Praktyczna aplikacja bazodanowa : Program do fakturowania - Cz. I

STRUKTURA BAZY DANYCH

Przed rozpoczęciem tworzenia formularza aplikacji musimy najpierw zdefiniować strukturę danych w których będą gromadzone informacje przetwarzane przez program. Dane w bazach są przechowywane w tabelach. Każda tabela składa się z rekordów, które z kolei zwierają pola określonego typu. W rekordach (wierszach) znajdują się dane opisujące określony element rzeczywistości . Poszczególne pola tworzą kolumny tabeli, stąd też czasami obydwu pojęć używa się zamiennie. Każde z pól [kolumn] w tabeli bazy danych musi być określonego typu - jest to jeden z mechanizmów zapewnienia integralności danych. W środowisku Delphi narzędziem do tworzenia tabel jest program Database Desktop (DBD). Pozwala on między innymi na definiowanie tabel w formacie Paradox, z którego będziemy korzystać. Format Paradox jest zalecany w przypadku tworzenia aplikacji jednowarstwowych (nie typu klient-serwer). Tabele w tym formacie są najlepiej obsługiwane przez komponenty dostępu do danych Delphi i kontrolki z palety BDE.

W tabelach Paradox można stosować pola następujących typów :

Typ : Alpha

Symbol : A

Rozmiar : 1 - 125

Opis : Wartości alfanumeryczne (łańcuchy tekstowe). Mogą to być litery, liczby, symbole specjalne i inne drukowalne znaki kodu ASCII. Wartość określającą rozmiar deklaruje się w momencie tworzenia tabeli

Typ : Number

Symbol : N

Opis : Zmiennoprzecinkowa wartość liczbowa

Typ : Money

Symbol : $

Opis : Pole tego typu , podobnie jak pole Number, może zawierać tylko liczby. Różnica między nimi jest taka ,że to ostatnie jest domyślnie formatowane w zapisie walutowym (zgodnie z ustawieniami formatu waluty w systemie Windows)

Typ : Short

Symbol : S

Opis : Pole do przechowywania krótkich wartości całkowitych (z przedziału od -32 767 do 32 767)

Typ : LongInteger

Symbol : I

Opis : Pole do przechowywania długich wartości całkowitych (z przedziału od -2 147 483 do 2 147 483 647)

Typ : BCD

Symbol : #

Rozmiar : 0-32

Opis : Pola liczb dziesiętnych kodowanych binarnie BCD (Binary Coded Decimal). Stosowany między innymi w obliczeniach o wysokim poziomie dokładności

Typ : Date

Symbol : D

Opis : Pole daty. Wpisując datę w takim polu, nie musimy sprawdzać jej poprawności

Typ : Time

Symbol : T

Opis : Pole godziny; ma analigiczne właściwości jak pole daty

Typ : Timestamp

Symbol : D

Opis : Pole "datownika" ; połączenie daty i godziny

Typ : Memo

Symbol : M

Rozmiar : 1-240

Opis : To pole służy do przechowywania dużych informacji tekstwowych (które nie mieszczą się w 255 bajtach pola typu Alpha). Teoretycznie, można w nim przechowywać dane dowolnej długości - jedynym ograniczeniem jest ilość wolnego miejsca na dysku. Mimo to dla pola Memo określa się rozmiar - są to bajty przechowywane wraz z tabelą, w któ®j zdefiniowano to pole. Pozostałe bajty są przechowywane w pliku o rozszerzeniu .MB i nazwie takiej samej jak nazwa tabeli

Typ : Formated Memo

Symbol : F

Rozmiar : 0-240

Opis : Przypomina pole typu Memo z tą różnicą, że zapisany w nim tekst może być sformatowany.

Typ : Graphic

Symbol : G

Rozmiar : 0 -240

Opis : Pole do przechowywania obrazów w formatach .BMP, .PCX, .TIF, .GIF, oraz .EPS

Typ : OLE

Symbol : O

Rozmiar : 0 -240

Opis : To pole umożliwia przechowywanie danych różnych typów, takich jak obrazy, dźwięki, dokumenty tekstowe itp. Nie trzeba określać rozmiaru tego pola, ponieważ jego zawartość jest przechowywana w oddzielnych plikach

Typ : Logical

Symbol : L

Opis : Pole logiczne. Domyślnie może przybierać wartość True lub False

Typ : AutoIncrement

Symbol : +

Opis : Pole, którego wartość jest automatycznie zwiększana po wstawieniu po wstawieniu kolejnego rekordu. Wartość ta nie może być edytowana. Pole to pozwala uzyskać niepowtarzalność rekordów - w danej tabeli nie znajdą się dwa rekordy o takiej samej wartości pola Autoincrement

Typ : Binary

Symbol : B

Rozmiar : 0 - 240

Opis : Pole binarne, służące do przechowywania danych, których nie jest w stanie zinterpretować program Database Desktop. Nie trzeba określać rozmiaru tego pola, ponieważ jego zawartość jest przechowywana w oddzielnych plikach

Typ : Bytes

Symbol : Y

Rozmiar : 1 -255

Opis : Pole bajtowe, służące do przechowywania danych, których nie jest w stanie zinterpretować program Database Desktop

Rozmiar pola Memo, Formated Memo, Grpahic, OLE oraz Binary określa liczbę bajtów przechowywanych w tabeli. Jeżeli np. rozmiar pola Memo wynosi 100, wówczas 100 pierwszych bajtów tego pola będzie przechowywanych w tabeli wraz z innymi danymi, natomiast reszta zawartości tego pola zostanie zapisana jako plik o rozszerzeniu .MB. Jeżeli przewidujesz ,że znacząca większość wartości w tym polu będzie zabierała mniej niż 240 bajtów, lub planujesz wykorzystywać początkowe bajty pola (np. do sortowania), określ odpowiedni rozmiar - w ten sposób prawdopodobnie przyspieszysz działanie aplikacji

