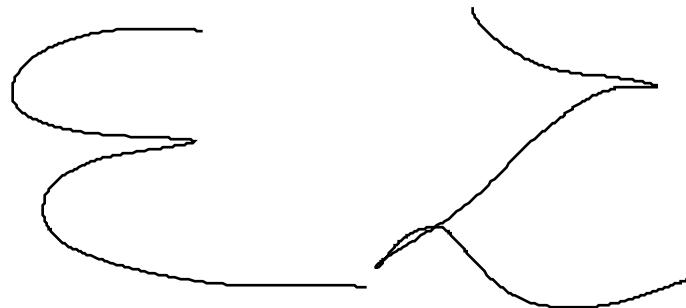




Olena Jewtuszenko, Michał Kuciej, Roman Trochimczuk

## *Bazy danych - MS ACCESS: przykłady i ćwiczenia*



Białystok 2018

**Recenzenci:**

Prof. dr hab. inż. Zbyszko Królikowski  
Instytut Informatyki, Politechnika Poznańska

Prof. dr hab. inż. Stanisław Kozielski  
Instytut Informatyki, Politechnika Śląska

**Redaktor wydawnictwa:**

Elżbieta Dorota Alicka

© Copyright by Politechnika Białostocka, Białystok 2018

ISBN 978-83-65596-64-2

ISBN 978-83-65596-65-9 (eBook)

<https://doi.org/10.24427/978-83-65596-65-9>



Publikacja jest udostępniona na licencji

Creative Commons Uznanie autorstwa-Użycie niekomercyjne-Bez utworów zależnych 4.0  
(CC BY-NC-ND 4.0)

Pełna treść licencji dostępna na stronie

[creativecommons.org/licenses/by-nc-nd/4.0/legalcode.pl](http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode.pl)

Publikacja jest dostępna w Internecie na stronie Oficyny Wydawniczej PB  
<http://pb.edu.pl/oficyna-wydawnicza/publikacje/publikacje/2018>

# Spis treści

<b>Przedmowa.....</b>	8
<b>I. Utworzenie przykładowej bazy danych w Microsoft Access 2016 ...</b>	10
<b>Rozdział 1. Podstawowe czynności .....</b>	10
1.1. Początek pracy z programem .....	10
1.2. Określenie domyślnego folderu .....	12
1.3. Dodanie zaufanej lokalizacji.....	13
1.4. Tworzenie pustej bazy danych .....	14
1.5. Zamknięcie, otwieranie, kompaktowanie bazy danych.....	18
<b>Rozdział 2. Projektowanie przykładowej tabeli .....</b>	19
2.1. Widok projektu tabeli.....	19
2.2. Utworzenie pierwszego pola tabeli .....	20
2.3. Klucz podstawowy .....	22
2.4. Zapisywanie tabeli.....	23
2.5. Dodanie pozostałych pól do projektu tabeli tOsoba .....	25
2.6. Zdefiniowanie właściwości pól tabeli.....	25
2.7. Wprowadzenie danych w widoku arkusza danych tabeli .....	35
<b>Rozdział 3. Najprostszy formularz .....</b>	37
3.1. Generowanie formularza do pracy z danymi tabeli .....	37
3.2. Widoki formularza .....	39
3.3. Pasek nawigacji.....	40
3.4. Zastosowanie formularza do wprowadzenia danych.....	40
3.5. Formularz dzielony .....	42
<b>Rozdział 4. Generowanie podstawowego raportu.....</b>	43
<b>Rozdział 5. Uzupełnienie bazy danych: nowe tabele i relacje .....</b>	46
5.1. Tabela tPokoje .....	47
5.2. Projekt tabeli tZakwaterowanie .....	48
5.3. Relacje .....	50

---

5.4. Formularz wspomagający wprowadzenie danych do tabeli tZakwaterowanie .....	58
5.5. Tabela tKoszty_zakwaterowania .....	60
5.6. Tabela tWpłaty .....	65
5.7. Przykład wprowadzenia zmian do projektu tabeli .....	70
<b>II. Kwerenda – podstawowe narzędzie bazy danych .....</b>	<b>72</b>
<b>Rozdział 6. Proste techniki znajdowania danych .....</b>	<b>72</b>
6.1. Przykładowa baza danych Northwind .....	73
6.2. Najprostsze metody wyszukiwania danych w tabeli .....	75
6.3. Metoda filtrowania danych według formularza .....	79
6.4. Informacje na temat kryteriów filtrowania .....	81
6.5. Filtr zaawansowany .....	87
6.6. Przykłady złożonych kryteriów filtrowania .....	90
Zadania do samodzielnego wykonania .....	91
<b>Rozdział 7. Kwerenda wybierająca .....</b>	<b>94</b>
7.1. Wprowadzenie .....	94
7.2. Kwerenda wybierająca zbudowana na jednej tabeli .....	95
7.3. Kwerenda wybierająca oparta na wielu tabelach .....	108
7.4. Używanie kwerendy jako źródła rekordów formularza .....	109
7.5. Przykład zastosowania w kwerendzie funkcji IIF .....	110
Zadania do samodzielnego wykonania .....	111
<b>Rozdział 8. Obliczenia za pomocą kwerend .....</b>	<b>114</b>
8.1. Pole obliczeniowe w kwerendzie .....	114
8.2. Wiersz podsumowujący w arkuszu danych kwerendy wybierającej .....	117
8.3. Funkcje agregujące w projekcie kwerendy wybierającej .....	118
8.4. Kwerenda krzyżowa .....	127
Zadania do samodzielnego wykonania .....	129
<b>Rozdział 9. Kwerendy funkcjonalne .....</b>	<b>131</b>
9.1. Kwerenda usuwająca .....	132
9.2. Kwerenda aktualizująca .....	133
9.3. Kwerenda tworząca tabelę .....	135
9.4. Kwerenda dołączająca .....	137
9.5. Kwerenda parametryczna .....	139
Zadania do samodzielnego wykonania .....	142
<b>III. Manipulowanie danymi za pomocą języka SQL .....</b>	<b>144</b>
<b>Rozdział 10. Pobieranie danych z jednej tabeli .....</b>	<b>146</b>
10.1. Instrukcja SELECT do wybierania określonych pól tabeli .....	146

10.2. Zastosowanie klauzuli WHERE .....	151
10.3. Sortowanie danych .....	157
10.4. Zapytania parametryczne .....	160
Zadania do samodzielnego wykonania .....	161
<b>Rozdział 11. Pobieranie danych z połączonych tabel .....</b>	<b>164</b>
11.1. Złączenie wewnętrzne (INNER JOIN) .....	164
11.2. Złączenie zewnętrzne lewostronne (LEFT JOIN) .....	166
11.3. Złączenie zewnętrzne prawostronne (RIGHT JOIN) .....	167
11.4. Instrukcja SELECT zbudowana na trzech tabelach .....	169
11.5. Pobieranie danych z tabel połączonych za pomocą podzapytania .....	169
11.6. Dodawanie synonimów do nazw tabel .....	171
Zadania do samodzielnego wykonania .....	172
<b>Rozdział 12. Instrukcje SQL do agregowania danych .....</b>	<b>173</b>
12.1. Przykłady zastosowania funkcji agregujących .....	173
12.2. Grupowanie .....	176
Zadania do samodzielnego wykonania .....	179
<b>Rozdział 13. Zapytania SQL do wprowadzenia, aktualizacji, usunięcia danych .....</b>	<b>180</b>
13.1. Wprowadzenie danych do tabeli .....	180
13.2. Aktualizacja danych w tabeli .....	182
13.3. Usunięcie danych z tabeli .....	183
Zadania do samodzielnego wykonania .....	185
<b>IV. Wprowadzenie do formularzy, raportów i makr .....</b>	<b>186</b>
<b>Rozdział 14. Formularze .....</b>	<b>186</b>
14.1. Struktura formularza w Widoku Projektu .....	187
14.2. Kontrolki (formanty) oraz ich arkusze właściwości .....	192
14.3. Formant obliczeniowy .....	195
14.4. Przykłady utworzenia formularzy w widoku projektu .....	201
14.5. Formularz nawigacji .....	211
Zadanie do samodzielnego wykonania .....	213
<b>Rozdział 15. Raporty .....</b>	<b>213</b>
15.1. Generowanie raportu przy użyciu narzędzia RAPORT .....	214
15.2. Tworzenie raportu przy użyciu narzędzia KREATOR RAPORTÓW .....	217
15.3. Tworzenie raportu w Widoku Projektu .....	218
Zadania do samodzielnego wykonania .....	225
<b>Rozdział 16. Makra .....</b>	<b>226</b>
16.1. Makra interfejsu użytkownika .....	226
16.2. Makra danych .....	233

---

16.3. Makro AutoExec .....	239
Zadania do samodzielnego wykonania .....	240
<b>V. Pierwsze kroki z programem Microsoft SQL Server Management Studio .....</b>	<b>242</b>
<b>Rozdział 17. Utworzenie najprostszej bazy danych przy zastosowaniu narzędzi graficznych .....</b>	<b>242</b>
17.1. Uruchomienie programu MS SQL Server Management Studio.....	242
17.2. Utworzenie pustej bazy danych .....	244
17.3. Stworzenie przykładowych tabel za pomocą narzędzi graficznych .....	246
17.4. Diagram bazy danych .....	252
17.5. Wprowadzenie danych do tabel .....	254
17.6. Modyfikacja tabeli za pomocą narzędzi graficznych .....	255
17.7. Modyfikacja opcji połączenia tabel .....	259
Zadania do samodzielnego wykonania .....	261
<b>Rozdział 18. Wprowadzenie do technik przeniesienia bazy danych ..</b>	<b>262</b>
18.1. Skrypt bazy danych .....	262
18.2. Utworzenie kopii bazy danych za pomocą operacji eksportowania .....	264
18.3. Utworzenie kopii zapasowej bazy danych (backup) .....	266
18.4. Odtworzenie bazy danych z kopii zapasowej .....	268
18.5. Odłączanie i dołączenie bazy danych .....	273
Zadania do samodzielnego wykonania .....	274
<b>Rozdział 19. Zastosowanie konstruktora Query Designer do utworzenia instrukcji Transact SQL .....</b>	<b>275</b>
19.1. Konstruowanie instrukcji SELECT do wyświetlenia danych .....	276
19.2. Konstruowanie zapytań agregujących .....	281
19.3. Konstruowanie instrukcji INSERT, UPDATE, DELETE .....	283
Zadania do samodzielnego wykonania .....	287
<b>Rozdział 20. Elementy wiedzy na temat instrukcji CREATE, ALTER, DROP .....</b>	<b>288</b>
20.1. Nazwa obiektu w systemie SQL Server .....	288
20.2. Przykłady zastosowania instrukcji Transact SQL do utworzenia, modyfikacji oraz usunięcia bazy danych .....	289
20.3. Przykłady zastosowania instrukcji Transact SQL do utworzenia, modyfikacji oraz usunięcia tabel i indeksów .....	292
20.4. Przykłady utworzenia widoków .....	302
Zadania do samodzielnego wykonania .....	306
<b>Rozdział 21. Zastosowanie programu Access do utworzenia aplikacji klienta bazy danych .....</b>	<b>307</b>

21.1. Połączenie programu Microsoft Access 2016 z bazą danych ulokowaną w systemie MSSQL Server 2014 .....	307
21.2. Wykonanie przykładowej aplikacji klienta bazy danych .....	314
<b>Rozdział 22. Wybrane konstrukcje programistyczne języka Transact SQL .....</b>	
22.1. Wsad, zmienna, przypisanie wartości .....	325
22.2. Przykłady narzędzi do sterowania przepływem wykonania .....	328
22.3. Procedura składowana .....	333
22.4. Instrukcja RETURN .....	341
<b>Literatura .....</b>	343

# PRZEDMOWA

W klasycznym podręczniku z podstaw baz danych J.D. Ulman oraz J. Widom [1] czytamy: „Czym jest baza danych? Baza danych nie jest w istocie rzeczy niczym więcej niż zbiorem danych istniejącym przez długi czas, często przez wiele lat. W potocznym rozumieniu termin „baza danych” odnosi się do zbioru danych zorganizowanego przez system zarządzania bazą danych”.

Do tej pory, w oparciu o różne modele, powstało wiele systemów zarządzania bazami danych (SZBD), jednak najczęściej wykorzystywane są relacyjne bazy danych. Trudność w nauce oprogramowania tego typu polega na tym, że jest ono tworzone przez różnych producentów, a poza tym jest bardzo skomplikowane i różnorodne. Tym niemniej relacyjne SZBD łączą wspólne cechy – zasady, na których tworzona jest relacyjna baza danych oraz wykorzystywany w nich język wysokiego poziomu (SQL). I to właśnie te zasady należy poznać, jeśli planowana jest praca z relacyjnymi bazami danych.

Nauka relacyjnych baz danych wymaga opanowania podstaw teoretycznych i jednocześnie niemożliwa jest w oderwaniu od konkretnego SZDB. Autorzy proponowanego podręcznika na praktyce przekonali się, że wstępna praktyczna znajomość z relacyjnymi bazami danych dobrze zacząć od wykonania i użytkowania własnej bazy danych w systemie Microsoft Access. Oprogramowanie to od wielu lat nie spada w światowym rankingu popularnych SZBD, ciągle pozostając w pierwszej dziesiątce wśród powyżej 340 wykorzystywanych SZBD wszystkich typów (aktualne informacje na ten temat można obejrzeć na stronie internetowej <https://db-engines.com/en/>). Jeśli chodzi o relacyjne bazy danych, to wśród 140 relacyjnych SZBD Microsoft Access wyprzedzają w popularności tylko takie systemy relacyjne, jak Oracle, MySQL, Microsoft SQL Server, PostgreSQL i DB2.

Proponowany podręcznik w oparciu o oprogramowanie Microsoft Access 2016 zawiera wprowadzenie do podstawowych operacji wykonywanych w relacyjnych bazach danych. W tym celu zaproponowano wykonanie licznych zadań wraz z opisami ich wykonania. Treść proponowanych zadań jest ze sobą logicznie powiązana. Nauka odbywa się metodą małych kroków – nowa wiedza bazuje na starej, w wyniku czego osoba, która wykonuje zaproponowane przykłady zadań, zaczyna rozumieć relacyjne bazy danych „od środka”.

Na początku podręcznika zaproponowano utworzenie od podstaw prostej bazy danych składającej się z pięciu tabel oraz stworzenie formularzy wspomagających gromadzenie danych. Następnie zamieszczono przykłady operacji wykonywanych w bazach danych. Po przyswojeniu takiej wiedzy czytelnik przechodzi do nauki języka SQL, która (ze względu na to, że operuje na poznanych wcześniej pojęciach) jest znacznie łatwiej i szybciej przyswajana.

Zaproponowano również wprowadzenie do zaawansowanych technik tworzenia formularzy, raportów i makr programu Access, które są niezbędne do tworzenia aplikacji bazodanowej.

Wiele uwagi w podręczniku poświęcono tworzeniu baz danych w systemie Microsoft SQL Server oraz zastosowaniu języka Transact SQL. Dodatkowo przedstawiono dokładny opis tworzenia prostej aplikacji bazodanowej klient-serwer, dla której baza danych jest umieszczona w systemie SQL Server 2014 Express With Advanced Services With Service Pack 1 32/64-bit, a modułem klienta jest aplikacja wykonana w programie Microsoft Access 2016. Na końcu zamieszczono wybrane informacje na temat narzędzi programistycznych języka Transact SQL.

W sumie w podręczniku zamieszczono powyżej 200 przykładów zadań wraz z opisami ich wykonania, a także sformułowano około 100 zadań do wykonania samodzielnego.

Niniejszy podręcznik jest zmodyfikowaną i w dużej mierze uzupełnioną wersją dwóch poprzednich podręczników autorów: Olena Jewtuszenko, Roman Trochimczuk „Praktyczne wprowadzenie do relacyjnych baz danych” oraz „Relacyjne bazy danych. Ćwiczenia praktyczne”, wydanych w Oficynie Wydawniczej Politechniki Białostockiej odpowiednio w latach 2010 i 2005.

W trakcie tworzenia podręcznika wykorzystano dokumentację oprogramowania Microsoft Access oraz Microsoft SQL Server, a także liczne wskazówki regularnie publikowane na stronie <https://msdn.microsoft.com/pl-pl/>.

Podręcznik jest adresowany głównie do studentów wydziałów mechanicznych wyższych uczelni technicznych.

Oczywiście, omawiane treści nie obejmują wszystkich aspektów relacyjnych baz danych. Czytelnik zawsze może poszerzyć swoją wiedzę na temat tego rodzaju baz danych za pomocą licznych monografii i podręczników. Listę niektórych z nich załączono na końcu podręcznika [1-9].

# I. Utworzenie przykładowej bazy danych w Microsoft Access 2016

W niniejszym rozdziale utworzymy przykładową bazę danych w środowisku systemu *Microsoft Access 2016*. Baza danych będzie obejmować osoby zakwaterowane w Domu Studenta pewnej uczelni wyższej. Zakładamy, że w tworzonej bazie danych będą przechowywane i wykorzystywane następujące informacje:

- a) dane osobowe studentów;
- b) dane o pokojach (numer pokoju, numer piętra, liczba miejsc);
- c) dane dotyczące zakwaterowania studentów (daty zakwaterowania i wykwaterowania studenta, nazwa wydziału, który go skierował do Domu Studenta, numer pokoju);
- d) dane dotyczące kosztów zakwaterowania studentów (miesięczna kwota do zapłaty, od kiedy obowiązuje, do kiedy obowiązuje);
- e) dane o wpłatach jednorazowych za zamieszkanie w Domu Studenta (wartość wpłaty, numer pokwitowania, data wpłaty, za który rok akademicki dokonano wpłaty).

