DO NOWEJ PODSTAWY PROGRAMOWEJ

Część 2

Kwalifikacja E.14

Tworzenie baz danych i administrowanie bazami

Podręcznik do nauki zawodu technik informatyk

Jolanta Pokorska

Helion Edukacja

Podręcznik dopuszczony do użytku szkolnego przez ministra właściwego do spraw oświaty i wychowania i wpisany do wykazu podręczników przeznaczonych do kształcenia w zawodzie technik informatyk, na podstawie opinii rzeczoznawców: mgr Marii Dziurzyńskiej-Ścibior, mgr Elżbiety Leszczyńskiej, dr. inż. Stanisława Szabłowskiego.

Nazwa kwalifikacji: Kwalifikacja E-14. Część 2. Tworzenie baz danych i administrowanie bazami.
Typ szkoły: technikum, szkoła policealna, kurs kwalifikacyjny.
Rok dopuszczenia 2013.

Numer ewidencyjny w wykazie: 50/2013

Redaktorzy prowadzący: Marcin Borecki
Projekt okładki: Maciej Pasek

Fotografia na okładce została wykorzystana za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie?e14te2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-6510-5

Printed in Poland.

Spis treści

Wstęp

Podręcznik *Tworzenie baz danych i administrowanie bazami danych* jest drugim z grupy podręczników przygotowanych dla kwalifikacji *E.14. Tworzenie aplikacji internetowych i baz danych oraz administrowanie bazami*.

Zawarte w podręczniku *Tworzenie baz danych i administrowanie bazami danych* treści opierają się na podstawie programowej kształcenia w zawodach wprowadzonej rozporządzeniem Ministra Edukacji Narodowej z 7 lutego 2012 r.

Obejmują one zagadnienia teoretyczne prowadzące do uzyskania wymienionych w podstawie programowej efektów kształcenia, projekty różnych zadań oraz ich praktyczną realizację. Tak skonstruowany podręcznik pomaga uczniowi w zdobywaniu wymaganej wiedzy oraz umożliwia mu samodzielne poszerzanie umiejętności.

Podręcznik składa się z sześciu rozdziałów. Ich budowa pozwala na realizowanie treści programowych w sposób wybrany przez nauczyciela.

Rozdział 1., „Zasady projektowania relacyjnych baz danych", zawiera omówienie zagadnień związanych z modelem relacyjnym baz danych oraz z planowaniem i projektowaniem relacyjnych baz danych. Dotyczy efektów związanych z projektowaniem i tworzeniem relacyjnych baz danych. Efektami tymi są: identyfikowanie podstawowych pojęć dotyczących relacyjnych baz danych, identyfikowanie elementów bazy danych, stosowanie zasad projektowania baz danych, tworzenie graficznych schematów baz danych, projektowanie tabel baz danych, identyfikowanie i stosowanie zasad normalizacji tabel, definiowanie związków między tabelami baz danych.

Rozdział 2., „Tworzenie lokalnych baz danych w programie MS Access", zawiera omówienie zagadnień związanych z tworzeniem bazy danych, tworzeniem obiektów bazy danych oraz zarządzaniem bazą danych w programie MS Access. Dotyczy efektów związanych z tworzeniem tabel, formularzy, zapytań i raportów, modyfikowaniem i rozbudową struktury baz danych, a także z importowaniem danych do bazy. Efektami tymi są: projektowanie formularzy do przetwarzania danych, wykorzystywanie podformularzy do przetwarzania danych, identyfikowanie i stosowanie różnych rodzajów kwerend do przetwarzania danych, projektowanie i wykonywanie raportów z bazy danych, modyfikowanie bazy danych oraz struktury bazy danych, rozbudowywanie struktury bazy danych.

Rozdział 3., „Systemy baz danych", zawiera omówienie zagadnień związanych z architekturą systemów baz danych oraz z systemami zarządzania bazami danych. Dotyczy efektów związanych z systemami baz danych oraz z systemami zarządzania bazami danych. Efektami tymi są: identyfikowanie różnych systemów zarządzania bazami danych, identyfikowanie funkcji systemów zarządzania bazami danych, identyfikowanie serwerów baz danych, identyfikowanie właściwości bazy danych.

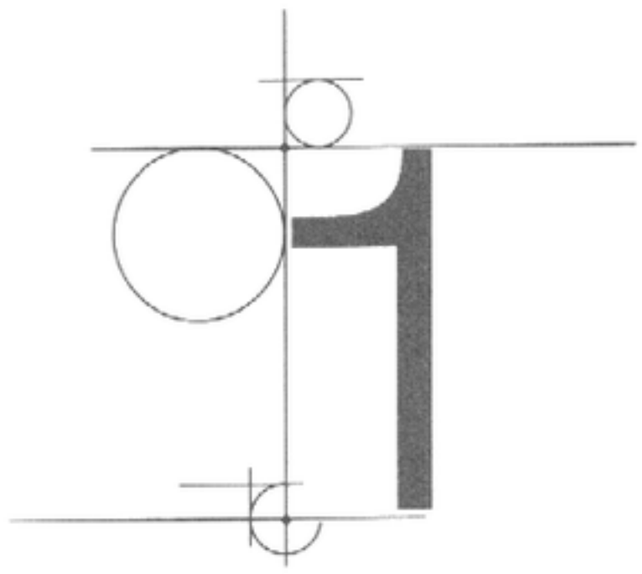
Rozdział 4., „Serwery bazodanowe", zawiera omówienie zagadnień związanych z instalowaniem i konfigurowaniem systemów bazodanowych oraz systemów zarządzania bazą danych. Dotyczy efektów związanych z instalowaniem systemów baz danych, konfigurowaniem serwerów baz danych, wykorzystaniem narzędzi graficznych do tworzenia baz danych oraz do zarządzania bazami. Efektami tymi są: instalowanie systemów baz danych, zarządzanie bazami danych, stosowanie narzędzi graficznych do tworzenia baz danych i schematów baz danych.

Rozdział 5., „SQL — strukturalny język zapytań", zawiera omówienie zagadnień związanych z tworzeniem bazy danych i zarządzaniem nią w języku SQL. Dotyczy efektów związanych z korzystaniem z funkcji strukturalnego języka zapytań oraz z posługiwaniem się strukturalnym językiem zapytań do obsługi baz danych. Efektami tymi są: identyfikowanie i stosowanie składni strukturalnego języka zapytań, stosowanie funkcji strukturalnego języka zapytań, stosowanie instrukcji strukturalnego języka zapytań, identyfikowanie i stosowanie transakcji przy użyciu strukturalnego języka zapytań.

Rozdział 6., „Administrowanie serwerami baz danych", zawiera omówienie zagadnień związanych z administrowaniem serwerami bazodanowymi. Dotyczy efektów związanych z określaniem uprawnień poszczególnych użytkowników, określaniem zabezpieczeń dla użytkowników, dobieraniem sposobów ustawienia zabezpieczeń dostępu do danych, udostępnianiem zasobów bazy danych w sieci, zarządzaniem kopiami zapasowymi baz danych i ich odzyskiwaniem, kontrolowaniem spójności bazy danych, dokonywaniem naprawy bazy danych. Efektami tymi są: zarządzanie bazą danych, kontrolowanie fizycznej spójności bazy danych, kontrolowanie logicznej spójności bazy danych, identyfikowanie i dobieranie sposobów ustawienia zabezpieczeń dostępu do danych, zarządzanie bezpieczeństwem bazy danych, identyfikowanie uprawnień użytkowników bazy danych, zarządzanie kopiami zapasowymi baz danych, zarządzanie odzyskiwaniem danych.

Część II

Tworzenie baz danych i administrowanie bazami danych



Zasady projektowania relacyjnych baz danych

1.1. Wprowadzenie do baz danych

Tradycyjne dane przechowywane są w postaci dokumentów papierowych. Dokumenty takie zawierają opisy przechowywanych obiektów. Obiektami mogą być na przykład samochody, książki lub osoby. Opisywane są także związki zachodzące między obiektami — kto jest właścicielem samochodu, kto wypożyczył książkę, gdzie pracuje dana osoba. Dokumenty opisujące obiekty i związki zachodzące między nimi są gromadzone w postaci kartotek, katalogów lub archiwów i są przechowywane w kopertach, teczkach albo segregatorach. Dostęp do tak opracowanych dokumentów jest trudny i zajmuje dużo czasu. Aby przyspieszyć wyszukiwanie danych, sporządza się różne spisy, wyciągi i katalogi.

Dobrym rozwiązaniem tych problemów jest przeniesienie takich dokumentów do komputera. Po zapisaniu danych w pamięci komputera można obsługiwać tak utworzoną bazę danych, korzystając z dostępnych narzędzi.

Zalety korzystania z komputerowych baz danych to:

- szybkie wyszukiwanie informacji,
- łatwe wykonywanie obliczeń,
- możliwość przechowywania dużej ilości danych na małej powierzchni,
- szybkie porządkowanie danych.

DEFINICJA

Baza danych to uporządkowany zbiór danych z określonej dziedziny tematycznej, zorganizowany w sposób ułatwiający do nich dostęp.

DEFINICJA

System zarządzania bazą danych to program zarządzający danymi w bazie i umożliwiający ich przetwarzanie.

DEFINICJA

System bazy danych to baza danych i system zarządzania bazą danych.

U podstaw konstruowania bazy danych leży założenie, że użytkownik, dla którego ta baza jest przeznaczona, nie musi być specjalistą z dziedziny baz danych, może w ogóle ich nie znać. Mimo to powinien bez problemów radzić sobie z obsługą zaprojektowanej bazy danych. Aby to było możliwe, podstawą tworzonej bazy musi być solidny projekt określający potrzeby użytkownika dotyczące gromadzenia, przechowywania i przetwarzania danych oraz definiujący czynności składające się na obsługę bazy danych.

Zadanie 1.1

Podaj przykłady możliwych zastosowań bazy danych.

1.2. Modele baz danych

Aby przechowywać dane na komputerze, konieczne jest określenie formy ich przechowywania. Na potrzeby baz danych zostały zdefiniowane klasyczne techniki organizowania informacji, zwane modelami baz danych.

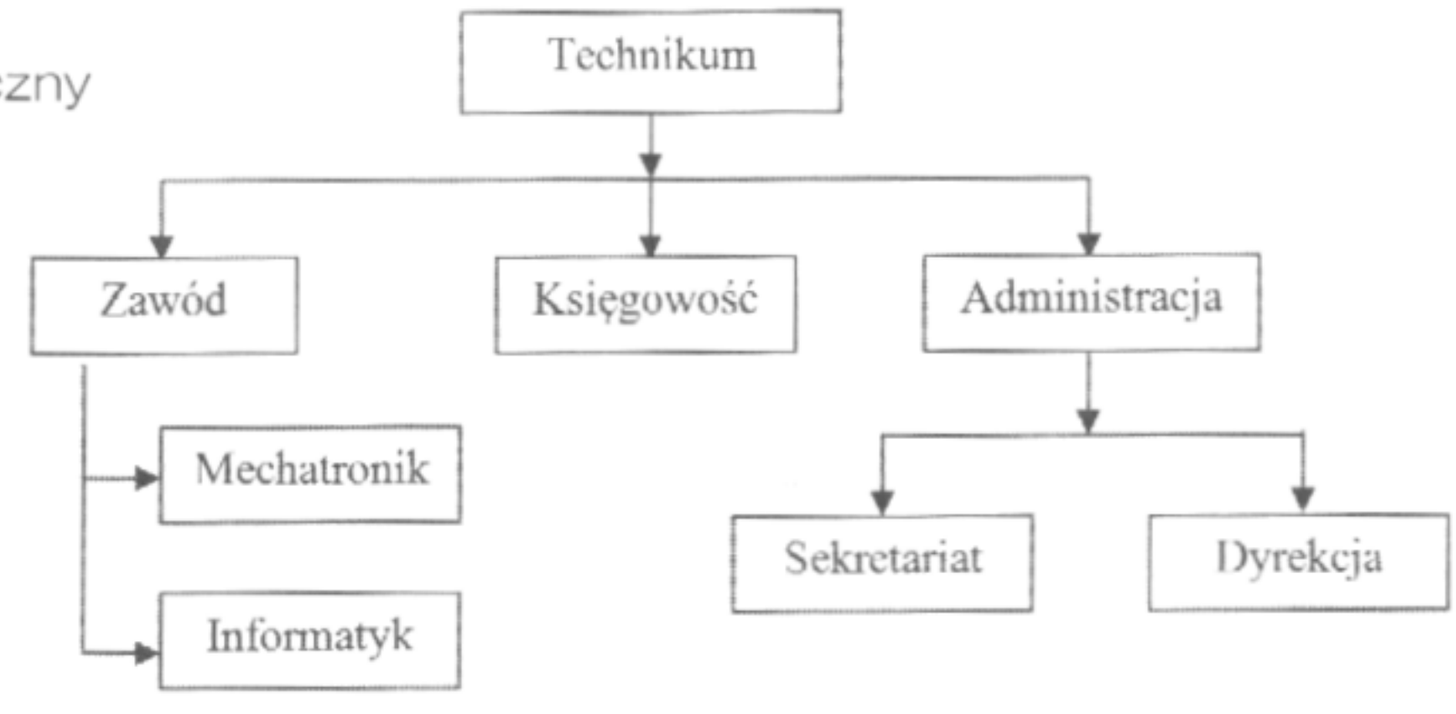
1.2.1. Model hierarchiczny

W tym modelu przechowywane dane są zorganizowane w postaci drzewa (rysunek 1.1). Informacja jest zawarta w dokumentach oraz w strukturze drzewa (podobnej do drzewa folderów na dysku komputera).

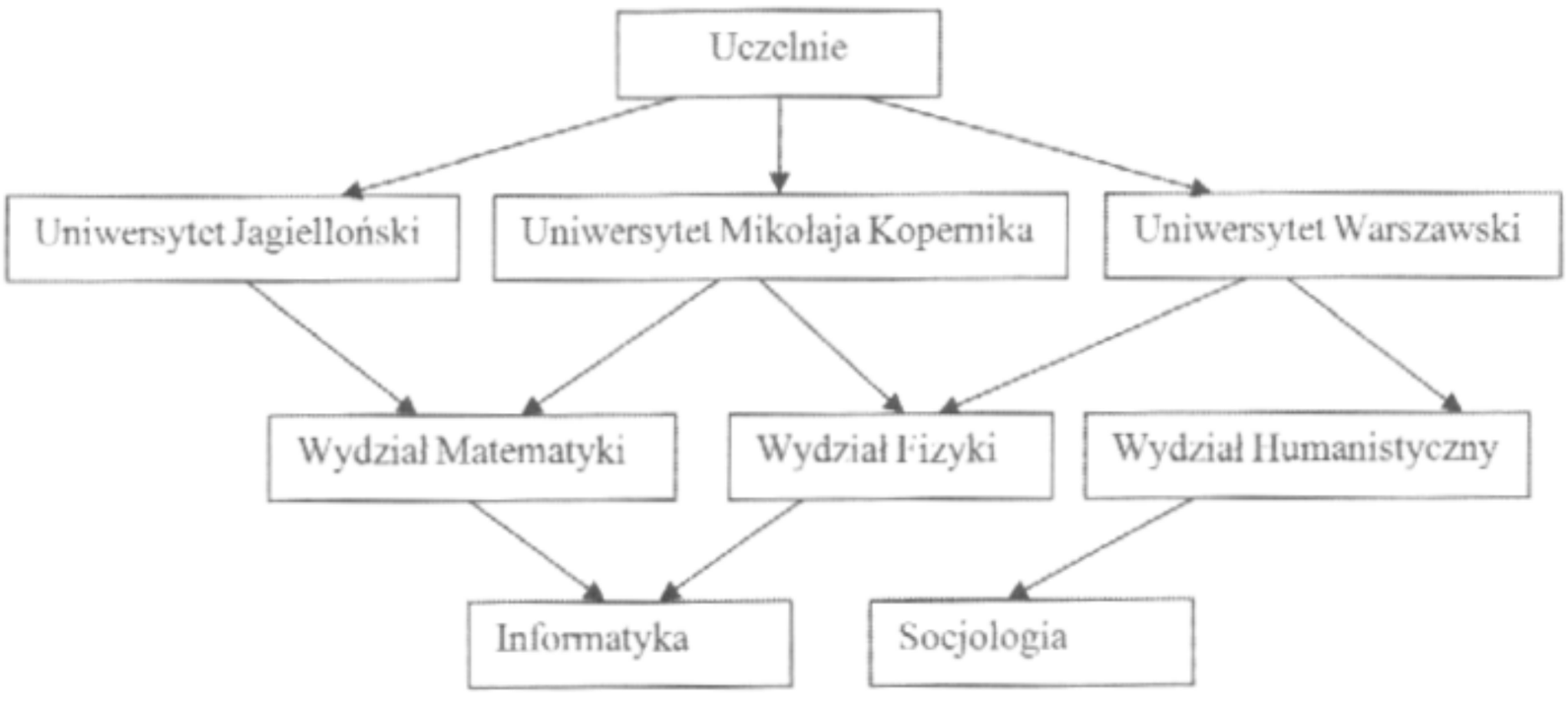
1.2.2. Model sieciowy

Połączenia między dokumentami tworzą sieć (rysunek 1.2). Informacja jest zawarta w dokumentach oraz w przebiegu połączeń sieci.

Rysunek 1.1.
Model hierarchiczny



Rysunek 1.2.
Model sieciowy



1.2.3. Model obiektowy

Model obiektowy łączy cechy programów komputerowych tworzonych w językach programowania obiektowego z cechami aplikacji bazodanowych. Obiekt w bazie reprezentuje obiekt w świecie rzeczywistym.

1.2.4. Model relacyjny

W relacyjnym modelu baz danych informacja jest zapisywana w tabeli w formie wierszy. Tabele tworzą między sobą powiązania zwane relacjami (rysunek 1.3).

Rysunek 1.3.
Model relacyjny

Ze względu na funkcjonalność model relacyjny jest najczęściej wykorzystywany przy projektowaniu baz danych.

1.2.5. Model postrelacyjny

W modelu postrelacyjnym baz danych zakłada się, że są to bazy relacyjne poszerzone na przykład o elementy obiektowości, obsługę XML, rozwiązania analityczne, zapytania historyczne. Model postrelacyjny powstał w wyniku rozszerzenia relacyjnego modelu baz danych o elementy ułatwiające opisanie skomplikowanej rzeczywistości. Są to na przykład złożone struktury danych, zagnieżdżone relacje, atrybuty wirtualne, abstrakcyjne typy danych czy funkcje rozszerzalne.

1.3. Relacyjny model danych

1.3.1. Model relacyjny według E.F. Codda

Twórcą teorii relacyjnych baz danych jest Edgar Frank Codd. W 1970 roku opublikował on pracę, która wprowadzała główne założenia dotyczące modelu relacyjnego dla danych, później uszczegółowione o terminy: algebra relacji oraz rachunek relacyjny.

Definicja relacji według E.F. Codda

Relacja *r* to dowolny podzbiór iloczynu kartezjańskiego jednego lub więcej zbiorów:

$r \subset D_1 \times D_2 \times \ldots \times D_k$

$D_1 \times D_2 \times \ldots \times D_k = \{(a_1, a_2, \ldots, a_k) : a_i \in D_i, i \in \{1, 2, \ldots, k\}\}$

Schematem R relacji nazywamy zbiór atrybutów $\{A_1, \ldots, A_n\}$.

Relacją r o schemacie $R = \{A_1, \ldots, A_n\}$ nazywamy skończony zbiór $r = \{t_1, \ldots, t_m\}$ odwzorowań $ti : R \rightarrow D$, gdzie D jest równe sumie dziedzin atrybutów $A_1, \ldots, A_n$ takich, że $t_i(A_j) \in D_j$ dla $i = 1, \ldots, m$, $j = 1, \ldots, n$.

Każde takie odwzorowanie nazywamy krotką.

W relacyjnym modelu danych relacje mogą być reprezentowane w postaci tabel.

Klucz

Kluczem schematu R relacji nazywamy taki zbiór atrybutów K tego schematu, że przez wartości atrybutów z tego zbioru można jednoznacznie zidentyfikować każdą krotkę.

Właściwości klucza:

- Wartość klucza pozwala jednoznacznie identyfikować krotki.
- Dany schemat może posiadać kilka kluczy.
- Każdy nadzbiór klucza jest kluczem.
- Klucz, którego żaden podzbiór właściwy nie jest kluczem, nazywa się kluczem właściwym lub kandydującym.

Wśród kluczy wybiera się jeden, który staje się kluczem głównym (podstawowym).

Integralność danych

W modelu relacyjnym danych występują następujące rodzaje więzów integralności:

- *Integralność encji* — każdy schemat relacji posiada klucz główny i żaden element klucza głównego nie może posiadać wartości pustej (*NULL*).
- *Integralność referencyjna* — każda wartość klucza obcego jest równa wartości klucza właściwego określonej krotki w relacji nadrzędnej lub wynosi *NULL*.
- *Więzy ogólne* — dodatkowe warunki dotyczące poprawności danych określane przez użytkowników lub administratorów baz danych.

Algebra relacji

Algebra relacji zawiera zbiór jawnych operacji, które pozwalają na tworzenie wymaganych relacji z relacji dostępnych w bazie danych.

Selekcja to wybór tych krotek relacji, które spełniają określone warunki.

Projekcja to wybór z relacji określonych atrybutów.

Złączenie to utworzenie z danych zawartych w dwóch relacjach jednej relacji. Najczęściej łączone są relacje, które mają taki sam atrybut.

Iloczynem kartezjańskim dwóch relacji — relacji r o schemacie $R = \{A_1, \ldots, A_k\}$ oraz relacji s o schemacie $S = \{B_1, \ldots, B_m\}$ takich, że $R \cap S = \emptyset$ — nazywamy relację $q = r \times s$ o schemacie $Q = R \cup S$ taką, że

$t \in q \Leftrightarrow (\exists u \in r)(t \mid R = u) \wedge (\exists v \in s)(t \mid S = v)$

$t \in q \Leftrightarrow t \mid R \in r \wedge t \mid S \in s.$

Jednym z kluczowych problemów relacyjnego modelu danych było podejście do brakującej informacji (na przykład nieznany numer telefonu lub numer domu). Ostatecznie Codd wprowadził do modelu relacyjnego dodatkową wartość *NULL*. Spowodowało to rozszerzenie logiki dwuwartościowej operatorów porównania (*Tak*, *Nie*) do logiki trójwartościowej (na każde pytanie można odpowiedzieć: *Tak*, *Nie*, *Nieznane*). Wartość *NULL* określa wartość atrybutu, która w danej chwili nie jest znana lub nie może zostać ustalona.

1.3.2. Relacyjny model baz danych

Cechy relacyjnego modelu baz danych:

- Podstawową formą przechowywania danych jest tabela.
- Poprzez użycie kombinacji „wartość klucza podstawowego, nazwa tabeli i nazwa kolumny" musi istnieć dostęp do dowolnych danych.
- Musi być obsługiwana wartość *NULL* (wartość *NULL* przedstawia brakujące lub bezużyteczne informacje, na przykład nieznany numer telefonu).
- Integralność danych powinna być naturalną cechą projektu bazy danych.

Korzyści płynące z używania modelu relacyjnego:

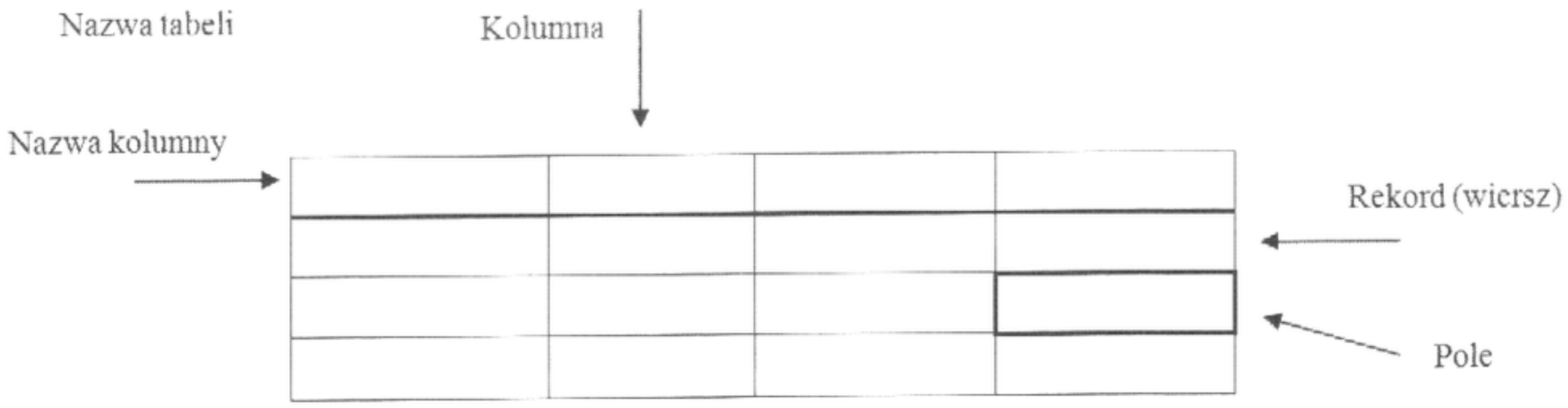
- efektywność przechowywania danych,
- pewność integralności danych,
- możliwość rozbudowy bazy danych,
- możliwość łatwej zmiany w strukturze bazy danych,
- zwiększenie szybkości dostępu do danych.

Istnieje wiele różnych podejść do relacyjnego modelu baz danych. Dwa główne to podejście formalne (opis relacyjnego modelu baz danych przez reguły matematyczne) oraz podejście intuicyjne (podejście do relacyjnego modelu baz danych w sposób czysto użytkowy).

Tabele

W modelu relacyjnym baz danych wszystkie dane są przechowywane w dwuwymiarowych tabelach (relacjach). W skład bazy danych może wchodzić wiele relacji.

Tabela to zbiór powiązanych ze sobą danych. Jest to układ poziomych wierszy, nazywanych rekordami lub krotkami, i pionowych kolumn, nazywanych polami rekordu lub atrybutami. Tabela jest identyfikowana poprzez nazwę (rysunek 1.4).



Rysunek 1.4. Struktura tabeli

Przecięcie kolumny i wiersza tworzy pole. Nazwa kolumny jest jednocześnie nazwą pola. Pojęcia „pole" używa się również do określenia kolumny.

Klucz podstawowy

W każdej tabeli musi znaleźć się pole, które dla każdego rekordu będzie przyjmowało inną, niepowtarzalną wartość. Pole takie jest potrzebne do jednoznacznego zdefiniowania rekordu. To klucz podstawowy, nazywany również kluczem głównym lub pierwotnym.

DEFINICJA

Klucz podstawowy jest to minimalna kombinacja pól identyfikująca każdy rekord w tabeli w sposób jednoznaczny.

Klucz podstawowy pozwala w sposób efektywny przeszukiwać i odczytywać dane w bazie oraz łączyć dane zapisane w różnych tabelach.

Klucz podstawowy nie może zawierać powtarzających się danych oraz nie może być pusty. Oznacza to, że w tabeli musi się znaleźć jedno lub kilka pól, które pozwolą odróżniać dane zapisane w jednym rekordzie od danych zapisanych w innym rekordzie.

Jeżeli tabela zawiera dane osobowe, takim kluczem może być kolumna z nazwiskiem. Każdą osobę zapisaną w tabeli rozpoznamy po nazwisku i odczytamy jej dane z właściwego rekordu. Co jednak zrobić, jeżeli w tabeli zapisaliśmy informacje o kilku osobach, które mają takie samo nazwisko?

Kluczem podstawowym może być kombinacja pól, czyli do pola *Nazwisko* możemy dodać pole *Imię*. Teraz każdą osobę w tabeli znajdziemy, podając jej nazwisko i imię. A co zrobić, jeśli w tabeli są umieszczone informacje o dwóch osobach, które mają identyczne nazwisko i imię? Można do klucza dodać kolejne pole, na przykład *Data urodzenia*.

W celu uniknięcia dodawania kolejnych pól do klucza bardzo często zastępuje się klucz podstawowy kluczem sztucznym. Najprostszym sposobem utworzenia klucza sztucznego jest dodanie do tabeli dodatkowego pola i umieszczenie w nim kolejnych numerów.

DEFINICJA

Klucz sztuczny to pole zawierające unikatowy numer identyfikacyjny nadany w sposób sztuczny każdemu obiektowi umieszczonemu w tabeli.

Jeżeli w tabeli zostało utworzone takie pole, staje się ono automatycznie kluczem podstawowym, ponieważ spełnia wszystkie wymagania dotyczące tego klucza.

Klucz obcy to jedno pole lub więcej pól tabeli (kolumn), które odwołują się do pola lub pól klucza podstawowego w innej tabeli. Klucz obcy pokazuje, w jaki sposób tabele są powiązane. Jest niezbędny do zdefiniowania połączenia między tabelami.

Relacje

Projektując bazę danych, dzielimy dane na wiele tabel tematycznych, tak aby każda informacja została zapisana tylko raz. Aby zestawić razem dane zapisane w różnych tabelach, tworzy się między nimi połączenia zwane relacjami.

DEFINICJA

Relacja jest to zdefiniowanie logicznego połączenia między tabelami bazy danych. W efekcie zmiana wiersza bieżącego w tabeli głównej powoduje automatyczną zmianę wiersza bieżącego w tabeli przyłączonej.

Typy relacji

W rzeczywistości (życiu codziennym) występują trzy typy relacji.

Relacja „jeden do jednego"

W relacji „jeden do jednego" każdemu rekordowi z pierwszej tabeli może odpowiadać tylko jeden rekord z drugiej tabeli i każdemu rekordowi z drugiej tabeli może odpowiadać tylko jeden rekord z pierwszej tabeli. Jest to nietypowy rodzaj relacji, ponieważ najczęściej informacje powiązane w ten sposób są przechowywane w jednej tabeli. Opisywanej relacji używa się w celu odizolowania części tabeli ze względu na bezpieczeństwo danych lub w celu podzielenia danych na podzbiory.

Relacja „wiele do jednego"

W relacji „wiele do jednego" każdemu rekordowi z pierwszej tabeli może odpowiadać najwyżej jeden rekord z drugiej tabeli, a każdemu rekordowi z drugiej tabeli może odpowiadać wiele rekordów z pierwszej tabeli. Jest to typ relacji najczęściej występujący w relacyjnych bazach danych.

Relacja „wiele do wielu"

W relacji „wiele do wielu" każdemu rekordowi z pierwszej tabeli może odpowiadać wiele rekordów z drugiej tabeli i każdemu rekordowi z drugiej tabeli może odpowiadać wiele rekordów z pierwszej tabeli.

1.4. Projektowanie bazy danych

W bazie danych przechowujemy tylko niektóre informacje o świecie rzeczywistym. Wybór właściwych wycinków rzeczywistości i dotyczących ich danych jest bardzo istotny — od niego zależy prawidłowe działanie bazy. Aby ten wybór był właściwy, należy wskazać informacje, które powinny być przechowywane w bazie danych, oraz określić ich strukturę.

1.4.1. Zasady projektowania bazy danych

Cały proces projektowania bazy danych możemy podzielić na kilka etapów:

- planowanie bazy danych,
- tworzenie modelu konceptualnego (diagramu ERD),
- transformacja modelu konceptualnego na model relacyjny,
- proces normalizacji bazy danych,
- wybór struktur i określenie zasad dostępu do bazy danych.

1.4.2. Podstawowe pojęcia

Z punktu widzenia relacyjnej bazy danych świat rzeczywisty widzimy i analizujemy jako zestaw **encji** i związków zachodzących między nimi.

Encja

DEFINICJA

Encją jest każdy przedmiot, zjawisko, stan lub pojęcie, czyli każdy obiekt, który potrafimy odróżnić od innych obiektów (na przykład: osoba, samochód, książka, stan pogody).

Encje podobne do siebie (opisywane za pomocą podobnych parametrów) grupujemy w zbiory encji. Projektując bazę danych, należy precyzyjnie zdefiniować encje i określić parametry, przy użyciu których będą opisywane.

Atrybut

Encje mają określone cechy wynikające z ich natury. Cechy te nazywamy atrybutami. Zestaw atrybutów, które określamy dla encji, zależy od potrzeb bazy danych.

Dziedzina

Atrybuty encji mogą przyjmować różne wartości. Projektując bazę danych, możemy określić, jakie wartości może przyjmować dany atrybut. Zbiór wartości atrybutu nazywamy dziedziną (domeną).

1.4.3. Planowanie bazy danych

Prostym przykładem bazy danych jest lista klientów sklepu internetowego zajmującego się sprzedażą książek. Zawiera ona zbiór informacji o wycinku otaczającej nas rzeczywistości, czyli o klientach dostępnego w internecie sklepu oraz o oferowanych książkach. Encją jest klient. Zbiór klientów tworzy zbiór encji.

Klient jest encją i jako encja jest określany przez kilka atrybutów (nazwisko, imię, adres zamieszkania, PESEL). Dla każdego atrybutu można ustalić zbiór wartości, na przykład atrybut *Nazwisko* to zbiór wszystkich kombinacji liter alfabetu łacińskiego. W rzeczywistości zbiór nazwisk jest podzbiorem tego zbioru, ponieważ niektóre kombinacje liter nigdy nie wystąpią w nazwisku.

Podobnie książka jest encją i jest określana przez atrybuty — tytuł, autor, cena, rok wydania, wydawnictwo, miejsce wydania, język.

Ocena, które z atrybutów encji są istotne, wynika z przeznaczenia bazy danych. Należy zwrócić uwagę, czy w bazie nie zostały umieszczone atrybuty, które tylko pozornie należą do danej encji, a naprawdę są atrybutami innej encji. Załóżmy, że w bazie danych trzeba określić narodowość autora książki. Jeżeli atrybut *Narodowość* zostanie dodany do encji *Książka*, nastąpi pomieszanie atrybutów, ponieważ narodowość jest atrybutem autora, a nie książki.

Można wymienić następujące niekorzystne skutki wspomnianego pomieszania informacji:

- Narodowość autora jest powtórzona tyle razy, ile książek tego autora znajdzie się w bazie.
- Jeżeli w bazie nie została umieszczona jeszcze żadna książka danego autora, nie ma o nim informacji w tej bazie.
- Znalezienie informacji o dowolnym autorze staje się wyjątkowo żmudne.
- Występują problemy z poprawianiem błędów, które popełniamy w czasie wprowadzania danych.

Ponieważ trudno uniknąć błędów podczas wprowadzania danych, należy odpowiednio zabezpieczyć się przed nimi na etapie projektowania bazy danych. Jeśli na przykład pojawią się różne narodowości dla tego samego autora, jak rozstrzygnąć, która jest właściwa? Jak usunąć ten błąd?

Właściwym postępowaniem jest zaprojektowanie oddzielnej encji dla autora. Każdy autor w tym zbiorze encji występuje raz, a jego atrybutami oprócz nazwiska i imienia mogą być: narodowość, okres tworzenia, rodzaj twórczości. Oto zalety tego rozwiązania:

- Wyszukiwanie autora i jego danych jest proste.
- Każdy autor występuje na liście tylko raz i jeżeli nawet jego książek na razie nie ma w bazie, to autor pojawia się na liście.
- Ewentualne błędy występują tylko raz i łatwo je poprawić.

Należy pamiętać, że prawidłowa klasyfikacja danych jest podstawą dobrego projektu bazy danych. Najgorsze efekty otrzymujemy, próbując zaprojektować bazę danych, która będzie gromadziła wszystkie możliwe dane i przetwarzała je na wszystkie możliwe sposoby.

Każda encja powinna mieć przynajmniej jeden atrybut lub kombinację kilku atrybutów, które identyfikują ją jednoznacznie. Ten atrybut to **klucz podstawowy** encji.

Projektując bazę danych na przykład dla klientów sklepu, widzimy, że tym atrybutem będzie nazwisko klienta. Aby uniknąć niejasności, gdy w bazie pojawi się kilku klientów o tym samym nazwisku, a czasami również imieniu, możemy przyjąć, że kluczem podstawowym będzie kombinacja atrybutów *Nazwisko*, *Imię* oraz *PESEL*. Jeśli klucz podstawowy składa się z kilku atrybutów, dobrym rozwiązaniem jest zastąpienie go **kluczem sztucznym**. Najczęściej zawiera on unikatowe liczby przypisane kolejnym encjom.

Projektowanie modelu bazy danych powinno składać się z następujących działań:

- określenie występujących zbiorów encji,
- określenie atrybutów przypisanych do poszczególnych encji,
- określenie dziedziny poszczególnych atrybutów,
- ustalenie kluczy podstawowych,
- określenie typów występujących związków,
- zweryfikowanie utworzonego modelu.

1.4.4. Diagramy związków encji (diagramy ERD)

Konceptualne projektowanie bazy danych to konstruowanie schematu danych niezależnego od wybranego modelu danych, docelowego systemu zarządzania bazą danych, programów użytkowych czy języka programowania.

Do tworzenia modelu graficznego schematu bazy danych wykorzystywane są diagramy związków encji, z których najpopularniejsze są diagramy ERD (ang. *Entity Relationship Diagram*). Pozwalają one na modelowanie struktur danych oraz związków zachodzących między tymi strukturami. Nadają się szczególnie do modelowania relacyjnych baz danych, ponieważ umożliwiają prawie bezpośrednie przekształcenie diagramu w schemat relacyjny.

Diagramy ERD składają się z trzech rodzajów elementów:

- zbiorów encji,
- atrybutów encji,
- związków zachodzących między encjami.

Encja to reprezentacja obiektu przechowywanego w bazie danych. Graficzna reprezentacja różnych encji została pokazana na rysunku 1.5.

Rysunek 1.5. Graficzna reprezentacja encji

Atrybut opisuje encję. Może on być liczbą, tekstem lub wartością logiczną. Graficzna reprezentacja atrybutów dla encji *Klient* została pokazana na rysunku 1.6.

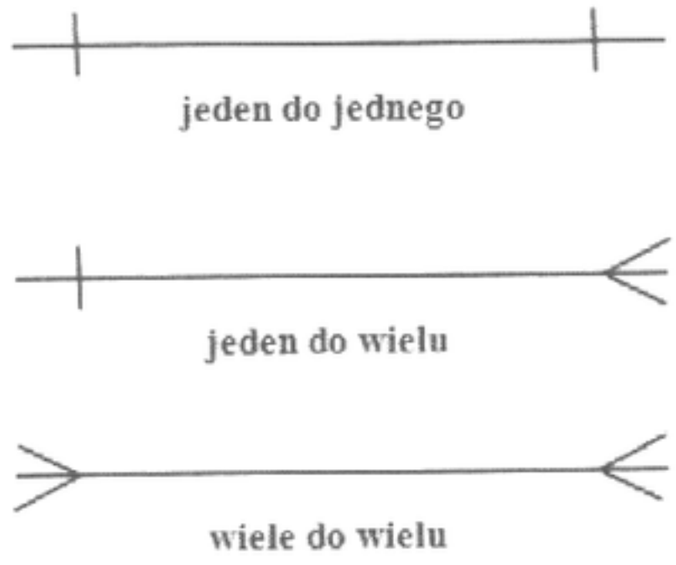
Rysunek 1.6. Atrybuty encji Klient

Związek jest to powiązanie między dwiema lub kilkoma encjami. Każdy związek ma dwa końce, do których są przypisane następujące atrybuty:

- nazwa,
- stopień związku,
- uczestnictwo lub opcjonalność związku. Atrybut ten określa, czy związek jest opcjonalny, czy wymagany.

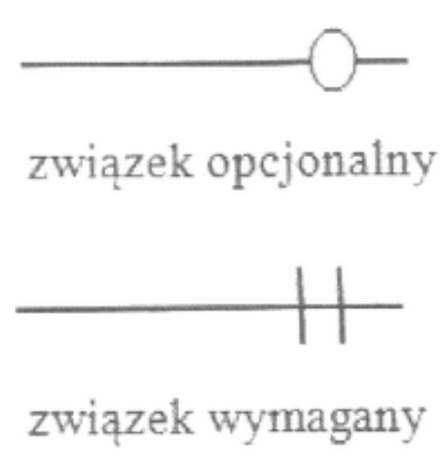
Opis reprezentacji graficznej stopnia związku został pokazany na rysunku 1.7.

Rysunek 1.7.
Graficzna reprezentacja związków zachodzących między encjami



Opis reprezentacji graficznej opcjonalności związku został pokazany na rysunku 1.8.

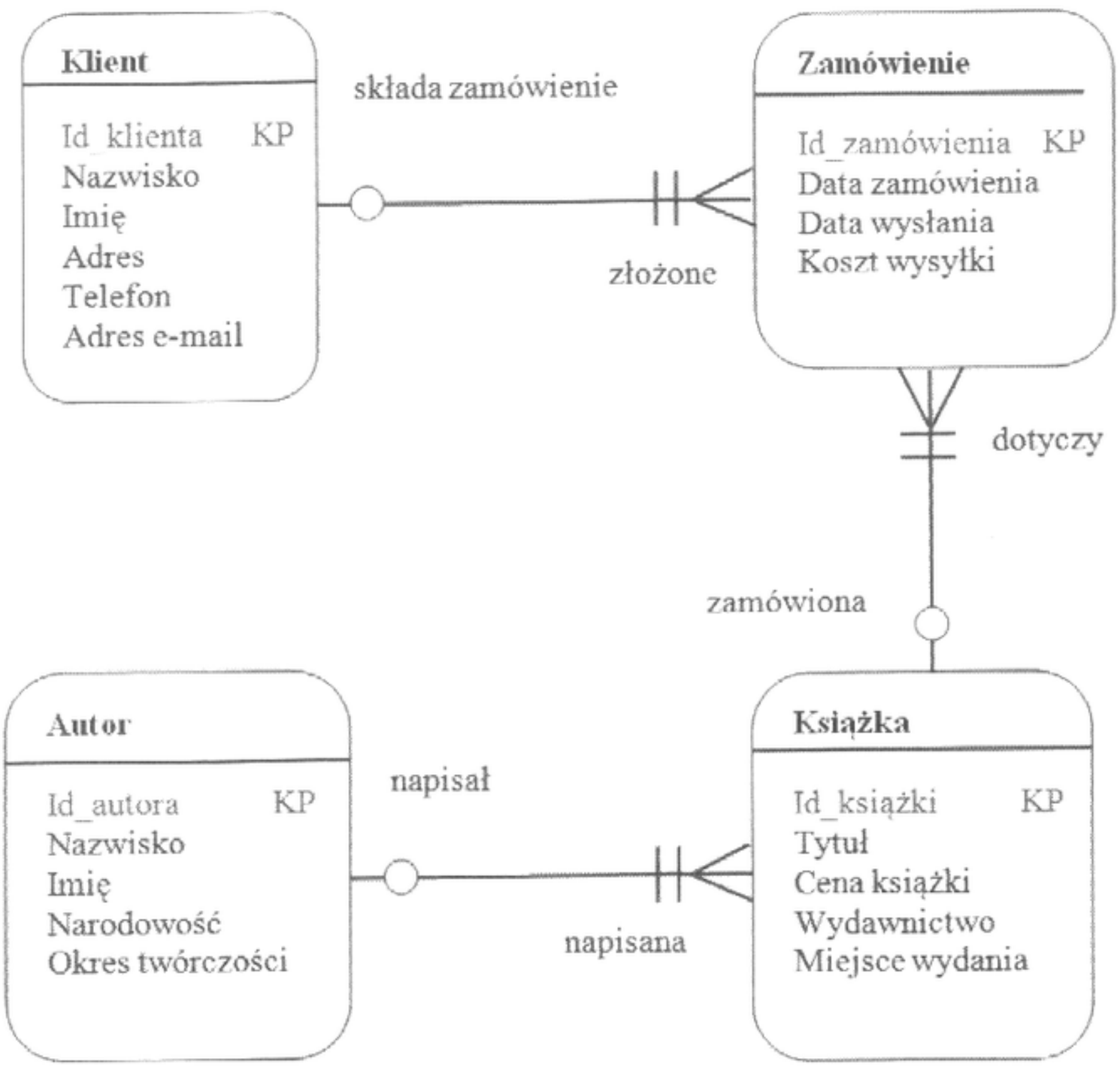
Rysunek 1.8.
Graficzna reprezentacja opcjonalności związku



Diagramy ERD spotyka się w wielu różnych notacjach, na przykład: Martina, Bachmana, Chena, IDEFIX. Istnieje wiele narzędzi wspomagających rysowanie diagramów ERD, ale jedynie w przypadku narzędzi klasy CASE (ang. *Computer Aided Software Engineering*) można mówić o określonej notacji.

Prosty przykład diagramu ERD w notacji Martina dla księgarni internetowej został przedstawiony na rysunku 1.9.

Rysunek 1.9.
Diagram ERD w notacji Martina dla księgarni internetowej



W pokazanym na rysunku schemacie encje zostały przedstawione za pomocą zaokrąglonych prostokątów zawierających listę atrybutów. Klucze główne (KP) zostały oznaczone kolorem czerwonym. Stopień związku i uczestnictwo zostały oznaczone liniami łączącymi z odpowiednimi symbolami opisującymi stopień oraz opcjonalność związku. Należy zwrócić uwagę na to, że w encjach nie umieszcza się kluczy obcych. Zostaną one dodane na etapie przekształcania encji na tabele.

Tak przygotowany diagram ERD pozwala na późniejszą weryfikację i optymalizację bazy danych, a także stanowi podstawową dokumentację projektowanej bazy danych. Można go również wykorzystać w jednym z narzędzi CASE do wygenerowania fizycznej struktury bazy danych.

1.4.5. Projektowanie bazy danych za pomocą narzędzi CASE

Narzędzia CASE (ang. *Computer Aided Software Engineering*) są wykorzystywane podczas projektowania różnego rodzaju oprogramowania, najczęściej wspomagają proces jego wytwarzania.

Narzędzia te pozwalają tworzyć modele graficzne odpowiadające konstrukcjom programistycznym. Wykorzystywane są tutaj edytory notacji graficznych, które dają możliwości tworzenia diagramów oraz powiązań między poszczególnymi elementami. Bardziej zaawansowane edytory umożliwiają przetwarzanie informacji i udostępnianie danych do aplikacji zewnętrznych, na przykład kodów w Visual Basicu, SQL-u, ODBC.

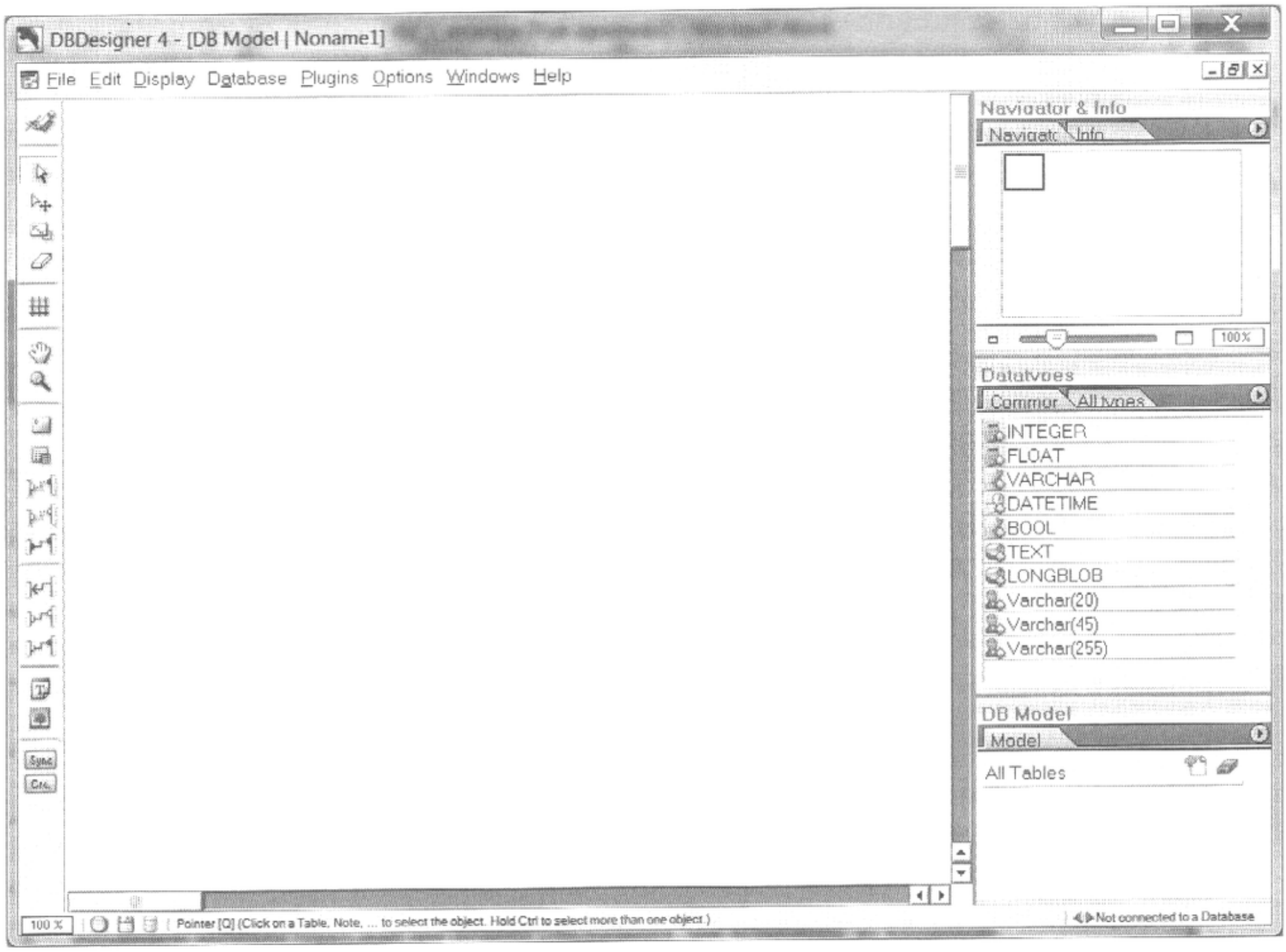
Narzędzia CASE mogą być stosowane do generowania kodu na podstawie zaprojektowanego modelu danych, można również za ich pomocą, na podstawie analizy kodu źródłowego, odtworzyć projekt i specyfikację bazy danych.

Przykładem narzędzia typu CASE jest program DBDesigner4. Jest to narzędzie do wizualnego projektowania, modelowania i tworzenia baz danych. Program został stworzony z myślą o bazie MySQL, ale obsługuje również bazy danych Oracle, SQLite, MS SQL. Jest rozpowszechniany jako open source i jest dostępny na stronie *http://fabforce.net/index.php*.

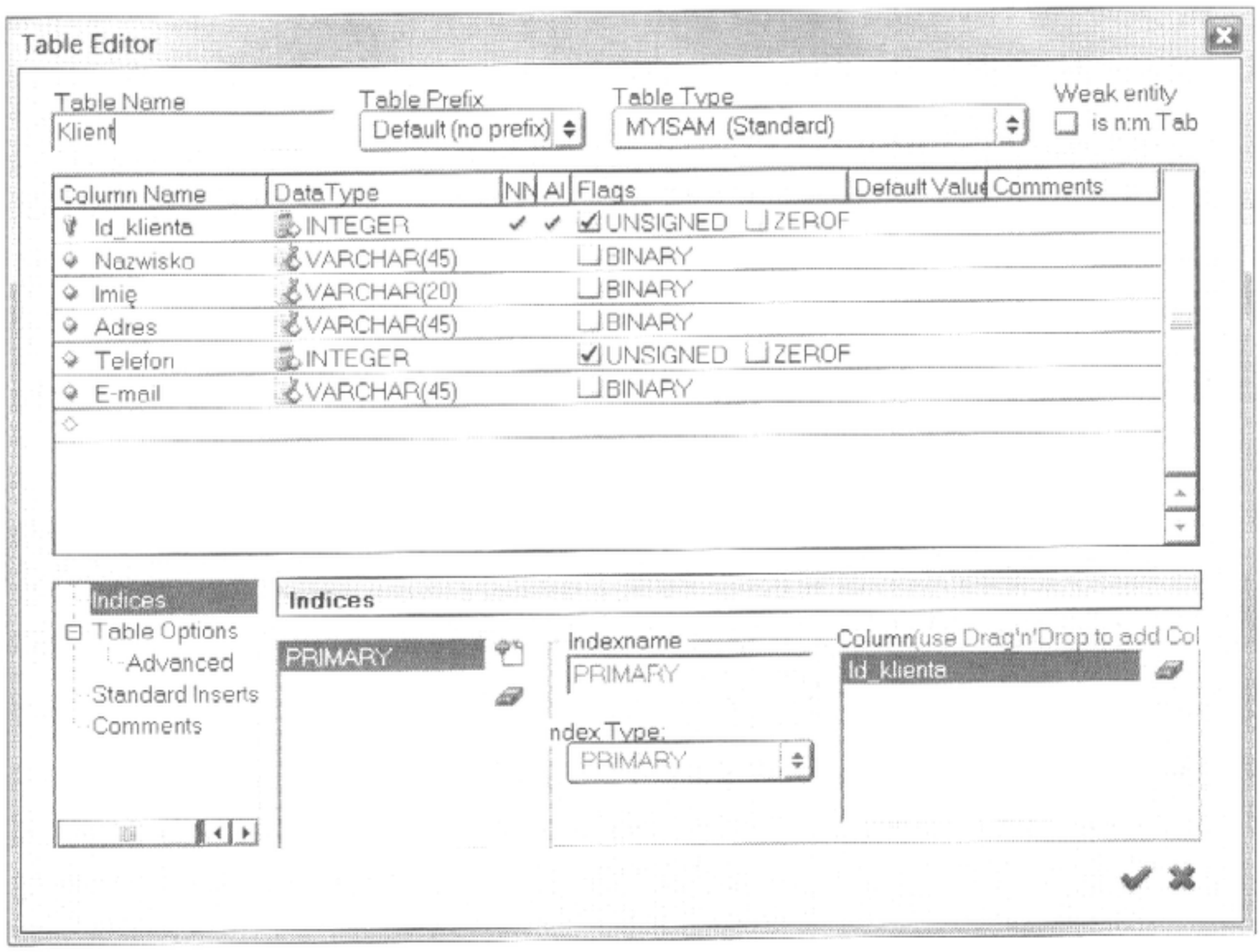
Zainstalowany program można wykorzystać do przygotowania logicznego projektu bazy danych.

Okno programu DBDesigner4 składa się z pięciu obszarów (rysunek 1.10). Pusty obszar na środku ekranu to obszar roboczy. Z lewej strony znajduje się pasek narzędzi. Z prawej w górnej części znajduje się okno nawigacji i informacji, na środku okno typów danych, a w dolnej części okno bieżącego modelu bazy danych.

Pracę w programie rozpoczynamy od utworzenia nowego projektu (menu *File/New*). Kolejnym etapem jest utworzenie pierwszej tabeli. Należy wybrać na pasku narzędzi ikonę i kliknąć obszar roboczy. W efekcie zostanie utworzona tabela. Po dwukrotnym kliknięciu tabeli można otworzyć okno jej edytowania, w którym należy wprowadzić nazwę tabeli, nazwy pól oraz wybrać ich typy (rysunek 1.11).



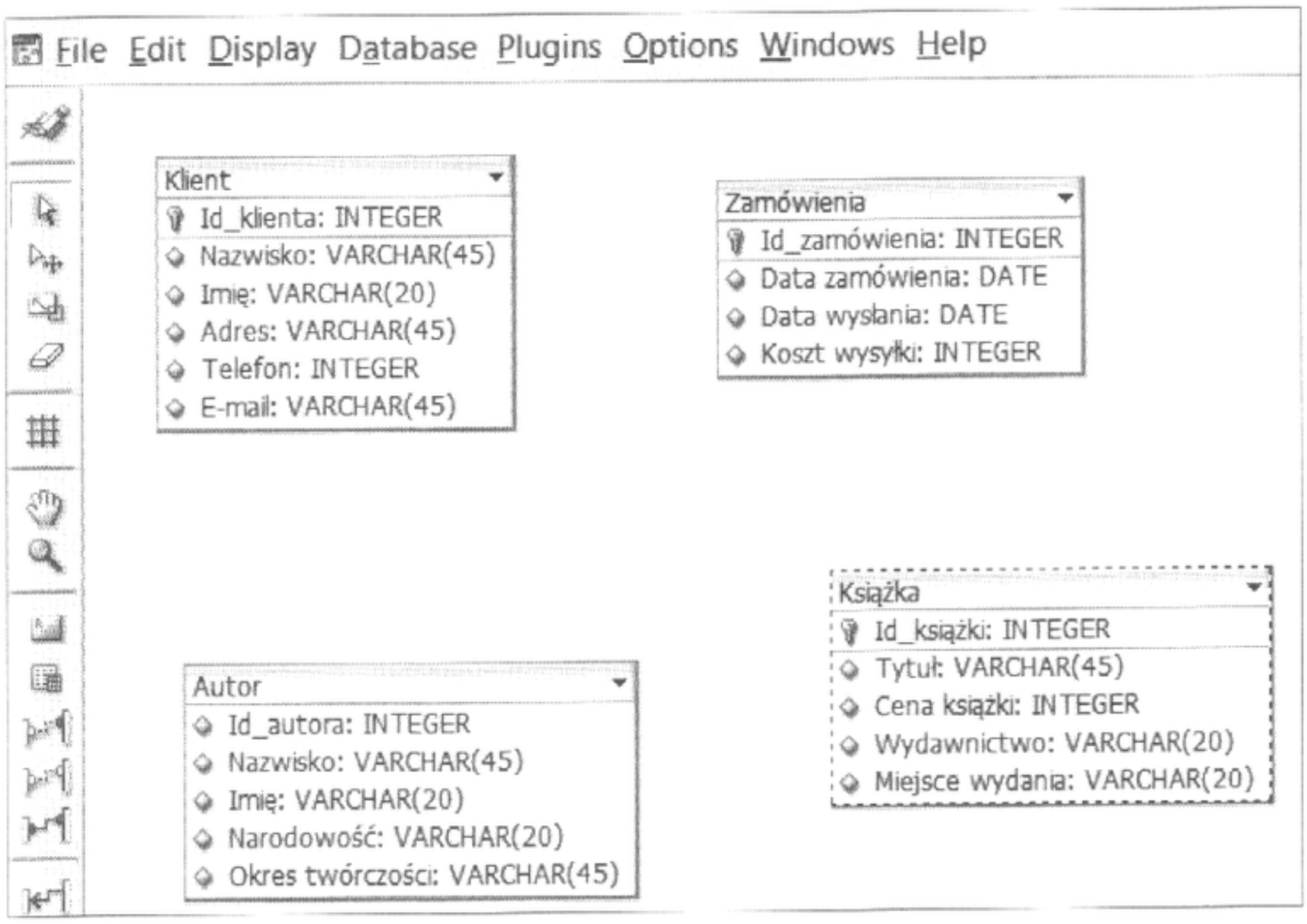
Rysunek 1.10. Okno programu DBDesigner4



Rysunek 1.11. Okno edytowania tabeli

W oknie edytowania tabeli w kolumnie *Column Name* należy wprowadzić nazwę tworzonego pola, w kolumnie *DataType* określić typ danych, w kolumnie *NN* określić, czy dozwolona jest wartość *NULL* (*NOT NULL*), w kolumnie *AI* zaznaczyć automatyczne zwiększanie wartości o 1 (*AUTO INCREMENT*), w kolumnie *Flags* zdefiniować dodatkowe opcje zależne od typu danych, w kolumnie *Default Value* ustawić wartość domyślną pola, a w kolumnie *Comments* wstawić komentarz.

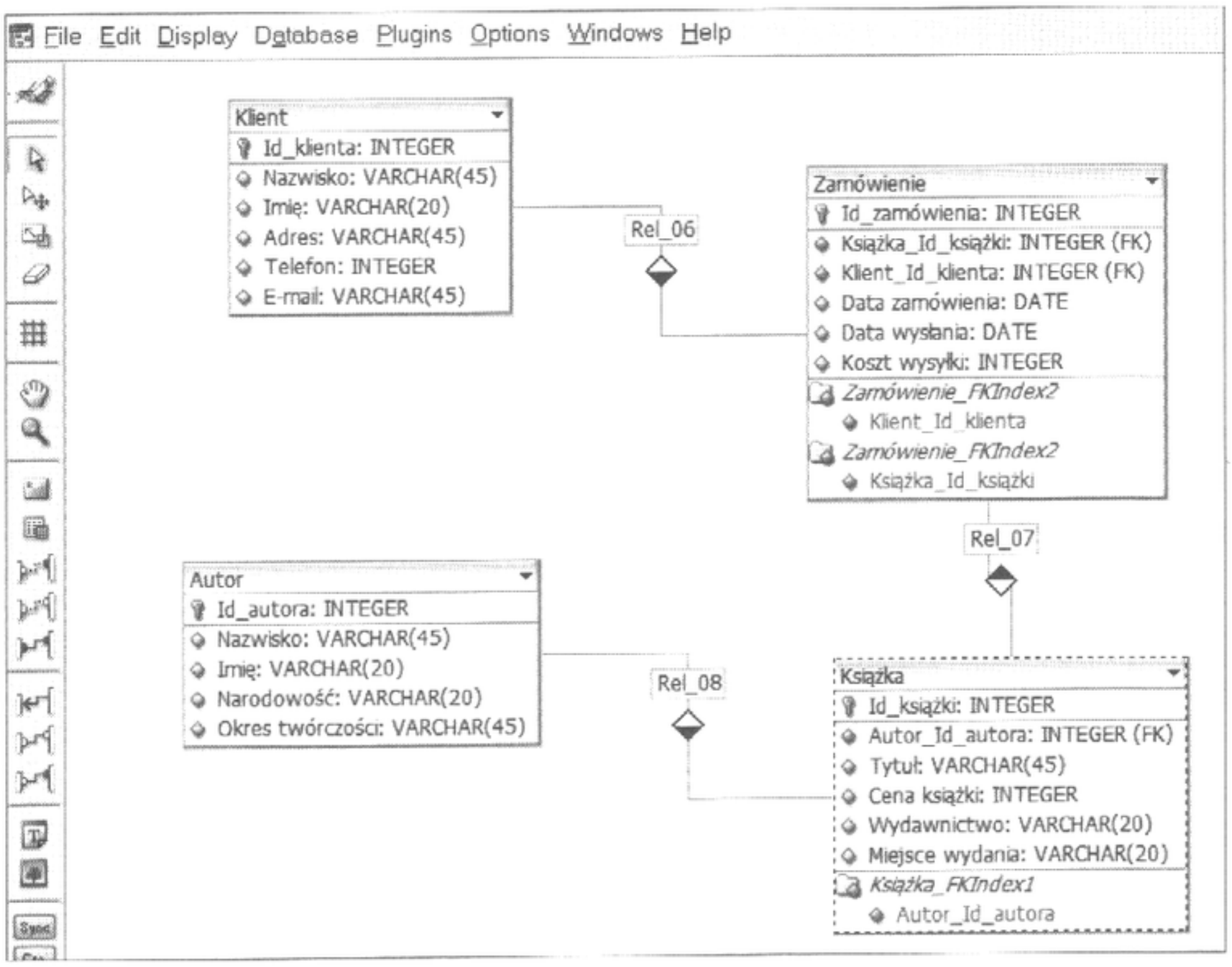
Dla projektowanego modelu graficznego bazy danych w podobny sposób należy utworzyć pozostałe tabele (rysunek 1.12).



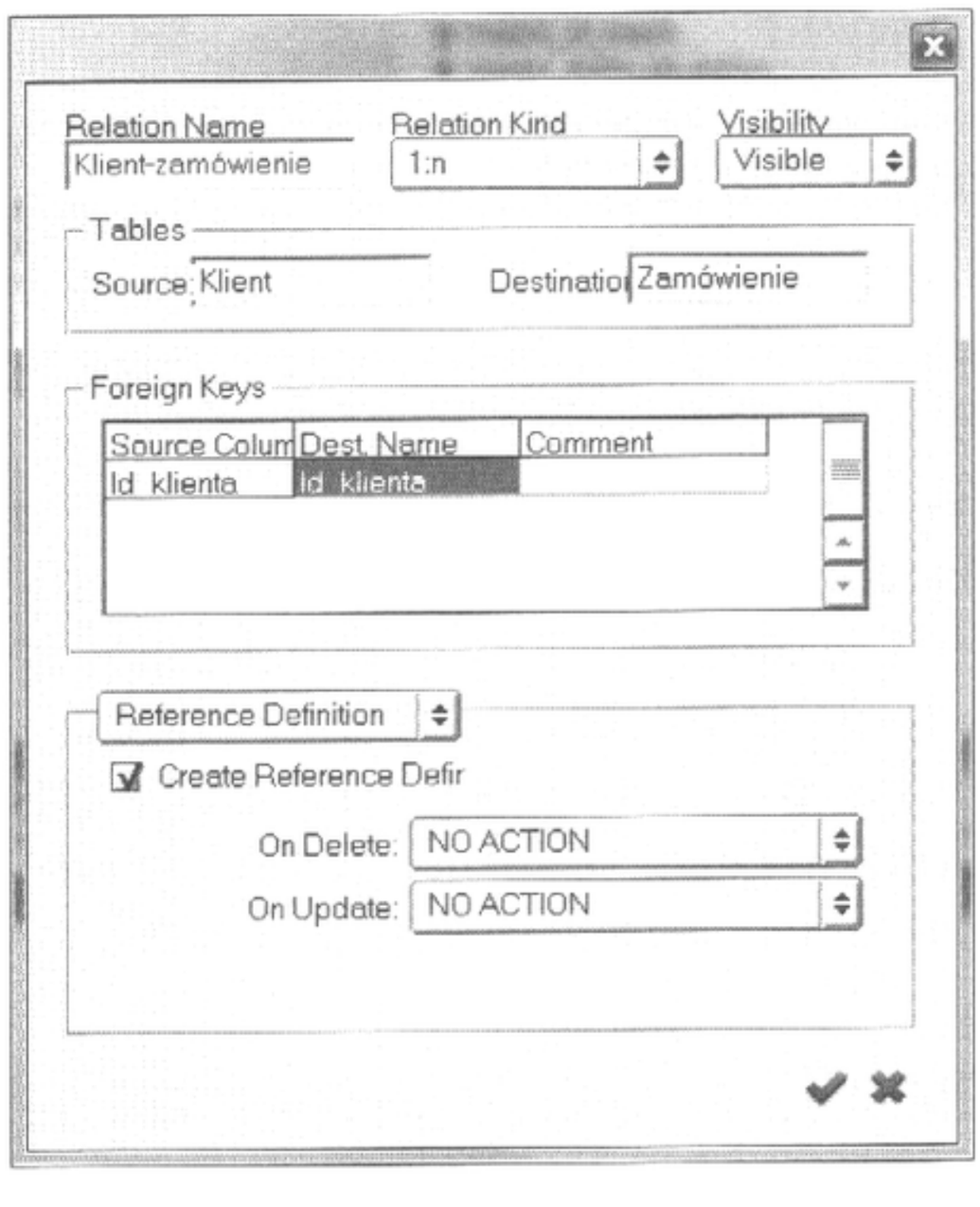
Rysunek 1.12. Tabele zaprojektowane w programie DBDesigner4

Po utworzeniu wszystkich tabel należy zdefiniować połączenia między nimi. Program DBDesigner4 obsługuje wszystkie rodzaje połączeń występujących w bazie danych. Ikony odpowiednich połączeń są dostępne na pasku narzędzi. Aby dodać połączenie typu „jeden do wielu", należy wybrać ikonę . Po wybraniu ikony rodzaju połączenia klikamy najpierw tabelę ze strony „jeden" (*Klient*), a następnie tabelę ze strony „wiele" (*Zamówienie*). W wyniku zdefiniowania połączenia w tabeli ze strony „wiele" (*Zamówienie*) pojawiło się nowe pole (*Klient_Id_klienta*), opisujące związek między tabelami, które stanie się kluczem obcym (rysunek 1.13).

Aby edytować utworzone połączenie, należy dwukrotnie kliknąć narysowaną linię. Zostanie otwarte okno edytowania połączenia, w którym można zmienić nazwę relacji oraz nazwę pola klucza obcego (rysunek 1.14).

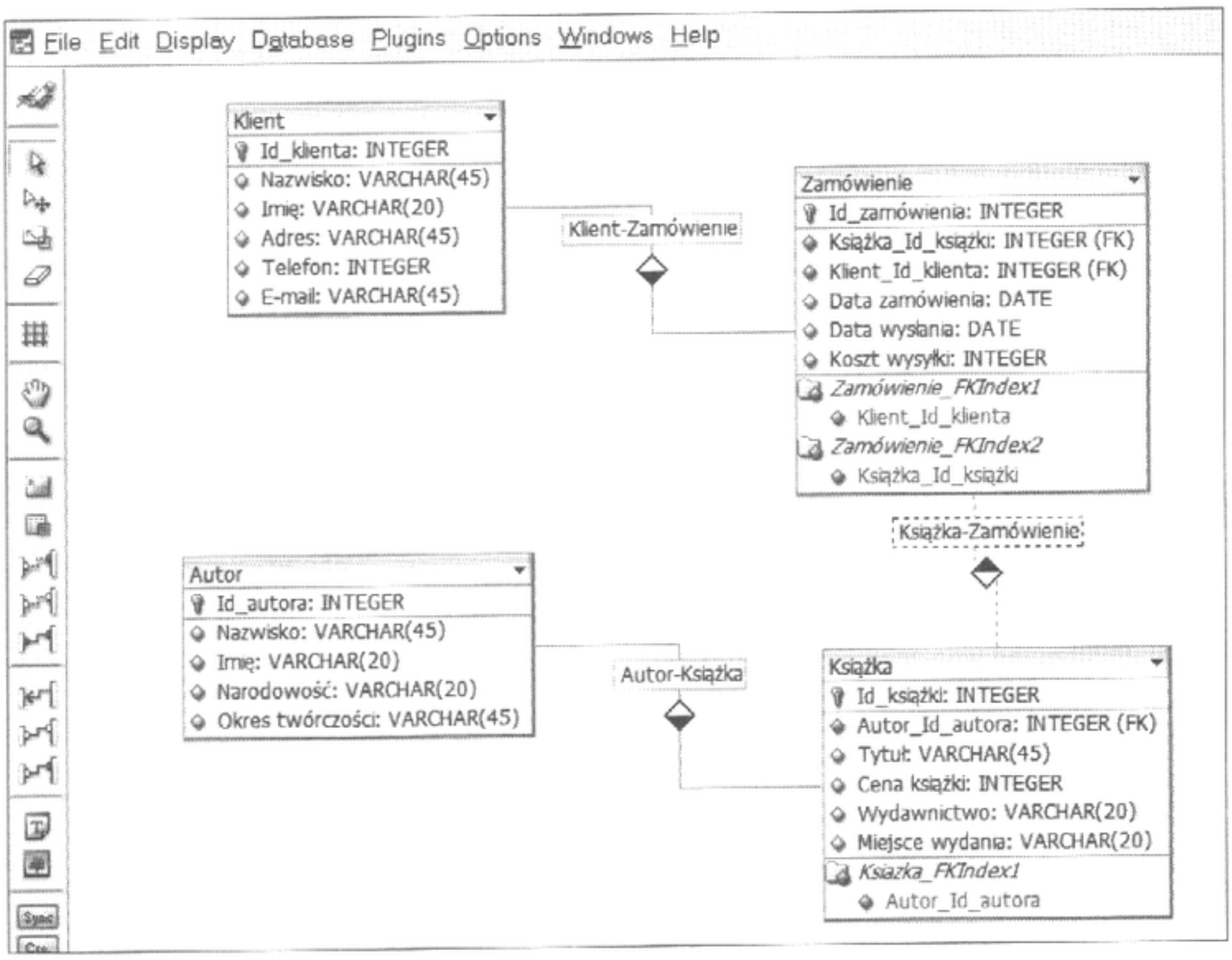


Rysunek 1.13. Definiowanie połączeń między tabelami



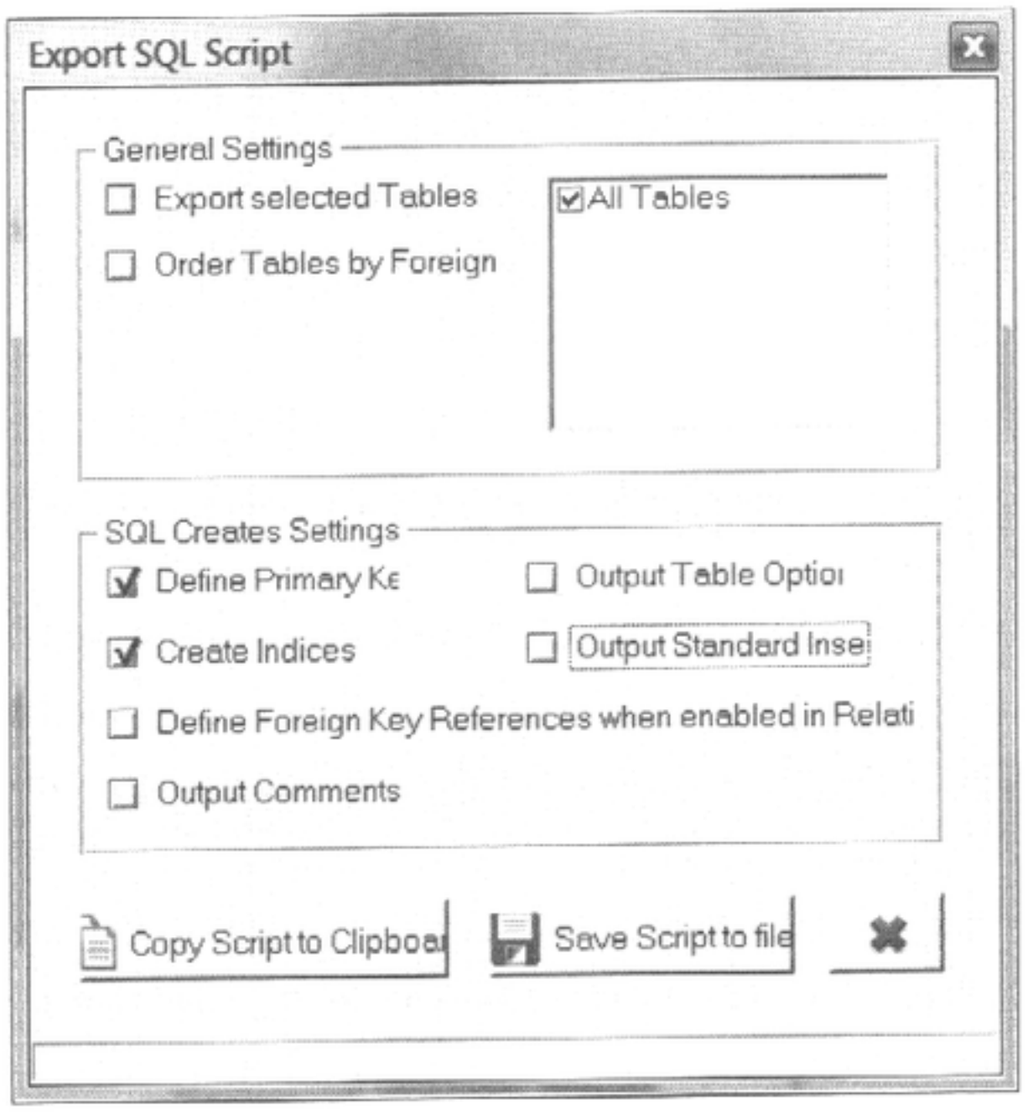
Rysunek 1.14. Okno edytowania połączenia

W podobny sposób należy zdefiniować wszystkie połączenia istniejące w bazie danych. Po utworzeniu połączeń uzyskamy efekt podobny do pokazanego na rysunku 1.15.



Rysunek 1.15. Schemat bazy danych uzyskany w programie DBDesigner4

Utworzony projekt należy zapisać w pliku, wybierając z menu *File/Save*. Można również zaprojektowaną bazę danych wyeksportować do pliku SQL. W tym celu należy wybrać z menu *File/Export/SQL Create Script* i w otwartym oknie zaznaczyć opcje, tak jak pokazano na rysunku 1.16, a następnie kliknąć przycisk *Save Script to file*.



Rysunek 1.16. Opcje eksportowania projektu bazy do kodu SQL

Po wykonaniu tych czynności zostanie wygenerowany skrypt, którego zawartość można zobaczyć, otwierając plik na przykład w edytorze tekstowym Notatnik (rysunek 1.17).

Schemat bazy_skrypt - Notatnik

Plik Edycja Format Widok Pomoc

```
CREATE TABLE Autor (
  Id_autora INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nazwisko VARCHAR(45) NULL,
  Imię VARCHAR(20) NULL,
  Narodowość VARCHAR(20) NULL,
  Okres twórczości VARCHAR(45) NULL,
  PRIMARY KEY(Id_autora)
);

CREATE TABLE Klient (
  Id_klienta INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nazwisko VARCHAR(45) NULL,
  Imię VARCHAR(20) NULL,
  Adres VARCHAR(45) NULL,
  Telefon INTEGER UNSIGNED NULL,
  E-mail VARCHAR(20) NULL,
  PRIMARY KEY(Id_klienta)
);

CREATE TABLE Książka (
  Id_książki INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Autor_Id_autora INTEGER UNSIGNED NOT NULL,
  Tytuł VARCHAR(45) NULL,
  Cena książki INTEGER UNSIGNED NULL,
  Wydawnictwo VARCHAR(20) NULL,
  Miejsce wydania VARCHAR(20) NULL,
  PRIMARY KEY(Id_książki),
  INDEX Książka_FKIndex1(Autor_Id_autora)
);

CREATE TABLE Zamówienie (
  Id_zamówienia INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Książka_Id_książki INTEGER UNSIGNED NOT NULL,
  Klient_Id_klienta INTEGER UNSIGNED NOT NULL,
  Data zamówienia DATE NULL,
  Data wysłania DATE NULL,
  Koszt wysyłki INTEGER UNSIGNED NULL,
  PRIMARY KEY(Id_zamówienia),
  INDEX Zamówienie_FKIndex1(Klient_Id_klienta),
  INDEX Zamówienie_FKIndex2(Książka_Id_książki)
);
```

Rysunek 1.17. Automatycznie wygenerowany kod SQL

Tak przygotowany skrypt może zostać uruchomiony na jednym z serwerów bazodanowych jako gotowa baza danych. Może również zostać wykorzystany do połączenia z wybranym serwerem bazodanowym.

1.4.6. Projektowanie bazy danych

Jeżeli w projektowanej bazie danych występuje kilka zbiorów encji (na przykład encje *Książki* i *Autor*), między tymi encjami zachodzą związki. Każda książka jest dziełem określonego autora, każdy autor ma przypisany zbiór książek. Zarówno zbiory encji, jak i związki zachodzące między encjami zapisujemy w tabelach projektowanej bazy danych. Przy projektowaniu tabel stosujemy reguły, które określają, w jaki sposób należy przekształcać opis świata rzeczywistego w schemat bazy danych.

Reguły projektowania tabel

1. Do opisu każdego zbioru podobnych encji stosuje się oddzielną tabelę. Jednej encji odpowiada jeden wiersz. Atrybutowi odpowiada kolumna. Dla każdego atrybutu określa się typ informacji.
2. Do opisu każdego dwustronnego związku między encjami można użyć oddzielnej tabeli. Kolumny tabeli są tworzone z kluczy encji należących do związku.

3. Zapis związku **jeden do jednego** lub **wiele do jednego** może być umieszczony w dodatkowych kolumnach tabel pozostających w związku (nie trzeba tworzyć oddzielnej tabeli do opisu tego związku). W przypadku związku **jeden do jednego** kolumna ta może znaleźć się w dowolnej tabeli, w przypadku związku **wiele do jednego** musi znaleźć się w tabeli ze strony „wiele". Dołączona kolumna zawiera klucz encji, z którą zachodzi związek.
4. Związek **wiele do wielu** opisuje się w oddzielnej tabeli, której kolumny tworzone są z kluczy encji należących do związku.
5. Jeśli klucze w tabeli opisującej związek składają się z wielu atrybutów lub są długie, należy zastąpić je kluczami sztucznymi.

1.4.7. Normalizacja tabel

Normalizację stosuje się, aby sprawdzić, czy zaprojektowane tabele mają prawidłową strukturę. Proces normalizacji rozpoczynamy, gdy zostanie utworzony wstępny projekt tabel. Pozwala ona określić, czy informacje przewidziane w projekcie bazy zostały przydzielone do właściwych tabel. Natomiast nie da odpowiedzi na pytanie, czy projekt bazy danych jest prawidłowy.

Korzyści płynące z normalizacji tabel są następujące:

- zlikwidowanie problemu powtarzania danych,
- optymalizacja objętości bazy danych,
- optymalizacja efektywności obsługi bazy danych,
- minimalizacja zagrożenia błędami przy wprowadzaniu danych.

Stosowane są cztery reguły normalizacji, ale w większości projektów baz danych wystarczy sprawdzić trzy pierwsze. Zostaną one omówione poniżej.

Dla każdej z nich stosowane są określenia: pierwsza postać normalna (**I PN**), druga postać normalna (**II PN**) i trzecia postać normalna (**III PN**).

Pierwsza postać normalna

DEFINICJA

Tabela jest w pierwszej postaci normalnej (**I PN**), gdy pojedyncze pole tabeli zawiera informację elementarną.

Oznacza to, że w komórce tabeli nie może wystąpić lista wartości, na przykład w polu *Narodowość autora* nie można umieścić dwóch narodowości — polskiej i angielskiej.

Załóżmy, że w projektowanej bazie danych dla księgarni internetowej została zaprojektowana tabela *Realizacja zamówień* z polami: *Nazwisko klienta*, *Imię*, *Adres*, *Telefon*, *PESEL*, *Tytuł książki*, *Liczba egzemplarzy*, *Cena*. W polu *Książki* będą umieszczane tytuły książek zakupionych przez klienta. Gdy klient kupi dwie książki, w polu *Tytuł książki*

należałoby wpisać dwa tytuły. Powstałaby lista wartości (rysunek 1.18). Tak zaprojektowana tabela nie jest w **I PN**. Nie będzie możliwe prawidłowe przetwarzanie danych zapisanych w tabeli. Rozwiązaniem jest zapisanie informacji o zakupionych książkach w dwóch wierszach. W pierwszym wierszu w polu *Tytuł książki* należy wpisać tytuł pierwszej książki, w drugim — tytuł drugiej książki, natomiast nazwisko klienta zostanie powtórzone w liczbie wierszy równej liczbie zakupionych książek (rysunek 1.19). Teraz tabela jest w **I PN**.

Realizacja zamówień

Nazwisko kli	Imię	Adres	Telefon	PESEL	Tytuł książki	Liczł	Cena
Nowak	Marek	Toruń	(56) 6589234	79120307431	Dziady	2	20,00 zł
Kowalski	Adam	Warszawa	(22) 3451234	80122401871	Balladyna, Tango	1	15,00 zł
Górecki	Grzegorz	Poznań	(45) 2367897	82061203983	Pan Tadeusz	1	21,00 zł
Zan	Marcin	Gdańsk	(33) 8373635	82020201875			

Rysunek 1.18. Tabela nie jest w I PN, ponieważ w polu *Tytuł książki* pojawiła się lista wartości

Realizacja zamówień

Nazwisko klie	Imię	Adres	Telefon	PESEL	Tytuł książki	Liczł	Cena
Nowak	Marek	Toruń	(56) 6589234	79120307431	Dziady	2	20,00 zł
Kowalski	Adam	Warszawa	(22) 3451234	80122401871	Balladyna	1	15,00 zł
Kowalski	Adam	Warszawa	(22) 3451234	80122401871	Tango	2	18,00 zł
Górecki	Grzegorz	Poznań	(45) 2367897	82061203983	Pan Tadeusz	1	21,00 zł

Rysunek 1.19. Tabela jest w I PN. W polu *Tytuł książki* występują pojedyncze wartości

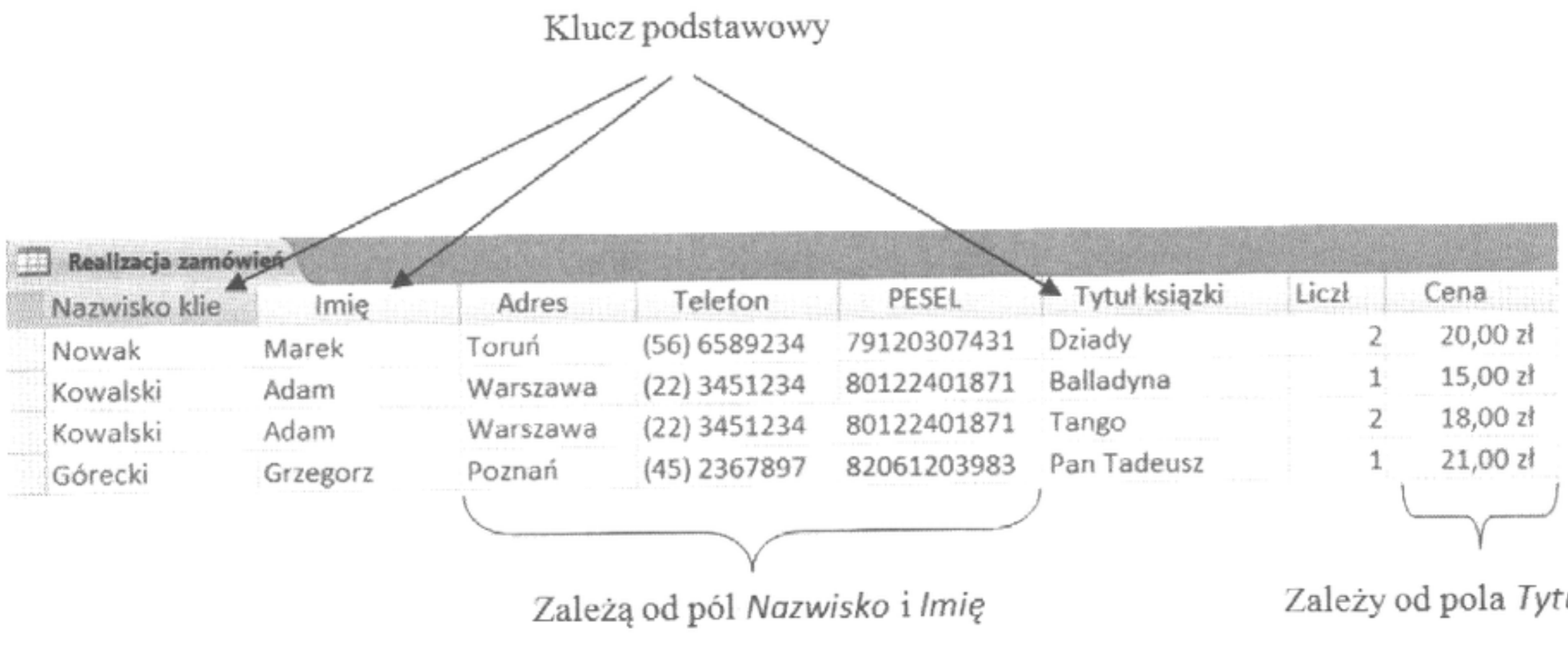
Druga postać normalna

DEFINICJA

Tabela jest w drugiej postaci normalnej (**II PN**), jeżeli jest w pierwszej postaci normalnej (**I PN**) oraz każde z pól niewchodzących w skład klucza podstawowego zależy od całego klucza, a nie od jego części.

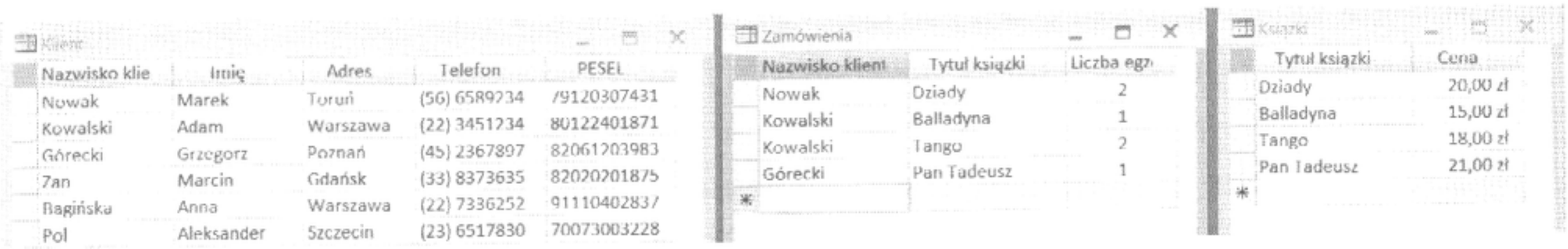
Ta reguła i następna służą do sprawdzenia, czy w tabeli i bazie danych nie doszło do redundancji, czyli niepotrzebnego powtarzania danych. Ponieważ **II PN** odnosi się do klucza podstawowego, należy określić ten klucz dla tabeli. Jeżeli klucz podstawowy składa się z jednego pola, tabela jest w **II PN**, ponieważ wszystkie pola, poza polem klucza podstawowego, muszą odnosić się do pola klucza podstawowego.

W zaprojektowanej tabeli *Realizacja zamówień* kluczem podstawowym będzie kombinacja pól *Nazwisko klienta*, *Imię* oraz *Tytuł książki*. Pola niewchodzące w skład klucza podstawowego (*Adres*, *Telefon*, *PESEL*) zależą od pól *Nazwisko* i *Imię*, natomiast nie zależą od pola *Tytuł książki*. Pole *Cena* zależy od pola *Tytuł książki* (rysunek 1.20). Tylko pole *Liczba egzemplarzy* zależy zarówno od pól *Nazwisko* i *Imię*, jak i od pola *Tytuł książki*. Tabela nie jest w **II PN**.



Rysunek 1.20. Tabela nie jest w II PN. Tylko pole *Liczba egzemplarzy* zależy od całego klucza

Normalizacja polega na podzieleniu tabeli na takie tabele, które spełnią warunek **II PN**. Tabelę *Realizacja zamówień* należy podzielić na trzy tabele: *Klient* z polami *Nazwisko*, *Imię*, *Adres*, *Telefon*, *PESEL*; *Zamówienia* z polami *Nazwisko klienta*, *Tytuł książki*, *Liczba egzemplarzy*; oraz *Książki* z polami *Tytuł książki* i *Cena*; oraz *Książki z* polami *Tytuł książki* i *Cena* (rysunek 1.21).. Kluczem podstawowym w tabeli *Klient* są pola *Nazwisko* i *Imię*, w tabeli *Zamówienia* — pola *Nazwisko klienta* i *Tytuł książki*, a w tabeli *Książki* pole *Tytuł książki*. Zostało zlikwidowane powtarzanie danych w tabeli *Realizacja zamówień* i wszystkie tabele są w drugiej postaci normalnej.



Rysunek 1.21. Podział tabeli *Realizacja zamówień* na tabele spełniające warunek II PN

Aby wyświetlić zestawienie dotyczące klienta i kupionych przez niego książek, należy zdefiniować relację między tabelami.

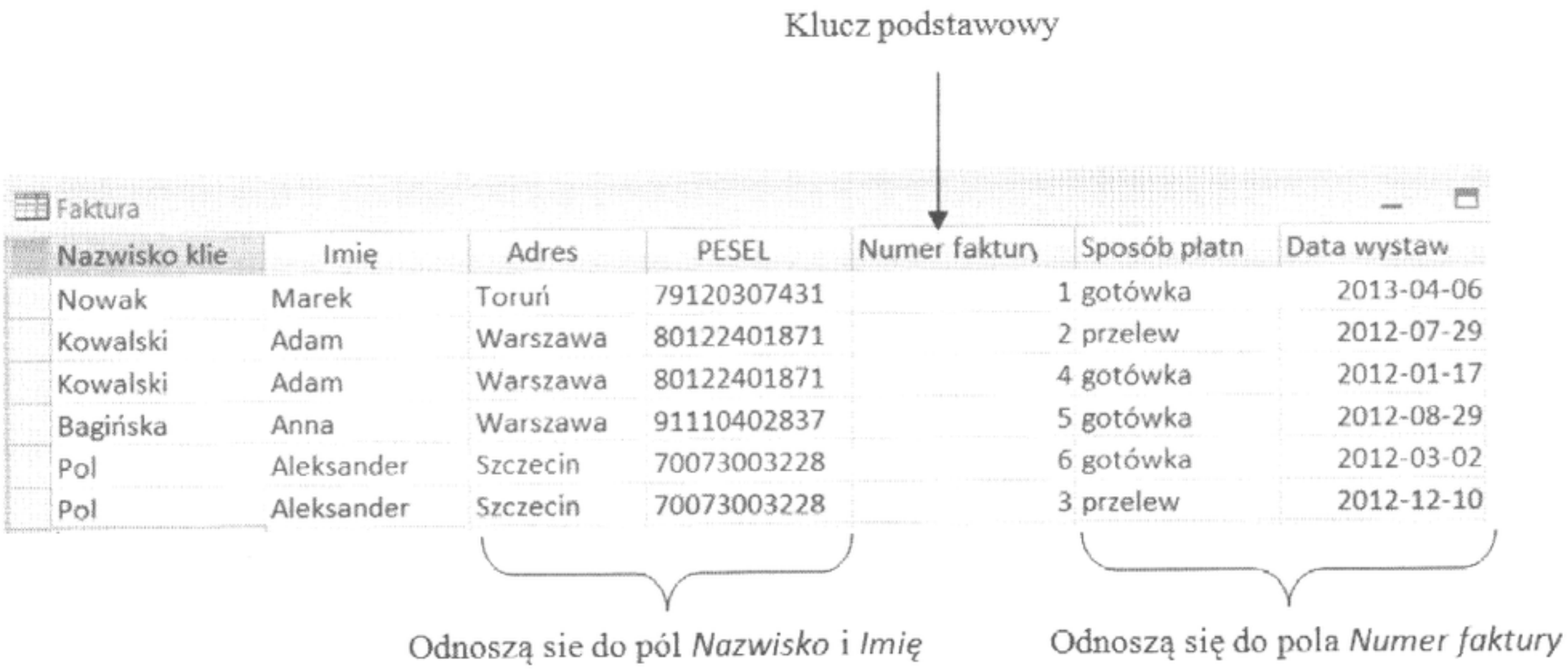
Trzecia postać normalna

DEFINICJA

Tabela jest w trzeciej postaci normalnej (**III PN**), jeżeli jest w pierwszej i w drugiej postaci normalnej oraz każde z pól niewchodzących w skład klucza podstawowego niesie informację bezpośrednio o kluczu i nie odnosi się do żadnego innego pola.

Załóżmy, że w projektowanej bazie danych dla księgarni internetowej została zaprojektowana tabela *Faktura* z polami: *Nazwisko klienta*, *Imię*, *Adres*, *PESEL*, *Numer faktury*, *Sposób płatności* i *Data wystawienia faktury*. Załóżmy również, że w pola tabeli będą

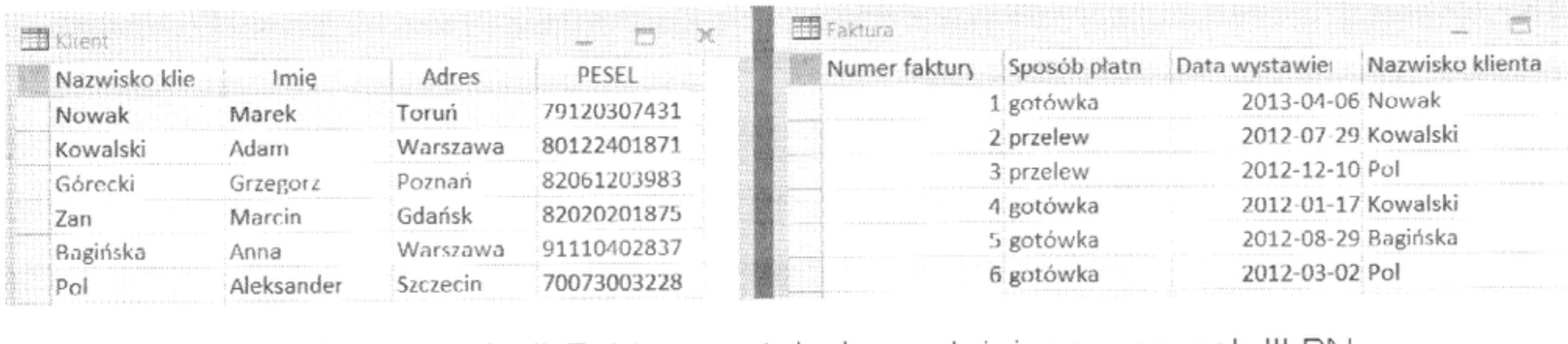
wpisywane tylko wartości elementarne, czyli tabela jest w **I PN**. Klucz podstawowy to pole *Numer faktury*. Wszystkie pola niewchodzące w skład klucza zależą od całego klucza, czyli tabela jest w **II PN**. Sprawdźmy, czy tabela jest w **III PN**. Pola *Sposób płatności* i *Data wystawienia faktury* odnoszą się do faktury, czyli zawierają informacje o kluczu. Natomiast pola *Adres* i *PESEL* zawierają informacje na temat klienta, a nie faktury (rysunek 1.22), czyli nie niosą informacji bezpośrednio o kluczu. Tabela nie jest w **III PN**.



Rysunek 1.22. Tabela nie jest w III PN. Pola *Adres* i *PESEL* nie niosą informacji o kluczu

Normalizacja, podobnie jak w przypadku **II PN**, polega na podzieleniu tabeli na takie tabele, które spełnią warunek **III PN**.

Tabelę *Faktura* należy podzielić na dwie tabele: *Faktura* (z polami *Numer faktury*, *Sposób płatności* i *Data wystawienia faktury*) oraz *Klient* (z polami *Nazwisko klienta*, *Imię*, *Adres*, *PESEL*) oraz *Klient* (z polami *Nazwisko klienta*, *Imię*, *Adres*, *PESEL*) (rysunek 1.23). Zostało zlikwidowane powtarzanie danych o kliencie w tabeli *Faktura*. Dane o kliencie będą zapisane tylko raz, w tabeli *Klient*.



Rysunek 1.23. Podział tabeli *Faktura* na tabele spełniające warunek III PN

Przykład 1.1

Przestrzegając reguł tworzenia tabel, po sprawdzeniu za pomocą normalizacji, czy tabele mają prawidłową strukturę, baza danych dla księgarni internetowej mogłaby składać się z następujących tabel:

Klient → *Id_klienta*, *Nazwisko*, *Imię*, *Kod pocztowy*, *Miejscowość*, *Ulica*, *Nr domu*, *PESEL*, *Telefon*, *Adres e-mail*.

Książki → *Id_książki*, *Tytuł*, *Id_autora*, *Cena książki*, *Wydawnictwo*, *Rodzaj literatury*, *Miejsce wydania*, *Język książki*, *Opis*.

Zamówienia → *Id_klienta*, *Id_książki*, *Data złożenia zamówienia*, *Data wysłania*, *Koszt wysyłki*, *Liczba egzemplarzy*.

Autorzy → *Id_autora*, *Nazwisko autora*, *Imię*, *Narodowość*, *Okres tworzenia*, *Rodzaj twórczości*, *Język tworzenia*, *Nota*.

1.4.8. Prawidłowy projekt bazy danych

Prawidłowy projekt bazy danych jest bardzo istotny dla efektywnej pracy, dlatego warto poświęcić trochę czasu, aby opanować zasady projektowania bazy.

Dobry projekt nie powinien zawierać powtarzających się danych. Aby osiągnąć ten cel, musimy podzielić dane na wiele tabel. Następnie powinniśmy zdefiniować połączenia między tabelami, aby można było tworzyć zestawienia danych pochodzących z różnych tabel. Na przykład w bazie danych *Księgarnia internetowa* podzieliliśmy dane na oddzielne zbiory (*Klienci*, *Książki*, *Zamówienia*, *Autorzy*), następnie zdefiniowaliśmy połączenia między tabelami, aby utworzyć zestawienie dotyczące realizacji zamówienia (*Nazwisko*, *Imię*, *Tytuł książki*, *Nazwisko autora*, *Cena książki*, *Liczba egzemplarzy*).

Proces projektowania bazy danych składa się z następujących kroków:

- **Określanie celu, jakiemu ma służyć baza danych**. Baza danych może na przykład służyć do gromadzenia informacji na temat sprzedaży książek, do wystawiania faktur dotyczących sprzedaży, do modyfikowania na bieżąco tych danych, do przetwarzania zgromadzonych danych.
- **Określenie zakresu potrzebnych informacji**. Należy określić, jakie informacje będą przechowywane w bazie, na przykład: nazwisko i imię klienta oraz jego dane osobowe, tytuły książek, informacje o autorach, informacje na temat realizacji zamówień.
- **Dzielenie informacji na tabele**. Zebrane informacje należy podzielić według tematów i dla każdego przewidzieć oddzielną tabelę, na przykład *Klient*, *Książki*.
- **Podzielenie elementów informacji na kolumny**. Trzeba zdecydować, jakie informacje mają być przechowywane w poszczególnych tabelach. Każdy element informacji zostanie przypisany do kolumny, na przykład tabela *Klient* będzie zawierała kolumny *Nazwisko klienta* i *Adres*.
- **Wybranie kluczy podstawowych**. Należy wybrać klucz podstawowy dla każdej tabeli, na przykład w tabeli *Klient* może to być identyfikator przypisany do każdego klienta lub PESEL.
- **Zastosowanie reguł normalizacji**. Za pomocą reguł normalizacji można sprawdzić, czy tabele mają prawidłową strukturę.

- **Poprawienie projektu**. Po sprawdzeniu, jeżeli to konieczne, trzeba skorygować projekt bazy.
- **Utworzenie relacji pomiędzy tabelami**. Należy przejrzeć projekt i zdecydować, jakie relacje powinny znaleźć się w bazie.

Po zaprojektowaniu bazy danych zgodnie z podanymi regułami można przystąpić do jej tworzenia, korzystając z aplikacji przeznaczonych do obsługi relacyjnych baz danych.

Przykład 1.2

Podczas analizowania przeznaczenia bazy danych tworzymy jej strukturę. Jeżeli w bazie danych dla księgarni internetowej zmienimy jej przeznaczenie, może okazać się, że tabela *Autorzy* nic jest potrzebna. Natomiast konieczne jest sporządzanie dla każdej sprzedaży faktury. Wtedy niezbędna będzie tabela do przechowywania informacji, które powinny znaleźć się na fakturze.

Baza danych mogłaby składać się z następujących tabel:

Klient → *Id_klienta*, *Nazwisko*, *Imię*, *Kod pocztowy*, *Miejscowość*, *Ulica*, *Nr domu*, *PESEL*, *Telefon*, *Adres e-mail*.

Książki → *Id_książki*, *Tytuł*, *Nazwisko i imię autora*, *Cena książki*, *Wydawnictwo*, *Rodzaj literatury*, *Miejsce wydania*, *Język książki*, *Opis*.

Zamówienia → *Id_klienta*, *Id_książki*, *Data złożenia zamówienia*, *Data wysłania*, *Koszt wysyłki*, *Liczba egzemplarzy*.

Faktura → *Numer faktury*, *Sposób płatności*, *Data wystawienia faktury*.

PYTANIA KONTROLNE

1. Wymień zalety korzystania z komputerowych baz danych.
2. Podaj definicję bazy danych.
3. Omów poznane modele baz danych.
4. Omów występujące w modelu relacyjnym rodzaje więzów integralności.
5. Wymień podstawowe cechy relacyjnego modelu baz danych.
6. Podaj definicję klucza głównego.
7. Jakiego typu relacje mogą wystąpić w bazie danych?
8. Co oznacza występujące w bazach danych pojęcie encji?
9. Do czego służą diagramy ERD?
10. Jakie zastosowanie w projektowaniu bazy danych mają narzędzia CASE?
11. W jaki sposób w tabelach opisywany jest związek „wiele do wielu”?
12. Na czym polega normalizacja tabel?

ZADANIE

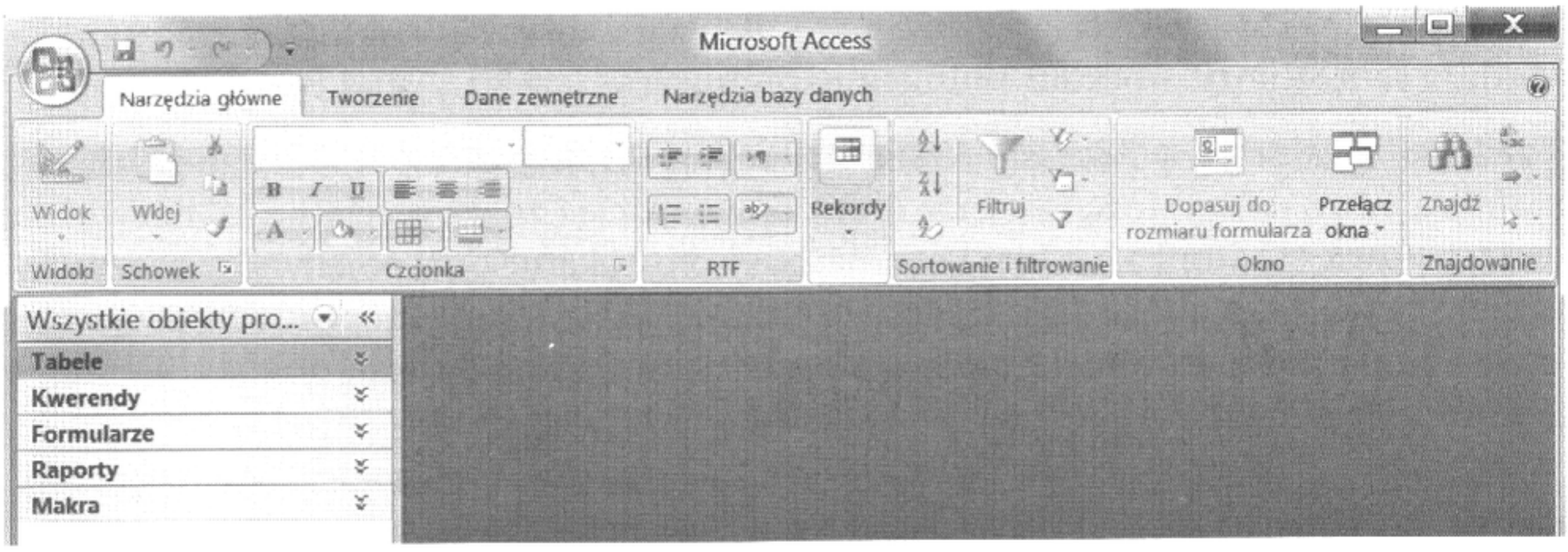
Zaprojektuj zgodnie z poznanymi zasadami bazę danych dotyczącą Twojej szkoły. Nazwij ją na przykład *Moja_szkoła*. Umieść w niej informacje na temat uczniów, klas, przedmiotów i nauczycieli, sal lekcyjnych i pracowni. Określ funkcje tworzonej bazy danych oraz zdefiniuj zbiory przechowywanych informacji. Wykorzystując diagramy ERD, opracuj model graficzny schematu bazy danych. Zaprojektuj tabele i sprawdź za pomocą reguł normalizacji, czy tabele mają prawidłową strukturę.

2

Tworzenie lokalnych baz danych w programie MS Access

2.1. Obiekty programu Access

W programie Access do budowania systemu obsługi bazy danych służy sześć rodzajów obiektów. Są to tabele, kwerendy, formularze, raporty, makropolecenia i moduły (rysunek 2.1).



Rysunek 2.1. Okno programu Access 2007

W oknie nawigacji po lewej widoczne są utworzone w bazie danych obiekty.

- Tabele

Tabele są podstawowymi obiektami bazy danych służącymi do przechowywania danych.

- **Formularze**

Formularze są obiektami, które służą do przedstawiania danych z tabeli lub kwerendy w sposób graficzny. Można ich używać do wprowadzania, edytowania lub wyświetlania danych. Za pomocą formularzy można sterować dostępem do danych. Do formularzy można dodawać różne elementy sterujące (na przykład przyciski) w celu zautomatyzowania często wykonywanych czynności. Formularze przyspieszają pracę z bazą danych i umożliwiają przedstawienie danych w atrakcyjny sposób. Mogą również zapobiegać wprowadzaniu nieprawidłowych danych.

- **Kwerendy**

Kwerendy umożliwiają przeszukiwanie, sortowanie i wyświetlanie danych. Służą do pobierania informacji z tabel, ich przetwarzania oraz wykonywania obliczeń. Za pomocą kwerend można wybrać grupę rekordów spełniających określone warunki i wykonać na tych rekordach działania.

- **Raporty**

Raporty są obiektami, które służą do wyświetlania i drukowania zestawień danych z tabeli lub kwerendy. Dzięki możliwości grupowania danych i tworzenia podsumowań nadają się do prezentowania wszelkiego rodzaju sprawozdań.

- **Makra**

Makropolecenia to procedury, których działanie powoduje wykonanie jednej lub kilku czynności zwanych akcjami. Każde makropolecenie to lista akcji wraz z argumentami akcji i warunkami decydującymi o tym, czy dana akcja będzie wykonana. Makropolecenia pozwalają uprościć i zautomatyzować obsługę baz danych. Są uruchamiane, gdy wystąpi określone zdarzenie, a połączenie zdarzenia z makropoleceniem odbywa się za pomocą właściwości zdarzeń.

- **Moduły**

Moduły są to zbiory procedur i funkcji napisanych w języku Visual Basic.

Każdemu obiektowi w Accessie trzeba nadać nazwę. Nazwa może składać się maksymalnie z 64 znaków. Może zawierać litery, cyfry, spacje i znaki specjalne. W nazwie nie mogą pojawić się kropka, wykrzyknik, nawiasy prostokątne oraz pojedynczy cudzysłów zamykający. Nazwa obiektu nie może zaczynać się od spacji, nie mogą również wystąpić w niej znaki niedrukowane. Należy nadawać nazwy, które opisują obiekty, ale nie są zbyt długie.

2.2. Tabela jako podstawowa forma organizacji danych

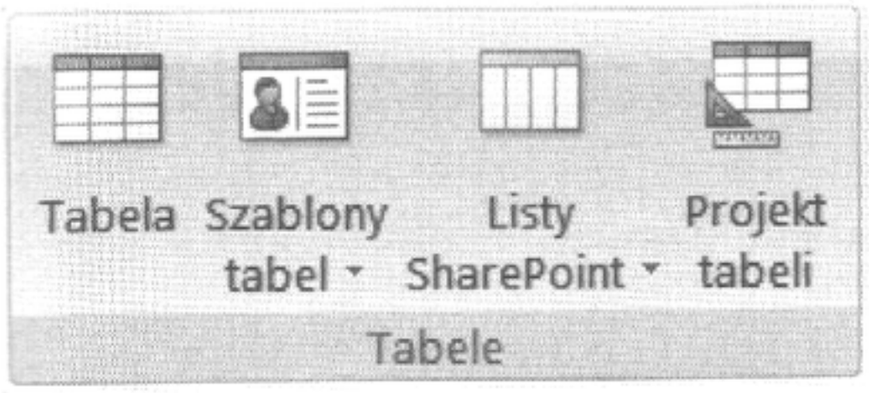
Dane w bazie danych są zapisywane w tabelach. Zbiór podobnych informacji jest umieszczony w jednej tabeli. W bazie danych może być wiele tabel przechowujących informacje

dotyczące różnych obiektów. Tabela powinna zostać właściwie zaplanowana i zaprojektowana, aby można było uniknąć konieczności późniejszych jej modyfikacji (rysunek 2.2).

Rysunek 2.2. Struktura tabeli w programie Access

Na karcie *Tworzenie*, w grupie *Tabele*, można wybrać ikonę określającą sposób tworzenia tabeli. Do wyboru mamy trzy opcje (rysunek 2.3):

- *Szablony tabel* — ta opcja umożliwia utworzenie nowej tabeli za pomocą gotowego szablonu.
- *Tabela* — ta opcja automatycznie tworzy nową tabelę.
- *Projekt tabeli* — ta opcja umożliwia utworzenie nowej tabeli przy użyciu widoku projektu.



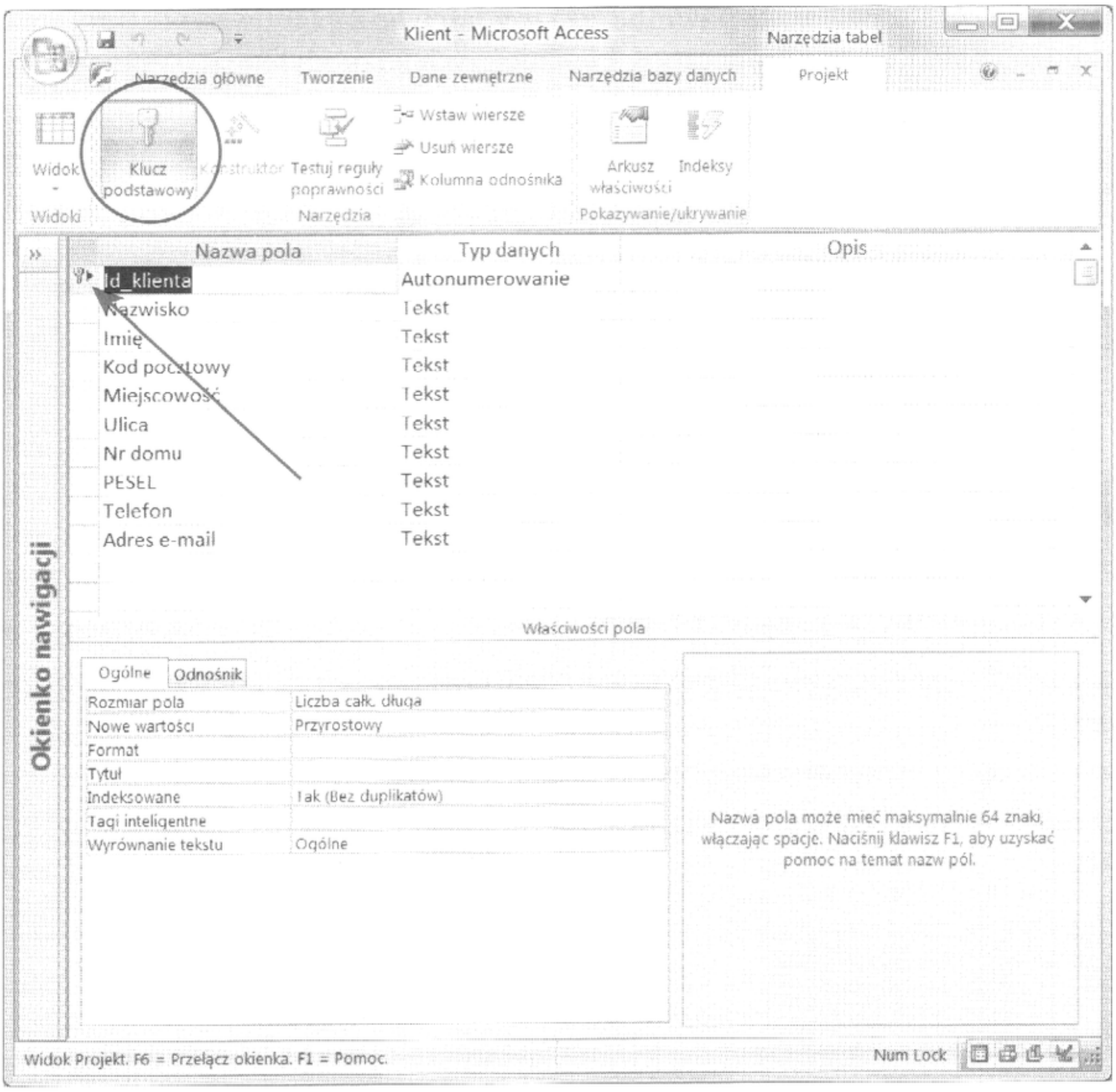
Rysunek 2.3. Opcje tworzenia tabeli

2.2.1. Okno projektu tabeli

Okno projektowania tabeli składa się z dwóch części. W górnej części definiujemy strukturę tabeli, w dolnej — właściwości pól tabeli (rysunek 2.4).

Minimalna deklaracja w projekcie tabeli musi zawierać nazwy pól i typy pól. Jeżeli nie zostanie zadeklarowany typ danych, domyślnie wybierany jest typ *Tekst*. Kolumna *Opis* może zawierać komentarze dotyczące projektowanych pól.

Tabelę po zaprojektowaniu trzeba zapisać. Można to zrobić, zamykając okno z projektem. Przy zapisywaniu tabeli należy nadać jej nazwę opisującą przechowywane informacje, na przykład *Dane osobowe*. W czasie projektowania tabeli warto zadeklarować klucz podstawowy. Każda tabela powinna mieć zdefiniowany klucz podstawowy. Jeżeli nie zdefiniowaliśmy klucza podstawowego, Access przy zapisywaniu tabeli zaproponuje zdefiniowanie tego klucza w sposób automatyczny.



Rysunek 2.4. Widok projektu tabeli. Ustawienie klucza podstawowego

2.2.2. Definiowanie klucza podstawowego

Pole pełniące funkcję klucza podstawowego ma kilka cech:

- Po pierwsze, jednoznacznie identyfikuje każdy rekord.
- Po drugie, nigdy nic jest puste ani nie ma wartości `NULL` — wartość zawsze istnieje.
- Po trzecie, jest niezbędne przy zakładaniu relacji.
- Po czwarte, jest rzadko zmieniane (najlepiej nigdy).

Za pomocą pól klucza podstawowego program Access szybko łączy dane z wielu tabel. Klucz podstawowy pozwala efektywnie przeszukiwać i odczytywać dane.

W celu ustawienia klucza podstawowego dla tabeli, a także jego zmiany lub usunięcia należy skorzystać z widoku projektu. W widoku tym poprzez kliknięcie selektora trzeba zaznaczyć wiersz z polem, które będzie pełniło funkcję klucza podstawowego (jeśli funkcję klucza podstawowego ma pełnić więcej pól, przy zaznaczaniu trzeba

przytrzymać klawisz *Ctrl*), i na karcie *Projekt* w grupie *Narzędzia* wybrać ikonę *Klucz podstawowy* (rysunek 2.4).

Jeśli pole klucza podstawowego zostało zdefiniowane automatycznie, program Access przypisuje mu nazwę pola *Identyfikator* oraz typ danych *Autonumerowanie*.

2.2.3. Typy danych

Każde pole definiowane w tabeli musi mieć określony typ danych i określoną długość lub format informacji. Cechy te ustalane są w chwili projektowania tabeli. Typy danych umożliwiają kontrolę poprawności wprowadzanych danych. W polu nie może pojawić się wartość niezgodna z typem. Typy danych zostały omówione w tabeli 2.1.

Tabela 2.1. Typy danych

Typ tekstowy		Ustawiany domyślnie. Jest to kombinacja tekstu i cyfr lub liczb niewymagających obliczeń (na przykład numery telefonów). Rozmiar *do 255 znaków*.
Typ Nota		Długi tekst lub kombinacja tekstu i liczb. W przypadku ręcznego wprowadzania danych można wprowadzać i wyświetlać maksymalnie *65 535 znaków*.
Typ Liczba		Dane numeryczne używane w obliczeniach matematycznych. Rozmiar *1, 2, 4, 8 lub 16 bajtów*. Ustawienia właściwości *Rozmiar* dla typu **Liczba** i ich wartości powiązane są w sposób przedstawiony poniżej.
	Bajt	Przechowuje liczby z przedziału od 0 do 255. Rozmiar *1 bajt*.
	Liczba całkowita	Przechowuje liczby z przedziału od –32 768 do 32 767 (bez ułamków). Rozmiar *2 bajty*.
	Liczba całkowita długa	Ustawienie domyślne. Przechowuje liczby z przedziału od –2 147 483 648 do 2 147 483 647 (bez ułamków). Rozmiar *4 bajty*.
	Pojedyncza precyzja	Przechowuje liczby z przedziału od –3,402823E38 do –1,401298E–45 w przypadku wartości ujemnych oraz od 1,401298E–45 do 3,402823E38 w przypadku wartości dodatnich. Rozmiar *4 bajty*.
	Podwójna precyzja	Przechowuje liczby z przedziału od –1,7976931348623lE308 do –4,940656458412 47E–324 w przypadku wartości ujemnych oraz od 4,940656458412 47E–324 do 1,7976931348623lE308 w przypadku wartości dodatnich. Rozmiar *8 bajtów*.
	Ułamek dziesiętny	Przechowuje liczby z przedziału od $-10^{28}-1$ do $10^{28}-1$. Rozmiar *12 bajtów*.
	Identyfikator replikacji	Unikatowy identyfikator globalny (GUID). Losowo generowane identyfikatory GUID są na tyle długie, że istnieje małe prawdopodobieństwo powtórzenia. Mają one szereg zastosowań, na przykład śledzenie towarów. Rozmiar *16 bajtów*.

Typ Data/Godzina	Wartości daty i godziny dla lat pomiędzy 100 i 9999. Rozmiar *8 bajtów*. Niezależnie od sposobu formatowania danych daty i godziny, pola typu **Data/Godzina** przechowują daty i godziny jako liczby zmiennoprzecinkowe o podwójnej precyzji.
Typ Waluta	Wartości walutowe i dane numeryczne używane w obliczeniach matematycznych dokonywanych na danych z dokładnością do czterech miejsc po przecinku. Wyniki obliczeń mają dokładność do 15 cyfr po lewej stronie separatora dziesiętnego i do 4 cyfr po prawej stronie separatora. Tego typu danych należy używać do przechowywania danych finansowych. Rozmiar *8 bajtów*.
Typ Autonumerowanie	Unikatowa liczba kolejna (zwiększana o 1) lub liczba losowa przypisywana przez program Microsoft Access przy dodawaniu nowego rekordu do tabeli. Pola *Autonumerowanie* nie można aktualizować. Typ zazwyczaj używany w kluczach podstawowych. Rozmiar *4 bajty*.
Typ Tak/Nie	Dane wartości logicznej TAK i NIE (prawda lub fałsz). W programie Access jest używana wartość –1 dla wszystkich wartości TAK i wartość 0 dla wszystkich wartości NIE.
Typ Obiekt OLE	Obiekt (na przykład arkusz kalkulacyjny programu Excel, dokument programu Word, grafika, dźwięk lub inne dane binarne) dołączony do tabeli lub osadzony w tabeli programu Access. Obiekty OLE tworzą obrazy map bitowych oryginalnego dokumentu lub innego obiektu, a następnie wyświetlają tę mapę w polach tabeli i formantach formularza lub raportu w bazie danych. Rozmiar *do 2 GB*.
Typ Hiperłącze	Tekst lub kombinacja tekstu i liczb przechowywana jako tekst i używana jako adres hiperłącza. Rozmiar *do 1 GB*.
Typ Załącznik	Do rekordów w bazie danych można dołączać obrazy, pliki arkuszy kalkulacyjnych, dokumenty, wykresy i obsługiwane pliki innego typu. Pola załączników zapewniają większą elastyczność niż pola typu *Obiekt OLE* i efektywniej wykorzystują miejsce na dysku, ponieważ nie tworzą obrazu mapy bitowej oryginalnego pliku.
Typ Kreator odnośników	Właściwie nie jest to typ danych, lecz obiekt, który powoduje wywołanie *Kreatora odnośników* pozwalającego utworzyć pole używające pola kombi do pobrania wartości z innej tabeli, kwerendy lub listy wartości.

UWAGI

Pola typu *Nota*, *Hiperłącze* i *Obiekt OLE* nie mogą być indeksowane.

W przypadku numerów telefonów, numerów elementów i innych liczb, które nie będą używane do obliczeń matematycznych, należy wybrać typ danych *Tekst*, a nie *Liczba*.

UWAGI

Typ *Waluta* powinien być stosowany w polach wymagających wielu obliczeń na danych o dokładności od jednego do czterech miejsc po przecinku. Pola o typach danych *Pojedyncza precyzja* i *Podwójna precyzja* wymagają obliczeń zmiennoprzecinkowych. Dla danych typu *Waluta* wykonywane są szybsze obliczenia stałoprzecinkowe.

Zmiana typu pola danych po wprowadzeniu danych do tabeli może spowodować długotrwały proces konwersji danych podczas zapisywania tabeli. Jeśli typ danych wpisanych w polu pozostaje w konflikcie ze zmienionym ustawieniem typu, może nastąpić utrata części danych.

2.2.4. Wprowadzanie danych

Po zaprojektowaniu tabeli, określeniu typów pól i wybraniu pola klucza podstawowego należy tabelę zapisać, podając jej nazwę, i przystąpić do wprowadzania danych. Trzeba pamiętać, że można wprowadzać tylko dane zgodne ze zdefiniowanym wcześniej typem pola.

Zadanie 2.1

Zaprojektuj prostą bazę danych składającą się z jednej tabeli. Tabela nazywa się Dane osobowe i zawiera następujące pola:

Nazwa kolumny	Typ	Rozmiar
Nr ewidencyjny	Autonumerowanie	
Nazwisko	Tekst	30 znaków
Imię	Tekst	25 znaków
Miejscowość	Tekst	50 znaków
Ulica	Tekst	45 znaków
Nr domu	Tekst	6 znaków
Data urodzenia	Data/Godzina	
Zarobki	Waluta	
Zdjęcie	Załącznik	
Uwagi	Nota	

W zaprojektowanej tabeli ustaw **klucz podstawowy**. Wpisz do tabeli przykładowe dane. Wstaw co najmniej jedno zdjęcie.

Zastanów się, dlaczego pole *Nr domu* jest typu *Tekst*. Wyjaśnij, które pole najlepiej nadaje się do pełnienia funkcji klucza podstawowego.

Zanim wstawisz zdjęcie, utwórz pliki ze zdjęciami i odpowiednio je nazwij, na przykład używając nazwisk osób. Otwórz tabelę i wybierz osobę, której zdjęcie chcesz dodać. Wybierz pole *Zdjęcie* i kliknij prawym przyciskiem myszy. Z menu podręcznego wybierz opcję *Zarządzaj załącznikami*, a następnie kliknij przycisk *Dodaj*. Odszukaj plik ze zdjęciem i zatwierdź wybór. W tabeli nie można zobaczyć zdjęcia, ponieważ dane są w niej zapisywane w trybie znakowym.

2.2.5. Właściwości pól tabeli

Każde pole zaprojektowane w tabeli ma właściwości. Za pomocą ustawień właściwości pola użytkownik może określać wygląd informacji, zapobiegać wprowadzaniu nieprawidłowych danych, ustawiać wartości domyślne, przyspieszać wyszukiwanie i sortowanie oraz sterować różnymi parametrami wyglądu lub zachowania.

Każdy typ danych ma inny zestaw właściwości. Najłatwiej definiować właściwości pól tabeli w widoku projektu. W dolnej części tego okna widoczne są właściwości wybranego pola. Właściwości pól tabeli zostały omówione w tabeli 2.2.

Tabela 2.2. Wybrane właściwości pól

Rozmiar pola	Określa maksymalną wielkość danej, jaka może być przechowywana w polu. Dotyczy danych typu *Tekst*, *Liczba* i *Autonumerowanie*.
Format	Określa sposób, w jaki wyświetlana jest na ekranie zawartość pola, na przykład sposób wyświetlania daty. Dla danych niektórych typów są zdefiniowane standardowe formy wyświetlania. Niektóre z nich zależą od ustawień w panelu sterowania Windows (symbol waluty). Można także określić własny format, używając znaków specjalnych.
Miejsca dziesiętne	Określenie liczby miejsc dziesiętnych używanych przy wyświetlaniu liczb dla typu *Liczba*.
Maska wprowadzania	Określa sposób wprowadzania danych do pola tabeli. Można na przykład wymusić na użytkownikach wprowadzanie numerów telefonów lub adresów w określonym formacie. Szczególnie przydatna dla danych typu: data, czas, numer telefonu, numer kodu pocztowego.
Tytuł	Można wprowadzić tekst opisujący kolumnę tabeli. Jeśli nie podamy tytułu, używana będzie nazwa pola.
Wartość domyślna	Ustawiana, aby automatycznie wprowadzać wartość w wybranym polu nowego rekordu (na przykład można zawsze dodawać bieżącą datę do nowych zamówień). Wartość domyślną można ustawić dla pól typu: *Tekst*, *Nota*, *Liczba*, *Data/Godzina*, *Waluta*, *Tak/Nie* oraz *Hiperłącze*.

Reguła sprawdzania poprawności	Ogranicza lub kontroluje dane, które użytkownicy mogą wprowadzać w polu tabeli.
Tekst reguły sprawdzania poprawność	Tekst komunikatu pojawiającego się po wprowadzeniu wartości niezgodnej z regułą poprawności.
Wymagane	Ustawienie rozstrzygające o tym, czy wprowadzenie danych w tym polu jest konieczne.
Zerowa długość dozwolona	Decyduje, czy mogą wystąpić ciągi znaków o zerowej długości. Umożliwia wprowadzanie ciągu o zerowej długości („”) w polu typu *Tekst* lub *Nota* po ustawieniu wartości na TAK.
Indeksowane	Przyspiesza dostęp do danych w tym polu. Indeks przyspiesza wyszukiwanie i sortowanie pól, ale może spowolnić aktualizację danych.

Przykład 2.1

Używając programu Access, utwórz bazę danych *Księgarnia internetowa* składającą się z tabel zaprojektowanych w przykładzie 1.1. Dla każdej z tabel ustaw klucz podstawowy. Dla zaprojektowanych pól określ typ oraz rozmiar lub format pola.

Wprowadź przykładowe dane do tabel.

2.2.6. Zasady tworzenia relacji między tabelami

W relacyjnych bazach danych dane zapisujemy w zdefiniowanych przez siebie tabelach, a między tabelami tworzymy połączenia, zwane w programie Access relacjami, aby zestawiać informacje z tabel wtedy, gdy okaże się to potrzebne. Następnie tworzymy kwerendy, formularze i raporty, dzięki którym są wyświetlane informacje z kilku tabel jednocześnie.

Relacje między tabelami:

- pomagają w projektowaniu kwerend na podstawie danych z wielu tabel,
- pomagają w projektowaniu formularzy i raportów opartych na danych z wielu tabel,
- poprzez wymuszanie więzów integralności zapobiegają powstawaniu rekordów odłączonych (rekord odłączony to rekord odwołujący się do rekordu, który nie istnieje).

