Framework IMPET

Dokumentacja techniczna

Software Factory
Maciej Szymczak,
Data utworzenia: 12.11.2000
Data ostatniej aktualizacji: 2012.03.31

soft@home.pl
tel. 604.22.46.58

Spis treści

Wstęp

Framework IMPET wspomaga proces tworzenia aplikacji bazodanowych w środowisku Delphi. Framework został napisany przez firmę Software Factory. Sztandarowy produkt, który powstał przy użyciu tego framework to www.plansoft.org

Framework IMPET stanowi zestaw modułów- cześć z nich to wzorce okien z wbudowaną standardową funkcjonalnością, z których należy korzystać przy tworzeniu aplikacji; część stanowi moduły użytkowe (np. okno do wywoływania listy wartości).

Pakiet IMPET działa w oparciu o połączenie ADO i zapewnia obsługę wszystkich baz danych obsługiwanych przez ADO, w szczególności obsługuje bazę danych ORACLE.
Wykonanie skomplikowanego formularza do przeglądania i aktualizacji danych, z możliwością wyszukiwania, generowania podsumowań, zobrazowania graficznego danych, drukowania, eksportowania danych do pakietu MS Office, tworzenia zestawień krzyżowych, dostosowywania wyglądu okna i parametrów modułu, sortowania danych wg wybranych kryteriów, wyszukiwanie za pomocą tzw. lokatora inkrementalnego itd. zajmuje sprawnemu developerowi około godziny.

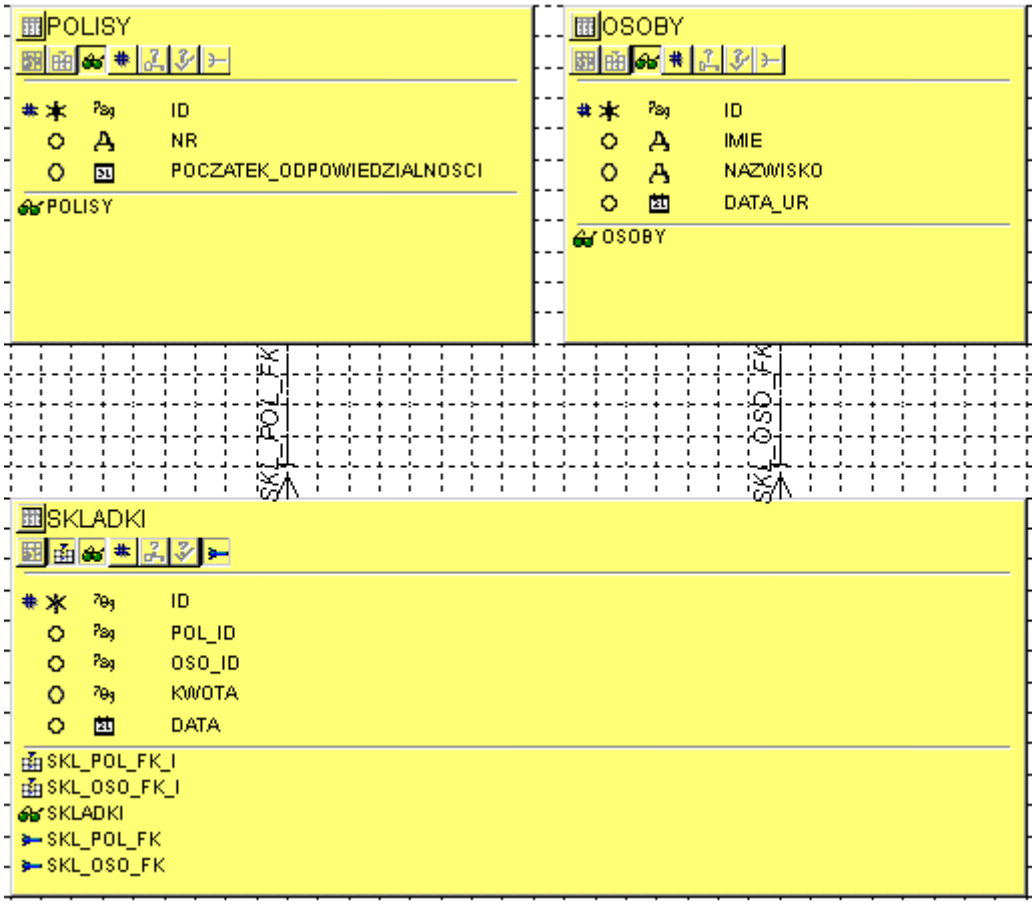
Pakiet stanowi wynik doświadczeń z narzędziami:
- Forms Developer firmy Oracle (idea modułu danych zintegrowanego z formularzem, lookupy),
- Clarion firmy TopSpeed (okno do przeglądania danych w liście i okno do aktualizacji pojedynczego rekordu),
- Lotus Notes (moduł filtrowania danych, okno do wybierania wartości z listy wartości, koncepcja uwierzytelniania za pomocą karty)
- Access (zestawienia krzyżowe)
- OAF – Oracle Application Framework (java)
- i oczywiście Delphi.

Sprawne projektowanie przy użyciu pakietu wymaga znajomości języka SQL oraz podstaw w zakresie programowania obiektowego. Aby skompilować pakiet IMPET, należy zainstalować pakiet Rx w wersji co najmniej 2.75 (Pakiet Rx jest dostępny na stronie Delphi Super Page). Najprostszą metodą skorzystania z pakietu IMPET jest rozpoczęcie tworzenia nowej aplikacji w oparciu o projekt o nazwie ExampleProject. Zawiera on wszystkie moduły pakietu IMPET. Projekt bazowy należy skopiować i zachować pod stosowaną nazwą.
W części **Tworzenie modułu danych** szczegółowo opisano jak utworzyć nowy moduł oraz jak rozwiązywać najczęściej występujące problemy podczas projektowana modułu.

Przykładowy problem

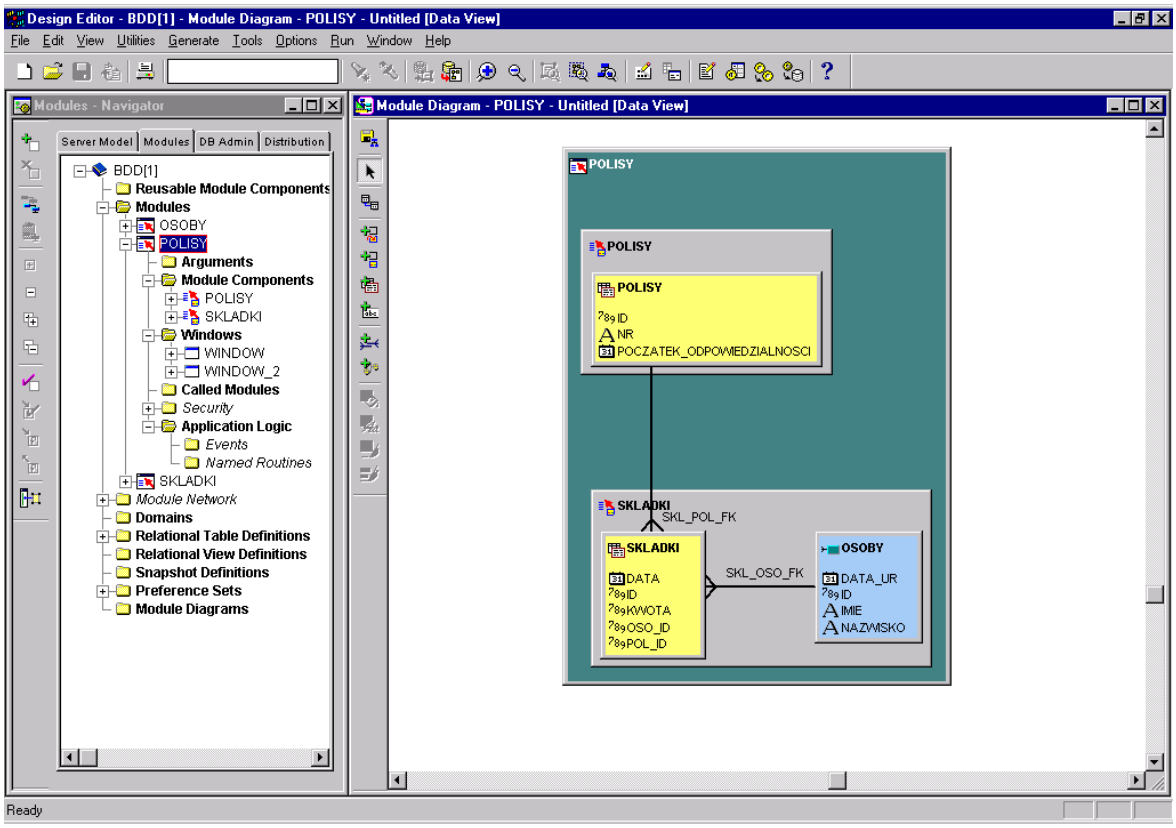
Załóżmy, że pewna firma ubezpieczeniowa zleciła wykonanie systemu przechowującego dane o polisach, wpływających na nie składkach i osobach, które wpłacają składki. Zastrzegła ona istnienie trzech kategorii użytkowników:
użytkownicy mający pełny dostęp do danych,
użytkownicy mający dostęp do wszystkich danych, ale możliwością dodawania wyłącznie danych o polisach,
użytkownicy mający dostęp do wszystkich danych, ale możliwością dodawania wyłącznie danych o osobach.

Schemat bazy danych może wyglądać jak na poniższym rysunku. Kontrola praw dostępu będzie po stronie serwera - zostaną utworzeni użytkownicy DEMO (pełny dostęp), REJ_POLISY (polisy) oraz REJ_OSOBY (osoby).



Rozwiązanie - Designer/ Forms Developer

Designer firmy Oracle jest narzędziem typu case wspomagającym proces wytwarzania aplikacji dwu- i trójwarstwowych. Istotnym elementem projektowania modułu (formularza) jest zdefiniowanie diagramu danych dla modułu. Poniższy rysunek przedstawia diagram danych dla modułu polisy. Szare prostokąty to bloki danych, tabele oznaczone kolorem żółtym to tabele bazowe, tabele niebieskie to tabele podglądane (ang. lookups).



