Framework

IMPET

Dokumentacja techniczna

Software Factory

Maciej Szymczak,

Data utworzenia: 12.11.2000

Data ostatniej aktualizacji: 20225.10.24

soft@home.pl

tel. 604.22.46.58

Spis treści

Wstęp 3

Przykładowy problem 3

Rozwiązanie - Designer/ Forms Developer 5

Rozwiązanie – Lotus Notes 6

Rozwiązanie – Delphi 7

Właściwości wzorca formularza 8

Warunki korzystania z framework 9

Wskazówki dotyczące tworzenia modelu danych 9

Nazewnictwo obiektów bazy danych 9

Budowanie fizycznego modelu danych 10

Opis powiązań między formularzami framework IMPET 11

Opis poszczególnych modułów framework 12

DM - Moduł danych 12

Fmain 13

UformConfig 13

Udostępniane elementy 13

Uwagi 13

FbrowseParent 14

FmoduleChart, FModuleConfigure, FmoduleFilter, FmoduleOleExport, FmodulePrint, FmodulePrintAddress, FmodulePrintAdvanced, FmoduleSummary, FmoduleSummaryConfig, FmoduleSummaryDlgOptions, FmoduleSummaryFields, FModuleSummarySQL, FdataBaseError 15

Niezależne użycie FmoduleFilter 15

FinfoParent 15

FlookupWindow 15

Udostępniane elementy 16

Uwagi: 16

Przykłady: 16

Fmultiselect 17

FreadDate 17

FreadString 17

FdatabaseLogin 17

FDatabaseLoginWithCard 17

Tworzenie nowego formularza 17

Najczęściej występujące zadania przy tworzeniu modułu 20

Siatka z rekordami podrzędnymi 20

Wyświetlanie rekordów podrzędnych dla wybranego nadrzędnego (filtr) 21

Naliczanie wielkości w rekordzie nadrzędnym zależnej od rekordów podrzędnych 22

Implementacja uprawnień hierarchicznych 23

Ustawienie formatu wyświetlania pól numerycznych 24

Uwagi do dbf 24

Odswiezanie rekordow podrzednych (siatka z rekordami odświeżana po sekundzie bezczynności) 24

