FORMULARZE Z POJEDYNCZYM ŹRÓDŁEM DANYCH

Zajmiemy się teraz przygotowaniem formularzy do wyświetlenia listy faktur, kontrahentów i towarów/usług. Chociaż tak naprawdę utworzymy tylko jeden formularz. Dlaczego? Jeśli uważniej przeanalizujemy zadania naszej aplikacji, szybko zauważymy, że formularze wyświetlające wspomniane listy są bardzo podobne. Różnica sprowadza się jedynie do danych wyświetlanych na poszczególnych listach oraz do formularzy, które będą wywoływane, gdy użytkownik zażąda edycji pojedynczego rekordu. Oczywiście nic nie stoi na przeszkodzie, aby opracować oddzielne formularze jeden wyświetlający listę kontrahentów, drugi listę towarów a trzeci listę faktur. Pytanie jednak po co?. Rozwiązanie które zastosujemy daje dwie istotne korzyści:

* W naturalny sposób uzyskujemy jednorodność interfejsu aplikacji

* Zmniejszają się rozmiary kodu źródłowego

Tworzenie formularza za pomocą kreatora

Do utworzenia formularza z listą rekordów użyjemy bardzo wygodnego narzędzia, jakim jest kreator formularzy - Form Wizard. Generuje on formularz ze wszystkimi komponentami niezbędnymi do przeglądania i edycji rekordów pojedynczej tabeli lub zapytania. Następnie dokonamy niezbędnych modyfikacji , aby formularz ten spełniał wymogi naszej aplikacji .Zanim jednak przystąpimy do realizacji konkretnych zadań, wyjaśnijmy jak Delphi uzyskuje dostęp do danych zgromadzonych w bazie. Jak pamiętamy, korzystamy z tabel w standardzie Paradox, obsługiwanych przez aparat bazy danych BDE (Borland Database Engine). Dostęp do BDE z poziomu Delphi uzyskujemy przez komponenty wywodzące się z klasy TDataSet, są to TTable lub TQuery. W przypadku operacji na pojedynczych tabelach z reguły stosuje się pierwszy z tych komponentów - Ttable. Do jego właściwości należą m.in. : DatabaseName, określająca nazwę bazy danych (w przypadku tabel Paradox może to być ścieżka do katalogu lub alias) , a także TableName, przechowująca nazwę tabeli obsługiwanej przez ten komponent. Pomiędzy obiektami TTable a kontrolkami wyświetlanymi na formularzu istnieje jeszcze jeden element pośredniczący - komponent TDataSource. Służy od do odseparowania źródła danych od kontrolek danych, co znacznie zwiększa elastyczność aplikacji. Za pomocą kreatora utwórzmy teraz formularz, w którym będzie wyświetlana lista kontrahentów:

1.Uruchamiamy środowisko Delphi

2.W menu File wybieramy polecenie Open, odszukujemy katalog z naszymi plikami, otwieramy plik projektu Faktury. Na ekranie pojawi się formularz startowy projektu MenuGlForm

3. W menu Database wybieramy polecenie Form Wizard. Pojawi się pierwsze okno dialogowe kreatora.

4.Upewnij się ,że zaznaczone są opcje Create a simple form (tworzenie formularza z jednym źródłem danych) i Create form using TTable objects (tworzenie formularza opartego na tabeli), a następnie kliknij Next

5.W polu Directories otwórz folder Dane

6.Na liście Table name zaznacz plik tabeli Kontrah, a następnie kliknij Next

7.Za pomocą przycisku oznaczonego symbolem >> przenieś wszystkie pola tej tabeli na listę Ordered Selected Fields

8.Kliknij Next

9.Zaznacz opcję In a grid i kliknij przycisk Next .Wybór tej opcji oznacza ,że dane będą prezentowane w kontrolce typu TDBGrid, a więc w postaci siatki, w której każdemu wierszowi odpowiada jeden rekord tabeli, a każdej kolumnie - pojedyncze pole rekordu

10.Upewnij się ,że opcja Generate a main form jest nieaktywna, zaznacz pcję Form and data module, a następnie kliknij Finish

Zostanie wygenerowany formularz zawierający obiekty typu TDBGrid oraz TDBNavigator. Obiekt TDBNavigator służy do wykonywania podstawowych operacji na zbiorze danych, takich jak przewijanie (cztery pierwsze przyciski), wstawianie, usuwanie, rozpoczynanie edycji, zatwierdzanie, anulowanie zmian i odświeżanie rekordów (kolejnych pięć przycisków) W razie potrzeby można ukryć niektóre przyciski, ustawiając odpowiednie wartości właściwości VisibleButtons obiektu TDBNavigator na False. Oprócz formularza na ekranie pojawi się moduł danych. Zaznaczenie pola wyboru Generate a main form w ostatnim oknie kreatora spowodowałoby utworzenie formularza głównego. Z kolei wybór opcji Form only pozwala na utworzenie pojedynczego modułu, w którym jest umieszczany zarówno interfejs formularza, jak i obiekty dostępu do danych. Choć takie rozwiązanie z technicznego punktu widzenia jest zupełnie prawidłowe, zaleca się tworzenie oddzielnego modułu danych (data module) co pozwala na odseparowanie interfejsu formularza od przetwarzanych danych. Takie podejście ma dwie zalety. Po pierwsze poprawia skalowalność aplikacji, a po drugie - pozwala nam stworzyć uniwersalny formularz, który po umieszczeniu w repozytorium może być wykorzystywany w innych aplikacjach. Wybór opcji Form and data module spowodował powstanie dodatkowego modułu danych (o domyślnej nazwie DataModule1), w którym znalazły się komponenty Table1 i DataSource1. W tej chwili w utworzonym formularzu nie widzimy żadnych rekordów, ponieważ właściwość Active komponentu Table1 ma wartość False. Jeśli w fazie projektowej chcesz wyświetlić rekordy tabeli, z którą są powiązane kontrolki danych, ustaw tę właściwość na True. W trakcie działania aplikacji , tabela będzie automatycznie otwierana w chwili tworzenia modułu danych - zadbał o to kreator, wstawiając odpowiednią procedurę w kodzie modułu danych :

Procedurę TDataModule1.DataModuleCreate(Sender : TObject)

begin

Table1.Open;

end;

Wywołanie metody Open obiektu TTable jest równoznaczne z przypisaniem właściwości Active wartości True.

11.Zaznacz komponent Table1 w module danych DataModule1, przejdź do Object Inspectora i zmodyfikuj właściwość DatabaseName na BAZYFAKT. Niestety kreator nie radzi sobie ze ścieżkami względnymi (np. "\dane") dlatego w trakcie jego działania jako źródło podaliśmy bezwzględną lokalizację pliku. Oczywiście aplikacja będzie bardziej elastyczna, jeżeli w tym miejscu zastosujemy alias.

12.Przejdź do właściwości Name nowego formularza i wpisz ListaForm

13.Przejdź do właściwości Name nowego modułu danych i wpisz KontrahDModule

14.Kliknij przycisk Save All. Zapisz nowy formularz jako plik Lista.pas (zapis formularza będzie proponowany jako pierwszy), natomiast modułu danych jako plik KontrahD.pas

>

Wstawianie formularza do repozytorium

Utworzony przed chwilą formularz chcemy wykorzystać również do wyświetlania list towarów oraz faktur. Najprostszym rozwiązaniem wydaje się skopiowanie odpowiednich plików *.dfm i *.pas a następnie zmodyfikowanie właściwości TableName w obiektach TTable powielonych modułów danych. W ten sposób uzyskaliśmy trzy całkowicie niezależne formularze. Jednak droga ta może prowadzić na manowce. Rozważmy najprostszy przykład. Po pewnym czasie doszliśmy do wniosku ,że pasek nawigatora powinien się znajdować nieco bardziej na prawo. Jeżeli zastosujemy powyższe rozwiązanie będzie to oznaczało konieczność dokonania zmian we wszystkich trzech formularzach. Co więcej, nie będziemy mieli pewności, że w każdym z tych formularzy dokonamy takiego samego przesunięcia. Pół biedy, gdy takie modyfikacje dotyczą tylko wyglądu formularzy - w ostateczności uzyskamy niejednorodny interfejs, który co najwyżej będzie irytował użytkownika. Może się jednak zdarzyć ,że zajdzie potrzeba wprowadzenia jakichś zmian do uniwersalnego kodu tego formularza, a wtedy już tylko krok do popełnienia błędu lub przeoczenia. Problemów tych możemy uniknąć w prosty sposób, wystarczy zapisać formularz jako szablon. W praktyce oznacza to zapamiętanie go w repozytorium obiektów. W repozytorium umieszczamy obiekty wielokrotnie wykorzystywane dla różnych aplikacji. Na przykład możemy dojść do wniosku ,że jakiś formularz jest na tyle doskonały ,że warto umieścić go w repozytorium. Obiekty wstawione do repozytorium powinny mieć charakter uniwersalny. Aby za taki można było uznać nasz formularz listy, musimy najpierw usunąć odwołania kontrolek danych do źródła danych w module KontrahDModule (każdy z formularzy potomnych będzie korzystał z innego modułu danych)

1.Zaznacz kontrolki DBNavigator i DBGrid , przejdź do Object Inspectora, a następnie usuń zawartość właściwości DataSource

2.Przejdź do kodu formularza ListaForm i usuń deklarację uses Unit1 z sekcji implementation. Po tych operacjach formularz przestanie być powiązany z konkretnym źródłem danych

3.Zapisz wprowadzone zmiany<br

4.Prawym przyciskiem myszki kliknij w dowolnym miejscu formularza ListaForm.