## 1. Podstawowe czynności

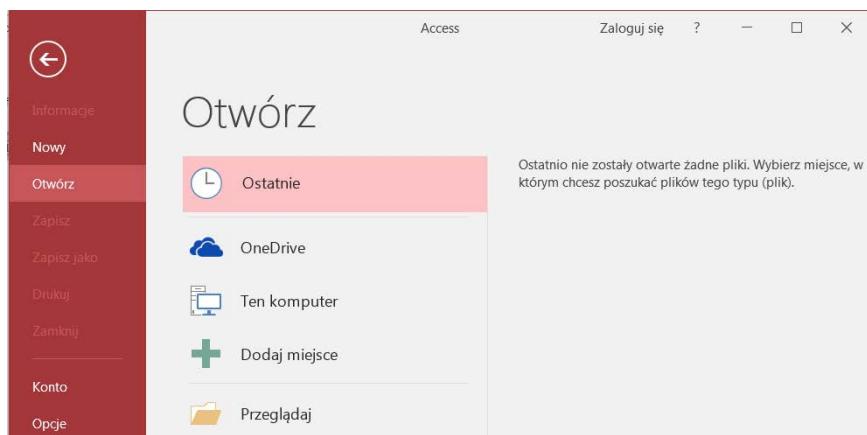
### 1.1. Początek pracy z programem

Uruchom program *Microsoft Access 2016*. Pierwsze okno programu o nazwie **Microsoft Office Backstage** zostało przedstawione na rysunku 1.1. W przypadku gdy na komputerze istnieją utworzone za pomocą *Microsoft Access 2016* bazy danych, ich wykaz zobaczymy pod napisem **Ostatnio używane**. Liczba wyświetlanych nazw plików jest zdefiniowana w opcjach programu. Jeśli otwieramy program po raz pierwszy, to okno ma wygląd taki, jak na rysunku 1.1.



Rys. 1.1. Widok Microsoft Office Backstage

Po prawej stronie okna znajdują się ikonki szablonów baz danych zaproponowane przez firmę *Microsoft*. Na górze został ulokowany monit z propozycją zalogowania się na osobistym koncie *Microsoft* (gdy go nie posiadamy, istnieje opcja założenia takiego konta). Za pomocą znaku ? możemy wywołać dokumentację programu.



Rys. 1.2. Drugie okno programu Microsoft Access 2016

Jeśli chcemy otworzyć inne pliki lub dodatkowe opcje programu, należy kliknąć napis **Otwórz inne pliki**. Zostanie wywołane kolejne okno programu (rys. 1.2).

## 1.2. Określenie domyślnego folderu

Baza danych systemu *Microsoft Access* zawarta jest w pliku o rozszerzeniu **accdb**. Ten plik należy przechowywać na dysku lokalnym komputera w określonym folderze.

O wiele sprawniej przebiega praca z programem, gdy na samym początku ustalimy roboczy folder dla wszystkich plików tworzonych w *Microsoft Access*.

### Zadanie 1\_1

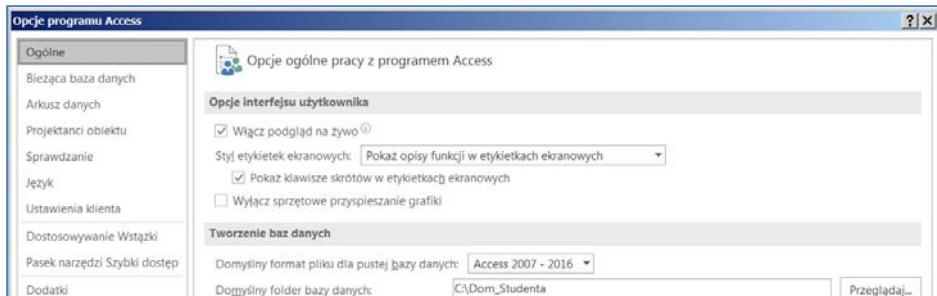
---

Utwórz roboczy folder o nazwie **Dom\_Studenta** na dysku lokalnym komputera (*C:* lub *D:*) dla baz danych programu *Microsoft Access*. W dalszej części tekstu będziemy odwoływać się do tego folderu, dlatego należy wskazać ten folder w środowisku programu jako domyślny.

---

### Wykonanie

- W pierwszym oknie (rys.1.1) programu **Access** kliknij napis **Otwórz inne pliki**.
- W kolejnym oknie (rys. 1.2) wybierz z menu **Opcje**.
- W oknie **Opcje programu Access**, w zakładce **Ogólne**, zdefiniuj swój folder (przykładowa ścieżka do folderu widoczna jest na rys. 1.3).
- Zatwierdź operację przyciskiem **OK** – okno opcji zostanie zamknięte.



Rys. 1.3. Okno **Opcje** programu *Microsoft Access* ze zdefiniowanym domyślnym folderem bazy danych

### 1.3. Dodanie zaufanej lokalizacji

Folder zawierający bazę danych (która niebawem utworzymy) należy zdefiniować jako zaufaną lokalizację.

#### Zadanie 1\_2

Zdefiniuj utworzony w **Zadaniu 1\_1** domyślny folder jako zaufaną lokalizację.

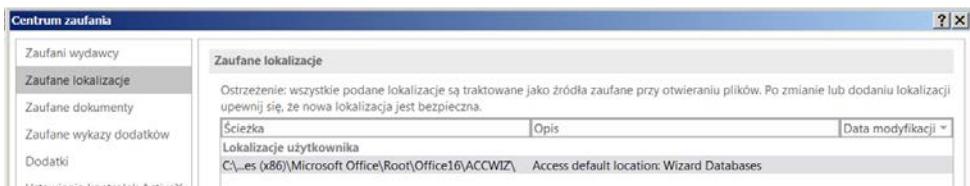
#### Wykonanie

- W oknie **Opcje** programu **Access** (rys. 1.3) wybierz zakładkę **Centrum zaufania**.
- W kolejnym oknie kliknij przycisk **Ustawienia Centrum zaufania** (rys. 1.4) – zostanie otworzone okno **Centrum zaufania** (rys. 1.5).



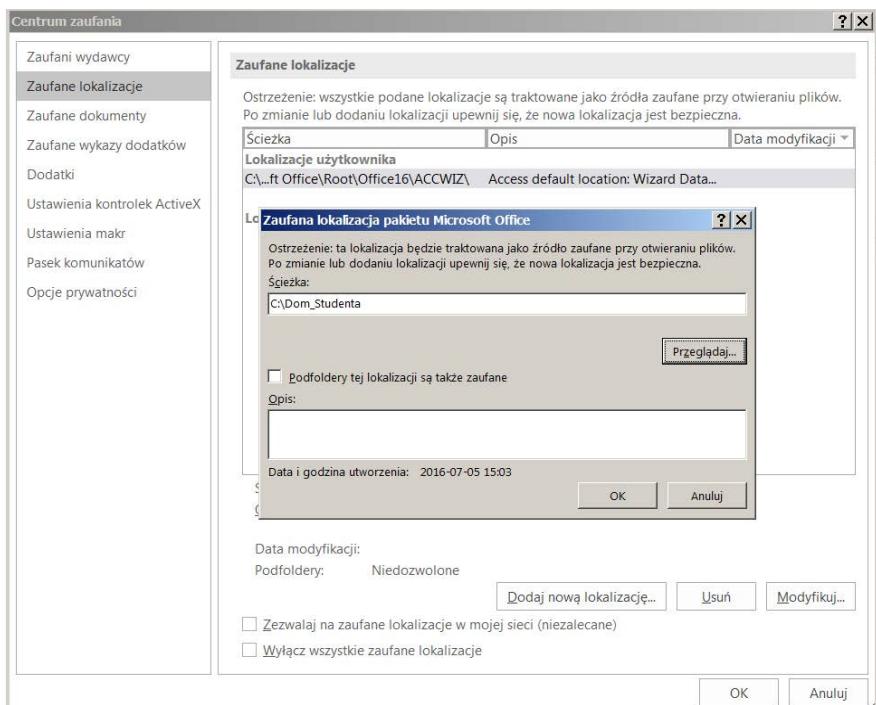
Rys. 1.4. Okno **Opcje** programu **Access**, zakładka **Centrum zaufania**

- W oknie **Centrum zaufania** wybierz zakładkę **Zaufane lokalizacje** (rys. 1.5).



Rys. 1.5. Okno **Centrum zaufania**, zakładka **Zaufane lokalizacje**

- d. W oknie **Centrum zaufania** naciśnij przycisk **Dodaj nową** i za pomocą przycisku **Przeglądaj** wybierz nazwę swojego folderu bazy danych (przykład okienka dialogowego z wprowadzoną ścieżką do folderu **Dom\_Studenta** przedstawia rys. 1.6).
- e. Następnie wybierz **OK** w kolejnych pojawiających się oknach (trzy razy), aż wrócisz do widoku podstawowego **Microsoft Office Backstage**.



Rys. 1.6. Zdefiniowanie **Zaufana lokalizacja pakietu Microsoft Office**

Zaufaną lokalizację można zmodyfikować lub usunąć, wybierając odpowiednie przyciski w oknie **Centrum zaufania**. Zaufaną lokalizację definiujemy jeden raz na stałe, gdy mamy na komputerze uprawnienia administratora. W innym przypadku (na innym koncie użytkownika) po wyłączeniu komputera powyższe ustawienia mogą zostać utracone.

## 1.4. Tworzenie pustej bazy danych

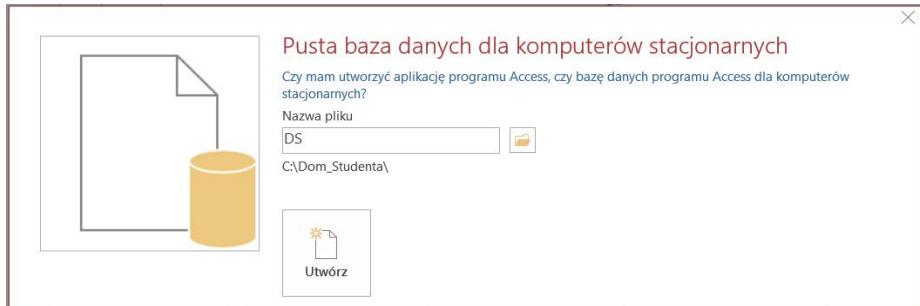
Program **Access** umożliwia tworzenie baz danych dla komputerów stacjonarnych oraz baz danych sieci Web. W tym rozdziale rozpatrzone zostały zasady tworzenia i użytkowania baz danych dla komputerów stacjonarnych.

## Zadanie 1\_3

Utwórz nową pustą bazę danych.

### Wykonanie

- a. W pierwszym oknie programu **Access** wybierz ikonkę **Pusta baza danych dla komputerów stacjonarnych** (rys. 1.1) – zostanie wyświetcone okno dialogowe do wprowadzenia nazwy bazy danych (rys. 1.7).



Rys. 1.7. Okno dialogowe do wprowadzenia nazwy nowej bazy danych

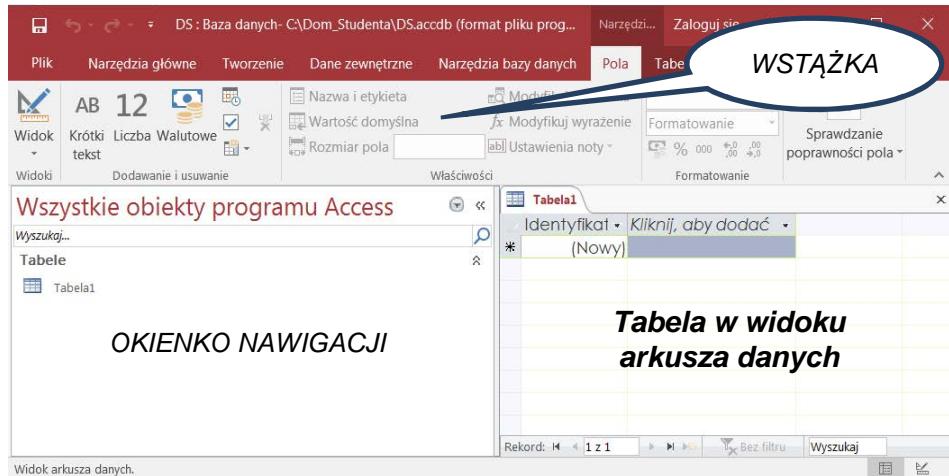
- b. W polu **Nazwa pliku** (rys. 1.7) skasuj **Database1.accdb** i wprowadź nazwę tworzonej bazy danych (na rys. 1.7 jest to nazwa **DS.accdb**). Baza danych zostanie utworzona w domyślnym folderze, który został określony jako zaufana lokalizacja.
- c. Kliknij przycisk **Utwórz** (rys. 1.7) – na ekranie zostanie wyświetlone okno utworzonej bazy danych z pustą tabelą o nazwie **Tabela1** (rys. 1.8).

Tabela wyświetlana jest w widoku arkusza danych. Arkusz pozwala na wprowadzenie danych do tabeli. W ten sposób można zacząć pracę z tabelą w przypadku, gdy ma ona bardzo prostą strukturę. Jednak w dalszej części podróżnika będziemy postępować inaczej, tzn. najpierw tabela zostanie zaprojektowana, a potem wprowadzimy do niej dane.

Na rysunku 1.8 zdefiniowano nazwy obszarów roboczych programu: „wstążka” oraz „okienko nawigacji”. Prezentują one podstawowe elementy środowiska graficznego programu **Access**.

**Wstążka** składa się z zakładek (kart), na których pogrupowano przyciski do wykonania poleceń. W poleceniach na wstążce jest uwzględniany aktywny obiekt, tzn. widok wstążki zmienia się w zależności od wybranego elementu.

Niekiedy może pojawić się potrzeba zwiększenia obszaru roboczego. W tym celu należy zwinąć wstążkę, pozostawiając tylko pasek z kartami poleceń.



Rys. 1.8. Pusta tabela o nazwie **Tabela1** otwarta w widoku arkusza danych

W celu **zamknięcia wstążki** można wybrać z jej menu kontekstowego polecenie **Minimalizuj Wstążkę**. Aby z powrotem **wstążkę wyświetlić**, należy kliknąć dwukrotnie jedną z kart poleceń.

**W okienku nawigacji wyświetlane są wszystkie obiekty znajdujące się w bazie danych.** Okienko to ułatwia porządkowanie obiektów baz danych, a także pełni rolę głównego mechanizmu otwierania tych obiektów i modyfikowania ich projektów. Okienko nawigacji można zminimalizować lub ukryć, nie można go jednak zasłonić przez otwarcie obiektów bazy danych na pierwszym planie.

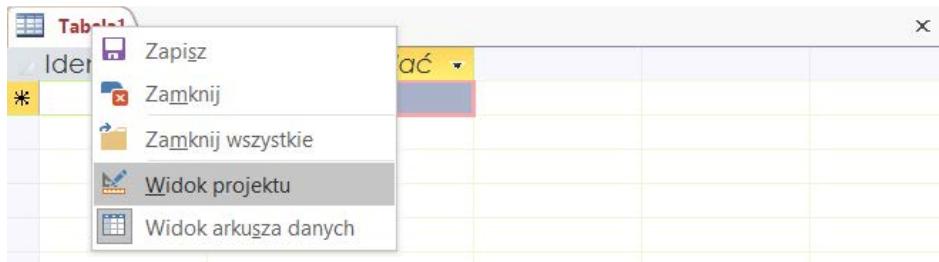
## Zadanie 1\_4

Zamknij tabelę otwartą przez program.

### Wykonanie

Kliknij prawym przyciskiem myszy na napisie **Tabela1** i wybierz polecenie **Zamknij** (rys. 1.9).

Takie menu, wywołane na obiekcie graficznym, w które klikniemy prawym klawiszem myszy, nazywane jest **menu podręcznym** danego obiektu.



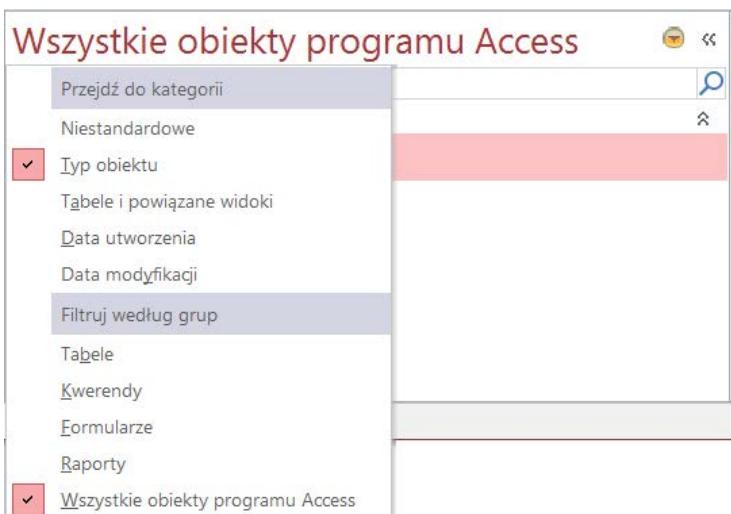
Rys. 1.9. Menu podręczne tabeli

Możemy także w celu zamknięcia tabeli po prostu kliknąć krzyżyk w prawym górnym rogu tabeli.

## Zadanie 1\_5

Rozwiń listę w okienku nawigacji – **Wszystkie obiekty programu Access**, jak na rysunku 1.10.

Zwróć uwagę, że zawartość okienka nawigacji jest podzielona na kategorie i grupy. Użytkownik może wybierać spośród wielu gotowych układów, a także tworzyć własne schematy.



Rys. 1.10. Polecenia okienka nawigacji

Domyślnie nowo utworzone bazy danych korzystają z kategorii **Typ obiektu**, w której poszczególne grupy odpowiadają różnym rodzajom obiektów.

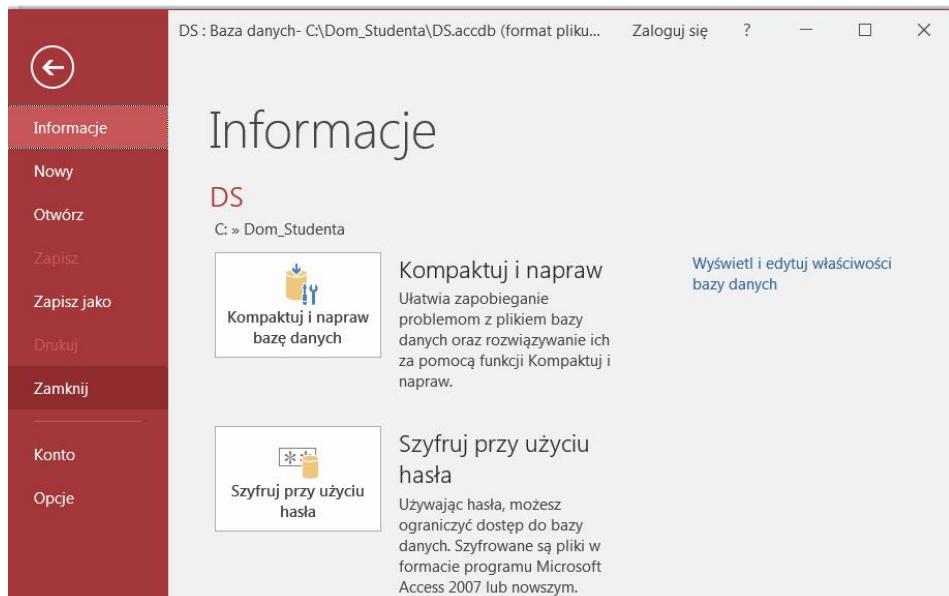
tów baz danych. Najwygodniej pracuje się, gdy zaznaczona jest opcja **Wszystkie obiekty programu Access** (rys. 1.10).

