

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Oracle 10g i Delphi. Programowanie baz danych

Autorzy: Artur Mościcki, Igor Kruk

ISBN: 83-246-0272-0

Format: B5, stron: 392



Oracle 10g to kolejna wersja chyba najbardziej znanej bazy danych. Jej stabilność i wydajność to już niemal legenda. W oparciu o Oracle tworzone są setki aplikacji bazodanowych operujących na milionach rekordów. Tego typu aplikacje często muszą być przygotowane w krótkim czasie. Do takich właśnie zastosowań doskonale nadaje się Delphi – środowisko programistyczne umożliwiające szybkie tworzenie aplikacji. Dzięki zaangażowaniu firm trzecich powstały komponenty, moduły i biblioteki pozwalające na współpracę Oracle i Delphi oraz sprawne budowanie aplikacji bazodanowych łączących te platformy.

Książka „Oracle 10g i Delphi. Programowanie baz danych” to podręcznik dla programistów zamierzających tworzyć takie właśnie aplikacje. Przedstawia podstawy działania i obsługi Oracle 10g oraz metody wykorzystywania tej bazy danych w aplikacjach pisanych w Delphi. Opisuje różne technologie dostępu do danych i komponenty pozwalające na realizację zadań związanych z obsługą bazy. Pokazuje również sposoby generowania raportów w Rave Reports z wykorzystaniem danych pochodzących z Oracle 10g.

- Instalacja Oracle 10g
- Korzystanie z narzędzi Enterprise Manager i SQL*Plus
- Projektowanie bazy danych
- Komunikacja z bazą danych za pomocą BDE
- Stosowanie komponentów ADO
- Korzystanie z dbExpress i ODAC
- Tworzenie raportów za pomocą Rave Reports
- Analiza danych

Poznaj nowoczesne metody budowania aplikacji dla Oracle



Spis treści

Wstęp	9
Część I Baza danych Oracle 10g	11
Dlaczego Oracle 10g?	13
Rozdział 1. Instalacja i konfiguracja środowiska Oracle 10g	15
Wymagania systemowe	15
Instalacja Oracle 10g	16
Rozdział 2. Pakiet Oracle 10g Companion	27
Wstęp	27
Instalacja pakietu Oracle 10g Companion	27
Konfiguracja DAD	33
Rozdział 3. Środowisko pracy użytkownika bazy danych Oracle 10g	35
Enterprise Manager	35
Uruchamianie	35
Nagłówek	41
Home	45
Performance	49
Administration	62
Maintenance	79
iSQL*Plus	83
Uruchamianie	84
Preferences	87
History	92
Workspace	93
Uprawnienia SYSDBA i SYSOPER	99
Oracle HTTP Server	105
Uruchamianie	106
Własna strona HTML	109
Rozdział 4. Przykładowa baza danych	111
Wstęp	111
Tworzenie przestrzeni tabel	111
Tworzenie schematu użytkownika	116
Profile użytkownika	116
Konta użytkowników	118
Role i uprawnienia	120
Projekt bazy danych	125

Tabele i relacje	125
Sekwencje	130
Rozdział 5. Tworzenie interfejsu użytkownika w środowisku Oracle 10g	135
Wstęp	135
Pakiety	136
Formularze	139
Część II Technologie dostępu do Oracle 10g	153
Rozdział 6. BDE	155
Wstęp	155
Tworzenie połączenia z bazą danych	156
Table	159
Specjalne cechy komponentu Table	161
Metoda Locate	162
Query	164
Zapytania z parametrami	167
Edycja danych	169
Stany zbioru danych	175
Field	176
Dodawanie atrybutów obliczeniowych	179
Tworzenie pól przeglądania	181
Database Form Wizard	183
Używanie SQL Monitora	187
Rozdział 7. ODBC	191
Wstęp	191
Mechanizm ODBC	191
Rozdział 8. ADO	197
Wstęp	197
ADOConnection	198
Data Link	202
Uzyskiwanie informacji o strukturze bazy danych	203
Dynamiczne właściwości komponentów ADO	206
ADOCmdnd	207
ADOTable	211
ADOQuery	212
ADODataset	215
Położenie i typy kursorów	219
Typy kursorów a metoda RecordCount	221
Typy blokad	221
Indeksy klienta	222
Klonowanie	223
Mechanizm transakcji	224
Transakcje zagnieżdżone	226
XML — eksport i import danych	226
Pliki Advanced Data TableGram (ADTG)	229
Zbiory rekordów odłączonych	229
Pooling połączeń	230
Model aktówki	231
Zakładki	231

