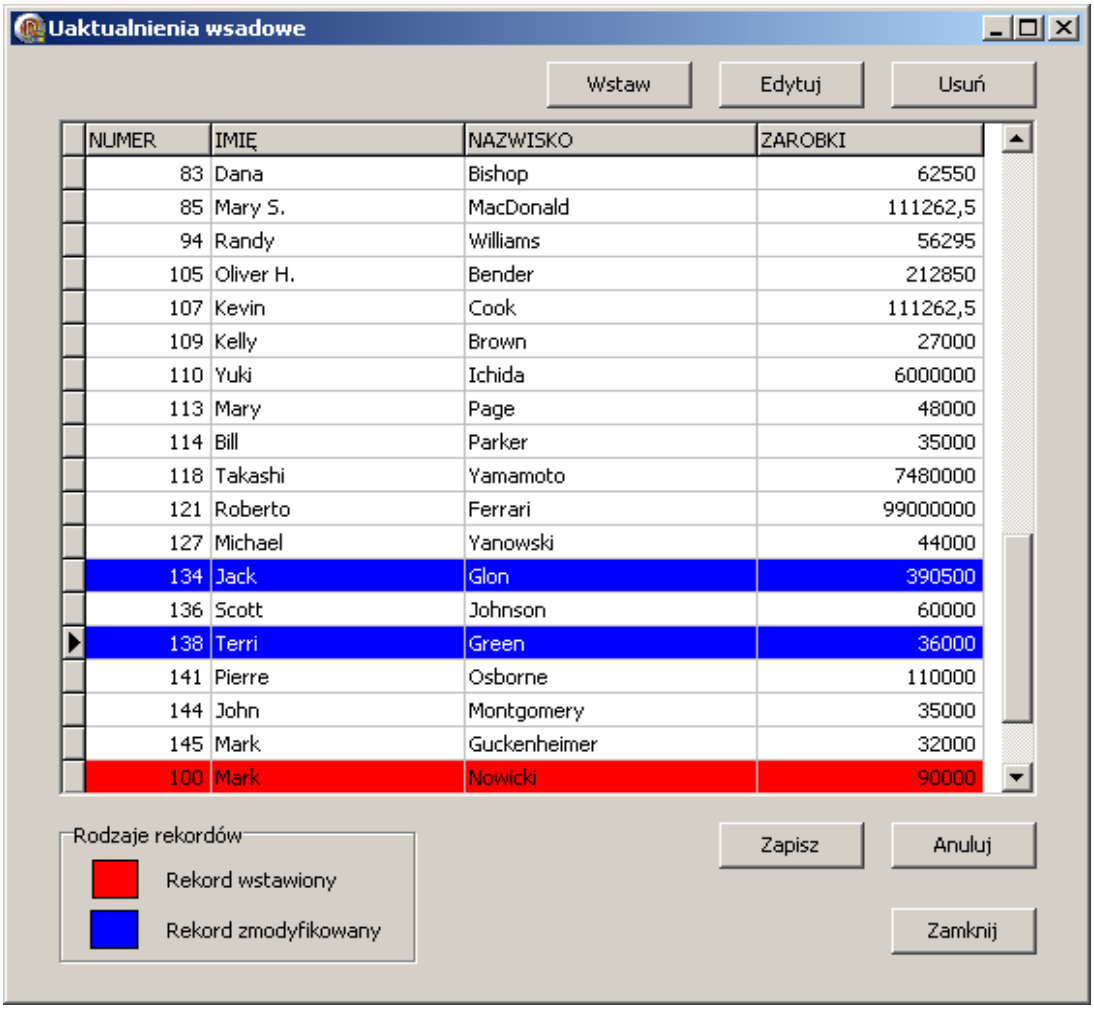
Bazy danych w TURBO DELPHI (2)

Zadanie: Poznanie mechanizmu uaktualnień wsadowych oraz klonowania zbiorów danych w ADO.

Uaktualnienia wsadowe w ADO (odpowiednik buforowanych uaktualnień w BDE i IBExpress) jest to mechanizm składającym się z trzech faz. Pierwsza z nich polega na pobraniu danych z serwera. Druga faza to lokalne edytowanie danych. Trzecia faza polega na jednoczesnym zatwierdzeniu wszystkich edytowanych danych.
Mechanizm klonowania zbiorów danych: klonowany zbiór danych jest nowym zbiorem danych, który ma wszystkie właściwości takie, jak oryginalny zbiór, z którego jest tworzony. Klony tworzy się za pomocą metody Clone zbioru danych.
Będziemy się łączyć z bazą danych *employee*. Skorzystamy z dostawcy LCPI OLE DB Provider for InterBase (wersja 3, free). (Przed rozpoczęciem pracy należy zainstalować powyższy sterownik!)
Przed uruchomieniem Turbo Delphi warto stworzyć tabelę z przykładowymi danymi, na której będziemy wykonywali operacje usuwania, dodawania i edycji. W naszym przypadku jest to tabela Pracownik, zawierająca kopię następujących danych: Emp_No, First_Name, Last_Name, Salary z tabeli Employee.