## 1.5. Zamykanie, otwieranie, kompaktowanie bazy danych

Możemy zamknąć otwartą bazę danych, nie zamykając programu **Microsoft Access**. W celu zamknięcia bazy danych wybierz na wstążce polecenie **Plik**, wówczas zostanie wyświetcone okno przedstawione na rysunku 1.11. Następnie wybierz polecenie **Zamknij**. Program **Microsoft Access** pozostanie otwarty, a baza danych zostanie zamknięta.

Aby otworzyć bazę danych, należy w oknie przedstawionym na rysunku 1.11 wybrać polecenie **Otwórz** i wskazać nazwę pliku bazy danych. Jeśli uruchamiamy polecenie **Otwórz** przy już otwartej innej bazie danych, wówczas ta baza zostanie zamknięta.

Po wykonaniu prac związanych z rozbudowywaniem bazy danych dobra praktyka jest naprawienie jej przed zamknięciem. Służy do tego przycisk **Kompaktuj i napraw baza danych** (rys. 1.11).



Rys. 1.11. Widok okna programu zawierającego polecenia do zarządzania bazą danych

## 2. Projektowanie przykładowej tabeli

Tabela jest zbiorem danych dotyczących określonego tematu. W tym rozdziale utworzymy tabelę od podstaw. Oznacza to, że utworzymy ją najpierw w oknie projektu, a następnie wprowadzimy dane.

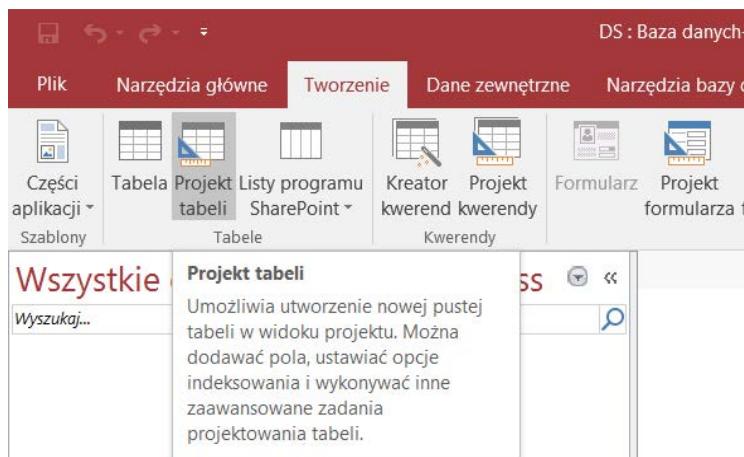
### 2.1. Widok projektu tabeli

#### Zadanie 2\_1

Utwórz pustą tabelę w **widoku projektu**.

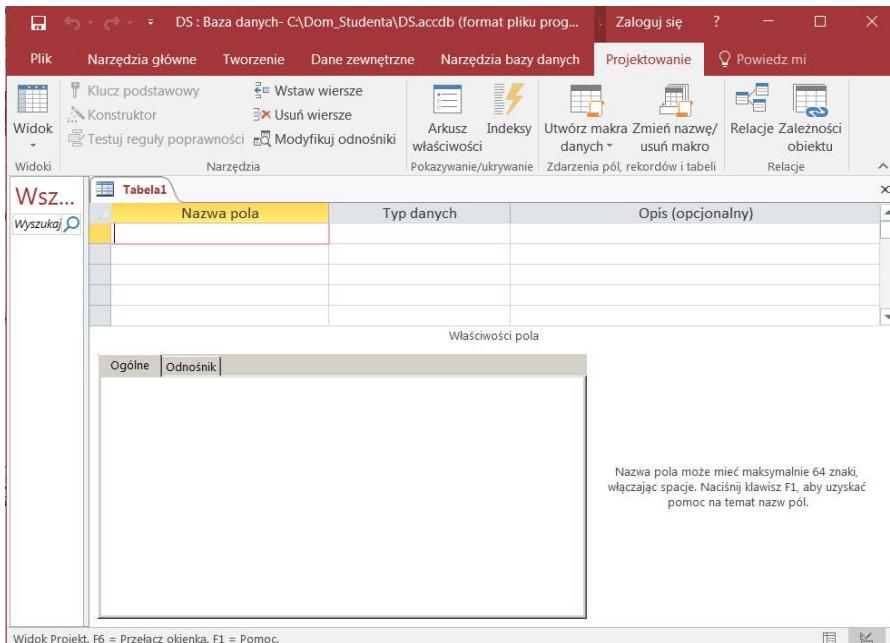
#### Wykonanie

Wybierz na wstążce w zakładce **Tworzenie** ikonkę **Projekt tabeli** (rys. 2.1).



Rys. 2.1. Widok wstążki z zaznaczoną ikonką **Projekt tabeli**

W wyniku wywołania tego polecenia pojawi się pusta tabela w **widoku projektu** (rys. 2.2).



Rys. 2.2. Widok pustej tabeli w **widoku projektu**

Pierwsza kolumna projektu tabeli zawiera nazwy kolumn (pół) tabeli, która ma powstać.

W drugiej kolumnie definiujemy typy danych pól. Typ danych jest cechą pola określającą, jaki rodzaj danych może być w nim przechowywany. Na przykład w polu, którego typem danych jest **KrótkiTekst**, mogą być przechowywane napisy o długości nie przekraczającej 255 znaków. W polu typu **Liczba** mogą być przechowywane tylko dane liczbowe. W kolumnie **Opis** możemy załączyć niezbędne komentarze.

## 2.2. Utworzenie pierwszego pola tabeli

W pierwszej tabeli tworzonej bazy danych będą gromadzone dane osobowe mieszkańców Domu Studenta.

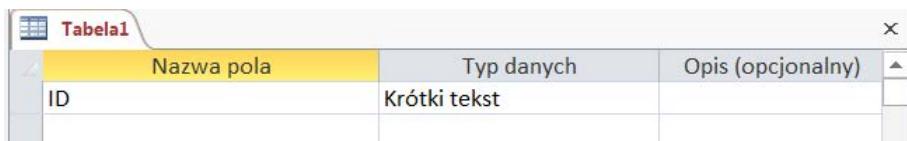
Pierwsza kolumna (pole) tabeli będzie zawierała unikatowe numery identyfikacji studentów, które tworzone są automatycznie.

## Zadanie 2\_2

Utwórz pierwsze pole tabeli typu **Autonumerowanie**, które będzie miało nazwę *ID*.

### Wykonanie

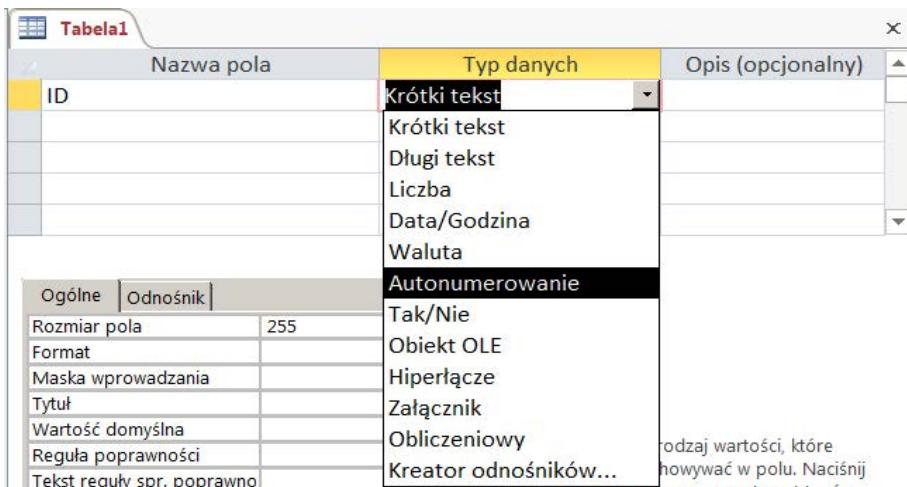
- a. Wprowadź do pierwszej kolumny projektu tabeli nazwę *ID*, a następnie kliknij myszką w kolumnę obok. Pierwszy wiersz projektu tabeli przyjmie postać przedstawioną na rysunku 2.3.



Nazwa pola	Typ danych	Opis (opcjonalny)
ID	Krótki tekst	

Rys. 2.3. Widok projektu tabeli po wprowadzeniu nazwy pierwszego pola

- b. Rozwiń listę typów danych (rys. 2.4), klikając strzałkę w prawym rogu komórki z typem pola i wybierz typ danych **Autonumerowanie**.



Nazwa pola	Typ danych	Opis (opcjonalny)
ID	Autonumerowanie	

Ogólne | Odnośnik

Rozmiar pola: 255  
Format:  
Maska wprowadzania:  
Tytuł:  
Wartość domyślna:  
Reguła poprawności:  
Tekst reguły spr. poprawno:

Rys. 2.4. Wybór typu danych pola

- c. W kolumnie **Opis** projektu tabeli wprowadź: „Numer identyfikacji studenta” (rys. 2.5).

Nazwa pola	Typ danych	Opis (opcjonalny)
ID	Autonumerowanie	Numer identyfikacji studenta

Rys. 2.5. Widok projektu tabeli po wprowadzeniu nazwy, typu danych oraz opisu pierwszego pola tabeli

## 2.3. Klucz podstawowy

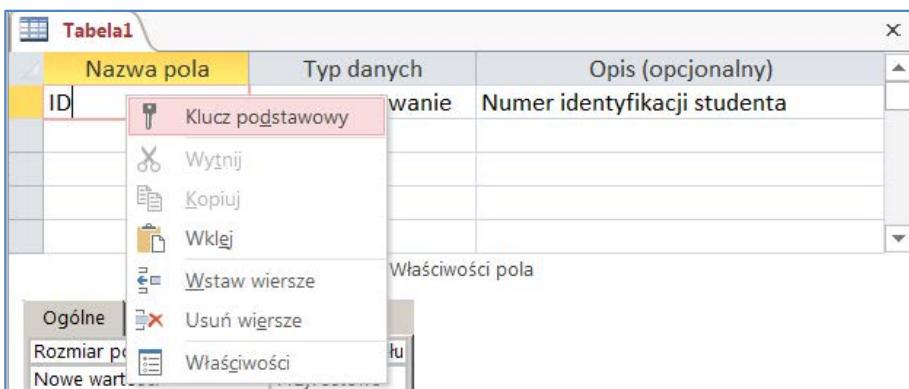
Teraz, gdy projekt naszej tabeli zawiera już treść, należy go zapisać. Nie możemy tego zrobić, dopóki nie zdefiniujemy klucza podstawowego. Służy on do jednoznacznej identyfikacji rekordów tabeli.

### Zadanie 2\_3

Określ pole *ID* jako **klucz podstawowy**.

#### Wykonanie

- Zaznacz pole *ID*, klikając wskaźnikiem myszy na szarej krawędzi po lewej stronie tabeli.
- Kliknij przycisk **Klucz podstawowy** na pasku narzędzi lub w menu podręcznym pola *ID* (na rys. 2.6 zostało przedstawione menu podręczne pola *ID*).



The screenshot shows the Microsoft Access 'Tabela1' table design view. A context menu is open over the 'ID' column header. The menu items are: Wytnij (Cut), Kopiuj (Copy), Wklej (Paste), Wstaw wiersze (Insert Rows), Usuń wiersze (Delete Rows), and Właściwości (Properties). The 'Klucz podstawowy' (Primary Key) option is highlighted with a red rectangle. The 'Właściwości' (Properties) option is also visible at the bottom of the menu.

Rys. 2.6. Menu podręczne pola *ID* z zaznaczonym kluczem podstawowym

Aby pole nadawało się na klucz podstawowy, musi posiadać kilka cech. Po pierwsze, powinno jednoznacznie identyfikować każdy wiersz. Po drugie, nie może być puste. Po trzecie, powinno rzadko (najlepiej nigdy) ulegać zmianie.

Często unikatowy numer identyfikacyjny, taki jak: identyfikator, numer seryjny, kod adresu zamieszkania, PESEL lub NIP, wykorzystywany jest w tabeli jako klucz podstawowy.

Możemy także zdefiniować jako klucz podstawowy zestaw pól, ale takie sytuacje nie są teraz rozważane.

## 2.4. Zapisywania tabeli

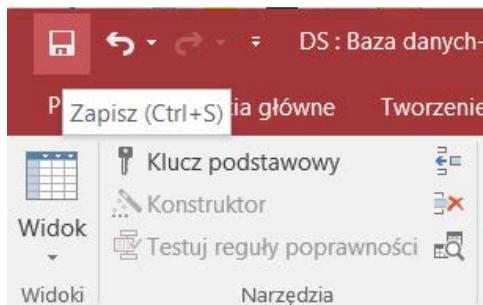
Każda operacja wykonywana na tabelach i na danych może być zrealizowana w programie **Access** na wiele sposobów. W kolejnym przykładzie i w dalszych zawartych w podręczniku zawsze będzie proponowany tylko jeden (lub dwa) z nich w celu najprostszej realizacji zadania.

### Zadanie 2\_4

Zapisz tabelę po raz pierwszy.

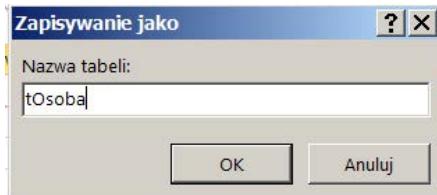
#### Wykonanie

W celu zapisania projektu tabeli skorzystaj z przycisku **Zapisz** na pasku narzędzi **Szybki dostęp** (rys. 2.7) lub wybierz go z menu podręcznego tabeli (rys. 1.9).



Rys. 2.7. Widok okna z ikoną do zapisywania obiektów

W oknie dialogowym zapisywania tabeli skasuj nazwę domyślną, wpisz nazwę: **tOsoba** (rys. 2.8) i zatwierdź przyciskiem **OK**.



Rys. 2.8. Widok okna dialogowego do zapisywania tabeli

Nazwy tabel, które zostaną w dalszej części utworzone, będą na początku zawierały literę „t”, raportów – literę „r”, formularzy – literę „f”, a kwerend – literę „k”.

Po zapisaniu tabeli jej nazwa pojawi się w okienku nawigacji. Teraz można otworzyć menu podrzczne i obejrzeć, jakie czynności można wykonać z utworzoną tabelą (rys. 2.9).

Nazwa pola	Typ danych
ID	Autonumerowanie

Rys. 2.9. Widok menu podrzcznego tabeli **tOsoba** wywołanego w okienku nawigacji

## 2.5. Dodanie pozostałych pól do projektu tabeli tOsoba

Na rysunku 2.10 przedstawiono widok projektu tabeli **tOsoba**, która dalej zamierzamy stworzyć.

	Nazwa pola	Typ danych
!	ID	Autonumerowanie
	Nazwisko	Krótki tekst
	Imiona	Krótki tekst
	Album	Liczba
	Płeć	Krótki tekst
	PESEL	Krótki tekst
	Imię_ojca	Krótki tekst
	Imię_matki	Krótki tekst
	Nazwisko_rodowe_matki	Krótki tekst
	e-mail	Krótki tekst
	Telefon	Krótki tekst
	Adres	Krótki tekst
	Miejscowość	Krótki tekst
	Województwo	Krótki tekst
	Kod_pocztowy	Krótki tekst
	Kraj	Krótki tekst
	Foto	Obiekt OLE
	Dok	Załącznik
	Uwagi	Długi tekst
	Strona_WWW	Hiperłącze

Rys. 2.10. Nazwy i typy pól tabeli **tOsoba**

### Zadanie 2\_5

Wprowadź pozostałe pola do projektu tabeli **tOsoba**, a także wybierz odpowiednie typy danych, wzorując się na rysunku 2.10. Następnie zapisz tabelę.

## 2.6. Zdefiniowanie właściwości pól tabeli

W dolnej części okna projektu tabeli pod listą pól widoczne są dwie zakładki: **Ogólne** i **Odnośnik**. Zamieszczone w nich informacje zawsze dotyczą pola, które jest w danej chwili zaznaczone. Wszystkie te opcje będziemy nazywać **właściwościami pola**.

Jeśli zaznaczone jest przykładowo pole *ID*, to zakładka **Ogólne** wyświetla właściwości tego pola tak, jak przedstawiono na rysunku 2.11.

Ogólne	Odnośnik
Rozmiar pola	Liczba całk. długa
Nowe wartości	Przyrostowo
Format	
Tytuł	
Indeksowane	Tak (Bez duplikatów)
Wyrównanie tekstu	Ogólne

Rys. 2.11. Właściwości pola *ID*

Dla każdego typu danych program proponuje domyślne wartości właściwości pola. W dalszej części tego rozdziału niektóre z tych właściwości zmienimy, uzupełniając tym samym projekt tworzonej tabeli **tOsoba**.

### Właściwość ROZMIAR POLA

Pola tekstowe i liczbowe posiadają właściwość „Rozmiar pola”.

Dane typu **Krótki tekst** są to ciągi znaków, których liczba nie może przekraczać 255. Dane typu **Długi tekst** nie mogą zawierać więcej niż 63999 znaków.

Jeśli zaznaczymy pole tekstowe (np. *Nazwisko*), to w dolnej części okna na zakładce **Ogólne** w wierszu **Rozmiar Pola** zobaczymy maksymalną liczbę znaków, które można do niego wpisać – 255. Możemy to zmienić, wpisując inną liczbę, przykładowo 50.

### Zadanie 2\_6

Zmniejsz rozmiary tekstowych pól *Płeć*, *PESEL*, *Kod\_pocztowy*, *Kraj*, *Miejscowość*, *Województwo*, zmieniając liczbę ustawioną domyślnie w wierszu **Rozmiar pola**.

Pole typu **Liczba** też posiada właściwość **Rozmiar pola**, którą definiujemy wybierając z listy rozwijalnej. Liczbowe pole może mieścić się w jednym, dwóch, czterech, ośmiu lub szesnastu bajtach.

### Właściwość FORMAT

Właściwość **Format** wpływa na sposób wyświetlania danych. Microsoft Access pozwala na korzystanie ze wstępnie zdefiniowanych formatów oraz

tworzonych przez użytkownika. W praktyce najczęściej używane są formaty dla pól typu **Waluta**, **Data/godzina**, **TAK/NIE**.

