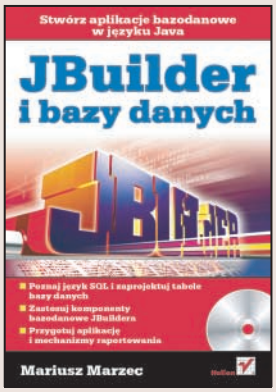
# JBuilder i bazy danych




Autor: Mariusz Marzec
ISBN: 83-7361-748-5
Format: B5, stron: 264

Informacja – najcenniejszy towar na współczesnym rynku, wymaga rozbudowanych narzędzi pozwalających na magazynowanie i korzystanie z niej. Do magazynowania informacji wykorzystywane są coraz bardziej rozbudowane systemy zarządzania bazami danych. Do przeglądania danych zgromadzonych w ich tabelach tworzone są mniej lub bardziej rozbudowane aplikacje. Dzięki nim informacje z baz danych przedstawiane są w formie możliwej do przeglądania, modyfikowania i drukowania.

Java – w początkowym okresie swojego rozwoju kojarzona była wyłącznie z Internetem i koszmarnymi apletami typu „padający śnieg", dziś jest uznanym i popularnym językiem programowania, coraz częściej wykorzystywanym do tworzenia aplikacji bazodanowych. Producenci środowisk programistycznych również zwrócili uwagę na popularność Javy, co zaowocowało powstaniem kilku bardzo udanych narzędzi, a wśród nich – JBuilder Enterprise.

„JBuilder i bazy danych" to podręcznik tworzenia aplikacji bazodanowych za pomocą środowiska JBuilder Enterprise. Przedstawia zarówno zagadnienia związane z bazami danych, jak i te, które wiążą się z wykorzystywaniem ich w aplikacjach w języku Java. Opisuje komponenty bazodanowe oferowane przez JBuildera oraz sposoby projektowania i implementowania aplikacji z ich wykorzystaniem. Zawiera również informacje dotyczące tworzenia mechanizmów raportujących i drukujących.

- Elementy języka SQL
- Tworzenie tabel baz danych
- Łączenie aplikacji z bazą danych
- Sterowniki JDBC
- SQL Server 2000
- Zastosowanie komponentów bazodanowych oraz komponentów graficznego interfejsu użytkownika w JBuilder
- Projektowanie aplikacji bazodanowej w JBuilder
- Raporty oraz drukowanie

**Przekonaj się, jak szybko i efektywnie możesz tworzyć aplikacje w środowisku JBuilder**

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

# Spis treści


**Wstęp ..... 7**

**Rozdział 1. Baza danych ..... 9**
Elementy baz danych ..... 10
Zasady projektowania baz danych ..... 12
Podsumowanie ..... 16

**Rozdział 2. Elementy języka SQL ..... 17**
Polecenia SQL ..... 19
SELECT ..... 19
INSERT ..... 22
UPDATE ..... 23
DELETE ..... 24
CREATE TABLE ..... 24
DROP TABLE ..... 27
ALTER TABLE ..... 28
CREATE VIEW ..... 29
DROP VIEW ..... 30
COMMIT ..... 30
ROLLBACK ..... 31
TRANSAKCJE ..... 31
WHERE ..... 31
TRIGGERY ..... 32
Procedury składowane (stored procedure) ..... 33
Funkcje agregujące ..... 34
Podsumowanie ..... 35

**Rozdział 3. Narzędzia wspomagające tworzenie i modyfikację baz danych ..... 37**
Przygotowanie projektu ..... 42
Tworzenie tablicy ..... 43
Tworzenie kluczy głównych ..... 45
Tworzenie powiązań między tablicami ..... 45
Tworzenie perspektyw ..... 47
Generowanie skryptów ..... 47
Generowanie dokumentacji projektu ..... 49
Podsumowanie ..... 50

**Rozdział 4. Połączenie z bazą danych ..... 51**
SQL Server 2000 ..... 52
DBase ..... 55
Access ..... 55
Sterowniki JDBC ..... 56
Podsumowanie ..... 59

**Rozdział 5. SQL Server 2000 ..... 61**
Instalacja pakietu ..... 64
Struktura SQL Servera 2000 ..... 69
Usługi SQL Server ..... 70
Najważniejsze narzędzia pakietu ..... 70
Przygotowanie projektu bazy ..... 81
Migracja bazy ..... 87
Podsumowanie ..... 90

**Rozdział 6. Tworzenie bazy danych Access ..... 91**
Przykładowy projekt bazy ..... 91
Użytkownicy i uprawnienia ..... 98
Podsumowanie ..... 99

**Rozdział 7. Przygotowanie bazy w systemie JDataStore ..... 101**
DataBase Pilot ..... 103
JDataStore Explorer ..... 104
Uprawnienia i użytkownicy ..... 111
JDataStore Server ..... 113
Podsumowanie ..... 115