Rozdział 9. dbExpress	235
Wstęp	235
Tworzenie połączenia z bazą danych	236
SQLQuery	237
Edycja danych	241
Monitorowanie bazy danych	251
Pozostałe komponenty zbioru danych	252
Obsługa transakcji	252
ClientDataSet i XML	254
Informacje o bazie danych	257
Metoda SetSchemaInfo	258
Rozdział 10. ODAC	259
Wstęp	259
Instalacja	259
Komponenty ODAC	261
Tworzenie połączenia z bazą danych	262
OraQuery	263
Atrybuty typu LOB	265
Typ LOB	265
Obsługa atrybutów typu BLOB za pomocą ODAC	266
ODAC i IntraWeb	272
Co to jest IntraWeb?	272
Tworzenie projektu	272
Uruchamianie projektu	273
Tworzenie aplikacji	273
OraSQL	279
Rozdział 11. Pozostałe komponenty bazodanowe	283
Wstęp	283
DBNavigator	283
Tekstowe kontrolki danych	284
Kontrolki danych dla list	284
DBCcheckBox	284
Zastosowanie kontrolek danych	285
Tworzenie niestandardowej siatki	286
Kolorowanie DBGrid	286
Dopasowanie szerokości kolumn	289
Siatka z polem wyboru	290
Siatka pozwalająca na zaznaczanie wielu rekordów	293
Siatka z polem Memo	294
Siatka wyświetlająca obrazki	295
Kontrolki przeglądania	297
Graficzne kontrolki danych	299
DBCtrlGrid	299
Standardowe kontrolki w aplikacjach bazodanowych	301
Część III Rave Reports — raportowanie baz danych	303
Dlaczego Rave Reports?	305
Rozdział 12. Raportowanie za pomocą kodu	307
Wstęp	307
Komponent RvSystem	307
Pierwszy raport	307

Raport kolumnowy	312
Umieszczanie grafiki w raporcie	316
Raportowanie baz danych za pomocą kodu	320
Umieszczanie atrybutów typu BLOB w raportach	325
Umieszczanie wykresów w raporcie	329
Metody rysowania w raportach generowanych za pomocą kodu	332
Rozdział 13. Podstawy Visual Designera	337
Wstęp	337
Wprowadzenie do Rave Visual Designera	337
Pierwszy raport	339
Komponent RvProject	341
Interakcja z aplikacją	342
Ustawienia globalne (Global Pages)	345
Zmienne po inicjalizacji (Post-Initialize Variables)	348
Druk warunkowy (Conditional Printing)	350
Rozdział 14. Inteligentna analiza danych w Rave Reports	353
Wstęp	353
Driver Data View	353
Pasma i regiony	356
Dodawanie pól	357
Dodawanie raportu do projektu	357
Pola obliczeniowe w raportach	359
Direct Data View	360
Projektowanie raportu typu Master-Detail	360
Rozdział 15. Pozostałe komponenty z palety Rave	365
Wstęp	365
RvNDRWriter i RvRenderPreview	365
RvRenderPrinter	367
Zapisywanie raportu w różnych formatach	367
Dodatki	371
Skorowidz	373

Rozdział 6.

BDE

Wstęp

BDE (*Borland DataBase Engine*) to technologia, która została stworzona jako alternatywa dla technologii ODBC. Powstała przy współpracy kilku firm, m.in. Borland i Oracle. Kiedyś była to podstawowa technologia dostępu do baz danych, obecnie jest wypierana przez inne, takie jak ADO czy dbExpress. Zaletą BDE jest większa w porównaniu z ODBC szybkość działania oraz prostota obsługi, wadą zaś — przenośność. Wynika to z tego, że BDE nie obsługuje systemu zarządzania baz danych bezpośrednio, lecz robi to poprzez pośrednika, jakim jest program SQL Links. Obecnie Borland rezygnuje z rozwijania tej technologii, niemniej można ją wykorzystać nawet do łączenia się z bazami danych stworzonymi w Oracle 10g. Przyjrzyjmy się paletce komponentów BDE dostępnych w Delphi (rysunek 6.1).