Postać formatów zależy od ustawień kraju/regionu (można je określić w **Panelu sterowania** systemu **Windows**). W programie **Access** wyświetlane są formaty właściwe dla wybranego kraju/regionu. Przykładowo, jeśli w *Panelu sterowania* systemu *Windows* zaznaczono lokalizację „Polska”, to wartość 12,34 w formacie **Waluta** jest wyświetlana jako 12,34 zł.

Na rysunku 2.12 zaprezentowano listę standardowych formatów dla danych liczbowych.

W tworzonej tabeli można pozostawić wiersz **Format** bez zmiany.

Ogólne	Odnośnik
Rozmiar pola	Liczba całk. długa
Format	Liczba ogólna
Miejsca dziesiętne	Liczba ogólna
Maska wprowadzania	Walutowy
Tytuł	Euro
Wartość domyślna	Stałoprzecinkowy
Reguła poprawności	Standardowy
Tekst reguły spr. poprawno	Procentowy
Wymagane	Wykładniczy
Indeksowane	Nie
Wyrównanie tekstu	Ogólne

Rys. 2.12. Lista formatów dla pola typu **Liczba całk. długa**

## Właściwość MASKA WPROWADZANIA

Aby ułatwić pracę osobie wprowadzającej dane do tabeli, często wykorzystuje się właściwość **Maska wprowadzania**.

## Zadanie 2\_7

Utwórz za pomocą kreatora maskę dla pola *Kod\_pocztowy*.

### Wykonanie

- Zaznacz pole *Kod\_pocztowy*.
- W zakładce **Ogólne** w prawym rogu wiersza **Maska wprowadzania** kliknij na przycisku (rys. 2.13).



Rys. 2.13. Wiersz z przyciskiem **Kreatora masek wprowadzania**

- Gdy pojawi się komunikat z pytaniem dotyczącym zapisywania tabeli, wybierz **Tak**.
- W oknie **Kreatora masek wprowadzania** wybierz maskę **Kod pocztowy**, a następnie wybierz **Dalej**.
- W kolejnym oknie wybierz **Dalej**, a w następnym wybierz opcję **Z symbolami w masce i Zakończ**.

## Zadanie 2\_8

Utwórz maskę niestandardową dla pola PESEL, aby do pola nie można było wprowadzić innego znaku niż cyfra. Cyfr tych powinno być 11.

### Wykonanie

Należy tu skorzystać z możliwości samodzielnego utworzenia masek. Numer PESEL zawiera 11 cyfr. Do maski wprowadzania należy wprowadzać symbole zastępujące cyfry. Takim symbolem według **Access** jest 0, więc do wiersza **Maska wprowadzania** wprowadź: **00000000000**, tak jak przedstawiono na rysunku 2.14.



Rys. 2.14. Maska wprowadzania dla pola PESEL

## Właściwość TYTUŁ

Każde pole posiada nazwę i tytuł. Nazwa pola widoczna jest w projekcie tabeli, tytuł zaś w arkuszu danych.

Zazwyczaj nazwa pola jest skrótem, nie zawiera znaków diakrytycznych oraz spacji. Takie nazewnictwo upraszcza pracę przy tworzeniu aplikacji bazodanowej lub przy dalszym eksportie bazy danych. Jednak w formularzach i raportach (jak i w widoku arkusza danych tabeli i kwerendy) utworzonych w programie **Access** lepiej będzie się prezentować właściwa nazwa pola tabeli. Będzie ona automatycznie pobierana z wiersza **Tytuł**.

Jeśli nic nie zostanie wprowadzone do wiersza **Tytuł**, to będzie on dokładnie taki sam jak odpowiednia nazwa pola.

### Zadanie 2\_9

Zdefiniuj właściwość Tytuł dla tych pól, których nazwa zawiera znak podkreślenia (tzn. pól *Imię\_matki*, *Nazwisko\_rodowe\_matki*, *Kod\_pocztowy*). Jako **Tytuł** wpisz ten sam tekst, ale znak podkreślenia zamień na spację.

### Wykonanie

Wykonaj zadanie, wzorując się na właściwości **Tytuł** pola *Imię\_ojca* zaprezentowanej na rysunku 2.15.

Ogólne	Odnośnik
Rozmiar pola	50
Format	
Maska wprowadzania	
Tytuł	Imię ojca

Rys. 2.15. Właściwość **Tytuł** pola *Imię\_ojca*

### Utworzenie listy możliwych wartości pola za pomocą Kreatora odnośników

Zakładka **Odnośnik**, podobnie jak zakładka **Ogólne**, zawiera różne opcje. Właśnie tam można zobaczyć listę możliwych wartości pola (odnośników), pod warunkiem, że taka lista została sporządzona. Określona w zakładce **Odnośnik** lista wartości służy do wybierania jednej z nich przy wprowadzeniu danych do tabeli. Rozważymy dalej taką możliwość, definiując

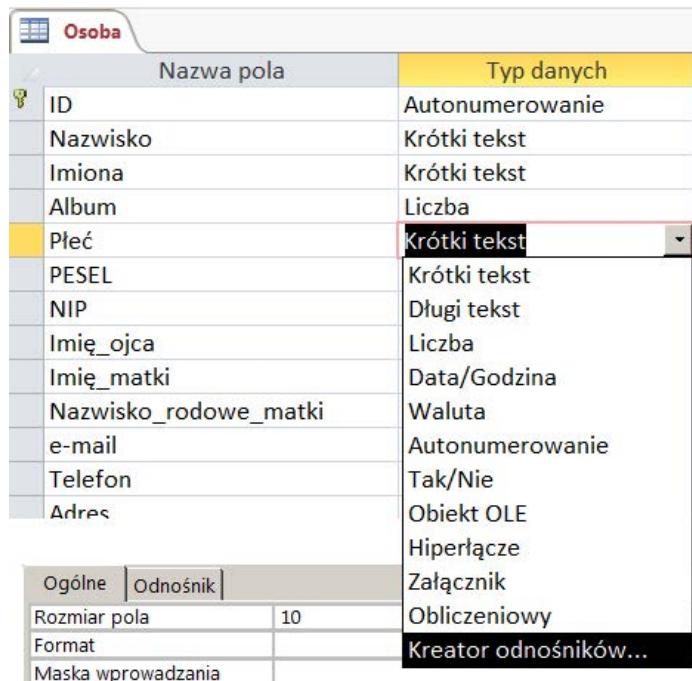
ając listę wartości pola *Płeć*. Będziemy tu posugiwać się **Kreatorem odnośników**.

## Zadanie 2\_10

Zdefiniuj listę wartości dla pola *Płeć*, wprowadzając dwie wartości tekstowe: „Mężczyzna” oraz „Kobieta”.

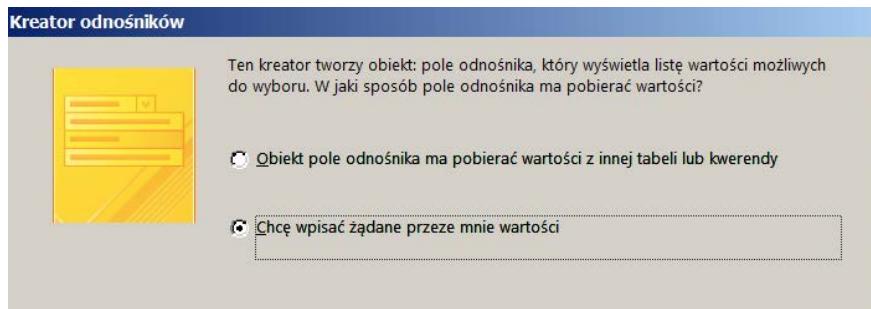
### Wykonanie

- Zaznacz w projekcie tabeli wiersz zawierający pole *Płeć*.
- W kolumnie **Typ danych** kliknij strzałkę i wybierz **Kreator odnośników**, jak pokazano na rysunku 2.16.



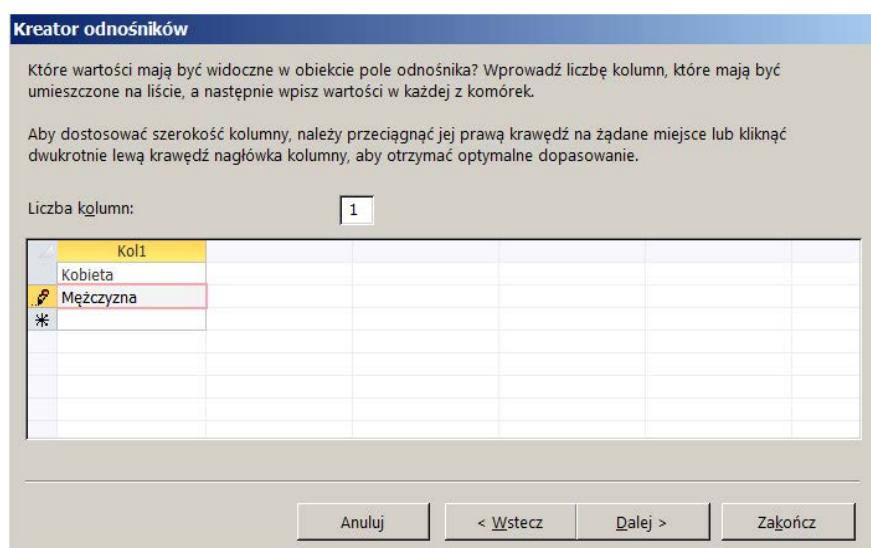
Rys. 2.16. Wybór polecenia **Kreator odnośników**

- W pierwszym oknie dialogowym **Kreatora odnośników** (rys. 2.17) wybierz opcję **Chcę wpisać żądane przede mnie wartości**, a następnie kliknij przycisk **Dalej**.



Rys. 2.17. Wybór opcji „Chcę wpisać żądane przeze mnie wartości”

- d. W kolejnym oknie dialogowym wprowadź do pierwszej kolumny wartości pola *Płeć* (rys. 2.18) i kliknij **Dalej**. W ostatnim oknie wybierz **Zakończ**.



Rys. 2.18. Lista wartości pola *Płeć* w oknie dialogowym **Kreatora odnośników**

Wprowadzone wartości pojawią się w zakładce **Odnośnik** (rys. 2.19).

Jeśli chcemy, aby użytkownik nie mógł dodawać do listy nowych danych (a właśnie tak jest w tym przypadku), możemy zmienić opcję **Ogranicz do listy**, wybierając opcję **Tak** (rys. 2.19).

**tOsoba**

Nazwa pola	Typ danych
ID	Autonumerowanie
Nazwisko	Krótki tekst
Imiona	Krótki tekst
Album	Liczba
Płeć	Krótki tekst

Właściwości pola

Ogólne	Odnośnik
Type kontrolki	Pole kombi
Typ źródła wierszy	Lista wartości
Źródło wierszy	"Kobieta";"Mężczyzna"
Kolumna powiązana	1
Liczba kolumn	1
Nagłówki kolumn	Nie
Szerokości kolumn	2.54cm
Liczba wierszy listy	16
Szerokość listy	2.54cm
Ogranicz do listy	Tak
Zezwalaj na wiele wartości	Nie
Zezwalaj na edycję listy war	Tak

Rys. 2.19. Lista wartości pola Płeć w zakładce **Odnośnik** projektu tabeli

### Wartość domyślna

Z pomocą właściwości **Wartość domyślna** można określić wartość, która przy dodawaniu nowego rekordu będzie wprowadzana do pola automatycznie. Łatwo jest zauważyć, że do pól liczbowych program automatycznie dodaje wartość domyślną 0. Nie zawsze jednak jest ona potrzebna – wtedy ją usuwamy.

## Zadanie 2\_11

Zdefiniuj wartość domyślną dla pola *Płeć*.

Skasuj wartość domyślną 0 dla pola *Album*, która była dodana przez program automatycznie.

### Wykonanie

Jako wartość domyślną wybieramy wartość, która będzie występować częściej przy dodawaniu rekordów do tabeli. Przykład wprowadzenia wartości domyślnej dla pola *Płeć* przedstawia rysunek 2.20. W tym przypadku zakładamy, że w tabeli będzie więcej danych o kobietach.

Ogólne	Odnośnik
Rozmiar pola	9
Format	
Maska wprowadzania	
Tytuł	
Wartość domyślna	"Kobieta"

Rys. 2.20. Wartość domyślna pola Płeć

Aby skasować wartość domyślną pola *Album*, zaznacz to pole i na zakładce **Ogólne** w wierszu **Wartość domyślna** usuń 0.

### Reguła poprawności oraz komunikat o błędzie

W celu zmniejszenia liczby możliwych błędów przy pracy z danymi definiujemy konkretne ograniczenia już na etapie projektowania tabeli. Ograniczenia te nazywają się regułami poprawności. Służą one do kontrolowania danych, które użytkownicy wprowadzają do pola tabeli.

## Zadanie 2\_12

Utwórz regułę poprawności dla pola *Album*, wychodząc z założenia, że wartość w tym polu nie może przekroczyć liczby 999 999. Dodaj tekst komunikatu o błędzie, który pojawi się, gdy użytkownik wprowadzi liczbę większą od 999 999.

### Wykonanie

- Zaznacz w projekcie tabeli **tOsoba** wiersz zawierający pole *Album*.
- W zakładce **Ogólne** wpisz do wiersza **Reguła poprawności** wyrażenie: <1 000 000, tak jak pokazano na rysunku 2.21.

Ogólne	Odnośnik
Rozmiar pola	Liczba całk. długa
Format	Liczba ogólna
Miejsca dziesiętne	Auto
Maska wprowadzania	
Tytuł	
Wartość domyślna	
Reguła poprawności	<1000000
Tekst reguły spr. poprawno	Numer albumu nie może przekraczać 999 999

Rys. 2.21. Reguła poprawności dla pola *Album*

- c. Zapisz w wierszu **Tekst reguły spr. poprawności** zdanie, które będzie wyświetlane w postaci komunikatu w przypadku wprowadzenia liczby większej od 999 999 (rys. 2.21).

Regułę poprawności możemy utworzyć dla danych wszystkich typów, z wyjątkiem typów **Autonumerowanie**, **Obiekt OLE** i **Załącznik**.

### Właściwość WYMA GANE

W bazie danych większości pól nie można pozostawiać pustych. Jeśli chcemy, aby pewne pole zawsze zawierało wartość, definiujemy jego właściwość **Wymagane** równą **Tak** (rys. 2.22).



Rys. 2.22. Ustawienie opcji **Tak** we właściwości **Wymagane**

## Zadanie 2\_13

---

Zdefiniuj pola *Nazwisko*, *Imiona*, *Album* jako wymagane.

Zalecane jest, aby wszystkie pola tabeli były wymagane. Zawsze można to zrealizować, definiując wartości domyślne.

### Właściwość INDEKSOWANE

Wiedząc, że w tabeli często będzie wykonywane wyszukiwanie danych według pewnego pola, tworzymy dla niego indeks. Może to być indeks unikatowy lub nieunikatowy.

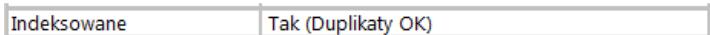
## Zadanie 2\_14

---

Utwórz indeks nieunikatowy dla pola *Nazwisko*.

### Wykonanie

- Zaznacz w projekcie tabeli **tOsoba** wiersz zawierający pole *Nazwisko*.
- W zakładce **Ogólne** w wierszu indeksowane wybierz z listy rozwijanej **Tak (Duplikaty OK)** (rys. 2.23).



Rys. 2.23. Ustawienie indeksowania dla pola *Nazwisko*

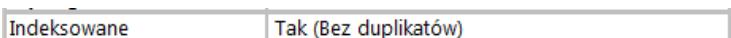
Utworzenie indeksu unikatowego sprawia, że do danego pola nie można będzie wprowadzić zduplikowanych wartości. W tabeli **tOsoba** takiego rodzaju indeksy możemy utworzyć na podstawie pól *PESEL* i *Album*.

## Zadanie 2\_15

Utwórz indeksy unikatowe dla pól: *PESEL* i *Album*.

### Wykonanie

- a. Zaznacz w projekcie tabeli **tOsoba** wiersz zawierający pole *PESEL*.
- b. W zakładce **Ogólne** w wierszu **Indeksowane** wybierz z listy rozwijanej **Tak (Bez duplikatów)** (rys. 2.24).



Rys. 2.24. Zdefiniowanie indeksów dla pól *PESEL* i *Album*

## 2.7. Wprowadzenie danych w widoku arkusza danych tabeli

Po zdefiniowaniu wszystkich właściwości mamy gotowy projekt tabeli. Tabela wciąż jest pusta i należy wypełnić ją danymi. Możemy to wykonać na wiele sposobów. Jednym z nich jest wprowadzenie danych w widoku arkusza danych tabeli. Przećwiczmy to, wykonując zadanie 2\_16.

## Zadanie 2\_16

Zapisz ponownie tabelę **tOsoba** i przejdź do **widoku arkusza danych**.

### Wykonanie

Skrajny przycisk w lewym rogu wstążki o nazwie **Widok** pozwala na zmianę wyglądu okna pracy z tabelą.

Jeżeli projekt tabeli jest otwarty, to przycisk **Widok** wygląda tak, jak na rysunku 2.25. Aby przejść do widoku arkusza danych, wystarczy kliknąć tę ikonkę. Nastąpi wówczas przejście do widoku arkusza danych, a postać ikonki zmieni się. W tym oknie przycisk **Widok** będzie miał postać jak na rysunku 2.26.

Jeśli tabela jest otwarta w widoku projektu, to ikonka **Widok** ma postać przedstawioną na rysunku 2.25. Po kliknięciu na niej zobaczymy tabelę w widoku arkusza danych, a ikonka będzie wyglądać jak na rysunku 2.26.



Rys. 2.25. Przycisk **Widok** w oknie projektu tabeli



Rys. 2.26. Przycisk **Widok** w oknie arkusza danych tabeli

Jeśli projekt tabeli **tOsoba** został już zamknięty, wybierz w okienku nawigacji, w menu podręcznym tabeli polecenie **Otwórz**, a otwarty zostanie widok arkusza danych tabeli.

Przykład fragmentu arkusza danych tabeli **tOsoba** zamieszczono na rysunku 2.27.