Aby w sposób prawidłowy utworzyć relację między tabelami, muszą być spełnione następujące warunki:

- Pola wspólne muszą mieć ten sam typ danych (jeśli klucz podstawowy tworzy pole typu *Autonumerowanie*, polem klucza obcego może być pole typu *Liczba*, ale *RozmiarPola* musi być taki sam).
- Pole wspólne w tabeli ze strony *jeden* musi być polem **klucza podstawowego**.
- Pola łączone muszą zawierać informacje wzajemnie sobie odpowiadające.

UWAGA

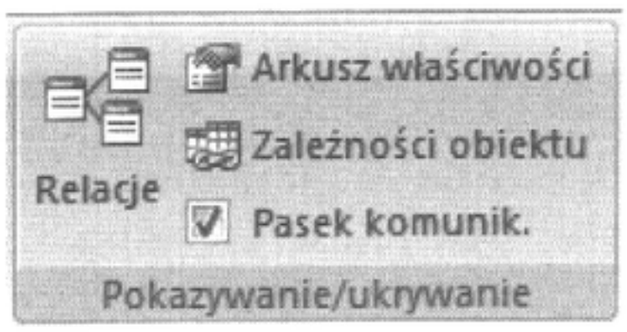
Pola wspólne nie muszą mieć tych samych nazw (choć często mają).

Relacja „jeden do wielu"

W bazach danych najczęściej mamy do czynienia z relacją typu „jeden do wielu".

Relacje między tabelami tworzymy, korzystając z okna *Relacje*. Na karcie *Narzędzia bazy danych*, w grupie *Pokazywanie/ukrywanie*, wybieramy ikonę *Relacje* (rysunek 2.5).

Rysunek 2.5.
Ikona tworzenia relacji



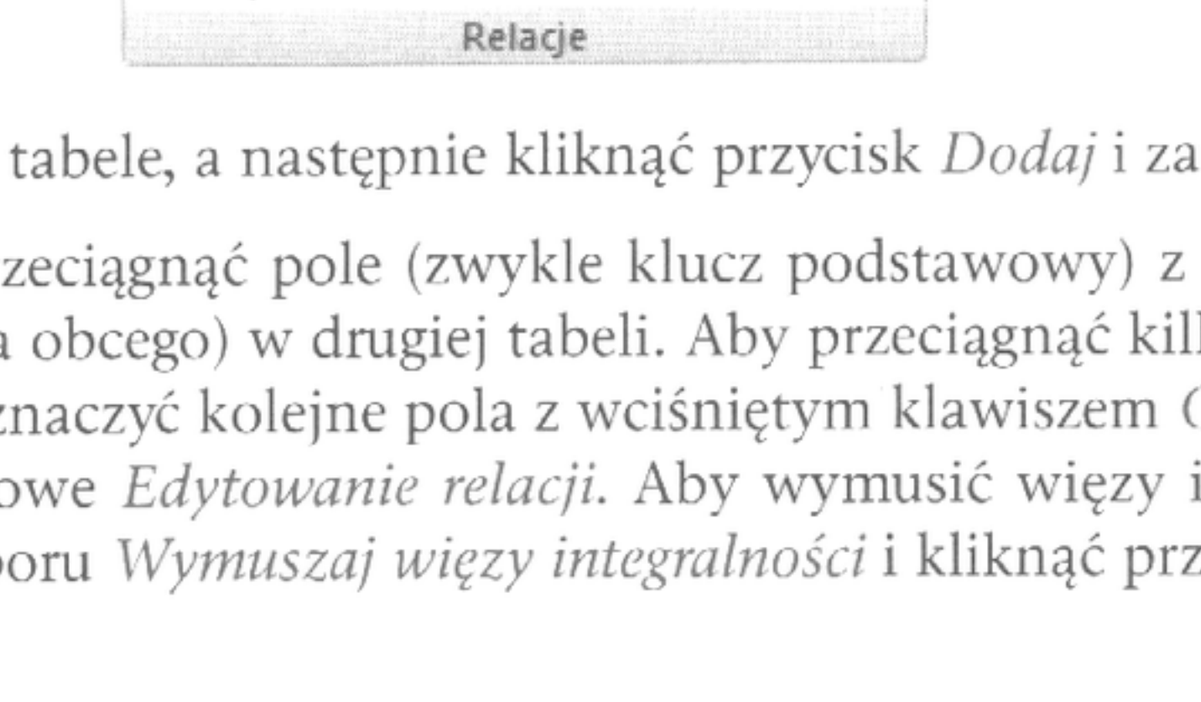
Jeśli relacji jeszcze nie zdefiniowano, automatycznie wyświetli się okno dialogowe *Pokazywanie tabeli*. Jeśli nie zostanie pokazane, na karcie *Projektowanie*, w grupie *Relacje*, wybieramy przycisk *Pokaż tabelę* (rysunek 2.6). W oknie dialogowym *Pokazywanie tabeli* są wyświetlane wszystkie tabele zawarte w bazie danych.

Rysunek 2.6.
Opcje tabel w oknie Relacje

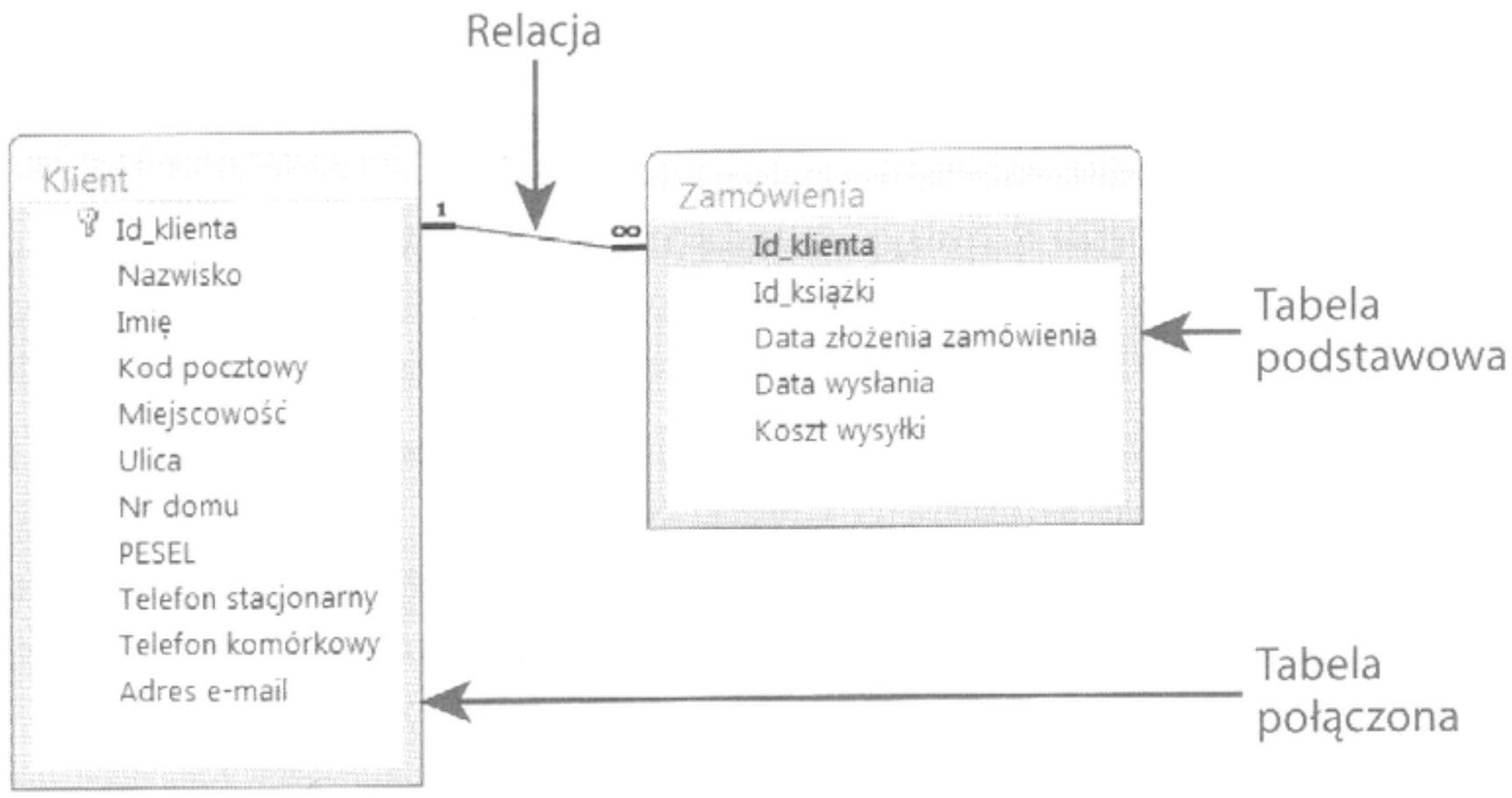
Należy zaznaczyć po kolei tabele, a następnie kliknąć przycisk *Dodaj* i zamknąć okno.

W oknie *Relacje* trzeba przeciągnąć pole (zwykle klucz podstawowy) z jednej tabeli do pola wspólnego (klucza obcego) w drugiej tabeli. Aby przeciągnąć kilka pól, przed przeciągnięciem trzeba zaznaczyć kolejne pola z wciśniętym klawiszem *Ctrl*. Zostanie wyświetlone okno dialogowe *Edytowanie relacji*. Aby wymusić więzy integralności, trzeba zaznaczyć pole wyboru *Wymuszaj więzy integralności* i kliknąć przycisk *Utwórz* (rysunek 2.7).

Rysunek 2.7.
Okno edytowania relacji



Między dwiema tabelami zostanie narysowana linia relacji. Jeśli zostało zaznaczone pole wyboru *Wymuszaj więzy integralności*, linia będzie grubsza na obu końcach. Ponadto nad grubszym odcinkiem po jednej stronie linii relacji zostanie wyświetlona liczba **1**, a nad grubszym odcinkiem po drugiej stronie linii — symbol nieskończoności (∞), tak jak pokazano na rysunku 2.8.



Rysunek 2.8. Okno Relacje po zdefiniowaniu relacji

DEFINICJA

Relacja jest to powiązanie między parą tabel bazy danych za pomocą pola **klucza podstawowego** jednej tabeli i odpowiadającego mu pola **klucza obcego** w drugiej tabeli.

Po zdefiniowaniu relacji między tabelami informacje na jej temat są wykorzystywane w projektach kwerend, w tworzeniu formularzy z podformularzami oraz przy projektowaniu pól kombi.

Jeżeli dwie tabele zostały połączone za pomocą relacji, ich obecność lub nieobecność w oknie *Relacje* nie ma wpływu na fakt istnienia związku między nimi. Relację można usunąć tylko przez usunięcie linii łączącej obydwie tabele. Usunięcie tabel z okna nie usuwa połączenia. Aby zobaczyć wszystkie powiązania zdefiniowane w bazie danych, należy wybrać na karcie *Projektowanie*, w grupie *Relacje*, ikonę *Wszystkie relacje*. Przycisk *Wyczyść układ* w grupie *Narzędzia* usuwa wszystkie tabele z okna *Relacje*, ale nie usuwa powiązań między nimi.

2.2.7. Reguły integralności bazy danych

Reguły integralności bazy danych zapewniają poprawność i spójność danych w bazie. Można je definiować na poziomie pól (typy danych, maski wprowadzania) oraz na poziomie tabel i relacji (wymuszanie więzów integralności).

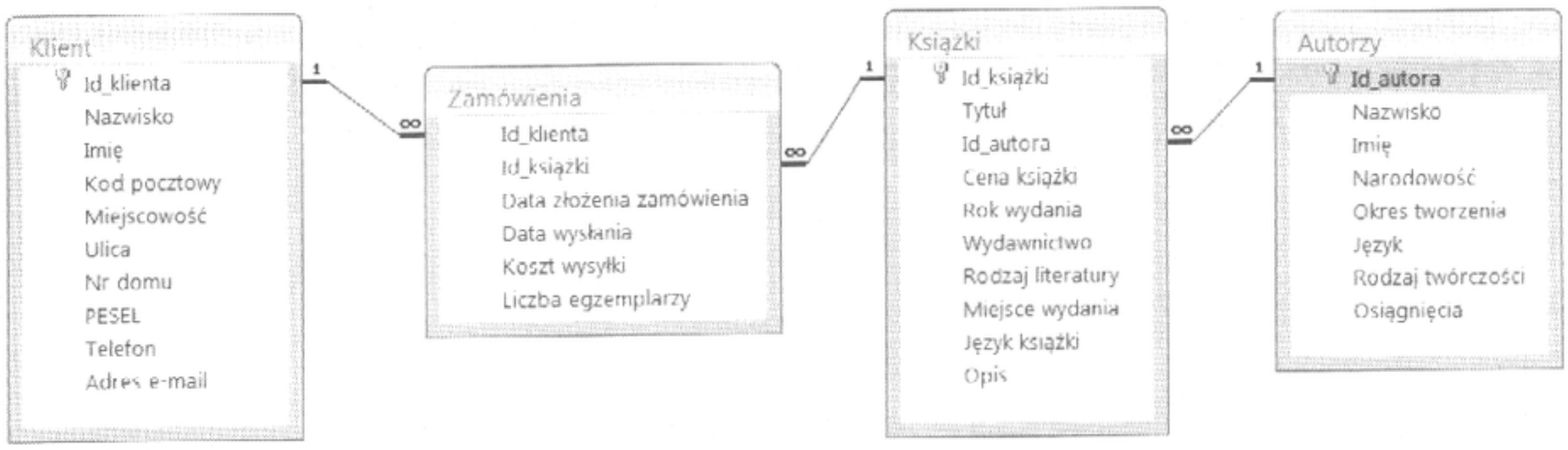
Więzy integralności, jak już wspomniano, mają na celu zapobieganie powstawaniu rekordów odłączonych i synchronizowanie odwołań. Wymusza się je przez włączenie opcji *Wymuszaj więzy integralności* podczas definiowania relacji między tabelami. Jeśli więzy integralności są wymuszone, program Access odrzuca każdą operację, która naruszyłaby więzy w relacji między tabelami.

Po włączeniu opcji *Wymuszaj więzy integralności* są stosowane następujące reguły:

- Nie można wprowadzić wartości w polu klucza obcego tabeli połączonej, jeśli ta wartość nie istnieje w polu klucza podstawowego tabeli podstawowej (czynność ta spowodowałaby powstanie rekordów odłączonych). Nie można na przykład dodać rekordu opisującego zamówienie do tabeli *Zamówienie* i w polu *Id_klienta* wprowadzić identyfikator klienta nieistniejącego w tabeli *Klient*.
- Nie można usunąć rekordu z tabeli podstawowej, jeśli w tabeli połączonej istnieją rekordy pasujące do niego. Nie można na przykład usunąć rekordu opisującego autora z tabeli *Autor*, jeśli w tabeli *Książka* istnicją książki tego autora. Można jednak zdecydować się na usunięcie rekordu podstawowego oraz wszystkich rekordów pokrewnych w ramach jednej operacji, zaznaczając pole wyboru *Kaskadowo usuń rekordy pokrewne*.
- Nie można zmienić wartości klucza podstawowego w tabeli podstawowej, jeśli spowodowałoby to powstanie rekordów odłączonych. Nie można na przykład zmienić identyfikatora klienta w tabeli *Klient*, jeśli w tabeli *Zamówienia* istnieją zamówienia przypisane do tego klienta. Można jednak zdecydować się na zaktualizowanie rekordu podstawowego oraz wszystkich rekordów pokrewnych w ramach jednej operacji, zaznaczając pole wyboru *Kaskadowo aktualizuj pola pokrewne*.

Przykład 2.2

Dla bazy danych *Księgarnia internetowa* sprawdź, czy są spełnione warunki niezbędne do prawidłowego zdefiniowania relacji, i zaprojektuj relacje między tabelami. Ustaw opcję *Wymuszaj więzy integralności* (rysunek 2.9).



Rysunek 2.9. Prawidłowo zdefiniowane relacje dla bazy Księgarnia internetowa

W otwartym oknie *Relacje* ustaw kursor myszy na polu *Id_klienta* w tabeli *Klient* i przeciągnij to pole do pola *Id_klienta* w tabeli *Zamówienia*. W otwartym oknie *Edytowanie relacji* zaznacz pole wyboru *Wymuszaj więzy integralności* i kliknij przycisk *Utwórz*. Podobnie postępuj podczas tworzenia pozostałych relacji.

UWAGA

Jeżeli relacja nie została prawidłowo zdefiniowana, po zaznaczeniu opcji *Wymuszaj więzy integralności* pojawi się komunikat informujący o przyczynie nieprawidłowości.

2.2.8. Sprawdzanie poprawności danych

W tabelach gromadzimy dane niezbędne do dalszej pracy. Jedną z podstawowych czynności po zaprojektowaniu tabel jest zapełnienie ich danymi. Aby zmniejszyć możliwość popełnienia błędu podczas wprowadzania danych do tabeli, możemy zdefiniować ograniczenia, które pomogą w kontrolowaniu poprawności danych.

Na poziomie projektowania tabel wprowadzamy ograniczenia poprzez określenie typów pól (nie można wprowadzić wartości niezgodnej z zadeklarowanym typem) oraz poprzez definiowanie właściwości pól odpowiedzialnych za wprowadzanie danych. Są to:

- wartość domyślna,
- maska wprowadzania,
- reguła poprawności.

Wartość domyślna

Jeśli dane w wybranym polu często przyjmują tę samą wartość, należy ją wpisać do właściwości pola *Wartość domyślna*. Wpisana wartość pojawi się automatycznie w każdym nowym rekordzie.

Wartość domyślna może być stałą lub wyrażeniem. Opis konstruowania wyrażeń znajduje się w dalszej części książki.

Wartość domyślną można określić dla danych wszystkich typów, poza typami *Obiekt OLE* i *Autonumerowanie*. Dla danych typu liczbowego i walutowego standardową wartością domyślną jest zero.

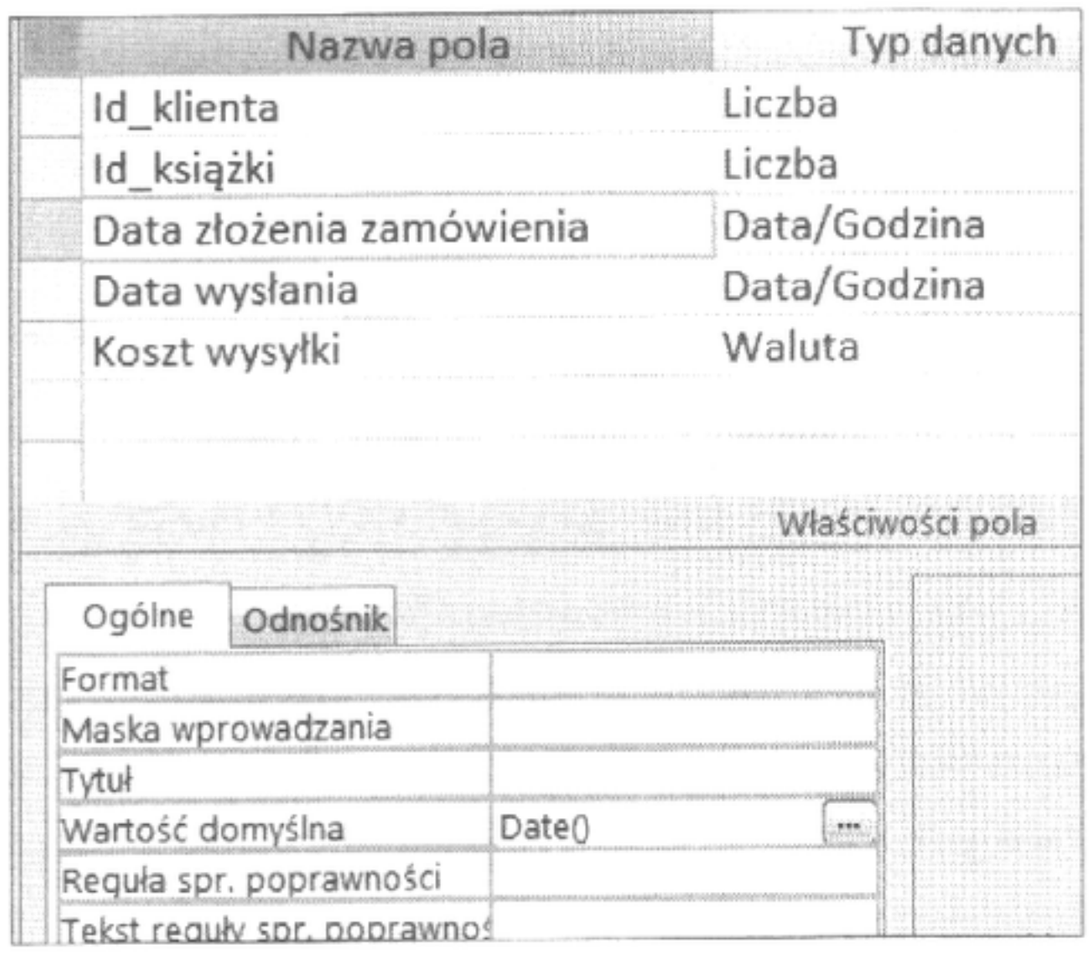
Przykład 2.3

Dla pola *Data złożenia zamówienia* w tabeli *Zamówienia* ustaw wartość domyślną na datę bieżącą. Dla pola *Język książki* w tabeli *Książki* ustaw wartość domyślną na *polski*.

Otwórz tabelę *Zamówienia* w widoku projektu. Wybierz pole *Data złożenia zamówienia*, a następnie w dolnej części okna we właściwościach pola wybierz właściwość *Wartość domyślna*.

Najprostsze przykłady wyrażeń to funkcje. Jedną z nich jest funkcja `Date()`, która zwraca datę bieżącą (rysunek 2.10). Wpisz tę funkcję jako wartość właściwości *Wartość domyślna*. W nowych rekordach automatycznie pojawi się data bieżąca. Podobnie ustaw właściwość *Wartość domyślna* dla pola *Język książki* w tabeli *Książki*.

Rysunek 2.10.
Ustawianie właściwości Wartość domyślna



Pamiętaj, że wartości domyślne będą wpisywane tylko w pola nowych rekordów, natomiast w rekordach już istniejących nie nastąpi zmiana danych.

Maska wprowadzania

Maska wprowadzania może być definiowana dla pól typu *Liczba*, *Data/Godzina* i *Tekst*. Za pomocą maski wprowadzania można osiągnąć trzy rodzaje efektów:

- wymuszenie wpisywania znaków określonego typu,
- automatyczne pojawianie się znaków, jeśli część z nich powtarza się,
- ograniczenie liczby znaków możliwych do wprowadzenia.

Maski wprowadzania są często używane w polach przeznaczonych do wpisywania kodów pocztowych lub numerów telefonów. Mogą być wykorzystane do wymuszania wstawiania wielkiej litery na początku tekstu lub do wprowadzania w polu tekstowym tylko liter lub cyfr. Symbole masek przedstawiono w tabeli 2.3. Przykłady zostały zaprezentowane w tabeli 2.4.

Tabela 2.3. Symbole maski wprowadzania

Symbol	Znaczenie
0	Cyfra — musi być wpisana, nie może być znaków + i –.
9	Cyfra lub spacja — nie musi być wpisana, nie może być znaków + i –.
#	Cyfra lub spacja — nie musi być wpisana, może być znak + lub –.
L	Litera — musi być wpisana.
?	Litera — nie musi być wpisana.
A	Litera lub cyfra — musi być wpisana.
a	Litera lub cyfra — nie musi być wpisana.
&	Dowolny znak — musi być wpisany.

Symbol	Znaczenie
<	Zamienia występujące po nim litery na małe.
>	Zamienia występujące po nim litery na wielkie.
!	Wymusza wypełnienie maski od prawej do lewej strony.
\	Następny znak nie jest traktowany jak kod maski, ale jak zwykły znak.

Tabela 2.4. Przykłady masek wprowadzania

Pole	Maska	Znaczenie
Data	`00-00-00`	W miejsce każdego zera musi być wpisana cyfra.
Data	`90-90-00`	Pola oznaczone dziewiątką mogą pozostać puste.
Nr dowodu osobistego	`>LLL0000000`	Numer dowodu składa się z dwóch części — serii oznaczonej wielkimi literami i właściwego numeru złożonego z siedmiu cyfr.
Nazwisko	`>L<??????????????`	W polu *Nazwisko* pierwsza litera powinna być wielka, pozostałe powinny być małe. Można wpisać maksymalnie 15 znaków.

Przykład 2.4

W tabeli *Klient* ustaw maski wprowadzania tak, aby:

- dla pól *Nazwisko* i *Imię* wymusić wstawianie wielkiej litery na początku tekstu, pozostałe litery powinny być małe;
- dla pola *Nr domu* zezwolić na wpisywanie liter i cyfr;
- dla pola *PESEL* zezwolić na wpisywanie cyfr;
- dla pola *Telefon stacjonarny* zezwolić na wpisywanie cyfr według wzoru *(56) 6412877*;
- dla pola *Kod pocztowy* zezwolić na wpisywanie znaków według wzoru *87-100*.

Otwórz tabelę *Klient* w widoku projektu. Wybierz pole *Nazwisko*, przejdź do właściwości pola i wybierz właściwość *Maska wprowadzania*. Wprowadź wartość `>L<L??????????????????`.

Tak zdefiniowana maska oznacza, że w polu można wprowadzać tylko litery, najkrótsze nazwisko może składać się z dwóch liter, a maksymalna długość nazwiska nie przekroczy liczby symboli umieszczonych w definicji maski wprowadzania (`L` oraz `?`). Pierwsza litera nazwiska jest wielka, pozostałe są małe. Postępując podobnie, ustaw maskę dla pola *Imię*.

Dla pola *Nr domu* wybierz typ pola *Tekst* oraz rozmiar, na przykład 4 znaki. W polu *Maska wprowadzania* wpisz wartość `aaaa`. Wymusi to wpisywanie numeru domu w postaci liter i cyfr, na przykład 12c.

Dla pola *PESEL* wybierz typ pola *Tekst* oraz rozmiar 11 znaków. Ustaw maskę wprowadzania na `00000000000`. Wymusi to wpisywanie cyfr na wszystkich pozycjach.

Dla pola *Telefon stacjonarny* wybierz typ pola *Tekst* oraz rozmiar 11 znaków. Ustaw maskę wprowadzania na `\(00") "0000000`.

Dla pola *Kod pocztowy* wybierz typ pola *Tekst* oraz rozmiar 6 znaków. Ustaw maskę wprowadzania na `00-000`.

Reguły poprawności

Reguły poprawności mogą być definiowane dla pojedynczego pola lub dla całego rekordu. Definiując regułę poprawności, określamy zakres danych, jakie mogą być wprowadzone w danym polu, lub zakres wypełnienia rekordu. Reguły poprawności można definiować samodzielnie lub za pomocą kreatora wyrażeń.

Reguła poprawności dla pola

Reguła poprawności dla pola pozwala sprawdzić wartości wpisywane w polu (sprawdzanie odbywa się przy opuszczaniu pola). Do definiowania reguły używamy wyrażeń. Najprościej zdefiniować regułę poprawności dla pól liczbowych poprzez określenie zakresu dopuszczalnych wartości dla danego pola, na przykład:

`>=50 and <=1500` dla wartości z zakresu (50, 1500);

`>=#1997-01-01# And <=#1998-1-1#` dla dat z zakresu od 1 stycznia 1997 do 1 stycznia 1998;

`<'Kowalski'` dla wszystkich nazwisk sprzed nazwiska *Kowalski* w porządku alfabetycznym.

Można też wyświetlić własny komunikat o błędzie. Treść komunikatu należy wpisać w wierszu właściwości *Tekst reguły sprawdzania poprawności*.

Reguła poprawności dla pola jest dziedziczona przez formularz oparty na tabeli.

Reguła poprawności rekordu

Reguły poprawności rekordu określają warunki, których spełnienie jest konieczne do zapisania całego rekordu (sprawdzanie odbywa się przy opuszczaniu rekordu). Reguły te mogą odwoływać się do różnych pól tej samej tabeli. Ustawia się je we właściwościach tabeli. W tym celu należy otworzyć tabelę w widoku *Widok projektu*, a następnie w obszarze *Narzędzia tabel*, na karcie *Projekt*, w grupie *Pokazywanie/ukrywanie*, wybrać ikonę *Arkusz właściwości*. Tę właściwość wykorzystuje się w celu porównania danych zapisanych w różnych polach, na przykład:

`[Data Urodzenia] < [Data Zatrudnienia]`,

`[Nazwisko] <> "" And [Imię]<>""` nie pozwoli zostawić tych pól pustych.

Przykład 2.5

W tabeli *Klient* przy rejestracji nowego klienta pola *Nazwisko* i *Imię* nie mogą pozostać puste. Ustaw odpowiednie warunki w regule poprawności rekordu. W bazie można rejestrować tylko klientów zamieszkałych w Warszawie, Krakowie lub Gdańsku. Ustaw odpowiednie warunki w regule poprawności pola.

W oknie właściwości tabeli wybierz właściwość *Reguła sprawdzania poprawności* i wprowadź wyrażenie:

```
[Nazwisko] <> "" And [Imię] <> ""
```

Zamknij okno *Arkusz właściwości*.

Wybierz pole *Miejscowość*, przejdź do właściwości *Reguła sprawdzania poprawności* w dolnej części okna i wpisz wyrażenie:

```
"Warszawa" Or "Kraków" Or "Gdańsk"
```

Przejdź do widoku tabeli i wpisz nowe dane, aby sprawdzić działanie ograniczeń wprowadzonych we właściwościach pól i tabeli.

2.2.9. Indeksowanie

Jeżeli dla wybranego pola tabeli często jest wykonywane wyszukiwanie lub sortowanie, to w celu przyspieszenia tych operacji można dla pola utworzyć indeks. Ale indeksy mogą zmniejszyć wydajność bazy przy dodawaniu lub aktualizowaniu danych. Nie zaleca się również indeksowania, jeżeli pole zawiera wiele takich samych wartości.

W niektórych przypadkach program Access tworzy indeks automatycznie, na przykład dla pola klucza podstawowego. Również użytkownik może utworzyć indeks dla jednego pola lub dla kilku pól tabeli.

Zwykle indeksy są tworzone dla pól, które są często przeszukiwane lub sortowane, oraz dla pól, które są połączone z polami innych tabel. Nie można indeksować pól typu *Obiekt OLE* lub *Załącznik*.

Definiowanie indeksu dla pola odbywa się przez ustawienie właściwości *Indeksowane*. W celu utworzenia indeksu należy otworzyć tabelę w widoku *Widok projektu*, a następnie wybrać pole, dla którego będzie tworzony indeks, i właściwość *Indeksowane*. Możliwe jest wybranie wartości *Nie*, *Tak (Duplikaty OK)*, *Tak (Bez duplikatów)*.

Nie — dla tego pola nie zostanie utworzony indeks.

Tak (Duplikaty OK) — dla tego pola zostanie utworzony indeks, wartości w polu mogą się powtarzać.

Tak (Bez duplikatów) — dla tego pola zostanie utworzony indeks, wartości w polu nie mogą się powtarzać.

2.3. Definiowanie wyrażeń w Accessie

Zadaniem bazy danych jest nie tylko przechowywanie informacji, ale również ich przetwarzanie. Projektując system obsługi bazy danych, często korzystamy z wyrażeń (formuł). Wyrażenia umożliwiają:

- wykonywanie obliczeń,
- pobieranie wartości pól lub formantów,
- definiowanie właściwości pól tabeli i formantów (wartość domyślna, reguła poprawności),
- tworzenie formantów i pól obliczeniowych,
- definiowanie kryteriów kwerendy,
- definiowanie poziomów grupowania w raporcie.

2.3.1. Elementy wyrażenia

Elementy wyrażenia to identyfikatory (nazwy pól, formantów lub właściwości), stałe, wartości, funkcje i operatory.

Identyfikatory

Są to nazwy pól tabeli, formanty w formularzach lub raportach albo właściwości tych pól lub formantów. Za ich pomocą można w wyrażeniach odwołać się do wartości związanej z polem, właściwością lub formantem, na przykład:

```
=[Data złożenia zamówienia] - [Data wysłania]
```

W skład identyfikatora wchodzą nazwa obiektu oraz operatory *!* (wykrzyknik) i . (kropka). Nazwy obiektów zawierające spację lub inne znaki specjalne muszą być ujęte w nawiasy kwadratowe. Jeśli w wyrażeniu odwołujemy się do innego obiektu, identyfikator ma postać:

```
[Nazwa rodzaju obiektu]![Nazwa obiektu].[Nazwa detalu].[Nazwa właściwości].
```

Oto przykład:

```
=Formularze![Form_książki].[Data_Zamówienia] -Formularze![Form_książki].
[Data_Wysłania]
=Raporty![R_książki].[Data_Zamówienia] - Raporty![R_książki].[Data_Wysłania]
=Formularze![Form_książki].[Data_Wysłania].[WartośćDomyślna]
```

Nazwy właściwości składających się z kilku słów piszemy bez spacji.

Jeśli odwołanie w wyrażeniu odnosi się do formantu w formularzu lub raporcie aktywnym, wystarczy podać tylko nazwę formantu.

W przypadku tabel pomijamy nazwę rodzaju obiektu, na przykład:

```
=[Klient].[Nazwisko]
```

Stałe

Stała jest nazwanym elementem, którego wartość nie zmienia się podczas działania programu Access. Stałe najczęściej używane w wyrażeniach to: `True`, `False` i `Null`.

Wartości

W wyrażeniach można stosować wartości literalne:

- liczby, na przykład:

```
1254; -234; 1,0E-6
```

- ciągi tekstowe (ujęte w cudzysłowy lub apostrofy), na przykład:

```
"Toruń"; 'Warszawa'; [Klient].[Miejscowość]="Poznań"
```

- wartości typu *Data/Godzina* (ujęte w znaki #), na przykład:

```
#03-07-07#; #07-Mar-07#; #07-Mar-2007#
```

Funkcje

Funkcje są wbudowanymi procedurami, które można stosować w wyrażeniach. Funkcja użyta w wyrażeniu ma postać:

```
Funkcja(argument, argument)
```

Niektóre funkcje nie mają argumentów, na przykład `Date()`. Jednak dla większości funkcji musimy określić argumenty, na przykład:

```
Year([DataUrodzenia])

Len("komputer")
```

Operatory

Operatory określają rodzaj działania wykonywanego w wyrażeniu.

- **Operatory arytmetyczne** służą do obliczania wartości z dwóch lub więcej liczb lub do zmieniania znaku liczby z + na –:

  +, –, *, / — dodawanie, odejmowanie, mnożenie i dzielenie;

  \ — dzielenie, a następnie obcięcie wyniku do liczby całkowitej;

  `MOD` — reszta z dzielenia;

  ^ — podniesienie liczby do potęgi podanej w wykładniku.

- **Operatory porównań** służą do porównywania wartości i zwracania wyniku w postaci logicznej *Prawda*, *Fałsz* lub nieokreślonej *NULL*:

  <, >, <=, >=, =, <>.

- **Operatory tekstowe** służą do łączenia dwóch wartości tekstowych w jeden ciąg.

  + — łączy łańcuchy tekstowe. Jeśli jeden z łańcuchów ma wartość *NULL*, wynik jest *NULL*.

`&` — łączy łańcuchy tekstowe. Jeśli jeden z łańcuchów ma wartość *NULL*, wynik jest równy drugiemu łańcuchowi.

- **Operatory logiczne** służą do łączenia dwóch wartości logicznych i zwracania wyniku w postaci logicznej: `Not`, `And`, `Or`, `Xor`.
- **Operatory specjalne** programu Access:

  `IS NULL` lub `IS NOT NULL` — porównuje dowolną wartość z wartością *NULL*.

  `Like "wzorzec"` — porównuje łańcuch tekstowy z podanym wzorcem. Wzorzec może być definiowany z użyciem symboli wieloznacznych `*` i `?`.

  `Between wartość1 And wartość2` — sprawdza, czy wartość wyrażenia mieści się w podanym zakresie wartości.

  `In(ciąg1, ciąg2...)` — określa, czy dana wartość należy do zestawu listy wartości.

Przykłady wyrażeń zostały pokazane w tabeli 2.5.

Tabela 2.5. Przykłady wyrażeń

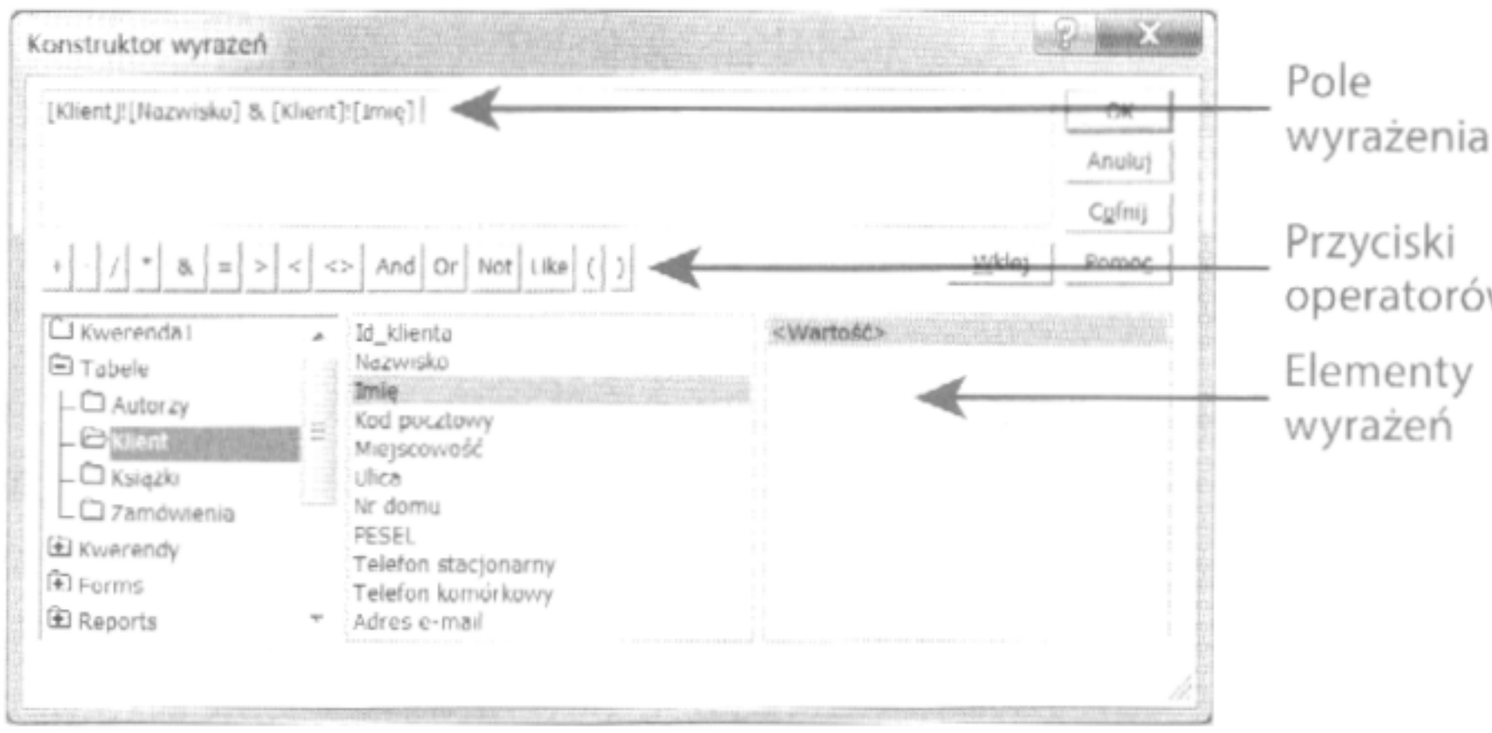
Wyrażenie	Znaczenie
`LIKE "A*"`	Użyte w kryterium kwerendy dla pola *Nazwisko* pokaże osoby, których nazwiska zaczynają się na literę *A*.
`IN ("Wrocław", "Warszawa", "Gdańsk")`	Użyte w kryterium kwerendy dla pola *Miejsce urodzenia* pokaże osoby urodzone w wymienionych miejscowościach.
`BETWEEN 50 AND 159`	Użyte w regule poprawności pozwoli wprowadzać wartości z podanego zakresu.
`BETWEEN #1-1-2008# AND #30-1-2008#`	Użyte w kryterium kwerendy pokaże osoby zarejestrowane *od 1 do 30 stycznia 2008 r.*
`BETWEEN #14-10-2008# AND Date()`	Użyte w kryterium kwerendy pokaże towary dostarczone w określonym czasie.
`Year(#15-12-2008#)`	Z podanej daty zostanie zwrócony rok.
`LIKE "[A-D]*"`	Użyte w kryterium kwerendy pokaże osoby, których nazwiska zaczynają się na litery *A*, *B*, *C*, *D*.
`LIKE "[a-z][a-g]*"`	Użyte w kryterium kwerendy pokaże ciągi tekstowe, w których na pierwszej pozycji jest dowolna litera, a na drugiej — litery z przedziału *a-g*.
`[Osoby].[Nazwisko]&" "&[Osoby].[Imię]`	W kwerendzie zostanie utworzone pole obliczeniowe zawierające połączone dane z pól *Nazwisko* i *Imię* oddzielone spacją.

Wyrażenie	Znaczenie
`IS NOT NULL`	Użyte w kryterium kwerendy pokaże rekordy, które mają wypełnione to pole.
`Year(Date())-Year([Data Urodzenia])`	W polu obliczeniowym kwerendy zostanie pokazany wiek osób.
`[Cena jednostkowa]*[ilość towaru]`	W polu obliczeniowym kwerendy zostanie wyliczona wartość towaru.

2.3.2. Konstruktor wyrażeń

Konstruktor wyrażeń jest narzędziem ułatwiającym tworzenie wyrażeń (rysunek 2.11). Umożliwia łatwy dostęp do pól i formantów w bazie danych oraz do funkcji dostępnych w programie. Za pomocą konstruktora wyrażeń można opracowywać nowe wyrażenia lub modyfikować już istniejące.

Rysunek 2.11. Okno konstruktora wyrażeń



Konstruktor można uruchamiać z większości miejsc, w których mogą być definiowane wyrażenia. Są to: właściwość formantu, właściwość pola tabeli (reguła sprawdzania poprawności, wartość domyślna), kryteria definiowane w kwerendzie, pola obliczeniowe określane w kwerendzie lub formanty w formularzu. Wszędzie tam, gdzie jest widoczny przycisk *Konstruuj* ..., można użyć konstruktora wyrażeń.

Pole wyrażenia

W górnej części konstruktora znajduje się *pole wyrażenia*, w którym tworzone jest wyrażenie. W polu tym można ręcznie wpisać wyrażenie lub wybrać elementy z trzech kolumn z dolnej części konstruktora i dodać je do pola wyrażeń.

Przyciski operatorów

W środkowej części konstruktora wyrażeń wyświetlane są przyciski służące do wstawiania operatorów arytmetycznych i logicznych. W celu wstawienia operatora w polu wyrażenia należy kliknąć odpowiedni przycisk.

Elementy wyrażeń

W dolnej części okna umieszczono trzy kolumny:

- W lewej kolumnie znajdują się foldery zawierające listę tabel, kwerend, formularzy i raportów bazy danych, a także dostępnych funkcji wbudowanych i zdefiniowanych przez użytkownika, stałych, operatorów i typowych wyrażeń.
- Środkowa kolumna zawiera listę elementów lub kategorii elementów określonych dla folderu zaznaczonego w lewej kolumnie.
- Prawa kolumna zawiera listę wartości (jeśli takie istnieją) dla elementów zaznaczonych w lewej i środkowej kolumnie.

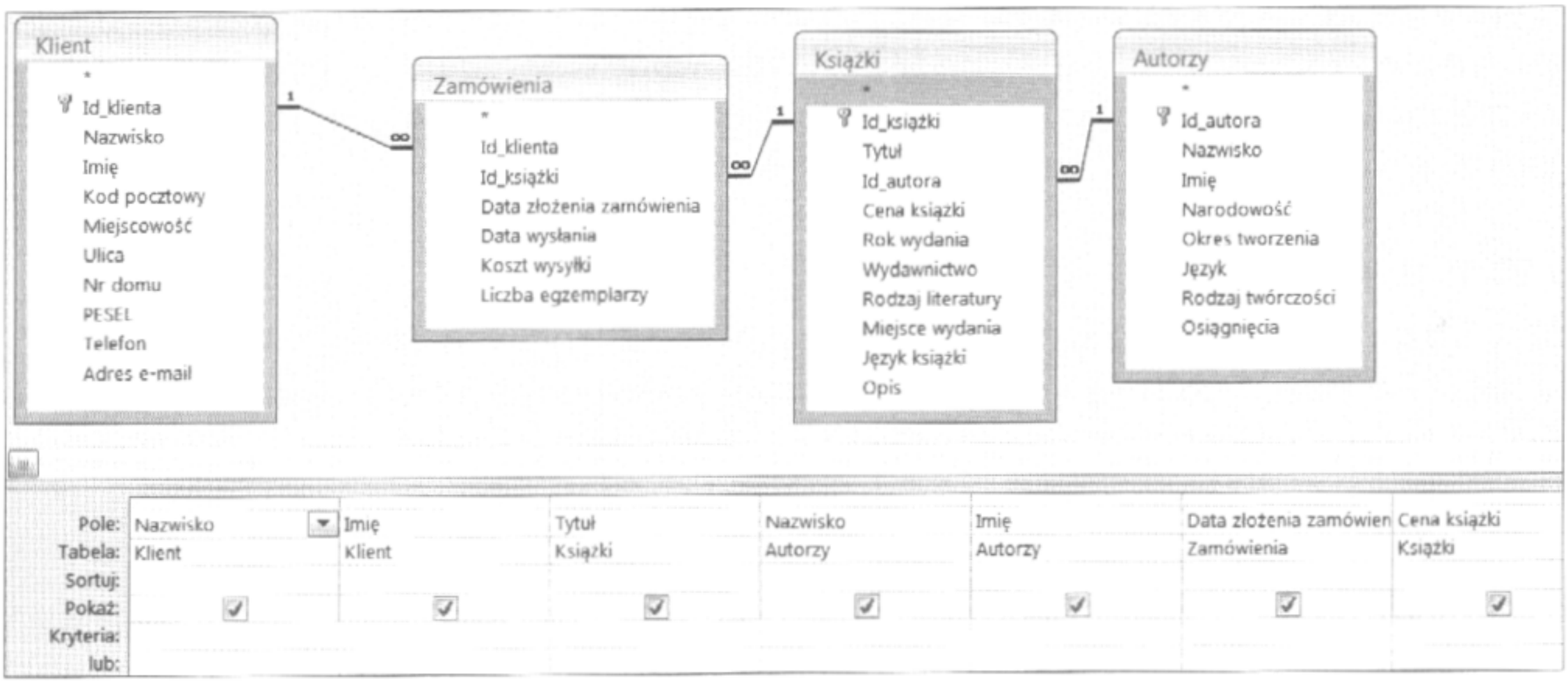
2.4. Kwerendy

Projektując bazę danych, umieszczamy w tabelach tylko wybrane dane. Aby korzystać z tych danych, trzeba określić sposób ich wybierania i przetwarzania. W systemie relacyjnych baz danych czynności te są wykonywane za pomocą **kwerend**. Do konstruowania kwerend i kierowania ich do bazy danych można w programie Access używać języka SQL (ang. *Structured Query Language*) lub QBE (ang. *Query By Example*).

QBE jest metodą tworzenia kwerend, za pomocą której można zbudować kwerendę, wybierając na karcie *Tworzenie* w grupie *Inne* ikonę *Projekt kwerendy* lub *Kreator kwerend*.

Projektując kwerendę, wybieramy tabele oraz pola, które będą brały udział w kwerendzie, określamy kryteria wyboru rekordów oraz sposób ich wyświetlania.

Okno projektowania kwerendy zostało podzielone na dwie części (rysunek 2.12). Górna zawiera obiekty, które biorą udział w kwerendzie (tabele i kwerendy), oraz zdefiniowane relacje, dolna stanowi szablon projektowania kwerendy. W kolejnych kolumnach szablonu umieszczamy pola, które będą dostępne w kwerendzie. Nazwy pól wybieramy z listy w wierszu *Pole* lub przenosimy z obiektów znajdujących się w górnej części okna, przeciągając je myszą do szablonu lub dwukrotnie klikając wybrane pole.



Rysunek 2.12. Okno projektowania kwerendy

- Wiersz *Tabela* w szablonie określa, z jakiego obiektu pochodzi pole. Wartość ta jest zwykle ustawiana automatycznie.
- Wiersz *Sortuj* ustala sposób porządkowania danych w kwerendzie.
- Wiersz *Pokaż* umożliwia ukrywanie pól, które służą do określania kryterium, ale nie muszą być widoczne.
- Wiersz *Kryteria* określa kryteria wyboru rekordu.

Kwerendy dzielimy na dwie grupy:

- kwerendy wybierające,
- kwerendy funkcjonalne.

2.4.1. Kwerendy wybierające

Kwerendy wybierające umożliwiają wybieranie z bazy danych rekordów spełniających określone kryteria. Do grupy tej należą również kwerendy, w których definiujemy pola obliczeniowe czy wykonujemy podsumowania w grupach, oraz kwerendy z parametrem.

Przykład 2.6

Zaprojektuj kwerendę wyświetlającą dane klientów mieszkających w Krakowie.

Kliknij na karcie *Tworzenie* w grupie *Inne* ikonę *Projekt kwerendy*. W otwartym polu dialogowym wybierz jako źródło kwerendy tabelę *Klient*. Zostanie otwarte okno *Projekt kwerendy*. Przenieś do szablonu w dolnej części pola *Nazwisko*, *Imię*, *Kod pocztowy*, *Miejscowość*, *Ulica*, *Nr domu*, *PESEL*, *Telefon*, *Adres e-mail*. W polu *Miejscowość* zdefiniuj kryterium `="Kraków"` (rysunek 2.13).

Rysunek 2.13. Projekt kwerendy ze zdefiniowanym kryterium

W górnym lewym rogu okna w grupie *Wyniki* kliknij ikonę *Widok*, aby zobaczyć wynik zaprojektowanej kwerendy. Jeżeli należy dokonać zmian w projekcie, wybierz ikonę *Widok* w grupie *Widoki* i wróć do projektu kwerendy. Jeśli kwerenda została zaprojektowana prawidłowo, zamknij okno. Program Access poprosi o podanie nazwy, pod jaką kwerenda zostanie zapisana w bazie.

Przykład 2.7

Zaprojektuj kwerendę wyświetlającą zamówienia złożone przez klienta o nazwisku Kowalski.

W kwerendzie będą wyświetlane dane klienta z tabeli *Klient* oraz szczegóły zamówień klienta o nazwisku Kowalski zapisanych w tabeli *Zamówienia*. W projekcie kwerendy wybierz jako źródło danych obie tabele. W oknie *Projekt kwerendy* widać strukturę obu tabel oraz zdefiniowane połączenie między nimi. Jeżeli między tabelami nie zostało zdefiniowane połączenie, teraz na potrzeby kwerendy można to zrobić. W projekcie kwerendy umieść pola *Nazwisko*, *Imię*, *Adres e-mail*, *Data złożenia zamówienia*, *Data wysłania* i *Koszt wysyłki*. W polu *Nazwisko* zdefiniuj kryterium `="Kowalski"`. Wynikiem kwerendy powinna być lista zamówień złożonych przez Kowalskiego. Zapisz kwerendę i sprawdź jej działanie. Jeżeli w bazie znalazłoby się kilku klientów o nazwisku Kowalski, należałoby zdefiniować dodatkowe kryterium dla pola *Imię*.

Do kwerend wybierających można zaliczyć:

- kwerendę podsumowującą,
- kwerendę z polem wyliczeniowym,
- kwerendę z parametrem,
- kwerendę krzyżową.

Kwerenda podsumowująca

Działanie kwerend podsumowujących polega na pogrupowaniu danych z tabeli (lub tabel) według wybranego pola, a następnie wykonaniu obliczeń na danych w każdej grupie. Obliczenia odbywają się z wykorzystaniem **funkcji agregujących**. Funkcje te działają na wartościach wybranego pola w grupie wierszy. Mogą zostać użyte do obliczenia sumy wartości (`Suma`), zliczania wierszy o danej wartości (`Policz`), wykonania obliczeń statystycznych (`Średnia`, `OdchStd`, `Wariancja`), a także wybrania wiersza z wartością najmniejszą lub największą (`Minimum`, `Maksimum`) lub takiego, w którym dana wartość występuje jako pierwsza lub ostatnia (`Pierwszy`, `Ostatni`). Funkcje agregujące mogą być używane tylko w kwerendach podsumowujących i są dostępne w oknie projektu kwerendy po rozwinięciu listy w wierszu *Podsumowanie* (rysunek 2.14).

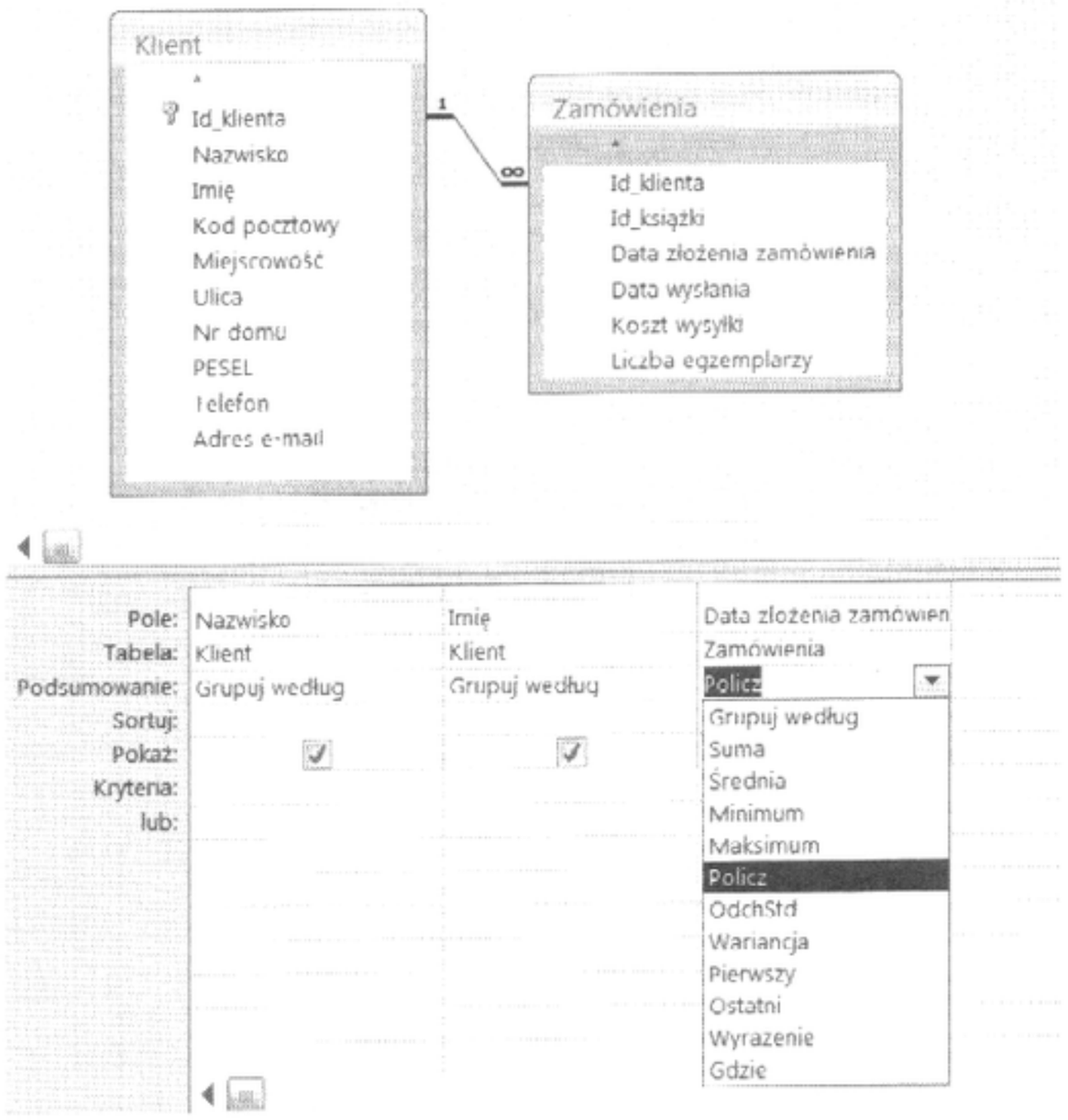
Kryteria wyboru rekordów w kwerendach podsumowujących są określane tak jak w zwykłych kwerendach wybierających. W wierszu *Kryteria* definiujemy określony dla pola warunek, a w wierszu *Podsumowanie* wybieramy pozycję *Gdzie*. Pola z tym wyrażeniem nie są wyświetlane.

Przykład 2.8

Zaprojektuj kwerendę wyświetlającą informację, ile razy każdy z klientów księgarni internetowej składał w niej zamówienie.

Aby wykonać to zadanie, w projekcie kwerendy dodaj wiersz *Podsumowanie*. Wybierz na wstążce, w obszarze *Narzędzia kwerend*, na karcie *Projektowanie*, w grupie *Pokazywanie/*

Rysunek 2.14.
Projekt kwerendy — wybór funkcji agregującej



ukrywanie, ikonę *Sumy*. Następnie w wierszu *Podsumowanie* wybierz w polach *Nazwisko* i *Imię* opcję *Grupowanie według*, a w polu *Data złożenia zamówienia* wskaż na liście funkcję `Policz`. Sprawdź działanie kwerendy. Gotową kwerendę zaprezentowano na rysunku 2.15.

Rysunek 2.15.
Kwerenda podsumowująca

Policz zamówienia

Nazwisko	Imię	PoliczOfData
Białoszewski	Mirek	2
Górecki	Grzegorz	3
Kowalski	Adam	5
Nowak	Marek	4
Pol	Aleksander	7

Zastrzeżenia budzi nazwa pola *PoliczOfData*. Można mu nadać własną nazwę, wracając do projektu kwerendy. Po kliknięciu prawym przyciskiem myszy w polu *Data złożenia zamówienia* wybierz z listy polecenie *Właściwości*, w otwartym oknie *Arkusz właściwości* wybierz właściwość *Tytuł* i wstaw wybrany przez siebie tekst (na przykład *Liczba zamówień*), który zostanie wyświetlony jako tytuł kolumny z liczbą zamówień.

Kwerenda z polem wyliczeniowym

Prawidłowo zaprojektowana baza danych nie przechowuje w tabelach wartości obliczonych. Jeżeli na podstawie danych zapisanych w tabelach należy obliczyć nowe wartości, można zaprojektować kwerendę z polem wyliczeniowym, w której obliczymy i wyświetlimy wymagane wyniki.

Przykład 2.9

W tabeli *Książki* pole *Cena* określa cenę każdej książki, a w tabeli *Zamówienia* w polu *Liczba egzemplarzy* jest podawana liczba egzemplarzy książki zamówionej przez klienta. Oblicz wartość każdego zamówienia, mnożąc cenę książki przez liczbę zamówionych egzemplarzy i dodając koszt wysyłki.

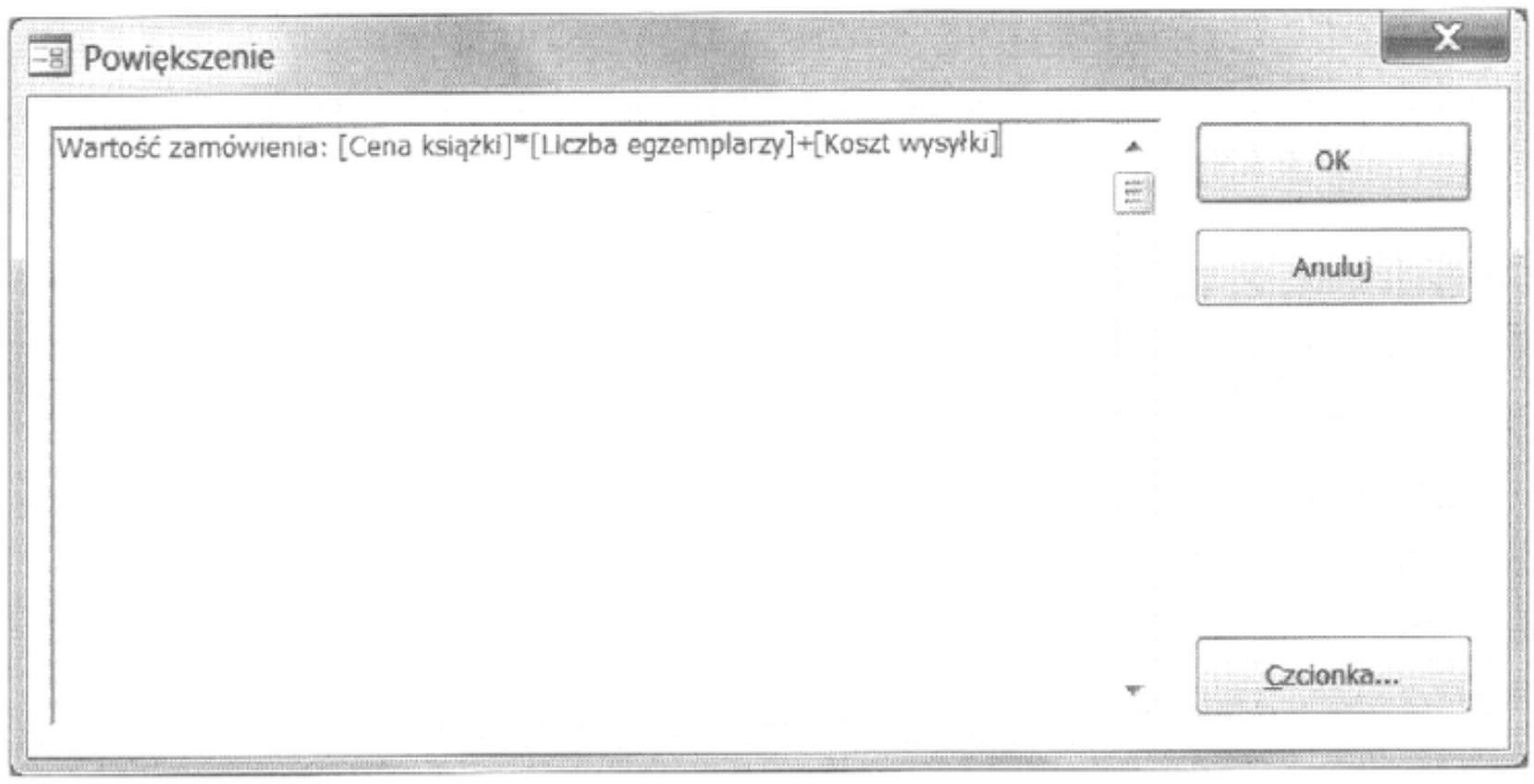
Utworzona kwerenda powinna zawierać nazwisko i imię klienta, datę złożenia zamówienia oraz obliczoną wartość zamówienia.

W projekcie kwerendy umieść pola *Nazwisko* i *Imię* z tabeli *Klient* oraz pole *Data złożenia zamówienia* z tabeli *Zamówienia*. W kolejnej kolumnie w wierszu *Pole* wpisz wyrażenie:

```
[Cena ksiażki]*[Liczba egzemplarzy]+[Koszt wysyłki]
```

Pole zostanie automatycznie uzupełnione o tekst *Wyr1:* i pod taką nazwą będzie wyświetlane. Aby to zmienić, usuń tekst *Wyr1* i w jego miejsce wpisz *Wartość zamówienia*. Jeżeli zdefiniowane wyrażenie nie jest widoczne w całości, można kliknąć pole prawym przyciskiem myszy i wybrać opcję *Powiększenie*. Zostanie otwarte nowe okno, w którym można zobaczyć całe wyrażenie i zmodyfikować je (rysunek 2.16). Sprawdź działanie kwerendy.

Rysunek 2.16. Okno Powiększenie



Kwerenda z parametrem

W bazie danych często tworzymy kwerendy, w których zmianie ulega tylko warunek zdefiniowany w kryterium wyboru. Załóżmy, że potrzebne jest nam zestawienie z listą osób mieszkających w Krakowie i drugie, z listą osób mieszkających w Poznaniu. Zostaną utworzone dwie kwerendy różniące się wartością umieszczoną w polu *Kryteria*. Zmieniając tylko kryterium wyboru, można tworzyć kolejne kwerendy. Można również utworzyć kwerendę, która w chwili uruchomienia wyświetli zapytanie o wartość parametru umieszczanego w kryterium wyboru. Podana wartość zostanie umieszczona w polu *Kryteria*. Wynikiem kwerendy będzie zestaw rekordów, które są zgodne z zadeklarowaną wartością. Przy kolejnym uruchomieniu kwerendy można wprowadzić inną wartość parametru i wynik kwerendy będzie inny.

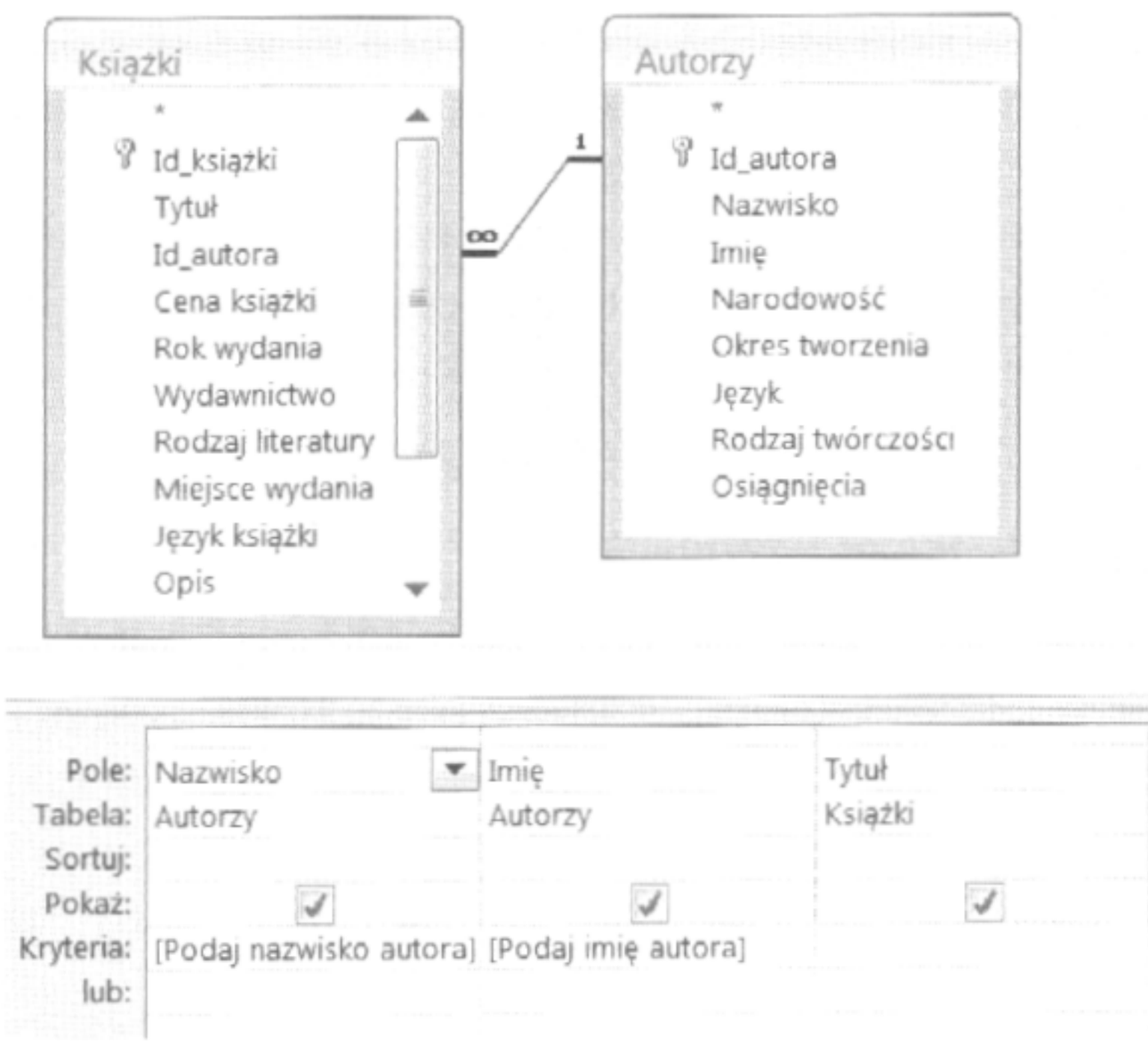
Przy projektowaniu takiej kwerendy należy wartość wpisywaną w polu *Kryteria* zastąpić tekstem wpisanym w nawiasach kwadratowych *[Tekst]*. Treść jest dowolna, ważne, aby była ujęta w nawiasy. Ta zmiana spowoduje, że przy uruchamianiu kwerendy pojawi się okno dialogowe z wpisanym tekstem i polem tekstowym, do którego trzeba wprowadzić wartość parametru. Tak działająca kwerenda została nazwana kwerendą z parametrem.

Przykład 2.10

Zaprojektuj kwerendę wyświetlającą listę książek wskazanego autora.

W projekcie kwerendy wybierz jako źródło danych tabele *Książki* i *Autorzy*. Przenieś do projektu kwerendy pola *Nazwisko* i *Imię* z tabeli *Autorzy* oraz *Tytuł* z tabeli *Książki*. W wierszu *Kryteria* dla pola *Nazwisko* wpisz tekst `[Podaj nazwisko autora]`, a dla pola *Imię* tekst `[Podaj imię autora]` (rysunek 2.17). Spowoduje to, że przy uruchamianiu kwerendy pojawią się po kolei dwa okna dialogowe z podanym tekstem i polem przeznaczonym do wpisania nazwiska i imienia autora. Nazwisko oraz imię autora są parametrami tworzonej kwerendy.

Rysunek 2.17. Projekt kwerendy z parametrem



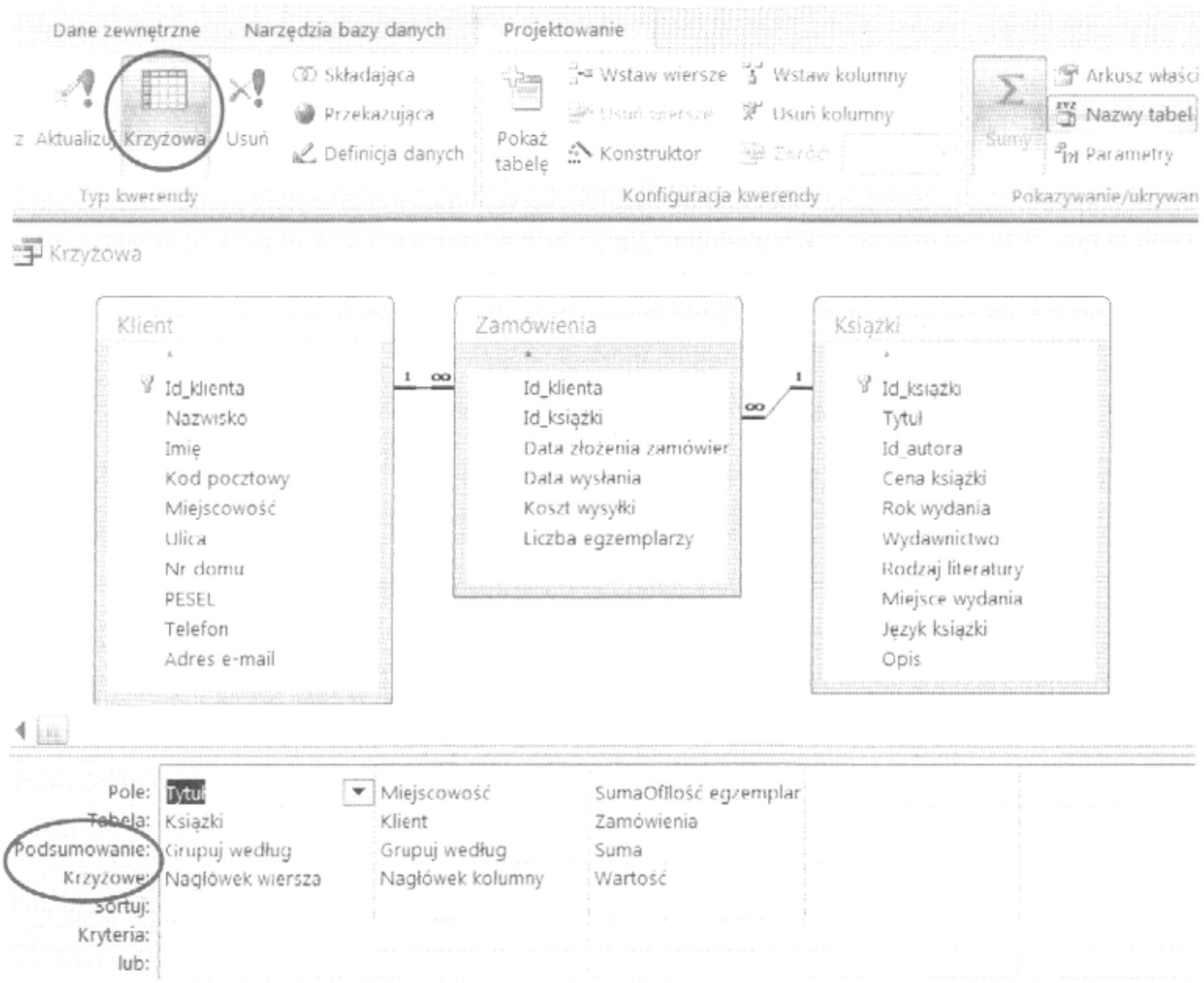
Sprawdź działanie kwerendy, uruchamiając ją kilkakrotnie z różnymi wartościami parametrów dla pól *Nazwisko* i *Imię* autora.

Kwerenda krzyżowa

Kwerenda krzyżowa jest odpowiednikiem tablicy dwóch zmiennych. Stosuje się ją zwykle do analizowania danych.

Aby zaprojektować kwerendę krzyżową, należy wybrać tabele, które będą źródłem danych, a następnie w oknie *Projekt kwerendy* określić pola, które będą brały udział w kwerendzie. Po wybraniu pól do kwerendy w obszarze *Narzędzia kwerend*, na karcie *Projektowanie*, w grupie *Typ kwerendy*, trzeba kliknąć ikonę *Krzyżowa*. W szablonie pojawią się dwa nowe wiersze: *Podsumowanie* i *Krzyżowe* (rysunek 2.18).

Rysunek 2.18. Projekt kwerendy krzyżowej



W wierszu *Podsumowanie* należy wybrać funkcje lub działania dla poszczególnych pól kwerendy. W wierszu *Krzyżowe* trzeba określić, które pole będzie źródłem wierszy, a które źródłem kolumn w wynikowym arkuszu danych. Utworzona kwerenda działa podobnie jak tabela przestawna w arkuszu kalkulacyjnym.

Przykład 2.11

Utwórz kwerendę wyświetlającą informację, ile egzemplarzy każdego tytułu zostało sprzedanych w miejscowościach, z których pochodzą klienci księgarni internetowej. Wynikiem tego zadania powinien być arkusz danych zawierający w kolumnach nazwy miejscowości, a w wierszach tytuły książek. Na przecięciu wiersza i kolumny pojawi się wartość określająca liczbę egzemplarzy każdego tytułu sprzedanych w określonej miejscowości.

W projekcie kwerendy wybierz jako źródło danych tabele *Klient*, *Książki* i *Zamówienia*. Następnie określ pola, które będą brały udział w kwerendzie (*Miejscowość*, *Tytuł*, *Liczba egzemplarzy*). Kliknij na wstążce ikonę *Krzyżowa*. W szablonie pojawią się dwa nowe wiersze: *Podsumowanie* i *Krzyżowe*. W wierszu *Podsumowanie* wybierz w polach *Miejscowość* oraz *Tytuł* funkcję *Grupowanie według*, a w polu *Liczba egzemplarzy* — funkcję *Suma*, aby zsumować egzemplarze tego samego tytułu. W wierszu *Krzyżowe* określ, które pole będzie źródłem wierszy, a które źródłem kolumn w wynikowym arkuszu danych. Dla pola *Liczba egzemplarzy*, na którym dokonujemy obliczeń, wybierz opcję *Wartość*. Sprawdź działanie kwerendy. Przykładowy rezultat zaprezentowano na rysunku 2.19.

Tytuł	Poznań	Szczecin	Toruń	Warszawa
Antygona		2		1
Balladyna			2	7
Chłopi	2			
Dziady		4		
Faraon	2			5
Fraszki	6			

Rysunek 2.19. Rezultat kwerendy krzyżowej

2.4.2. Wybór typu sprzężenia w kwerendzie

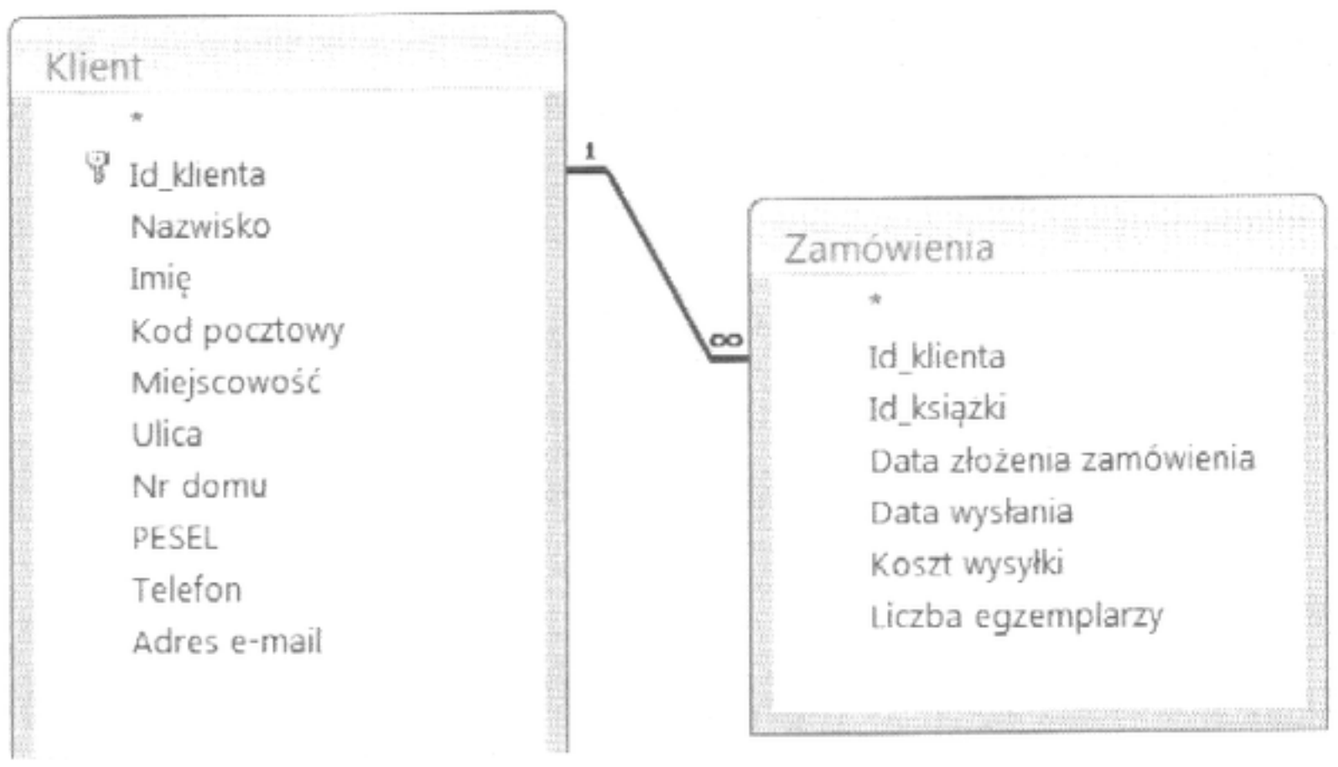
Zdefiniowanie *relacji* między tabelami jest konieczne do budowania prawidłowych kwerend odwołujących się do kilku tabel w bazie danych. Zdefiniowane wcześniej połączenia między tabelami są widoczne w oknie projektu kwerendy. Jeżeli relacje między tabelami nie zostały zdefiniowane, można je określić w oknie projektu kwerendy, należy tylko pamiętać, że tak zdefiniowane połączenia będą działały tylko w projektowanej kwerendzie. W innym przypadku kwerenda nie będzie działała prawidłowo. Również na potrzeby projektowanej kwerendy można usunąć wcześniej zdefiniowane połączenie między tabelami.

Domyślnie zdefiniowane połączenie powoduje wybranie w kwerendzie tylko tych rekordów tabel źródłowych, dla których wartości połączonych pól są równe. Oznacza to, że projektowana kwerenda wyświetli na przykład listę tylko tych klientów, którzy dokonali już zakupów w księgarni internetowej. Jeżeli klient zarejestrował się w bazie, ale nie złożył jeszcze zamówienia, jego nazwisko nie pojawi się na liście. Rozwiązaniem problemu jest zmiana atrybutów połączenia.

Istnieją trzy podstawowe typy połączeń:

- **Połączenie zawężające** (wewnętrzne lub równopołączenie) — oznacza wybranie w kwerendzie tylko tych rekordów tabel źródłowych, dla których wartości połączonych pól są równe. Ten rodzaj połączenia jest definiowany domyślnie (rysunek 2.20).

Rysunek 2.20. Połączenie zawężające



- **Połączenie rozszerzające** (zewnętrzne) — można wyodrębnić połączenie zewnętrzne lewe i prawe. W połączeniu zewnętrznym lewym kwerenda uwzględnia wszystkie

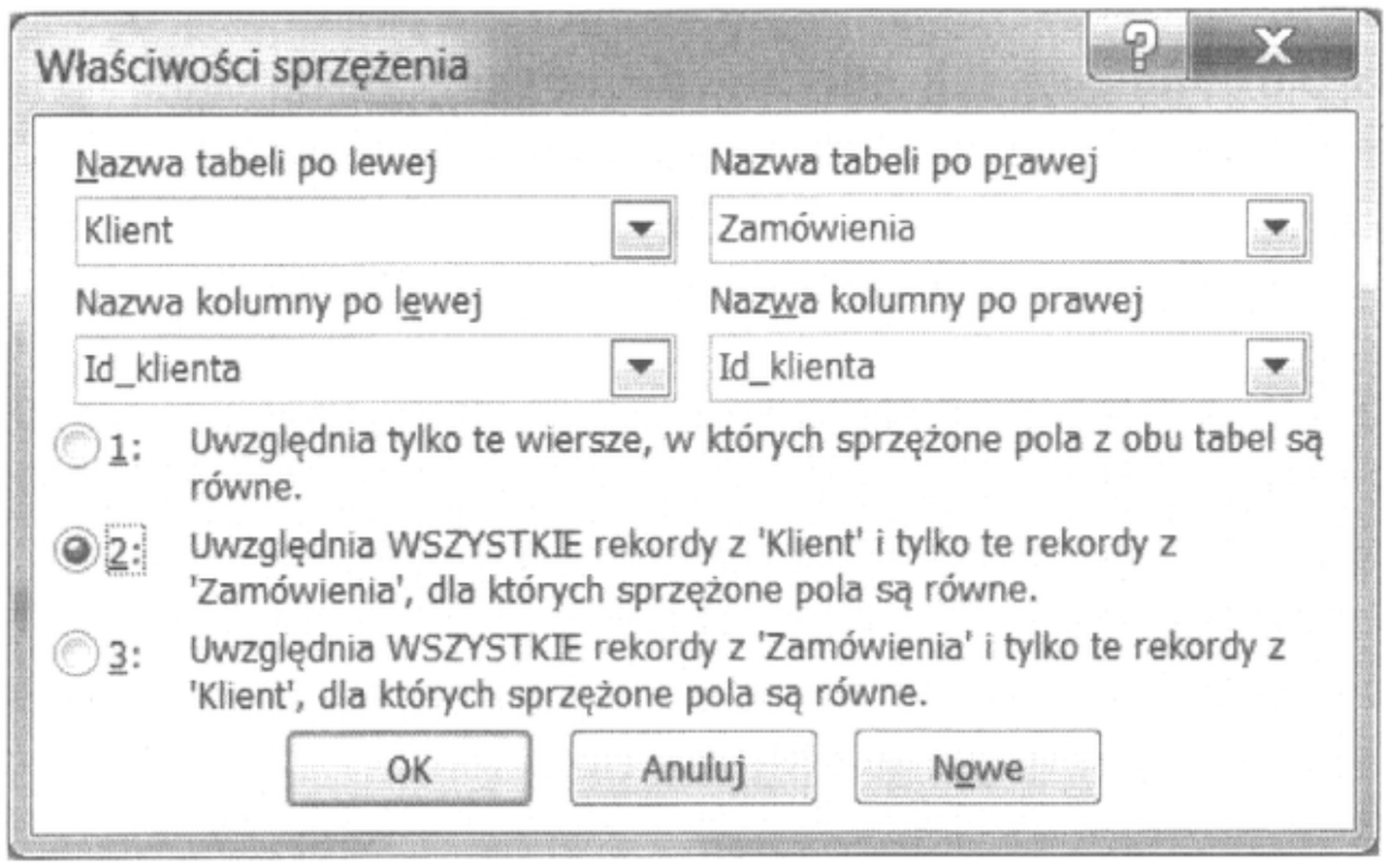
wiersze pierwszej tabeli oraz tylko te wiersze drugiej tabeli, w których pole łączące zawiera wartości wspólne dla obu tabel. W połączeniu zewnętrznym prawym kwerenda uwzględnia wszystkie wiersze drugiej tabeli oraz tylko te wiersze pierwszej tabeli, w których pole łączące zawiera wartości wspólne dla obu tabel. Zmiana rodzaju sprzężenia odbywa się w projekcie kwerendy we właściwościach połączenia.

- **Samopołączenie** — pozwala połączyć dwa pola tej samej tabeli, aby odzwierciedlić hierarchiczną strukturę danych. Samopołączenie wymaga umieszczenia w oknie projektu kwerendy kilku egzemplarzy tej samej tabeli.

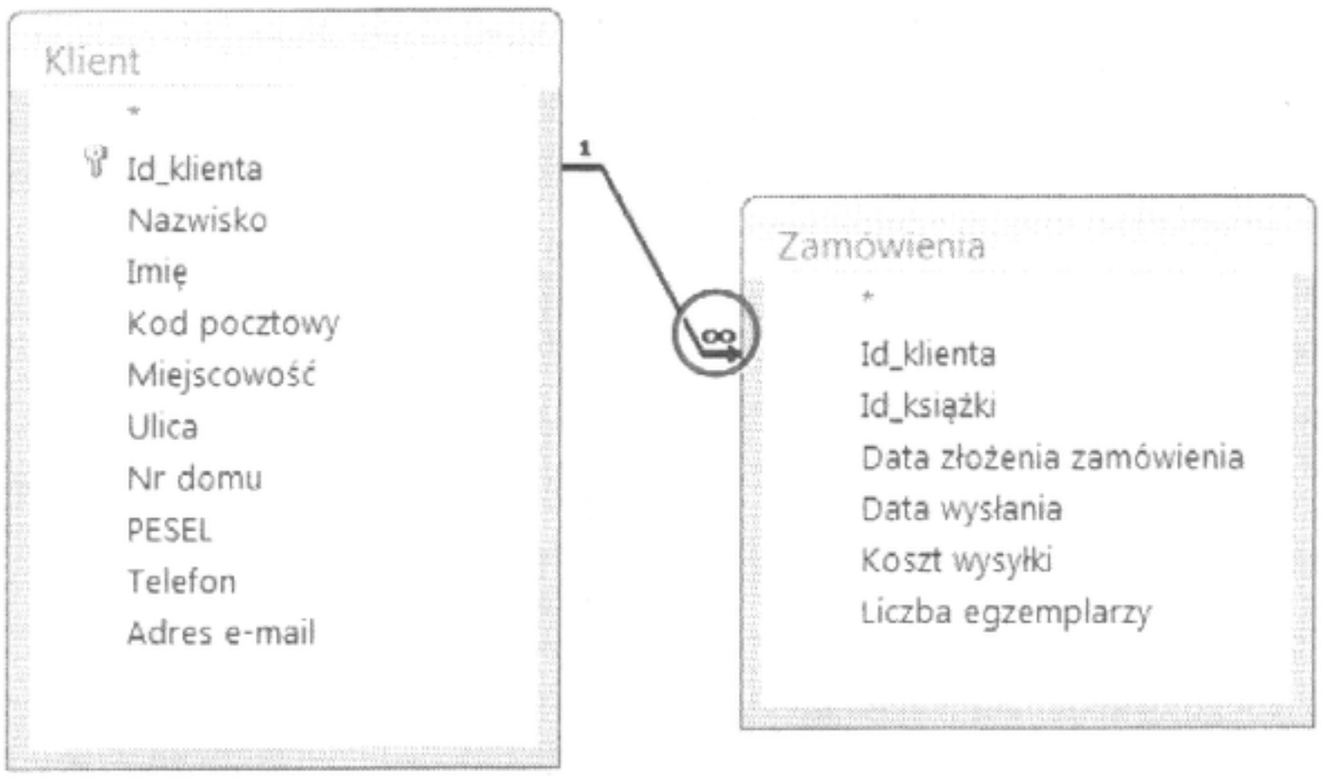
Po dwukrotnym kliknięciu myszą połączenia między tabelami widocznego w projekcie kwerendy zostanie wyświetlone okno *Właściwości sprzężenia* (rysunek 2.21). Mamy do wyboru połączenie zawężające lub rozszerzające.

Jeśli zostanie zdefiniowane połączenie rozszerzające, w oknie projektowania kwerendy pojawi się specjalny symbol (rysunek 2.22).

Rysunek 2.21.
Okno
Właściwości sprzężenia



Rysunek 2.22.
Sprzężenie rozszerzające



Przykład 2.12

Zaprojektuj kwerendę wyświetlającą listę wszystkich klientów księgarni internetowej wraz z tytułami książek, które zamówili. Na liście klientów powinni znaleźć się również klienci, którzy zarejestrowali się w księgarni, ale jeszcze nie złożyli żadnego zamówienia (rysunek 2.23). Jeżeli utworzymy kwerendę wybierającą dla tabel *Klient* i *Zamówienia*,

domyślnie zostanie wybrane połączenie zawężające między tabelami i klienci, którzy nie złożyli żadnego zamówienia, zostaną pominięci na liście (rysunek 2.24).

Rysunek 2.23. Niektórzy klienci nie złożyli zadnego zamówienia

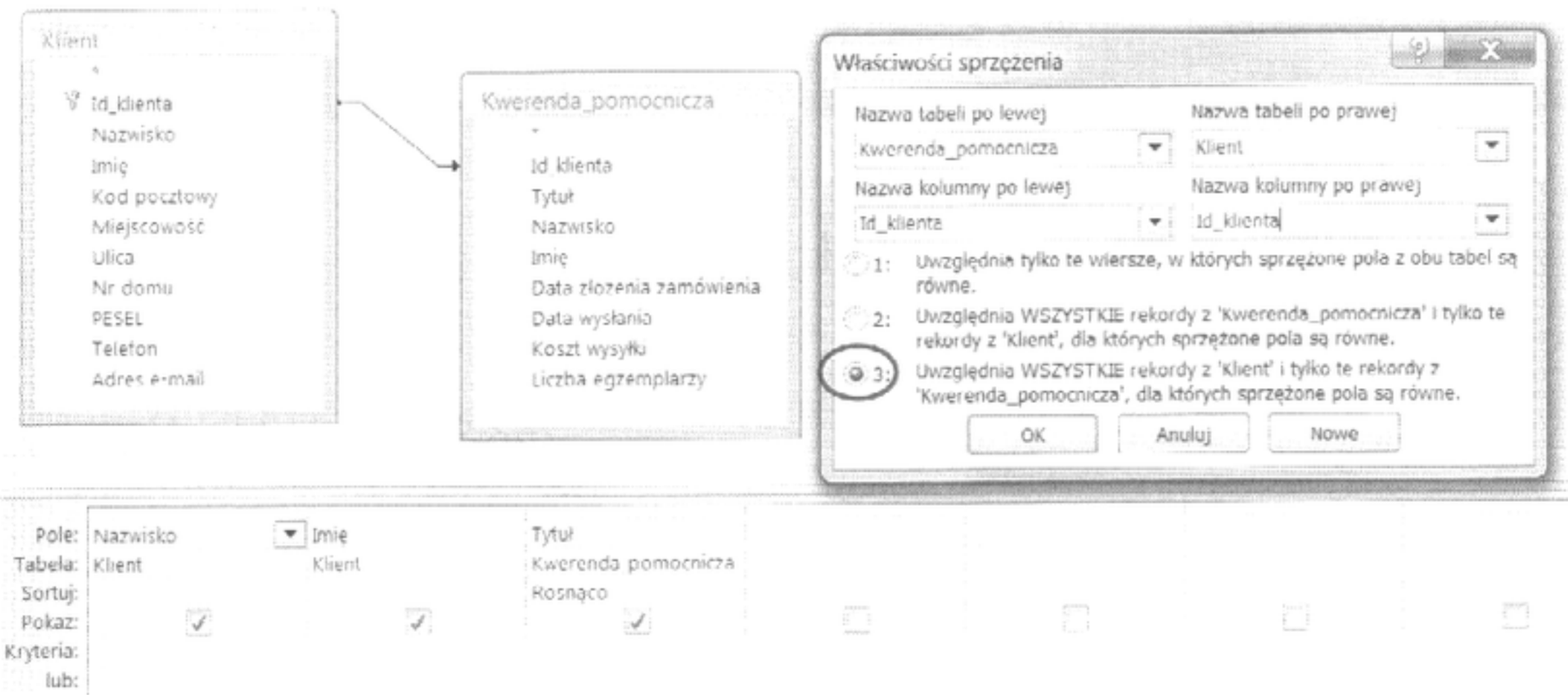
Nazwisko klienta	Imię klienta	Tytuł
Bagińska	Anna	
Zan	Marcin	
Kowalski	Adam	Antygona
Pol	Aleksander	Antygona
Nowak	Marek	Balladyna
Kowalski	Adam	Balladyna
Górecki	Grzegorz	Chłopi
Pol	Aleksander	Dziady
Białoszewski	Mirek	Faraon
Górecki	Grzegorz	Faraon
Górecki	Grzegorz	Fraszki

Rysunek 2.24. Klienci, którzy nie złożyli żadnego zamówienia, nie pojawili się na liście

Nazwisko klienta	Imię klienta	Tytuł
Pol	Aleksander	Antygona
Kowalski	Adam	Antygona
Kowalski	Adam	Balladyna
Nowak	Marek	Balladyna
Górecki	Grzegorz	Chłopi
Pol	Aleksander	Dziady
Białoszewski	Mirek	Faraon
Górecki	Grzegorz	Faraon
Górecki	Grzegorz	Fraszki

W pierwszym etapie utwórz kwerendę pomocniczą, dla której źródłem będą tabele *Zamówienia* i *Książki* z polami *Id klienta* z tabeli *Zamówienia* oraz *Tytuł* z tabeli *Książki*. Kwerendę nazwij na przykład *Pomocnicza*.

Następnie zaprojektuj właściwą kwerendę, której źródłem danych będzie tabela *Klient* oraz kwerenda *Pomocnicza*. Określ pola, które będą brały udział w kwerendzie, czyli *Nazwisko* i *Imię* z tabeli *Klient* oraz *Tytuł* z kwerendy *Pomocnicza*. W projekcie kwerendy kliknij dwukrotnie połączenie między tabelą *Klient* i kwerendą *Pomocnicza*. Zostanie wyświetlone okno dialogowe *Właściwości sprzężenia*. Wybierz sprzężenie rozszerzające, które uwzględnia wszystkie rekordy z tabeli *Klient* (rysunek 2.25).



Rysunek 2.25. Wybór sprzężenia rozszerzającego

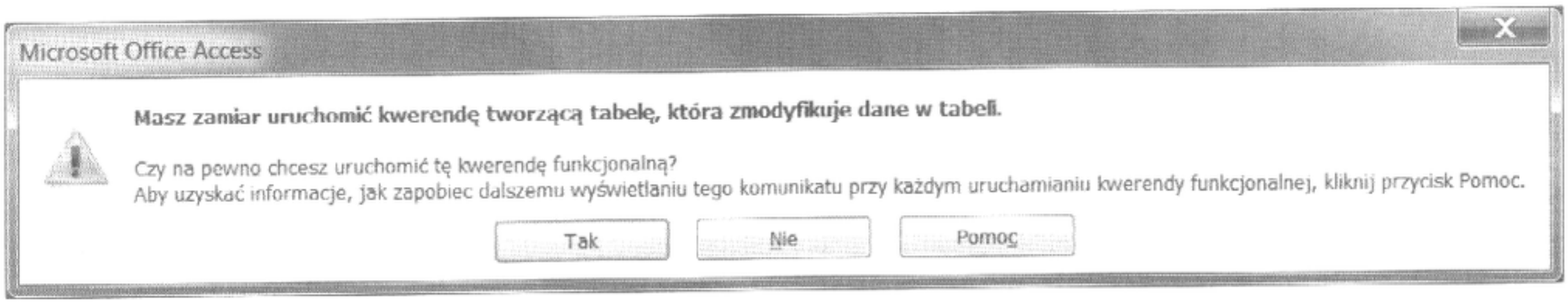
Po zmianie rodzaju połączenia na *rozszerzające* na liście znajdą się wszyscy klienci, niezależnie od tego, czy zamówili książki, czy nie.

2.4.3. Kwerendy funkcjonalne

Kwerendy wybierające są tworzone domyślnie. Dopóki nie zdecydujemy się na inny rodzaj kwerendy, każda nowa kwerenda będzie kwerendą wybierającą.

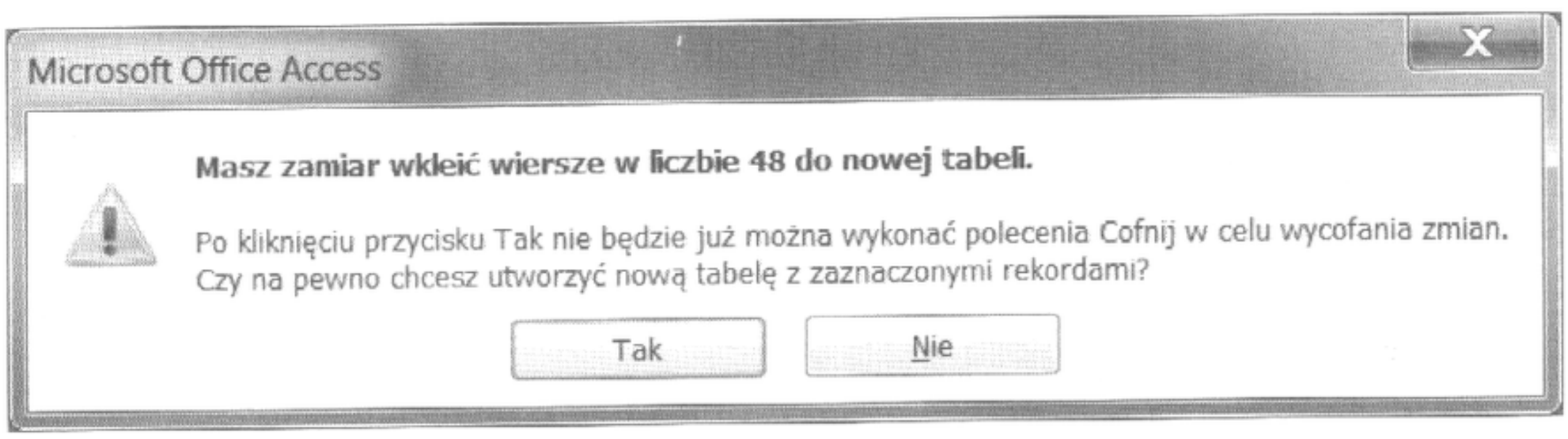
Kwerendy funkcjonalne, oprócz wybierania określonej grupy rekordów, wykonują na nich pewne operacje. Najczęściej jest to zmiana danych w tabeli. Zmian spowodowanych uruchomieniem kwerendy funkcjonalnej nie można w prosty sposób cofnąć, dlatego przed wykonaniem takiej kwerendy należy upewnić się, jak działa wybrana kwerenda i czy mamy kopię zapasową bazy danych.

Program Access ma wbudowane narzędzia do kontrolowania pracy z kwerendami funkcjonalnymi. Są to komunikaty, które pojawiają się w trakcie uruchamiania kwerendy. Pierwszy z nich informuje, że zostanie wykonana kwerenda funkcjonalna, która może zmodyfikować dane w tabeli (rysunek 2.26).



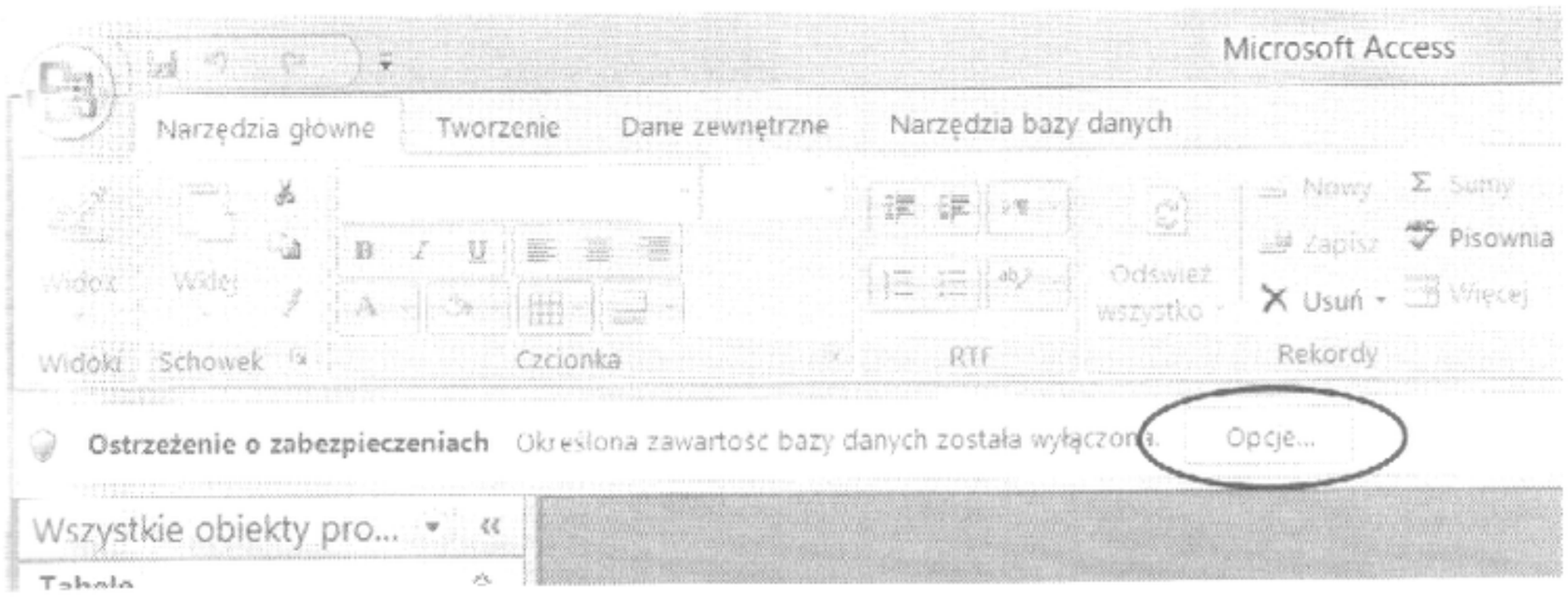
Rysunek 2.26. Komunikat ostrzegający o możliwości zmodyfikowania danych w tabeli

Drugi informuje, jakie zmiany zostaną dokonane w tabeli (rysunek 2.27).



Rysunek 2.27. Komunikat informujący o rodzaju zmian w tabeli

Program Access domyślnie uniemożliwia wykonywanie kwerend funkcjonalnych. Aby włączyć możliwość wykonywania tych kwerend, należy na pasku komunikatów kliknąć przycisk *Opcje* (rysunek 2.28) i w wyświetlonym oknie zaznaczyć opcję *Włącz tę zawartość*.



Rysunek 2.28. Pasek komunikatów

Jeśli nie widać paska komunikatów, trzeba wybrać kartę *Narzędzia bazy danych* i w grupie *Pokazywanie/ukrywanie* zaznaczyć opcję *Pasek komunikatów*.

Wyróżnia się cztery rodzaje kwerend funkcjonalnych:

- kwerendę tworzącą tabele,
- kwerendę aktualizującą,
- kwerendę dołączającą,
- kwerendę usuwającą.

Kwerendy funkcjonalne są projektowane podobnie jak kwerendy wybierające. Po kliknięciu na karcie *Tworzenie*, w grupie *Inne*, ikony *Projekt kwerendy* zostanie otwarte okno projektu kwerendy. Dla kwerend funkcjonalnych należy w obszarze *Narzędzia kwerend*, na karcie *Projektowanie*, w grupie *Typy kwerend*, kliknąć ikonę odpowiedniego rodzaju kwerendy.

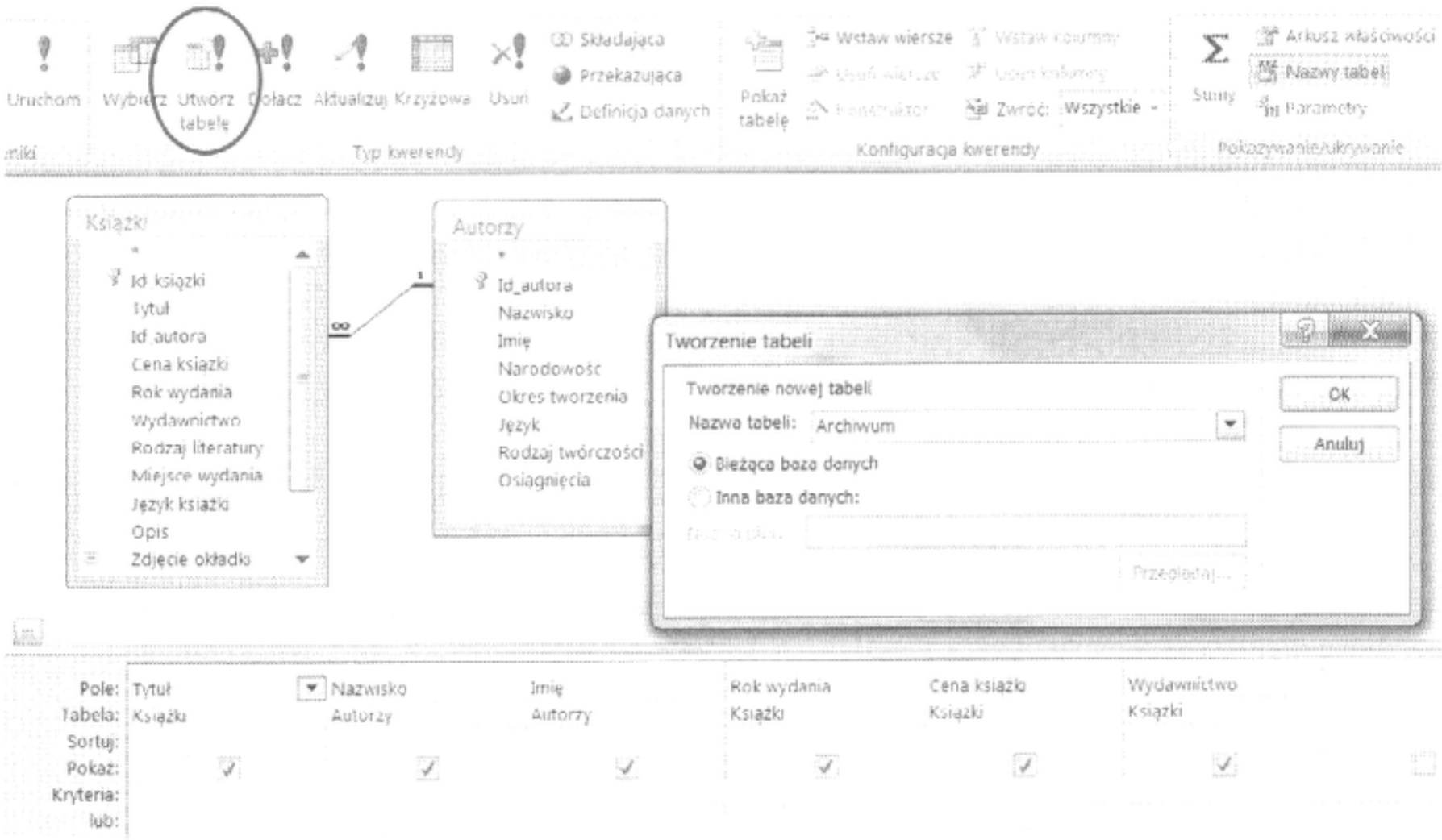
Kwerenda tworząca tabelę

W wyniku uruchomienia kwerendy następuje utworzenie nowej tabeli zawierającej pola i rekordy określone przez kwerendę. Tego typu kwerendy najczęściej stosujemy w celu utworzenia kopii zapasowej lub zarchiwizowania danych.

Przykład 2.13

Zaprojektuj kwerendę przenoszącą informacje o książkach (*tytuł*, *nazwisko* i *imię autora*, *cenę* i *nazwę wydawnictwa*) wydanych przed rokiem 2000 do tabeli *Archiwum*. Wynikiem wykonania tej kwerendy powinna być nowa tabela, zawierająca wybrane przez nas informacje z tabel *Książki* i *Autorzy*.

Kwerendę zaprojektuj podobnie jak kwerendę wybierającą dla danych z tabel *Książki* i *Autorzy*. Po jej zaprojektowaniu na karcie *Projektowanie* w grupie *Typ kwerendy* wybierz ikonę *Utwórz tabelę*. W oknie, które się pojawi, wpisz nazwę nowej tabeli, czyli *Archiwum* (rysunek 2.29).



Rysunek 2.29. Projekt kwerendy tworzącej tabelę

Zapisz kwerendę. Na liście kwerenda tworząca tabelę została oznakowana specjalną ikoną . Sprawdź działanie kwerendy.

Kwerenda aktualizująca

W wyniku uruchomienia kwerendy tego typu następuje zaktualizowanie wybranych za pomocą kwerendy pól i rekordów. Kwerendy aktualizujące stosujemy, gdy chcemy zmienić dane w pewnej grupie rekordów według tego samego wzoru.

Przykład 2.14

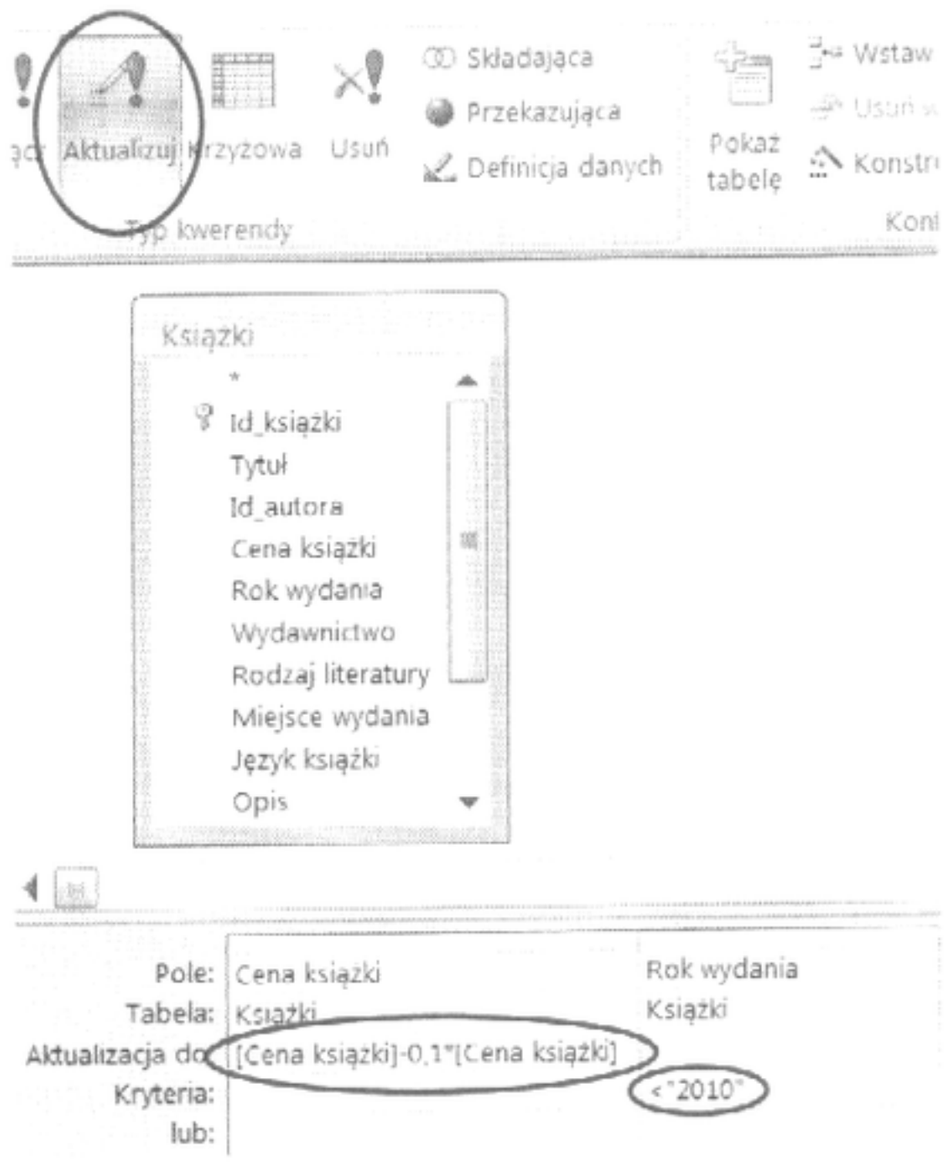
Zaprojektuj kwerendę, która w tabeli *Książki* obniży cenę książek wydanych przed rokiem 2010 o 10%. Pamiętaj, że skutków działania kwerendy aktualizującej nie można cofnąć.

Podobnie jak w poprzednim przykładzie, projektowanie kwerendy zacznij od utworzenia kwerendy wybierającej dla tabeli *Książki*. Po jej zaprojektowaniu na karcie *Projektowanie* w grupie *Typ kwerendy* wybierz ikonę *Aktualizuj*. W projekcie kwerendy zostanie dodany nowy wiersz *Aktualizacja do* — wpisz w nim dla pola *Cena książki* wyrażenie realizujące aktualizację danych (`[Cena książki]-0,1*[Cena książki]`). Aby aktualizacja dotyczyła wybranych rekordów, w polu *Rok wydania* zdefiniuj kryterium, które tylko książkom wydanym przed rokiem 2010 zmieni wartość pola *Cena książki*: `<2010` (rysunek 2.30).

Zapisz projekt kwerendy. Na liście kwerend kwerenda aktualizująca została oznakowana specjalną ikoną . Sprawdź działanie kwerendy. Pamiętaj o tym, że każde uruchomienie kwerendy będzie modyfikowało zawartość pola *Cena książki*.

Kwerenda dołączająca

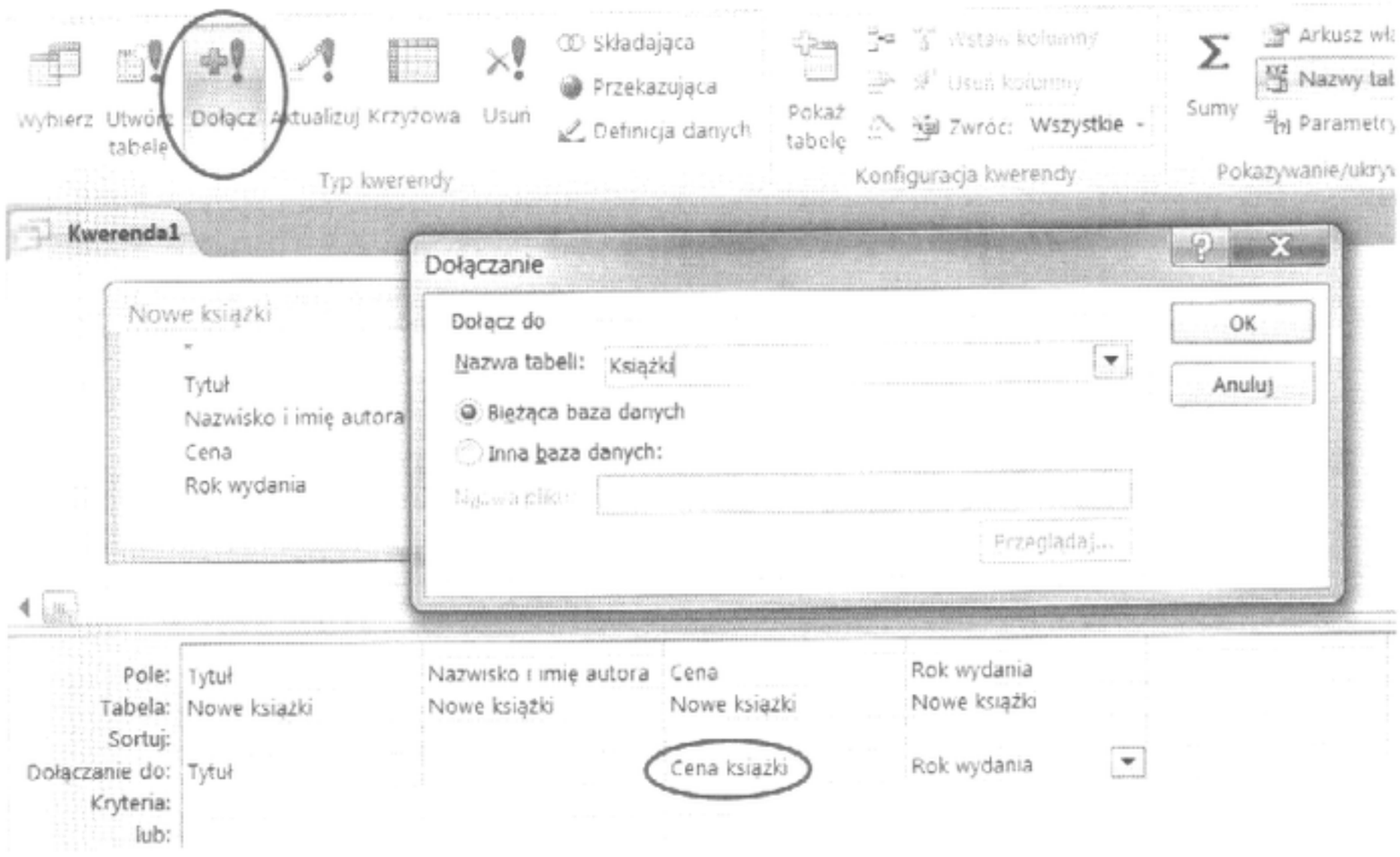
W wyniku uruchomienia kwerendy następuje dołączenie do wybranej tabeli rekordów innej tabeli. Tego typu kwerendy stosujemy w celu dodania nowych rekordów do istniejącej tabeli.



Rysunek 2.30. Projekt kwerendy aktualizującej

Przykład 2.15

Zaprojektuj kwerendę, która do tabeli *Książki* dołączy dane o książkach zapisane w tabeli pomocniczej *Nowe książki*. Tabela *Nowe książki* zawiera informacje o nowych książkach dostępnych w księgarni internetowej (*Tytuł*, *Nazwisko i imię autora*, *Cena*, *Rok wydania*). Projektowanie kwerendy zacznij od utworzenia kwerendy wybierającej. Źródłem kwerendy jest tabela *Nowe książki*, z której dane zostaną dołączone do tabeli *Książki*. Po zaprojektowaniu kwerendy na karcie *Projektowanie* w grupie *Typ kwerendy* wybierz ikonę *Dołącz*. W oknie, które się pojawi, wybierz z listy tabelę *Książki* (rysunek 2.31). W projekcie kwerendy pola, które mają takie same nazwy, zostaną automatycznie skojarzone. Pole *Cena* trzeba samodzielnie skojarzyć z polem *Cena książki*, natomiast dane z pola *Nazwisko i imię autora* nie zostaną dołączone do tabeli *Książki*, ponieważ nie ma w niej pola, z którym można by skojarzyć te dane.



Rysunek 2.31. Projekt kwerendy dołączającej

Zapisz projekt kwerendy. Na liście kwerend kwerenda dołączająca została oznakowana specjalną ikoną . Sprawdź działanie kwerendy. Pamiętaj o tym, że każde uruchomienie kwerendy będzie dołączało do tabeli *Książki* dane z tabeli *Nowe książki*.

Kwerenda usuwająca

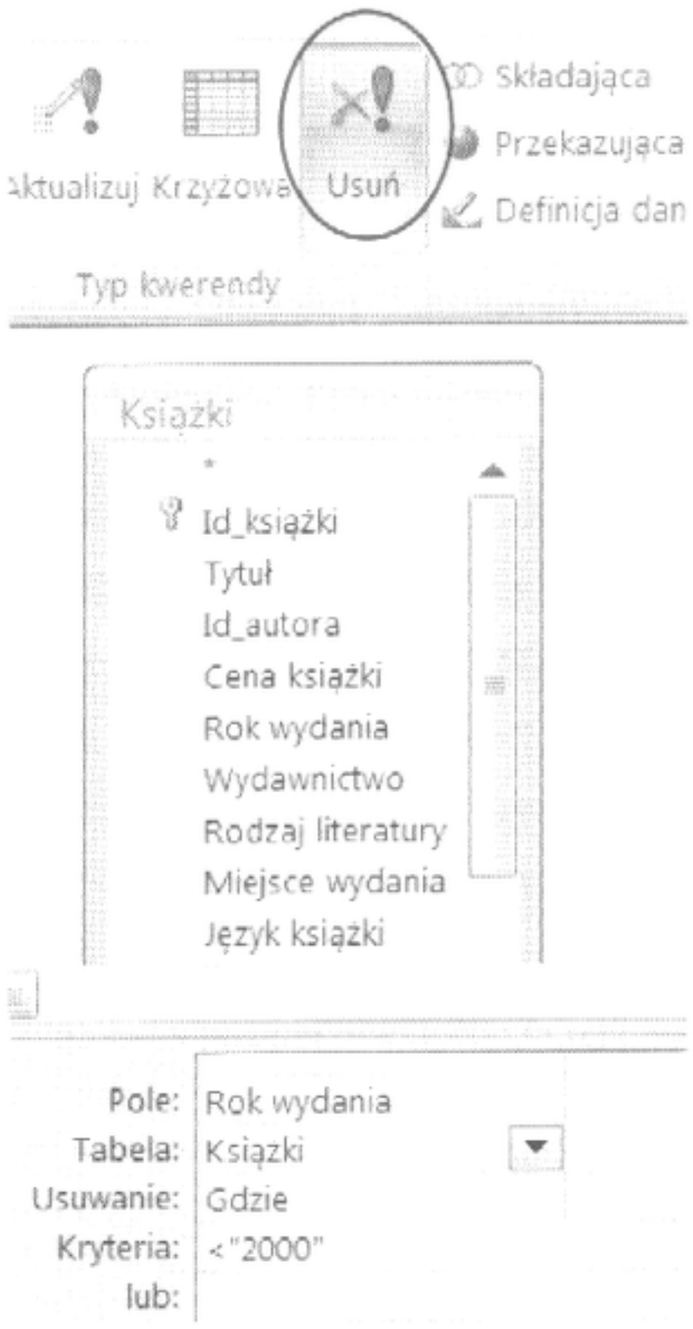
W wyniku uruchomienia tej kwerendy następuje usunięcie z tabeli wybranych rekordów.

Przykład 2.16

Zaprojektuj kwerendę usuwającą z tabeli *Książki* dane o książkach wydanych przed rokiem 2000 (dane te zostały umieszczone w tabeli *Archiwum*).

Projektowanie kwerendy zacznij od utworzenia kwerendy wybierającej dla tabeli *Książki*. Po jej zaprojektowaniu na karcie *Projektowanie* w grupie *Typ kwerendy* wybierz ikonę *Usuń* (rysunek 2.32).

Rysunek 2.32.
Projekt kwerendy usuwającej

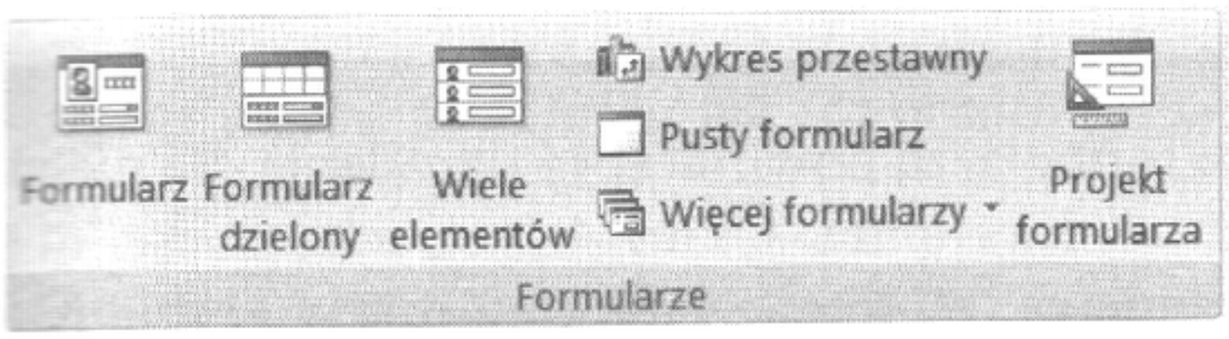


W kryteriach dla pola *Rok wydania* wpisz warunek usuwania rekordów (<2000). Pamiętaj, że operacja ta jest nieodwracalna. Jeżeli nie zostaną zdefiniowane kryteria, z tabeli będą usunięte wszystkie rekordy. Zapisz projekt kwerendy. Na liście kwerend kwerenda usuwająca została oznakowana specjalną ikoną . Sprawdź działanie kwerendy.

2.5. Formularze

Formularz służy do prezentacji danych z tabeli lub kwerendy w sposób graficzny. Można go zaprojektować do wprowadzania, edytowania lub wyświetlania danych, za jego pomocą można również sterować dostępem do danych. Formularze zapewniają użytkownikowi bazy danych dostęp do zgromadzonych danych.

Nowy formularz w bazie danych może być tworzony na różne sposoby. Na karcie *Tworzenie* w grupie *Formularze* należy wybrać ikonę określającą sposób jego budowania (rysunek 2.33).

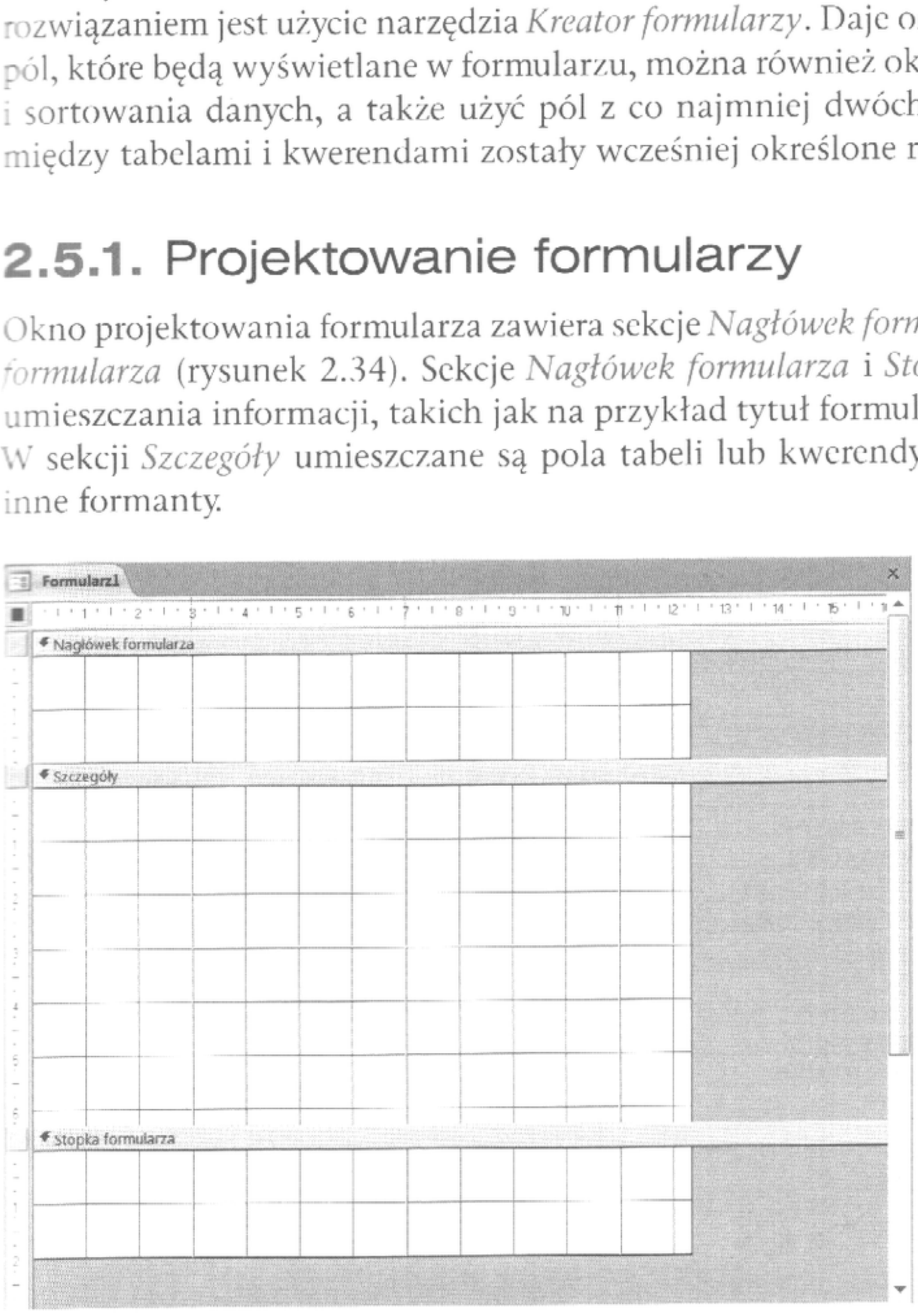


Rysunek 2.33. Opcje tworzenia formularza

Dla użytkowników mało doświadczonych w projektowaniu formularzy najlepszym rozwiązaniem jest użycie narzędzia *Kreator formularzy*. Daje ono możliwość wybierania pól, które będą wyświetlane w formularzu, można również określić sposób grupowania i sortowania danych, a także użyć pól z co najmniej dwóch tabel lub kwerend, jeśli między tabelami i kwerendami zostały wcześniej określone relacje.

