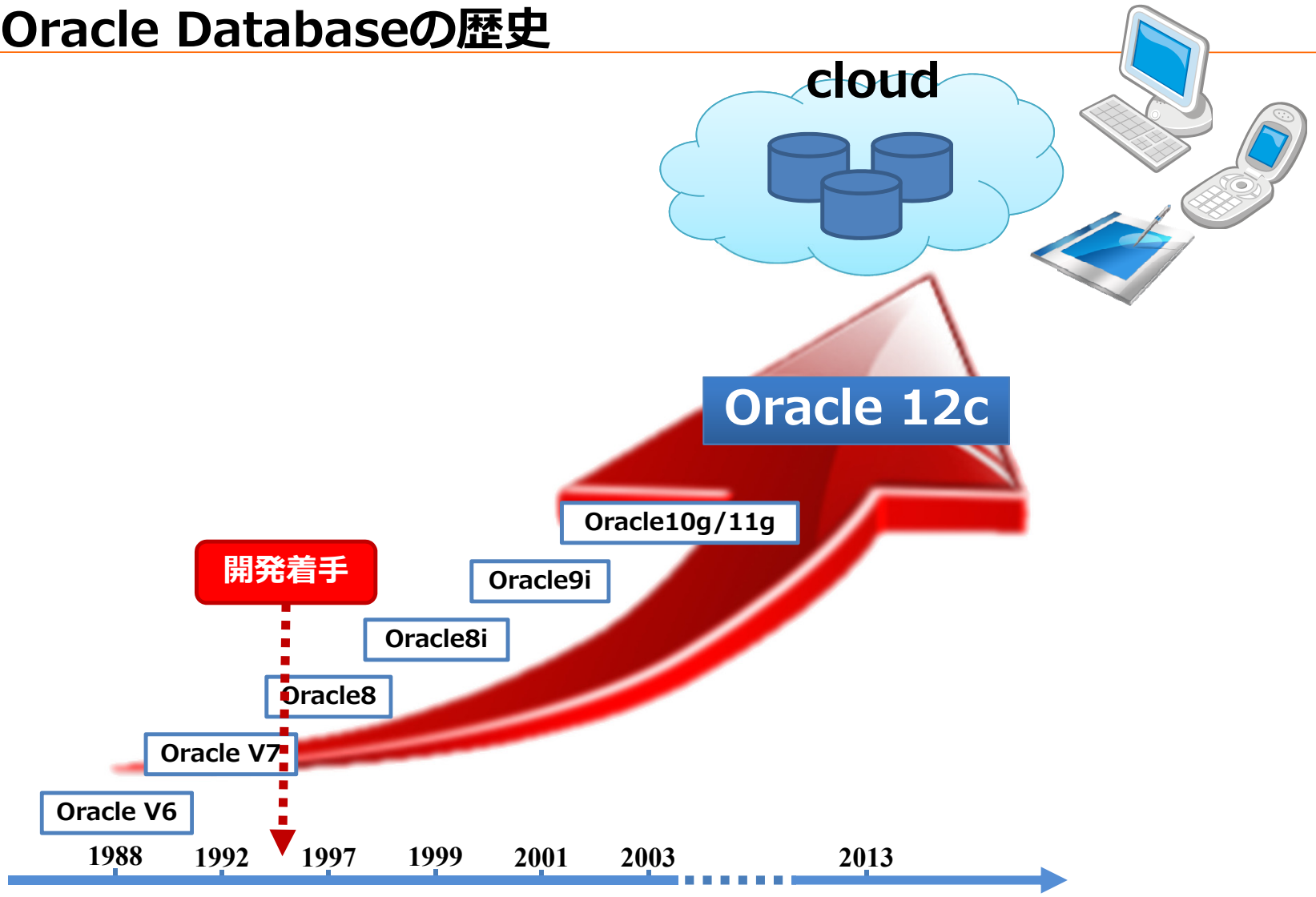
- **テーブル、ビューの表示・編集機能**
  行の編集／追加／削除／全行削除
  各種形式(XML／HTML／Excel／CSV‥)でのデータ出力

動作環境

- **パソコンの動作環境**
  - Windows日本語版
  - Vista／7／8(8.1)（32bit／64bit共に可）

- **オラクルのバージョン／Excelのバージョンについて**
  - Oracleクライアントがインストール済であること
  - Oracleクライアントと接続するOracleサーバが以下のバージョンであること
  - Oracle 9.0.1 ～ 12.1.0（64bit Oracleには未対応）
  - 文字コードはSJISのみ（UniCodeは未対応）
  - 仕様書出力・Excel形式でデータを出力する場合は、Microsoft Excel（97以降）がインストール済みであること