5.W menu podręcznym wybierz polecenie Add To Repository. Na ekranie pojawi się okno dialogowe o takiej samej nazwie.

6.Wypełnij go : Title - Lista; Description - Lista rekordów ; Page - Forms Author - ????. Repozytorium jest standardowo podzielony na kilka stron (można tworzyć własne), na których są pogrupowane obiekty różnych kategorii. W naszym przykładzie zastosowaliśmy stronę Forms

7.Kliknij przycisk OK. Formularz zostanie umieszczony w repozytorium obiektów. Oznacza to, że będzie można go wielokrotnie wykorzystywać w tej i innych aplikacjach. Sprawdźmy teraz, czy operacja umieszczenia w repozytorium powiodła się .

8.W menu Tools wybierz polecenie Repository

9.Na liście Pages zaznacz pozycję Forms. Na liście Objects pojawią się wszystkie elementy zapamiętane na stronie Forms repozytorium, a wśród nich szablon formularza ListaForm. Okno dialogowe umożliwia reorganizowanie repozytorium (tworzenie i usuwanie stron, przemieszczanie obiektów między stronami itp.)

10.Kliknij Save All aby zapisać wprowadzone zmiany. Jak sama nazwa wskazuje formularz ListaForm jest teraz szablonem - nie będziemy więc go bezpośrednio wykorzystywać w aplikacji, lecz użyjemy do

stworzenia trzech formularzy pochodnych, w których znajdą się oddzielne listy kontrahentów, towarów i faktur.

Pobieranie formularza z repozytorium

Istnieją trzy sposoby korzystania z formularzy umieszczonych w repozytorium:

* Kopiowanie formularzy

* Dziedziczenie po formularzach

* Operowanie bezpośrednio na formularzach

Pierwsza z wymienionych metod nie różni się niczym od skopiowania plików DFM i PAS formularza, jest tylko nieco wygodniejsza, gdy wielokrotnie powtarzamy ten proces. Otrzymany w ten sposób formularz jest całkowicie niezależną kopią swojego pierwowzoru. Druga metoda jest o wiele bardziej elastyczna, gdyż prowadzi do powstania formularza dziedziczącego po formularzu-rodzicu. Co to oznacza w praktyce? Wszelkie zmiany wprowadzane w klasie macierzystej będą automatycznie odzwierciedlone w klasach potomnych. Jeżeli na przykład w formularzu - rodzicu przesuniemy pasek nawigatora, o taką sama wartość przesunie się on w formularzach potomnych. Jeżeli we wzorcowym formularzu wstawimy nowy przycisk, pojawi się on natychmiast w jego kopiach. Czyż nie o to nam chodziło? Dzięki dziedziczeniu w prosty sposób uzyskamy jednorodny interfejs. Trzecia metoda polega ba wprowadzaniu zmian bezpośrednio w formularzu umieszczonym w repozytorium. Zmiany będą odzwierciedlone we wszystkich projektach wykorzystujących ten formularz .Zanim wstawimy nowy formularz listy, musimy nieco "oszukać" środowisko Delphi. Niestety nie radzi ono sobie dobrze w sytuacji, gdy w projekcie umieszczamy kilka formularzy dziedziczących po jednym formularzu z repozytorium. Problem polega na tym ,że wstawienie formularza w trybie dziedziczenia oznacza również konieczność wstawienia formularza nadrzędnego - w naszym przypadku jest to ListaForm. Gdy spróbujemy wstawić formularz dziedziczący, pojawi się komunikat "The project already contains a form or module named Lista" (projekt zawiera już formularz lub moduł o nazwie Lista). Oto jeden ze sposobów poradzenia sobie z tym problemem.

1.Kliknij przycisk View Unit, a następnie na liście wybierz plik projektu Faktury. Na ekranie pojawi się główny plik projektu

2.W edytorze usuń wiersz w klauzuli uses związany z modułem Lista.pas. Usuń również wywołanie Applcation.CreateForm(TListaForm,ListaForm), aby moduł projektu Faktury.dpr uzyskał postać :

```
program Faktury;

uses

Forms,

MenuGl in 'MenuGl.pas' {MenuGlForm},

Konfig in 'Konfig.pas' KonfigForm),

KontrahD in 'KontrahD.pas' {KontrahDModule : TDataModule};

{$R *.res}

begin

Application.Initialize;
```

Application.CreateForm(TMenuGlForm, MenuGlForm);

Application.CreateForm(TKonfigForm, KonfigForm);

Application.CreateForm(TKontrahDModule, KontrahDModule);

Application.Run;

end.

Zwróć uwagę aby klauzula uses kończyła się średnikiem! Spróbujmy teraz utworzyć formularz listy oparty na formularzu z repozytorium

3.W menu File, New wybierz polecenie Other. Pojawi się okno dialogowe NewItems

4.Przejdź do karty Forms i zaznacz ikonę Lista. Zawartość tego okna w rzeczywistości jest zawartością repozytorium, widzimy więc tutaj ikonę symbolizującą formularz Lista

5.Zaznacz opcję Inherit, a następnie kliknij przycisk OK. Wybór tej opcji spowoduje utworzenie nowego formularza dziedziczącego po klasie TListaForm. Możemy się o tym przekonać , sprawdzając moduł formularza.

6.Przejdź do widoku kodu modułu formularza i odszukaj definicję jego klasy. Powinna ona wyglądać następująco

type

TListaForm1 = class(TListaForm)

private

{Private declarations}

public

{Public declarations}

end;

Gdybyśmy pozostawili opcję Copy, powstałaby klasa formularza dziedzicząca bezpośrednio po klasie TForm i zbudowana identycznie jak klasa TListaForm, zapamiętana w repozytorium:

type

TListaForm1 = class(TListaForm)

DBGrid1 : TDBGrid;

DBNavigator : TDBNavigator;

Panel1 : TPanel;

Panel2: TPanel;

private

{Private declarations}

public

{Public declarations}

end;

Przypisanie źródła danych formularzowi pochodnemu

Kolejnym zadaniem do wykonania jest ponowne powiązanie formularza z modułem danych. Jak pamiętasz ,ze względu na uniwersalny charakter formularza ListaForm, usunęliśmy powiązania jego komponentów bazodanowych ze źródłem danych. Oznacza to ,że w formularzach pochodnych musimy jeszcze raz zdefiniować te wartości.

1.Naciśnij klawisz F12, aby przenieść się do widoku kodu formularza i w sekcji implementation (za dyrektywą {$R *.DFM}) umieść deklarację :

uses KontrahD;

2.W Object Inspectorze odszukaj obiekt formularza KontrahLForme, przejdź do karty Events, a następnie dwukrotnie kliknij w polu procedury zdarzenia OnCreate dla tego formularza

3.Uzupełnij powstałą procedurę obsługi zgodnie z poniższym:

procedure TKontrahLForm.FormCreate(Sender : TObject);

begin

inherited;

DBNavigator.DataSource : = KontrahDModule.DataSource1;

DBGrid1.DataSource := KontrahDModule.DataSource1;

end;

Wartości te moglibyśmy przypisać w fazie projektwej , korzystając z Obiekt Inspectora. Niestety pojawiają się tu pewne niekorzystne konsekwencje zastosowania dziedziczenia formularzy. Jeżeli w jakiś sposób zmieniasz formularz macierzysty (ListForm), jego właściwości zostaną odwzorowane w formularzach pochodnych. Ponieważ w formularzu macierzystym usunęliśmy wartości DataSource z kontrolki nawigatora i siatki, może to spowodować to usunięcie tych właściwości w kontrolkach formularzy pochodnych. Ponadto opcja KontrahDModule.DataSource1 w polu właściwości DataSource komponentów formularza będzie niedostępna, jeżeli uprzednio nie otworzysz pliku, w którym znajduje się wskazywane źródło danych. W powyższym przykładzie, jeśli plik KontrahD.pas nie jest otwarty (nie widać jego zakładki w edytorze kodu), kliknij przycisk View Unit i wybierz moduł KontrahD. Aby uniknąć tych niedogodności , przypisywanie źródła danych będzie się odbywało podczas każdego tworzenia listy, w procedurze obsługi zdarzenia OnCreate. Pozostało nam jeszcze do wykonania kilka prostych czynności porządkowych, takich jak nazwanie nowego formularza, nadaniu mu etykiety i zapisanie zmian na dysku.

4.Przejdź do widoku View Form nowego formularza

5.W Object Inspectorze odszukaj właściwość Name z zmień ja na KontrahLForm

6.Odszukaj właściwość Caption i zmień ją na Lista kontrahentów

7.Kliknij przycisk Save i zapisz moduł formularza jako plik KontrahL

Utworzenie dodatkowych źródeł danych

Zanim w analogiczny sposób utworzysz formularze wyświetlające listy towarów/usług i faktur, musimy najpierw zdefiniować odpowiednie moduły danych. Tym razem nie skorzystamy z pomocy kreatora

1.W menu File, New, wybierz polecenie Data mdule

2.Do nowego modułu wstaw komponent TTable (z palety BDE) oraz TDateSource (z palety Data Access)

3.W Object Inspectorze zmień właściwości obiektu Table1:

* DatabaseName na BAZAFAKT

* TableName na TowUslug

4.Dwukrotnie kliknij ikonę komponentu Table1. Na ekranie pojawi się okno z listą pól. W tej chwili powinno być ono puste.