Positions - spec\_authors - authors, roles 25

Nie używać lokupcombo dla tabel > 5000 rekordów, zamiast tego 27

# 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](http://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.

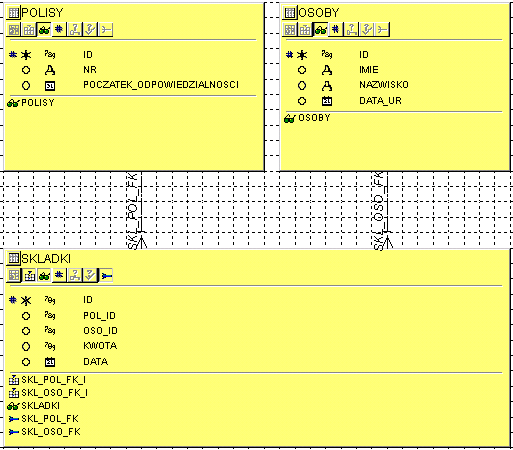
## Przykładowy problem

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,

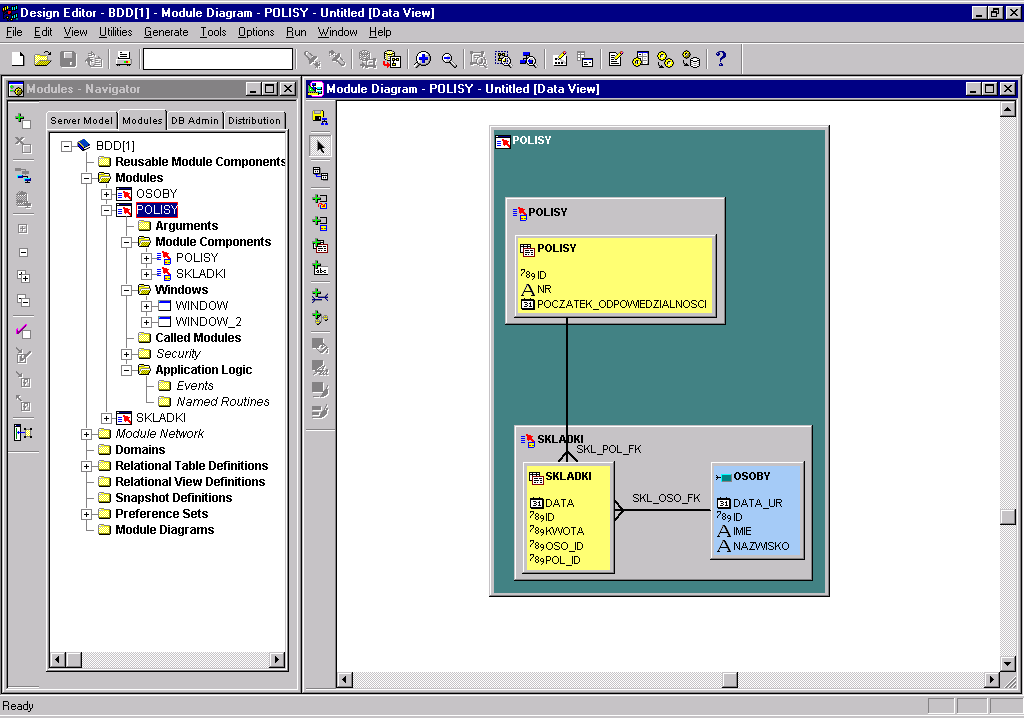
użytkownicy mający dostęp do wszystkich danych, ale możliwością dodawania wyłącznie danych o polisach,

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:

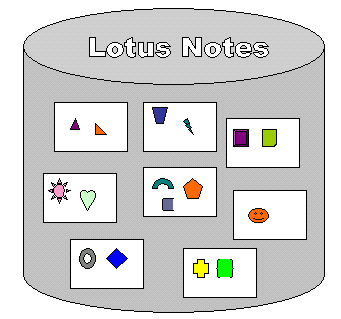
* zestawu preferencji użytych podczas generowania,
* użytej biblioteki obiektów,
* zastosowanego szablonu formularza.

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

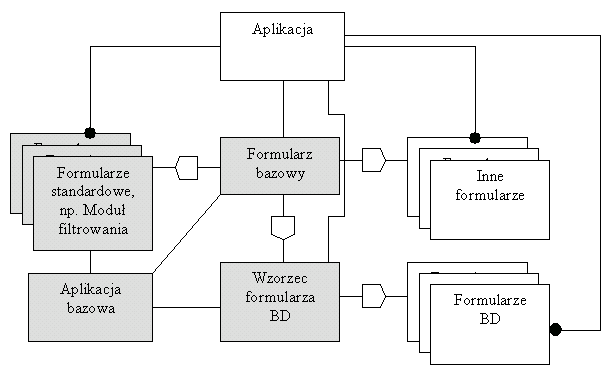
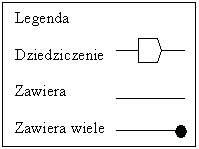
Przystępując do pisania aplikacji w Delphi, pierwszym intuicyjnym rozwiązaniem wydaje się utworzenie aplikacji składającej się z wielu niezależnych formularzy. Dopiero po pewnym czasie okazuje się, że te same problemy rozwiązywane są przez wielu deweloperów wielokrotnie, nierzadko w inny sposób, pojawiają 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.