Oracle Databaseの歴史



Oracle接続の仕組み

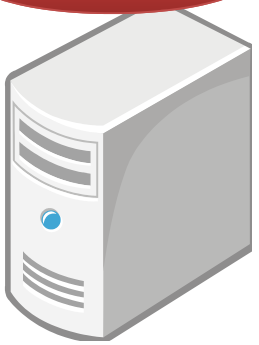
tnsnames.ora

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)
      (HOST = host1)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = orcl)
    )
  )
```

ネットサービス名

ネットワーク
定義ファイル

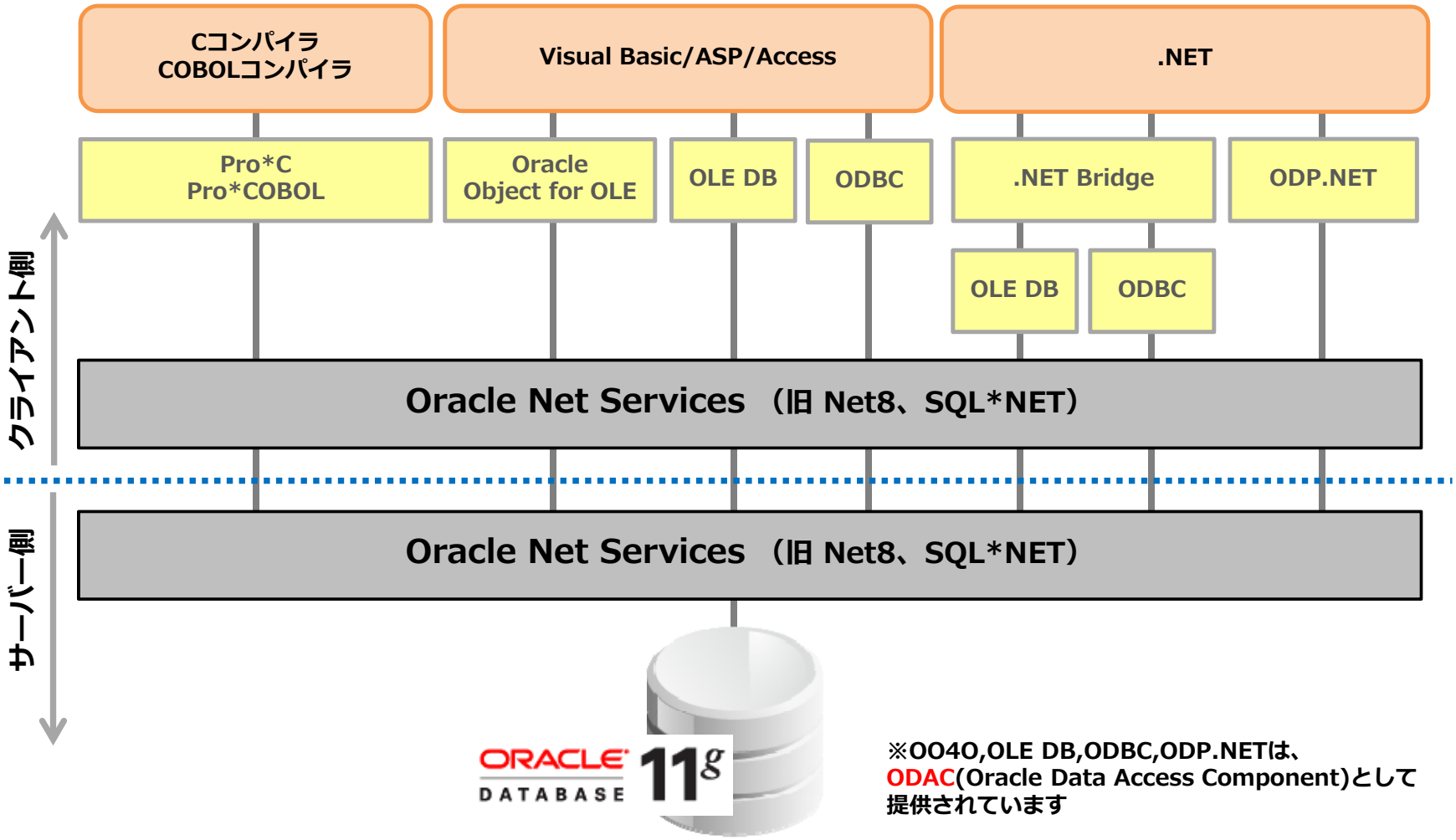
リスナー



host1

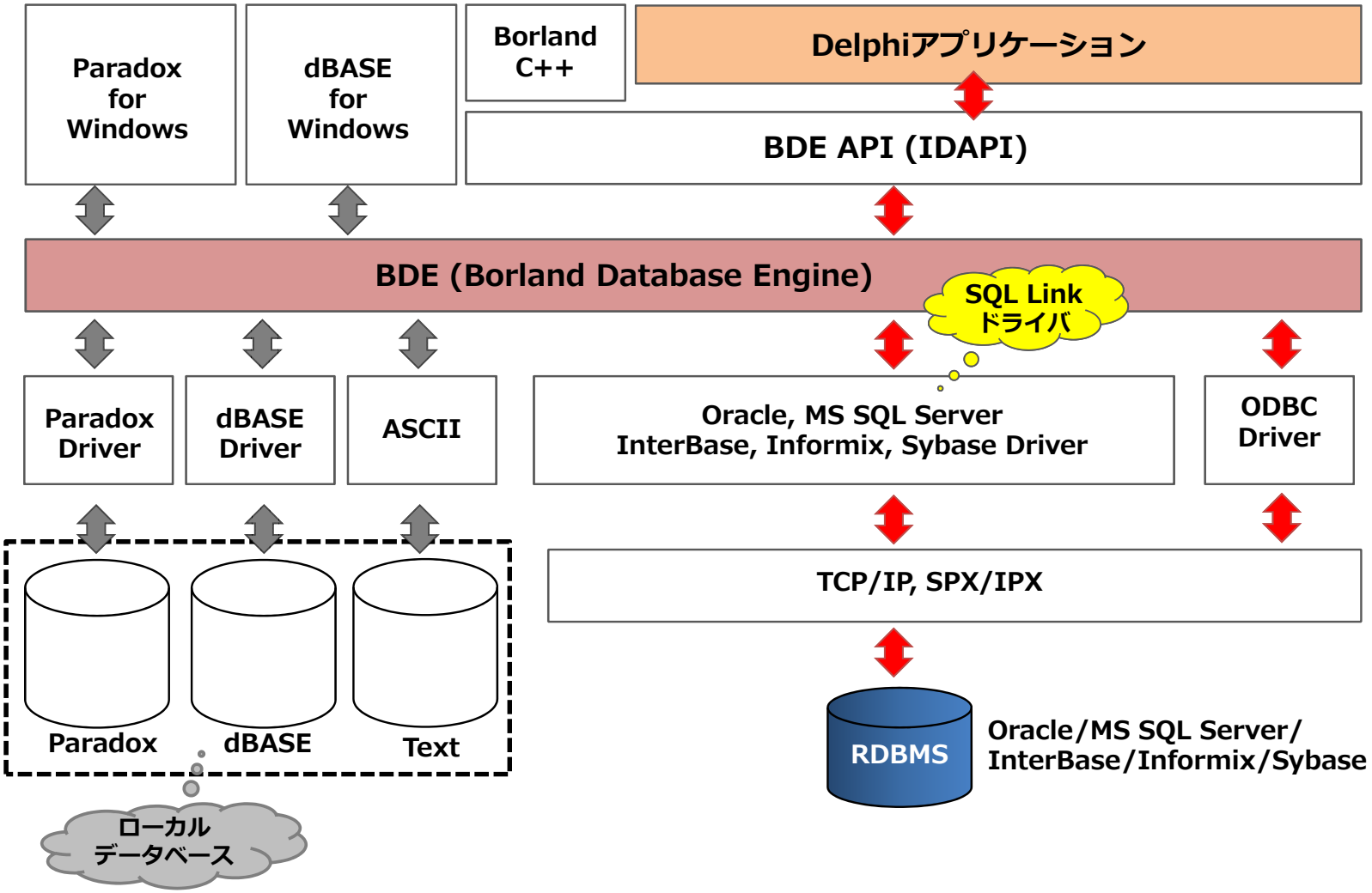
Connect
ユーザー名／パスワード@ORCL

Oracle社が提供するミドルウェア



※OO4O,OLE DB,ODBC,ODP.NETは、
ODAC(Oracle Data Access Component)として
提供されています

BDE(Borland Database Engine)の構造



BDEの現状

- **BDEは2002年に開発・保守が終了**
  不具合があっても修正パッチの提供はされていない

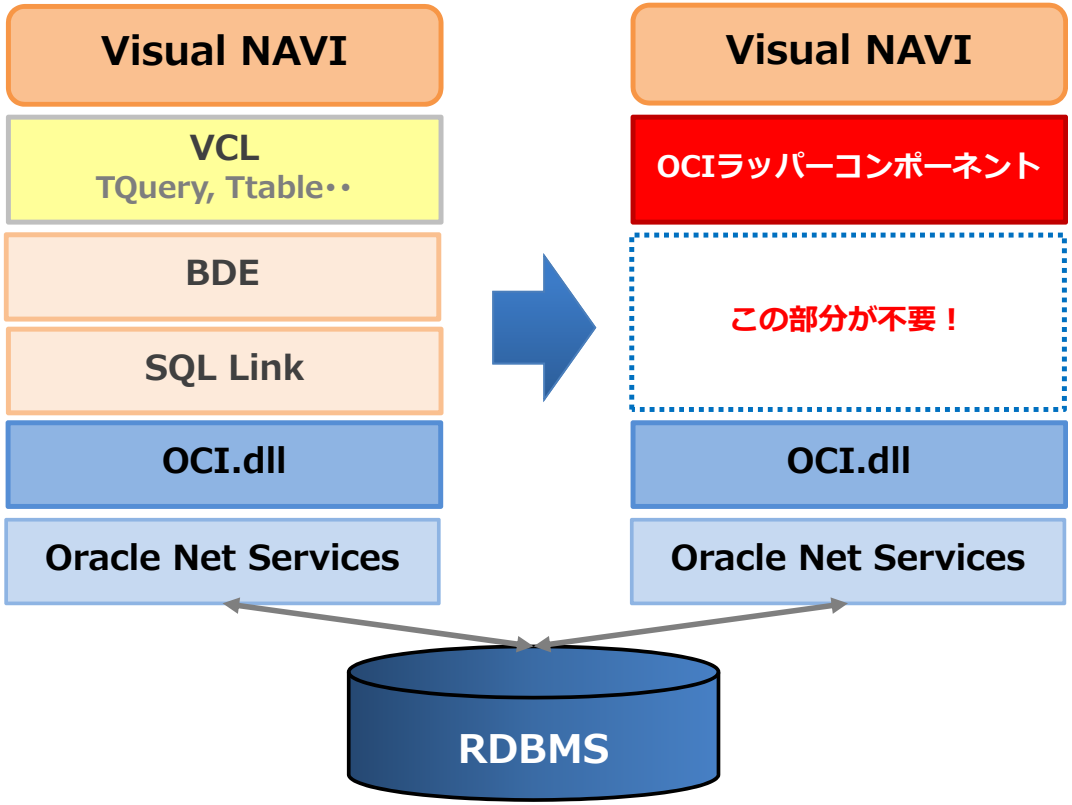
- **BDEの最新バージョンは 5.2**
  Delphi 7／C++Builder 6以降、BDEのバージョンは更新されていない
  現在のDelphi／C++Builder製品に付属するBDEは、あくまで過去の資産保守用

- **動作保証プラットフォームは Windows XPまで**
  Windows XP以降にリリースされたOSバージョンは動作保証がされていない
  （例えば、Windows Vista／7／8、Windows2003／2008 Serverなど）

- **RAD Studio XE7ではBDEがインストールされない**
  しかし、別途インストーラがダウンロードできます‥
  http://support.embarcadero.com/jp/print/44077