Plik bazy danych w formacie Paradox

Baza danych w formacie Paradox składa się z wielu plików. Każda utworzona tabela jest zapisywana w oddzielnym pliku. Osobno są również zapisywane poszczególne indeksy i plik z warunkami poprawności. Rozwiązanie takie ma swoje zalety i wady .Dzięki podziałowi bazy danych na pliki mamy do nich większy dostęp z poziomu systemu operacyjnego i możemy w wygodny sposób kontrolować ich lokalizację oraz wykonywać różnego rodzaju operacje (usuwanie zawartości pojedynczej tabeli itp.) Z drugiej strony , baza danych jest w znacznym stopniu uzależniona od działania systemu operacyjnego, co oznacza ,że aparat bazy danych ma mniejsze szanse na zastosowanie własnych mechanizmów zarządzania i optymalizacji , które odgrywają coraz większą rolę w nowoczesnych

systemach zarządzania relacyjnymi bazami danych. Poniżej mamy opisane rozszerzenia plików powstających podczas tworzenia tabel w formacie Paradox:

DB : plik z danymi

MB : plik z danymi typu Memo, Formated Memo lub Graphic

PX : plik z indeksem (kluczem) podstawowym

Xnn, Ynn : Pliki z prostymi indeksami dodatkowymi

XGn, YGn : Pliki ze złożonymi indeksami dodatkowymi

VAL : Plik z regułami poprawności oraz integralności referencyjnej

Tworzymy pierwszą tabelę

Po wstępie teoretycznym możemy przystąpić do realizacji pierwszego zadania praktycznego. Spróbujemy utworzyć tabele ,na które będzie operować nasza aplikacja. Rozpoczniemy od najprostszej z nich, przeznaczonej do gromadzenia danych kontrahentów.

Tabela kontrahentów

1.W menu File, New wybierz Table. Pojawi się okno dialogowe Create Table

2.Upewnij się ,że polu Table Type zaznaczona jest opcja Paradox 7, a następnie kliknij przycisk OK. Na ekranie pojawi się okno dialogowe Create Paradox 7 Table. Możemy teraz przystąpić do definiowania pól tabeli. Program Database Desktop umożliwia również definiowanie tabel w innych formatach, na przykład dBAse IV, Oracle, Informix czy MSSQL. Ponieważ tworzymy prostą aplikację jednowarstwową, format Pardox wystarcza całkowicie. Ponadto użycie tego formatu zapewnia największą swobodę w korzystaniu z komponentów Delphi.

3.Gdy zaznaczone jest pierwsze pole w kolumnie Field Name, wpisz IDKontrah i naciśnij klawisz [Tab]. Zostanie zaznaczone pole Type

4.W polu Type wpisz A i naciśnij klawisz [Tab]. Identyfikator kontrahenta (pole IDKontrah) będzie typu znakowego, umożliwi to przypisywanie kontrahentom zarówno symboli liczbowych , jak i nazw symbolicznych (skróconych)

5.W pole Size wpisz 12 i naciśnij [Tab]. Wartość ta oznacza ,że identyfikator klienta nie może być dłuższy niż 12 znaków

6. W pole Key wpisz * . Gwiazdka (*) oznacza ,że to pole należy do indeksu podstawowego. Indeks podstawowy decyduje o uporządkowaniu rekordów w tabeli. Wynika z tego ,że tworzona tabela kontrahentów będzie porządkowana według ich identyfikatorów. Podzbiór pól wchodzących w skład klucza podstawowego musi być niepowtarzalny. W naszym przypadku, ponieważ do indeksu podstawowego tej tabeli będzie należeć tylko pole IDKontrah, wszystkie jego wartości w tej tabeli muszą być unikatowe. Na szczęście zadba o to sam aparat bazy danych - jeżeli w trakcie działania programu użytkownik spróbuje wprowadzić kontrahent o kodzie, który jest już używany w innym rekordzie , pojawi się komunikat o błędzie i nowy rekord nie zostanie zapisany w bazie. Zadaniem programisty pozostaje jedynie odpowiednie obsłużenie takiego błędu - użytkownik powinien zostać poinformowany o zaistniałej sytuacji, by mógł na nią właściwie zareagować.

7.Naciśnij klawisz [Tab]. Do tabeli zostanie wstawiony nowy wiersz i zaznaczona będzie kolumna Field Name

8.Posługując się poniższa tabelą , zdefiniuj pozostałe pola tabeli:

Nazwa pola : Nazwa

Typ : A

Rozmiar : 80

Nazwa pola : NIP

Typ : A

Rozmiar : 13

Nazwa pola : Miasto

Typ : A

Rozmiar : 80

Nazwa pola : Ulica

Typ : A

Rozmiar : 80

Nazwa pola : Bank

Typ : A

Rozmiar : 40

Nazwa pola : Konto

Typ : A

Rozmiar : 40

Nazwa pola : Telefon

Typ : A

Rozmiar : 40

Nazwa pola : Telefon_kom

Typ : A

Rozmiar : 20

Nazwa pola : Kontakt

Typ : A

Rozmiar : 80

Nazwa pola : Odbier_fakture

Typ : A

Rozmiar : 80

Definiując nazwy pól, staraj się unikać stosowania polskich znaków oraz odstępów w nazwach. Chociaż Database Desktop i Delphi akceptują takie ustawienia, możesz mieć wiele kłopotów, gdy w przyszłości zajdzie potrzeba połączenia twoje bazy danych z jakimś innym systemem. Z tego samego względu staraj się również tworzyć stosunkowo krótkie nazwy pól - niektóre starsze systemy baz danych mają limitowane długości nazw. Oczywiście skracanie nazwy pola nie powinno się odbywać kosztem jej czytelności - żaden programista nie ucieszy się , gdy będzie musiał napisać aplikację operująca na tabeli, w której pola nazywają się np. P1, P2, W itp. Jeśli chcesz używać nazwy wieloczłonowej, poszczególne jej elementy połącz znakami podkreślenia