2.5.1. Projektowanie formularzy

Okno projektowania formularza zawiera sekcje *Nagłówek formularza*, *Szczegóły* i *Stopka formularza* (rysunek 2.34). Sekcje *Nagłówek formularza* i *Stopka formularza* służą do umieszczania informacji, takich jak na przykład tytuł formularza czy numeracja stron. W sekcji *Szczegóły* umieszczane są pola tabeli lub kwerendy, pola obliczeniowe oraz inne formanty.



Rysunek 2.34. Widok Projekt formularza

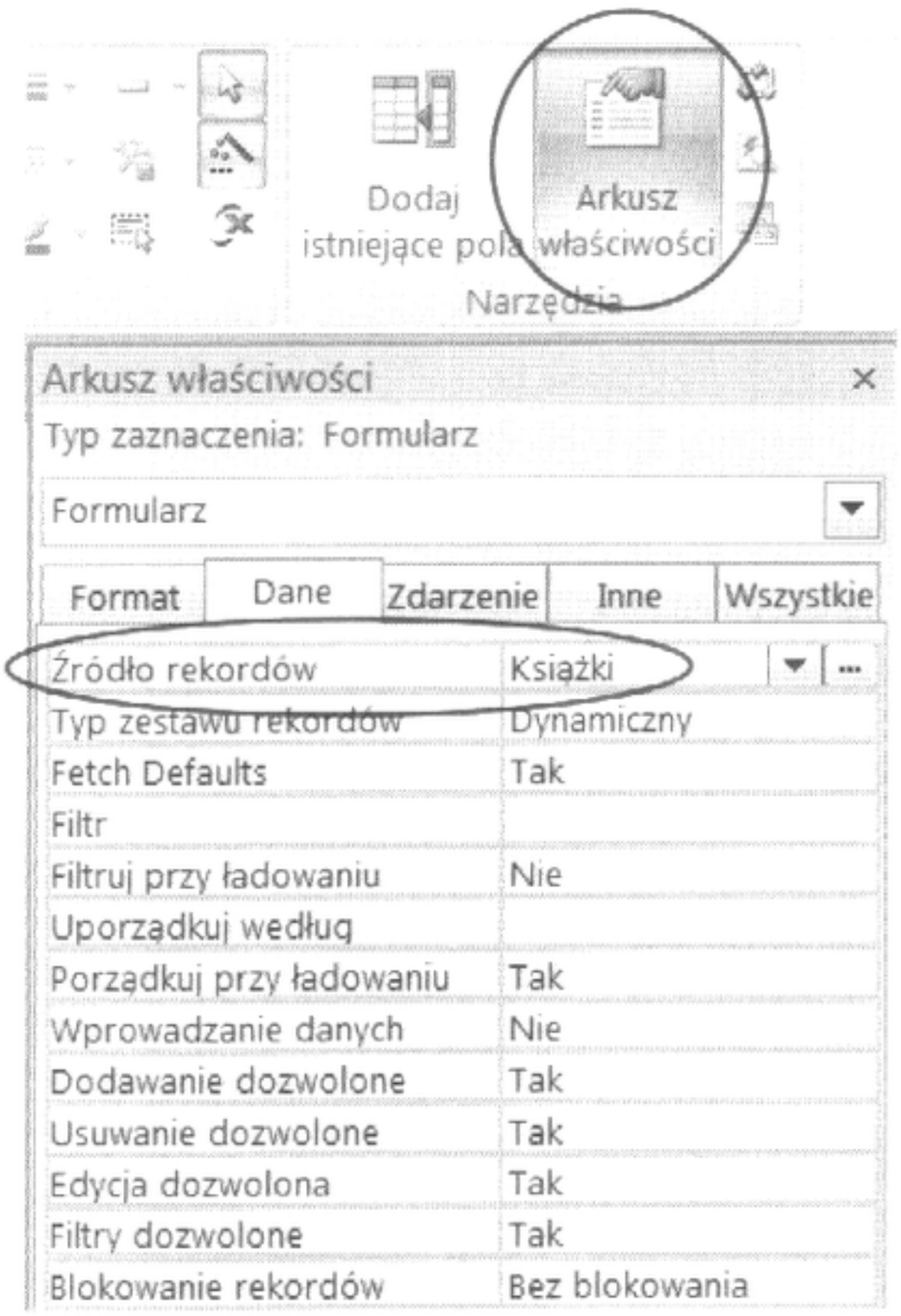
Widok *Projekt formularza* umożliwia dodawanie do formularza różnych obiektów, takich jak: pola tekstowe, etykiety, obrazy, linie i prostokąty, przyciski poleceń i pola kombi. Obiekty umieszczane w formularzu nazywamy **formantami**. W tym widoku można również zmieniać właściwości formularza lub ustawiać właściwości formantów. Siatka i linijka widoczne w projekcie ułatwiają rozmieszczenie formantów.

Pierwszym krokiem przy projektowaniu formularza powinno być określenie dla niego źródła rekordów. Może nim być tabela lub kwerenda. Okno *Arkusz właściwości* dla formularza umożliwi ustalenie tego parametru. Na karcie *Projektowanie* w grupie *Narzędzia* należy wybrać przycisk *Arkusz właściwości*. We właściwości *Źródło rekordów* trzeba z listy wybrać odpowiednią tabelę lub kwerendę (rysunek 2.35).

Najprostszym sposobem umieszczania w projekcie formularza pola tabeli lub kwerendy jest jego przeciąganie bezpośrednio z okna *Lista pól* do odpowiedniej sekcji formularza. Aby otworzyć to okno, na karcie *Projektowanie* w grupie *Narzędzia* należy wybrać przycisk *Dodaj istniejące pola*.

Do budowania obiektów formularza (formantów) służą narzędzia z grupy *Formanty* dostępne na karcie *Projektowanie* (rysunek 2.36).

Rysunek 2.35. Definiowanie źródła rekordów



Rysunek 2.36. Formanty

Aby utworzyć nowy formant w projekcie formularza, należy wybrać ikonę odpowiadającą typowi formantu, który ma być dodany, i w siatce projektu formularza narysować formant o wymaganych wymiarach. Do projektowania niektórych formantów można użyć narzędzia *Kreator formantów*. Jeśli na karcie *Projektowanie* w grupie *Formanty* nie jest zaznaczona opcja *Użyj kreatorów formantów*, trzeba włączyć tę opcję, klikając ikonę *Użyj kreatorów formantów*  . Za pomocą *kreatora formantów* można tworzyć przyciski poleceń, pola listy, podformularze, pola kombi i grupy opcji.

Za pomocą narzędzi z grupy *Formanty* na karcie *Projektowanie* można dodać do formularza logo, tytuł, numerację stron lub datę i godzinę.

2.5.2. Formanty

Formanty są obiektami, które służą do wyświetlania danych (pola tekstowe) i wykonywania akcji (przyciski poleceń); zwiększają one funkcjonalność formularza (etykiety, pola wyboru i formanty podformularza czy podraportu), pozwalają na pokazywanie elementów graficznych (obrazy, linie, prostokąty). Najczęściej używanym formantem jest pole tekstowe.

Formanty mogą być powiązane, niepowiązane i obliczeniowe.

- **Formant powiązany** — jego źródłem danych jest pole tabeli lub kwerendy. Formanty powiązane służą do wyświetlania wartości pól bazy danych. Pole tekstowe w formularzu, które wyświetla nazwisko klienta, uzyskuje te informacje z pola *Nazwisko* w tabeli *Klient*.
- **Formant niepowiązany** — do tego formantu nie przypisano źródła danych (na przykład pola lub wyrażenia). Formantów niepowiązanych można używać do wyświetlania informacji, linii, prostokątów i obrazów. Przykładem formantu niepowiązanego jest etykieta, w której wyświetla się tytuł formularza.
- **Formant obliczeniowy** — źródłem danych jest wyrażenie, a nie pole. Wartość, która stanowi źródło danych formantu, określana jest przez zdefiniowane wyrażenie.

Etykiety

Etykiety są to formanty, które wyświetlają treść tekstową, na przykład tytuł, nazwę lub opis pola. Jeżeli w formularzu ma się pojawić dowolny tekst, zapisujemy go w etykiecie. Można zmieniać zarówno treść etykiet, jak i ich wygląd.

Pole tekstowe

Pole tekstowe służy do przeglądania i edytowania danych. W polach tekstowych są wyświetlane dane z pól tabeli lub kwerendy. Za pomocą pól tekstowych można również wykonywać obliczenia.

Aby w polu tekstowym przedstawić dane pochodzące z tabeli lub kwerendy, należy utworzyć powiązane pole tekstowe. Najlepszym sposobem na utworzenie powiązanego pola tekstowego jest przeciągnięcie pola z okna *Lista pól* do formularza. Access

automatycznie utworzy pola tekstowe dla pól o typach danych, takich jak: *Tekst*, *Nota*, *Liczba*, *Data/Godzina*, *Waluta*, *Hiperłącze*. Przeciąganie pól innych typów danych powoduje tworzenie formantów innych rodzajów. Należy pamiętać, że zmiany wprowadzone w polu tekstowym zostaną odzwierciedlone w tabeli źródłowej.

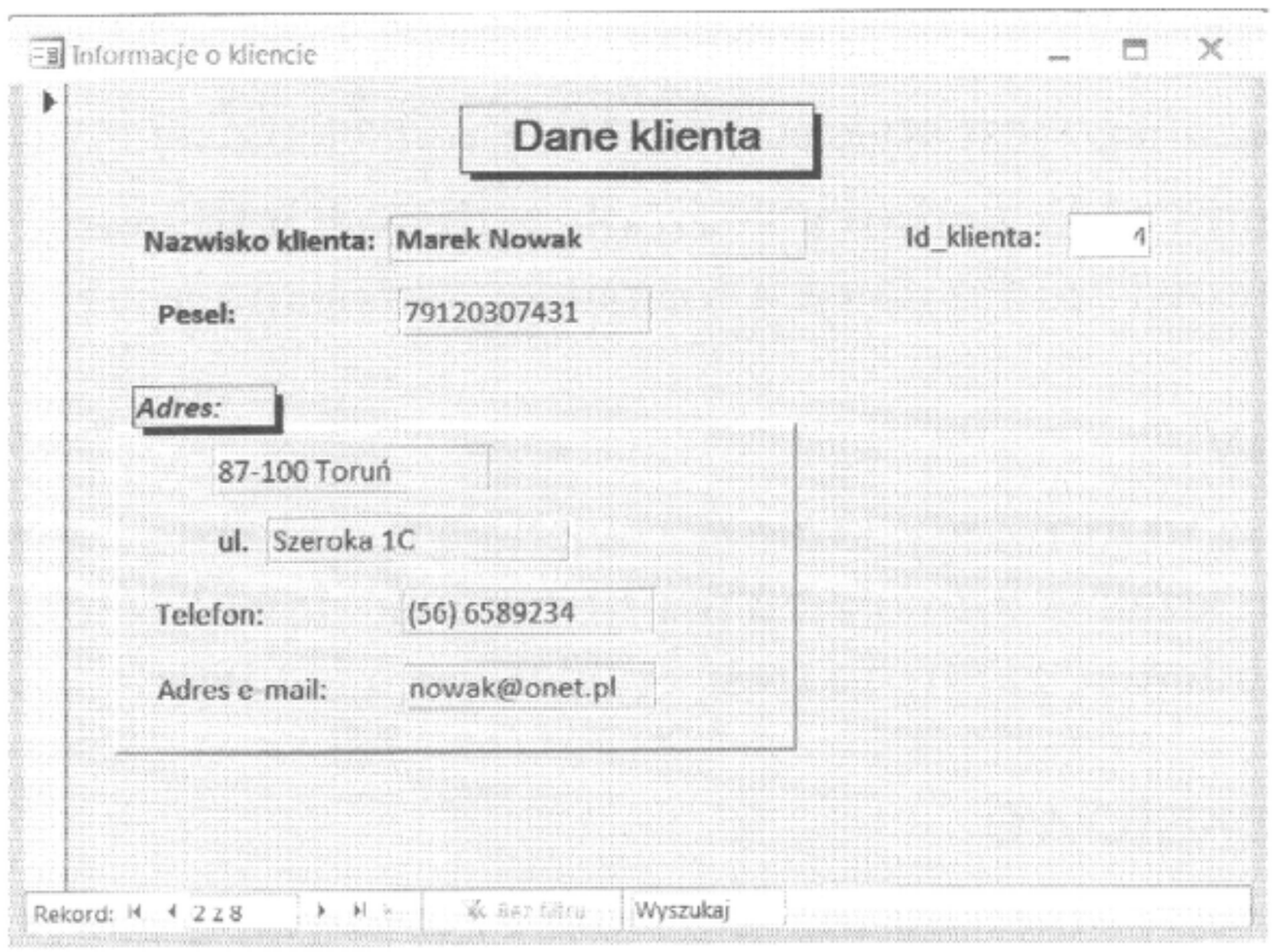
Inną metodą tworzenia pola tekstowego jest wybranie na karcie *Projektowanie* w grupie *Formanty* ikony *Pole tekstowe* **ab|**, a następnie wstawienie tego pola do formularza. Tak utworzone pole tekstowe jest niepowiązane. Aby je powiązać z polem tabeli lub kwerendy, należy we właściwościach pola tekstowego ustawić właściwość *Źródło formantu*. Właściwość ta określa, czy pole tekstowe jest powiązane, niepowiązane, czy obliczeniowe. Jeśli wartość w tej właściwości jest pusta, pole tekstowe jest niepowiązane. Jeżeli zmienimy wartość właściwości *Źródło formantu* na nazwę pola tabeli, pole tekstowe zostanie powiązane z tym polem. Jeśli wartość w tej właściwości jest wyrażeniem, pole tekstowe jest polem obliczeniowym.

Aby pole tekstowe stało się polem obliczeniowym, trzeba umieścić kursor w polu tekstowym i bezpośrednio w tym polu wpisać wyrażenie służące do wykonania obliczeń lub wpisać je we właściwości *Źródło formantu* pola tekstowego. Można również utworzyć wyrażenie za pomocą *konstruktora wyrażeń*, klikając przycisk ... obok pola właściwości *Źródło formantu*.

Przykład 2.17

Dla tabeli *Klient* zaprojektuj formularz zawierający wszystkie pola z tej tabeli. Pola *Nazwisko* i *Imię* powinny zostać połączone i wyświetlone w jednym wspólnym polu (rysunek 2.37).

Rysunek 2.37. Formularz zaprojektowany dla tabeli Klient



Na karcie *Tworzenie* w grupie *Formularze* wybierz ikonę *Projekt formularza*. We właściwościach formularza wybierz *Źródło rekordów* i na liście kliknij tabelę *Klient*.

Tytuł formularza *Dane klienta* zostanie utworzony za pomocą formantu *Etykieta*. Na karcie *Projektowanie* w grupie *Formanty* kliknij ikonę *Etykieta* i w górnej części formularza narysuj prostokąt. Wpisz do niego tytuł formularza. Naciśnij przycisk *Enter*, aby

zatwierdzić tekst. Wprowadzony tekst może być formatowany za pomocą narzędzi dostępnych w grupie *Czcionka*. Pola z tabeli *Klient* umieść w formularzu, przeciągając je z okna *Lista pól*.

Pole obliczeniowe *Nazwisko i imię klienta* utwórz, korzystając z *konstruktora wyrażeń*. Zdefiniowane wyrażenie ma połączyć zawartość dwóch pól: *Nazwisko* oraz *Imię*, i ma postać:

```
=[Klient.Nazwisko] & " " & [Klient.Imię]
```

Aby utworzyć pole obliczeniowe, na karcie *Projektowanie* w grupie *Formanty* kliknij ikonę *Pole tekstowe* i na formularzu narysuj prostokąt. Kliknij dwukrotnie utworzone pole tekstowe, aby otworzyć okno *Arkusz właściwości*. W tym oknie na karcie *Dane* wybierz właściwość *Źródło formantu* i wpisz podane wyżej wyrażenie.

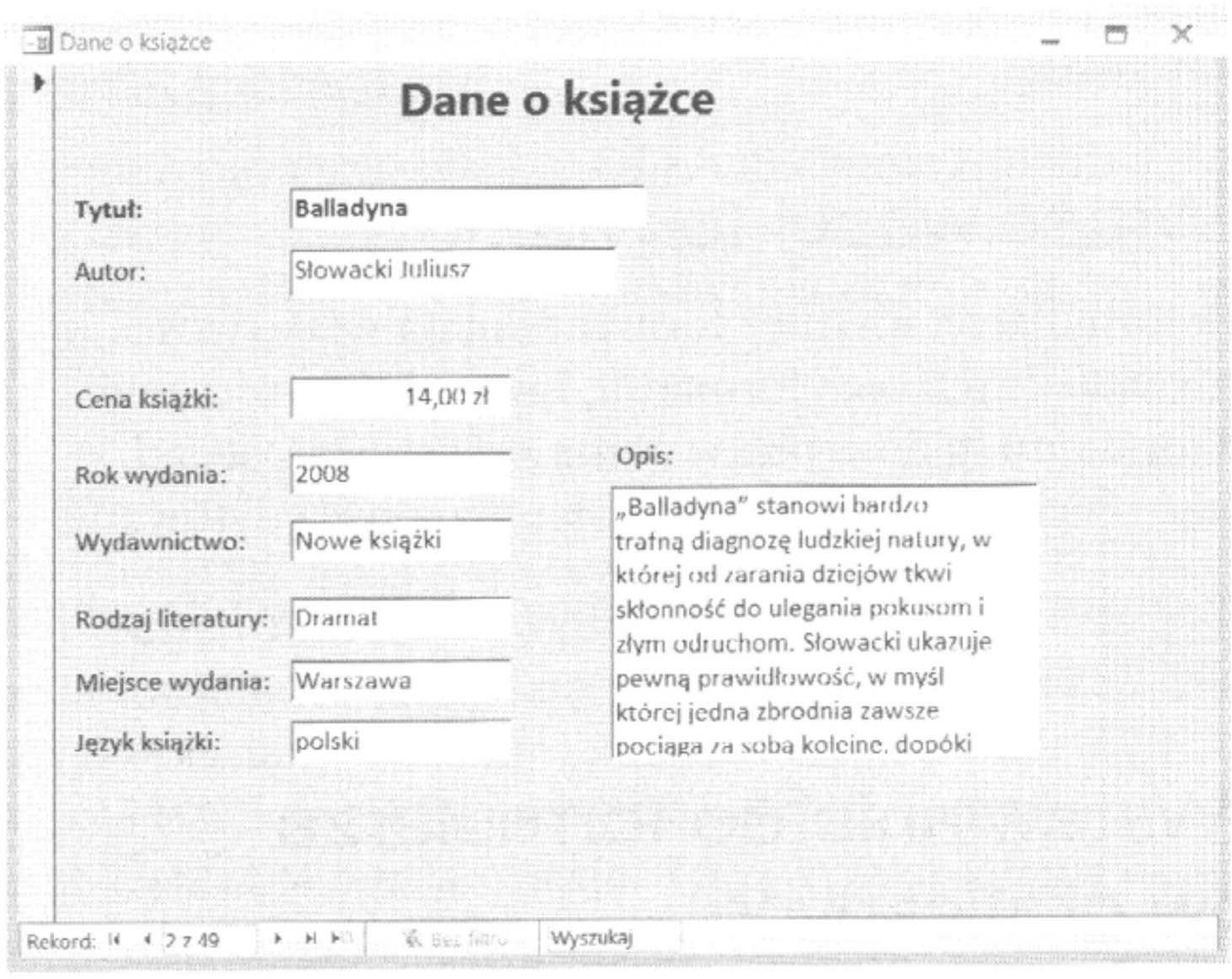
Przejdź do widoku formularza, aby sprawdzić, czy został on zaprojektowany prawidłowo. Zapisz i zamknij formularz. Nazwij go *Dane klienta*.

Nawigacja między rekordami formularza odbywa się za pomocą pola *Rekord* w dolnej części okna formularza. Zamknięcie, zminimalizowanie lub zmiana wielkości okna formularza są możliwe przy użyciu ikon w prawym górnym rogu okna formularza.

Przykład 2.18

Zaprojektuj kwerendę zawierającą wszystkie pola z tabel *Książki* i *Autorzy*. Nazwij kwerendę *O książce*. Zaprojektuj dla niej formularz zawierający wszystkie pola z tabeli *Książki* oraz pola *Nazwisko* i *Imię* z tabeli *Autorzy* (rysunek 2.38). W projekcie formularza jako *Źródło rekordów* wybierz z listy kwerendę *O książce*. Formularz nazwij *Dane o książce*.

Rysunek 2.38. Formularz zaprojektowany dla kwerendy

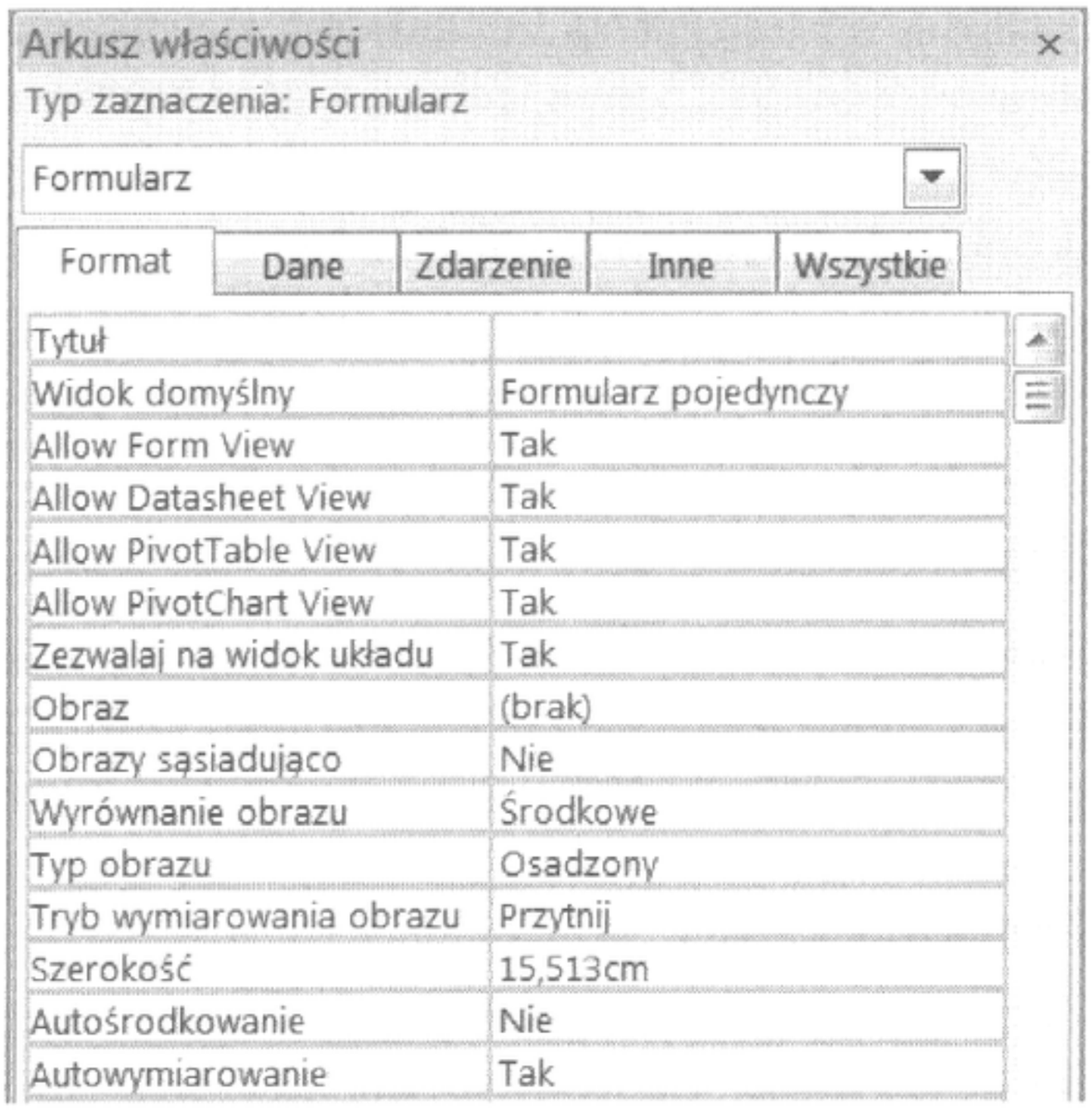


Formularz został utworzony podobnie jak formularz z poprzedniego ćwiczenia, ale zawiera informacje o książce oraz nazwisko i imię autora książki dodane z tabeli *Autorzy*.

2.5.3. Właściwości formularza

Formularze, tak jak inne obiekty, mają właściwości, które umożliwiają definiowanie indywidualnych cech. Tych właściwości jest kilkadziesiąt, ale tylko niektóre mają bezpośredni wpływ na pracę z bazą danych. Aby ułatwić pracę przy ich definiowaniu, właściwości zostały podzielone w oknie *Arkusz właściwości* na kilka grup (rysunek 2.39).

Grupa *Format* zawiera właściwości związane z wyglądem formularza. W grupie *Dane* definiujemy właściwości związane z danymi, które można wyświetlić w formularzu, i ze sposobem dostępu do tych danych. Tu znajduje się jedna z najczęściej wykorzystywanych właściwości, czyli *Źródło rekordów*. Za jej pomocą można powiązać formularz z dowolną tabelą lub kwerendą. Nieprawidłowe określenie tej właściwości prowadzi do poważnych błędów w funkcjonowaniu bazy danych. Grupa *Zdarzenie* zawiera właściwości, przy użyciu których możemy zdefiniować działania związane ze zdarzeniami zachodzącymi w trakcie pracy z formularzem, a grupa *Inne* zawiera pozostałe właściwości. Można również wybrać grupę *Wszystkie* i przeglądać wszystkie dostępne właściwości.



Rysunek 2.39.
Arkusz właściwości formularza

2.5.4. Właściwości formantów

Podobnie jak formularze, również formanty mają właściwości, które umożliwiają definiowanie indywidualnych cech formantu. Lista właściwości zależy od rodzaju formantu. Właściwości formantu są dostępne w oknie *Arkusz właściwości*. W polu *Typ zaznaczenia* można wybrać obiekt, którego właściwości mają być edytowane.

Każdy typ formantu ma domyślny zestaw właściwości decydujący o jego ogólnym wyglądzie i zachowaniu.

2.5.5. Wstawianie do formularza obiektów graficznych

Użycie formularza oznacza wykorzystanie graficznych możliwości aplikacji. Do formularza można wstawiać elementy graficzne, korzystając z trzech rodzajów formantów:

Obraz, *Niezwiązana ramka obiektu* i *Związana ramka obiektu* (rysunek 2.40). Każdy z nich ma inne zastosowanie oraz inny sposób działania.

Rysunek 2.40. Opcje wstawiania obiektów

Obraz

Służy do wyświetlania statycznych obrazów. Nie można go edytować. Rysunek jest wyświetlany w utworzonym formacie obrazu. Można zmieniać jego rozmiar i proporcje. We właściwościach tego formantu można określić tryb wymiarowania.

Oto właściwości obrazu:

- *Obetnij* — rysunek jest wyświetlany w aktualnej wielkości; jeśli jest za duży, zostanie przycięty.
- *Rozciągnij* — rysunek jest skalowany tak, aby zmieścił się wewnątrz formantu. Może to prowadzić do zniekształcenia obrazu.
- *Powiększ* — rysunek jest wyświetlany w całości po przeskalowaniu, tak aby jego wysokość lub szerokość została dopasowana do odpowiednich wymiarów formantu. Wartość ta nie zniekształca obrazu.

Użycie formantu *Obraz* do umieszczania i wyświetlania rysunków jest sposobem bardziej wydajnym niż związane i niezwiązane ramki obiektów.

Niezwiązana ramka obiektu

Służy do wyświetlania niezwiązanego obiektu OLE. Obiekt nie zmienia się przy przechodzeniu od rekordu do rekordu. Jeżeli nie jest to konieczne do umieszczania rysunków, lepiej wykorzystać formant *Obraz*. Tryb wymiarowania określa się w ten sam sposób jak dla formantu *Obraz*.

Związana ramka obiektu

Formant ten jest przeznaczony do wyświetlania obiektów OLE, których wartości są przechowywane w kolejnych rekordach wybranego pola tabeli. Przy przechodzeniu od rekordu do rekordu formularza zmieniają się wyświetlane obiekty. Taki rodzaj formantów może być wykorzystany do wyświetlania danych z pola *Zdjęcie* (typ OLE). Przy przeciągnięciu pola obiektu OLE z okna *Lista pól* do formularza automatycznie jest tworzony formant *Związana ramka obiektu*. Tryb wymiarowania określa się w ten sam sposób jak dla formantu *Obraz*.

Elementy graficzne do bazy danych należy dodać na etapie końcowym, gdy zostały zaprojektowane i przetestowane wszystkie obiekty bazy danych. Im więcej elementów graficznych, tym więcej czasu i pamięci potrzeba do ich przedstawienia.

Zadanie 2.2

Dla bazy danych *Księgarnia internetowa* zaprojektuj formularz dotyczący realizacji zamówień, zawierający: dane klienta (*Nazwisko*, *Imię* oraz *Adres*), informacje o zamówionej książce (*Tytuł książki*, *Nazwisko i imię autora*, *Liczba egzemplarzy*) oraz informacje dotyczące realizacji zamówienia (*Data złożenia zamówienia*, *Data wysłania*, *Wartość zamówienia*: cena książek i koszt wysyłki). W formularzu umieść element graficzny związany z księgarnią internetową (na przykład logo firmy).

W formularzu wyświetlającym dane o uczniach biorących udział w zawodach umieść pole wyświetlające zdjęcie ucznia, tak aby po zmianie rekordu w formularzu zdjęcie ucznia również uległo zmianie.

2.5.6. Przyciski poleceń w formularzu

Formant *Przycisk polecenia* w formularzu służy do uruchamiania akcji lub zestawu akcji. Tą akcją może być otwieranie innego formularza, znajdowanie rekordu, usuwanie rekordu czy przejście do kolejnego rekordu. Aby przypisać akcję do przycisku polecenia, należy zaprojektować makro lub napisać procedurę zdarzenia i dołączyć ją do odpowiedniej właściwości przycisku polecenia. Istnieje także możliwość automatycznego utworzenia procedury zdarzenia przy użyciu kreatora formantów.

Kreator przycisków poleceń

Za pomocą kreatora przycisków poleceń można tworzyć przyciski służące do wykonywania różnych zadań, takich jak: zamykanie formularza, znajdowanie rekordu lub drukowanie raportu.

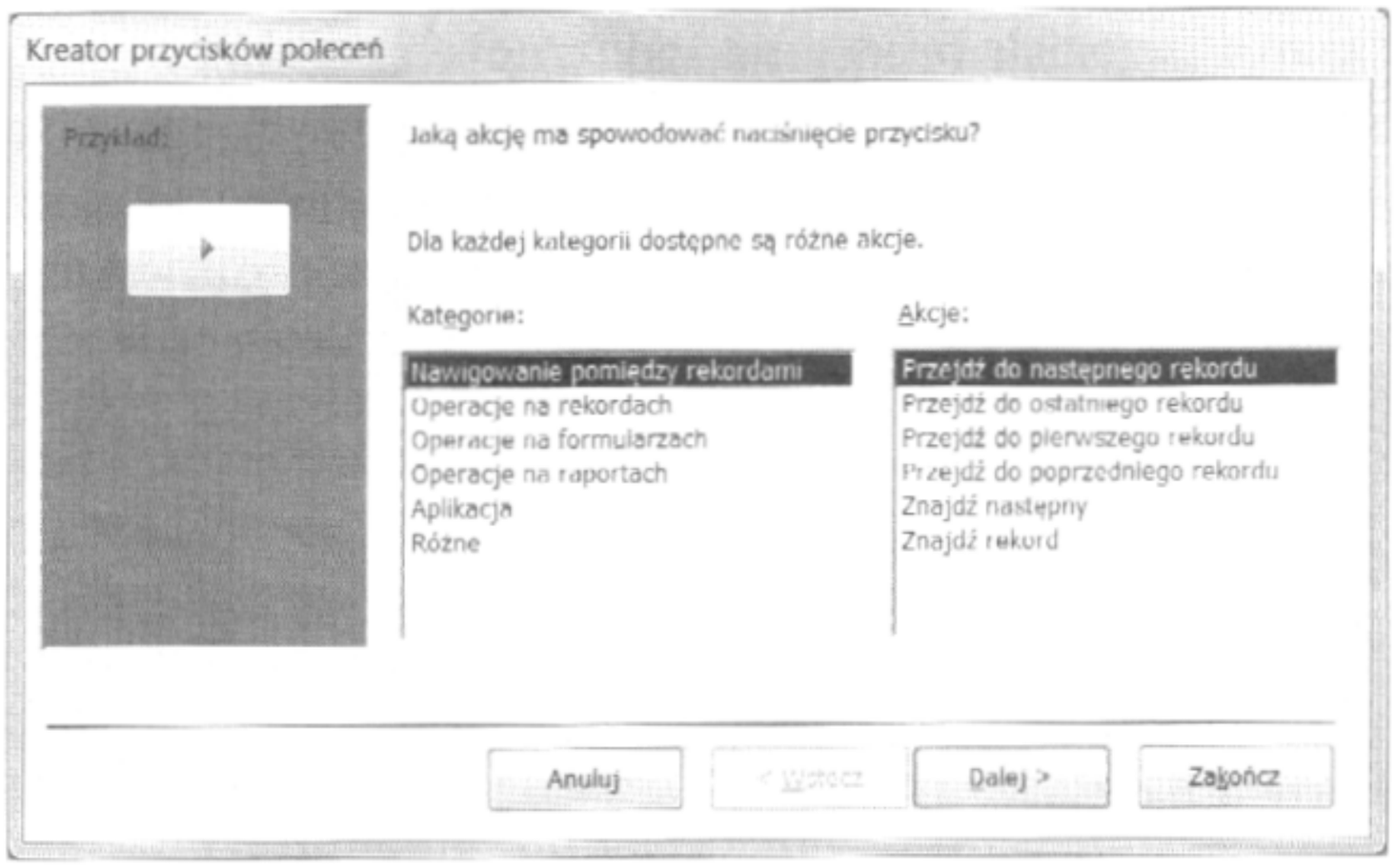
Aby zaprojektować przycisk poleceń przy użyciu kreatora przycisków poleceń, należy w obszarze *Widok projektu*, na karcie *Projektowanie*, w grupie *Formanty* włączyć opcję *Użyj kreatorów formantów*, następnie w tej samej grupie wybrać ikonę *Przycisk* i w projekcie formularza kliknąć w miejscu, gdzie ma być wstawiony przycisk. Zostanie uruchomiony *Kreator przycisków poleceń*, który krok po kroku utworzy przycisk polecenia i w jego właściwości zdarzenia *Przy kliknięciu* osadzi makro. Makro zawiera akcje służące do wykonywania zadania wybranego w kreatorze. Na karcie *Zdarzenie* w arkuszu właściwości obok właściwości *Przy kliknięciu* powinien być wyświetlany tekst *[Makro osadzone]*.

Przykład 2.19

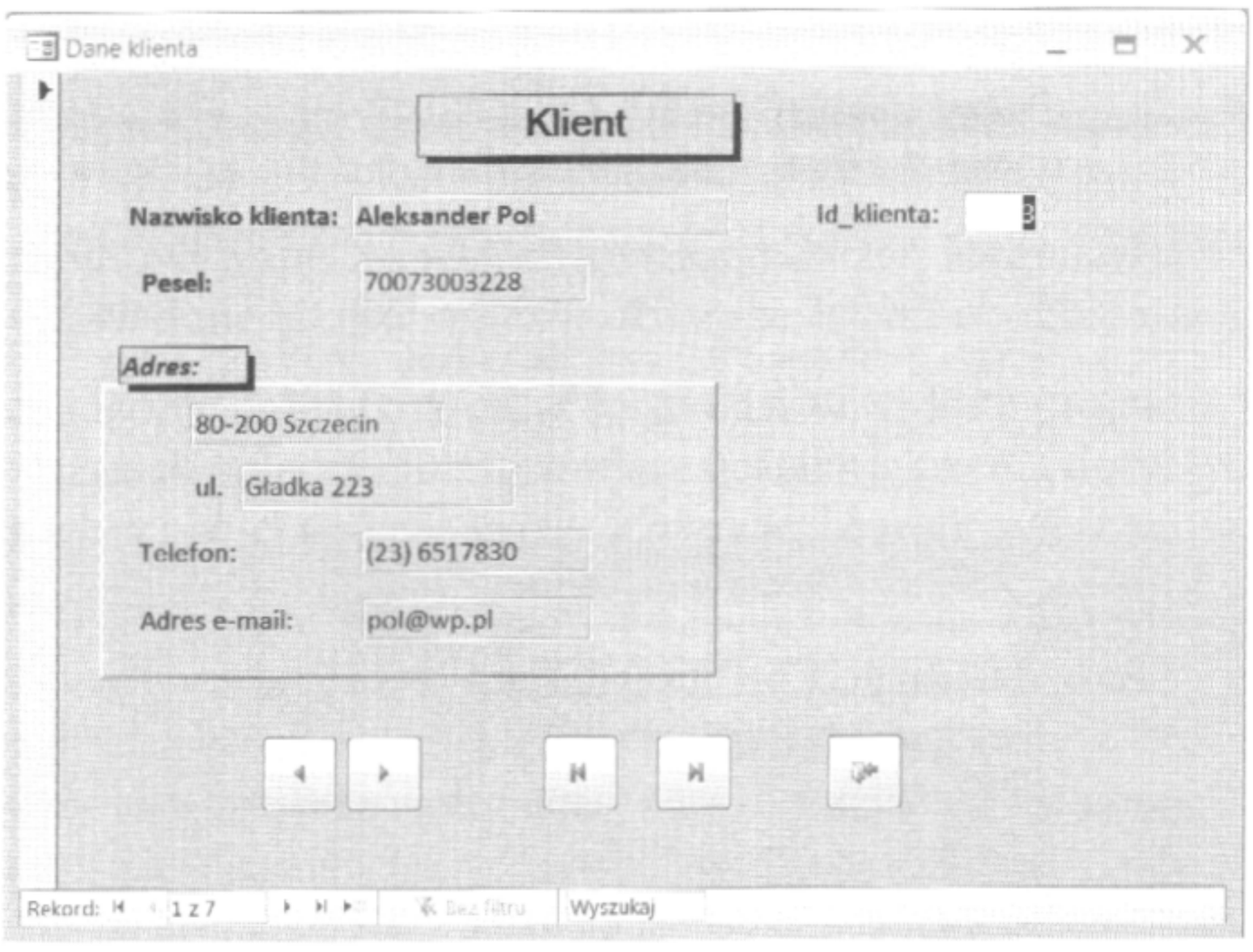
W formularzu *Dane klienta* z przykładu 2.17 utwórz za pomocą **kreatora przycisków poleceń** następujące przyciski służące do nawigacji między rekordami: *Następny klient*, *Poprzedni klient*, *Pierwszy*, *Ostatni* oraz przycisk *Zamknij* do zamykania formularza.

Projektowanie przycisku zacznij od uruchomienia kreatora. Dla przycisku *Następny klient* wybierz kategorię *Nawigowanie między rekordami* oraz akcję *Przejdź do następnego rekordu*, po czym kliknij przycisk *Dalej* (rysunek 2.41).

W następnym kroku określ, czy na przycisku powinien pojawić się *Tekst* opisujący działanie przycisku, czy *Obraz* ilustrujący to działanie. Po wybraniu odpowiedniej opcji kliknij *Zakończ*. Pozostałe przyciski do nawigacji zaprojektuj w podobny sposób. Dla przycisku *Zamknij* wybierz kategorię *Operacje na formularzach* oraz akcję *Zamknij formularz*. Dalej postępuj podobnie jak przy projektowaniu poprzednich przycisków. Gotowy formularz pokazano na rysunku 2.42.



Rysunek 2.41. Okno narzędzia Kreator przycisków poleceń



Rysunek 2.42. Formularz z przyciskami poleceń

Właściwości zdarzeń

W programie Access do obiektów takich jak formularze, raporty oraz formanty zostały przypisane różne właściwości zdarzeń. Właściwości zdarzeń zawierają listę zdarzeń, które mogą towarzyszyć danemu obiektowi (na przykład kliknięcie myszą, otwarcie formularza). Zdarzenia mogą też być związane z działaniami występującymi poza programem Access (na przykład zdarzenia systemowe). Użytkownik może przypisać do zdarzeń zachodzących podczas pracy z bazą danych określone działanie poprzez

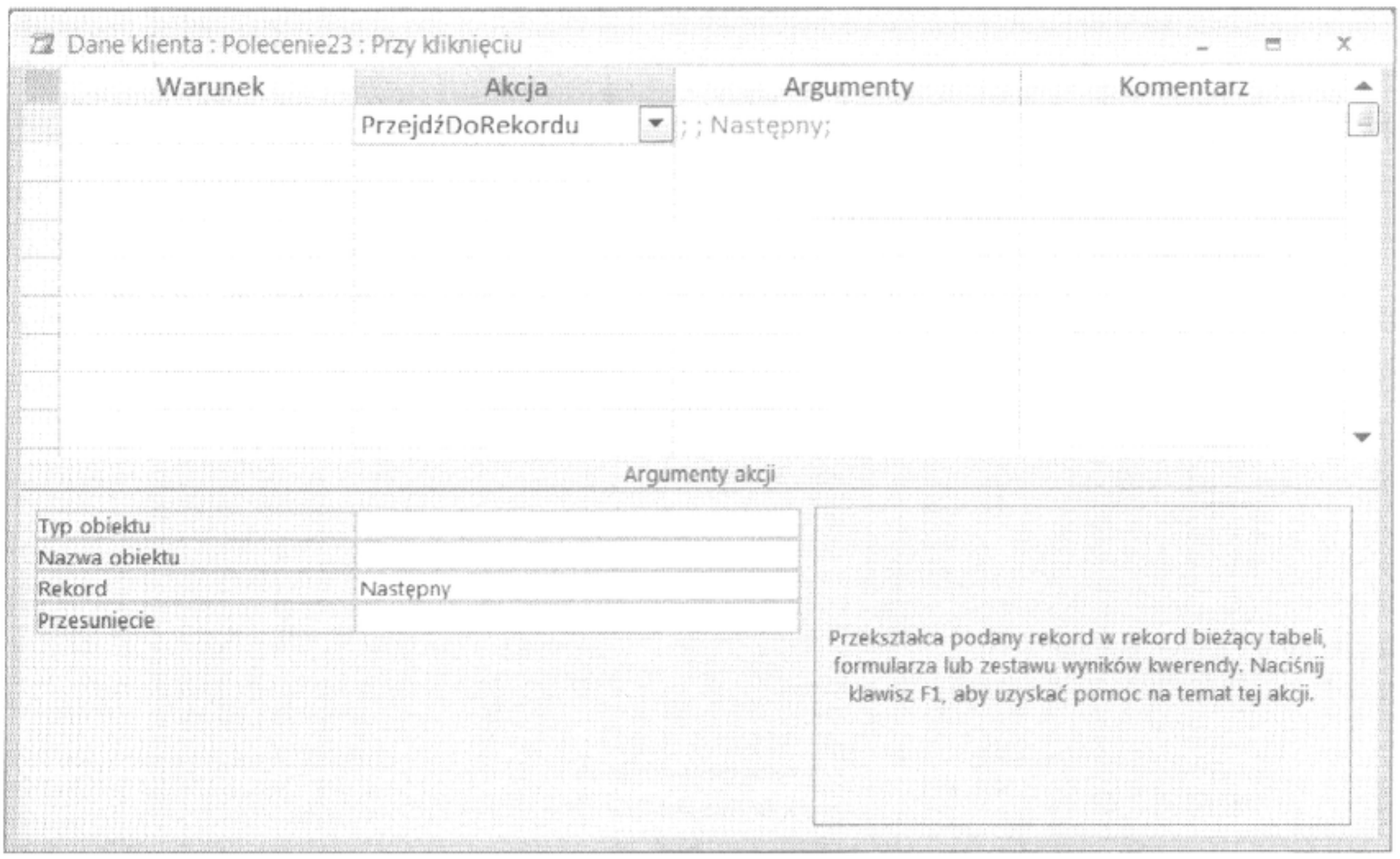
zdefiniowanie makropoleceń lub napisanie procedur. Zaistnienie zdarzenia spowoduje wykonanie przypisanej do niego procedury lub makropolecenia.

2.5.7. Makropolecenia

Makropolecenia umożliwiają automatyzację zadań wykonywanych w bazie. Zadania te można wiązać ze zdarzeniami zachodzącymi w obiektach bazy danych. Jeśli w formularzu dodano przycisk polecenia, zdarzenie *Przy kliknięciu* można związać z zaprojektowanym makropoleceniem, które będzie wykonywane za każdym razem, gdy przycisk zostanie kliknięty. Projektowanie makra odbywa się przez wybór akcji z listy rozwijanej, a następnie uzupełnienie akcji dodatkowymi informacjami, na przykład określeniem argumentów akcji. Zaprojektowane makro może składać się z wielu akcji wykonywanych kolejno.

Definiowanie makropoleceń

Makropolecenia najlepiej projektować za pomocą konstruktora makr. Uruchamia się go, wybierając na karcie *Tworzenie* w grupie *Inne* przycisk *Makro*. Zostanie otwarte okno projektowania makr, a w obszarze *Narzędzia makr* zostanie udostępniona karta *Projektowanie* (rysunek 2.43).



Rysunek 2.43. Okno projektowania makropolecenia

W kolejnych wierszach kolumny *Akcja* wybieramy z listy akcje, z których będzie się składało makro. Większość akcji wymaga co najmniej jednego argumentu. Niektóre argumenty są wymagane, inne są opcjonalne. Argumenty można zobaczyć w obszarze *Argumenty akcji* u dołu konstruktora. Dodatkowo do każdego makra można dodać warunki określające sposób uruchamiania każdej akcji. Warunki określają kryteria, które

muszą zostać spełnione, zanim akcja będzie wykonana. Warunki wykonania makra są definiowane w kolumnie *Warunek*. Domyślnie kolumna *Warunek* jest wyłączona. Można ją włączyć, wybierając na wstążce ikonę *Warunki*.

Akcje makropoleceń

Akcje są podstawowymi elementami konstrukcyjnymi makr. Program Access udostępnia wiele akcji, spośród których można wybierać najprzydatniejsze przy wykonywaniu poleceń, na przykład umożliwiające otworzenie raportu, znalezienie rekordu, wyświetlenie okna komunikatu lub zastosowanie filtru do formularza lub raportu.

Makropolecenia warunkowe

Warunek określa kryteria, jakie muszą zostać spełnione, zanim akcja będzie wykonana. Można użyć dowolnego wyrażenia zwracającego wynik *Prawda/Fałsz* lub *Tak/Nie*. Akcja nie zostanie wykonana, jeśli wyrażenie zwróci wartość *Fałsz*, *Nie* lub *0* (zero). Jeśli wyrażenie zwróci inną wartość, akcja zostanie wykonana.

Wiązanie makropoleceń ze zdarzeniami

W programie Access makro lub grupa makr mogą zostać umieszczone w obiekcie makro (czasem nazywanym makrem autonomicznym). Makro może być również osadzone w dowolnej właściwości zdarzenia formularza, raportu lub formantu. Makra osadzone stają się częścią obiektu lub formantu. Makra autonomiczne są widoczne w oknie nawigacji w obszarze *Makra*, natomiast makra osadzone są niewidoczne.

Makra osadzone są przechowywane we właściwościach zdarzeń formularzy, raportów i formantów. Nie są wyświetlane jako obiekty w obszarze *Makra* w oknie nawigacji.

Aby utworzyć makro osadzone, należy otworzyć formularz w widoku projektu, wybrać formant (na przykład przycisk poleceń) lub formularz (jeżeli makro ma dotyczyć formularza), w oknie *Arkusz właściwości* wybrać kartę *Zdarzenie*, określić właściwość zdarzenia, w którym ma zostać osadzone makro, a następnie obok pola kliknąć przycisk ....

W otwartym oknie dialogowym trzeba kliknąć narzędzie *Konstruktor makr* i przycisk *OK*. W konstruktorze poleceń w kolumnie *Akcja* należy wybrać odpowiednią akcję, wpisać wymagane argumenty w oknie *Argumenty akcji* i przejść do kolejnego wiersza akcji. Po zakończeniu projektowania makro należy zapisać i zamknąć okno.

Makro będzie uruchamiane przy każdym zaistnieniu określonego zdarzenia.

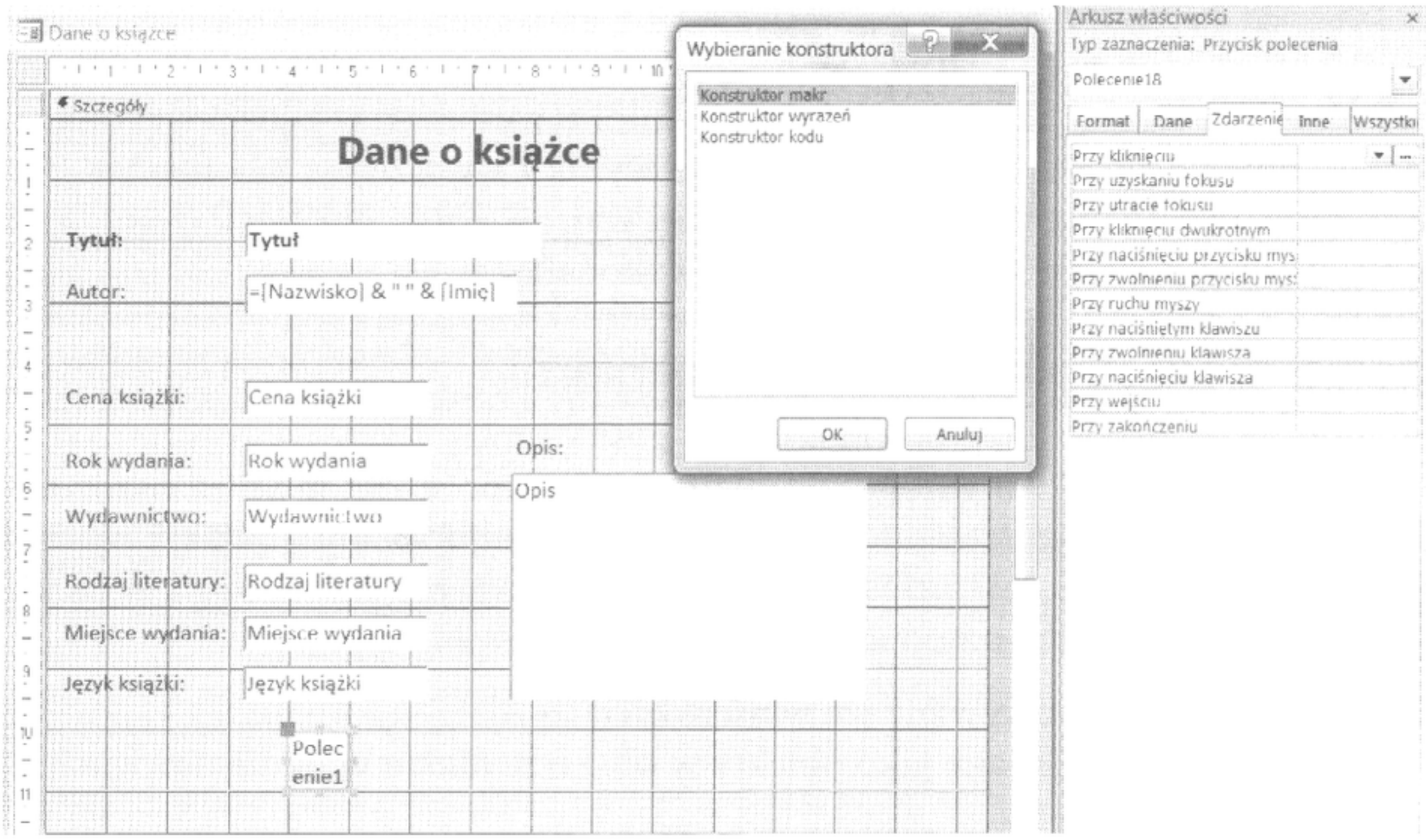
Przykład 2.20

W formularzu *Dane o książce* z przykładu 2.18 utwórz za pomocą makr osadzonych przyciski służące do nawigacji między rekordami (*Następna książka*, *Poprzednia książka*, *Pierwsza*, *Ostatnia*) oraz przycisk *Zamknij* do zamykania formularza. Nie używaj kreatora przycisków poleceń.

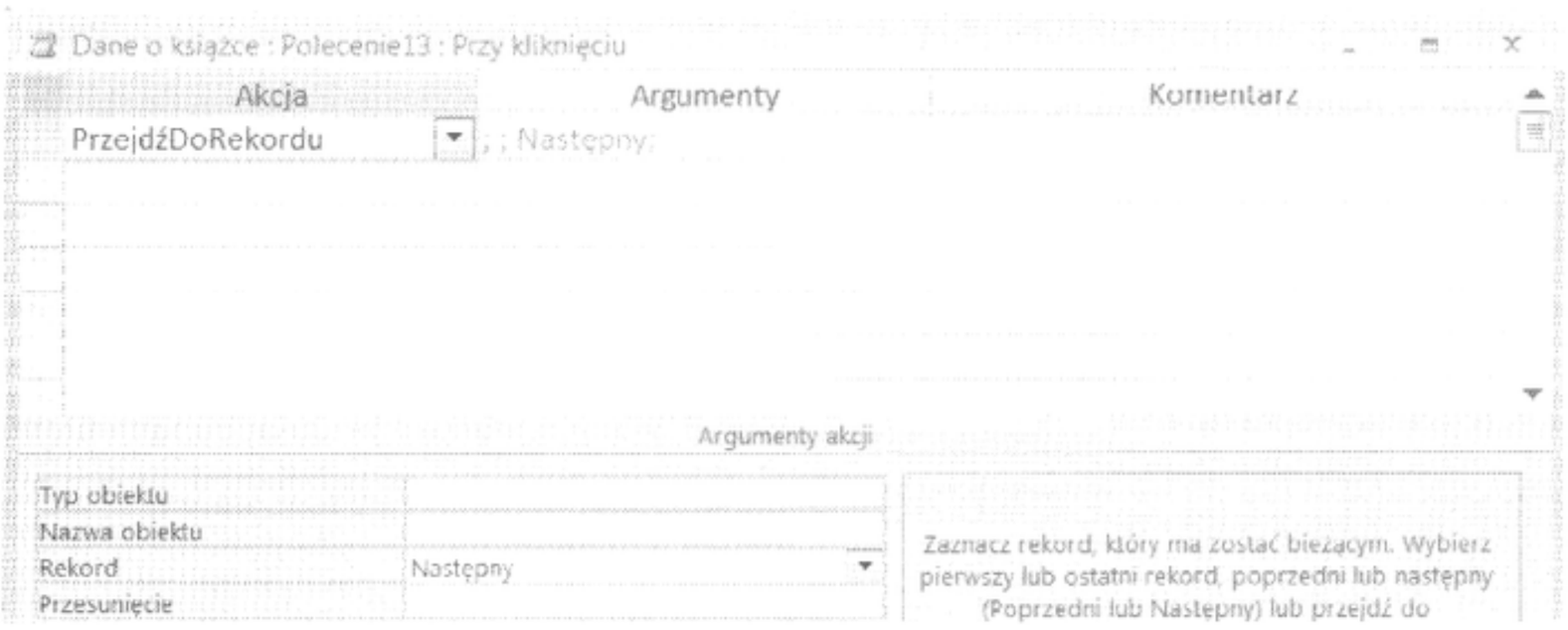
Otwórz formularz w widoku projektu, wybierz projektowanie formantu *Przycisk* (*Kreator formantów* jest wyłączony). We właściwościach zdarzeń formantu *Przycisk* wybierz właściwość *Przy kliknięciu* i uruchom konstruktor makr (rysunek 2.44).

W oknie projektowania *Makro* wybierz akcję *PrzejdźDoRekordu* i w argumentach akcji — *Następny* (rysunek 2.45). Zapisz zaprojektowane makro.

Rozmiar i położenie tak zaprojektowanego przycisku możesz zmienić bezpośrednio w projekcie formularza lub ustawić na karcie *Format* właściwości: *Szerokość*, *Wysokość*, *Górny*, *Lewy*.



Rysunek 2.44. Uruchamianie konstruktora makr z widoku projektu formularza



Rysunek 2.45. Wybór akcji i argumentów akcji

Jeżeli na przycisku ma zostać wyświetlona ikona, we właściwości *Obraz* wpisz ścieżkę dostępu i nazwę pliku obrazu. Możesz również uruchomić *Konstruktor obrazów* i wybrać odpowiedni obraz spośród profesjonalnych, które oferuje Access.

Pozostałe przyciski do nawigacji między rekordami zaprojektuj w podobny sposób, zmieniając odpowiednio argumenty akcji *Rekord*. Projektując przycisk *Zamknij* do zamykania formularza, wybierz akcję *Zamknij*. Dla argumentu *Typ obiektu* wybierz z listy *Formularz*, a dla argumentu *Nazwa obiektu* — nazwę zamykanego formularza (*Dane o książce*). Argumenty akcji *Typ obiektu* i *Nazwa obiektu* mogą pozostać niewypełnione — wtedy uruchomienie makra spowoduje zamknięcie bieżącego aktywnego obiektu.

Zadanie 2.3

Dla pozostałych formularzy bazy danych *Księgarnia internetowa* utwórz za pomocą makr osadzonych przyciski służące do nawigacji między rekordami, takie jak: *Następny rekord*, *Poprzedni rekord*, *Pierwszy rekord*, *Ostatni rekord*, oraz przycisk *Zamknij* do zamykania formularza (nie używaj kreatora przycisków poleceń).

2.5.8. Funkcje formularza

Formularze służą do przedstawiania danych z tabeli lub kwerendy w sposób graficzny. Przez formularze użytkownik bazy danych ma dostęp do zgromadzonych danych. Formularz może służyć do przeglądania, wprowadzania lub edytowania danych, można poprzez formularz sterować dostępem do danych. Do formularza można dodać elementy, które pozwolą na sterowanie obsługą bazy danych.

Funkcje dostępu do danych i sposobu ich obsługi są realizowane poprzez definiowanie właściwości formularza. Sterowanie obsługą bazy danych odbywa się przez umieszczanie w formularzu elementów sterujących i definiowanie obsługi zdarzeń.

Zaprojektowane w ćwiczeniach formularze dla bazy danych *Księgarnia internetowa* nie mają zdefiniowanych ograniczeń dostępu do danych. Nadszedł czas, aby poznać narzędzia, które pozwolą na sterowanie dostępem do danych.

Formularz do przeglądania danych

Przykład 2.21

Wcześniej zaprojektowany formularz *Dane klienta* zmodyfikuj tak, aby pozwalał na przeglądanie danych bez możliwości ich dodawania, modyfikowania i usuwania. Dodatkowo zablokuj w formularzu wyświetlanie paska *Rekord* w dolnej części okna i działanie ikony *Zamknij* w prawym górnym rogu okna. Do nawigacji między rekordami i zamykania formularza użyj wyłącznie przycisków poleceń. Ikony w prawym górnym rogu okna formularza powinny być niedostępne.

Aby zablokować możliwość dodawania, modyfikowania i usuwania danych, w projekcie formularza otwórz *Arkusz właściwości* i w zakładce *Dane* ustaw wartość *Nie* właściwości *Dodawanie dozwolone*, *Usuwanie dozwolone* i *Edycja dozwolona* (rysunek 2.46).

Aby zablokować nawigację między rekordami i zamykanie okna, w oknie *Arkusz właściwości* w zakładce *Format* ustaw *Nie* dla właściwości *Przyciski nawigacyjne* i *Przycisk Zamknij*.

Rysunek 2.46.
Arkusz właściwości formularza, zakładka Dane

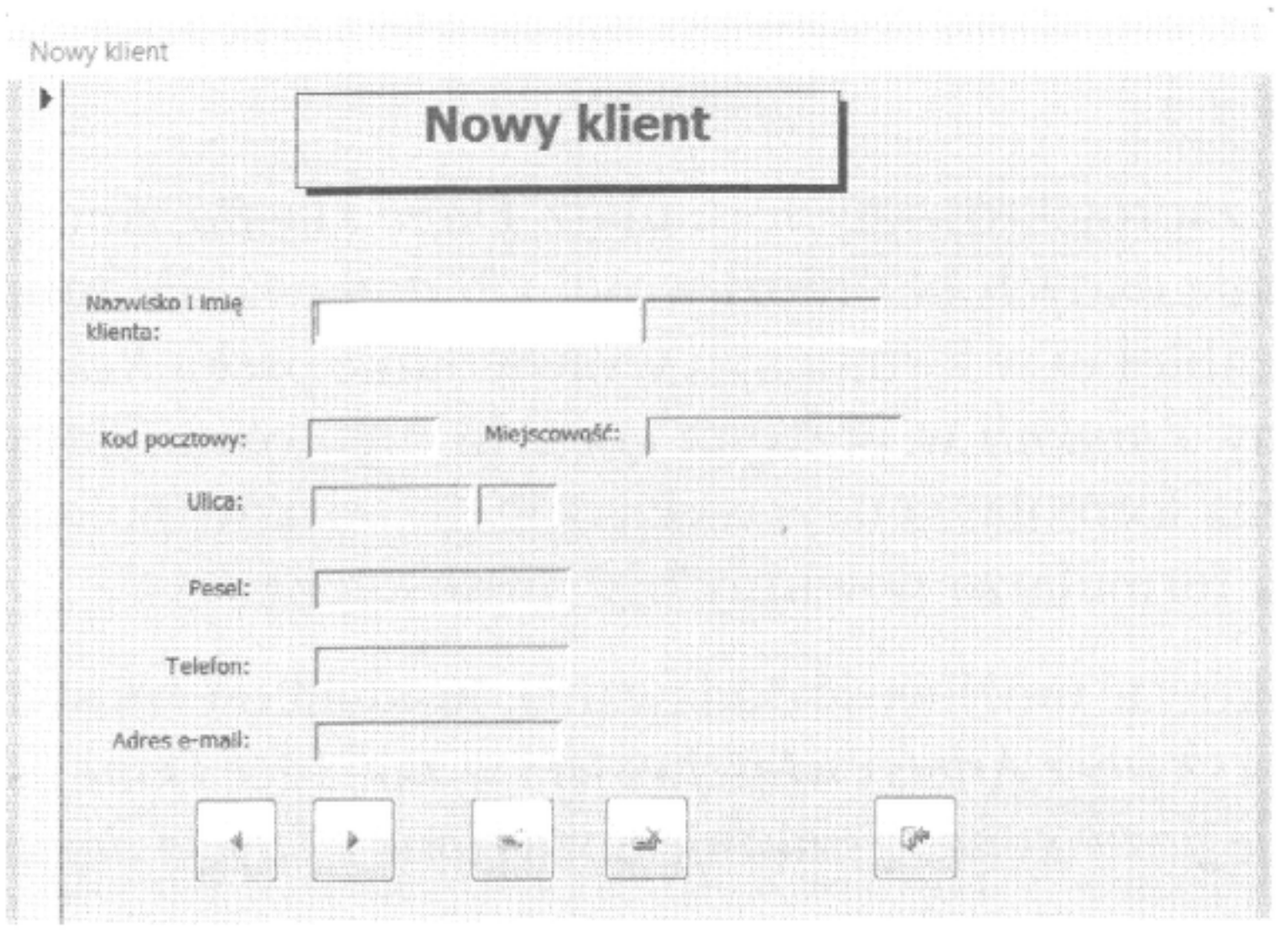
Dodatkowo możesz zablokować ikonę menu sterowania (ikona w lewym górnym rogu okna formularza) przez ustawienie wartości *Nie* właściwości *Pole menu sterowania* oraz zablokować zmianę wielkości okna przez ustawienie wartości *Żaden* właściwości *Przyciski Min Maks*. We właściwości *Tytuł* umieść tekst `Informacje o kliencie`. Teraz ten tekst będzie pojawiał się w tytule formularza.

Formularz do wprowadzania danych

Przykład 2.22

Zaprojektuj nowy formularz oparty na tabeli *Klient*. Będzie on przeznaczony do wprowadzania danych nowych klientów. Umieść w nim wszystkie pola z tej tabeli i nazwij go *Nowy klient*. Formularz wprowadzania danych powinien otwierać się z pustymi polami gotowymi do wprowadzania danych, tak jak na rysunku 2.47.

Rysunek 2.47.
Formularz do wprowadzania danych



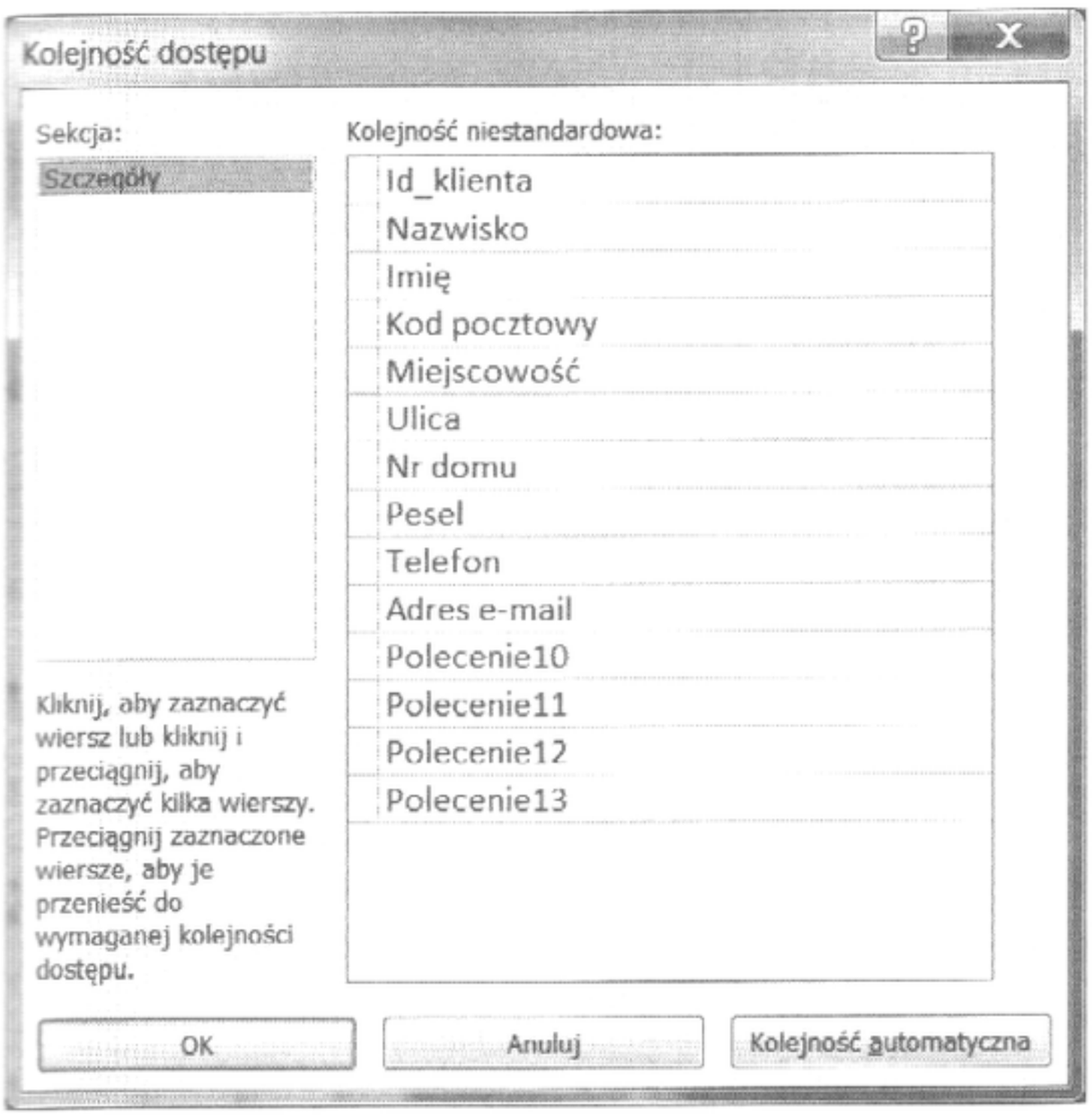
Dla tabeli *Klient* zaprojektuj formularz i umieść w nim wszystkie pola tabeli. W projekcie formularza otwórz *Arkusz właściwości* i w zakładce *Dane* ustaw wartość *Tak* właściwości *Wprowadzanie danych*. Określ również poznane właściwości wpływające na wygląd

formularza. Zaprojektuj przyciski poleceń *Następny klient*, *Poprzedni klient* i *Zamknij formularz*. Wprowadź dane kilku klientów, aby sprawdzić działanie formularza.

Zauważ, że można przeglądać, modyfikować i usuwać dane ostatnio wprowadzone, ale nie można oglądać danych wcześniej wprowadzonych do tabeli *Klient*. Dane wprowadzane poprzez formularz są zapisywane w tabeli *Klient*. Zamknięcie formularza kończy sesję wprowadzania danych i nie są one dostępne do modyfikacji z tego formularza.

Aby usprawnić pracę z formularzem, utworzymy dwa przyciski: *Nowy klient* do wprowadzania danych następnego klienta oraz *Usuń* do usuwania błędnie wpisanych rekordów.

Przycisk *Nowy klient* zaprojektuj, wybierając w tworzonym makropoleceniu akcję *PrzejdźDoRekordu*; w argumencie akcji *Rekord* wybierz *Nowy*. Przycisk *Usuń* zaprojektuj, wybierając w tworzonym makropoleceniu akcję *UruchomPolecenie*, a w argumencie akcji *Polecenie* wybierz *UsuńRekord*.



Rysunek 2.48. Okno Kolejność dostępu

Wprowadzając dane do formularza, można zauważyć, że nie zawsze kolejność dostępnych do wypełniania okien jest odpowiednia. Domyślnie okna do wypełniania pól są dostępne w takiej kolejności, w jakiej były projektowane, ale można to zmienić. W tym celu wybierz w widoku projektu, na karcie *Rozmieszczanie*, w grupie *Sterowanie układem*, ikonę *Kolejność klawisza Tab*. W otwartym oknie dialogowym (rysunek 2.48) zaznacz wiersz z wybranym polem i przeciągnij go w odpowiednie miejsce na liście. Kolejność obsługiwanych pól zostanie zmieniona zgodnie z zaprojektowaną listą.

Zadanie 2.4

Wcześniej zaprojektowane formularze w bazie danych *Księgarnia internetowa*: *Dane o książce* i *Realizacja zamówień* powinny służyć do przeglądania danych, bez możliwości ich dodawania, modyfikowania i usuwania. Ustaw odpowiednie właściwości formularzy.

Zaprojektuj nowy formularz oparty na tabeli *Książki* przeznaczony do wprowadzania informacji o nowych książkach i nazwij go *Nowe książki*. Umieść w nim wszystkie pola z tej tabeli. Wykorzystaj wiadomości z przykładu 2.22.

Zaprojektuj nowy formularz oparty na tabeli *Autorzy* i nazwij go *Nowy autor*. Umieść w nim wszystkie pola z tej tabeli. Będzie on przeznaczony do wprowadzania informacji o nowych autorach książek. Wykorzystaj wiadomości z poprzednich przykładów.

2.5.9. Formularze pojedyncze i ciągłe

W formularzu można wyświetlać rekordy pojedynczo lub w sposób ciągły. W tym drugim przypadku wyświetlane są jeden po drugim identyczne formularze pokazujące dane z kolejnych rekordów tabeli lub kwerendy. Sposób wyświetlania danych w formularzu można określić, ustawiając właściwość *Widok domyślny* formularza w oknie *Arkusz właściwości*, po wybraniu zakładki *Format*. Wartość *Formularz pojedynczy* jest ustawiona domyślnie i wyświetla dane tylko z jednego rekordu. Aby zobaczyć wiele rekordów w jednym oknie, należy wybrać wartość *Formularze ciągłe*. W tym widoku w formularzu powielana jest sekcja *Szczegóły* dla każdego rekordu tabeli lub kwerendy. Wybór wartości *Arkusz danych* powoduje wyświetlanie danych w postaci tabeli. W arkuszu danych wyświetlane są tylko pola wybrane do formularza.

Przykład 2.23

Zaprojektuj nowy formularz oparty na tabeli *Książki* i nazwij go *Lista książek*. W formularzu umieść pola *Tytuł*, *Cena* i *Rodzaj literatury*. Formularz będzie przeznaczony do przeglądania danych o książkach dostępnych w bazie. Dane powinny zostać wyświetlone w postaci formularzy ciągłych.

Ze względu na to, że w formularzach ciągłych powielana jest sekcja *Szczegóły*, w niej należy umieścić pola tekstowe związane z polami tabeli. Etykiety z nazwami pól powinny zostać umieszczone w nagłówku formularza. Przycisk polecenia *Zamknij formularz* również nie powinien być powtarzany, dlatego trzeba umieścić go w nagłówku lub stopce formularza.

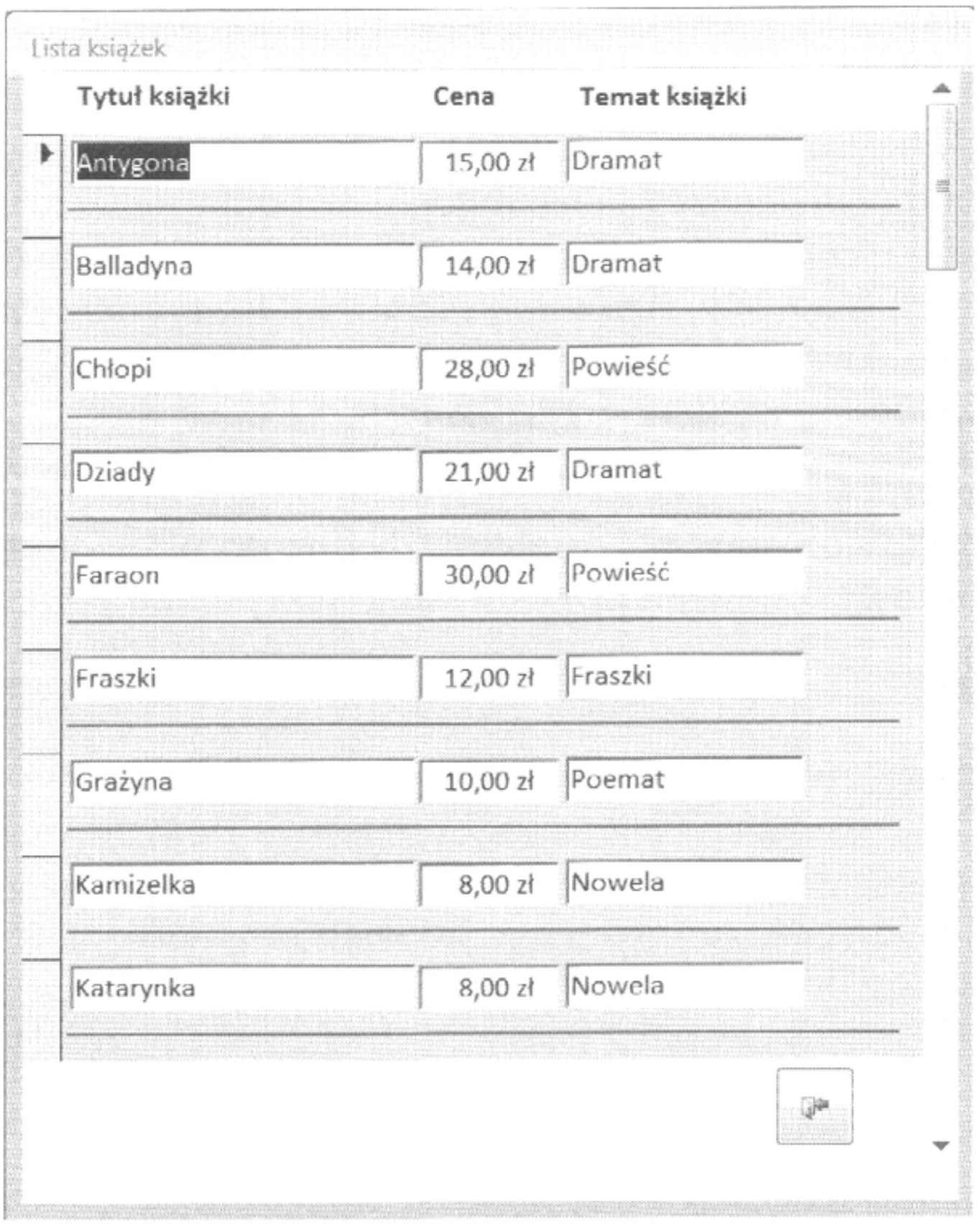
Na karcie *Tworzenie* w grupie *Formularze* wybierz ikonę *Projekt formularza*. We właściwości *Źródło rekordów* na karcie *Dane* wybierz tabelę *Książki*. Z okna *Lista pól* przenieś do sekcji *Szczegóły* pola: *Tytuł*, *Cena* i *Rodzaj literatury*. Usuń etykiety pól, a pola tekstowe umieść w jednej linii, jak najbliżej krawędzi górnej. Do widoku dodaj sekcje *Nagłówek formularza* i *Stopka formularza*. Aby to zrobić, kliknij prawym przyciskiem myszy krawędź *Szczegóły* i wybierz z listy *Nagłówek/stopka formularza*. W nagłówku formularza umieść etykiety opisujące pola położone poniżej (*Tytuł książki*, *Cena*, *Temat książki*). W nagłówku lub stopce formularza zaprojektuj przycisk polecenia *Zamknij formularz*. Zmniejsz wysokość każdej sekcji do minimalnych rozmiarów. Otwórz *Arkusz właściwości* formularza i w karcie *Format* ustal właściwość *Widok domyślny*. Wybierz wartość *Formularze ciągłe* (rysunek 2.49).

Sprawdź w opcji *Widok formularza*, jak prezentuje się utworzony formularz. Ustaw potrzebne właściwości formularza.

2.5.10. Formularz z podformularzem

Podformularz to formularz wstawiony do innego formularza, nazywanego **formularzem głównym**. Podformularze są wykorzystywane, gdy trzeba przedstawić dane pochodzące z tabel połączonych relacją „jeden do wielu". Formularz główny zawiera dane ze strony *jeden*, a podformularz — dane ze strony *wiele* tej relacji.

Rysunek 2.49.
Widok formularza
Formularze ciągłe



Można na przykład utworzyć formularz zawierający podformularz w celu przedstawienia danych z tabel *Autorzy* i *Książki*. Dane w tabeli *Autorzy* to strona *jeden* relacji. Dane w tabeli *Książki* to strona *wiele* relacji.

Podformularz i formularz są połączone, dzięki czemu w podformularzu wyświetlane są tylko rekordy związane z bieżącym rekordem formularza głównego. Jeśli na przykład w formularzu głównym wyświetla się nazwisko autora — Mickiewicz — w podformularzu można zobaczyć tylko książki tego autora. Podformularz może być wyświetlany w widokach *Arkusz danych* lub *Formularze ciągłe*.

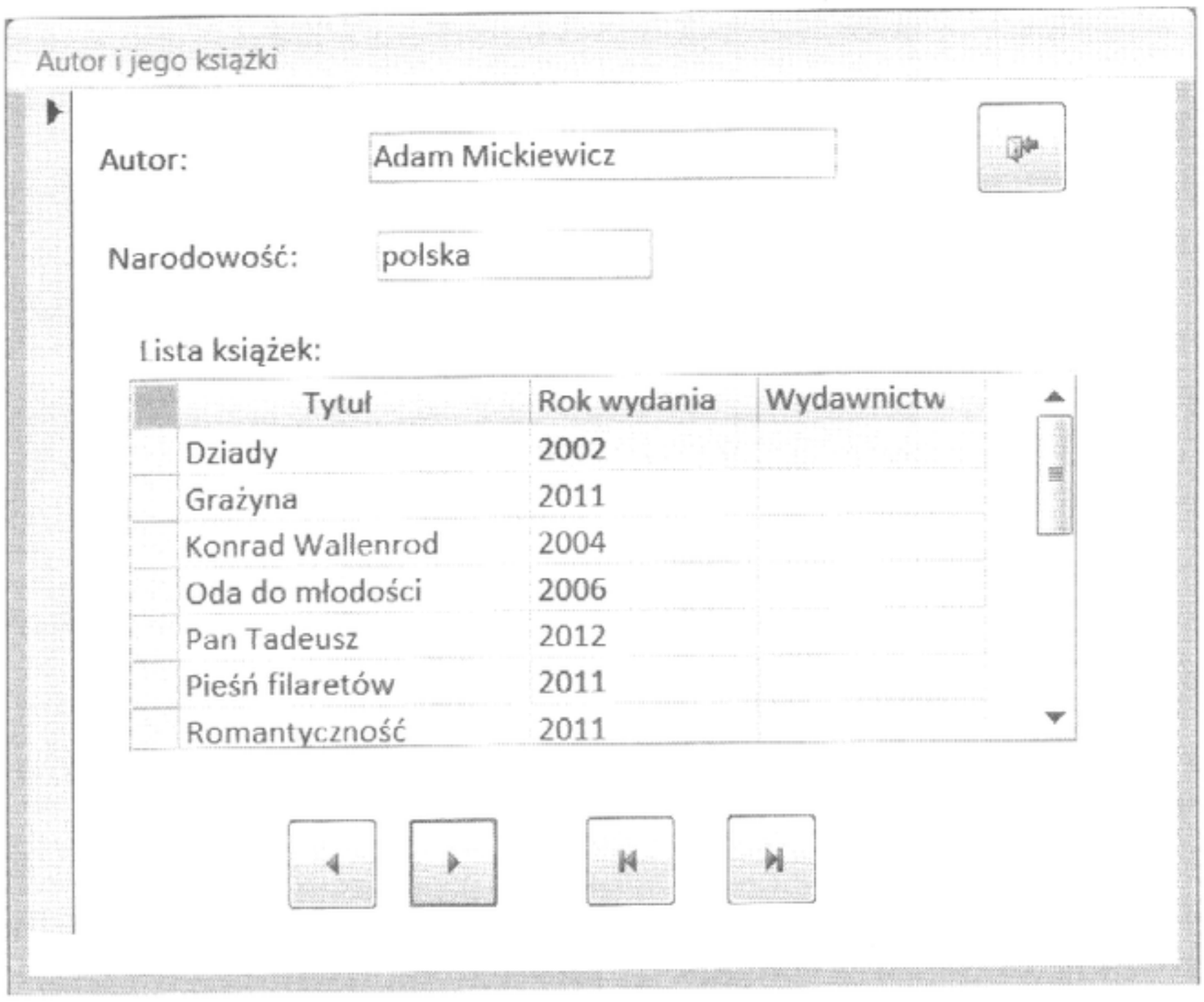
Podformularz może być projektowany za pomocą kreatora formularzy lub samodzielnie, poprzez zaprojektowanie dwóch formularzy: pierwszego dla tabeli ze strony *jeden* i drugiego dla tabeli ze strony *wiele*. W formularzu dla tabeli ze strony *jeden* należy zaprojektować formant *Podformularz/Podraport* i związać go we właściwościach formantu z formularzem ze strony *wiele*, wybierając we właściwości *Obiekt źródłowy* nazwę podformularza. Właściwości *Podrzędne pole łączące* i *Nadrzędne pole łączące* powinny zostać automatycznie wypełnione nazwami pól, przez które zostało zdefiniowane połączenie między tabelami.

Formularz główny może zawierać dowolną liczbę podformularzy. Podformularze można także umieszczać jeden w drugim aż do siódmego poziomu zagnieżdżenia.

Przykład 2.24

Zaprojektuj formularz z podformularzem oparty na tabelach *Autorzy* (tabela ze strony *jeden*) i *Książki* (tabela ze strony *wiele*). W formularzu głównym umieść pola *Nazwisko i imię autora* oraz *Narodowość* z tabeli *Autorzy*. W podformularzu umieść pola *Tytuł*, *Rok wydania* i *Wydawnictwo* z tabeli *Książki*. W formularzu głównym dodatkowo zaprojektuj przyciski poleceń *Następny*, *Poprzedni*, *Pierwszy*, *Ostatni* oraz *Zamknij formularz*. Aby dane w podformularzu były pokazywane w postaci tabelki, we właściwości podformularza *Widok domyślny* wybierz *Arkusz danych*.

Po zmianie autora w formularzu głównym automatycznie powinna się zmienić lista książek. Dodatkowo ustaw pozostałe niezbędne do prawidłowej pracy właściwości formularzy. Gotowy formularz przedstawiono na rysunku 2.50.



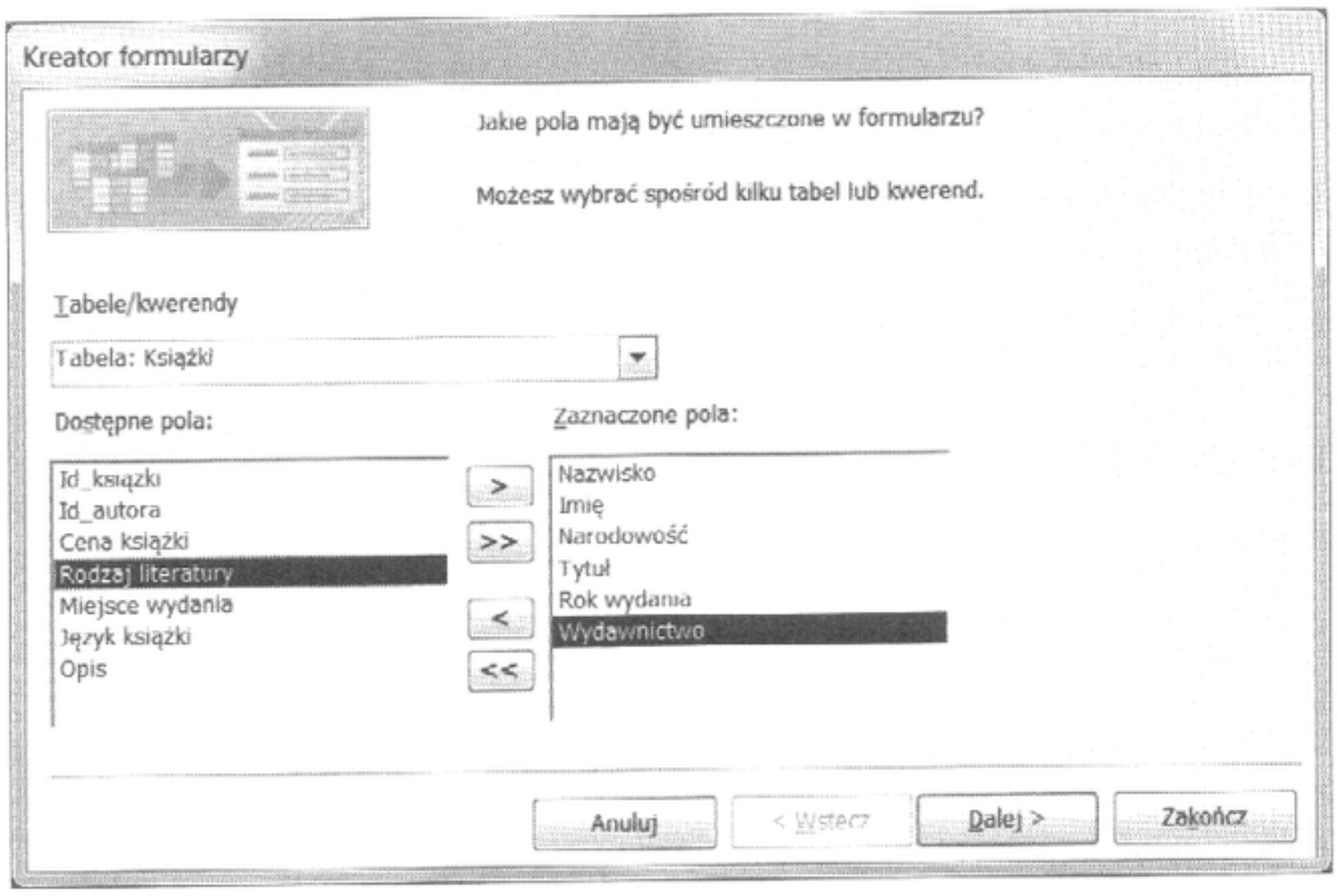
Rysunek 2.50. Formularz z podformularzem

Na karcie *Tworzenie* w grupie *Formularze* wybierz ikonę *Więcej formularzy*. Z dostępnej listy wybierz *Kreator formularzy*. Zostanie otwarte okno projektowania formularza (rysunek 2.51).

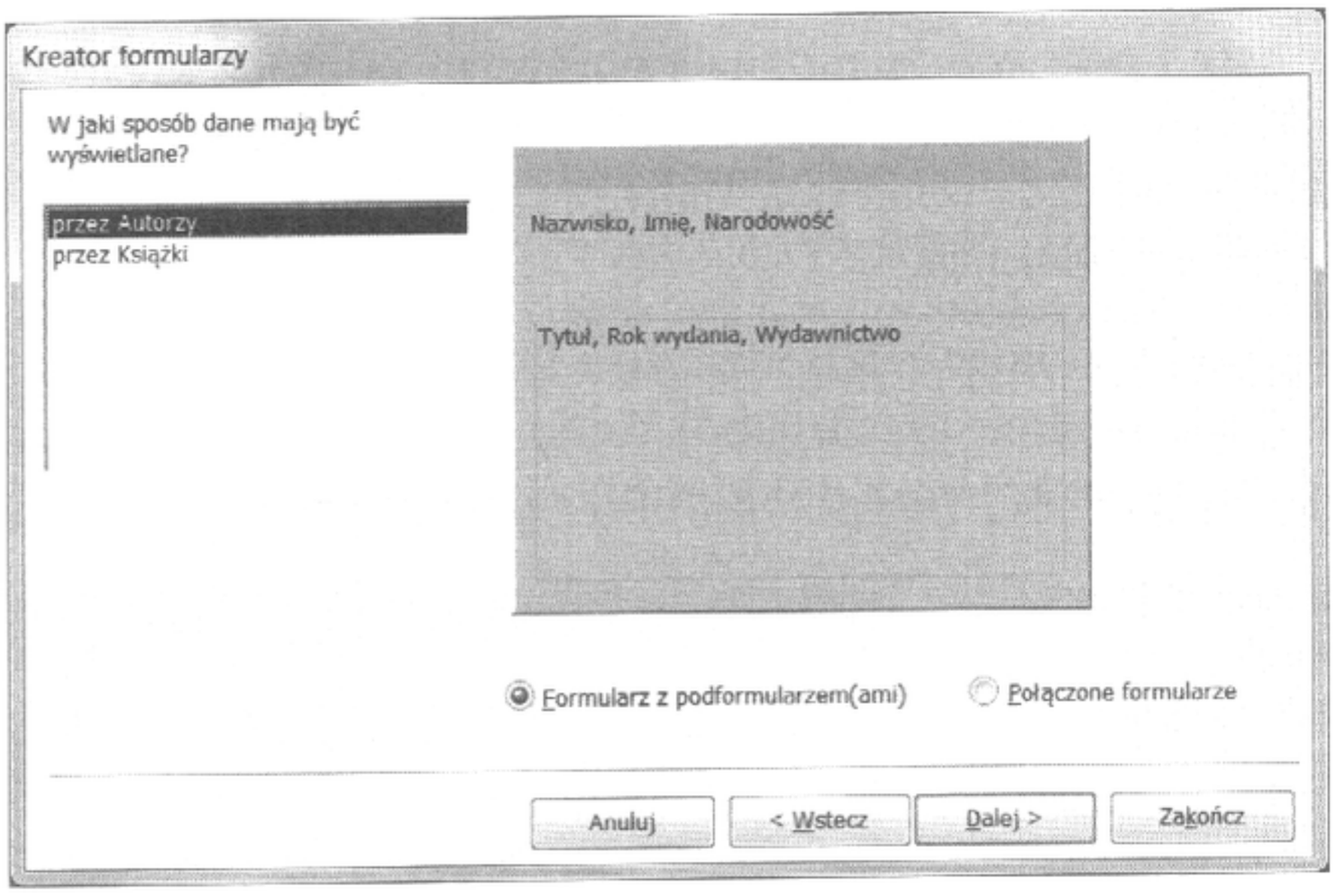
Wybierz pola, które będą używane w formularzu (*Nazwisko*, *Imię*, *Narodowość* z tabeli *Autorzy*) oraz w podformularzu (*Tytuł*, *Rok wydania* i *Wydawnictwo* z tabeli *Książki*). Przejdź do kolejnego okna kreatora, w którym określisz sposób wyświetlania danych. Wybierz wyświetlanie danych *przez Autorzy* (rysunek 2.52). Spowoduje to włączenie opcji *Formularz z podformularzem* i zaprojektowanie formularza w tym układzie.

Przejdź do następnego okna, w którym określisz widok wyświetlania zawartości podformularza. Najczytelniejszy jest widok *Arkusz danych*. Wybierz go. W kolejnym oknie wybierz dowolny styl formularza i w następnym kroku wpisz nazwę formularza *Autor* oraz nazwę podformularza *Lista książek*. Zakończ projektowanie formularza i nazwij go *Autor i jego książki*. Sprawdź działanie formularza. Teraz możesz w widoku projektu zmodyfikować utworzony formularz, dodając do niego odpowiednie przyciski poleceń

Rysunek 2.51. Kreator formularzy



Rysunek 2.52. Kreator formularzy — zaznaczona opcja tworzenia podformularza



oraz ustawiając właściwości formularza i podformularza. Możesz również próbować zaprojektować formularz z podformularzem bez użycia kreatora formularzy według podanego wyżej opisu.

2.5.11. Formanty listy

W programie Access występują dwa formanty listy: *pole listy* i *pole kombi*. Obiekty te są tworzone w formularzu. Ułatwiają pracę przy wprowadzaniu danych (zamiast wpisywać jakąś wartość, wystarczy wybrać ją z listy), zmniejszają też prawdopodobieństwo popełnienia błędu przy wprowadzaniu danych oraz pomagają w wyszukiwaniu rekordów. Listy można stosować tylko wtedy, gdy dane mają ograniczony zbiór wartości.

Pola listy i pola kombi mogą również zostać utworzone w tabeli. Tak powstałe pola pobierają wartości z innej tabeli, kwerendy lub listy wartości i wyświetlają je jako dane tabeli. Dotyczy to głównie pól klucza obcego, które służą do tworzenia relacji między tabelami.

Pole listy

Pól listy można używać w formularzach i na stronach dostępu do danych. Taka lista jest zawsze widoczna, a wartość formantu jest ograniczona do ustalonego zbioru elementów listy. Jeśli do wprowadzania lub edytowania danych użyto formularza, w formancie nie można ustawić wartości spoza ustalonej listy.

Pole kombi

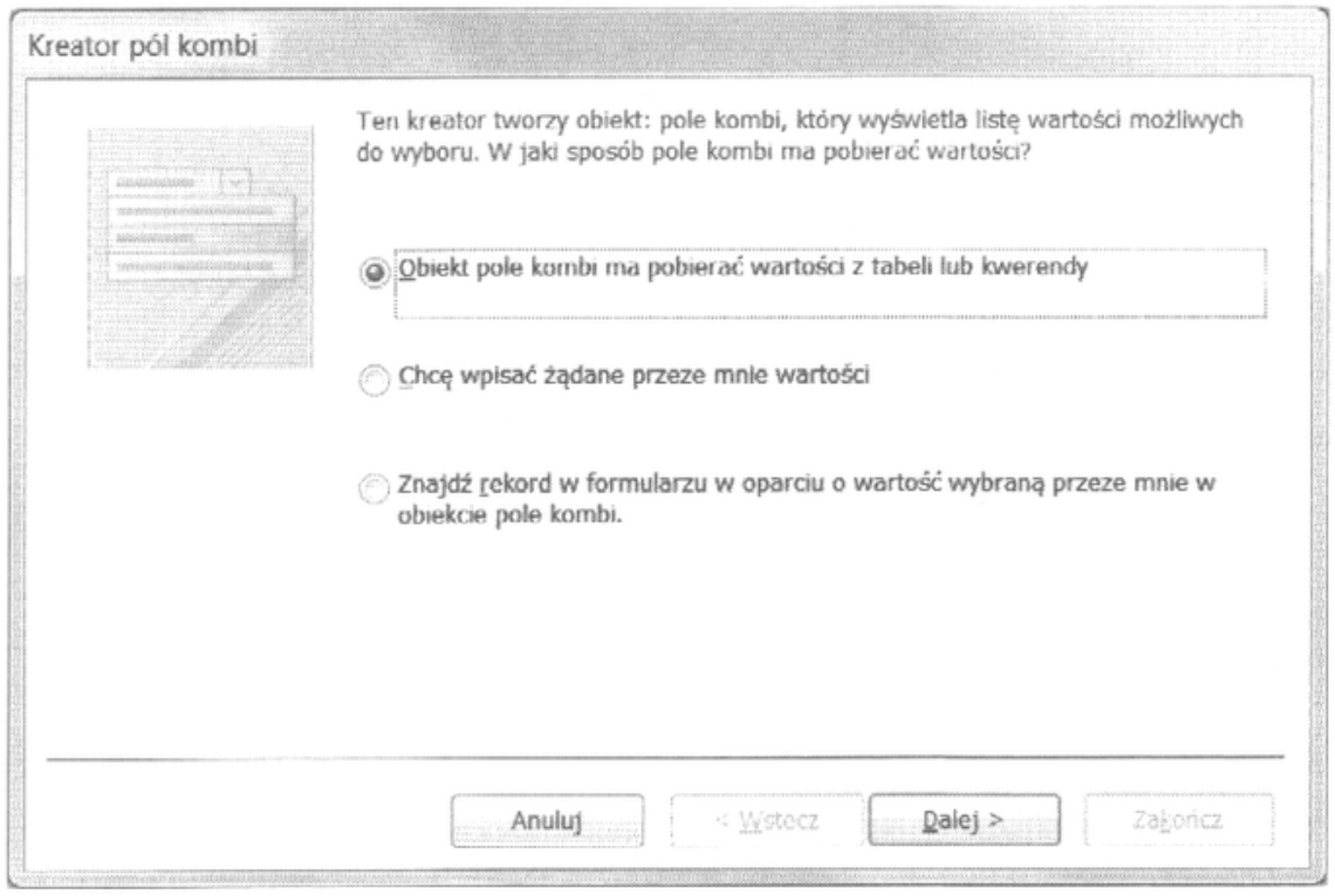
W polu kombi lista nie jest wyświetlana, dopóki nie zostanie otwarta, dzięki czemu formant ten zajmuje w formularzu mniej miejsca. Można tu także zezwolić na wprowadzenie dowolnej wartości lub zabronić wprowadzania wartości spoza zbioru wartości występujących na liście.

Aby w obu rodzajach list szybko przejść do wartości zaczynającej się na konkretną literę, wystarczy wpisać tę literę.

Pole listy i *pole kombi* można tworzyć za pomocą kreatora list lub samodzielnie. Po zbudowaniu formantu można definiować jego właściwości. Aby utworzyć listę, należy ustalić, skąd będą pobierane dane oraz do czego zostaną użyte wartości wybrane z pola listy lub pola kombi.

Dane do listy mogą pochodzić ze zdefiniowanej przez nas listy wartości (druga opcja wybierana w oknie kreatora pól kombi), z pola lub pól tabeli albo z kwerendy (pierwsza opcja w oknie kreatora pól kombi, zaznaczona na rysunku 2.53).

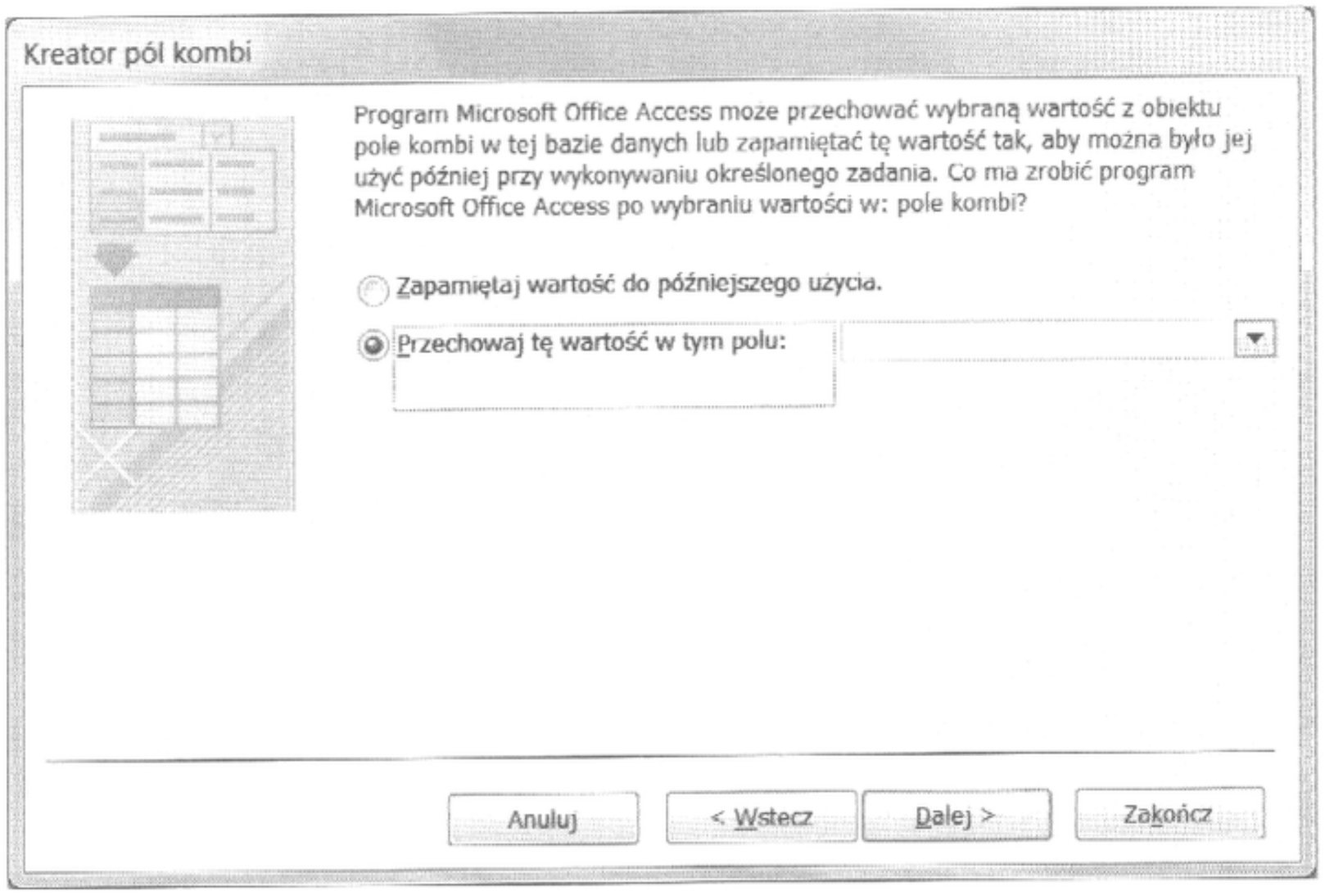
Rysunek 2.53. Kreator pól kombi — wybór listy wartości



Wartości wybrane z pola listy mogą zostać zapisane w polu tabeli (druga opcja w ostatnim oknie kreatora) lub posłużyć do określenia zawartości innego formantu (pierwsza opcja w ostatnim oknie kreatora, zaznaczona na rysunku 2.54).

Projektowanie listy za pomocą kreatora

Za pomocą narzędzia *Kreator formantów* można w ten sam sposób tworzyć pola listy i pola kombi służące do wprowadzania danych do tabel lub do wyszukiwania rekordu.



Rysunek 2.54. Kreator pól kombi — określenie sposobu zapamiętania wybranej wartości

Aby zaprojektować pole kombi przy użyciu kreatora formantów, należy w widoku projektu, na karcie *Projektowanie*, w grupie *Formanty*, włączyć opcję *Użyj kreatorów formantów*, a następnie w tej samej grupie wybrać ikonę *Pole kombi* i w projekcie formularza narysować pole kombi. Zostanie uruchomiony *Kreator pola kombi*. W pierwszym oknie kreatora trzeba zaznaczyć opcję decydującą, do jakiego celu zostanie użyte pole kombi, w drugim wybrać źródło danych, a w trzecim określić, które z pól źródła danych powinny być widoczne w polu kombi. W ostatnim kroku trzeba ustalić, gdzie będzie zapamiętana wartość wybrana z listy.

Wprowadzanie danych za pomocą pola kombi

Przykład 2.25

We wcześniej zaprojektowanym formularzu *Nowe książki*, opartym na tabeli *Książki*, przeznaczonym do wprowadzania danych o nowych publikacjach dostarczonych do księgarni internetowej, umieść pole pozwalające wybrać z listy autora książki. Funkcję zrealizuj za pomocą pola kombi zawierającego listę autorów z tabeli *Autorzy*. Wybór autora z listy powinien spowodować przypisanie książki do wybranego autora.

Otwórz formularz *Nowe książki* w widoku projektu. Włącz *Kreator formantów* i na karcie *Projektowanie* w grupie *Formanty* kliknij ikonę *Pole kombi*. W pierwszym oknie kreatora wybierz opcję pobierania wartości do pola kombi z tabeli (pierwsza opcja). Przejdź do następnego okna. Wybierz z listy tabelę *Autorzy*, gdyż z niej będą pobierane wartości do pola kombi. Przejdź dalej i wybierz pola *Nazwisko* i *Imię*, gdyż z tych pól będą pobierane wartości do pola kombi. W następnym kroku możesz określić kolejność sortowania wartości w polu kombi. Wybierz pole *Nazwisko*. Ostatni krok to ustalenie, gdzie w tabeli *Książki* zostanie zapamiętana wartość wybrana z listy. Wybierz opcję *Przechowaj tę wartość w tym polu* i z listy wybierz pole *Id_autora*, ponieważ w tabeli *Książki* w tym polu przechowywana jest informacja o autorze książki. Zakończ projektowanie pola kombi i sprawdź, jak działa nowy formant. Gotowy formularz z polami kombi zaprezentowano na rysunku 2.55.

Rysunek 2.55. Formularz z polami kombi

Przykład 2.26

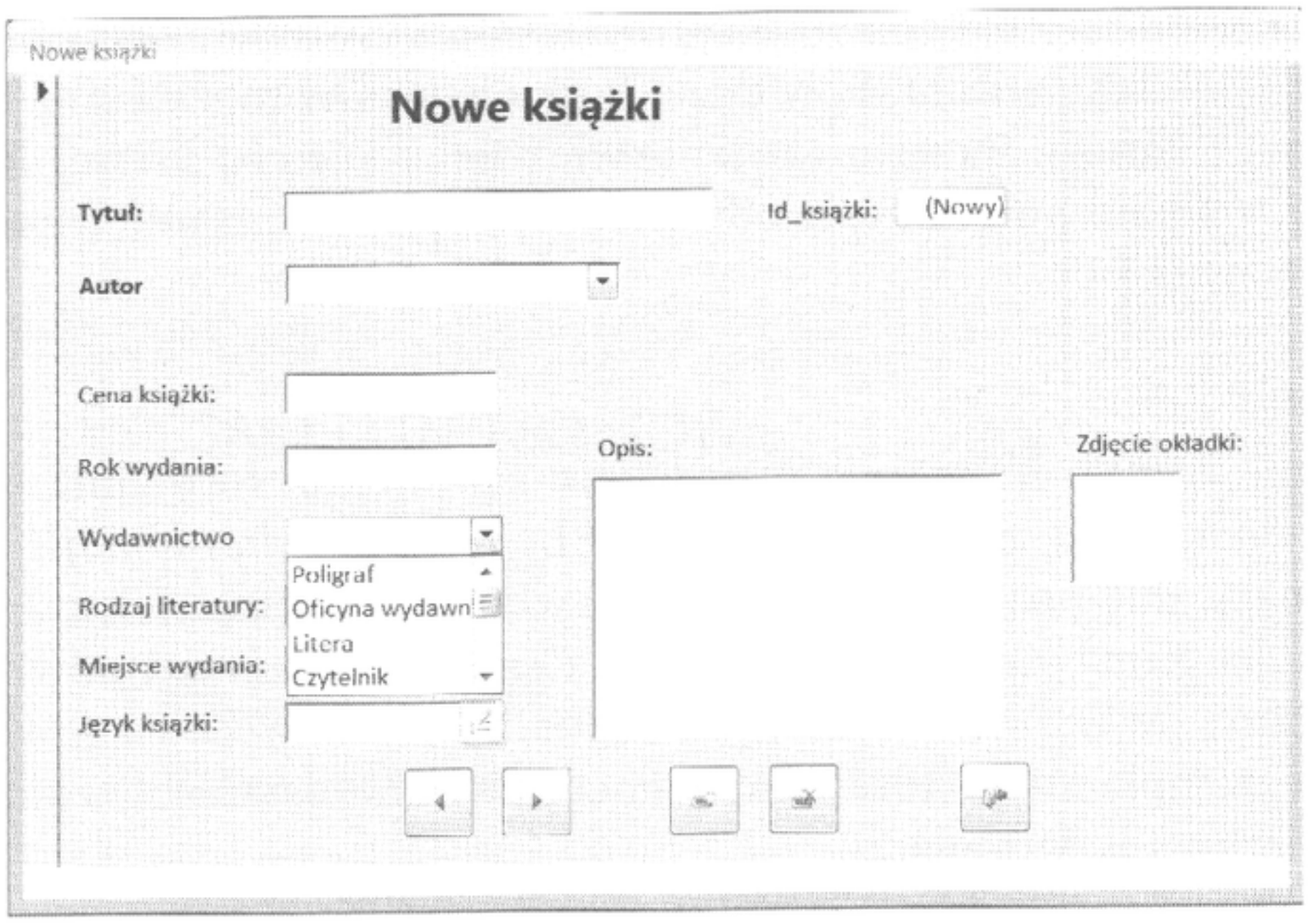
Zmodyfikuj formularz *Nowe książki* z przykładu 2.25, umieszczając w nim pole kombi z listą wartości, na podstawie których będzie można wybrać wydawnictwo. Przyjmij założenie, że na liście mogą pojawić się następujące wydawnictwa: *Poligraf*, *Oficyna wydawnicza*, *Litera*, *Czytelnik*, *Lektury*.

Otwórz formularz *Nowe książki* w widoku projektu. Włącz *Kreator formantów* i na karcie *Projektowanie* w grupie *Formanty* kliknij ikonę *Pole kombi*. W pierwszym oknie kreatora wybierz drugą opcję — *Chcę wpisać żądane przeze mnie wartości*. Przejdź do następnego okna i wpisz listę wartości, które będą dostępne w polu kombi (*Poligraf*, *Oficyna wydawnicza*, *Litera*, *Czytelnik*, *Lektury* — rysunek 2.56).

Rysunek 2.56. Lista wartości pola kombi

	Kol1
	Poligraf
	Oficyna wydawnicza
	Litera
	Czytelnik
	Lektury
*	

Ostatni krok to określenie, gdzie w tabeli *Książki* zostanie zapamiętana wartość wybrana z listy. Wybierz opcję *Przechowaj tę wartość w tym polu* i z listy wybierz pole *Wydawnictwo*, ponieważ w tabeli *Książki* w tym polu przechowywana jest informacja o wydawcy książki. Zakończ projektowanie pola kombi i sprawdź, jak działa nowy formant. Gotowy formularz zaprezentowano na rysunku 2.57.



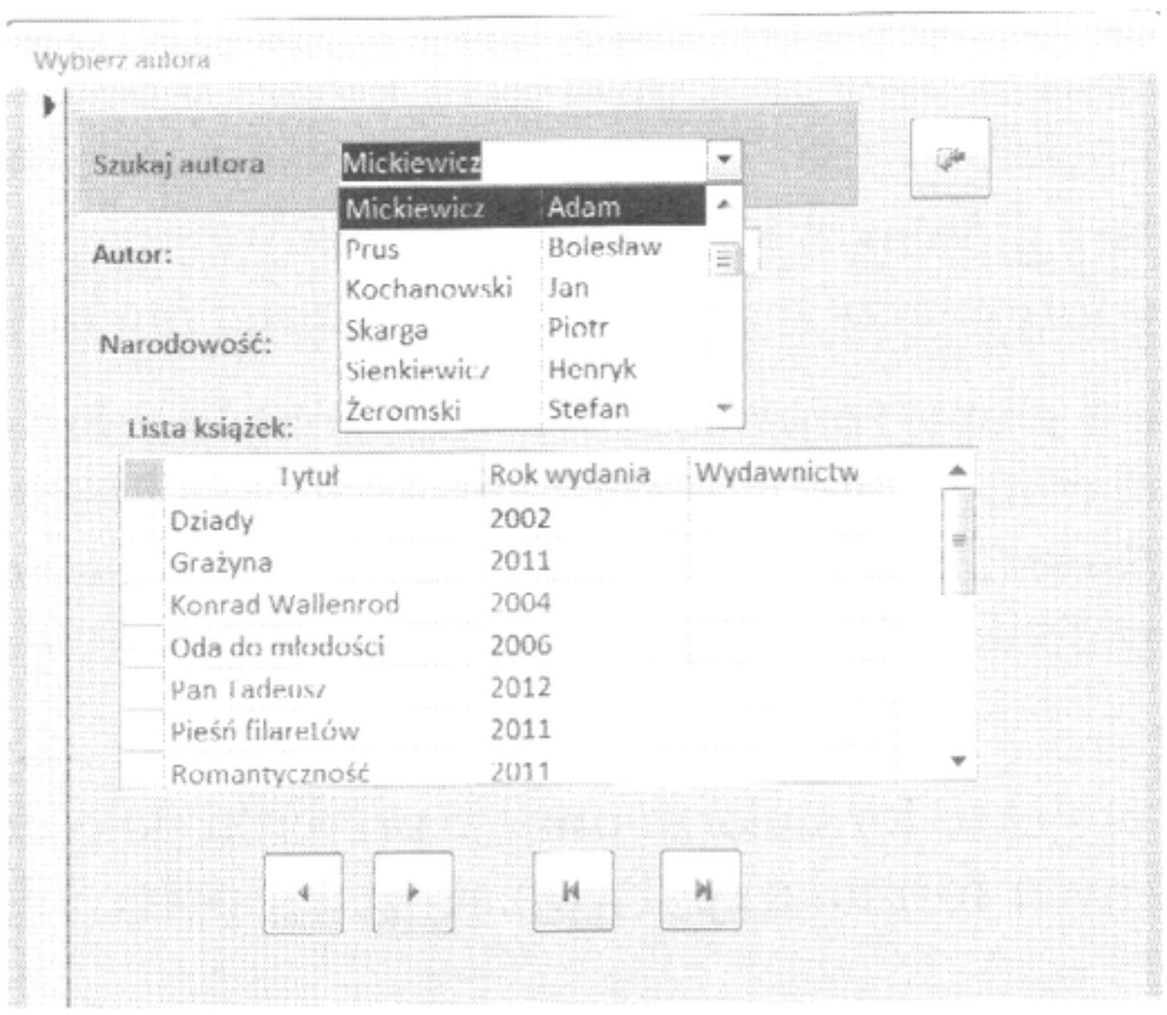
Rysunek 2.57. Formularz z listą wartości

Wyszukiwanie rekordu za pomocą pola kombi

Przykład 2.27

Dla zaprojektowanego w przykładzie 2.24 formularza z podformularzem *Autor i jego książki* opartego na tabelach *Autorzy* (tabela ze strony *jeden*) i *Książki* (tabela ze strony *wiele*) zaprojektuj mechanizm wyszukiwania autora na liście (lista rozwijana) i wyświetlenia listy jego książek.

Otwórz formularz *Autor i jego książki* w widoku projektu. Włącz *Kreator formantów* i na karcie *Projektowanie* w grupie *Formanty* kliknij ikonę *Pole kombi*. W pierwszym oknie wybierz *Znajdź rekord w formularzu* (trzecia opcja). Przejdź do następnego okna. Wybierz z listy pola *Nazwisko* i *Imię*, gdyż z nich będą pobierane wartości dla pola kombi. W następnym kroku możesz określić szerokość kolumny. W ostatnim kroku wpisz nazwę etykiety *Wybierz autora*. Zakończ projektowanie pola kombi. Zapisz zmieniony formularz. Sprawdź jego działanie. Gotową postać formularza zaprezentowano na rysunku 2.58.



Rysunek 2.58. Formularz z polem kombi do wyszukiwania rekordów

Wybranie autora z listy powinno powodować wyświetlenie w podformularzu informacji o książkach tego autora. Przy dużej liczbie autorów można zmieniać liczbę pozycji na liście rozwijanej poprzez ustawienie właściwości *Liczba wierszy listy* dla formantu *Pole kombi*, a wybór autora może odbywać się poprzez wpisanie w polu kombi pierwszych liter nazwiska.

Zadanie 2.5

W bazie danych *Księgarnia internetowa* zaprojektuj formularz do rejestracji nowych zamówień. Dane klienta (*Adres*, *PESEL*, *Telefon*, *Adres e-mail*) powinny pojawić się automatycznie po wybraniu z listy rozwijanej nazwiska klienta. Również informacje dotyczące zamawianej książki (*Autor*, *Cena*, *Rok wydania*) powinny pojawić się automatycznie po wybraniu z listy rozwijanej tytułu książki. Po wpisaniu liczby zamawianych egzemplarzy wartość zamówienia powinna zostać obliczona automatycznie i być wyświetlona w formularzu.

2.5.12. Formularz sterujący

Tworząc system obsługi bazy danych, powinniśmy się starać, aby użytkownik miał szybki dostęp do danych. Zapewnią to prawidłowo opracowane formularze, oparte na tabelach i kwerendach. Wykorzystanie odpowiednich narzędzi w czasie projektowania formularzy pozwoli kontrolować dane i zapewni ich bezpieczeństwo. Narzędzia sterowania umożliwiają nawet niedoświadczonym użytkownikom sprawne posługiwanie się bazą danych. Elementy graficzne podnoszą atrakcyjność projektu i podkreślają profesjonalizm jego twórców.

Przy projektowaniu systemu obsługi bazy danych zakładamy, że użytkownik będzie miał dostęp do danych tylko przez formularze. Projektujemy liczne formularze przeznaczone do różnych zadań. Potrzebne jest narzędzie, które będzie sterowało obsługą aplikacji. Może to być formularz główny, w którym za pomocą przycisków poleceń będziemy sterować zadaniami aplikacji. Ostateczny wybór narzędzi i działań zależy od tego, jakie zadania zostały określone dla bazy danych. W najprostszej postaci przy użyciu tego formularza można sterować dostępem do innych formularzy.

Przykład 2.28

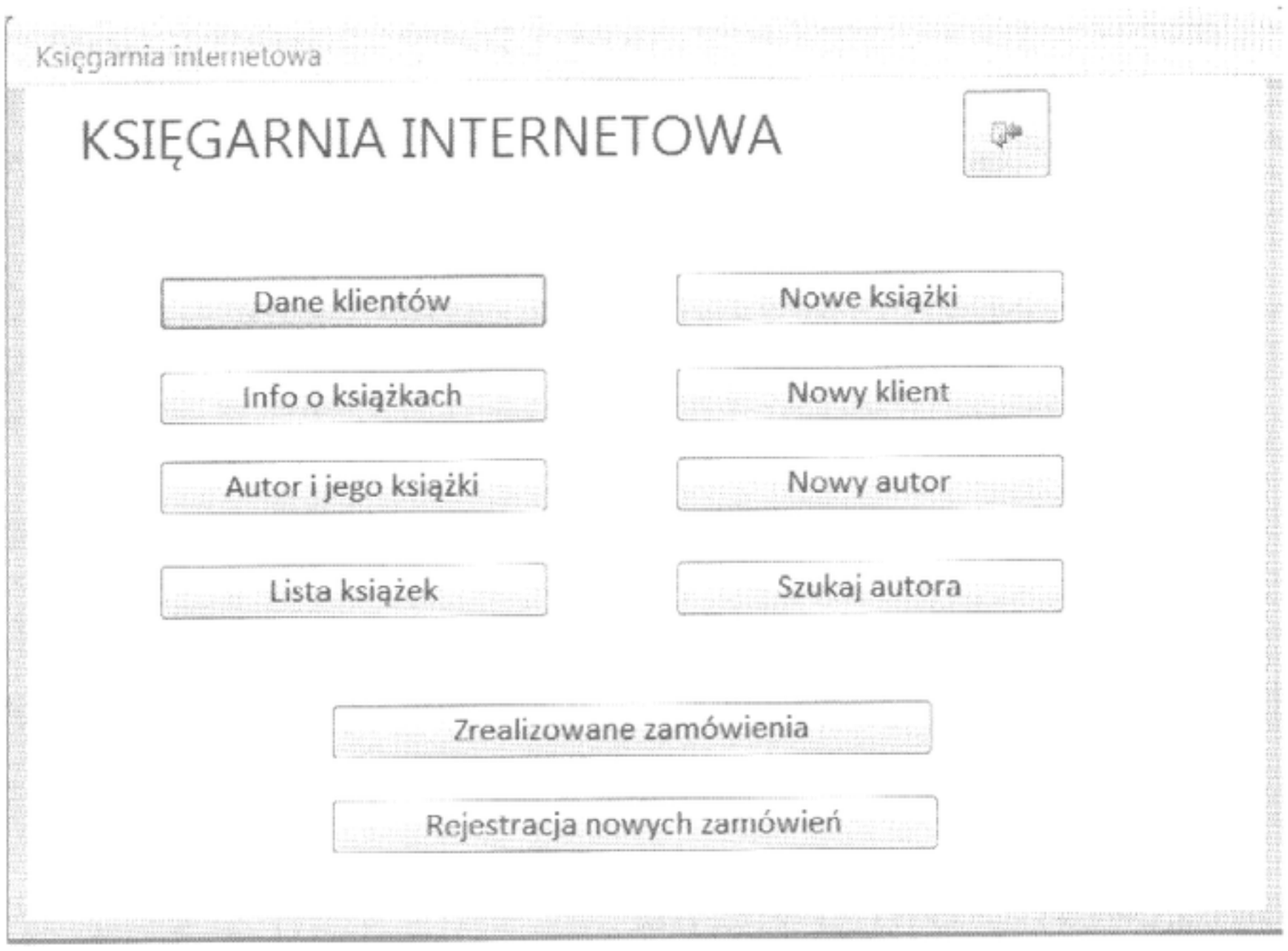
Zaprojektuj formularz główny o nazwie *Główny*, który przeznaczony będzie do sterowania dostępem do formularzy bazy danych *Księgarnia internetowa*.

Na karcie *Tworzenie* w grupie *Formularze* wybierz ikonę *Projekt formularza*. Tytuł umieść w nagłówku formularza, wybierając formant *Tytuł* lub *Etykieta*. Obok w nagłówku umieść przycisk *Zakończ*, który spowoduje zakończenie pracy z aplikacją. W tym celu, projektując makropolecenie dla tego przycisku, wybierz akcję *Zakończ*, która kończy pracę z aplikacją.

W sekcji *Szczegóły* projektu formularza umieść przyciski poleceń do otwierania wszystkich zaprojektowanych formularzy. Tworząc makropolecenia dla tych przycisków,

wybierz akcję *Otwórz formularz*, a w argumencie akcji *Nazwa formularza* wybierz nazwę formularza, który będzie otwierany projektowanym przyciskiem (rysunek 2.59). Ustaw niezbędne właściwości formularza głównego, wyłączając dostęp do nieużywanych funkcji formularza.

Rysunek 2.59. Formularz sterujący bazą danych



Następnym etapem będzie zaprojektowanie narzędzia, które spowoduje automatyczne uruchomienie formularza *Główny* w chwili otwierania bazy danych. Jednym ze sposobów jest zaprojektowanie makropolecenia o nazwie `AutoExec`. Makropolecenie to zostało przypisane do zdarzenia wyjątkowego, jakim jest otworzenie bazy danych. Jeśli makropolecenie o takiej nazwie znajdzie się w bazie danych, zostanie uruchomione w chwili jej otwierania. Makropolecenie to, poza nazwą, niczym nie różni się od innych makropoleceń. Można zapisać w nim ukrycie okna bazy danych, ukryć lub zmodyfikować wbudowane paski narzędzi, zdefiniować autoryzację użytkowników lub uruchomić formularz sterujący.

Przykład 2.29

Zaprojektuj makropolecenie o nazwie `AutoExec`, którego zadaniem będzie uruchomienie formularza *Główny* sterującego obsługą bazy danych *Księgarnia internetowa*.

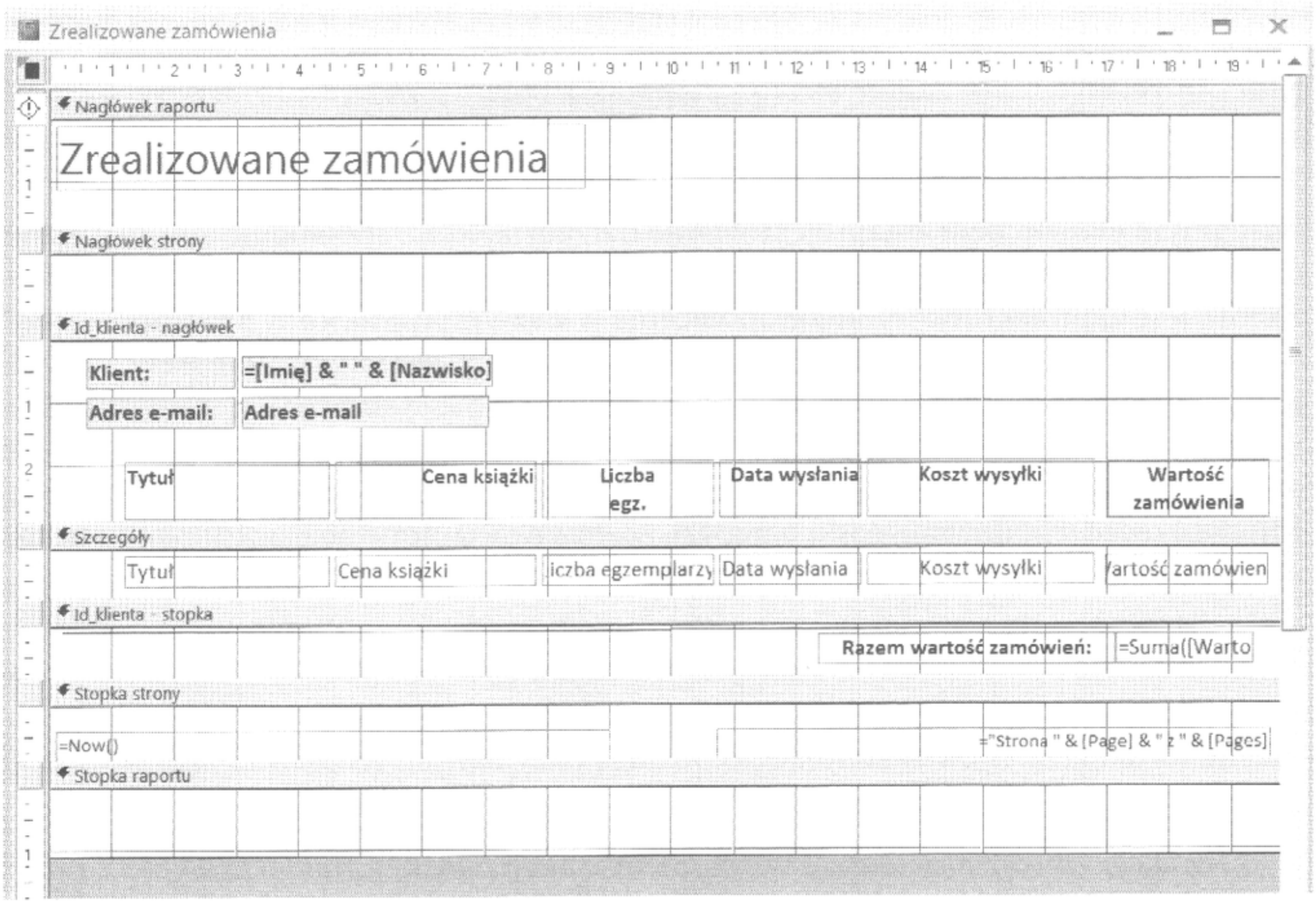
Na karcie *Tworzenie* w grupie *Inne* kliknij ikonę *Makro*. W oknie projektowania makra wybierz akcję *Otwórz formularz*, a w argumencie akcji *Nazwa formularza* wybierz z listy nazwę formularza *Główny*. Zapisz makro pod nazwą `AutoExec`. Na liście obiektów bazy danych powinno pojawić się zaprojektowane makro. Zamknij bazę danych i otwórz ją ponownie, klikając ikonę bazy. Formularz *Główny* powinien zostać otwarty automatycznie. Przetestuj działanie wszystkich przycisków poleceń.

2.6. Raporty

Zawartość tabeli, kwerendy czy formularza może zostać wydrukowana, ale obiektem specjalnie przeznaczonym do drukowania jest **raport**. Raporty projektujemy, gdy chcemy tworzyć dla danych grupy i podsumowania oraz gdy planujemy przeprowadzenie analizy danych. Wyniki grupowania rekordów i podsumowań można przeglądać na ekranie, ale raporty głównie służą do sporządzania wydruków. Procedura tworzenia raportu jest pracochłonna, dlatego wygodniej jest zaprojektować raport z użyciem kreatora raportu, a później zmodyfikować go w widoku projektu i doprowadzić do ostatecznej postaci.

Grupowane zbiory rekordów zawierają takie elementy, jak nagłówek i podsumowanie. Grupa składa się z nagłówka grupy, grup zagnieżdżonych (jeżeli występują), rekordów szczegółów i stopki grupy zawierającej podsumowania.

Aby ułatwić tworzenie raportu, okno *Projekt raportu* zostało podzielone na *sekcje* (rysunek 2.60).



Rysunek 2.60. Okno Projekt raportu

- **Nagłówek raportu** — sekcja drukowana jednorazowo na początku raportu. W nagłówku raportu można zamieszczać informacje, które zwykle znajdują się na okładce, na przykład logo, tytuł i datę. Nagłówek raportu jest drukowany przed nagłówkiem strony.
- **Nagłówek strony** — sekcja drukowana u góry każdej strony. W nagłówku strony można na przykład powtarzać tytuł raportu na każdej stronie.