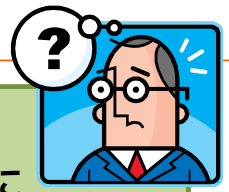
ネイティブ接続

- Delphi用OCI*1 ラッパーコンポーネントを開発
- ODBC、ADO、OO4O、BDE等は一切不要
- Oracleのすべての機能が使用可能
- 処理が高速



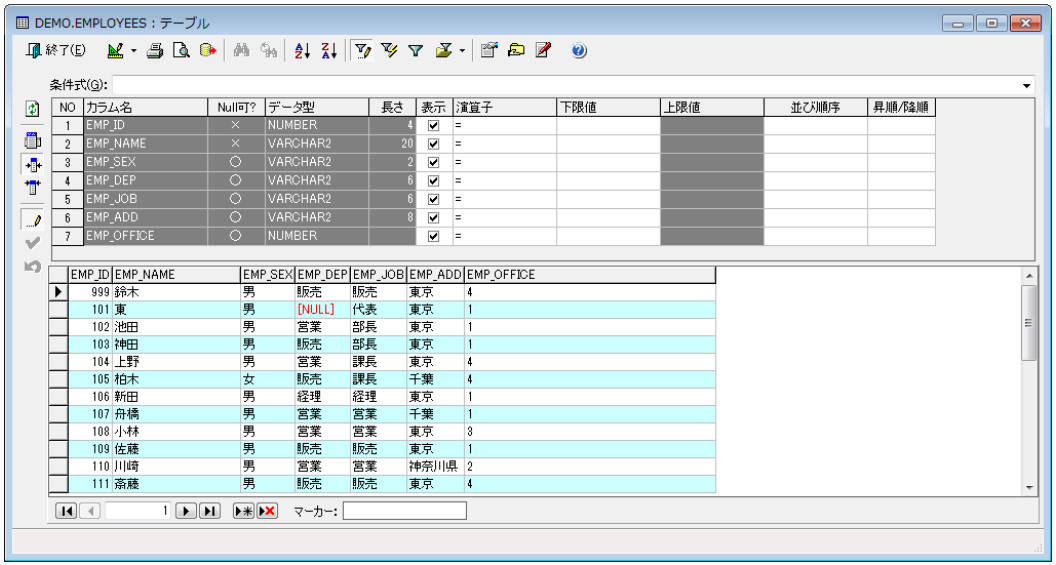
*1 OCI（Oracle Call Interface）はOracleデータベースが提供するAPI（Application Program Interface）の1つです。

リポジトリを使う・・



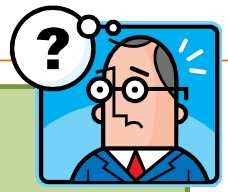
リポジトリとは・・?
Delphiのソフトウェア資産の再利用のための機能で、Form等で共通した機能を共有するのために使用する。また、チームで開発する場合はテンプレートとしてリポジトリの共有が行える。

frmBaseClassMDIFormを継承



- Form情報の読込／保存
- TaskBarの追加／削除
：

Frameを使う・・

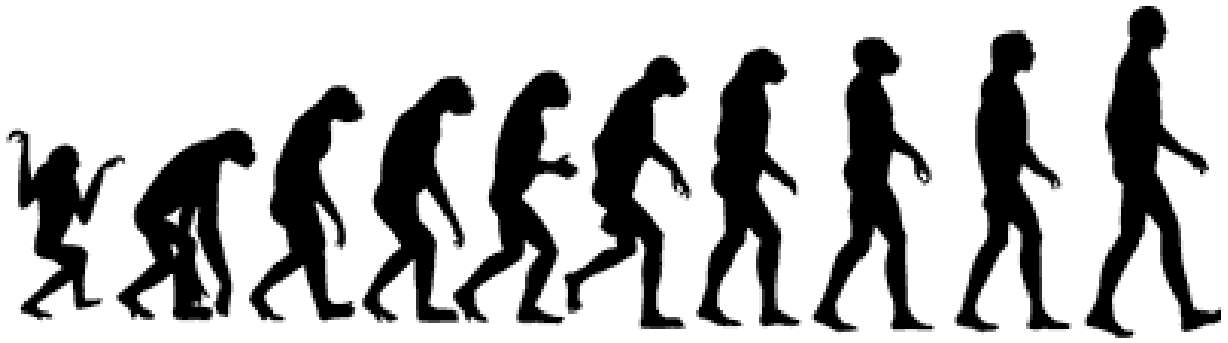


フレーム(Frame)とは・・?
Formと同じように、他のコンポーネントのコンテナ。複数のコンポーネントをまとめて配置し、部品化することができる。

EMP_ID	EMP_NAME	EMP_SEX	EMP_DEP	EMP_JOB	EMP_ADD	EMP_OFFICE
999	鈴木	男	販売	販売	東京	4
101	東	男	[NULL]	代表	東京	1
102	池田	男	営業	部長	東京	1
103	神田	男	販売	部長	東京	1
104	上野	男	営業	課長	東京	4
105	柏木	女	販売	課長	千葉	4
106	新田	男	経理	経理	東京	1
107	舟橋	男	営業	営業	千葉	1
108	小林	男	営業	営業	東京	3
109	佐藤	男	販売	販売	東京	1
110	川崎	男	営業	営業	神奈川県	2
111	斎藤	男	販売	販売	東京	4

1 マーカー:

The Next Steps

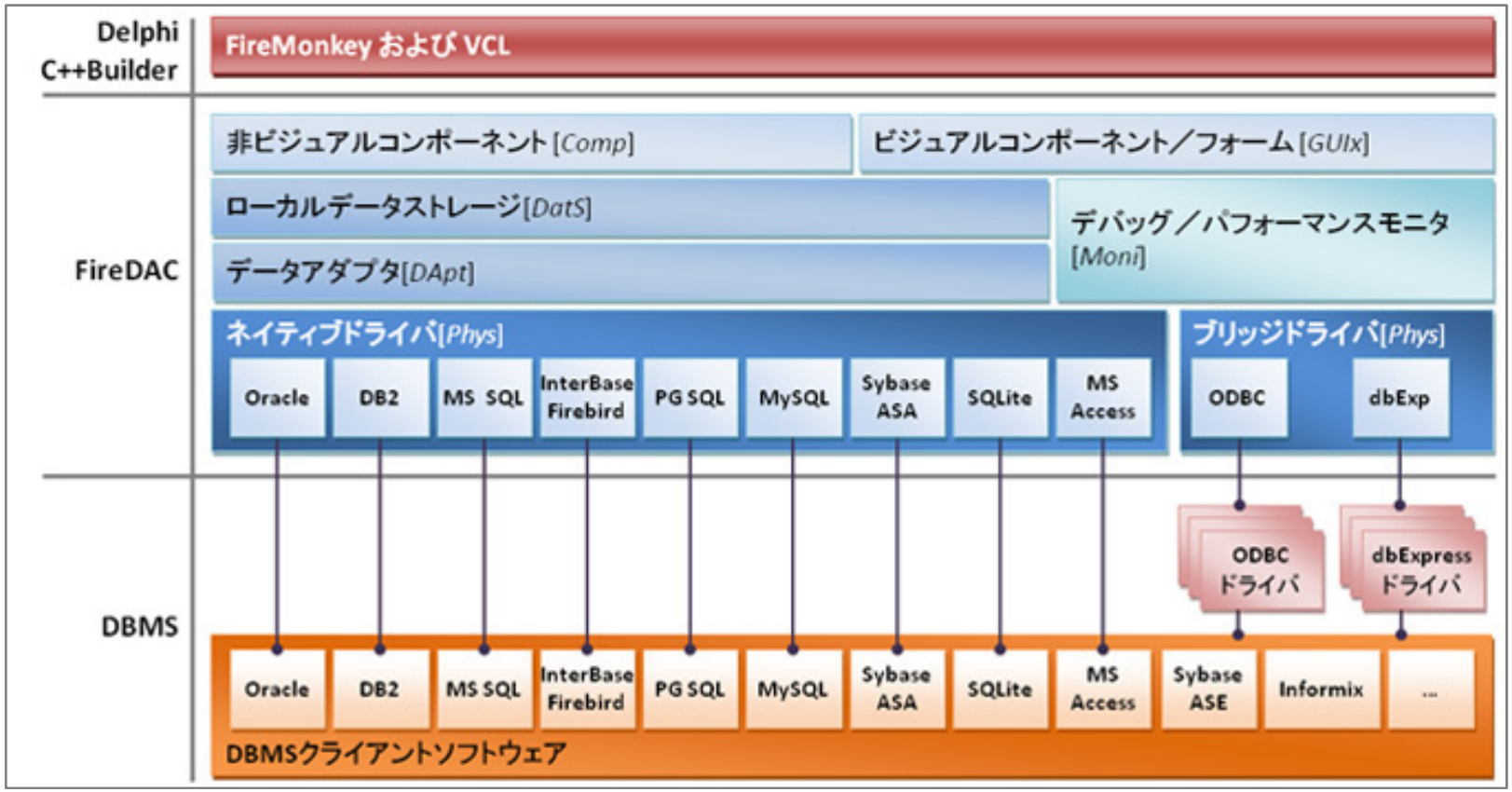


今後の進化

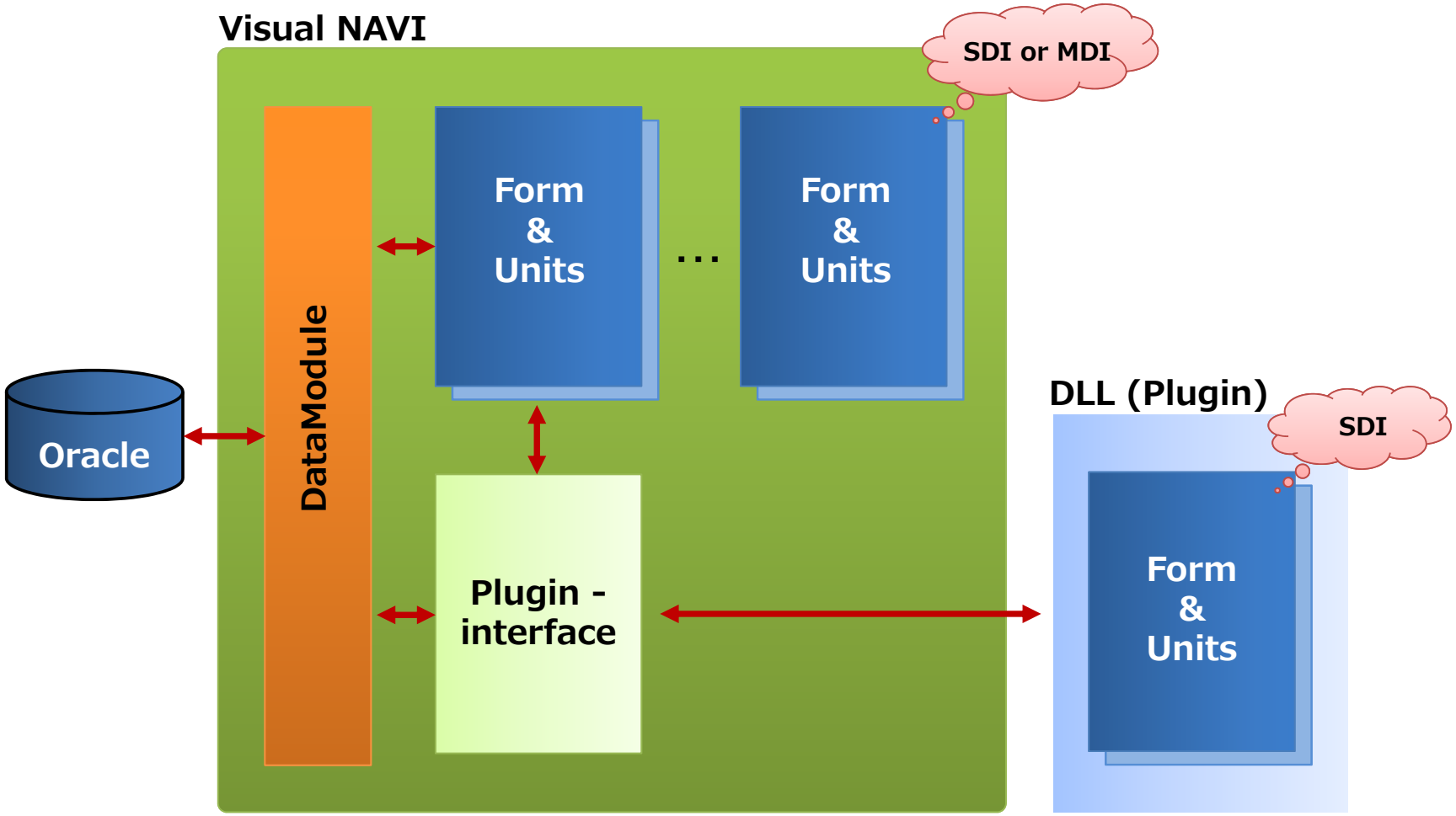
- **マルチデータベースのサポート**
  あらゆるデータベースのサポート（Oracle、MS-SQL Server、MySQL ・・）
- **ソフトウェア構造の根本的な見直し**
  ソフトウェア資産の流用と拡張性への対応
- **各種プログラミング言語への対応**
  Delphi以外の言語への対応（課題は、bool、int 、string ・・）
- **64ビット化**
  データベースも急激に32ビットから64ビットにシフトしている
- **DataSnapによる多層化**
  インフラ、プラットフォームの変化に柔軟に対応
- **マルチデバイスへの対応**
  Mac、スマートディバイス等の普及拡大に対応
- **多言語化**
  グローバルな開発環境への対応（英語、中国語・・）

マルチデータベースのサポート

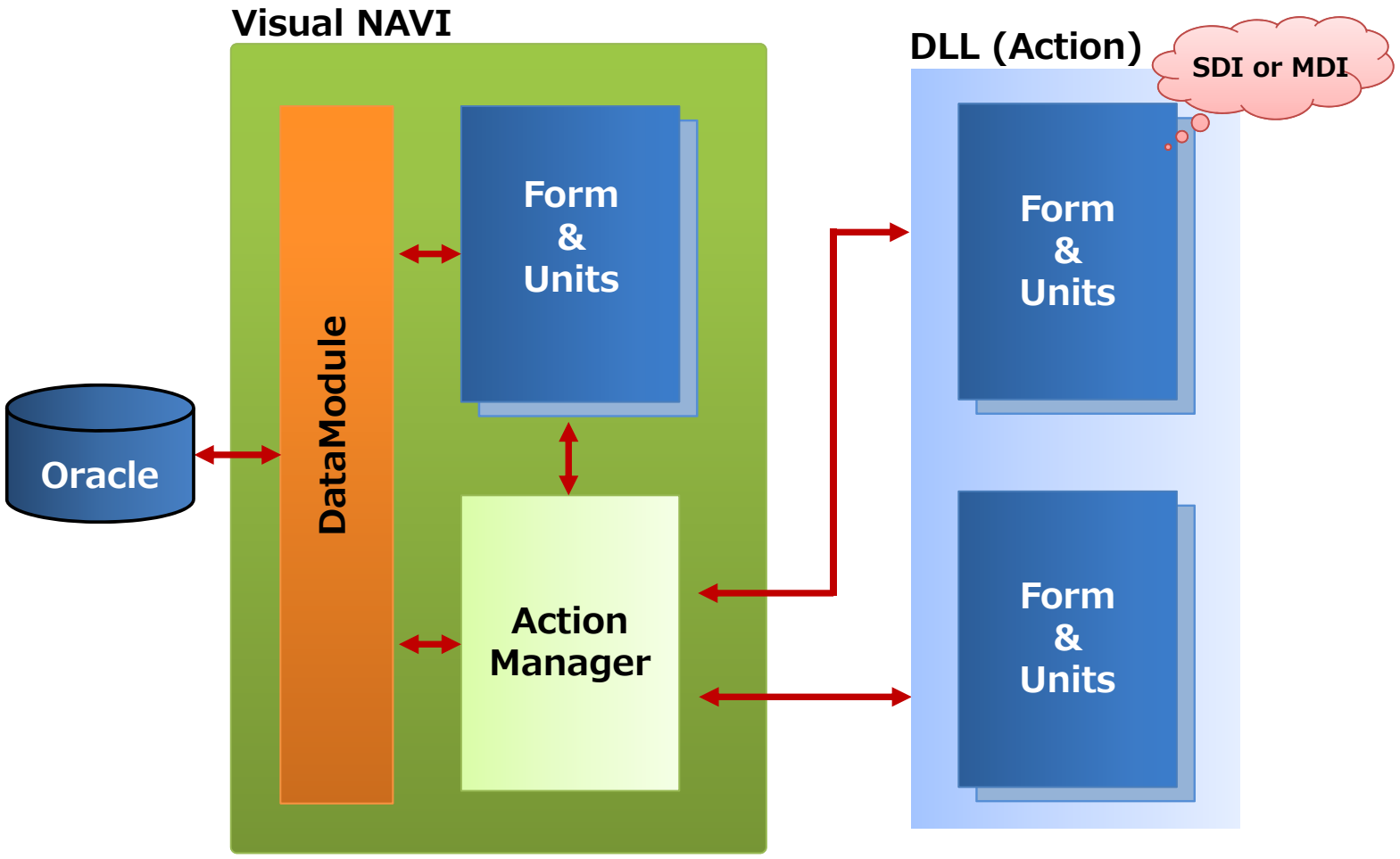
- FireDACがサポートする主なDBMSのサポート
- 第一段としてOracle。続いてMS-SQL Server、MySQLをリリース予定



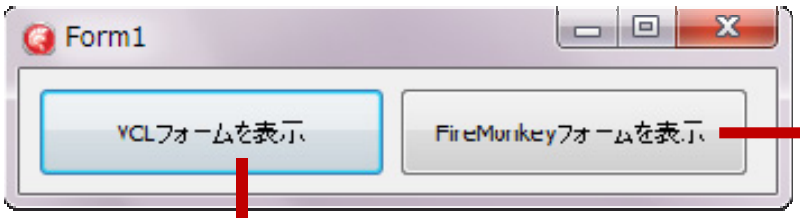
ソフトウェア構造の根本的な見直し (Before)

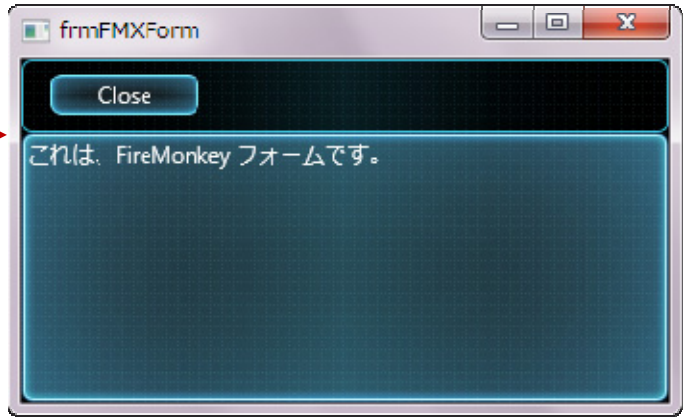


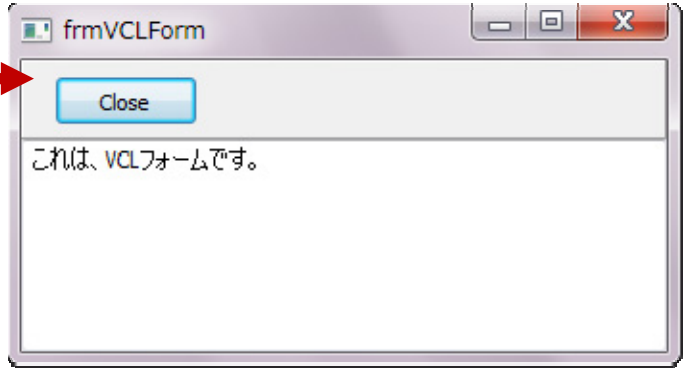
ソフトウェア構造の根本的な見直し (After)



無謀にも‥ VCLからFireMonkey(FMX)フォームを呼び出す





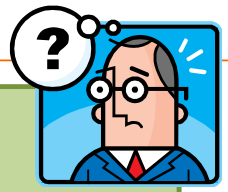


FireMonkeyフォームの動的呼出

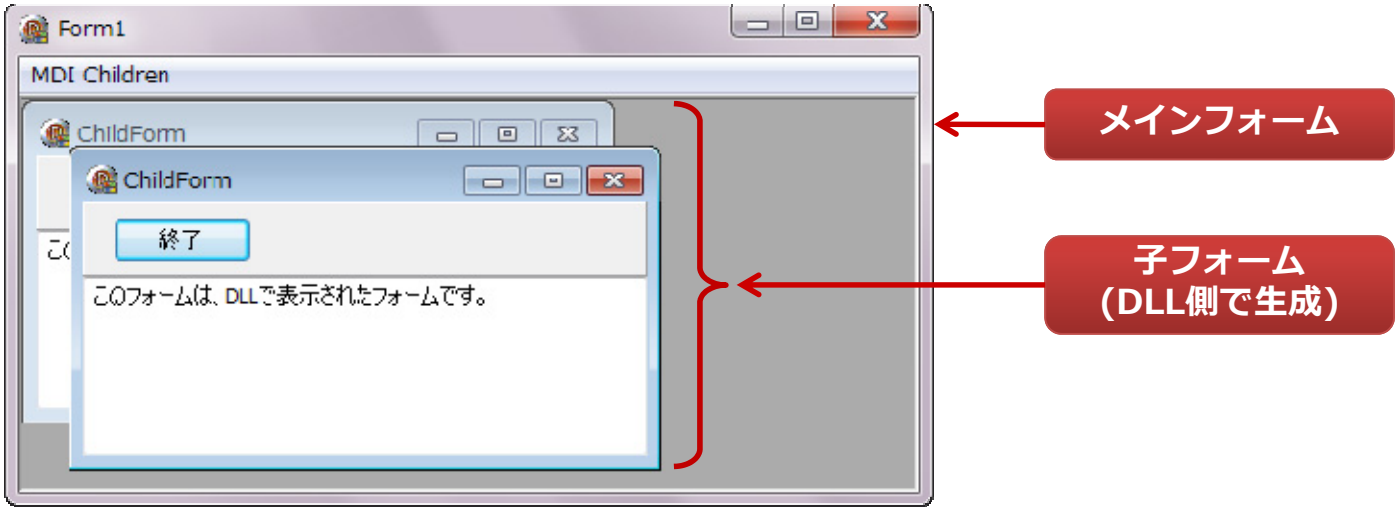
```
type
  TShowFMXForm = procedure stdcall;