5.Prawym przyciskiem myszy kliknij w pustym oknie i z menu podręcznego wybierz polecenie Add all fields. Na liście pojawią się wszystkie pola tabeli TowUslug

6.Zamknij okno z listą pól

7.W Object Inspectorze wybierz właściwość Active obiektu Table1 i ustaw ją na True. Ustawienie tej właściwości na True spowoduje ,że tabela będzie dostępna już w fazie prjektowej. Równoważna dla tej właściwości jest metoda Open, obiektu TTable; z reguły wywołuje się ją w procedurze zdarzenia Create formularza. W kodzie programu można by więc zapisać:<br.>

Table1.Active := true; lub po prostu Table1.Open

8.Zaznacz obiekt DataSource1 i ustaw jego właściwość DataSet na Table1

9.Zaznacz okno modułu danych i zmień jego nazwę (właściwość Name) na TowarDModule

10.Kliknij przycisk Save i zapisz utworzony moduł jako plik TowarD.pas

11.Zamknij okno modułu danych

Powtórz czynności z punktów 1- 11 ,aby zdefiniować moduł danych dla tabeli z nagłówkami faktur. W punkcie 3 jako nazwę tabeli (właściwość TableName) podaj TranNagl. Zmień nazwę modułu na FaktDModule i zapisz go jako plik FaktD.pas

Tworzenie pozostałych formularzy

Czas na nieco samodzielnej pracy. W analogiczny sposób jak w poprzedniej części musisz wstawić do aplikacji kolejne dwa formularze - do wyświetlenia list towarów oraz faktur. Ponieważ należy wykonywać identyczne czynności jak powyżej, nie będziemy powielali ich opisu. Zmienią się jedynie nazwy utworzonych obiektów i niektóre ich właściwości. Nie zapomnij od opcji Inherit podczas wstawiania formularza dziedziczącego. Pamiętaj też aby przed każdą operacją wstawienia takiego formularza usuną wiersz Lista in 'LISTA.pas' z klauzuli uses modułu projektu. Po utworzeniu formularza dla listy towarów zmień jego właściwości:

* Name na TowarLForm

* Caption na Lista towarów i usług

W kodzie modułu dołącz klauzulę uses TowarD Utwórz procedurę zdarzenia OnCreate dla formularza TowarLForm i na jej końcu wstaw polecenia:

DBNavigator.DataSource := TowarDModule.DataSource1;

DBGrid1.DataSource := TowarDModule.DataSource1;

Zapisz formularz jako plik TowarL. Po utworzeniu formularza dla listy faktur zmień jego właściwości:

* Name na FaktLForm

* Caption na Lista faktur

W kodzie modułu dołącz klauzulę uses FaktD. Utwórz procedurę zdarzenia OnCreate dla formularza TowarLForm i na jej końcu wstaw polecenia:

DBNavigator.DataSource := FaktDModule.DataSource1;

DBGrid.DataSource := FaktDModule.DataSource1;

Zapisz formularz jako plik FaktL. Nasza aplikacja wzbogaciła się o kolejne trzy moduły danych, Jak widać, jej złożoność wzrosła w stosunku do stanu z poprzedniej części. Jednak dzięki zachowaniu spójnej konwencji nazewniczej, struktura modułów i formularzy jest przejrzysta. "Sztuczka" z usuwaniem modułu Lista może spowodować nieco zamieszania w klauzuli uses modułu projektu. Sprawdź czy ma on prawidłowa budowę:

1.W oknie edytora kodu przejdź do zakładki Faktury. Tekst pliku projektu Faktury.dpr powinien mieć postać:

program Faktury;

uses

Forms, MenuGl in 'MenuGl.pas' {MenuGlForm},

Konfig in 'Konfig.pas' {KonfigForm},

Lista in 'LISTA.pas' {ListaForm},

KontrahD in 'KontrahD.pas' {KontrahDModule : TDataModule},

KontrahL   in 'KontrahL.pas' {KontrahLForm},

TowarL in 'TowarL.pas' {TowarLForm},

TowarD   in 'TowarD.pas' {TowarDModule : TDataModule},

FaktD in 'FaktD.pas' {FaktDModule : TDataModule},

FaktL in 'FaktL.pas' {FaktLForm},

{$R *.res}

begin

Application.Initialize;

Application.CreateForm(TMenuGlForm, MenuGlForm);

Application.CreateForm(TKonfigForm, KonfigForm);

Application.CreateForm(TKontrahDModule, KontrahDModule);

Application.CreateForm(TKontrahLForm, KOntrahLForm);

Application.CreateForm(TTowarDModule, TowarDModule);

Application.CreateForm(TTowarLForm, TowarLForm);

Application.CreateForm(TFaktDmodule, FaktDModule);

Application.CreateForm(TFaktLForm, FaktLForm);

Application.Run;

end.

Kolejność modułów może być dowolna z dwoma wyjątkami:

* Pierwsza metoda CreateForm powinna tworzyć formularz startowy aplikacji (w naszym przykładzie MenuGlForm)

* Tworzenie modułów danych powinno poprzedzać tworzenie formularzy, z którymi te moduły są związane. W przeciwnym razie podczas tworzenia formularza może dojść do wywołania metody Open obiektu TTable, gdy nie został on jeszcze utworzony

2.Wprowadź ewentualne zmiany i kliknij przycisk Save All

3.Skompiluj zmodyfikowany kod, aby sprawdzić czy nie zawiera błędów

Powiązanie nowych formularzy z formularzem głównym

Utworzone przed chwilą formularze powinny być wywoływane z poziomu formularza głównego MenuGlForm poprzez wybranie polecenia menu lub kliknięcie przycisku na pasku narzędzi. Musimy więc zdefiniować odpowiednie procedury w formularzu głównym aplikacji. Będziemy postępować tak samo jak w przypadku formularza KonfigForm. Wspólnie zdefiniujemy odpowiednie mechanizmy dla formularza KontrahForm. Powiązanie pozostałych formularzy z formularzem głównym pozostawimy do samodzielnej realizacji

1.Na pasku narzędzi kliknij przycisk VewForm i wybierz formularz MenuGlForm.

2.Dwukrotnie kliknij drugi przycisk paska narzędzi (przedstawiający segregatory), uzupełnij procedurę obsługi zdarzenia, aby wyglądała następująco:

procedure TMenuGlForm.ToolButton3Click(Sender : TObject);

begin

WyswietlKontrah;

end;

3.Powróć do formularza, klikając przycisk Toggle Form/Unit , w menu Słwniki wybierz polecenie Kontrahenci, a następnie uzupełnij procedurę obsługi zdarzenia aby wyglądała następująco:

procedure TMenuGlForm.Kontrahenci1Click(Sender : TObject);

begin

WyswietlKontrah;

end;

4.Poniżej utworzonej przed chwilą procedury wpisz:

procedure TMenuGlForm.WyswietlKontrah;

begin

KontrahLForm.ShowModal;

end;

5.Przejdź do definicji klasy TMenuGlForm i wstaw w niej deklarację procedury (przed słowem kluczowym private):

procdure WyswietlKontrah;

6.W klauzuli uses w sekcji implementation dołącz moduł KontrahL. Klauzula ta powinna mieć teraz postać :

uses Konfig, KontrahL;

7.Powtórz czynności z punktów 1-6 dla formularzy TowarLForm i FaktLForm. Tworząc procedury w punkcie 4 nazwij je odpowiednio WyswietlTowar i WyswietlFakt

8.Kliknij przycisk Save, aby zapisać wprowadzone zmiany

9.Skompiluj i uruchom aplikację

Sprawdź czy wszystkie formularze działają prawidłowo. Do ich wyświetlenia użyj opcji menu oraz przycisków paska narzędzi. Spróbuj wpisywać przykładowe dane w listach towarów i kontrahentów. Obserwuj jak zmienia się stan przycisków nawigatora.

Modyfikacja właściwości pól

Jeżeli zgodnie z zaleceniami poświęciłeś nieco czasu na zapoznanie się z działaniem nowego formularza, z pewnością dostrzegłeś kilka elementów, które warto by zmienić. Przede wszystkim nagłówki poszczególnych kolumn siatki, choć zgadzają się z nazwami pól tabel, z pewnością nie wyglądają zbyt ładnie w formie skrótów, w dodatku pozbawionych polskich znaków diakrytycznych. Ponadto niektóre kolumny mogłyby być nieco węższe, co pozwoliłoby wyświetlić więcej informacji na ekranie, a inne - całkowicie ukryte przed użytkownikiem, na przykład kolumna IDTranNagl z tabeli TranNagl (wartości tej kolumny będą wykorzystywane wewnętrznie przez aplikację i nie ma potrzeby kłopotać nimi użytkownika) Odpowiednie zmiany uzyskamy, modyfikując pola obiektu TTable

1.Przejdź do widoku modułu KontrahDmodule

2.Dwukrotnie kliknij obiekt Table1. Na ekranie pojawi się okno Field Editor z listą pól zdefiniowanych w obiekcie Table1. Powinny być to wszystkie pola tabeli Kontrah. Jeżeli na liście brak któregoś pola lub chcesz usunąć jakieś pole z listy, kliknij ją prawym przyciskiem myszki i wybierz odpowiednie polecenie (Add fields lub Delete)

3.Kliknij pole IDKontrah, przejdź do Object Inspector i zmień właściwość DisplayLabel na Kod

4.Kliknij pole nazwa, przejdź do Object Inspectora i zmień właściwość DisplayWidth na 40. Właściwość DisplayWidth określa liczbę znaków wyświetlanych jednocześnie na ekranie. Nie wpływa ona na liczbę znaków przechowywanych w tabeli ani też na liczbę znaków, które można wpisać w kontrolce ekranowej