**Rozdział 8. Komponenty bazodanowe w JBuilderze ..... 117**
Wstęp ..... 117
Java i bazy danych ..... 119
JBuilder i bazy danych ..... 123
DataExpress ..... 124
DataBase ..... 124
TableDataSet ..... 125
QueryDataSet ..... 126
QueryResolver ..... 129
ProcedureDataSet ..... 129
ProcedureResolver ..... 129
ParameterRow ..... 130
DataSetView ..... 132
DataStore ..... 134
StorageDataSet ..... 134
DataStoreConnection ..... 134
TxManager ..... 135
DataStoreServer ..... 135
DataStorePump ..... 136
DataStoreSync ..... 136
JDBCDataSource ..... 137
Podsumowanie ..... 137

**Rozdział 9. Komponenty graficzne dbSwing ..... 139**
JFrame ..... 141
JPanel ..... 142
JToolBar ..... 142

JMenuBar, JMenu, JMenuItem ... 143
JOptionPane — standardowe okna dialogowe ... 143
JDBRadioButton ... 147
JDBCheckBox ... 148
JDBToggleButton ... 149
JDBLabel ... 149
JDBTextField ... 149
JDBTextArea ... 150
JDBTextPane ... 152
JDBEditorPane ... 152
JDBComboBox ... 153
JDBList ... 156
JDBSlider ... 156
JDBTree ... 157
JDBTable ... 158
TableScrollPane ... 160
JDBStatusLabel ... 161
JDBNavToolBar ... 161
DBPasswordPrompter ... 162
Zdarzenia ... 162
Podsumowanie ... 165

**Rozdział 10. Projektowanie aplikacji bazodanowej ... 167**
Etapy projektu aplikacji klient-serwer ... 168
Charakterystyka aplikacji typu klient-serwer ... 169
Projektowanie aplikacji z użyciem JBuilder8 ... 169
JBuilder 8 — wiadomości ogólne ... 170
Elementy używane do projektowania aplikacji ... 175
Projekt aplikacji na platformie Access ... 176
Założenia projektowe ... 176
Projekt bazy ... 177
Projekt aplikacji ... 179
Projekt aplikacji na platformie SQL Server 2000 ... 196
Założenia projektowe ... 197
Projekt bazy ... 198
Projekt aplikacji ... 206
Wywoływanie okien z menu ... 228
Podsumowanie ... 230

**Rozdział 11. Prezentacja wyników pracy aplikacji: raporty i wydruki ... 231**
Projektowanie wydruku ... 234
Dołączanie rysunków do wydruków ... 237
Używanie wydruków podczas pracy aplikacji ... 238
Podsumowanie ... 239

**Rozdział 12. Przygotowanie pakietu aplikacji ... 241**
Tworzenie pliku archiwum ... 241
Tworzenie pliku uruchomieniowego ... 244
Podsumowanie ... 246

**Skorowidz ... 247**

Rozdział 7.

# Przygotowanie bazy w systemie JDataStore

W pakiecie JBuilder, poza środowiskiem projektowania aplikacji, zawarto również kilka narzędzi, służących do przygotowania i pracy z bazami danych w systemie JDataStore. Jest to rozwiązanie, które możemy stosować wszędzie tam, gdzie nie ma potrzeby używania systemów bazodanowych innych firm. Korzystanie z plików **.jds* [1] nie wymaga stosowania sterowników ODBC, połączenie jest nawiązywane całkowicie za pomocą języka Java, przy użyciu sterownika *com.borland.datastore.jdbc.DataStoreDriver*. JDataStore jest wydajnym, w całości napisanym w języku Java rozwiązaniem, stworzonym na potrzeby magazynowania danych dla aplikacji tworzonych w JBuilderze. Dostęp do bazy danych uzyskujemy przez sterowniki JDBC i interfejs DataExpress. System obsługuje transakcyjność i pracę wielu użytkowników, magazynowanie obiektów serializowanych, tabel i innych plików.

Bazy danych tego typu charakteryzują się następującymi parametrami:

- minimalna wielkość bloku danych: 1 KB,
- maksymalna wielkość bloku danych: 32 KB,
- domyślna wielkość bloku danych: 4 KB,
- maksymalna wielkość pliku JDataStore: 2 biliony bloków (2 G).
  Dla domyślnego rozmiaru bloku maksymalna wielkość bazy będzie miała 8 796 093 022 208 bajtów (8 TB),
- maksymalna liczba wierszy w tablicy: 4 biliony (4 G),
- maksymalna długość wiersza: 1/3 wielkości bloku,
- maksymalna wielkość danych typu BLOB: 2 GB każda,
- maksymalna wielkość strumienia plikowego: 2 GB dla każdego,
- znak separatora katalogów: /,

[1] Pliki zawierające strukturę bazy danych JDataStore.

- zarezerwowane nazwy:
  - *SYS/Connections*,
  - *SYS/Queries*,
  - *SYS/Users*.

W bazach typu JDataStore wykorzystywane są typy danych przedstawione w tabeli 7.1.