	ID	Nazwisko	Imiona	Album	Płeć	PESEL
[+]	1	Kulesza	Paweł	34567	Mężczyzna	9011221231
[+]	2	Poniatowska	Maria	87890	Kobieta	9409098976
[+]	3	Gólbiecki	Krzysztof	88900	Mężczyzna	9603037557
[+]	4	Kowalski	Marek	89891	Mężczyzna	9012053439
[+]	5	Malinowska	Irena	89892	Kobieta	9302052232

Rys. 2.27. Widok arkusza danych tabeli **tOsoba**

Wprowadzenie danych w oknie arkusza danych wymaga standardowej wiedzy o narzędziach pakietu *Microsoft Office*. Dalej wystarczy umiejętność pracy z klawiaturą oraz przyciskami: **Kopiuj**, **Wytnij**, **Wklej**.

Należy nadmienić, że wszystkie przykłady i ćwiczenia zaproponowane w dalszej części książki można wykonać tylko wtedy, gdy baza danych będzie zawierała prawidłowe dane. Dane te należy wymyślić, ale nie w sposób dowolny. Na przykład numer PESEL posłuży nam do pewnych operacji na danych osobowych, więc należy wpisywać przykładowe numery zgodnie z przyjętymi ogólnie zasadami.

Wprowadzenie danych nie jest czynnością, którą można zlekceważyć lub trywializować. W dalszej części będziemy budować specjalne formularze, aby można było wprowadzić właściwe dane do pewnych tabel.

Podsumowując, im więcej wprowadzisz danych testowych, tym łatwiej uczyć się baz danych. Należy więc potraktować tę sprawę poważnie. Im więcej danych testowych wprowadzisz, tym lepiej opanujesz naukę tworzenia bazy danych.

## Zadanie 2\_17

Wprowadź do tabeli dane testowe:

- przechodząc kolejno od wiersza do wiersza, wprowadź do tabeli 3 wiersze przykładowych danych; gdy przechodzimy do następnej linijki, poprzedni wiersz automatycznie zostaje zapisany do bazy danych;
- nie wypełniaj pól *Foto* i *Dok.*

Po wpisaniu danych testowych zamknij tabelę (krzyżyk w prawym rogu tabeli).

## 3. Najprostszy formularz

Formularz jest podstawowym obiektem interfejsu graficznego użytkownika bazy danych. Można go używać między innymi do wprowadzania lub edytowania danych w tabeli oraz do wyświetlania tych danych. Zapoznamy się z nim, tworząc formularz do wprowadzenia i przeglądu danych tabeli **tOsoba**.

### 3.1. Generowanie formularza do pracy z danymi tabeli

## Zadanie 3\_1

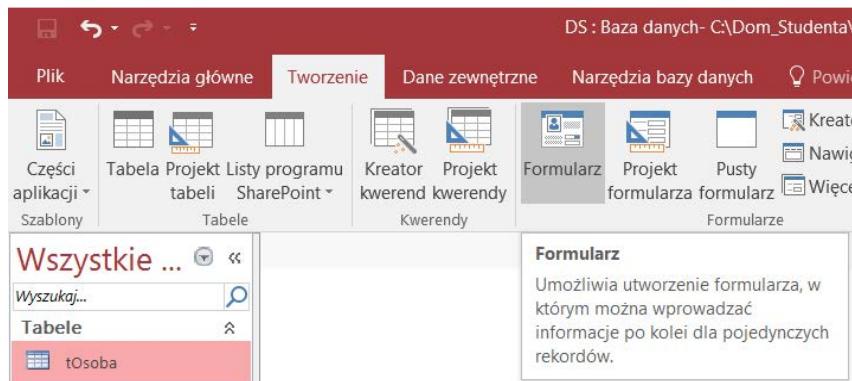
Zastosuj ikonkę wstążki do automatycznego utworzenia formularza, który pozwoli na wykonywanie dowolnych operacji na danych tabeli **tOsoba**. Zapisz formularz. Nadaj formularzowi nazwę **fOsoba**.

### Wykonanie

- a. Zaznacz w **Okienku nawigacji** nazwę tabeli **tOsoba**.

- b. Wybierz na wstążce zakładkę **Tworzenie**, a następnie ikonkę **Formularz** (rys. 3.1). Formularz zostanie utworzony automatycznie i będzie przedstawiony w **Widoku układu** (rys. 3.2), o czym informują pomarańczowe krawędzie pola *ID*. W formularzu wyświetlono dane wprowadzone do tabeli **tOsoba**. Nazwy pól przedstawione tak, jak je zdefiniowano we właściwościach **Tytuł** (nazwy pól *Imię\_ojca*, *Imię\_matki*, *Kod\_pocztowy*, *Strona\_WWW* nie zawierają spacji).

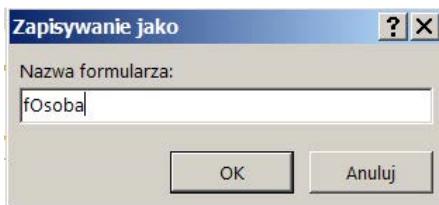
- c. Zapisz formularz za pomocą przycisku (tak jak w przypadku tabeli) lub polecenia **Zapisz** menu podręcznego formularza. Widok okna zapisywania utworzonego formularza przedstawia rysunek 3.3.



Rys. 3.1. Widok zaznaczonej w okienku nawigacji tabeli **tOsoba** oraz wstążki z wybraną ikonką **Formularz**

ID	1	Adres	ul. Studzienn
Nazwisko	Kulesza	Miejscowość	Supraśl
Imiona	Paweł	Województwo	podlaskie
Album	34567	Kod pocztowy	16-456
Płeć	Mężczyzna	Kraj	Polska
PESEL	90112212345	Foto	
Imię ojca	Czesław	Dok	

Rys. 3.2. Widok formularza **fOsoba** natychmiast po utworzeniu



Rys. 3.3. Widok okna zapisu stworzonego formularza

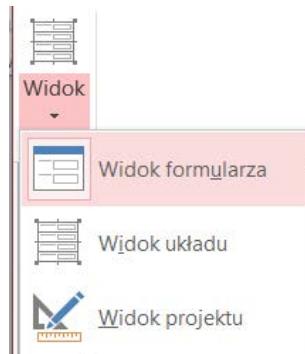
### 3.2. Widoki formularza

Każdy formularz może być wyświetlony przez program **Access** w jednej z trzech postaci: w **widoku formularza**, w **widoku układu**, w **widoku projektu**.

W momencie kiedy utworzony zostanie nowy formularz, jest on przedstawiony przez program w **widoku układu**. **Widok projektu** i **widok układu** są przeznaczone dla projektanta aplikacji bazodanowej.

Natychmiast po wygenerowaniu formularza jest on przedstawiany w **widoku układu** (rys. 3.2). W tym widoku możemy łatwo zmienić szerokość i wysokość pól. Wystarczy „pociągnąć” za żółtą ramkę pola w potrzebnym kierunku. Tak samo możemy zmienić treść i postać napisów, używając standardowych przycisków umieszczonych na wstążce w zakładce **Narzędzia główne**.

Przejście między różnymi widokami formularza zapewnia ikonka **Widok** umieszczona na wstążce w zakładce **Narzędzia główne** (rys. 3.4).



Rys. 3.4. Wybór widoku formularza

Użytkownik bazy danych zawsze widzi formularz w **widoku formularza**. Tylko w **widoku formularza** możemy wprowadzić nowe dane. Po utworzeniu formularza należy przejść do widoku formularza, jeśli mamy zamiar przeglądać lub dodawać nowe dane.

## Zadanie 3\_2

---

Przejdz od **widoku układu** do **widoku formularza fOsoba**.

---

### 3.3. Pasek nawigacji

Zbudowany formularz **fOsoba** jest narzędziem do przeglądania, wprowadzenia i usunięcia danych tabeli **tOsoba**. Za pomocą **paska nawigacji**, który jest ulokowany w dolnej części formularza, możemy nawigować między rekordami tabeli **tOsoba** (rys. 3.5).



Rys. 3.5. Widok paska nawigacji formularza

Można sprawdzić, jak działają strzałki na pasku narzędzi, klikając po koleją każdą z nich:

- strzałka otworzy kolejny rekord formularza;
- strzałka spowoduje powrót do poprzedniego rekordu;
- za pomocą strzałki przejdziemy do ostatniego rekordu tabeli;
- strzałka pozwoli wrócić na początek tabeli.

Jeśli naciśniemy strzałkę na ostatnim rekordzie, wyświetli się nam okno do wprowadzenia nowego rekordu. W tym celu możemy także skorzystać z przycisku .

### 3.4. Zastosowanie formularza do wprowadzenia danych

## Zadanie 3\_3

---

Dodaj jeszcze 3 rekordy do tabeli **tOsoba**. Wykorzystaj do tego formularz **fOsoba**, pracując w **widoku formularza**.

Zamknij formularz, otwórz tabelę **tOsoba** w widoku arkusza danych i obejrzyj wprowadzone 6 rekordów danych.  
Zamknij tabelę **tOsoba**.

## Wykonanie

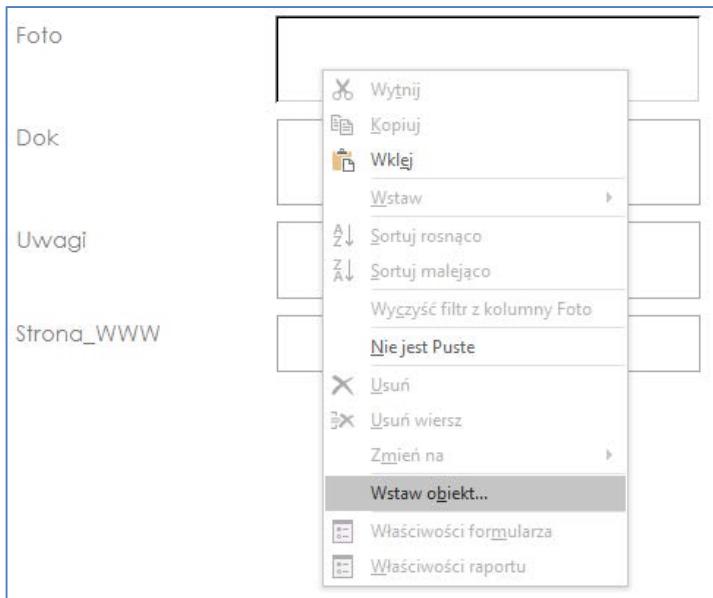
Dane tekstowe i liczbowe wprowadzaj na tych samych zasadach, jak wcześniej były wprowadzane do tabeli. Do kolejnych rekordów przechodź za pomocą strzałek na pasku nawigacji.

## Zadanie 3\_4

Otwórz formularz **fOsoba** w widoku **formularza** i wprowadź do pola *Foto* jednego z rekordów plik o rozszerzeniu *BMP*.

## Wykonanie

W widoku formularza **fOsoba** skorzystaj z menu podręcznego pola *Foto* (rys. 3.6). Wybierz polecenie **Wstaw obiekt**. Dalej wybierz lub stwórz plik o rozszerzeniu *BMP*.



Rys. 3.6. Menu podręczne pola *Foto*

Pole *Dok* ma typ **Załącznik**. Możemy w takim polu dodać jeden lub więcej plików — dokumentów, prezentacji, obrazów itp. Objętość załączonych danych może maksymalnie wynosić dwa gigabajty (maksymalny rozmiar w przypadku bazy danych programu **Access**). Pojedyncze pliki nie mogą przekraczać rozmiaru 256 megabajtów. Załączone pliki można zapisywać w lokalizacjach na dysku twardym lub w sieci.

## Zadanie 3\_5

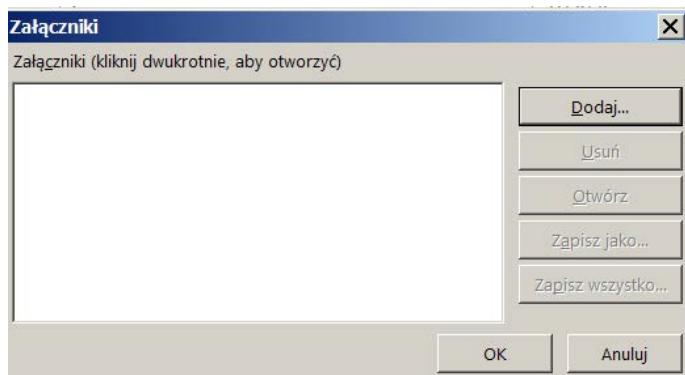
---

Otwórz formularz **fOsoba** w **widoku formularza** i wprowadź dowolny plik do pola *Dok* jednego z rekordów. Zamknij formularz.

---

### Wykonanie

W **widoku formularza fOsoba** kliknij dwukrotnie pole *Dok* i skorzystaj z przycisków okienka **Załączniki** (rys. 3.7), które się wyświetli.



Rys. 3.7. Widok okna **Załączniki**

## 3.5. Formularz dzielony

Wykonany wcześniej formularz **fOsoba** był formularzem **kolumnowym**. **Access** pozwala na stworzenie innych rodzajów formularzy. Jeden z nich to **formularz dzielony**. Formularz ten udostępnia jednocześnie dwa widoki danych — widok formularza i widok arkusza danych. Te dwa widoki są połączone z tym samym źródłem danych i zawsze są ze sobą zsynchronizowane. Zaznaczenie pola w jednej części formularza powoduje zaznaczenie tego samego pola w drugiej części formularza.

## Zadanie 3\_6

Utwórz formularz dzielony do pracy z rekordami tabeli **tOsoba**.

### Wykonanie

- Upewnij się, że zamknięta jest tabela **tOsoba** i formularz **fOsoba**.
- Zaznacz tabelę **tOsoba** w okienku nawigacji.
- Wybierz na wstążce, w karcie **Tworzenie** listę rozwijaną **Więcej formularzy**.
- Rozwiń listę i kliknij polecenie **Formularz dzielony** – formularz zostanie wyświetlony w widoku układu.
- Zapisz formularz, nadając mu nazwę **fOsoba1**.
- Przejdź do widoku formularza i zobacz, jak działa.
- Zamknij formularz.

Na rysunku 3.8 przedstawiono przykładową postać formularza dzielnego **fOsoba1**.

Imię ojca	Imię matki	Nazwisko r.	e-mail	Telefon	Adres	Miejscowo...	Wojewódz...	Kraj
Czesław	Irena	Bogucka	kulesza34@gi	111-222-333	ul. Studzien... Supraśl	podlaskie	1	
Bogumił	Fryderyka	Iwaszkiewicz	poniatowska	567-244-667	ul. Slenkiewic... Białystok	podlaskie	1	
Andrzej	Anna	Przybylska	wars_goleble	789-456-865	ul. Marszałki... Warszawa	mazowieckie	0	

Rys. 3.8. Przykładowy widok formularza dzielnego **fOsoba1**

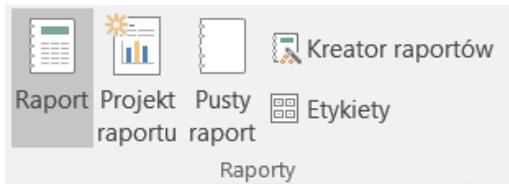
## 4. Generowanie podstawowego raportu

Tabela służy do przechowywania danych, a formularze związane z tabelą wykorzystywane są do tworzenia przyjaznego środowiska użytkownika

bazy danych. Gdy zachodzi potrzeba wydrukowania danych czy utworzenia pliku reprezentującego żądaną przez nas treść, należy utworzyć raport.

Raport jest obiektem specjalnie dostosowanym do potrzeb prezentacji i podsumowywania informacji o danych w postaci dokumentu przygotowanego do druku.

Program **Access** dysponuje narzędziami do automatycznego generowania podstawowych raportów. W zakładce **Tworzenie** na wstążce umieszczona jest grupa ikonek **Raporty** (rys. 4.1).



Rys. 4.1. Polecenie **Raport** na karcie **Tworzenie** wstążki

Narzędzie **Raport** umożliwia szybkie utworzenie raportu, nie wymagające podawania dodatkowych informacji. Jest użyteczne, gdy tabela, której dane chcemy wydrukować, zawiera niewielką liczbę pól.

W przypadku tabeli **tOsoba** lepiej skorzystać z **Kreatora raportów**, który pozwoli wybrać pola do wyświetlenia oraz zdefiniować potrzebną strukturę raportu.

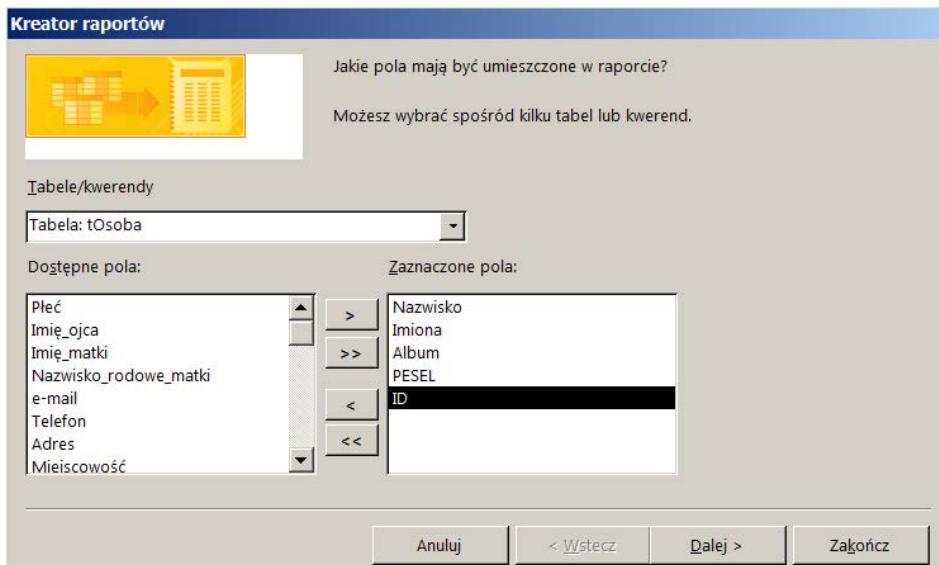
## Zadanie 4\_1

Wygeneruj raport do wyświetlenia następujących pól tabeli **tOsoba**: *Nazwisko, Imiona, ALBUM, PESEL, ID*. Dane mają być uporządkowane według pól: *Nazwisko, Imiona, Album*. Zastosuj **Kreator raportów**.

### Wykonanie

- Na karcie **Tworzenie**, w grupie **Raporty** kliknij ikonkę **Kreator raportów** (rys. 4.1).
- W oknie **Kreatora raportów** (rys. 4.2) za pomocą przycisku wybierz zadane pola, a następnie wybierz przycisk **Dalej**.
- W kolejnym oknie **Kreatora raportów** wybierz **Dalej**.
- W oknie dotyczącym sortowania wybierz po kolejno pola: *Nazwisko, Imiona, Album* (sortowanie według albumu zostanie wykonane tylko

dla tych rekordów, których wartości pól *Nazwisko* oraz *Imiona* są takie same); kliknij **Dalej**.