- **Nagłówek grupy** — sekcja drukowana na początku każdej grupy rekordów. W nagłówku grupy można drukować nazwę grupy. W raporcie grupowanym według klientów można na przykład drukować w nagłówku grupy nazwisko klienta.
- **Szczegóły** — sekcja drukowana jednorazowo dla każdego rekordu ze źródła rekordów. Umieszczane są w niej formanty, które tworzą główną część raportu.
- **Stopka grupy** — sekcja drukowana na końcu każdej grupy rekordów. W stopce grupy można drukować informacje podsumowujące dla grupy.
- **Stopka strony** — sekcja drukowana u dołu każdej strony. W stopce strony można drukować numery stron oraz informacje dotyczące poszczególnych stron.
- **Stopka raportu** — sekcja drukowana jednorazowo na końcu raportu. W stopce raportu można drukować sumy lub inne informacje podsumowujące dla całego raportu.

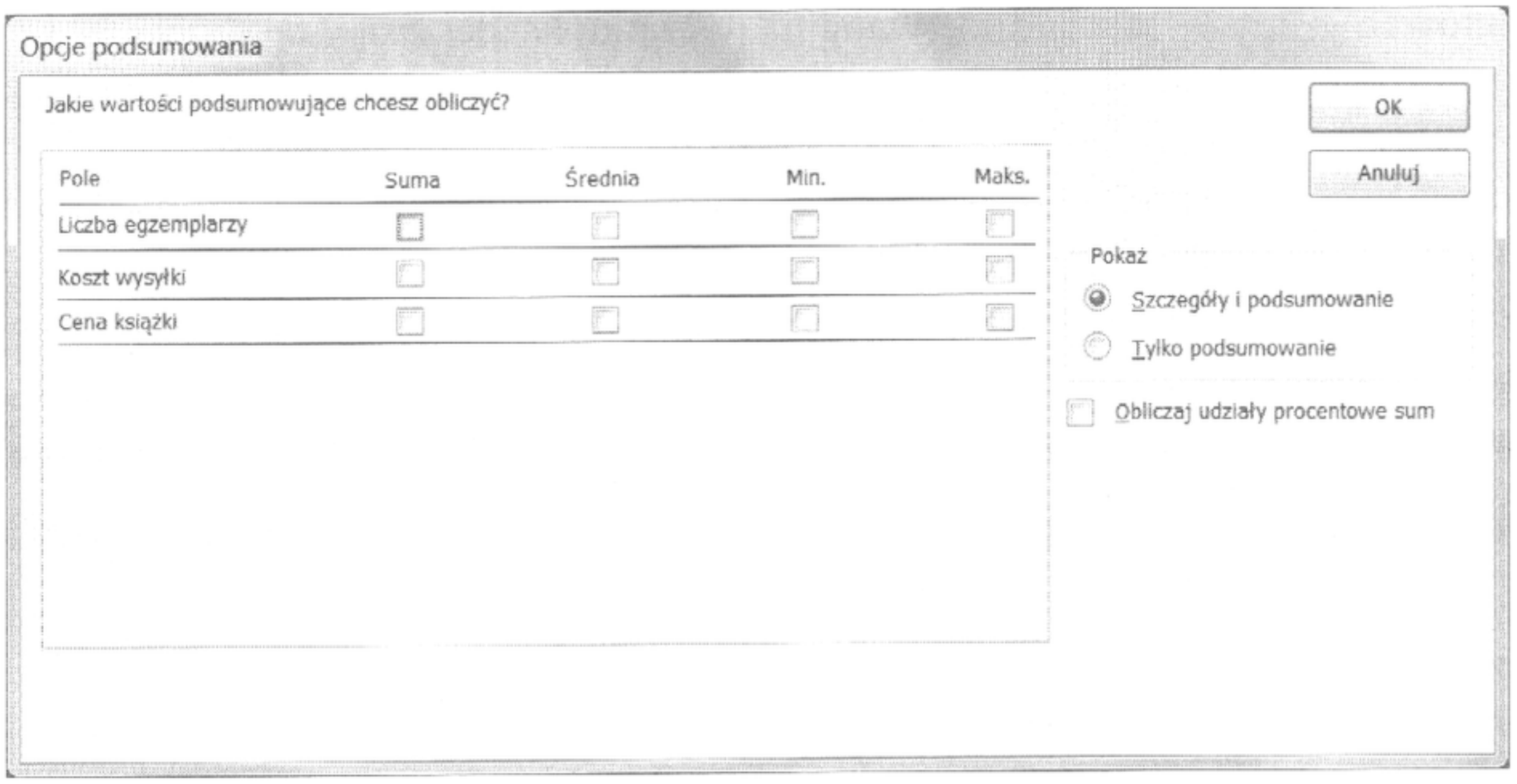
2.6.1. Projektowanie raportu przy użyciu kreatora

Przed przystąpieniem do tworzenia raportu należy określić informacje, jakie ten raport ma zawierać. Wśród wybranych pól muszą być te, według których rekordy będą grupowane lub podsumowywane. Źródłem rekordów dla raportu może być tabela lub kwerenda.

Przykład 2.30

Zaprojektuj raport, który będzie zawierał zestawienie zbiorcze ze zrealizowanych przez księgarnię zamówień. Dane w raporcie pogrupuj według klientów składających zamówienia. W nagłówku każdej grupy umieść nazwisko klienta oraz jego adres e-mail. W grupie umieść tytuł książki, jej cenę oraz liczbę zamówionych egzemplarzy, datę wysłania, koszt wysyłki oraz wartość zamówienia. W podsumowaniu grupy podaj całkowitą wartość zrealizowanych zamówień. Na końcu każdej strony umieść numer strony i datę wydrukowania raportu.

Na karcie *Tworzenie* w grupie *Raporty* wybierz ikonę *Kreator raportów*. W pierwszym oknie kreatora wybierz dla tabeli *Klient* pola *Nazwisko*, *Imię*, *Adres e-mail*, dla tabeli *Książki* pola *Tytuł* i *Cena książki*, a dla tabeli *Zamówienia* pola *Liczba egzemplarzy*, *Data wysłania* i *Koszt wysyłki*. W następnym oknie określ sposób grupowania danych. Pogrupuj dane według klienta (domyślny sposób grupowania). W kolejnym oknie można zdefiniować podgrupy. Skoro w tworzonym raporcie grupowanie odbywa się tylko według klienta, przejdź do następnego okna, w którym można określić porządek sortowania danych. Wybierz z listy sortowanie według tytułu książki. W tym oknie można również określić opcje podsumowań. Podsumowania można tworzyć tylko dla pól liczbowych. Jeżeli takich nie ma, nie wykonamy podsumowań za pomocą kreatora raportów. Po wybraniu przycisku *Opcje podsumowania* zostanie otwarte okno, w którym można zaznaczyć rodzaj wykonywanej operacji (rysunek 2.61). W projektowanym raporcie podsumowanie będzie dotyczyło wartości zrealizowanych zamówień (pole obliczeniowe), zatem nie można zrealizować tego podsumowania za pomocą kreatora.



Rysunek 2.61. Definiowanie opcji podsumowania

W kolejnym kroku wybierz układ raportu (na przykład *Konspekt*) i dalej styl raportu (na przykład *Access 2007*). Zakończ projektowanie raportu, nazwij go *Zamówienia* i wyświetl w widoku *Podgląd wydruku*.

Powstał raport, który nie jest zgodny z oczekiwaniami, dlatego należy go zmodyfikować. Otwórz raport w widoku projektu, w nagłówku raportu zmień tekst etykiety z tytułem raportu na `Zrealizowane zamówienia`, w nagłówku grupy *Id_klienta* dobierz odpowiednią szerokość pól *Liczba egzemplarzy* i *Data wysłania* oraz *Nazwisko* i *Imię* klienta (możesz również zmodyfikować nazwy etykiet tych pól lub zdefiniować pola obliczeniowe). W raporcie brakuje pola obliczeniowego *Wartość zamówienia*. Jedną z metod jego dodania jest zaprojektowanie pola obliczeniowego w kwerendzie, która jest źródłem raportu.

Wybierz we właściwościach raportu zakładkę *Dane*, następnie właściwość *Źródło rekordów*. Kliknij ikonę ..., w otwartym oknie projektowania kwerend utwórz pole obliczeniowe i wpisz wyrażenie `[Cena książki]*[Liczba egzemplarzy]+[Koszt wysyłki]`. Występujący przed wyrażeniem napis *Wyr1* zmień na `Wartość zamówienia`. Zamknij okno i zapisz zmiany wprowadzone do kwerendy.

W projekcie raportu z okna *Lista pól* wybierz pole *Wartość zamówienia* i umieść je w obszarze *Szczegóły* po formancie *Koszt wysyłki*. W raporcie zostanie wyświetlona wartość zrealizowanego zamówienia. W nagłówku grupy umieść nad zdefiniowanym polem etykietę *Wartość zamówienia*. Po modyfikacji raport powinien wyglądać tak jak na rysunku 2.62.

Zrealizowane zamówienia

Zrealizowane zamówienia

Klient: **Aleksander Pol**
Adres e-mail: **pol@wp.pl**

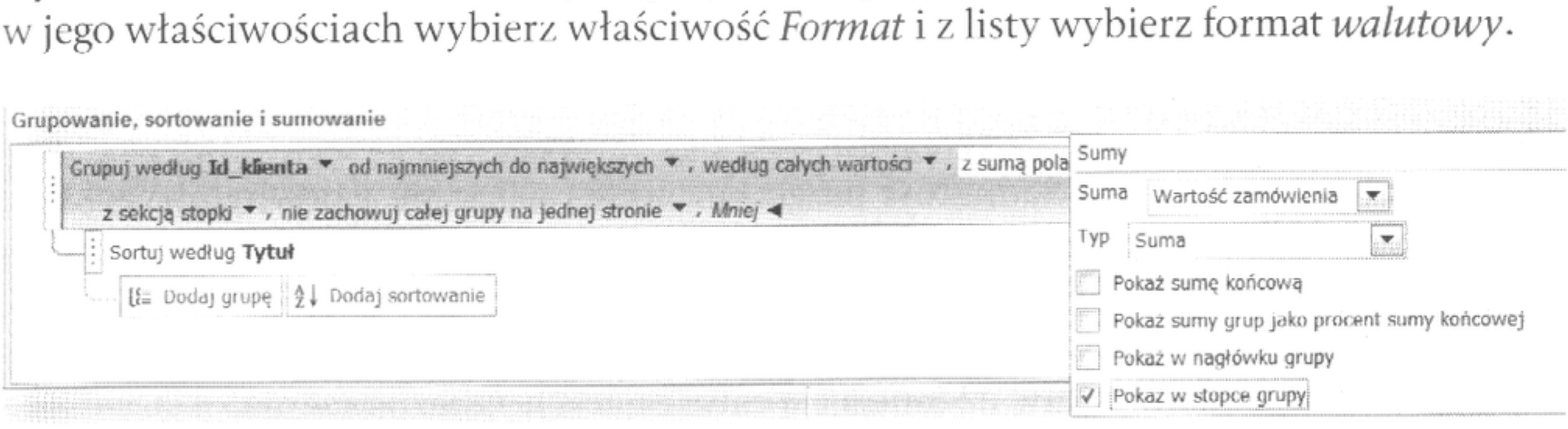
Tytuł	Cena książki	Ilość egz.	Data wysłania	Koszt wysyłki	Wartość zamówienia
Dziady	21,00 zł	4	2011-08-26	12,00 zł	96,00 zł
Konrad Wallenrod	10,00 zł	8	2011-09-05	21,00 zł	101,00 zł
Oda do młodości	10,00 zł	3	2011-10-09	12,00 zł	42,00 zł
Treny	13,00 zł	3	2011-11-18	12,00 zł	51,00 zł
Powrót posła	12,00 zł	6	2011-11-19	21,00 zł	93,00 zł
Wesele	18,00 zł	7	2011-12-15	21,00 zł	147,00 zł
Wiersze	16,00 zł	3	2012-02-26	12,00 zł	60,00 zł
Antygona	15,00 zł	2	2012-08-12	8,00 zł	38,00 zł
				Razem wartość zamówień:	628,00 zł

Klient: **Marek Nowak**
Adres e-mail: **nowak@onet.pl**

Tytuł	Cena książki	Ilość egz.	Data wysłania	Koszt wysyłki	Wartość zamówienia
Katarynka	8,00 zł	2	2011-08-09	8,00 zł	24,00 zł
Lalka	38,00 zł	4	2011-09-08	12,00 zł	164,00 zł
Nie-Boska komedia	16,00 zł	3	2011-09-13	12,00 zł	60,00 zł

Rysunek 2.62. Raport dotyczący zrealizowanych zamówień

W zestawieniu brakuje podsumowania dotyczącego wartości wszystkich zamówień jednego klienta. Na karcie *Projekt* w obszarze *Grupowanie i sumy* kliknij przycisk *Grupuj i sortuj*. Na dole ekranu zostanie otwarte okno *Grupowanie, sortowanie i sumowanie*, w którym zostały pokazane operacje zdefiniowane dla grupy *Klient*. Ponieważ w podsumowaniu grupy należy podać całkowitą wartość zrealizowanych dla klienta zamówień, w otwartym oknie kliknij przycisk *Więcej* i wybierz opcję *Bez sum*. W otwartym oknie w polu *Suma* wybierz z listy *Wartość zamówienia*, w polu *Typ* wybierz *Suma* i zaznacz opcję *Pokaż w stopce grupy* (rysunek 2.63). Jeżeli wartość wszystkich zamówień klienta nie jest wyświetlana w formacie waluty, w projekcie raportu zaznacz ten formant. Następnie w jego właściwościach wybierz właściwość *Format* i z listy wybierz format *walutowy*.



Rysunek 2.63. Definiowanie podsumowania w grupie

Zauważ, że w stopce strony kreator automatycznie umieścił formanty obliczeniowe z funkcjami `Now()` i `Page()`, które spowodują wydrukowanie daty bieżącej i numerów stron. Zapisz zmiany w raporcie.

Po zapisaniu projektu raportu można go wielokrotnie używać. Każdy wydruk raportu będzie zawierał bieżące dane. Jeśli będzie potrzebny nieco inny raport, można zmodyfikować projekt lub utworzyć nowy.

2.6.2. Drukowanie raportu

Raport można wydrukować z poziomu podglądu wydruku lub widoku projektu, a także z poziomu okna nawigacji. Przed wydrukowaniem warto dokładnie sprawdzić ustawienia strony, takie jak szerokość marginesów i orientacja papieru. Ustawienia strony są zapisywane razem z raportem, dzięki czemu wystarczy je określić tylko raz.

2.7. Moduły

Za pomocą makr programu Access lub kodu VBA (*Visual Basic for Application*) możliwe jest dodawanie do bazy danych nowych funkcji. Po utworzeniu makra lub procedury języka VBA wystarczy związać je z odpowiednim zdarzeniem, którego wystąpienie będzie uruchamiało makro lub procedurę. Tak naprawdę akcje makropoleceń to zestawy poleceń dostępnych w języku VBA, a konstruktor makropoleceń tylko je udostępnia, umożliwiając użytkownikowi dodanie kodu programu do obiektów bez wymagania znajomości języka VBA.

Procedury tworzone w języku VBA mogą stanowić zagrożenie dla bezpieczeństwa danych, mogą również uszkodzić pliki na komputerze. Dlatego należy włączać kod VBA tylko wtedy, gdy wiadomo, z jakiego źródła pochodzi baza danych. Natomiast gdy użytkownik tworzy bazę, która będzie używana przez inne osoby, powinien unikać dołączania do niej procedur.

W celu zapewnienia bezpieczeństwa bazy danych powinno się używać, jeżeli tylko to możliwe, makropoleceń, a procedury VBA stosować jedynie wtedy, gdy wykonanie operacji nie jest możliwe za pomocą akcji makropoleceń.

2.8. Ochrona danych

2.8.1. Centrum zaufania

Korzystanie z niektórych składników programu Access wiąże się z pewnym ryzykiem. Chodzi o **kwerendy funkcjonalne** (kwerendy, które wstawiają, usuwają lub zmieniają dane), **makra, wyrażenia** (funkcje, które zwracają pojedynczą wartość) oraz kod **VBA**. Aby zagwarantować większe bezpieczeństwo danych, program przy każdym otwarciu bazy danych wykonuje zestaw czynności sprawdzających zabezpieczenia.

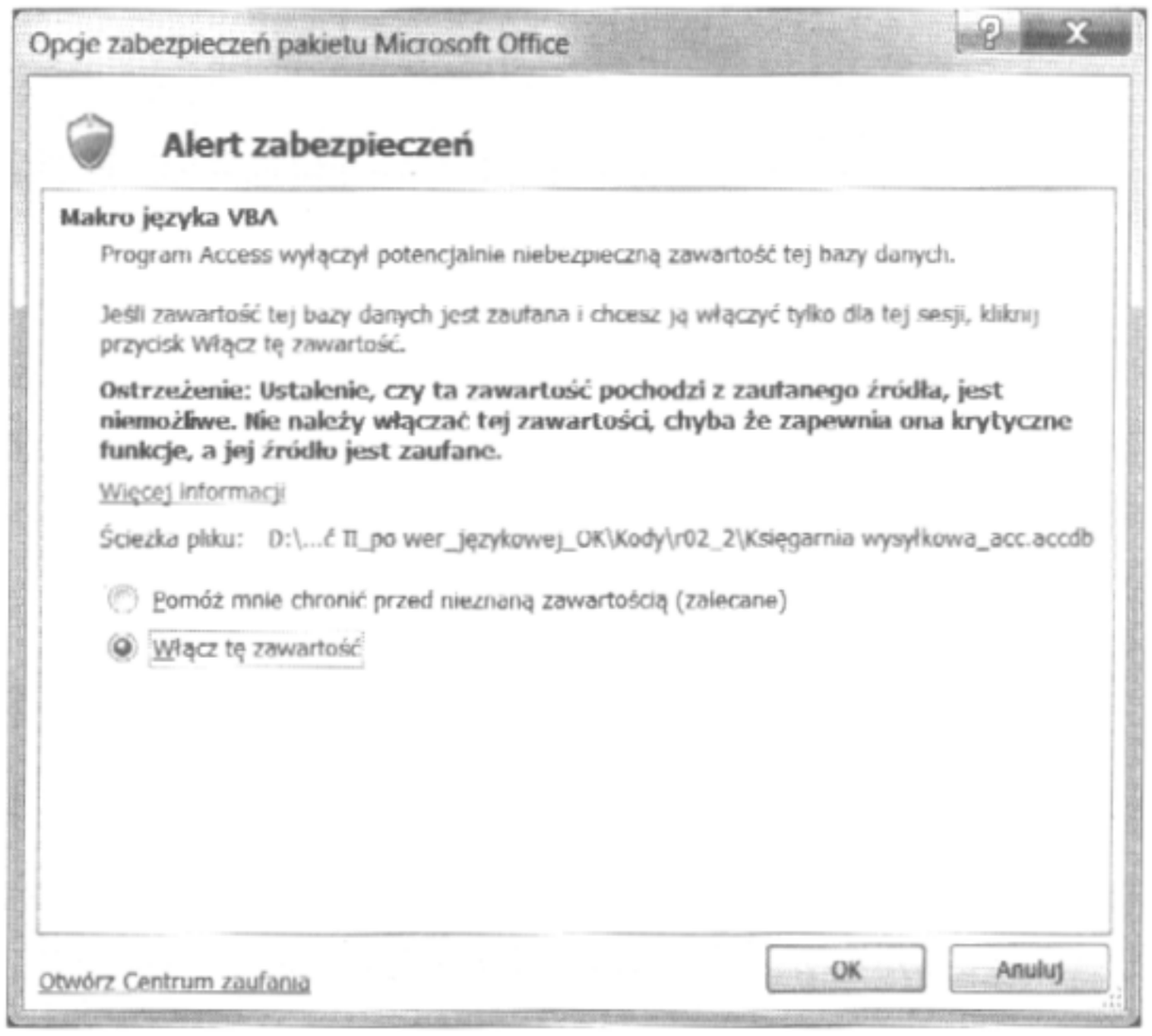
Po otwarciu pliku informacje o jego lokalizacji są przesyłane do **Centrum zaufania.** Centrum sprawdza te informacje. Po sprawdzeniu program Access wyłącza bazę danych lub otwiera ją jako w pełni funkcjonalną.

Jeżeli centrum wyłączy jakąkolwiek zawartość, to w trakcie otwierania bazy danych, jeśli nie znajduje się ona w zaufanej lokalizacji, pojawi się pasek komunikatów (rysunek 2.64).

Rysunek 2.64. Pasek komunikatów

Jeżeli wiadomo, że baza danych jest zaufana, można włączyć wyłączoną zawartość. W tym celu należy kliknąć przycisk *Opcje*, a następnie zaznaczyć w wyświetlonym oknie dialogowym opcję *Włącz tę zawartość* (rysunek 2.65).

Rysunek 2.65. Okno Alert zabezpieczeń



UWAGA

Określenie, że pliki bazy danych znajdują się w zaufanej lokalizacji, oznacza, że znajdują się w folderze plików lub udziałach sieciowych ustawionych jako bezpieczna lokalizacja.

Program Access włączy wyłączoną zawartość, a baza danych zostanie ponownie otwarta jako w pełni funkcjonalna. W przeciwnym razie wyłączone składniki nie będą działać.

Jeżeli pasek komunikatów jest niewidoczny, należy na karcie *Narzędzia bazy danych* w grupie *Pokazywanie/ukrywanie* zaznaczyć opcję *Pasek komunikatów*.

UWAGA

Wykonanie tych czynności włączy całą zawartość, w tym potencjalnie złośliwy kod, który może uszkodzić dane lub oprogramowanie komputera.

Możliwe jest także definiowanie opcji zabezpieczeń w oknie *Centrum zaufania*. W tym celu należy wybrać *Przycisk pakietu Office*, następnie przycisk *Opcje programu Access* i z prawej strony z menu opcję *Centrum zaufania*. Następnie trzeba kliknąć przycisk *Ustawienia Centrum zaufania*. W otwartym oknie można definiować zaawansowane opcje zabezpieczeń bazy danych, takie jak tworzenie zaufanych lokalizacji czy ocenianie bezpieczeństwa składników bazy danych.

W przypadku otwarcia bazy danych, która została utworzona we wcześniejszym formacie pliku (*mdb* lub *mde*) i nie jest podpisana ani zaufana, program Access domyślnie wyłącza każdą wykonywalną zawartość.

2.8.2. Zabezpieczenia na poziomie użytkownika

Program Access nie zapewnia zabezpieczeń na poziomie użytkownika. Niektóre składniki bazy mogą potencjalnie być niebezpieczne (kwerendy funkcjonalne, makra, kod VBA), więc *Centrum zaufania* przy każdym otwarciu bazy sprawdza jej stan zabezpieczeń. Jeśli baza danych zostanie oceniona jako niezaufana, cała wykonywalna zawartość bazy zostanie wyłączona. Po umieszczeniu bazy danych w zaufanej lokalizacji będzie ona traktowana jako zaufana.

Umieszczanie bazy danych w zaufanej lokalizacji

W celu zdefiniowania zaufanych lokalizacji dla baz danych należy wybrać *Przycisk pakietu Office*, później przycisk *Opcje programu Access* i w otwartym oknie dialogowym kliknąć położoną po prawej stronie opcję *Centrum zaufania* i przycisk *Ustawienia Centrum zaufania*. Następnie trzeba wybrać pozycję *Zaufane lokalizacje* i zapisać ścieżkę do jednej lub kilku lokalizacji. Można również kliknąć przycisk *Dodaj nową lokalizację* i w otwartym oknie po kliknięciu przycisku *Przeglądaj* wybrać odpowiednią lokalizację.

Po zdefiniowaniu zaufanych lokalizacji trzeba umieścić bazę danych w jednej z tych lokalizacji.

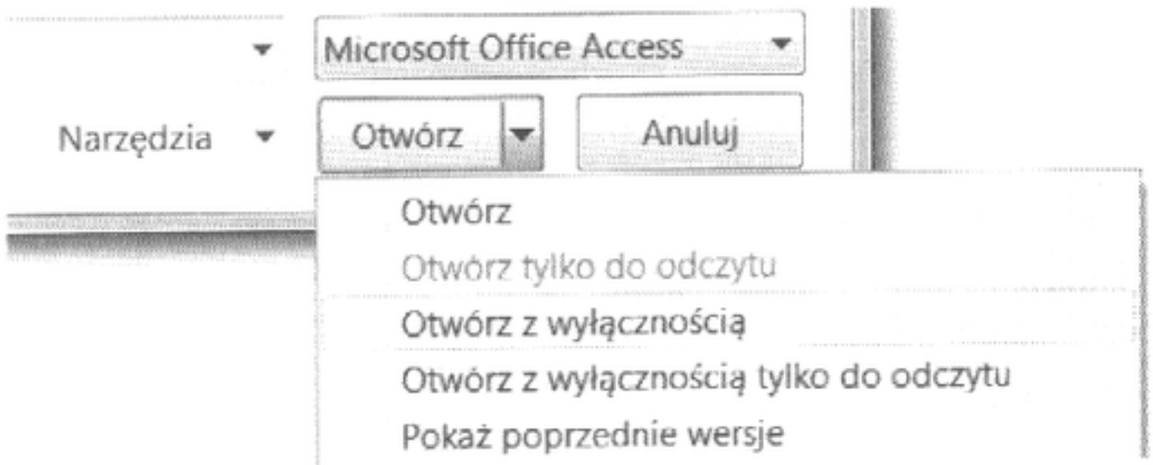
Jeżeli baza danych została umieszczona w zaufanej lokalizacji, to program Access przy jej otwieraniu uruchamia wszystkie makra, kwerendy funkcjonalne, bezpieczne wyrażenia oraz kody VBA.

Szyfrowanie bazy danych

Bazę danych można zaszyfrować przy użyciu hasła. Baza danych staje się niedostępna, jeśli nie podano hasła do niej.

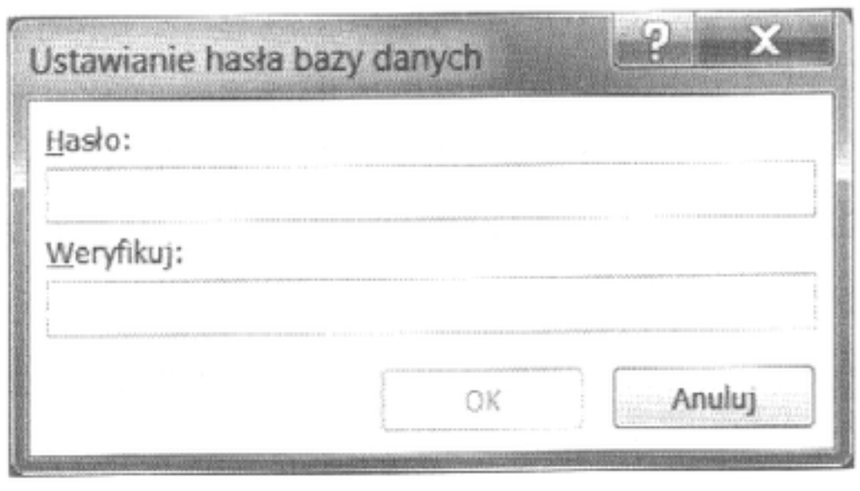
W celu zaszyfrowania bazy danych należy otworzyć ją w trybie wyłączności. W tym celu trzeba wybrać *Przycisk pakietu Office*, a następnie polecenie *Otwórz*. W otwartym oknie należy odnaleźć plik z bazą danych, następnie kliknąć strzałkę obok przycisku *Otwórz* i wybrać polecenie *Otwórz z wyłącznością* (rysunek 2.66).

Rysunek 2.66. Wybór trybu wyłączności w otwieranej bazie danych



Po otwarciu bazy danych na karcie *Narzędzia bazy danych* w grupie *Narzędzia bazy danych* należy wybrać ikonę *Szyfruj przy użyciu hasła*. Zostanie otwarte okno, w którym trzeba wpisać hasło zabezpieczające bazę danych (rysunek 2.67). Zapomnianego hasła nie można odtworzyć.

Rysunek 2.67. Definiowanie hasła do bazy danych



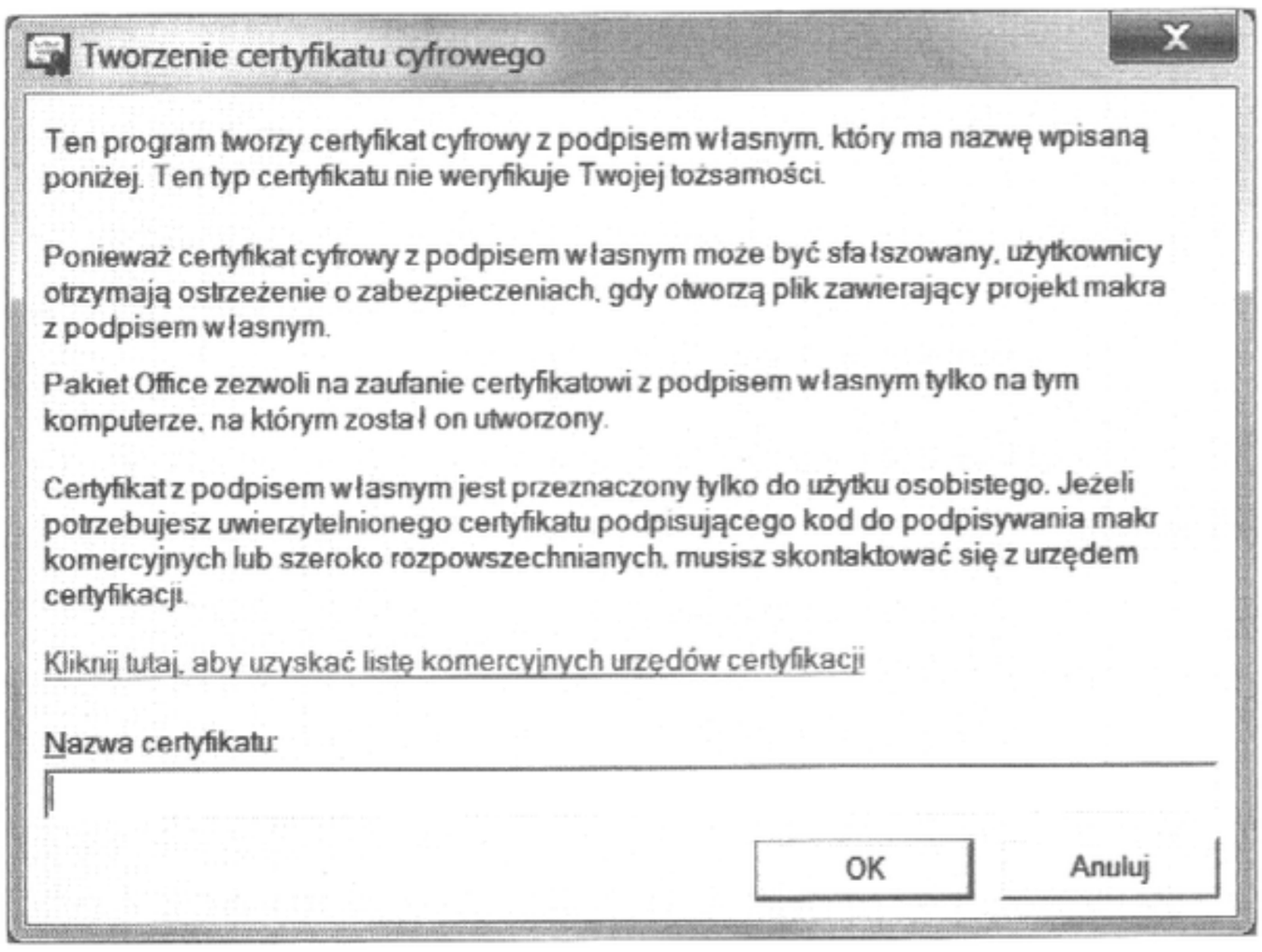
Powinno się używać silnych haseł, zawierających małe i duże litery, cyfry i inne znaki.

W celu odszyfrowania bazy danych trzeba ją otworzyć w zwykły sposób, podając hasło. Następnie na karcie *Narzędzia bazy danych* w grupie *Narzędzia bazy danych* należy wybrać ikonę *Odszyfruj bazę danych*. Zostanie otwarte okno, w którym należy wpisać hasło i nacisnąć przycisk *OK*.

Rozpowszechnianie bazy danych

Jeżeli baza danych będzie udostępniana innym użytkownikom, można dla niej zastosować dostępne w programie Access narzędzie do pakowania i tworzenia podpisu cyfrowego. Podpis cyfrowy można stosować tylko do plików zapisanych w formacie *accdb* i tylko w ramach pakowania bazy danych. W wyniku takiego działania powstanie plik z rozszerzeniem *accdc*, który zostanie umieszczony w lokalizacji wskazanej przez użytkownika.

Aby można było skorzystać z tego narzędzia, musi być dostępny przynajmniej jeden certyfikat zabezpieczeń. W celu uzyskania takiego certyfikatu należy kliknąć przycisk *Start*, wybrać *Wszystkie programy*, a następnie *Microsoft Office* i *Narzędzia Microsoft Office* oraz *Certyfikat cyfrowy dla projektów VBA*. Zostanie wyświetlone okno *Tworzenie certyfikatu cyfrowego*. W polu *Nazwa certyfikatu* trzeba wpisać nazwę nowego certyfikatu i kliknąć *OK* (rysunek 2.68).



Rysunek 2.68. Tworzenie certyfikatu cyfrowego dla bazy danych

Po utworzeniu certyfikatu zabezpieczeń można przystąpić do tworzenia podpisu cyfrowego. W tym celu należy otworzyć bazę danych, którą zamierzamy spakować i podpisać. Następnie trzeba wybrać *Przycisk pakietu Office*, a po nim polecenia *Publikuj* i *Spakuj i podpisz*. Po wybraniu certyfikatu cyfrowego zostanie otwarte okno, w którym należy wybrać lokalizację spakowanego i podpisanego pakietu i podać jego nazwę. Po kliknięciu przycisku *Utwórz* w wybranej lokalizacji zostanie utworzony plik z rozszerzeniem *accdc*.

Podsumowanie

Powyższe metody pomagają w zabezpieczaniu baz danych, ale najpewniejszym sposobem ochrony danych jest przechowywanie tabel na serwerze, a formularzy i raportów na komputerach lokalnych lub na udziałach sieciowych.

2.9. Język SQL w programie Access

Do konstruowania kwerend w programie Access można używać języka **SQL** (ang. *Structured Query Language*) lub **QBE** (ang. *Query By Example*).

Język SQL jest zbliżony do języka formalnego. Gdy tworzymy kwerendę, korzystając z szablonu QBE, program Access w tle konstruuje odpowiadające jej wyrażenie w języku SQL.

SQL formalnie jest podobny do języków programowania wysokiego poziomu, ale jest od nich o wiele prostszy. Edytowanie poleceń w języku SQL wymaga przejścia do trybu SQL. Uruchomienie zdefiniowanej w tym języku kwerendy odbywa się tak jak uruchomienie innych kwerend zdefiniowanych za pomocą szablonu, a wynik wyświetlany jest w postaci arkusza danych. Dla kwerend tworzonych w języku SQL program Access tworzy w tle szablon QBE.

2.9.1. Kwerenda wybierająca w języku SQL

Ogólna postać kwerendy wybierającej, dla której źródłem danych jest jedna tabela, ma postać:

```
[PARAMETERS
  parametr_1 typ_danych [,
  parametr_2 typ_danych [, ...]];]
SELECT [ALL/DISTINCT/DISTINCTROW/TOP n [PERCENT]]
  {*/[nazwa_tabeli.]pole_1 [AS nazwa_1][,
  [nazwa_tabeli.]pole_2 [AS nazwa_2][, ...]]}
FROM
  Nazwa_tabeli
  [AS nowa_nazwa]
  [IN baza_danych]
[WHERE
  wyrażenie]
[GROUP BY
  lista_pól
  [HAVING
    wyrażenie]]
[ORDER BY
  kryterium1 [,
  kryterium2 [, ...]]];
```

Wyrażenie można zapisać w jednej linii, a jego strukturalność została wprowadzona w celu zwiększenia przejrzystości zapisu. Wyrażenie musi kończyć się średnikiem.

W wyrażeniu są stosowane następujące konwencje:

- Słowa kluczowe pisane są dużymi literami.
- Argumenty wyrażenia pisane są małymi literami.
- W nawiasach kwadratowych umieszczone są opcjonalne elementy wyrażeń.
- Pionowa kreska pełni funkcję słowa „lub" (wybór jednego elementu z listy).
- Nawiasy klamrowe używane są, gdy jeden z elementów musi wystąpić w wyrażeniu.

2.9.2. Definiowanie kwerendy wybierającej

Jedynie słowa kluczowe `SELECT` i `FROM` wraz z odpowiednimi argumentami muszą wystąpić w definicji kwerendy wybierającej. Argumentami dla słowa `SELECT` będą nazwy pól, a dla słowa `FROM` będzie to nazwa tabeli źródłowej kwerendy. Słowo `SELECT`

występuje na początku każdego wyrażenia SQL, a zamiast nazwy pola można użyć wyrażenia operującego na wartościach tego pola.

Przykład 2.31

```
SELECT
Klient.Imię, Klient.Nazwisko, Year(Date())-Year([Data urodzenia])
FROM Klient;
```

Po dodaniu do tabeli *Klient* pola *Data urodzenia* wynikiem wykonania wyrażenia będzie arkusz danych z polami *Imię* i *Nazwisko* z tabeli *Klient* oraz z polem obliczeniowym, w którym zostanie obliczony wiek każdego klienta. Argumentem słowa `SELECT` może być gwiazdka (*), która oznacza wybór wszystkich pól tabeli źródłowej. Nazwy pól mogą być poprzedzone nazwą tabeli źródłowej i kropką. Jest to konieczne przy wyborze pól z kilku tabel źródłowych. Kolumny powstającego arkusza danych będą nazywały się tak samo jak pola tabeli źródłowej.

Parametr AS

Parametr ten pozwala ustawić nazwę kolumny w arkuszu danych. Może służyć również do nadania nazwy tabeli.

Przykład 2.32

```
SELECT
Imię, Nazwisko, Year(Date())-Year([Data urodzenia]) AS Wiek
FROM Klient;
```

Parametr WHERE

Parametr ten stosujemy, gdy chcemy wyświetlić w kwerendzie tylko rekordy spełniające określony warunek. Po słowie `WHERE` musi wystąpić warunek w postaci wyrażenia. W arkuszu danych znajdą się tylko te rekordy, które spełniają warunek.

Przykład 2.33

```
SELECT
  Imię, Nazwisko, Year(Date())-Year([Data urodzenia]) AS Wiek
  FROM Klient
  WHERE [Miejscowość]="Toruń" ;
```

W arkuszu danych pojawią się tylko dane klientów z Torunia.

Parametr ORDER BY

Parametr ten stosujemy, gdy chcemy uporządkować dane. Po słowach `ORDER BY` należy wpisać nazwę pola lub wyrażenie, według którego nastąpi porządkowanie danych. Dodatkowo można określić sposób porządkowania za pomocą słów: `ASC` — rosnąco,

DESC — malejąco. Program Access domyślnie sortuje dane rosnąco, więc słowo ASC może zostać pominięte.

Przykład 2.34

```
SELECT
  Tmię, Nazwisko, Year(Date())-Year([Data urodzenia]) AS Wiek
FROM Klient
ORDER BY Year(Date())-Year([Data urodzenia]) DESC, Nazwisko ;
```

Wynikiem wykonania wyrażenia będzie arkusz danych zawierający listę wszystkich klientów uporządkowaną malejąco według wieku. Nazwiska klientów w tym samym wieku zostaną posortowane alfabetycznie.

PARAMETERS

Parametr ten stosujemy, gdy chcemy utworzyć kwerendę z parametrem. Słowo PARAMETERS umieszczamy na początku całego wyrażenia, jeszcze przed słowem SELECT. Po nim deklarujemy nazwy parametrów wraz z typem danych. Typy danych dla SQL zostały omówione w dalszej części rozdziału.

Przykład 2.35

```
PARAMETERS
[Podaj nazwisko klienta:] Text ;
SELECT *
FROM Klient
WHERE Nazwisko = [Podaj nazwisko klienta:] ;
```

Uruchomienie kwerendy spowoduje wyświetlenie pola dialogowego z tekstem, który jest nazwą parametru. Po wprowadzeniu parametru, czyli nazwiska klienta, zostanie wyświetlony arkusz danych zawierający dane klienta o podanym nazwisku. Nazwisko jest parametrem kwerendy.

Jeżeli chcemy wyświetlić dane jednego klienta i uniknąć wyświetlania danych wszystkich klientów noszących to samo nazwisko, należy zadeklarować drugi parametr, na przykład imię.

Przykład 2.36

```
PARAMETERS
[Podaj nazwisko klienta:] Text ,
[Podaj imię klienta:]  Text ;
SELECT *
FROM Klient
WHERE Nazwisko = [Podaj nazwisko klienta:] AND Imię= [Podaj imię klienta:];
```

Parametry ALL, DISTINCT, DISTINCTROW, TOP

Są to predykaty używane do określenia, czy wyświetlane wartości mogą się powtórzyć i czy należy wyeliminować wiersze, w których wartości się powtarzają.

`ALL` — używany domyślnie.

`DISTINCT` — używany, gdy chcemy ograniczyć wystąpienia takich samych wartości w kolejnych rekordach. Odpowiada atrybutowi *WARTOŚCI UNIKATOWE*.

`DISTINCTROW` — używany wyłącznie w kwerendzie opartej na kilku tabelach. Eliminuje w zestawieniu identyczne rekordy. Odpowiada atrybutowi *REKORDY UNIKATOWE*.

`TOP n` — pozwala wybrać *n* pierwszych rekordów spełniających warunki określone w kwerendzie. Odpowiada atrybutowi *NAJWYŻSZE WARTOŚCI*. Liczbę wybranych rekordów można określać procentowo, używając predykatu `TOP` ze słowem `PERCENT`.

Przykład 2.37

```
SELECT
  Miejscowość
FROM Klient
WHERE [Miejscowość]<>"Poznań" ;
```

Uruchomienie tej kwerendy spowoduje wyświetlenie informacji, z jakich miejscowości, poza Poznaniem, pochodzą klienci. Nazwa miejscowości pojawi się na liście tyle razy, ilu klientów jest z danej miejscowości. Aby ograniczyć się do pojedynczych wystąpień nazw miejscowości, należy dodać predykat `DISTINCT`.

Przykład 2.38

```
SELECT DISTINCT
  Miejscowość
FROM Klient
WHERE [Miejscowość]<>"Poznań" ;
```

Uruchomienie kolejnej kwerendy spowoduje wyświetlenie danych trzech najstarszych klientów.

Przykład 2.39

```
SELECT TOP 3 *
FROM Klient
ORDER BY [Data urodzenia] ;
```

GROUP BY ... HAVING

Polecenie to służy do grupowania wybranych w kwerendzie rekordów według identycznych wartości pól, których nazwy zostały wymienione w tym poleceniu. W tym przypadku można w wyrażeniu SELECT używać funkcji agregujących.

Przykład 2.40

```
SELECT
  Miejscowość , Count(Nazwisko) AS [Liczba klientów]
FROM Klient
GROUP BY Miejscowość
ORDER BY Miejscowość ;
```

Wynikiem wykonania wyrażenia będzie pogrupowanie klientów według wartości pola *Miejscowość*. W każdej grupie liczone będą rekordy i wyświetlony zostanie rezultat obliczeń.

Aby wyeliminować rekordy po podziale na grupy, należy dodatkowo zastosować warunek HAVING.

Przykład 2.41

```
PARAMETERS
[Z jakiej miejscowości:] Text  ;
SELECT
  Miejscowość , Count(Nazwisko) AS [Liczba klientów]
FROM Klient
GROUP BY Miejscowość
HAVING Miejscowość = [Z jakiej miejscowości:];
```

Jeśli kwerenda zawiera polecenie GROUP BY, wymagane jest użycie każdego z pól wymienionych w wyrażeniu SELECT jako kryterium grupowania lub jako argumentu funkcji agregującej.

IN

Słowo to może wystąpić po słowie FROM i służy do określenia bazy danych, z której pochodzi tabela. Argumentem słowa IN może być ścieżka dostępu do pliku bazy danych.

2.9.3. Typy danych w języku SQL

W tabeli 2.6 zostały przedstawione typy danych stosowane w języku SQL.

Tabela 2.6. Typy danych w SQL

Typ danych	Rozmiar	Synonimy	Opis
BINARY	1 bajt na znak	VARBINARY	Można przechowywać dowolny rodzaj danych. Sposób prezentacji danych zależy od sposobu ich umieszczania w polu.
BIT	1 bajt	BOOLEAN, LOGICAL1	Wartości typu logicznego.
BYTE	1 bajt	INTEGER1, TINYINT	Liczba całkowita od 0 do 255.
COURRENCY	8 bajtów	MONEY	Liczba całkowita mieszcząca się na 8 bajtach.
DATETIME	8 bajtów	DATE, TIME, TIMESTAMP	Wartość daty lub godziny dla lat od 100 do 9999.
SINGLE	4 bajty	REAL	Liczby zmiennoprzecinkowe pojedynczej precyzji.
DOUBLE	8 bajtów	FLOAT, NUMBER, NUMERIC	Liczby zmiennoprzecinkowe podwójnej precyzji.
SHORT	2 bajty	SMALLINT	Liczba całkowita mieszcząca się na 2 bajtach.
LONG	4 bajty	INTEGER, INT	Liczba całkowita długa mieszcząca się na 4 bajtach.
TEXT	1 bajt na znak	CHAR, CHARACTER, STRING	Tekst zawierający od 0 do 255 znaków.

2.9.4. Definiowanie połączeń między tabelami

Podczas definiowania kwerendy w wyrażeniu FROM można używać wielu tabel. Aby dane z tabel zostały prawidłowo powiązane, należy zdefiniować połączenia między tabelami. Kwerendy oparte na kilku tabelach różnią się od kwerend opartych na jednej tabeli tylko sposobem określenia źródła danych, więc zmiany ogólnej postaci kwerendy dotyczą jedynie wyrażenia FROM.

```
FROM
  Tabela1
{ INNER/LEFT/RIGHT } JOIN ( tabela2
{ INNER/LEFT/RIGHT } JOIN ( tabela3
[ { INNER/LEFT/RIGHT } JOIN [(] tabelax
[ { INNER/LEFT/RIGHT } JOIN … ) ]
ON tabela3.pole3 = tabelax.polex) ]
ON tabela2.pole2 = tabela3.pole3)
ON tabela1.pole1 = tabela2.pole2 ;
```

W wyrażeniu uwzględniono możliwość powiązania i wykorzystania w kwerendzie wielu tabel źródłowych. Oprócz określenia tabel źródłowych wyrażenie musi zawierać nazwy odpowiadających sobie pól z każdej tabeli. Oto postać wyrażenia dla dwóch tabel źródłowych:

```
FROM
Tabela1
{ INNER/LEFT/RIGHT } JOIN tabela2
ON tabela1.pole1 = tabela2.pole2 ;
```

Przed słowem kluczowym `JOIN` (połączenie) występuje określenie rodzaju połączenia:

- `INNER JOIN` — połączenie zawężające,
- `LEFT JOIN` — lewe połączenie rozszerzające,
- `RIGHT JOIN` — prawe połączenie rozszerzające.

Przykład 2.42

```
SELECT
[Klient].[Imię] , [Klient].[Nazwisko],
[Książki].[Tytuł]
FROM Klient
INNER JOIN [Zamówienia]
ON [Klient].[Id klienta] = [Zamówienia].[Id klienta]
INNER JOIN [Książki]
ON [Zamówienia].[Id_książki] = [Książki].[Id książki]
ORDER BY [Klient].[Nazwisko] ;
```

Wynikiem wykonania kwerendy będzie arkusz danych zawierający listę wszystkich klientów wraz z tytułami książek, które zamówili; lista będzie uporządkowana alfabetycznie według nazwisk.

Przykład 2.43

Przeanalizuj zamówienia książek składane przez klientów. Zestawienie powinno zawierać listę nazwisk wszystkich klientów wraz z informacją o dacie złożenia zamówienia i o dacie wysyłki. Na liście muszą pojawić się też nazwiska klientów, którzy nie złożyli żadnego zamówienia, ale są zarejestrowani w bazie. W kwerendzie zdefiniuj połączenie między tabelami *Klient* i *Zamówienia*.

```
SELECT
 [Klient].[Nazwisko], [Klient].[Imię], [Zamówienia].[Data złożenia zamó-
wienia], [Zamówienia].[Data wysłania]
FROM [Klient]
LEFT JOIN [Zamówienia]
ON [Klient].[Id_klienta] = [Zamówienia].[Id_klienta]
ORDER BY [Klient].[Nazwisko] ;
```

Zostało zdefiniowane połączenie rozszerzające lewe, zatem z tabeli *Klient* zostaną wybrani wszyscy klienci. To, czy należy zastosować połączenie rozszerzające lewe, czy prawe, zależy od kolejności zapisywania nazw tabel w definiowanym połączeniu.

PYTANIA KONTROLNE

1. Jakie cechy powinno posiadać pole tabeli pełniące rolę klucza podstawowego?
2. Jaki typ pola najlepiej wybrać do wpisania numeru telefonu? Uzasadnij swoją odpowiedź.
3. Kiedy powinien być stosowany typ pola *Liczba, pojedyncza precyzja*?
4. W jakim celu ustawiane są właściwości pola?
5. Wymień warunki niezbędne do poprawnego zaprojektowania relacji między tabelami.
6. Jaką rolę w bazie danych spełniają więzy integralności?
7. W jakim celu stosuje się indeksowanie pól tabeli?
8. Czym jest występujący w wyrażeniu identyfikator?
9. Jak w wyrażeniu zapisywane są wartości typu *data*, *liczba*, *tekst*?
10. Wymień operatory specjalne występujące w programie Access.
11. Jaką rolę w bazie danych spełniają kwerendy wybierające?
12. Czym różni się działanie połączenia zawężającego od połączenia rozszerzającego?
13. Jakie działania mogą być realizowane za pomocą kwerend funkcjonalnych?
14. Jaką rolę w bazie danych spełniają formularze?

ZADANIE

W programie Access utwórz zaprojektowaną w rozdziale 1. bazę danych *Moja_szkoła*. Utwórz tabele, określ typy pól i ich rozmiar lub format. Utwórz relacje między tabelami. Zbuduj kwerendy określające sposób wybierania i przetwarzania danych zgromadzonych w tabelach. Utwórz formularze przedstawiające w postaci graficznej podstawowe informacje przechowywane w tabelach oraz informacje dostępne dzięki zaprojektowanym wcześniej kwerendom. Zdefiniuj przeznaczenie tych formularzy. Utwórz raporty, które będą prezentowały w formie podsumowań wyniki uczniów.

3

Systemy baz danych

3.1. Wprowadzenie

Do obsługi baz danych tworzone są złożone systemy zawierające zbiory gotowych narzędzi zapewniających dostęp do danych. Umożliwiają one manipulowanie danymi zgromadzonymi w systemach komputerowych i aktualizowanie tych danych. Są to systemy zarządzania bazą danych, w skrócie nazywane SZBD (ang. *Database Management Systems*).

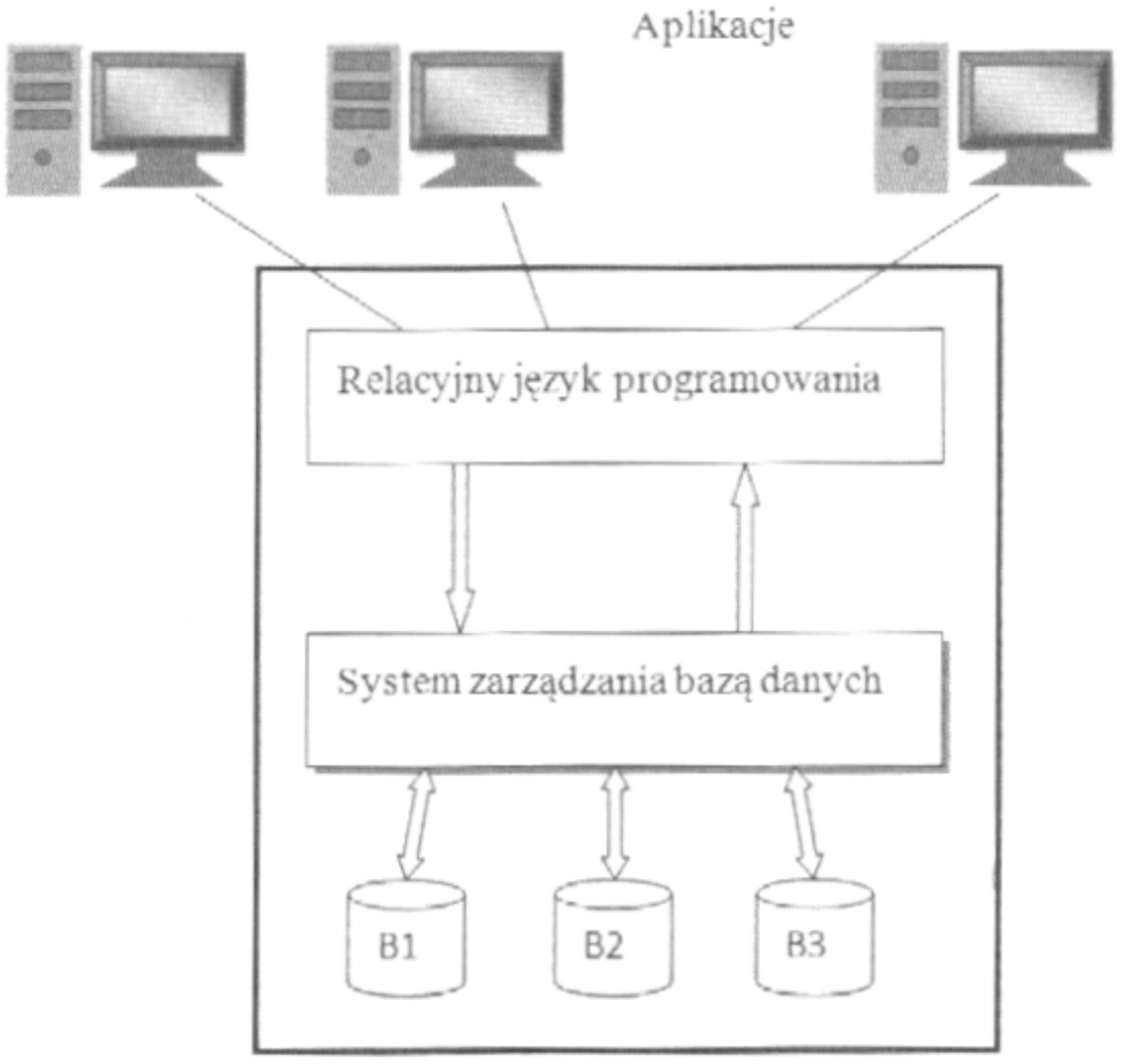
Do najważniejszych cech SZBD można zaliczyć:

- operowanie na dużych i bardzo dużych zbiorach danych,
- zarządzanie złożonymi strukturami.

3.2. Architektura systemu baz danych

System zarządzania bazą danych wraz z bazami danych i językiem komunikowania się tworzą system baz danych (rysunek 3.1).

Rysunek 3.1.
Architektura systemu baz danych



Interakcja programu użytkowego (aplikacji) z bazą danych odbywa się najczęściej za pomocą języka SQL. Jest to jedyny sposób komunikowania się aplikacji z bazą danych.

Język SQL jest narzędziem dostępu do bazy danych stosowanym głównie przez projektantów aplikacji, projektantów baz danych i administratorów baz danych. Użytkownicy komunikują się z bazą danych za pomocą aplikacji i dopiero aplikacje komunikują się z bazą danych za pomocą poleceń SQL.

W praktyce stosuje się dwa sposoby komunikacji z bazą danych:

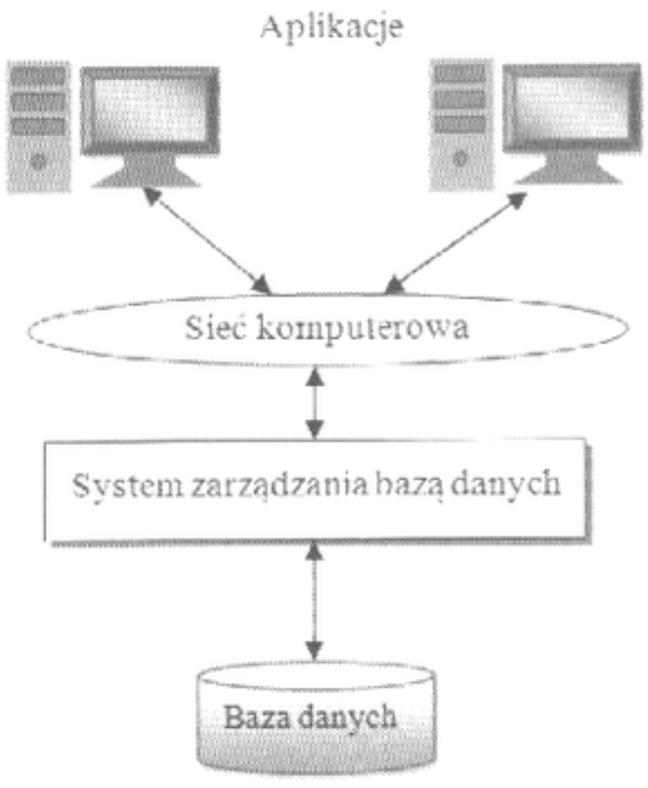
- architektura klient-serwer,
- architektura 3-warstwowa.

Architektura klient-serwer

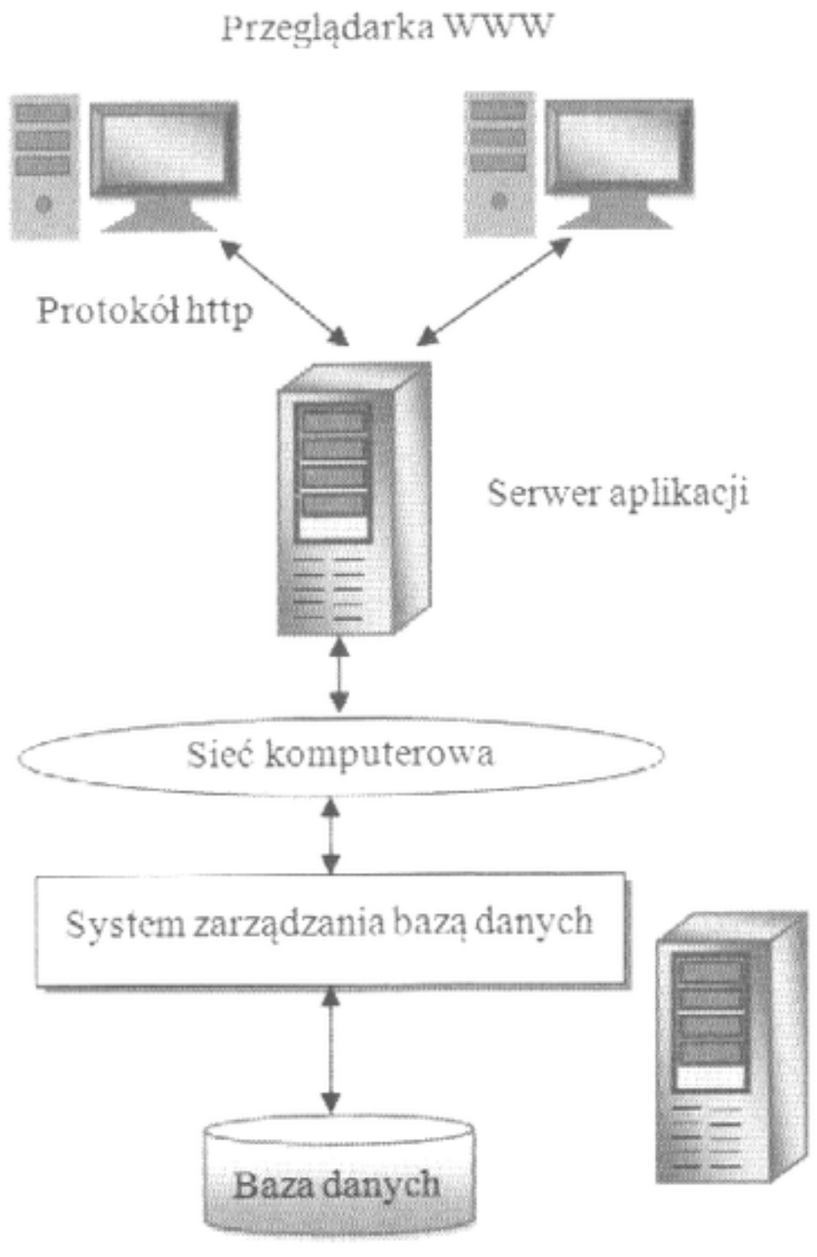
W architekturze klient-serwer aplikacje zainstalowane na stacjach użytkowników komunikują się z bazą danych, wykorzystując sieciowe oprogramowanie dedykowane do komunikacji z systemem zarządzania bazą danych (rysunek 3.2).

Architektura 3-warstwowa

W architekturze 3-warstwowej pomiędzy użytkownikami a serwerem bazy danych znajduje się tzw. serwer aplikacji, który udostępnia umieszczone na nim aplikacje. Jest to architektura typowa dla aplikacji WWW. Aplikacje są udostępniane przez serwer aplikacji w postaci stron internetowych. Użytkownik komunikuje się z bazą danych przez przeglądarkę stron WWW. W odpowiedzi na polecenia użytkownika serwer aplikacji wysyła odpowiednie żądania do systemu zarządzania bazą danych, który wykonuje polecenia i przesyła ich wyniki do serwera aplikacji. Serwer aplikacji przesyła te wyniki do aplikacji użytkownika (rysunek 3.3).



Rysunek 3.2. Architektura komunikacyjna klient-serwer



Rysunek 3.3. Architektura komunikacyjna 3-warstwowa

3.3. Baza danych

Baza danych to zbiór danych z określonej dziedziny posiadający ściśle zdefiniowaną wewnętrzną strukturę. Baza danych powinna charakteryzować się następującymi cechami:

- trwałość danych,
- integralność danych,
- bezpieczeństwo danych,
- współdzielenie danych,
- abstrakcja danych,
- niezależność danych,
- integracja danych.

Trwałość

Trwałość danych zapisanych w bazie jest podstawową cechą baz danych. Trwałość danych oznacza, że dane zostały zapisane w bazie w sposób nieulotny. Wszystkie współczesne systemy baz danych muszą spełniać ten wymóg. Trwałość danych jest niezależna od działania aplikacji oraz od platformy sprzętowej i programowej. Dane gromadzone w bazie danych są przechowywane w pamięci zewnętrznej (dyskowej, optycznej, taśmowej). Powinny one być przechowywane w pamięci tak długo, jak długo wymagają tego użytkownicy systemu baz danych.

Integralność, czyli poprawność danych

Integralność (spójność) danych oznacza, że dane muszą:

- wiernie odzwierciedlać dane rzeczywiste,
- spełniać ograniczenia nałożone przez użytkowników,
- wykazywać brak anomalii wynikających ze współbieżnego dostępu do danych.

Integralność bazy danych to także odporność na błędy i awarie wynikające z zawodności sprzętu i oprogramowania oraz odporność na błędy użytkowników.

Wyróżniamy dwa rodzaje integralności danych:

- semantyczna — oznacza zgodność danych z rzeczywistością, czyli poprawność odwzorowania rzeczywistości;
- bazowa — oznacza poprawność procesów zachodzących w bazie.

Na poziomie struktur bazy danych można wyróżnić następujące rodzaje integralności:

- referencyjna — oznacza, że każdej wartości klucza obcego odpowiada dokładnie jedna wartość klucza podstawowego;
- encji — oznacza, że każdy atrybut encji musi mieć określoną wartość;
- atrybutu — oznacza, że wartość każdego atrybutu należy do jego dziedziny.

Bezpieczeństwo danych

Bezpieczeństwo danych oznacza, że dostęp do bazy danych mają tylko jej użytkownicy identyfikowani unikalną nazwą (loginem) i hasłem. Ponadto każdy użytkownik posiada określone uprawnienia w bazie danych. Bezpieczeństwo danych oznacza również, że baza danych jest zabezpieczona przed awarią sprzętu lub oprogramowania.

Współdzielenie danych

Współdzielenie danych oznacza, że istnieje możliwość równoczesnej pracy wielu użytkowników z tą samą bazą danych. Użytkownicy mogą też jednocześnie pracować z tym samym zbiorem danych. Skutkiem takiej pracy mogą być konflikty w dostępie do danych, gdy jeden użytkownik modyfikuje zbiór danych, a drugi próbuje ten sam zbiór odczytać lub zmodyfikować w inny sposób. W bazie danych muszą istnieć mechanizmy zapewniające poprawne rozwiązanie takich konfliktów.

Abstrakcja danych

Baza danych zawiera pewien model rzeczywistości. Ale przechowywane są w niej tylko niektóre dane o obiektach. Powinny one opisywać wyłącznie istotne aspekty obiektów świata rzeczywistego. Baza danych to abstrakcyjny model pewnego wycinka rzeczywistości, a jej struktura powinna w poprawny sposób odzwierciedlać obiekty świata rzeczywistego i powiązania pomiędzy tymi obiektami.

Niezależność danych

Niezależność danych polega na oddzieleniu danych od aplikacji, które używają tych danych. Niezależność danych może być rozpatrywana w dwóch aspektach:

- **Logiczna niezależność danych** — niezależność danych widzianych przez użytkownika (przez jego aplikacje) od logicznej struktury bazy danych (schematu pojęciowego) oraz zmian tej struktury w czasie.
- **Fizyczna niezależność danych** — niezależność logicznej struktury bazy danych (schematu pojęciowego) od sposobu przechowywania danych w pamięci zewnętrznej (na nośnikach zewnętrznych).

We współczesnych bazach danych niezależność danych jest osiągana tylko częściowo.

Integracja danych

Integracja danych polega na zapewnieniu współpracy danych przechowywanych w różnych miejscach i obsługiwanych przez różne systemy. Często bazy danych fizycznie są umieszczane na różnych komputerach połączonych ze sobą w taki sposób, że użytkownik nie wie, iż dane, z którymi pracuje, pochodzą z różnych baz i komputerów. Zmiany zawartości bazy na jednym komputerze powinny być uwzględniane również na innych komputerach. Tak funkcjonujące bazy nazywamy rozproszonymi bazami danych. Bazy danych powinny być zaprojektowane tak, aby dane były zawsze dostępne, niezależnie od tego, gdzie i w jakiej postaci zostały zapisane i jak są przechowywane. Serwery

bazodanowe powinny zapewniać integrację rozproszonych danych znajdujących się na wielu komputerach. Mechanizmy integracyjne dostępne na nich powinny działać w czasie rzeczywistym i współpracować z różnorodnymi bazami danych.

3.4. System zarządzania bazą danych

System zarządzania bazą danych (ang. *Database Management System* — DBMS) decyduje o sposobie pobierania i przechowywania danych w relacyjnych bazach danych. Może on udostępniać bazę lokalnie na jednym komputerze (*bazy jednostanowiskowe* — rozdział 2.) lub może być serwerem bazy danych udostępniającym zasoby komputerom połączonym w sieć (*bazy typu klient-serwer*).

System zarządzania bazą danych musi posiadać mechanizmy, które pozwalają administrować zbiorami danych umieszczonymi w bazie, zapewniają bezpieczeństwo i integralność danych, umożliwiają dostęp do danych za pomocą języka zapytań (najczęściej SQL), zapewniają wielodostępność danych (na przykład przez transakcje) oraz pozwalają na autoryzację dostępu do danych.

Często takie systemy posiadają narzędzia do przechowywania i przetwarzania danych multimedialnych, narzędzia do konwersji danych w celu współpracy z innymi systemami baz danych, graficzne środowiska do tworzenia aplikacji oraz narzędzia do udostępniania bazy danych w internecie.

W systemach zarządzania bazą danych można wyodrębnić dwa elementy:

- **serwer** — przechowuje dane, umożliwia ich pobieranie i aktualizowanie oraz zapewnia ich integralność i bezpieczeństwo;
- **oprogramowanie pośredniczące** (ang. *middleware*) — realizuje komunikację między użytkownikiem a serwerem. Wyposażone jest w mechanizmy pozwalające wykorzystywać pobierane dane, na przykład w narzędzia do tworzenia i obsługi formularzy oraz raportów.

Systemy zarządzania bazą danych zwykle działają w trybie *klient-serwer*, czyli baza umieszczona na serwerze jest udostępniana klientom poprzez oprogramowanie pośredniczące (systemy bazodanowe). Przykładami takich systemów są: MySQL, MS SQL Server, Oracle, PostgreSQL, DB2.

Istnieją również systemy, które nie uwzględniają podziału typu klient-serwer. Przykładem takiego systemu jest poznany wcześniej Microsoft Access.

System zarządzania bazą danych (serwer bazodanowy), który należy do architektury klient-serwer, składa się z dwóch części, które ze sobą współpracują. Jedna część, działająca na serwerze, odpowiedzialna jest za wydajność, bezpieczeństwo, kopie zapasowe itp. Druga, działająca po stronie klienta, zapewnia interfejs użytkownika, dzięki któremu możliwe jest przeprowadzanie operacji na bazie danych. Najczęściej serwer bazodanowy komunikuje się z bazą danych za pomocą języka SQL.

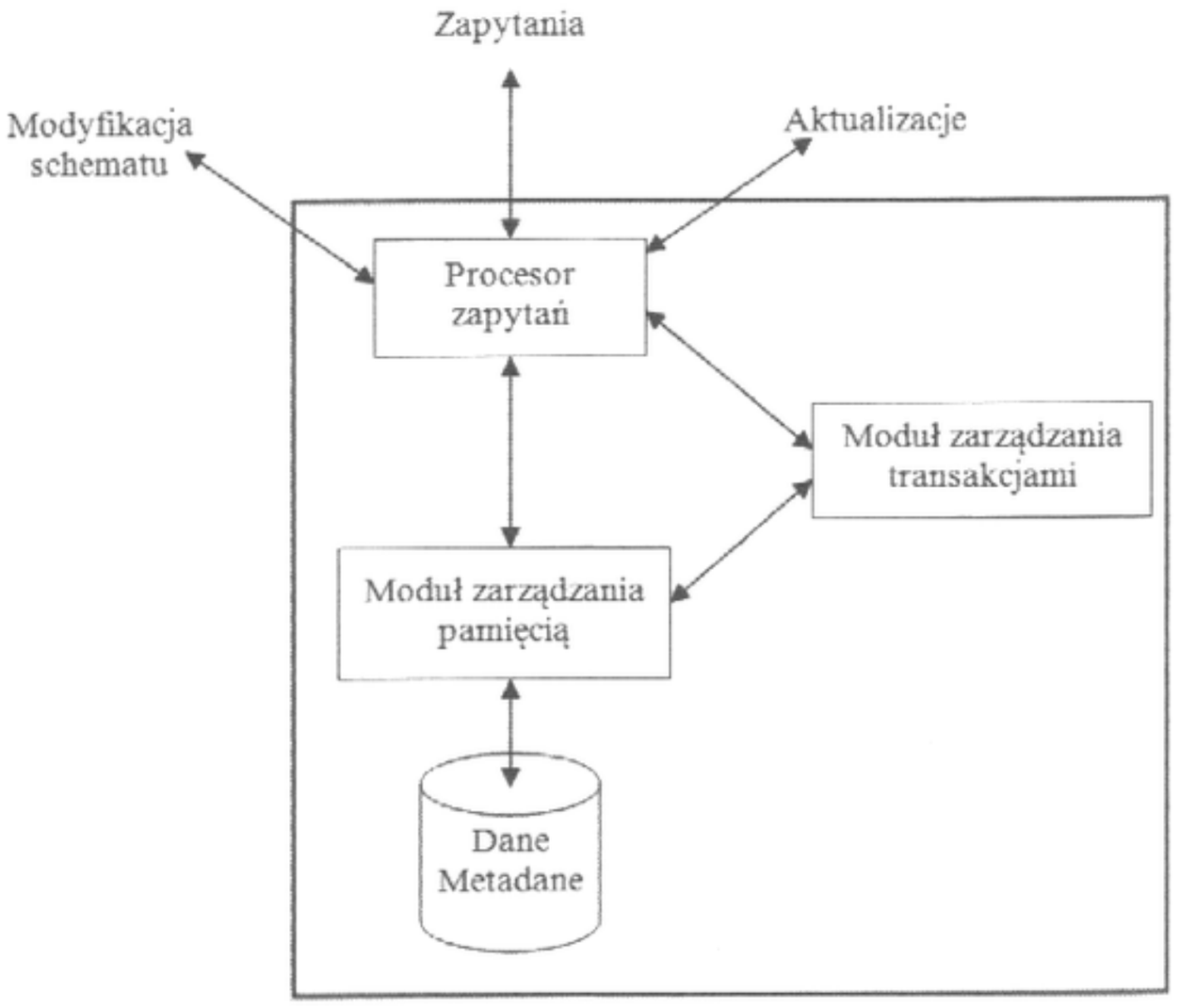
Schemat SZBD

Schemat systemu zarządzania bazą danych wraz z otaczającym go środowiskiem został pokazany na rysunku 3.4.

Moduł zarządzania pamięcią przechowuje informacje o miejscu zapisania plików bazy danych na dysku oraz obsługuje pamięć operacyjną.

Procesor zapytań przekształca zapytanie lub operację języka zapytań w ciąg poleceń żądających określonych danych.

Moduł zarządzania transakcjami kontroluje poprawność i kompletność wykonania wszystkich transakcji.



Rysunek 3.4.
Architektura systemu zarządzania bazą danych

PYTANIA KONTROLNE

1. Podaj definicję Systemu Zarządzania Bazą Danych.
2. Z jakich elementów składa się *System Baz Danych*?
3. Czym charakteryzuje się architektura *klient-serwer*?
4. Czym charakteryzuje się architektura *3-warstwowa*?
5. Na czym polega trwałość bazy danych?
6. Co oznacza pojęcie „integralność bazy danych”?
7. Na czym polega współdzielenie danych w bazie?
8. Co oznacza pojęcie „abstrakcja danych w bazie”?
9. Na czym polega niezależność danych w bazie?

ZADANIE

Zbuduj model komunikowania się z bazą *Moja_szkoła* użytkowników korzystających z sieci dostępnej w Twojej szkole. Określ miejsce gromadzenia danych oraz serwer, na którym będzie znajdował się System Zarządzania Bazą Danych. Sprawdź, czy baza posiada takie cechy jak trwałość, integralność, abstrakcja, niezależność danych. Zastanów się, jak zapewnić bezpieczeństwo oraz współdzielenie danych.

4 Serwery bazodanowe

4.1. Wprowadzenie

Do najpopularniejszych serwerów bazodanowych należą:

- SQL Server firmy Microsoft;
- Oracle Database firmy Oracle;
- MySQL — produkt powstały jako *open source*;
- PostgreSQL — darmowy serwer baz danych powstały na Uniwersytecie Kalifornijskim;
- DB2 firmy IBM;
- InterBase firmy Borland.

Do pracy z bazami danych można wybrać dowolny serwer bazodanowy. Niezależnie od tego, czy jest to produkt darmowy, czy komercyjny, zestaw podstawowych elementów języka SQL w większości z nich jest podobny. Natomiast każdy serwer zawiera niestandardowe rozszerzenia języka SQL.

4.2. Serwer MySQL

MySQL to bardzo wydajny i stabilny serwer o małych wymaganiach sprzętowych. Charakterystyczne cechy MySQL to:

- praca w zasadzie na wszystkich dostępnych platformach;
- udostępnianie różnych silników bazodanowych (na przykład bardzo szybkie tabele MyISAM lub tabele HEAP);
- podstawowa implementacja złączeń;
- wykorzystywanie systemu przesyłania skompresowanych danych pomiędzy klientem i serwerem;

- udostępnianie serwera w postaci osobnego programu lub biblioteki;
- obsługa zapytań rozproszonych;
- udostępnianie mechanizmów replikacji.

Zalety serwera MySQL:

- bardzo szybki, nadający się do obsługi często odwiedzanych stron WWW;
- małe wymagania sprzętowe;
- darmowy i nielimitowany.

Wady serwera MySQL:

- transakcje wymagają korzystania z silnika bazodanowego InnoDB;
- licencja GPL uniemożliwia sprzedaż produktów, których działanie jest powiązane z serwerem MySQL.

4.2.1. Instalowanie serwera w systemach Ubuntu i Debian

Większość dystrybucji systemu Linux (na przykład Ubuntu i Debian) zawiera pakiet MySQL. Może on być standardowo dostarczony w trakcie instalacji systemu lub może zostać przez użytkownika doinstalowany.

Aby doinstalować MySQL należy w terminalu wpisać następujące polecenie:

```
sudo apt-get install mysql-server
```

Podczas instalacji pojawi się pytanie o ustawienie hasła do konta root serwera MySQL.

Instalacja MySQL dla Ubuntu 12.4

W Ubuntu 12.04 serwer MySQL 5.5 jest instalowany domyślnie podczas instalowania systemu. Po zakończeniu instalacji jest on uruchamiany automatycznie.

W celu sprawdzenia czy serwer MySQL jest uruchomiony należy w terminalu wpisać następujące polecenie:

```
sudo service mysql status
```

W wyniku wykonania polecenia zostanie wyświetlona informacja zbliżona do podanej niżej:

```
mysql start/running, process 1218
```

Jeżeli serwer MySQL nie działa prawidłowo należy wpisać polecenie:

```
sudo service mysql restart
```

Po wykonaniu tych czynności można przystąpić do pracy z bazą.

Do pracy z bazą można wykorzystywać utworzone konto użytkownika oraz utworzoną bazę, logując się do serwera przez polecenie:

```
mysql -u nazwa_uzytkownika -phasło nazwa_bazy
```

lub bez wybierania bazy danych — przez polecenie:

```
mysql -u nazwa_uzytkownika -p
```

Po uzyskaniu połączenia należy wybrać bazę danych, wpisując polecenie:

```
use nazwa_bazy;
```

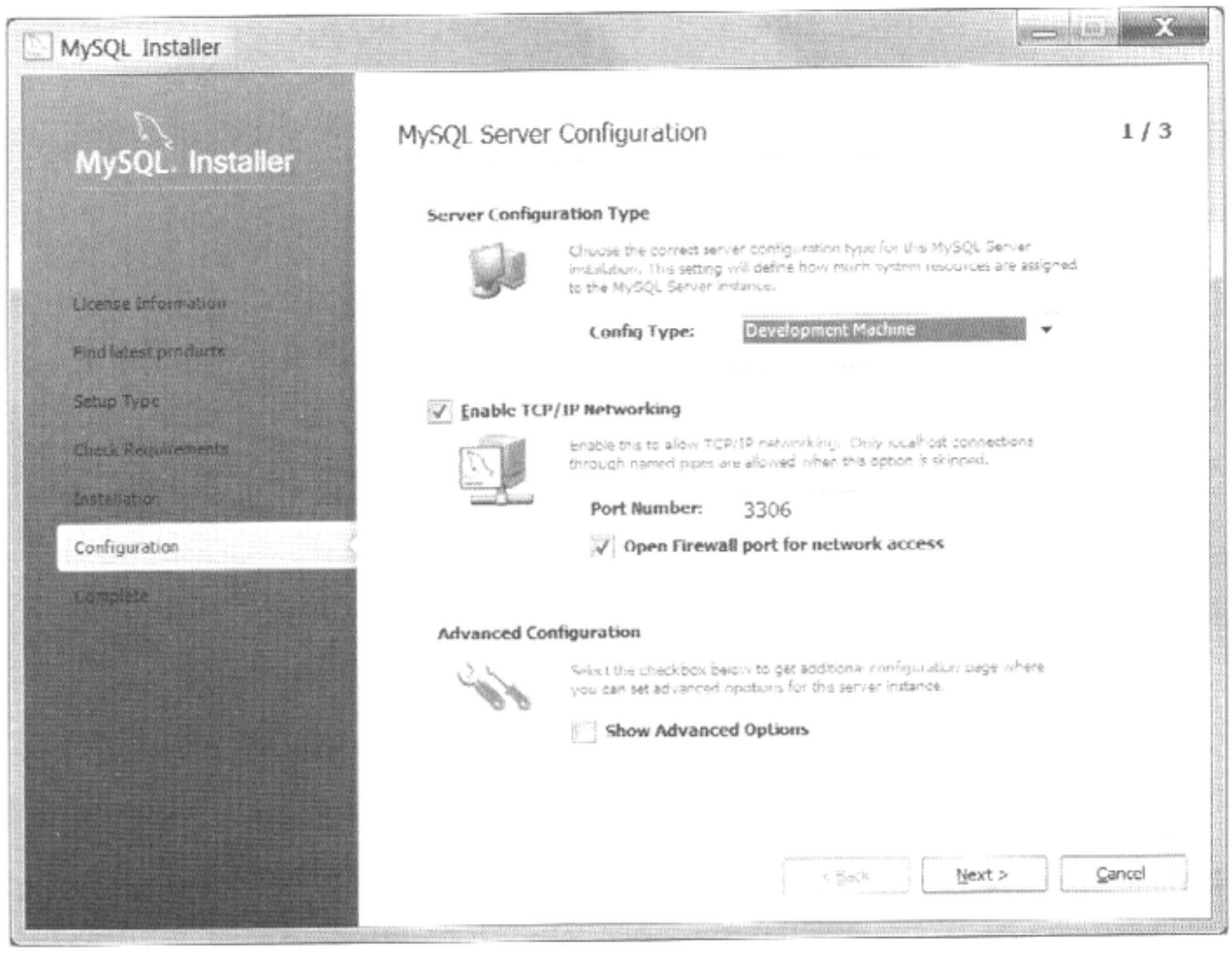
Do rozłączenia się z serwerem służy polecenie:

```
quit
```

4.2.2. Instalowanie serwera w systemie Windows

Pakiet MySQL jest dostępny dla wszystkich wersji systemu Windows i proces jego instalowania dla każdej wersji jest w zasadzie taki sam. Aby zainstalować MySQL, trzeba ze strony *www.mysql.com* pobrać pakiet instalacyjny programu. Po jego pobraniu należy uruchomić plik instalacyjny, a instalator MySQL Installer for Windows poprowadzi nas przez cały proces. Po zakończeniu instalowania pojawi się okno z opcją *Launch the MySQL Instance Configuration Wizard*. Po zaznaczeniu tej opcji zostanie uruchomiony kreator konfiguracji, który pozwoli określić sposób uruchamiania usługi MySQL (rysunek 4.1).

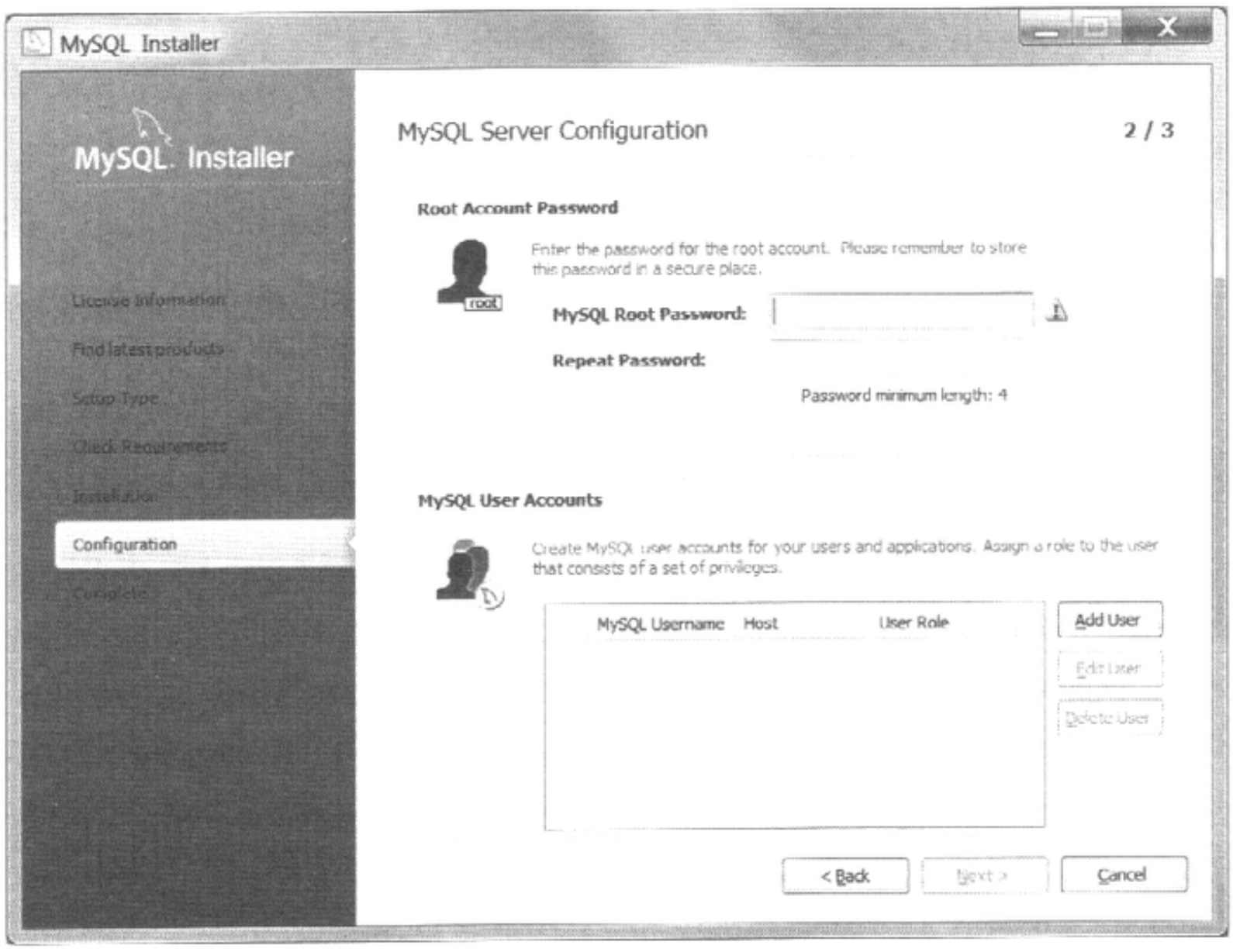
Rysunek 4.1. Okno kreatora konfiguracji serwera MySQL



W pierwszym oknie kreatora, wybierając typ serwera, określamy ilość dostępnej dla niego pamięci operacyjnej i czasu procesora. Pierwsza opcja oznacza przyznanie serwerowi MySQL najmniejszej ilości zasobów komputera, ostatnia — największej. Zaznaczenie opcji *Enable TCP/IP Networking* jest wymagane, jeżeli planujemy łączenie się

z serwerem przez sieć. Należy również podać numer portu serwera (domyślny to *3306*) oraz odblokować zaporę systemową.

W kolejnym oknie trzeba podać hasło dla administratora (*Root Account Password*) oraz zdefiniować użytkowników baz danych i nadać im odpowiednie uprawnienia (rysunek 4.2).



Rysunek 4.2. Konfigurowanie użytkowników baz danych

W ostatnim oknie należy ustawić serwer MySQL jako automatycznie uruchamianą usługę.

Do serwera można zalogować się jako administrator, wpisując z konsoli systemowej polecenie:

```
mysql -u root -p
```

Po wpisaniu polecenia

```
CREATE DATABASE nazwa_bazy;
```

zostanie utworzona nowa baza danych.

Po wpisaniu polecenia

```
GRANT ALL ON nazwa_bazy.* TO nazwa_uzytkownika IDENTIFIED BY
'hasło_użytkownika';
```

zostanie utworzone konto użytkownika o podanej nazwie i podanym haśle. Użytkownik będzie miał dostęp do bazy, której nazwa została podana w klauzuli `GRANT ALL ON`. Będzie mógł na niej wykonywać wszystkie operacje.

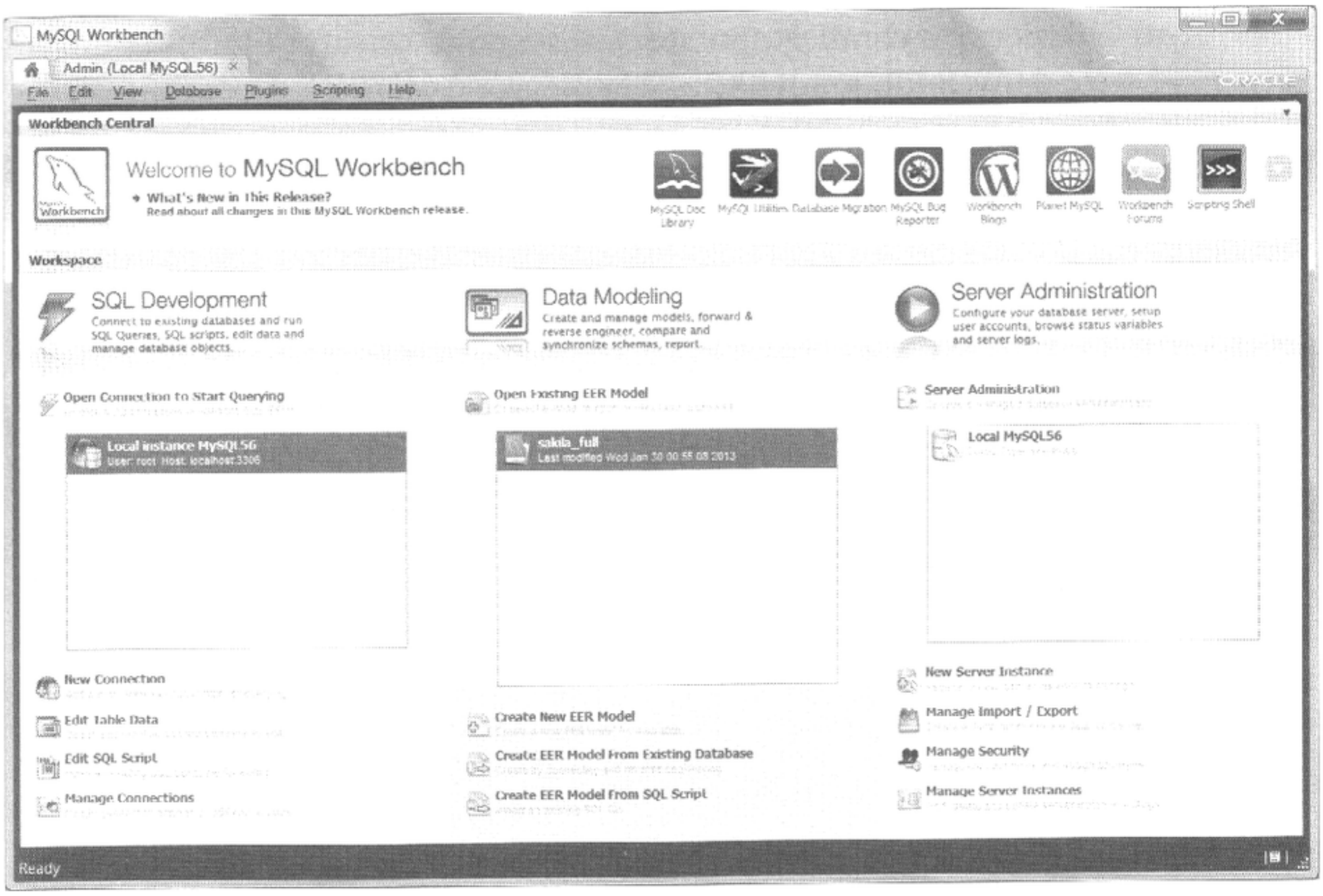
Połączenie z serwerem bazodanowym nastąpi po wpisaniu polecenia:

```
mysql -u nazwa_uzytkownika -phasło nazwa_bazy
```

4.2.3. MySQL Workbench

MySQL Workbench to narzędzie, za pomocą którego można na poziomie interfejsu graficznego zarządzać bazami danych MySQL. Po zainstalowaniu serwera można tę aplikację uruchomić, wybierając *Start/Wszystkie programy/MySQL/MySQL Workbench*.

Po uruchomieniu aplikacji pojawi się okno główne programu (rysunek 4.3).



Rysunek 4.3. Okno główne aplikacji MySQL Workbench

Składa się ono z dwóch części: *Workbench Central* i *Workspace*. W części *Workbench Central* znajdują się bieżące informacje o wydarzeniach i zasobach związanych z serwerem MySQL. Część *Workspace* zawiera narzędzia, które ułatwiają pracę z serwerem. Ta część została podzielona na trzy obszary:

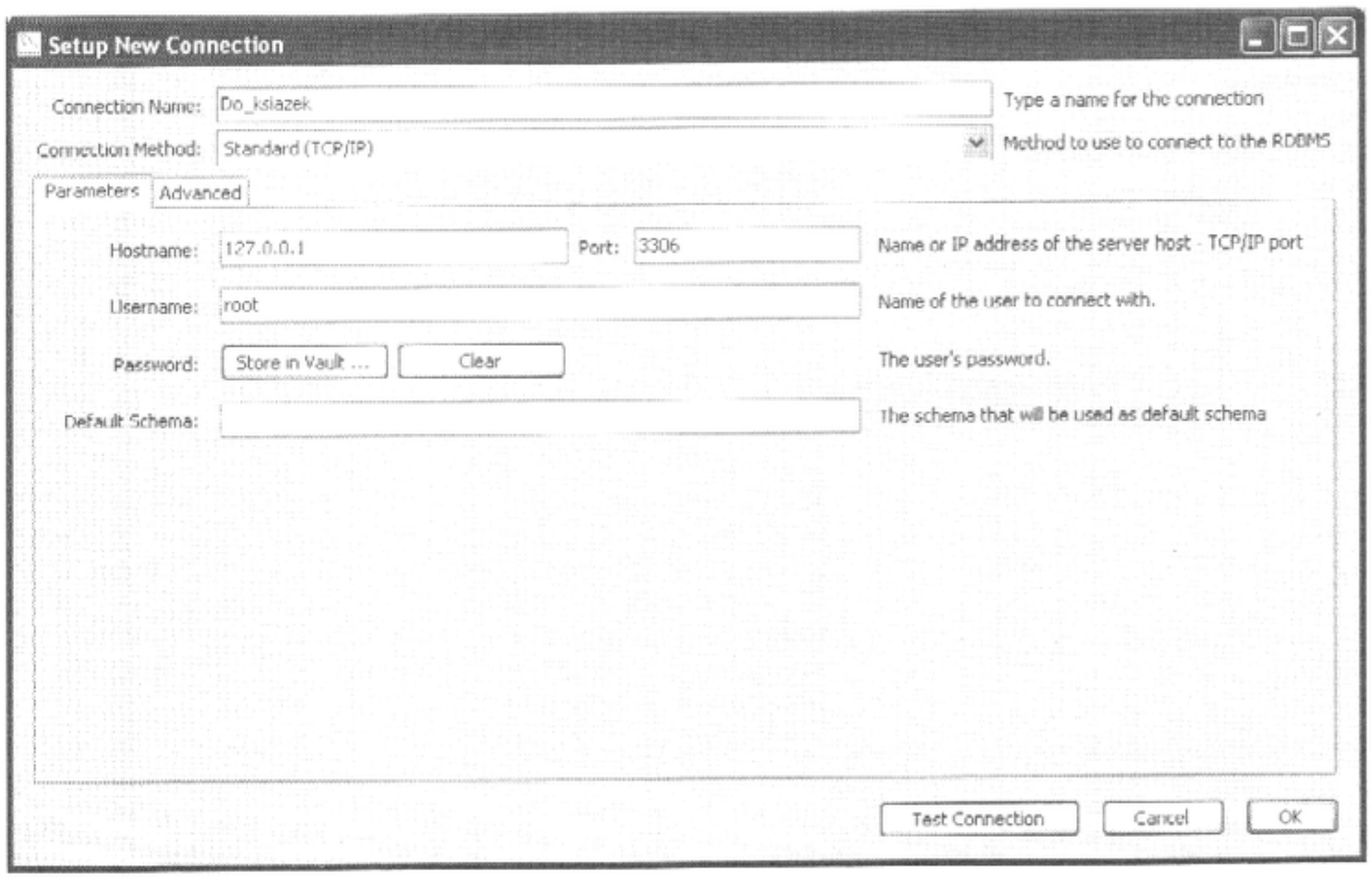
- *SQL Development* — zawiera narzędzia do konfigurowania połączeń, tworzenia baz danych i zapytań SQL (tworzenie połączenia, edytowanie danych w tabelach, zarządzanie połączeniami, edytowanie plików SQL).
- *Data Modeling* — umożliwia projektowanie baz danych (tworzenie i modyfikowanie modelu graficznego, zapisywanie schematu bazy do pliku, tworzenie i edytowanie tabel i danych).
- *Server Administration* — zawiera funkcje zarządzania instancją serwera (tworzenie nowych instancji, zarządzanie serwerem, konfigurowanie serwera, zarządzanie importem i eksportem danych, zarządzanie bezpieczeństwem).

SQL Development

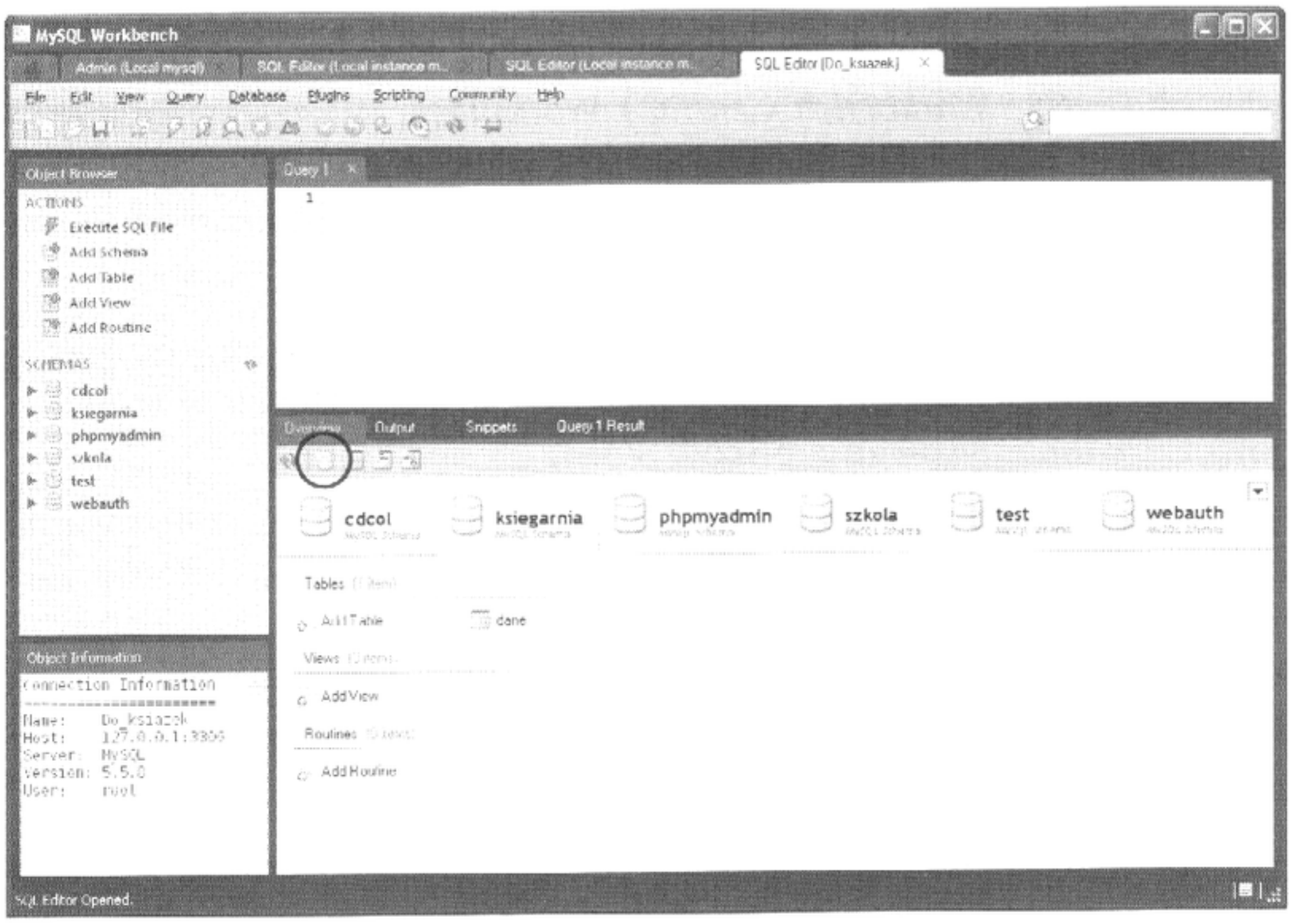
W panelu *SQL Development* możliwe jest zdefiniowanie połączenia dla tworzonej bazy danych oraz utworzenie tej bazy.

Nowe połączenie można skonfigurować po wybraniu opcji *New Connection*. W otwartym oknie (rysunek 4.4) ustawiamy parametry połączenia. Po jego zdefiniowaniu nowe połączenie pojawi się na liście połączeń.

Po dwukrotnym kliknięciu wybranego połączenia zostanie otwarte okno, w którym można tworzyć bazę danych lub modyfikować istniejącą (rysunek 4.5).



Rysunek 4.4. Konfigurowanie nowego połączenia

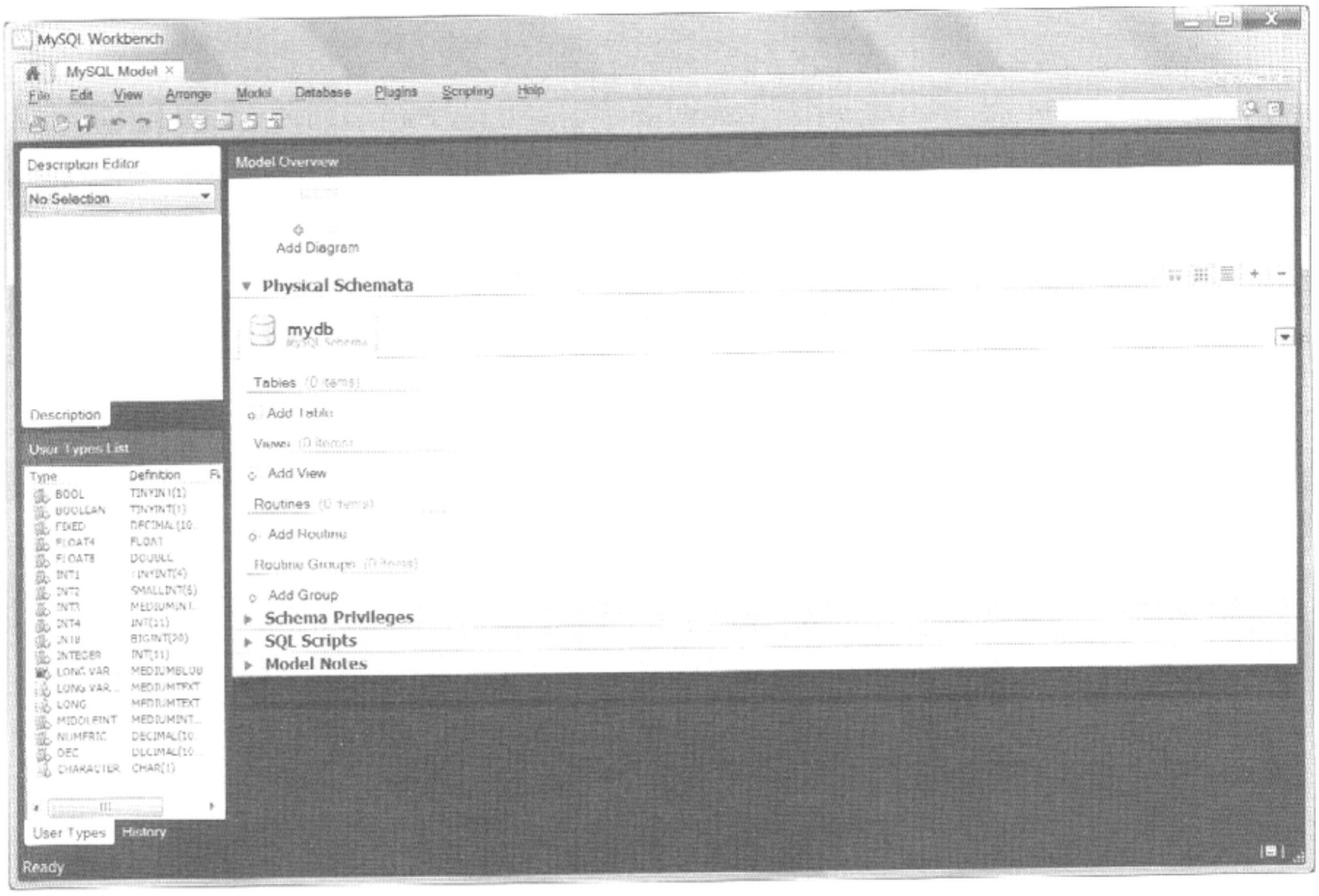


Rysunek 4.5. Okno edytowania baz danych

Po wybraniu ikony *Create a New Schema* można tworzyć nowy schemat bazy, po kliknięciu opcji *Add Table* można tworzyć nową tabelę. Podczas jej tworzenia można modyfikować kod SQL wykonywanej operacji.

Data Modeling

W panelu *Data Modeling* możliwe jest tworzenie i modyfikowanie modelu graficznego, zapisywanie schematu bazy do pliku, tworzenie i edytowanie tabel i danych baz danych. Po wybraniu opcji *Create New EER Model* możliwe jest tworzenie schematów tabel, widoków i diagramów baz danych (rysunek 4.6).



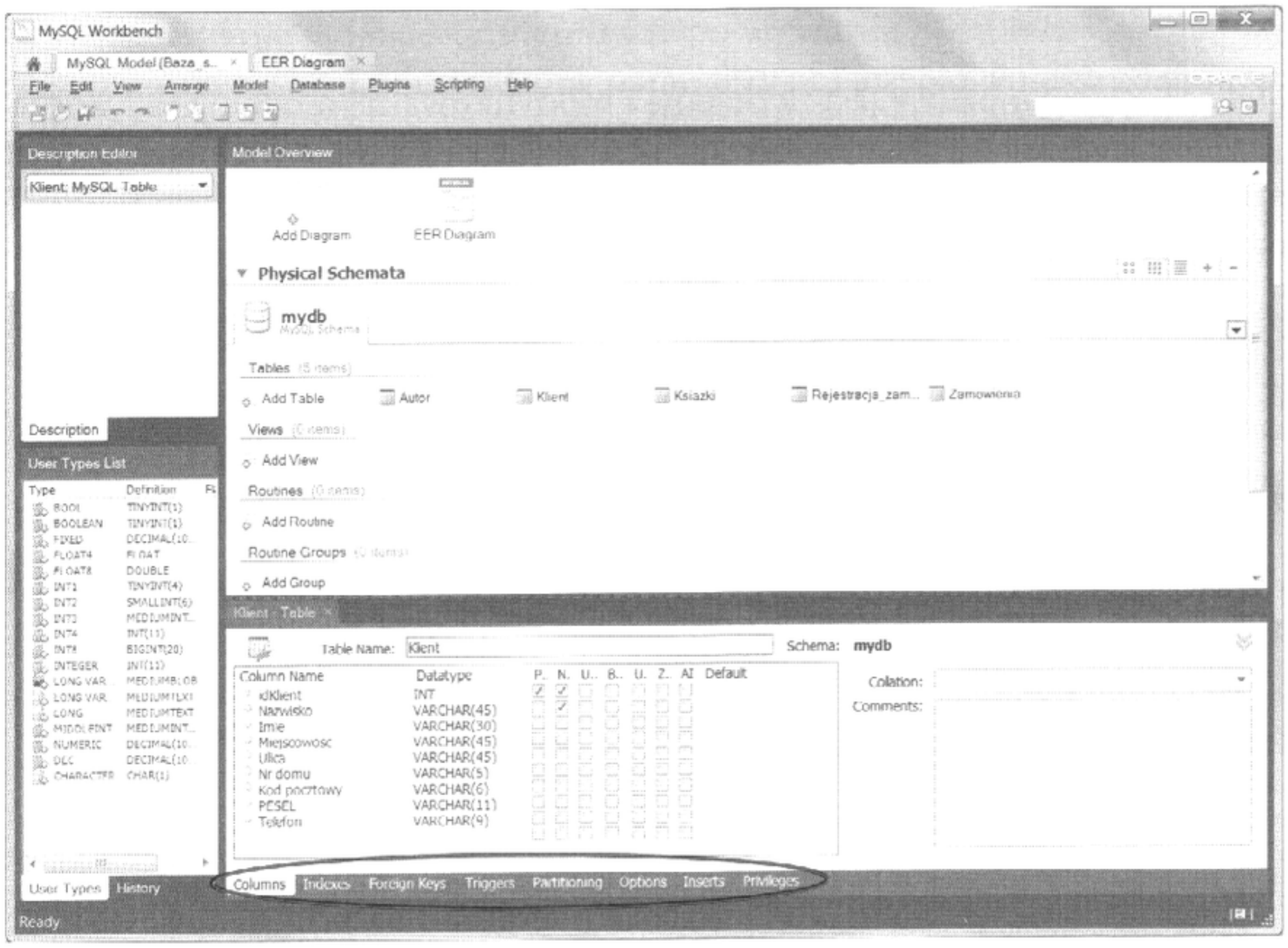
Rysunek 4.6. Okno tworzenia tabel i widoków bazy danych

Tworzenie struktury tabel jest realizowane po dwukrotnym kliknięciu opcji *Add Table* w otwartym panelu. Przy projektowaniu tabeli należy podać nazwę każdego pola oraz typ danych. Dodatkowo można określić właściwości każdego pola (na przykład definiować klucz podstawowy albo ustalić, czy wymagane jest wypełnienie pola). W dolnej części okna, wybierając odpowiednie opcje, można definiować indeksy, klucze obce, uprawnienia do tabeli oraz wprowadzać dane (rysunek 4.7).

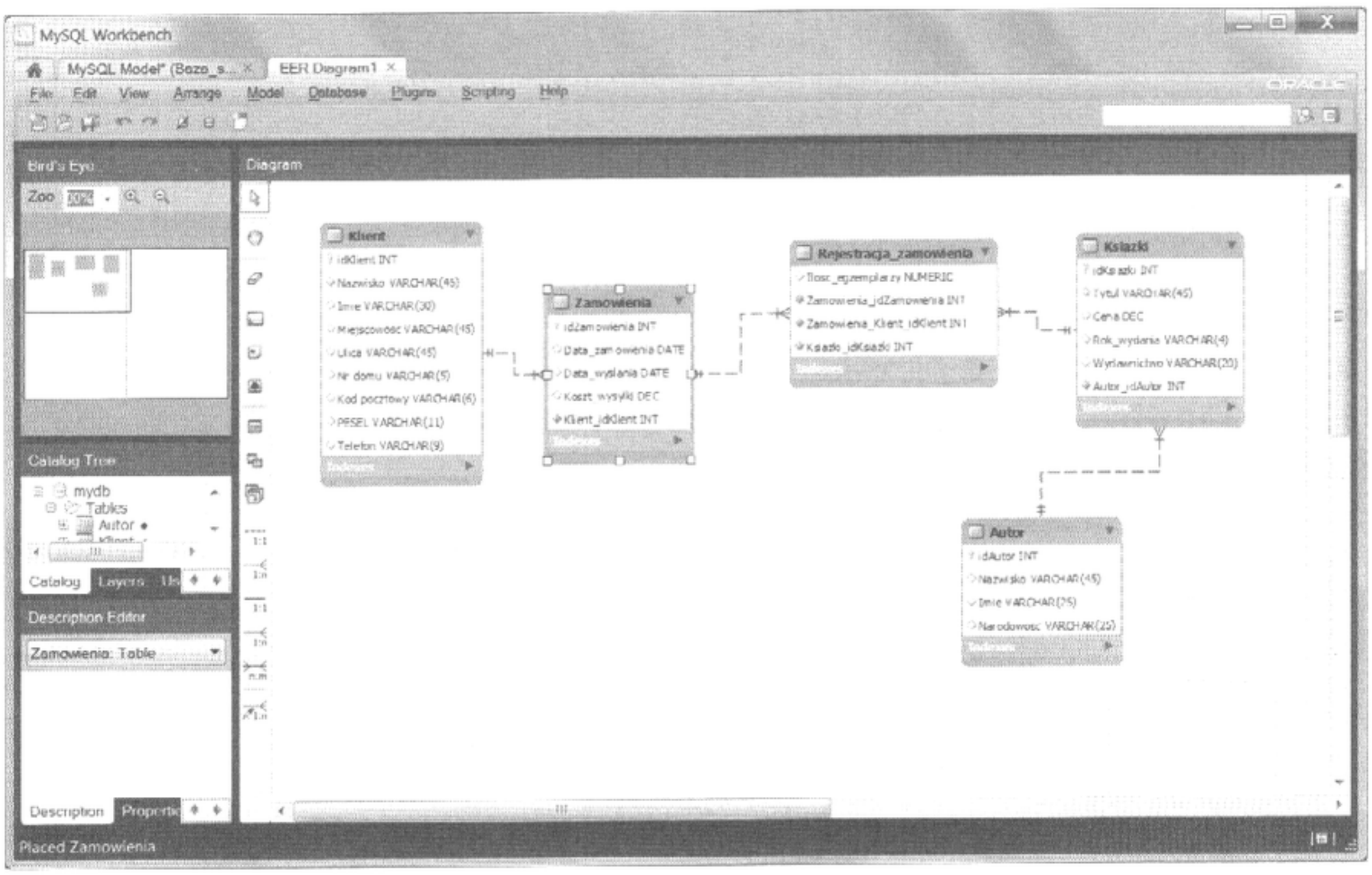
W panelu *Create New EER Model* po dwukrotnym kliknięciu opcji *Add Diagram* można zdefiniować połączenia i utworzyć diagram EER dla zaprojektowanych tabel (rysunek 4.8).

Utworzony model bazy danych można zapisać do pliku. Kolejnym krokiem po zapisaniu bazy do pliku jest utworzenie bazy danych na podstawie zaprojektowanego modelu. W tym celu należy wybrać w menu *Database/Synchronize Model* i w otwartym oknie określić parametry bazy (rysunek 4.9). Następnie w oknie *Select Schema to Synchronize* trzeba wybrać model tworzonej bazy danych. Zostanie wygenerowany skrypt tworzący bazę i po zatwierdzeniu baza danych zostanie utworzona.

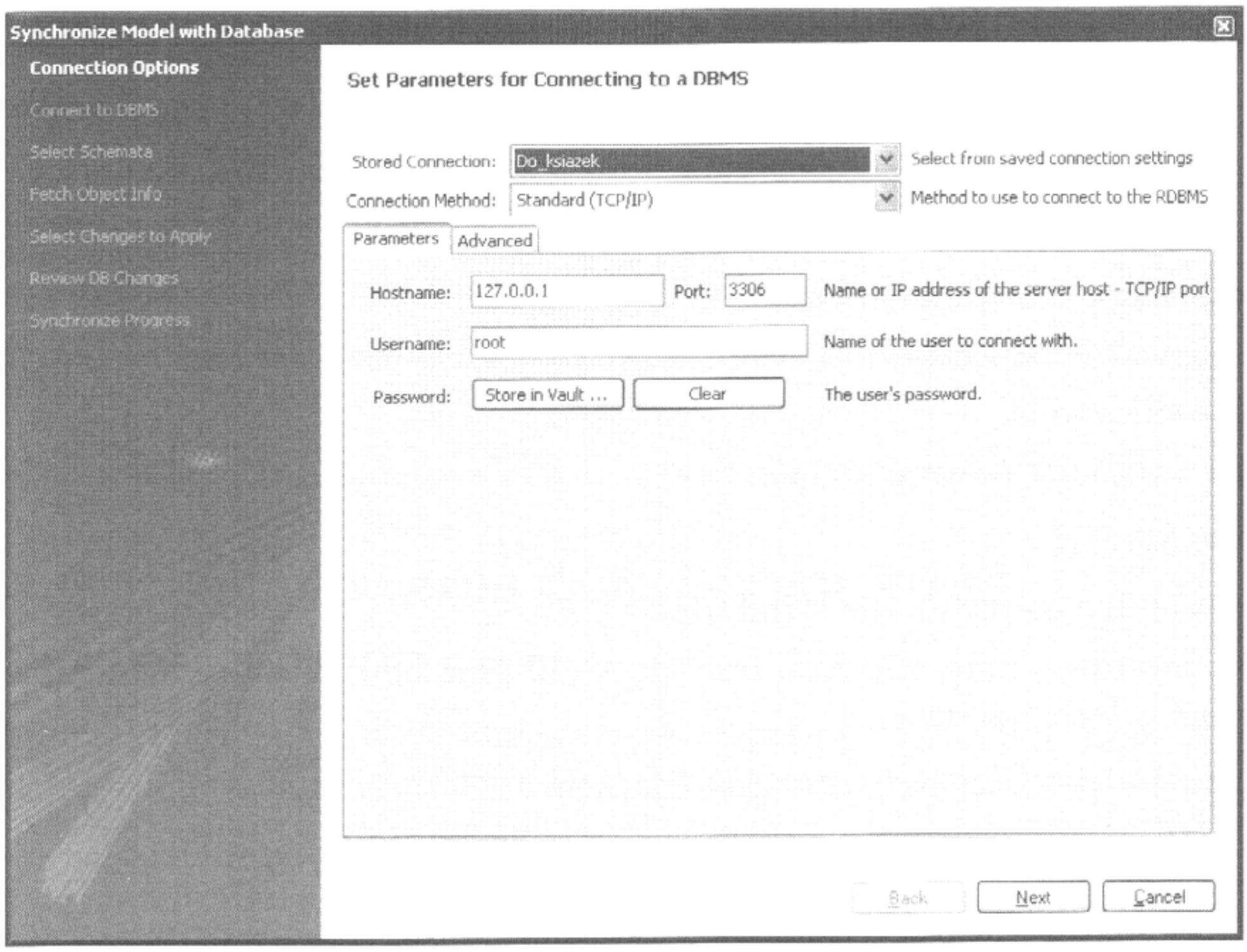
Tworzenie nowego użytkownika bazy danych i nadawanie mu praw dostępu do bazy można realizować w obszarze *Schema Privileges* po dwukrotnym kliknięciu opcji *Add User* (rysunek 4.10).



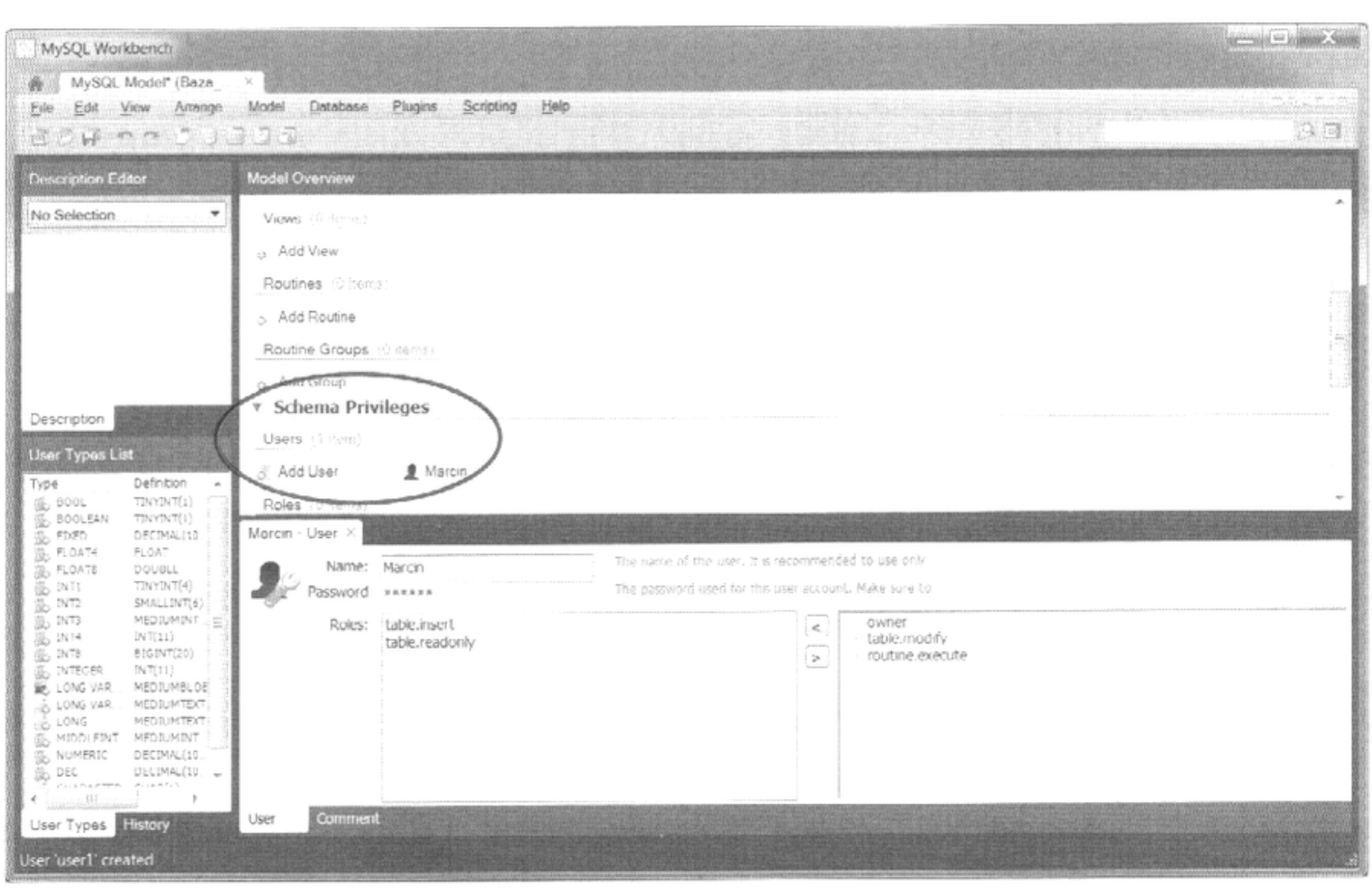
Rysunek 4.7. Okno projektowania tabeli



Rysunek 4.8. Okno tworzenia diagramu EER



Rysunek 4.9. Okno tworzenia bazy danych na podstawie jej modelu



Rysunek 4.10. Okno tworzenia nowego użytkownika

4.3. MS SQL Server

MS SQL Server należy do zaawansowanych serwerów bazodanowych. Jego zalety to:

- restrykcyjne mechanizmy zapewniające bezpieczeństwo systemu,
- wbudowane mechanizmy replikacji i synchronizacji danych,
- partycjonowanie danych (zwiększenie wydajności),
- raportowanie danych, dzięki którym można przeprowadzać dokładne analizy danych,
- łatwość instalowania,
- dostępna wersja darmowa systemu.

Wady serwera MS SQL:

- dostępność tylko na platformie Windows,
- w wersji darmowej limit rozmiaru bazy do 4 GB oraz brak niektórych narzędzi dostępnych w wersji płatnej.

Cechy systemu zwiększające bezpieczeństwo danych:

- autoryzacja występująca zarówno przy instalowaniu, jak i na poziomie baz danych,
- dwa tryby uwierzytelniania (Windows i SQL Server),
- zarządzanie dzięki zastosowaniu ról,
- szyfrowanie danych,
- zastosowanie certyfikatów,
- szybkie przywracanie systemu.

System Microsoft SQL Server (MS SQL) jest pakietem komercyjnym, ale występuje również w wersji Express, z której można korzystać nieodpłatnie. Oprogramowanie można pobrać ze strony *www.microsoft.com/sqlserver/2008/en/us/default.aspx* lub *http://www.microsoft.com/poland/sql/default.mspx*.

Wersję Express możemy wykorzystać zarówno do celów naukowych, jak i komercyjnych. MS SQL Server w wersji Express został wyposażony w dodatkowe narzędzie do zarządzania serwerem — Express Manager.

Do zainstalowania serwera w systemie Windows potrzebne są dodatkowe składniki (nie dotyczy Windows 7 i wyżej).

Są to:

- Microsoft .NET Framework 3.5 SP1,
- Microsoft Windows Installer 4.5,
- Microsoft PowerShell 1.0.

Po uruchomieniu pakietu rozpocznie się proces instalowania, który umożliwia również konfigurację serwera. Po zakończeniu instalowania usługa jest uruchamiana automatycznie po każdym uruchomieniu komputera.

Po zainstalowaniu serwera można zalogować się do serwera i utworzyć bazę danych. W celu zalogowania się należy w wierszu poleceń wpisać:

```
sqlcmd -Snazwa_komputera\nazwa_usługi
```

Po zalogowaniu należy utworzyć nową bazę, wpisując polecenie:

```
CREATE DATABASE nazwa_bazy
```

Przy kolejnych połączeniach z serwerem podczas logowania można wskazać bazę danych, dodając w poleceniu opcję `-d`:

```
sqlcmd -Snazwa_komputera\nazwa_usługi -d nazwa_bazy
```

4.3.1. Instalowanie SQL Server 2008 Express Edition

Microsoft SQL Server 2008 Express to bezpłatna wersja MS SQL Server, która zawiera narzędzia graficznego interfejsu (SQL Management Studio) i pozwala na bezpłatne rozpowszechnianie własnych aplikacji bazodanowych. Zalety tego serwera to:

- graficzne narzędzia do administrowania serwerem,
- możliwość łatwego instalowania w systemach Windows,
- duża zgodność ze standardem SQL3.

Przykład 4.1. Proces instalowania SQL Server 2008 Express Edition

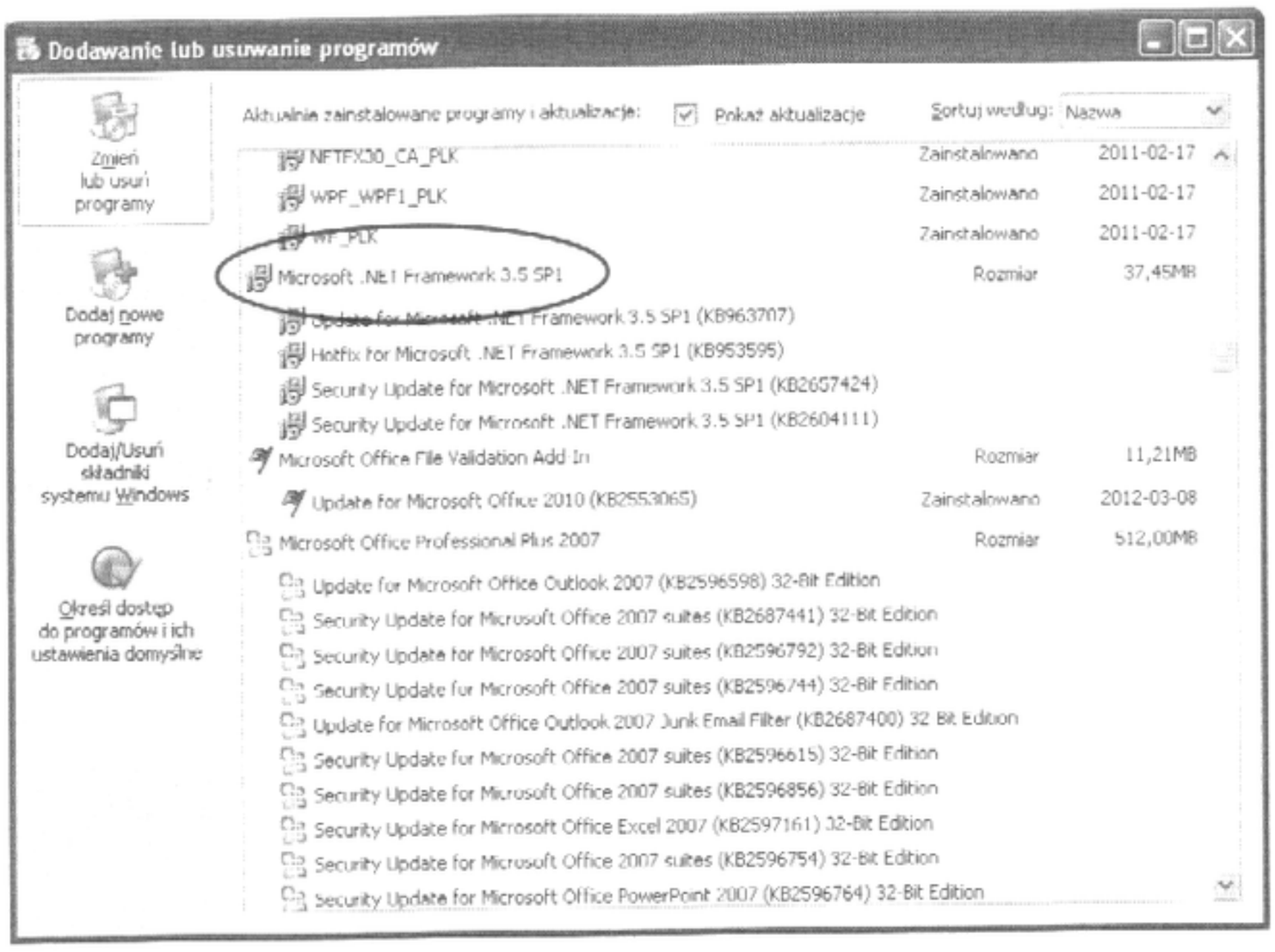
Jeżeli zamierzamy zainstalować SQL Server 2008 Express Edition, pierwszym krokiem powinno być pobranie pliku instalacyjnego tej aplikacji, na przykład ze strony *http://www.microsoft.com/en-us/download/details.aspx?id=25052*.

Przed przystąpieniem do instalowania sprawdzamy, czy na komputerze jest zainstalowany Microsoft .NET Framework 3.5 SP1 (nie dotyczy Windows 7 i wyżej). Można to zrobić, otwierając *Panel sterowania* i wybierając *Dodaj/usuń programy* (rysunek 4.11). Podobnie możemy sprawdzić, czy zainstalowane są pozostałe składniki niezbędne do prawidłowego procesu instalowania MS SQL — Microsoft Windows Installer 4.5 oraz Microsoft PowerShell 1.0.