**Tabela 7.1.** *Typy danych obsługiwane przez JDataStore*

| Typ danych w JDataStore | Opis | Synonim |
|---|---|---|
| *SMALLINT* | Wartość numeryczna z zakresu –32768 do 32767 | *SHORT* |
| *INT* | Wartość numeryczna z zakresu –2147483648 do 2147483647 | *INTEGER* |
| *BIGINT* | Wartość numeryczna z precyzją –9223372036854775808 do 9223372036854775807 | *LONG* |
| *DECIMAL* (*p*, *d*) | Wartość rzeczywista z precyzją: *p* oznacza liczbę cyfr przed przecinkiem, *d* — po przecinku. Maksymalna ich suma może być równa 72 | *BIGDECIMAL* |
| *REAL* | Wartość numeryczna z przedziału od 1.4E-45 do 3.4E38 z precyzją 23 bitów | |
| *DOUBLE* | Wartość rzeczywista z zakresu od 4.9E-324 do 1.8E308 z precyzją 52 bitów | *DOUBLE_PRECISION* |
| *FLOAT* (*p*) | Wartość numeryczna z precyzją do 52 bitów | |
| *VARCHAR* (*p*, *m*) | Ciąg znaków zmiennej długości: *p* — długość całkowita ciągu, *m* — liczba znaków w wierszu. Domyślnie wielkość *p* nie jest określona, *m* domyślnie przyjmuje wartość 64 | *STRING* |
| *VARBINARY* (*p*, *m*) | Ciąg binarny zmiennej długości. Znaczenie parametrów takie samo jak powyżej | *INPUTSTREAM BINARY* |
| *OBJECT* (*t*, *m*) | Ciąg obiektów języka Java, składowanych w bazie dzięki *serializacji* obiektów: *t* oznacza klasę obiektu, *m* — liczbę obiektów | |
| *BOOLEAN* | Zmienna dwuwartościowa *true* / *false* | *BIT* |
| *DATE* | Data liczona od roku 0 | |
| *TIME* | Czas określany w zakresie: 00:00:00 do 23:59:59 | |
| *TIMESTAMP* | Zwraca czas i datę w zależności od strefy czasowej | |

Dołączone narzędzia, poza przeglądaniem danych, dodatkowo zapewniają nam proste mechanizmy administracyjne, tzn. zarządzanie użytkownikami, przydzielanie praw, tworzenie kopii bezpieczeństwa bazy, import danych z innych platform bazodanowych itp. Do wykorzystania mamy trzy aplikacje, z których każda jest używana do innych funkcji.

- *DataBase Pilot* zapewnia możliwość przeglądania baz, których aliasy są zdefiniowane w ODBC. Przy jego wykorzystaniu możemy podejrzeć zawartość każdej bazy, do której posiadamy sterownik ODBC, a nie tylko JDataStore.
- *JDataStore Explorer* — za pomocą tego narzędzia mamy możliwość tworzenia struktury bazy JDataStore, tworzenia nowych baz, testowania zapytań SQL itd.
- *JDataStore Server* — serwer baz danych *JDataStore*. Jeżeli chcemy stworzyć bazę zdalną, z którą aplikacje klientów będą łączyły się przez sieć, lub umożliwić podłączenie wielu użytkowników, musimy na komputerze uruchomić serwer JDataStore.

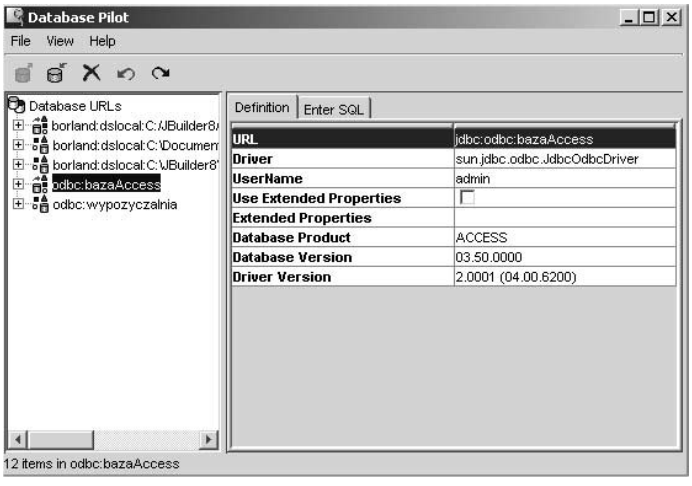
# DataBase Pilot

Aplikację możemy wykorzystać na przykład w sytuacji, gdy jest potrzebna ręczna modyfikacja bazy, a nie posiadamy narzędzi administracyjnych, które powinny być dostarczone z określonym systemem bazodanowym. Na rysunku 7.1 widzimy interfejs aplikacji z listą dostępnych baz danych do przeglądania w lewym oknie. Prawe okno, w zależności od wybranej zakładki, udostępnia dwie funkcje:

- informacje o źródle danych, gdy wybrana jest zakładka *Definition*. Na przykładzie widać dane dotyczące bazy danych stworzonej za pomocą Accessa. W zależności od rodzaju wybranej bazy, znajdują się tam podstawowe informacje dotyczące sterownika, nazwy użytkownika, hasła itd.,
- podręczny edytor poleceń SQL, jeżeli wybierzemy *Enter SQL*. Za jego pomocą możemy edytować polecenia SQL i wykonywać je natychmiastowo — *EXECUTE*. Możliwe jest również wczytywanie skryptów zawierających odpowiednie polecenia i wykonywanie ich za pomocą przycisku *Load SQL*. Na dole zakładki znajduje się wynik wykonania polecenia `SELECT` wraz z komponentem nawigacyjnym, za pomocą którego możemy edytować określone wartości. Jeżeli dane są pobierane z kilku tablic, należy pamiętać o wypełnieniu wszystkich wymaganych danych przy uwzględnieniu struktury bazy, czyli ograniczeń i kluczy. W przypadku gdy dane będą niewłaściwe lub niekompletne, zostanie wygenerowany komunikat o błędzie.