Rys. 4.2. Okno **Kreatora raportów**

- W kolejnym oknie wybierz **układ kolumnowy**; naciśnij **Dalej**.
- W ostatnim okienku **Kreatora raportów** należy podać tytuł raportu. Wpisz: „Dane osobowe” (bez cudzysłowu) i wybierz **Zakończ**.

Nazwisko	Imiona	Album	PESEL	ID
Goleblecki	Krzysztof	88900	880303755	3
Pełzyński	Pawel			

Rys. 4.3. Widok raportu **Dane osobowe** w widoku **podglądu wydruku**

Po wykonaniu tych czynności program utworzy raport, pokazując go w widoku **Podgląd wydruku** (rys. 4.3). Nazwa raportu będzie identyczna jak wprowadzony tytuł.

W widoku **Podgląd wydruku** ikonki wstążki pozwalają dopasować wszystkie parametry raportu do ewentualnego jego wydruku na wybranej drukarce, a także na utworzenie plików różnych typów w celu ich eksportu.

Gdy chcemy wprowadzić zmiany do struktury raportu, należy zamknąć widok **podglądu wydruku** i otworzyć raport w **widoku układu** lub **widoku projektu raportu**. Przykłady pracy w tych widokach będą rozważane w rozdziale 15 niniejszego podręcznika.

## 5. Uzupełnienie bazy danych: nowe tabele i relacje

Kontynuując budowę przykładowej bazy danych, utworzymy jeszcze cztery tabele w naszej bazie danych:

- **tPokoje** – dotyczącą pokojów w Domu Studenta;
- **tZakwaterowanie** – dotyczącą zakwaterowania studentów (przy założeniu, że zakwaterowanie i wykwaterowanie studenta odbywa się w każdym roku akademickim);
- **tKoszty\_zakwaterowania** – dotyczącą miesięcznych kosztów zakwaterowania każdego studenta (zakładamy, że koszt zależy od rodzaju pokoju; przewidziane są tu także ulgi mieszkaniowe);
- **tWpłaty** – dotyczącą wpłat studentów za zamieszkanie w Domu Studenta.

Naszym celem jest utworzenie relacji między tabelami oraz wypełnienie tabel danymi testowymi.

Należy tu zauważyć, że większość zaproponowanych w podręczniku przykładów opiera się na tych tabelach, więc dane testowe mają być spójne. W tym celu proponuje się zbudować odpowiednie formularze do wprowadzenia danych.

Wszystkie niezbędne czynności przedstawiono w postaci zadań oraz opisów do ich wykonania. Każde kolejne zadanie opiera się na poprzednich. Wykonanie większości z nich wymaga od czytelnika zastosowania wiedzy już przyswojonej na poprzednich etapach nauki. Pozwoli to wyeliminować niepotrzebne powtórzenia znanych już treści.

## 5.1. Tabela tPokoje

Informacje o pokojach Domu Studenta będziemy przechowywać w tabeli o nazwie **tPokoje**.

W oparciu o wiedzę nabycą przy tworzeniu tabeli **tOsoba** zbuduj tabelę samodzielnie, wykonując podane niżej zadanie 5.1.

### Zadanie 5\_1

- a. W bazie danych **Dom Studenta** utwórz projekt tabeli o nazwie **tPokoje**.

Wymagania do projektu tabeli:

- ❖ Tabela powinna zawierać cztery pola, jak ilustruje to rysunek 5.1 A:
  - *Nr\_pokoju* - numer unikatowy, który pełni rolę klucza podstawowego;
  - *Piętro* - zawiera numer piętra, na którym jest dany pokój;
  - *Liczba\_miejsc* - zawiera informację o liczbie miejsc w danym pokoju;
  - *Uwagi* - służy do przechowywania aktualnych komentarzy.

	Nazwa pola	Typ danych
!	Nr_Pokoju	Liczba
!	Piętro	Liczba
!	Liczba_miejsc	Liczba
!	Uwagi	Krótki tekst

Rys. 5.1 A. Widok projektu tabeli **tPokoje**

- ❖ Pole *Piętro* jest wymagane.
- ❖ Właściwość **Rozmiar pola** dla pola *Nr\_pokoju* zdefiniuj jako **Liczba całkowita**.
- ❖ Właściwość **Rozmiar pola** dotycząca pól *Piętro* oraz *Liczba\_miejsc* zdefiniuj jako **Bajt**.
- ❖ Zdefiniuj właściwość **Tytuł** dla pól *Nr\_pokoju* oraz *Liczba\_miejsc*.
- ❖ Za pomocą **Kreatora odnośników** stwórz listę możliwych wartości pola *Piętro*.
- ❖ Do pola *Liczba\_miejsc* dodaj regułę poprawności oraz tekst odpowiedniego komunikatu (np.:  $\geq 0$  and  $\leq 3$ ).

- ❖ W polach liczbowych *Nr\_pokoju*, *Piętro*, *Liczba\_miejsc* usuń wartość domyślną 0 dodaną przez program.

	Nr Pokoju	Piętro	Liczba miejsc	Uwagi
	101	1	2	

Rys. 5.1 B. Widok projektu tabeli **tPokoje**

- b. Przejjdź do widoku arkusza danych tabeli i wprowadź do tabeli **tPokoje** co najmniej 5 wierszy różnych wartości. Przykład jednego wiersza podano na rysunku 5.1 B.

## 5.2. Projekt tabeli tZakwaterowanie

Tabela **tOsoba** zawiera dane personalne studentów. Dane o ich zamieszkaniu w Domu Studenta umieścimy w innej tabeli i nazwiemy ją **tZakwaterowanie**. Przed projektowaniem tabeli sformułujemy założenia do jej treści.

### Założenia do treści tabeli

1. Student zakwaterowuje się na początku roku akademickiego, a w końcu roku akademickiego wykwaterowuje się. Zakwaterowanie rozpoczyna się 1 października, a wykwaterowanie powinno zakończyć się do 1 lipca.
2. Będziemy przechowywać w bazie danych tylko jeden numer pokoju dla każdego zakwaterowania. Jeżeli osoba w czasie zakwaterowania w Domu Studenta przeprowadza się do innego pokoju, to tylko aktualny numer pokoju będzie uwzględniany, a informacja o poprzednim zniknie.

## Zadanie 5\_2

W bazie danych **Dom Studenta** zbuduj projekt tabeli o nazwie **tZakwaterowanie**. Tabela powinna zawierać 6 pól, tak jak pokazano na rysunku 5.2.

### Przeznaczenie pól:

- ❖ *ID\_zakwaterowanie* – to unikatowy numer automatycznie przypisywany do każdego rekordu tabeli;

Nazwa pola	Typ danych
ID_Zakwaterowanie	Autonumerowanie
ID	Liczba
Data_zakwaterowania	Data/Godzina
Nr_pokoju	Liczba
Data_wykwaterowania	Data/Godzina
Wydział	Krótki tekst

Rys. 5.2. Widok projektu tabeli **tZakwaterowanie**

- ❖ *ID* – numer osoby (numer osoby był zdefiniowany w tabeli **tOsoba**; po połączeniu tych dwóch tabel będzie pełnił w tabeli **tZakwaterowanie** rolę klucza obcego);
- ❖ *Data\_zakwaterowania* – data zakwaterowania studenta;
- ❖ *Nr\_pokoju* – aktualny numer pokoju zamieszkania;
- ❖ *Data\_wykwaterowania* – data wykwaterowania studenta;
- ❖ *Wydział* – nazwa wydziału uczelni, który skierował studenta do Domu Studenta.

Zdefiniuj właściwości pól tabeli zgodnie z poniższymi wytycznymi:

- ❖ Właściwość **Rozmiar pola** *ID* zdefiniuj jako **Liczba całkowita dłuża**.
- ❖ Właściwość **Rozmiar pola** *Nr\_pokoju* zdefiniuj jako **Liczba całkowita**.
- ❖ Za pomocą **Kreatora odnośników** stwórz listę wartości pola *Wydział*, ogranicz wartości do listy stworzonej z czterech nazw wydziałów.
- ❖ Zmień rozmiar pola *Wydział*.
- ❖ Określ wartość domyślną pola *Wydział*, wybierając jedną z już wykonanej listy wartości tego pola.
- ❖ Wszystkie pola, oprócz *Data\_wykwaterowania*, określ jako wymagane.
- ❖ Zdefiniuj właściwość **Tytuł** wszędzie, gdzie jest to potrzebne.
- ❖ Ustaw właściwość **Format** pól *Data\_zakwaterowania* i *Data\_wykwaterowania* jako **Data krótka**(rys. 5.3);
- ❖ W polach liczbowych *ID* oraz *Nr\_pokoju* usuń wartość domyślną 0 dodaną przez program.
- ❖ Do pola *Data\_zakwaterowania* dodaj przykładową regułę poprawności oraz tekst odpowiedniego komunikatu (przykład podano na rys. 5.3).
- ❖ Zdefiniuj klucz podstawowy i zapisz tabelę.

Rys. 5.3. Właściwości pola Data\_zakwaterowania

Wprowadzenie danych do tabeli **tZakwaterowanie** nie jest tak proste, jak w przypadku tabeli **tPokoje**. Sytuację komplikuje obecność w tabeli **tZakwaterowanie** pola *ID*. Dobrym rozwiązaniem jest utworzenie specjalnego formularza do wprowadzenia danych na dwóch połączonych tabelach: **tOsoba** oraz **tZakwaterowanie**. Dalej taki formularz zostanie wykonany, ale najpierw należy połączyć te tabele, tzn. utworzyć między nimi relację.

### 5.3. Relacje

Relacyjna baza danych składa się z połączonych ze sobą tabel. W programie *Microsoft Access* możemy połączyć tabele, tworząc tak zwane **relacje**. Budując relację, można narzucić więzy integralności, a także uwzględnić możliwość kaskadowej aktualizacji oraz kaskadowego usuwania danych, co sprawi, że baza danych będzie bardziej odporna na ewentualne błędy.

Na połączonych tabelach można budować inne obiekty bazy danych (formularze, kwerendy, raporty) pozwalające wyświetlać informacje z wielu tabel za jednym razem.

Najprostszym sposobem na utworzenie relacji jest zastosowanie narzędzi graficznych w oknie **Relacje**.

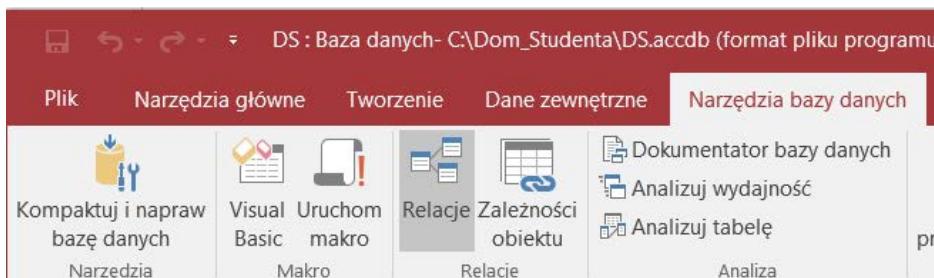
*Przykład połączenia tabel w oknie Relacje.  
Klucz obcy*

Pole **ID** w tabeli **Zakwaterowanie** jest przeznaczone do połączenia z polem **ID** tabeli **tOsoba**.

### Zadanie 5\_3

Połącz tabele **tOsoba** i **tZakwaterowanie**, działając według zaproponowanego poniżej algorytmu.

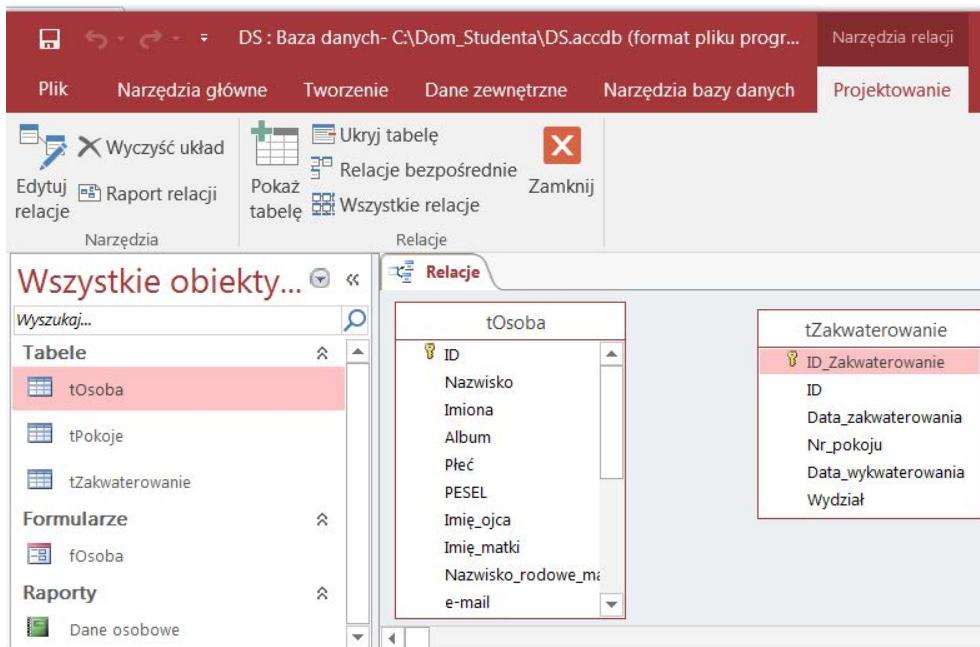
- Kliknij kartę **Narzędzia bazy danych** na wstążce i wybierz ikonkę **Relacje** (rys. 5.4).



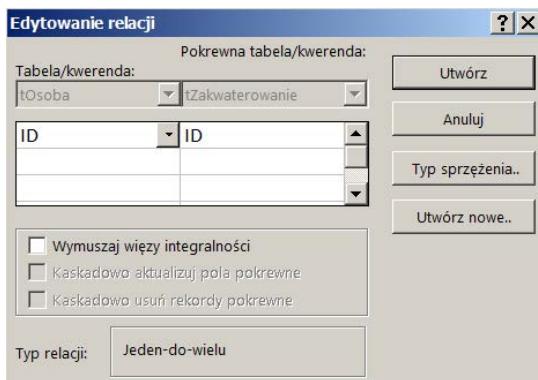
Rys. 5.4. Polecenie **Relacje**

- W okienku **Pokazywanie tabeli**, które zawiera nazwy utworzonych tabel bazy danych, wybierz najpierw **tOsoba** i naciśnij **Dodaj**. Następnie wybierz **tZakwaterowanie**, dodaj i następnie naciśnij **Zamknij**. Okno programu będzie wówczas wyglądać podobnie do pokazanego na rysunku 5.5.
- Jeśli okienko **Pokazywanie tabeli** nie otworzy się automatycznie, można skorzystać z ikonki **Pokaż tabelę** na wstążce lub nacisnąć w oknie **Relacje** na prawy klawisz myszki i wybrać polecenie **Pokaż tabelę** w menu podręcznym.
- Aby utworzyć połączenie między tabelami (relację), kliknij pole **ID** w tabeli **tOsoba**, a następnie przeciągnij je za pomocą wskaźnika myszki do pola **ID** w tabeli **tZakwaterowanie** – wyświetli się okienko **Edytowanie relacji** (rys. 5.6).
- W okienku **Edytowanie relacji**:
  - zaznacz opcję **Wymuszaj więzy integralności**;

- zaznacz opcję **Kaskadowo usuń rekordy pokrewne**;
- kliknij **Typ sprzężenia**;
- w okienku **Właściwości sprzężenia** zaznacz typ złączania wierszy: „**Uwzględnia wszystkie rekordy z ‘tOsoba’ i tylko te rekordy z ‘Zakwaterowanie’, dla których sprzężone pola są równe**”, jak przedstawia rysunek 5.7, a następnie wybierz **OK**;

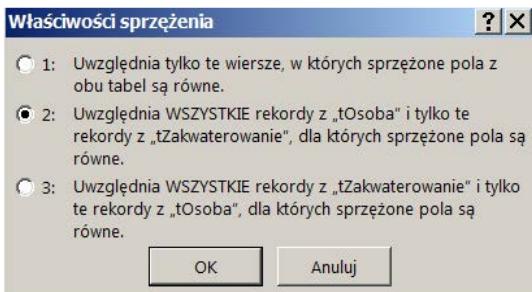


Rys. 5.5. Okno **Relacje** po umieszczeniu w nim tabel **tOsoba** oraz **tZakwaterowanie**



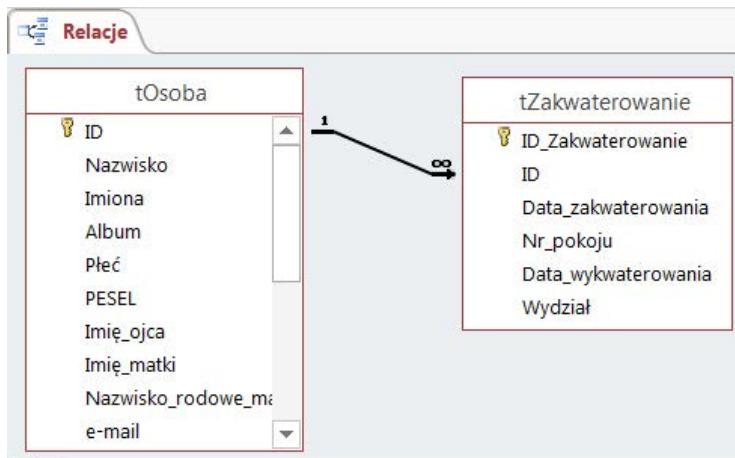
Rys. 5.6. Okienko **Edytowanie relacji**

- o kliknij przycisk **Utwórz** w okienku **Edytowanie relacji** – zobaczysz graficzną reprezentację relacji (rys. 5.8); linia między tabelami **tOsoba** i **tZakwaterowanie** jest pogrubiona, ponadto nad grubszym fragmentem linii po jednej stronie relacji jest wyświetlana cyfra 1, a nad grubszym fragmentem po drugiej stronie jest wyświetlany symbol nieskończoności ( $\infty$ ), co wskazuje na wymuszenie więzów integralności; linia reprezentuje typ związku **jeden do wielu**, zaś kierunek strzałki zależy od typu sprzężenia.



Rys. 5.7. Okienko **Właściwości sprzężenia**

e. Zapisz układ i zamknij okno.