9.Kliknij przycisk Save As. Pojawi się okno dialogowe Save Table As

10.W polu Nazwa pliku wpisz Kontrah, a następnie kliknij przycisk Zapisz

W przypadku nazw tabel zaleca się stosowanie podobnych zasad ,jak przy definiowaniu nazw pól tabeli - należy unikać stosowania znaków diakrytycznych i odstępów. Dodatkowo warto stosować nazwy nie dłuższe niż 8 znaków. Ułatwi to zarządzanie tabelami w aplikacjach i ich integrację z innymi systemami baz danych

Definiowanie warunków poprawności pól tabeli

Zanim zapiszemy zdefiniowaną tabelę Kontrah, wprowadzimy jeszcze kilka warunków poprawności (validity checks) .Pozwalają one dodatkowo zawęzić przedział danych, które można wprowadzać w poszczególnych polach. Warunki te wprowadzimy w polu IDKontrah - chcemy ,aby wpisywane litery były zawsze wielkie.

1.W menu Tools, Utilities wybierz polecenie Restructure

2.W oknie dialogowym Select File zaznacz plik Kotrah i kliknij przycisk Otwórz. Na ekranie pojawi się okno dialogowe Restructure Paradox 4 Table, przedstawiające strukturę tabeli Kontrah. Podczas zapisywania struktury tabeli program Database Desktop analizuje zastosowane typu pol i wybiera jak najprostszy format zapisu. W przypadku tabeli Kontrah wystarczył zapis w formacie Paradox 4. Mechanizm ten pozwala na uzyskanie zgodności z aplikacjami starszego typu , przy jednoczesnym zachowaniu pełnej funkcjonalności tabel (wcześniejsze wersje formatu Paradox są zgodne z nowszymi)

3.Upewnij się ,że zaznaczone jest pole IDKontrah oraz, że na liście rozwijanej Table Properties wybrana jest opcja Validity Chekcs, a następnie zaznacz pole wyboru Required Field. Zaznaczenie tej opcji powoduje ,że wpisanie wartości we wskazanym polu będzie wymagane.

4.Kliknij w polu tekstowym Picture i wpisz w nim *!. Wykrzynik oznacza ,że wszystkie wpisane w tym polu znaki będą przekształcane na wielkie litery. Gwiazdka oznacza ,że umieszczony za nią znak może wystąpić wielokrotnie. W rezultacie uzyskujemy maskę, która pozwala na wprowadzenie dowolnej liczby znaków, przy czym znaki alfabetu będą zapisywane wielkimi literami. Pole tekstowe Picture oferuje o wiele więcej możliwości. Jeżeli chcesz je przetestować, kliknij znajdujący się poniżej przycisk Assist. Pojawi się wówczas okno dialogowe Picture Assistance. Możesz skorzystać z przykładowych masek zdefiniowanych na liście rozwijanej Sample pictures lub utworzyć nową maskę w polu Picture. Omawiane okno dialogowe pozwala również na natychmiastowe przetestowanie utworzonej maski. Wystarczy przejść do pola Sample value i wpisać jakąś wartość. Jeżeli jest ona niezgodna z maską, zostanie wygenerowany sygnał dźwiękowy i pojawi się komunikat Picture does not accept value. Poniższa tabela przedstawia pełny wykaz symboli stosowanych w maskach :

Symbol : #

Opis : Cyfra

Symbol : ?

Opis : Dowolna litera (mała klub wielka)

Symbol : &

Opis : Dowolna litra zamieniana na wielką

Symbol : ~

Opis : Dowolna litera zamieniana na małą

Symbol : @

Opis : Dowolny znak

Symbol : !

Opis : Dowolny znak, litery zamieniane na wielkie

Symbol : ;

Opis : (średnik) Następny znak będzie traktowany dosłownie, a nie jako symbol maski

Symbol : *

Opis : Dowolna liczba powtórzeń następnego znaku (najczęściej jednego z symboli maski)

Symbol : [abc]

Opis : Opcjonale znaki : a,b lub c]

Symbol : [a,b,c]

Opis : Opcjonalne znaki : a,b lub c

Na przykład maska do wprowadzania kodu pocztowego mogłaby wyglądać następująco :

####

Masek nie należy jednak nadużywać. Zbytnie ograniczenie zakresu danych , które może wprowadzić użytkownik, czasami staje się dla niego kłopotliwe. Rozważmy przedstawiony przed chwilą przykład kodu pocztowego. Jeżeli w tabeli z takim polem zapisujemy tylko adresy krajowe, wszystko będzie w porządku. Co jednak zrobić, jeżeli użytkownik zechce wprowadzić kod pocztowy kraju, w którym stosowane są oznaczenia 6 cyfrowe lub oprócz cyfr występują również litery? Na tym etapie projektowania bazy danych pojawia się wiele podobnych pytań : czy zastosowana maska nie jest zbyt restrykcyjna, czy długość pola jest wystarczająca? Szukając na nie odpowiedzi ,trzeba wykazać się dużą dalekowzrocznością. Jeżeli mamy wątpliwości, warto zakładać na wyrost. Na przykład skoro nie ma przeciwwskazań, by w tabeli były gromadzone również adresy zagraniczne, dlaczego ograniczać format kodu pocztowego? Nad każdym z takich pytań trzeba się dobrze zastanowić, gdyż poprawki, wprowadzane w dalszych fazach pracy, będą o wiele bardziej kosztowne. Maskę wprowadzania można również zdefiniować w kontrolkach tyupu TDBEit, umieszczonych na formularzach. Jeżeli to jednak możliwe, odpowiednie ograniczenia należy wprowadzić już na poziomie struktur danych, a nie dopiero w aplikacji klienckiej. Sprzyja to zmniejszeniu rozmiarów kod, co upraszcza logikę aplikacji i zmniejsza ryzyko popełnienia ewentualnych błędów. Oprócz określenia maski można również

definiować dodatkowe warunki poprawności pól. Różnią się one w zależności od typu pola. Na przykład dla pola tekstowego (typ A) można określić wartość minimalną(Minimum value), maksymalną (Maximum value) oraz domyślną (Default value) . Opcje te są dostępne z prawej strony okna dialogowego Restructure/Create Table , gdy na liście rozwijanej Table Properties aktywna jest opcja Validity Checks