Po zdefiniowaniu modułów w Designerze następuje proces generowania modułów do środowiska Forms Developer. Właściwości wygenerowanego modułu zależą od:

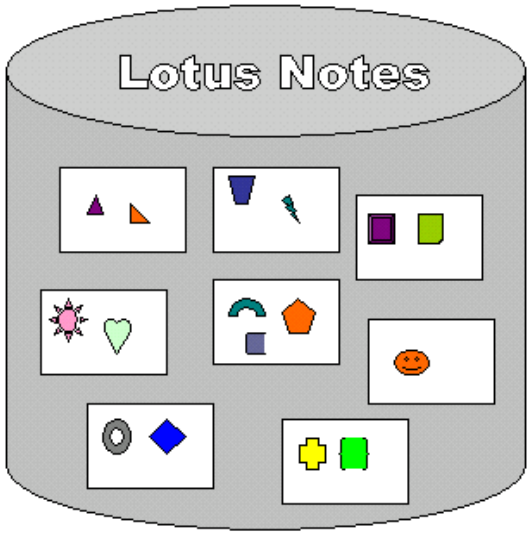
- zestawu preferencji użytych podczas generowania,
- użytej biblioteki obiektów,
- zastosowanego szablonu formularza.

Szablon formularza może posiadać elementy stałe, takie jak nagłówek/ stopkę z nazwą firmy, nazwą formularza, numerem wersji, przyciski itd.

Środowisko Forms Developer ma wbudowane standardowe mechanizmy obsługi bazy danych, takie jak blokowanie rekordu, maksymalna liczba rekordów pobieranych z serwera, wyszukiwanie danych wg wprowadzonych kryteriów, standardowe klawisze skrótów itd. **Dzięki temu wysiłki deweloperów skupiają się wyłącznie na kwestiach merytorycznych związanych z wykonaniem systemu**. Stąd pomysł utworzenia w Delphi wzorca formularza z wbudowaną standardową funkcjonalnością, który zostanie zaprezentowany w dalszej części artykułu.

Rozwiązanie – Lotus Notes

Lotus Notes w odróżnieniu od bazy relacyjnej służy do przechowywania danych o nieregularnej strukturze. Baza Lotus Notes często porównywana jest do worka z mieszkami z różną zawartością wewnątrz. Worek to baza danych a mieszek to pojedyncza dana w bazie, zawierająca informacje o danych i strukturze tych danych (zob. rys).



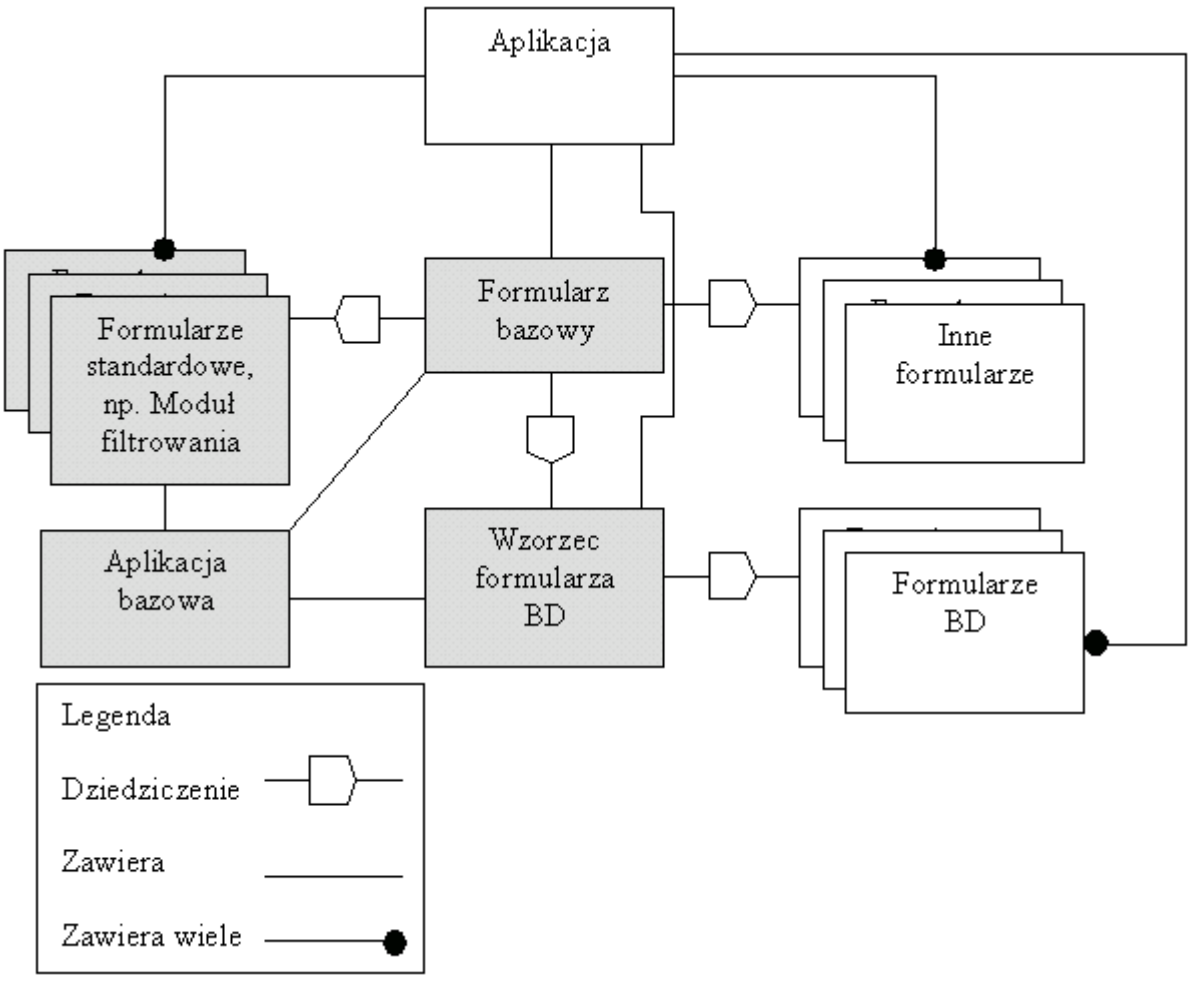
Na uwagę zasługuje zaawansowany mechanizmem wyszukiwania danych. Umożliwia on wprowadzenie dowolnej liczby warunków z podaniem pól i zakresu wartości dla tych pól. Pomysł ten stał się inspiracją dla wykonania modułu filtrowania danych w prezentowanym rozwiązaniu w Delphi.

Rozwiązanie – Delphi

Przystępując do pisania aplikacji w Delphi, pierwszym intuicyjnym rozwiązaniem wydaje się utworzenie aplikacji składającej się z wielu niezależnych formularzy. Dopiero po pewnym czasie okazuje się, że te same problemy rozwiązywane są przez wielu deweloperów wielokrotnie, nierzadko w inny sposób, pojawiają się problemy z konserwacją oprogramowania, czas wytwarzania i testowania systemu zwiększa się wykładniczo, co jakiś czas pojawiają się nowe, nieprzewidziane błędy.

Aby uniknąć opisanych problemów można utworzyć wzorzec formularza z wbudowaną standardową funkcjonalnością, dziedziczony przez pozostałe formularze bazodanowe aplikacji. Dodatkowo w wielu projektach informatycznych pojawia się wymóg wytworzenia aplikacji wielojęzycznej, przechowującej treści komunikatów i wyświetlanych tekstów w plikach konfiguracyjnych. Można zatem wykonać wzorzec formularza bazowego z wbudowaną funkcjonalnością odczytywania i zapisywania wizualnych elementów okna w plikach konfiguracyjnych (zob. formularz bazowy na rysunku)
Wzorzec formularza bazy danych powinien dziedziczyć właściwości formularza bazowego.

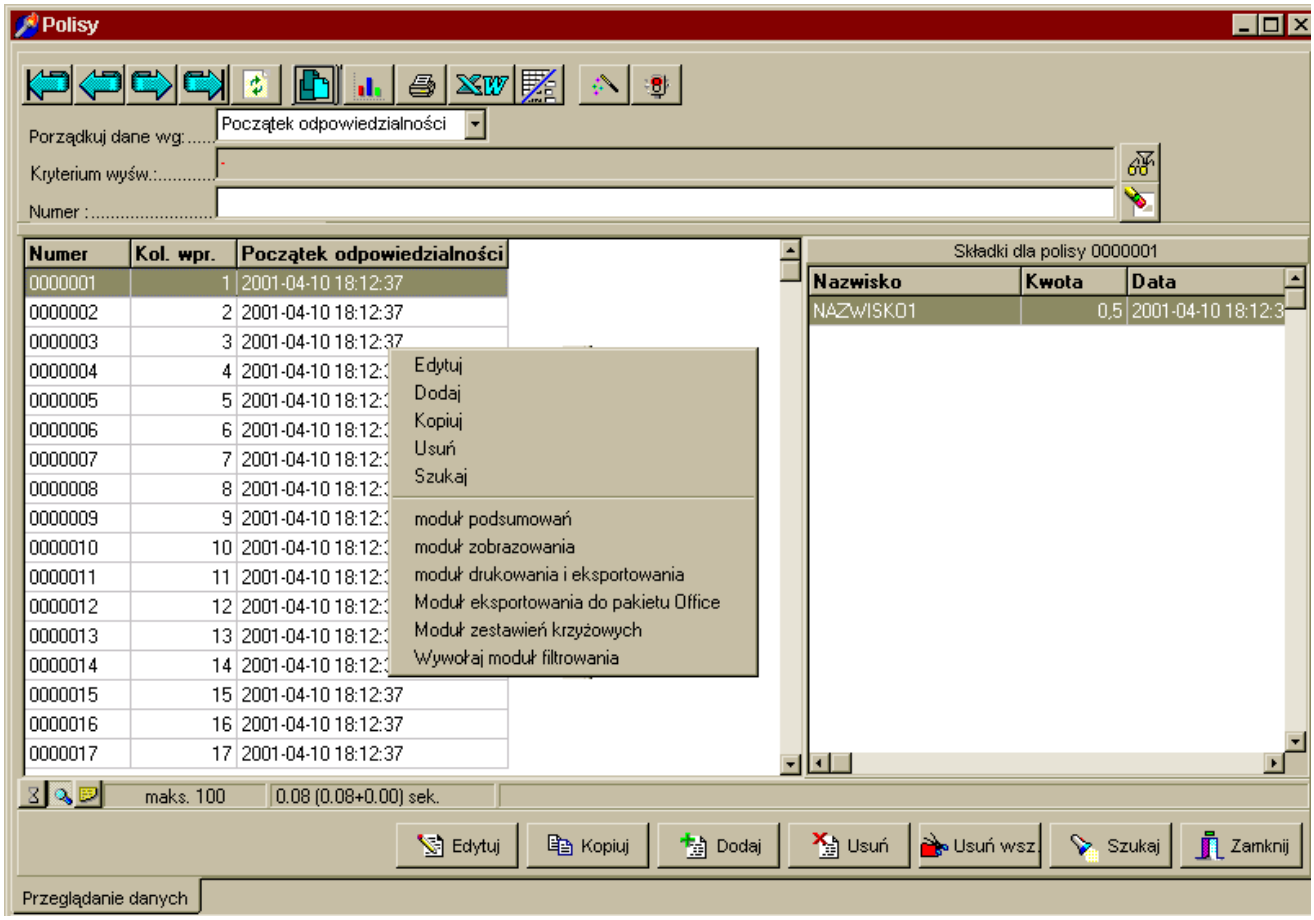
Dzięki otwartości środowiska Delphi wzorzec bazowy, wzorzec formularza bazy danych i pozostałe formularze standardowe, np. moduł filtrowania, moduł drukowania, moduł podsumowań itd. mogą być współdzielone przez wiele aplikacji. Dzięki temu ulepszenie formularza zmienia funkcjonalność nie jednej lecz wielu aplikacji.