Brakujące składniki można pobrać ze stron:

- *http://www.microsoft.com/en-us/download/details.aspx?id=22* — Microsoft .NET Framework 3.5 SP1;
- *http://microsoft-windows-installer-vista-7-32-bits.en.softonic.com* — Microsoft Windows Installer 4.5;
- *http://www.microsoft.com/en-us/download/details.aspx?id=7217* — Microsoft PowerShell 1.0.

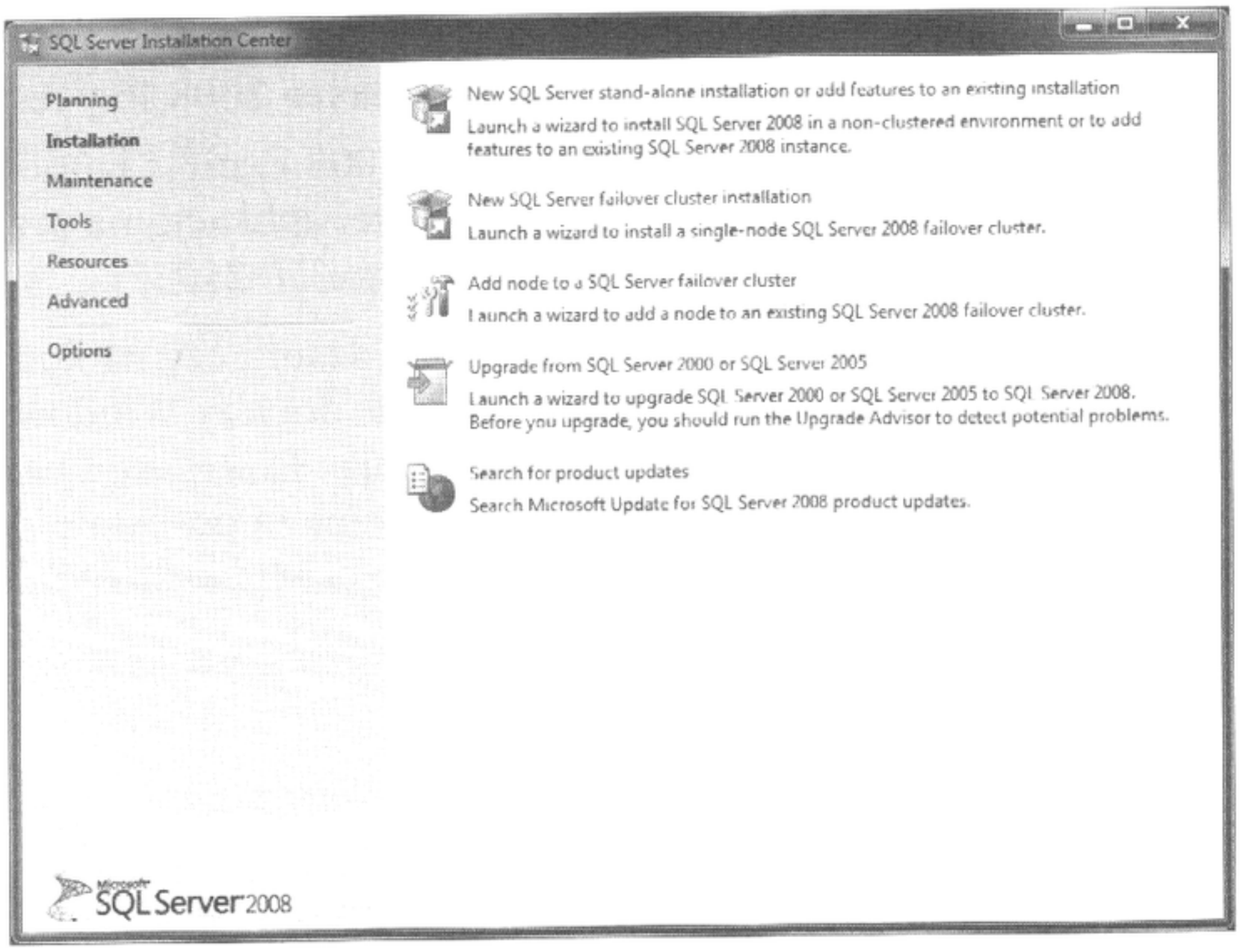
Rysunek 4.11. Panel sterowania, opcja Dodaj/usuń programy



Po zainstalowaniu powyższych narzędzi możemy przystąpić do instalowania SQL Server 2008 Express Edition.

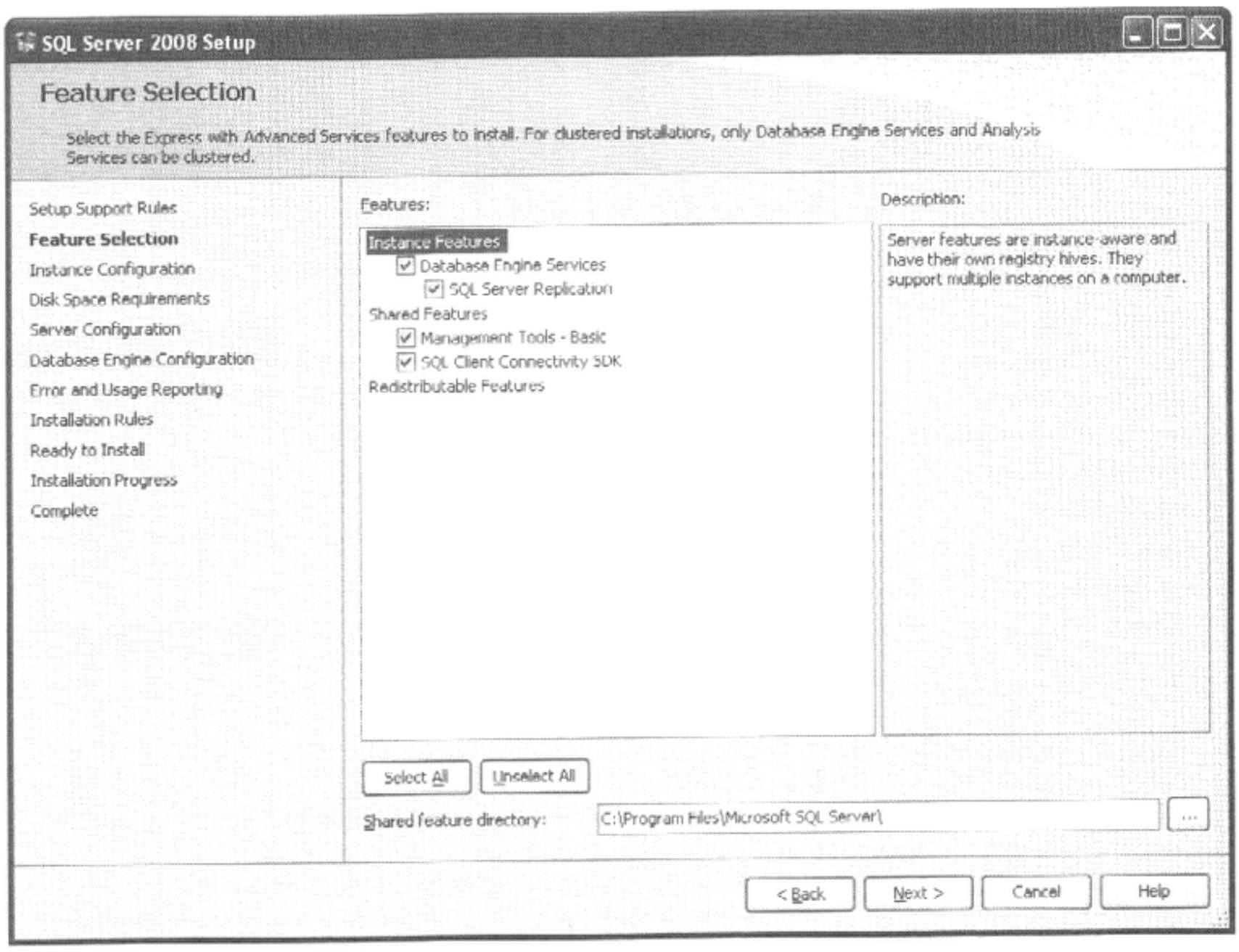
Uruchamiamy plik instalatora i w pierwszym oknie kreatora instalacji wybieramy opcję *Instalation*, a następnie opcję *New SQL Server stand-alone installation or add features to an existing instalation* (rysunek 4.12).

Rysunek 4.12. Rozpoczęcie instalowania SQL Server



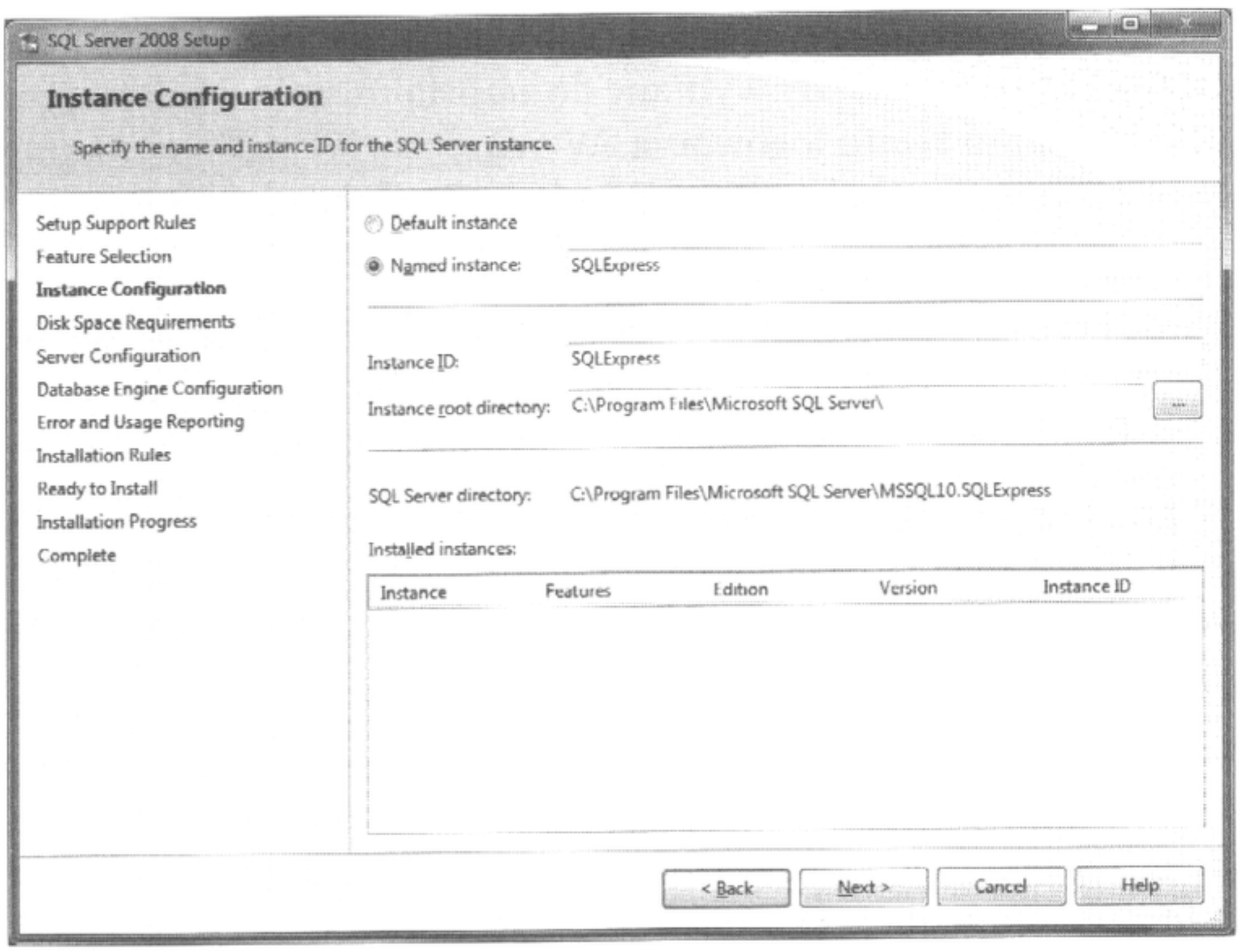
Wybieramy darmową wersję programu SQL Server, akceptujemy licencję i w kolejnym oknie zaznaczamy składniki systemu, które powinny zostać zainstalowane (najlepiej wybrać wszystkie dostępne — rysunek 4.13).

Rysunek 4.13. Wybór składników, które powinny zostać zainstalowane

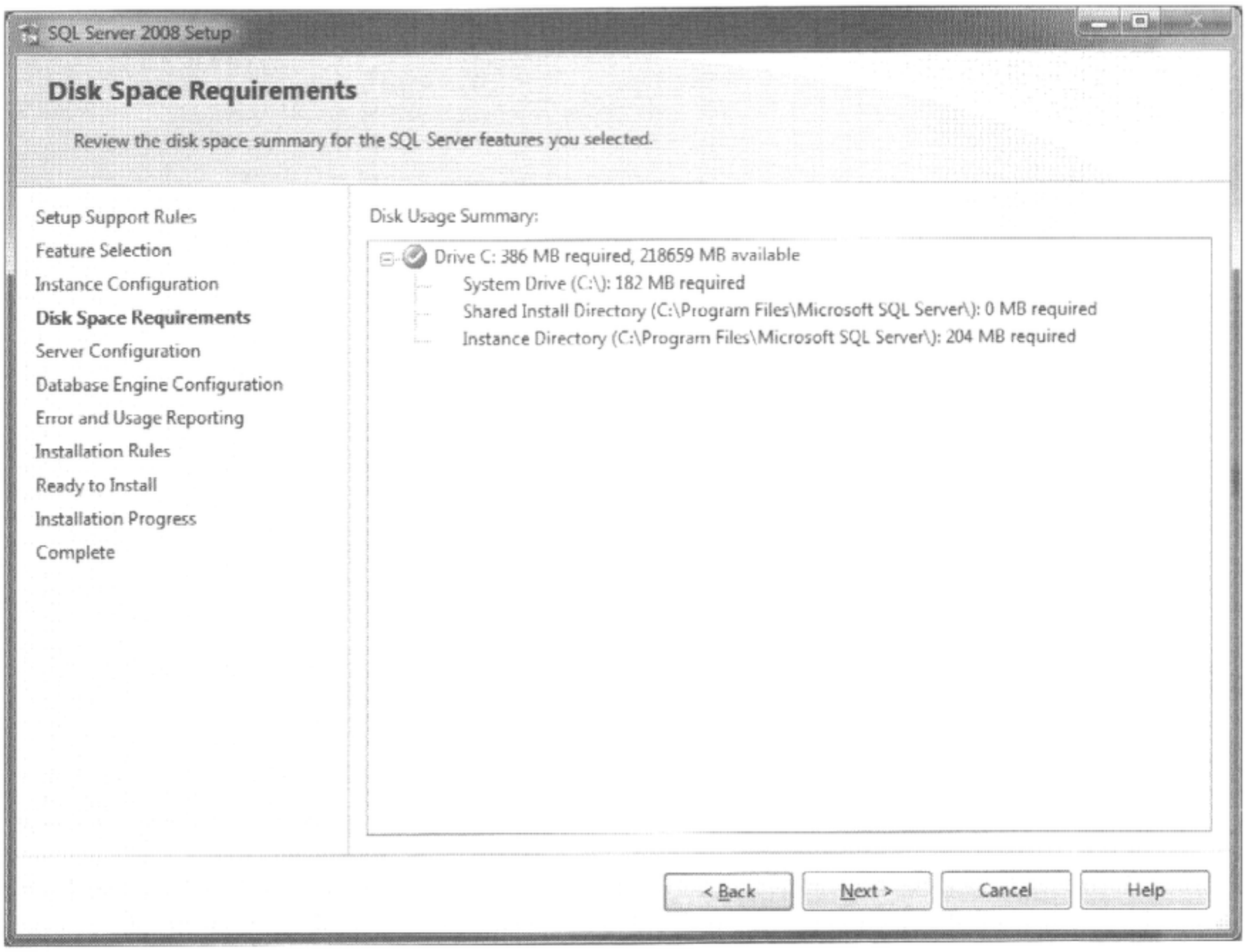


W oknie *Instance Configuration* określamy, czy instancja zostanie zainstalowana z domyślną nazwą, czy z nazwą zdefiniowaną przez użytkownika (rysunek 4.14). Wybieramy domyślną nazwę instancji i klikamy *Next*.

Rysunek 4.14. Wybór nazwy instancji



W oknie *Disk Space Requirements* następuje obliczenie miejsca potrzebnego do zainstalowania wybranych składników oraz sprawdzenie, czy na dysku jest wymagana ilość miejsca (rysunek 4.15). Klikamy *Next*.



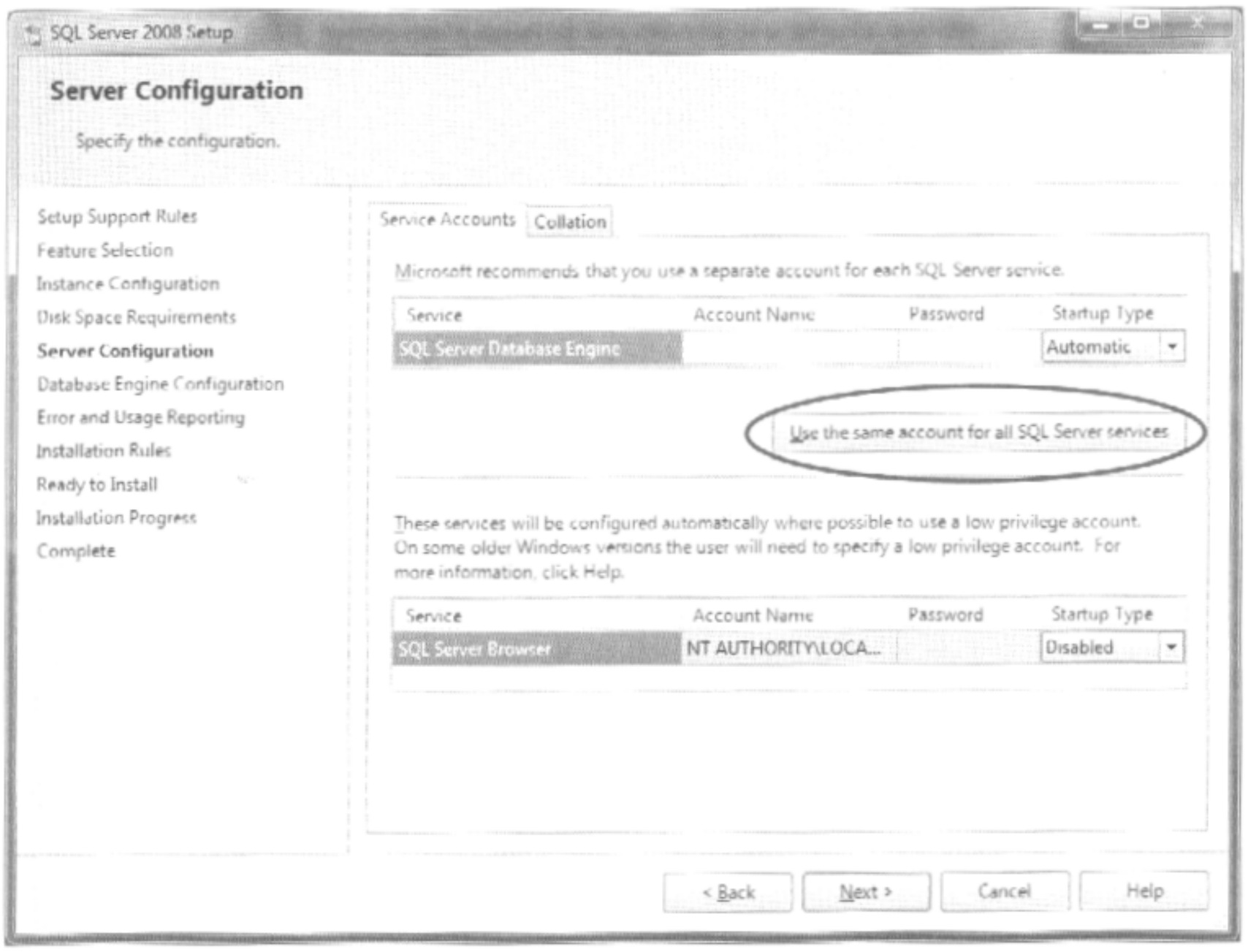
Rysunek 4.15. Sprawdzanie miejsca na dysku

Na karcie *Server Configuration* są konfigurowane usługi (rysunek 4.16). Ich lista zależy od funkcji, które zostały wybrane do zainstalowania. Najlepiej przypisać do wszystkich usług to samo konto logowania. W Windows XP *Account Name* dla usługi *SQL Server service* może pozostać puste, w Windows 7 należy zdefiniować użytkownika, wybierając przycisk *Use the same account for all SQL Server services*.

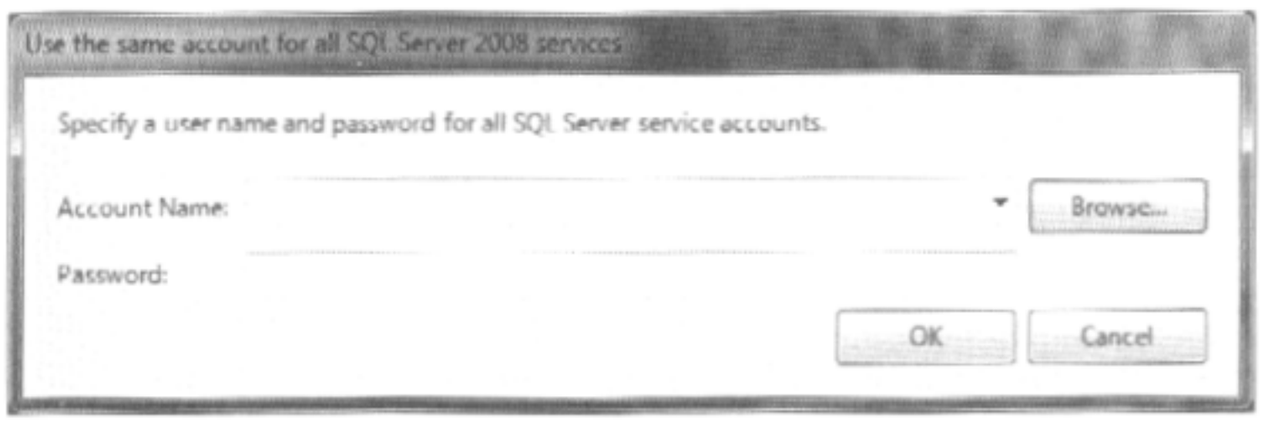
Po wybraniu przycisku zostanie otwarte okno wyboru nazwy użytkownika (rysunek 4.17), gdzie należy wpisać nazwę konta lub odszukać ją, klikając przycisk *Browse...* a następnie wprowadzić hasło przypisane do konta.

W kolejnym oknie, *Database Engine Configuration* (rysunek 4.18), w opcji *Authentication Mode*, można określić tryb zabezpieczeń. Może to być uwierzytelnienie systemu Windows lub tryb mieszany uwierzytelnienia. Wybieramy uwierzytelnienie systemu Windows.

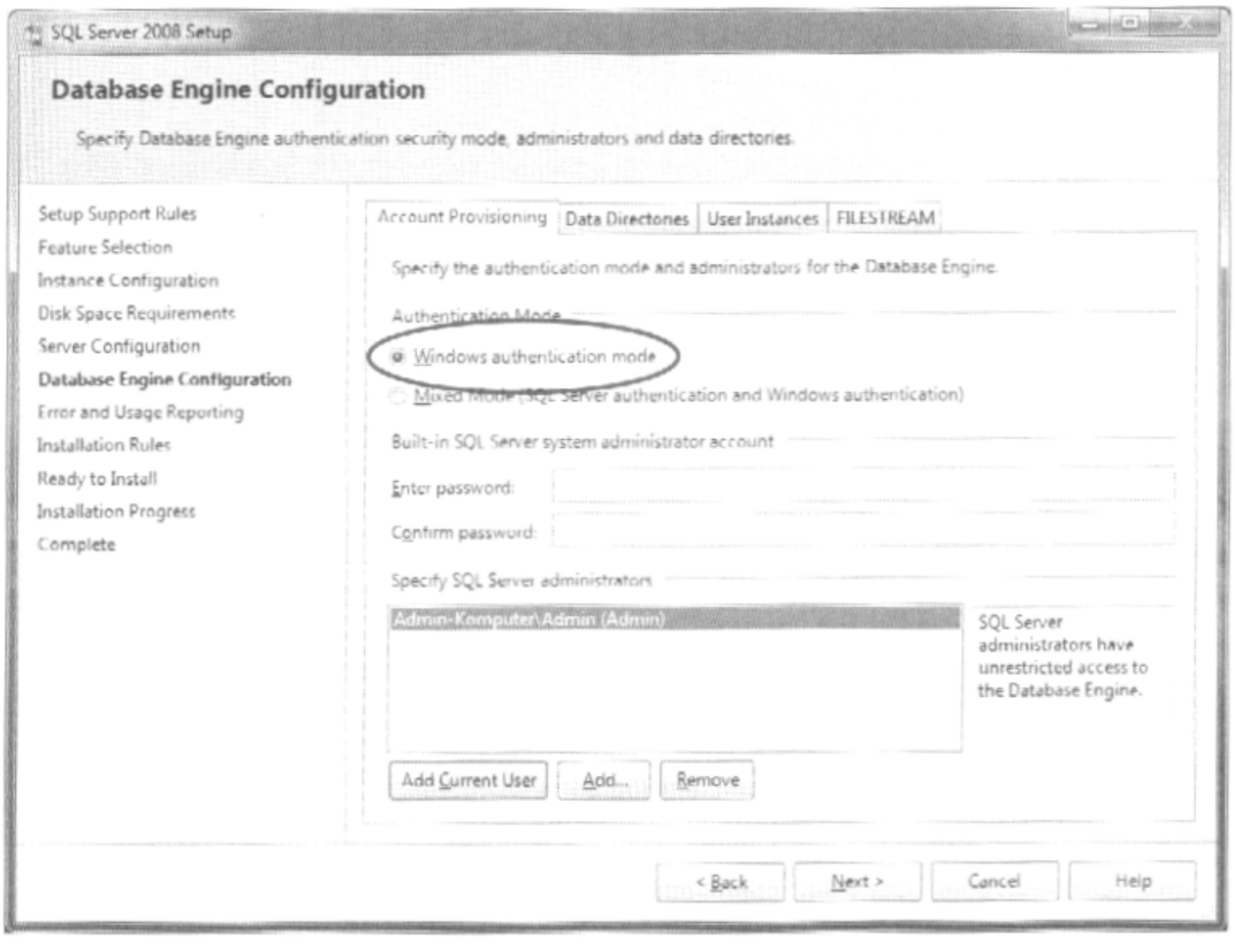
W opcji *Specify SQL Server administrators* można określić administratorów programu SQL Server. Wybieramy przycisk *Add Current User* (dodaj bieżącego użytkownika). Jeżeli chcemy nadać innym użytkownikom, grupom lub komputerom uprawnienia administratora dla programu SQL Server, klikamy przycisk *Add...* i modyfikujemy listę użytkowników. Następnie klikamy *Next*.



Rysunek 4.16. Konfigurowanie usług



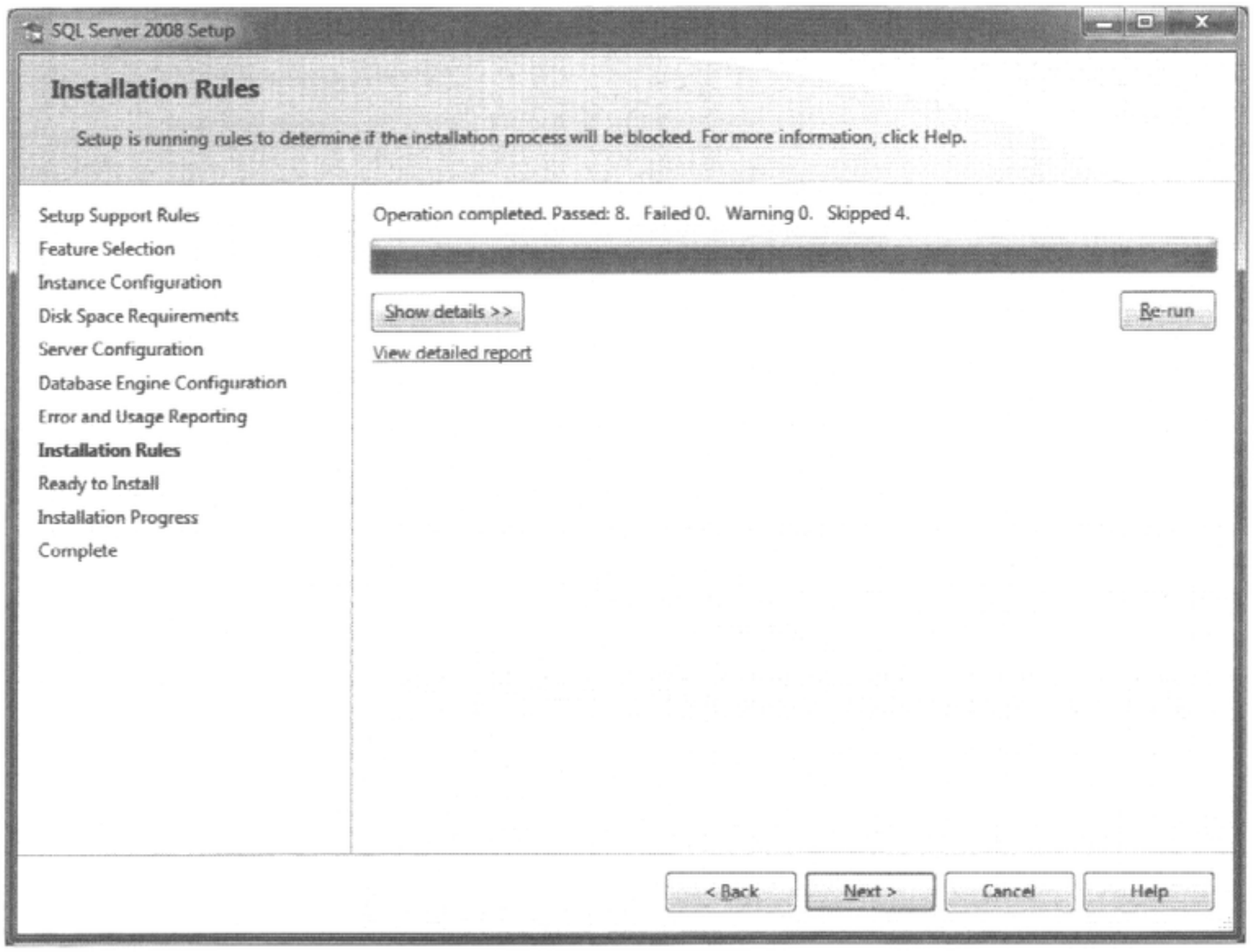
Rysunek 4.17. Wybór użytkownika dla instalowanych usług



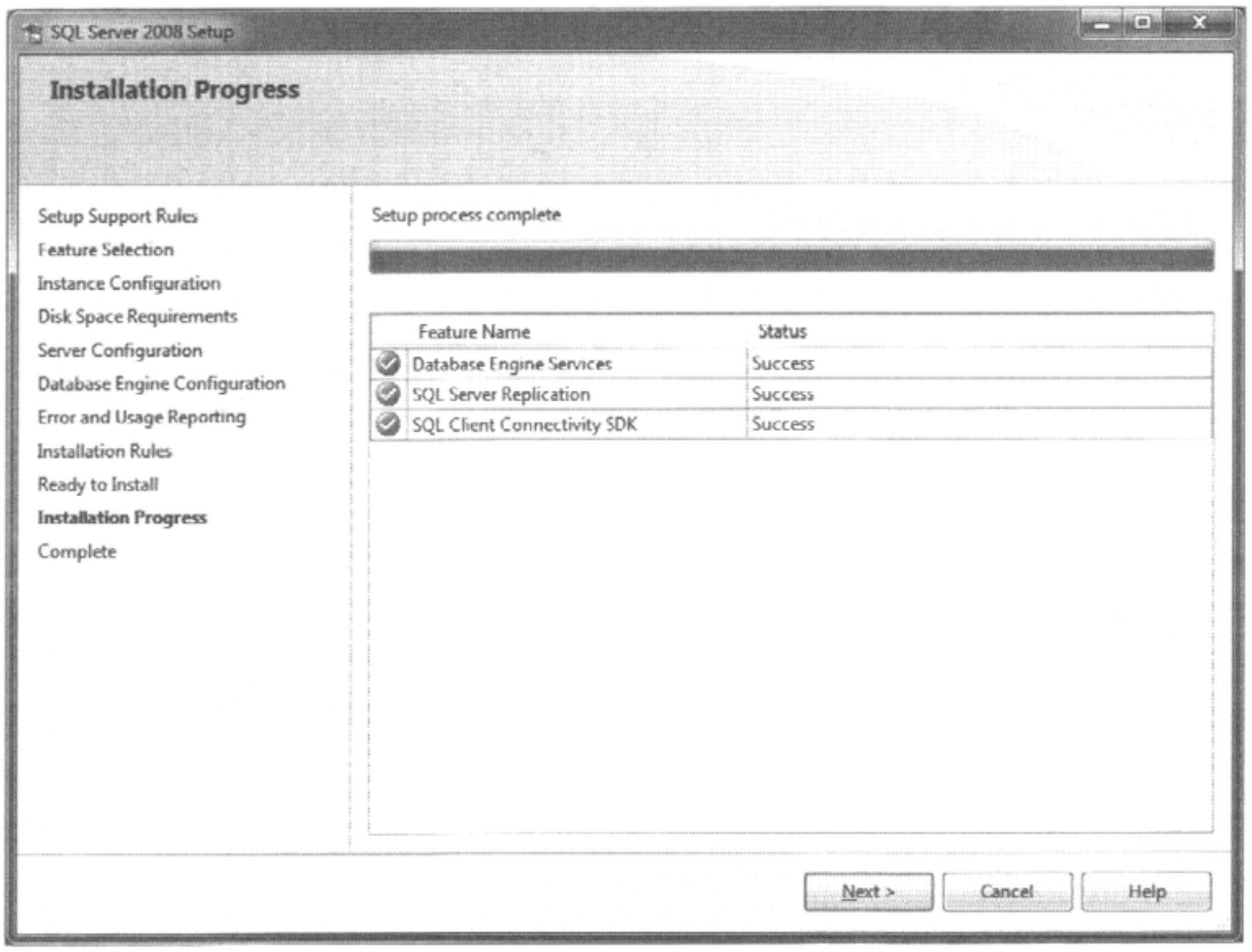
Rysunek 4.18. Definiowanie trybu zabezpieczeń

Na kolejnej karcie, *Error and Usage Reporting* (raportowanie błędów i użycia), można określić, jakie informacje na temat błędów powinny być przesyłane do firmy Microsoft. Bez wprowadzania jakichkolwiek zmian klikamy przycisk *Next*.

Następnie zatwierdzamy ustawienia, klikając kolejne przyciski *Next*. Możemy obserwować postęp procesu instalowania (rysunki 4.19 i 4.20), aż do uzyskania ostatecznego rezultatu (rysunek 4.21).

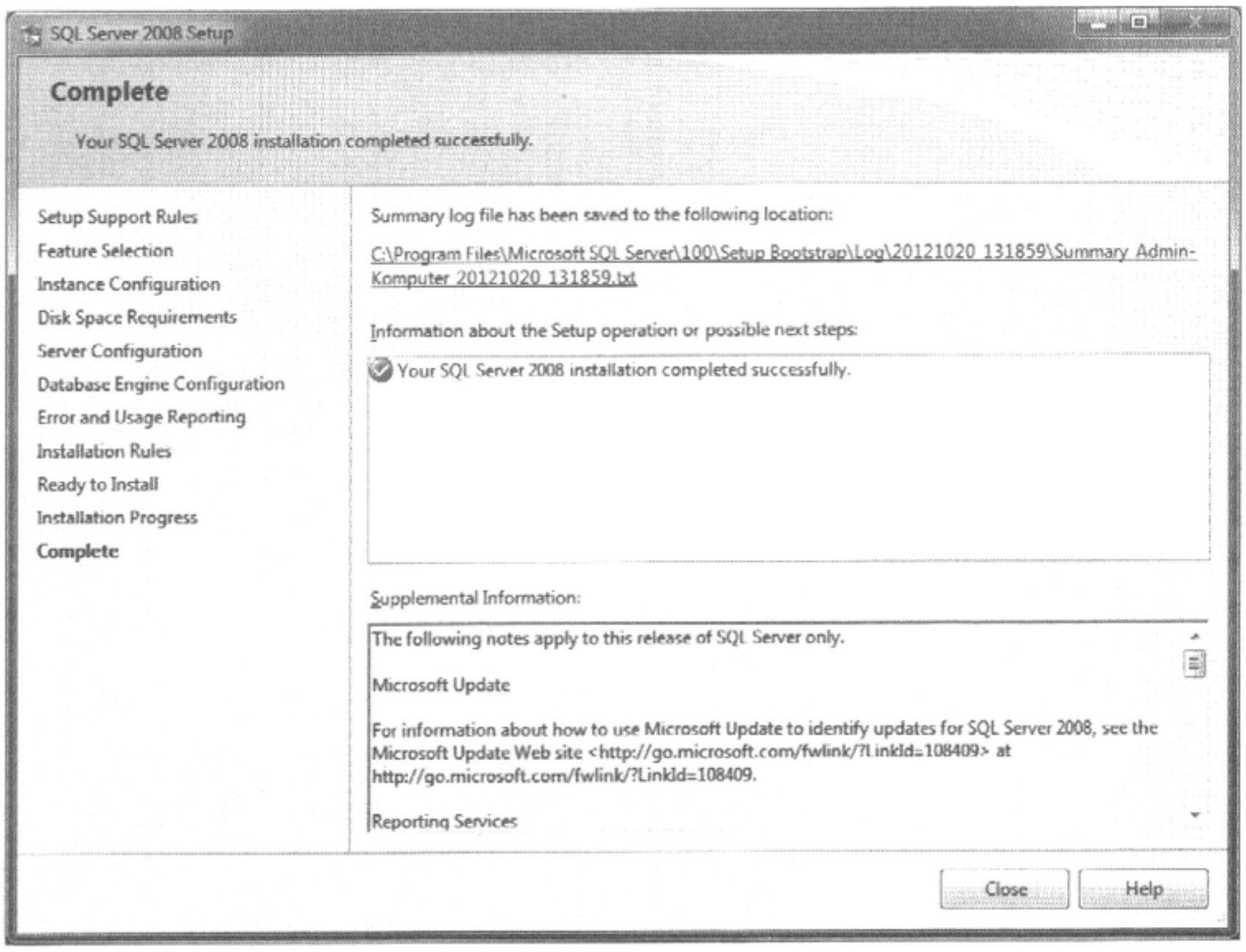


Rysunek 4.19. Przebieg procesu instalowania SQL Server 2008 Express Edition



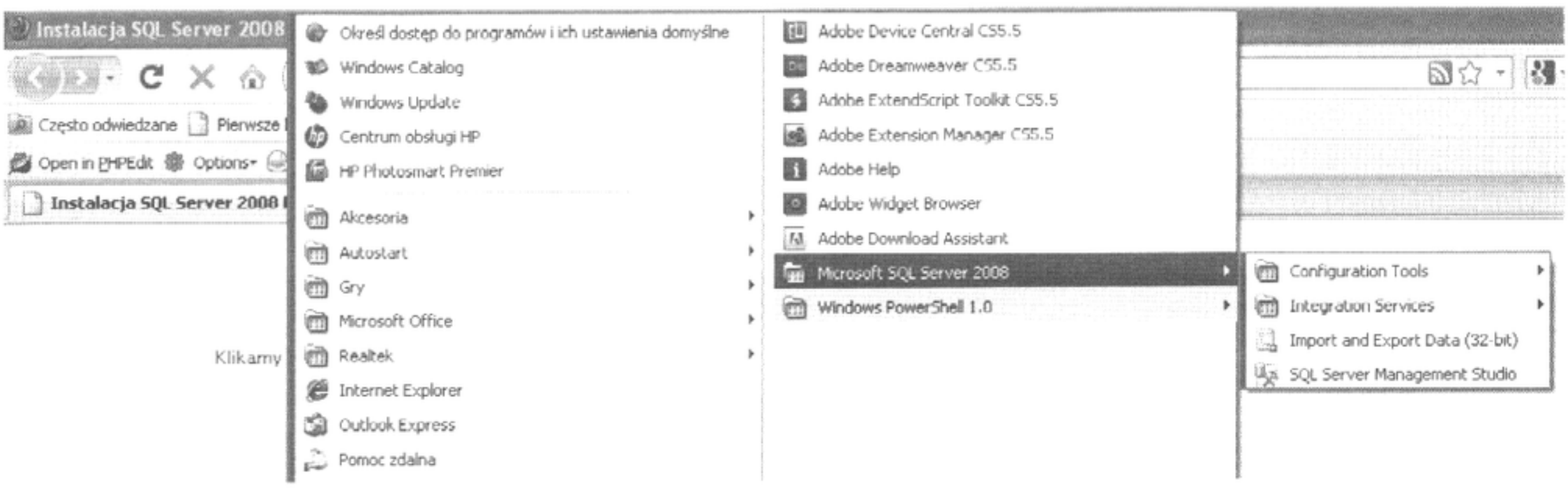
Rysunek 4.20. Instalowanie SQL Server 2008 Express Edition

Rysunek 4.21. Finał instalowania SQL Server 2008 Express Edition



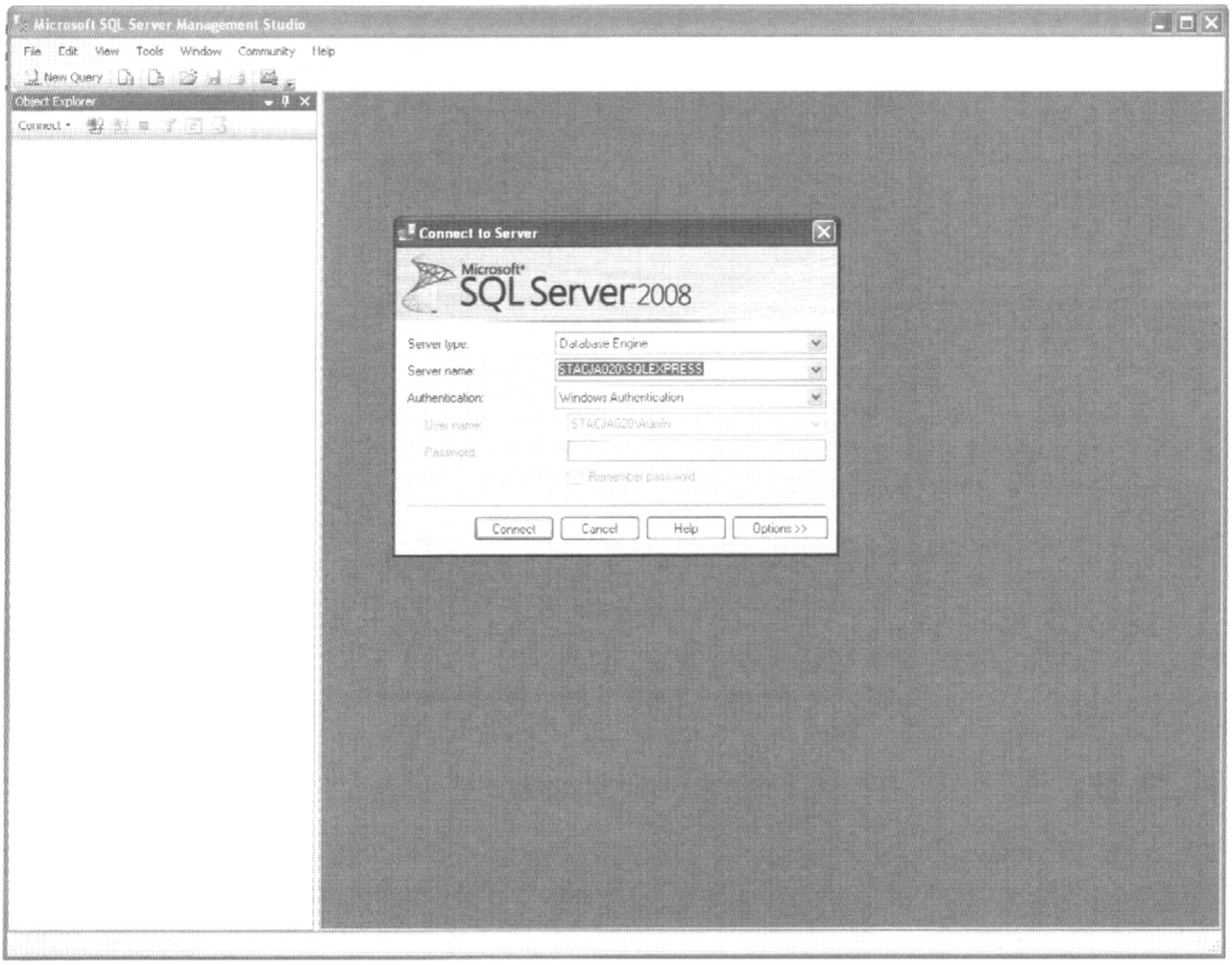
4.3.2. SQL Server Management Studio

Po zainstalowaniu serwera można uruchomić aplikację, wybierając *Start/Wszystkie programy/Microsoft SQL Server 2008/SQL Server Management Studio* (rysunek 4.22).



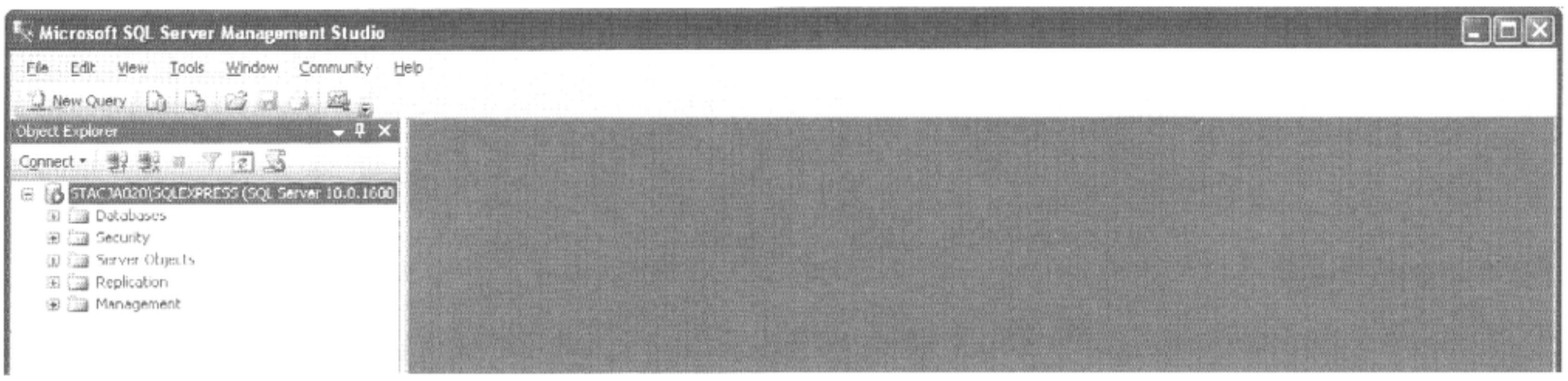
Rysunek 4.22. Uruchomienie SQL Server 2008 Express Edition

SQL Server Management Studio to bezpłatne narzędzie, za pomocą którego można na poziomie interfejsu graficznego zarządzać bazami danych SQL Server Express. Może zostać wykorzystane również do zarządzania instancjami silnika SQL Server Database Engine. Jest uzupełnieniem darmowych serwerów baz danych. Po uruchomieniu aplikacji pojawi się okno logowania do serwera; w oknie tym należy podać nazwę serwera (*Server name*) lub wybrać ją z listy (rysunek 4.23). Przy domyślnej instalacji serwera, aby połączyć się z bazą, w polu *Server name* należy wpisać *localhost\SQLEXPRESS*.



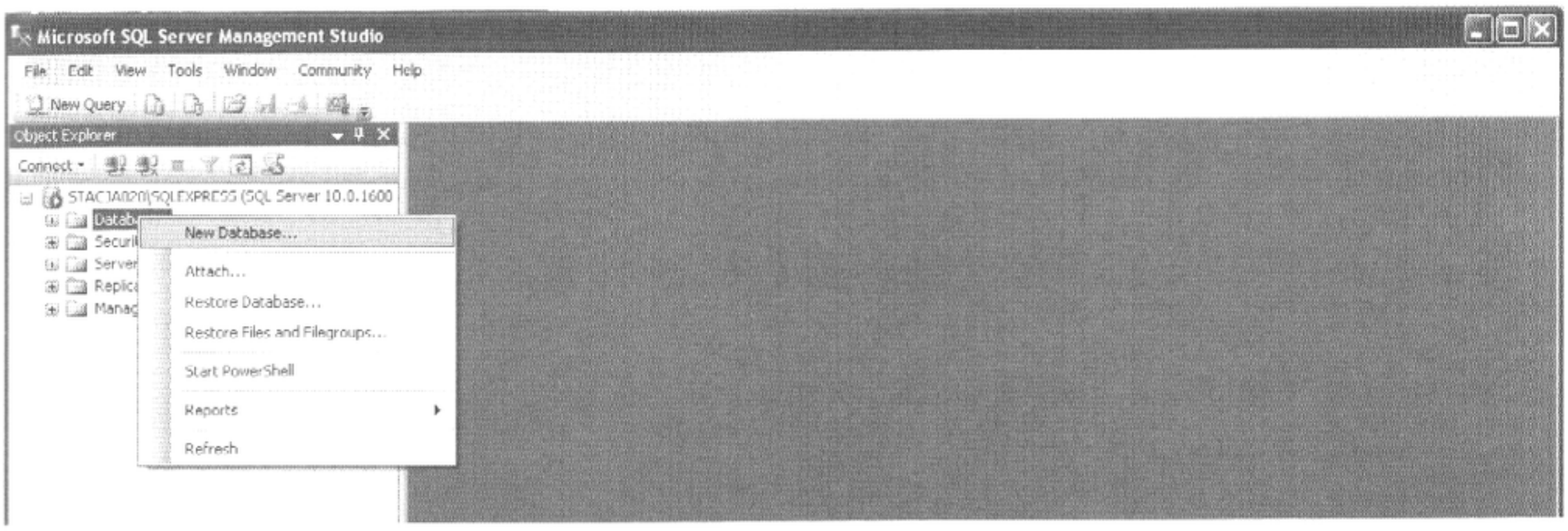
Rysunek 4.23. Logowanie do serwera baz danych

Po zalogowaniu uzyskujemy dostęp do serwera za pośrednictwem SSMS (*SQL Server Management Studio* — rysunek 4.24).

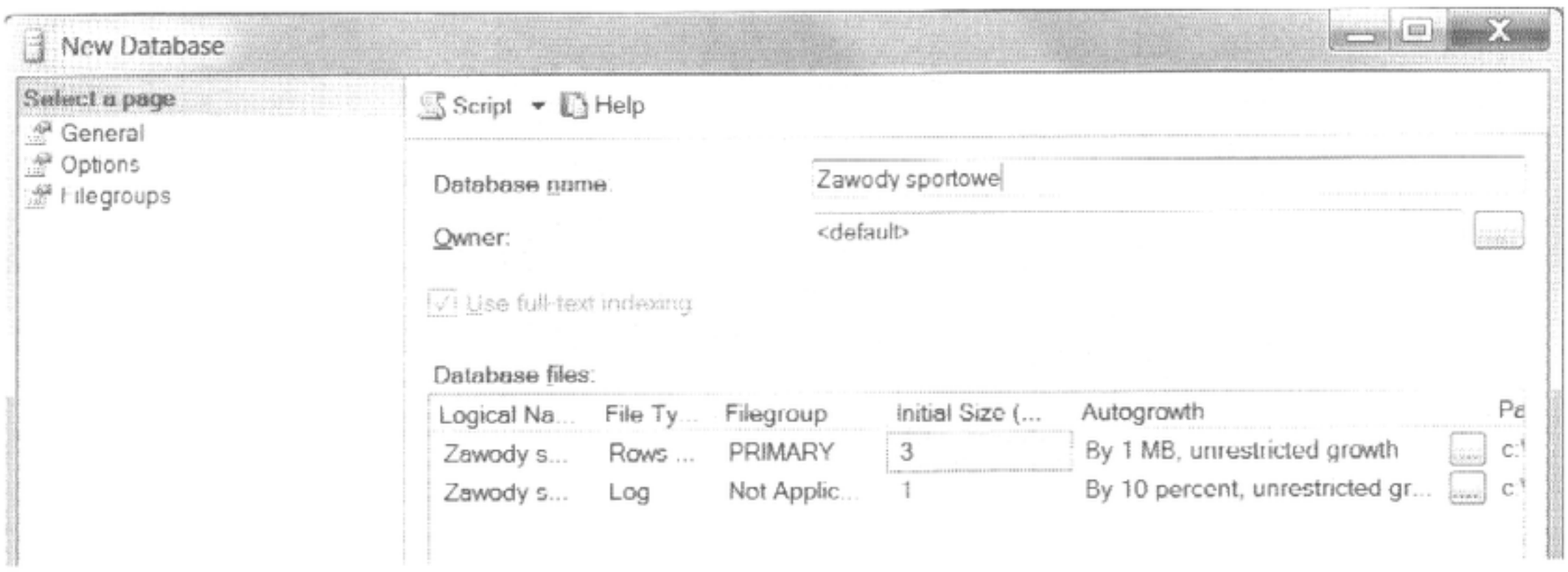


Rysunek 4.24. Okno SQL Server Management Studio

Aby rozpocząć pracę z nową bazą danych, klikamy prawym przyciskiem myszy folder *Databases* i wybieramy opcję *New Database...* (rysunek 4.25), a następnie w otwartym oknie wprowadzamy nazwę tworzonej bazy danych (rysunek 4.26).

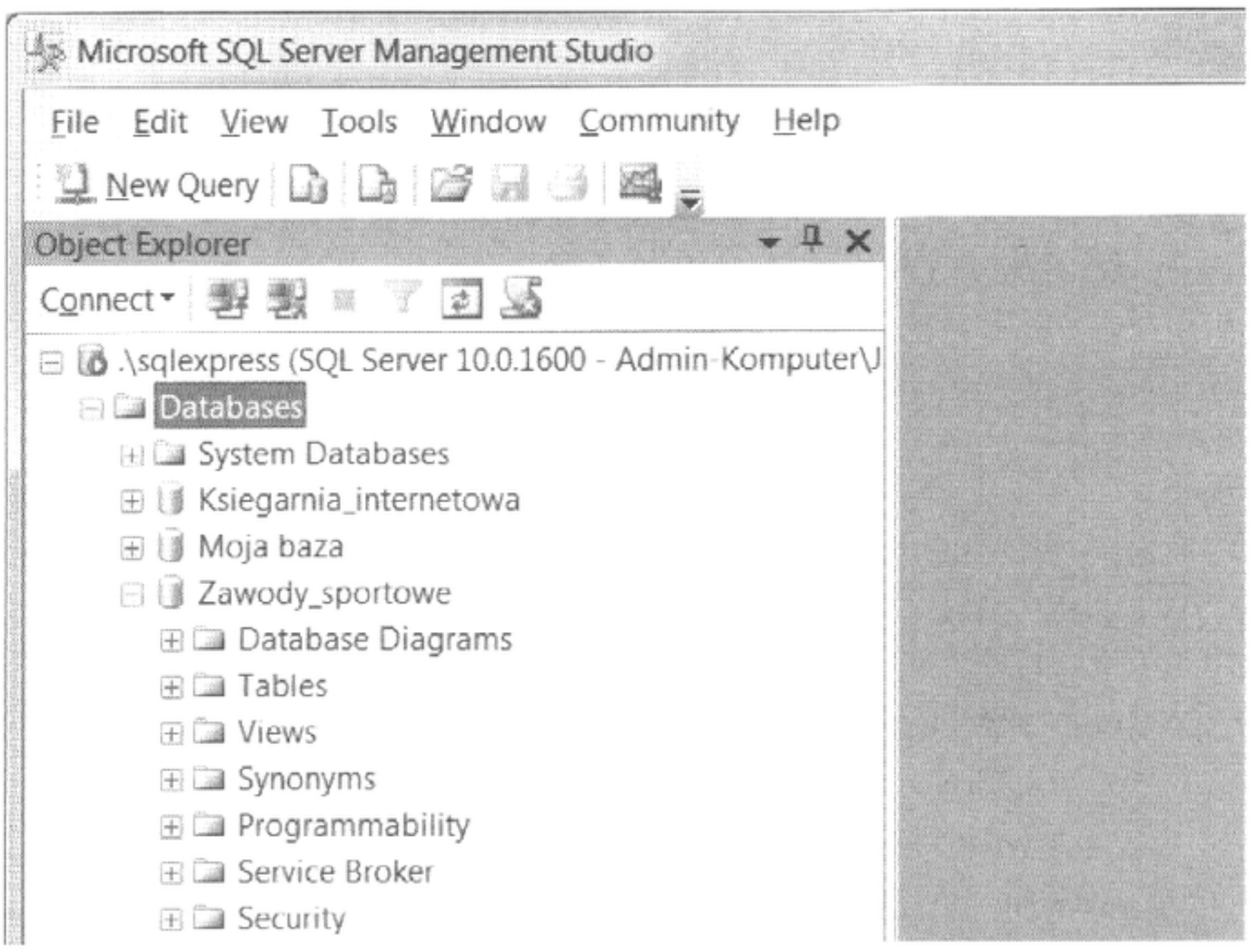


Rysunek 4.25. Opcja tworzenia nowej bazy danych



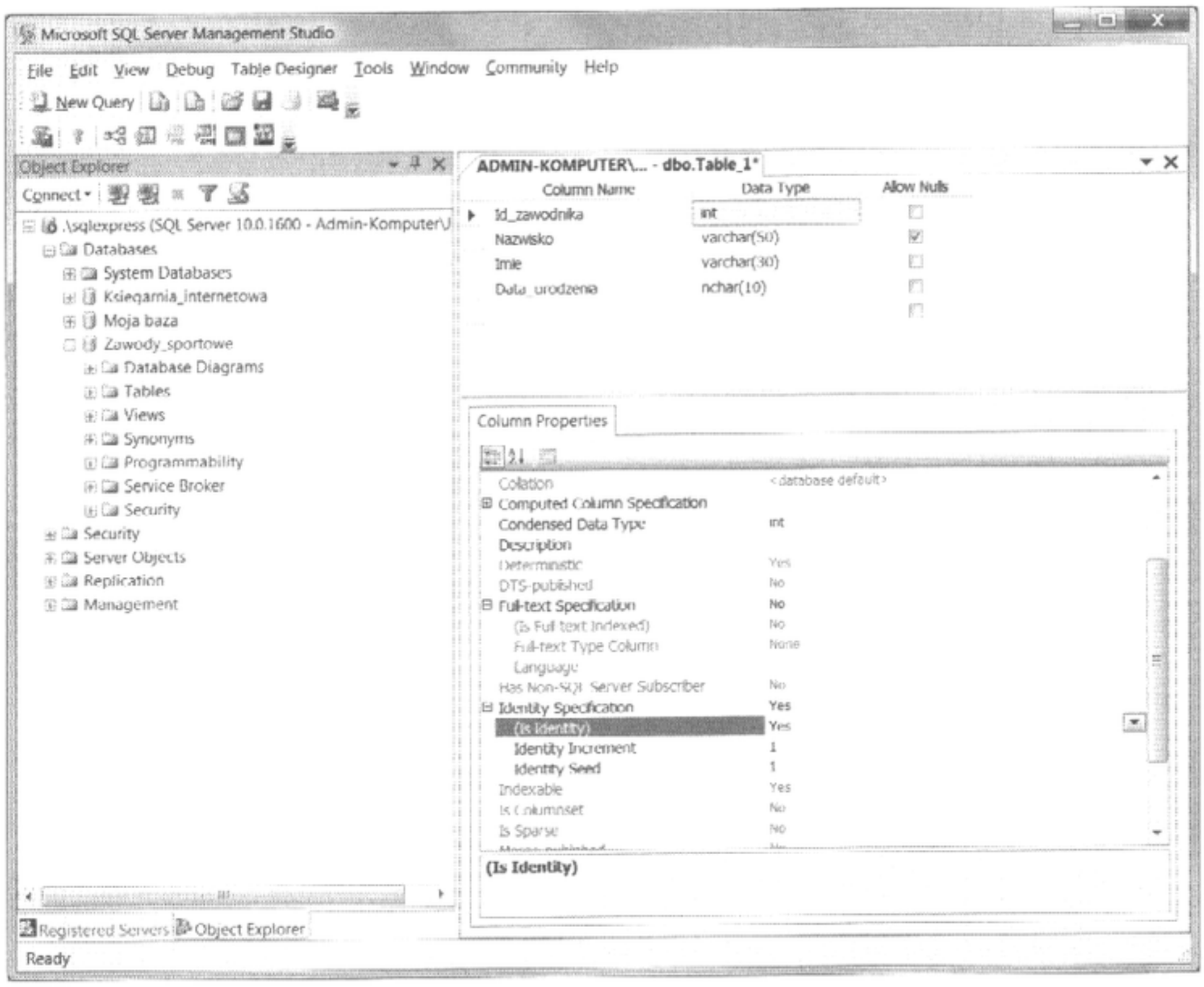
Rysunek 4.26. Tworzenie nowej bazy danych

W wyniku tych działań uzyskamy na serwerze bazodanowym pustą bazę danych (rysunek 4.27).



Rysunek 4.27. Zawartość utworzonej bazy danych

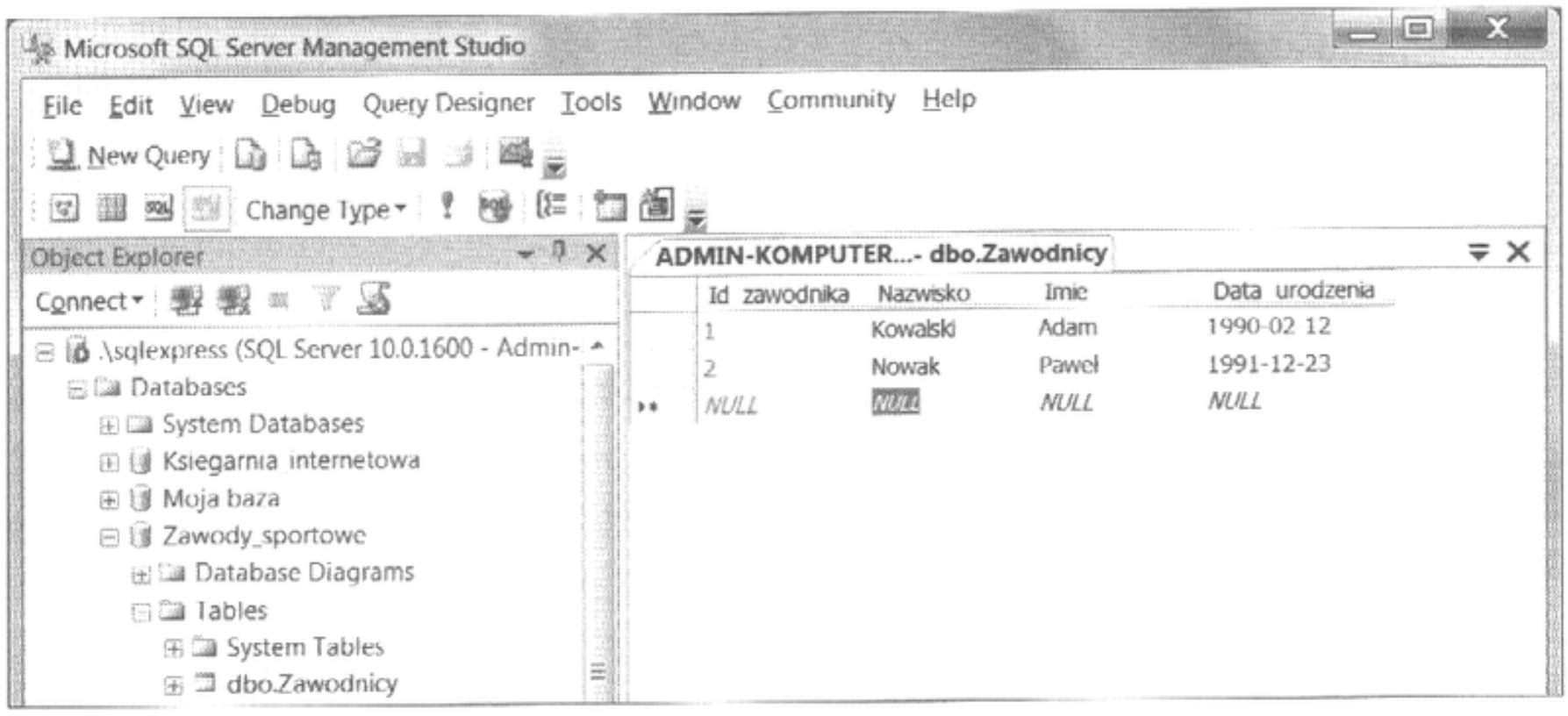
Aby utworzyć w bazie danych tabelę, należy w oknie *Object Explorer* wybrać opcję *Tables* i po jej kliknięciu prawym przyciskiem myszy wybrać z menu *New Table*.... W otwartym oknie można zaprojektować strukturę tabeli, podając nazwy kolejnych kolumn oraz typy pól (rysunek 4.28). W dolnej części okna można ustawić właściwości tworzonego pola.



Rysunek 4.28. Tworzenie nowej tabeli

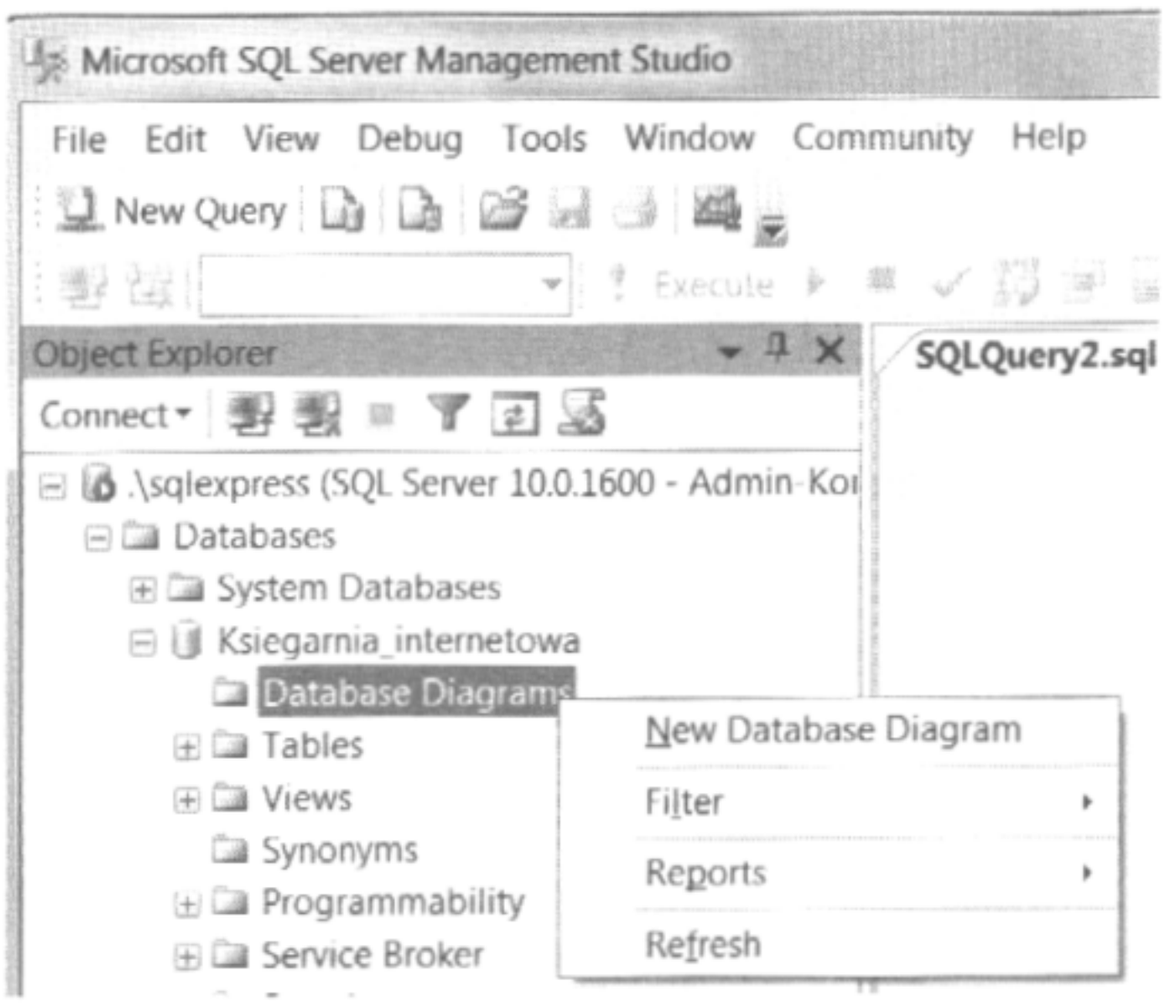
Po utworzeniu wszystkich pól tabeli, aby zapisać jej strukturę, należy wybrać z menu *File/Save Table_1*, podać nazwę tabeli i zatwierdzić jej utworzenie. W podobny sposób można tworzyć struktury kolejnych tabel.

Aby wprowadzić do tabeli dane, trzeba kliknąć prawym przyciskiem myszy wybraną tabelę i z listy wybrać opcję *Edit Top 200 Rows*. Po wybraniu tej opcji zostanie otwarte okno do wprowadzania danych do tabeli (rysunek 4.29).



Rysunek 4.29. Wprowadzanie danych do tabeli

W programie SQL Server Management Studio można tworzyć diagramy istniejącej bazy danych. W tym celu należy w wybranej bazie danych rozwinąć drzewo, wybrać opcję *Database Diagrams* i kliknąć prawym przyciskiem myszy (rysunek 4.30).



Rysunek 4.30. Tworzenie diagramu dla wybranej bazy danych

Z wyświetlonego menu trzeba wybrać opcję *New Database Diagram* i w otwartym oknie wskazać tabele, które będą użyte w diagramie. Do okna diagramu zostaną przeniesione wybrane tabele (rysunek 4.31).

Rysunek 4.31. Struktura bazy danych Księgarnia internetowa

Utworzony diagram można zapisać. Po zapisaniu diagram będzie dostępny na liście diagramów.

PYTANIA KONTROLNE

1. Wymień popularne serwery baz danych.
2. Wymień podstawowe cechy serwera MySQL.
3. Wymień podstawowe cechy serwera MS SQL Server.

ZADANIA

1. Dla bazy danych *Moja_szkoła* utwórz w programie MySQL Workbench tabele oraz schemat połączeń. Zdefiniuj klucze podstawowe oraz typy pól. Wprowadź przykładowe dane.
2. Dla bazy danych *Moja_szkoła* w programie SQL Server Management Studio utwórz tabele oraz schemat połączeń. Zdefiniuj klucze podstawowe oraz typy pól. Wprowadź przykładowe dane.

5

SQL — strukturalny język zapytań

5.1. Wprowadzenie

Język SQL (ang. *Structured Query Language*) jest to uniwersalny język zapytań stosowany w systemach relacyjnych baz danych do komunikowania się z bazą. Jest on również podstawowym językiem programowania baz danych, pozwalającym na tworzenie i modyfikowanie obiektów bazy danych (na przykład tabel).

Język SQL jest językiem **deklaratywnym**. W językach deklaratywnych definiuje się warunki, jakie musi spełniać końcowy wynik, natomiast nie definiuje się sposobu, w jaki ten wynik zostanie osiągnięty. W instrukcjach języka SQL nie znajdziemy informacji, w jaki sposób serwer bazodanowy powinien uzyskać wymagany wynik. Sposób wykonania instrukcji zależy od serwera baz danych i to zadaniem serwera jest znalezienie najlepszego sposobu osiągnięcia spodziewanego wyniku.

5.2. Standardy języka SQL

Każdy z dostępnych na rynku systemów bazodanowych zawiera specyficzne, niedostępne w innych systemach elementy. W różnych systemach mogą być stosowane różne wersje języka SQL. W celu ujednolicenia wersji języka SQL Amerykański Narodowy Instytut Standardów (ang. *American National Standards Institute* — ANSI) oraz Międzynarodowa Organizacja Normalizacyjna (ang. *International Organization for Standardization*— ISO) opracowują i publikują standardy języka SQL. W roku 1999 został przyjęty standard ANSI SQL99 (SQL3) i obecnie większość producentów systemów bazodanowych tworzy systemy zgodne z tym standardem.

Standard SQL99 nie definiuje wielu rozszerzeń języka SQL, dlatego w roku 2003 został opublikowany czwarty standard języka SQL, który jest rozszerzeniem SQL3. Zawiera on następujące rozszerzenia:

- obsługa nowych typów danych,

- rozszerzenia w sposobie wywoływania procedur,
- poszerzona instrukcja CREATE TABLE,
- funkcje rankingu,
- retrospektywne sprawdzanie więzów integralności.

Mimo zdefiniowanych standardów systemy zarządzania bazą danych nadal dodają własne rozszerzenia, ponieważ użytkownicy oczekują od tych systemów funkcji nieujętych w standardzie języka. Można jednak przyjąć, że wszystkie typowe operacje są wykonywane tak samo, niezależnie od używanego systemu zarządzania bazą danych.

Dialekty języka SQL

Najbardziej znane dialekty języka SQL to:

- T-SQL (ang. *Transact-SQL*) — stosowany na serwerach Microsoft SQL Server i Sybase Adaptive Server;
- PL/SQL (ang. *Procedural Language/SQL*) — stosowany na serwerach firmy Oracle;
- PL/pgSQL (ang. *Procedural Language/PostgreSQL Structured Query Language*) — podobna do PL/SQL wersja języka zaimplementowana na serwerze PostgreSQL;
- SQL PL (ang. *SQL Procedural Language*) — wersja używana na serwerach firmy IBM.

Terminatory SQL

W języku SQL mamy do czynienia z dwoma rodzajami terminatorów. Są to terminatory poleceń i terminatory wsadowe. Terminatory poleceń kończą każde polecenie napisane w języku SQL. Terminatory wsadowe kończą ciąg poleceń języka SQL. Powodują one przesłanie ciągu poleceń do serwera.

Terminatory poleceń najczęściej związane są z dialektami. W niektórych dialektach języka SQL, na przykład Oracle i InterBase, wymagane jest kończenie każdego polecenia średnikiem.

Terminatory wsadowe najczęściej związane są ze stosowanymi narzędziami. Na przykład edytor pakietu Sybase wymaga zakończenia ciągu poleceń słowem kluczowym GO, a w edytorze WINSQL średnik na końcu ciągu poleceń jest opcjonalny.

5.3. Składnia języka SQL

Język SQL realizuje trzy podstawowe typy zadań — definiowanie danych, manipulowanie danymi i kontrolowanie danych. W związku z tym jego instrukcje można podzielić na trzy kategorie:

- Instrukcje **DDL** (ang. *Data Definition Language*) — tworzą język definiowania danych i służą do tworzenia, modyfikowania i usuwania obiektów bazy danych.
- Instrukcje **DML** (ang. *Data Manipulation Language*) — tworzą język manipulowania danymi i służą do odczytywania i modyfikowania danych.

- Instrukcje **DCL** (ang. *Data Control Language*) — tworzą język kontroli dostępu do danych i umożliwiają nadawanie i odbieranie uprawnień użytkownikom.

UWAGA

Żaden z systemów bazodanowych nie jest w pełni zgodny ze standardem języka SQL. Na potrzeby podręcznika został wybrany SQL Server 2008 Express Edition firmy Microsoft. Przedstawiane w przykładach polecenia języka SQL w większości wykorzystują jego standardowe cechy i prawdopodobnie będą obsługiwane również w innych systemach bazodanowych.

5.3.1. Instrukcje języka SQL

Instrukcje języka SQL składają się z wyrażeń, klauzul i warunków. Wyrażenie jest poleceniem, które mówi, co należy zrobić. Klauzula określa ograniczenia dla danego polecenia i jest definiowana w formie warunków (na przykład klauzula `WHERE`).

Przykład 5.1

```
SELECT nazwisko, Imie
FROM Klient
WHERE miejscowosc="Warszawa"
```

Wyrażenie `SELECT` mówi, że trzeba wybrać z bazy dane, które zostały wymienione za poleceniem. W następnej linii zostało określone miejsce, z którego mają pochodzić dane. W ostatniej linii został zdefiniowany warunek, który musi zostać spełniony przy wybieraniu danych. Polecenie kończy się średnikiem. Taką składnię języka już poznaliśmy w rozdziale 2., definiując w programie Access kwerendy z użyciem języka SQL.

Składnia języka jest zbiorem reguł, których należy przestrzegać. W języku SQL stosuje się trzy kategorie pojęć składniowych: identyfikatory, literały i operatory.

Identyfikator jednoznacznie definiuje obiekt bazy danych. Każdy obiekt bazy danych (baza, tabela, kolumna) musi posiadać niepowtarzalną nazwę. Identyfikatory muszą być zgodne ze zdefiniowanymi w standardzie regułami:

- nie mogą być dłuższe niż 128 znaków;
- mogą zawierać litery, cyfry oraz znaki: @, $, #;
- nie mogą zawierać spacji ani innych znaków specjalnych;
- muszą zaczynać się literą;
- nie mogą być słowami kluczowymi języka SQL.

Dodatkowo zaleca się, aby nazwy były krótkie, ale jednoznacznie opisywały obiekt. Wielkość liter powinna być zgodna z przyjętymi regułami.

Literał jest stałą wartością. Wszystkie wartości liczbowe, ciągi znaków i daty, jeżeli nie są identyfikatorami, są traktowane jako stałe, czyli literały.

Typy liczbowe mają dopuszczalną postać liczby, na przykład: `150`, `-375`, `5.39`, `3E4`.

Ciągi znaków muszą być umieszczone w apostrofach, na przykład: `'Gdańsk'`, `'KOMPUTER'`.

Typy daty muszą być umieszczone w apostrofach, na przykład: `'20-09-2012'`, `'2010--02-13'`.

Operatory odgrywają rolę łączników. Ze względu na zastosowanie zostały podzielone na:

- **arytmetyczne** — suma +, różnica -, iloczyn *, iloraz /, modulo %;
- **znakowe** — konkatenacja (złączenie ciągu znaków) +, symbol zastępujący dowolny ciąg znaków %, symbol zastępujący jeden znak _;
- **logiczne** — koniunkcja `AND`, alternatywa `OR`, negacja `NOT`;
- **porównania** — =, <, >, <=, >=, <>;
- **operatory specjalne** (zależą od systemu bazodanowego) — `IN` (przynależność do listy wartości), `BETWEEN...AND...` (przynależność do domkniętego przedziału), `LIKE` (zgodność ze wzorem).

Słowa kluczowe to wyrazy zastrzeżone, interpretowane w ściśle określony sposób. W systemie baz danych mają dokładnie określone znaczenie. Do słów kluczowych należą:

- instrukcje języka SQL,
- klauzule języka SQL,
- nazwy typów danych,
- nazwy funkcji systemowych,
- terminy zarezerwowane do późniejszego użycia w systemie.

5.3.2. Typy danych

Typy danych, jak wiemy, określają rodzaj informacji przechowywanej w kolumnach tabeli. Określają również, jakiego rodzaju dane mogą być przekazywane jako parametry do procedur i funkcji, lub jakie dane mogą być przechowywane w zmiennych. Typy danych mogą różnić się nieznacznie od standardowych typów w różnych systemach baz danych, nawet jeżeli mają takie same nazwy. Możemy je podzielić według kategorii (tabela 5.1) na:

- typy liczbowe,
- typy daty i czasu,
- typy znakowe,
- typy waluty,
- typy binarne,
- typy specjalne.

Tabela 5.1. Typy danych w MS SQL Server

Kategoria	Typ danych
typy liczbowe	`int`, `smallint`, `bigint`, `tinyint`, `float`, `real`, `decimal`, `numeric`
typy daty i czasu	`datetime`, `smalldatetime`
typy znakowe	`char`, `varchar`, `nchar`, `ntext`, `nvarchar`
typy waluty	`money`, `smallmoney`
typy binarne	`binary`, `varbinary`
typy specjalne	`text`, `image`, `xml`, `bit`

Wartość NULL

`NULL` jest wartością specjalną. Oznacza wartość pustą (w komórce tabeli nie została umieszczona żadna wartość). Wartość `NULL` reprezentuje w bazie danych nieznaną lub nieistotną wartość. Jest ona różna od wartości *0* i od pustego ciągu znaków. Ze względu na to, że systemy bazodanowe muszą przetwarzać wartość `NULL` jako brakującą informację, obowiązuje w nich logika trójwartościowa. Oznacza to, że w wyniku porównania wartości `NULL` z inną wartością otrzymamy wartość nieznaną (ang. *Unknown*). Wynikiem dowolnej operacji wykonanej na wartości `NULL` jest zawsze wartość `NULL`.

5.3.3. Hierarchia obiektów bazy danych

Obiekty dostępne na serwerze bazodanowym tworzą hierarchię. Serwer zawiera wiele baz danych, baza danych może zawierać wiele schematów, w każdym schemacie może występować wiele tabel, a każda tabela może składać się z wielu kolumn. Każdy z tych obiektów powinien mieć nadaną niepowtarzalną nazwę (ang. *Alias*).

Przy odwoływaniu się w instrukcjach do obiektu pełna jego nazwa powinna zawierać następujące elementy:

```
nazwa_serwera.nazwa_bazy_danych.nazwa_schematu.nazwa_obiektu
```

W praktyce niektóre z tych elementów mogą zostać pominięte.

- Nazwa serwera wskazuje serwer bazodanowy, na którym znajduje się obiekt. Jeżeli ten element zostanie pominięty, to instrukcja zostanie wykonana przez serwer, z którym jesteśmy połączeni.
- Nazwa bazy danych wskazuje bazę danych, w której znajduje się obiekt. Jeżeli pominiemy tę nazwę, serwer wykona instrukcję w bazie danych, z którą jesteśmy połączeni.
- Nazwa schematu wskazuje schemat, w którym znajduje się obiekt. Schemat jest zbiorem powiązanych ze sobą obiektów, tworzonym w bazie danych przez użytkownika w celu usprawnienia administrowania bazami danych. Jeżeli nazwa schematu

zostanie pominięta, serwer przyjmie, że obiekt znajduje się w domyślnym schemacie użytkownika wykonującego instrukcję. Jeżeli nie zadeklarujemy schematu, domyślnym schematem użytkownika staje się schemat *dbo*.

Różni użytkownicy bazy danych mogą mieć przypisane różne schematy domyślne, dlatego podając nazwę schematu, unikniemy ewentualnych błędów. Poza tym posługiwanie się nazwami schematów skraca czas wykonania instrukcji.

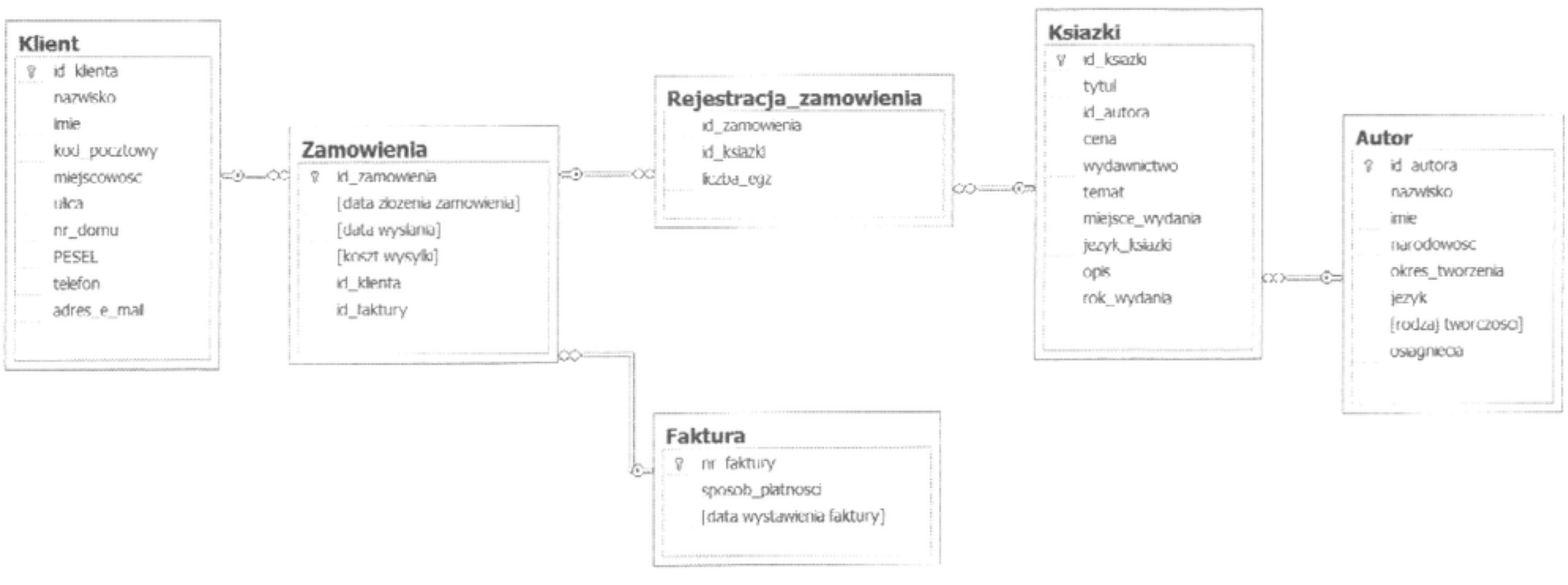
5.4. Język definiowania danych (DDL)

Język **DDL** używany jest do tworzenia, modyfikowania i usuwania bazy danych oraz jej obiektów. Instrukcje wchodzące w skład tego języka to:

- `CREATE` *nazwa_obiektu* — tworzy nowy obiekt;
- `ALTER` *nazwa_obiektu* — zmienia strukturę istniejącego obiektu;
- `DROP` *nazwa_obiektu* — usuwa istniejący obiekt.

Ponieważ instrukcje te wykorzystują indywidualne cechy używanego przez użytkownika systemu bazodanowego, ich składnia może być różna.

Pracę z bazą danych rozpoczniemy od jej utworzenia. Na potrzeby ćwiczeń w tej części podręcznika zostanie wykorzystana wcześniej zaprojektowana baza danych *Księgarnia internetowa* po niewielkich modyfikacjach (rozdział 2., przykład 1.2). Struktura bazy danych *Księgarnia internetowa* została pokazana na rysunku 5.1.



Rysunek 5.1. Baza danych Księgarnia internetowa

Tworzenie bazy danych jest realizowane za pomocą instrukcji:

```
CREATE DATABASE nazwa_bazy
```

Przykład 5.2

```
CREATE DATABASE ksiegarnia_internetowa;
```

Tak samo nieskomplikowanie wygląda polecenie usuwające bazę danych:

```
DROP DATABASE nazwa_bazy
```

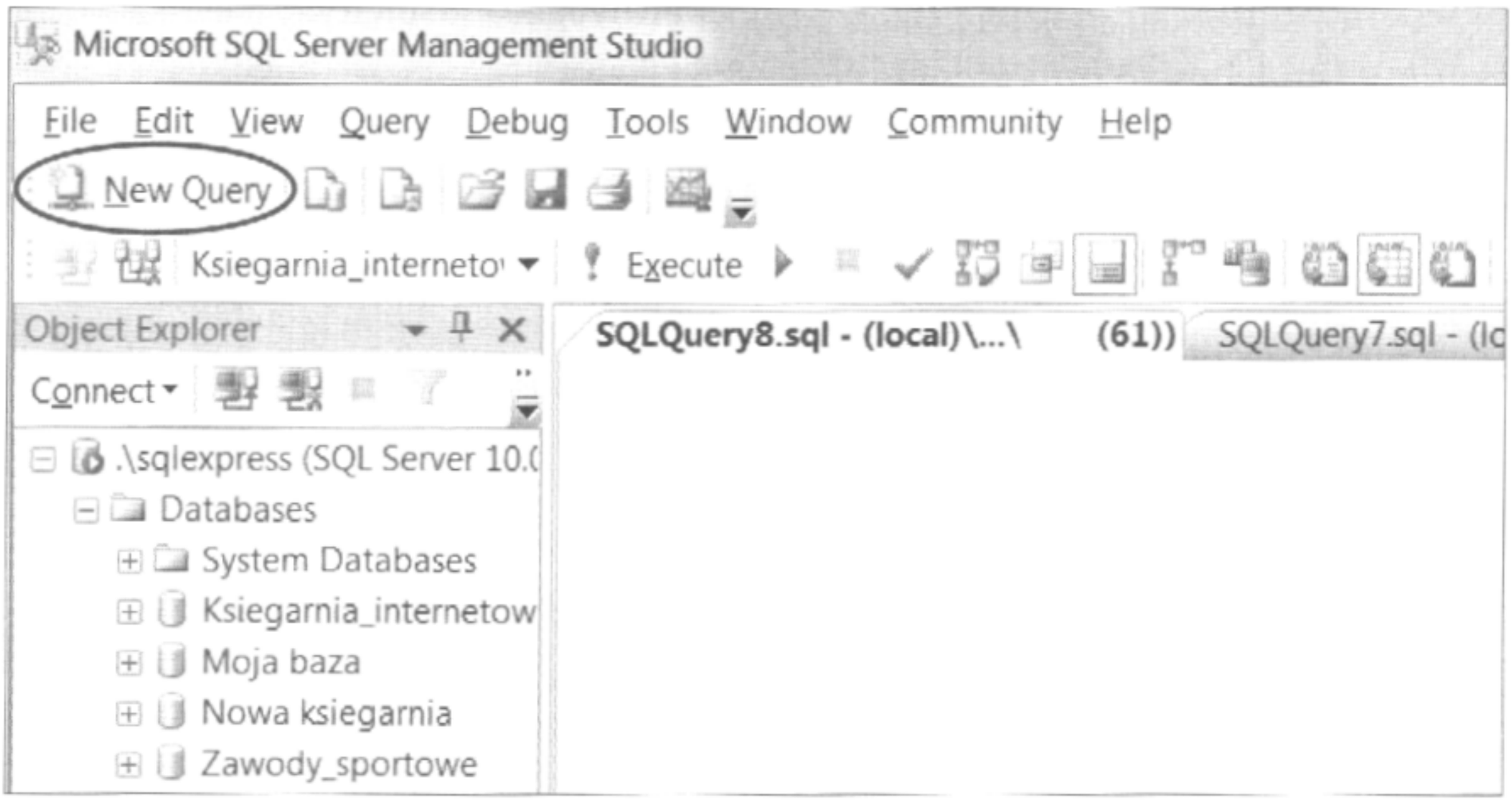
Przykład 5.3

```
DROP DATABASE ksiegarnia_internetowa;
```

Usunąć bazę danych może tylko użytkownik, który ma odpowiednie uprawnienia, na przykład administrator serwera bazodanowego lub właściciel bazy danych. W trakcie usuwania bazy nikt nie może być z nią połączony.

W praktyce te i wszystkie inne operacje można wykonać za pomocą SQL Server Management Studio bez znajomości poleceń języka SQL.

Jeżeli baza danych będzie tworzona w SQL Server Management Studio za pomocą skryptów, należy kliknąć przycisk *New Query*. Spowoduje to otwarcie nowego okna, w którym trzeba wpisać odpowiednie polecenie (rysunek 5.2).



Rysunek 5.2. Okno tworzenia zapytań dla bazy danych

Przykład 5.4

Utwórz bazę danych *ksiegarnia_internetowa*, korzystając ze skryptów języka SQI.

Klikamy przycisk *New Query* i w otwartym oknie wprowadzamy poniższy skrypt:

```
CREATE DATABASE ksiegarnia_internetowa;
```

Po wprowadzeniu kodu klikamy przycisk *Execute* (lub wybieramy z klawiatury *F5*). W wyniku wykonania skryptu zostanie utworzona baza danych *ksiegarnia_internetowa*.

5.4.1. Tworzenie i usuwanie tabel

Po utworzeniu bazy danych następnym etapem pracy jest tworzenie tabel oraz połączeń, tak aby została zbudowana cała struktura dla bazy danych.

Do tworzenia tabel służy polecenie `CREATE TABLE` w postaci:

```
CREATE TABLE nazwa_tabeli
(
    nazwa_kolumny_1 typ_kolumny_1 [atrybuty],
    nazwa_kolumny_2 typ_kolumny_2 [atrybuty],
    . . . .
    nazwa_kolumny_n typ_kolumny_n [atrybuty],
)
```

Podczas tworzenia tabel należy pamiętać, że:

- każda tabela musi mieć unikatową nazwę i unikatowego właściciela;
- każda kolumna musi mieć unikatową nazwę;
- nazwy tabel i kolumn muszą być zgodne z zasadami dotyczącymi standardów SQL;
- każda kolumna w tabeli musi mieć zdefiniowany typ;
- jeżeli kolumna jest typu znakowego, trzeba podać jej maksymalną długość;
- utworzone tabele są puste.

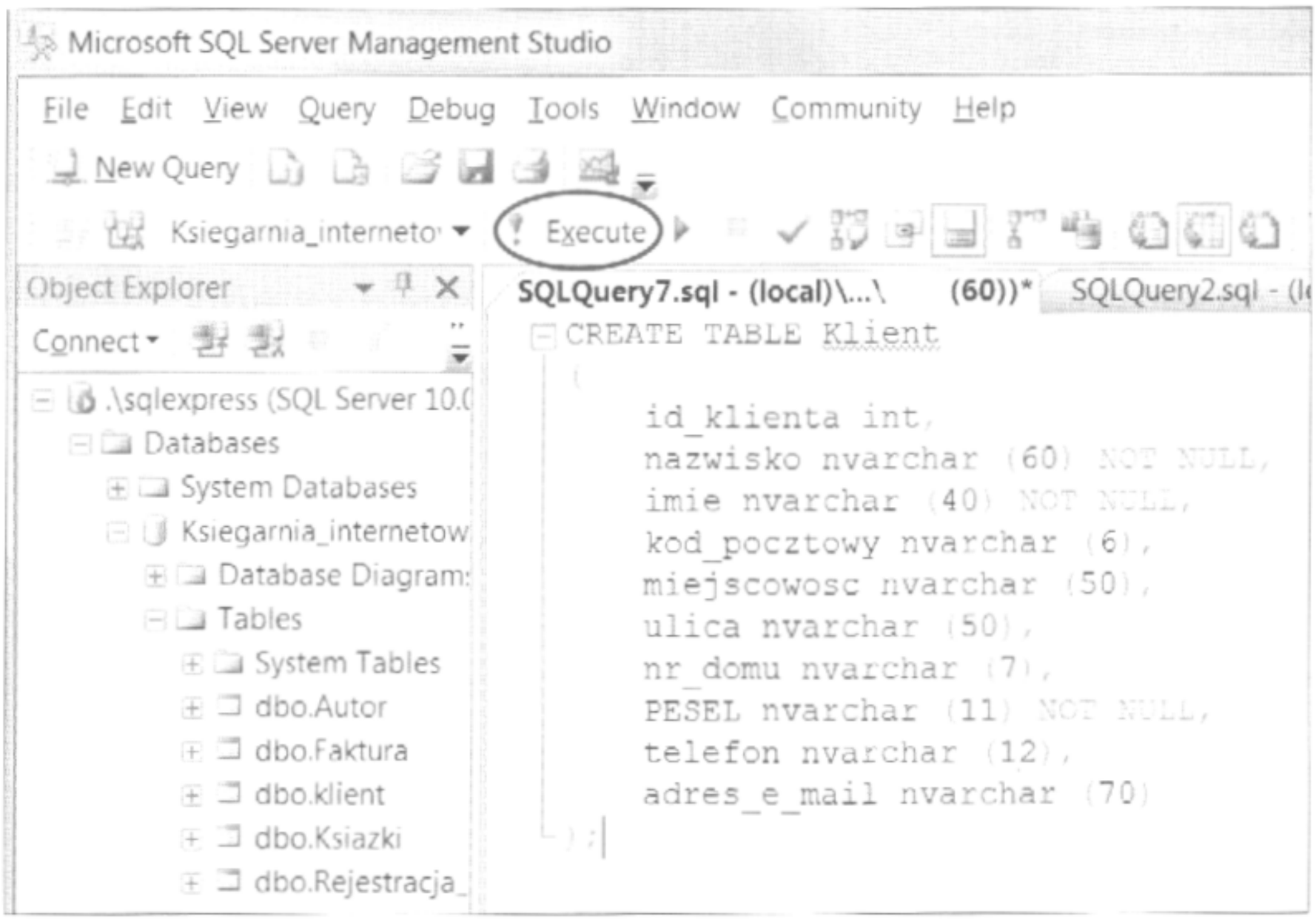
Przykład 5.5

Dla bazy danych *ksiegarnia_internetowa* utwórz tabelę *Klient* z kolumnami takimi jak widoczne na rysunku 5.1. Ustaw odpowiednie typy dla kolumn tworzonej tabeli.

Podobnie jak w przykładzie 5.4, klikamy przycisk *New Query* i w otwartym oknie wprowadzamy podany niżej skrypt:

```
USE ksiegarnia_internetowa;
CREATE TABLE Klient
(   id_klienta int,
    nazwisko nvarchar (60) NOT NULL,
    imie nvarchar (40) NOT NULL,
    kod_pocztowy nvarchar (6),
    miejscowosc nvarchar (50),
    ulica nvarchar (50),
    nr_domu nvarchar (7),
    PESEL nvarchar (11) NOT NULL,
    telefon nvarchar (12),
    adres_e_mail nvarchar (70)
);
```

Po wprowadzeniu skryptu klikamy przycisk *Execute* (lub *F5* — rysunek 5.3). W wyniku wykonania skryptu do bazy danych zostanie dodana tabela *Klient*. Na serwerze może istnieć wiele baz danych. W celu określenia, dla której bazy danych będą wykonywane kolejne polecenia, została użyta instrukcja `USE`.



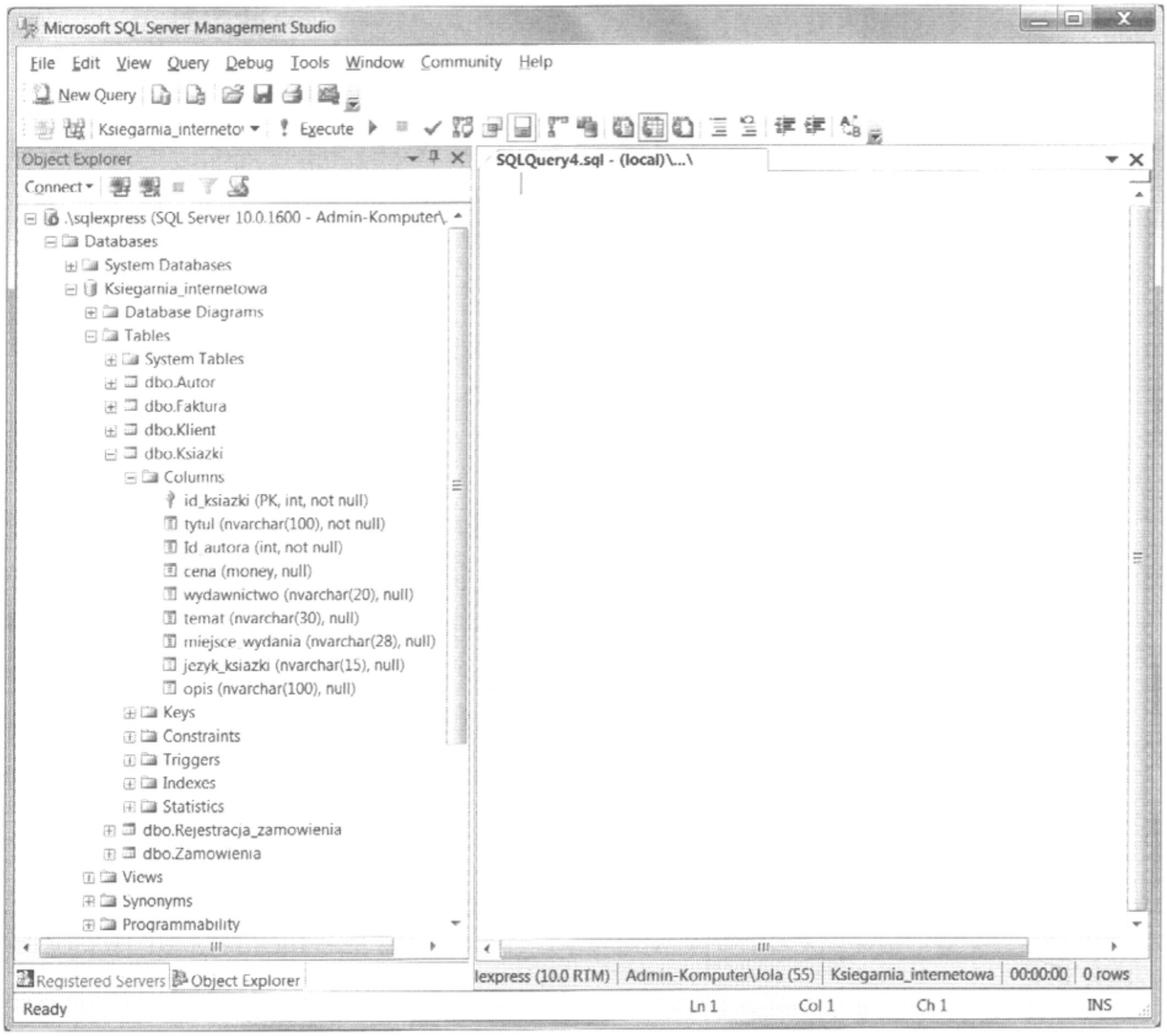
Rysunek 5.3. Definiowanie polecenia tworzącego tabelę Klient

Przykład 5.6

Utwórz tabelę *Ksiazki* z kolumnami jak na rysunku 5.1. Polecenie tworzące tabelę będzie miało postać:

```
CREATE TABLE Ksiazki
(
    id_ksiazki INT,
    tytul nvarchar (100) NOT NULL,
    autor nvarchar (80) NOT NULL,
    cena money,
    wydawnictwo nvarchar (20),
    temat nvarchar (30),
    miejsce_wydania nvarchar (28),
    jezyk_ksiazki nvarchar (15),
    opis nvarchar (100),
);
```

Rysunek 5.4 pokazuje strukturę bazy danych i tabel w oknie *Server Management Studio*.



Rysunek 5.4. Struktura bazy danych ksiegarnia_internetowa

Do usuwania tabel służy instrukcja `DROP TABLE` w następującej postaci:

```
DROP TABLE nazwa_tabeli
```

Przykład 5.7

```
DROP TABLE Klient;
```

W wyniku wykonania instrukcji z bazy danych zostanie usunięta tabela *Klient*.

5.4.2. Schematy

Baza danych może zawierać wiele tabel. Zarządzanie bazą, która zawiera kilkadziesiąt, a czasami kilkaset tabel, może być bardzo trudne. W celu usprawnienia administrowania takimi bazami danych możemy tworzyć **schematy** i przydzielać do nich obiekty bazy danych.

Obiekty bazy danych powinny być przydzielane do schematów według powiązań, jakie zachodzą między nimi. Jeżeli obiekty bazy zostaną przydzielone do schematów, to

administrator bazy danych będzie mógł zdefiniować uprawnienia użytkownika nie na poziomie poszczególnych tabel, ale na poziomie całych schematów.

Schemat jest tworzony za pomocą instrukcji `CREATE SCHEMA`, która ma postać:

```
CREATE SCHEMA nazwa_schematu
```

Przykład 5.8

```
CREATE SCHEMA Zasoby;
```

Podczas tworzenia schematu można tworzyć tabele, widoki, definiować prawa dostępu. Obiekty, które są tworzone instrukcją `CREATE SCHEMA`, są umieszczane wewnątrz definiowanego schematu.

Przykład 5.9

```
CREATE SCHEMA Magazyn
CREATE TABLE Ksiazki
(   id_ksiazki INT,
    tytul nvarchar (100) NOT NULL,
    cena money,
    wydawnictwo nvarchar (20));
CREATE TABLE Autor
(   id_autora INT,
    nazwisko nvarchar (100) NOT NULL,
  imie nvarchar (30) NOT NULL);
```

Przypisanie obiektu do schematu może nastąpić na dwa sposoby: jawnie i niejawnie. Aby jawnie przypisać tabelę do schematu, należy poprzedzić jej nazwę nazwą schematu:

```
CREATE TABLE nazwa_schematu.nazwa_tabeli
```

Każdy obiekt tworzony w bazie danych należy do jakiegoś schematu. Jeżeli podczas tworzenia tabeli nie przypiszemy jej jawnie do schematu, zostanie ona domyślnie umieszczona w schemacie, w którym obecnie pracujemy. Najprawdopodobniej będzie to schemat *dbo*. Ale gdyby zalogowany użytkownik znajdował się domyślnie w schemacie *Zasoby*, to tabela zostałaby dodana do schematu *Zasoby*. Niejawne przypisywanie do schematów nie jest zalecane.

Przykład 5.10

```
CREATE TABLE Zasoby.ksiazki
(
    id_ksiazki INT,
    tytul nvarchar (100) NOT NULL,
```

```
    autor nvarchar (80) NOT NULL,
    cena money,
    wydawnictwo nvarchar (20),
    temat nvarchar (30),
    miejsce wydania nvarchar (28),
    jezyk ksiazki nvarchar (15),
    opis nvarchar (100)
);
```

Schemat może mieć tylko jednego właściciela, ale jeden użytkownik może mieć wiele schematów. Każdy użytkownik ma zdefiniowany domyślny schemat, który może zostać zmieniony poleceniem:

```
DEFAULT_SCHEMA CREATE USER
```

lub:

```
ALTER USER
```

Jeżeli domyślny schemat nie został zdefiniowany, użytkownikowi zostanie przypisany schemat *dbo*.

5.4.3. Zmiana struktury tabeli

Zmiana struktury tabeli może polegać na dodaniu kolumny, usunięciu kolumny, dodaniu atrybutu lub usunięciu atrybutu. Do modyfikowania struktury tabeli służy polecenie:

```
ALTER TABLE nazwa_tabeli zmiana
```

Przykład 5.11

Dodanie kolumny do tabeli *Ksiazki*:

```
ALTER TABLE Ksiazki
ADD liczba_stron nvarchar (5);
```

Przykład 5.12

Zmiana definicji istniejącej kolumny w tabeli *Ksiazki*:

```
ALTER TABLE Ksiazki
ALTER COLUMN rok_wydania nvarchar (4) NOT NULL;
```

Przykład 5.13

Usunięcie kolumny z tabeli *Ksiazki*:

```
ALTER TABLE Ksiazki
DROP COLUMN liczba_stron;
```

Przykład 5.14

Usunięcie z tabeli *ksiegarnia_internetowa* kolumny *autor* i dodanie kolumny *id_autora*:

```
Use ksiegarnia_internetowa;
GO
ALTER TABLE Ksiazki
DROP COLUMN Autor;
ALTER TABLE Ksiazki
ADD id_autora int NOT NULL;
```

5.4.4. Atrybuty kolumn

Każda kolumna tabeli może mieć zdefiniowane za pomocą atrybutów ograniczenia, które określają, jakie dane mogą zostać w niej zapisane. Ograniczenia dotyczące kolumn mogą być definiowane w trakcie tworzenia tabeli lub w trakcie jej modyfikowania.

PRIMARY KEY

Klucz podstawowy (*Primary Key*) to kolumna lub kombinacja kolumn, które w sposób jednoznaczny definiują wiersz w tabeli.

Do określenia, która kolumna tabeli będzie kluczem podstawowym, stosuje się atrybut `PRIMARY KEY`. Kolumna z tym atrybutem jest unikatowa i automatycznie indeksowana.

Przykład 5.15

Definiowanie klucza podstawowego podczas tworzenia tabeli *Zamowienia*:

```
CREATE TABLE Zamowienia
(
    id_zamowienia int PRIMARY KEY,
    id_klienta int NOT NULL
    [data zlozenia zamowienia] datetime,
    [data wysłania] datetime,
    [koszt wysylki] money,
    id_faktury int
);
```

NOT NULL

Atrybut `NOT NULL` oznacza, że w kolumnie nie mogą wystąpić wartości puste. Aby zabronić wstawiania do kolumny wartości `NULL`, podczas tworzenia tabeli należy po nazwie kolumny wpisać `NOT NULL`.

Można również jawnie zezwolić na wprowadzanie do kolumny wartości NULL, wpisując po jej nazwie słowo NULL.

Przykład 5.16

Blokowanie wartości NULL podczas tworzenia tabeli *Ksiazki*:

```
CREATE TABLE Ksiazki
(
    id_ksiazki int NOT NULL PRIMARY KEY,
    tytul nvarchar (100) NOT NULL,
    ....
);
```

IDENTITY

Atrybut IDENTITY oznacza automatyczny wzrost wartości w kolumnie, dla której został zdefiniowany. Na przykład IDENTITY (1,1) oznacza wzrost wartości kolumny o 1, począwszy od wartości 1. Niemożliwe jest nadawanie atrybutu IDENTITY istniejącej kolumnie.

Przykład 5.17

Definiowanie ograniczeń dla kolumn podczas tworzenia tabeli *Autor*:

```
CREATE TABLE Autor
(
    id_autora INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    nazwisko nvarchar (50) NOT NULL,
    imie nvarchar (30) NOT NULL,
    narodowosc nvarchar (30),
    okres_tworzenia nvarchar (35),
    jezyk nvarchar (30),
    [rodzaj tworczosci] nvarchar (35),
    osiagniecia nvarchar (100));
```

DEFAULT

Atrybut DEFAULT jest stosowany do wprowadzania do kolumny wartości domyślnej.

Przykład 5.18

Definiowanie wartości domyślnej podczas tworzenia tabeli *Ksiazki*:

```
CREATE TABLE Ksiazki
(
    ....
    rok_wydania nvarchar (4) NOT NULL DEFAULT '2012'
    ....
);
```

UNIQUE

Atrybut `UNIQUE` jest stosowany, jeśli wartości w kolumnie nie mogą się powtarzać. Ograniczenie powtarzalności w kolumnie nie blokuje możliwości wpisania do niej wartości `NULL`.

Przykład 5.19

Definiowanie wartości unikatowych podczas tworzenia tabeli *Ksiazki*:

```
CREATE TABLE Ksiazki
(
    ....
    tytul nvarchar (100) NOT NULL UNIQUE,
    ....
);
```

Warunek logiczny CHECK

Atrybut `CHECK` pozwala na zdefiniowanie warunków ograniczających zakres danych wprowadzanych do kolumny. Dla każdej kolumny można definiować wiele warunków. Można również tworzyć za pomocą operatorów `NOT`, `AND` i `OR` złożone warunki ograniczające.

Przykład 5.20

Definiowanie ograniczeń podczas tworzenia tabeli *Ksiazki*:

```
CREATE TABLE Ksiazki
(
    . . . .
    rok_wydania int CHECK (BETWEEN 2010 AND 2014)
    . . . .
);
```

Przykład 5.21

Definiowanie atrybutów ograniczających podczas tworzenia tabeli *Klient*:

```
CREATE TABLE Klient
(
    id_klienta int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    nazwisko nvarchar (60) NOT NULL,
    imie nvarchar (40) NOT NULL,
    kod_pocztowy nvarchar (6),
    miejscowosc nvarchar (50) DEFAULT 'Warszawa',
    ulica nvarchar (50),
    nr_domu nvarchar (7),
    PESEL nvarchar (11) NOT NULL,
    telefon nvarchar (12) UNIQUE,
    adres_e_mail nvarchar (70)
);
```

Przykład 5.22

Definiowanie atrybutów ograniczających podczas tworzenia tabeli *Ksiazki*:

```
CREATE TABLE Ksiazki
(
    id_ksiazki int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    tytul nvarchar (100) NOT NULL,
    id_autora int NOT NULL,
    cena money,
    wydawnictwo nvarchar (20),
    temat nvarchar (30),
    miejsce_wydania nvarchar (28),
    jezyk_ksiazki nvarchar (15),
    opis nvarchar (100),
);
GO
ALTER TABLE Ksiazki
ADD rok_wydania nvarchar (4) NOT NULL DEFAULT '2012';
```

Zadanie 5.1

W bazie danych *ksiegarnia_internetowa* utwórz tabele zgodnie ze schematem zamieszczonym na rysunku 5.1. Określ dopuszczalne wartości w kolumnach tabel i zdefiniuj klucze podstawowe.

5.5. Manipulowanie danymi (DML)

Jednym z zadań realizowanych przez język SQL jest pobieranie i modyfikowanie danych. Instrukcje do tego służące tworzą tak zwany język manipulacji danymi (ang. *Data Manipulation Language* — **DML**). Są to:

- `SELECT` — wybiera dane z bazy danych;
- `INSERT` — umieszcza nowe wiersze w tabeli;
- `UPDATE` — zmienia zawartość istniejącego wiersza;
- `DELETE` — usuwa wiersze z tabeli.

5.5.1. Instrukcja INSERT

Instrukcja `INSERT` służy do wstawiania nowych wierszy do tabeli i ma postać:

```
INSERT INTO nazwa_tabeli (kolumna1, kolumna2, ... )
VALUES (wartość1, wartość2, ...)
```

W wyniku wykonania instrukcji do tabeli zostanie dodany nowy wiersz. W polu *kolumna1* zostanie zapisana wartość *wartość1*, w polu *kolumna2* zostanie zapisana wartość *wartość2* itd.

Jeżeli jawnie nie podamy, do jakich kolumn powinny zostać wstawione wartości, to dane podane w klauzuli `VALUES` zostaną wstawione do kolejnych kolumn tabeli.

Przykład 5.23

```
INSERT INTO Ksiazki (tytul, id_autora, cena, wydawnictwo, temat)
VALUES ('Projektowanie stron internetowych', 1, 35, 'Helion', 'Internet');
```

Po wykonaniu tego polecenia do tabeli *Ksiazki* zostanie dodany nowy wiersz, a w polach: *tytul*, *id_autora*, *cena*, *wydawnictwo*, *temat* zostaną wstawione podane ciągi znaków.

Jeżeli wstawiony wiersz odczytamy, okaże się, że oprócz wymienionych pól została wstawiona również wartość dla pola *id_ksiazki*, ponieważ podczas definiowania tabeli pole to zostało wybrane jako pole klucza podstawowego z automatycznym wstawianiem kolejnych wartości, poczynając od liczby 1 (`id_ksiazki int IDENTITY (1, 1) NOT NULL PRIMARY KEY`). Jeżeli dla jakiejś kolumny została zdefiniowana wartość domyślna (klauzula `DEFAULT`), to podobnie zostanie ona automatycznie wstawiona do niej, chyba że podczas wstawiania wiersza podamy jej wartość.

Jeżeli podczas definiowania tabeli dla określonej kolumny zostało zabronione zapisywanie wartości nieznanej (NULL), to próba zapisania nowego wiersza bez podania wartości dla tej kolumny zakończy się niepowodzeniem.

Ze względu na to, że pojedyncze wstawianie wierszy jest uciążliwe, niektóre serwery pozwalają na wpisanie w klauzuli VALUES wartości, które zostaną umieszczone w kilku wierszach.

Przykład 5.24

```
INSERT INTO Ksiazki (tytul, id_autora, cena, wydawnictwo, temat)
VALUES ('Aplikacje internetowe', 2, 57, 'Helion', 'Internet'),
('Programowanie w PHP', 2, 72, 'Helion', 'Internet'),
('SQL Server 2008', 3, 45, 'PWN', 'Bazy danych');
```

W wyniku wykonania instrukcji do tabeli *Ksiazki* zostaną dodane trzy nowe wiersze z podanymi wartościami.

Przykład 5.25

```
INSERT INTO Klient (nazwisko, imie, PESEL)
VALUES ('Nowak', 'Andrzej','78021203121'),
('Kowalski', 'Jan','81092902821'),
('Górski', 'Antoni','89120217239');
```

W wyniku wykonania instrukcji do tabeli *Klient* zostaną dodane trzy nowe wiersze z danymi klientów.

5.5.2. Instrukcja UPDATE

Do aktualizowania danych służy instrukcja UPDATE w postaci:

```
UPDATE ... SET ...
```

W klauzuli UPDATE podajemy nazwę tabeli, a w klauzuli SET nazwę modyfikowanej kolumny oraz przypisaną jej nową wartość. Warto pamiętać, że wykonanie takiej instrukcji spowoduje zmianę we wszystkich wierszach podanej kolumny. Dlatego jeżeli modyfikacja będzie dotyczyła tylko wybranych wierszy, do instrukcji należy dołączyć klauzulę WHERE.

Przykład 5.26

```
UPDATE Ksiazki SET [jezyk ksiazki] ='polski'
WHERE [rok wydania]>=2012;
```

W podanym przykładzie dla tabeli *Ksiazki* kolumnie *jezyk ksiazki* zostanie przypisana wartość *polski*, ale tylko dla książek wydanych w roku 2012 lub później.

Przy użyciu instrukcji UPDATE możliwe jest modyfikowanie wielu kolumn jednocześnie. Jest to zalecane rozwiązanie, ponieważ modyfikacja wielu kolumn za pomocą jednej instrukcji jest dużo bardziej wydajna niż modyfikowanie pojedynczych kolumn z wykorzystaniem kilku operacji.

Przykład 5.27

```
UPDATE Klient SET kod_pocztowy ='87-100', miejscowosc ='Toruń',
ulica='Szeroka', nr_domu=34
WHERE nazwisko='Nowak' AND imie='Andrzej';
```

W podanym przykładzie dla tabeli *Klient* zostaną zmodyfikowane kolumny *kod_pocztowy*, *miejscowosc*, *ulica* i *nr_domu*, dla klienta *Nowak Andrzej*.

5.5.3. Instrukcja DELETE

Instrukcja DELETE usuwa wiersze z wybranej tabeli.

Aby usunąć wszystkie wiersze z tabeli, wystarczy w klauzuli FROM podać nazwę tabeli.

Przykład 5.28

Usunięcie wszystkich wierszy z tabeli *Klient*:

```
DELETE FROM Klient;
```

Po dodaniu klauzuli WHERE ze zdefiniowanym warunkiem z tabeli zostaną usunięte te wiersze, dla których warunek będzie prawdziwy.

Przykład 5.29

Usunięcie z tabeli *Klient* klientów mieszkających w Opolu:

```
DELETE FROM Klient
WHERE miejscowosc = 'Opole';
```

Jeżeli usunięcie danych mogłoby spowodować utratę spójności danych, instrukcja nie zostanie wykonana.

5.5.4. Instrukcja SELECT

Instrukcja SELECT określa, jakie dane zostaną zwrócone w wyniku jej wykonania. Ogólną postać instrukcji SELECT definiującej kwerendę wybierającą w programie Access poznaliśmy w rozdziale 3. Jej najprostsza postać dla serwera SQL Server wygląda następująco:

```
SELECT [ALL | DISTINCT] [TOP n [PERCENT] ]
{nazwa_kolumny | wyrażenie | IDENTITYCOL | ROWGUIDCOL} [[AS] nazwa_kolumny]
| nazwa_kolumny = wyrażenie } [, ... n]
FROM Nazwa_tabeli
```

gdzie:

`IDENTITYCOL` zwraca wartość kolumny *IDENTITY*,

`ROWGUIDCOL` zwraca wartość globalnego identyfikatora.

Przykład 5.30

```
SELECT nazwisko, imie
FROM Klient;
```

W podanym przykładzie instrukcja `SELECT` zwróci zawartość pól *nazwisko* i *imie* z tabeli *Klient*.

Wybieranie niektórych kolumn z tabeli nazywane jest **projekcją** lub **selekcją pionową** tabeli.

W poleceniu `SELECT` zamiast wypisywania listy wszystkich pól tabeli można użyć symbolu `*`.

Przykład 5.31

Chcemy otrzymać wszystkie kolumny tabeli *Klient*.

```
SELECT *
FROM Klient;
```

Może się zdarzyć, że w wyniku zastosowania symbolu * otrzymamy błędne wyniki. Sytuacja taka ma miejsce, gdy ktoś inny zmienia kolejność kolumn lub dodaje kolumnę do tabeli albo ją usuwa. Jeżeli chcemy odczytać zawartość tylko niektórych kolumn, nie powinniśmy definiować klauzuli odczytującej całą tabelę.

Klauzula DISTINCT

Klauzula `DISTINCT` eliminuje z wyświetlania wyniku zapytania powtarzające się wiersze.

Przykład 5.32

Chcemy otrzymać informację o tym, z jakich miejscowości pochodzą klienci naszej bazy danych.

```
SELECT DISTINCT kod_pocztowy, miejscowosc
FROM Klient;
```

Wyrażenia w instrukcji SELECT

W instrukcji `SELECT` oprócz nazw kolumn mogą występować wyrażenia. Tworzone są one z nazw kolumn, funkcji systemowych, stałych i operatorów i muszą zwracać pojedyncze wartości.

Przykład 5.33

Chcemy obliczyć marżę narzuconą na książki. Marża wynosi 7% ceny książki.

```
SELECT tytul, cena, cena*0.07 AS [Marża]
FROM Ksiazki;
```

Wynik zostanie obliczony dla wszystkich wierszy tabeli *Ksiazki*.

Przykład 5.34

Chcemy uzyskać ostateczną cenę książki, uwzględniającą marżę.

```
SELECT tytul, cena+cena*0.07 AS [Cena sprzedaży]
FROM Ksiazki;
```

Przykład 5.35

Chcemy połączyć dane klienta z kilku kolumn tabeli *Klient*.

```
SELECT nazwisko+' '+imie AS Klient, kod_pocztowy+' '+miejscowosc AS Miasto,
ulica+' '+nr_domu AS Adres
FROM Klient;
```

Sortowanie

W celu posortowania danych należy dodać klauzulę `ORDER BY`. W klauzuli umieszcza się nazwy lub numery kolumn, według których nastąpi sortowanie.

Przykład 5.36

```
SELECT tytul, cena
FROM Ksiazki
ORDER BY tytul;
```

Książki zostaną posortowane według kolumny *tytul*.

Inny przykład:

```
SELECT tytul, cena
FROM Ksiazki
ORDER BY 2;
```

Książki zostaną posortowane według kolumny *cena*.

Domyślnie dane są sortowane rosnąco. Do tego rodzaju sortowania można użyć również słowa kluczowego `ASC`. Aby posortować dane malejąco, należy po nazwie lub jej numerze użyć słowa kluczowego `DESC`.

Przykład 5.37

```
SELECT tytul, cena
FROM Ksiazki
ORDER BY cena DESC;
```

Książki zostaną ustawione w kolejności od najdroższej do najtańszej.

Dane mogą być sortowane według wielu kolumn. Klauzule definiujące sposób sortowania są od siebie niezależne. Można również odwoływać się do kolumn niewymienionych w klauzuli `SELECT`.

Przykład 5.38

```
SELECT tytul, rok_wydania, cena
FROM Ksiazki
ORDER BY rok_wydania ASC, cena DESC;
```

Sortowanie zwykle wydłuża czas wykonywania zapytania, dlatego jeżeli wynik zapytania nie musi być posortowany, należy unikać używania klauzuli `ORDER BY`.

Wybieranie wierszy — klauzula WHERE

Projektując zapytanie, najczęściej chcemy ograniczyć wynik do interesujących nas danych. Do zwracania tylko wybranych wierszy służy poznana już klauzula `WHERE`.

Wybieranie niektórych wierszy z tabeli nazywane jest **selekcją rekordów**.

Klauzula `WHERE` musi wystąpić bezpośrednio po klauzuli `FROM`.

Przykład 5.39

```
SELECT tytul, cena
FROM Ksiazki
WHERE rok_wydania=2012;
```

lub:

```
SELECT tytul, cena
FROM Ksiazki
WHERE cena BETWEEN 50 AND 100;
```

Przykład 5.40

Chcemy odczytać nazwiska klientów zaczynające się na literę *A*.

```
SELECT nazwisko, imie
FROM Klient
WHERE nazwisko LIKE'A%';
```

Przykład 5.41

Chcemy odczytać nazwiska klientów, którzy nie podali swojego adresu e-mail.

```
SELECT nazwisko, imie
FROM Klient
WHERE adres_e_mail IS NULL;
```

Przykład 5.42

Złożony warunek logiczny:

```
SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc ='Warszawa' AND adres_e_mail IS NULL;
```

W wyniku zostaną odczytane nazwiska i imiona klientów, którzy mieszkają w *Warszawie* i nie podali swojego adresu *e_mail*.

Klauzula `WHERE` może być używana na podobnych zasadach w instrukcjach `UPDATE` oraz `DELETE`. Składnia jej jest we wszystkich przypadkach taka sama.

Klauzula TOP

Klauzula `TOP` służy do wybrania określonej liczby wierszy. Liczba wierszy może być podana jawnie lub procentowo. Klauzula `TOP` musi wystąpić bezpośrednio po instrukcji `SELECT` przed nazwami kolumn.

Przykład 5.43

```
SELECT TOP 1 tytul, wydawnictwo, rok_wydania, cena
FROM Ksiazki
ORDER BY cena DESC;
```

Zostanie zwrócony wiersz opisujący najdroższą książkę w bazie.

Zwykle razem z klauzulą `TOP` występuje klauzula `ORDER BY`. Bez niej klauzula `TOP` jest właściwie bezużyteczna, ponieważ to klauzula `ORDER BY` określa kolejność wierszy wyświetlanych w wyniku zapytania.

Otrzymany wynik jest poprawny tylko wtedy, gdy w bazie danych jest tylko jedna książka z tą ceną. A jeżeli książek z tą samą ceną jest więcej?

Wtedy można użyć rozszerzonej składni klauzuli `TOP` w postaci: `TOP n WITH TIES`. W wyniku zostaną zwrócone wszystkie wiersze z tą samą najwyższą ceną książki. Użycie rozszerzonej składni klauzuli TOP pokazuje przykład podany niżej.

Przykład 5.44

```
SELECT TOP 1 WITH TIES tytul, cena
FROM Ksiazki
ORDER BY cena DESC;
```

Zostaną zwrócone wszystkie wiersze opisujące książki o tej samej najwyższej cenie.

5.5.5. Grupowanie danych

Funkcje agregujące

Funkcje agregujące działają na wartościach wybranego pola w grupie wierszy. Wynikiem może być suma, średnia, liczba wierszy, wybranie wartości maksymalnej lub minimalnej. Funkcje te zwracają pojedyncze wartości i są wywoływane w instrukcji `SELECT`. Funkcje agregujące są jednym z najważniejszych narzędzi relacyjnych baz danych.

Podstawowe funkcje agregujące to:

- `COUNT` (*nazwa_kolumny*) — zwraca liczbę wierszy w grupie;
- `SUM` (*nazwa_kolumny*) — zwraca sumę wartości w grupie dla wskazanej kolumny;
- `AVG` (*nazwa_kolumny*) — zwraca średnią wartości w grupie dla wskazanej kolumny;
- `MAX` (*nazwa_kolumny*) — zwraca największą wartość w grupie dla wskazanej kolumny;
- `MIN` (*nazwa_kolumny*) — zwraca najmniejszą wartość w grupie dla wskazanej kolumny.

Oprócz powyższych funkcji agregujących niektóre serwery bazodanowe akceptują inne, mniej popularne funkcje.

Funkcja `COUNT` może zostać wywołana z symbolem * zamiast nazwy kolumny.

Przykład 5.45

```
SELECT COUNT(*) AS 'Liczba klientów'
FROM Klient;
```

W wyniku wykonania polecenia zostaną policzone wszystkie wiersze tabeli *Klient*, czyli otrzymamy odpowiedź na pytanie: „Ilu klientów jest zarejestrowanych w bazie danych *ksiegarnia_internetowa*?”. W wyniku takiego wywołania funkcji `COUNT()` zostaną policzone także puste wiersze. Jest to jedyny przypadek, gdy funkcja agregująca uwzględnia wartość `NULL`.

Natomiast wywołanie tej funkcji z jawnym podaniem nazwy kolumny spowoduje pominięcie wierszy, dla których wybrana kolumna ma wartość `NULL`.

Przykład 5.46

```
SELECT COUNT(kod_pocztowy) AS 'Liczba klientów'
FROM Klient;
```

Jeżeli dla niektórych klientów kolumna *kod_pocztowy* nie została wypełniona, zostaną oni pominięci w obliczeniach.

Przykład 5.47

```
SELECT AVG(cena) AS 'Średnia cena książek'
FROM Ksiazki;
```

Wynikiem będzie średnia cena książek znajdujących się w księgarni internetowej.

Jeżeli funkcja agregująca w swoich obliczeniach uwzględnia tylko wartości niepowtarzające się, to argumentem funkcji staje się słowo kluczowe `DISTINCT`.

Przykład 5.48

```
SELECT COUNT (DISTINCT miejscowosc) AS 'Liczba miejscowości'
FROM Klient;
```

W wyniku otrzymamy liczbę miejscowości, z których pochodzą klienci księgarni internetowej. Słowo kluczowe `DISTINCT` zostało umieszczone w nawiasie `()` jako argument funkcji `COUNT()`, ponieważ dotyczy kolumny *miejscowosc*, a nie tej funkcji.

Jeżeli w poleceniu dodamy klauzulę ograniczającą `WHERE`, to obliczenia będą dotyczyły tylko wierszy, które spełniają warunek zdefiniowany w klauzuli.

Przykład 5.49

```
SELECT COUNT (tytul) AS 'Liczba tytułów'
FROM Ksiazki
WHERE rok_wydania>2008;
```

Wynikiem będzie liczba książek wydanych po roku 2008.

Funkcje agregujące mogą być częścią wyrażeń.

Przykład 5.50

```
SELECT MAX (cena) - MIN(cena)
FROM Ksiazki;
```

W wyniku otrzymamy różnicę między maksymalną i minimalną ceną książki.

Klauzula GROUP BY

Funkcje agregujące mają zastosowanie głównie dla danych, które zostały pogrupowane. Do grupowania wierszy służy klauzula `GROUP BY`.