Rysunek 6.1.
*Paleta komponentów
BDE*



Na zakładce BDE znajduje się kilka komponentów:

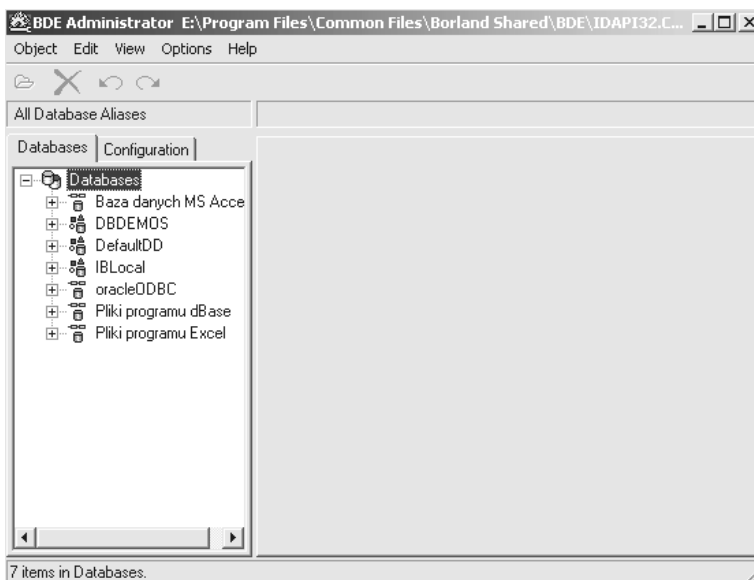
- ◆ Table — umożliwia pracę z tabelami bazy danych;
- ◆ Query — umożliwia wykonywanie zapytań SQL;
- ◆ StoredProc — umożliwia wykonywanie procedur składowanych na serwerze baz danych;
- ◆ Database — umożliwia zdefiniowanie połączenia z bazą danych;
- ◆ Session — służy do globalnej kontroli komponentów Database. Domyślnie tworzony jest automatycznie (właściwość `SessionName` komponentu Database ustawiana jest na `Default`). Jeśli jednak użytkownik tworzy aplikacje wielowątkową, to powinien dla każdego wątku używać oddzielnego komponentu;

- ♦ BatchMove — służy do kopiowania zbioru danych. Może być używany do konwersji tabel na różne formaty baz danych lub do tworzenia zbioru rekordów odłączonych;
- ♦ UpdateSQL — pozwala na wykonywanie instrukcji SQL (UPDATE). Buforuje zbiór danych;
- ♦ NestedTable — umożliwia hermetyzowanie zbioru danych zagnieżdżonego w innej tabeli.

Tworzenie połączenia z bazą danych

Aby stworzyć nowe połączenie z bazą danych wykorzystując technologię BDE, należy użyć programu *BDE Administrator*. Program ten można uruchomić poza Delphi 7 poprzez *Start/Programy/Delphi 7/BDE Administrator*. Ukaże się wówczas okno programu (rysunek 6.2), na którym będą widoczne dwie zakładki *DataBase* i *Configuration*. Na zakładce *Configuration* można zobaczyć, jakie sterowniki SQL Links (gałąź *Drivers/Native*) i ODBC (gałąź *Drivers/ODBC*) są dostępne.