Rys. 5.8. Relacja między tabelami **tOsoba** i **tZakwaterowania**

Tabele **tOsoba** oraz **tZakwaterowanie** zostały połączone dlatego, że w tabeli **tZakwaterowanie** ulokowano kolumnę **ID**, która zawiera wartości pola **ID** (klucza podstawowego) tabeli **tOsoba**. Kolumna **ID** w tabeli

**tZakwaterowanie** jest tak zwanym **kluczem obcym** tabeli i służy do połączenia z tabelą **tOsoba**.

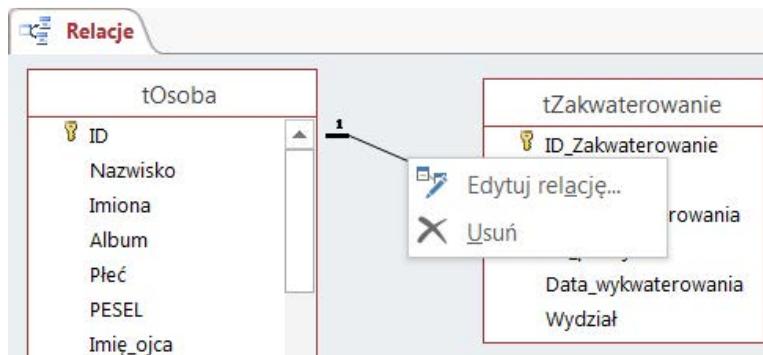
**Klucz obcy** to pole (kolumna), które czerpie swoje wartości z tej samej dziedziny co i klucz podstawowy tabeli powiązanej tabeli. Warto zauważyć, że klucz obcy może być złożony – zbudowany na więcej niż jednej kolumnie.

W dokumentacji programu *Microsoft Access* określono następujące zasady działania na danych połączonych:

- Nie można wprowadzić wartości w polu klucza obcego tabeli powiązanej, jeśli ta wartość nie istnieje w polu klucza podstawowego tabeli podstawowej – czynność ta powoduje powstanie rekordów odłączonych.
- Nie można usunąć rekordu z tabeli podstawowej, jeśli w tabeli powiązanej istnieją pasujące do niego rekordy. 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. Można jednak zdecydować się na zaktualizowanie rekordu podstawowego oraz wszystkich rekordów pokrewnych w ramach jednej operacji, zaznaczając pole wyboru **Kaskadowo aktualizuj pola pokrewne**.

W dalszej części podręcznika będziemy wykonywać wiele operacji w bazie danych Dom Studenta. Przy nieudanej próbie działania na danych zawsze należy się zastanowić, czy nie zostały naruszone opisane powyżej reguły.

W trakcie ćwiczeń z bazą danych często wynika potrzeba zmiany lub usunięcia relacji. Aby tego dokonać, należy najpierw kliknąć **lewy** klawiszem myszy linię oznaczającą relację. Potem kliknąć **prawym** przyciskiem myszy zaznaczoną linię, a następnie wybrać z menu podręcznego potrzebne polecenie (rys. 5.9).



Rys. 5.9. Polecenia do edytowania lub usunięcia relacji

## Przykład połączenia tabel za pomocą Kreatora odnośników

Dwie tabele można połączyć ze sobą, używając **Kreatora odnośników**.

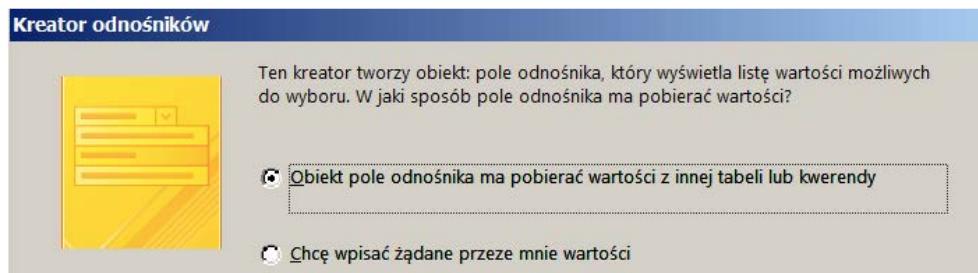
### Zadanie 5\_4

Utwórz relację między tabelami **tZakwaterowanie** i **tPokoje**.

Przy tworzeniu tabeli **tZakwaterowanie** zakładaliśmy, że w bazie danych będzie przechowywany numer pokoju, w którym student mieszkał przy każdym zakwaterowaniu. Oczywiście jest, że wartości wpisane do pola *Nr\_pokoju* w tabeli **tZakwaterowanie** nie mogą być dowolne; powinny to być takie liczby, które występują w tabeli **tPokoje**. Utworzenie połączenia tabel **tPokoje** i **tZakwaterowanie** oraz zdefiniowanie więzów integralności spowoduje, że przy wprowadzeniu numerów pokojów do tabeli **tZakwaterowanie** nie będzie można wpisać wartości niewłaściwych (niestniejących).

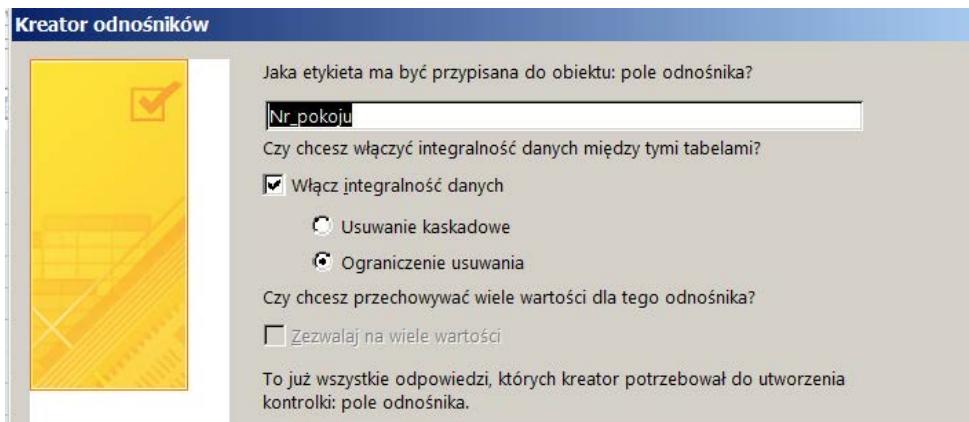
W celu połączenia tabel **tZakwaterowanie** i **tPokoje** postępuj według następujących wskazówek:

- Otwórz tabelę **tZakwaterowanie** w widoku projektu.
- Zaznacz pole *Nr\_pokoju*.
- Otwórz listę typów i kliknij **Kreator odnośników**.
- W dialogowym okienku (rys. 5.10) wybierz opcję „**Obiekt pole odnośnika ma pobierać wartości z tabeli lub kwerendy**” i naciśnij przycisk **Dalej**.
- Kliknij nazwę tabeli **tPokoje** i wybierz **Dalej**.
- Wybierz za pomocą przycisku pole *Nr\_pokoju*, kliknij **Dalej**.
- W celu sortowania numerów pokoi wybierz *Nr\_pokoju*, a następnie **Dalej**.



Rys. 5.10. Pierwsze okno **Kreatora odnośników**

- h. W ostatnim oknie **Kreatora odnośników** zaznacz opcję **Włącz integralność danych** (rys. 5.11) i wybierz **Zakończ**.



Rys. 5.11. Ostatnie okno **Kreatora odnośników**

- Po zamknięciu okna **Kreatora odnośników** pojawi się okienko dialogowe z poleceniem zapisania tabeli. Należy wybrać: „**Tak**”.
- W celu obejrzenia zmian dokonanych w projekcie tabeli **tZakwaterowanie** należy zaznaczyć pole *Nr\_pokoju* i kliknąć zakładkę **Odnośnik** (rys. 5.12). W linijce **Źródło wierszy** zobaczymy utworzone automatycznie polecenie języka **SQL** do wyświetlania listy numerów pokojów z tabeli **tPokoje**.

tZakwaterowanie		
	Nazwa pola	Typ danych
!	ID_Zakwaterowanie	Autonumerowanie
!	ID	Liczba
!	Data_zakwaterowania	Data/Godzina
!	Nr_pokoju	Liczba
!	Data_wykwaterowania	Data/Godzina
!	Wydział	Krótki tekst

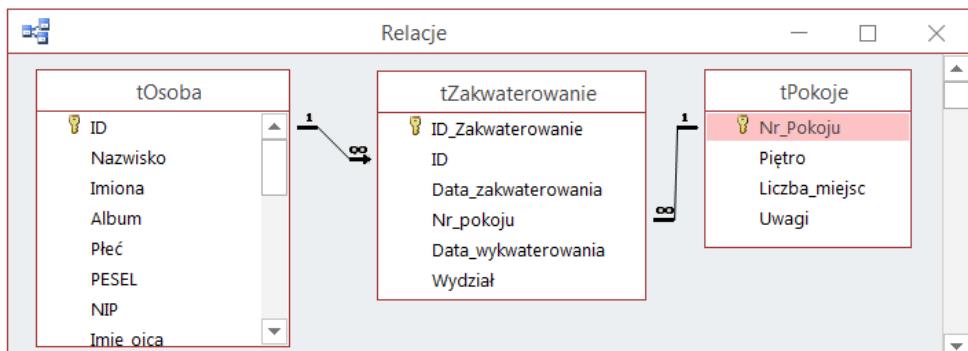
Właściwości pola

Ogólne	Odnośnik
Typ kontrolki	Pole kombi
Typ źródła wierszy	Tabela/Kwerenda
Źródło wierszy	SELECT [tPokoje].[Nr_Pokoju] FROM tPokoje ORDER BY [Nr_Pokoju];

Rys. 5.12. Zakładka **Odnośnik** pola *Nr\_pokoju* tabeli **tZakwaterowanie** po zastosowaniu **Kreatora odnośników**

Aby obejrzeć utworzone połączenie między tabelami **tZakwaterowanie** oraz **tPokoje**:

- Zamknij tabelę **tZakwaterowanie**.
- Otwórz okienko **Relacje**.
- Kliknij prawym klawiszem myszki i w menu podręcznym wybierz polecenie **Pokaz wszystko**. Wówczas w okienku pojawi się tabela **tPokoje** połączona z tabelą **tZakwaterowanie**, jak widać na rysunku 5.13.

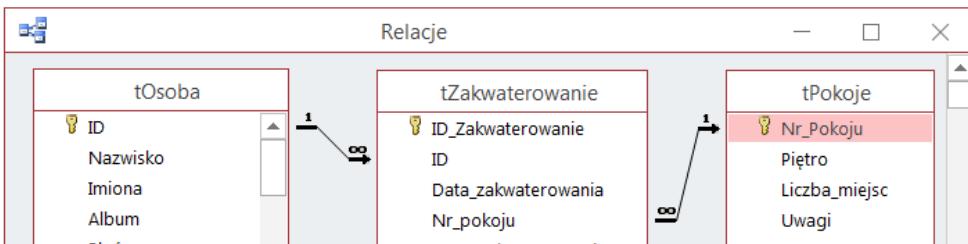


Rys. 5.13. Widok okna Relacje po utworzeniu relacji między tabelami **tZakwaterowanie** oraz **tPokoje**

## Zadanie 5\_5

Edytuj relację między tabelami **tPokoje** i **tZakwaterowanie**: zmień typ sprzężenia, ustawiając „Uwzględniaj WSZYSTKIE rekordy z **tZakwaterowanie** i tylko te rekordy z **tPokoje**, dla których sprzężone pola są równe”. Zapisz i zamknij okno **Edytuj Relacje**.

Po wykonaniu **Zadania 5\_5** okienko **Relacje** powinno wyglądać podobnie do przedstawionego na rysunku 5.14.



Rys. 5.14. Widok okna Relacje po edycji relacji między tabelami **tZakwaterowanie** oraz **tPokoje**

## 5.4. Formularz wspomagający wprowadzenie danych do tabeli tZakwaterowanie

Wprowadzanie danych testowych do tabeli w arkuszu danych może doprowadzić do licznych błędów, gdyż wcale nie jest oczywiste uzupełnienie wartości pola *ID*. W takiej sytuacji najlepiej zbudować specjalny formularz w oparciu o dwie tabele, które łączy relację: **tOsoba** oraz **tZakwaterowanie**. W tym przypadku program **Access** może wygenerować taki formularz automatycznie.

### Zadanie 5\_6

---

Utwórz formularz, który pozwoli na manipulowanie danymi o studentach oraz o ich zakwaterowaniu.

---

W tym celu dokonaj następujących czynności:

- a. Zaznacz tabelę **tOsoba** w okienku nawigacji.
- b. Na zakładce **Tworzenie** wybierz ikonkę **Formularz** – wówczas zostanie utworzony formularz z podformularzem. W dokumentacji **Access** podformularz definiujemy jako formularz wstawiony do innego formularza.
- c. Zapisz utworzony formularz, nadając mu nazwę **fOsoba2**.

Na rysunku 5.15 przedstawiono formularz **fOsoba2**. Jest on w **widoku układu**. Właśnie w tym zostanie wyświetlony formularz od razu po utworzeniu. O tym, że formularz jest widoczny w widoku układu, informuje pomarańczowa ramka wokół pola *ID*. Taka ramka pojawia się wokół każdego obiektu formularza, gdy klikamy w niego myszką.

W widoku układu łatwo jest przemieszczać obiekty formularza oraz zmieniać ich rozmiary. W zależności od postaci kurSORA myszy możliwe jest przemieszczenie lub zmiana rozmiaru obiektu.

W dolnej części formularza **fOsoba2** widoczne są dwa paski z przyciskami nawigacyjnymi. Pasek dolny służy do przewijania rekordów tabeli **tOsoba**, górny – rekordów tabeli **tZakwaterowanie**.

Formularz na rysunku 5.15 zawiera pusty podformularz. Możemy wprowadzić tutaj dane o zakwaterowaniu w różnych latach studenta o numerze *ID* równym 1. Aby wprowadzić dane o zakwaterowaniu kolejnej osoby, należy skorzystać z dolnego paska przewijania, przechodząc do kolejnego rekordu tabeli **tOsoba**.

Rys. 5.15. Formularz **fOsoba2** w widoku układu

Wprowadzenie nowych danych wymaga **przełączenia się z widokiem układu do widoku formularza**.

Przykład podformularza z wprowadzonymi danymi o zakwaterowaniu osoby o numerze *ID* równym 1 jest przedstawiony na rysunku 5.16.

Rys. 5.16. Widok podformularza formularza **fOsoba2** w momencie przeglądania danych osoby o numerze *ID* równym 1

## Zadanie 5\_7

Przejdź do widoku formularza **fOsoba2**. Wprowadź trzy wiersze o zakwaterowaniu pierwszej z kolej osoby (z tabeli **tOsoba**)

oraz po jednym wierszu o zakwaterowaniu kolejnych pięciu osób. Po wprowadzeniu danych zamknij formularz i otwórz tabelę **tZakwaterowanie** w widoku arkusza danych, aby upewnić się, czy jest ona wypełniona.

## 5.5. Tabela tKoszty\_zakwaterowania

Zakładamy, że na początku roku akademickiego każdy student ma ustaloną wysokość opłaty, którą musi wnosić co miesiąc za swoje zamieszkanie. Dlatego utworzymy tabelę o nazwie **tKoszty\_zakwaterowania**, która będzie zawierać wartości miesięcznej opłaty dla każdej osoby oraz daty, od kiedy i do kiedy taka kwota obowiązuje.

### *Projekt tabeli tKoszty\_zakwaterowania*

W oparciu o nabycią wiedzę na temat projektowania tabel wykonaj zadanie 5\_8 w celu stworzenia jeszcze jednej tabeli w swojej bazie danych.

### Zadanie 5\_8

W bazie danych **Dom Studenta** utwórz projekt tabeli o nazwie **tKoszty\_zakwaterowania**. W tabeli tej będą przechowywane kwoty miesięcznych opłat studentów za zamieszkanie w Domu Studenta.

#### Wytyczne do projektu:

- ❖ Tabela powinna zawierać kolumny o nazwach (rys. 5.17):
  - **ID\_Koszty** – numer unikatowy, który pełni rolę klucza podstawowego;

Nazwa pola	Typ danych
 <b>ID_Koszty</b>	Autonumerowanie
<b>ID_Zakw</b>	Liczba
<b>Od_kiedy</b>	Data/Godzina
<b>Do_kiedy</b>	Data/Godzina
<b>Kwota_mies_opłaty</b>	Waluta

Rys. 5.17. Projekt tabeli **tKoszty\_zakwaterowania**

- **ID\_Zakw** – numer zakwaterowania z tabeli **Zakwaterowanie** (pobiera się z pola **ID\_zakwaterowanie** tej tabeli; w tabeli **tKoszty\_zakwaterowania** pełni rolę klucza obcego);

- o *Od\_kiedy* – data ustalenia miesięcznej kwoty opłaty za zamieszkanie w Domu Studenta;
  - o *Do\_kiedy* – data końca ważności obowiązywania miesięcznej kwoty opłaty za zamieszkanie w Domu Studenta;
  - o *Kwota\_mies\_opłaty* – ustalona wartość opłaty za jeden miesiąc zamieszkania w Domu Studenta.
- ❖ Typy danych należy zdefiniować tak, jak pokazano na rys. 5.17.
- ❖ W polu *ID\_zakw* właściwość **Rozmiar** zdefiniuj jako **Liczba całkowita dłuża**.
- ❖ Zdefiniuj format dla pól zawierających daty.
- ❖ Zdefiniuj właściwość **Tytuł** dla pól *Od\_kiedy*, *Do\_kiedy*, *Kwota\_mies\_opłaty*.
- ❖ Wartość domyślna pola *Kwota\_mies\_opłaty* powinna być równa 0 zł.
- ❖ Pole *ID\_Zakw* nie powinno mieć wartości domyślnej.
- ❖ Wszystkie pola powinny być wymagane.

### *Połączenie tabel tZakwaterowanie oraz tKoszty\_zakwaterowania*

Każdy rekord o zakwaterowaniu osoby ma być połączony z rekordem zawierającym dane o wartości miesięcznej opłaty w trakcie tego zakwaterowania. Należy utworzyć relację między tabelą **tZakwaterowanie** i **tKoszty\_zakwaterowania**.