Właściwości wzorca formularza

Uwaga: Właściwości wzorca formularza w tym artykule zostały omówione bardzo ogólnie. Aby dowiedzieć się więcej na temat wzorca formularza, uruchom prezentacje lub przykładową aplikację.

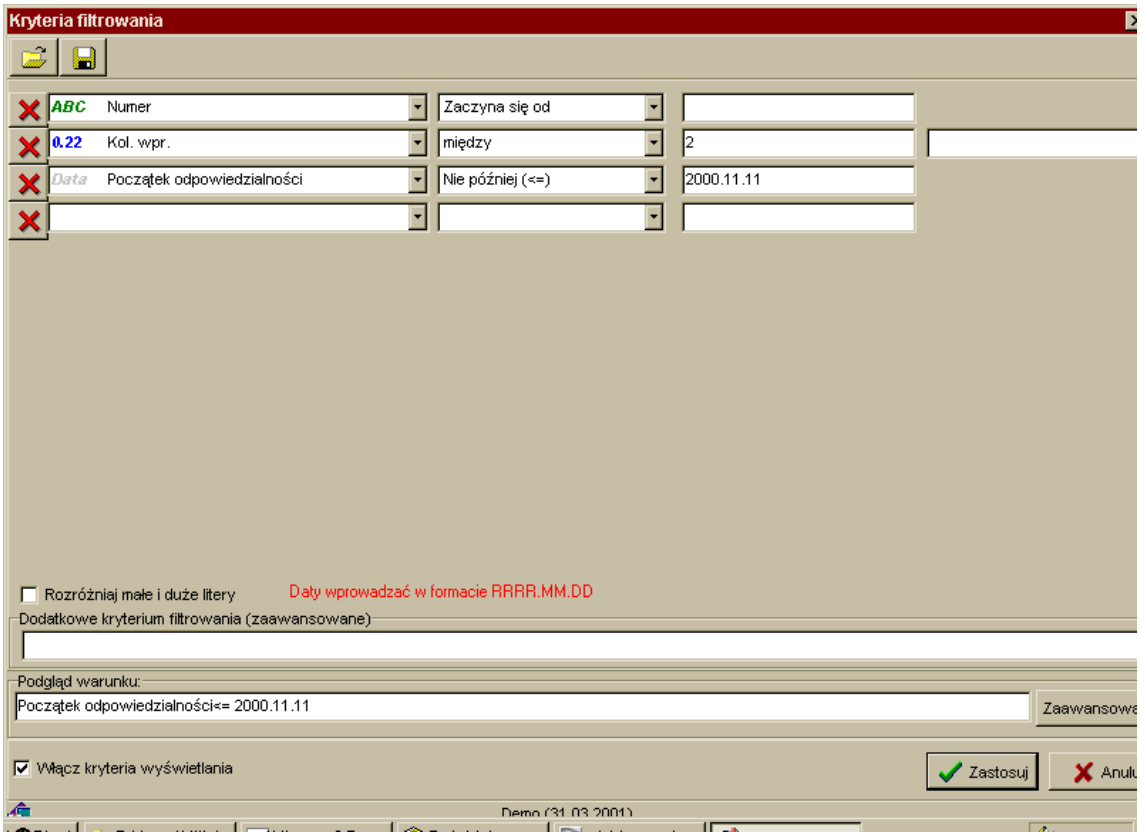
Poniższy rysunek przedstawia wygląd przykładowego formularza, zbudowanego na bazie wzorca formularza bazodanowego.



Wzorzec formularza zapewnia:

- Tworzenie wielojęzycznych aplikacji. Wizualne atrybuty okna są przechowywane w pliku konfiguracyjnym;
- Możliwość wywołania formularza w trybach: przeglądania, wybierania rekordu, wybierania wielu rekordów, wykonania akcji na jednym rekordzie;
- Jednolity wygląd formularza, tj. siatkę z rekordami z menu podręcznym, standardowe przyciski w stopce formularza, standardowy pasek narzędzi, pasek statusu z informacjami o maks. liczbie wyświetlanych rekordów i czasach dostępu do danych;
- Obsługę i rejestrację błędów w logu systemowym;
- Rejestrację zbyt długiego czasu odpowiedzi systemu w logu systemowym;
- Zliczanie liczby rekordów, określanie maksymalnej liczby rekordów pobieranych z serwera;

- Wywołanie modułu filtrowania danych, zapewniającego wyszukiwanie danych wg kryteriów wprowadzonych przez użytkownika:



- Wyszukiwanie rekordów na postawie wprowadzonych liter;
- Sortowanie rekordów wg jednego ze zdefiniowanych porządków sortowania;
- Wywołanie modułów: wydruków, wykresów, podsumowań, eksportów do pakietu Office, zestawień krzyżowych, ustawień konfiguracyjnych formularza.

Warunki korzystania z framework

Framework IMPET jest udostępniany za darmo i bez żadnych ograniczeń. Może być stosowany również w celach komercyjnych. Framework może zostać pobrany ze strony www firmy Software Factory.

Firma Software Factory udziela wsparcia technicznego w zakresie posługiwania się framework oraz w zakresie tworzenia aplikacji bazodanych w Delphi. Opłata za wsparcie techniczne wg aktualnego cennika.

Wskazówki dotyczące tworzenia modelu danych

Nazewnictwo obiektów bazy danych

Gdy nad fizycznym modelem bazy danych pracuje zespół projektantów, nieodzowne jest przyjęcie jednolitych zasad nazewnictwa obiektów w bazie danych. Praktyka pokazuje, że najistotniejsze z nich to:

- Przestrzegać stosowania nazewnictwa w jednym języku. Preferowany język: angielski.
- Nazwa tabeli musi być w liczbie mnogiej. Każda tabela musi mieć ALIAS.
- Każda tabela musi mieć jednokolumnowy klucz główny ID NUMBER
- Nazwy atrybutów muszą być jasne i zrozumiałe
- Jeżeli nazwa atrybutu jest kilkuczłonowa, człony łączyć znakiem "_"

- Stosować wyłącznie duże litery, nie stosować znaków diakrytycznych
- Jeżeli atrybuty stanowią grupę (np. adres), grupę poprzedzać prefiksem (patrz przykład)
- Nie stosuj żadnych nowomodnych tricków, np. obiektowych typów danych, w innym razie napotkasz na problemy implementacyjne
- Z każdą tabelą związana jest sekwencja o nazwie <ALIAS TABELI>_SEQ (Oracle)

Przykład:

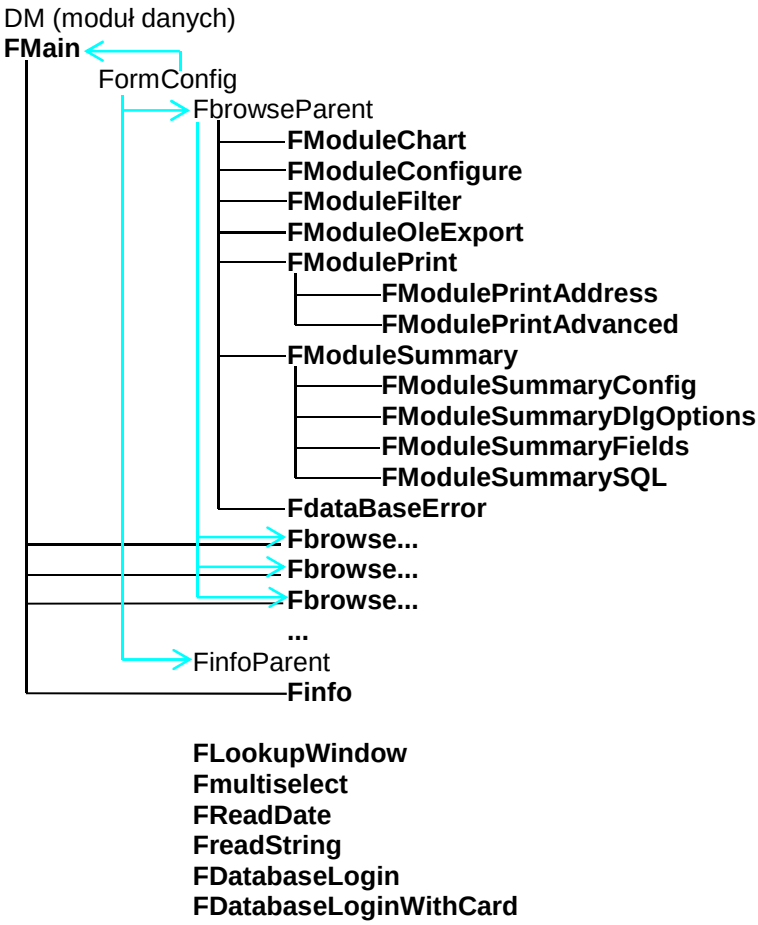
```
Create table CUSTOMERS
 (ID INTEGER
 ,FIRST_NAME            VARCHAR2(30),
 ,ADDRESS_STREET        VARCHAR2(200));
 ,ADDRESS_CITY          VARCHAR2(200));
 ,ADDRESS_COUNTRY       VARCHAR2(200));

Create Sequence CUS_SEQ;
```