5.Kliknij przycisk Save.Pojawi się okno dialogowego Restructur Warning, Okno to jest wyświetlane, gdy zmiany wprowadzone w strukturze tabeli mogą wpłynąć na zawarte w niej dane. W naszym przypadku wprowadziliśmy warunek poprawności wymuszający stosowanie wielkich liter w identyfikatorach klientów. W związku z tym musimy zadecydować czy dotychczas zgromadzone w tabeli dane (oczywiście w tej chwili jest ona jeszcze pusta) będą zmodyfikowane zgodnie z wprowadzonym ograniczeniem. Jeżeli chcesz uzyskać wspomniany efekt zaznacz opcję Yes

6.Kliknij przycisk OK. Zmiany wprowadzone w strukturze tabeli Kontrah zostaną zapisane

Tworzymy pozostałe tabel

Po utworzeniu pierwszej tabeli z pewnością nie będziesz miał trudności z wprowadzeniem następnej definicji Oto wykaz tabel, które musisz zdefiniować, zanim przystąpisz do tworzenia aplikacji. Postępuj zgodnie z opisem definiowania nowej tabeli , podanym przy "Tabeli kontrahentów"

Tabela towarów i usług

W tej tabeli będzie przechowywany wykaz towarów i usług sprzedawanych przez naszą firmę

1.Zdefiniuj tabelę TowUslug zgodnie z poniższym opisem

Nazwa pola : IDTowUslug

Typ : A

Rozmiar : 12

Klucz : *

Nazwa pola : Nazwa

Typ : A

Rozmiar : 80

Klucz :

Nazwa pola : KWiU

Typ : A

Rozmiar : 30

Klucz :

Nazwa pola : J_miary

Typ : A

Rozmiar : 10

Klucz :

Nazwa pola : Stan

Typ : N

Rozmiar :

Klucz :

Nazwa pola : Stan_min

Typ : N

Rozmiar :

Klucz :

Nazwa pola : Cena_zak

Typ : $

Rozmiar :

Klucz :

Nazwa pola : Cena_netto

Typ : $

Rozmiar :

Klucz :

Nazwa pola : St_VAT

Typ : A

Rozmiar : 2

Klucz :

W omawianej tabeli wykorzystujemy , oprócz pola znakowego także dwa inne typu pól : N - pole numeryczne oraz $ - pole walutowe .W zasadzie obydwa pola służą do przechowywania wartości zmiennoprzecinkowych. Podstawowa różnica między nimi polega na tym ,że pole typu Currency ($) jest wyświetlane w formacie walutowym - z odstępem co trzy pozycje dziesiętne, dwoma miejscami po przecinku i symbolem waluty. Domyślnie do wyświetlania stosowany jest format waluty zdefiniowany w systemie Windows

2.W polu IDTowUslug definiuj maskę w postaci *!

3.Kliknij przycisk SaveAs , w polu Nazwa pliku wpisz TowUslug i kliknij przycisk Zapisz

Tabela nagłówka transakcji

Tabele Kontrah i TowUslug utworzone w powyżej, chociaż bardzo ważne, mają charakter pomocniczy. Kluczowe informacje związane ze sprzedażą będą jednak przechowywane w dwóch innych tabelach, które utworzymy za chwilę. Pierwsza z nich będzie zawierać dane nagłówka faktury, takie jak jej numer, data wystawienia, identyfikator kontrahenta itp. W drugiej znajdują się rekordy związane z poszczególnymi sprzedanym towarami lub usługami. Rozpoczniemy od utworzenia tabeli nagłówka faktury.

Zdefiniuj tabelę TrangNagl, zgodnie z powyższym opisem:

Nazwa pola : IDTrangNagl

Typ : +

Rozmiar :

Klucz : *

Nazwa pola : Typ

Typ : A

Rozmiar : 5

Klucz :

Nazwa pola : Seria

Typ : A

Rozmiar : 12

Klucz :

Nazwa pola : Numer

Typ : I

Rozmiar :

Klucz :

Nazwa pola : Dopisek

Typ : A

Rozmiar : 12

Klucz :

Nazwa pola : Data_wystaw

Typ : D

Rozmiar :

Klucz :

Nazwa pola : Data_trans

Typ : D

Rozmiar :

Klucz :

Nazwa pola : Term_platn

Typ : D

Rozmiar :

Klucz :

Nazwa pola : Forma_platn

Typ : A

Rozmiar : 12

Klucz :

Nazwa pola : IDKontrah

Typ : A

Rozmiar : 12

Klucz :

Nazwa pola : Uwagi

Typ : M

Rozmiar : 80

Klucz :

Tej tabeli musimy poświęcić nieco więcej uwagi. Pole IDTranNagl, jak można się domyślić, będzie zawierało niepowtarzalny identyfikator każdego rekordu nagłówka transakcji. Jest ono typu AutoIncrement (o czym informuje znak "+"), a więc o automatyczne zwiększanie jego zawartości zadba sam aparat bazy danych . Polu temu jest przypisywany następny wolny numer wyznaczany na podstawie wewnętrznego licznika tabeli. Numer raz wykorzystany nigdy nie jest używany powtórnie. Jeżeli na przykład do tabeli TranNagl wprowadzimy pięć rekordów i w polu IDTranNagl uzyskują one kolejno wartości 1,2,3,4 i 5, a następnie usuniemy rekordy 4 i 5, to po utworzeniu nowego rekordu uzyska on wartość 6. Pole Typ definiujemy nieco na wyrost, gdyż będzie ono przeznaczone do określania typu transakcji. W naszym programie co prawda ograniczymy się do transakcji będących dokumentami sprzedaży, czyli fakturami. Nic nie stoi jednak na przeszkodzie, aby w przyszłości tych samych tabel użyć do gromadzenia dokumentów zakupu, dokumentów wydania