Przykład 5.51

Mamy policzyć książki o tym samym temacie i znaleźć trzy tematy z największą liczbą książek w bazie danych.

```
SELECT TOP 3 COUNT(tytul), temat
FROM Ksiazki
GROUP BY temat
ORDER BY 1 DESC;
```

Użycie klauzuli `GROUP BY` spowoduje pogrupowanie wierszy z tabeli *Ksiazki* według wartości w kolumnie *temat*. Funkcja `COUNT()` zliczy wiersze w każdej grupie, klauzula `ORDER BY` uporządkuje otrzymane wyniki od największej (`DESC`) do najmniejszej wartości w kolumnie *1*, a klauzula `TOP 3` ograniczy wynik zapytania do trzech pierwszych wierszy.

Klauzula HAVING

Klauzula `HAVING` jest ściśle powiązana z klauzulą `GROUP BY`. Określa, które wiersze zostaną zwrócone przez klauzulę `GROUP BY`.

Przykład 5.52

Chcemy otrzymać informację o tematach, dla których w bazie danych jest co najmniej pięć tytułów książek.

```
SELECT temat, COUNT(tytul)
FROM Ksiazki
GROUP BY temat
HAVING COUNT(tytul)>=5
ORDER BY 2 DESC;
```

Istotne jest zrozumienie różnicy między klauzulami `WHERE` oraz `HAVING`. Klauzula `WHERE` filtruje wiersze przed grupowaniem i obliczeniami, tym samym określa, dla których wierszy funkcje agregujące będą wykonywały obliczenia. Klauzula `HAVING` wybiera wiersze już pogrupowane, po wykonaniu obliczeń przez funkcje agregujące. Klauzula `WHERE` nie może zawierać funkcji agregujących. Klauzula `HAVING` zawsze zawiera funkcje agregujące. Można użyć klauzuli `HAVING` bez funkcji agregujących, ale wykona ona wtedy te same działania co klauzula `WHERE` z tym samym warunkiem i będzie od niej mniej efektywna.

Podsumowując, klauzula `WHERE` pozwala filtrować wiersze, natomiast klauzula `HAVING` pozwala filtrować grupy zwracane przez zapytanie.

5.6. Łączenie tabel

Omawiane do tej pory zapytania dotyczyły pojedynczych tabel, ale wiemy, że istotą relacyjnych baz danych jest odwoływanie się w zapytaniach do wielu powiązanych ze sobą tabel. Połączenia realizowane są przez porównanie wartości kolumny lub kilku kolumn z jednej tabeli z podobnymi kolumnami z drugiej tabeli.

Połączenia są najważniejszym mechanizmem relacyjnych baz danych. Dzięki nim możemy wybierać pasujące do siebie dane z wielu tabel, tworzyć raporty i podsumowania danych pochodzących z wielu tabel. Poprawne tworzenie połączeń jest podstawą projektowania i tworzenia profesjonalnych systemów baz danych.

Połączenia są realizowane między kluczem podstawowym jednej tabeli i kluczem obcym drugiej. Klucz obcy jest kolumną lub kombinacją kolumn, które są kluczem podstawowym w innej tabeli.

Wartości klucza podstawowego są niepowtarzalne, natomiast w kolumnie klucza obcego każda z wartości może powtórzyć się wielokrotnie. Wtedy tworzone powiązanie jest związkiem typu „jeden do wielu".

5.6.1. Połączenie wewnętrzne i zewnętrzne

W języku SQL połączenie między tabelami jest definiowane w sekcji `FROM` zapytania. Słowo kluczowe `INNER JOIN` definiuje połączenie między tabelami. Klauzula realizująca połączenie ma postać:

```
tabela1 INNER JOIN tabela2
ON tabela1.kolumna1= tabela2.kolumna2
```

Przykład 5.53

Mamy prześledzić historię zamówień w księgarni internetowej w czerwcu 2012 roku. Wynikiem zapytania ma być nazwisko i imię klienta, data złożenia zamówienia, liczba zamówionych egzemplarzy książki oraz tytuł i nazwisko autora książki.

```
SELECT Klient.nazwisko, Klient.imie, Zamowienia.[data zlozenia zamowienia],
Rejestracja_zamowienia.[liczba_egz], Ksiazki.tytul, Autor.nazwisko, Autor.
imie
FROM Klient INNER JOIN Zamowienia
ON Klient.id_klienta=Zamowienia.id_klienta
INNER JOIN Rejestracja_zamowienia
ON Zamowienia.id_zamowienia=Rejestracja_zamowienia.id_zamowienia
INNER JOIN Ksiazki
ON Rejestracja_zamowienia.id_ksiazki=Ksiazki.id_ksiazki
INNER JOIN Autor
```

```
ON Ksiazki.id_autora=Autor.id_autora
WHERE Zamowienia.[data zlozenia zamowienia] BETWEEN '2012-06-1' AND
'2012-06-30';
```

Aby prawidłowo utworzyć zapytanie, należy zdefiniować połączenia między tabelami bazy danych. Definicji połączeń można używać wielokrotnie, chociaż może to wpłynąć na szybkość działania zapytania.

Przedstawione w przykładzie połączenie jest **połączeniem wewnętrznym** (`INNER JOIN`). Oznacza to, że wynikiem zapytania są tylko wiersze zawierające w polu klucza podstawowego i w polu klucza obcego pasujące do siebie dane. Połączenie wewnętrzne jest domyślnym typem połączenia.

Innym rodzajem połączenia jest **połączenie zewnętrzne** (`OUTER JOIN`). Przy takim połączeniu wynikiem zapytania są wszystkie wiersze w jednej tabeli i pasujące do nich wiersze z drugiej tabeli.

Złączenia *zewnętrzne* dzielimy na:

- `LEFT OUTER JOIN` — zapytanie zwraca wszystkie wiersze z pierwszej tabeli i pasujące wiersze z drugiej tabeli;
- `RIGHT OUTER JOIN` — zapytanie zwraca wszystkie wiersze z drugiej tabeli i pasujące wiersze z pierwszej tabeli;
- `FULL OUTER JOIN` — zapytanie zwraca wszystkie pasujące i niepasujące wiersze z obu tabel.

Gdybyśmy chcieli uzyskać listę wszystkich zamówień zrealizowanych w księgarni internetowej wraz z numerami wystawionych faktur, to przy zastosowaniu domyślnego połączenia między tabelami (`INNER JOIN`) nie uzyskalibyśmy informacji o zamówieniach, dla których nie wystawiono faktury.

Jeżeli zastosujemy połączenie zewnętrzne, otrzymamy wszystkie wiersze z tabeli *Zamowienia* oraz numery faktur dla zamówień, dla których faktury zostały wystawione.

Przykład 5.54

```
SELECT Zamowienia.[data zlozenia zamowienia], Zamowienia.[koszt wysylki],
Faktura.nr_faktury, Faktura.sposob_platnosci
FROM Zamowienia LEFT OUTER JOIN Faktura
ON Zamowienia.nr_faktury=Faktura.nr_faktury;
```

Tak jak lewostronne połączenie zewnętrzne (`LEFT OUTER JOIN`) zwraca wszystkie wiersze z lewej tabeli, tak prawostronne połączenie zewnętrzne (`RIGHT OUTER JOIN`) zwraca wszystkie wiersze z prawej tabeli. Zmieniając kolejność tabel w klauzuli `FROM`, można zastąpić połączenie lewostronne połączeniem prawostronnym.

Połączenie zewnętrzne obustronne (`FULL OUTER JOIN`) zwraca wszystkie wiersze obu połączonych tabel, również te, które nie spełniają warunku połączenia.

Przykład 5.55

Jeżeli przyjmiemy, że w tabeli *Zamowienia* znajdują się zamówienia, dla których nie zostały wystawione faktury, a w tabeli *Faktury* znajdują się informacje o fakturach, które nie są powiązane z żadnym zamówieniem, to zapytanie, które wyświetli informację o wszystkich zamówieniach i fakturach, będzie miało postać:

```
SELECT Zamowienia.[data zlozenia zamowienia], Zamowienia.[koszt wysylki],
Faktura.nr_faktury, Faktura.sposob_platnosci
FROM Zamowienia FULL OUTER JOIN Faktura
ON Zamowienia.nr_faktury=Faktura.nr_faktury;
```

5.6.2. Połączenie krzyżowe

Wszystkie możliwe połączenia wierszy dwóch tabel nazywamy *iloczynem kartezjańskim* lub *połączeniem krzyżowym* (`CROSS JOIN`). W tego typu połączeniu nie określa się warunku połączenia. Połączeniem krzyżowym można połączyć dowolne dwie tabele. Wynik tak zaprojektowanego zapytania dla tabel o pięciu i dziesięciu wierszach to tabela o 50 wierszach. Przy większej liczbie wierszy w tabelach wynik może być bardzo duży.

Tego typu połączenia są rzadko stosowane w relacyjnych bazach danych.

Przykład 5.56

```
SELECT Klient.nazwisko, Klient.imie, Zamowienia.[data zlozenia zamowienia],
Zamowienia.[koszt wysylki]
FROM Klient CROSS JOIN Zamowienia;
```

5.6.3. Połączenia wielokrotne

Projektując zapytanie, możemy definiować w nim dowolną liczbę połączeń między tabelami. Maksymalna liczba dopuszczalnych połączeń zależy od serwera bazodanowego. Przy ich definiowaniu należy pamiętać, że dołączenie w zapytaniu każdej następnej tabeli powoduje zmniejszenie wydajności zapytania. Mechanizm obsługiwania przez serwer zapytań zawierających zdefiniowane połączenie między tabelami działa tak, że zawsze łączone są dwie tabele. Po połączeniu dwóch pierwszych tabel powstaje struktura pośrednia, która jest łączona z kolejną tabelą i tworzona jest kolejna struktura pośrednia. Tworzenie struktur pośrednich trwa aż do połączenia wszystkich tabel.

5.6.4. Złączenie tabeli z nią samą

Złączenie tabeli z nią samą jest definiowane w podobny sposób jak połączenia różnych tabel. Przy łączeniu tej samej tabeli, jej nazwy w klauzuli `FROM` byłyby takie same. Aby odróżnić je od siebie, należy nadać im nowe nazwy (*aliasy*).

Przykład 5.57

```
SELECT K1.nazwisko, K1.imie
FROM Klient AS K1 CROSS JOIN Klient AS K2;
```

Ponieważ w obu wirtualnych tabelach istnieją kolumny *nazwisko* i *imie*, nazwy te są niejednoznaczne i muszą być poprzedzone nazwą tabeli.

Taki sposób połączenia tabeli z nią samą może zostać wykorzystany do wykrycia klientów, którzy zarejestrowali się w bazie danych kilkakrotnie.

Przykład 5.58

```
SELECT K1.nazwisko, K1.imie
FROM Klient AS K1 CROSS JOIN Klient AS K2
WHERE K1.nazwisko=K2.nazwisko AND K1.imie=K2.imie AND K1.PESEL=K2.PESEL;
```

5.7. Więzy integralności

Dane przechowywane w bazie danych powinny spełniać wymogi integralności wynikające z założeń przyjętych podczas projektowania bazy.

5.7.1. Definiowanie klucza obcego

Jeżeli w bazie danych *ksiegarnia_internetowa* do tabeli *Ksiazki* zostanie wpisana nowa książka z numerem autora 27, a autora o numerze 27 nie ma w bazie, to znaczy, że powstał błąd, który łatwo popełnić przy wprowadzaniu danych. Aby tego uniknąć, należy odpowiednio zdefiniować więzy integralności — wtedy serwer bazodanowy będzie automatycznie sprawdzał poprawność dokonywanych wpisów.

Sprawdzanie spójności w bazie danych odbywa się po jawnym zdefiniowaniu klucza obcego. Dla tabeli *Ksiazki* możemy zdefiniować klauzulę, która poinformuje bazę, że *id_autora* w tej tabeli to klucz obcy pochodzący z kolumny *id_autora* w tabeli *Autor*. Klauzulę dotyczącą ograniczeń klucza obcego trzeba umieścić za definicją kolumn w poleceniu `CREATE TABLE` lub `ALTER TABLE`.

Ogólna postać polecenia wygląda następująco:

```
[CONSTRAINT nazwa] FOREIGN KEY (kolumna1, kolumna 2, ...)
    REFERENCE nazwa_tabeli (kolumna1, kolumna 2, ...)
```

gdzie:

- `CONSTRAINT` *nazwa* jest nazwą ograniczenia, może zostać pominięta — wtedy ograniczeniu zostanie nadana nazwa systemowa;
- `FOREIGN KEY` (*kolumna1*, *kolumna 2*, . . .) określa kolumny zawierające klucz obcy;

- REFERENCE *nazwa_tabeli* (*kolumna1*, *kolumna 2*, . . .) określa, z której tabeli pochodzi klucz obcy i które kolumny są w niej kluczem podstawowym.

Przykład 5.59

Podczas tworzenia tabeli *Ksiazki* należy zdefiniować klauzulę ograniczeń klucza obcego dla kolumny *id_autora*:

```
CREATE TABLE Ksiazki
(
    id_ksiazki INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    tytul nvarchar (100) NOT NULL,
    id_autora INT REFERENCES Autor (id_autora),
    cena money,
    rok_wydania nvarchar (4),
    wydawnictwo nvarchar (20),
    temat nvarchar (30),
    miejsce_wydania nvarchar (28),
    jezyk_ksiazki nvarchar (15),
    opis nvarchar (100),
);
```

W istniejącej tabeli (*Ksiazki*) można definiować klauzulę ograniczeń klucza obcego dla nowej kolumny (*id_autora*) podczas dodawania tej kolumny do tabeli:

```
ALTER TABLE Ksiazki
ADD id_autora INT REFERENCES Autor (id_autora);
```

W istniejącej tabeli (*Ksiazki*) dla istniejącego pola klucza obcego (*id_autora*) należy zdefiniować klauzulę ograniczeń klucza obcego przez zmodyfikowanie kolumny:

```
ALTER TABLE Ksiazki
ADD CONSTRAINT Ksiazki_FK FOREIGN KEY (id_autora)
REFERENCES Autor (id_autora);
```

Klauzula REFERENCES we wszystkich przypadkach zdefiniuje ograniczenie dla klucza obcego. Nie będzie możliwe wpisanie w kolumnie *id_autora* tabeli *Ksiazki* wartości, która nie istnieje w kolumnie *id_autora* tabeli *Autor*.

W klauzuli można również zdefiniować nazwę ograniczenia klucza obcego.

Przykład 5.60

```
ALTER TABLE Ksiazki
ADD id_autora INT CONSTRAINT Ksiazki_FK
REFERENCES Autor (id_autora);
```

Klauzula `CONSTRAINT Ksiazki_FK` nadaje ograniczeniu klucza obcego nazwę *Ksiazki_FK*.

Przy tak zdefiniowanym ograniczeniu nałożonym na klucz obcy w tabeli *Ksiazki* zmiana lub usunięcie klucza podstawowego z tabeli *Autor* powoduje skutki w tabeli *Ksiazki* wynikające z zasad kaskadowego usuwania i aktualizowania danych.

5.7.2. Kaskadowe usuwanie i aktualizowanie danych

Efekty modyfikacji klucza podstawowego w powiązanych tabelach są odzwierciedlane przez zdefiniowanie kaskadowego usuwania lub aktualizowania danych.

- Aktualizowanie klucza podstawowego wymaga aktualizacji wartości w powiązanym z nim kluczu obcym. W relacyjnej bazie danych klucze podstawowe nie powinny być w ogóle modyfikowane, więc kaskadowe aktualizowanie definiujemy tylko w wyjątkowych przypadkach.
- Usunięcie wiersza w tabeli nadrzędnej lub wartości klucza podstawowego wymaga usunięcia lub zaktualizowania wartości w powiązanym z nim kluczu obcym. Kaskadowe usuwanie może doprowadzić do usunięcia wielu wierszy z różnych tabel, a w konsekwencji do usunięcia istotnych danych, zatem należy je definiować wyłącznie dla tabel pomocniczych opisujących związki „wiele do wielu". Aktualizowanie wartości klucza obcego jest bezpieczniejsze. Jeżeli kolumna klucza obcego zezwala na wpisywanie wartości `NULL`, należy wartości usuniętego klucza podstawowego zastąpić tą wartością. Jeżeli jest to niemożliwe, trzeba zastąpić wartości `NULL` specjalnie zdefiniowaną wartością domyślną.

Kaskadowe usuwanie i aktualizowanie danych definiuje się w klauzulach `ON UPDATE` i `ON DELETE` z wartościami:

- `NO ACTION` — dane w powiązanych tabelach nie będą automatycznie modyfikowane. Jest to domyślna wartość.
- `CASCADE` — modyfikacja ma zostać automatycznie powtórzona we wszystkich powiązanych tabelach.
- `SET NULL` — zmodyfikowane wartości klucza podstawowego mają zostać zastąpione wartością `NULL` w powiązanych kolumnach klucza obcego.
- `SET DEFAULT` — zmodyfikowane wartości klucza podstawowego mają zostać zastąpione w powiązanych kolumnach klucza obcego wartością domyślną.

Przykład 5.61

```
ALTER TABLE Ksiazki
DROP CONSTRAINT Ksiazki_FK;
ALTER TABLE Ksiazki
```

```
ADD CONSTRAINT Ksiazki_FK FOREIGN KEY (id_autora)
REFERENCES Autor (id_autora)
ON UPDATE CASCADE
ON DELETE SET NULL;
```

Polecenie DROP CONSTRAINT usuwa wcześniej zdefiniowane ograniczenie. Następnie zostaje zdefiniowane nowe ograniczenie klucza obcego, w którym polecenie ON UPDATE CASCADE określa, że aktualizacja wykonana w tabeli nadrzędnej ma zostać powtórzona w kolumnie klucza obcego tabeli podrzędnej, a polecenie ON DELETE SET NULL oznacza, że gdy będzie usuwany wiersz w tabeli nadrzędnej, w kolumnie klucza obcego tabeli podrzędnej zostanie wstawiona wartość NULL.

5.8. Łączenie wyników zapytań

W języku SQL istnieją mechanizmy, które pozwalają łączyć wyniki kilku zapytań. Do połączenia zapytań można użyć jednej z trzech instrukcji: UNION, INTERSECT, EXCEPT.

Instrukcja UNION

Instrukcja UNION łączy wyniki zapytań i ma postać:

```
zapytanie1 UNION zapytanie2
```

Warunkiem wykonania instrukcji jest, aby łączone zapytania miały taką samą liczbę kolumn oraz aby typy kolumn były takie same.

Przykład 5.62

```
SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc='Warszawa'
UNION
SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc='Gdańsk';
```

Pierwsze zapytanie zwróci listę klientów z Warszawy, drugie z Gdańska. Wynikiem połączenia zapytań będzie lista klientów z Warszawy i z Gdańska.

Instrukcja UNION domyślnie powoduje usunięcie powtarzających się wierszy. W celu zachowania w wyniku wszystkich wierszy należy użyć instrukcji UNION ALL. Takie użycie instrukcji spowoduje dużo szybsze zwrócenie wyniku, ponieważ nie wymaga wyszukiwania i usuwania powtarzających się wierszy.

Instrukcja UNION lub UNION ALL może zastępować w zapytaniu operator OR.

Przykład 5.63

```
SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc = 'Warszawa' OR miejscowosc = 'Gdańsk';
```

Instrukcja INTERSECT

Instrukcja `INTERSECT` zwraca część wspólną wyników dwóch zapytań i ma postać:

```
zapytanie1 INTERSECT zapytanie2
```

Podobnie jak poprzednio, obydwa zapytania powinny zwracać taką samą liczbę kolumn o takich samych typach.

Przykład 5.64

```
SELECT nazwisko, imie
FROM Klient
WHERE telefon IS NULL
INTERSECT
SELECT nazwisko, imie
FROM Klient
WHERE adres_e_mail IS NULL;
```

Instrukcja `INTERSECT` zwróci dane klientów, którzy nie podali numeru telefonu ani adresu e-mail.

Instrukcja EXCEPT

Instrukcja `EXCEPT` ma postać:

```
zapytanie1 EXCEPT zapytanie2
```

Zwraca te wiersze, które wystąpiły w wyniku pierwszego zapytania, ale nie było ich w wyniku drugiego zapytania. Zmiana kolejności zapytań w tej instrukcji spowoduje zmianę wyniku.

Przykład 5.65

```
SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc='Warszawa'
EXCEPT
SELECT nazwisko, imie
FROM Klient
WHERE [telefon komorkowy] IS NOT NULL;
```

Instrukcja EXCEPT zwróci klientów, którzy mieszkają w Warszawie i nie podali numeru telefonu komórkowego.

5.9. Podzapytania

Zapytanie definiowane przy użyciu instrukcji SELECT może zostać umieszczone wewnątrz innej instrukcji SELECT. Tak zagnieżdżone zapytanie nazywamy **podzapytaniem**. Możemy używać również nazw **zapytanie zagnieżdżone** lub **zapytanie wewnętrzne**. Ponieważ serwery bazodanowe najpierw wykonują zapytania wewnętrzne, wyniki tych właśnie zapytań mogą być używane jako warunki logiczne kolejnych zapytań. Zagnieżdżanie zapytań można stosować w instrukcjach SELECT, INSERT, UPDATE i DELETE w klauzuli WHERE lub FROM. Każde podzapytanie klauzuli WHERE lub FROM jest umieszczane wewnątrz nawiasów okrągłych i może zawierać dowolną liczbę podzapytań.

Najczęściej wynikiem zapytania typu SELECT jest tabela składająca się z kolumn i wierszy, więc można na niej wykonywać kolejne zapytania typu SELECT.

Przy takim założeniu ogólna postać zapytania może wyglądać tak:

```
SELECT kolumna1, kolumna2, kolumna3, ...
FROM Tabela1
WHERE kolumna1 =
(SELECT kolumnaA1
FROM TabelaA1
WHERE warunek_podzapytania AS nazwa)
```

5.9.1. Podzapytania klauzuli WHERE

Zapytania wewnętrzne używane w klauzuli WHERE mogą zwracać pojedynczą wartość, listę wartości lub dane tabelaryczne (zawierające kilka kolumn).

Przykład 5.66

Na podstawie numeru faktury chcemy odczytać nazwisko i imię klienta, dla którego faktura została wystawiona.

```
SELECT Klient.nazwisko, Klient.imie
FROM Klient
WHERE Klient.id_klienta =
(SELECT Zamowienia.id_klienta
FROM Zamowienia
WHERE Zamowienia.nr_faktury=2);
```

Wynik zapytania wewnętrznego jest pojedynczą wartością. Zwraca *id_klienta*, dla którego została wystawiona faktura o numerze *2*. Zapytanie zewnętrzne zwraca imię i nazwisko klienta o wskazanym *id_klienta*.

Jeżeli takie same nazwy kolumn powtarzają się w kilku tabelach, w podzapytaniach powinny być one poprzedzone nazwami tabel — pozwoli to uniknąć błędów serwera.

Jeżeli zapytanie wewnętrzne zwraca tylko jedną wartość, to w zapytaniu zewnętrznym w klauzuli `WHERE` można użyć operatorów: =, <, >, <=, >=, <>.

Gdyby zwróciło listę wartości, użycie tych operatorów byłoby nielogiczne. Serwer bazodanowy nie potrafiłby jednoznacznie określić ich znaczenia. Jeżeli zaistnieje taka sytuacja, to do porównania należy użyć operatora `IN`. Zwróci on prawdę, jeżeli chociaż jedna wartość z listy spełnia zdefiniowany warunek.

Przykład 5.67

Na podstawie daty wysłania chcemy odczytać tytuł i cenę książek wysłanych do klientów 1 czerwca 2012 roku.

```
SELECT Ksiazki.tytul, Ksiazki.cena
FROM Ksiazki
WHERE Ksiazki.id_ksiazki IN
(SELECT Rejestracja_zamowienia.id_ksiazki
FROM Rejestracja_zamowienia INNER JOIN Zamowienia
ON Rejestracja_zamowienia.id_zamowienia=Zamowienia.id_zamowienia
WHERE [data wyslania]='2012-06-01');
```

Zapytanie wewnętrzne może wywoływać funkcje grupujące.

Przykład 5.68

Chcemy odczytać tytuły książek, których cena dwukrotnie przekracza średnią cenę książek zapisanych w bazie danych.

```
SELECT Ksiazki.tytul
FROM Ksiazki
WHERE Ksiazki.cena>
(SELECT AVG(Ksiazki.cena)*2
FROM Ksiazki);
```

5.9.2. Podzapytania klauzuli FROM

Wynikiem zapytania typu `SELECT` jest tabela zawierająca określone w zapytaniu kolumny, zatem możliwe jest wykonanie na niej kolejnego zapytania typu `SELECT`. W ten sposób tworzone są podzapytania w klauzuli `FROM`. Tabeli zwróconej przez podzapytanie użyte w klauzuli `FROM` należy przypisać nazwę.

Tabele pochodne

Jeżeli zapytanie wewnętrzne zwraca wartości w postaci danych tabelarycznych, to tworzone są tabele pochodne. Są one dostępne tylko w zapytaniu, w którym zostały utworzone.

Przykład 5.69

```
SELECT Li.tytul, Li.cena
FROM
(SELECT tytul, cena, miejsce_wydania, rok_wydania
FROM Ksiazki
WHERE rok_wydania<2012 AND cena>70) AS Li
WHERE miejsce_wydania='Warszawa';
```

W podanym przykładzie w wyniku wykonania zapytania wewnętrznego otrzymaliśmy wybrane wiersze i kolumny z tabeli *Ksiazki*. Tak powstałej tabeli pochodnej została nadana nazwa (*alias*) *Li*. W zapytaniu zewnętrznym zawartość tej tabeli jest odczytywana w podobny sposób jak zawartość zwykłej tabeli.

Tabele pochodne są stosowane w celu uproszczenia zapytań i poprawienia ich czytelności.

5.9.3. Operatory zapytań wewnętrznych

Podzapytania zwracające w zapytaniu wewnętrznym listę wartości wymagają zastosowania w klauzuli `WHERE` specjalnych operatorów. Są to operatory: `IN`, `EXISTS`, `ANY`, `SOME`, `ALL`.

`IN` sprawdza, czy chociaż jedna wartość z listy spełnia zdefiniowany warunek. Lista wartości musi składać się z jednej kolumny.

`EXISTS` służy do sprawdzenia, czy w wyniku wykonania zapytania wewnętrznego zostały zwrócone jakiekolwiek wartości.

`ANY`, `SOME` są synonimami i działają dokładnie tak samo. Sprawdzają wartość wybranego wiersza wyniku podzapytania.

`ALL` sprawdza wartość wszystkich wierszy wyniku podzapytania.

Operator EXISTS

Operator `EXISTS` zwraca tylko prawdę lub fałsz. Jeżeli w wyniku wykonania zapytania wewnętrznego zostały zwrócone jakiekolwiek wartości, operator `EXISTS` zwraca prawdę, a jeżeli nie zostały zwrócone żadne wartości, zwraca fałsz.

Przykład 5.70

```
SELECT Klient.nazwisko, Klient.imie
FROM Klient
WHERE EXISTS
(SELECT *
FROM Zamowienia
WHERE Klient.id_klienta =Zamowienia.id_klienta);
```

W wyniku wykonania zapytania zostanie zwrócona lista klientów, którzy złożyli zamówienia na książki.

Przykład 5.71

```
SELECT nazwisko, imie
FROM Klient AS K1
WHERE EXISTS
(SELECT *
FROM Klient AS K2
WHERE K1.nazwisko=K2.nazwisko AND K1.imie=K2.imie AND K1.PESEL=K2.PESEL);
```

W podanym przykładzie operator `EXISTS` został użyty do wyszukania powtarzających się danych klientów. Dane te mogły przez pomyłkę zostać kilkakrotnie wpisane do bazy. W celu stwierdzenia braku danych operator `EXISTS` może zostać zastosowany razem z operatorem `NOT`.

W przykładzie poniżej dodanie operatora `NOT` spowoduje wyświetlenie listy klientów, którzy nie złożyli żadnego zamówienia na książki.

Przykład 5.72

```
SELECT Klient.nazwisko, Klient.imie
FROM Klient
WHERE NOT EXISTS
(SELECT *
FROM Zamowienia
WHERE Klient.id_klienta =Zamowienia.id_klienta);
```

Operator ANY lub SOME

Operator `ANY` zwraca prawdę, jeśli którakolwiek ze zwróconych wartości wyniku zapytania wewnętrznego spełnia poprzedzający je warunek.

Przykład 5.73

```
SELECT Klient.nazwisko, Klient.imie
FROM Klient INNER JOIN Zamowienia
ON Klient.id_klienta=Zamowienia.id_klienta
WHERE Zamowienia.[data zlozenia zamowienia]= ANY
(SELECT Faktura.[Data wystawienia faktury]
FROM Faktura);
```

Podzapytanie zwróci nazwiska klientów, dla których faktury zostały wystawione w dniu złożenia zamówienia.

Operator ALL

Operator `ALL` zwraca prawdę, jeżeli wszystkie ze zwróconych wartości wyniku zapytania wewnętrznego spełniają poprzedzający je warunek.

Przykład 5.74

```
SELECT Ksiazki.tytul
FROM Ksiazki
WHERE Ksiazki.cena< ALL
(SELECT Zamowienia.[koszt wysylki]
FROM Zamowienia INNER JOIN Rejestracja_zamowienia
ON Zamowienia.id_zamowienia=Rejestracja_zamowienia.id_zamowienia);
```

Podzapytanie zwróci tytuły książek, które są tańsze od najniższych kosztów wysyłki. Najprawdopodobniej niewielu klientom opłaca się zamawiać książki, których koszt wysyłki przekracza cenę książki.

Podzapytania w instrukcjach modyfikujących dane

Podzapytania mogą być definiowane w instrukcjach modyfikujących dane, takich jak: `INSERT`, `UPDATE`, `DELETE`.

Przykład 5.75

W bazie danych zostanie utworzona nowa tabela *Archiwum*, która będzie zawierała imiona i nazwiska klientów oraz informacje o złożonych zamówieniach.

```
CREATE TABLE Archiwum
(
    nazwisko nvarchar (60),
    imie nvarchar (40),
[data zlozenia zamowienia] datetime
 );
```

Powstałą tabelę wypełnimy danymi z tabel *Klient* i *Zamowienia*. Możemy to zrealizować za pomocą instrukcji `INSERT` z odpowiednio zdefiniowanym podzapytaniem.

```
INSERT INTO Archiwum (nazwisko, imie,[data zlozenia zamowienia])
(SELECT nazwisko, imie,[data zlozenia zamowienia]
FROM Klient, Zamowienia
WHERE Klient.id_klienta=Zamowienia.id_klienta);
```

Przykład 5.76

Skoro w utworzonej tabeli nazwiska klientów powtarzają się tyle razy, ile razy zamawiali oni książki, można tak zmodyfikować polecenie, aby w tabeli zostały umieszczone nazwiska i imiona klientów z liczbą zamówień złożonych przez każdego z nich (w tym celu w tabeli *Archiwum* należy dodać pole *Liczba zamówień*). W podzapytaniu trzeba użyć grupowania oraz funkcji agregujących.

```
INSERT INTO Archiwum (nazwisko, imie, Liczba_zamowien)
(SELECT nazwisko, imie, COUNT([data zlozenia zamowienia]) AS 'Liczba
zamówień'
FROM Klient INNER JOIN Zamowienia
ON Klient.id_klienta=Zamowienia.id_klienta
GROUP BY Klient.nazwisko, Klient.imie);
```

5.9.4. Podzapytania skorelowane

Używane do tej pory podzapytania proste były wykonywane tylko jeden raz. W podzapytaniach skorelowanych następuje odwołanie się w zapytaniu wewnętrznym do wyniku zapytania zewnętrznego. W związku z tym zapytanie wewnętrzne wykonywane jest osobno dla każdego wiersza zwróconego przez zapytanie zewnętrzne.

Przykład 5.77

Mamy sprawdzić, które książki mają cenę wyższą niż średnia cena książek wydanych w tym samym roku.

```
SELECT tytul, cena, rok_wydania
FROM Ksiazki
WHERE cena >
   (SELECT AVG(cena)
   FROM Ksiazki AS Ks
   WHERE Ksiazki.rok_wydania=Ks.rok_wydania)
ORDER BY cena;
```

Dla każdej książki rozpatrywanej w zapytaniu zewnętrznym zostanie wyliczona w zapytaniu wewnętrznym średnia cena książek wydanych w tym samym roku co analizowana książka. W zapytaniu wewnętrznym tabela *Ksiazki* otrzymała nazwę *Ks*, aby warunek w klauzuli `WHERE` zapytania wewnętrznego miał sens.

5.10. Transakcje

W bazach danych transakcja to zbiór wykonywanych operacji, które stanowią całość. Muszą zostać wykonane wszystkie operacje wchodzące w skład transakcji lub nie zostanie wykonana żadna z nich. Przykładem transakcji jest wykonywanie przelewu z jednego konta na drugie. Operacja przelewu musi zostać wykonana w całości, a jeżeli nie jest to możliwe, należy powrócić do stanu sprzed rozpoczęcia wykonywania operacji przelewu.

Transakcja składa się z trzech etapów:

- rozpoczęcia,
- wykonania,
- zakończenia.

5.10.1. Właściwości transakcji

Transakcja ma przypisane jej własności:

- `atomic` — transakcja jest niepodzielna, czyli jest wykonywana w całości lub w całości jest odwoływana.
- `consistent` — transakcja nie zmienia spójności (integralności) bazy danych, czyli wykonanie transakcji nie doprowadzi do utraty spójności danych. Jeżeli baza danych była spójna przed wykonaniem transakcji, to jest spójna również po jej zakończeniu.
- `isolated` — transakcja musi być izolowana, czyli nie może istnieć konflikt z innymi transakcjami wykonywanymi w tym samym czasie na tym samym zbiorze danych.
- `durable` — transakcja jest trwała, czyli działania wykonane w transakcji są trwałe niezależnie od tego, co się będzie działo po jej zakończeniu.

W trakcie wykonywania transakcji stosowany jest **mechanizm blokowania**, który gwarantuje spójność bazy danych. Oznacza to, że podczas wykonywania transakcji dane, na których jest ona wykonywana, nie ulegną zmianie lub usunięciu.

Pierwsze litery własności transakcji **ACID** tworzą skrót określający reguły, które muszą być spełnione przez serwery bazodanowe, aby transakcje mogły być wykonywane.

W serwerze SQL Server transakcja może być wykonywana na trzy sposoby:

- *Explicit* — jawnie. Rozpoczęcie transakcji jest realizowane za pomocą polecenia `BEGIN TRANSACTION`.
- *Autocommit* — automatycznie. Operacje wykonywane na serwerze są standardowo traktowane jako transakcje, w związku z czym nie ma potrzeby ich rozpoczynania

poleceniem `BEGIN TRANSACTION`. Po poprawnym wykonaniu każde z poleceń jest automatycznie zatwierdzane.

- *Implicit* — niejawnie. Transakcje są wywoływane przez programy użytkowe działające na bazie danych.

Transakcje Explicit

Transakcje *Explicit* są wykonywane, jeżeli zadeklarujemy chęć wykonania zapytania lub bloku zapytań w ramach transakcji za pomocą polecenia `BEGIN TRANSACTION`.

Zatwierdzenie zmian i zakończenie transakcji deklarujemy za pomocą instrukcji `COMMIT`. Polecenie to zdejmuje blokady z tabel założone na czas trwania transakcji. Natomiast użycie instrukcji `ROLLBACK` odwołuje transakcję i odrzuca wszystkie zmiany dokonane podczas trwania transakcji. Polecenie to również zdejmuje blokady z tabel założone na czas trwania transakcji.

Przykład 5.78

Załóżmy, że od nowego roku ceny książek w księgarni internetowej wydanych po roku 2010 wzrosły o 5%, a ceny książek wydanych przed rokiem 2008 zmalały o 2%. Wykonywanie operacji zmiany cen książek zostanie zabezpieczone transakcją.

```
BEGIN TRANSACTION

UPDATE Ksiazki

SET cena = cena + cena *0.05

WHERE rok_wydania > 2010


IF @@ERROR <> 0

   BEGIN

      RAISERROR ('Błąd, Operacja zakończona niepowodzeniem !', 16, -1)

      ROLLBACK

   END


UPDATE Ksiazki

SET cena = cena - cena *0.02

WHERE rok_wydania < 2008


IF @@ERROR <> 0

   BEGIN

      RAISERROR ('Błąd, Operacja zakończona niepowodzeniem !', 16, -1)

      ROLLBACK

   END


COMMIT;
```

Pierwsza instrukcja (BEGIN TRANSACTION) uruchomiła transakcję, następna (UPDATE) przeprowadziła operację na danych. Kolejny blok to instrukcje sprawdzające, czy transakcja się powiodła. Ponieważ zmienna systemowa @@ERROR standardowo zwraca informację z numerem ostatniego błędu, może zostać użyta do sprawdzenia, czy w trakcie operacji zmiany ceny książek wystąpił błąd. Jeśli błąd nie wystąpił, zmienna ma wartość zero. Jeśli błąd wystąpił, zostanie wyświetlony komunikat (instrukcja RAISERROR) i transakcja zostanie odwołana (ROLLBACK). Jeżeli operacja przebiegła pomyślnie, przechodzimy do kolejnej operacji zmiany ceny książek.

Pojedyncza instrukcja UPDATE (dotyczy to również instrukcji DELETE i INSERT) nie musiałaby być poprzedzona uruchomieniem transakcji, ponieważ dla niej transakcja zostanie uruchomiona automatycznie. W podanym wyżej przykładzie modyfikowane dane są ze sobą wzajemnie powiązane, czyli niepowodzenie przy modyfikacji jednego zbioru danych powinno spowodować anulowanie modyfikacji drugiego zbioru danych — dlatego należy użyć transakcji.

Transakcje Autocommit

Transakcje *Autocommit* podobnie jak transakcje *Implicit* inicjuje MS SQL Server za każdym razem, gdy zostanie wydane polecenie modyfikowania danych. Ale po wykonaniu polecenia serwer automatycznie zatwierdza lub odrzuca transakcję.

Dla serwera MS SQL Server tryb automatyczny jest domyślnym trybem zarządzania transakcjami. Należy pamiętać, że jeżeli pracujemy w trybie *Autocommit*, każde wydane polecenie jest osobną transakcją.

Przykład 5.79

Wykorzystując funkcję systemową @@TRANCOUNT, sprawdzimy działanie trybu automatycznego transakcji. Funkcja ta zwraca liczbę otwartych w danym momencie transakcji.

```
SELECT @@TRANCOUNT;
UPDATE Ksiazki SET cena=20
WHERE rok_wydania<2008;
SELECT @@TRANCOUNT;
```

W wyniku wykonania kodu zobaczymy, że przed rozpoczęciem wykonywania instrukcji UPDATE i po jej zakończeniu nie było otwartych żadnych transakcji.

Transakcje Implicit

Transakcje *Implicit* inicjuje MS SQL Server, wydając niejawnie polecenie BEGIN TRANSACTIONS. Transakcje niejawne są przeprowadzane, gdy wykonywana jest jedna z następujących komend: ALTER TABLE, CREATE, DELETE, DROP, FETCH, GRANT, INSERT, OPEN, REVOKE, SELECT, TRUNCATE, UPDATE. Naszym zadaniem jest zakończenie transakcji i jej zatwierdzenie lub wycofanie. Aby serwer pracował w trybie transakcji niejawnych, należy je włączyć za pomocą polecenia:

```
SET IMPLICIT_TRANSACTIONS ON
```

Po wykonaniu tego polecenia do końca sesji będą realizowane transakcje niejawne. W celu wcześniejszego zakończenia pracy w trybie transakcji niejawnych trzeba wpisać polecenie:

```
SET IMPLICIT_TRANSACTIONS OFF
```

Ten tryb pracy może stwarzać problemy z bazą danych wtedy, gdy zapomnimy zatwierdzić lub wycofać transakcję. Pozostanie ona otwarta i będzie blokowała dostęp do danych. Jego zaletą jest możliwość wycofywania przypadkowych lub błędnych modyfikacji danych.

Przykład 5.80

Wykorzystując polecenie `SET IMPLICIT_TRANSACTIONS ON`, ustawimy tryb niejawny transakcji. Następnie stosując zmienną systemową `@@TRANCOUNT`, sprawdzimy działanie trybu niejawnego transakcji.

```
SET IMPLICIT_TRANSACTIONS ON;
SELECT @@TRANCOUNT;
UPDATE Ksiazki SET cena=30
WHERE rok_wydania<2007;
SELECT @@TRANCOUNT;
```

Przed rozpoczęciem wykonywania instrukcji `UPDATE` nie było otwartych żadnych transakcji. Natomiast po jej zakończeniu pozostała rozpoczęta jedna transakcja, ponieważ nie została ona automatycznie zamknięta. Użytkownik powinien zamknąć transakcję, zatwierdzając zmiany lub je wycofując. Aby zamknąć transakcję, należy wykonać polecenie:

```
COMMIT;
SET IMPLICIT_TRANSACTIONS OFF;
```

Dopóki transakcja nie zostanie zamknięta, dostęp do tabeli *Ksiazki* będzie niemożliwy.

Zagnieżdżanie transakcji

W ramach już rozpoczętej transakcji można umieścić kolejną instrukcję `BEGIN TRANSACTION`. Wynikiem takiego działania jest zwiększenie liczby otwartych transakcji, a nie rozpoczęcie nowej transakcji.

Mechanizm zagnieżdżonych transakcji wygląda następująco: wykonanie kolejnej instrukcji `BEGIN TRANSACTION` zwiększa o jeden liczbę otwartych transakcji, wykonanie instrukcji `COMMIT` zmniejsza o jeden liczbę otwartych transakcji, natomiast wykonanie instrukcji `ROLLBACK` zamyka transakcje i ustawia liczbę otwartych transakcji na zero.

Przykład 5.81

```
BEGIN TRANSACTION;
SELECT @@TRANCOUNT;
UPDATE Ksiazki SET cena-10
WHERE rok_wydania<2005;

BEGIN TRANSACTION;
SELECT @@TRANCOUNT;
   UPDATE Ksiazki SET cena=12
WHERE rok_wydania=2006;

BEGIN TRANSACTION;
SELECT @@TRANCOUNT;
   UPDATE Ksiazki SET cena=18
WHERE rok_wydania=2007;
COMMIT;

SELECT @@TRANCOUNT;
ROLLBACK;
SELECT @@TRANCOUNT;
```

W podanym przykładzie po pierwszym odczycie zmiennej `@@TRANCOUNT` otrzymamy jedną otwartą transakcję, następnie dwie, później trzy, znowu dwie i na końcu zero otwartych transakcji.

Punkty przywracania

W większości serwerów bazodanowych można wycofać nie tylko całą transakcję, ale także jej część. W tym celu należy utworzyć punkty przywracania za pomocą instrukcji `SAVE TRANSACTION`.

Przykład 5.82

```
BEGIN TRANSACTION;
INSERT INTO Klient (nazwisko, imie, PESEL)
VALUES ('Gordon', 'Michał', '79020908765');
SAVE TRANSACTION P1;
INSERT INTO Klient (nazwisko, imie, PESEL)
VALUES ('Zieliński', 'Tomasz', '89110803456');
ROLLBACK TRANSACTION P1;
```

W podanym przykładzie instrukcja `BEGIN TRANSACTION` uruchomi transakcję. Po dodaniu do tabeli *Klient* danych jednego klienta instrukcja `SAVE TRANSACTION P1` utworzy punkt przywracania. Po dodaniu do tabeli *Klient* danych kolejnego klienta, transakcja zostanie odwołana, ale nie zostaną odrzucone wszystkie zmiany dokonane podczas trwania transakcji, tylko zmiany wprowadzone po zdefiniowanym punkcie przywracania. W rezultacie, mimo że transakcja została odwołana, w tabeli *Klient* zostaną zapisane dane klienta o nazwisku *Gordon*, natomiast dane klienta o nazwisku *Zieliński* zostaną anulowane.

UWAGA

Należy pamiętać o zamknięciu transakcji. Inaczej dostęp do tabeli *Klient* będzie niemożliwy.

Zadanie 5.2

Jaki będzie ostateczny wynik wykonania kodu w przykładzie 5.81? Jakie ceny zostaną zachowane w kolumnie *cena*?

5.11. Współbieżność

Współbieżność to zdolność do jednoczesnego realizowania wielu procesów (wątków) opartych na wspólnych danych. Polega ona na przełączaniu między procesami w bardzo krótkich przedziałach czasu, co sprawia wrażenie, że procesy wykonywane są równocześnie. Ten mechanizm znajduje szerokie zastosowanie w serwerach bazodanowych, które muszą obsługiwać jednocześnie wielu użytkowników. Aby każdy z użytkowników mógł pracować tak, jakby był jedynym użytkownikiem bazy danych, konieczne jest odizolowanie operacji (transakcji) wykonywanych przez tych użytkowników.

5.11.1. Kontrola współbieżności

Kontrola współbieżności w serwerach bazodanowych może odbywać się na podstawie:

- modelu optymistycznego,
- modelu pesymistycznego.

Model optymistyczny

Model optymistyczny opiera się na założeniu, że modyfikowanie i odczytywanie tych samych danych przez różnych użytkowników jest mało prawdopodobne, chociaż nie niemożliwe. W związku z tym można przeprowadzić transakcję bez blokowania zasobów. Jedynie w sytuacji modyfikowania danych zasoby bazy są sprawdzane w celu wykrycia konfliktów.

Model pesymistyczny

Model pesymistyczny zakłada wystąpienie konfliktów, dlatego dane są blokowane za każdym razem, gdy jakaś transakcja próbuje je odczytać lub modyfikować.

5.11.2. Blokowanie danych

Blokowanie danych stosujemy w celu zagwarantowania integralności i spójności danych w trakcie realizowania transakcji. Dzięki temu użytkownicy nie mogą odczytywać danych, które są właśnie zmieniane przez innych użytkowników, oraz nie mogą równocześnie modyfikować tych samych danych. Bez blokad dane znajdujące się w bazie bardzo szybko stałyby się niespójne, a definiowane na nich zapytania dawałyby błędne wyniki.

Serwery bazodanowe automatycznie ustawiają blokady, ale na potrzeby własnej aplikacji można modyfikować ich standardowe ustawienia.

Tryby blokad

Istnieją dwa tryby blokad. Decydują one, czy możliwe jest założenie blokady na dane, które wcześniej zostały zablokowane przez inny proces.

Blokady współdzielone S (ang. *Shared*) są zakładane domyślnie na odczytywanych obiektach tylko na czas wykonania zapytania. Jeżeli dane zostały zablokowane w trybie *S*, to możliwe jest założenie na nie blokady *S* przez inne procesy.

Blokady wyłączne X (ang. *eXclusive*) są zakładane na modyfikowanych obiektach i domyślnie utrzymywane do zakończenia całej transakcji. Użytkownicy modyfikujący dane blokują innych użytkowników.

Zakresy blokad

Blokady mogą być zakładane na różnym poziomie szczegółowości — na poziomie wierszy, kluczy indeksów, stron, tabel, zakresów lub bazy. Dla każdej transakcji dynamicznie jest określany odpowiedni poziom, na którym należy założyć blokadę. Poziomy, na których zakładane są blokady, są ustawiane i kontrolowane przez serwer bazodanowy. To on sprawdza, czy blokady przydzielone na jednym poziomie respektują blokady ustawione na innym poziomie.

Im większe obiekty są blokowane, tym dłużej nie są one dostępne dla użytkownika (mniejsza współbieżność), ale dzięki temu zmniejsza się liczba blokad, którymi musi zarządzać serwer.

Zakleszczenia

Zakleszczenie (ang. *Deadlock*) powstaje, gdy jeden proces próbuje założyć blokadę powodującą konflikt z blokadą, którą próbuje założyć inny proces. W wyniku nie mogą zostać założone blokady wymagane do ukończenia rozpoczętych procesów. Serwery bazodanowe posiadają algorytmy, które automatycznie wykrywają zakleszczenia i przerywają wykonywanie jednej z transakcji.

5.11.3. Izolowanie transakcji

Wiemy, że transakcja musi być izolowana, czyli próba odczytania danych z tabeli, na której przeprowadzana jest transakcja przez innego użytkownika, zakończy się niepowodzeniem. Dopiero pojawienie się polecenia `COMMIT` powoduje właściwą modyfikację danych, zdjęcie blokad z danych i umożliwia dostęp do nich innym użytkownikom.

W zależności od istniejącego na serwerze bazodanowym poziomu izolowania transakcji może pojawić się jeden z następujących problemów:

- **Utrata aktualizacji** (ang. *Lost or buried updates*) — problem pojawi się, gdy dwie transakcje modyfikują te same dane. Żadna z transakcji nie wie, że druga transakcja korzysta z tych samych danych. W efekcie transakcja, która zakończy się później, zmodyfikuje dane, niszcząc zmiany dokonane przez transakcję, która zakończyła się wcześniej. Domyślnie skonfigurowany serwer nie dopuści do utraty aktualizacji.
- **Brudne odczyty** (ang. *Dirty reads*) — problem pojawia się, gdy jedna transakcja dokonuje zmian danych, a druga w tym czasie odczytuje te dane. W efekcie transakcja odczytuje dane, które nie zostały jeszcze zatwierdzone i mogą zostać wycofane. Domyślnie skonfigurowany serwer nie dopuści do brudnych odczytów.
- **Niepowtarzalne odczyty** (ang. *Non-repeatable reads*) występują, gdy powtórzenie w ramach transakcji tego samego odczytu daje inny wynik. Może to być spowodowane tym, że po pierwszym odczycie (a nie po zakończeniu transakcji) zostaną zdjęte blokady założone na odczytywane dane. Niezablokowane dane mogą zostać zmienione przez inne działania, a powtórne ich odczytanie da inny wynik. Domyślnie skonfigurowany serwer dopuszcza niepowtarzalne odczyty.
- **Odczyty widma** (ang. *Phantom reads*) występują, gdy pomiędzy dwoma odczytami tych samych danych w ramach jednej transakcji zmieni się liczba odczytywanych wierszy (na przykład w wyniku wykonania w międzyczasie instrukcji *INSERT* lub *DELETE* na tych danych). Domyślnie skonfigurowany serwer dopuszcza odczyty widma.

5.11.4. Poziomy izolowania transakcji

Blokad używamy w celu zabezpieczenia danych w trakcie współbieżnego wykonywania transakcji. Pozwala to na wykonywanie transakcji w pełnej izolacji od innych transakcji i zapewnia przez cały czas poprawność danych w bazie.

Dzięki temu możliwe jest wykonywanie kilku transakcji w tym samym czasie, tak jakby były wykonywane jedna po drugiej, czyli szeregowo.

Transakcje nie zawsze wymagają pełnej izolacji. Możemy wpływać na sposób zakładania blokad przez serwery bazodanowe, zmieniając poziom izolacji transakcji.

Poziom izolacji określa stopień, do którego dana transakcja musi być izolowana od innych transakcji. Najniższy poziom izolacji to maksymalny stopień współbieżności, ale jest on okupiony najmniejszym stopniem spójności danych. Natomiast wyższy stopień

izolacji transakcji zwiększa poprawność danych, ale zmniejsza stopień współbieżności. Najwyższy poziom izolacji gwarantuje najwyższy poziom spójności danych kosztem ograniczenia do minimum współbieżności.

Serwery bazodanowe pozwalają ustawiać na poziomie serwera, baz danych lub pojedynczych sesji poziom izolowania transakcji.

Standard SQL3 definiuje cztery poziomy izolacji transakcji:

- *Read Uncommitted*,
- *Read Committed*,
- *Repeatable Read*,
- *Serializable*.

Read Uncommitted

Read Uncommitted to tryb niezatwierdzonego odczytu. Jest to najniższy poziom izolacji transakcji. Odczyt danych nie powoduje założenia blokady współdzielonej. Tylko fizycznie uszkodzone dane nie będą odczytywane. Na tym poziomie pojawiają się brudne odczyty, niepowtarzalne odczyty i odczyty widma. Nie występuje jedynie problem z utratą aktualizacji. Ten tryb może być stosowany do odczytywania danych, o których wiemy, że w czasie odczytywania nie będą modyfikowane.

Przykład 5.83

W ramach jednej sesji w oknie edytora SQL rozpoczniemy transakcję, której zadaniem będzie zmodyfikowanie danych klienta w tabeli *Klient*.

```
BEGIN TRANSACTION;

UPDATE Klient

SET ulica = 'Mickiewicza', nr_domu=94

WHERE nazwisko='Nowak' AND imie='Andrzej';
```

Nie zamykając tej transakcji w ramach kolejnej sesji, jako inny użytkownik (nowe okno edytora SQL) zmienimy poziom izolowania transakcji i spróbujemy odczytać dane modyfikowane przez pierwszego użytkownika:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

GO

SELECT miejscowosc, ulica, nr_domu

FROM Klient

WHERE nazwisko='Nowak' AND imie='Andrzej';
```

Mimo że użytkownik pierwszy nie zamknął transakcji, udało się odczytać zmienione dane klienta.

Kończąc przykład, należy zamknąć transakcję bez zatwierdzania zmian.

Read Committed

Read Committed to tryb odczytu zatwierdzonego. Jest to domyślny poziom izolacji stosowany przez MS SQL Server. Podczas odczytu danych zostanie założona na nie blokada współdzielona. Założona blokada eliminuje brudne odczyty. Natomiast nadal występują niepowtarzalne odczyty oraz odczyty widma.

Przykład 5.84

Otwieramy dwa okna edytora SQL. W pierwszym oknie ustawiamy tryb odczytu zatwierdzonego, rozpoczynamy transakcję i odczytujemy dane klienta:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;

SELECT miejscowosc, ulica, nr_domu
FROM Klient
WHERE nazwisko='Nowak' AND imie='Andrzej';
```

Pozostawiamy otwartą transakcję i w drugim oknie zmieniamy adres klienta:

```
UPDATE Klient
SET ulica = 'Sienkiewicza', nr_domu=21
WHERE nazwisko='Nowak' AND imie='Andrzej';
```

Ponownie wracamy do okna pierwszego. W ramach tej samej transakcji drugi raz odczytujemy dane klienta:

```
SELECT miejscowosc, ulica, nr_domu
FROM Klient
WHERE nazwisko='Nowak' AND imie='Andrzej';
COMMIT;
```

Adres ponownie odczytany jest inny niż odczytany wcześniej. Wystąpił efekt niepowtarzalnych odczytów.

Repeatable Read

Repeatable Read to tryb powtarzalnego odczytu. Założona na dane blokada współdzielona jest utrzymywana aż do zakończenia całej transakcji. Dzięki temu, że inny proces nie może zmodyfikować odczytywanych danych, zostały wyeliminowane niepowtarzalne odczyty. Natomiast nadal występują odczyty widma.

Przykład 5.85

Otwieramy dwie sesje edytora SQL. W pierwszej ustawiamy tryb powtarzalnego odczytu, rozpoczynamy transakcję i odczytujemy nazwiska klientów mieszkających w Poznaniu:

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN TRANSACTION;


SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc='Poznań';
```

Pozostawiamy otwartą transakcję i w drugiej sesji zmieniamy miejsce zamieszkania jednego z klientów, który nie mieszkał w Poznaniu, na Poznań:

```
UPDATE Klient
SET miejscowosc = 'Poznań'
WHERE nazwisko='Kowalski' AND imie='Jan';
```

Ponownie wracamy do pierwszej sesji. W ramach tej samej transakcji drugi raz odczytujemy nazwiska klientów mieszkających w Poznaniu:

```
SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc='Poznań';
```

Tym razem liczba odczytywanych wierszy uległa zmianie. Jest ich więcej. Pojawił się wiersz widmo.

Zmiana danych w drugiej sesji była możliwa, ponieważ dotyczyła danych, które nie były odczytywane w transakcji otwartej w pierwszej sesji.

Na zakończenie przykładu należy wykonać instrukcję zamknięcia transakcji `COMMIT`.

Sytuacja ulegnie zmianie, gdy w drugiej sesji będziemy próbowali zmienić dane odczytywane w pierwszej sesji.

Przykład 5.86

Otwieramy dwie sesje edytora SQL. W pierwszej ustawiamy tryb powtarzalnego odczytu, rozpoczynamy transakcję i odczytujemy nazwiska klientów mieszkających w Gdańsku:

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN TRANSACTION;


SELECT nazwisko, imie
FROM Klient
WHERE miejscowosc='Gdańsk';
```

Pozostawiamy otwartą transakcję i w drugiej sesji zmieniamy miejsce zamieszkania jednego z klientów, który mieszkał w Gdańsku, na Kraków:

```
UPDATE Klient
SET miejscowosc = 'Kraków'
WHERE miejscowosc = 'Gdańsk';
```

Ta instrukcja będzie oczekiwać na zakończenie transakcji w pierwszej sesji i zdjęcie blokady z danych.

Aby aktualizacja danych była możliwa, w pierwszej sesji należy wykonać instrukcję zamknięcia transakcji `COMMIT`. Efekt niepowtarzalnego odczytu został zablokowany.

Serializable

Serializable to tryb szeregowania. Transakcje odwołujące się do tych samych danych są szeregowane, czyli wykonywane jedna po drugiej. Jest to najwyższy poziom izolacji transakcji, gdzie transakcje są w pełni izolowane od siebie. Na czas trwania transakcji blokowane są całe obiekty, a nie tylko odczytywane dane, co pozwala wyeliminować odczyty widma, ale powoduje, że inni użytkownicy nie mogą modyfikować przechowywanych w obiekcie danych. Na przykład odczytując jeden wiersz tabeli, blokujemy możliwość modyfikowania danych w całej tabeli.

Przykład 5.87

Otwieramy dwie sesje edytora SQL. W pierwszej ustawiamy tryb szeregowania, rozpoczynamy transakcję i odczytujemy informacje o wybranym kliencie:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;

SELECT miejscowosc, ulica, nr_domu
FROM Klient
WHERE nazwisko='Adamek' AND imie='Marek';
```

Jeżeli w drugiej sesji spróbujemy zmienić dane innego klienta, to okaże się, że aktualizacja została zablokowana:

```
UPDATE Klient
SET miejscowosc = 'Warszawa'
WHERE nazwisko='Borewicz' AND imie='Adam';
```

Aktualizacja będzie możliwa dopiero po zakończeniu transakcji w pierwszej sesji.

Na zakończenie przykładu należy zamknąć obydwie sesje bez zatwierdzania transakcji.

W tym trybie odczytywane dane zawsze są takie same. Jednak przez czas trwania transakcji pozostali użytkownicy nie mogą modyfikować zablokowanych tabel. Powoduje to znaczne wydłużenie czasu dostępu do danych i w praktyce, jeśli zmian nie jest dużo, zaleca się przełączenie do modelu optymistycznego.

5.12. Widoki

Widoki (perspektywy) w języku SQL to wirtualne tabele tworzone na podstawie zapytań. Składają się z kolumn i wierszy pobranych z prawdziwych tabel. Pokazywane w widoku dane są zawsze aktualne, ponieważ widoki są tworzone w momencie wykonania zapytania.

Widoki nie przechowują zapisanych w tabelach danych. W bazie danych jest zapamiętywana tylko definicja widoku i związane z nią metadane. Natomiast widoki są tworzone za każdym razem, gdy do widoku zostanie skierowane zapytanie. Umożliwiają wykonywanie operacji podobnych do tych wykonywanych na tabelach. Zapewniają bezpieczeństwo danych przez ograniczenie dostępu do danych zapisanych w tabelach. Na podstawie istniejących widoków można tworzyć następne.

Tworzenie i usuwanie widoku

W celu zapisania zapytania w postaci widoku należy poprzedzić je poleceniem `CREATE VIEW`.

Przykład 5.88

```
CREATE VIEW Zbior_ksiazek AS
SELECT tytul, nazwisko, imie, cena
FROM Autor INNER JOIN Ksiazki
ON Autor.id_autora= Ksiazki.id_autora;
```

Do usuwania zdefiniowanego widoku używamy polecenia `DROP VIEW`.

Przykład 5.89

```
DROP VIEW Zbior_ksiazek;
```

W wyniku wykonania polecenia z bazy danych zostanie usunięty widok *Zbior_ksiazek;*.

Modyfikowanie widoku

Zdefiniowany wcześniej widok można zmodyfikować. Modyfikując widok, należy jeszcze raz zapisać tworzącą go instrukcję `SELECT`. Różnica między tworzeniem widoku od nowa a modyfikowaniem istniejącego polega na tym, że widok modyfikowany zachowuje nadane mu wcześniej uprawnienia. Widok modyfikujemy instrukcją `ALTER VIEW`.

Przykład 5.90

```
ALTER VIEW dbo. Zbior_ksiazek AS
SELECT tytul, nazwisko, imie, cena
FROM Ksiazki INNER JOTN Autor
ON Ksiazki.id_autora = Autor.id_autora
WHERE cena > 50;
```

UWAGA

W widokach nie można stosować klauzuli `ORDER BY`. Jedyny wyjątek to użycie klauzuli `ORDER BY` do określenia wierszy, które są zwracane przez klauzulę `TOP`.

Widoki są szczególnie przydatne, gdy w zapytaniach często pobieramy dane z wielu połączonych między sobą tabel. Zamiast każdorazowego definiowania tych połączeń można do pracy z bazą wykorzystać zdefiniowane widoki. Dane poprzez widoki są odczytywane tak samo jak bezpośrednio z tabel.

Przykład 5.91

```
CREATE VIEW Sprzedaz AS
SELECT Klient.nazwisko AS K_Nazw, Klient.imie AS K_Im, Ksiazki.tytul, Autor.
nazwisko AS A_Nazw, Autor.imie AS A_Im, Ksiazki.cena
FROM klient INNER JOIN Zamowienia
ON Klient.id_klienta=Zamowienia.id_klienta
INNER JOIN Rejestracja_zamowienia
ON Zamowienia.id_zamowienia=Rejestracja_zamowienia.id_zamowienia
INNER JOIN Ksiazki
ON Rejestracja_zamowienia.id_ksiazki=Ksiazki.id_ksiazki
INNER JOIN Autor
ON Ksiazki.id_autora=Autor.id_autora
WHERE Zamowienia.[data zlozenia zamowienia] = '2012-06-01';
```

Zaleca się, aby użytkownicy baz danych ze względu na bezpieczeństwo danych nie mieli bezpośredniego dostępu do tabel. Dlatego dobrym rozwiązaniem jest definiowanie widoków i udostępnianie danych poprzez utworzone widoki.

5.13. Indeksy

Indeksy są definiowane w celu zwiększenia prędkości wykonywania operacji na tabeli. Są to uporządkowane struktury zawierające dane z wybranych kolumn tabeli. Zaletą stosowania indeksów jest ograniczenie ilości danych odczytywanych z bazy, przyspieszenie wyszukiwania informacji oraz sortowanie danych. Ich wadą jest to, że zajmują na dysku dodatkowe miejsce, muszą być na bieżąco aktualizowane, a każde wstawienie, usunięcie lub aktualizacja danych w tabeli wiąże się z aktualizacją wszystkich zdefiniowanych dla niej indeksów.

Indeksy można budować na etapie tworzenia tabel lub definiować je w już istniejącej tabeli. Instrukcja tworzenia indeksu ma postać:

```
CREATE INDEX nazwa_indeksu ON nazwa_tabeli (nazwa_kolumny)
```

Przykład 5.92

Tworzenie indeksu w istniejącej tabeli:

```
CREATE INDEX Indeks_tytul ON Ksiazki (tytul);
```

Tworzenie unikatowego indeksu ma postać:

```
CREATE UNIQUE INDEX nazwa_indeksu
ON nazwa_tabeli (nazwa_kolumny)
```

W tak zaindeksowanej kolumnie powtarzające się wartości są niedozwolone.

UWAGA

Instrukcja tworzenia indeksów w innych serwerach bazodanowych może różnić się od podanej.

Aktualizacja tabeli z indeksami zajmuje więcej czasu niż aktualizacja tabeli bez indeksów. Dlatego należy tworzyć indeksy tylko dla kolumn, które będą często przeszukiwane. Przed utworzeniem indeksów warto przeanalizować zapytania i podjąć decyzję, gdzie i jakie indeksy utworzyć. Na pewno indeksy powinny być utworzone dla pól kluczy głównych i kluczy obcych.

Do usuwania indeksu służy instrukcja `DROP INDEX`:

```
DROP INDEX nazwa_tabeli.nazwa_indeksu
```

5.14. T-SQL

T-SQL (*Transact-SQL*) jest rozszerzeniem języka SQL stosowanym dla serwerów MS SQL Server. W jego skład oprócz standardowych poleceń języka SQL wchodzą instrukcje tworzenia pętli, instrukcje warunkowe, zmienne, wyzwalacze, funkcje i procedury.

W języku T-SQL ciąg poleceń bezpośrednio kierowany do serwera powinien kończyć się słowem kluczowym `GO`. Realizacja ciągu poleceń na poziomie edytora jest realizowana po wybraniu przycisku *Execute* lub wciśnięciu na klawiaturze *F5*.

Pojedyncza instrukcja może kończyć się średnikiem, ale brak średnika nie jest błędem. Instrukcje mogą być pisane w jednej linii. Jednak w celu większej czytelności kodu zaleca się umieszczanie każdej z nich w oddzielnej linii i stosowanie wcięć.

5.14.1. Zmienne

Zmienne deklarowane przez użytkownika są zmiennymi lokalnymi i istnieją tylko w obrębie skryptu. Muszą być poprzedzone znakiem `@`. Deklaracja zmiennych jest realizowana za pomocą instrukcji:

```
DECLARE @zmienna typ_danych
```

W jednej instrukcji można deklarować wiele zmiennych.

Przykład 5.93

```
DECLARE @Imie nvarchar (10), @Nazwisko nvarchar (30);
```

W podanym przykładzie zostały zadeklarowane dwie zmienne lokalne @Imie i @Nazwisko.

Zmienne systemowe

Zmienne systemowe oznaczone są dwoma znakami `@@`.

Przykłady zmiennych systemowych:

- `@@ERROR` — zwraca numer ostatniego błędu (0 — brak błędu);
- `@@FETCH_STATUS` — ustala, czy kursor pobrał wiersz (0 — pobrał);
- `@@IDENTITY` — zwraca ostatnio wygenerowaną wartość;
- `@@ROWCOUNT` — zwraca liczbę wierszy, dla których została wykonana instrukcja SQL;
- `@@VERSION` — zwraca informację o wersji SQL;
- `@@TRANCOUNT` — zwraca liczbę otwartych transakcji.

Wartość do zmiennej można przypisać za pomocą instrukcji `SELECT`.

```
SELECT @zmienna=wyrażenie [FROM ...]
```

Przykład 5.94

```
SELECT @Imie='Jan', @Nazwisko='Kowalski';
```

Innym sposobem przypisania zmiennej wartości jest użycie instrukcji `SET`. Instrukcją `SET` można przypisać tylko jedną zmienną.

Przykład 5.95

```
SET @Imie='Jan';

SET @Nazwisko='Kowalski';
```

Zmiennym można przypisywać wartości zwrócone przez zapytanie. Zapytanie takie musi jednak zwracać dokładnie jeden wiersz.

Przykład 5.96

```
SELECT @Imie=imie, @Nazwisko=nazwisko

FROM Klient

WHERE id_klienta=1;
```

Jeśli zapytanie zwróci więcej wierszy, to zmiennym zostaną przypisane wartości z ostatniego wiersza. Natomiast jeśli zapytanie nie zwróci żadnego wiersza, zmienne zachowają swoje dotychczasowe wartości.

Wartość przypisaną zmiennej można odczytać za pomocą instrukcji SELECT.

Przykład 5.97

```
INSERT INTO Ksiazki (tytul) VALUES ('Dolina Issy')

SELECT @@IDENTITY;
```

Instrukcja INSERT INTO doda do tabeli *Ksiazki* nowy rekord i w polu *tytul* umieści podaną wartość. Dodatkowo dla pola *id_ksiazki*, które nie może pozostać puste, zostanie wygenerowana wartość liczbowa. Zapytanie SELECT @@IDENTITY zwróci wartość wygenerowaną dla tego pola.

5.14.2. Instrukcja warunkowa

Instrukcja warunkowa używana jest do określania sposobu wykonania kodu. Ogólna postać instrukcji wygląda następująco:

```
IF warunek_logiczny
   polecenie
ELSE
   polecenie
```

Przykład 5.98

```
DECLARE @Imie nvarchar (10), @Nazwisko nvarchar (30);

SELECT @Imie='Adam', @Nazwisko='Nowak';

IF NOT EXISTS (SELECT TOP 1 FROM Klient WHERE imie=@Imie AND nazwisko- @
Nazwisko)

INSERT INTO Klient (imie, nazwisko)

VALUES ('Adam', 'Nowak');
```

Wynikiem będzie sprawdzenie, czy klient o nazwisku Adam Nowak został zarejestrowany w bazie, a jeżeli nie został zarejestrowany — dodanie go do tabeli *Klient*.

Przykład 5.99

```
IF object_id ('Zbior_ksiazek','X') IS NOT NULL

DROP VIEW dbo.Zbior_ksiazek;

GO

CREATE VIEW Zbior_ksiazek AS

SELECT tytul, nazwisko,imie, cena

FROM Autor INNER JOIN Ksiazki

ON Autor.id_autora- Ksiazki.id_autora;
```

Funkcja systemowa `object_Id (nazwa_obiektu)` zwraca numer identyfikacyjny obiektu i została użyta do sprawdzenia, czy podany obiekt istnieje. Jeżeli tak jest, obiekt ten zostanie usunięty.

5.14.3. Wyrażenie CASE

Wyrażenie `CASE` może być użyte wewnątrz niektórych poleceń języka SQL. Należą do nich polecenia: `SELECT`, `UPDATE`, `DELETE`, `SET` oraz klauzule: `IN`, `WHERE`, `ORDER BY`, `HAVING`. Wyrażenie `CASE` sprawdza dla każdego wiersza wartość otrzymaną w zapytaniu i porównuje ją z wynikiem wyrażenia. W zależności od wyniku zwraca wartość zapisaną po słowie kluczowym `THEN`.

```
CASE
{WHEN wyrażenie_logiczne THEN wyrażenie_wynikowe}
[ELSE wyrażenie_wynikowe]
END
```

Przykład 5.100

```
SELECT [cena ksiazki],
CASE
WHEN [cena ksiazki] < 20 THEN 'Tania książka'
WHEN [cena ksiazki] < 80 THEN 'Książka w średniej cenie'
ELSE 'Książka z najwyższej półki'
END
FROM Ksiazki;
```

Wyrażenie CASE sprawdza dla każdego wiersza tabeli *Ksiazki* wartość pola *cena ksiazki* i porównuje ją z wartościami 20 i 80. W zależności od wyniku porównania zwraca wartość zapisaną po słowie kluczowym THEN.

5.14.4. Wyrażenia tablicowe (CTE)

Wyrażenia tablicowe umożliwiają definiowanie wirtualnych tabel, do których można się odwoływać wielokrotnie za pomocą ich nazwy. W celu utworzenia wyrażenia tablicowego należy użyć polecenia `WITH` w postaci:

```
WITH nazwa_wirtualnej_tabeli
AS
(zapytanie)
```

Po wykonaniu zapytania zostanie utworzona wirtualna tabela, z którą można pracować jak ze zwykłą tabelą.

Wyrażenia tablicowe mogą być stosowane:

- do tworzenia zapytań cyklicznych,
- jako odpowiedniki widoków, jeżeli nie ma potrzeby przechowywania definicji metadanych,
- przy wielokrotnym odwoływaniu się do tabeli wynikowej w tym samym zapytaniu.

Zaletą stosowania wyrażeń tablicowych jest większa czytelność złożonych zapytań. Mogą one służyć do budowania prostych bloków logicznych, z których tworzone są bardziej złożone zapytania. Mogą być definiowane i używane w funkcjach i procedurach składowanych oraz w wyzwalaczach i widokach.

Wykonanie wyrażenia tablicowego następuje po podaniu polecenia `SELECT`:

```
SELECT lista_kolumn
FROM nazwa_wirtualnej_tabeli
```

Przykład 5.101

```
WITH CTE_Klient
AS
(SELECT id_klienta, imie +' ' +nazwisko AS Dane,
kod_pocztowy+' '+miejscowosc AS Adres,
ulica+' '+nr_domu AS Ul
FROM Klient)
SELECT id_klienta, Dane, Adres, Ul
FROM CTE_Klient
WHERE id_klienta > 20;
```

Najpierw została utworzona wirtualna tabela *CTE_Klient*, a następnie na podstawie tej tabeli w zapytaniu zostały wybrane dane do pokazania.

Jeżeli połączymy ze sobą wyniki dwóch wyrażeń tabelarycznych, uzyskamy możliwość wykonywania rekurencyjnych zapytań.

5.14.5. Procedury składowane

Procedury składowane to polecenie lub kilka poleceń, które są wykonywane jako całość w jednym bloku. Mogą mieć zadeklarowane zarówno parametry wejściowe, jak i wyjściowe. Mogą również zawierać polecenia kontrolowania kodu, takie jak `IF` oraz `WHILE`. Zaleca się, aby wszystkie powtarzalne czynności w bazie danych były definiowane przy użyciu procedur składowanych.

Zdefiniowana przez użytkownika procedura jest tworzona w bieżącej bazie danych. Wyjątkiem są procedury tymczasowe — one zawsze są tworzone w folderze *tempdb*.

Definiowanie procedury składowanej wygląda następująco:

```
CREATE PROCEDURE Nazwa_procedury
(
@nazawa_parametru_1 typ_parametru, ..., @nazawa_parametru_n typ_parametru
)
AS
Treść_procedury
```

Przykład 5.102

Utworzona procedura będzie pobierała parametr z ceną książek i zwracała listę tylko tych książek, których ceny są niższe niż cena podana jako parametr.

```
CREATE PROCEDURE cena_ksiazki
(@cena_ks money)
AS
BEGIN
Print 'Książki o cenie niższej niż' + CAST (@cena_ks AS varchar (10));
SELECT tytul, [cena ksiazki]
FROM Ksiazki
WHERE [cena ksiazki] < @cena_ks;
END
GO
```

Pierwsza instrukcja tworzy procedurę o nazwie `cena_ksiazki` z parametrem wejściowym `@cena_ks` typu `money`.

Kolejne dwie instrukcje to instrukcje procedury. Pierwsza instrukcja to polecenie wydrukowania podanego tekstu połączonego z wartością parametru wejściowego, którego typ został zmieniony z `money` na `varchar (10)`. Druga to zapytanie, w którym jako warunek klauzuli `WHERE` występuje parametr wejściowy, zwracające listę wszystkich książek, które kosztują mniej niż wartość parametru wejściowego.

Procedury składowane mogą być wykonywane w podany sposób:

```
EXECUTE nazwa_procedury;
```

lub:

```
EXEC nazwa_procedury;
```

Przykład 5.103

Aby uruchomić procedurę dla książek tańszych niż 90 zł, należy wpisać polecenic `EXECUTE` w podanej postaci:

```
EXECUTE cena_ksiazki 90,00;
GO
```

Przykład 5.104

Mamy utworzyć procedurę, która po podaniu nazwiska i imienia klienta będzie zwracała jego dane.

```
CREATE PROCEDURE Dane_klienta
(
@Nazwisko nvarchar (50),  @Imie nvarchar (40)
)
AS
SET NOCOUNT ON;
SELECT nazwisko, imie, kod_pocztowy, miejscowosc, ulica, nr_domu, telefon
FROM Klient
WHERE nazwisko=@Nazwisko AND imie=@Imie;
GO
```

Procedura może zostać wykonana w następujący sposób:

```
EXECUTE Dane_klienta 'Kowalski', 'Jan';
```

lub:

```
EXEC Dane_klienta @Nazwisko =  'Kowalski', @Imie  = 'Jan';
```

Parametry do procedury można przekazywać anonimowo (bez określania, któremu parametrowi zostanie przypisana wartość). Jest to dobre rozwiązanie, gdy procedura ma jeden parametr wejściowy. Natomiast w przypadku, gdy tych parametrów jest więcej (jak w podanym przykładzie), lepiej stosować jawne przypisanie wartości do parametru.

Jeżeli procedura jest wywoływana jako pierwszy element w skrypcie, słowo kluczowe `EXECUTE` może zostać pominięte.

Użyta w procedurze składowanej opcja `SET NOCOUNT ON` zapobiega wyświetlaniu komunikatu pokazującego, na ilu wierszach działa instrukcja.

UWAGA

Komunikat `DONE_IN_PROC` jest wysyłany do użytkownika dla każdej instrukcji umieszczonej w procedurze składowanej. W przypadku procedur składowanych zawierających kilka instrukcji, które nie zwracają danych, oraz procedur zawierających pętle ustawienie opcji `SET NOCOUNT = ON` może znacznie przyspieszyć wykonywanie zapytań.

Przykład 5.105

```
CREATE PROCEDURE Sprzedaz_ksiazek
(
@Tyt_ksiazki (100)
)
 AS
SET NOCOUNT ON;
SELECT Ksiazki.tytul, SUM(liczba_egz) AS [Liczba książek]
 FROM Ksiazki INNER JOIN Rejestracja_zamowienia
ON Ksiazki.id_ksiazki = Rejestracja_zamowienia.id_ksiazki
WHERE Ksiazki.tytul=@Tyt_ksiazki
GROUP BY Ksiazki.tytul;
```

Po podaniu tytułu książki procedura zwróci liczbę sprzedanych książek o podanym tytule. Procedura może zostać wielokrotnie użyta w analizie statystycznej dotyczącej sprzedaży poszczególnych tytułów.

Wbudowane procedury

Oprócz definiowania własnych procedur pracę z bazą danych może usprawnić stosowanie procedur wbudowanych.

- `sys.sp_addrolemember` — ta procedura dodaje użytkownika do roli bazy danych. Parametry to `@rolename` i `@membername`.
- `sys.sp_adduser` — ta procedura dodaje użytkownika do bazy danych. Parametry to: `@loginame`, `@name_in_db`, `@grpname`.
- `sys.sp_catalogs` — ta procedura zwraca listę katalogów serwera. Parametr to `@server_name`.
- `sys.sp_columns` — ta procedura zwraca szczegółowe informacje dotyczące kolumn wybranej tabeli. Parametry to: `@table_name`, `@table_ovner`, `@table_qualifier`, `@column_name`, `@ODBCVer`.
- `sys.sp_databases` — ta procedura zwraca listę wszystkich baz danych dostępnych na serwerze wraz z informacją o ich rozmiarze.
- `sys.sp_help` — ta procedura zwraca szczegółowe informacje (typ, precyzja itp.) wskazanego obiektu bazodanowego (na przykład tabeli).

5.14.6. Funkcje składowane

Funkcje składowane działają podobnie jak procedury składowane, ale sposób ich wykorzystania jest zupełnie inny. Jedną z cech takiej funkcji jest to, że należy zadeklarować w niej typ zwracanego wyniku. Instrukcja `SELECT` wykorzystywana w procedurze może zwracać pewną wartość, ale nie musi, natomiast w funkcji jest to wymagane. Dlatego

zalecane jest, aby procedury były wykorzystywane do zapisywania, modyfikowania i usuwania rekordów, czyli działań, które nie zwracają wyniku, zaś funkcje do obliczeń typu: liczenie średniej, przeliczanie wartości itp.

Podstawowa postać funkcji to:

```
CREATE FUNCTION nazwa_funkcji
(
@nazwa_parametru_1 typ_parametru, ..., @nazwa_parametru_n typ_parametru
)
RETURNS typ_zwracany
AS
BEGIN
Treść_funkcji
RETURN wartość_zwracana
END;
```

Podczas definiowania funkcji trzeba zdefiniować nagłówek (nazwa, lista parametrów, typ zwracanej wartości) i treść funkcji. Ostatnim poleceniem treści powinna być instrukcja RETURN, która określa wartości zwracane przez funkcję.

Przykład 5.106

```
CREATE FUNCTION Data_pl
(
@pl_data datetime,
@odstep varchar (8)
)
RETURNS (nvarchar (32))
AS
BEGIN
RETURN
CONVERT (nvarchar (32), DATEPART (dd, @pl_data))
+@odstep
+ CONVERT (nvarchar (32), DATEPART (mm, @pl_data))
+@odstep
+ CONVERT (nvarchar (32), DATEPART (yy, @pl_data))
END
```

Funkcja zmienia format daty z 2012-10-20 na bardziej przyjazny 20-10-2012.

Jeżeli w tabeli *Klient* zostanie umieszczone pole *data urodzenia*, to wywołanie funkcji może mieć postać:

```
SELECT nazwisko, imie, data_pl([data urodzenia])
FROM Klient;
```

Przykład tej funkcji pokazuje, jak duże możliwości daje stworzenie własnej biblioteki ze skryptami zawierającymi funkcje, które będziemy mogli wykorzystywać w wielu tworzonych projektach.

Funkcje wbudowane

Podobnie jak w przypadku procedur, użytkownik może skorzystać z istniejących wbudowanych funkcji. Najczęściej wykorzystywane funkcje to:

- `AVG` — obliczanie wartości średniej;
- `SUM` — sumowanie wartości;
- `COUNT` — zliczanie liczby wystąpień;
- `GETDATE` — zwracanie wartości bieżącej daty i czasu;
- `DAY` — zwracanie dnia miesiąca jako liczby całkowitej;
- `YEAR` — zwracanie roku jako liczby całkowitej;
- `MONTH` — zwracanie miesiąca jako liczby całkowitej.

5.14.7. Wyzwalacze

Wyzwalacze (ang. *triggers*) to specjalny rodzaj procedur składowanych, które są automatycznie wykonywane, gdy na serwerze wystąpi określone zdarzenie. Tymi zdarzeniami są operacje przeprowadzane przez użytkownika. Dopiero po wykonaniu instrukcji uruchamiany jest wyzwalacz. Polecenia wykonywane w ramach wyzwalacza są traktowane jako transakcje rozpoczęte jawnie lub niejawnie przez użytkownika.

Głównym przeznaczeniem wyzwalaczy jest wymuszenie integralności danych. Tylko właściciel tabeli może utworzyć powiązany z nią wyzwalacz. Praw do tworzenia wyzwalaczy nie można nikomu przekazać. Możliwe jest definiowanie dla tabeli dowolnej liczby powiązanych z nią wyzwalaczy. Nie można definiować wyzwalaczy tabel tymczasowych. Wyzwalacze nie zwracają żadnych danych. Są automatycznie uruchamiane w odpowiedzi na działania użytkownika.

Do tworzenia wyzwalaczy służy instrukcja `CREATE TRIGGER`.

W języku SQL występują trzy rodzaje wyzwalaczy:

- wyzwalacze DML,
- wyzwalacze DDL,
- wyzwalacze logowania.

Wyzwalacze DML

Wyzwalacze DML są wywoływane, gdy zostanie wykonane jedno z poleceń języka DML, na przykład `INSERT`, `UPDATE` lub `DELETE`. Takie wyzwalacze są użyteczne:

- przy modyfikowaniu danych w powiązanych tabelach (jednak lepsze efekty uzyskamy, stosując w tej sytuacji kaskadowe więzy integralności);
- przy egzekwowaniu warunków ograniczających zakres danych wprowadzanych do kolumny zdefiniowany w instrukcjach `INSERT`, `UPDATE` i `DELETE` (w przeciwieństwie do warunków ograniczających, definiowanych przy użyciu atrybutu `CHECK`, wyzwalacze DML mogą odwoływać się do kolumn w innych tabelach);
- przy podejmowaniu działań na podstawie odczytu stanu tabeli przed modyfikacją i po modyfikacji.

Można wyróżnić następujące typy wyzwalaczy DML:

- Wyzwalacze *After* — są uruchamiane po instrukcji generującej zdarzenie. Mogą być definiowane tylko dla tabel. Wyzwalacz zostanie uruchomiony tylko wtedy, gdy wszystkie operacje generujące zdarzenie zostały wykonane pomyślnie.
- Wyzwalacze *Instead of* — są wykonywane zamiast instrukcji generującej zdarzenie. Są definiowane dla widoków opartych na tabelach i aktualizują zawartość tych widoków.
- Wyzwalacze *CLR* — mogą być wyzwalaczami *After* lub *Instead of*. Mogą być również wyzwalaczami DLL. Zamiast wykonywania transakcji dla procedury składowanej wykonują kod zewnętrzny utworzony w .NET Framework i przesłany do serwera SQL Server.

Postać instrukcji `CREATE TRIGGER` dla wyzwalaczy DML:

```
CREATE TRIGGER nazwa_wyzwalacza ON nazwa_tabeli/widoku
{FOR/AFTER/INSTEAD OF} {INSERT, DELETE, UPDATE}
AS
Polecenia/EXTERNAL NAME nazwa_wywoływanej_procedury
```

gdzie:

- `nazwa_tabeli/widoku` — tabela lub widok, dla którego wykonywany jest wyzwalacz. Podanie nazwy tabeli lub wyzwalacza jest opcjonalne. Widok może wystąpić tylko w wyzwalaczu *Instead of*. Wyzwalaczy DML nie można tworzyć dla tabel tymczasowych.
- `FOR/AFTER` — wyzwalacz zostanie uruchomiony tylko wtedy, gdy wszystkie operacje generujące zdarzenie zostały wykonane pomyślnie. Zdefiniowane kaskadowe aktualizowanie więzów integralności oraz warunki ograniczające również muszą dać pozytywne efekty.
- `INSTEAD OF` — wyzwalacz jest wykonywany zamiast generującej go instrukcji. Tylko jeden wyzwalacz tego typu może zostać zdefiniowany dla tabeli lub widoku.

Natomiast można zdefiniować widoki dla widoku i każdy z nich może posiadać swój wyzwalacz typu INSTEAD OF.

- INSERT, UPDATE i DELETE — określają instrukcje modyfikacji danych, które mogą uruchomić wyzwalacz dla określonej tabeli lub dla określonego widoku. Co najmniej jedna z instrukcji musi zostać wybrana. W wyzwalaczach typu INSTEAD OF instrukcja DELETE nie jest dozwolona dla tabel, które mają zdefiniowaną opcję kaskadowego usuwania danych. Podobnie jak instrukcja INSERT, nie jest ona dozwolona w przypadku tabel, które mają zdefiniowaną opcję kaskadowego aktualizowania danych.

Przykład 5.107

```
CREATE TRIGGER modyfikuj
ON dbo.klient
AFTER INSERT, UPDATE
AS RAISERROR ('Uwaga! Zmiana danych', 16, 10);
```

Wyzwalacz wysyła do użytkownika komunikat, gdy ktoś próbuje dodać dane do tabeli *Klient* lub zmienić je.

RAISERROR(...) to funkcja generująca komunikat o błędzie.

Wyzwalacze DDL

Wyzwalacze DDL są uruchamiane w wyniku zdarzeń zachodzących po wykonaniu poleceń języka DDL. Dotyczy to poleceń CREATE, ALTER i DROP. Często są wykorzystywane do sprawdzania stanu odpowiedzi wysyłanych przez procedury składowane do systemu. Mogą być wykorzystywane do zadań administracyjnych, takich jak audyt czy regulowanie operacji bazodanowych.

Są użyteczne, gdy trzeba:

- uniemożliwić wprowadzenie zmian w schemacie bazy danych,
- wykonać działania w bazie danych w odpowiedzi na zmiany w schemacie bazy,
- zapamiętać zmiany w schemacie bazy danych.

Wyzwalacze DDL są uruchamiane tylko po poleceniach języka DDL. Nie mogą być używane wyzwalacze INSTEAD OF.

Postać instrukcji CREATE TRIGGER dla wyzwalaczy DDL:

```
CREATE TRIGGER nazwa_wyzwalacza ON ALL SERVER/DATABASE
{FOR /AFTER} {nazwa_zdarzenia}
AS
Polecenia/EXTERNAL NAME nazwa_wywoływanej_procedury
```

gdzie:

DATABASE — działanie wyzwalacza dotyczy bieżącej bazy danych;

ALL SERVER — działanie wyzwalacza dotyczy bieżącego serwera;

nazwa_zdarzenia — nazwa zdarzenia, po którym nastąpi wywołanie wyzwalacza.

Przykładowe zdarzenia wykorzystywane przez wyzwalacze DDL:

- CREATE_TABLE,
- ALTER_ TABLE,
- CREATE_SCHEMA,
- GRANT_DATABASE,
- CREATE_INDEX,
- DROP_INDEX.

Słowa kluczowe zawsze są oddzielone znakiem podkreślenia _.

Przykład 5.108

```
CREATE TRIGGER blokuj
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
   PRINT 'Musisz wyłączyć wyzwalacz "blokuj", aby zmienić lub usunąć tabelę!'
   ROLLBACK;
```

Zdefiniowany wyzwalacz zadziała zawsze, gdy w bazie danych wystąpi zdarzenie DROP_ TABLE lub ALTER_TABLE. Zdarzenia te są generowane instrukcjami DROP TABLE i ALTER TABLE.

Wyzwalacze logowania

Wyzwalacze logowania uruchamiają się w odpowiedzi na zdarzenia związane z logowaniem. Uruchamiane są po zakończeniu fazy uwierzytelnienia logowania, ale przed rozpoczęciem sesji użytkownika. Nie są uruchamiane, gdy logowanie nie powiedzie się.

Postać instrukcji CREATE TRIGGER dla wyzwalaczy logowania:

```
CREATE TRIGGER nazwa_wyzwalacza ON ALL SERVER
{FOR /AFTER} LOGON
AS
Polecenia/EXTERNAL NAME nazwa_wywoływanej_procedury
```

Przykład 5.109

```
USE master;
GO
CREATE LOGIN Marek WITH PASSWORD = 'Mar!@#ek (MUST_CHANGE M@r%k',
    = CHECK_EXPIRATION ON;
GO
GRANT VIEW SERVER STATE TO login_test;
GO
CREATE TRIGGER limit
ON ALL SERVER WITH EXECUTE AS 'Marek'
FOR LOGON
AS
BEGIN
IF ORIGINAL_LOGIN () = 'Marek' AND
    (SELECT COUNT (*) FROM sys.dm_exec_sessions
            WHERE is_user_process = 1 AND
                original_login_name = 'login_test')> 3
    ROLLBACK;
    END;
```

W podanym przykładzie wyzwalacz logowania zabrania logowania do serwera SQL zainicjowanego przez *login_test* wtedy, gdy istnieją już trzy sesje logowania utworzone przez tego użytkownika.

Stosując ten mechanizm, można kontrolować aktywność logowania do serwera i ograniczać liczbę sesji dla określonego loginu.

Uprawnienia

Do tworzenia wyzwalaczy DML wymagane są uprawnienia `ALTER` dla tabeli lub widoku, dla którego tworzony jest wyzwalacz.

Do tworzenia wyzwalaczy DDL dla serwera (`ON ALL SERVER`) oraz wyzwalaczy logowania wymagane są uprawnienia `CONTROL SERVER` na serwerze.

Do tworzenia wyzwalaczy DDL dla bazy danych (`ON DATABASE`) wymagane są uprawnienia `ALTER DATABASE`.

Załącznik 1. Tabele wchodzące w skład bazy danych ksiegarnia_internetowa

Tabela Klient

- *id_klienta*,
- *nazwisko*,
- *imie*,
- *kod_pocztowy*,
- *miejscowosc*,
- *ulica*,
- *nr_domu*,
- *PESEL*,
- *telefon*,
- *adres_e_mail*.

Tabela Ksiazki

- *id_ksiazki*,
- *tytul*,
- *id_autora*,
- *cena*,
- *wydawnictwo*,
- *temat*,
- *miejsce_wydania*,
- *jezyk_ksiazki*,
- *opis*.

Tabela Zamowienia

- *id_zamowienia*,
- *id_klienta*,
- *[data zlozenia zamowienia]*,
- *[data wyslania]*,
- *[koszt wysylki]*,
- *nr_faktury*.

Tabela Autor

- *id_autora*,

- *nazwisko*,
- *imie*,
- *narodowosc*,
- *okres_tworzenia*,
- *jezyk*,
- *[rodzaj tworczosci]*,
- *osiagniecia*.

Tabela Rejestracja_zamowienia

- *id_zamowienia*,
- *id_ksiazki*,
- *liczba_egz*.

Tabela Faktura

- *nr_faktury*,
- *sposob_platnosci*,
- *[data wystawienia faktury]*.

Skrypty tworzące tabele dla bazy ksiegarnia_internetowa

Tabela Autor

```
CREATE TABLE Autor
(
    id_autora INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    nazwisko nvarchar (50) NOT NULL,
    imie nvarchar (30) NOT NULL,
    narodowosc nvarchar (30),
    okres_tworzenia nvarchar (35),
    jezyk nvarchar (30),
    [rodzaj tworczosci] nvarchar (35),
    osiagniecia nvarchar (200)
);
```

Tabela Faktura

```
CREATE TABLE Faktura
(
    nr_faktury int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    sposob_platnosci nvarchar (30),
    [data wystawienia faktury] datetime
);
```

Tabela Klient

```
CREATE TABLE Klient
(
    id_klienta int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    nazwisko nvarchar (60) NOT NULL,
    imie nvarchar (40) NOT NULL,
    kod_pocztowy nvarchar (6),
    miejscowosc nvarchar (50) DEFAULT 'Warszawa',
    ulica nvarchar (50),
    nr_domu nvarchar (7),
    PESEL nvarchar (11) NOT NULL,
    telefon nvarchar (12) UNIQUE,
    adres_e_mail nvarchar (70)
);
```

Tabela Ksiazki

```
CREATE TABLE Ksiazki
(
    id_ksiazki int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    tytul nvarchar (100) NOT NULL,
    id_autora int NOT NULL,
    cena money,
    wydawnictwo nvarchar (20),
    temat nvarchar (30),
    miejsce_wydania nvarchar (28),
    jezyk_ksiazki nvarchar (15),
    opis nvarchar (100),
    rok_wydania nvarchar (4)
```

```
);
```

Tabela Rejestracja_zamowienia

```
CREATE TABLE Rejestracja_zamowienia
(
    id_zamowienia int,
    id_ksiazki int,
    liczba_egz int
);
```

Tabela Zamowienia

```
CREATE TABLE Zamowienia
(
    id_zamowienia int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    [data zlozenia zamowienia] datetime,
    [data wyslania] datetime,
    [koszt wysylki] money,
    id_klienta int NOT NULL,
    id_faktury int NOT NULL
);
```

Skrypty tworzące tabele dla bazy ksiegarnia_internetowa (wykorzystane do tworzenia diagramu bazy danych)

```
CREATE TABLE Ksiazki
(
    id_ksiazki int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    tytul nvarchar (100) NOT NULL,
    id_autora INT REFERENCES Autor (id_autora),
    cena money,
    wydawnictwo nvarchar (20),
    temat nvarchar (30),
    miejsce_wydania nvarchar (28),
    jezyk_ksiazki nvarchar (15),
    opis nvarchar (100),
    rok_wydania nvarchar (4)
```

```
);

CREATE TABLE Zamowienia
(
    id_zamowienia int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    [data zlozenia zamowienia] datetime,
    [data wyslania] datetime,
    [koszt wysylki] money,
    id_klienta INT REFERENCES Klient (id_klienta),
    id_faktury INT REFERENCES faktura (nr_faktury)
);

CREATE TABLE Klient
(
    id_klienta int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    nazwisko nvarchar (60) NOT NULL,
    imie nvarchar (40) NOT NULL,
    kod_pocztowy nvarchar (6),
    miejscowosc nvarchar (50) DEFAULT 'Warszawa',
    ulica nvarchar (50),
    nr_domu nvarchar (7),
    PESEL nvarchar (11) NOT NULL,
    telefon nvarchar (12) UNIQUE,
    adres_e_mail nvarchar (70)
);

CREATE TABLE Faktura
(
    nr_faktury int IDENTITY (1, 1) NOT NULL PRIMARY KEY,
    sposob_platnosci nvarchar (30),
    [data wystawienia faktury] datetime
);

CREATE TABLE Autor
(
    id_autora INT   IDENTITY (1, 1) NOT NULL PRIMARY KEY,
```

```
    nazwisko nvarchar (50) NOT NULL,
    imie nvarchar (30) NOT NULL,
    narodowosc nvarchar (30),
    okres_tworzenia nvarchar (35),
    jezyk nvarchar (30),
    [rodzaj tworczosci] nvarchar (35),
    osiagniecia nvarchar (200)
);
```

```
CREATE TABLE Rejestracja_zamowienia
(
    id_zamowienia INT REFERENCES Zamowienia (id_zamowienia),
    id_ksiazki INT REFERENCES Ksiazki (id_ksiazki),
    liczba_egz int
);
```

Widoki

Sprzedaż książek

```
SELECT dbo.klient.nazwisko AS K_Nazw, dbo.klient.imie AS K_Im, dbo.Ksiazki.
tytul, dbo.Autor.nazwisko AS A_Nazw, dbo.Autor.imie AS A_Im, dbo.Ksiazki.cena
FROM dbo.Klient INNER JOIN dbo.Zamowienia
ON dbo.Klient.id_klienta = dbo.Zamowienia.id_klienta
INNER JOIN dbo.Rejestracja_zamowienia
ON dbo.Zamowienia.id_zamowienia = dbo.Rejestracja_zamowienia.id_zamowienia
INNER JOIN dbo.Ksiazki
ON dbo.Rejestracja_zamowienia.id_ksiazki = dbo.Ksiazki.id_ksiazki
INNER JOIN dbo.Autor
ON dbo.Ksiazki.id_autora = dbo.Autor.id_autora
```

Książki i autorzy

```
SELECT dbo.Autor.nazwisko, dbo.Autor.imie, dbo.Ksiazki.tytul, dbo.Ksiazki.
cena
FROM dbo.Autor INNER JOIN
                dbo.Ksiazki ON dbo.Autor.id_autora = dbo.Ksiazki.id_autora
WHERE (dbo.Ksiazki.rok_wydania > 2010)
```

PYTANIA KONTROLNE

1. Wymień podstawowe cechy języka SQL.
2. Z jakich elementów składa się język SQL?
3. Jak wygląda hierarchia obiektów bazy danych w języku SQL?
4. Jakiego typu działania są realizowane za pomocą języka DDL?
5. Jakiego typu działania są realizowane za pomocą języka DML?
6. Za pomocą jakich poleceń języka SQL realizowane są połączenia zewnętrzne?
7. Jaką rolę w zapytaniu spełniają podzapytania (zapytania zagnieżdżone)?
8. Kiedy należy stosować transakcje?
9. Wymień właściwości transakcji.
10. Na czym polega współbieżność?
11. Na czym polega izolowanie transakcji?
12. Co to jest procedura składowana?
13. Jaką rolę w języku T-SQL spełniają wyzwalacze?

ZADANIE

Korzystając z wybranego serwera baz danych i języka SQL, utwórz dla bazy danych *Moja_szkoła* zapytania określające sposób wybierania i przetwarzania danych zgromadzonych w tabelach. Wykorzystując narzędzia języka SQL, zbuduj pełną obsługę tej bazy.

6

Administrowanie serwerami baz danych

6.1. Wprowadzenie

Serwer baz danych to program, który zarządza bazami danych. Jego zadaniem jest bezpieczne przechowywanie danych, kontrolowanie dostępu do danych, umożliwienie wielu użytkownikom jednoczesnego korzystania z przechowywanych danych oraz umożliwienie przetwarzania danych za pomocą języka SQL.

6.1.1. Zadania administratora baz danych

Podstawowym celem działań administratora baz danych powinno być zapewnienie niezawodności i bezawaryjnego działania systemu. Wiąże się to z zarządzaniem dostępem użytkowników do baz danych, zabezpieczeniem przed częściową lub całkowitą utratą danych oraz monitorowaniem pracy serwera baz danych.

Bezpieczeństwo danych to:

- uwierzytelnienie użytkownika,
- autoryzacja użytkownika,
- zarządzanie dostępem do baz danych,
- replikacja bazy danych,
- tworzenie kopii zapasowych.

Administrowanie bazami danych na poziomie użytkownika to:

- dodawanie nowego konta z różnymi uprawnieniami,
- usuwanie konta,

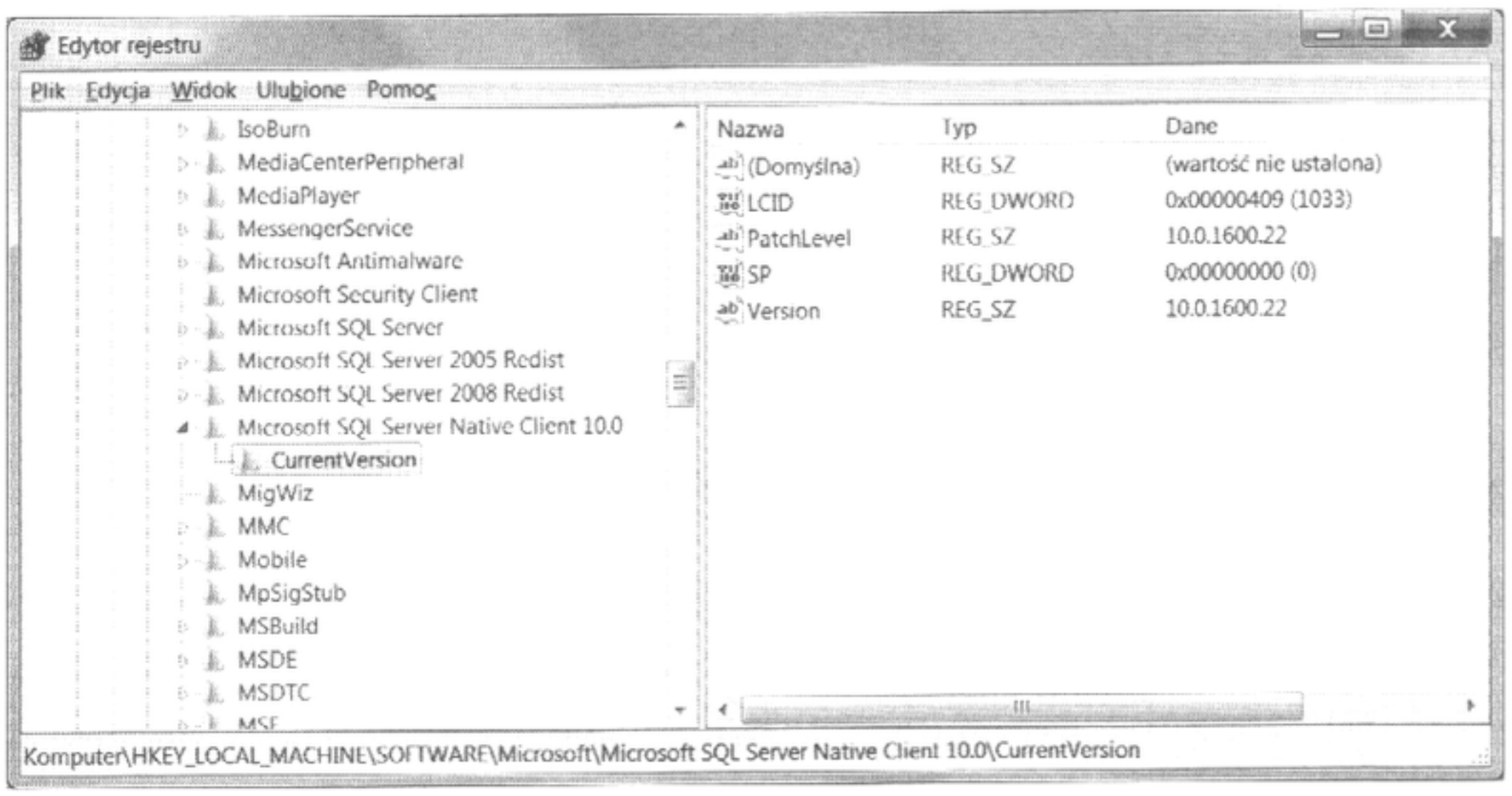
- wprowadzanie ograniczeń i nadawanie przywilejów dla konta,
- zarządzanie bazą danych przez definiowanie ról,
- dołączanie lub odłączanie baz danych,
- zakładanie baz danych,
- konserwacja i aktualizacja baz danych,
- monitorowanie bieżącego działania baz danych.

6.2. MS SQL Server

6.2.1. Konfiguracja serwera w systemie operacyjnym

W systemie Windows ustawienia konfiguracyjne serwera SQL są przechowywane w kluczach rejestru systemu (rysunek 6.1). Są to:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server
HKEY_CURRENT_USER\Software\Microsoft\Microsoft SQL Server
```



Rysunek 6.1. Konfiguracja klienta w rejestrze systemowym

Konfiguracja serwisów (instancji) jest przechowywana w następującym kluczu:

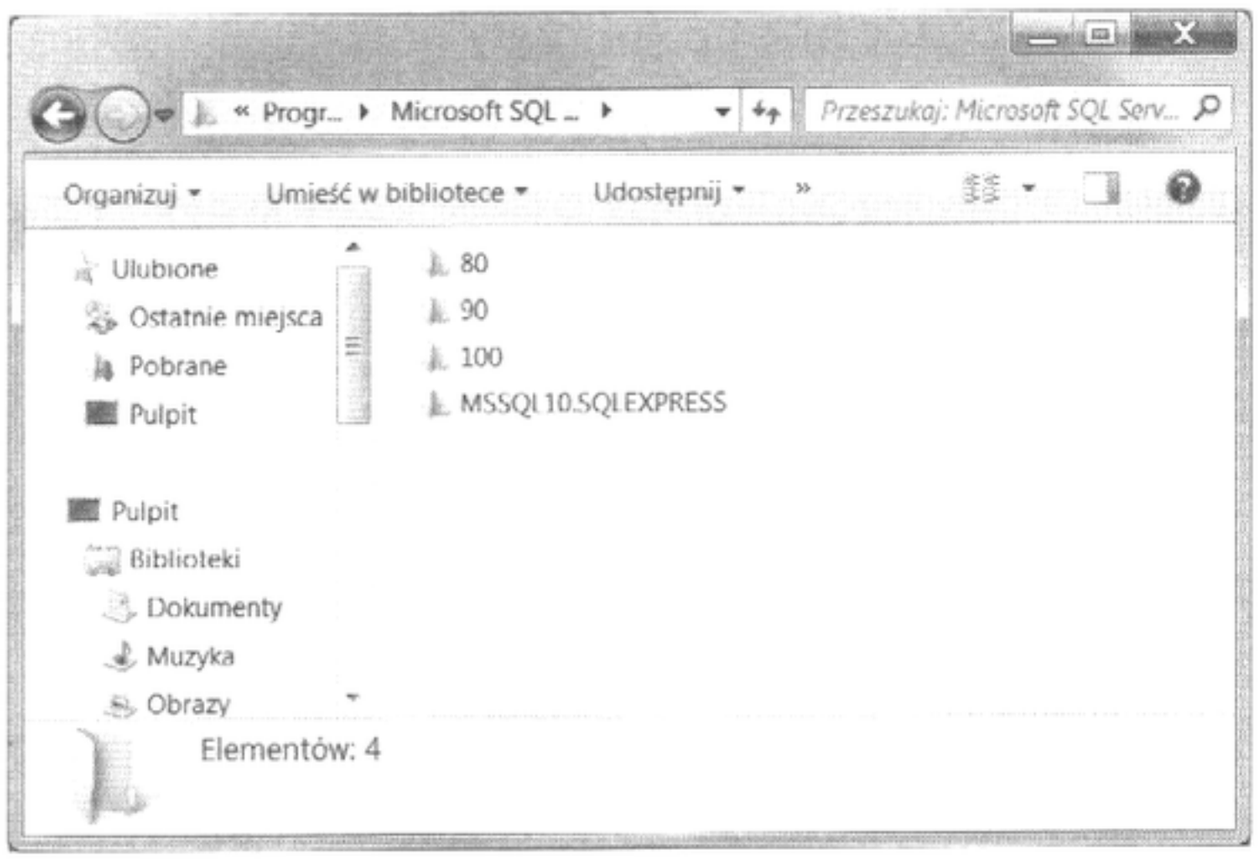
```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
```

Konfiguracja klienta jest przechowywana w następującym podkluczu:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Native Client
```

Domyślnym folderem instalowania serwera SQL 2008 jest *C:\Program Files\Microsoft SQL Server* (rysunek 6.2). Znajdują się tam foldery zawierające biblioteki i programy skojarzone (*80*, *90*, *100*) oraz folder przechowujący pliki instancji serwera (*MSSQL10.SQLEXPRESS*).

Rysunek 6.2.
Foldery serwera SQL 2008

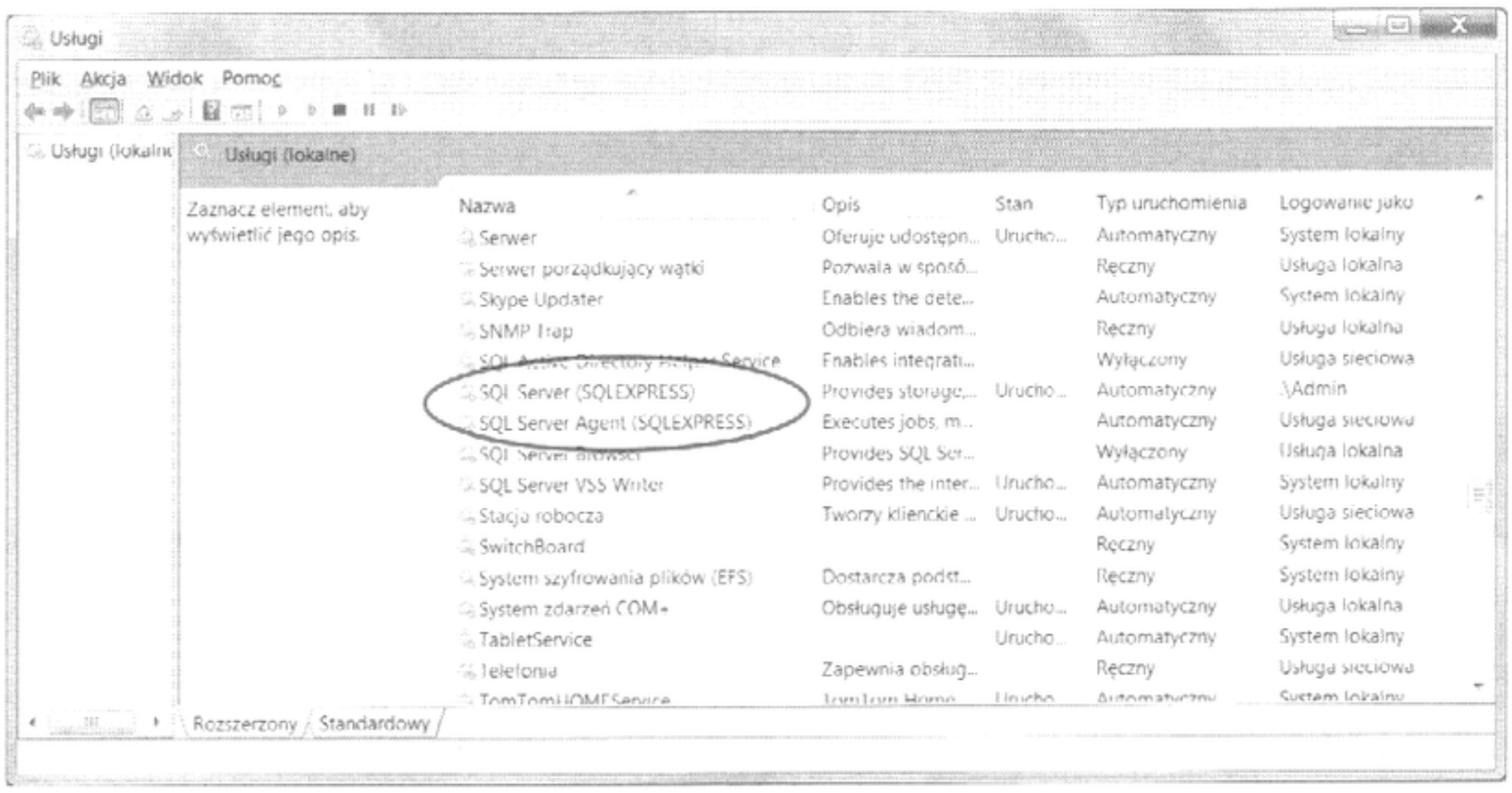


Istotne z punktu widzenia administratora mogą być foldery:

- *\Backup* — przeznaczony na kopie zapasowe;
- *\Binn* — zawiera biblioteki i narzędzia serwera;
- *\Data* — zawiera pliki baz danych i dziennika serwera;
- *\Log* — zawiera dziennik błędów serwera;
- *\repldata* — używany w trakcie replikacji;
- *\Tools\Binn* — zawiera biblioteki i programy narzędziowe serwera;
- *\Sdk* — zawiera pliki dla programistów (biblioteki, pliki nagłówkowe);
- *\Shared* — zawiera zestaw bibliotek współdzielonych;
- *\Com* — zawiera pliki wspólne dla wszystkich kopii serwera oraz innych narzędzi usługowych.

Uruchamianie serwera i logowanie

Uruchamianie i zatrzymanie serwera można wykonać za pomocą systemowych narzędzi administracyjnych przez zatrzymanie lub uruchomienie usługi (rysunek 6.3).



Rysunek 6.3. Uruchamianie serwera SQL 2008 jako usługi

Można go również uruchomić lub zatrzymać, wpisując w wierszu poleceń jedno z poniższych poleceń:

```
NET START MSSQL$SQLEXPRESS
NET STOP MSSQL$SQLEXPRESS
```

gdzie `MSSQL$SQLEXPRESS` jest nazwą instancji (serwisu).

6.2.2. Tryby uwierzytelnienia

Domyślnie tylko administratorzy komputera mają pełny dostęp do serwera SQL Server.

Podczas instalowania serwera SQL Server istnieje możliwość wybrania trybu uwierzytelnienia. Możliwe są dwa tryby: *Windows Authentication* (uwierzytelnienie Windows) oraz *Mixed mode* (tryb mieszany). Tryb uwierzytelnienia Windows akceptuje uwierzytelnienie systemu Windows i wyłącza uwierzytelnienie przez SQL Server. Logowanie w trybie *Windows Authentication* jest automatyczne. Nie trzeba wprowadzać nazwy użytkownika i hasła. Jeżeli użytkownik należy do grupy administratorów lokalnych komputera, zostanie mu przyznana automatycznie rola **sysadmin**. Uwierzytelnienia systemu Windows nie można wyłączyć.

Tryb mieszany akceptuje uwierzytelnienie systemu Windows i uwierzytelnienie przez SQL Server. Ten tryb logowania umożliwia logowanie na specjalne konto SQL Server. Domyślnie zawsze istniejącym w bazie użytkownikiem jest **sa** (*System Administrator*), który należy do roli **sysadmin**.

Uwierzytelnienie za pomocą systemu Windows jest bezpieczniejsze od uwierzytelnienia SQL Server. Uwierzytelnienie to wykorzystuje specjalny protokół zabezpieczeń (*Kerberos*), który zapewnia egzekwowanie zasad haseł, wsparcie dla blokady kont i obsługuje wygaśnięcia hasła. Połączenie dokonywane za pomocą uwierzytelnienia systemu Windows jest też nazywane zaufanym połączeniem.

Podczas korzystania z uwierzytelnienia przez SQL Server (tryb mieszany) loginy są tworzone i przechowywane na serwerze SQL Server. Użytkownicy korzystający z takiego uwierzytelnienia muszą podawać swój login i hasło za każdym razem, gdy chcą się połączyć z bazą.

Rodzaj uwierzytelnienia dla użytkownika bazy danych zależy od konfiguracji systemu komputerowego.

6.2.3. Autoryzacja i autentykacja

Za bezpieczeństwo przechowywanych w bazach danych, jak również za bezpieczny dostęp do nich odpowiadają autentykacja oraz autoryzacja.

Aby użytkownik mógł się zalogować i połączyć z bazą danych, musi mieć konto na serwerze. SQL Server umożliwia dostęp do bazy użytkownikom posiadającym lokalne lub domenowe konto w systemie Windows. W takiej sytuacji podczas logowania do

systemu operacyjnego użytkownik jest identyfikowany, a podczas łączenia się z serwerem SQL Server autentykacja jest pomijana.

Drugą metodą logowania jest posiadanie loginu na serwerze SQL Server. W takim przypadku łączenie z instancją SQL Server wymaga użycia tego loginu.

Model zabezpieczeń w serwerze SQL Server opiera się na następującej zasadzie — tylko uwierzytelnieni użytkownicy mogą się połączyć z serwerem bazodanowym.

Rozpoczynając pracę z bazą danych, użytkownik musi podać swój login i wpisać odpowiednie hasło. Jest to mechanizm *uwierzytelniania użytkownika*, czyli *autentykacja użytkownika*. Ponadto, jeżeli użytkownik próbuje wykonać jakąkolwiek operację na bazie danych, serwer sprawdza, czy ma on do tego wystarczające uprawnienia. Zabronienie korzystania lub zezwolenie użytkownikowi na korzystanie z niektórych zasobów, czyli *autoryzacja użytkownika*, są przeprowadzane automatycznie.

Podczas instalowania serwera bazodanowego automatycznie tworzone są loginy dla grupy lokalnych administratorów systemu operacyjnego (wszyscy administratorzy systemu), dla konta użytkownika, na którym to koncie instalowany jest serwer, oraz gdy został wybrany tryb mieszany dla użytkownika **sa** (*System Administrator*).

Wszyscy oni mają nieograniczone uprawnienia na serwerze bazodanowym. Pozostali użytkownicy serwera bazodanowego muszą mieć zdefiniowany login.

Oto instrukcja tworząca login na serwerze bazodanowym:

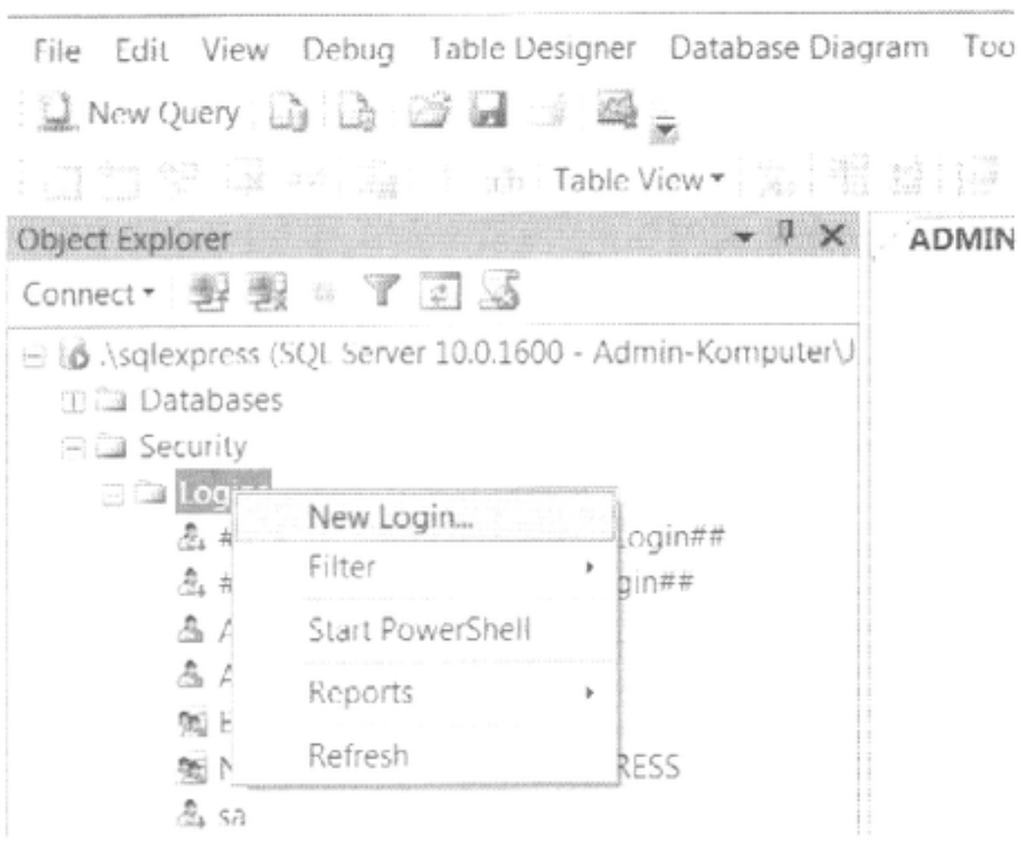
```
CREATE LOGIN [Komp\User1]
FROM WINDOWS;
```

W tak zapisanej instrukcji *Komp* to nazwa komputera, a *User1* to nazwa konta użytkownika.

Login można też utworzyć z poziomu SQL Server Management Studio (rysunek 6.4).

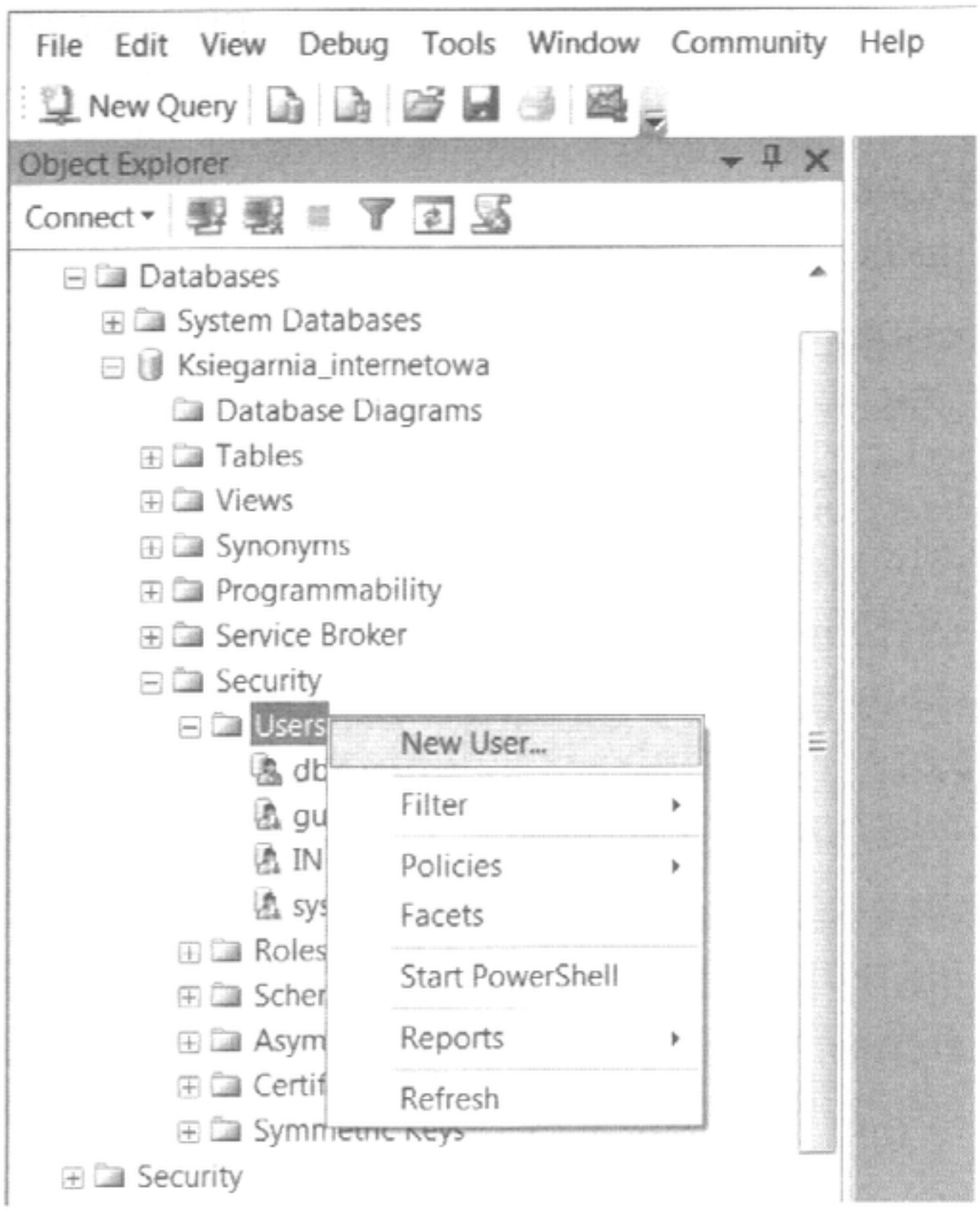
Po zdefiniowaniu loginu użytkownika użytkownik ma dostęp do serwera baz danych, ale nie ma uprawnień, aby uzyskać dostęp do bazy danych. Aby uzyskać dostęp do bazy danych, należy z poziomu wybranej bazy danych utworzyć użytkownika bazy i połączyć go z loginem. Oto instrukcja tworząca użytkownika bazy:

```
CREATE USER Janek
FOR LOGIN [Komp\User1]
WITH PASSWORD = 'hasło'
CHECK_POLICY = ON
```



Rysunek 6.4. Tworzenie nowego loginu z poziomu SQL Server Management Studio

Użytkownika bazy danych można utworzyć za pomocą SQL Server Management Studio. Należy pamiętać, aby tworzyć go z poziomu wybranej bazy danych (rysunek 6.5).



Rysunek 6.5. Tworzenie użytkownika bazy danych w SQL Server Management Studio

Login i user to dwa odrębne elementy systemu baz danych. Login zapewnia dostęp do instancji SQL Server, natomiast user to użytkownik, któremu można nadać uprawnienia do wybranych obiektów konkretnej bazy danych. Określony login może mieć przypisanych wielu użytkowników. Dzięki temu przy użyciu jednego konta logowania użytkownik może mieć zdefiniowane różne uprawnienia do różnych baz danych.

Do tworzonego loginu możemy przypisać domyślną bazę danych, z którą użytkownik będzie mógł automatycznie się połączyć:

```
CREATE LOGIN [Komp\User1]
FROM WINDOWS
WITH DEFAULT_DATABASE=Nazwa_bazy;
GO
```

Po utworzeniu loginu powiązanego z kontem wystarczy, że użytkownik zaloguje się do systemu, aby miał dostęp do bazy.

Z poziomu serwera bazodanowego można też utworzyć konto użytkownika w systemie Windows, wykonując instrukcję `CREATE USER`.

Przykład 6.1

```
USE master;
CREATE USER Pracownik1
FOR LOGIN [Komp\Pracownik1];
```

Jeden login może zostać powiązany z wieloma dostępnymi na serwerze bazami danych.

6.2.4. Zarządzanie bazami danych

SQL Server przechowuje informacje o istniejących bazach użytkowników w specjalnej bazie systemowej *master*. Z poziomu tej bazy jest realizowane zarządzanie pozostałymi bazami danych. Aby połączyć się z dowolną bazą, należy wykonać instrukcję `USE`.

Połączenie z bazą *master* uzyskamy, wpisując:

```
USE master;
```

Do połączenia z bazą *master* wymagane są uprawnienia administratora.

Systemowe bazy danych

Podczas instalowania serwera SQL Server tworzone są systemowe bazy danych, którymi można zarządzać za pomocą narzędzi serwera.

master

Baza *master* jest najważniejszą bazą danych w systemie. Zawiera podstawowe dane dla serwera oraz informacje o wszystkich utworzonych w systemie bazach danych. Ze względu na bezpieczeństwo powinno się regularnie tworzyć jej kopie zapasowe.

model

Baza *model* jest szablonem bazy danych. W momencie tworzenia nowej bazy danych jest tworzona kopia bazy *model*, a następnie kopia ta jest dostosowywana do wymagań tworzonej bazy. Jeżeli baza *model* zostanie zmodyfikowana, na przykład przez dodanie jakiegoś obiektu, to wszystkie tworzone nowe bazy danych będą zawierały ten obiekt.

tempdb

Baza *tempdb* jest przeznaczona do wykonywania operacji, które wymagają tymczasowej przestrzeni. Jest wykorzystywana przy operacjach typu sortowanie, złączenie itp.

msdb

Baza *msdb* przechowuje informacje na temat replikacji, zdarzeń, zadań (na przykład tworzenia kopii zapasowych, odzyskiwania danych).

Tworzenie bazy danych

Do tworzenia bazy danych służy polecenie `CREATE DATABASE`. W trakcie tworzenia bazy danych można określić takie parametry, jak: rozmiar bazy, jej nazwę logiczną oraz lokalizację pliku z danymi.

Przykład 6.2

```
CREATE DATABASE Towary ON PRIMARY
(NAME= Sprzedaz_towarow,
FILENAME = "C:Program Files\Microsoft SQL Server\MSSQL\DATA\Towary.mdf",
SIZE = 10MB,
MAXSIZE = 30MB,
FILEGROWTH = 5MB)
```

Została utworzona baza danych *Towary*. Dla powstałej bazy została określona nazwa logiczna, podana lokalizacja pliku z danymi oraz jej rozmiar.

Zmiana parametrów bazy danych

Parametry bazy danych, takie jak rozmiar lub nazwa, można zmienić poleceniem `ALTER DATABASE`.

Przykład 6.3

```
ALTER DATABASE Towary
ADD FILE
(NAME = Sprzedaz_towarowl,
FILENAME = "C:Program Files\Microsoft SQL Server\MSSQL\DATA\Towary1.mdf"",
SIZE - 5,
MAXSIZE - 15,
FILEGROWTH = 3)
```

Poleceniem `ALTER DATABASE` można do istniejącej bazy danych dodać dziennik transakcji.

Przykład 6.4

```
ALTER DATABASE Towary
ADD LOG FILE
(NAME = Sprzedaz_Dziennik,
FILENAME = "C:Program Files\Microsoft SQL Server\MSSQL\DATA\Towary.mdf",
SIZE = 2,
MAXSIZE = 6,
FILEGROWTH = 1)
```

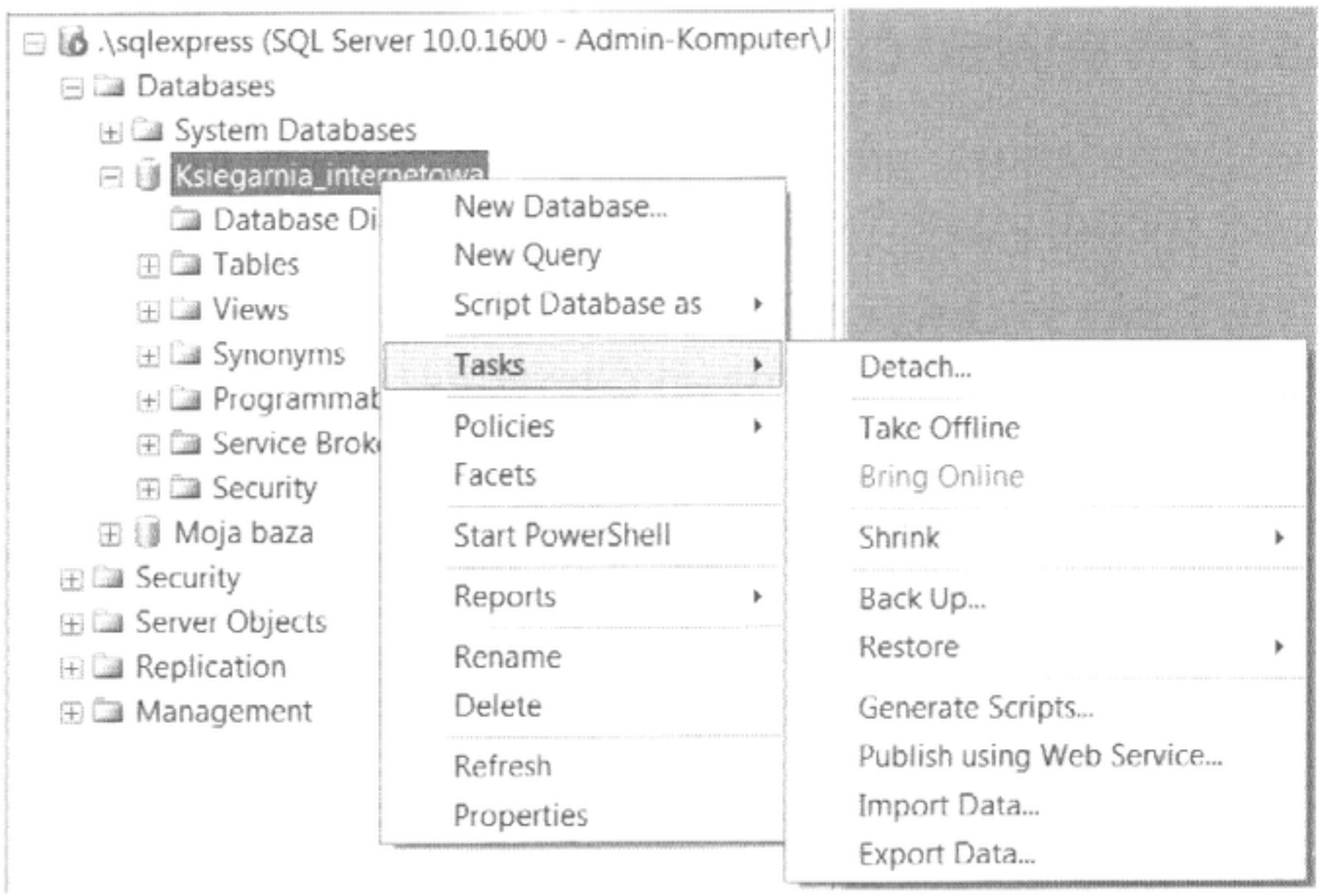
W SQL Server Management Studio w menu kontekstowym wybranej bazy danych znajduje się opcja *Tasks* (rysunek 6.6) zawierająca polecenia, które pomogą w administrowaniu bazą danych.

Detach... służy do odłączania bazy danych od danej instancji serwera. Jest to narzędzie przydatne, gdy zachodzi potrzeba przeniesienia bazy danych z jednego serwera na inny.

Shrink służy do zmniejszenia rozmiarów bazy danych. Przy jego użyciu można zmniejszyć rozmiary albo pojedynczych plików używanych przez bazę, albo całej bazy. Zmniejszanie rozmiaru bazy danych ma na celu zwolnienie miejsca niewykorzystywanego przez bazę.

Back Up... umożliwia zapisanie bazy danych na dysku lokalnym serwera.

Rysunek 6.6.
Narzędzia zarządzania bazą danych



Restore służy do przywrócenia bazy danych na serwerze SQL z pliku kopii zapasowej.

Generate Scripts... to polecenie, które po uruchomieniu kreatora pozwala na wygenerowanie skryptów bazy danych. W zależności od wybranych opcji można określić, jakie elementy bazy danych powinny znaleźć się w skrypcie.

Poza grupą *Tasks* w menu kontekstowym bazy danych znajdują się opcje:

Rename służy do zmiany nazwy bazy danych.

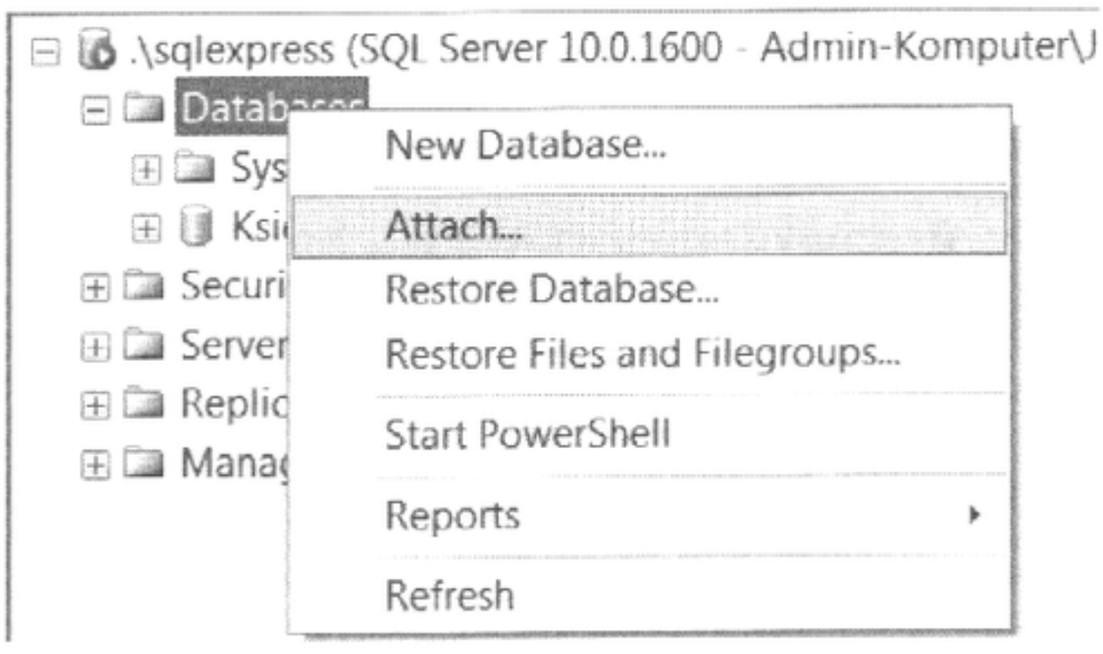
Delete powoduje usunięcie bazy danych.

Refresh odświeża wyświetlane w drzewie obiekty bazy.

Properties powoduje wyświetlenie okna właściwości bazy danych.

Odłączoną bazę danych można dołączyć do wybranej instancji poleceniem *Attach...* (rysunek 6.7).

Rysunek 6.7.
Narzędzie dołączania bazy danych



Zadanie 6.1

Na potrzeby kursów przygotowujących do egzaminu na prawo jazdy utwórz nową bazę danych *Prawo jazdy*. Baza danych powinna zawierać: dane osób zarejestrowanych, listę dostępnych kursów, przydział uczestników do różnych kursów, rejestrację jazd dla każdego uczestnika kursu. W trakcie tworzenia bazy danych określ jej rozmiar, nazwę logiczną oraz położenie pliku z danymi. Wprowadź przykładowe dane.

6.3. Prawa dostępu do serwera

Serwery bazodanowe posiadają mechanizmy, które nie pozwalają korzystać ze swoich zasobów niezidentyfikowanym użytkownikom.

MS SQL Server w chwili tworzenia bazy danych wraz z nią tworzy konta domyślnych użytkowników oraz przypisuje im domyślne role i uprawnienia. Aby lepiej zabezpieczyć bazę danych, administrator może zmienić domyślne ustawienia serwera baz danych.

Tylko uwierzytelnieni użytkownicy mają dostęp do wybranych baz danych. Kolejny poziom zabezpieczeń tworzą uprawnienia nadawane poszczególnym użytkownikom.

6.3.1. Role

Utworzone dla użytkownika konta umożliwiają połączenie się z bazą danych, ale nie pozwalają na odczytanie danych czy ich zmodyfikowanie. Użytkownik domyślnie nie ma żadnych uprawnień. Wynika to z zasad bezpieczeństwa stosowanych przez serwery bazodanowe. Aby użytkownik mógł wykonywać jakiekolwiek operacje w bazie danych, musi mieć nadane odpowiednie uprawnienia.

Podczas nadawania uprawnień powinno się stosować zasadę minimalnych uprawnień. Nie należy nadawać żadnych uprawnień poza minimalnymi, niezbędnymi do prawidłowej pracy z bazą. Uprawnienia można nadawać użytkownikom albo grupom użytkowników. SQL Server umożliwia zarządzanie uprawnieniami poprzez tworzenie ról, czyli grupowanie użytkowników według potrzeb i przypisywanie im uprawnień.

W serwerze SQL Server wyróżniamy trzy rodzaje ról:

- serwerowe,
- bazodanowe,
- definiowane przez użytkownika.

Role pozwalają grupować użytkowników, którzy powinni mieć takie same prawa dostępu do obiektów bazy danych. Role nadawane na poziomie instancji serwera służą do nadawania uprawnień operatorom serwera. Role bazodanowe służą do nadawania użytkownikom uprawnień dostępu do bazy danych. Role definiowane przez użytkownika to uprawnienia do wybranych obiektów bazy lub uprawnienia mieszane, na przykład prawo tylko do odczytu w jednej tabeli, a w innej do odczytu i do zapisu.

W serwerze SQL Server istnieją predefiniowane role serwerowe i role baz danych.

Role serwerowe

`sysadmin` — użytkownicy przypisani do tej roli mają prawo wykonywać każdą operację na serwerze i są właścicielami każdej bazy danych. Nie można im zabronić wykonania żadnej operacji. Rola ta jest przydzielona użytkownikowi **sa** i nie można jej mu odebrać.

`serveradmin` — użytkownicy przypisani do tej roli to administratorzy serwera. Nie administrują oni bazami danych, ale mają prawo do konfigurowania systemu i do zarządzania serwerem.

`setupadmin` — rola jest przypisywana administratorom, którzy konfigurują system i nim zarządzają.

`securityadmin` — rola jest przypisywana administratorom, którzy zarządzają bezpieczeństwem systemu (tworzą konta, przyznają prawo tworzenia baz danych, wykonują procedury związane z bezpieczeństwem).

`processadmin` — rola jest przypisywana administratorom, którzy kontrolują procesy uruchomione na serwerze, na przykład kasowanie działających zapytań.

`dbcreator` — użytkownicy przypisani do tej roli mogą tworzyć, modyfikować, usuwać i odzyskiwać bazy danych.

`diskadmin` — użytkownicy przypisani do tej roli zarządzają plikami.

`bulkadmin` — użytkownicy przypisani do tej roli mogą uruchamiać polecenie `BULK INSERT` (operację masowego wstawiania danych).

Role bazy danych

Role bazy danych to uprawnienia przyznawane użytkownikowi do określonej bazy danych.

`db_owner` — rola przypisywana właścicielowi bazy danych. Użytkownicy przypisani do tej roli mogą wykonywać operacje DDL (wyjątkiem są polecenia `GRANT`, `DENY` i `REVOKE`).

`db_accessadmin` — użytkownicy przypisani do tej roli decydują, które konta logowania mają prawa dostępu do bazy danych.

Właściciel bazy danych należy do roli `db_owner` i ma do niej wszystkie prawa. Każda baza danych ma tylko jednego właściciela (`dbo`).

Właścicielem obiektu bazy danych (`dboo`) jest użytkownik, który stworzył ten obiekt. Użytkownicy należący do ról `db_owner` lub `db_ddladmin` mogą tworzyć obiekty pod swoją nazwą lub jako `dbo`.

Tworząc konto logowania (login) i użytkownika (user), który będzie wykorzystywany podczas pracy z bazą danych, należy przypisać użytkownikowi odpowiednie role serwerowe i bazodanowe.

Tworzenie i usuwanie ról

Rola w bazie danych jest tworzona za pomocą polecenia `CREATE ROLE`.

