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

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 6.
BDE

Wstęp

BDE (*Borland DataBase Enigme*) 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 palecie komponentów BDE dostępnych w Delphi (rysunek 6.1).

Rysunek 6.1. *Paleta komponentów BDE*

Na zakładce BDE znajduje się kilka komponentów:

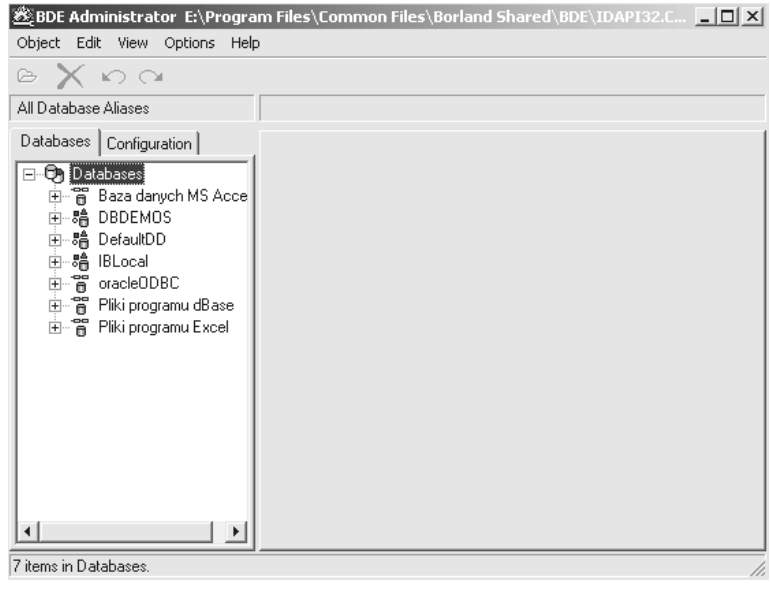
- `Table` — umożliwia pracę z tabelami bazy danych;
- `Query` — umożliwia wykonywanie zapytań SQL;
- `StoredProc` — umożliwia wykonywanie procedur składowanych na serwerze baz danych;
- `Database` — umożliwia zdefiniowanie połączenia z bazą danych;
- `Session` — służy do globalnej kontroli komponentów `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

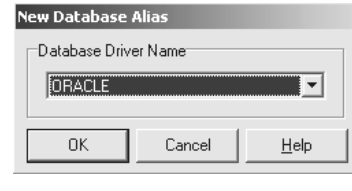
Aby stworzyć nowe połączenie z bazą danych wykorzystujące technologię BDE, należy użyć programu *BDE Administrator*. Program ten można uruchomić poza Delphi 7 poprzez *Start/Programy/Delphi 7/BDE Administrator*. Ukaże się wówczas okno programu (rysunek 6.2), na którym będą widoczne dwie zakładki *DataBase* 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*

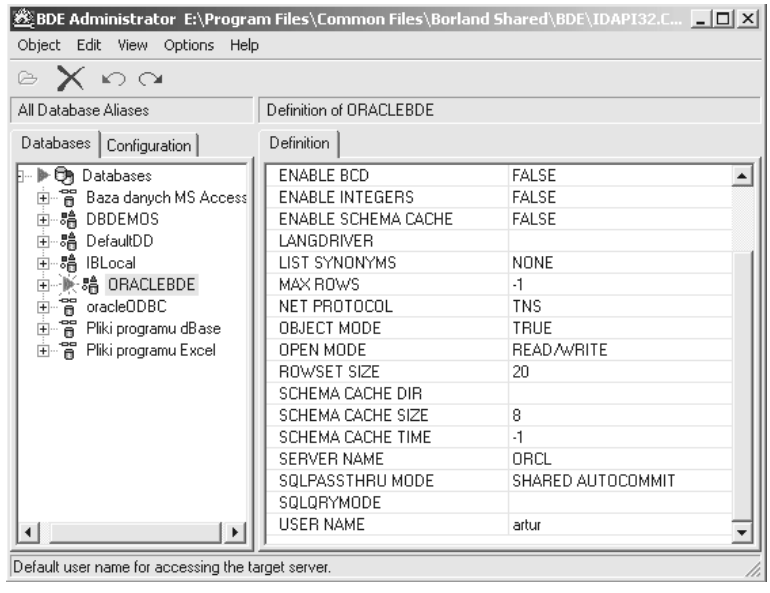


Przejdź na zakładkę *Database* i wybierz z menu głównego *Object/New*. Pojawi się nowe okienko *New Database Alias* (rysunek 6.3).