(WZ) itp. Rodzaj transakcji, którego dotyczy określony rekord, będzie wskazywany właśnie przez pole Typ. Pola Seria, Numer i Dopisek w obrębie danego typu transakcji powinny tworzyć niepowtarzalne oznaczenie transakcji, na przykład "FA/31/2012" lub "FA/124/C" .Więcej o tych polach napiszemy później. Kolejne trzy pola nie powinny przysporzyć problemu .Są one typu Date i będą w nich przechowywane odpowiednio : data wystawienia dokumentu (Data_wystaw), data sprzedaży (Data_trans) oraz termin płatności (Term_platn). W polu IDKontrah znajdzie się identyfikator kontrahenta, którego dotyczy transakcja. Zwróć uwagę ,że nazwa tego pola jest taka sama ,jak nazwa indeksu podstawowego w tabeli Kontrah. Nie jest to żadna reguła, lecz dobra praktyka programistyczna - w ten sposób sygnalizujemy ,że obydwa pola są ze sobą powiązane relacją. Oczywiście ,ze zbieżności nazw w praktyce nie nie wynika - konieczne jest jeszcze zdefiniowanie odpowiedniej zależności (relacji) między tabelami. Ostatnie pole, Uwagi, jest przeznaczone do przechowywania informacji niezwiązanych bezpośrednio ze sprzedawanym towarem. Może się w nim znaleźć na przykład numer zamówienia, dodatkowe informacje dotyczące dostawy lub nota o naliczaniu odsetek w przypadku zwłoki z płatnością. Pole to jest typu Memo, co oznacza ,że ilość zawartych w nim danych jest praktycznie nieograniczona.

Definiowanie indeksu dodatkowego w tabeli nagłówka transakcji

Aby uzyskać pewność ,że w tabeli nagłówków transakcji nie pojawią się dwa rekordy o takim samym typie, serii, numerze i dopisku, zdefiniujemy indeks dodatkowy (secondary index) bez powtórzeń .(Indeksy dodatkowe, w przeciwieństwie do indeksów podstawowych, mogą być z powtórzeniami lub bez)

1.W menu Tools, Utilities wybierz polecenie Restructure

2.Zaznacz plik TranNagl i kliknij przycisk Otwórz. Pojawi się okno dialogowe Restructure Table przedstawiające definicję tabeli TranNagl

3.Na liście Table properties z prawej strony wybierz opcję Secondary Indexes

4.Kliknij przycisk Define. Na ekranie pojawi się okno dialogowe Define Secondary Index

5.Na liście z lewej strony dwukrotnie kliknij pole Typ. Pole to zostanie przeniesione na listę Indexed fields

6.W ten sam sposób przenieś na listę pól indeksu kolejno pola Seria, Dopisek i Numer

7.Zaznacz opcję Unique .Zaznaczenie tej opcji spowoduje powstanie indeksu bez powtórzeń

8.Kliknij przycisk OK. Pojawi się okno dialogowe Save Index As

9.W polu Index name wpisz dokument, a następnie kliknij przycisk OK. Ponownie aktywne stanie się okno Restructure Table. Na liście z prawej strony widoczna będzie nazwa zdefiniowanego indeksu.

10.Kliknij przycisk Save

Utworzony przed chwilą indeks da nam również inną korzyść, mianowicie w prosty sposób będziemy mogli sortować i filtrować dokumenty określonych typów. Z tego względu istotna jest kolejność, w jakie te pola zostały umieszczone w definicji indeksu. Dotyczy to zwłaszcza pola Typ. Dzięki temu , że jest ono na pierwszym miejscu, znacznie uprości się filtrowanie danych. Jeżeli na przykład wszystkie dokumenty sprzedaży będą miały w tym polu wpisaną wartość SPRZ a dokumentu zakupu wartość ZAK, odfiltrowanie poszczególnych kategorii dokumentów nie przysporzy problemu .Dzięki temu użytkownik będzie widział tylko dokumenty sprzedaży lub tylko dokumenty zakupu. Gdyby to pole znalazło się na dalszej pozycji , konieczne byłoby utworzenie dla niego indeksu dodatkowego (secondary index). W przeciwnym razie w trakcie filtrowania tworzony byłby indeks doraźny lub następowałoby skanowanie całej tabeli - gdy w tabeli będzie już wiele danych, mogłoby to powodować znaczne spowolnienie programu

Tabela pozycji transakcji

Pozostało nam jeszcze zdefiniowane w tabeli TranSzcz, w której gromadzone będą rekordy dotyczące poszczególnych pozycji transakcji. Zdefiniuj tabelę TranSzcz zgodnie z poniższym opisem :

Nazwa pola : IDTranNagl

Typ : I

Rozmiar :

Klucz : *<br

Nazwa pola : IDTranSzcz

Typ : +

Rozmiar :

Klucz : *<br

Nazwa pola : IDTowUslug

Typ : A

Rozmiar : 12

Klucz : <br

Nazwa pola : Nazwa

Typ : A

Rozmiar : 80

Klucz : <br

Nazwa pola : KWiU

Typ : A

Rozmiar : 30

Klucz : <br

Nazwa pola : J_miary

Typ : A

Rozmiar : 10

Klucz : <br

Nazwa pola : Ilosc

Typ : N

Rozmiar :

Klucz : <br

Nazwa pola : Cena_jedn

Typ : $

Rozmiar :

Klucz : <br

Nazwa pola : St_VAT

Typ : A

Rozmiar : 2

Klucz :