Program powinien działać w następujący sposób:
Wczytujemy do DBGrid dane z nowo utworzonej tabeli Pracownik (w naszym przypadku jest to tabela zawierająca kopię następujących danych: Emp_No, First_Name, Last_Name, Salary z tabeli Employee). Następnie stosując przyciski: Wstaw, Edytuj, Usuń dokonujemy zmian, które jednak nie będą natychmiast zatwierdzone. Wszystkie zmienione rekordy powinny być odpowiednio oznaczone. Po dokonaniu zmian możemy je zatwierdzić (przycisk Zapisz) lub wycofać (przycisk Anuluj).

Okno programu może wyglądać następująco:



Tworząc program będziemy korzystać z następujących komponentów:

a) Z zakładki dbGo
   i) ADOConnection
   ii) ADODataSet
b) Z zakładki Data Access
   i) DataSource
c) Z zakładki Data Controls
   i) DBGrid
d) Z zakładki Standard
   i) Button
   ii) GroupBox
   iii) Label
e) Z zakładki Additional
   i) Shape

1) Utwórz nowy projekt. Utwórz moduł danych. Zmień nazwę modułu na *DataModuleDane*, nazwę formatki na *FormGlowny*. Zapisz projekt w wybranym wcześniej katalogu. Moduł będzie służyć do obsługi danych – tutaj będziemy umieszczać komponenty dostępu do danych.

2) Wstaw do modułu ADOConnection1 i kliknij dwa razy na tym komponencie.
Zaznacz: Use Connection String i kliknij: Build.
Na zakładce Provider wybierz LCPI OLE DB Provider for InterBase
Na zakładce Connection wypełnij pola:
Database description: employee
Full path to database file: pełna ścieżka dostępu do pliku bazy *employee.fdb*
User: SYSDBA
Password: masterkey
Zaznacz Save password
Zaznacz Enable automatic transaction
Na zakładce Advanced wypełnij pola:
Database Client: Firebird 2.0
Dynamic library: pełna ścieżka dostępu do pliku *fbclient.dll*
Przejdź na zakładkę Connection i sprawdź połączenie - kliknij Test connection.
Zamknij okno właściwości łącza danych.

Ustaw kolejne własności w komponencie ADOConnection1:
Własność LoginPrompt ustaw na FALSE
Własność Connected ustaw na TRUE

3) Wstaw do modułu ADODataSet1
Dla własności Connection wybierz ADOConnection1
Dla własności CommandText wpisz: select * from Pracownik
Aby uzyskać uaktualnienia wsadowe, musimy zmienić dwie następujące własności:
    własność LockType na ltBatchOptimistic
    własność CursorLocation na clUseClient
Ustaw własność Active na TRUE

4) Wstaw do modułu DataSource1
Dla własności DataSet wybierz ADODataSet1

5) Wrzuć na formatkę DBGrid1
Dla własności DataSource wybierz DataModuleDane.DataSource1
W parametrze Options zaznacz dgRowSelect (dzięki temu DBGrid nie będzie edytowalny dopóki nie wciśniemy odpowiedniego przycisku).

6) Dla zdarzenia FormActivate powinny być wykonywane następujące akcje:

```
DataModuleDane.ADODataSet1.Open;
ButtonZapisz.Enabled:=false;
ButtonAnuluj.Enabled:=false;
```

7) Procedura „rysująca” DBGrid (z uwzględnieniem kolorów zmienionych rekordów) może wyglądać następująco (korzystamy tu z właściwości zbioru danych UpdateStatus, która informuje jaka zmiana została dokonana dla konkretnego rekordu)

Dla zdarzenia OnDrawColumnCell komponentu DBGrid1:

```
procedure TFormGlowny.DBGrid1DrawColumnCell(Sender: TObject; const Rect: TRect;
                DataCol: Integer; Column: TColumn; State: TGridDrawState);
begin
    with TDBGrid(Sender).Canvas do
    begin
            case DataModuleDane.ADODataSet1.UpdateStatus of
                usInserted : Brush.Color := clRed;
                usModified :
                        begin
                        Brush.Color := clBlue;
                        Font.Color := clWhite;
                        end;
            end;
    end;
    TDBGrid(Sender).DefaultDrawColumnCell(Rect,DataCol,Column,State);
end;
```