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` (tabele 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 (*user_name*) i hasło (*password*), 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 (*DataSet*) jest otwarty.
`AutoCalcFields`	Specyfikuje, kiedy zdarzenie `OnCalcFields` jest uruchamiane i kiedy wartości pól typu *Lookup* są obliczane.
`AutoRefresh`	Specyfikuje, czy wartości pól przechowywane na serwerze są automatycznie odświeżane (wartość domyślna `False`).
`CashedUpdates`	Specyfikuje, czy możliwe jest stosowanie trybu *Cashed Updates* (*buforowane uaktualnienie*). Jeśli ten tryb pracy jest włączony (wartość ustawiona jest ustawiona na `True`), 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 `False`.
`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 `TField`.
`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 `True`).
`Exclusive`	Specyfikuje, czy dana aplikacja ma wyłączność na połączenie z bazą danych (wartość domyślna `False`).
`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 `False`.
`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 `IndexName`.
`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 *master-detail* między tabelami. Właściwość tę należy ustawić po wypełnieniu właściwości `MasterSource`.
`MasterSource`	Specyfikuje nazwę zbioru danych, który będzie traktowany jako tabela nadrzędna dla danej tabeli.

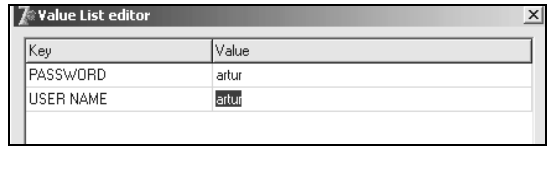
Tabela 6.3. *Zdarzenia komponentu Table — ciąg dalszy*

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 modułem 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 *BDE* odnajduje rekordy przy operacjach typu `UPDATE`.
`UpdateObject`	Specyfikuje komponent typu `UpdateObject`, pozwalający na aktualizację danych zwracanych jako tylko do odczytu lub w trybie *Cached Updates*.

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

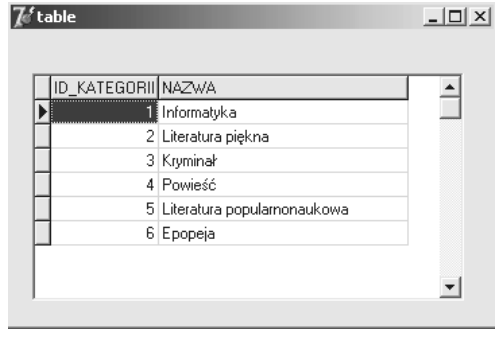
1. Umieść na formularzu komponenty `Database1`, `Table1` z palety *BDE*, *DataSource1* z karty *Data Access* i *DBGrid1* z karty *Data Control*.
2. Ustaw właściwość `AliasName` komponentu `Database1` na stworzone przed momentem połączenie *ORACLEBDE*.
3. We właściwości `DatabaseName` komponentu `Database1` 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 `orcl`.
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*

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;
```

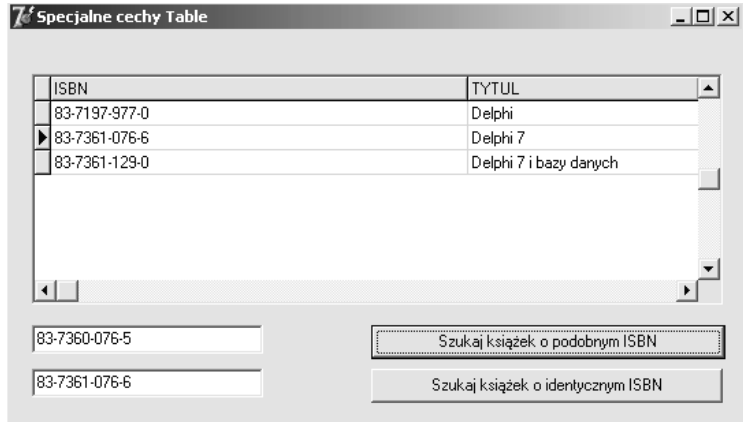
Parametrami obu metod są tablice stałych. Każdy element tablicy odpowiada indeksowanemu atrybutowi. W przykładzie jest jeden indeks na atrybucie `ISBN`, dlatego wyszukiwanie odbywa się tylko po tym jednym atrybucie.

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`. Zaprezentujmy wykorzystanie metody `Locate` na przykładzie.

1. Wykorzystaj przykład z podrozdziału „Table". Zmień właściwość `TableName` komponentu `Table1` tak, aby wskazywała na tabelę `Ksiazki`.
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('Tytul',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('Tytul;ISBN',VarArrayOf([Edit1.Text,Edit2.Text]),[loPartialKey]);
```