Budowanie fizycznego modelu danych

Przy tworzeniu fizycznego modelu danych warto przestrzegać zasad:

- w celu zmniejszenia rozmiaru bazy danych kolumny, w których występują z reguły puste wartości umieszczać na końcu. Większość baz danych rezerwuje dla rekordu rozmiar rzeczywisty + założony margines, a nie rozmiar nominalny;
- stosować powiązania 1-1 w celu umieszczania rzadziej używanych atrybutów w innych przestrzeniach tabel. Większość baz danych pobiera dane z całego bloku. Dzięki umieszczeniu rzadko używanych atrybutów w innej przestrzeni nie będą one obciążały systemu;
- odpowiednio dobrać parametry PRC FREE i USED (dotyczy serwera Oracle). Zostawienie pustych przestrzeni w bazie zapobiegnie defragmentacji bazy danych;
- tabele i indeksy umieszczać w osobnych przestrzeniach tabel. Dzięki temu możliwe jest zrównoleglenie pracy serwera przez przyporządkowanie przestrzeni tabel osobnym zasobom sprzętowym. Drugim powodem jest różny stopień defragmentacji dla indeksów i dla tabel;
- Dobrać odpowiednią liczbę segmentów wycofania; ich liczba zależy liniowo od liczby użytkowników;
- Ograniczyć audyt do niezbędnych zdarzeń. Wykonywanie audytu obciąża zasoby sprzętowe;
- Dobrać odpowiednio przestrzenie sortowania, pamięci współdzielonej. Za mała pamięć powoduje migotanie stron, za duża powoduje wykorzystanie wirtualnej a nie fizycznej przestrzeni pamięci komputera i w rezultacie zwolnienie pracy serwera bazy danych.

Opis powiązań między formularzami framework IMPET



Legenda:
Linie czarne – schemat wywołań modułów
Linie błękitne – schemat dziedziczenia modułów
Pogrubione – moduł widziany przez użytkownika końcowego
Nie pogrubione – moduł dziedziczony przez inne moduły, nie widziany przez użytkownika końcowego

Opis poszczególnych modułów framework

DM - Moduł danych

Moduł zawiera komponent DataBase : TdataBase oraz komponenty Qwork : TQuery.
Wszystkie kontakty z bazą danych realizowane są przez połączenie udostępniane przez DataBase.
Komponenty Qwork służą do wykonywania operacji na bazie danych. Służą do tego metody obiektu DModule:

SQL (wykonuje podane polecenie SQL stosując Qwork1),
OpenSQL(otwiera kursor na podstawie podanego polecenia SELECT stosując Qwork1),
OpenSQL2, OpenSQL3(jw. stosując Qwork2, Qwork3).
SingleValue (zwraca wartość zapytania. Zapytanie powinno zwrócić jeden rekord i jedną kolumnę. Stosuje Qwork1)
SingleValue2 (jw. Stosuje Qwork2)
SingleValue3 (jw. Stosuje Qwork3)

Tworząc aplikacje przy użyciu pakietu IMPET pamiętaj, że moduł danych zawiera wyłącznie komponent do połączenia z bazą danych i kwerendy pomocnicze. Kwerendy z konkretnymi zapytaniami SQL znajdują się na stosownych formularzach (Zob. FbrowseParent.Query). Jeżeli korzystasz z tzw. Lookupów, również umieszczaj je na stosownych formularzach. Ideowo podejście jest zgodne z metodyką firmy Oracle, gdzie dla konkretnego modułu tworzy się model danych. Zatem model danych jest powiązany z formularzem, a nie z modułem danych.

Fmain

Okno główne systemu. Zawiera menu z funkcjami: Wyjście i O programie oraz przycisk zamknij.

UformConfig

W module zdefiniowano formularz FormConfig dziedziczony przez wszystkie formularze aplikacji.
Zawiera on wszystkie mechanizmy wspólne dla wszystkich formularzy.
Zapewnia on przechowywanie właściwości okien w tekstowych plikach konfiguracyjnych.
Pliki konfiguracyjne przechowywane są w lokalizacji: C:\Documents and Settings\..
Nazwa pliku konfiguracyjnego: Application.Title + '.' + Form.Name + '.'+ Extension
Każdy moduł w aplikacji powinien być jego potomkiem.

Czasem zachodzi konieczność przywrócenia pierwotnej postaci pliku konfiguracyjnego, np. gdy użytkownik wprowadził nieprawidłowe kryterium filtrowania danych powodujące wystąpienie błędu i nie potrafi sobie poradzić samodzielnie z problemem. Wówczas, gdy pojawi mu się komunikat o błędzie (okno FDatabaseError) , powinien nacisnąć przycisk **Napraw Automatycznie**. Spowoduje to zapisanie z pliku rejestru informacji o tym, żeby przy kolejnym otwieraniu aplikacji jednorazowo nie odczytywać zawartości pliku konfiguracyjnego. Dzięki temu przy ponownym uruchamianiu aplikacji nie zostaną odczytane nieprawidłowe kryteria filtrowania.
Można także wyjść z aplikacji i ręczenie usunąć plik.

Udostępniane elementy

Klasa TformConfig (TForm)
W obecnej wersji formularz FormConfig zawiera:

- Pasek statusu na którym wyświetlana jest nazwa aplikacji i nazwa użytkownika
- Przycisk Edytuj plik konfiguracyjny formularza
- Zdarzenia OnCreate (odczytanie pliku konfiguracyjnego), OnClose (zachowanie pliku konfiguracyjnego), OnShow (odświeżenie paska statusu)

Function GetFileNameWithExtension(Form : TForm; Extension : ShortString) : ShortString;
Funkcja zwraca pełną ścieżkę dostępu dla podanego formularza i rozszerzenia,
np. GetFileNameWithExtension(Self,'cfg')

Procedure SaveFormConfiguration(Form : TForm);
Procedura zachowuje ustawienia podanego formularza w pliku konfiguracyjnym

Procedure LoadFormConfiguration(Form : TForm);
Procedura odczytuje ustawienia podanego formularza w pliku konfiguracyjnym

Uwagi

Przechowaniu w pliku konfiguracyjnym podlega:
TForm.WindowState
TForm.Caption
TForm.Left
TForm.Top
TForm.Width
TForm.Height
TForm.HelpFile
TForm.HelpContext
Tcontrol.Hint
Tcontrol.ShowHint
Tlabel.Caption
TcomboBox.ItemIndex
TRxDBLookupCombo.DisplayEmpty
Tbutton.Caption

TmenuItem.Caption
TcheckBox.Caption
TcheckBox. Checked (gdy .Tag And 67108864 = 67108864)
TradioButton.Caption
TgroupBox.Caption
TradioGroup.Caption
TradioGroup.ItemIndex
TradioGroup.ItemIndex
TtabbedNoteBook. (obsługa wyłączona)
Tfield.DisplayLabel
Tnotebook.Pages
TtoolButton.Caption
TspeedButton.Caption
TtabSheet.Caption
TDBGrid. .Columns[.].Title.Caption (gdy NIE .Tag And 33554432 = 33554432)
TtabSet.Tabs
Tpanel.Caption
TChart .Title.Text
TDBChart.Title.Text
Tedit.Text (gdy .Tag And 67108864 = 67108864)
TdateEdit.DialogTitle
TdateEdit.Text (gdy .Tag And 67108864 = 67108864)
TspinEdit.Value (gdy .Tag And 67108864 = 67108864)
Tmemo.Lines (gdy .Tag And 67108864 = 67108864)
TrxQuery.SQL (procedura próbuje przywrócić właściwość .Active)
TstrHolder.Strings (gdy .Tag And 67108864 = 67108864)

FbrowseParent

Moduł udostępnia formularz FbrowseParent. Formularz jest podstawą tworzenia modułów do modyfikacji bazy danych i zapewnia następujące cechy:

- Implementacja praw dostępu do danych (Wirtualne funkcje CanShow, CanEditIntegrity, CanEditPermission, CanInsert, CanDelete, CanConfigure)
- Wywołanie okna w trybach:
  - Przeglądania i aktualizacji rekordów (Procedure ShowModalAsBrowser)
  - Wybierania rekordu z listy (Function ShowModalAsSelect(Var aID : ShortString) : TModalResult)
  - Wybierania wielu rekordów z listy (Function ShowModalAsMultiSelect(Var aIDs : String) : TModalResult; //Klucze zwracane są w postaci : 1|2|3... użyj funkcji WordCount oraz ExtractWord do ich odczytania w pętli)
  - Dodawania, aktualizacji lub usuwania pojedynczego rekordu (Function ShowModalAsSingleRecord(Action : Integer; Var ID : ShortString) : TModalResult)
- Wywoływanie modułów
  - wyszukiwania (filtrowania) danych
  - podsumowań
  - zobrazowania graficznego
  - Drukowania
  - Eksportowania danych do Worda, Excela i notatnika
  - Dostosowywania wyglądu okna i parametrów modułu
  - Dostosowania parametrów
- Ergonomiczny dostęp do danych
  - Sortowanie danych wg wybranych kryteriów
  - Wyszukiwanie za pomocą tzw. lokatora inkrementalnego
  - Przyciski Edytuj, Kopiuj, Dodaj itd. wraz ze standardową obsługą
  - Menu podręczne siatki
  - Skróty klawiszowe (INSERT, DELETE, ENTER, F8 itd.)
  - Pozyskiwanie z bazy danych tylko określonej liczby rekordów
  - Przechowywanie zmiany położenie i rozmiarów okna
- Estetyczny i jednolity w ramach aplikacji wygląd modułu (Ikony, podpowiedzi, menu podręczne)

Ponadto formularz ma zaszyte różne mechanizmy eliminujące typowe problemy występujące podczas tworzenia aplikacji bazodanych w Delphi i jest wynikiem długoletnich doświadczeń autora w tej dziedzinie np.