Indeksem podstawowym tej tabeli są pola IDTranNagl i IDTranSzcz. Pierwsze z nich wskazuje nagłówek transakcji, do której należą poszczególne pozycje. Razem z pola te muszą tworzyć unikatową parę - wymaga tego ograniczenie klucza podstawowego. Ponieważ pole IDTranSzcz jest typu AutoIncrement , już samo to pole zapewnia niepowtarzalność wartości klucza. Kolejne pole to IDTowUslug. Jak można się domyślić, znajdziesz w nim identyfikator towaru lub usługi pobrany z Tabeli TowUslug. Uważny czytelnik dostrzeże tutaj pozorną niekosekwencję. Podczas definiowana tabeli TranNagl wystąpiła analogiczna sytuacja - w nagłówku transakcji pojawił się identyfikator kontrahenta, lecz nie było w nim już innych pól związanych z kontrahentem. W tym przypadku jednak, oprócz identyfikatora występuje również nazwa, symbol KWiU (Klasyfikacja Wyrobów i Usług) oraz jednostka miary. Zgadza się , w tej tabeli pojawia się pewna nadmiarowość, jest ona jednak zamierzona .Dzięki niej użytkownik będzie mógł umieszczać na fakturze pozycje pobrane z tabeli TowUslug oraz wpisywać inne pozycje (dotyczy to zwłaszcza usług), które nie zostały zdefiniowane w tej tabeli. Zwróćmy również uwagę na fakt ,że w powyższej definicji brakuje takich pól jak cena brutto oraz kwota podatku VAT. Nie musimy zapamiętywać wartości tych pól w tabeli, ponieważ można je w każdej chwili wyliczyć. Dzięki temu unikniemy przechowywania nadmiarowych danych w tabeli. Brak wspomnianych pól ułatwia również zachowanie integralności danych - wyobraźmy sobie sytuację ,w której istnieją pola Cena_jedn, St_VAT, Ilosc i Cena_brutto. Na skutek błędu w programie mogłoby dojść do sytuacji , że cena brutto nie zostanie zaktualizowana wraz ze zmianą jednej z trzech poprzednich wartości - innymi słowy doszłoby do utraty integralności danych w bazie. Likwidowanie nadmiarowości danych nazywane jest normalizacją bazy danych. Jest to dosyć szerokie zagadnienie. Jednak, każdy "szanujący się" projektant baz danych powinien się z nim zapoznać i potrafić doprowadzić bazę danych przynajmniej do trzeciej postaci normalnej (3NF) .Czasami się zdarza ,że celowo unikamy normalizacji bazy danych a nawet denoromalizujemy ją (z postaci znormalizowanej wracamy do postaci nieznormalizowanej). Ten odwrotny proces stosuje się ,gdy bardziej niż na jakości struktur danych zależy nam na czasie dostępu do nich - niestety im bardziej znormalizowana jest baza, tym więcej czasu zabiera przetwarzania danych, gdyż w zapytaniach trzeba wykonywać wiele operacji złączeń i obliczeń. Prostszym od normalizacji sposobem uzyskiwania poprawnych struktur baz danych jest modelowanie jednostka - związek (entity - relation), nazywane również modelowaniem ER

Tabela stawek podatku VAT

Z poprzedniego podrozdziału wynika ,że wystarczy w tabeli TranSzcz przechowywać odpowiedni procent podatku , a następnie używać go do obliczania kwoty podatku oraz wartości brutto. Stwierdzenie to byłoby prawdziwe, gdyby wszystkie stawki podatku VAT były wyrażone liczbowo. Niestety istniej kilka wariantów stawek zerowych , rozliczanych na podstawie różnych kryteriów. Nie będziemy się wgłębiać w rachunkowość, wystarczy ,że zauważymy następujący problem - jeżeli stawka podatku jest wyrażona liczbowo, wystarczy dokonać konwersji tej liczby (pamiętajmy ,że pole St_VAT z tabeli TransSzcz jest typu tekstowego) i przeprowadzić stosowne obliczenia. Z drugiej strony, jeżeli stawki podatku nie można konwertować na liczbę, wówczas jako procent należy przyjąć 0. Zaznaczam , że problem ten dałoby się rozwiązać bez tworzenia dodatkowej tabeli, jednak z przyczyn opisanych w dalszych rozdziałach, zaproponowane tutaj rozwiązanie będzie dla nas najwygodniejsze.

1.Zdefiniuj tabelę St_vat, zgodnie z poniższym opisem:

Nazwa pola : IDSt_VAT

Typ : A

Rozmiar : 2

Klucz : *

Nazwa pola : Pozycja

Typ : S

Rozmiar :

Klucz :

Nazwa pola : Procent

Typ : N

Rozmiar :

Klucz :

Nazwa pola : Opis

Typ : A

Klucz :

2.Pozostając w programie Database Desktop, otwórz tabelę St_vat (polecenie File, Open, Table)

3.Naciśnij klawisz [F9], aby przejść do trybu edycji

4.Wypełnij tabelę :

St_vat : 1,2,3,4

IDst_VAT : 0 ,22, 7 ,zw

Pozycja : 0 ,23,7 -1

Procent : 0,00 ; 23,00 ; 7,00 ; 0,00

Opis : stawka 0% , stawka 23%, stawka 7% , zwolniona

Zwróć uwagę na kolumnę Pozycja. Określa ona kolejność ,w jakiej będą występowały poszczególne stawki na różnego rodzaju zestawieniach i wydrukach. Oczywiście można by tutaj zastosować ciągłą numeracją np. 1,2,3,4 ale przyjęte rozwiązanie ułatwia nam dodanie ewentualnych nowych stawek - gdyby zaszła potrzeba wstawienia stawki np. 12%, otrzyma ona pozycję 12 i nie trzeba będzie zmieniać kolejności w pozostałych rekordach .Przypisanie wartości -1 stawce "zw" oznacza ,że będzie ona drukowana jako pierwsza (przed pozycją 0, która jest przypisywana stawce 0%). Podsumowując, dzięki zastosowaniu dodatkowej tabeli uzyskamy następujące korzyści:

-Nie trzeba będzie tworzyć skomplikowanych zapytań SQL , konwertujących ciągi znaków na liczby

-Podczas wypełniania faktury użytkownik będzie mógł wprowadzić tylko jedną ze zdefiniowanych stawek podatku (w tym celu trzeba będzie zdefiniować relację)

-Będziemy mieli kontrolę nad kolejnością poszczególnych stawek VATY w wydrukach

Ujemnymi stronami przyjętego rozwiązania jest nieznaczne zwiększenie złożoności programu i konieczność stosowania dodatkowej operacji złączenia w zapytaniach operujących na stawkach podatku VAT.