Bardzo proste menu główne aplikacji zawiera trzy pozycje:

- *File* — podmenu, udostępnia nam funkcje:
  - *New* tworzy nowy URL, czyli źródło danych na liście dostępnych,
  - *Open* otwiera bazę danych stworzoną wcześniej za pomocą opcji *New*,
  - *Close* zamyka aktualne źródło danych,
  - *Create Table* tworzy tabelę w wybranej bazie,
  - *Apply* uaktualnia zmiany w bazie,

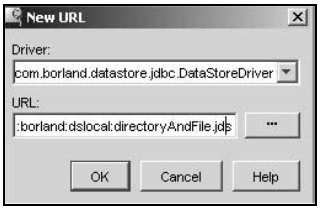



**Rysunek 7.1.** *Okno główne aplikacji DataBase Pilot*

- *Cancel* cofa zmiany dokonane w bazie,
- *Delete* usuwa URL z listy.

- *View* — ustawienia dotyczące sposobu prezentowania danych w aplikacji. Dodatkowa opcja pozwala na podgląd danych typu BLOB.

W przykładzie na rysunku 7.1 w oknie z listą URL znajdują się trzy bazy JDataStore oraz dwie Accessa. Aby dodać nowy alias bazy danych do listy, używamy funkcji *New*. W oknie dialogowym, jak na rysunku 7.2, wybieramy sterownik, za pomocą którego będziemy się komunikowali z bazą (w polu *Driver*), oraz źródło danych.

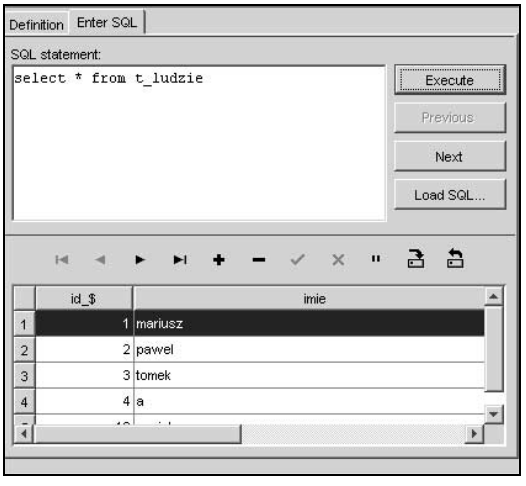



**Rysunek 7.2.** *Konfiguracja nowego URL-a*

W przypadku sterownika ODBC jest to alias bazy danych z Administrator ODBC, w przypadku bazy JDataStore jest to plik **.jds* w określonej lokalizacji. Dostępne sterowniki omówię przy komponencie `DataBase` w rozdziale 8. Kiedy już połączenie z bazą zostanie nawiązane, możemy za pomocą zakładki *Execute Sql* i odpowiednich poleceń operować na danych. Na rysunku 7.3 widać efekty wykonania prostego zapytania na przykładowej bazie.

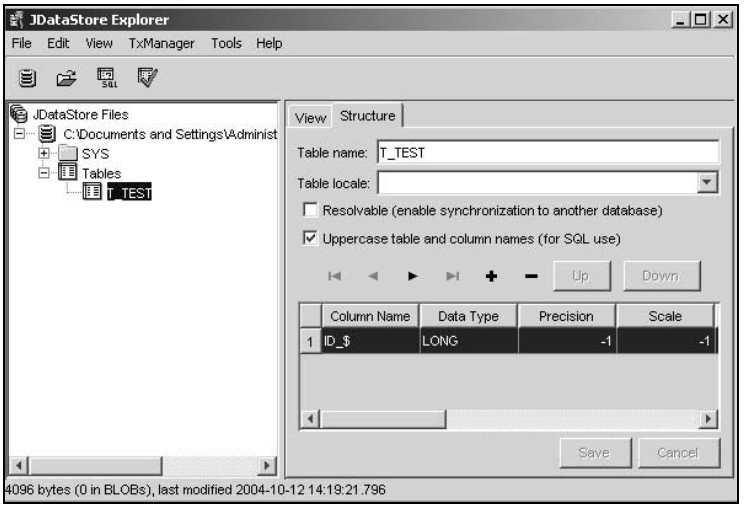
# JDataStore Explorer

Drugie z opisywanych narzędzi jest przeznaczone dla projektantów bazy JDataStore. Zawiera ono szereg przydatnych funkcji, za pomocą których możemy wykonać wszystkie charakterystyczne operacje, znane z innych platform bazodanowych. Okno główne aplikacji




**Rysunek 7.3.** *Wykorzystanie zakładki Enter Sql do pracy z bazą danych*

(rysunek 7.4) jest podzielone na dwie części. W lewej znajduje się drzewo rozwijane z aktualnie otwartymi źródłami JDataStore oraz ich elementami, w prawej pojawiają się różne opcje, w zależności od wybranego obiektu.