procedure TForm1.btnShowFMXFormClick(Sender: TObject);
var
  ShowFMXForm: TShowFMXForm;
  DLLHandle: THandle;
begin
  DLLHandle := LoadLibrary('VCLdll.dll');
  if DLLHandle <> 0 then
  begin
    @ShowFMXForm := GetProcAddress(DLLHandle, 'ShowFMXForm');
    if Assigned(ShowFMXForm) then ShowFMXForm();
    FreeLibrary(DLLHandle); *1
  end;
end;
```

*1 ライブラリの解放(FreeLibrary呼出)でダンマリ状態になる。（正しく終了しない）
VCL(.exe/.dll)とFireMonkey(.dll/.exe)とをアプリ内で混在させることは想定されていない。
詳細は次のQualityCentral(QC)を参照。
[DLL developed with FireMonkey crash under FireMonkey app/ VCL app after FreeLibrary to unload the DLL]
http://qc.embarcadero.com/wc/qcmain.aspx?d=123874

DLL内でMDI子フォームを生成する



ポイントは・・
DLL内のApplicationとScreenの値を、呼出側の値と一致させる。

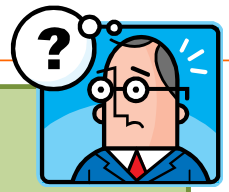


実行時ライブラリを使っても実現できますが、ここではレガシィなやり方で実装します。
作成したForm(Unit)をリポジトリとして登録しておくと、クラス継承が容易となります。

DLL内でMDI子フォームを生成する (DLL側コードの抜粋)