- Utrzymanie focusu podczas dodawania, edycji, usuwania rekordu,
- Eliminacja błędu Invalid BLOB cursor handle przez odłączanie pól memo,
- Blokowanie nieaktywnych przycisków i pozycji menu podręcznego (np. polecenia usuń dla pustej listy)
- Można zmieniać rozmiary okna tylko w założonym zakresie
- Obsługa zdarzenia usunięcia rekordu przez kogoś innego
- Ustawienie dla formularza: KeyPreview = True, Position = poScreenCenter, Scaled = False

FmoduleChart, FModuleConfigure, FmoduleFilter, FmoduleOleExport, FmodulePrint, FmodulePrintAddress, FmodulePrintAdvanced, FmoduleSummary, FmoduleSummaryConfig, FmoduleSummaryDlgOptions, FmoduleSummaryFields, FModuleSummarySQL, FdataBaseError

Moduły stanowią integralną część pakietu, są wywoływane przez FbrowseParent. Nie zostały odokumentowane, ponieważ ich znajomość nie jest niezbędna dla stosowania pakietu IMPET.

Niezależne użycie FmoduleFilter

Inicjalizacja

```
================
 KolumnyFiltrowane.Strings.CommaText :=
  '"DESC1 (niewidoczne)|DESC1|ftString",'
 +'"DESC2 (niewidoczne)|DESC2|ftString",'
 +'"GROUP_TYPE (niewidoczne)|GROUP_TYPE|ftString",'
 +'"Kol. wpr.|ID|ftFloat",'

+'Kolor|COLOUR|ftFloat,Licznoœæ|NUMBER_OF_PEOPLES|ftFloat,Nazwa|NAME|ftString,Skrót|ABB
REVIATION|ftString,'
 +'"Typ grupy|GROUP_TYPE_NAME|ftString"';
 initializeFilerSettings ( UstawieniaFiltra.strings ) ;
 UFModuleFilter.GetSQLAndNotes(UstawieniaFiltra.Strings, KolumnyFiltrowane.Strings);
```

pobranie biezacych ustawien

```
============================
 filer := UstawieniaFiltra.Strings.Values['Notes'];
 sql   := UstawieniaFiltra.Strings.Values['SQL']
```

wywolanie okna

```
================
 If UFModuleFilter.ShowModal(UstawieniaFiltra.Strings, KolumnyFiltrowane.Strings) = mrOK Then
Begin
   ...
 End;
```

FinfoParent

Moduł do tworzenia okna **O programie**

FlookupWindow

Moduł służy do wywoływania okna dialogowego do wyboru rekordu z listy rekordów.

Udostępniane elementy

Function LookupWindow(aTableName, aKeyField, aDisplayFields, aDisplayCaption, aFindField, aWhereClause, aDefaultMask : ShortString; Var KeyValue : ShortString) : TModalResult;
Funkcja służy do wywoływania okna dialogowego do wyboru rekordu z listy rekordów.

ATableName	**Nazwa tabeli (lub nazwy tabel zgodnie ze składnią SQL)**
AKeyField	**Klucz tabeli, domyślnie ID**
ADisplayFields	**Kolumny, które zostaną wyświetlone**
ADisplayCaption	**Nagłówek dla kolumn**
AFindField	**Kolumna, po której odbywa się szukanie i sortowanie rekordów (pierwszych 5000)**
AWhereClause	**Warunek WHERE**
ADefaultMask	**Domyślna maska, domyślnie %**
KeyValue	**Zwracana wartość klucza wybranego rekordu (gdy Result = mrOK)**
Result	**mrOK lub mrCancel**

Uwagi:

- Polecenie SQL konstruowane jest w następujący sposób:
  `'SELECT '+KeyField+','+DisplayFields+' FROM '+TableName+' WHERE '+FindField+' LIKE '''+Percent(BFILTER.Text)+''' AND '+WhereClause+' AND ROWNUM < 5000 ORDER BY '+FindField`
- W razie niepowodzenia pojawia się komunikat o błędzie a polecenie SQL zostaje umieszczone w schowku.

Przykłady:

- `Result := LookupWindow('WYKAZ_DYZUROW','WYD_ID','NAZWA_DYZURU','Nazwa dyżuru','NAZWA_DYZURU','JED_JED_ID='+CONJEDNOSTKA.TEXT,'%',KeyValue);`
- `Result := LookupWindow('OSOBY','OSO_ID','NAZWISKO','Nazwisko','NAZWISKO','STA_STA_ID IN (SELECT STA_ID FROM STANOWISKA WHERE JED_JED_ID = '+JED_ID+')','%',KeyValue);`
- Złączenie dwóch tabel
  `Result := LookupWindow('OSOBY_WYKAZ_DYZUROW OSOWYD, OSOBY','OSOWYD.OSO_OSO_ID','NAZWISKO','Nazwisko','NAZWISKO','OSOWYD.OSO_OSO_ID = OSO_ID AND WYD_WYD_ID = '+WYD_ID,'%',KeyValue);`

Fmultiselect

Okno do wybierania wielu wartości z listy wartości.

FreadDate

Okno do pobierania daty z klawiatury.

FreadString

Okno do pobierania ciągu znaków z klawiatury.

FdatabaseLogin

Okno do wprowadzenia nazwy użytkownika i hasła.

FDatabaseLoginWithCard

Okno do wprowadzenia nazwy użytkownika, hasła i pliku identyfikacyjnego – karty dostępu.

Wskazówki dotyczące nazywania formantów

- Formularze nazywać rozpoczynając od litery F, moduły zachowywać UF

Tworzenie nowego formularza

1. Utworzyć potomka FBrowseParent
   File/ New/ Karta <Nazwa aplikacji>/ FBrowseParent
2. Wpisać Form.Caption, Form.Name, Unit.Name
   Form.Name = FBrowse+ <nazwa tabeli>,
   Unit.Name = UFBrowse + <nazwa tabeli>.
3. Wpisać kwerendę w Query.SQL
   Query.Active := True (jeżeli jest błąd unkown database, to wyświetl moduł DM i ADOConnection ustaw na true. Na końcu z powrotem ADOConnection ustaw na false)
   Nie zassysać pól w Query
   Nie usuwać %TOPRECORDS , %SortOrder, %Conditionals, %Search
   Jeżeli przewiduje się wykorzystanie uruchamianie programu z kilku kont, poprzedzać nazwy tabel nazwą schematu np. KON.ARTYKULY.
4. Poprawić wyglad siatki
   Zassać pola (Menu podręczne/ Columns Editor/ Add All Fields)
   Zmienić nagłówki (Title/ Caption - format „Jak w zdaniu”, NIE „Jak W Zdaniu”)
   Pobrać również kolumny attribute1..attribute15
5. opcjonalnie [Uzupełnić Holdery: Kolumny, SortOrder]
6. Położyć komponenty (TDBEdit itd.) i podłączyć je pod Source
   Użyj aplikacji Case, lub wykonaj czynność ręcznie
   Komponent.Nazwa := <nazwa atrybutu z tabeli>,
   Etykieta.Nazwa := L + <nazwa atrybutu z tabeli>
   Etykieta.Font.Color := clRed dla atrybutów wymaganych
   MaxLength dla TDBEdit i innych komponentów
   [Wpisać zapytania w Lookupy, Active ustawić na False
   Lookupy zostaną automatycznie otwarte przy otwieraniu okna]
7. Poprawić TabOrder i rozmiary okna
8. [Udostępnić przyciski BPodrzedne (Visible = True)]

```
Przykładowa zawartość metody OnClick:
AutoCreate.PAYMENTS_PIECESShowModalAsBrowser('',Query['ID']);
If MainPage.ActivePage = Update Then Begin
  BRefreshRESTClick(nil);                    ← odświeżaj kolumny wyliczane rekordu
 End Else BOdswiezClick(nil);          ← odświeżaj wygląd siatki
```

9. Query.Active := False
10. Wersja uproszczona pkt 11 i 12 procedury:

```
 public
   Function  CheckRecord : Boolean; override;
   Procedure DefaultValues; override;
uses DM, UUtilityParent;
Function  TFBrowseARTYKULY.CheckRecord : Boolean;
Begin
  Result := CheckValid.ReducRestrictEmpty(Self, [NAZWA, DOS_ID]);
End;

Procedure TFBrowseARTYKULY.DefaultValues;
Begin
 Query.FieldByName('DOS_ID').AsString := CONDOSTAWCY.Value;
 Query['ID'] := DModule.SingleValue('SELECT KON.ART_SEQ.NEXTVAL FROM DUAL');
 If SygnaturaUzytkownika <>'' Then Query['SYGNATURA'] := SygnaturaUzytkownika;
```

```
End;
```

11. Dopisać CheckRecord

```
public Function  CheckRecord : Boolean; override;, np.:
Function  TFBrowse<NAZWA TABELI>.CheckRecord : Boolean;
Begin
 With UUtilityParent.CheckValid Do Begin //Metody: Init, AddError(S : String), AddWarning(S : String),   ShowMessage :
Boolean = czy wszystko jest ok ?
 Init(Self);
 RestrictEmpty([ZAM_ID, ART_ID ]);
 If Empty(NAZWA.TEXT) Then AddError(MESSAGES.Strings[0]);
 If Empty(IDAPARAT.Value)    Then AddError(MESSAGES.Strings[0]);
 If Empty(IDPRAC.Value) Then AddError(MESSAGES.Strings[1]);
 Try
 P_DNAST.CheckValidDate;
 If P_DNAST.Date = 0  Then AddWarning(MESSAGES.Strings[2]);
 Except AddError(MESSAGES.Lines[3]); End;
 {komunikaty formułować następująco : NAZWA POLA (dużymi literami) musi zostać wpisana/wybrana  (zależnie, czy pole
to EDIT czy COMBO)}
 Result := ShowMessage;
 End;
End;
```

UWAGA:W MESSAGES są przechowyane w pliku tekstowym. Można tu przechowywać dowolne komunikaty.