5.W analogiczny sposób zmodyfikuj szerokość pól Miasto i Ulica

6.Kliknij pole Telefon_kom, a następnie przejdź do Object Inspector i zmień właściwość DisplayLabel na Tel.komórkowy. Ta wartość będzie wyświetlana w nagłówku kolumny w komponencie siatki

7.Zaznacz jednocześnie pole Kontakt i Odbier_fakture i Object Inspectorze ustaw ich właściwość Visible na False. To ustawienie spowoduje, że obydwa pola nie będą wyświetlane w komponencie siatki

8.Kliknij przycisk Save ,aby zapisać wprowadzone zmiany

Odwoływanie się do pól obiektu typu DataSet

Jeżeli pola tabeli lub zapytania zostaną dodane do listy pól, jest to równoznaczne ze zdefiniowaniem na poziomie klasy odpowiednich obiektów TFielld (w rzeczywistości będą obiekty typów przypisanych do poszczególnych typów danych w zestawie. Np. poniżej mamy kilka definicji pól dla tabeli TranNagl:

Table1Seria : TStringField;

Table1Numer : TIntegerField;

Table1Dopisek: TStringField;

Table1Data_wystaw: TDateField;

Jeżeli np. chcemy przypisać polu Seria wartość "FA", wystarczy użyć konstrukcji:

Tabel1Seria.AsString := "FA"

Takie rozwiązanie jest wygodniejsze , jeżeli zależy nam na możliwości edytowania właściwości tych pól w fazie projektowej. Czasami jednak, gdy chcemy zachować większą elastyczność kodu, lub gdy korzystamy tylko z niewielu pól zestawu danych, rezygnujemy z definiowania obiektów przedstawiających te pola. Lista pól jest wtedy pusta. Mimo to pola tabeli lub zapytania są dostępne - zwykle odwołujemy się do nich stosując metodę FiledByName zdefiniowaną w klasie TDataSet. Na przykład

Table1.FieldByName('Seria'),AsString := "FA"

Czynności opisane w punktach 1-8 wykonaj dla pozostałych komponentów TTable umieszczonych w modułach danych TowarDmodule i FaktDModule. Modyfikując właściwości, skorzystaj z poniższych wartości:

Pole : IDTowUslug

Właściwość : DisplayLabel

Nowa wartość : Kod

Pole : J_miary

Właściwość : DisplayLabel

Nowa wartość : J.m

Pole : Stan_min

Właściwość : DisplayLabel

Nowa wartość : Stan min

Pole : Cena_zak

Właściwość : DisplayLabel

Nowa wartość : Cena zakupu

Pole : Cena_sprz

Właściwość : DisplayLabel

Nowa wartość : Cena sprzed.

Pole : St_VAT

Właściwość : DisplayLabel

Nowa wartość : St. VAT

Pole : Data_wystaw

Właściwość : DisplayLabel

Nowa wartość : Data wystaw

Pole : Data_trans

Właściwość : DisplayLabel

Nowa wartość : Data sprzed.

Pole : Term_plat

Właściwość : DisplayLabel

Nowa wartość : Termin płatn.

Pole : Forma_platn

Właściwość : DisplayLabel

Nowa wartość : Forma płatn.

Pole : IDKOntrah

Właściwość : DisplayLabel

Nowa wartość : Forma płatn.

Pole : IDTranNagl

Właściwość : Visible

Nowa wartość : False

Zwróć uwagę ,że w tabeli TranNagl (moduł FaktDModule) ukrywamy pole z identyfikatorem transakcji. Pole to służy jedynie do powiązania nagłówka transakcji z pozycjami transakcji - dla użytkownika nie niesie ono żadnych informacji a więc nie powinno być wyświetlane. Na zakończenie skompiluj i uruchom aplikacje .Sprawdź efekt wprowadzonych modyfikacji

Definiowanie maski edycji

Jak już wspomnieliśmy, chcemy aby identyfikatory kontrahentów i towarów/usług były wyświetlane wielkimi literami. W tym celu podczas opracowywania struktury tabeli zdefiniowaliśmy dla odpowiednich pól właściwość Picture, umożliwiającą formatowanie wprowadzonego tekstu. Niestety mechanizm ten jest dostępny tylko z poziomu programu Database Desktop. Jeżeli chcemy, aby takie same reguły obowiązywały również podczas wypełniania rekordów z poziomu formularzy , musimy je ponownie zdefiniować w aplikacji. Umożliwia to właściwość EditMask

1.Przejdź do widoku modułu KontrahDModule

2.Dwukrotnie kliknij obiekt Table1

3.Zaznacz pole IDKontrah

4.Przejdź do Object Inspectora i we właściwości EditMask wpisz następujący ciąg znaków : >aaaaaaaaaaaa;1;_

Jest to tzw. maska edycji. Składa się ona z trzech części rozdzielonych średnikami. Pierwsza część określa samą maskę. W powyższym przykładzie symbol większości oznacza ,że wszystkie wpisane znaki alfabetu będą przekształcane na wielkie litery. Dwanaście małych liter "a oznacza ,że w polu edycji można wpisać co najwyżej tyle znaków. W ten sposób formułujemy warunek ,że kod kontrahenta może się składać maksymalnie z dwunastu znaków. Wartość 1 za średnikiem oznacza ,że w odpowiednim polu bazy danych będą wpisywane tylko wartości wprowadzane przez użytkownika. Np. jeżeli maska edycji ma postać 00-000;1;_, to po wpisaniu wartości 44-310 zostanie ona zapamiętana w polu bazy danych wraz z elementem stałym (w tym przypadku jest nim łącznik), a więc 44-310. Gdyby maska miała postać 00-000;0;_, wtedy w polu bazy danych znalazłaby się wartość 44310. Ostatni element maski określa znak jaki będzie się pojawiał w jeszcze niewypełnionych pozycjach maski. Jeżeli na przykład wprowadzimy kod kontrahenta ADA, to w polu tekstowym będzie on widoczny jako ADA_____. Nie zawsze to dobrze wygląda, dlatego zamiast znaku podkreślenia można użyć spacji

5.Kliknij przycisk Save, aby zapisać wprowadzone zmiany

6.Skompiluj u uruchom aplikację, Przejdź do formularza z listą towarów/usług. Spróbuj wpisać dowolny kod towaru. Jak widać , tekst jest wpisywany wielkimi literami. Spróbuj samodzielnie zdefiniować taką samą maskę edycji dla kodu towaru (obiekt Table1 w module danych TowarDModule)

Tworzenie formularza do edycji pojedynczego rekordu

Dane mogą być wyświetlane w formie listy lub pojedynczych rekordów. Praca z listą nie zawsze jest wygodna. Z jednej strony dzięki niej widzimy zawsze więcej rekordów, z drugiej jednak, jeżeli zawierają one zbyt wiele pól, nie wszystkie mieszczą się na ekranie, Nie można definitywnie stwierdzić ,że któraś w wymienionych form prezentacji danych jest lepsza, Ich przydatność wynika z potrzeb użytkownika, dlatego dobra aplikacja bazodanowa powinna oferować obydwie możliwości. Opracujemy teraz formularz do edycji pojedynczego rekordu z tabeli Kontrah

1.W menu Database wybierz polecenie FormWizard. Pojawi się pierwsze okno kreatora

2.Upewnij się ,że zaznaczone są opcje Create simple form i Create a form using TTable objects i kliknij przycisk Next<br

3.W kolejnym oknie kreatora na liście Directories przejdź do folderu Dane, na liście Table name zaznacza tabelę Kontrah i kliknij przycisk Next

4.W kolejnym oknie kreatora kliknij przycisk >>, aby przenieść wszystkie pola tabeli na listę Ordered Selected Fields. Kliknij przycisk Next

5.W następnym oknie kreatora upewnij się,że zaznaczona jest opcja Horizontally i kliknij pzycisk Next. Pola na tworzonym formularzu zostaną umieszczone w wierszach jedno obk drugiego Etykiety opisujące przeznaczenie poszczególnych pól zostaną umieszczone ponad nimi. Druga opcja (Verticaly) powoduje umieszczanie pól jedno pod drugim

6.W ostatnim oknie kreatora upewnij się ,że pole wyboru Generate a main form nie jest zaznaczone oraz ,że jest aktywna opcja Form only. Na ekranie pojawi się gotowy formularz d edycji pojedynczego rekordu tabeli Kontrah. Oprócz kontrolek typu TDBEdit widać na nim pasek nawigacyjny oraz komponenty TTable i TDataSource. W tym przypadku zrezygnowaliśmy z tworzenia oddzielnego modułu danych - ten formularz będzie przeznaczony wyłącznie do edycji danych kontrahentów, a więc odseparowanie źródła danych od interfejsu nie przyniosłoby na istotnych korzyści. Oczywiście utworzony w ten sposób formularz warto jeszcze nieco doszlifować , dobierając odpowiednie wielkości czcionek, rozmieszczając komponenty itp.

7.Gdy zaznaczony jest nowo utworzony formularz, przejdź do karty Properties Object Inspectora i zmień właściwość Caption na Dane kontrahenta oraz właściwość Name na KontrahForm.

8.Kliknij przycisk Save i zapisz moduł nowego formularza jako plik Kontrah.pas.