Rysunek 6.2.
BDE Administrator



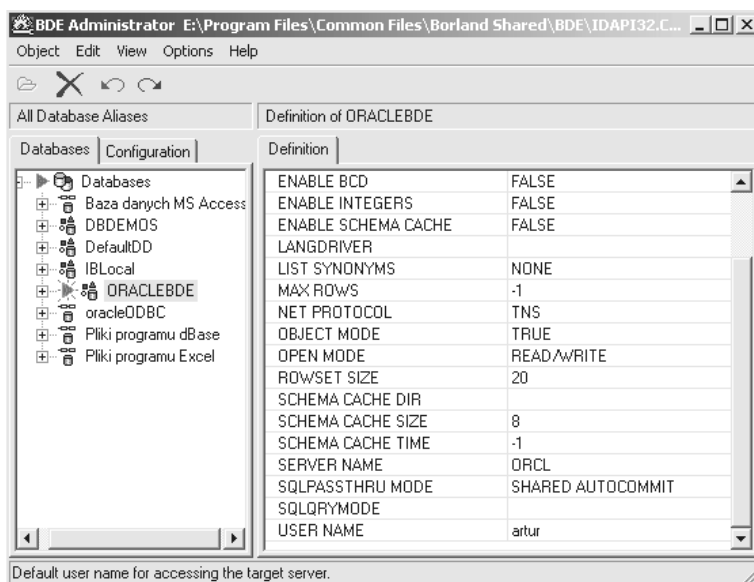
Przejdź na zakładkę *Database* i wybierz z menu głównego *Object/New*. Pojawi się nowe okienko *New Database Alias* (rysunek 6.3).

Rysunek 6.3.
Okno New Database Alias



W oknie tym wybierz sterownik do Oracle i kliknij przycisk *OK*. Pojawi się ponownie okno *BDE Administrator*, w którym do zakładki *Database* dodana zostanie nowa pozycja *ORACLE1*. Zmień jej nazwę na *ORACLEBDE* (rysunek 6.4).

Rysunek 6.4.
Okno *BDE Administrator* po dodaniu nowej pozycji *ORACLEBDE*



Aby stworzyć nowy sterownik BDE, musisz jeszcze zmienić dwie właściwości znajdujące się po prawej stronie okna w *BDE Administratorze*:

- ♦ w polu *Server Name* wpisz *orcl*;
- ♦ w polu *User Name* wpisz nazwę użytkownika bazy danych.

Na koniec naciśnij przycisk *Apply* (niebieska strzałka) w celu zapisania nowego połączenia BDE.

Przetestujmy teraz stworzone przed chwilą połączenia BDE.

1. Otwórz Delphi i dodaj do formularza komponent *Database1* z palety BDE.
2. Ustaw właściwość *AliasName* komponentu *Database1* na stworzone przed momentem połączenie *ORACLEBDE*.
3. We właściwości *DatabaseName* komponentu *Database1* wpisz nazwę bazy danych *orcl*.
4. Zmień właściwość *Connected* komponentu *Database1* na *True*. Pojawi się okno, w którym będziesz musiał podać hasło dostępu do bazy danych. Jeśli je podasz i właściwość *Connected* zostanie zmieniona na *True*, oznacza to, że połączenie się powiodło.

Przyjrzyjmy się jeszcze właściwościom i zdarzeniom komponentu *Database* (tabela 6.1 i 6.2).

Tabela 6.1. Właściwości komponentu Database

Właściwość	Opis
AliasName	Specyfikuje alias BDE wykorzystywany przez ten komponent.
Connected	Specyfikuje, czy połączenie z bazą danych jest aktywne. Ustawienie właściwości na True uaktywnia połączenie, natomiast na False powoduje odłączenie od bazy danych.
DatabaseName	Specyfikuje nazwę bazy danych związaną z komponentem Database.
DriverName	Specyfikuje nazwę sterownika BDE wykorzystywanego przez DataBase.
Exclusive	Specyfikuje, czy dana aplikacja ma wyłączność na połączenie z bazą danych (wartość domyślna False).
HandleShared	Specyfikuje, czy uchwyt do bazy danych może być współdzielony (wartość domyślna False).
KeepConnection	Specyfikuje, czy aplikacja utrzymuje połączenie z bazą danych, nawet jeśli nie jest otwarty żaden zbiór danych.
LoginPrompt	Specyfikuje, czy przed otwarciem nowego połączenia będzie wyświetlane okno dialogowe służące do logowania (wartość domyślna True).
Name	Specyfikuje nazwę komponentu.
Params	Specyfikuje zestaw parametrów dla połączenia z bazą danych, np. nazwę użytkownika i hasło.
ReadOnly	Specyfikuje, czy połączenie z bazą danych umożliwia przeprowadzenie tylko operacji odczytu (wartość domyślna False).
SessionName	Specyfikuje nazwę sesji wykorzystywanej przez ten komponent.
Tag	Dodatkowa właściwość komponentu. Pozwala na wprowadzenie dowolnej wartości. Może zastępować zmienną w kodzie. Właściwość ta znajduje się w większości komponentów.
TransIsolatio	Specyfikuje poziom izolacji transakcji zarządzanych przez BDE.