12. Dopisać: DefaultValues:

```
Query.FieldByName('ID').AsString := DModule.SingleValue('SELECT AKW_ADMIN_UPRAWNIENIA_SEQ.NEXTVAL
FROM DUAL');
```

13. Umieścić moduł w AvailableForms

Project/ Options

14. W module AutoCreate dopisać:

```
interface:
procedure <NAZWA TABELI>Create;
Procedure <NAZWA TABELI>ShowModalAsBrowser;
Function <NAZWA TABELI>ShowModalAsSelect(VAR ID : ShortString) : TModalResult;
Procedure <NAZWA TABELI>Free;

implementation:
Uses UFBrowse<NAZWA TABELI>

procedure <NAZWA TABELI>Create;
begin
If Not Assigned(FBrowse<NAZWA TABELI>) Then FBrowse<NAZWA TABELI> := TFBrowse<NAZWA
TABELI>.Create(Application);
end;

Procedure <NAZWA TABELI>ShowModalAsBrowser;
Begin
 <NAZWA TABELI>Create;
 FBrowse<NAZWA TABELI>.ShowModalAsBrowser('');
 If GetSystemParam('SAVERESOURCES') = 'Yes' Then <NAZWA TABELI>Free;
End;

Function <NAZWA TABELI>ShowModalAsSelect(VAR ID : ShortString) : TModalResult;
Begin
 <NAZWA TABELI>Create;
 Result := FBrowse<NAZWA TABELI>.ShowModalAsSelect(ID);
 If GetSystemParam('SAVERESOURCES') = 'Yes' Then <NAZWA TABELI>Free;
End;

Procedure <NAZWA TABELI>Free;
Begin
 If Assigned(FBrowse<NAZWA TABELI>) Then If FBrowse<NAZWA TABELI>.Visible Then Exit;
 If Assigned(FBrowse<NAZWA TABELI>) Then FBrowse<NAZWA TABELI>.Free;
 FBrowse<NAZWA TABELI> := Nil;
End;
```

15. Zdefiniuj nieaktualizowalne kolumny

```
procedure TFBrowseREQ_PAYMENTS.FormCreate(Sender: TObject);
begin
  inherited;
  SetNotUpdatable([FAK_ID, BSELECTFAK_ID, BClearEXKND_ID, AMOUNT, AMOUNT_PERCENT], [LabelFAK_ID,
LabelAMOUNT, LabelAMOUNT_PERCENT]);
end;
```

16. Ewentualnie nadpisać metody (wszystkie override w sekcji public)

```
 Procedure DodajClick;
 Procedure EdytujClick;
 Procedure UsunClick;
 Procedure UsunAllClick;
 Procedure BeforeDelete;
 Function  CanShow      : Boolean;
 Function  CanEditIntegrity : Boolean;
 Function  CanEditPermission : Boolean;
 Function  CanInsert    : Boolean;
 Function  CanDelete    : Boolean;
 Function  CanConfigure : Boolean;
 Function  CheckRecord  : Boolean;
```

```
Procedure DefaultValues;
Procedure BeforeEdit;
Procedure CustomConditionals;

Np.
Procedure TFBrowse+<nazwa tabeli>.BeforeDelete; override;
Begin
 SQL('DELETE FROM P_TYRODZ WHERE IDTYP='+ViewQuery.FieldByName(KeyField).AsString);
End;

Procedure TFUpd+<nazwa tabeli>.DefaultValues; Override  //Wartości domyślne podczas dodawania nowego rekordu
Begin
 Query.FieldByName('IDAPA').AsString := ID.Value;
End;
- Dopisać: OnFormCreate: FirstControl := <pierwszy komponent>;  (Domyślnie: FirstControl = Formant z karty Update o
TabOrder = 0)
```

17. Dmodule.database.connected = false
18. Obsługa dodatkowych atrybutów:

Dodać do sekcji others tłumaczenie aliasów:

```
ALIAS:ATTRIBS_01=LECTURERS.ATTRIBS_01
ALIAS:ATTRIBS_02=LECTURERS.ATTRIBS_02
ALIAS:ATTRIBS_03=LECTURERS.ATTRIBS_03
ALIAS:ATTRIBS_04=LECTURERS.ATTRIBS_04
ALIAS:ATTRIBS_05=LECTURERS.ATTRIBS_05
ALIAS:ATTRIBS_06=LECTURERS.ATTRIBS_06
ALIAS:ATTRIBS_07=LECTURERS.ATTRIBS_07
ALIAS:ATTRIBS_08=LECTURERS.ATTRIBS_08
ALIAS:ATTRIBS_09=LECTURERS.ATTRIBS_09
ALIAS:ATTRIBS_10=LECTURERS.ATTRIBS_10
ALIAS:ATTRIBS_11=LECTURERS.ATTRIBS_11
ALIAS:ATTRIBS_12=LECTURERS.ATTRIBS_12
ALIAS:ATTRIBS_13=LECTURERS.ATTRIBS_13
ALIAS:ATTRIBS_14=LECTURERS.ATTRIBS_14
ALIAS:ATTRIBS_15=LECTURERS.ATTRIBS_15
ALIAS:ATTRIBD_01=LECTURERS.ATTRIBD_01
ALIAS:ATTRIBD_02=LECTURERS.ATTRIBD_02
ALIAS:ATTRIBD_03=LECTURERS.ATTRIBD_03
ALIAS:ATTRIBD_04=LECTURERS.ATTRIBD_04
ALIAS:ATTRIBD_05=LECTURERS.ATTRIBD_05
ALIAS:ATTRIBD_06=LECTURERS.ATTRIBD_06
ALIAS:ATTRIBD_07=LECTURERS.ATTRIBD_07
ALIAS:ATTRIBD_08=LECTURERS.ATTRIBD_08
ALIAS:ATTRIBD_09=LECTURERS.ATTRIBD_09
ALIAS:ATTRIBD_10=LECTURERS.ATTRIBD_10
ALIAS:ATTRIBD_11=LECTURERS.ATTRIBD_11
ALIAS:ATTRIBD_12=LECTURERS.ATTRIBD_12
ALIAS:ATTRIBD_13=LECTURERS.ATTRIBD_13
ALIAS:ATTRIBD_14=LECTURERS.ATTRIBD_14
ALIAS:ATTRIBD_15=LECTURERS.ATTRIBD_15
ALIAS:ATTRIBN_01=LECTURERS.ATTRIBN_01
ALIAS:ATTRIBN_02=LECTURERS.ATTRIBN_02
ALIAS:ATTRIBN_03=LECTURERS.ATTRIBN_03
ALIAS:ATTRIBN_04=LECTURERS.ATTRIBN_04
ALIAS:ATTRIBN_05=LECTURERS.ATTRIBN_05
ALIAS:ATTRIBN_06=LECTURERS.ATTRIBN_06
ALIAS:ATTRIBN_07=LECTURERS.ATTRIBN_07
ALIAS:ATTRIBN_08=LECTURERS.ATTRIBN_08
ALIAS:ATTRIBN_09=LECTURERS.ATTRIBN_09
ALIAS:ATTRIBN_10=LECTURERS.ATTRIBN_10
ALIAS:ATTRIBN_11=LECTURERS.ATTRIBN_11
ALIAS:ATTRIBN_12=LECTURERS.ATTRIBN_12
```

ALIAS:ATTRIBN_13=LECTURERS.ATTRIBN_13
ALIAS:ATTRIBN_14=LECTURERS.ATTRIBN_14
ALIAS:ATTRIBN_15=LECTURERS.ATTRIBN_15

UWAGI DODATKOWE:
- Moduł jest elementem pakietu IMPET; tworząc nową aplikację należy skorzystać z projektu ProjektBazowy, zawierający ten pakiet.
- obsługiwana tabela musi posiadać klucz unikalny oparty na jednej kolumnie(parametr Others.KeyField określa nazwę klucza unikalnego, domyślnie jest to ID)
- Moduł wywoływać za pomocą funkcji ShowModalAsBrowser, ShowModalAsSelect, ShowModalAsSingle zdefiniowanych w module AutoCreate
- w celu odświeżenia Enable/ disable dla przycisków wywołuj: UstalDostepnoscPrzyciskow

Najczęściej występujące zadania przy tworzeniu modułu

Siatka z rekordami podrzędnymi

Skopiuj rozwiązanie stąd: Tkaniny, moduł UFBrowseFaktury
Dodatkowa podpowiedź:

1. Dodaj QueryDetails, DSDetails
2. Dodaj TimerDetails (interval= 1000,)
3. Dodaj GridDetails i PanelDetails (powyżej QueryDetails) (rozdziel splitterem od okna głównego)

```
procedure TFBrowseFAKTURY.QueryAfterScroll(DataSet: TDataSet);
begin
  inherited;
   Counter := 3;
end;

procedure TFBrowseFAKTURY.TimerDetailsTimer(Sender: TObject);
Var ID : ShortString;
begin
  inherited;
  If Counter > 0 Then Counter := Counter - 1;
  If Counter = 1 Then Begin
      QueryDetails.Close;
      If Query.IsEmpty Then Begin
        QueryDetails.MacroByName('ID_FAKTURY').AsString := '= -1';
        PanelDetails.Caption := '';
      End Else Begin
        ID := NVL(Query.FieldByName('ID').AsString,'-1');
        QueryDetails.MacroByName('ID_FAKTURY').AsString := '= '+ID;
        PanelDetails.Caption := 'Pozycje faktury
'+Query.FieldByName('NUMER').AsString;
      End;
      QueryDetails.Open;
  End;
end;

procedure TFBrowseFAKTURY.CommisionFilterClick(Sender: TObject);
begin
  inherited;
  CommisionGroupBox.Visible := CommisionFilter.Checked;
  SearchCounter := 3;
end;

procedure TFBrowseFAKTURY.PayedCommisionChange(Sender: TObject);
```

```
begin
  inherited;
  Others.Strings.Values['PayedCommision'] :=
IntToStr(PayedCommision.ItemIndex);
  SearchCounter := 3;
end;
```

… podobnie dla pozostałych elementów które zmieniają zawartość siatki głównej

Wyświetlanie rekordów podrzędnych dla wybranego nadrzędnego (filtr)

Okno podrzędne:

Położyć Lookup :
Name = CON<TABELA MASTER>,
Owner = CustomPanel,
LookupSource = ...