Określanie zależności między tabelami

W tej chwili nasza baza danych składa się z pięciu tabel : Kontrah, TowUslug, TranNagl i St_vat. Trudną ją jednak nazwać relacyjną, ponieważ zdefiniowane tabele nie są ze sobą jeszcze w żaden sposób powiązane. W tym podrozdziale wyjaśnimy , na czym polega definiowanie relacji i jakie to ma praktyczne znaczenie dla funkcjonowania baz danych oraz zarządzających nimi aplikacji. Istnieją następujące typy relacji : jeden - do - jednego, jeden - do - wielu i wiele - do - wielu . Najczęściej występuje drugi typ. Relacja jeden - do -wielu oznacza ,że z jednym rekordem w tabeli nadrzędnej jest powiązanych wiele rekordów w tabeli podrzędnej .Spróbujmy to wyjaśnić na przykładzie tabel TranNagl i Kontrah. Danemu kontrahentowi może być przypisanych wiele nagłówków transakcji (faktury), natomiast każdemu nagłówkowi transakcji jest przypisany tylko jeden kontrahent. Mamy więc do czynienia z relacją typu jeden - do - wielu . Zależność taką często przedstawia się w formie graficznej. Identyfikator kontrahenta (pole IDKontrah) w tabeli TranNagl nazywany jest kluczem obcym lub kluczem zewnętrznym (foreign key), ponieważ pochodzi z innej, zewnętrznej tabeli. Jednym z najważniejszych zadań systemu zadań systemu zarządzania relacyjnymi bazy danych (SZRBD) jest kontrola poprawności powiązań między tabelami, czyli dbanie o zachowanie integralności referencyjnej .W naszym przykładzie możemy powiedzieć , że baza danych zachowuje integralność referencyjną, jeżeli dla każdego pola IDKontrah w tabeli TranNagl istnieje odpowiedni powiązany rekord w tabeli Kontrah, innymi słowy - dla każdego identyfikatora kontrahenta w tabeli nagłówków transakcji istnieje odpowiedni zapis w tabeli kontrahentów. Łatwo się domyślić,co stałoby się , gdyby ten warunek nie był spełniony - najlepszym razie zostałaby wydrukowana faktura bez odbiorcy. Ponieważ wiemy do czego zmierzamy, pozostało nam zakodować potrzebne informacje w strukturze bazy danych .Do tego celu użyjemy oczywiście programu narzędziowego Database Desktop

1.W menu Tools, Utilities wybierz polecenie Restructure

2.W oknie dialogowym SelectFile zaznacz tabelę TranNagl, a następnie kliknij przycisk Otwórz. Pojawi się okno dialogowe Restructure Table, przedstawiające strukturę tabeli TranNagl

3.Na liście Table Properties z prawej strony okna dialogowego wybierz opcję Referential Integrity. Zawartość okna pod listą zmieni się.

4.Kliknij przycisk Define. Pojawi się okno dialogowego Referential Integrity . Z lewej strony tego okna znajduje się lista pól tabeli podrzędnej (TranNagl), z prawej widoczne są wszystkie zdefiniowane nazwy tabel

5.Na liście Fields zaznacz pole IDKontrah i kliknij przycisk strzałki w prawo .Wskazane pole znajdzie się na liście Child fields

6.Na liście Tables zaznacz plik tabeli Kontrah i klkinij przyciski strzałki w lewo. Na liście Parent′s key zostanie umieszczony indeks podstawowy tabeli Kontrah. Jak pamiętamy , został on utworzony w oparciu o pole IDKontrah. W grupie opcji Update rule mamy do wyboru dwie możliwości : Cascade oraz Prohibit. Gdy zaznaczona jest pierwsza , wszystkie zmiany dokonane w indeksie głównym tabeli nadrzędnej będą kaskadowo (stąd nazwa) wprowadzane w tabelach podrzędnych. W naszym przykładzie oznacza to ,że jeżeli użytkownik zmieni identyfikator kontrahenta, aparat bazy danych spróbuje zmienić wszystkie identyfikatory tego kontrahenta występujące w rekordach tabeli TranNagl. Aby przeprowadzić ten proces, tabela podrzędna (tutaj TranNagl) jest blokowana na wyłączność - w trakcie modyfikacji kaskadowej ,żaden inny użytkownik nie będzie mógł wprowadzić zmian w tabeli. Z drugiej strony, jeżeli jakiś użytkownik spróbuje zmienić identyfikator klienta w tabeli Kontrah, a równocześnie zawartość tabeli TranNagl jest edytowana

przez kogoś innego, próba utworzenia blokady wyłączonej tabeli nie powiedzie się, co z kolie uniemożliwi dokonanie zmiany identyfikatora. Druga opcja, Prohibit, po prostu uniemożliwia zmianę klucza podstawowego (indeksu głównego w tabeli nadrzędnej) ,jeżeli w tabeli podrzędnej istnieją jakieś rekordy powiązane z tym kluczem. Wracając do naszego przykładu - gdy ta opcja będzie zaznaczona, nie będzie można usunąć ani zmienić identyfikatora kontrahenta w tabeli Kontrah, jeżeli istnieją już dla niego jakieś transakcje. Jak widać , pierwsza opcja zapewnia większą elastyczność i należy ją stosować jeżeli to tylko możliwe (ograniczenia stosowania tej opcji występuje w przypadku niektórych formatów przechowywania danych, innych niż Paradox) .Zaznaczona domyślna opcja Strict referential integrity powoduje ,że tabela będzie chroniona przed uszkodzeniem przez wcześniejsze wersje programu Database Desktop. Niestety sprawia to , że tabela ta stanie się niedostępna z poziomu bazy Paradox dla systemu DOS. Oczywiście ma to znaczenie tylko wtedy jeżeli przewidujesz wykorzystanie tabel w aplikacjach starszego typu