8) Do oprogramowania przycisków Wstaw, Edytuj, Usuń używamy metod odpowiednio: Insert, Edit, Delete. Dla zdarzeń OnClick dla przycisków Wstaw, Edytuj, Usuń wpisujemy odpowiednio kod:

```
DataModuleDane.ADODataSet1.Insert;
DataModuleDane.ADODataSet1.Edit;
DataModuleDane.ADODataSet1.Delete;
```

W celu umożliwienia edycji w DBGrid, czyli dla przycisków Wstaw i Edytuj musimy dodatkowo zmienić opcje komponentu DBGrid

```
DBGrid1.Options:= DBGrid1.Options - [dgRowSelect] + [dgEditing];
```

9) Następująca funkcja sprawdza, czy w danym zbiorze danych jakieś rekordy są nadal w trakcie uaktualniania (innymi słowy, czy są jeszcze jakieś niezatwierdzone zmiany). Skorzystamy z mechanizmu klonowania oraz z właściwości FilterGroup zbioru danych.

```
Function TDataModuleDane.ADOUpdatesPending(AdodataSet:TCustomADODataSet):Boolean;
var Clone:TADODataSet;
begin
   Clone:=TADODataSet.Create(nil);
   try
        Clone.Clone(AdodataSet);                  //klonuj zbiór danych
        Clone.FilterGroup:=fgPendingRecords;  //wybierz te rekordy, dla których zmiany jeszcze
                                                          nie są zatwierdzone
        Clone.Filtered:=True;                      //włącz filtr
        Result:=not(Clone.Bof and Clone.Eof);  //jeżeli ten zbiór danych jest niepusty, to są
                                                 niezatwierdzone zmiany
        Clone.Close;
   finally
```

```
            Clone.Free;
        end;
    end;
```

10) Po każdej zmianie pozycji w DBGrid należy znowu zablokować możliwość edycji (zdarzenie AfterScroll dla komponentu ADODataSet1):

```
procedure TDataModuleDane.ADODataSet1AfterScroll(DataSet: TDataSet);
begin
    FormGlowny.DBGrid1.Options:= FormGlowny.DBGrid1.Options + [dgRowSelect];
    if ADOUpdatesPending(ADODataSet1) then
        begin
            FormGlowny.ButtonZapisz.Enabled:=True;
            FormGlowny.ButtonAnuluj.Enabled:=True;
        end;

end;
```

11) Zatwierdzanie zmian:

```
procedure TFormGlowny.ButtonZapiszClick(Sender: TObject);
begin
    with DataModuleDane.ADODataSet1 do
        begin
            if (not DataModuleDane.ADOConnection1.InTransaction) then
                DataModuleDane.ADOConnection1.BeginTrans;
        try
            UpdateBatch;
            DataModuleDane.ADOConnection1.CommitTrans;
            Refresh;
         except
            DataModuleDane.ADOConnection1.RollbackTrans;
            MessageDlg('Zmiany nie mogą być zapisane!',mtError,[mbOK],0);
            Refresh;
        end;
     end;

     if not(DataModuleDane.ADOUpdatesPending(DataModuleDane.ADODataSet1)) then
        begin
            ButtonAnuluj.Enabled:=False;
            ButtonZapisz.Enabled:=False;
        end;
end;
```

12) Wycofywanie zmian:

```
procedure TFormGlowny.ButtonAnulujClick(Sender: TObject);
begin
     DataModuleDane.ADODataSet1.CancelBatch(arAll);
     if not DataModuleDane.ADOUpdatesPending(DataModuleDane.ADODataSet1) then
        begin
            ButtonAnuluj.Enabled:=False;
            ButtonZapisz.Enabled:=False;
        end;
end;
```

13) Wrzuć na formatkę komponent GroupBox1. Ustaw własność Caption na Rodzaje rekordów. Umieść na polu GroupBox1 następujące elementy:

Shape1,Shape2

Label1, Label2.

Ustaw kolory kwadratów Shape – własność Brush – odpowiednio na clRed,clBlue.

Ustaw własności Caption dla komponentów Label odpowiednio jako: Rekord wstawiony, Rekord Zmodyfikowany.

14) Wrzuć na formatkę przycisk Button, ustaw Caption na Zamknij, zdarzenie OnClick niech wykonuje:

```
FormGlowny.Close;
```

15) Zanim zamkniemy formatkę, należy sprawdzić, czy jakieś rekordy są w trakcie uaktulaniania (zdarzenie OnCloseQuery formatki)

```
procedure TFormGlowny.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  CanClose:=true;
  if (DataModuleDane.ADOUpdatesPending(DataModuleDane.ADODataSet1)) then
        CanClose:=(MessageDlg('Zmiany nie sa zatwierdzone, czy zamknąć?',
        mtConfirmation,[mbyes,mbno],0)  =mryes);
end;
```