Przycisk BOdswiez<TABELA MASTER>;, zdarzenie BOdswiez<TABELA MASTER>Click(nil):
DBUtils.RefreshQuery(...);

onChange: BOdswiezClick(nil);
UZUPEŁNIĆ Query o Makro: ... AND %CON<TABELA MASTER> AND ...

Tylko gdy istnieje bezpośrednie powiązanie:

```
Procedure TFBrowse<nazwa tabeli>.DefaultValues;
Begin
 Query.FieldByName('<ALIAS TABELI MASTER>_ID').AsString := CON<TABELA
MASTER>.Value;
 ...
End;

Procedure TFBrowse<nazwa tabeli>.CustomConditionals; override;
Begin
 If CON<ALIAS TABELI MASTER>.Value = '' Then Query.MacroByName('CON<ALIAS
TABELI MASTER>').AsString := '0=0'
                          Else Query.MacroByName('CON<ALIAS TABELI
MASTER>').AsString := '<ALIAS TABELI MASTER>_ID ='+CON<ALIAS TABELI
MASTER>.Value;
                               // inaczej gdy nie istnieje bezpośrednie
powiązanie
End;
```

AutoCreate:

```
Procedure <nazwa tabeli>ShowModalAsBrowser(aCON<NAZWA TABELI MASTER> :
ShortString);
Begin
 <nazwa tabeli>Create;
 With FBrowse<nazwa tabeli> Do Begin
  If aCON<NAZWA TABELI MASTER> <> '' Then BRefresh<NAZWA TABELI
MASTER>Click(nil);
  CON<NAZWA TABELI MASTER>.Value := aCON<NAZWA TABELI MASTER>;
  ShowModalAsBrowser;
  If GetSystemParam('SAVERESOURCES') = 'Yes' Then <nazwa tabeli>Free;
 End;
End;
```

```
Function <nazwa tabeli>ShowModalAsSelect(aCON<NAZWA TABELI MASTER> :
ShortString; VAR ID : ShortString) : TModalResult;
Begin
 <nazwa tabeli>Create;
 With FBrowse<nazwa tabeli> Do Begin
  If aCON<NAZWA TABELI MASTER> <> '' Then BRefresh<NAZWA TABELI
MASTER>Click(nil);
  CON<NAZWA TABELI MASTER>.Value := aCON<NAZWA TABELI MASTER>;
  Result := ShowModalAsSelect(ID);
  If GetSystemParam('SAVERESOURCES') = 'Yes' Then <nazwa tabeli>Free;
 End;
End;
```

Poprawić odwołania do AutoCreate

Naliczanie wielkości w rekordzie nadrzędnym zależnej od rekordów podrzędnych

Zwykle nie można tego zrobić całkowicie na bazie ze względu na problem "mutanting tables"

Idea:
Nadrzedny.beforePost
Podrzedny.Po dodaniu, aktualizacji (aktualizacja nowego i starego przydziału !!!), usunięciu

UCOMMON:

```
Procedure Get<NAZWA ATRYBUTU>For<TABELA MASTER>(ID : ShortString; Var
<NAZWA ATRYBUTU>: Extended);
Begin
  With DModule Do Begin
   SingleValue('SELECT NVL(SUM(CENA),0), NVL(SUM(CENA_EURO),0) FROM
POZYCJEZAMOWIEN WHERE ZAM_ID ='+ID);
   <NAZWA ATRYBUTU>     := QWork.Fields[0].AsFloat;
End;
End;
```

<TABELA MASTER>:

```
procedure TFBrowse<TABELA MASTER>.BRefreshRESTClick(Sender: TObject);
begin
  inherited;
  Query.FieldByName('<NAZWA ATRYBUTU>').AsFloat := UCommon.Get<NAZWA
ATRYBUTU>for<TABELA MASTER> (Query['ID']);
end;

Procedure TFBrowse<TABELA MASTER>.BeforePost;
Begin
 BRefresh<NAZWA ATRYBUTU>Click(nil);
End;

procedure TFBrowse<TABELA MASTER>.BPodrzedne1Click(Sender: TObject);
begin
  inherited;
  AutoCreate.PAYMENTS_PIECESShowModalAsBrowser('',Query['ID']);
  If MainPage.ActivePage = Update Then Begin
   BRefresh<NAZWA ATRYBUTU>Click(nil);
  End;
```

```
end;
```

<TABELA DETAIL>:

```
  private
    MASTER_<ALIAS MASTERA>_ID : ShortString;
    MASTER_PREV_<ALIAS MASTERA>_ID : ShortString;
    Procedure ExeUpdate;
  public
    Procedure AfterPost; override;
    Procedure BeforeDelete; override;
    Procedure AfterDelete; override;
    Procedure BeforeEdit; override;

Procedure TFBrowse<TABELA DETAIL>.BeforeEdit;
Begin
  MASTER_PREV_<ALIAS MASTERA>_ID := <ALIAS MASTERA>_ID.text;
End;

Procedure TFBrowse<TABELA DETAIL>.ExeUpdate;
Begin
 DMOdule.SQL('UPDATE ZAMOWIENIA X SET (WARTOSC_ZAMOWIENIA ,
WARTOSC_ZAMOWIENIA_EURO ) = (SELECT SUM(CENA), SUM(CENA_EURO) FROM
POZYCJEZAMOWIEN WHERE ZAM_ID = X.ID) WHERE ID ='+MASTER_<ALIAS
MASTERA>_ID);
 If Trim(MASTER_PREV_<ALIAS MASTERA>_ID) <> '' Then
   If MASTER_<ALIAS MASTERA>_ID <> MASTER_PREV_<ALIAS MASTERA>_ID Then
     DMOdule.SQL('UPDATE ZAMOWIENIA X SET (WARTOSC_ZAMOWIENIA ,
WARTOSC_ZAMOWIENIA_EURO ) = (SELECT SUM(CENA), SUM(CENA_EURO) FROM
POZYCJEZAMOWIEN WHERE ZAM_ID = X.ID) WHERE ID ='+MASTER_PREV_<ALIAS
MASTERA>_ID);
End;

Procedure TFBrowse<TABELA DETAIL>.AfterPost;
Begin
 MASTER_<ALIAS MASTERA>_ID := <ALIAS MASTERA>_ID.text;
 ExeUpdate;
End;

// gdyby wystąpił problem z kompilacją tego fragmentu, to nazwę komponentu
ID trzeba zmienić na ID_
Procedure TFBrowse<TABELA DETAIL>.BeforeDelete;
Begin
  MASTER_<ALIAS MASTERA>_ID := DMOdule.SingleValue('SELECT <ALIAS
MASTERA>_ID FROM <TABELA DETAIL> WHERE ID='+ID);
End;

Procedure TFBrowse<TABELA DETAIL>.AfterDelete;
Begin
 ExeUpdate;
End;
```

Implementacja uprawnień hierarchicznych

Każdy użytkownik posiada MASKA oraz SYGNATURA
W każdej tabeli jest kolumna SYGNATURA

```
Procedure DefaultValues;
Begin
 ...
```

```
 If FTest.Sygnatura <>'' Then Query['SYGNATURA'] := FTest.Sygnatura;
End;

Query.SQL: ... AND %MASKA AND ...

Procedure CustomConditionals;
Begin
  Query.MacroByName('MASKA').AsString := '('+TableName+'.SYGNATURA LIKE
'''+FTest.MASKA+''' OR '+TableName+'.SYGNATURA IS NULL )';
End;
```

Ustawienie formatu wyświetlania pól numerycznych

```
procedure TFBrowse<NAZWA TABELI>.QueryAfterOpen(DataSet: TDataSet);
begin
  TFloatField(Query.FieldByName('WARTOSC_ZAMOWIENIA')).DisplayFormat :=
'###,###,###,##0.00';
  TFloatField(Query.FieldByName('WARTOSC_ZAMOWIENIA')).EditFormat :=
'############0.00';
  // KONIECZNIE BEZ PRZECINKÓW !
end;
```

Uwagi do dbf

Ukryć przycisk Zatwierdź dla dBase
nie działa pole like '1%' dla pól numerycznych (co powoduje problemy z lokatorem inkrementalnym)
// CheckRecord dla DBF:

```
 If AkcjaAktualizacji in [AINSERT,ACOPY] Then
 If Trim(NRUM.TEXT) <> '' Then
 If DModule.SingleValue('SELECT COUNT(*) FROM '+TABLENAME+' WHERE
NRUM='+NRUM.TEXT) <> '0' Then AddError(MESSAGES.Strings[0]);
```

Odswiezanie rekordow podrzednych (siatka z rekordami odświeżana po sekundzie bezczynności)

Położyć QueryDetails, DSDetails, GridDetails, PANELDetails, TIMERDetails(Interval=1000)
- Query nie aktywne
- Query z makrem, Default "-1"

Const Counter : Integer = 3;

```
procedure TFBrowseZAMOWIENIA.QueryAfterScroll(DataSet: TDataSet);
begin
  inherited;
  Counter := 3;
end;

procedure TFBrowseFAKTURY.RefreshTimerTimer(Sender: TObject);
Var ID : ShortString;
begin
  inherited;
  If Counter > 0 Then Counter := Counter - 1;
  If Counter = 1 Then Begin
      QueryDetails.Close;
      If Query.IsEmpty Then Begin
        QueryDetails.MacroByName('ID_FAKTURY').AsString := '= -1';
```

```
        PanelDetails.Caption := '';
      End Else Begin
        ID := NVL(Query.FieldByName('ID').AsString,'-1');
        QueryDetails.MacroByName('ID_FAKTURY').AsString := '= '+ID;
        PanelDetails.Caption := 'Pozycje faktury
'+Query.FieldByName('NUMER').AsString;
      End;
      QueryDetails.Open;
  End;
end;
```