```
var
  SaveApplication: TApplication;
  SaveScreen: TScreen;

procedure ShowChildForm(AApplication: TApplication; AScreen: TScreen); stdcall;
begin
  if not Assigned(SaveApplication) then
  begin
    // Tapplication･Tscreenのセーブ
    SaveApplication := Application;
    SaveScreen := Screen;
    // 親のTapplication･Tscreenをセット
    Application := AApplication;
    Screen := AScreen;
  end;
  frmChild := TfrmChild.Create(Application);
end;

initialization
  SaveApplication := nil;

finalization
  // Tapplication･Tscreenを元に戻す
  if Assigned(SaveApplication) then
  begin
    Application := SaveApplication;
    Screen := SaveScreen;
  end;
end;
```

FireMonkeyの状態保存（保存）



XE7からアプリケーションの状態の保存／復帰機能が追加されています。
これにより、従来INIファイルで処理していたことが簡単に実現することができます。

```
procedure TForm1.FormSaveState(Sender: TObject); *1
var
  BW: TBinaryWriter;
begin
  SaveState.Stream.Clear;
  // 何かが編集されたときにのみ現在の状態が保存される
  // 何も変更されていない場合は、状態がこのように削除される
  if Edit1.Text.Length > 0 then
  begin
    // 入力したテキストを Edit1 コントロールに保存する
    BW := TBinaryWriter.Create(SaveState.Stream);
    try
      BW.Write(Edit1.Text);
      BW.Write(Edit2.Text);   *2
    finally
      BW.Free;
    end;
  end;
end;
```

*1 OnSaveStateイベントに設定
*2 保存するコントロールを指定する。

FireMonkeyの状態保存 (復帰)

```
uses
  System.IOUtils;

procedure TForm1.FormCreate(Sender: TObject);
var
  BR: TBinaryReader;
begin
  SaveState.StoragePath := TPath.GetHomePath; *1
  if SaveState.Stream.Size > 0 then
  begin
    // 前に入力したテキストを Edit1 コントロールに回復する
    BR := TBinaryReader.Create(SaveState.Stream);
    try
      Edit1.Text := BR.ReadString;
      Edit2.Text := BR.ReadString;   *2
    finally
      BR.Free;
    end;
  end;
end;
```

*1 保存先を指定。この場合はホームパス (C:¥Users¥<ユーザー名>¥AppData¥Roaming)に保存される。
*2 読込むコントロールを指定する。

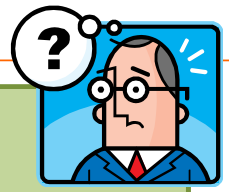
FireMonkeyでクリップボードを扱う

```
uses
  FMX.Platform;


// クリップボードへ文字列をコピーする
procedure ClipboardAsString(Text: String);
var
  ClipboardService: IFMXClipboardService;
begin
  ClipboardService :=
IFMXClipboardService(TPlatformServices.Current.GetPlatformService(IFMXClipboardService));
  ClipboardService.SetClipboard(text);
end;