7.Kliknij przycisk OK. Pojawi się okno dialogowe Save Referential Integrity As

8.Wpisz KontrahRI i kliknij OK. Ponownie aktywne stanie się okno dialogowe Restructure Table. Na liście z prawej strony okna będzie widoczny zdefiniowany przed chwilą warunek integralności. W ten sposób zdefiniowaliśmy reguły integralności referencyjnej między tabelami TrangNagl i Kontrah. Kluczem obcym w warunku integralności w tabelach typu Paradox może być tylko indeks podstawowy

W analogiczny sposób zdefiniujemy teraz warunki integralności między tabelami TranNagl i TranSzcz

1.W menu Tools, Utilities wybierz polecenie Restructure

2.W oknie dialogowym SelectFile zaznacz plik TranSzcz i kliknij przycisk Otwórz. Pojawi się okno dialogowe Restructure Table przedstawiające strukturę tabeli TranSzcz

3.Na liście rozwijanej Table properties zaznacz opcję Referential Integrity

4.Kliknij przycisk Define .Pojawi się okno Referential Integrity

5.Na liście z lewej strony dwukrotnie kliknij pole IDTranNagl. Na liście z prawej strony dwukrotnie kliknij tabelę TranNagl

6.Upewnij się ,że zaznaczone są opcje Cascade i Strict referential integrity, a następnie kliknij przycisk OK

7.W oknie dialogowym Save Referentia;l Integrity As wpisz TranNaglRI i kliknij przycisk OK

Kolejny warunek integralności zdefiniujemy dla pola IDTowUslug, będącego identyfikatorem towaru lub usługi w tabeli szczegółów transakcji

1.Ponownie kliknij Define

2.W oknie dialogowym Referential Integrity z listy po lewej stronie wybierz pole IDTowUSlug, natomiast z listy po prawej wybierz TowUslug. Dzięki tej relacji , użytkownik nie będzie mógł wprowadzić w pozycji transakcji identyfikator towaru, który nie został zdefiniowany w tabeli TowUslug

3.Upewnij się ,że zaznaczone są opcje Cascade i Strict referential integrity, a następnie kliknij przycisk OK

4.W oknie dialogowym Save Referential Integrity As wpisz TowUSlugRI i kliknij przycisk OK

Ostatni warunek integralności w tej tabeli będzie dotyczył pola ze stawką podatku VAT. Będziesz postępować jak w poprzednim ćwiczeniu.

1.Ponownie kliknij przycisk Define

2.W oknie dialogowym Referential Integrity z listy po lewej stronie wybierz pols St-VAT, natomiast z listy po prawej wybierz tabelę St_vat. Dzięki tej relacji, użytkownik nie będzie mógł wprowadzić w pozycji transakcji stawki podatku VAT, która nie została zdefiniowana w tabeli St_vat

3.Upewnij się ,że zaznaczone są opcje Cascade i Strict referential integrity , a następnie kliknij przyciks OK

4.W oknie dialogowym Save Referential Integrity As wpisz St_VATRI i kliknij przycisk OK.

5.W oknie dialogowym Restructure Table kliknij przycisk Save

W tej chwili wszystkie niezbędne relacje między tabelami naszej bazy danych są już zdefiniowane. Warto tutaj nadmienić ,że wspomniane zależność można definiować również w samym programie klienckim, który w tym przypadku będzie aplikacja do wystawiania faktur. Oznacza to ,że w praktyce etap definiowana relacji na poziomie aparatu bazy danych można by w ogóle pominąć. Takie postępowanie jest jednak niewskazane. Projektując bazy danych, wszystkie zależności i ograniczenia należy definiować na możliwie jak najniższym poziomie .Sprzyja to zwiększeniu niezawodności systemu i odciąża aplikacje użytkowe - relacje między tabelami są definiowane w aparacie bazy danych tylko raz

Tabela pomocnicza Liczniki

W tabeli Liczniki będą zapisywane informacje o ostatnich wykorzystanych numerach faktur. Ponieważ dla każdego typu transakcji, serii i dopisku mogą być prowadzone oddzielne numeracje, oprócz ostatniego numeru faktury w tej tabeli musimy również przechowywać informacje o wartościach z pól Typ, Seria i Dopisek z tabeli TranNagl

Zdefiniuj tabelę Liczniki zgodnie z poniższym opisem:

Nazwa pola : Licznik

Typ : A

Rozmiar : 40

Klucz : *

Nazwa pola : OstatniNumer

Typ : I

Rozmiar :

Klucz :

Tabela ta ma tylko dwa pola. W pierwszym polu będzie przechowywana nazwa licznika (np. nazwa serii faktury),natomiast w drugim jego wartość (np. ostatni numer faktury dla określonej serii).

Podsumowanie

W tej części zaprojektowałeś kompletną strukturę bazy danych. W ramach tego procesu utworzyłeś sześć tabel : Kontrah, Liczniki, St_vat, TowUslug, TranNagl i TranSzcz. Ponadto zdefiniowałeś warunki

integralności, zapewniając zachowanie prawidłowych powiązań między tabelami. Gdy baza danych jest już zdefiniowana, możemy przystąpić do tworzenia szkieletu aplikacji użytkowej, za pomocą której będziemy wprowadzać i edytować dane oraz sporządzać wydruki (…)