**Rysunek 7.4.** *Okno główne aplikacji JDataStore Explorer*

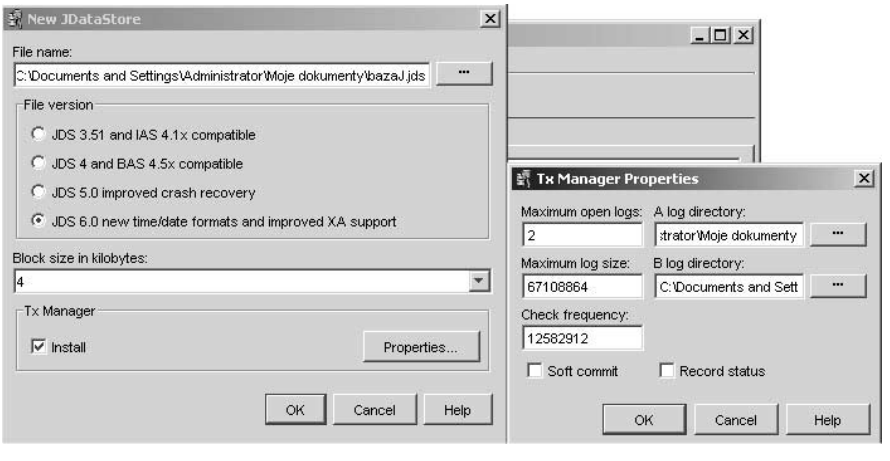
W przedstawionym przykładzie z drzewa została tabela — *T_TEST*. Z prawej strony możemy podejrzeć jej zwartość na karcie *View* lub strukturę (karta *Structure*), co widać na rysunku.

Menu aplikacji zostało podzielone na kilka sekcji:

- *File* — za pomocą funkcji tu umieszczonych tworzymy nowe bazy danych lub otwieramy istniejące,
- *Edit* — funkcje edycji, takie jak usuwanie, zmiana nazwy, cofanie operacji usuwania,

- *View* — opcje rozwijania i zwijania poszczególnych elementów struktury drzewa,
- *TxManager* — manager zarządzania transakcjami,
- *Tools* zawiera funkcje służące do administracji i zarządzania bazą danych:
  - *Import* — wczytywanie z innych platform bazodanowych struktury bazy,
  - *Sql* — testowanie poleceń SQL,
  - *Create Table* — tworzenie tabeli,
  - *Create Index* — tworzenie indeksów,
  - *Refresh JDataStore* — odświeżanie bazy danych,
  - *Save JDataStore* — zapis zmian do pliku *jds*,
  - *Delete JDataStore* — usunięcie wybranej bazy danych,
  - *Verify JDataStore* — weryfikacja poprawności bazy,
  - *Pack JDataStore* — wykonywanie kopii bezpieczeństwa bazy,
  - *Copy JDataStore* — kopiowanie pliku *jds*,
  - *Administer Users* — zarządzanie użytkownikami i prawami.

W celu zapoznania się z programem, przedstawię proces przygotowania kilku tabel w nowej bazie danych. Pierwszym krokiem jest oczywiście stworzenie pliku *jds*. Wybieramy z menu opcję *File/New* i w oknie dialogowym *New JDataStore*, jak na rysunku 7.5, ustalamy wymagane parametry.




**Rysunek 7.5.** *Parametry nowo tworzonej bazy danych JDataStore*

Najważniejsze parametry to ustalenie: pliku *jds*, w którym będzie przechowywana nasza baza, wersji sterownika JDS, wielkości bloku w kilobajtach (*Block size in kilobytes*) oraz parametrów dodatkowych, dotyczących używania logów. Jeżeli zdecydujemy, że nasza baza nie powinna obsługiwać transakcji, pole *Install* w sekcji *Tx Manager* powinno

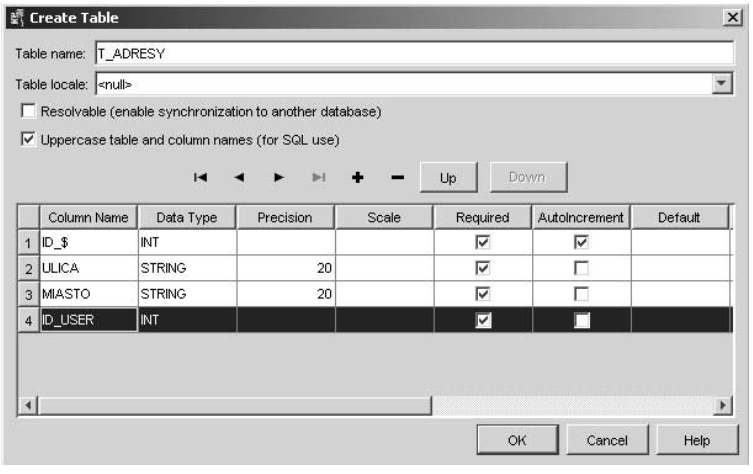
pozostać odznaczone. W przeciwnym przypadku wybieramy przycisk *Properties* i ustalamy dodatkowe parametry, jak widać na tym samym rysunku w oknie *Tx Manager Properties*. W oknie zawarto kilka parametrów związanych z obsługą logów:

- *Maximum open logs* decyduje o liczbie otwartych plików logów. Jeżeli używamy więcej niż jednego, warto rozmieścić je w różnych lokalizacjach lub nawet napędach. Zabezpieczy to naszą bazę przed konsekwencjami awarii,
- *Maximum log size* — maksymalna wielkość pliku logów, domyślnie 64 MB,
- *A, B log directory* — lokalizacja plików logów. Jeżeli pola pozostawimy puste, pliki logów zostaną umieszczone w tej samej lokalizacji, co plik bazy danych,
- *Check frequency* — częstotliwość zapisywania do plików. Decyduje, jak często powstają zapisy do plików logów. Im częstsze zapisy, tym większe bezpieczeństwo, ale wolniejsza praca bazy,
- *Soft Commit* określa, czy używać opcji programowego zatwierdzania transakcji. Programowa realizacja gwarantuje odzyskanie danych w momencie awarii aplikacji. Nie zabezpiecza natomiast przed awariami spowodowanymi przez system operacyjny lub sprzęt, dlatego też jest zalecana dla stabilnego środowiska pracy aplikacji. Opcja ta poprawia działanie poprzez wstrzymywanie zapisów na dysk i wykonywanie operacji bazodanowych w pamięci komputera,
- *Record status* kontroluje, czy komunikaty generowane podczas pracy z bazą są zapisywane do pliku logów *STATUS log*, który jest przechowywany w lokalizacji określonej w *A log directory* (rysunek 7.5).

Po stworzeniu bazy przystępujemy do zaprojektowania struktury. Możemy do tego celu wykorzystać zewnętrzne narzędzia projektowe (jak chociażby wspomniane w rozdziale 3.) i po wygenerowaniu skryptów SQL wczytać je i wykonać na naszej bazie, lub wykorzystać funkcje programu. Aby zapoznać Czytelnika z działaniem JDataStore Explorera, zaprojektujemy bazę za jego pomocą. Na początku dodamy dwie tablice, zawierające proste dane, później utworzymy indeksy i powiązania. W celu dodania nowej tablicy wybieramy z menu *Tools/Create Table*. Na rysunku 7.6 widzimy okno dialogowe edytora tablicy.

Dla naszych potrzeb przygotujemy dwie tablice, zawierające ponownie dane ludzi i dane adresowe. Jedną z nich widzimy na rysunku 7.6, zawiera ona trzy kolumny:

- *ID_$* — identyfikator, typ *INT*, pole ustalone jako wymagane (*Required*) oraz automatycznie zwiększające wartość w przypadku wykonywania polecenia `INSERT` (*AutoIncrement*). Dodatkowo dla każdej kolumny można ustalić wartość domyślną w kolumnie *Default*,
- *MIASTO* — nazwa miasta, typ *STRING* o długości 20 znaków, pole również musi być wypełnione,
- *ULICA* — definicja jest dokładnie taka sama, jak w przypadku poprzedniej kolumny,
- *ID_USER* — kolumna, która zostanie wykorzystana jako klucz obcy do kolumny *ID_$* w tabeli *T_LUDZIE*. Również zdefiniujemy ją jako wymaganą i ustalimy taki sam typ, jak dla kolumn *ID_$*.




**Rysunek 7.6.** *Definiowanie nowej tablicy o nazwie T_ADRESY*

Druga o nazwie *T_LUDZIE* będzie zawierała następujące kolumny:

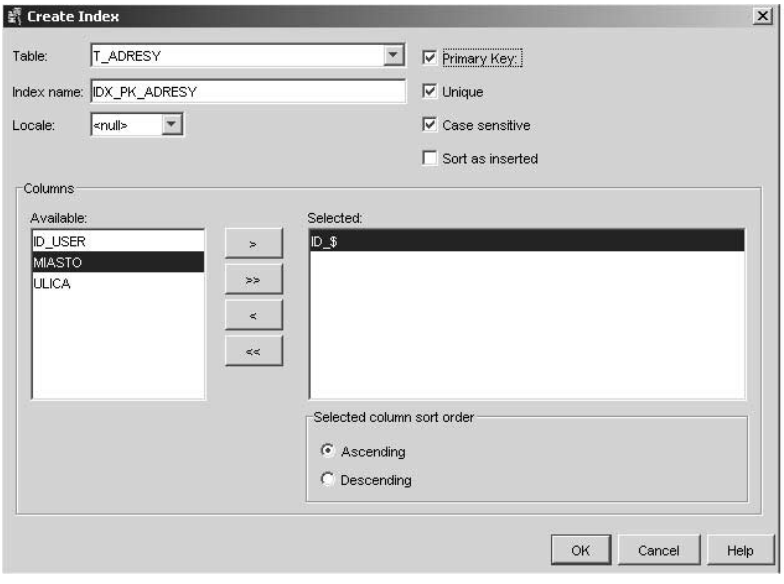
- *ID_$* — identyfikator, typ *INT*, pole ustalone jako wymagane (*Required*) oraz automatycznie zwiększające wartość w przypadku wykonywania polecenia `INSERT` (*AutoIncrement*),
- *IMIE* — nazwa miasta, typ *STRING* o długości 15 znaków, pole również musi być wypełnione,
- *NAZWISKO* — definicja jest dokładnie taka sama, jak w przypadku poprzedniej kolumny,
- *ROK* — rok urodzenia.