// クリップボードから文字列を取得する
function GetClipboardAsString: String;
var
  ClipboardService: IFMXClipboardService;
begin
  ClipboardService :=
IFMXClipboardService(TPlatformServices.Current.GetPlatformService(IFMXClipboardService));
  Result := ClipboardService.GetClipboard.AsString;
end;
```

以下を参照して下さい。

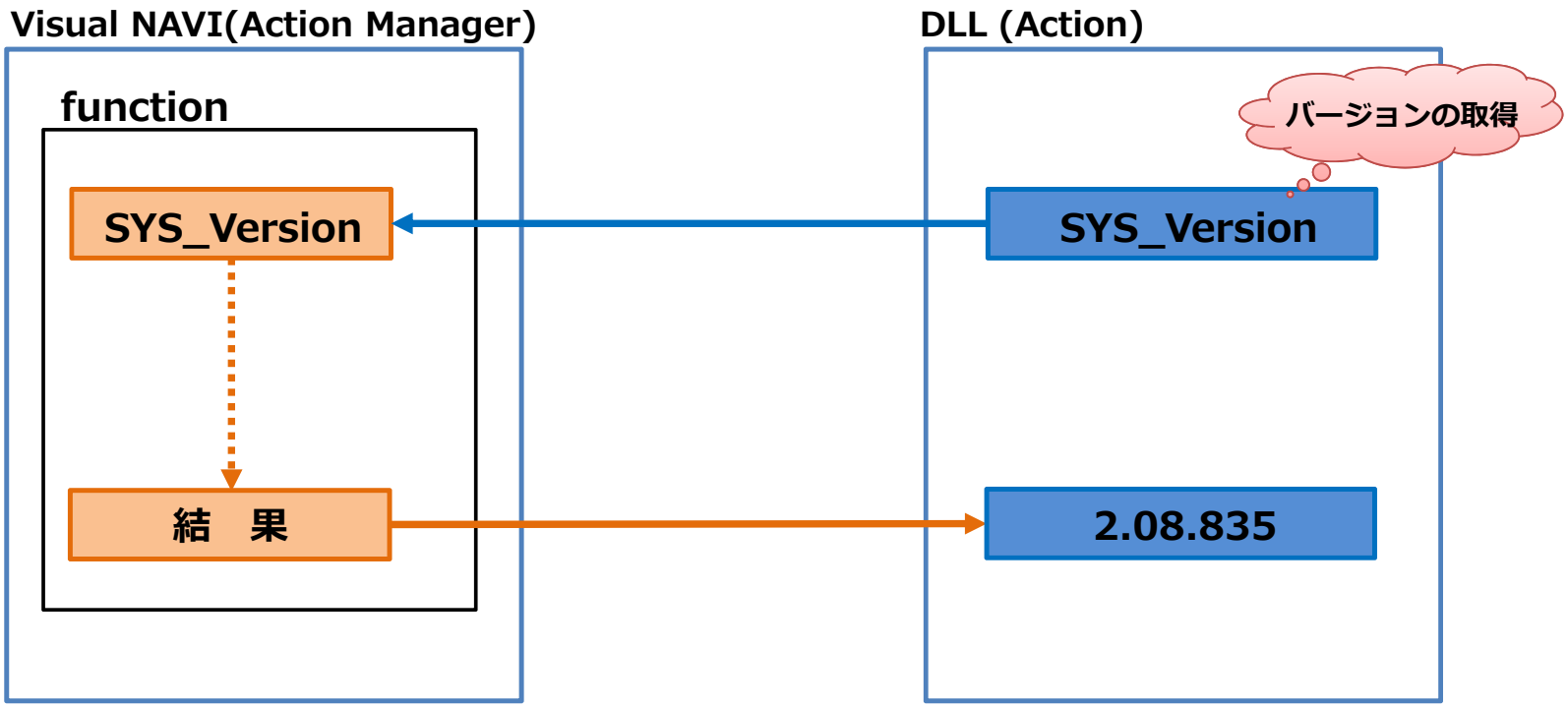
- **TPlatformServices** : http://docwiki.embarcadero.com/Libraries/XE7/ja/FMX.Platform.TPlatformServices
- **IFMXClipboardService**: http://docwiki.embarcadero.com/Libraries/XE7/ja/FMX.Platform.IFMXClipboardService

コア技術は、コールバック関数！



コールバック関数とは‥?
プログラム中で、呼び出し先の関数の実行中に実行されるように、あらかじめ指定しておく関数。

(IT用語辞典より引用)



コールバック関数の体系

関数名	機能概要
SYSTEM info functions • SYS_Version • SYS_RootDir • SYS_IniFileName :	**Visual NAVIの各種情報を取得** • バージョンを取得 • 起動Dirを取得 • 環境設定ファイル名を取得 :
IDE functions • IDE_Connected • IDE_ConnectionInfo • IDE_GetAppHandle :	**IDEの情報を取得** • DB接続の有無を取得（True:接続済み） • DB接続情報の取得 • アプリケーションハンドル名を取得 :
SQL functions • SQL_Execute • SQL_FieldCount • SQL_Eof :	**SQLの実行と結果の取得** • SQLの実行 • 項目（コンポーネント）の数を取得 • データセットの最後かどうかを取得 :
Oracle functions • ORA_Version • ORA_OracleHome :	**Oracleの各種情報取得、処理の依頼** • Oracleのバージョンを取得 • Oracle-Homeを取得 • :

コールバック関数の登録

DLL側で登録するコールバック関数を「RegisterCallback」手続き内で定義をします。
Visual NAVI起動時に、自動的にコールバック関数が登録される

```
var
  // コールバック関数の定義
  SYS_Version   : function: PWideChar; stdcall;
  SYS_RootDir   : function: PWideChar; stdcall;
  ORA_Version   : function: PWideChar; stdcall;
  ORA_OracleHome: function: PWideChar; stdcall;
       :

// コールバック関数の登録
procedure RegisterCallback(Index: Integer; Addr: Pointer); stdcall;
begin
  case Index of
     1: @SYS_Version    := Addr;
     2: @SYS_RootDir    := Addr;
              :
    91: @ORA_Version    := Addr;
    92: @ORA_OracleHome := Addr;
              :
  end;
end;
```

コールバック関数の呼出

Visual NAVI側からDLLの「OnMenuClick」を呼び出すと、DLL側で処理が実行される。

```
procedure OnMenuClick(Index: Integer); stdcall;
begin
  case Index of
    1: ShowLogonInfo;
    2: ShowMessage(IntToStr(IDE_GetWindowType));
    3: IDE_CreateWindow(3, 'select * from tab', True);
    5: ShowMessage(SYS_Version);
    6: ShowMessage(SYS_RootDir);
    7: ShowMessage(ORA_Version);
  end;
end;
```

実行例

SQLの実行

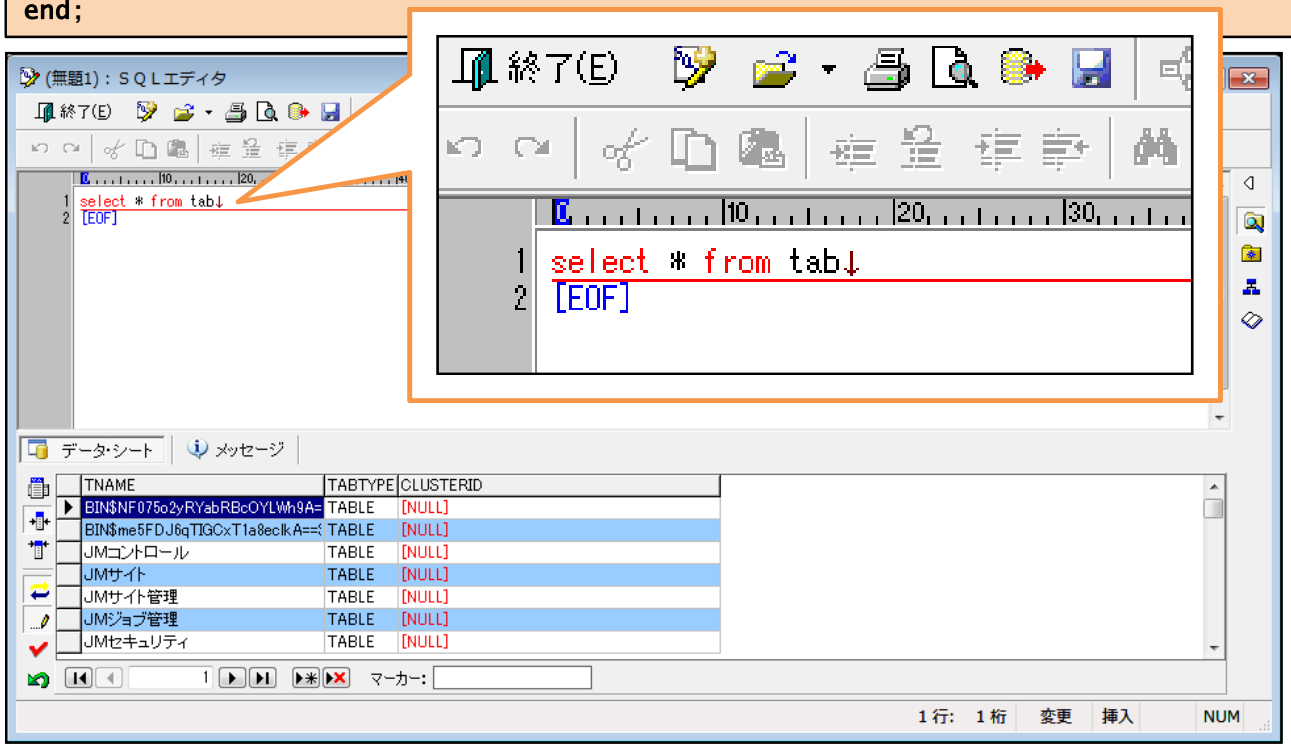
次のソースコードは、Oracleの「データディクショナリビュー」からユーザー一覧を取得するものです。（赤字がコールバック関数）