Wykonując czynności opisane w punktach1-8, w analogiczny sposób przygotuj formularz do edycji pojedynczych rekordów z tabeli TowUslug. Po utworzeniu formularza zmień jego nazwę na TowarForm. Zapisz go jak Towar.pas.

Powiązanie formularza do edycji pojedynczego rekordu z formularzem-listą

Formularze do edycji pojedynczych rekordów z tabel kontrahentów i towarów są już gotowe, musimy je tylko ze sobą powiązać. Dobrze jest, gdy aplikacja oferuje kilka dróg osiągnięcia tego samego celu. Na przykład w wielu popularnych edytorach tekstu zmianę czcionki można uzyskać za pomocą menu, paska narzędzi, menu podręcznego lub klawisza skrót. Taka wszechstronność pozwala na dużą elastyczność w użytkowaniu aplikacji i jest ceniona zwłaszcza przez zaawansowanych użytkowników, dążących zwykle do optymalizacji wykonywanych czynności. W naszej aplikacji wprowadzimy dwa sposoby przejścia do edycji pojedynczego rekordu: poprzez kliknięcie odpowiedniego przycisku w formularzu listy oraz poprzez podwójne kliknięcie wybranego wiersza na liście (to drugie rozwiązanie prawdopodobnie będzie preferowane przez zaawansowanych użytkowników aplikacji ponieważ pozwoli im zmniejszyć liczbę wykonywanych czynności). Najpierw jednak musimy umieścić na formularzach odpowiedni przycisk. Przy okazji będziemy mieli możliwość docenienie zalet korzystania z formularzy dziedziczonych z repozytorium - przycisk wstawiamy tylko w formularzu - rodzicu, co spowoduje jego automatyczne pojawienie się we wszystkich formularzach potomnych.

1.Na pasku narzędzi kliknij przycisk View Form

2.Dwukrotnie kliknij pozycję ListaForm. Na ekranie pojawi się formularz macierzysty dla wszystkich formularzy list używanych w aplikacji.

3.Na palecie Additional wybierz komponent TBitBtn i umieść go na formularzu obok paska nawigatora

4.W Object Inspectorze zmień właściwość Name przycisku na EdytujRekord-BitBtn oraz właściwość Caption na &Edytuj rekord

5.Kliknij przycisk Save.

Musimy teraz wprowadzić kod wyświetlający odpowiednie formularze. Ponieważ w zależności od typu listy będziemy wyświetlać różne dane, odpowiedni kod zdefiniujemy już nie w formularzu wzorcowym , lecz w poszczególnych formularzach list

1.Wyświetl na ekranie formularz KontrahLForm

2.Dwukrotnie kliknij przycisk EdytujRekordBitBtn

3.W utworzonej procedurze , poniżej słowa kluczowego Inherit, wpisz poniższy kod:

WyswietlKontrah;

Zwróć uwagę, aby wpisywany kod znalazł się pod słowem kluczowym Inherit. Zostało ono tutaj wstawione automatycznie, aby wykonać ewentualne operacje przypisane analogicznej procedurze zdarzenia na formularzu - rodzicu. Ostatecznie definiowana procedura powinna mieć postać:

procedure TKontrahLForm.EdytujRekrdBitBtnClick(Sender : TObject);

begin

inherited

WyswietlKontraj;

end;

4.Naciśnij klawisz F12, aby powrócić do widoku formularza i zaznacz komponent DBGrid1

5.W Object Inspectorze wybierz zakładkę Events i dwukrotnie kliknij w polu tekstowym obok zdarzenia OnDblClick

6.W utworzonej procedurze wpisz poniższy kod:

WyswietlKontrah;

Dzięki temu podwójne kliknięcie dowolnego wiersza siatki danych będzie powodowało wyświetlenie formularza z pojedynczym rekordem kontrahenta

7.Na końcu modułu formularza wpisz procedurę:

procedure TKontrahLForm.WyswietlKontrah;

begin

KontrahForm.ShowModal

end;

8.Uzupełnij deklarację procedury w sekcji interfejsu modułu (powyżej słowa kluczowego private), wpisując:

procedure WyswietlKontrah;

Wykonaj analogiczne czynności z punktów 1-8 dla formularza TowarLForm. Utwórz w nim procedurę WyswietlTowar, w której będzie wykonywana metoda TowarForm.ShowModal

9.Kliknij przycisk Save All ,aby zapisać zmiany

10.Przetestuj działanie aplikacji

Synchronizacja danych między formularzami

Czy wprowadziłeś przynajmniej kilka rekordów do tabeli Kontrah? Jeżeli nie, zrób to.Gdy będziesz w widoku listy, dwukrotnie kliknij któryś z dalszych rekordów (nie pierwszy). Jak się można było spodziewać, nastąpi przejście do formularza KontrahForm. Jednak czeka nas przykra niespodzianka. Zamiast rekordu który klikaliśmy, zobaczymy pierwszy rekord z tabeli Kontrah. Nic dziwnego - komponent TDataSource na tym formularzu korzysta z innego obiektu TTable niż siatka na formularzu KontarhLForm. Krótko mówiąc, operacje wykonywane w siatce (np. zmiana bieżącego rekordu) nie są synchronizowane z formularzem KontrahForm. Z tym problemem można sobie poradzić na kilka sposobów, mu jednak zdecydujemy się na najprostsze rozwiązanie. Aby uzyskać synchronizację danych między formularzami, wystarczy aby komponenty TDataSource, do których odwołują się kontrolki tych formularzy, korzystały z tych samych komponentów TTable

1.Przejdź do widoku formularza KontrahForm

2.Usuń komponent Table1

3.Zaznacz komponent DataSource1

4.Przejdź do zakładki Properties w oknie Object Inspectora i w polu właściwości DataSet wpisz KontrahDModule.Table1

5.Przejdź do widoku kodu modułu formularza, odszukaj procedurę obsługi zdarzenia FormCreate i usuń w niej wiersz

Table1.Open Procedura ta jest tworzona automatycznie przez kreator formularzy. Ponieważ obiekt Table1 w tym formularzu przestał istnieć, pozostawienie powyższej instrukcji spowodowałoby błąd kompilacji. Czynności z punktów 1-5 wykonaj w formularzu TowarForm, Jako właściwość DataSet wpisz TowarDModule.Table1

6.Kliknij przycisk Save All, aby zapisać wprowadzone zmiany

7.Skompiluj i uruchom aplikację. Przetestuj wprowadzone modyfikacje.

Synchronizacja między formularzami została osiągnięta.

Podsumowanie

W tej części utworzyłeś formularz ListaForm, który posłużył jako formularz macierzysty do zdefiniowania formularzy wyświetlających listy faktur, towarów i kontrahentów. Formularze te powiązałeś z menu głównym aplikacji, dzięki czemu są dostępne po jej uruchomieniu. Przy okazji nauczyłeś się umieszczać formularze w repozytorium i używać go z wykorzystaniem mechanizmu dziedziczenia. Dodatkowo utworzyłeś formularze do edycji pojedynczych rekordów tabel TowUslug i Kontrah,. Wszystkie zadania , które wykonałeś w tej i poprzednich częściach miały na celu przygotowanie aplikacji do wprowadzenia najważniejszego formularza, jakim jest formularz do wprowadzania faktur. Jego tworzenie zaczniemy w kolejnej części

FORMULARZE Z WIELOMA ŹRÓDŁAMI DANYCH

Większość aplikacji ma swój "punkt węzłowy", na którym koncentruje się znaczna część działań użytkownika i w którym są zgromadzone kluczowe elementy logiki aplikacji .Wszystkie pozostałe fragmenty kodu mają wobec wspomnianego punktu węzłowego charakter służebny - ich zadaniem jest zabezpieczenie integralności i poprawności danych oraz zapewnienie maksymalnej wygody

użytkownika. W naszej aplikacji kluczowym elementem jest formularz do sporządzania faktur. Stwierdzenie to nie wymaga chyba specjalnego uzasadnienia. Wszystkie listy towarów i kontrahentów tworzymy by móc wypełnić fakturę, a wydruki służą do graficznego zaprezentowania jednej lub wielu faktur. Formularz ten będzie najbardziej złożonym w opracowywanej aplikacji - połączymy w nim dane z kilku źródeł (kontrahenci, towary itp.) i jednocześnie poddamy wielorakiej weryfikacji. Oczywiście w bardziej złożonych aplikacjach (czy pakietach aplikacji) punktów węzłowych może być kilka, na przykład obok formularza do sporządzania faktur może się nim okazać formularz do przygotowywania listy wynagrodzeń

Tworzenie formularza do edycji dwóch tabel powiązanych relacją

Do przygotowania formularza faktur użyjemy znanego nam już kreatora formularzy. Jednak sposób postępowania będzie nieco inny niż w poprzednich przykładach. Do tej pory tworzyliśmy formularze wyświetlające zawartość jednej tabeli. Planowany formularz będzie wyświetlał zawartość dwóch tabel. Pierwszą z nich jest tabela TranNagl, w której jak pamiętamy, są gromadzone informacje z nagłówka faktury, takiej jak jej numer, data wystawienia, identyfikator kontrahenta itp. Druga tabela to TranSzcz, w której znajdują się poszczególne pozycje faktury wraz z wyszczególnieniem ceny i ilości. Pierwsza tabela będzie nadrzędna (master) w stosunku do drugiej (detail). Stąd te z tego typu formularze określa się jako master/detail. Aby tworzony formularz wyświetlał zawartość dwóch tabel, wykonaj następujące kroki:

1.Uruchom Delphi

2.W menu Database, wybierz polecenie Form Wizard

3.W pierwszym oknie kreatora zaznacz opcję Create a master/detail form i pozostaw zaznaczoną opcję Create form using TTable objects, a następnie kliknij Next. Pojawi się drugie okno kreatora, w którym należy podać nazwę tabeli głównej (nadrzędnej)

4.W polu Directories przejdź do folderu Dane, a następnie na liście Table Name zaznacz tabele TranNagl i kliknij Next

5.Za pomocą przycisku >> przenieś wszystkie pola tabeli TranNagl na listę Ordered Selected Fields i kliknij Next

6.Zaznacz opcję Horizontally i kliknij Next. W tworzonym formularzu w danym momencie będziemy wyświetlać tylko jedną fakturę, a więc wybór opcji In a grid nie miałby sensu. W kolejnych oknach kreatora będziemy odpowiadali na podobne pytania lecz dotyczące tabeli szczegółowej

7.Zaznacz tabelę TranSzcz i kliknij Next

8.Kliknij przycisk >> aby przenieść wszystkie pola tabeli TranSzcz na listę Ordered Selected Fields, a następnie kliknij Next

9.Upewnij się, że jest zaznaczona opcja In a grid i kliknij Next. Na ekranie pojawi się okno kreatora, z którym dotychczas nie mieliśmy do czynienia. Musimy w nim określić pola łączące tabelę główną z tabelą szczegółową. Jeśli dobrze zaprojektowaliśmy strukturę bazy danych ta operacja nie powinna nastręczyć trudności. Wyjaśnimy w kilku słowach budowę tego okna. Po pierwsze w polu Available Index widnieje tekst Primary Oznacza to ,że do złączenia dwóch źródeł bazy danych posłuży indeks pdstawowy tabeli TranNagl. W skład tego indeksu wchodzi jedno pole, IDTranNagl. Na liście Detail Fields są wyświetlane wszystkie elementy indeksu podstawowego tabeli TranSzcz, a więc pola IDTranNagl i IDTranSzcz. Przycisk Add służy do określenia pól złączonych (joined fields), to znaczy takich, których wartości w obu polach powinny być równe

10.Na liście Detail Fields zaznacz pole IDTranNagl

11.Pole o takiej samej nazwie zaznacz na liście Master Fields, a następnie kliknij Add .W polu Joined Fields pojawiło się wyrażenie : IDTranNagl → IDTranNagl. Zapsi ten oznacza ,że w formularzu pojawią się tylko te rekordy z tabeli TranSzcz których pole IDTranNagl ma taką samą wartość ,jak pole IDTranNagl w bieżącym rekordzie tabeli TranNagl. Jeśli do połączenie używamy indeksu złożonego (składającego się z więcej niż jednego pla), par pól na liście Joined Fields może być więcej

12.Kliknij przycisk Next

13.Upewnij się, że w ostatnim oknie kreatora zaznaczona jest opcja Form Only oraz ,że pole wyboru Genrate main form jest zaznaczone, a następnie kliknij przycisk Finish. Formularz został utworzony.

14.Przejdź do Object Inspectora i zmień właściwość Name formularza na FakturaForm

15.Przejdź do właściwości Caption i wpisz Faktura

16.Zaznacz komponent Table2 i zmień jego właściwości Database na BAZAFAKT. Komponentu Table1 nie modyfikujemy ponieważ za chwilę zostanie on usunięty (w celu uzyskania synchronizacji formularzy)

17.Kliknij przycisk Save i zapisz moduł nowego formularza pod nazwą Faktura

Aby nasz formularz stał się dostępny z poziomu formularza z listą faktur, musimy jeszcze zdefiniować odpowiednie powiązania. Ponieważ czynności te wykonywaliśmy już trzykrotnie, nie powinieneś mieć problemów z ich powtórzeniem. A tej chwili musisz utworzyć odpowiednie procedury obsługi zdarzeń w module FaktL.pas, wywołujące formularz FakturaForm. Informacje znajdziesz w sekcji "Powiązanie formularza do edycji pojedynczego rekordu z formularzem-listą". Aby uzyskać synchronizację danych wyświetlanych na liście faktur i w formularzu jednej faktury, musisz usunąć z tego ostatniego komponent Table1, a następnie właściwości DataSource1.DataSet przypisać komponent typu TTable umieszczony w module danych FaktDModule (nie zapomnij wstawić klauzuli uses FaktD; w sekcji implementation modułu Faktura.pas)

Określanie zależności między tabelą nadrzędną a podrzędną

Przeanalizujemy teraz budowę formularza FakturaForm. Interesuje nas zwłaszcza powiązanie miedzy wyświetlanymi na nim dwiema tabelami. Gdybyśmy teraz wprowadzili kilka faktur, zauważylibyśmy następujący fakt - przy przejściu do innej faktury (zmianie bieżącego rekordu w tabeli TranNagl), w siatce pojawiają się tylko pozycje faktury (rekordy z tabeli TranSzcz), które dotycza tylko tej właśnie faktury. By lepiej zrozumieć ten mechanizm, musimy na chwilę cofnąć się do struktury bazy danych. Pamiętamy, że między tabelami TranNagl i TranSzcz zachodzi relacja jeden-do-wielu. Relację tę definiują pola IDTranNagl zdefiniowane w obydwu tabelach. W przypadku tabeli TranSzcz pole IDTranNagl jest jednym z dwóch składników klucza podstawowego (należy do niego również pole Pozycja). Gdy na utworzonym przed chwilą formularzu wyświetlany jest jakiś rekord z tabeli TranNagl, np. o numerze 5, wówczas na obiekcie Table2, reprezentującym tabelę TranSzcz, zakładany jest filtr, który udostępnia rekordy z tej tabeli o tym samym identyfikatorze, zgodnym z identyfikatorem tabeli nadrzędnej. Zwróćmy uwagę ,że mechanizm ten wykorzystuje powiązanie zdefiniowane w trakcie działania kreatora , a nie poziomie struktury bazy danych. Powiązania tego typu definiujemy często bez pomocy kreatora, warto więc dowiedzieć się w jaki sposób są one zapisane na formularzu:

1.Przejdź do widoku formularza FakturaForm. Na formularzu widoczne są dwa komponenty typu TDataSource i jedne komponent TTable

2.Zaznacz komponent Table2 , a następnie w Object Inspectorze wyświetl kartę Properties. W arkuszu właściwości zwróć uwagę na dwa pola : MasterSource i MasterFields. Pierwsze z nich wskazuje źródło danych, z którym powiązany jest ten obiekt TTable. Jak widać jest nim obiekt DataSource1, powiązany z tabelą TranNagl (poprzez komponent FaktDModule.Table1). Z kolei w polu MasterFields znajdują się nazwy pól z tabeli nadrzędnej, wchodzące w relacje z kluczem podstawowym tabeli podrzędnej - u nsa jest to pole IDTranNagl

Wiedzę tę wykorzystamy do zdefiniowana kolejnego typu zależności na formularzu faktury

Stosowanie pól kombi i pól memo

Niektóre informacje wprowadzane przez kreator są niepotrzebne, część kontrolek trzeba nieco zmodyfikować, zmienić ich położenie oraz inne właściwości itp. Tu skoncentrujemy się na takich "edytorskich" operacjach. Pokażemy jak zastąpić standardowe kontrolki edycyjne zaproponowane przez kreator bardziej zaawansowanymi narzędziami. Po pierwsze założymy ,że liczba możliwych form płatności jest ograniczona do co najwyżej kilku. Jeżeli chcemy zawęzić zbiór wartości, którymi powinien dysponować użytkownik, przy czym zbiór ma charakter stały, dobrym rozwiązaniem jest zastosowanie pola listy lub pola kombi. Ponieważ nasz formularz będzie wyświetlał wiele różnych informacji, a jego powierzchni jest ograniczona zdecydujemy się na tę drugą kontrolkę

1.Zaznacz kontrolkę EditForma_platn, a następnie usuń ją naciskając klawisz Del

2.Na palecie Data Controls zaznacz kontrolę TDBComboBoxm a następnie umieść ja w miejscu z którego zostało usunięte pole tekstowe. W razie potrzeby przesuń nieco pozostałe kontrolki, aby uzyskać miejsce dla wstawionego obiektu

Uwaga : Zanim wstawisz nowa kontrolkę , zaznacz komponent typu TScrollBox, na który zostały umieszczone pozostałe komponenty związane z nagłówkiem faktury. TScrollBox jest tak zwanym komponentem posiadającym, co oznacza ,że może być właścicielem innych komponentów (ich właściwości Owner można przypisać komponentowi TScrollBx). Jeżeli nie postąpisz zgodnie z tym zaleceniem, wstawiana kontrolka może zostać przejęta przez inny komponent posiadający ,co utrudni późniejszą pracę z formularzem - w szczególności nie będzie można prawidłowo ustawić kolejności poruszania się za pomocą klawisza Tab. Po prawidłowym wstawieniu kontrolka powinna być widoczna w okienku Object TreeView jako odgałęzienie obiektu TScrollBox. Komponentem posiadającym oprócz komponentu typu TScrollBox jest m.in. formularz, kontrolki typu TPanel, TradioGroup lub TGroupBox itp. Hierarchię posiadania najlepiej obserwować w okienku Object TreeView, umieszczonym nad okienkiem Object Inspectora. Jak widać, kontrolki danych związane z tabelą TranNagl są w posiadaniu komponentu TScrollBox, który z kolei jest zarządzany przez komponent Panel2, będący własnością formularza FakturaForm

Na formularzu pojawi się nowa kontrolka o nazwie DBComboBox. Kontrolka ta nie jest jeszcze powiązana ze źródłem danych

3.W Object Inspectorze odszukaj właściwość Name i wpisz Forma_platnDBCombo

4.Przejdź do właściwości DataSource i z listy wybierz źródło danych DataSource1 .W tym momencie nowa kontrolka została powiązana ze źródłem danych DataSource1.