Tabela 6.2. Zdarzenia udostępniane przez komponent Database

Zdarzenie	Opis
AfterConnect	Zdarzenie obsługiwane po połączeniu się aplikacji z bazą danych.
AfterDisconnect	Zdarzenie obsługiwane po rozłączeniu się z bazą danych.
BeforeConnect	Zdarzenie obsługiwane tuż przed połączeniem się aplikacji z bazą danych.
BeforeDisconnect	Zdarzenie obsługiwane tuż przed rozłączeniem się aplikacji z bazą danych.
OnLogin	Zdarzenie obsługiwane w momencie podłączania się do bazy danych. W parametrze LoginParams przekazywane są nazwa użytkownika (<i>user_name</i>) i hasło (<i>password</i>), pobrane z właściwości Params komponentu. W zdarzeniu tym można umieścić obsługę własnego okna logowania (wtedy właściwość LoginPrompt musi być ustawiona na True).

Table

Najprostszym sposobem dostępu do danych przechowywanych w bazie danych *orcl* jest użycie komponentu *Table*. Komponent ten oznacza po prostu tabelę bazy danych. Przyjrzyjmy się najpierw właściwościom udostępnianym przez komponent (tabela 6.3). Zdarzenia komponentu *Table* są bardzo podobne do udostępnianych przez komponent *Query* i zostaną omówione w jednym z następnych podrozdziałów.

Tabela 6.3. Zdarzenia komponentu *Table*

Właściwość	Opis
Active	Specyfikuje, czy zbiór danych (<i>DataSet</i>) jest otwarty.
AutoCalcFields	Specyfikuje, kiedy zdarzenie <i>OnCalcFields</i> jest uruchamiane i kiedy wartości pól typu <i>Lookup</i> są obliczane.
AutoRefresh	Specyfikuje, czy wartości pól przechowywane na serwerze są automatycznie odświeżane (wartość domyślna <i>False</i>).
CachedUpdates	Specyfikuje, czy możliwe jest stosowanie trybu <i>Cached Updates</i> (<i>buforowane uaktualnienie</i>). Jeśli ten tryb pracy jest włączony (wartość ustawiona jest ustawiona na <i>True</i>), to wszystkie operacje wykonywane na zbiorze danych odbywają się w buforze po stronie aplikacji klienta. Po zakończeniu pracy wszystkie zmiany mogą zostać wysłane na serwer w ramach jednej transakcji. Wartość domyślna <i>False</i> .
Constraints	Opis więzów integralności na poziomie rekordu. Tutaj można utworzyć więzy bazujące na wartości kilku atrybutów. W razie wystąpienia potrzeby wykorzystania więzów na poziomie pojedynczej kolumny, należy je stworzyć w odpowiednim komponencie klasy <i>TField</i> .
DatabaseName	Specyfikuje nazwę bazy danych związaną z tym komponentem.
DefaultIndex	Specyfikuje, czy dane w tabeli mają być uporządkowane według domyślnego indeksu, czyli według klucza głównego (wartość domyślna <i>True</i>).
Exclusive	Specyfikuje, czy dana aplikacja ma wyłączność na połączenie z bazą danych (wartość domyślna <i>False</i>).
FieldDefs	Specyfikuje listę definicji atrybutów w zbiorze danych.
Filter	Właściwość używana w celu nałożenia filtru na zbiór danych.
Filtered	Specyfikuje, czy filtr jest aktywny. Wartość domyślna to <i>False</i> .
FilterOptions	Specyfikuje dodatkowe opcje związane z filtrami.
IndexDefs	Przechowuje informacje o definicjach indeksów tabeli w formie tablicy.
IndexFieldNames	Specyfikuje kolumny, które są używane jako indeks tabeli. Kolumny są rozdzielane średnikami. Ustawienie tej właściwości czyści właściwość <i>IndexName</i> .
IndexFiles	Pozwala na odczyt lub ustawienie pól wykorzystywanych w kluczu.
IndexName	Określa indeks wykorzystywany w tabeli. Jeśli właściwość nie jest zdefiniowana, to do określenia sortowania wykorzystywany jest klucz główny.
MasterFields	Specyfikuje jeden lub więcej atrybutów w tabeli nadrzędnej, które są połączone z atrybutami w danej tabeli, w celu ustalenia związku <i>master-detail</i> między tabelami. Właściwość tę należy ustawić po wypełnieniu właściwości <i>MasterSource</i> .
MasterSource	Specyfikuje nazwę zbioru danych, który będzie traktowany jako tabela nadrzędna dla danej tabeli.