```
CREATE ROLE Blok1;

CREATE ROLE Pracownik;
```

Istniejącą rolę można usunąć, wpisując polecenie `DROP ROLE`.

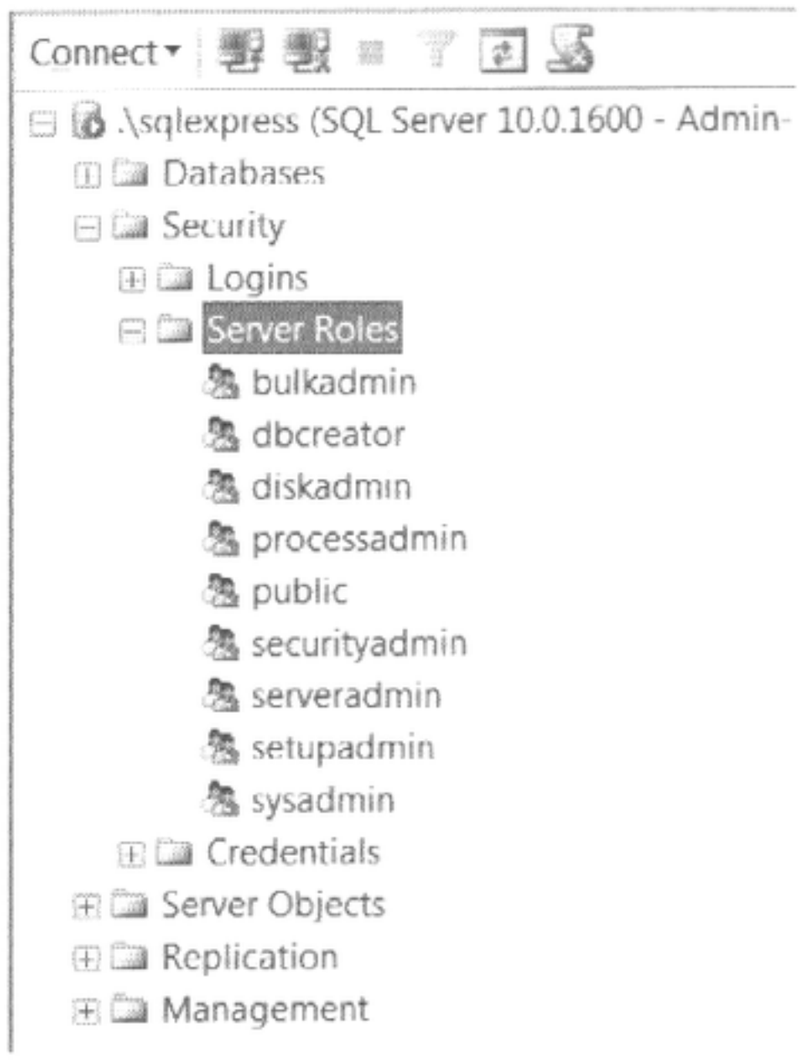
```
DROP ROLE Pracownik;
```

Przykład 6.5

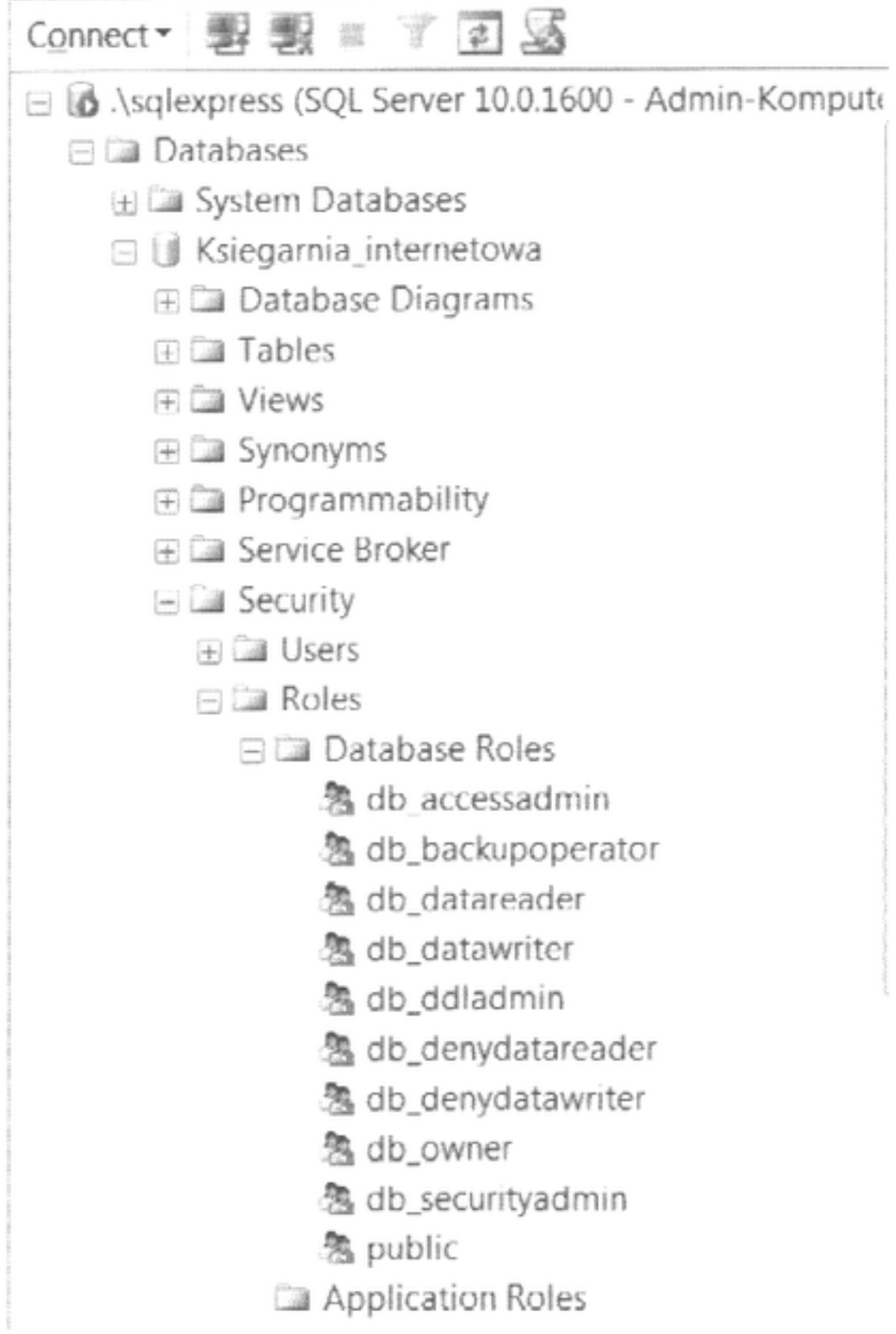
```
CREATE ROLE Blok1;

CREATE ROLE Operator;
```

Można usuwać tylko role użytkowników. Natomiast nie mogą być usuwane wbudowane role serwera oraz bazy danych.

Z poziomu SQL Server Management Studio dostęp do ról serwerowych i bazodanowych został pokazany na rysunkach 6.8 i 6.9.



Rysunek 6.8. Role serwerowe dostępne z poziomu SQL Server Management Studio



Rysunek 6.9. Role bazodanowe dostępne z poziomu SQL Server Management Studio

Przypisywanie do ról

Do utworzonych ról można przypisać dowolną liczbę użytkowników. Użytkownicy są przypisywani do ról poleceniem `GRANT ... TO ...`.

Przykład 6.6

```
GRANT Blok1 TO Maciej;

GRANT Blok1 TO Michał;

GRANT Blok1 TO Marcin;
```

Ten sam użytkownik może zostać przypisany do różnych ról.

```
GRANT Operator TO Michał;
```

Każdy z użytkowników jest automatycznie przypisany do roli `PUBLIC`. Jest to specjalna grupa, z której nie można żadnego użytkownika usunąć. Pozwala ona nadać lub odebrać uprawnienia wszystkim użytkownikom bazy danych.

6.3.2. Uprawnienia

Uprawnienia nadawane użytkownikom dzielimy na dwie kategorie.

1. Uprawnienia do wykonania określonych instrukcji. Są to:
   - » uprawnienia do modyfikowania ról (`ALTER ANY ROLE`);
   - » uprawnienia do modyfikowania kont użytkowników (`ALTER ANY USER`);
   - » uprawnienia do wykonywania kopii zapasowych bazy danych (`BACKUP DATABASE`);
   - » uprawnienia do tworzenia funkcji (`CREATE FUNCTION`);
   - » uprawnienia do tworzenia procedur (`CREATE PROCEDURE`);
   - » uprawnienia do tworzenia schematów (`CREATE SCHEMA`);
   - » uprawnienia do tworzenia tabel (`CREATE TABLE`);
   - » uprawnienia do przejmowania obiektów na własność (`TAKE OWNERSHIP`);
   - » uprawnienia do odczytywania metadanych obiektów (`VIEW DEFINITION`).
2. Uprawnienia obiektowe — użytkownicy mogą odwoływać się do wybranych obiektów bazy danych i wykonywać określone operacje. Możliwe do nadania uprawnienia zależą od typu obiektu:
   - » Do tabel i widoków można nadawać uprawnienia `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `REFERENCE`. Uprawnienia te pozwalają na pobieranie, wstawianie, modyfikowanie i usuwanie danych oraz na tworzenie kluczy obcych.
   - » Do kolumn można nadawać uprawnienia `SELECT` i `UPDATE`.
   - » Do procedur składowanych oraz funkcji można nadać uprawnienie `EXECUTE`.
   - » Do schematów można nadać wszystkie wymienione wyżej uprawnienia.

Nadawanie uprawnień — instrukcja GRANT

Uprawnienia do obiektów bazy danych nadawane są rolom lub użytkownikom instrukcją `GRANT`.

```
GRANT {uprawnienia}
ON Obiekt
TO Użytkownicy/Role
```

Przykład 6.7

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON dbo.Ksiazki
TO Blok1;
```

Zamiast nadawać uprawnienia poszczególnym użytkownikom, należy nadawać je rolom. W podanym przykładzie wszyscy członkowie roli *Blok1* mogą odczytywać i modyfikować dane zapisane w tabeli *Ksiazki*.

Odbieranie uprawnień — instrukcja REVOKE

Instrukcją `REVOKE` można usunąć wcześniej nadane lub odebrane uprawnienia do obiektu.

Przykład 6.8

```
REVOKE INSERT, UPDATE
ON dbo.Ksiazki
FROM Michał;
```

W podanym przykładzie użytkownikowi *Michał* zostały odebrane uprawnienia do wstawiania i modyfikowania danych w tabeli *Ksiazki*. Ale *Michał* należy do roli *Blok1*, która posiada odebrane mu uprawnienia. Z tego powodu będzie on nadal mógł wstawiać i modyfikować dane w tabeli *Ksiazki*.

Instrukcja DENY

Jawne odebranie użytkownikowi uprawnień dostępu do obiektów bazy danych nastąpi po użyciu instrukcji `DENY`.

```
DENY {Uprawnienia}
ON Obiekt
TO Użytkownicy/Role
```

Przykład 6.9

```
DENY INSERT, DELETE
ON Ksiazki
TO Michał;
```

Po wykonaniu tej operacji użytkownik *Michał* mimo przynależności do ról nie będzie mógł dodać wiersza do tabeli *Ksiazki* ani go z niej usunąć.

Dziedziczenie uprawnień

W serwerach bazodanowych funkcjonuje mechanizm dziedziczenia uprawnień. Obiekty bazy danych tworzą hierarchię, zgodnie z którą działają mechanizmy dziedziczenia.

Najwyżej w hierarchii znajdują się obiekty serwera bazodanowego (bazy danych, loginy), poniżej są obiekty bazy danych (konta użytkowników, role, schematy), jeszcze niżej — obiekty schematów (tabele, widoki, procedury, funkcje), a na samym dole hierarchii są obiekty tabel, czyli kolumny.

Uprawnienia nadane do obiektu znajdującego się wyżej w hierarchii są domyślnie dziedziczone przez obiekty położone niżej. Jednak uprawnienia mogą być nadawane i odbierane na dowolnym poziomie.

Przykład 6.10

```
REVOKE UPDATE
ON dbo.Ksiazki (tytul)
FROM Maciej;
```

W podanym przykładzie użytkownik *Maciej* może aktualizować wszystkie kolumny tabeli *Ksiazki* oprócz kolumny *tytul*.

Właściciel obiektu

Każdy obiekt zdefiniowany w bazie danych ma swojego właściciela. Domyślnie właścicielem obiektu jest użytkownik, który go utworzył. Ma on nieograniczone uprawnienia do obiektu. Możliwe jest jawne określenie innego właściciela obiektu, jeśli podczas definiowania obiektu jego nazwę poprzedzimy nazwą nowego właściciela. Możliwa jest również zmiana właściciela istniejącego obiektu. Właściciel obiektu może udzielać uprawnień do tego obiektu.

Zaleca się, aby właścicielem wszystkich obiektów bazy danych był ten sam użytkownik.

Przykład 6.11

```
USE Ksiegarnia_internetowa GRANT CREATE VIEW TO Blok1;
GO
CREATE VIEW Marcin.widok1 AS
SELECT Autor.nazwisko, Autor.imie, Ksiazki.tytul, Ksiazki.cena
FROM Autor INNER JOIN Ksiazki
ON Autor.id_autora=Ksiazki.id_autora
WHERE Ksiazki.rok_wydania>2010;
GO
```

W podanym przykładzie pierwsze polecenie nadaje prawo tworzenia widoków w bazie *Ksiegarnia_internetowa* użytkownikom należącym do roli *Blok1*. Następne polecenie tworzy widok, którego właścicielem zostaje użytkownik *Marcin*.

Podsumowanie

Jedną z podstawowych zasad bezpieczeństwa danych jest nadawanie minimalnych uprawnień. Oznacza to, że należy nadawać użytkownikowi tylko te uprawnienia, które są niezbędne do wykonywania jego obowiązków.

Jeżeli chcemy ograniczyć użytkownikom dostęp do poszczególnych kolumn w tabelach, można nadawać uprawnienia do tych kolumn, ale lepszym rozwiązaniem jest tworzenie widoków i nadawanie odpowiednich uprawnień do widoków.

6.4. Replikacja bazy danych

Replikacja bazy danych polega na kopiowaniu (powielaniu) i przesyłaniu danych lub obiektów bazodanowych między serwerami oraz na synchronizowaniu tych danych w celu utrzymania ich spójności. Dane kopiowane nazywamy danymi źródłowymi, dane docelowe — repliką.

Replikacja danych najczęściej wykorzystywana jest w systemach rozproszonych baz danych, gdzie dane z jednego zdalnego węzła (serwera) są kopiowane do innych zdalnych węzłów. Celem replikacji jest skrócenie czasu dostępu do danych oraz uniezależnienie się od czasowej niedostępności serwerów i awarii sieci. Wadą replikacji jest konieczność aktualizowania repliki w przypadku zmian danych źródłowych.

Proces uaktualniania nazywamy synchronizacją (ang. *synchronization*) lub odświeżaniem (ang. *refreshing*) repliki.

Dzięki replikacji możemy poprawić niezawodność i wydajność systemu bazodanowego.

Replikując dane:

- umożliwiamy lokalny dostęp do danych użytkownikom z oddalonych miejsc;
- pozwalamy na częściową niezależność serwerów bazodanowych;
- możemy podzielić dane w sposób odpowiadający wymaganej strukturze;
- możemy fizycznie rozdzielić serwery bazodanowe realizujące różne zadania oparte na tych samych danych.

6.4.1. Model replikacji

Model replikacji serwera SQL Server definiuje trzy role, które mogą zostać przypisane serwerom bazodanowym: serwer może zostać skonfigurowany jako **dystrybutor** (ang. *distributor*), **wydawca** (ang. *publisher*) oraz **subskrybent** (ang. *subscriber*).

Wydawca to serwer baz danych, który przesyła dane do innego serwera lub do innej bazy danych. Jego zadaniami są: utrzymanie wzorcowej bazy danych, udostępnianie tej bazy innym serwerom (wysyłanie danych do subskrybenta) oraz monitorowanie zmian w replikowanych danych i przesyłanie informacji o tych zmianach do serwera pełniącego funkcję dystrybutora.

Dystrybutor to serwer baz danych, który zarządza przepływem danych między wydawcą a subskrybentami. Serwer ma bazę dystrybucyjną (ang. *distribution*), która jest automatycznie tworzona podczas przypisywania serwerowi roli dystrybutora. Ponadto

dystrybutor zarządza informacjami związanymi z replikacją danych, takimi jak: historia zmian danych, transakcje przeprowadzone na serwerach, konfiguracja serwerów biorących udział w replikacji.

Subskrybent to serwer (lub baza danych), który otrzymuje replikowane dane od innego serwera (lub od innej bazy danych) i przechowuje ich lokalną kopię. Dane przechowywane przez subskrybenta mogą zostać udostępnione użytkownikom tylko do odczytu lub do odczytu i modyfikacji.

Dane (publikacje) mogą być dostarczane automatycznie (ang. *push*) przez wydawcę lub mogą być pobierane okresowo (ang. *pull*) przez subskrybenta, który w celu ich pobrania łączy się z bazą dystrybucyjną.

Replikacja typu *push* zalecana jest w przypadku synchronizowania poufnych danych pomiędzy małą liczbą serwerów i wymaga wydajnego serwera pełniącego funkcję dystrybutora. Replikacja typu *pull* zalecana jest w przypadku synchronizowania danych pomiędzy dużą liczbą serwerów.

6.4.2. Typy replikacji

W serwerze SQL Server występują trzy typy replikacji: migawkowa, transakcyjna i łączeniowa.

Replikacja migawkowa (ang. *Snapshot replication*) polega na systematycznym przesyłaniu danych z określonych momentów od wydawcy do subskrybentów. Ten typ replikacji stosowany jest dla danych, które nie są często modyfikowane.

Replikacja transakcyjna (ang. *Transactional replication*) — wszystkie zmiany w danych źródłowych są na bieżąco przesyłane do replik w kolejności, w jakiej zostały wprowadzone. Ponieważ zmiany zapisywane są na bieżąco, nie występują konflikty. Ten typ replikacji wymaga szybkiego i niezawodnego połączenia wszystkich serwerów i dlatego w praktyce jest stosowany głównie w sieciach lokalnych.

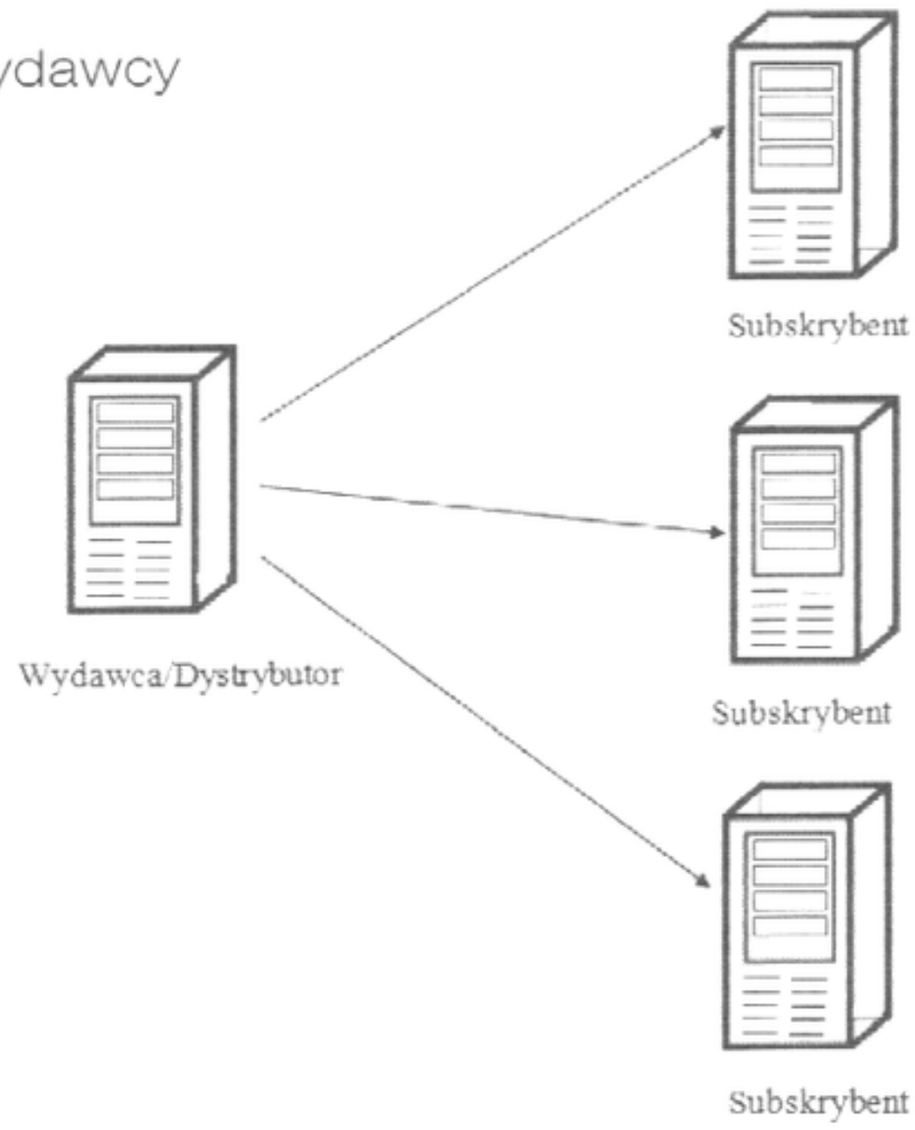
Replikacja łączeniowa (ang. *Merge replication*) — zmodyfikowane dane są przechowywane i w określonym czasie przesyłane do dystrybutora, który rozwiązuje konflikty i wysyła dane do subskrybentów.

Konfiguracje replikacyjne

Każdy typ replikacji może zostać zaimplementowany w jednym z fizycznych modeli replikacji. Najczęściej spotykane modele to:

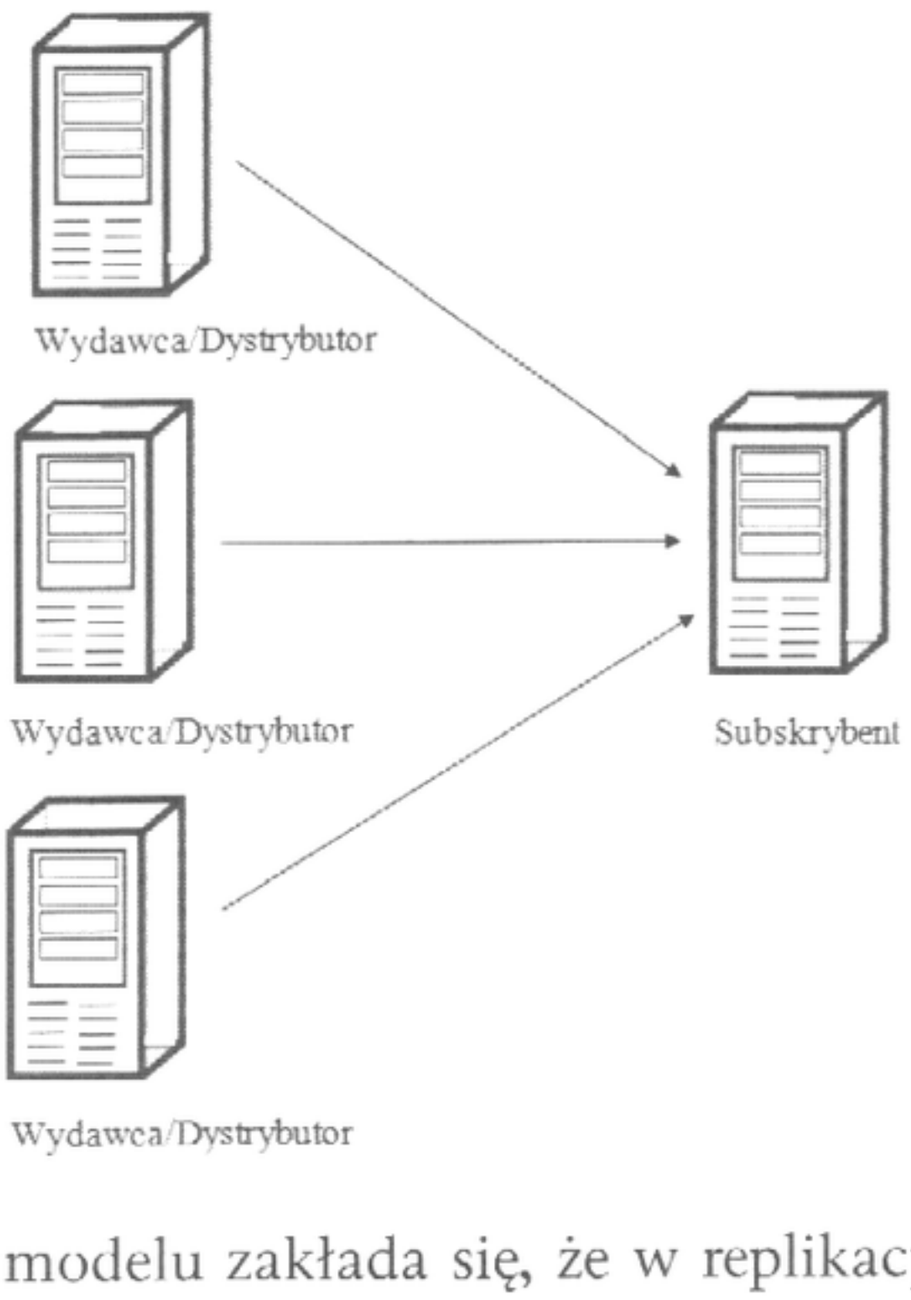
- **model centralnego wydawcy** — w tym modelu jeden serwer pełni funkcje wydawcy oraz dystrybutora, który replikuje dane do dowolnej liczby subskrybentów. Często role wydawcy i dystrybutora są przypisywane temu samemu serwerowi. Jest to najczęściej spotykana konfiguracja (rysunek 6.10).

Rysunek 6.10.
Model centralnego wydawcy



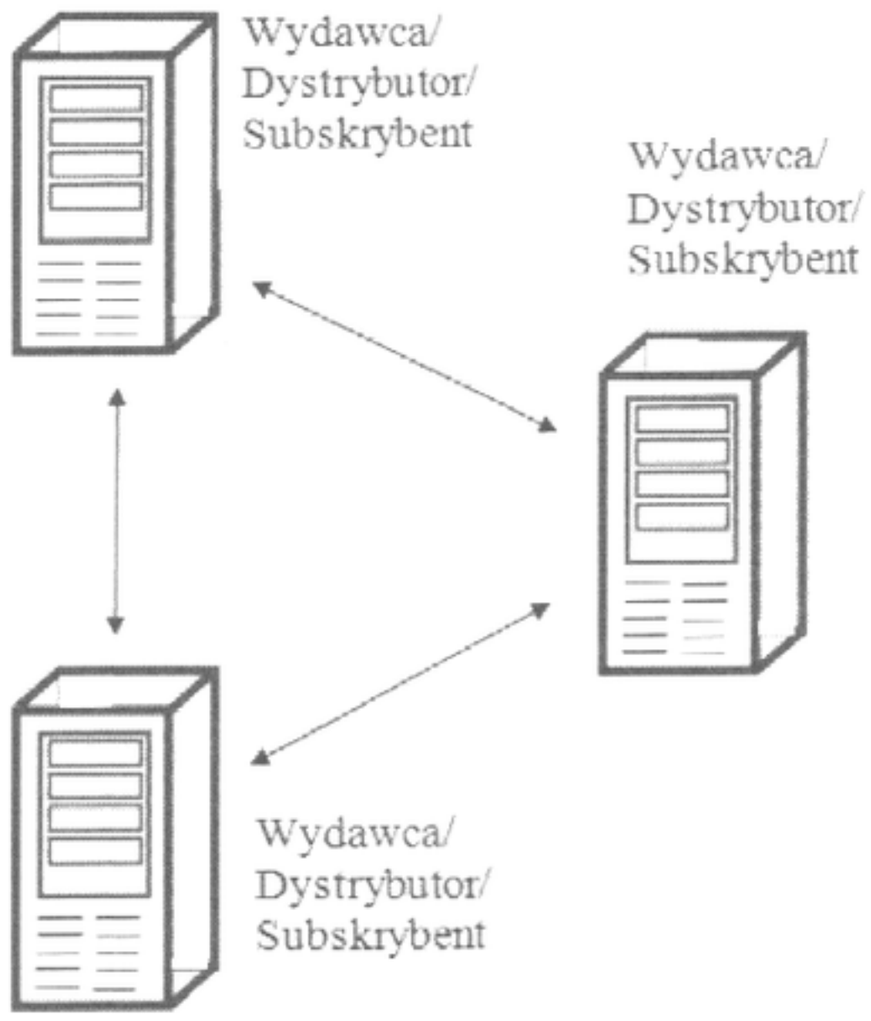
- **model centralnego subskrybenta** — w tym modelu zakłada się, że dowolna liczba wydawców przesyła dane do jednego subskrybenta, gdzie są analizowane (rysunek 6.11).

Rysunek 6.11.
Model centralnego subskrybenta



- **model równorzędny** — w tym modelu zakłada się, że w replikacji bierze udział dowolna liczba wydawców oraz dowolna liczba subskrybentów (rysunek 6.12).

Rysunek 6.12.
Model równorzędny



W serwerze SQL Server do wykonania replikacji wymagane jest uruchomienie usługi SQL Server oraz usługi SQL Server Agent.

Za replikację danych odpowiadają usługi replikacji zwane agentami. Są to:

- agent migawki (ang. *Snapshot Agent*),
- agent transakcji (ang. *Log Reader Agent*),
- agent scalania (ang. *Merge Agent*),
- agent dystrybucji (ang. *Distribution Agent*),
- agent kolejkowania (ang. *Queue Reader Agent*).

6.4.3. Replikacja migawkowa

Replikacja migawkowa jest najprostszą z metod replikacji. Polega ona na wykonywaniu kopii wszystkich danych i obiektów wydawcy i przesyłaniu ich do subskrybentów. W tego typu replikacji udział biorą agent migawki oraz agent dystrybucji. Usługi te powinny zostać uruchomione na serwerze pełniącym funkcję dystrybutora.

Agent migawki (ang. *Snapshot Agent*) pracuje na dystrybutorze i tworzy plik migawki, który zawiera schemat i dane z publikowanych tabel i obiektów bazy danych. Agent dystrybucji (ang. *Distribution Agent*) dostarcza plik do subskrybenta.

W języku SQL migawkę tworzy się poleceniem: `CREATE SNAPSHOT` lub `CREATE MATERIALIZED VIEW`. Składnia polecenia ma postać:

```
CREATE SNAPSHOT nazwa_migawki
BUILD {IMMEDIATE | DEFERRED}
REFRESH sposób_odświeżania
START WITH data_rozpoczęcia_odświeżania
```

```
NEXT częstotliwość_odświeżania
WITH {ROWID | PRIMARY KEY}
AS zapytanie;
```

Przykład 6.12

```
CREATE SNAPSHOT sprzedaz
BUILD IMMEDIATE
REFRESH COMPLETE
START WITH sysdate+(1/(24*60))
NEXT sysdate+(1/(24*60*6))
AS
SELECT * FROM sprzedaz@tor
WHERE data_zamowienia like '%2013';
```

W podanym przykładzie została zdefiniowana migawka o nazwie *sprzedaz*. Migawka została wypełniona danymi w momencie tworzenia (`BUILD IMMEDIATE`). W definicji migawki przy użyciu klauzuli `BUILD IMMEDIATE` lub `BUILD DEFERRED` można określić moment jej wypełnienia danymi. Migawka może zostać wypełniona danymi natychmiast po jej utworzeniu (klauzula `BUILD IMMEDIATE`) lub w momencie jej pierwszego odświeżenia (klauzula `BUILD DEFERRED`). Klauzula `BUILD IMMEDIATE` jest klauzulą domyślną. Zastosowane zostało odświeżanie pełne (`REFRESH COMPLETE`), moment rozpoczęcia odświeżania automatycznego ustawiono na 1 minutę po utworzeniu migawki (`START WITH sysdate+(1/24*60)`), a częstotliwość odświeżania na 10 sekund (`NEXT sysdate+(1/24*60*6)`). Migawka udostępnia informacje o sprzedaży produktów w roku 2013 z tabeli wskazywanej łącznikiem `tor`.

Przykład 6.13

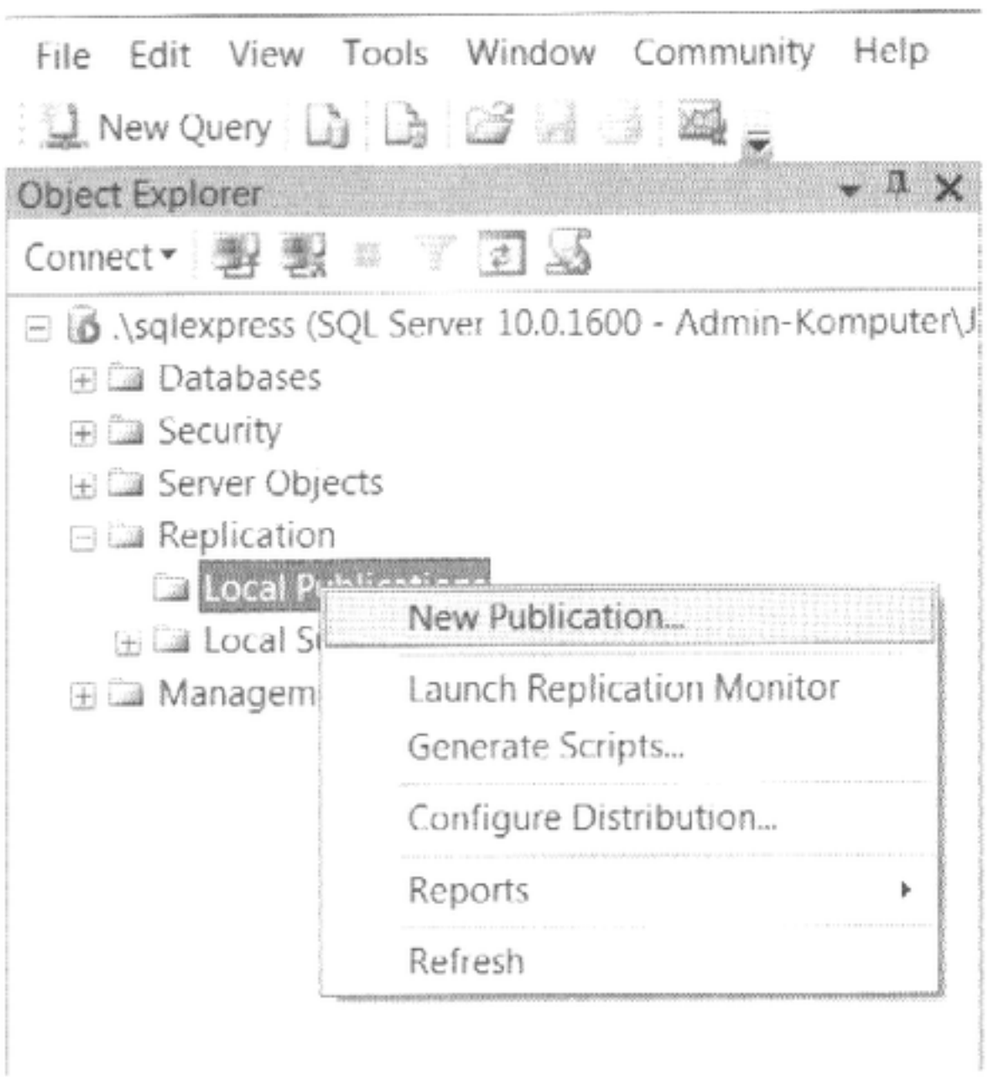
```
CREATE SNAPSHOT sprzedaz1
BUILD DEFERRED
REFRESH FORCE
START WITH sysdate + (1/(24*6))
NEXT sysdate+(1/(24*60))
WITH PRIMARY KEY
AS
SELECT * FROM sprzedaz@tor
WHERE id_klienta=1;
```

W podanym przykładzie migawka zostanie wypełniona danymi podczas pierwszego odświeżenia, tj. 10 minut po jej utworzeniu.

Tworzenie replikacji za pomocą SQL Server Management Studio

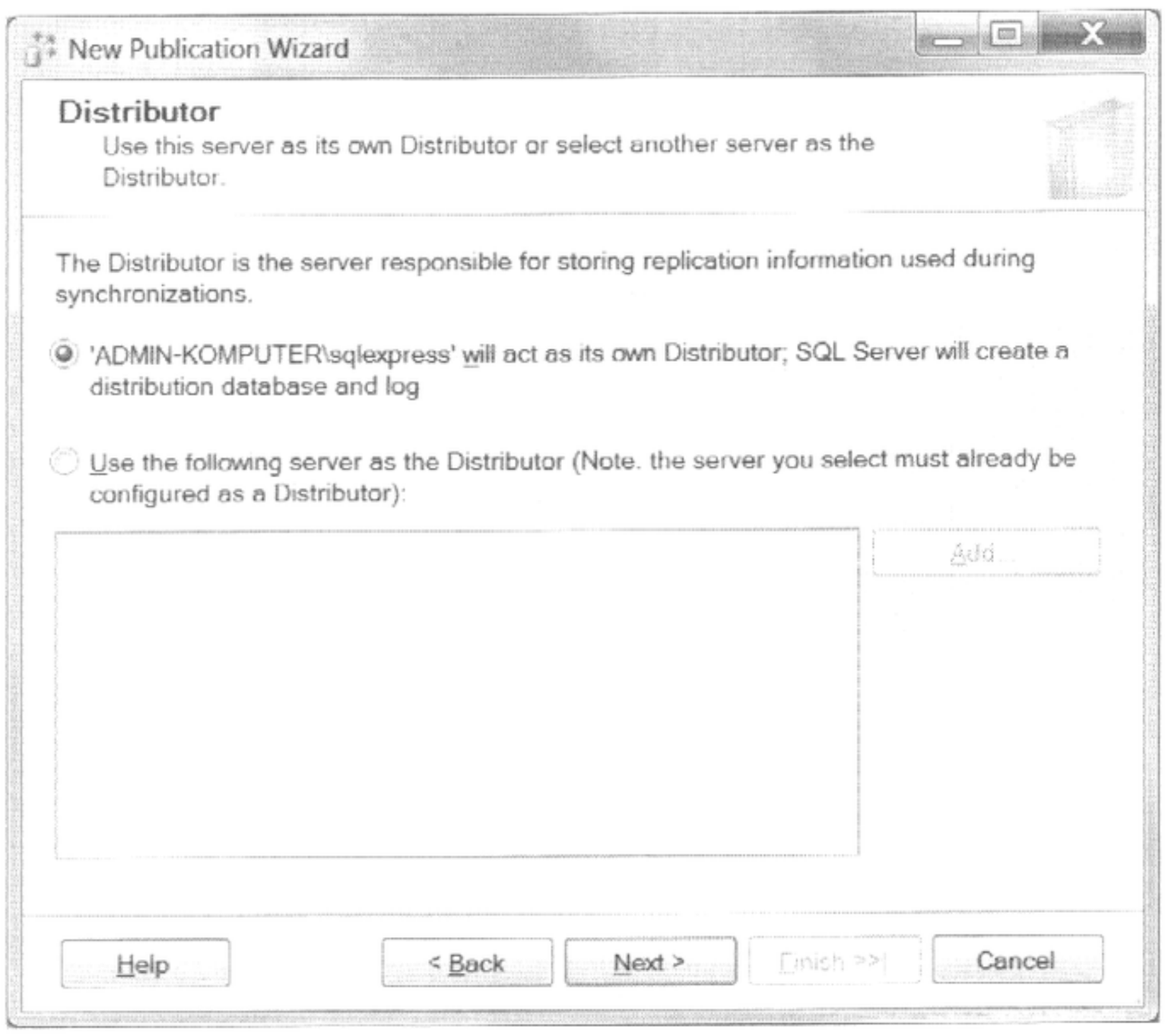
Za pomocą aplikacji SQL Server Management Studio można utworzyć replikację bazy danych znajdującej się na tym serwerze.

Po uruchomieniu SQL Server Management Studio i połączeniu się z wybraną instancją należy wybrać folder *Replication/Local Publication*, a następnic kliknąć prawym przyciskiem myszy i wybrać opcję *New Publication...* (rysunek 6.13).



Rysunek 6.13. Uruchomienie replikacji danych

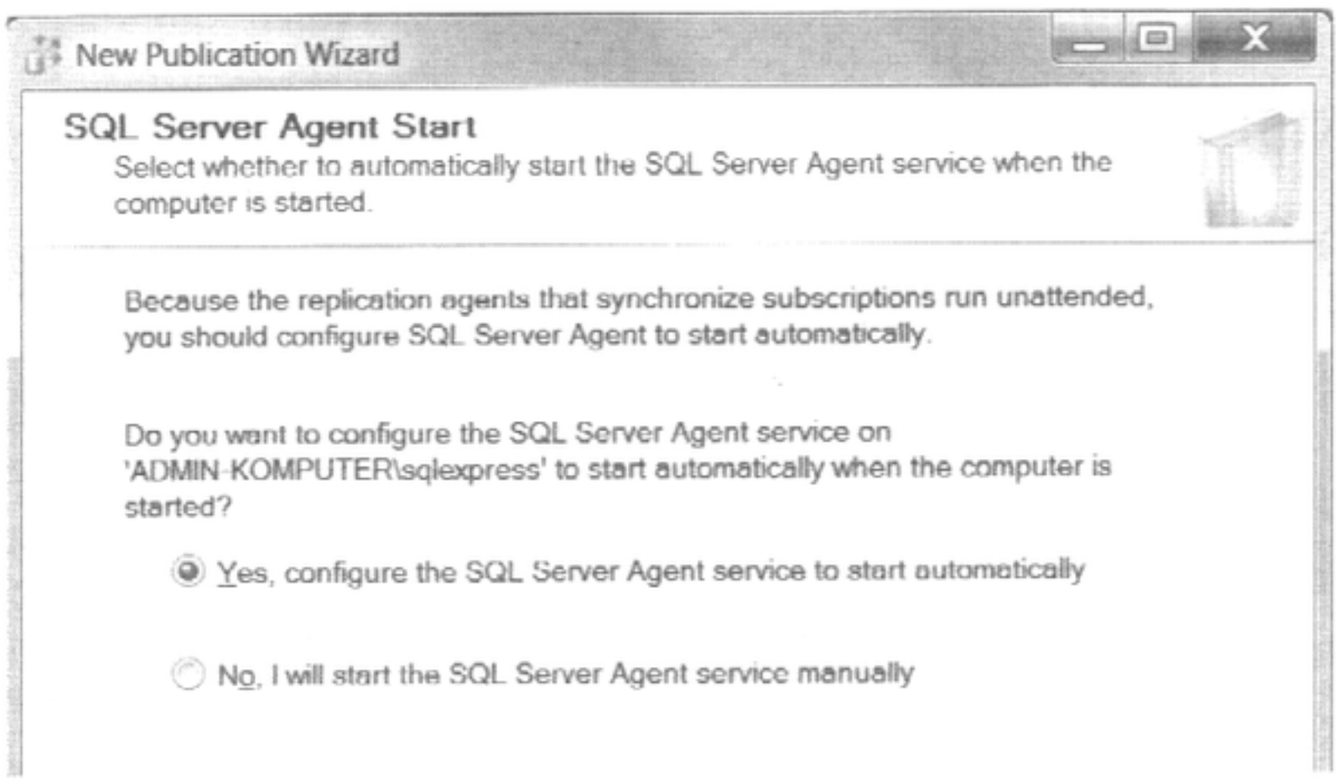
Zostanie uruchomiony kreator publikacji (ang. *Publication Wizard*), który poprowadzi nas przez proces tworzenia publikacji. W pierwszym oknie można określić, czy serwer, na którym aktualnie pracujemy, ma zostać jednocześnie wydawcą i dystrybutorem, czy też jako dystrybutor zostanie wybrany inny serwer (rysunek 6.14).



Rysunek 6.14. Okno wyboru dystrybutora

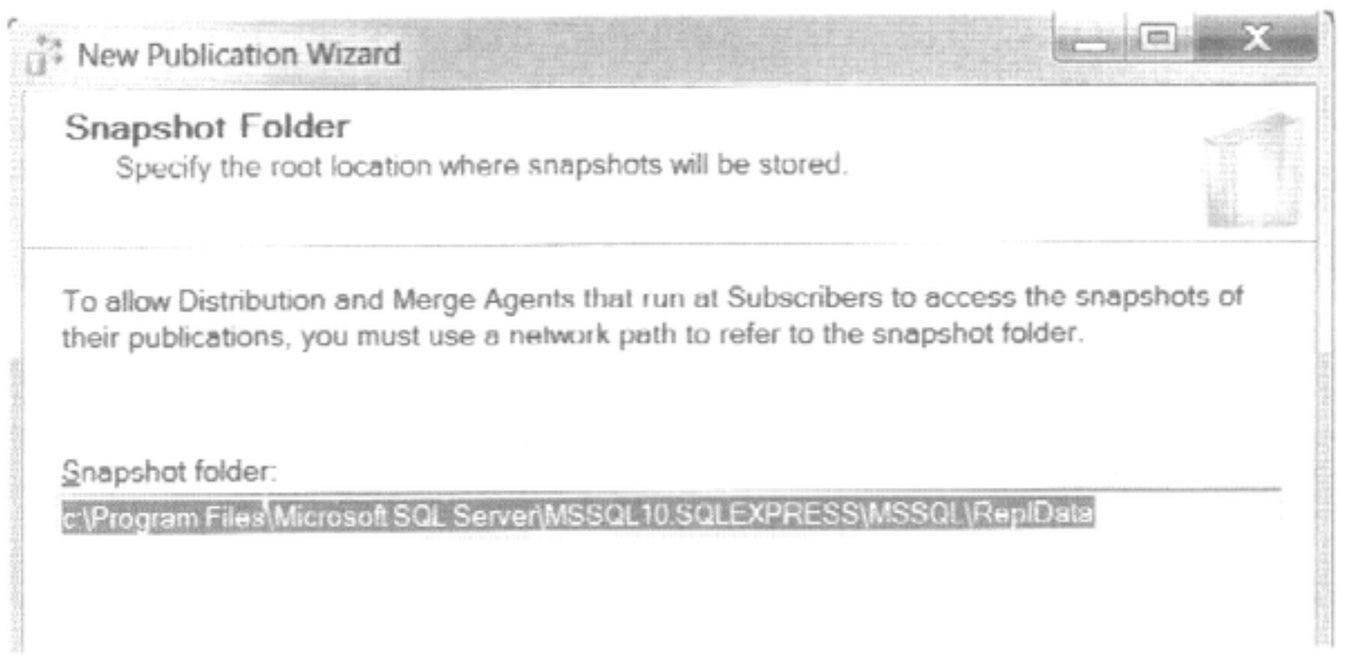
W kolejnym oknie należy skonfigurować *agenta dystrybucji* (jeżeli wcześniej nie został uruchomiony — rysunek 6.15).

Rysunek 6.15. Okno konfigurowania agenta dystrybucji



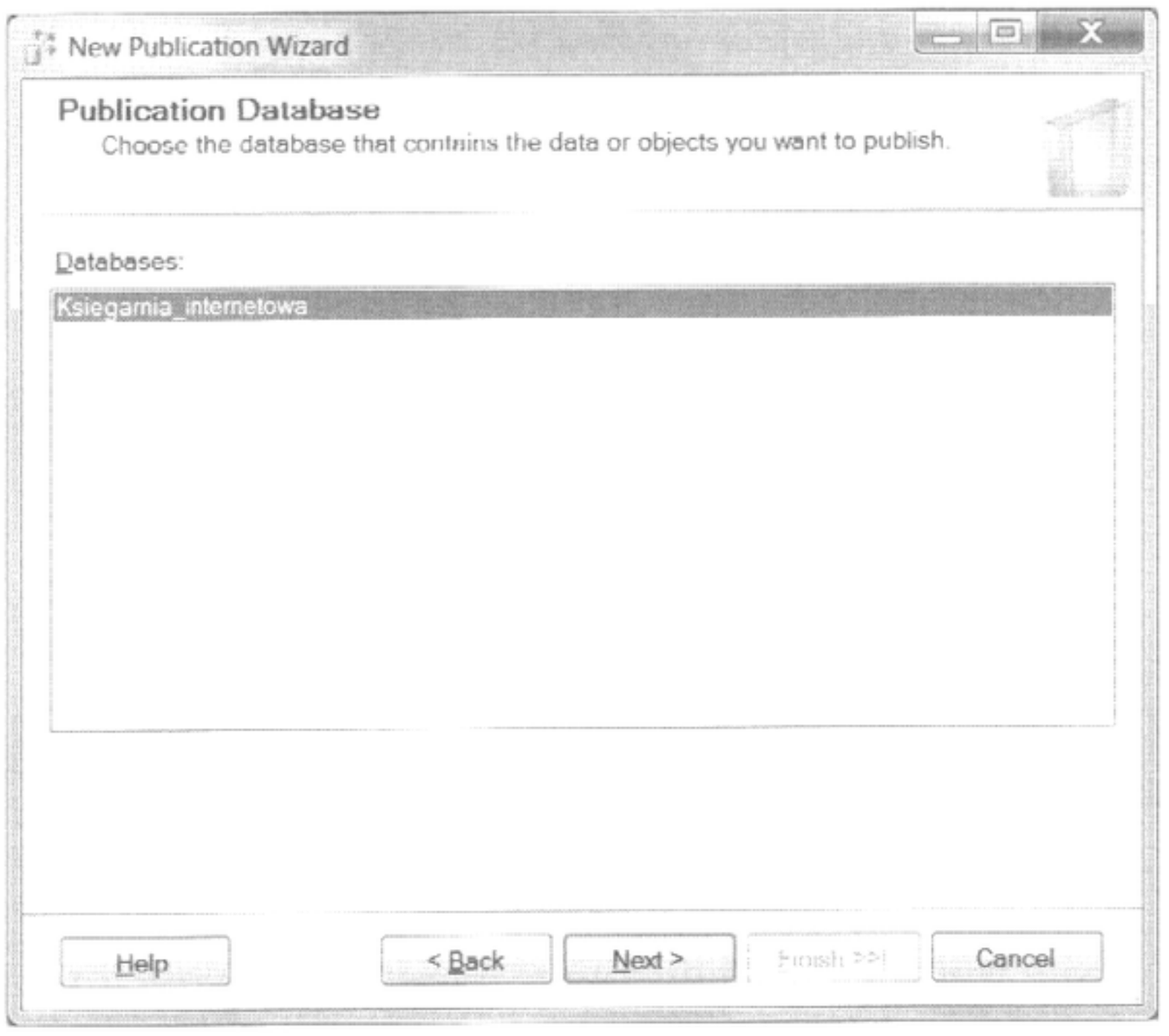
Aby agent dystrybucji miał dostęp do migawek publikacji, należy określić ścieżkę dostępu do folderu migawki (rysunek 6.16). Jeżeli dane są replikowane między dwoma różnymi serwerami SQL, powinna zostać zdefiniowana ścieżka do współdzielonego zasobu sieciowego.

Rysunek 6.16. Okno konfigurowania agenta dystrybucji



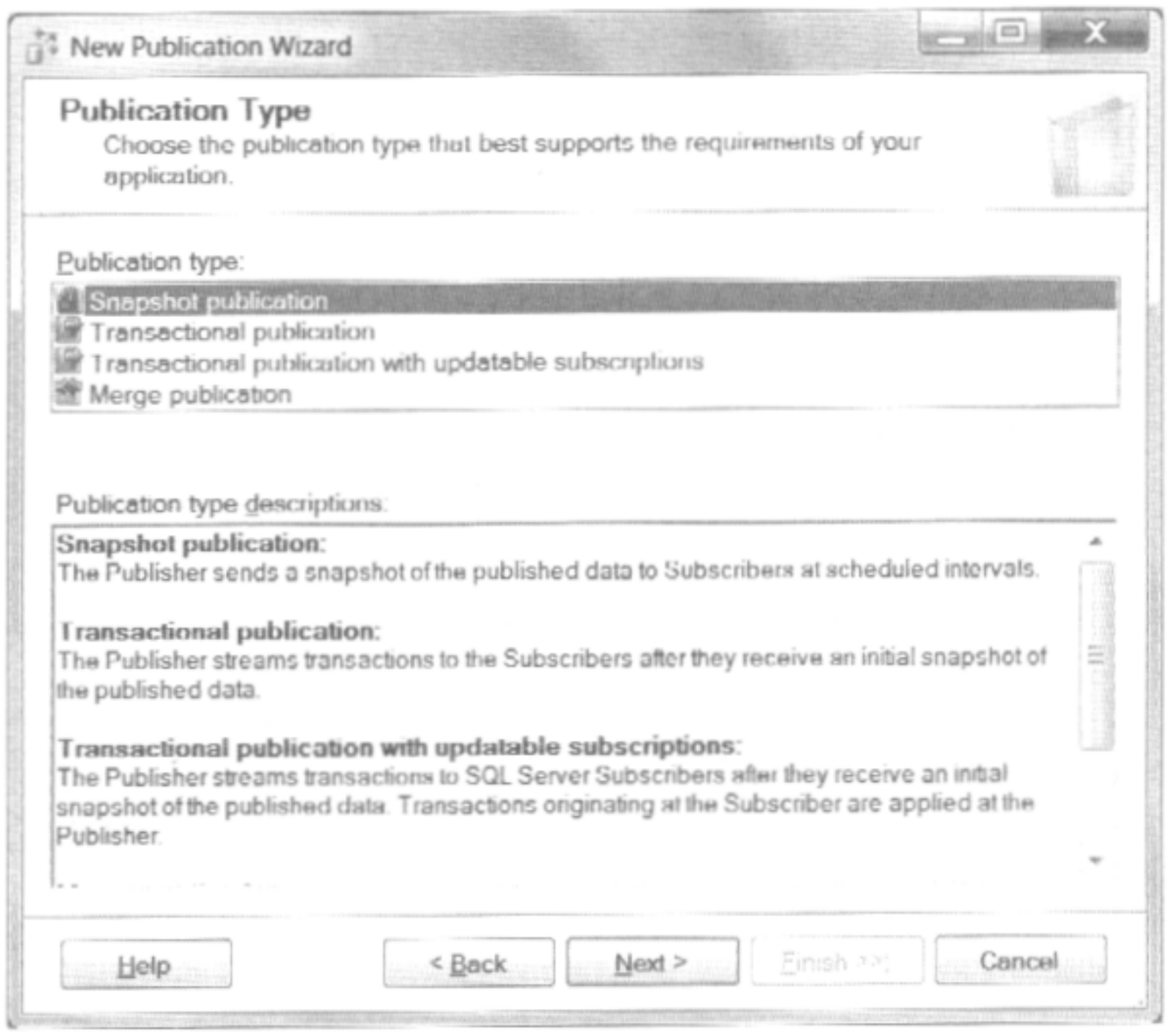
Kolejny krok to wybór bazy, która ma być replikowana (rysunek 6.17).

Rysunek 6.17. Wybór bazy do replikacji



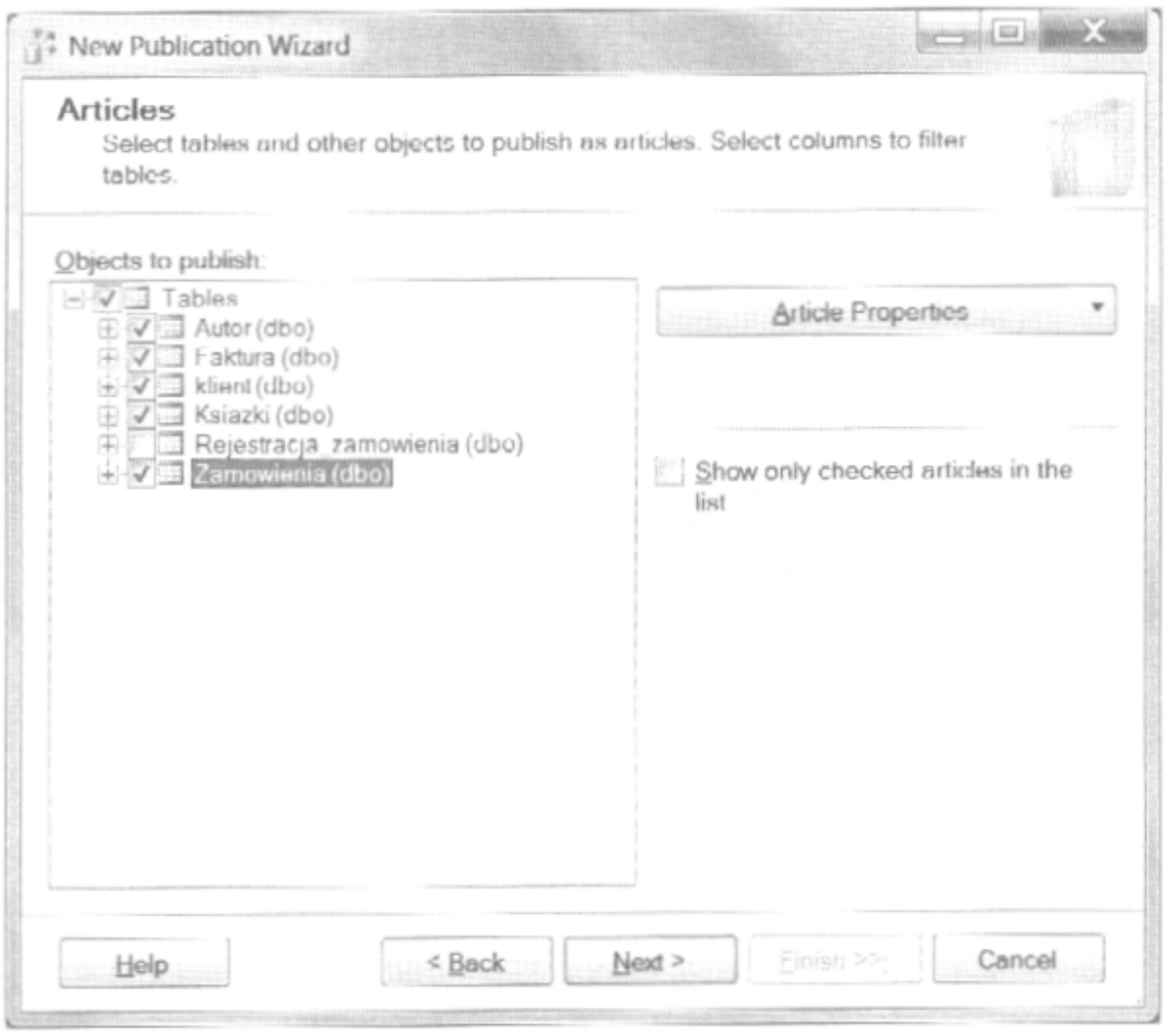
W kolejnym oknie należy określić typ replikacji. Ponieważ będzie tworzona replikacja migawkowa, wybieramy opcję *Snapshot publication* (rysunek 6.18).

Rysunek 6.18. Wybór typu replikacji



W następnym oknie należy wybrać tabele do publikacji (rysunek 6.19).

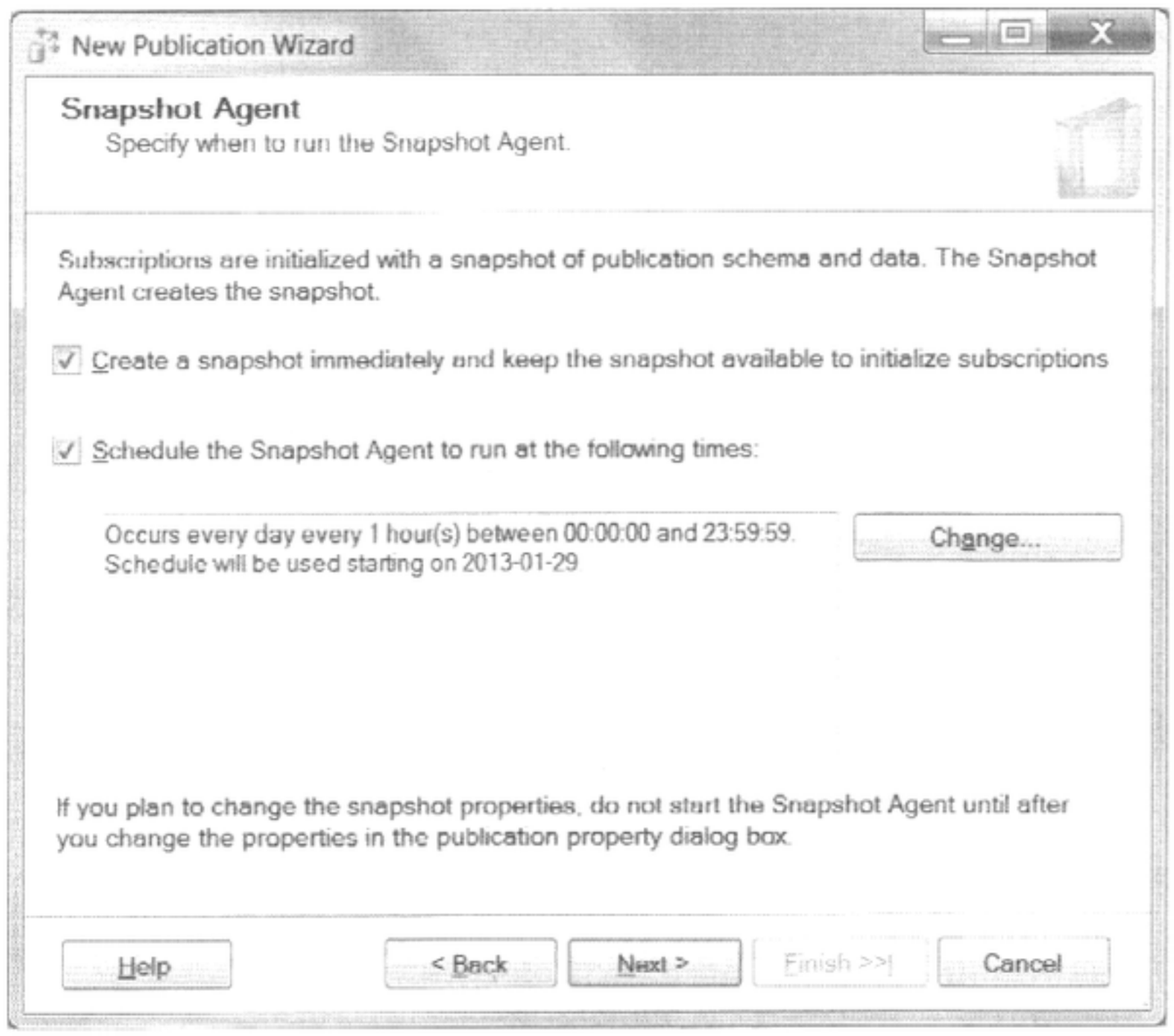
Rysunek 6.19. Wybór tabel do publikacji



Jeżeli chcemy publikować tylko wybrane fragmenty tabel, wybieramy je w kolejnym kroku, klikając *Add* i zaznaczając odpowiednie kolumny.

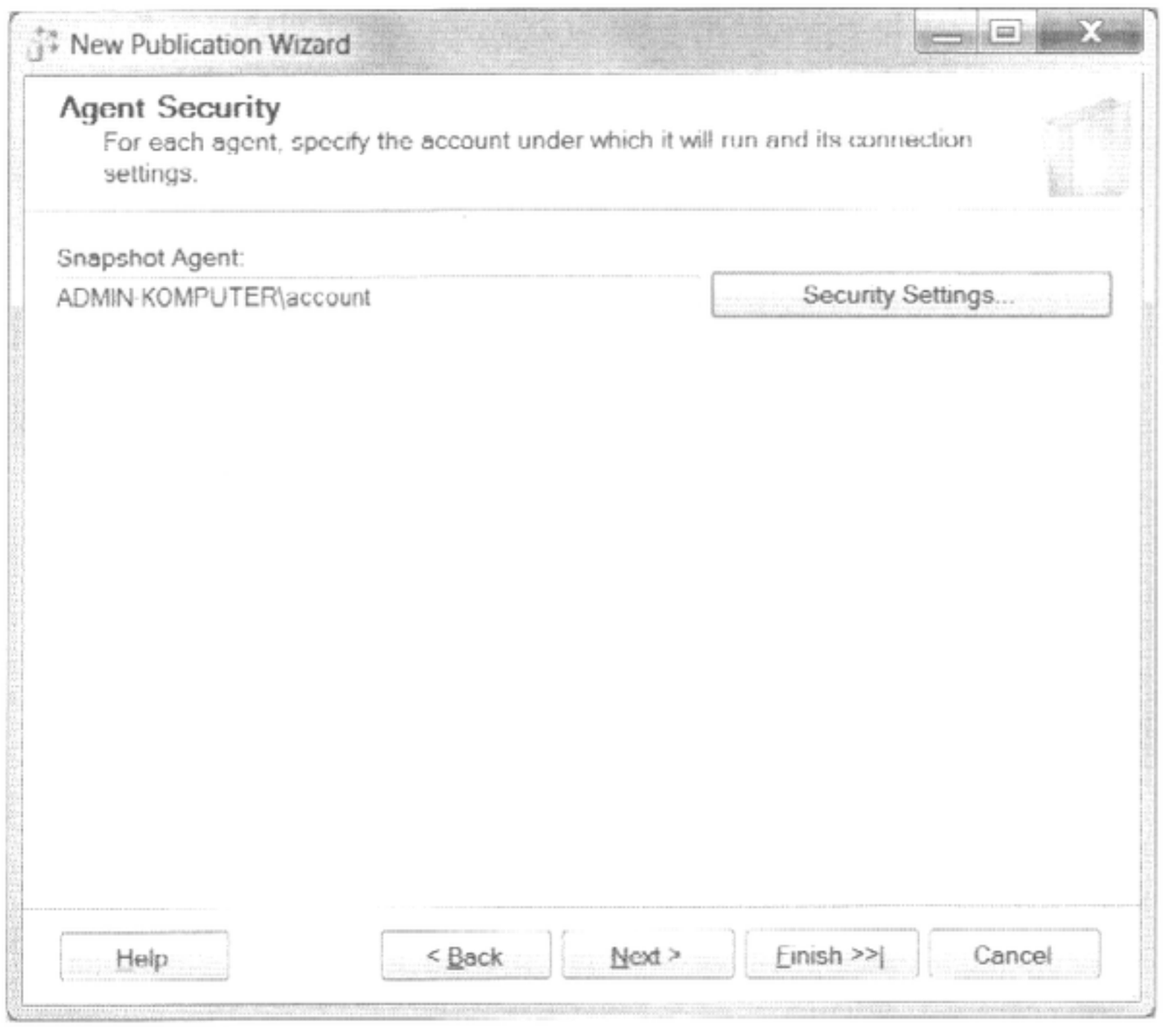
Po przejściu do kolejnego okna należy zaznaczyć opcje tworzenia migawki oraz skonfigurować harmonogram pracy agenta migawki (rysunek 6.20).

Rysunek 6.20. Konfigurowanie pracy agenta migawki



W kolejnym oknie (*Agent Security*) po kliknięciu przycisku *Security Settings...* należy zaznaczyć konto, które będzie działało jako agent migawki (rysunek 6.21).

Rysunek 6.21. Ustalenie konta dla agenta migawki

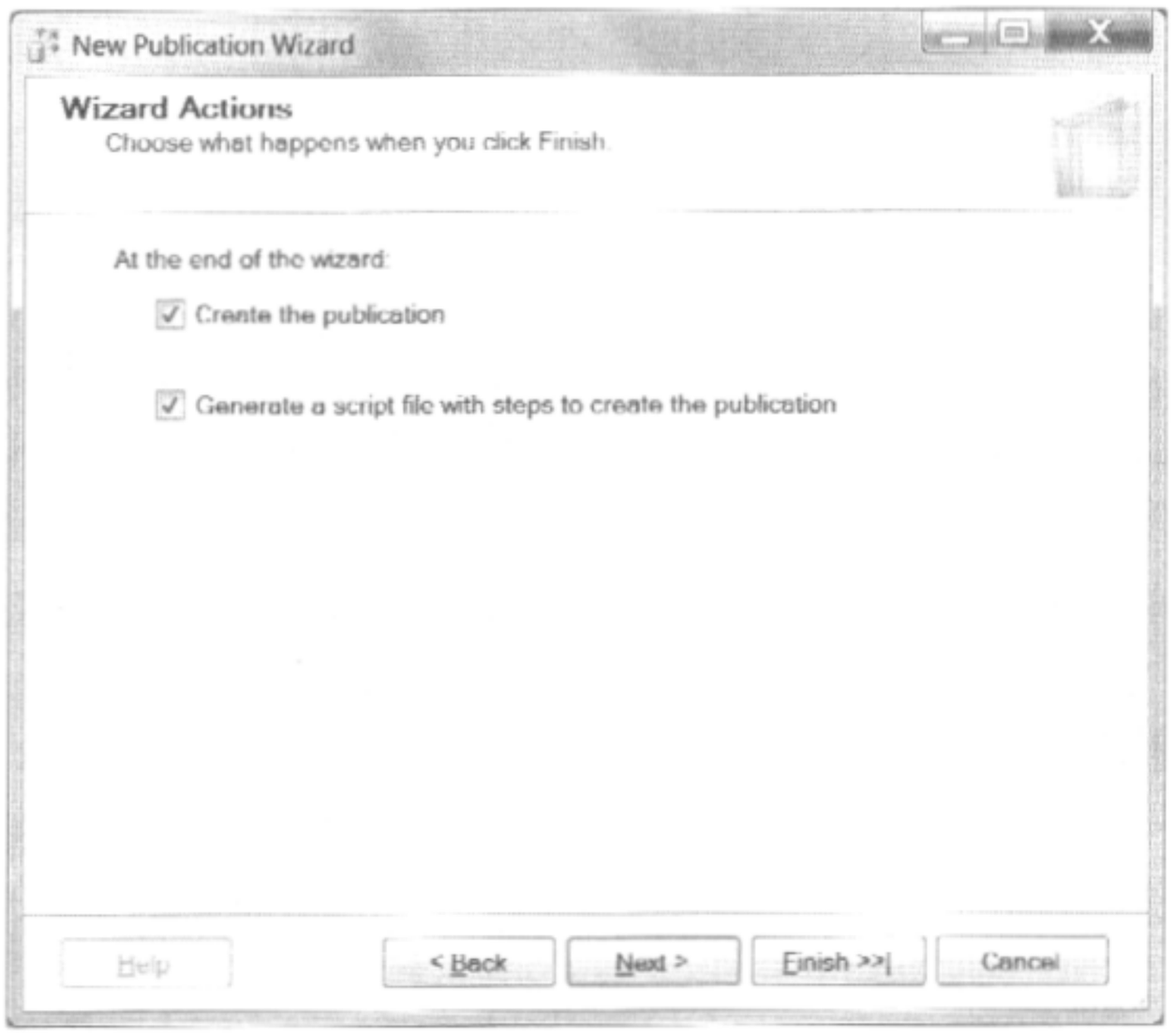


W kolejnym oknie trzeba zaznaczyć opcje automatycznego utworzenia repliki po zakończeniu pracy kreatora (rysunek 6.22) oraz określić, gdzie utworzony skrypt powinien zostać zapisany (rysunek 6.23).

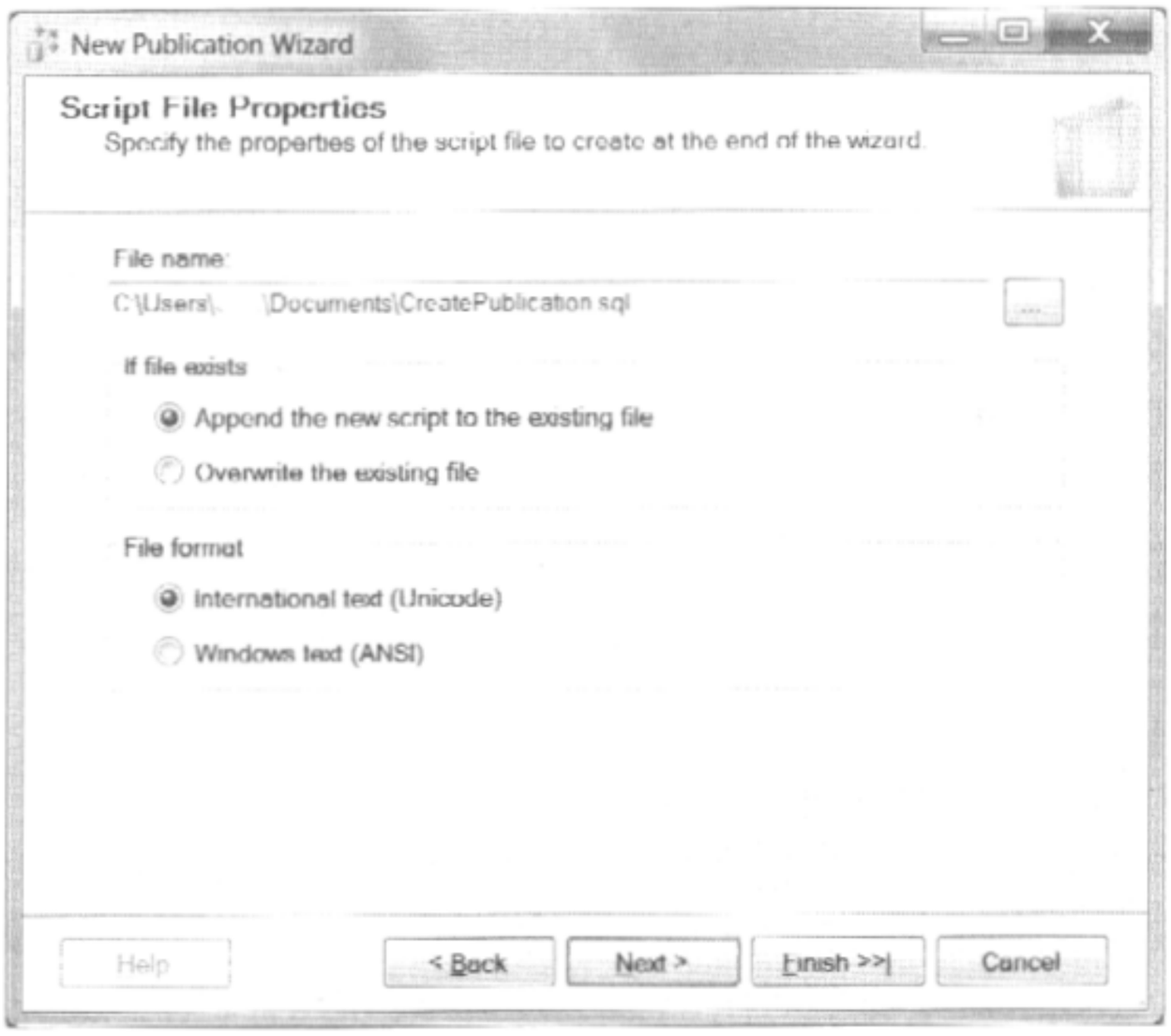
Następnie należy kliknąć przycisk *Finish* i w otwartym oknie nadać publikacji nazwę (rysunek 6.24). Kolejne kliknięcie przycisku *Finish* zakończy pracę kreatora publikacji.

Po wykonaniu wszystkich operacji publikacja zostanie utworzona.

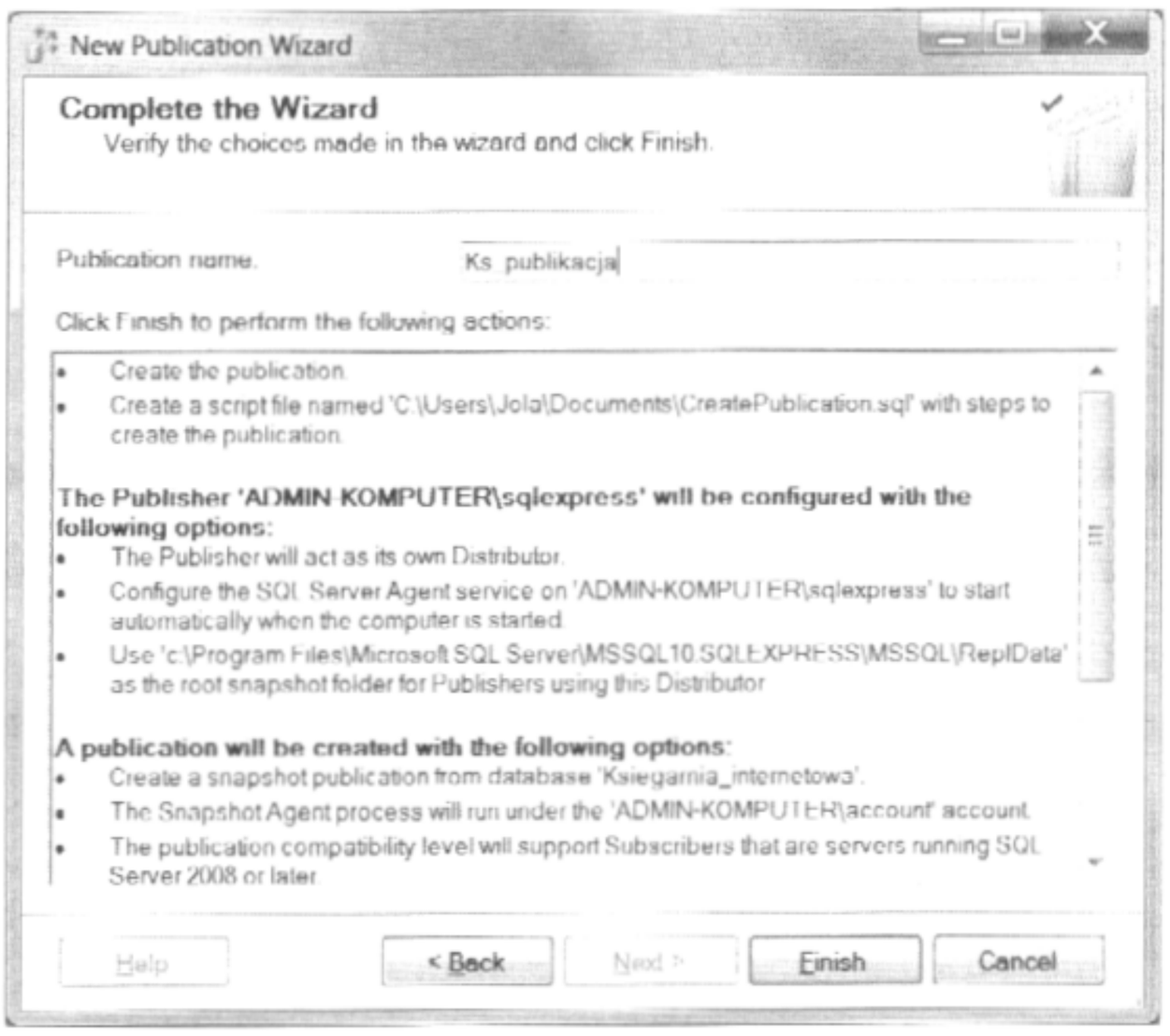
Rysunek 6.22.
Opcje tworzenia migawki



Rysunek 6.23.
Definiowanie ścieżki do skryptu migawki

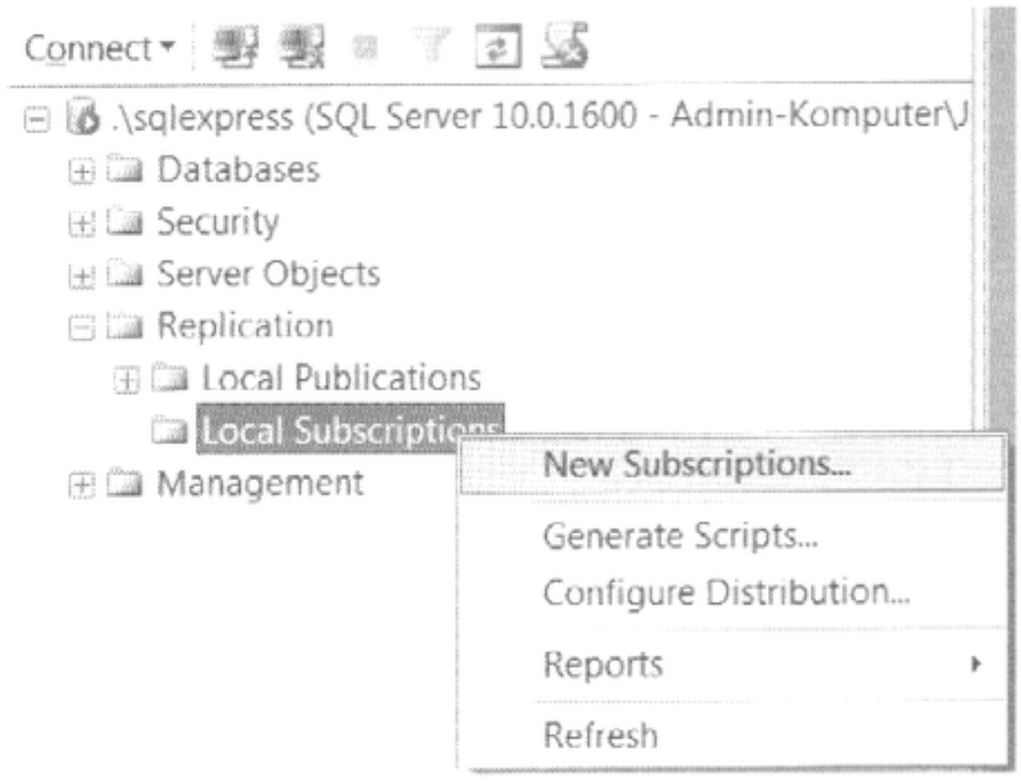


Rysunek 6.24.
Kończenie pracy kreatora publikacji



Tworzenie subskrypcji

W celu replikowania bazy danych należy połączyć się z instancją, do której będzie ona replikowana. W SQL Server Management Studio trzeba wybrać opcję tworzenia subskrypcji (rysunek 6.25).



Rysunek 6.25. Tworzenie subskrypcji

Zostanie uruchomiony kreator subskrypcji (ang. *Subscription Wizard*), który poprowadzi nas przez proces tworzenia subskrypcji.

Po wybraniu z listy skonfigurowanego poprzednio wydawcy (*Publisher*) należy wybrać bazę danych oraz nazwę utworzonej publikacji (*Databases and publications*). W kolejnym oknie wybieramy opcję wykonania zadania przez agentów na dystrybutorze (*Run all agents the Distributor*). Następnie wybieramy nazwę subskrybenta i z menu rozwijanego docelową bazę danych (*Subscription Database*). W kolejnym oknie wybieramy agenta subskrypcji (*Agent Subscriber*) oraz, po kliknięciu przycisku ...., konfigurujemy użytkownika i jego hasło. Kolejne okno pozwoli ustawić opcje pracy agenta subskrypcji (*Agent Schedule*). Do wyboru mamy tryb ciągły (*Run continuously*) lub na żądanie (*Run on demand only*). W następnym oknie można określić moment rozpoczęcia subskrypcji migawki (*Initialize When*). Do wyboru są opcje natychmiastowa (*Immediately*) lub przy pierwszej synchronizacji (*At first synchronization*). Na koniec zaznaczamy opcję wygenerowania skryptu z przeprowadzonych działań i klikając przycisk *Finish*, kończymy pracę kreatora.

Aby sprawdzić poprawność działania replikacji, łączymy się z instancją, w której znajduje się publikacja, i modyfikujemy zawartość wybranej tabeli. Przechodzimy do instancji, w której została skonfigurowana subskrypcja. Otwieramy zmodyfikowaną przed chwilą tabelę i sprawdzamy, czy wprowadzone zmiany zostały zapisane w bazie danych po stronie subskrybenta.

6.5. Kopie bezpieczeństwa

Jednym z podstawowych działań administratora baz danych jest zabezpieczenie przed ich utratą. Jedynie posiadanie kopii bezpieczeństwa bazy danych gwarantuje odzyskanie utraconych danych.

6.5.1. Tworzenie kopii bezpieczeństwa

Strategię wykonywania kopii bezpieczeństwa wypracowuje administrator serwera. Niezależnie od przyjętej strategii kopia bezpieczeństwa bazy danych powinna być wykonana zawsze, gdy:

- utworzono lub zmodyfikowano strukturę bazy danych,
- utworzono indeks,
- usunięto nieaktywną część dziennika.

Kopie bezpieczeństwa bazy danych mogą tworzyć administrator serwera oraz właściciel bazy danych. SQL Server umożliwia wykonanie kopii bezpieczeństwa całej bazy danych, wybranych plików (lub grup plików) bazy danych lub kopii dziennika transakcyjnego. Kopię bezpieczeństwa każdego typu można utworzyć, wykonując instrukcję `BACKUP`.

Instrukcją `BACKUP LOG` można wykonać kopię dziennika transakcyjnego wybranej bazy danych.

Pełna kopia bazy danych

Aby zachować wszystkie elementy bazy danych niezbędne do jej odtworzenia, powinniśmy wykonywać pełną kopię bazy danych. Przechowuje ona wszystkie informacje zapisane zarówno w plikach bazy danych (strukturę obiektów bazodanowych oraz dane tabel i indeksów), jak i w plikach dziennika transakcyjnego oraz wszystkie dane z aktywnej części dziennika.

Pełna kopia bazy danych jest wymagana do odtworzenia kopii przyrostowej lub kopii dziennika transakcyjnego bazy danych.

Przykład 6.14

```
BACKUP DATABASE Ksiegarnia_internetowa
 TO DISK = 'E:\SQLServerBackups\Ksiegarnia_kopia.bak'
   WITH FORMAT;
```

W podanym przykładzie została utworzona pełna kopia bazy danych *Ksiegarnia_internetowa*. Kopia ta została zapisana na dysku *E:* w pliku *Ksiegarnia_kopia*.

Przyrostowa kopia bazy danych

Tworzenie przyrostowych kopii baz danych pozwala skrócić czas potrzebny na odtworzenie bazy z plików kopii dziennika transakcyjnego oraz czas potrzebny na wykonanie kopii plików bazodanowych. Do pliku przyrostowej kopii bazy danych zostaną zapisane wszystkie dane, które zostały zmodyfikowane od czasu wykonania ostatniej pełnej kopii bazy danych.

Przykład 6.15

```
BACKUP DATABASE Ksiegarnia_internetowa
TO Ksiegarnia_kopia_P WITH DIFFERENTIAL
```

W podanym przykładzie została utworzona przyrostowa kopia bazy danych *Ksiegarnia_internetowa*. Będzie ona zawierała tylko zapis zmian, które nastąpiły od momentu wykonania pełnej kopii bazy.

W przypadku bardzo dużych baz danych codzienne wykonywanie kopii całej bazy może być trudne. W tym przypadku zamiast wykonywania kopii zapasowej całej bazy można utworzyć kopię wybranego pliku (lub grupy plików).

Przykład 6.16

```
BACKUP DATABASE Ksiegarnia_internetowa
FILE = 'Dane', FILEGROUP = 'Grupa1' TO Kopia_ksiegarnia
```

W podanym przykładzie została wykonana kopia zapasowa tylko wybranych plików bazy danych *Ksiegarnia_internetowa*.

6.5.2. Sprawdzanie spójności bazy danych

Przed wykonaniem kopii bezpieczeństwa powinno się sprawdzić spójność bazy danych. Utworzenie kopii niespójnej bazy danych może uniemożliwić późniejsze jej odtworzenie. Spójność bazy można sprawdzić za pomocą polecenia `DBCC CHECKDB`. Polecenie `DBCC CHECKCATALOG` sprawdza integralność referencyjną tabel systemowych, polecenie `DBCC CHECKALLOC` — informacje na temat alokacji, a polecenie `DBCC CHECKTABLE` — spójność na poziomie jednej tabeli.

Polecenie `DBCC CHECKDB` ma następujące opcje:

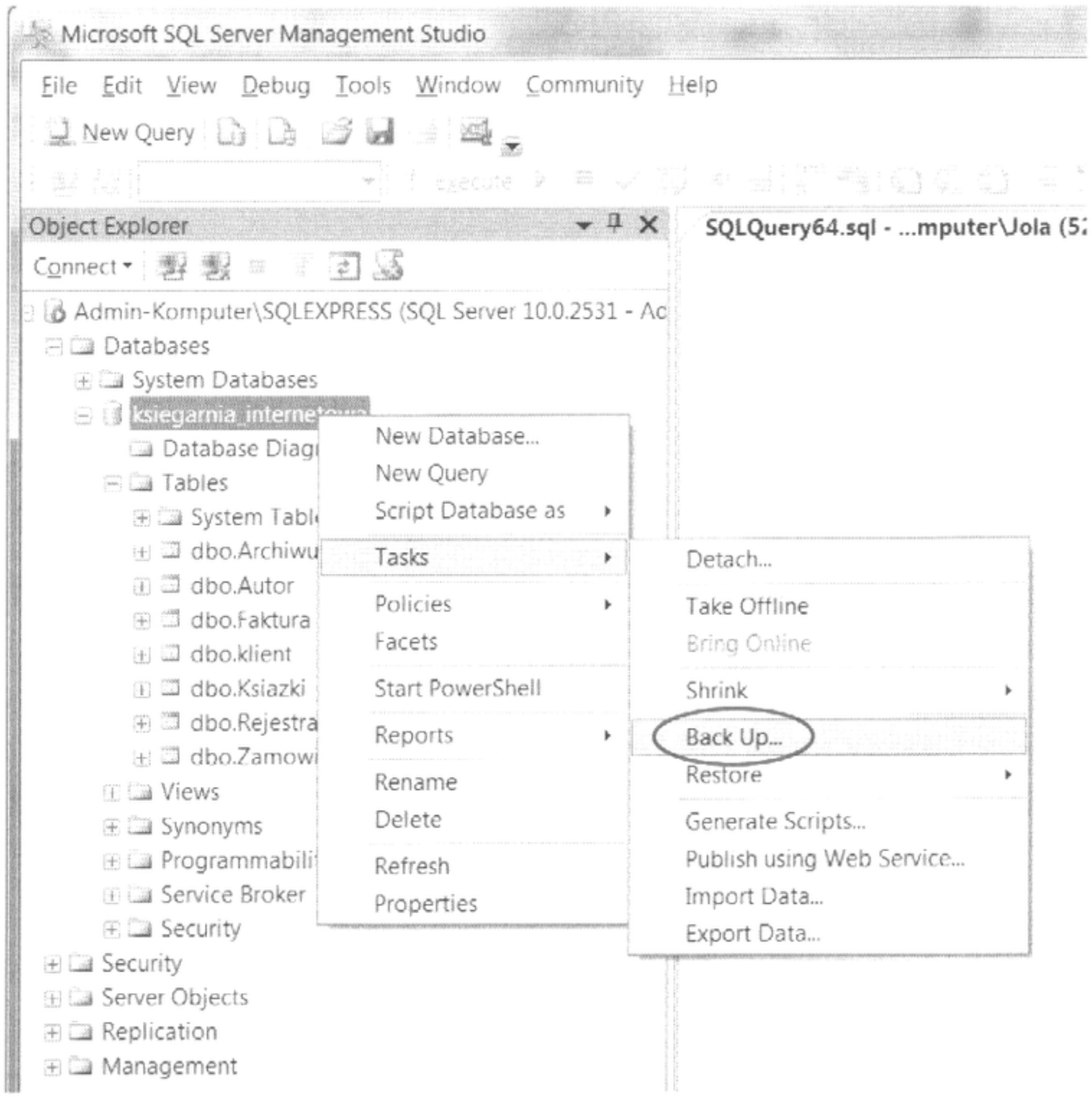
- `Repair_Allow_Data_Loss` — naprawia uszkodzone indeksy. Uszkodzone strony są czyszczone, nawet jeśli spowoduje to utratę danych.
- `Repair_fast` — naprawia uszkodzone indeksy, jeżeli nie spowoduje to utraty danych.
- `Repair_Rebuild` — naprawia uszkodzone indeksy, ale może tworzyć je ponownie.

Przykład 6.17

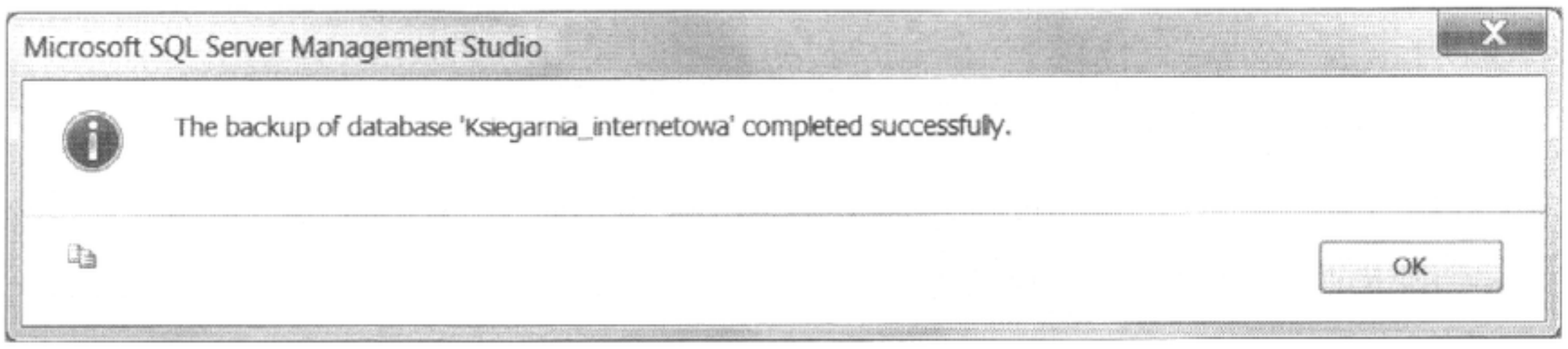
```
DBCC CHECKDB ('Ksiegarnia_internetowa', Repair_fast)
```

Kopię zapasową w programie SQL Server Management Studio wykonuje się, wybierając w oknie *Object Explorer* określoną bazę. Następnie klikamy prawym przyciskiem myszy bazę i wybieramy opcję *Tasks/Back Up...* (rysunek 6.26).

W otwartym oknie należy sprawdzić ustawienia dotyczące bazy danych. Można zmienić miejsce docelowe oraz nazwę pliku z kopią bazy. Po kliknięciu *OK* zostanie utworzona kopia zapasowa bazy (rysunek 6.27).



Rysunek 6.26. Wykonanie kopii zapasowej bazy danych



Rysunek 6.27. Potwierdzenie utworzenia kopii zapasowej

6.5.3. Przywracanie bazy danych z kopii bezpieczeństwa

Przywracanie bazy danych z kopii bezpieczeństwa odbywa się przy użyciu polecenia `RESTORE DATABASE`. Istnieje kilka metod przywracania bazy:

- przywracanie całej bazy danych z kopii zapasowej (pełne przywracanie),
- przywracanie części bazy danych (częściowe przywracanie),
- przywracanie wybranych plików lub grup plików do bazy danych (plik przywracania),
- przywracanie dziennika transakcji do bazy danych (plik dziennika transakcji).

Przykład 6.18

Przywracanie pełnej bazy danych:

```
RESTORE DATABASE Ksiegarnia_internetowa
FROM Ksiegarnia_kopia.bak;
```

Przykład 6.19

Nadpisywanie istniejącej bazy danych:

```
RESTORE DATABASE Ksiegarnia_internetowa
FROM Ksiegarnia_kopia.bak WITH replace;
```

6.6. Import i eksport danych

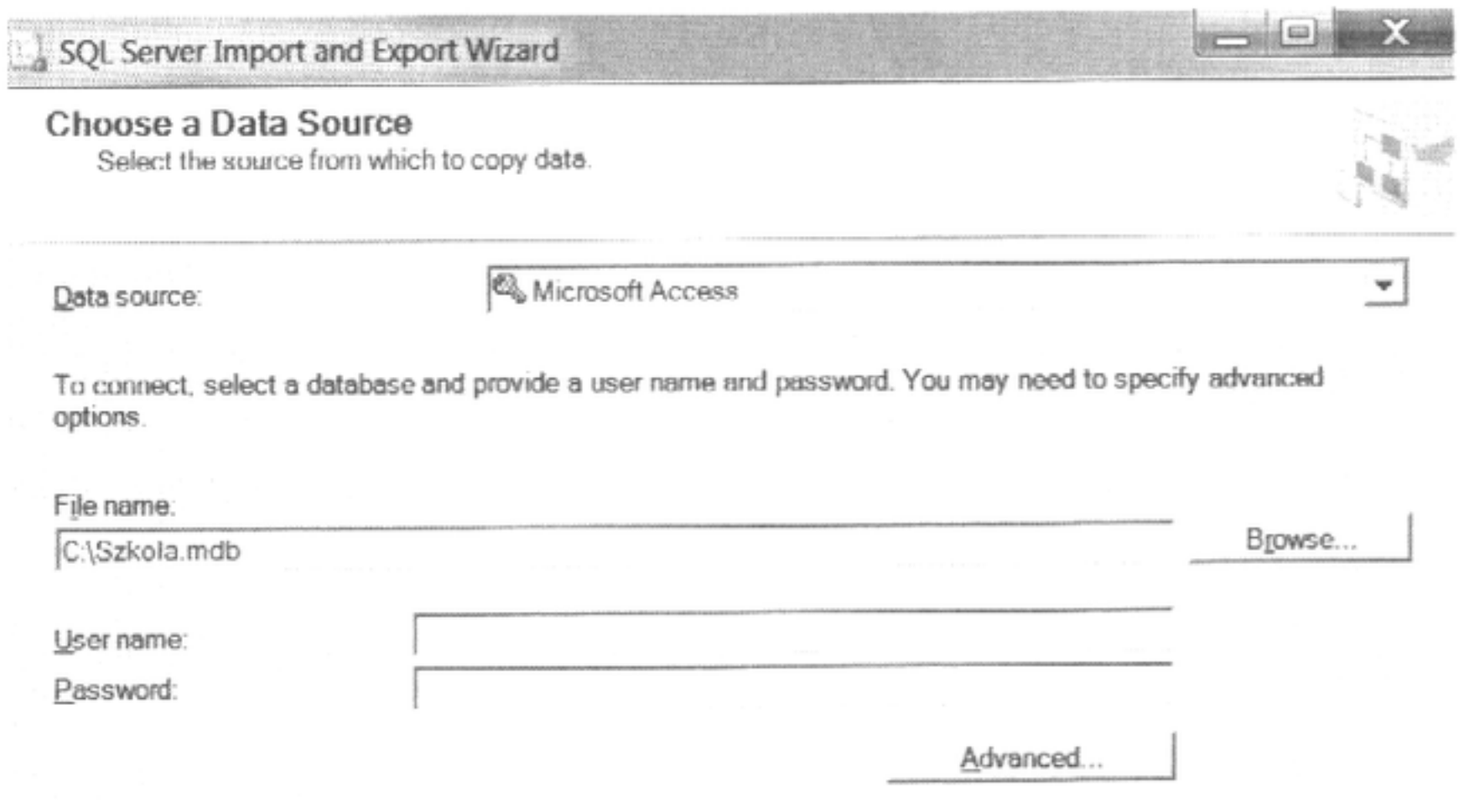
Podczas instalowania SQL Server Management Studio instalowany jest program DTS (*Data Transformation Service*), który może być bardzo przydatnym narzędziem umożliwiającym import danych do serwera SQL Server lub eksport danych z niego.

Źródłem danych dla serwera SQL Server mogą być dane z: *.NET Framework*, *OLE DB*, *SQL Server Native Client*, *ADO.NET*, *MS Excel*, *MS Access*. W zależności od pochodzenia danych można ustawić opcje określające tryb uwierzytelnienia, nazwę serwera, nazwę bazy danych, nazwę pliku.

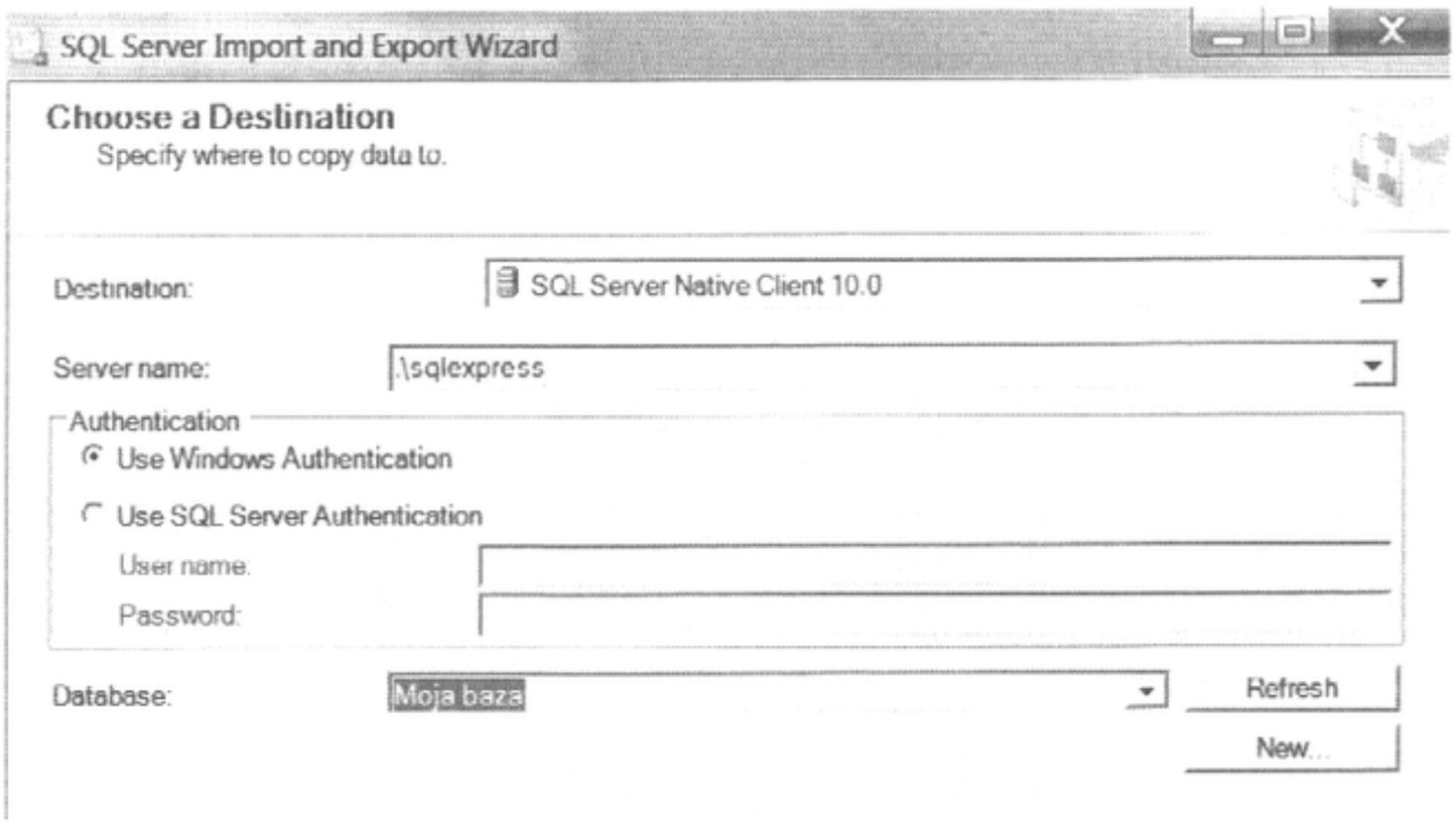
Dane mogą zostać wyeksportowane do: *.NET Framework*, *OLE DB*, *SQL Server Native Client*, *ADO.NET*, *MS Excel*, *MS Access*.

Aby uruchomić program DTS, należy z menu *Start* systemu Windows wybrać *Wszystkie programy/Microsoft SQL Server 2008/Importuj i eksportuj dane*. Zostanie uruchomiony kreator importu i eksportu danych.

W pierwszym oknie kreatora trzeba wybrać typ źródła danych (rysunek 6.28), a w kolejnym miejsce docelowe danych (rysunek 6.29).

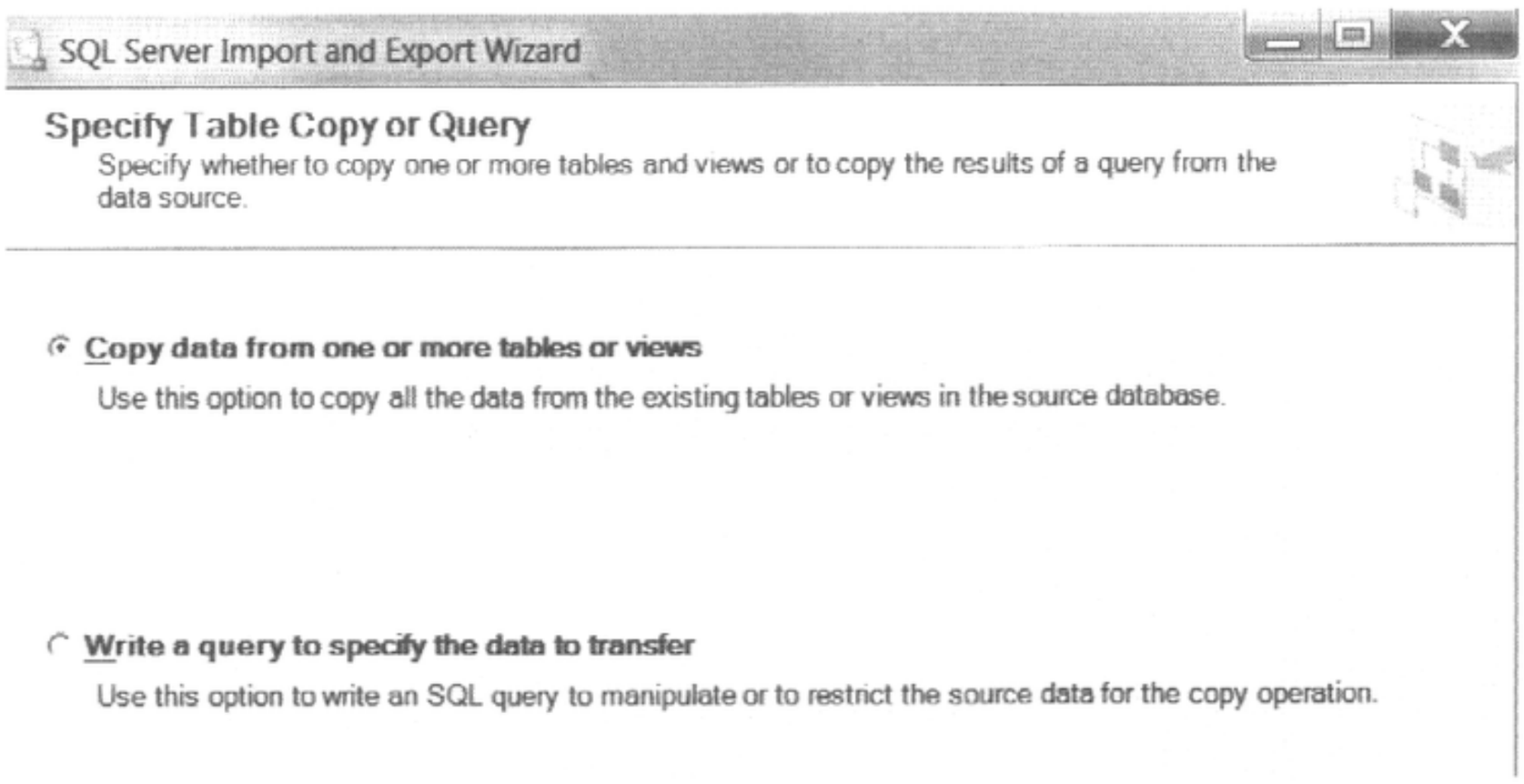


Rysunek 6.28. Wybór źródła importowanych danych



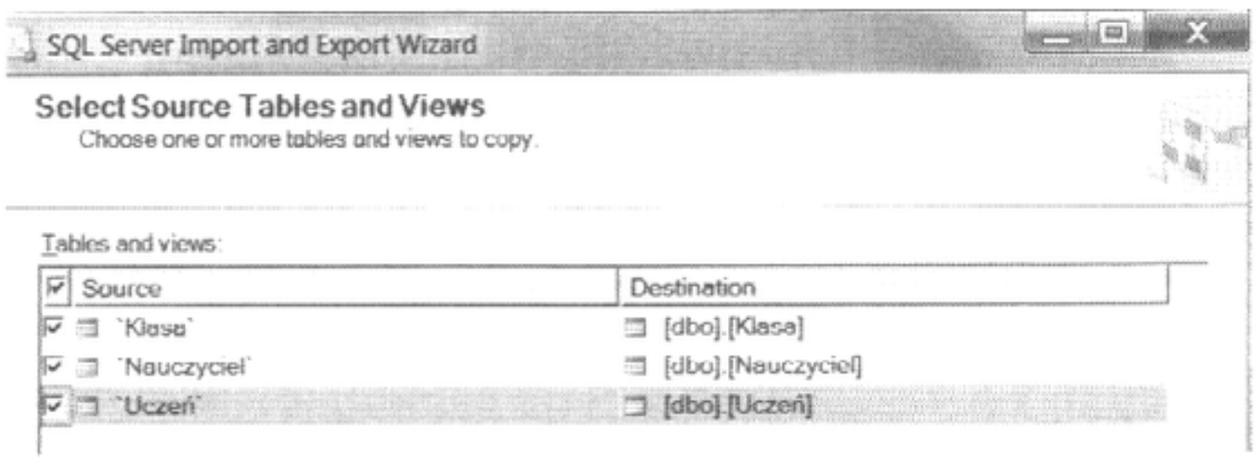
Rysunek 6.29. Wybór miejsca docelowego importowanych danych

W kolejnym oknie importowania plików można określić, czy będą przenoszone dane z tabel lub widoków, czy wyniki zapytania (rysunek 6.30).



Rysunek 6.30. Wybór rodzaju importowanych danych

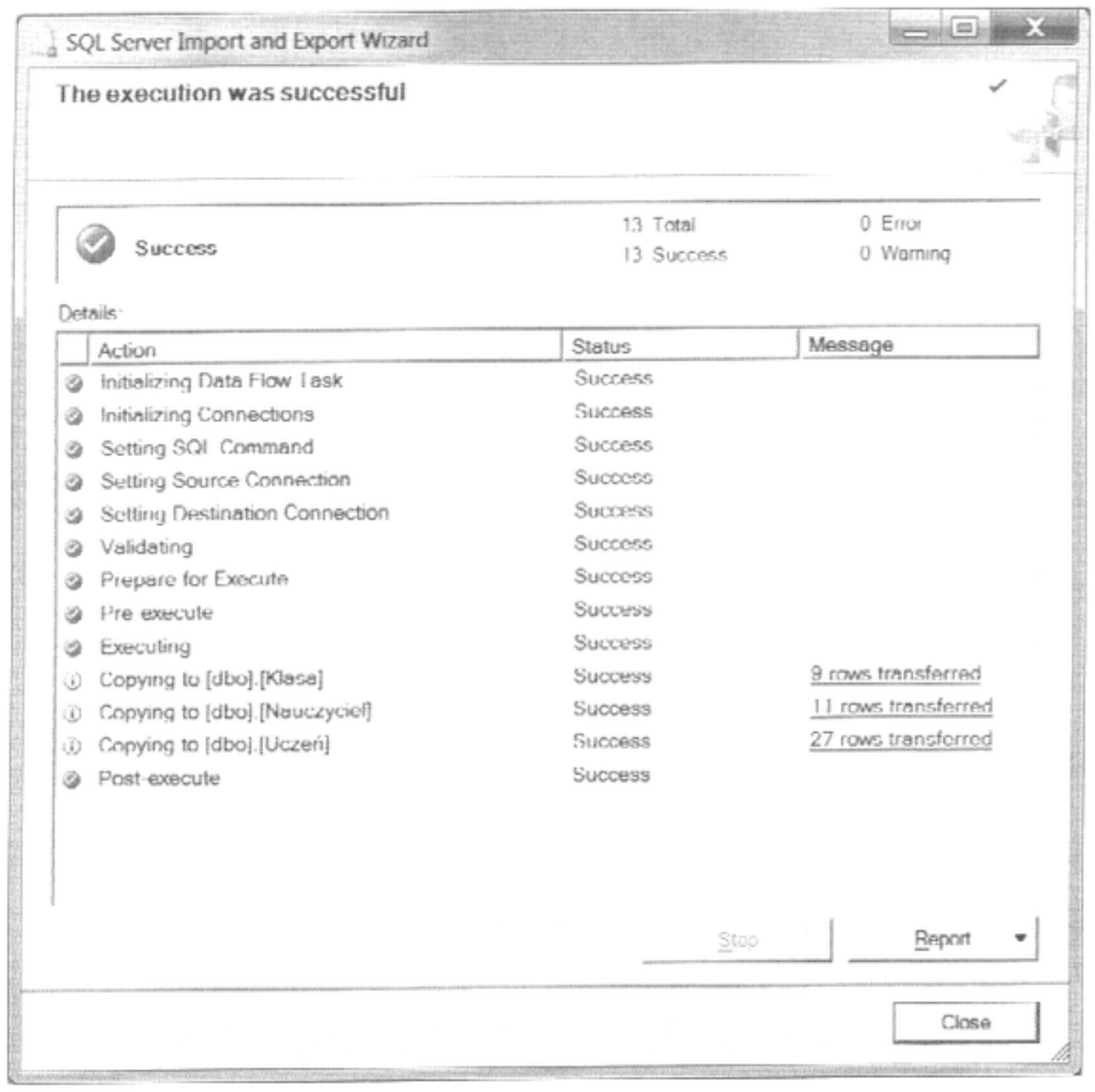
Następne okno zawiera listę obiektów, z których należy wybrać te, które zostaną zaimportowane (rysunek 6.31).



Rysunek 6.31. Wybór importowanych danych

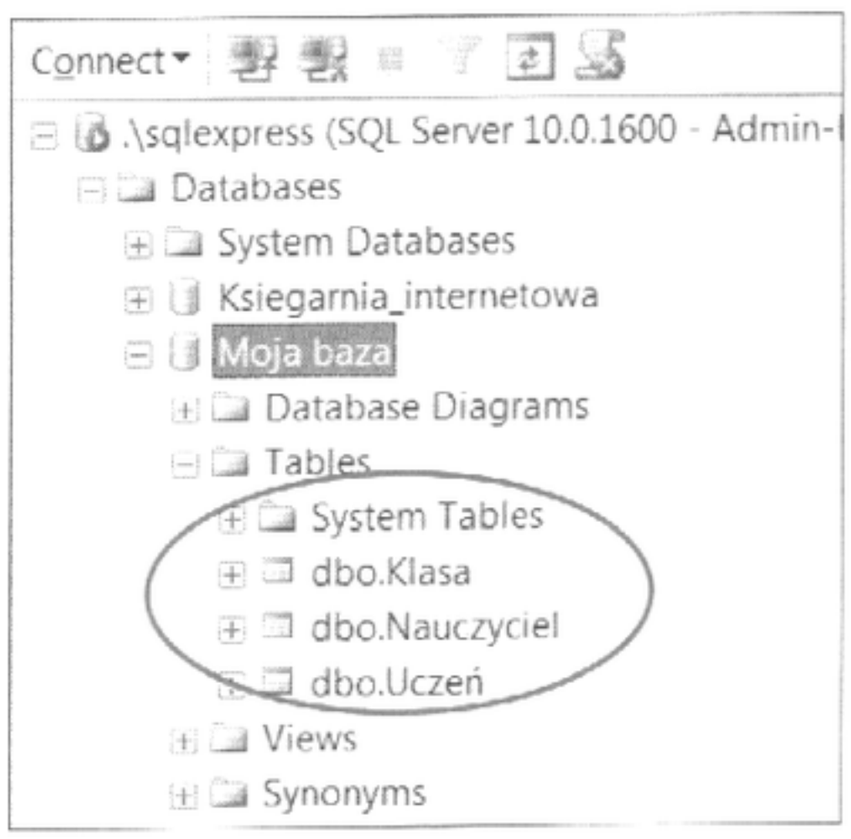
Po wykonaniu tych czynności dane zostaną zaimportowane do bazy docelowej (rysunek 6.32).

Rysunek 6.32. Efekt importowania danych do bazy



Dane bazy danych programu Access zostały zaimportowane do bazy SQL Server (rysunek 6.33).

Rysunek 6.33. Dane zaimportowane do bazy SQL Server



W ten sam sposób można wykonać operację eksportowania danych, określając jako źródło danych bazę SQL Server, a jako miejsce docelowe jednego z wcześniej wymienionych dostawców danych.

Zadanie 6.2

Tabele utworzonej wcześniej w programie MS Access bazy danych *Księgarnia internetowa* zaimportuj do bazy SQL Server.

6.7. Udostępnianie zasobów w sieci

Jeśli użytkownik chce połączyć się z bazą danych znajdującą się na serwerze z innego komputera, powinien włączyć protokół TCP/IP.

Konfigurację protokołu TCP/IP można zrealizować za pomocą narzędzia SQL Server Configuration Manager. W tym celu należy w menu *Start* wybrać *Wszystkie programy/Microsoft SQL Server 2008/Configuration Tools/SQL Server Configuration Manager*.

Po uruchomieniu tego programu w sekcji *SQL Server Network Configuration* trzeba wybrać pozycję *Protocols for <InstanceName>*.

InstanceName to nazwa instancji, która jest dostępna na serwerze. Domyślnie jest to instancja *sqlexpress*.

Na liście protokołów należy kliknąć prawym przyciskiem myszy protokół, który zostanie włączony (na przykład TCP/IP), a następnie kliknąć przycisk *Enable*.