Kiedy tablice zostaną już przygotowane, należy zdefiniować — jeśli potrzeba — indeksy i klucze w poszczególnych tablicach. W celu utworzenia indeksu i klucza w tabeli wybieramy ją z listy i wywołujemy opcję menu *Tools/Create Index*. Powoduje to otwarcie okna dialogu jak na rysunku 7.7, w którym określimy wymagane parametry.

Aby przygotować indeks, wystarczy podać jego nazwę w polu *Index Name* i przenieść w sekcji *Columns* kolumny (jedną lub więcej) z okna *Available* do *Selected*. W tym przykładzie tworzymy indeks na kolumnie *ID_$*. Ma on być kluczem głównym, dlatego zaznaczamy pole *Primary Key* oraz *Unique*. Podczas tworzenia indeksów w JDataStore zauważamy, że nie można stworzyć indeksu na kolumnie z atrybutem *AutoIncrement* (numerowanie automatyczne). W ten sposób możemy przygotować indeksy i klucze główne we wszystkich wymagających tego tabelach w bazie. Nazwy nadawane indeksom powinny być dla jasności nazywane w zgodzie z przyjętą przez projektanta konwencją. Oczywiście każdy projektant może przyjąć własny sposób nazewnictwa. W naszym przykładzie przyjęliśmy schemat:

- dla klucza głównego — *IDX_PK_nazwaTabeli*, pierwszy człon oznacza, że jest to indeks, drugi oznacza PK — *primary key*, FK — *foreign key*,
- dla klucza obcego — *IDX_FK_nazwaKolumny*.

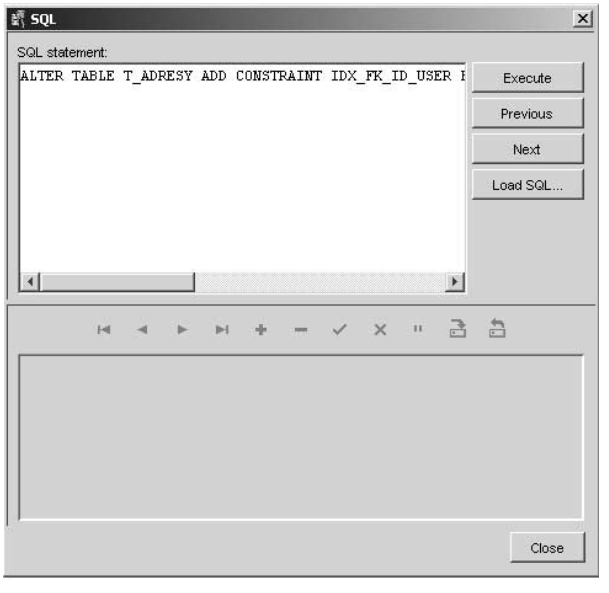
**Rysunek 7.7.**
*Edytor indeksów*



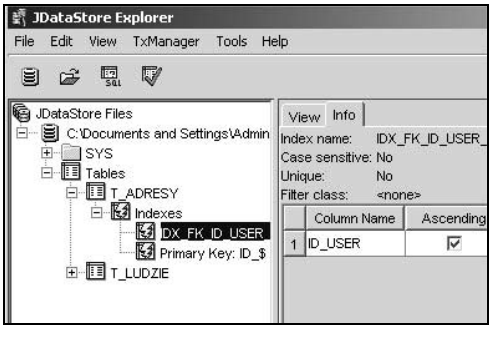

O ile do tworzenia tabel, indeksów i kluczy głównych możemy wykorzystać odpowiednie pozycje menu, przygotowanie pozostałych elementów bazy nie jest już takie proste. Aplikacja nie udostępnia narzędzi do tworzenia ograniczeń czy wizualnego tworzenia relacji pomiędzy tabelami (jak chociażby prezentowany w poprzednim rozdziale MS Access). W celu zapewnienia spójności referencyjnej bazy oraz ustalenia dodatkowych parametrów jej struktury musimy wykorzystać język SQL i za pomocą odpowiednich poleceń utworzyć wymagane składniki. Na szczęście autorzy aplikacji wśród dostępnych funkcji umieścili edytor poleceń SQL. Praca z nim nie jest może szczególnie wygodna, ale spełnia on swoje zadania. My go wykorzystamy do przygotowania kluczy obcych oraz dodania ograniczeń na kolumny tabel. Edytor można uruchomić na dwa sposoby, wybierając z menu *Tools/SQL* lub ikonę SQL z paska narzędzi. Okno edytora widzimy na rysunku 7.8, zawiera ono kilka przycisków, które możemy wykorzystać do wykonania polecenia SQL (*Execute*), przywrócenia poprzedniego polecenia (*Previous*), wybrania następnego (*Next*) oraz wczytania zewnętrznego pliku z poleceniami SQL (*Load SQL*). Za pomocą tego ostatniego możemy wczytać skrypty przygotowane przez zewnętrzne narzędzia projektowe typu *CASE*. W pierwszym kroku przygotujemy klucz obcy w tabeli *T_ADRESY* do tabeli *T_LUDZIE* o nazwie *IDX_FK_ID_USER*. Na podstawie wiedzy zdobytej w rozdziale 2. edytujemy następujące polecenie:

```
ALTER TABLE T_ADRESY ADD CONSTRAINT IDX_FK_ID_USER FOREIGN KEY(ID_USER) REFERENCES
T_LUDZIE (ID_$) ON DELETE CASCADE ON UPDATE CASCADE
```