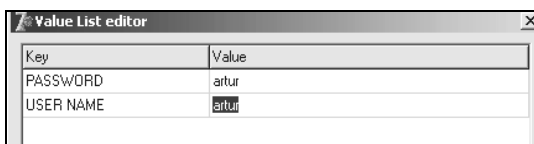
Tabela 6.3. Zdarzenia komponentu Table — ciąg dalszy

Właściwość	Opis
Name	Specyfikuje nazwę komponentu.
ObjectView	Specyfikuje, czy pola są przechowywane w sposób hierarchiczny czy sekwencyjny.
ReadOnly	Specyfikuje, czy tabela będzie dostępna w trybie tylko do odczytu (wartość domyślna False).
SessionName	Specyfikuje nazwę sesji wykorzystywanej przez ten komponent.
StoreDefs	Ustawienie tej właściwości na True powoduje, że definicje atrybutów (właściwość FieldDefs) i indeksów (właściwość IndexDefs), wypełnione przy projektowaniu aplikacji, mają zostać zapisane razem z modulem lub formularzem. Pozwoli to na tworzenie tabeli w bazie danych poprzez wywołanie metody CreateTable.
TableName	Specyfikuje nazwę tabeli w bazie danych, która będzie reprezentowana przez komponent Table.
TableType	Specyfikuje strukturę tabeli z bazy danych reprezentowanej przez dany komponent.
Tag	Dodatkowa właściwość komponentu. Pozwala na wprowadzenie dowolnej wartości. Może zastępować zmienną w kodzie. Właściwość ta znajduje się w większości komponentów w Delphi.
UpdateMode	Specyfikuje, w jaki sposób <i>BDE</i> odnajduje rekordy przy operacjach typu UPDATE.
UpdateObject	Specyfikuje komponent typu UpdateObject, pozwalający na aktualizację danych zwracanych jako tylko do odczytu lub w trybie <i>Cached Updates</i> .

Zastosujmy komponent Table w praktyce. Napişmy program, który wyświetli dane z tabeli Kategorie oraz pozwoli na ich modyfikację.

1. Umieść na formularzu komponenty Database1, Table1 z palety *BDE*, DataSource1 z karty *Data Access* i DBGrid1 z karty *Data Control*.
2. Ustaw właściwość AliasName komponentu Database1 na stworzone przed momentem połączenie *ORACLEBDE*.
3. We właściwości DatabaseName komponentu Database1 wpisz nazwę bazy danych orcl.
4. Kliknij dwa razy właściwość Params i w oknie *Value List Editor* (rysunek 6.5) wpisz dwa parametry: USER NAME i PASSWORD oraz odpowiadające im wartości (nazwę użytkownika i hasło). Parametry te będą potrzebne po to, by podczas próby łączenia się z bazą danych program nie pytał o login i hasło użytkownika.