5.Przejdź do właściwości DataField i z listy wybierz pole Forma_platn .W kolejnym kroku musimy zdefiniować możliwe formy płaności

6.Przejdź do właściwości Items i kliknij przycisk wielokropka. Ponieważ elementy pola Items są typu TStrings, na ekranie pojawi się okno edytora łańcuchów znaków

7.W oknie dialogowym String List Editor wpisz jedno pod drugim następujące słowa:

gotówka

przelew

czek

8.Kliknij przycisk OK

W tej chwili użytkownik będzie mógł zarówno wybierać z listy jaki i wpisywać swoje własne propozycje z klawiatury. Jeżeli chcesz aby w danym polu użytkownik mógł wybierać tylko wartości proponowane na liście, ustaw właściwość Style pola kombi na csDropDownList

Kolejny problem z jakim musimy sobie poradzić to pole Uwagi. Kreator utworzył dla niego kontrolkę typu TDBEdit. Jednak jak pamiętamy pole to zostało zdefiniowane w bazie danych jako pole typu Memo. Gdybyś teraz skompilował aplikację i spróbował coś wpisać w polu EditUwagi na formularzu FakturaForm, nie będzie to możliwe. Musimy więc zastosować kontrolkę odpowiedniego typu

1.Zaznacz pole EditUwagi i usuń je, naciskając klawisz Del

2.Na palecie DataControls zaznacz kontrolkę typu TDBMemo, a następnie umieść ją na formularzu. Na formularzu pojawi się nowa kontrolka DBMemo1

3.Dopasuj rozmiary wstawionej kontrolki aby była w całości widoczna i nie przesłaniała innych obiektów

4.w Object Inspectorze odszukaj właściwość DataSource i z listy wybierz źródło danych DataSource1

5.Przejdź do właściwości DataField i z listy wybierz pole Uwagi. Kontrolka zostanie powiązana z polem Uwagi w tabeli TranNagl

6.Przejdź do właściwości Name i wpisz UawgiDBMemo

Zmiana kolejności klawisza Tab

Do kolejnych kontrolek ekranu, których właściwość TabStop jest ustawiona na True, można przechodzić za pomocą klawisza Tab .W aplikacjach bazodanowych jest to bardzo dużym ułatwieniem, gdyż użytkownik w trakcie wprowadzania danych z klawiatury nie musi co chwila sięgać po myszkę. Podczas tworzenia formularza kreator domyślnie tak ustawia kolejność klawisza Tab, aby jego naciśnięcie powodowało przechodzenie do sąsiadujących kontrolek. Często jednak zdarza się tak ,że dokonujemy modyfikacji formularza, które polegają na wstawianiu nowych kontrolek edycyjnych lub zmianie uporządkowania już istniejących. Nowe kontrolki są zawsze umieszczane na końcu listy klawisza Tab. W takiej sytuacji programista sam musi zadbać ,aby poszczególne kontrolki były dostępne we właściwej kolejności. W ty celu trzeba kliknąć prawym klawiszem myszki w dowolnym miejscu komponentu posiadającego, a następnie wybrać polecenie Tab Order i w oknie dialogowym ustawić kontrolki we właściwej kolejności. Pracując nas swoimi aplikacjami ,zwróć szczególną uwagę na ten aspekt. Nic tak nie deprymuje użytkownika jak kursor przemieszczający się po ekranie w nieprzewidziany sposób

Ukrywanie wybranych pól

Chociaż kreator bardzo się starał, wygląd formularza wciąż jest daleki od naszych oczekiwań Przede wszystkim, dwa pola z tabeli TranNagl (IDTranNagl i Typ) są używane wewnętrznie przez aplikację, a więc użytkownik nie musi ich widzieć

1.Na formularzu FakturaForm zaznacz kontrolki EditIDTranNagl, EditTyp orz towarzyszące im etykiety

2.Naciśnij klawisz Delete

Zaznaczone kontrolki zostaną usunięte z ekranu i z definicji klasy. Nie oznacza to jednak ,że tracimy do nich dostęp - w dalszym ciągu możemy na ich operować programowo przez komponent TTable1, zdefiniowany w module danych FaktModule

Zmiana tekstu etykiety na formularzu

Kreator tworząc formularz wygenerował również etykiety dla każdej kontrolki TDBEdit. Ich właściwości Caption zostały zdefiniowane na podstawie nazwy pól obsługiwanych przez odpowiednie kontrolki TDBEdit. Ponieważ uzyskane nazwy nie zawsze są wystarczając czytelne, warto zmodyfikować nieco treść etykiet

1.Zaznacz etykietę z tekstem Data_wystaw, przejdź do Object Inspectora i zmień jej właściwość Caption na Data wystawienia

2.W analogiczny sposób zmień właściwości Caption pozostałych etykiet:

Data_trans : Data sprzedaży

Term_platn : Termin płatności

Forma_platn : Forma płatności

IDKontrah : Kontrahent

Rozmieszczanie kontrolek na formularzu

Czynności, które zamierzamy teraz wykonać, choć bardzo ważne z punktu widzenia ergonomiczności i estetyki interfejsu użytkownika, mają charakter manualny. Z tego względu nie będziemy się tym zajmować szczegółowo.

1.Rozmieść kontrolki związane z nagłówkiem faktury. W razie potrzeby zwiększ nieco rozmiary komponentu Panel2 (w pionie) i samego formularza (szerokość)

2.Gdy zakończysz modyfikowanie formularza zapisz go

Modyfikacja właściwości pól wyświetlanych w komponencie TDBGrid

Analogiczne czynności wykonamy teraz na nagłówkach siatki danych z szczegółami faktury. W ich przypadku musimy zastosować takie same zasady postępowania , jak podczas zmiany nagłówków kolumn w formularzach list

1.Dwukrotnie kliknij komponent Table2, aby wyświetlić listę jego pól

2.Na liście zaznacza pole IDTranNagl

3.W Object Inspectorze odzsukać właściwość Visible, a następnie ustaw ją na False

4.Zmdyfikuj inne właściwości pól obiektu Table2. Skorzystaj z tego:

Pole : IDTranSzcz

Właściwość : Visible

Nowa wartość : False

Pole : IDTowUslug

Właściwość : DisplayLabel

Nowa wartość: Kod

Pole : Nazwa

Właściwość : DisplayWidth

Nowa wartość : 40

Pole : KWiU

Właściwość : DisplayWidth

Nowa wartość 12

Pole : J_miary

Właściwość : DisplayLabel

Nowa wartość: j.m

Pole : J_miary

Właściwość : DisplayWidth

Nowa wartość : 4

Pole : Ilosc

Właściwość : DisplayLabel

Nowa wartość Ilość

Pole : Cena_jedn

Właściwość : DisplayLabel

Nowa wartość : Cena jedn.

Pole : St_VAT

Właściwość : DisplayLabel

Nowa wartość : St. VAT

Podobnie jak w przypadku nagłówka faktury, również pole IDTranSzcz z tabeli TranSzcz nie powinno być wyświetlane w siatce danych - służy ono wyłącznie do zdefiniowania relacji z tabelą TranNagl

6.Kliknij przycisk Save All, aby zapisać wprowadzone zmiany

7.Skompiluj i uruchom aplikację

8.Przejdź do formularza FakturaForm i spróbuj wpisać fikcyjną fakturę (nie podawaj kodu kontrahenta). Wypełnij pozycje faktury

Defniowanie podrzędnego źródła danych

W obecnej postaci na formularzu wyświetlany jest tylko identyfikator kontrahenta. Takie rozwiązanie jest nie do przyjęcia - nie możemy zmuszać użytkownika aby pamiętał identyfikatory wszystkich klientów. Wynika z tego ,że na ekranie powinny się również pojawić takie podstawowe informacje o kliencie, jak nazwa oraz adres. Aby uzyskać dane ,zdefiniujemy kolejne podrzędne źródło danych

1.Na palecie BDE zaznacz komponent TTable i umieść go na formularzu obok komponentu DataSource2. Na formularzu pojawi się nowy komponent TTable1. Otrzymał taką nazwę ponieważ usunęliśmy poprzedni komponent TTable1 wygenerowany przez kreator. Ponieważ źródło danych , które za chwilę wstawimy otrzyma nazwę DataSource3,zmienimy teraz nazwę komponentu Table1 na Table3 co ułatwi nam kojarzenie tych dwóch obiektów

2.Gdy zaznaczony jest nowy komponent TTable przejdź do Object Inspectora i zmień właściwość Name na Table3

3.Zmodyfikuj dalsze właściwości komponentu Table3 zgodnie z tym:

DatabaseName : BAZAFAKT

TableName : Kontrah

MasterSource : FaktDModule.DataSource1

ReadOnly : True

Pierwsze dwie właściwości .DatabaseName i TableName, służą do wskazania tabeli w bazie danych, z którą zostanie powiązany komponent Table3. Za pomocą właściwości MasterSource definiujemy nadrzędne źródło danych, którym w tym przypadku jest tabelaTranNagl (wskazywana przez źródło danych FaktDModule.DataSource1). Dane kontrahenta wyświetlane na tym formularzu nie powinny być dostępne do edycji. W takiej sytuacji warto ustawić właściwość ReadOnly na True. Jeśli zestaw danych jest tylko do odczytu, zaleca się ustawienie tej właściwości na True, gdyż zwiększa to wydajność aplikacji i oszczędza zasoby systemowe.