Stworzy ono ograniczenie na kolumnie *ID_USER* w tablicy *T_ADRESY*, które nie zezwoli, aby do tej kolumny została wpisana wartość, która nie występuje w kolumnie *ID_$* w tabeli *T_LUDZIE*. Jest to podobna relacja, jaką tworzyliśmy w rozdziale 6., jednak w tym przypadku projektujemy ją bez użycia interfejsu graficznego. Po dodaniu ograniczenia odpowiednia gałąź pojawia się w lewym oknie struktury bazy (rysunek 7.9).




**Rysunek 7.8.** *Edytor poleceń SQL w aplikacji JDataStore Explorer*




**Rysunek 7.9.** *Struktura bazy z dodanymi kluczami primary key i foreign key*

Należy pamiętać o zgodności typów kolumn, wykorzystywanych w takim ograniczeniu. W ten sposób możemy tworzyć zależności pomiędzy innymi tablicami w bazie. Zasada działania jest taka sama, jak w innych systemach bazodanowych i zapewnia spójność referencyjną bazy. Drugie ograniczenie, jakie dodamy do naszej bazy, będzie dotyczyło kolumny *ROK* w tablicy *T_LUDZIE*. Ma ono na celu uniemożliwienie wpisywania roku poniżej wartości 1900. W tym celu wykorzystamy poniższą instrukcję i za pomocą polecenia Execute edytora SQL wykonamy ją na bazie.

```
ALTER TABLE T_LUDZIE ADD CONSTRAINT MIN_ROK_LUDZIE CHECK(ROK>1900)
```

Tym sposobem możemy przygotować wszelkie inne polecenia SQL i wykonać je na bazie JDataStore.

Poza standardowymi funkcjami, dostarczanymi przez język SQL, bazy JDataStore dostarczają kilka dodatkowych funkcji, które programista może wykorzystać przy tworzeniu poleceń SQL. Należy jednak pamiętać, że instrukcje SQL, wykorzystujące te funkcje, będą wymagały zmian, gdy przeniesiemy je na inną platformę bazodanową. Krótka lista najczęściej wykorzystywanych funkcji:

- ABSOLUTE zwraca wartość bezwzględną z podanej

```
SELECT ABSOLUTE(WARTOSC) FROM TABLICA1
```

- CURRENT_DATE, CURRENT_TIME i CURRENT_TIMESTAMP — bardzo przydatne funkcje, zwracające aktualną datę i czas. Podajemy nazwę funkcji w poleceniu select, jak widać poniżej, nazwa tablicy jest nieistotna, znajduje się jedynie w celu poprawności składni SQL.

```
SELECT (CURRENT_TIMESTAMP)FROM T_LUDZIE
```

  W wyniku zapytania otrzymamy aktualną datę i czas.

- LOWER and UPPER — użycie funkcji spowoduje zmianę wszystkich liter w ciągu lub kolumnie podanej jako parametr na duże lub małe.

- EXTRACT służy do wydzielenia z danych typu DATE i TIME.

  Za pomocą pierwszego polecenia pobierzemy miesiąc z podanej daty i będzie to wartość 5.

```
EXTRACT(MONTH FROM DATE '1999-05-17')
```

  Za pomocą drugiego godzinę — 19.

```
EXTRACT(HOUR FROM TIME '19:00:00'.
```

  Podczas używania funkcji należy pamiętać o odpowiedniej formie podanej wartości, jak widać w przykładach.

- POSITION zwraca pozycję zadanego ciągu znaków wewnątrz innego. Wykonanie polecenia

```
POSITION('BCD' IN 'ABCDEFG') da w rezultacie 2.
```

- SUBSTRING zwraca w wyniku zapytania *substring* (wycinek tekstu) pobrany z kolumny ciąg znaków.

```
SUBSTRING('ABCDEFG' FROM 2 FOR 3) zwraca ciąg 'BCD'
```

- TRIM — z pobranego ciągu znaków usuwa spacje lub inne zadane znaki

```
TRIM(' Usuwanie spacji   ')
```

  W wyniku wykonania tej funkcji otrzymamy string 'Usuwanie spacji', wykonanie polecenia z parametrem spowoduje usunięcie podanego znaku — tutaj '0'

```
TRIM(LEADING '0' FROM '00000789.75') w wyniku otrzymamy '789.75'.
```