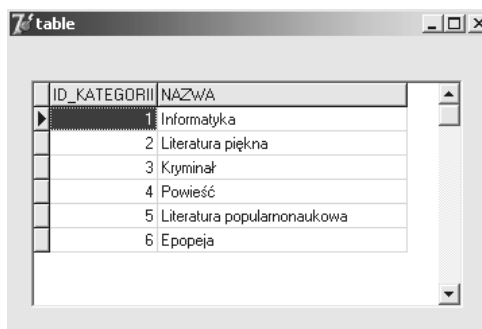
Rysunek 6.5.
Wpisane parametry
w oknie Value List
Editor



5. Zmień właściwość LoginPrompt komponentu Database1 na False. Od tego momentu nie będzie pojawiało się okno z prośbą o podanie loginu i hasła.
6. Zmień właściwość Connected komponentu Database1 na True.

7. Ustaw właściwość `DatabaseName` komponentu `Table1` na `orc1`.
8. Ustaw właściwość `TableName` komponentu `Table1` na tabelę `Kategorie`.
9. Ustaw właściwość `DataSet` komponentu `DataSource1` na `Table1`.
10. Ustaw właściwość `DataSource` komponentu `DBGrid1` na `DataSource1`. Dwa ostatnie kroki wykonuje się w celu wyświetlenia danych pobranych z bazy danych na komponencie `DBGrid1`.
11. Na koniec zmień właściwość `Active` komponentu `Table1` na `True`. Od tego momentu w komponencie `DBGrid1` wyświetlone będą dane z tabeli `Kategorie`. Możesz zapisać i uruchomić projekt (rysunek 6.6).

Rysunek 6.6.
Program podczas działania



ID_KATEGORII	NAZWA
1	Infomatyka
2	Literatura piękna
3	Kryminał
4	Powieść
5	Literatura popularnonaukowa
6	Epopeja

OK, program wyświetla dane. Ale jak dodać nowe wpisy lub modyfikować istniejące? Żeby zmodyfikować wpis, wystarczy ustawić kursor w odpowiednim polu siatki (po prostu kliknij myszą interesujące Cię pole) i poprawić istniejący tam wpis. Natomiast żeby dodać nowy wpis, trzeba ustawić kursor w dowolnym miejscu, a następnie za pomocą klawiszy strzałek przejść na koniec tabeli. Gdy pojawią się puste pola, należy wpisać w nie nowe wartości. Oczywiście w kolumnie `Id_kategorii` nie można wpisywać wartości, które już istnieją, ponieważ atrybut `Id_kategorii` jest kluczem głównym tabeli `Kategorie` i jego wartości muszą być niepowtarzalne. W kolumnie `Id_kategorii` można wpisywać tylko wartości numeryczne. W przeciwnym wypadku program zwróci komunikat o błędzie. W następnych podrozdziałach napiszemy, jak stworzyć program, w którym użytkownik nie będzie mógł ingerować w wartości atrybutu będącego kluczem głównym tabeli.

Specjalne cechy komponentu Table

Komponent `Table` posiada specjalne cechy, których nie mają inne komponenty zbioru danych, np. `Query`. Otóż w komponencie tym dostępne są specjalne metody wyszukiwania, takie jak: `FindKey`, `FindNearest`, `GotoKey`, `GotoNearest`. My przyjrzymy się bliżej dwóm pierwszym metodom. Odpowiedzialne są one za wyszukiwanie przybliżonych wartości (`FindNearest`) oraz za wyszukiwanie dokładne (`FindKey`). Metody te wyszukują tylko po atrybutach będących indeksami lokalnymi. Natomiast indeksy lokalne można założyć tylko na atrybutach będących kluczem głównym tabeli.

Zmodyfikujmy przykład z poprzedniego podrozdziału tak, aby można było wyszukiwać książki na podstawie ISBN.