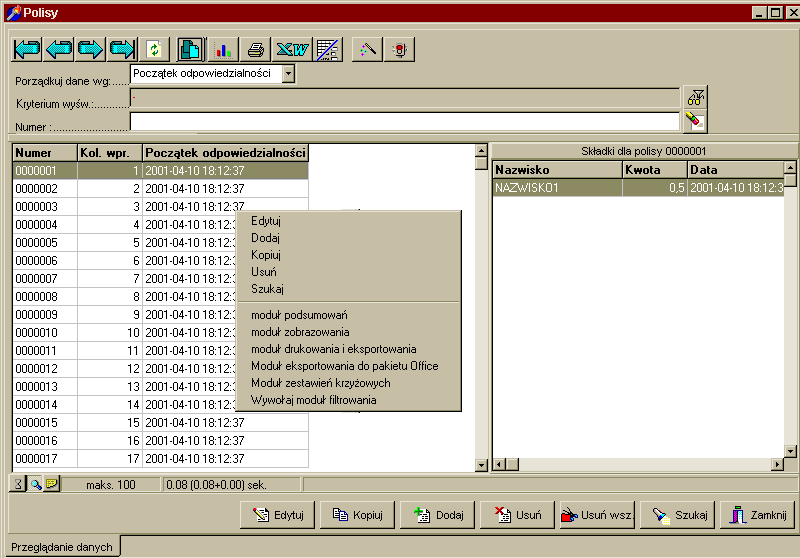
Dzięki otwartości środowiska Delphi wzorzec bazowy, wzorzec formularza bazy danych i pozostałe formularze standardowe, np. moduł filtrowania, moduł drukowania, moduł podsumowań itd. mogą być współdzielone przez wiele aplikacji. Dzięki temu ulepszenie formularza zmienia funkcjonalność nie jednej lecz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



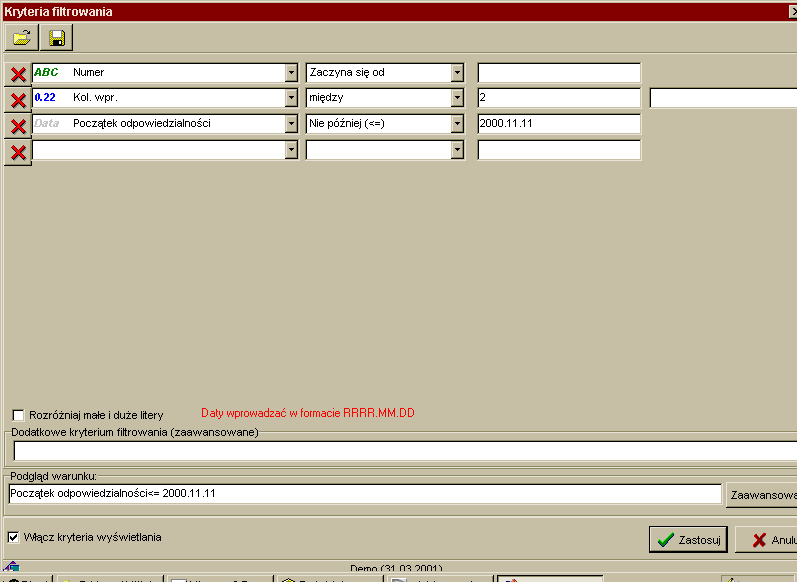
### Właściwości wzorca formularza

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

DM (moduł danych)

**FMain**

FormConfig

FbrowseParent

**FModuleChart**

**FModuleConfigure**

**FModuleFilter**

**FModuleOleExport**

**FModulePrint**

**FModulePrintAddress**

**FModulePrintAdvanced**

**FModuleSummary**

**FModuleSummaryConfig**

**FModuleSummaryDlgOptions**

**FModuleSummaryFields**

**FModuleSummarySQL**

**FdataBaseError**

**Fbrowse...**

**Fbrowse...**

**Fbrowse...**

**...**

FinfoParent

**Finfo**

**FLookupWindow**

**Fmultiselect**

**FReadDate**

**FreadString**

**FDatabaseLogin**

**FDatabaseLoginWithCard**

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|ABBREVIATION|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

1. Wpisać Form.Caption, Form.Name, Unit.Name

Form.Name = FBrowse+ <nazwa tabeli>,

Unit.Name = UFBrowse + <nazwa tabeli>.

1. 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

1. 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

1. opcjonalnie [Uzupełnić Holdery: Kolumny, SortOrder]
2. 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]

1. Poprawić TabOrder i rozmiary okna
2. [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

1. Query.Active := False
2. 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;

1. 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.

1. Dopisać: DefaultValues:

Query.FieldByName('ID').AsString := DModule.SingleValue('SELECT AKW\_ADMIN\_UPRAWNIENIA\_SEQ.NEXTVAL FROM DUAL');

1. Umieścić moduł w AvailableForms

Project/ Options

1. 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;

1. 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;

1. 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)

1. Dmodule.database.connected = false
2. 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>,’’);

1. 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;