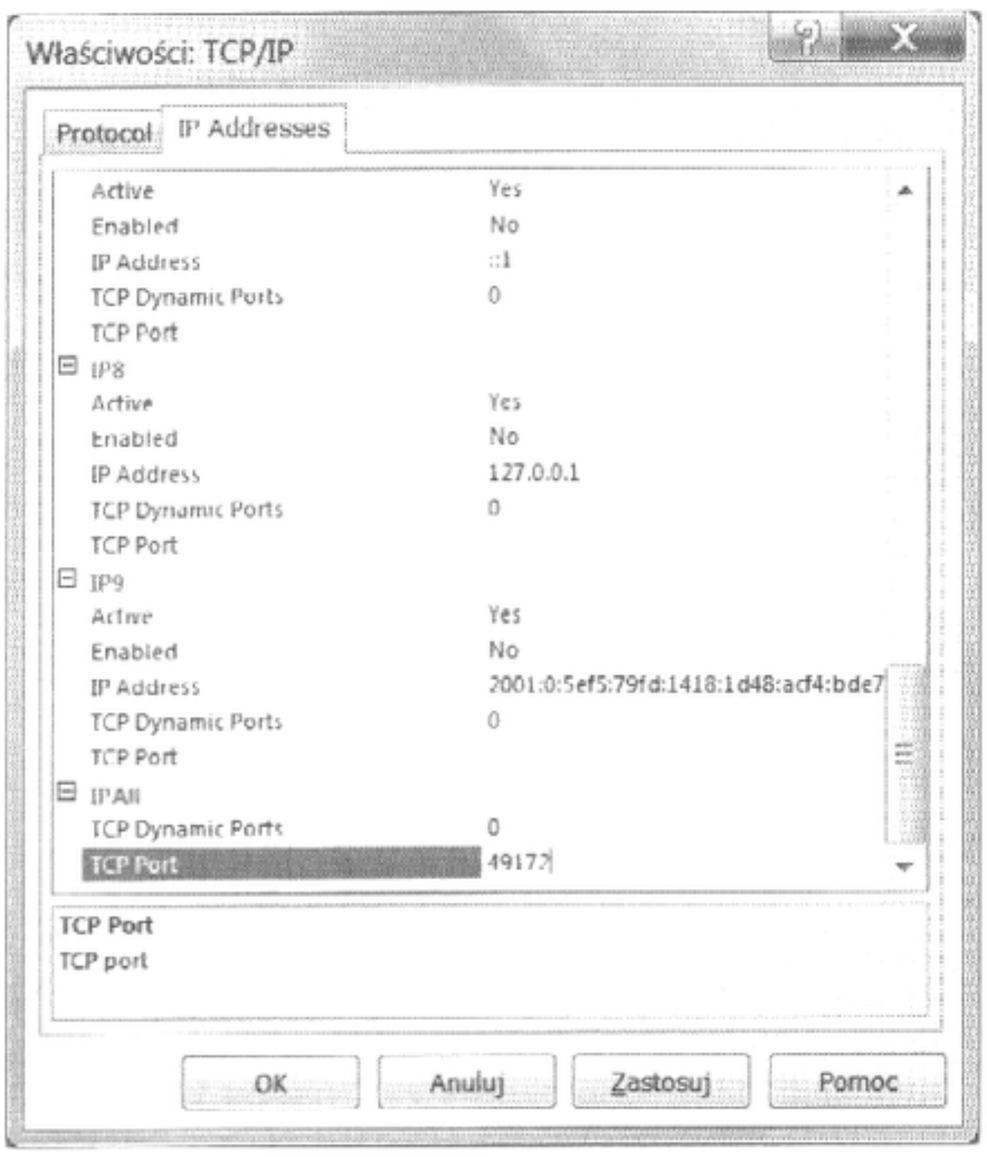
6.7.1. Konfigurowanie portu

W celu zwiększenia poziomu zabezpieczeń w systemach Microsoft Windows włączona jest zapora systemu. Jeśli użytkownik chce połączyć się z serwerem SQL Server z innego komputera, w zaporze trzeba otworzyć port komunikacyjny. Domyślnie aparat bazy danych nasłuchuje na porcie *1433*. Jednak utworzone instancje nasłuchują na portach dynamicznych. Dlatego przed otwarciem portu w zaporze należy skonfigurować serwer bazy danych w celu nasłuchiwania na określonym porcie, w przeciwnym razie po każdym uruchomieniu serwer bazy danych może nasłuchiwać na innym porcie.

Konfigurowanie serwera SQL Server do nasłuchiwania na określonym porcie

W programie SQL Server Configuration Manager należy wybrać opcję *SQL Server Network Configuration*, a następnie kliknąć instancję serwera, która ma zostać skonfigurowana. W prawym panelu trzeba dwukrotnie kliknąć protokół *TCP/IP*. Zostanie otwarte okno dialogowe *TCP/IP Properties*, w którym trzeba wybrać kartę *IP Addresses*.

W polu *TCP Port* w sekcji *IPAll* trzeba wpisać dostępny numer portu, na przykład *49172* (rysunek 6.34), i zamknąć okno dialogowe. Jeśli pojawi się okno z ostrzeżeniem o ponownym uruchomieniu usługi, należy kliknąć przycisk *OK*.



Rysunek 6.34. Konfigurowanie serwera do nasłuchiwania na określonym porcie

W oknie *SQL Server Configuration Manager* w lewym panelu należy wybrać opcję *SQL Server Services* i w prawym panelu, po kliknięciu prawym przyciskiem myszy opcji *SQL Server*, wybrać opcję *Restart*. Po ponownym uruchomieniu serwera SQL Server aparat bazy danych będzie nasłuchiwał na wskazanym porcie — *49172*.

Otwieranie portu w zaporze systemu Windows

Otwieranie zapory systemu Windows zostało przedstawione na przykładzie Windows XP.

Aby otworzyć port w zaporze, trzeba w menu *Start* wybrać *Panel sterowania/Połączenia sieciowe i internetowe/Zapora systemu Windows*, a następnie kartę *Wyjątki* i kliknąć przycisk *Dodaj port*. W oknie dialogowym *Dodaj port* w polu *Nazwa* należy wpisać *Serwer SQL<nazwa instancji>*. W polu *Numer portu* wpisujemy numer portu instancji bazy danych. W przypadku instancji domyślnej trzeba użyć portu *1433*. W przypadku konfigurowania utworzonej instancji należy wpisać numer portu, który został skonfigurowany do nasłuchiwania w serwerze SQL Server, na przykład *49172*. Ustawienia te należy wybrać dla protokołu TCP.

Łączenie się z aparatem bazy danych z innego komputera

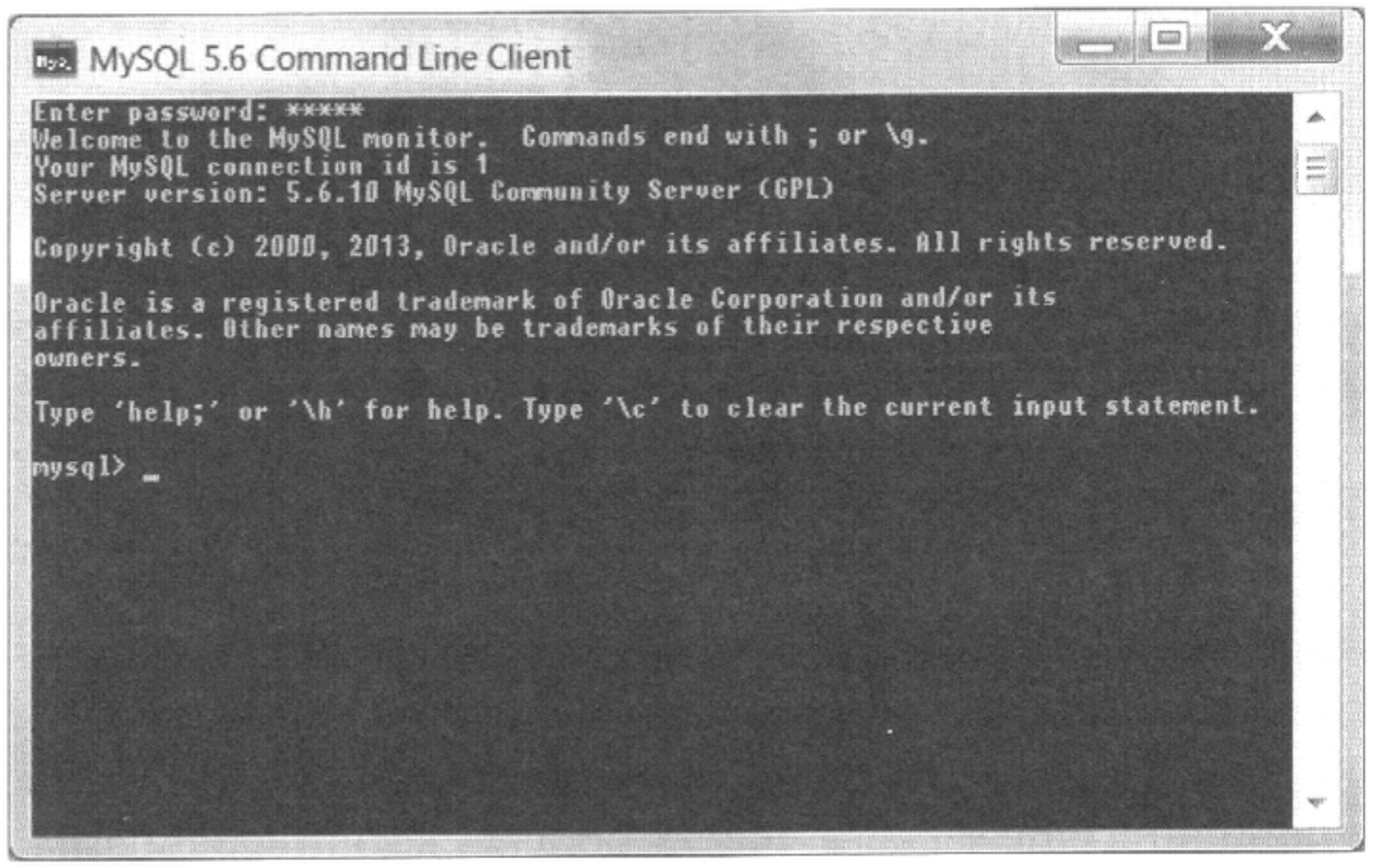
Po skonfigurowaniu serwera bazy danych do nasłuchiwania na porcie stałym i po otwarciu w zaporze odpowiedniego portu można połączyć się z programem SQL Server z innego komputera.

Komputer, z którego nastąpi połączenie z serwerem SQL Server, musi posiadać narzędzia klienta programu SQL Server. Po zalogowaniu się i uruchomieniu programu SQL Management Studio w oknie dialogowym *Connect to Server*, w polu *Server type*, powinna być wybrana pozycja *Database Engine*. W polu *Server name* należy wpisać `tcp:\nazwa_komputera, numer_portu` (na przykład `tcp:\SQLEXPRESS,1433` dla instancji domyślnej lub `tcp:\nazwa,49172` dla instancji utworzonej). W polu *Authentication* powinna zostać wybrana opcja *Window Authentication*. Po tych ustawieniach trzeba wybrać przycisk *Connect*, aby połączyć się z bazą SQL Server.

6.8. MySQL

6.8.1. Konfigurowanie serwera

Konfigurowanie serwera MySQL przez administratora można przeprowadzić za pomocą wiersza poleceń (rysunek 6.35) lub odpowiednich narzędzi, na przykład programów MySQL Workbench, MySQL Administrator lub phpMyAdmin. Narzędzia te są dostępne zarówno na platformie Windows, jak i Linux. Konfiguracja serwera polega na modyfikacji pliku *my.ini* w systemach Windows lub *my.cnf* w systemach Linux.



Rysunek 6.35. Wiersz poleceń programu MySQL

Konfigurowanie w systemie Windows

W systemie Windows ustawienia konfiguracyjne serwera MySQL są przechowywane w kluczu rejestru systemu:

```
HKEY_LOCAL_MACHINE\Software\MySQL AB\MySQL Server 5.5
```

Konfigurację serwisu przechowuje klucz

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MySQL
```

Domyślnym folderem instalacyjnym serwera MySQL jest *C:\Program Files\MySQL\MySQL Server 5.6*.

Istotne z punktu widzenia administratora mogą być foldery:

- *bin* — serwer MySQL, programy użytkowe;
- *data* — bazy danych, pliki dzienników;
- *examples* — przykładowe programy i skrypty;
- *include* — pliki nagłówkowe;
- *lib* — biblioteki;
- *scripts* — skrypty użytkowe;
- *share* — pliki błędów, zestawy znaków.

Uruchamianie serwera

Uruchamianie i zatrzymanie serwera w systemie Windows można wykonać za pomocą systemowych narzędzi administracyjnych przez zatrzymanie lub uruchomienie usługi.

Można go również uruchomić lub zatrzymać, wpisując w wierszu poleceń jedną z poniższych komend:

```
NET START mysql
```

```
NET STOP mysql

mysql -u root -h localhost -p
```

Parametr `-u` określa konto, na które chcemy się zalogować; parametr `-h` określa komputer, na którym pracuje serwer baz danych; parametr `-p` określa uwierzytelnienie za pomocą hasła. W wyniku wykonania podanego wyżej polecenia zostaniemy zalogowani do serwera jako użytkownik *root*.

Konfigurowanie w systemie Linux

W systemie Linux konfiguracja serwera dla ustawień globalnych jest przechowywana w plikach */etc/my.cnf*, konfiguracja dla specyficznych ustawień — w plikach *~/.my.cnf*. Niektóre ustawienia mogą być przekazywane jako opcje w trakcie uruchamiania serwera z linii komend lub zmieniane w trakcie pracy za pomocą polecenia `SET`. Aby odczytać opcje konfiguracyjne serwera, należy podczas jego uruchamiania z linii komend zdefiniować dodatkowe parametry uruchamiania:

```
mysql --verbose --help
```

Zainstalowany i skonfigurowany serwer można przetestować.

Sprawdzenie, czy serwer odpowiada, można przeprowadzić poleceniem:

```
mysqladmin version
```

lub:

```
mysqladmin variables
```

Sprawdzenie, czy serwer zwraca dane, można przeprowadzić poleceniem:

```
mysqlshow
```

lub:

```
mysqlshow -u root mysql
```

Uruchamianie serwera

Uruchamianie serwera w systemie Linux polega na wykonaniu polecenia:

```
/usr/bin/mysqld_safe
```

Inną metodą uruchomienia i zatrzymania serwera jest wpisanie poleceń:

```
/usr/share/mysql/mysql.server start

/usr/share/mysql/mysql.server stop
```

W systemach Windows i Linux do uruchomienia i zatrzymania serwera można użyć poleceń:

`/usr/sbin/mysqld` (uruchomienie serwera)

i `usr/bin/mysqladmin -u root -p shutdown` (zatrzymanie serwera).

6.8.2. Tryby uwierzytelnienia

Aby połączyć się z serwerem, użytkownik musi potwierdzić swoją tożsamość, podając login i hasło. Jest to proces *uwierzytelnienia użytkownika*. Jeżeli użytkownik zalogowany do bazy danych próbuje wykonać jakąkolwiek operację (na przykład odczytanie lub modyfikowanie danych), serwer sprawdza, czy ma on wystarczające uprawnienia do wykonania tej operacji. Jeżeli użytkownik nie ma wymaganych uprawnień, nastąpi przerwanie wykonywania operacji. Ten proces nazywany jest *autoryzacją*.

Podczas instalowania serwera MySQL tworzone jest wbudowane konto *root*. Ma ono pełne uprawnienia i w codziennej pracy nie powinno być używane nawet przez administratora. Korzystając z konta *root*, wyłączamy cały mechanizm autoryzacji i obniżamy poziom bezpieczeństwa serwera. Założenie konta użytkownika i nadanie użytkownikowi minimalnych uprawnień potrzebnych do wykonywania jego pracy powinno być podstawową zasadą bezpieczeństwa podczas korzystania z serwera baz danych.

Serwer MySQL powinien być uruchamiany z konta zwykłego użytkownika, specjalnie w tym celu utworzonego, z minimalnymi prawami, które są potrzebne do pracy. Aby serwer uruchamiał się z tego konta, w pliku *my.cnf* należy dodać wpis:

```
[mysqld]
user=mysql
```

W serwerze MySQL konta użytkownika związane są z nazwą komputera, z którego użytkownik korzysta. Jeżeli użytkownik łączy się z serwerem z wielu komputerów, to dla każdego z nich możliwe jest ustawienie innego hasła i innych praw dostępu.

Utworzenie użytkownika realizuje polecenie `CREATE USER`.

Przykład 6.20

```
CREATE USER adam IDENTIFIED BY 'haslo';
CREATE USER adam IDENTIFIED BY PASSWORD 'tajne_haslo';
```

W wyniku wykonania polecenia w tabeli *mysql.user* zostanie utworzony wpis oznaczający, że użytkownik *adam* może korzystać z dowolnego komputera do połączenia się z serwerem.

Aby usunąć użytkownika, należy użyć polecenia `DROP USER`.

Przykład 6.21

```
DROP USER adam;
```

Nazwę bieżącego użytkownika można uzyskać po wpisaniu polecenia:

```
SELECT CURRENT_USER()
```

Zmianę hasła bieżącego użytkownika uzyskamy po wpisaniu polecenia:

```
SET PASSWORD = PASSWORD('haslo');
```

Zmiana hasła dla innego konta wymaga odpowiednich uprawnień i ma postać:

```
SET PASSWORD FOR adam = PASSWORD('haslo_adama');
```

Zadanie 6.4

Skonfiguruj zainstalowany w wybranym przez siebie systemie operacyjnym serwer MySQL. Utwórz trzech nowych użytkowników: *Operator* z hasłem, *Instruktor* z hasłem oraz *Gracz* bez hasła.

6.8.3. Zarządzanie bazami danych

Podczas instalowania serwera MySQL automatycznie tworzone są dwie bazy systemowe: *information_schema* oraz *mysql*. Baza *information_schema* zawiera metadane na temat pozostałych baz danych. Baza *mysql* zawiera między innymi informacje o użytkownikach serwera i ich uprawnieniach.

Tworzenie bazy danych

Do tworzenia bazy danych służy polecenie `CREATE DATABASE`, do zmiany parametrów bazy — polecenie `ALTER DATABASE`, zaś do usunięcia bazy polecenie `DROP DATABASE`.

Przykład 6.22

```
CREATE DATABASE Towary;

ALTER DATABASE Towary CHARACTER SET latin2;

DROP DATABASE Towary;
```

Informację o bazie danych można wyświetlić, używając polecenia `SHOW DATABASES`. Informację o tabelach w aktualnej bazie danych wyświetlimy, wpisując polecenie `SHOW TABLES`. Informacje o wybranej tabeli w aktualnej bazie danych można wyświetlić, używając polecenia `DESCRIBE Nazwa_tabeli`.

6.8.4. Typy tabel w MySQL

W serwerze MySQL występuje wiele mechanizmów obsługi tabel. Każdy z nich jest przeznaczony do innego zastosowania. Przy użyciu polecenia `CREATE DATABASE` i jego klauzul `ENGINE` lub `TYPE` można zdefiniować mechanizm przechowywania danych, który zostanie użyty w definiowanej tabeli.

MyISAM

Jest to domyślny typ tabeli. Tego typu tabele pozwalają na szybkie odczytywanie danych, ale nie obsługują transakcji. Powinny być wykorzystywane do przechowywania rzadko zmienianych danych.

MEMORY

Tabele tego typu są przechowywane w pamięci operacyjnej, a dane w nich zapisane są tracone bezpowrotnie po wyłączeniu serwera. Wykorzystywane są w nich indeksy

mieszane (ang. *Hash*), dlatego modyfikacje danych są bardzo szybkie. Tabele tego typu powinny być używane jako tabele tymczasowe. Każda tabela typu `MEMORY` przechowywana jest w postaci pojedynczego pliku z rozszerzeniem *frm*, który zawiera jedynie definicję tabeli.

FEDERATED

Tabele tego typu definiują dane znajdujące się na innym serwerze. W przeciwieństwie do tabel innych typów tabele FEDERATED są bezpośrednio dostępne ze zdalnych serwerów MySQL. Serwer lokalny łączy się z serwerem zdalnym i bezpośrednio zapisuje dane w tabeli lub odczytuje je.

BLACKHOLE

Dane zapisywane w tego typu tabelach nie są przechowywane. Zapisywane w nich wiersze są automatycznie usuwane. Ze względu na to, że informacje o operacjach wykonywanych na nich są zapisywane w dzienniku zdarzeń serwera MySQL, tabele tego typu są przydatne podczas diagnozowania i testowania baz danych.

CSV

Tabele tego typu służą do przechowywania danych w plikach tekstowych typu *csv*. Są używane do importowania i eksportowania danych pomiędzy serwerem MySQL a innymi serwerami lub programami.

ARCHIVE

Tabele tego typu służą do przechowywania dużych ilości danych. Dane są przechowywane bez indeksów i są kompresowane przed umieszczeniem w tabeli. Podczas odczytu danych następuje ich dekompresja. Zmienianie lub usuwanie tych danych jest niedozwolone. Tabele typu `ARCHIVE` są używane głównie do przechowywania zarchiwizowanych danych lub danych diagnostycznych.

InnoDB

Są to tabele obsługujące transakcje. Serwer MySQL automatycznie blokuje odczytywane i modyfikowane wiersze tych tabel. Tabele `InnoDB` pozwalają również definiować i sprawdzać ograniczenia klucza obcego.

Aby wyświetlić informacje o dostępnych na serwerze typach tabel, należy użyć polecenia:

```
SHOW ENGINES\G;
```

6.9. Narzędzia administracyjne

Zarządzanie serwerem i obsługę baz danych można realizować w trybie wiersza poleceń lub za pomocą odpowiednich narzędzi, na przykład programu MySQL Workbench.

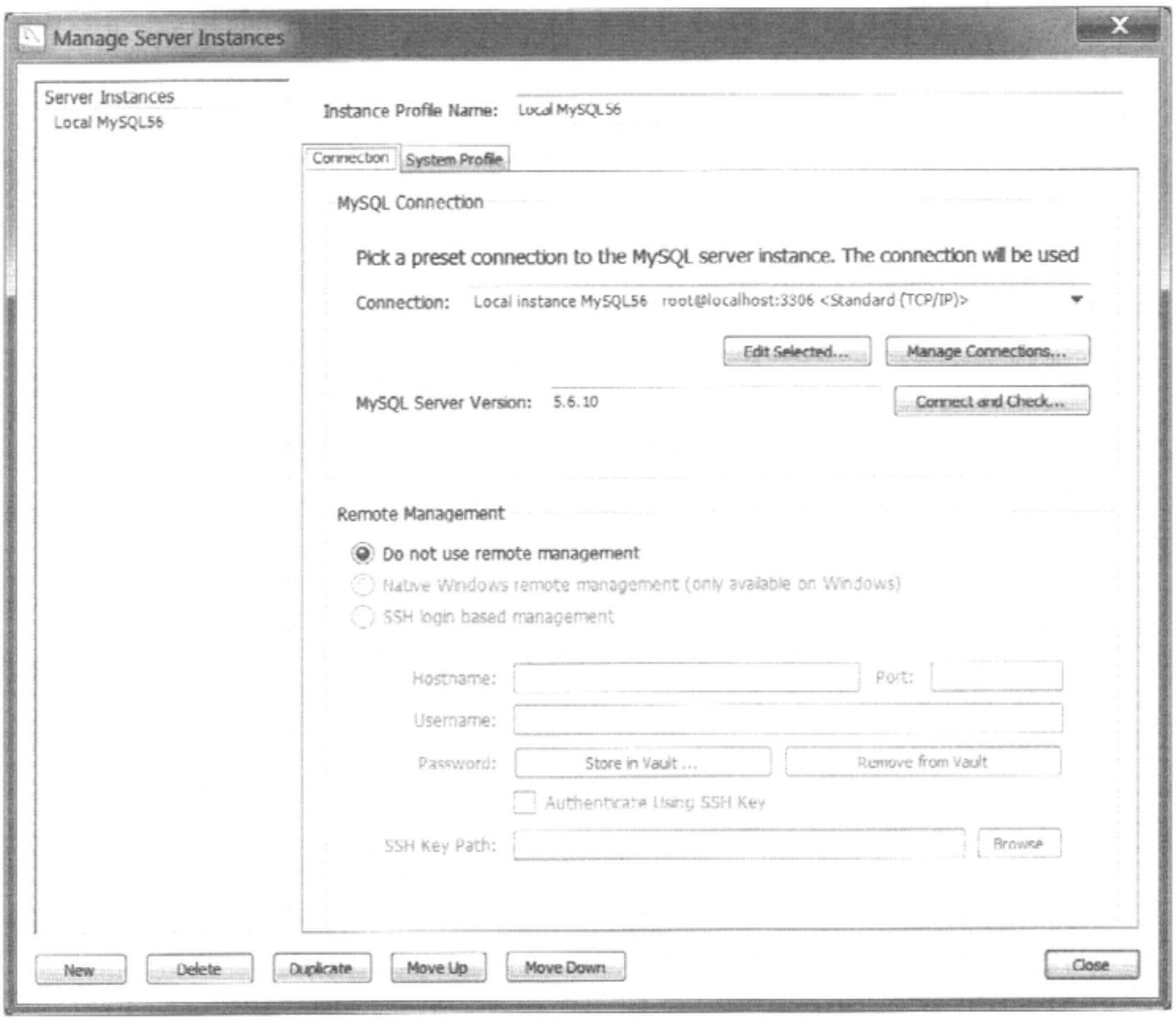
6.9.1. MySQL Workbench

Konfiguracja serwera i zarządzanie serwerem w programie MySQL Workbench są realizowane w panelach *SQL Development* i *Server Administration*.

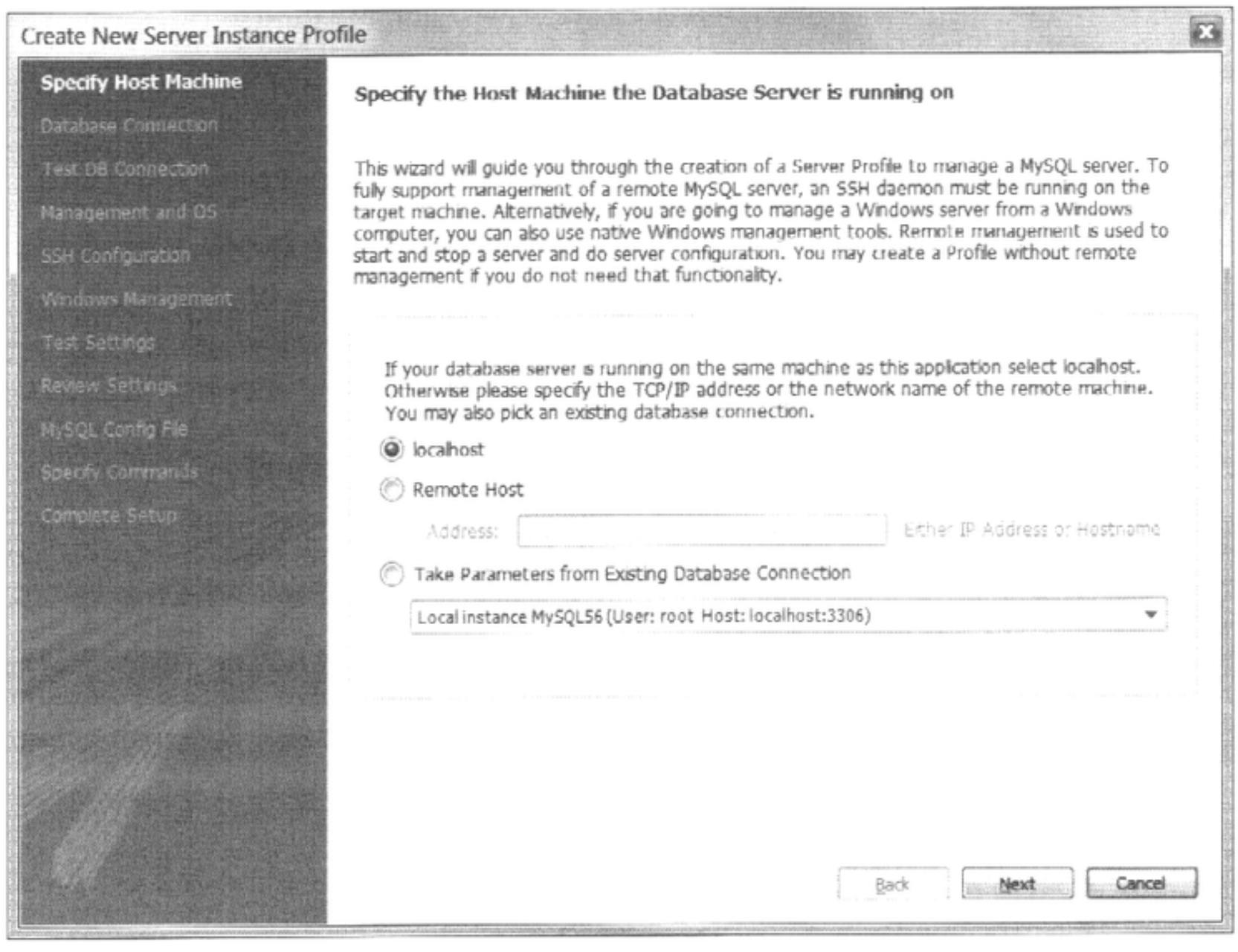
Server Administration

W panelu *Server Administration* po wybraniu opcji *Manage Server Instances* można zmieniać konfiguracje istniejących instancji serwera (rysunek 6.36). Natomiast wybranie opcji *New Server Instance* spowoduje uruchomienie kreatora tworzenia nowej instancji (rysunek 6.37).

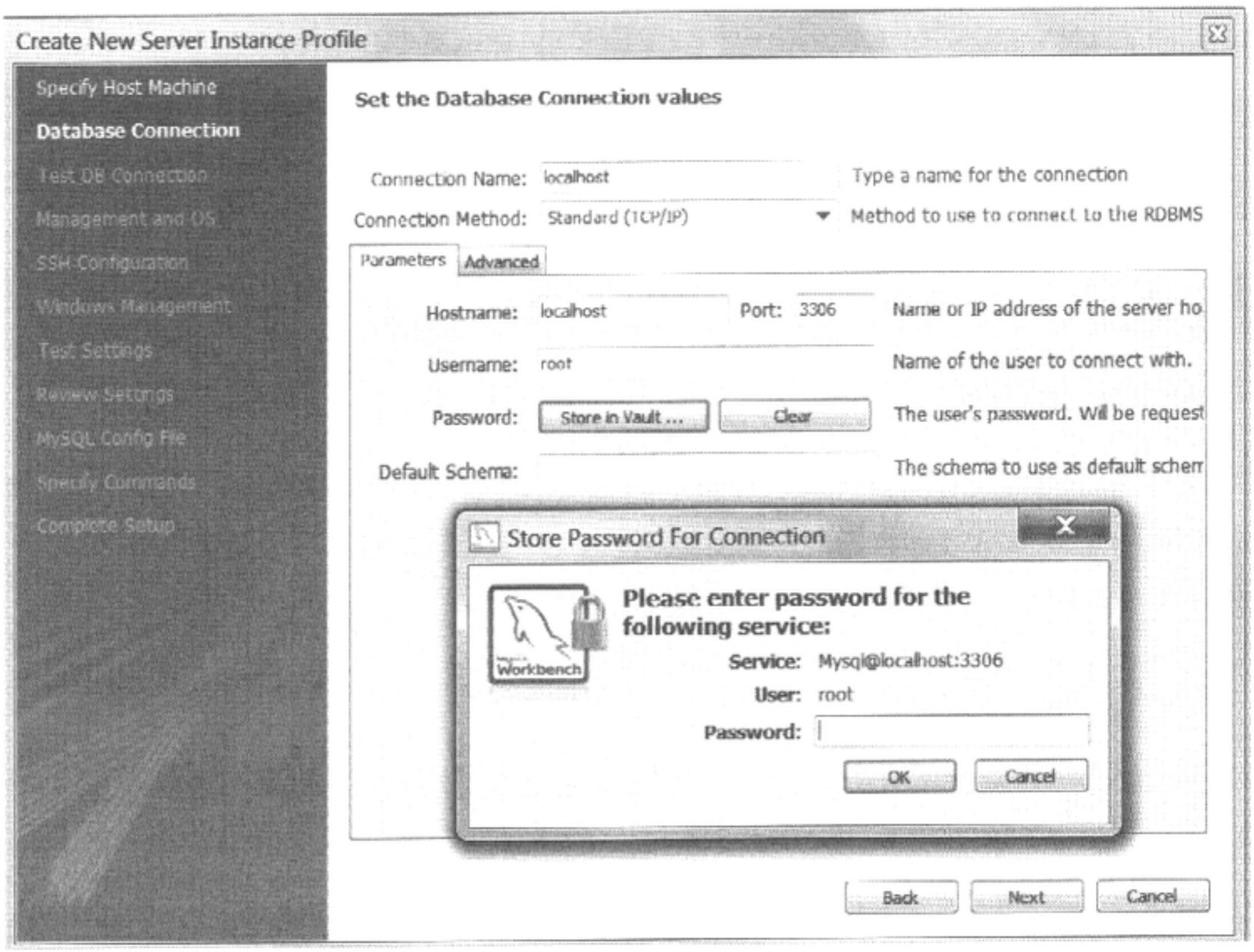
Rysunek 6.36. Okno konfigurowania instancji serwera



Rysunek 6.37. Kreator tworzenia nowej instancji

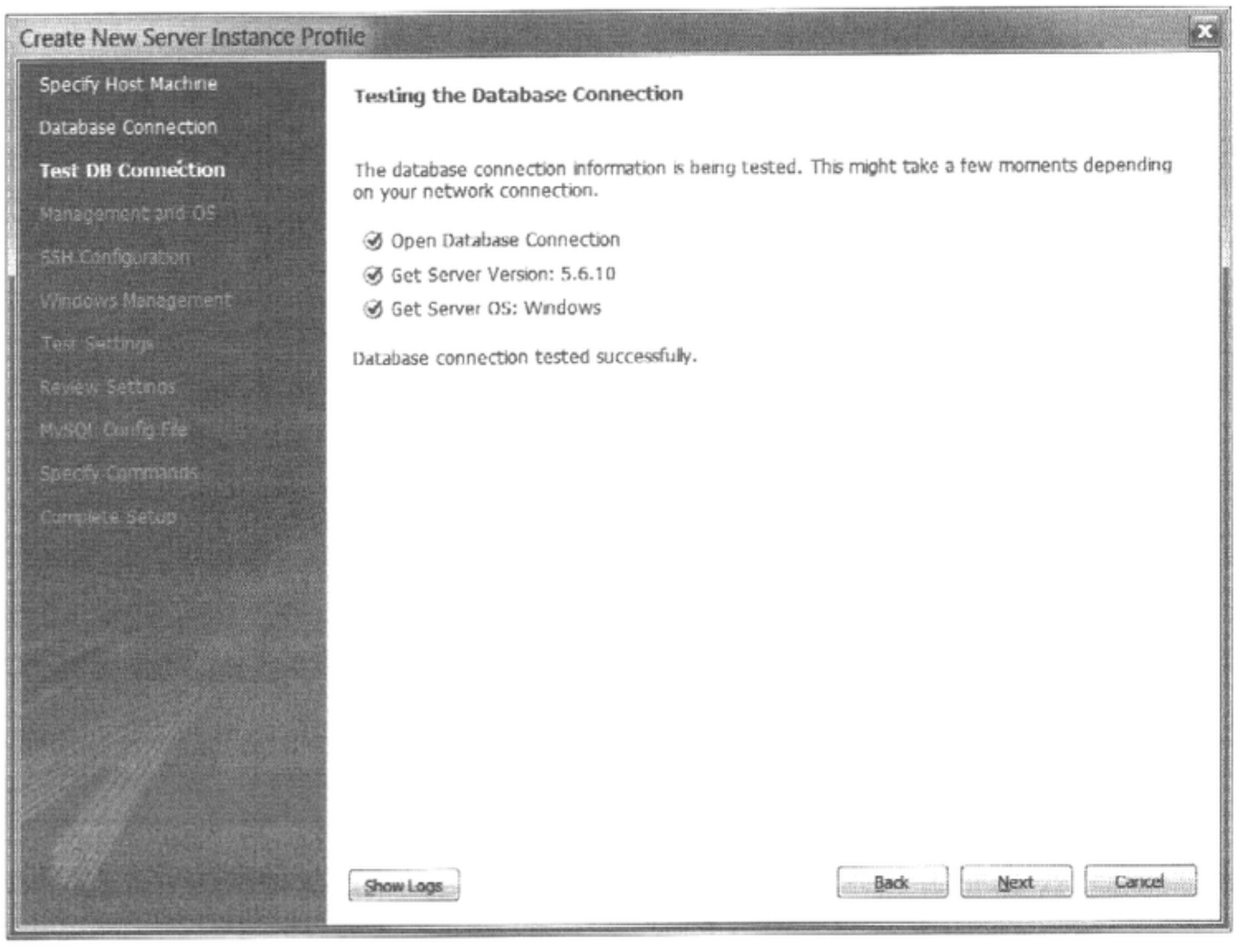


Dla tworzonej instancji należy skonfigurować nowe połączenie lub wybrać już istniejące. Po jego ustawieniu wyświetlą się parametry skonfigurowanego połączenia (rysunek 6.38). Jeżeli serwer ma ustawione hasło do konta głównego, można je wprowadzić po kliknięciu przycisku *Store in Vault*. Umożliwi to łączenie się z serwerem bez konieczności wprowadzania hasła przy każdym uruchamianiu instancji.



Rysunek 6.38. Parametry połączenia zdefiniowanego dla nowej instancji

Po przetestowaniu połączenia (rysunek 6.39) i sprawdzeniu, czy możliwy jest dostęp do pliku konfiguracyjnego i do poleceń *Start* i *Stop*, nastąpi utworzenie nowej instancji, a jej nazwa pojawi się w głównym oknie programu MySQL Workbench.



Rysunek 6.39. Test połączenia dla nowej instancji

Zadanie 6.5

Utwórz nową instancję na serwerze MySQL. Skonfiguruj dla niej połączenie.

Zadanie 6.6

Wykorzystując program MySQL Workbench, utwórz zaprojektowaną wcześniej bazę danych *Prawo jazdy*. Baza danych powinna zawierać dane osób zarejestrowanych, listę dostępnych kursów, przydział uczestników do różnych kursów, rejestrację jazd dla każdego uczestnika kursu. W trakcie tworzenia bazy danych określ jej rozmiar, nazwę logiczną oraz położenie pliku z danymi. Wprowadź przykładowe dane.

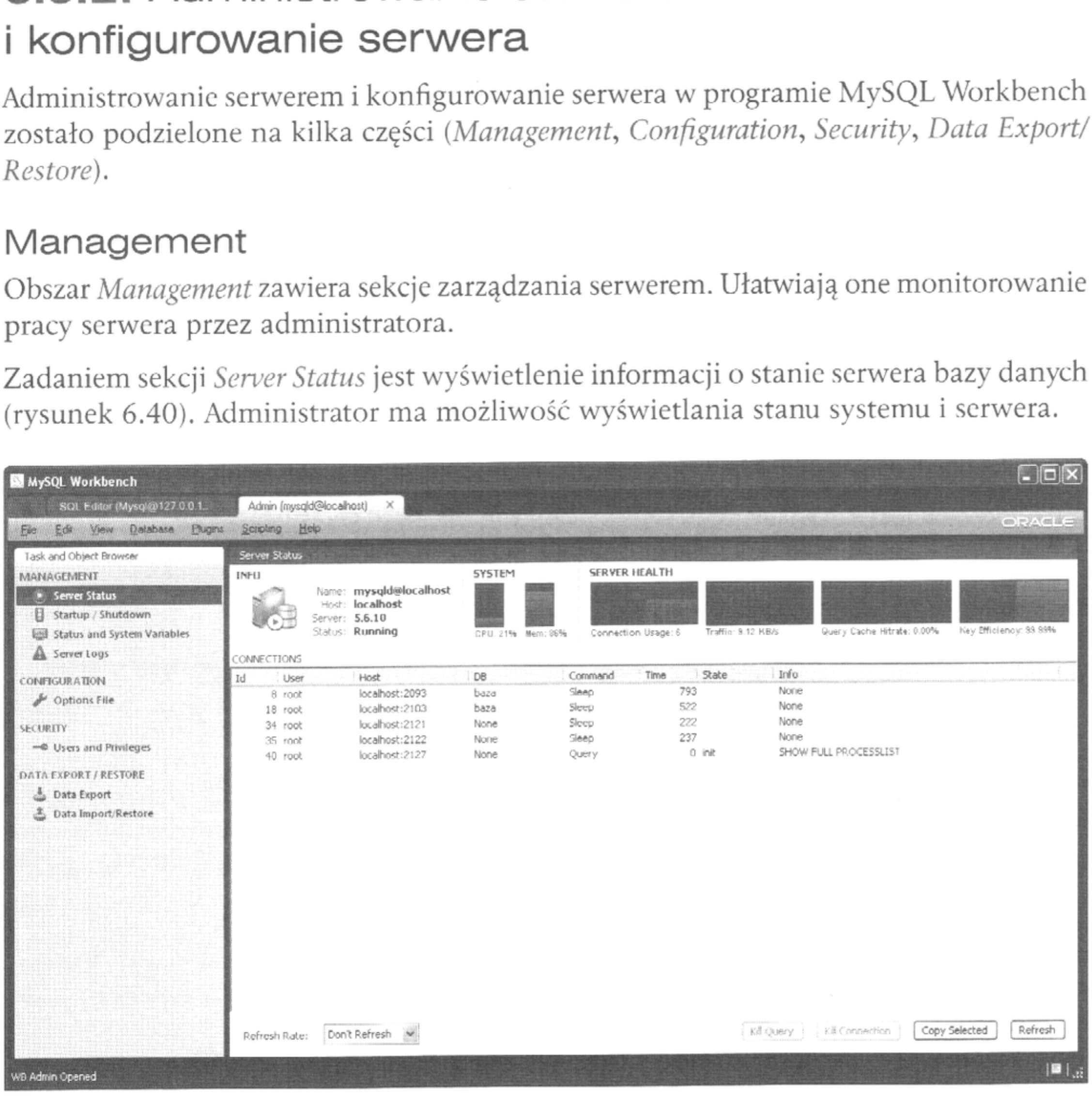
6.9.2. Administrowanie serwerem i konfigurowanie serwera

Administrowanie serwerem i konfigurowanie serwera w programie MySQL Workbench zostało podzielone na kilka części (*Management*, *Configuration*, *Security*, *Data Export/Restore*).

Management

Obszar *Management* zawiera sekcje zarządzania serwerem. Ułatwiają one monitorowanie pracy serwera przez administratora.

Zadaniem sekcji *Server Status* jest wyświetlenie informacji o stanie serwera bazy danych (rysunek 6.40). Administrator ma możliwość wyświetlania stanu systemu i serwera.

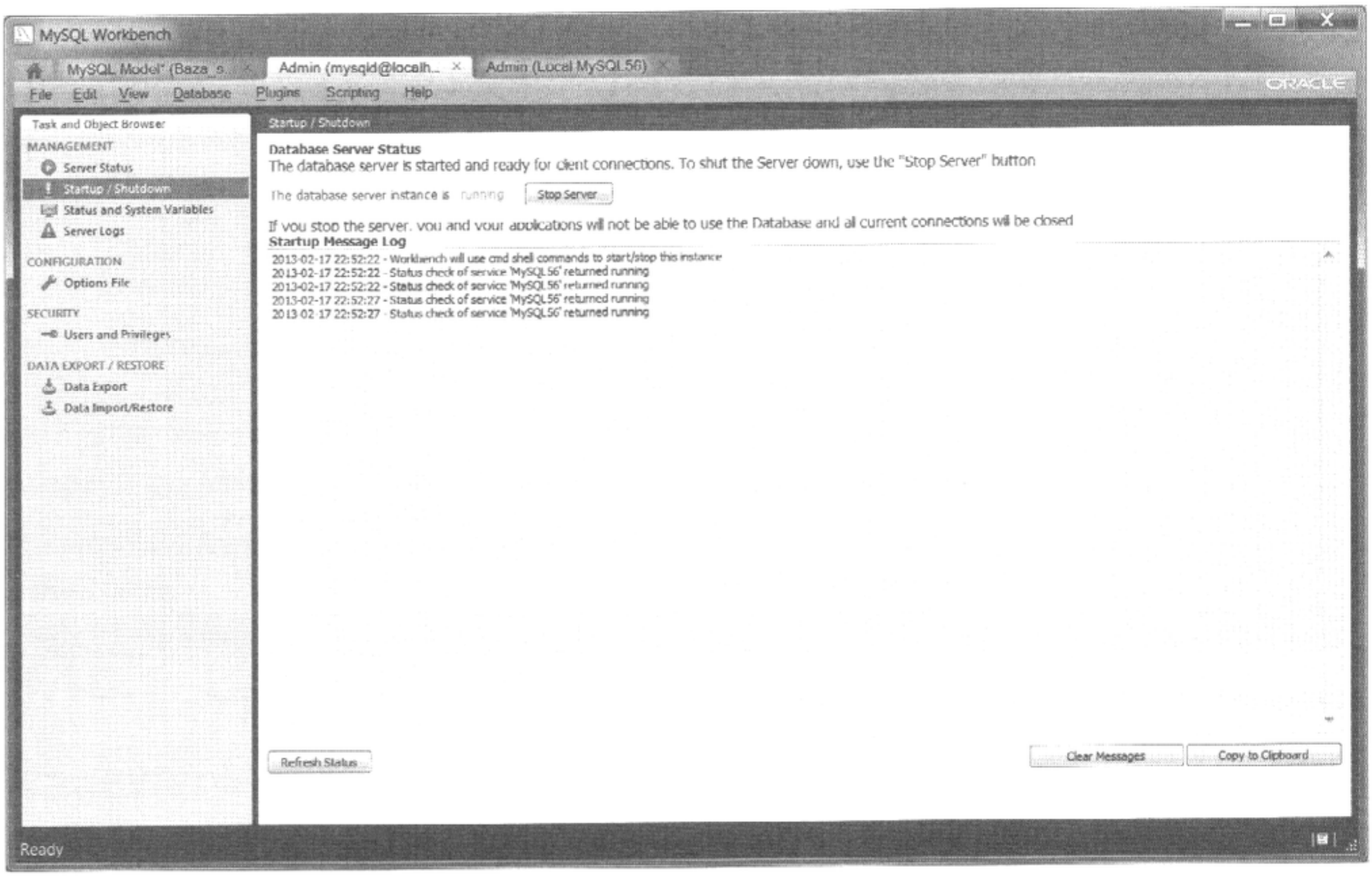


Rysunek 6.40. Funkcje administrowania serwerem w programie MySQL Workbench

Stan systemu to: wykorzystanie procesora, użycie pamięci, stan połączeń.

Stan serwera to: dostępne połączenia, przepływ informacji, wydajność.

Sekcja *Startup/Shutdown* umożliwia uruchomienia i zatrzymania serwera (rysunek 6.41).



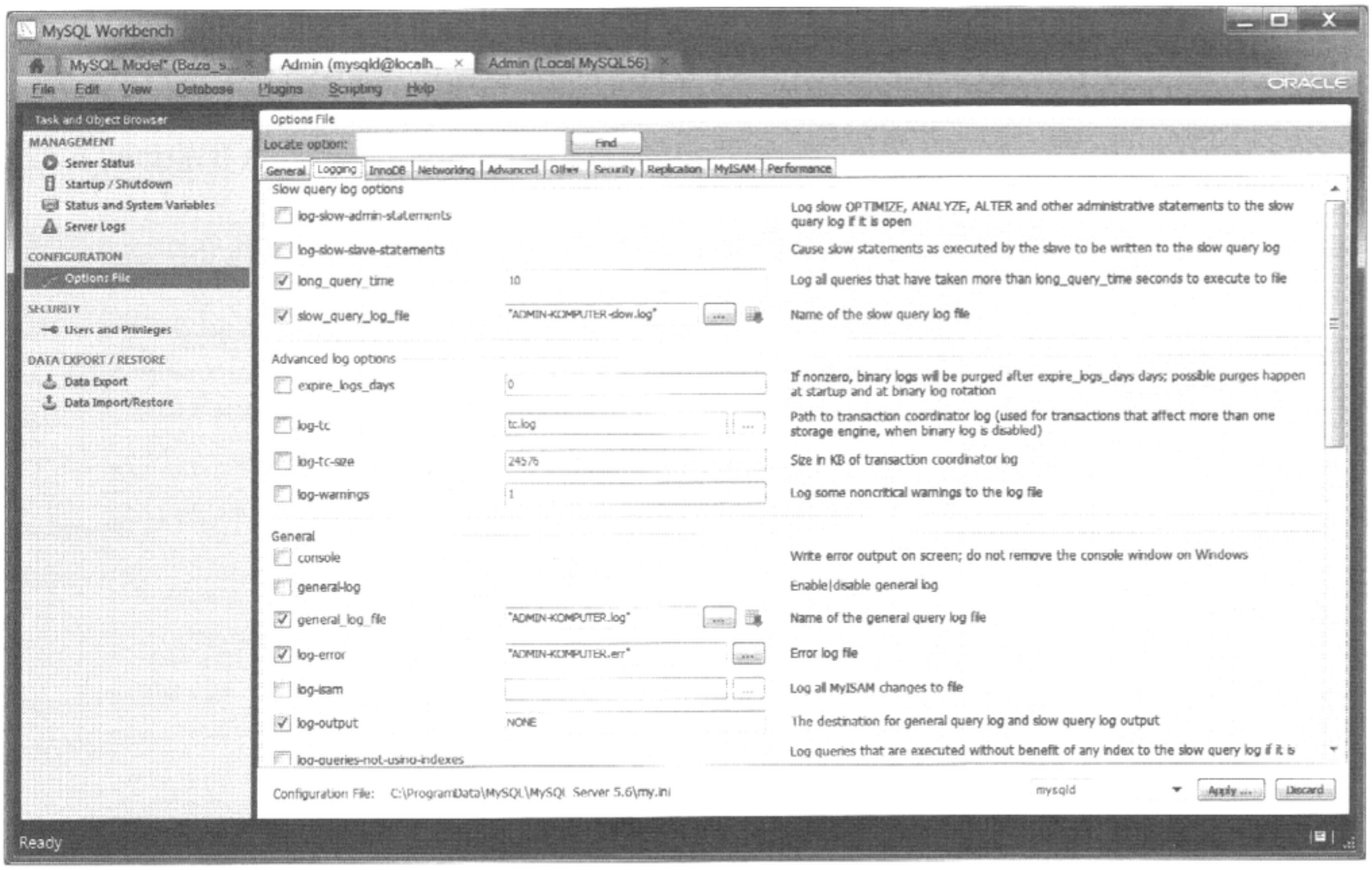
Rysunek 6.41. Opcja Startup/Shutdown

Po wybraniu sekcji *Server Logs* w zakładce *Error Log File* znajdują się informacje o uruchamianiu i zatrzymaniu głównej usługi serwera. Wyświetlana jest zawartość plików dziennika serwera (dzienniki błędów, dzienniki binarne, dzienniki `InnoDB`).

W sekcji *Status and System Variables* zostały zapisane informacje o stanie zmiennych systemowych.

Configuration

Sekcja *Configuration* daje możliwość edytowania pliku konfiguracyjnego serwera MySQL (*my.ini* lub *my.cnf*). Zawiera wiele różnych zakładek, dzięki którym można konfigurować ustawienia serwera (rysunek 6.42). Umożliwiają one monitorowanie pracy serwera i reagowanie przez administratora na pojawiające się problemy. Znajdują się tu takie zakładki, jak: *Performance* (wydajność), *Log Files* (pliki z logami), *Replication* (replikacja), *Security* (zabezpieczenia).



Rysunek 6.42. Opcje konfigurowania serwera

6.10. Prawa dostępu do serwera

Podczas pracy z serwerem MySQL należy przestrzegać zasad bezpieczeństwa. Trzeba pamiętać o ustawieniu hasła dla konta administratora oraz konta anonimowego (`User=""`). Jeżeli po wpisaniu polecenia `mysql -u root` nie pojawi się komunikat z prośbą o podanie hasła, to znaczy, że hasło nie zostało ustawione. Nie należy pracować na koncie *root*. Nie należy definiować dostępu do bazy *mysql* innym użytkownikom poza administratorem. Użytkownikom należy nadawać jak najmniejsze uprawnienia — tylko te, które są konieczne. Nie należy nadawać praw dostępu do wszystkich baz danych, a jedynie do wybranych.

Prawa dostępu serwera MySQL mogą być nadawane użytkownikom na poziomie:

- całego serwera,
- bazy danych (na przykład: `CREATE`, `ALTER`, `DROP`),
- obiektów bazy danych (na przykład: `SELECT`, `INSERT`, `UPDATE`, `DELETE`).

Informacje na temat praw dostępu do serwera MySQL są przechowywane w tabelach słownikowych bazy danych *mysql*. Są to tabele: *host*, *db*, *user*, *tables_priv*, *columns_priv* i *procs_priv*.

- *user* przechowuje informacje o prawach dostępu użytkownika niezależnie od bazy danych.
- *db* przechowuje informacje o prawach dostępu użytkownika w zależności od bazy danych.

- *host* przechowuje informacje w kontekście komputera, z którego łączy się użytkownik.

Pozostałe tabele przechowują informacje szczegółowe dotyczące praw dostępu do tabel, kolumn itp.

Po zalogowaniu się do bazy *mysql* można wyświetlić listę dostępnych tabel.

Przykład 6.23

```
USE mysql;
SHOW TABLES;
SELECT Host, User FROM user WHERE -"adam";
```

Użyte w przykładzie polecenie `SELECT` pozwoli wyświetlić prawa dostępu użytkownika *adam*.

W serwerze MySQL prawa są dziedziczone. Oznacza to, że prawa nadane do obiektu znajdującego się wyżej w hierarchii są domyślnie dziedziczone przez obiekty znajdujące się niżej. Jednak prawa mogą być nadawane i odbierane na dowolnym poziomie.

6.10.1. Uprawnienia

Nadawanie praw

Prawa są nadawane użytkownikom instrukcją `GRANT`. Składnia polecenia ma postać:

```
GRANT [prawo] ON baza_danych.* TO '[użytkownik]'&'[host]' IDENTIFIED BY
'[hasło]';
```

Prawa mogą być nadawane na różnych poziomach.

Prawa nadawane globalnie:

```
GRANT UPDATE ON *.* TO Marcin;
```

Prawa nadawane na poziomie domyślnej bazy danych:

```
GRANT UPDATE ON ksiegarnia_internetowa.* TO Marcin;
```

lub:

```
GRANT UPDATE ON * TO Marcin;
```

Prawa nadawane na poziomie obiektów bazy danych (na przykład do wybranej tabeli):

```
GRANT INSERT ON ksiegarnia_internetowa.ksiazki TO Marcin;
```

Prawa nadawane na poziomie wybranych kolumn tabeli:

```
GRANT UPDATE(ksiazki.tytul),INSERT ON ksiegarnia_internetowa.ksiazki TO
Marcin;
```

Do przyznania wszystkich przywilejów służy w poleceniu `GRANT` opcja `ALL PRIVILEGES` lub `ALL`.

Przykład 6.24

```
GRANT ALL PRIVILEGES on *.* TO Marcin;
```

Polecenie `GRANT` może zawierać dodatkowe klauzule:

- `MAX_QUERIES_PER_HOUR` — ogranicza liczbę zapytań;
- `MAX_UPDATES_PER_HOUR` — ogranicza liczbę zmian wprowadzanych do bazy;
- `MAX_CONNECTIONS_PER_HOUR` — ogranicza liczbę logowań użytkownika w ciągu godziny;
- `MAX_USER_CONNECTIONS` — ogranicza liczbę jednoczesnych połączeń uzyskiwanych z jednego konta.

Przykład 6.25

```
GRANT USAGE ON *.* TO Marcin WITH MAX_QUERIES_PER_HOUR 1;
```

Prawo `USAGE` oznacza, że użytkownikowi nie zostały nadane żadne prawa.

Odbieranie praw

Instrukcją `REVOKE` można usunąć wcześniej nadane prawa.

Przykład 6.26

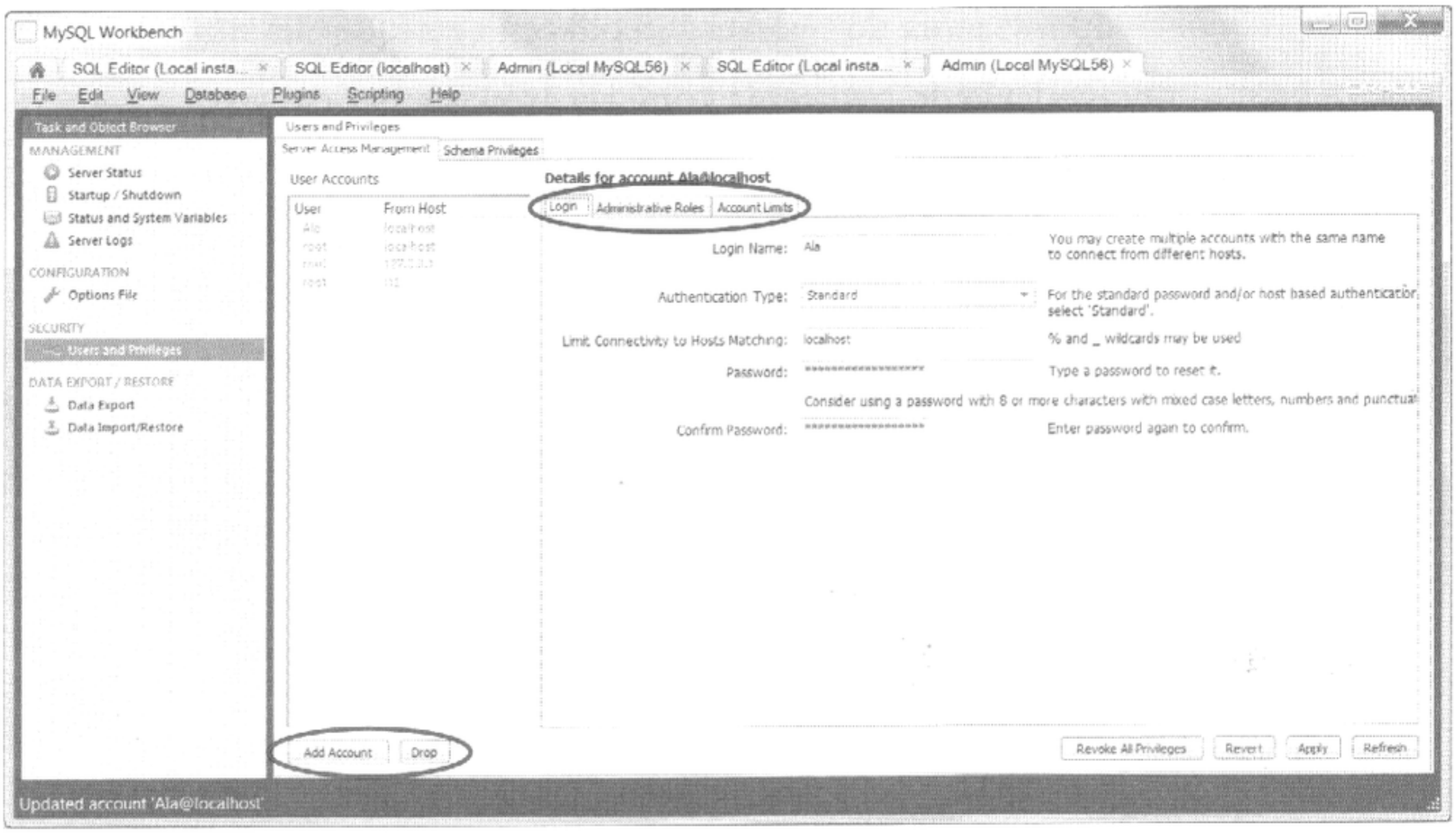
```
REVOKE UPDATE ON ksiegarnia_internetowa.Ksiazki FROM Marcin;
```

W podanym przykładzie użytkownikowi *Marcin* zostały odebrane uprawnienia do modyfikowania danych w tabeli *Ksiazki*.

Lista uprawnień

- `ALL` — użytkownik otrzymuje wszystkie prawa poza `GRANT OPTION`.
- `CREATE` — użytkownik może tworzyć tabele.
- `SELECT` — użytkownik może wyświetlać zawartość tabel.
- `INSERT` — użytkownik może dodawać nowe dane do tabel.
- `SHOW DATABASES` — użytkownik może przeglądać listy dostępnych baz danych.
- `USAGE` — umożliwia tworzenie użytkownika bez uprawnień.
- `GRANT OPTION` — użytkownik może nadawać uprawnienia innym użytkownikom.

W programie MySQL Workbench zarządzanie użytkownikami i ich prawami jest realizowane w panelu *Server Administration* w opcji *Manage Security*. W otwartym oknie dostępne są dwie zakładki: *Server Access Management* oraz *Schema Privileges* (rysunek 6.43).



Rysunek 6.43. Okno nadawania uprawnień dla użytkownika

Zakładka *Server Access Management* służy do tworzenia, modyfikowania i usuwania kont użytkownika. Nowi użytkownicy są tworzeni po wybraniu w dolnej części okna ikony *Add Account*. Usunięcie użytkownika następuje po wybraniu ikony *Drop*. Ustawienia konfiguracyjne użytkownika są modyfikowane w zakładce *Login*. Role są przydzielane w zakładce *Administrative Roles*, a ograniczenia są ustawiane w zakładce *Account Limits*.

Aby ułatwić przypisywanie uprawnień użytkownikom, w MySQL została wprowadzona koncepcja ról administracyjnych. Za pomocą ról można w szybki sposób przyznawać użytkownikowi zestaw uprawnień potrzebnych do pracy na serwerze. Użytkownikowi można nadać jedną rolę lub kilka ról.

Dostępne role

DBA — wszystkie uprawnienia.

MaintenanceAdmin — uprawnienia do utrzymania serwera.

ProcessAdmin — uprawnienia do monitorowania i zatrzymywania procesów użytkownika.

UserAdmin — uprawnienia do tworzenia użytkowników i ustawiania haseł.

SecurityAdmin — uprawnienia do zarządzania kontami oraz nadawania i odbierania uprawnień serwera.

MonitorAdmin — uprawnienia do monitorowania serwera.

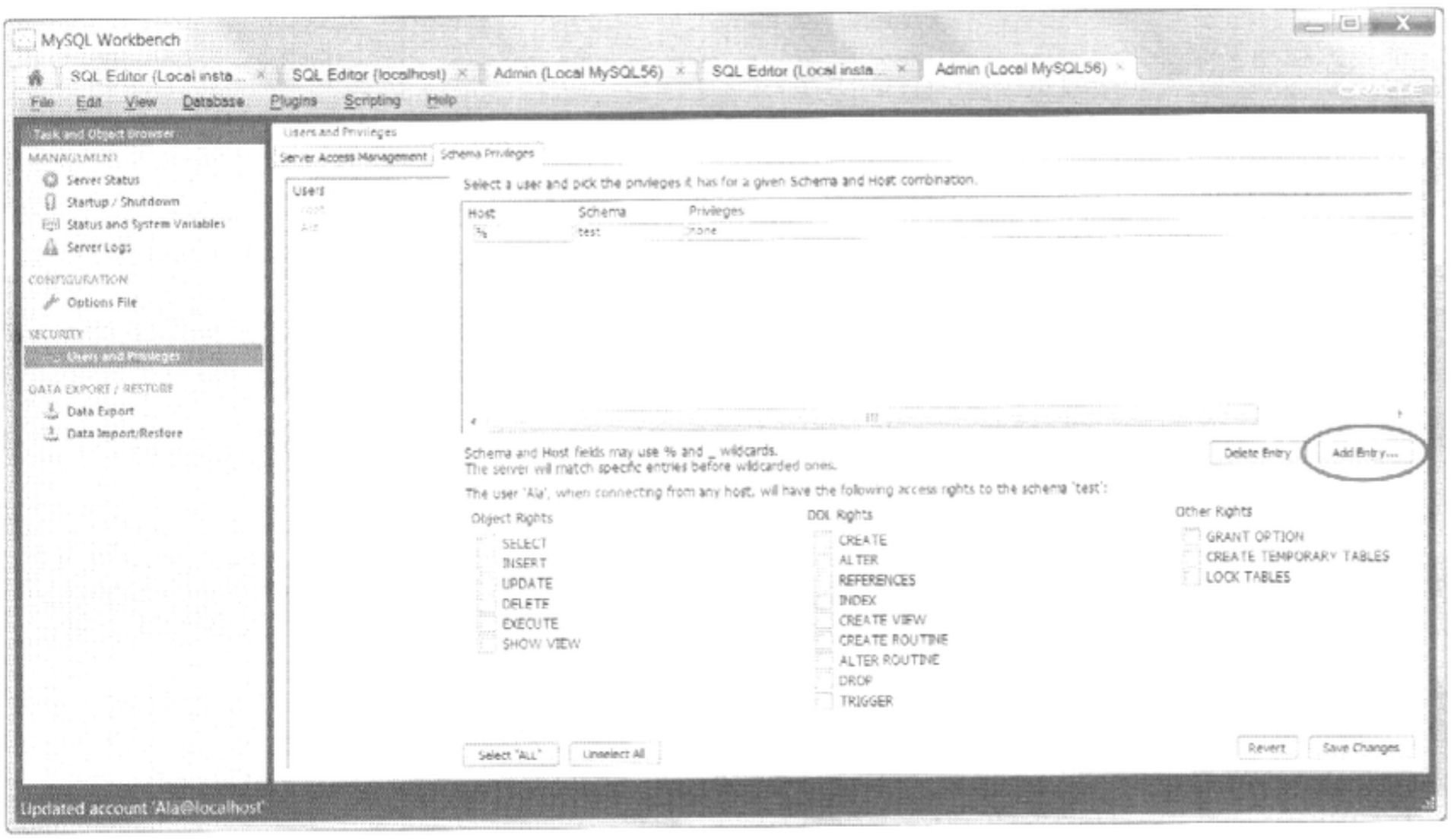
DBManager — uprawnienia do zarządzania bazami danych.

DBDesigner — uprawnienia do tworzenia i modyfikowania schematów baz danych.

ReplicationAdmin — uprawnienia do tworzenia replikacji i zarządzania nią.

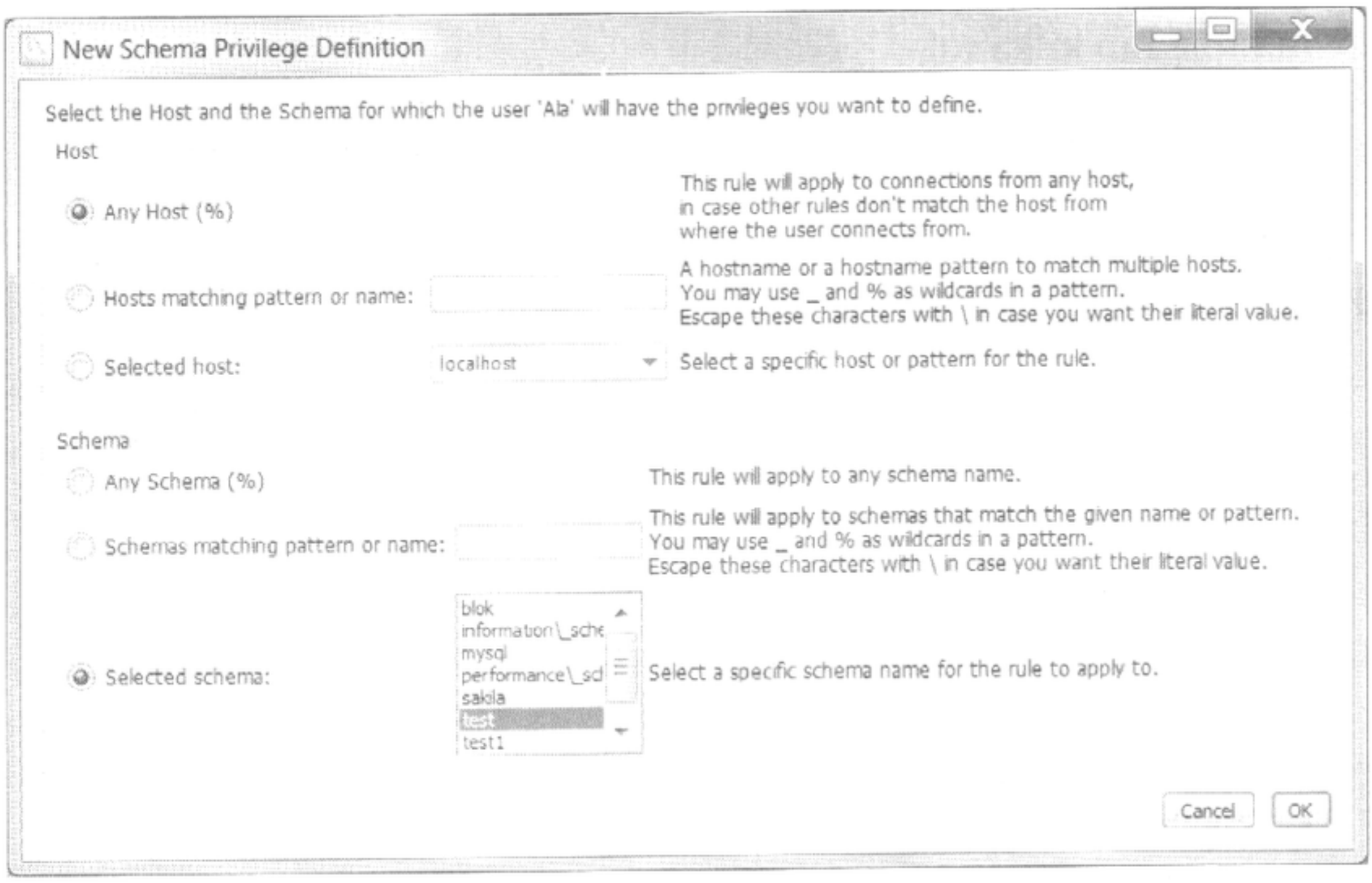
BackupAdmin — uprawnienia do tworzenia kopii zapasowych baz danych.

W zakładce *Schema Privileges* można nadawać i odbierać użytkownikowi określone prawa do wybranych baz danych (rysunek 6.44).



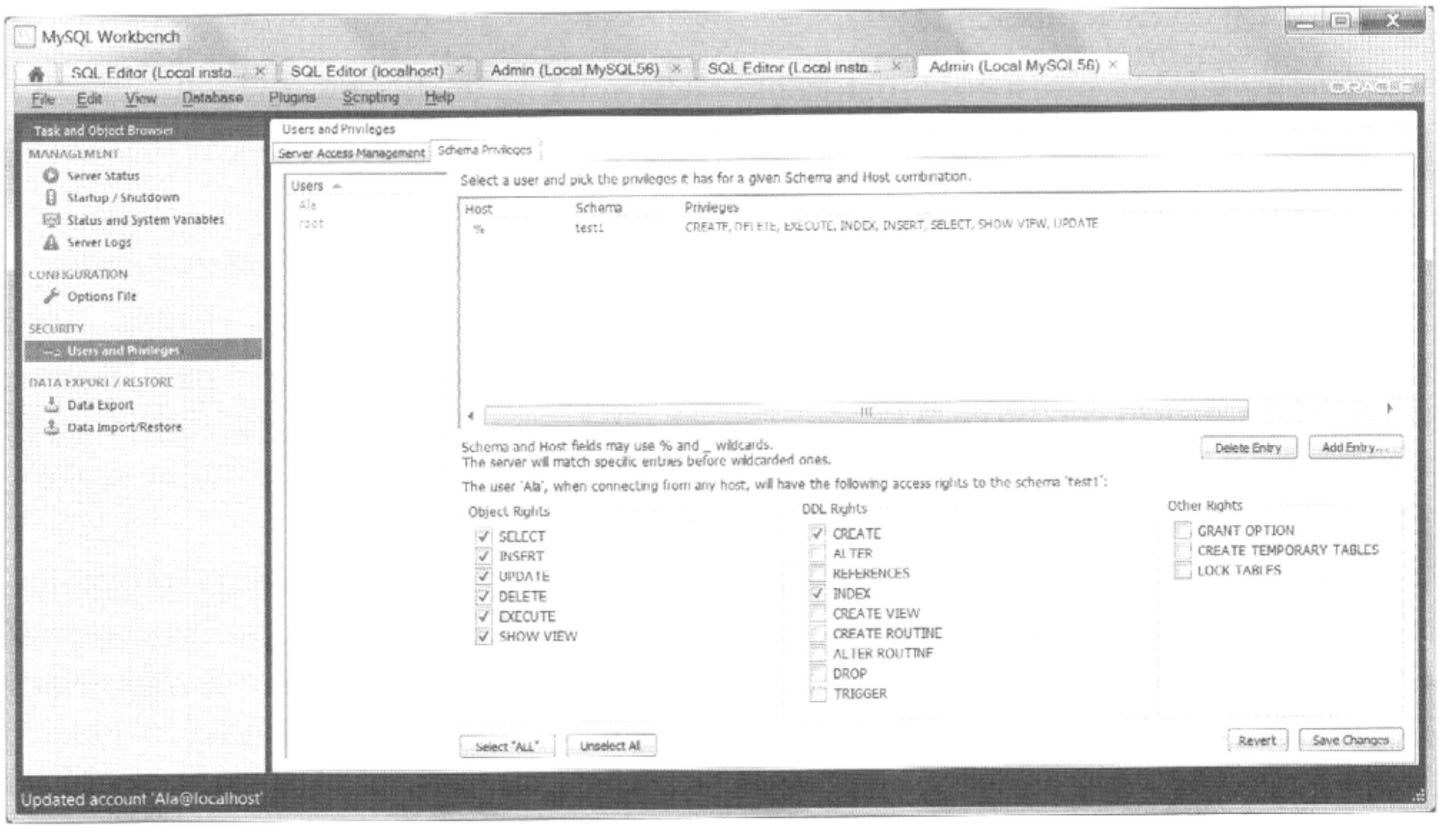
Rysunek 6.44. Okno odbierania praw do bazy danych

Po wybraniu użytkownika należy kliknąć opcję *Add Entry...*. Zostanie otwarte okno *New Schema Privilege Definition*, w którym można określić, jakiego hosta (obszar *Host*) i jakiej bazy danych (obszar *Schema*) będą dotyczyły prawa przyznawane użytkownikowi (rysunek 6.45).



Rysunek 6.45. Okno wyboru hosta i bazy danych

Po wybraniu bazy danych i kliknięciu przycisku *OK* w zakładce *Schema Privileges* można nadawać i odbierać użytkownikowi prawa do tej bazy danych (rysunek 6.46). Po określeniu praw do bazy danych należy zatwierdzić je przyciskiem *Save Changes*.



Rysunek 6.46. Nadawanie praw do bazy danych

Zadanie 6.7

Utworzonym wcześniej użytkownikom przypisz prawa do bazy danych *Prawo jazdy*. *Operator* powinien mieć pełne prawa do bazy. *Instruktor* powinien mieć prawo do przeglądania zawartości wszystkich tabel. Dodatkowo powinien mieć prawo dodawania nowych danych do tabeli rejestrującej jazdy każdego uczestnika kursu. *Gracz* powinien mieć prawo przeglądania tabeli rejestrującej jazdy każdego uczestnika kursu. Sprawdź prawa dostępu różnych użytkowników do bazy danych.

6.11. Replikacja bazy danych

Replikacja na serwerze MySQL działa na podstawie schematu *master-slave*. Serwer *master* wszelkie zmiany wprowadzane w bazie danych (aktualizację, usuwanie danych) zapisuje w plikach logowania binarnego (*binary logs*). Dlatego, aby wykonać replikację bazy danych, należy włączyć rejestrowanie binarne na serwerze *master*. Serwer (lub serwery) *slave* pobiera dane i zapisuje je w replice. Każda zmiana danych na serwerze głównym (*master*) powoduje identyczną zmianę danych na serwerze zapasowym (*slave*). Ten typ replikacji jest replikacją jednokierunkową. Po podłączeniu do serwera *master* serwer *slave* przesyła informacje o stanie uaktualnienia wpisów z dziennika replikacji. Jeżeli z jakiegoś powodu połączenie między serwerami *master* i *slave* zostanie przerwane, serwer *slave* będzie próbował okresowo łączyć się z serwerem *master*, aż połączenie

ponownie zostanie nawiązane. Po uzyskaniu połączenia serwer *slave* zaktualizuje swoją bazę danych. Serwer *slave* może być skonfigurowany jako serwer *master* dla innych serwerów MySQL.

Podstawowe cechy replikacji to:

- *skalowalność* — możliwość rozłożenia obciążenia na wiele serwerów;
- *bezpieczeństwo* — tworzona kopia istniejącej bazy danych może pomóc w przypadku awarii serwera głównego;
- *separacja* — możliwość udostępnienia kopii bazy danych wielu użytkownikom.

Zasada działania replikacji polega na zapisywaniu do dziennika przez serwer *master* każdej czynności, którą wykonał. Wykorzystuje się do tego *bin-logi*. Są to pliki binarne zawierające instrukcje, które wykonał serwer *master*. Serwer *slave* odczytuje te pliki i kolejno wykonuje zapisane w nich instrukcje. Efektem takiego działania są dwie identyczne bazy danych.

Po skonfigurowaniu mechanizmu replikacji na serwerze *master* pojawia się dodatkowy wątek dla każdego serwera *slave*, który odpowiada za wysyłanie *bin-logów* do serwerów *slave*. Każdy serwer *slave* tworzy dwa wątki. Pierwszy (wątek *I/O Thread*) odpowiada za odbieranie dziennika replikacji z serwera *master* i zapisanie go na dysku w plikach tymczasowych (*relay-log*). Drugi (wątek *SQL Thread*) zajmuje się parsowaniem plików tymczasowych i wykonaniem zapytań do bazy.

Bin-log

Bin-logi przechowują informacje o wszystkich zdarzeniach, które prowadziły do zmian w bazie danych. Są wykorzystywane do:

- tworzenia replikacji,
- odzyskiwania danych.

UWAGA

Bin-logi zastąpiły dzienniki aktualizacji dostępne we wcześniejszych wersjach MySQL.

Włączenie rejestrowania binarnego na serwerze zmniejsza jego wydajność, jednak korzyści płynące ze skonfigurowania replikacji są na tyle istotne, że przewyższają spadek wydajności serwera.

6.11.1. Rodzaje replikacji

W serwerze MySQL występują trzy rodzaje replikacji:

- **SBR** (*statement-based replication*) — serwer zapisuje do *bin-logów* tylko te zapytania, które wykonał. Jest to najszybsza metoda replikacji. Jednak w przypadku bardziej

skomplikowanych zapytań taki sposób zapisu może prowadzić do problemów z tworzeniem replikacji.

- **RBR** (*row-based replication*) — serwer zapisuje do *bin-logów* wyniki działań zapytań na serwerze *master*. Jest to metoda bezpieczniejsza od metody SBR, ale znacznie wolniejsza. Powoduje też zwiększenie ilości przesyłanych danych między serwerami biorącymi udział w replikacji.
- **MFL** (*mixed-format logging*) — jest to połączenie dwóch poprzednich metod. W tej metodzie w większości przypadków zapytania są zapisywane tak jak w metodzie SBR, natomiast dla bardziej skomplikowanych zapytań, których wyniku nie można przewidzieć, uruchamiana jest metoda RBR.

6.11.2. Konfigurowanie replikacji

Konfigurowanie replikacji składa się z następujących etapów:

1. Uaktywnienie dziennika transakcji na serwerze *master*.

W tym celu należy wprowadzić zmiany w pliku konfiguracyjnym serwera. Najczęściej jest to plik *my.cnf* lub *my.ini*. Powinien on wyglądać tak:

```
[mysqld]
log-bin = mysql-bin
binlog_format = mixed
max_binlog_size =50M
server-id =1
```

`log-bin = mysql-bin` — zostanie uruchomiony mechanizm zapisywania zmian w bazie danych. Utworzony zostanie plik dziennika o nazwie `<nazwa komputera>-bin`.

`binlog_format = mixed` — zostanie ustawiony format zapisu danych do *bin-logów*. W podanym przykładzie będzie to format mieszany.

`max_binlog_size =50M` — zostanie ustawiony maksymalny rozmiar dziennika logów.

`server-id =1` — zostanie ustalony unikatowy numer serwera w obrębie replikacji. Musi on mieć unikalną wartość na każdym z serwerów i musi być różny od 0.

Domyślnie replikowane są wszystkie bazy danych i wszystkie tabele dostępne na serwerze. Jednak poprzez ustawienie odpowiednich opcji można replikować wybraną bazę lub tabelę. Dodanie w pliku konfiguracyjnym opcji `binlog-do-db=database` spowoduje replikację tylko wybranych baz danych.

Przykład 6.27

```
binlog-do-db=ksiegarnia
```

Aby sprawdzić stan serwera *master*, należy go zrestartować (`mysqld restart`) i po zalogowaniu się poleceniem `mysql -u root -p` wprowadzić następujące polecenie:

```
SHOW MASTER STATUS\G
```

W wyniku powinien zostać wyświetlony zdefiniowany wcześniej stan serwera *master*.

II. Utworzenie konta replikacji na serwerze *master*.

Utworzone konto (z prawami do replikacji) zostanie wykorzystane przez serwer *slave* i będzie odpowiadało za jego autoryzację.

W celu utworzenia konta i nadania mu praw do replikacji należy wykonać polecenia:

```
CREATE USER 'nazwa_użytkownika'@'%' IDENTIFIED BY 'hasło';
GRANT REPLICATION SLAVE ON *.* TO 'nazwa_użytkownika'@'%' IDENTIFIED BY
'hasło';
FLUSH PRIVILEGES;
```

III. Utworzenie kopii baz danych serwera *master*.

W celu synchronizacji serwerów *slave* i *master* należy zablokować możliwość dodawania, usuwania i edytowania baz danych. W tym celu wykonujemy polecenie:

```
FLUSH TABLES WITH READ LOCK;
```

Następnie uruchamiamy na serwerze *master* polecenie:

```
SHOW MASTER STATUS;
```

Polecenie to wyświetli stan serwera (nazwę pliku dziennika, jego pozycję oraz nazwy baz, które będą replikowane). Należy zapamiętać wartości kolumn *File* oraz *Position*, które będą potrzebne podczas konfigurowania serwera *slave*.

Teraz można wykonać zrzut bazy danych, używając narzędzia `mysqldump` lub dowolnego innego. Polecenie programu `mysqldump` może mieć postać:

```
mysqldump --add-drop-table --master-data --quick -u uzytkownik -p
--all-databases > /tmp/kopia_db.sql
```

Po wykonaniu zrzutu można odblokować tabele poleceniem:

```
UNLOCK TABLES;
```

Po wykonaniu tych poleceń trzeba zrestartować serwer *master*.

IV. Skonfigurowanie serwera *slave*.

W celu skonfigurowania serwera *slave* należy, podobnie jak w przypadku serwera *master*, wprowadzić zmiany w pliku konfiguracyjnym *my.cnf* lub *my.ini*. Powinny one wyglądać tak:

```
[mysqld]
master-host=adres_ip_serwera
master-user='nazwa_użytkownika'
master-password=hasło
master-connect-retry=30
server-id=2
```

`server-id =2` — zostanie ustalony numer serwera *slave*. Jeżeli istnieje kilka takich serwerów, trzeba nadać im kolejne identyfikatory.

`master-connect-retry=30` — w przypadku braku połączenia próby połączenia są ponawiane co 30 sekund.

Po zmianie konfiguracji serwer należy zrestartować. Po jego ponownym uruchomieniu trzeba przenieść zrzut baz danych z serwera *master* na serwer *slave*. Można to zrobić w dowolny sposób, na przykład:

```
mysql -u uzytkownik -p -h nazwa_bazy < kopia_db.sql
```

Po wykonaniu tych poleceń należy zatrzymać serwer *slave* poleceniem:

```
SLAVE STOP;
```

V. Ustawienie parametrów replikacji.

Po zatrzymaniu serwera *slave* trzeba skonfigurować na nim dostęp do serwera *master*, ustawiając parametry replikacji:

```
CHANGE MASTER TO MASTER_HOST='adors_ip_serwera',
MASTER_USER='nazwa_użytkownika',
USER_PASSWORD='hasło_użytkownika',
MASTER_LOG_FILE='nazwa_pliku',
MASTER_LOG_POS='pozycja';
```

Po wykonaniu tych poleceń należy uruchomić serwer *slave* poleceniem:

```
SLAVE START;
```

W wyniku wykonanych operacji bazy MySQL z serwera *master* będą replikowane na serwerze *slave*.

VI. Weryfikacja.

Do sprawdzenia poprawności działania replikacji można użyć polecenia:

```
SHOW SLAVE STATUS;
```

W wyniku zostanie wyświetlona informacja o serwerze *master*, replikowanych bazach danych oraz pozycji w dzienniku.

Do wyświetlenia listy procesów na serwerach służy polecenie:

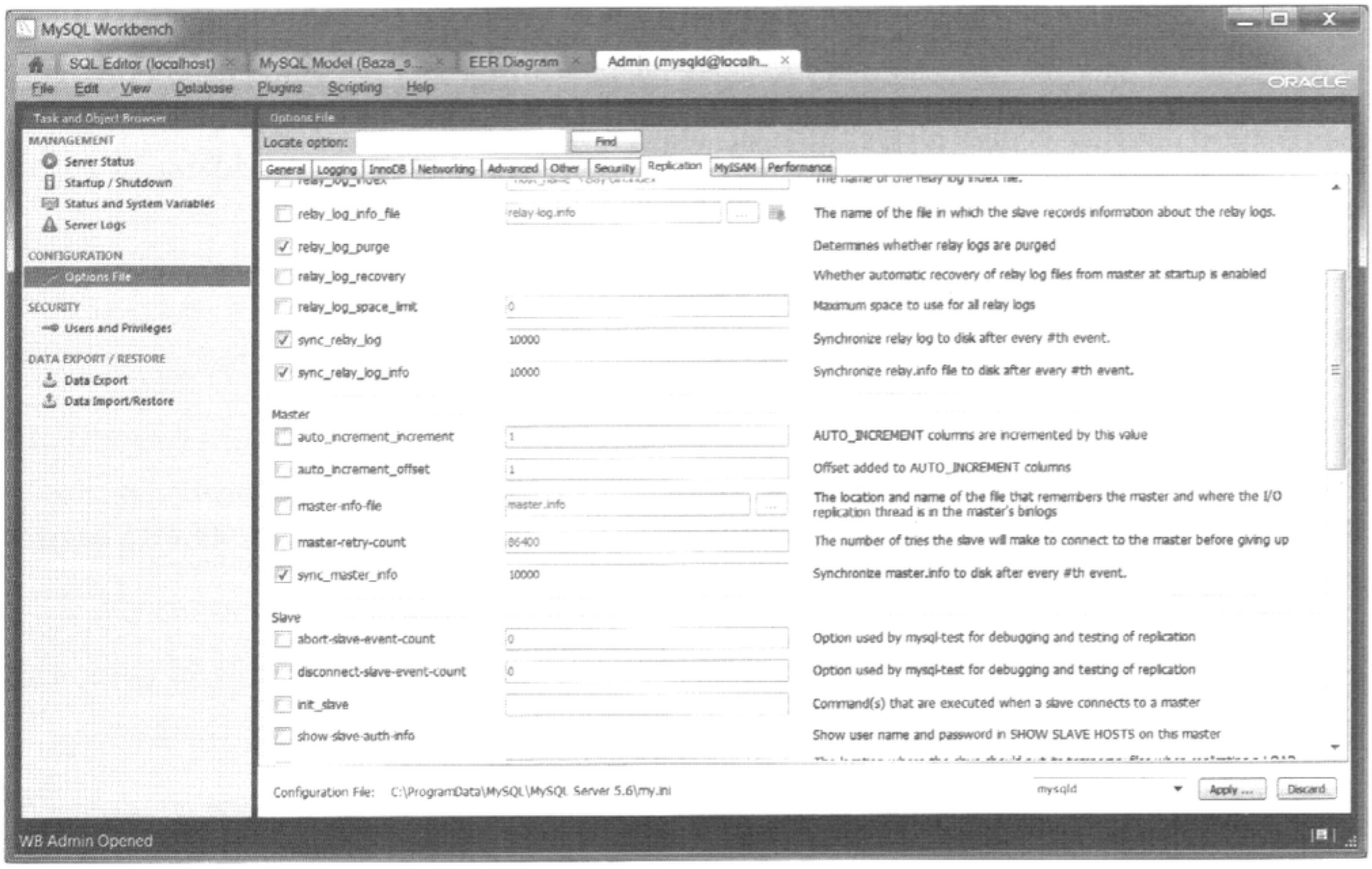
```
SHOW PROCESSLIST;
```

Działanie replikacji można sprawdzić również poleceniem:

```
SHOW STATUS LIKE 'Slave%';
```

W programie MySQL Workbench replikację można skonfigurować w panelu *Server Administration*. Po wybraniu bieżącej instancji w otwartym oknie na liście funkcji administrowania serwerem należy zaznaczyć *Configuration/Options File* i po wybraniu

zakładki *Replication* (rysunek 6.47) ustawić parametry konfiguracyjne tworzonej replikacji. Po kliknięciu przycisku *Apply* zostanie utworzona replikacja wybranych zasobów.



Rysunek 6.47. Okno konfigurowania replikacji

6.12. Kopie bezpieczeństwa

Jednym z podstawowych działań administratora jest zapewnienie bezpieczeństwa danych przez tworzenie ich kopii.

6.12.1. Sprawdzanie spójności bazy danych

Przed wykonaniem kopii bezpieczeństwa należy sprawdzić, czy baza danych nie zawiera błędów, i w miarę potrzeby wykonać jej naprawę. Do sprawdzania poprawności tabel `MyISAM`, `InnoDB` i `ARCHIVE` służy polecenie `CHECK TABLE` w postaci:

```
CHECK TABLE Tabela1, Tabela2;
```

Polecenie sprawdzania poprawności tabel może mieć następujące opcje:

- `QUICK` — sprawdzanie poprawności tabel bez sprawdzania, czy wiersze mają właściwe referencje;
- `FAST` — sprawdzanie dotyczy tylko tabel, które nie zostały poprawnie zamknięte;
- `CHANGED` — sprawdzanie dotyczy tylko tabel, które zmieniły się od ostatniej kontroli lub nie zostały poprawnie zamknięte;

- MEDIUM — sprawdzanie dotyczy poprawności wierszy i niektórych referencji;
- EXTENDED — sprawdzane są wszystkie referencje, używane, gdy potrzebna jest pewność co do spójności danych w tabeli. Podane opcje mogą być łączone w jednym poleceniu.

Przykład 6.28

```
CHECK TABLE Ksiazki FAST QUICK;

CHECK TABLE Autor MEDIUM;
```

6.12.2. Naprawa bazy danych

Do naprawienia uszkodzonej bazy danych z tabelami typu MyISAM lub ARCHIVE może zostać użyte polecenie REPAIR TABLE w postaci:

```
REPAIR TABLE Tabela1;
```

Do naprawy tylko pliku indeksu można użyć polecenia w postaci:

```
REPAIR TABLE Tabela1 QUICK;
```

6.12.3. Tworzenie pełnej kopii danych

Strategia tworzenia kopii bezpieczeństwa zaleca tworzenie w określonych momentach pełnych kopii baz danych oraz częste tworzenie różnicowych kopii bezpieczeństwa.

Utworzenie pełnej kopii baz danych umożliwia program narzędziowy mysqldump. W wyniku użycia tego narzędzia tworzony jest plik *sql* zawierający komendy źródłowej bazy danych.

Przed wykonaniem kopii bazy należy zablokować bazę poleceniem FLUSH TABLES.

Składnia polecenia do wykonania kopii zapasowej ma postać:

```
mysqldump -u użytkownik -p[hasło] nazwa_bazy] > plik.sql
```

Składnia polecenia do przywracania bazy danych wygląda podobnie:

```
mysqldump -u użytkownik -p[hasło] nazwa_bazy] < plik.sql
```

Podstawowe parametry polecenia mysqldump to:

- databases — kopiowana jest zawartość bazy danych i jej struktura. Opcja pozwala na zapisanie wielu baz danych w jednym pliku.
- all-databases (lub -A) — kopiowana jest zawartość wszystkich baz danych i tabel dostępnych na serwerze. Nie ma możliwości wybierania pojedynczych baz.
- no-create-info — w utworzonej kopii nie zostanie zapisana informacja o strukturze tabel (nazwy pól, nazwy tabel, indeksów). Opcja tworzy kopię samych danych.
- no-data — tworzona jest kopia tylko struktury bazy i tabel.

- `default-character-set=charset_name` — możliwość ustawienia kodowania znaków podczas robienia kopii bezpieczeństwa.
- `opt` *nazwa_bazy* — tworzona jest kopia bazy danych wraz z rozszerzonymi informacjami MySQL na przykład na temat blokowania tabel.
- `single-transaction` — tworzona jest spójna kopia działającej bazy danych.
- `flush-logs` — przed wykonaniem kopii bezpieczeństwa nastąpi zapisanie dziennika transakcji.
- `master-data` — w pliku kopii zostaną zapisane informacje o bieżącym dzienniku transakcji, takie jak: nazwa, stan, pozycja, potrzebne do ustawienia parametrów replikacji dla serwera *slave*.
- `add-drop-database` — przywraca bazę danych z pliku przy jednoczesnym usunięciu istniejącej bazy.
- `add-drop-table` — przywraca tabelę (strukturę i dane lub strukturę) z pliku przy jednoczesnym usunięciu istniejącej tabeli.
- `host` (lub `-h`) — nazwa lub adres IP komputera, z którego mają zostać skopiowane lub przywrócone dane.
- `user` (lub `-u`) — deklaracja użytkownika, który ma prawo do wykonywania kopii zapasowych bazy.
- `password` (lub `-p`) — opcja wykorzystywana do uwierzytelnienia użytkownika na serwerze bazodanowym.

Przykład 6.29

```
mysqldump -u user -phaslo databases > kopia_baz.sql
```

W wyniku wykonania polecenia podanego w przykładzie zostanie utworzona kopia wszystkich baz danych serwera.

Przykład 6.30

```
mysqldump -u user -phaslo ksiegarnia > kopia_ksiegarnia.sql
```

W wyniku wykonania polecenia zostanie utworzona kopia bazy danych *ksiegarnia*.

Przykład 6.31

```
mysql -u root -phaslo
mysql -u root -phaslo ksiegarnia < /katalog/kopia_ksiegarnia.sql
```

W wyniku wykonania poleceń po zalogowaniu się do serwera zostanie przywrócona z pliku *kopia_ksiegarnia.sql* baza danych *ksiegarnia*.

Przykład 6.32

```
mysqldump -u root -p --single-transaction --all-database > kopia_baz.sql
```

W podanym przykładzie zostanie utworzona kopia bezpieczeństwa baz transakcyjnych zawierających tabele typu `BDB` lub `InnoBD`.

Przykład 6.33

```
mysqldump -u root -p --single-transaction --flush-logs --master-data
--all-database > kopia_baz.sql
```

W wyniku wykonania polecenia z przykładu będzie utworzona spójna kopia działającej bazy danych, a także zapisane zostaną dziennik transakcji oraz informacje o nim.

Tworzenie kopii bezpieczeństwa dla tabel typu `MyISAM` polega na skopiowaniu plików typu *frm*, *MYD* oraz *MYI*.

Inną metodą tworzenia kopii bezpieczeństwa dla tabel typu InnoDB może być zatrzymanie serwera MySQL i skopiowanie plików: *ibd*, *ib_logfile** , *frm*, *my.cnf*.

6.12.4. Tworzenie przyrostowej kopii danych

Tworzenie przyrostowej kopii bezpieczeństwa jest związane z dziennikiem transakcji (ang. *binary log*). Są w nim zapisywane wszystkie operacje dotyczące modyfikowania bazy danych wykonywane od momentu utworzenia ostatniej kopii bezpieczeństwa. W przypadku awarii istnieje możliwość odtworzenia bazy danych na podstawie tych zapisów. Dzienniki transakcji są traktowane jako kopie przyrostowe bazy danych. Aby w MYSQL operacje były zapisywane do dziennika transakcji, należy uruchomić serwer z opcją `--log-bin` lub ustawić tę opcję w pliku konfiguracyjnym. Przy tworzeniu nowego pliku dziennika do jego nazwy dodawany jest kolejny numer. Nowy dziennik transakcji jest tworzony zawsze, gdy:

- uruchamiany jest serwer,
- zostanie wykonane polecenie `FLUSH LOGS`,
- aktualny plik dziennika osiągnie maksymalny rozmiar.

Pliki dziennika transakcji mogą zostać usunięte poleceniem:

```
RESET MASTER;
```

lub:

```
PURGE MASTER LOGS;
```

Do lokalizacji plików dzienników transakcji służy polecenie:

```
SHOW BINLOG EVENTS;
```

Aby odtworzyć stan bazy danych sprzed awarii, potrzebna jest pełna kopia bazy oraz plik dziennika transakcji.

6.12.5. Odzyskiwanie danych

Odzyskiwanie bazy danych z kopii bezpieczeństwa odbywa się w dwóch etapach:

- odzyskanie bazy danych z pełnej kopii (na przykład utworzonej programem `mysqldump`) poleceniem:

```
mysql -u root -p < kopia.sql
```

- odzyskanie wszystkich danych zmodyfikowanych po utworzeniu pełnej kopii bezpieczeństwa poleceniem:

```
mysqlbinlog -u root -p binlog.2 | mysql
```

Zadanie 6.8

Po uruchomieniu serwera MySQL w trybie rejestracji dziennika transakcji wykonaj przy użyciu programu `mysqldump` kopię bezpieczeństwa bazy *Prawo jazdy*. Kopia bezpieczeństwa powinna być spójna i powinien zostać utworzony dziennik transakcji. Zapisany dziennik transakcji potraktuj jako kopię przyrostową bazy. Dodaj nowych kursantów do tabeli *Osoby*. Zrestartuj serwer.

Wykonaj kilka działań na bazie *Prawo jazdy*. Dodaj następnych kursantów do tabeli *Osoby*, usuń wszystkie kursy z tabeli *Kursy*, zmodyfikuj dane w tabeli *Rejestracja jazd*.

Przy założeniu, że wykonane po restarcie działania były związane z awarią serwera MySQL, ponownie zrestartuj go i po restarcie spróbuj odzyskać utracone dane (dane wprowadzone do bazy po wykonaniu kopii bezpieczeństwa, przed pierwszym restartem), wykorzystując posiadane kopie bezpieczeństwa i dzienniki transakcji.

6.13. Eksport i import danych

6.13.1. Eksport poleceniem SELECT

Do szybkiego wyeksportowania danych do pliku tekstowego może zostać użyte polecenie `SELECT ... INTO OUTFILE`, które umożliwia zapisanie wyniku wykonania zapytania do pliku. W celu uzyskania określonego formatu zapisu danych mogą zostać zdefiniowane dodatkowe opcje polecenia.

```
SELECT *
INTO OUTFILE 'c:/temp/plik1.txt'
FIELDS
TERMINATED BY '\t'
ESCAPED BY ''
OPTIONALLY ENCLOSED BY '"'
LINES
TERMINATED BY '\r\n'
FROM nazwa tabeli
WHERE warunek;
```

Aby móc eksportować zawartość tabel za pomocą polecenia `SELECT`, wymagane jest posiadanie prawa `FILE`. Użytkownik posiadający takie uprawnienia może zapisywać pliki na dysku. Z powodów bezpieczeństwa należy przypisywać prawo `FILE` tylko administratorom.

Jeżeli plik o nazwie podanej w klauzuli `INTO OUTFILE` istnieje, jego zawartość nie zostanie zapisana. Opcje zapisane w klauzulach `FIELDS` oraz `LINES` decydują o sposobie zapisu danych do pliku.

6.13.2. Eksport danych przy użyciu mysqldump

Program `mysqldump`, za pomocą którego wykonywana jest kopia bezpieczeństwa, może zostać wykorzystany do przeniesienia danych na inny serwer, który potrafi odczytywać pliki zapisane w jego formacie. Wykorzystując odpowiednie opcje programu, można skopiować do pliku tekstowego interesujące nas informacje, na przykład polecenia tworzące struktury tabel oraz zapisane w nich dane.

Przykład 6.34

```
mysqldump --all-databases --opt no-data --user=root --password=mysql
--result-file=c:\temp\baza.sql
```

W wyniku wykonania podanego polecenia zostanie skopiowana do pliku tekstowego tylko struktura bazy danych, bez zawartości tabel.

Przykład 6.35

```
mysqldump --databases ksiegarnia --tables ksiazki autor --skip-quote-names
--comments=1 --complete-insert --create-options --add-drop-table --skip-opt
--user=root --password=mysql --result-file=c:\temp\zbior.sql
```

W podanym przykładzie do pliku tekstowego z bazy danych *ksiegarnia* zostaną skopiowane polecenia tworzące struktury tabel *ksiazki* i *autor* oraz ich zawartości.

6.13.3. Import danych

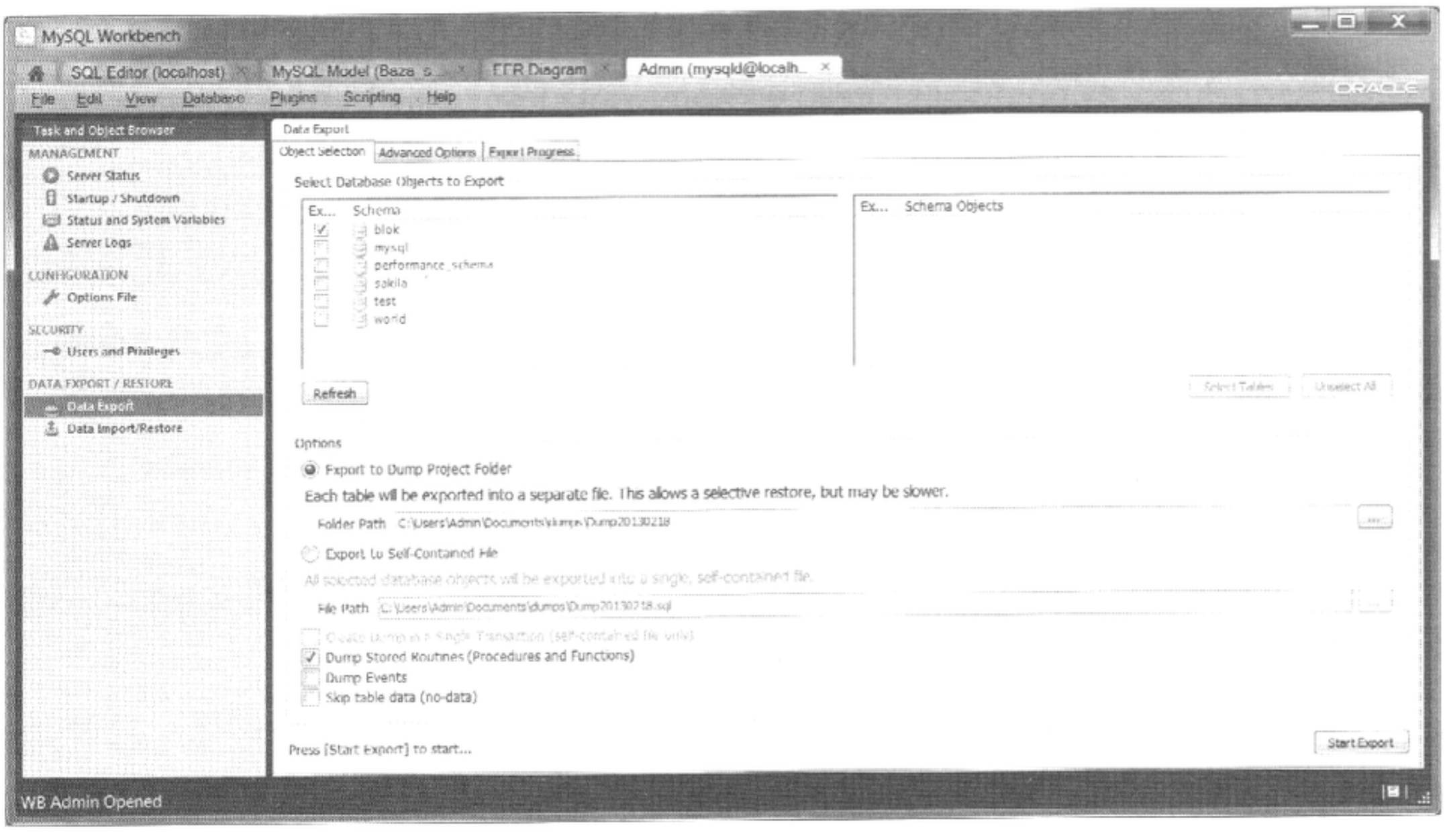
Import danych można zrealizować za pomocą programu narzędziowego `mysqlimport`.

Przykład 6.36

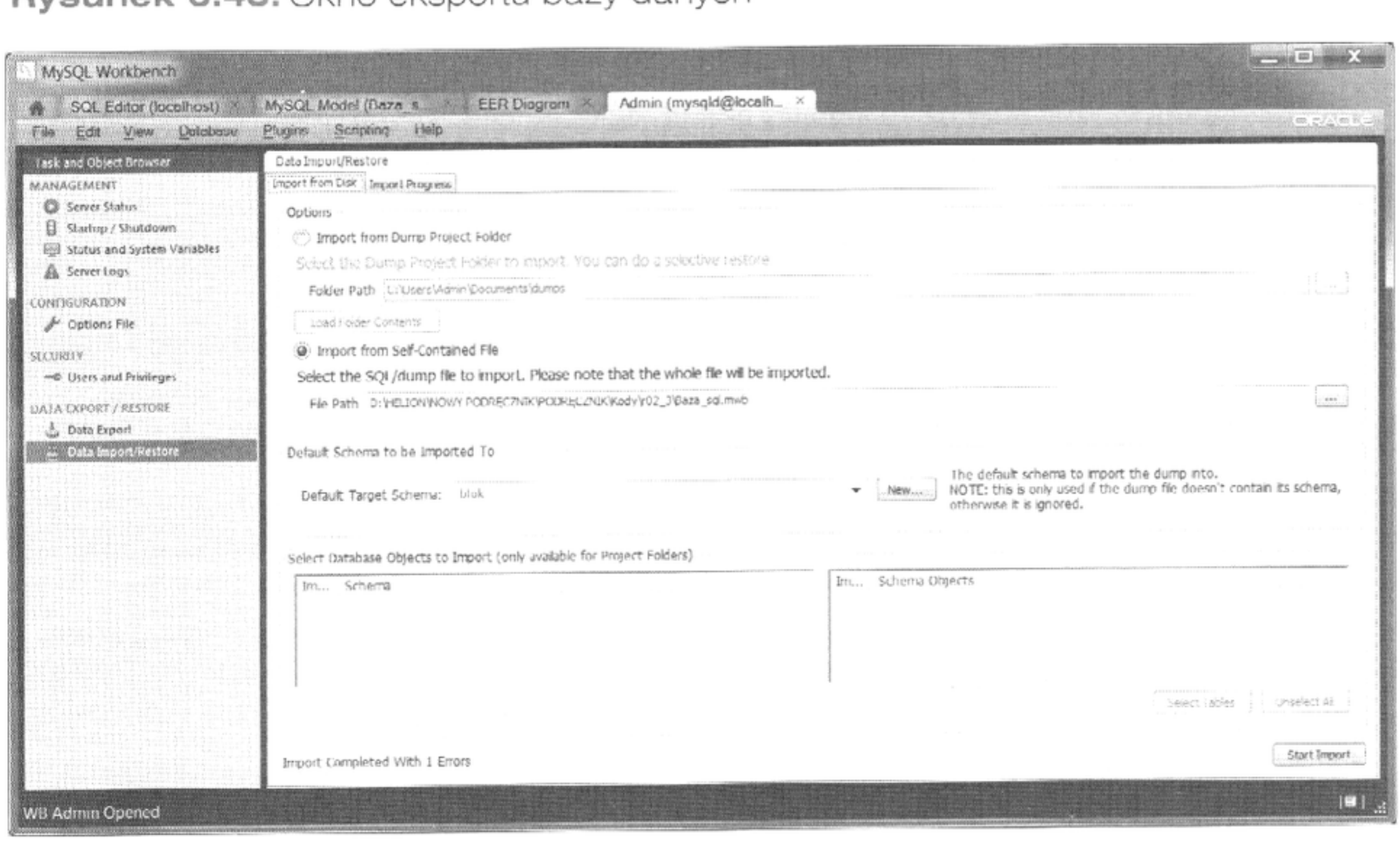
```
mysqlimport  user=root  --password=xxx  --fields-terminated-by=,
--fields-enclosed-by=' --lines-terminated-by=\r\n --delete ksiegarnia c:\
temp\zbior.sql
```

W programie MySQL Workbench import i eksport danych są realizowane za pomocą opcji *Manage Import/Export* dostępnej w oknie głównym w panelu *Server Administration*. Przy użyciu tego narzędzia można utworzyć zrzut z bazy danych lub przywrócić dane z pliku do bazy. Po wybraniu opcji w otwartym oknie należy wybrać zakładkę *Export to Disk*, aby wykonać eksport bazy danych (rysunek 6.48). Eksportować można schemat bazy oraz tabele.

Import danych jest realizowany w zakładce *Import from Disk* (rysunek 6.49). Po podaniu ścieżki do pliku można importować wcześniej wyeksportowany projekt (lub dowolny inny projekt). Zakładka *Advanced Export Options* poprzez wybór różnych opcji umożliwia ustawienie parametrów eksportu danych.



Rysunek 6.48. Okno eksportu bazy danych



Rysunek 6.49. Okno importu bazy danych do wskazanego schematu

6.14. Udostępnianie zasobów

Zainstalowany i skonfigurowany serwer baz danych MySQL pracuje lokalnie i nasłuchuje na lokalnym porcie 3306 (*localhost: 3306*).

Zdalny dostęp do zasobów serwera ze względu na bezpieczeństwo danych jest wyłączony.

Dostęp do serwera można zdefiniować dla użytkownika *root* lub dla dowolnego użytkownika z pełnymi uprawnieniami.

Przykład 6.37

```
GRANT ALL PRIVILEGES ON *.* TO 'marcin'@'%' IDENTYFIED BY 'haslo' WITH
GRANT OPTION;

FLUSH PRIVILEGES;
```

W podanym przykładzie użytkownikowi *marcin* pozwalamy połączyć się z dowolnego hosta (%) z bazą danych z pełnymi uprawnieniami.

Użytkownik *marcin* uzyskał również prawo nadawania uprawnień. Polecenie `FLUSH PRIVILEGES` spowoduje zatwierdzenie przydzielonych uprawnień.

Jeżeli dostęp do serwera baz danych chcemy ograniczyć do jednego hosta, to jego adres powinien zostać wpisany w poleceniu zamiast znaku %. Następnie należy zmienić ustawienia serwera tak, aby nasłuchiwał na wskazanych adresach, a nie tylko lokalnie.

W tym celu trzeba zmodyfikować plik konfiguracyjny *my.ini* lub *my.cnf*. Należy w nim odszukać linię:

```
bind-address = 127.0.0.1
```

Linię tę można usunąć i wtedy serwer będzie nasłuchiwał na wszystkich skonfigurowanych adresach IP. Można też w linii wpisać wybrany adres — wtedy serwer będzie dostępny tylko na tym adresie. Aby zatwierdzić wprowadzone zmiany, serwer trzeba zrestartować.

Zadanie 6.9

Ustaw opcje konfiguracyjne serwera MySQL w taki sposób, aby umożliwić użytkownikom sieci lokalnej połączenie się z programem MySQL komputera, na którym została umieszczona baza danych **Prawo jazdy**.

Skonfiguruj uprawnienia tych użytkowników do bazy danych.

6.15. Optymalizacja wydajności SZBD

6.15.1. Optymalizacja wydajności systemu bazodanowego

Wydajność SZBD określana jest najczęściej za pomocą następujących parametrów:

- liczby operacji przeprowadzanych na sekundę, czyli przepustowości systemu;
- czasu potrzebnego na zwrócenie wyniku, czyli czasu reakcji systemu.

Wartość pierwszego parametru zależy przede wszystkim od dostępnych dla serwera baz danych zasobów systemu komputerowego (pamięci operacyjnej, mocy obliczeniowej, wydajności systemu wejścia-wyjścia, przepustowości lokalnej sieci). Wartość drugiego zależy głównie od logicznej i fizycznej struktury bazy danych i od aplikacji klienckich. Wpływ na wydajność systemu bazodanowego ma również stosowana przez serwer i klientów metoda pobierania i przetwarzania danych. Mała wydajność jednego z podanych składników może negatywnie wpływać na wyniki monitorowania wydajności pozostałych składników.

Monitorowanie wydajności systemu bazodanowego należy przeprowadzać, zaczynając od sprawdzenia wydajności komputera. Następnie trzeba prześledzić pracę systemu operacyjnego, sprawdzić wykorzystanie zasobów systemowych przez serwer baz danych oraz przeanalizować strukturę bazy danych. Powinno się również monitorować wydajność aplikacji klienckiej.

6.15.2. Optymalizacja bazy danych

Jednym z czynników wpływających na wydajność pracy serwera jest odpowiednie skonstruowanie logicznej i fizycznej struktury bazy danych oraz aplikacji klienckich. Najczęstszymi przyczynami małej efektywności pracy z bazą danych są:

- nieefektywne projekty schematów baz,
- źle utworzone indeksy,
- źle skonstruowane zapytania,
- nieoptymalna struktura przechowywania danych.

Większość baz pracuje na ogromnej ilości danych, dlatego powinny one być projektowane tak, aby jak najmniej obciążały serwer, oraz aby dane nie zajmowały zbyt dużo miejsca na dysku. Wykonanie optymalizacji bazy danych może pomóc w zwiększeniu jej wydajności oraz zmniejszeniu jej rozmiarów. Podstawowymi działaniami, które powinny zostać wykonane, są:

- normalizacja bazy danych,
- prawidłowe tworzenie indeksów,
- optymalizacja zapytań i operacji wykonywanych w bazie danych.

Normalizacja

Prawidłowo opracowany projekt schematu bazy danych jest pierwszym elementem optymalizacji. Zasady poprawnej budowy schematu bazy są definiowane za pomocą postaci normalnych. Normalizacja chroni przed powielaniem tej samej informacji w tabelach. Brak normalizacji prowadzi do wielu utrudnień (anomalii bazy danych) w trakcie korzystania z bazy i jej modyfikowania. W celu zwiększenia wydajności bazy danych należy zadbać, aby schemat baz danych spełniał wymogi normalizacji.

Indeksy

Indeksy w bazie danych pozwalają na szybkie wyszukiwanie danych, co powoduje znacznie szybsze wykonywanie zapytań do bazy.

Zaletą stosowania indeksów jest ograniczenie ilości danych odczytywanych z bazy, przyspieszenie wyszukiwania informacji oraz sortowanie danych. Dostęp do rekordu odszukanego przez indeks jest bardzo szybki i efektywny. Wadą indeksów jest to, że zajmują na dysku dodatkowe miejsce, a ponadto muszą być na bieżąco aktualizowane. Każde wstawienie, usunięcie lub aktualizacja danych w tabeli wiąże się z aktualizacją wszystkich zdefiniowanych dla niej indeksów. Zastosowanie indeksowania daje najlepsze rezultaty przy wybieraniu małej liczby rekordów z dużego zbioru. Wtedy najlepiej widać korzyść z ograniczenia ilości danych, które muszą zostać odczytane z dysku. Przyjmuje się, że indeks jest opłacalny, gdy z tabeli odczytywane jest nie więcej niż około 15% rekordów.

Zastosowanie indeksu przyspiesza odczyt danych. Spowalnia jednak modyfikowanie danych, ponieważ wymaga aktualizowania informacji w indeksach. Jeżeli tabela zawiera kilka indeksów, to czas modyfikacji danych może być wielokrotnie dłuższy w porównaniu z aktualizacją tej tabeli bez indeksów. Jeżeli na przykład tylko dodajemy nowe dane do bazy, użycie indeksów może znacznie spowolnić jej działanie.

Poniżej został pokazany przykład wpływu indeksów na wydajność bazy danych.

Przykład 6.38

Utworzone zostały dwa indeksy, każdy dla innego pola:

```
CREATE INDEX indeks_imie ON Klient (imie);

CREATE INDEX indeks_nazw ON Klient (nazwisko);
```

Przykład 6.39

Utworzony został jeden indeks dla dwóch kolumn jednocześnie:

```
CREATE INDEX indeks_imie_nazw ON Klient (imie, nazwisko);
```

Po wykonaniu zapytania:

```
SELECT * FROM Klient WHERE imie='Jan' AND nazwisko='Kowalski';
```

w pierwszym przypadku wykonane zostaną następujące czynności:

- wyszukane zostaną wszystkie rekordy dla *imie* = *Jan*,
- wyszukane zostaną wszystkie rekordy dla *nazwisko* = *Kowalski*,
- zostanie wybrana wspólna część zbiorów rekordów z pierwszego i drugiego wyszukania i zwrócona jako wynik zapytania.

W drugim przypadku potrzebne dane zostaną wyszukane w jednym kroku, w którym równocześnie sprawdzone zostaną wartości w polach *imie* i *nazwisko*.

Każdy indeks zajmuje pewną ilość miejsca na dysku, dlatego nie należy tworzyć indeksów dla każdej kolumny w bazie. Należy przeanalizować zapytania, które będą wykonywane w bazie, i podjąć decyzję, gdzie i jakie indeksy utworzyć. Indeksy należy tworzyć dla kluczy głównych oraz dla kluczy obcych, gdyż te pola są najczęściej przeszukiwane.

Optymalizacja zapytań i operacje wykonywane w bazie danych

Podczas wykonywania zapytania oprócz uzyskania interesujących nas informacji bardzo ważna jest szybkość wykonania tego zapytania. W systemach zarządzania bazą danych występuje narzędzie służące do optymalizacji zapytań (ang. *query optimizer*). Aby zapewnić wydajność bazy danych, optymalizator analizuje zapytanie i tworzy możliwie optymalny plan jego wykonania (ang. *execution plan*). Plan ten zawiera opis użytych indeksów, sposób dostępu do danych z tabeli, sposób i kolejność łączenia tabel, uwzględnienie warunków z sekcji `WHERE`.

Aby przyspieszyć wykonywanie zapytań, należy je konstruować tak, by były wykonywane na jak najmniejszej ilości danych. Zmniejszenie ilości przetwarzanych danych zmniejsza ilość potrzebnych zasobów oraz zwiększa efektywność działania indeksów.

Jedną z metod zmniejszenia ilości przetwarzanych danych jest ograniczenie listy kolumn w zapytaniu `SELECT`.

Przykład 6.40

Zamiast stosować polecenie w postaci:

```
SELECT * FROM Klient WHERE imie='Jan' AND nazwisko='Kowalski';
```

można zapisać:

```
SELECT id_klienta, imie, nazwisko FROM Klient WHERE imie='Jan' AND
nazwisko='Kowalski';
```

Inną metodą prowadzącą do zmniejszenia ilości przetwarzanych danych jest ich filtrowanie za pomocą klauzuli `WHERE`:

```
SELECT nazwisko, imie, miejscowosc, ulica, nr_domu
FROM Klient
WHERE miejscowosc='Kraków';
```

Nie powinno się stosować porządkowania danych, jeżeli nie jest to konieczne (klauzula `ORDER BY`). Dane należy pobierać w takiej kolejności, w jakiej są zapisane w bazie. Trzeba unikać zagnieżdżonych zapytań i klauzuli `GROUP BY`.

Transakcje

Sama transakcja nie jest narzędziem optymalizacji, natomiast może mieć na nią wpływ. Zatwierdzenie transakcji instrukcją `COMMIT` powoduje zapisanie zmian dokonanych w bazie danych. Natomiast brak zatwierdzenia transakcji prowadzi do utrzymywania rosnącego zbioru wartości danych na wypadek wycofania zmian instrukcją `ROLLBACK`. Najczęściej sposób obsługi transakcji jest wymuszany względami bezpieczeństwa danych i częstsze wykonywanie instrukcji `COMMIT` jest możliwe tylko w nielicznych przypadkach.

Procedury składowane w bazie danych

Inną możliwością optymalizacji bazy danych jest tworzenie procedur i funkcji składowanych. Dzięki temu, że są one przechowywane i wykonywane w bazie danych, zostają wyeliminowane opóźnienia związane z przesyłaniem danych poza bazę. W związku z tym możliwy jest wzrost wydajności.

Struktura bazy danych

Wielkość bazy danych, jej uporządkowanie oraz operacje wykonywane na plikach bazodanowych mają istotny wpływ na wydajność bazy danych. Aby zapewnić maksymalną wydajność, należy:

- umieścić pliki bazy danych na oddzielnym dysku twardym,
- pliki dziennika transakcyjnego umieścić na oddzielnym, wydajnym dysku twardym,
- utworzyć dla bazy danych kilka plików i umieścić je na oddzielnych dyskach twardych,
- dla tabel oraz powiązanych z nimi indeksów utworzyć grupy plików i zapisać je na oddzielnych dyskach,
- wykonywać regularne defragmentowanie dysków,
- uporządkować logiczną strukturę plików bazy danych.

6.15.3. Optymalizacja wydajności serwera MS SQL

Tak jak wydajność każdego serwera bazodanowego, również wydajność serwera MS SQL Server zależy w dużej mierze od zasobów systemu komputerowego. Na przykład mała ilość pamięci operacyjnej spowoduje częstsze odwoływanie się do danych zapisanych na dysku, a to poskutkuje obniżeniem wydajności systemu. Powstaną kolejki procesów czekających na obsłużenie. Skutkiem powstałej sytuacji bieżące wysyłanie danych do klientów i ich odbieranie stanie się niemożliwe.

SQL Server działa w środowisku systemu Windows, dlatego wszystkie zasoby systemowe można monitorować, korzystając z narzędzi *Monitor systemu* oraz *Dzienniki wydajności*

i alerty. Domyślna konfiguracja serwera SQL Server umożliwia dynamiczne przydzielanie dostępnych zasobów sprzętowych potrzebnych do pracy serwera i w większości przypadków nie ma potrzeby jej zmieniania.

Pamięć operacyjna

Menedżer pamięci w programie Microsoft SQL Server eliminuje konieczność ręcznego zarządzania pamięcią dostępną dla serwera SQL Server. Po uruchomieniu SQL Server dynamicznie określa ilość potrzebnej pamięci na podstawie ilości pamięci systemu operacyjnego i innych aktualnie używanych aplikacji. Jest to optymalny przydział, zapewniający najlepszą wydajność pracy serwera.

6.15.4. Poprawa wydajności serwera MySQL

Czynnikami wpływającymi na wydajność pracy serwera MySQL mogą być sprzęt lub konfiguracja serwera. Na przykład zwiększenie ilości pamięci operacyjnej może znacząco poprawić wydajność systemu.

Konfigurację serwera można sprawdzić i zmodyfikować w pliku konfiguracyjnym *my.ini* lub *my.cnf*. Dostępne są następujące parametry:

- `key_buffer_size` — opisuje ilość pamięci dostępnej do przechowywania kluczy indeksu. Domyślna wartość to 8 MB, ale parametr może zostać ustawiony nawet na 25% pamięci RAM.
- `query_cache_size` — opisuje ilość pamięci zarezerwowanej na zapytania. Domyślna wartość jest równa 0. Jeżeli w bazie danych występuje dużo powtarzających się zapytań, wartość tę należy zwiększyć.
- `table_open_cache` — określa liczbę deskryptorów tabeli MySQL przechowywanych w pamięci podręcznej. Domyślną wartością jest 64. Jeśli wielu użytkowników korzysta jednocześnie z tabel, wartość ta powinna zostać zwiększona.

Statystyki działania serwera można uzyskać po uruchomieniu programu narzędziowego `mysqlreport`. Wyświetli on raport z informacją o konfiguracji serwera oraz o wykorzystaniu zasobów przez serwer. Na podstawie tego raportu można ustalić obciążenie serwera, sprawdzić, czy konfiguracja jest poprawna, a także czy struktura bazy danych jest prawidłowa i czy zapytania są optymalne.

Zadanie 6.10

W wybranym przez siebie serwerze baz danych (SQL Server lub MySQL) przeprowadź optymalizację bazy danych *Księgarnia internetowa*. Przeanalizuj poprawność schematu bazy danych. Sprawdź zdefiniowane klucze podstawowe, klucze obce i utworzone indeksy. Przeanalizuj i zmodyfikuj konstrukcję utworzonych zapytań.

PYTANIA KONTROLNE

1. Co należy do podstawowych zadań administratora baz danych?
2. Na czym polega tryb uwierzytelnienia na serwerze MS SQL Server?
3. Na czym polega autoryzacja użytkownika na serwerze MS SQL Server?
4. W jaki sposób na serwerze MS SQL Server nadawane są uprawnienia użytkownikom?
5. Na czym polega replikacja bazy danych na serwerze MS SQL Server?
6. Jak działa replikacja migawkowa na serwerze MS SQL Server?
7. Kiedy powinno się wykonywać pełną kopię bazy danych?
8. Na czym polega sprawdzanie spójności bazy danych?
9. Jak działa tryb uwierzytelnienia na serwerze MySQL?
10. W jaki sposób na serwerze MySQL nadawane są uprawnienia użytkownikom?
11. Na czym polega replikacja bazy danych na serwerze MySQL?
12. Kiedy powinna być tworzona kopia przyrostowa bazy danych?
13. Na czym polega optymalizacja wydajności Systemu Zarządzania Bazą Danych?

ZADANIA

1. Korzystając z wybranego serwera baz danych, skonfiguruj dla bazy danych *Moja_szkoła* konta użytkowników. Przypisz im odpowiednie role i uprawnienia do bazy.
2. Wykonaj replikację bazy danych *Moja_szkoła*. Przetestuj działanie repliki.
3. Utwórz pełną kopię bezpieczeństwa bazy danych *Moja_szkoła*. Określ miejsce jej przechowywania.
4. Ustaw opcje konfiguracyjne serwera w taki sposób, aby umożliwić użytkownikom sieci lokalnej połączenie się z serwerem, na którym została umieszczona baza danych *Moja_szkoła*. Skonfiguruj uprawnienia tych użytkowników do bazy danych.
5. Przeprowadź optymalizację bazy danych *Moja_szkoła*. Przeanalizuj poprawność schematu bazy danych. Sprawdź zdefiniowane klucze podstawowe, klucze obce i utworzone indeksy. Przeanalizuj i zmodyfikuj konstrukcję utworzonych zapytań.

Skorowidz

E

F

G

H

I

J

K

S

T

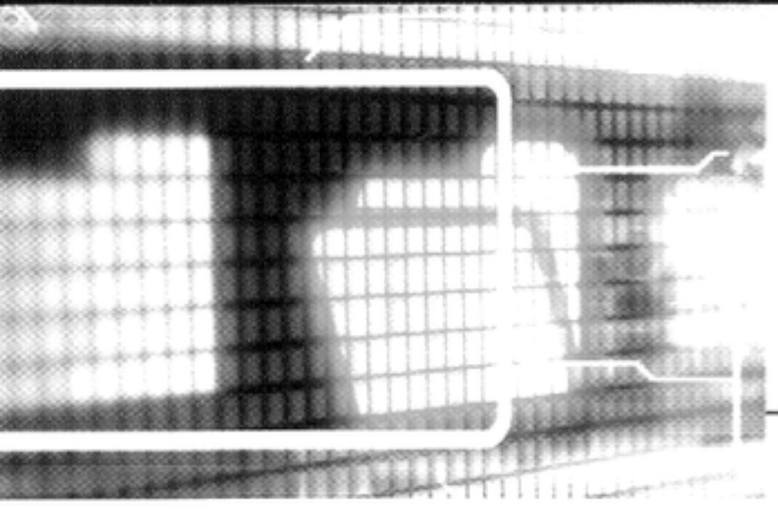
U

V

W

X

Z



Kwalifikacja E.14 Część 2

Tworzenie baz danych i administrowanie bazami

Podręcznik do nauki zawodu **technik informatyk**

Technik informatyk nie jest zwykłym użytkownikiem komputerów. Jeśli uczeń wybiera szkołę kształcącą w tym zawodzie, z czasem staje się prawdziwym komputerowym ekspertem.

Kwalifikacja E.14 dotyczy tematów związanych z tworzeniem aplikacji internetowych oraz projektowaniem i tworzeniem baz danych. Pierwszy podręcznik z tej serii zawierał wiedzę z zakresu projektowania i tworzenia stron internetowych. Drugi, *Tworzenie baz danych i administrowanie bazami*, wyjaśnia zasady projektowania baz danych i zarządzania tymi bazami. Podręcznik jest zgodny z najnowszą podstawą programową. Dzięki realizacji zawartych w nim zadań uczniowie zaczną bez problemu identyfikować terminy związane z relacyjnymi bazami danych, będą stosować zasady normalizacji tabel oraz poznają funkcje programu MS Access. Przyszli technicy informatycy poznają pojęcia związane z architekturą systemów bazodanowych. Nauczą się także instalować i konfigurować serwery baz danych oraz nimi zarządzać. Informacje dotyczące języka SQL ułatwią im tworzenie bazy danych oraz pozwolą na odczytywanie i modyfikowanie zawartych w niej danych, a także kontrolowanie dostępu do nich. Uczniowie zapoznają się też z zasadami stosowania transakcji, procedur składowanych oraz wyzwalaczy. Podręcznik składa się z sześciu części, a jego budowa pozwala na realizowanie treści programowych w sposób wybrany przez nauczyciela.

Technik informatyk to doskonały, charakteryzujący się wysoką jakością, kompletny zestaw edukacyjny, przygotowany przez dysponującego ogromnym doświadczeniem lidera na rynku książek informatycznych — wydawnictwo Helion.

W skład zestawu *Technik informatyk* wchodzą także:

Kwalifikacja E.12. Montaż i eksploatacja komputerów osobistych oraz urządzeń peryferyjnych. Podręcznik do nauki zawodu technik informatyk

Kwalifikacja E.13. Projektowanie lokalnych sieci komputerowych i administrowanie sieciami. Podręcznik do nauki zawodu technik informatyk

Kwalifikacja E.14. Część 1. Tworzenie stron internetowych. Podręcznik do nauki zawodu technik informatyk

Kwalifikacja E.14. Część 3. Aplikacje internetowe. Podręcznik do nauki zawodu technik informatyk

Podręczniki oraz inne pomoce naukowe należące do tej serii zostały opracowane z myślą o wykształceniu kompetentnych techników, którzy świetnie poradzą sobie z trudnymi zadaniami w świecie współczesnej informatyki. Według nowych przepisów, aby otrzymać tytuł technika informatyka, należy potwierdzić trzy kwalifikacje wyodrębnione w tym zawodzie — to niewątpliwie wyzwanie i dla adeptów nauki o komputerach, i dla ich pedagogów. Ta książka pozwoli zarówno przygotować się do egzaminów, jak i uzyskać wiedzę i umiejętności oczekiwane na rynku pracy lub przydatne w przyszłej pracy.

helion.pl
księgarnia internetowa

Nr katalogowy: 12163

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Helion

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-6510-5