```
function GetAllUsers(UserList: TStrings): Boolean;
var
  SQL: TStrings;
  ErrMessage: PWideChar;
begin
  Result := False;
  SQL := TStringList.Create;
  try
    // SQL文の編集
    SQL.Clear;
    SQL.Add('/* Visual NAVI */ SELECT');
    SQL.Add('  DISTINCT USERNAME');
    SQL.Add('FROM');
    SQL.Add('  sys.ALL_USERS');
    SQL.Add('ORDER BY');
    SQL.Add('  USERNAME');
    // SQLの実行
    if (SQL_Execute(PWideChar(SQL.Text), ErrMessage) = 0) then
    begin
      // Itemsをクリア
      UserList.Clear;
      // ユーザー名をUserListに溜め込む
      while not SQL_Eof do
      begin
        UserList.Add(SQL_Field(0));
        SQL_Next;
      end;
      Result := True;
    end
    else
      IDE_OutputEvents(2, 'ユーザー一覧取得(SELECT)', ErrMessage);
  finally
    FreeAndNil(SQL);
  end;
end;
```

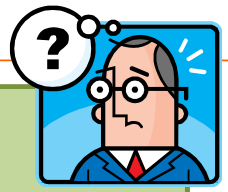
Visual NAVI のSQLエディタを起動・実行する

Visual NAVI側から呼ばれた「OnMenuClick」内で、コールバック関数を実行する。

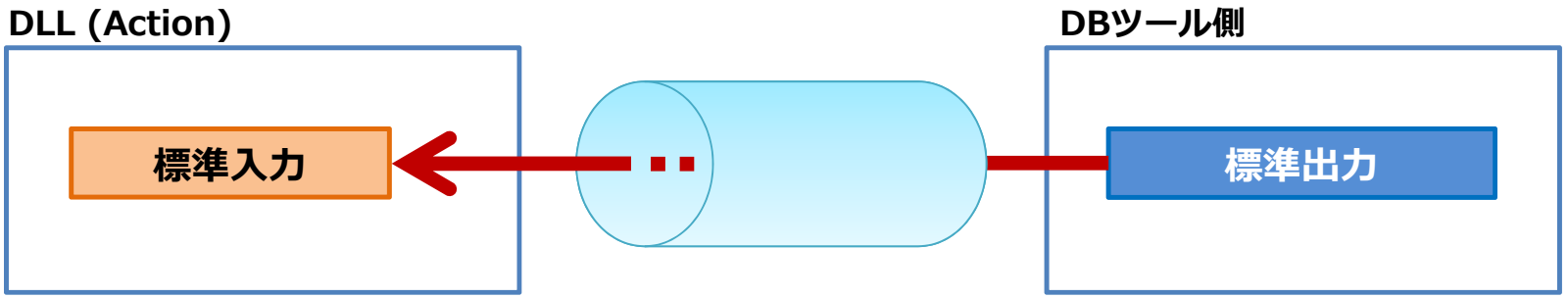
```
procedure OnMenuClick(Index: Integer); stdcall;
begin
  case Index of
    1: ‥;
         :
    3: IDE_CreateWindow(3, 'select * from tab', True);
         :
  end;
end;
```

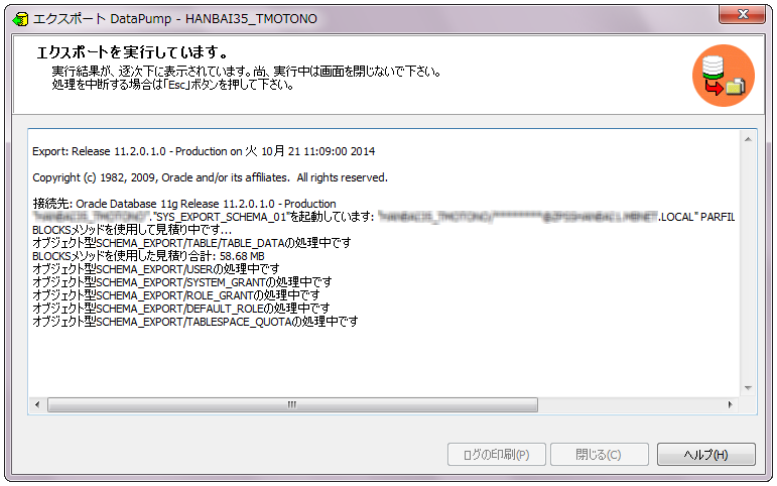


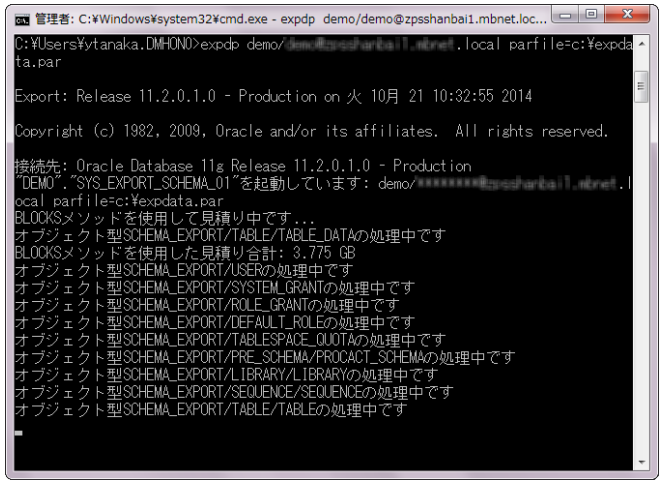
パイプを使ってメッセージを横取りする！



パイプ(pipe) とは‥?
あるプログラムの出力を別のプログラムの入力に引き渡す機能。
DBツールにはコマンドラインで実行されるものが多く、ツールが標準出力するメッセージを
インターセプト(intercept)します。







デモ

- DLL内でMDI子フォームを生成する
- コールバック関数を使う
- VCLからFireMonkey(FMX)フォームを呼び出す

ご清聴
ありがとうございました


Yoshiki.tanaka-avsoft@nifty.com