4.Przejdź do właściwości MasterField i kliknij przycisk wielokropka. Pojawi się okno dialogowe Field Link Desginer

5.Na liście Detail Fields zaznacz pole IDKontrah (z tabeli Kontrah), natomiast na liście Master Fields odszukaj pole o takiej samej nazwie (lecz należące do tabeli TranNagl)

6.Kliknij przycisk Add,, a następni OK. Pole łączące budiwe tabele zostało zdefiniowane. Utworzenie takiej zależności jest możliwe, ponieważ pole IDKontrah jest kluczem obcym w tabeli TranNagl i kluczem podstawowym w tabeli Kontrah. Dzięki temu w zestawie danych Table3 będzie dostępny tyko ten rekord z tabeli Kontrah, którego wartość IDKontrah jest równa wartości z pola IDKontrah, ale w tabeli TranNagl

7.Z palety Data Access pobierz komponent TDataSource i umieść obok komponentu Table3. Pojawi się nowy komponent DataSource3

8.Przejdź do Object Inspectora i ustaw właściwość DataSet komponentu DataSource3 na Table3. Możemy przystąpić do wstawiania komórek danych w których będą wykorzystywane informacje o kontrahencie. Do ich wyświetlenia wykorzystamy pola typu TDBEdit

9.W pobliżu pola EditIDKontrah umieść cztery pola TDBEdit oraz jedno pole TLabel, i rozmieść je na formularzu

10.Zaznacz wszystkie nowe pola TDBEdit (skorzystaj z klawisza Shift) i zmdyfikuj ich wspólne właściwości :

DataSource : DataSource3

Enabled : False

Color : clSkyBlue

Zmiana właściwości Enabled na False powoduje ,że w polu tym nie będzie można ustawić kursora. Zmiana właściwości Color oznacza ,że tło pola będzie wyświetlane w innym kolorze - to również jeden ze sposobów poinformowania użytkownika, że wartości w tym polu nie podlegają edycji

11.Poszczególnym komponentom TDBEdit przypisz właściwości Name i DataField :

TDBEdit1 : Name - EditNazwa :DataField - Nazwa

TDBEdit2 : Name - EditMiasto :DataField - Miasto

TDBEdit3 : Name - EditUlica :DataField - Ulica

TDBEdit1 : Name - EditNIP :DataField - NIP

12.Zmień właściwość Caption nowej etykiety na NIP

Operacje związane z otwieraniem wstawionej tabeli

Musimy jeszcze zadbać o to, aby komponent Table2 otwierał się w czasie tworzenia formularza

1.Naciśnij F12 aby przejść do widoku kodu, odszukaj procedurę zdarzenia FormCreate i uzupełnij ją o wiersz kodu:

Table3.Open

2.Kliknij przycisk Save, aby zapisać wprowadzone zmiany

3.Przejdź do formularza FakturaForm, w polu Kontrahent wpisz identyfikator NOWAK i naciśnij klawisz TAB aby przejść do następnej kontrolki. Jeżeli podany identyfikator istnieje w tabeli Kontrah, w nowych kontrolkach TDBEdit pojawią się dane kontrahenta

4.Korzystając z paska nawigatora, przejdź do innego rekordu tabeli TranNagl

5.Powróć do poprzedniego rekordu

Jak widać, dzięki powiązaniu zdefiniowanemu w komponencie Table3 (właściwość MasterSource i MasterFields), wartości w kontrolkach związanych z danymi kontrahenta są natychmiast odświeżane. Oczywiście nie możemy oczekiwać, że użytkownik będzie pamiętał identyfikatory wszystkich kontrahentów lub przed wystawieniem każdego dokumentu sprzedaży wychodził z formularza FakturaForm, wracał do menu i wyświetlał sobie listę kontrahentów, a następnie wracał znowu do wystawienia faktur. Wniosek jest oczywisty - na formularzu FakturaForm powinna być możliwość podglądu zdefiniowanych kodów kontrahentów

Stosowanie pól TDBLookupCombo do wyświetlania listy dopuszczalnych wartości

Zwróć uwagę ,że jedyną informacją o kontrahencie przechowywaną w tabeli TranNagl jest jego identyfikator zaś pozostałe jego dane są uzyskiwane poprzez relację z tabelą Kontrah, której kluczem jest właśnie pole IDKontrah. Z tego spostrzeżenia wynika fakt ,że nie będzie można wystawić faktury

dla kontrahenta, którego danych nie wprowadzono do tabeli Kontrah. Spróbujmy wykorzystać te wnioski do zmodyfikowania formularza FakturaForm. Jeżeli wiemy ,że w jakimś polu może się pojawić tylko wartość, dal której istnieje odpowiedni zapis w tabeli podrzędnej (tabela Kontrah), najwygodniej jest użyć komponentu TDBLookupCombo. W polu tego typu są wyświetlane wartości z tabeli podrzędnej, nazywanej również tabelą odnośników (lookup table)

1.Usuń z formularza komponent EditIDKontrah

2.Zaznacz komponent TDBLookupCombo na palecie Data Controls i umieść go w tym samym miejscu , w którym znajdowało się usunięte pole tekstowe. Musimy jeszcze zdefiniować źródło danych dla nowej kontrolki. Potrzebne są do tego komponenty TTable i TDataSource<nr>

3.Z palety BDE pobierz komponent TTable i umieść na formularzu. Zmień jego właściwości :

Name : Table4 DatabaseName: BAZAFAKT TableName : Kontrah ReadOnly : True Active : True

Chociaż na formularzu jest już jeden zestaw danych odwołujący się do tabeli Kontrah, nie możemy z niego skorzystać ponieważ został powiązany relacją master / detail z tabelą TranNagl - po rozwinięciu listy w polu kombi pojawiłby się tylko identyfikator kontrahenta zgodny z wartością przechowywaną w polu IDKontrah w tabeli TranNagl

4.Obok komponentu Table4 umieść komponent TDataSource z palety Data Access. Na formularzu pojawi się komponent DataSource4

5.Przejdź do Object Insepctora i ustaw właściwość DataSet nowego komponentu na Table4. Po zdefiniowaniu źródła danych musimy jeszcze określić sposób działania pola kombi.

6.Zaznacz komponent DBLookupComboBox1 i zmień jego właściwości:

DataSource : DataSource1 DataField : IDKontrah ListSource : DataSource4 KeyField : IDKontrah ListField : IDKontrah; Nazwa DropDownWidth : 200 NullValueKey : Del Name : IDKontrahDBLookup

Właściwości DataSource i DataField , podobnie jak w innych kontrolkach danych, wskazują ple tabeli, w którym będzie przechowywana wartość wybrana w tym polu kombi. Z kolei właściwości ListSource i KeyField wskzują źródło danych i pole, którego wartość powinna być zgodna z wartością przechowywaną w polu wskazanym we właściwości DataField. We właściwości ListField podajemy pole lub pola, które będą wyświetlane po rozwinięciu listy. W naszym przypadku chcemy aby obok identyfikatora kontrahenta wyświetlała się również jego nazwa. Ponieważ wyświetlane informacje zajmą dużo miejsca, musimy odpowiednio zmodyfikować wartość DropDownWidth (domyślna wartość 0 oznacza ,że rozwinięta lista ma taką samą szerokość jak pole kombi). I wreszcie NullValueKey. Pole kombi ma tą cechę, że podczas wpisywania w nim kolejnych znaków, automatycznie proponowane są najbardziej pasujące wartości klucza. Może się jednak zdarzyć ,że po wypełnieniu pola IDKontarhenta z jakichś względów będziemy chcieli usunąć wyświetlaną po nim wartość. Nie jest to jednak zwykłe pole tekstowe, a wiec tradycyjne metody nie zadziałają. Musimy wtedy skorzystać z właściwości NullValueKey. Przypisujemy jej nazwę klawisza (lub kombinacji klawiszy), który będzie odpowiedzialny za czyszczenie tego pola W naszym przykładzie wybraliśmy Del

7.Przejdź do widoku kodu i uzupełnij procedurę zdarzenia FormCreate o poniższy wiersz:

Table4.Open;

8.Kliknij przycisk Save All , aby zapisać wprowadzone zmiany

9.Skompiluj i uruchom aplikację.

10.Przejdź do formularza Faktura i rozwiń listę w polu kombi

11.Z listy kontrahentów wybierz jeden z kodów. Po wybraniu kodu powinny się zaktualizować pozostałe pola dotyczące kontrahent na formularzu Faktura

Podsumowanie

W tej części poznałeś zasady tworzenia formularzy korzystających z kliku powiązanych ze sobą zestawów danych. Formularz FakturaForm korzysta obecnie z jednego nadrzędnego zestawu danych (tabela TranNagl) oraz trzech zestawów podrzędnych (jeden dla tabeli TranSzcz i dwóch dla tabeli Kontrah). Dzięki temu, w trakcie edycji pojedynczej faktury będą wyświetlane wszystkie rekordy z tabeli TranSzcz o takim samym identyfikatorze transakcji oraz dane kontrahenta o identyfikatorze zgodnym z identyfikatorem zapamiętanym w nagłówku transakcji. Dowiedziałeś się również ,jak używać pola TDBLookupCombo do wyświetlania listy dopuszczalnych wartości utworzonej tabeli odnośników.