1. Otwórz projekt z poprzedniego podrozdziału.
2. Zmień właściwość `TableName` komponentu `Table1` tak, aby wskazywała na tabelę `Ksiazki`.
3. Zmień właściwość `Active` komponentu `Table1` na `True` w celu wyświetlenia wszystkich rekordów.
4. Ustaw indeks na atrybucie `ISBN`. W tym celu ustaw właściwość `IndexFieldNames` komponentu `Table1` na `ISBN`.
5. Dodaj do formularza dwa komponenty `Edit` i dwa przyciski (`Button`).
6. Zmień tekst wyświetlany na przyciskach (właściwość `Caption`) na `Wyszukaj książki o podobnym ISBN` i `Wyszukaj książki o identycznym ISBN`.
7. Oprogramuj zdarzenia `onClick` przycisków:

8. procedure `TForm1.Button1Click(Sender: TObject);`

```
begin
    Table1.FindNearest([Edit1.Text]); //wyszukaj książki o podobnym ISBN
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    //wyszukaj książki o identycznym ISBN, a jeśli takich nie ma, to wyświetl
    komunikat
    if Table1.FindKey([Edit2.Text])=false then
        showmessage('Książka o takim ISBN nie istnieje w bazie danych');
end;
```

Parametrami obu metod są tablice stałych. Każdy element tablicy odpowiada indeksowanemu atrybutowi. W przykładzie jest jeden indeks na atrybucie `ISBN`, dlatego wyszukiwanie odbywa się tylko po tym jednym atrybucie.

Rysunek 6.7 pokazuje program podczas działania. Po wpisaniu w polach edycyjnych wartości takich, jak na rysunku, wskaźnik (czarny trójkąt) zostanie ustawiony na drugim rekordzie. Widać, że metoda `FindNearest` znajduje odpowiedni rekord, pomimo iż wartość `ISBN` różni się dwoma cyframi.

Metoda Locate

Metody `FindKey` i `FindNearest` szukały rekordy tylko po atrybucie, który był indeksem. Metoda `Locate` pozwala wyszukiwać rekordy według dowolnego atrybutu tabeli. Posiada ona trzy parametry:

- ♦ pierwszym jest nazwa lub nazwy atrybutów, według których będzie prowadzone wyszukiwanie;
- ♦ drugim jest tablica wartości poszukiwanych;

Rysunek 6.7.
 Program
 wykorzystujący
 metody *FindKey*
 i *FindNearest*



- ♦ trzecim są opcje metody *Locate*. Dostępne są dwie opcje:
 - ♦ *loCaseInsensitive* — atrybuty i wartości poszukiwane są porównywane bez uwzględnienia wielkości liter,
 - ♦ *loPartialKey* — jeśli podane parametry wyszukiwania spełniają przynajmniej w części wartości występujące w polach, to metoda zwraca wartość *True*.

Metoda *Locate* jako rezultat zwraca wartość typu *Boolean*. Jeśli znaleziony został rekord odpowiadający poszukiwanej wartości, to wynikiem działania metody jest wartość *True*. Zaprezentujemy wykorzystanie metody *Locate* na przykładzie.

1. Wykorzystaj przykład z podrozdziału „Table”. Zmień właściwość *TableName* komponentu *Table1* tak, aby wskazywała na tabelę *Książki*.
2. Zmień właściwość *Active* komponentu *Table1* na *True* w celu wyświetlenia wszystkich rekordów.
3. Dodaj do formularza komponenty *Edit1* i *Button1*. W kontrolce *Edit1* będziemy wpisywać tytuł poszukiwanej książki.
4. Zmień właściwość *Caption* komponentu *Button1* na *Szukaj książek o tytule*.
5. Oprogramuj zdarzenie *onClick* komponentu *Button1*.

```

procedure TForm1.Button1Click(Sender: TObject);
begin
    //jeśli wpisano tytuł w kontrolce Edit1
    if Edit1.text<>' then
        //spróbuj znaleźć książkę o podanym tytule
        if Table1.Locate('Tytuł',VarArrayOf([Edit1.Text]),[loPartialKey])=False
        then
            //jeśli nie znaleziono książki o podanym tytule, to wyświetl komunikat
            showMessage('Nie znaleziono książki zawierającej taki tytuł');
end;

```

Jeśli chcesz wyszukiwać rekordy według dwóch atrybutów, to musisz nieco zmodyfikować parametry metody *Locate*:

```
Table1.Locate('Tytuł;ISBN',VarArrayOf([Edit1.Text,Edit2.Text]),[loPartialKey]);
```