Positions - spec_authors - authors, roles

```
Procedure PutAuthorsAndRoles(JoinAuthors    : String; POS_ID  : String);
Var
    AuthorsCount   : Integer;
    AuthorAndRole  : ShortString;
    Authors        : Array[1..100] Of ShortString;
    Roles          : Array[1..100] Of ShortString;
    t              : Integer;
    AUT_ID         : ShortString;
    ROL_ID         : ShortString;
begin
  AuthorsCount := WordCount(JoinAuthors,[',']);
  For t := 1 To AuthorsCount Do Begin
   AuthorAndRole := Trim(ExtractWord(t,JoinAuthors,[',']));
   Authors[t] := Trim(ExtractWord(1,AuthorAndRole,['[']));
   Roles[t]   := Copy(Trim(ExtractWord(2,AuthorAndRole,['['])),1,1);
  End;

  DModule.SQL('DELETE FROM SPEC_AUTHORS WHERE POS_ID ='+POS_ID);
  For t := 1 To AuthorsCount Do Begin
     If Authors[t] <> '' Then Begin
       AUT_ID := NVL(DModule.SingleValue('SELECT ID FROM LIBRARY.AUTHORS
WHERE NAME='''+Authors[t]+''''),'NULL');
       IF AUT_ID = 'NULL' Then Begin
        AUT_ID := DModule.SingleValue('SELECT AUT_SEQ.NEXTVAL FROM DUAL');
        DModule.SQL('INSERT INTO LIBRARY.AUTHORS (ID, NAME, DESC1) VALUES
('+AUT_ID+','''+Authors[t]+''',''Dane dodane automatycznie'')');
       End;
     End Else AUT_ID := 'NULL';

     If Roles[t] <> '' Then Begin
       ROL_ID := NVL(DModule.SingleValue('SELECT ID FROM LIBRARY.ROLES
WHERE SHORTCUT='''+Roles[t]+''''),'NULL');
       IF ROL_ID = 'NULL' Then Begin
        ROL_ID := DModule.SingleValue('SELECT ROL_SEQ.NEXTVAL FROM DUAL');
        DModule.SQL('INSERT INTO LIBRARY.ROLES (ID, SHORTCUT, NAME, DESC1)
VALUES ('+ROL_ID+','''+Roles[t]+''','''+Roles[t]+''',''Dane dodane
automatycznie'')');
       End;
     End Else ROL_ID := 'NULL';

     try
     DModule.SQL('INSERT INTO SPEC_AUTHORS (ID, ROL_ID, AUT_ID, POS_ID)
VALUES (SPEAUT_SEQ.NEXTVAL,'+ROL_ID+','+AUT_ID+','+POS_ID+')');
     except end;
  End;
end;
```

```
Function GetAuthorsAndRoles(POS_ID : ShortString) : String;
Var JoinAuthors : String;
Begin
 JoinAuthors := '';
 With DModule Do Begin
 OPENSQL(
  'SELECT AUT.NAME, ROL.SHORTCUT '+
  'FROM  LIBRARY.SPEC_AUTHORS SPEAUT, '+
  '      LIBRARY.ROLES        ROL, '+
  '      LIBRARY.AUTHORS      AUT, '+
  '      LIBRARY.POSITIONS    POS '+
  'WHERE ROL_ID = ROL.ID(+) AND '+
  '      AUT_ID = AUT.ID    AND '+
  '      SPEAUT.POS_ID = POS.ID AND SPEAUT.POS_ID='+String(POS_ID)+
  ' ORDER BY AUT.NAME');

  JoinAuthors := '';
  QWork.First;
  While Not QWOrk.EOF Do Begin
   If JoinAuthors <> '' Then JoinAuthors := JoinAuthors + ', ';
   JoinAuthors := JoinAuthors + QWork.Fields[0].AsString;
   If QWork.Fields[1].AsString<>'' Then JoinAuthors := JoinAuthors + '[' +
QWork.Fields[1].AsString +']';
   QWork.Next;
  End;
 End;
 Result := JoinAuthors;
End;
```

POSITIONS:

```
Procedure TFBrowsePOSITIONS.BeforeEdit;
Begin
  Query.FieldByName('CALC_AUTHORS').AsString :=
GetAuthorsAndRoles(_ID.text);
End;

Procedure TFBrowsePOSITIONS.AfterPost;
Begin
 PutAuthorsAndRoles(CALC_AUTHORS.Text, _ID.text);
End;
```

SPEC_AUTHORS:

```
Procedure TFBrowseSPEC_AUTHORS.BeforeEdit;
Begin
  MASTER_PREV_POS_ID := POS_ID.Value;
End;

Procedure TFBrowseSPEC_AUTHORS.ExeUpdate;
Begin
 DMOdule.SQL('UPDATE LIBRARY.POSITIONS SET CALC_AUTHORS =
'''+GetAuthorsAndRoles(MASTER_POS_ID)+''' WHERE ID='+MASTER_POS_ID);
 If Trim(MASTER_PREV_POS_ID) <> '' Then
   If MASTER_POS_ID <> MASTER_PREV_POS_ID Then
     DMOdule.SQL('UPDATE LIBRARY.POSITIONS SET CALC_AUTHORS =
'''+GetAuthorsAndRoles(MASTER_PREV_POS_ID)+''' WHERE
ID='+MASTER_PREV_POS_ID);
End;
```

```
Procedure TFBrowseSPEC_AUTHORS.AfterPost;
Begin
 MASTER_POS_ID := POS_ID.Value;
 ExeUpdate;
End;

Procedure TFBrowseSPEC_AUTHORS.BeforeDelete;
Begin
  MASTER_POS_ID := DModule.SingleValue('SELECT POS_ID FROM
LIBRARY.SPEC_AUTHORS WHERE ID='+ID);
End;

Procedure TFBrowseSPEC_AUTHORS.AfterDelete;
Begin
 ExeUpdate;
End;
```

AUTHORS:

```
Procedure TFBrowseAUTHORS.AfterPost;
VAR POS_ID : ShortString;
Begin
  With DModule Do Begin
   OPENSQL('SELECT POS_ID FROM LIBRARY.SPEC_AUTHORS WHERE AUT_ID ='+ID);
   QWork.First;
   While Not QWOrk.EOF Do Begin
    POS_ID := QWork.Fields[0].AsString;
    SQL2('UPDATE LIBRARY.POSITIONS SET CALC_AUTHORS =
'''+UCommon.GetAuthorsAndRoles(POS_ID)+''' WHERE ID='+POS_ID);
    QWork.Next;
   End;
  End;
End;
```

Nie używać lokupcombo dla tabel > 5000 rekordów, zamiast tego

```
DBEdit.Name = <NAZWA POLA>
      .hint = <NAZWA POLA>
      .Visible := False;
      .DataSource := Source;
      .DataField  := <NAZWA POLA>
      .OnChange   := DModule.RefreshLookupEdit(Self,
TControl(Sender).Name,'TITLE',<nazwa tabeli nadrz>,'');

TEdit.Name = <NAZWA POLA>_value
     .ReadOnly = True

TBitBtn.Name := BSelect<NAZWA POLA>
       .HINT := "Wybierz daną z tabeli" ...
      .OnClick =
Var ID : ShortString;
begin
  inherited;
  ID := <NAZWA POLA>.Text;
  If AutoCreate.<NAZWA tabeli nadrz>ShowModalAsSelect(ID) = mrOK Then Begin
    Query.FieldByName(<NAZWA POLA>).AsString := ID;
  End;
end;
```

```
TBitBtn.Name := BClear<NAZWA POLA>
       .OnClick := Query.FieldByName(<NAZWA POLA>).Clear;
```

Uwagi:
Przykłady wywołania RefreshLookupEdit:
1. dla tabeli lookup bez lookupów
DModule.RefreshLookupEdit(Self, TControl(Sender).Name,'TITLE',<nazwa tabeli nadrz>,'');
2. dla tabeli lookup z lookupami:
DModule.RefreshLookupEdit(Self, TControl(Sender).Name,'CUR1.ABBERVATION || "->" || CUR2.ABBERVATION || ": " || EXRAT.EXCHANGE_VALUE || " (" ||TO_CHAR(EXRAT.EXCHANGE_DATE,"YYYY-MM-DD") || ")"','KON.EXCHANGE_RATES EXRAT, KON.CURRENCIES CUR1, KON.CURRENCIES CUR2','EXRAT.CUR_FROM_ID = CUR1.ID AND EXRAT.CUR_TO_ID = CUR2.ID AND EXRAT.ID='), co odpowiada zapytaniu:

```
'SELECT CUR1.ABBERVATION || "->" || CUR2.ABBERVATION || ": " || EXRAT.EXCHANGE_VALUE
|| " (" ||TO_CHAR(EXRAT.EXCHANGE_DATE,"YYYY-MM-DD") || ")" '+
 'FROM   KON.EXCHANGE_RATES EXRAT, '+
 '      KON.CURRENCIES CUR1, '+
 '      KON.CURRENCIES CUR2 '+
 'WHERE  EXRAT.CUR_FROM_ID = CUR1.ID AND '+
 '      EXRAT.CUR_TO_ID   = CUR2.ID AND EXRAT.ID='+EXRAT_ID);
```

Implementacja:

```
Procedure TDModule.RefreshLookupEdit(S : TForm; DBEditName, DisplayField,
TableName, WhereClause : String);
Var DBEdit     : TDBEdit;
    Edit       : TEdit;
    Edit_Value : TEdit;
Begin

  Edit_Value   := TEdit(S.FindComponent(DBEditName+'_VALUE'));

  If S.FindComponent(DBEditName) is TDBEdit Then Begin
    DBEdit := TDBEdit(S.FindComponent(DBEditName));
    If WhereClause = '' Then WhereClause := 'ID='+DBEdit.TEXT
                        Else WhereClause := WhereClause;
    If Trim(DBEdit.TEXT) <> '' Then Edit_Value.Text := SingleValue('SELECT
'+DisplayField+' FROM '+TableName+' WHERE '+WhereClause)
                               Else Edit_Value.Text := '-';
  End Else Begin
   Edit := TEdit(S.FindComponent(DBEditName));
   If WhereClause = '' Then WhereClause := 'ID='+Edit.TEXT
                       Else WhereClause := WhereClause+Edit.TEXT;

   If Trim(Edit.TEXT) <> '' Then Edit_Value.Text := SingleValue('SELECT
'+DisplayField+' FROM '+TableName+' WHERE ID='+Edit.TEXT)
                            Else Edit_Value.Text := '-';
  End;
End;
```