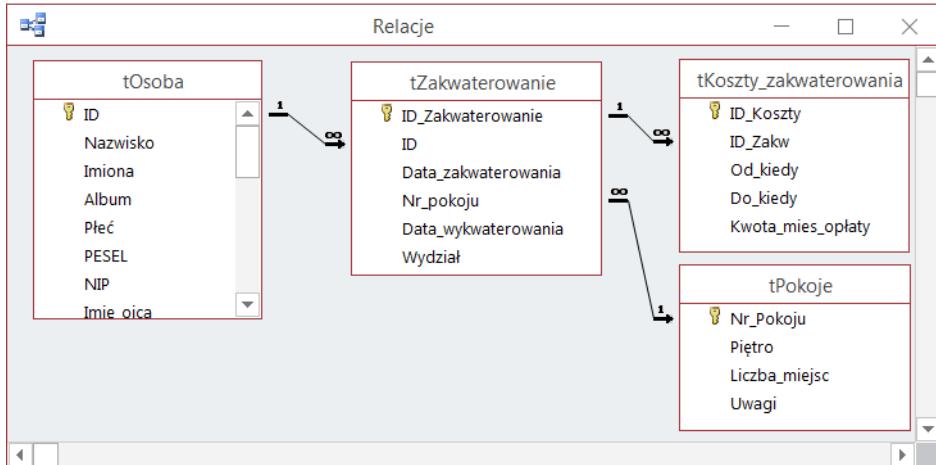
Postępując tak samo jak przy połączeniu tabel **tOsoba** i **tZakwaterowanie**, wykonaj zadanie 5\_9.

### **Zadanie 5\_9**

Utwórz relację pomiędzy tabelami **Zakwaterowanie** oraz **tKoszty\_zakwaterowania** w takiej postaci, jak na rysunku 5.18. W oknie **Edytowanie relacji** wybierz opcje **Wymuszaj więcej integralności** oraz **Kaskadowo usuń rekordy pokrewne**.

### *Formularz wspomagający wprowadzenie danych do tabeli tKoszty\_zakwaterowania*

Utworzona tabela zawiera kolumnę *ID\_zakw*, w której nie można zamieszować dowolnych danych. Jeśli spróbujemy wpisywać przykładowe dane testowe w widoku arkusza danych tabeli, może nam się nie udać zapisać żadnego z powodu naruszenia więzów integralności. Dlatego utworzymy formularz, który usprawni proces wprowadzenia prawidłowych wartości do tabeli **tKoszty\_zakwaterowania**.



Rys. 5.18. Okno **Relacje** po utworzeniu połączenia między tabelami **Zakwaterowanie** oraz **tKoszty\_zakwaterowania**

Formularz zostanie zbudowany w trakcie wykonywania zadania 5\_10.

## Zadanie 5\_10

Utwórz formularz o nazwie **fKoszty** z dwoma podformularzami w oparciu o tabele **tOsoba**, **tZakwaterowanie** oraz **tKoszty\_zakwaterowania**.

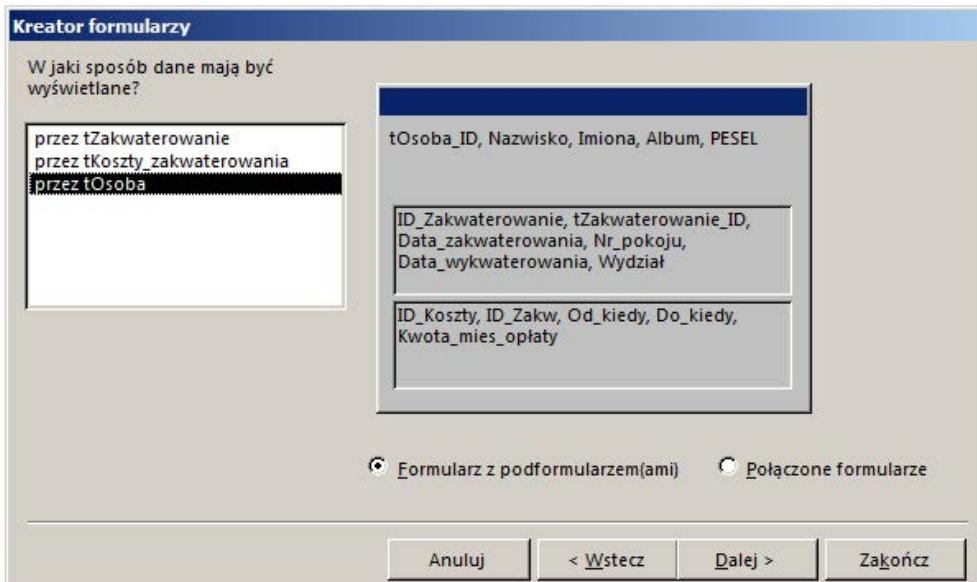
Skorzystaj z podanych niżej podpowiedzi w celu uzyskania takiej samej postaci formularza jak na rysunku 5.20.

Wprowadź przykładowe dane o kosztach zakwaterowania. Należy wprowadzić tyle wierszy, aby każdemu rekordowi z tabeli **tZakwaterowanie** odpowiadał co najmniej jeden rekord w tabeli **tKoszty\_zakwaterowania**.

## Wykonanie

Uruchom **Kreatora formularzy**.

- W pierwszym oknie **Kreatora formularzy** należy wybrać po kolej:
  - o pola *ID*, *Nazwisko*, *Imiona*, *Album*, *PESEL* z tabeli **tOsoba**,
  - o wszystkie pola tabeli **tZakwaterowanie**,
  - o wszystkie pola tabeli **tKoszty\_zakwaterowania**,
  - o przycisk **Dalej**.
- W kolejnym oknie **Kreatora formularzy** należy ustawić opcje tak, jak na rysunku 5.19.



Rys. 5.19. Widok drugiego okna Kreatora formularzy przy generowaniu formularza **fKoszty\_zakwaterowania**

- c. W kolejnym oknie **Kreatora formularzy** nie wprowadzamy żadnych zmian.
- d. W ostatnim oknie **Kreatora** należy wpisać tytuły formularza i podformularzy, jak pokazano na rysunku 5.20, i nacisnąć **Zakończ**.



Rys. 5.20. Nazwy formularza oraz podformularzy w ostatnim oknie **Kreatora formularzy** przy generowaniu formularza **fKoszty\_zakwaterowania**

Po zamknięciu ostatniego okna **Kreatora formularzy** na ekranie zostanie wyświetlony formularz podobny do tego przedstawionego na rysunku 5.21.

Aby wykonany formularz wyglądał tak, jak przedstawiono na tym rysunku, należy przełączyć się na widok układu i za pomocą wskaźnika myszy dopasować rozmiary wszystkich elementów.

**fKoszty**

ID	1	Album	34567
Nazwisko	Kulesza	PESEL	90112212345
Imiona	Paweł		

Zakwaterowanie

ID_Zakwaterowanie	ID	Data zakwaterowania	Nr pokoju	Data wykwaterowania	Wydział
1	1	2014-10-01	101	2015-06-30	Mechaniczny
2	1	2015-09-25	202	2016-06-30	Mechaniczny
3	1	2016-10-01	411		Mechaniczny
*	(Nowy)				Mechaniczny

Rekord: 1 z 3 ▶ ▶ Bez filtru Wyszukaj

Koszty miesięczne pobytu w DS

ID_Koszty	ID_Zakw	Od kiedy	Do kiedy	Kwota miesięcznej opłaty
*	(Nowy)	1		0,00 zł

Rekord: 1 z 6 ▶ ▶ Bez filtru Wyszukaj

Rys. 5.21. Widok formularza **fKoszty** przed wprowadzeniem danych

Każda osoba może mieć przypisanych kilka wierszy w tabeli **tZakwaterowanie**, a z każdym z tych wierszy może być połączonych kolejnych kilka wierszy w tabeli **tKoszty\_zakwaterowania**.

Aby wprowadzić dane o kwotach miesięcznej opłaty dla dowolnej osoby, należy:

- przejść od **Widoku układu do Widoku formularza**;
- wybrać osobę;
- zaznaczyć pierwszy z wierszy** podformularza **fZakwaterowanie**;
- wprowadzić wiersz do podformularza **Koszty miesięczne pobytu w DS**, jak pokazano na rysunku 5.22.

Jeżeli dana osoba ma więcej niż jeden wpis w tabeli **tZakwaterowanie** (tak jak na rys. 5.22), wówczas należy:

- zaznaczyć drugi z wierszy** podformularza **fZakwaterowanie** (dotyczący kolejnego zakwaterowania) i wprowadzić wiersz danych do podformularza **fKoszty miesięczne pobytu w DS**;
- zaznaczając po kolei wszystkie wiersze podformularza **fZakwaterowanie**, należy postępować analogicznie, aż każdy rekord podformularza **fZakwaterowanie** będzie miał połączony z nim rekord w podformularzu **fKoszty miesięczne pobytu w DS**; tym samym wprowadzimy wszystkie kwoty miesięcznej opłaty, które obowiązywały na każdym roku studiów danego studenta.

fKoszty

ID	1	Album	34567
Nazwisko	Kulesza	PESEL	90112212345
Imiona	Paweł		

Zakwaterowanie

ID_Zakwaterowanie	ID	Data zakwaterowania	Nr pokoju	Data wykwaterowania	Wydział
1	1	2014-10-01	101	2015-06-30	Mechaniczny
2	1	2015-09-25	202	2016-06-30	Mechaniczny
3	1	2016-10-01	411		Mechaniczny
*	(Nowy)	1			Mechaniczny

Rekord: 1 z 3 ▶ ▶ Bez filtru Wyszukaj

Koszty miesięczne pobytu w DS

ID_Koszty	ID_Zakw	Od kiedy	Do kiedy	Kwota miesięcznej opłaty
1	1	2014-10-01	2015-06-30	280.00 zł
*	(Nowy)	1		0.00 zł

Rekord: 1 z 6 ▶ ▶ Bez filtru Wyszukaj

Rys. 5.22. Widok formularza **fKoszty** z danymi o koszcie miesięcznego pobytu Pawła Kuleszy w 2014 roku

Następnie wybieramy kolejną osobę i zaznaczając każdy wiersz w pod-formularzu **fZakwaterowanie**, wprowadzamy po jednym wierszu do pod-formularza **fKoszty miesięczne pobytu w DS**.

## 5.6. Tabela tWpłaty

Teraz utworzymy tabelę rzeczywistych wpłat, które każda osoba wnosiła za zamieszkanie w Domu Studenta.

### Projekt tabeli tWpłaty

W celu utworzenia projektu tabeli należy wykonać zadanie 5\_11.

## Zadanie 5\_11

Utwórz projekt tabeli o nazwie **tWpłaty**.

Wytyczne do projektu:

- ❖ Tabela powinna składać się z pól o nazwach (rys. 5.23):

- o *ID\_wpłaty* - numer unikatowy, który pełni rolę klucza podstawowego;
- o *ID* - wartość pola *ID* z tabeli **tOsoba**, pełniąca rolę klucza obcego;
- o *Pokwitowanie* - zawiera numer potwierdzenia wpłaty z banku;
- o *Data\_wpłaty* - data wpłaty (zakładamy, że student wielokrotnie wnosi opłatę za zamieszkanie w Domu Studenta);
- o *Wpłata* - kwota jednorazowej opłaty za zamieszkanie w Domu Studenta; każdorazowo kwoty te mogą przyjmować różne wartości, ale ich suma za dany rok akademicki powinna odpowiadać wymaganej;
- o *Za\_rok\_akad* - liczba całkowita, która wskazuje, za który rok akademicki była wpłacona kwota za zamieszkanie; tu będziemy wprowadzać tylko pierwszą z dwóch liczb określających rok akademicki – przykładowo, aby wprowadzić wartość roku akademickiego 2015/2016, będziemy wprowadzać do tabeli tylko wartość 2015.

	Nazwa pola	Typ danych
!	<i>ID_Wpłaty</i>	Autonumerowanie
	<i>ID</i>	Liczba
	<i>Pokwitowanie</i>	Krótki tekst
	<i>Data_wpłaty</i>	Data/Godzina
	<i>Wpłata</i>	Waluta
	<i>Za_rok_akad</i>	Liczba

Rys. 5.23. Projekt tabeli **Wpłaty**

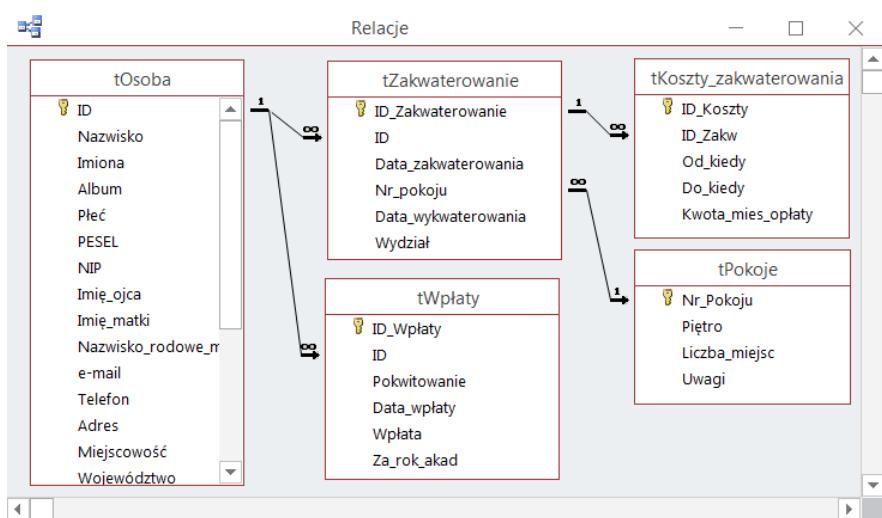
- ❖ Typy danych należy zdefiniować tak, jak pokazano na rysunku 5.23.
- ❖ Właściwość **Rozmiar** pola *ID* zdefiniuj jako **Liczba całkowita dłuża**.
- ❖ Zdefiniuj format dla pola *Data\_wpłaty*.
- ❖ Właściwość **Rozmiar pola** *Za\_rok\_akad* zdefiniuj jako **Liczba całkowita**.
- ❖ Zdefiniuj właściwość **Tytuł** dla pól: *Data\_wpłaty* oraz *Za\_rok\_akad*.
- ❖ Zdefiniuj wartość domyślną dla pola *Za\_rok\_akad*.
- ❖ Wartość domyślna dla pola **Wpłata** ma być równa 0.
- ❖ Pole *ID* nie powinno mieć wartości domyślnej.
- ❖ Wszystkie pola powinny być wymagane.

## Połączenie tabel **tOsoba** oraz **tWpłaty**

Bazując na już nabycie wiedzy odnośnie do budowania relacji, połącz tabele **tOsoba** i **tWpłaty**, wykonując kolejne zadanie.

### Zadanie 5\_12

Utwórz relację pomiędzy tabelami **tOsoba** oraz **tWpłaty** w takiej postaci, jak na rysunku 5.24. W tym celu otwórz okienko **Relacje** i umieść w nim tabelę **tWpłaty** wraz z linią relacji. W oknie **Edytowanie relacji** wybierz opcje **Wymuszaj więzy integralności** oraz **Kaskadowo usuń rekordy pokrewne**.



Rys. 5.24. Okno **Relacje** po utworzeniu połączenia między tabelami **tOsoba** oraz **tWpłaty**

*Formularz wspomagający wprowadzenie danych do tabeli **tWpłaty***

### Zadanie 5\_13

Utwórz formularz o nazwie **fWpłaty** oparty na tabelach **tOsoba** oraz **tWpłaty**, zawierający dane o wpłatach każdej osoby. Wprowadź przykładowe dane do tego formularza. Formularz powinien mieć postać przedstawioną na rysunku 5.26.

## Wykonanie

Zastosuj **Kreatora formularzy**:

- ❖ W pierwszym oknie **Kreatora formularzy** wybierz kolejno pola: *ID*, *Nazwisko*, *Imiona*, *Album* tabeli **tOsoba** oraz wszystkie pola tabeli **tWpłaty**.
- ❖ W kolejnym oknie **Kreatora formularzy** wybierz opcję, aby dane były „wyświetlane przez **tOsoba**”.
- ❖ Zapisz formularz, nie wnosząc dalszych zmian. Nadaj mu nazwę **fWpłaty**, a podformularzowi – nazwę **WpłatyPodformularz**.
- ❖ Przejdź do **widoku układu** formularza i powiększ go, zwalniając miejsce dla kolejnego podformularza, jak pokazano na rysunku 5.25.

Rys. 5.25. Nieuukończony formularz **fWpłaty**

- ❖ Zaznacz w oknie nawigacji nazwę tabeli **tZakwaterowanie** i za pomocą wskaźnika myszy przeciągnij ją do formularza **fWpłaty** na puste miejsce. Spowoduje to pojawienie się w formularzu jeszcze jednego podformularza. Ten ostatni podformularz pozwoli na podgląd danych dotyczących zakwaterowania, co oznacza, że pomoże przy wprowadzeniu właściwych danych testowych o wpłatach studenta za zamieszkanie w Domu Studenta (rys.5.26).

Formularz fWpłyty

Wpłaty za zamieszkanie w Domu Studenta

ID	<input type="text" value="1"/>
Nazwisko	<input type="text" value="Kulesza"/>
Imiona	<input type="text" value="Paweł"/>
Album	<input type="text" value="34567"/>

**Wpłaty**

ID_Wpłaty	ID	Pokwitowanie	Data wpłaty	Wpłata	Za rok akademicki
*	(Nowy)	1		0.00 zł	2016
<b>Rekord:</b> 1 z 1   Bez filtru   Wyszukaj					

ID_Zakwaterowanie	Data zakwaterowania	Nr pokoju	Data wykwaterowania	Wydział
1	2014-10-01	101	2015-06-30	Mechaniczny
2	2015-09-25	202	2016-06-30	Mechaniczny
3	2016-10-01	411		Mechaniczny
*	(Nowy)			Mechaniczny
<b>Rekord:</b> 1 z 6   Bez filtru   Wyszukaj				

Rys. 5.26. Ostateczna postać formularza **fWpłyty**

### Informacje do wprowadzenia danych testowych

Przy wprowadzeniu danych przykładowych do podformularza **fWpłyty** podformularz z danymi dotyczącymi zakwaterowania służy tylko do podglądu i kontroli, aby wprowadzone dane dotyczyły tego okresu, kiedy student faktycznie zamieszkiwał w Domu Studenta.

Zakładamy, że w polu *Wpłata* możemy wpisywać różne kwoty, niekoniecznie takie same jak w tabeli **tKoszty\_zakwaterowania**. Suma wpłat za dany rok akademicki powinna odpowiadać wymaganej, co może być osobno sprawdzane w bazie danych.

Na rysunku 5.26 przedstawiono formularz **fWpłyty** po wprowadzeniu danych testowych dotyczących osoby, której numer *ID* jest równy 1. Widzimy w polu *Za\_rok\_akad* pojedyncze liczby, np. 2014, 2015 lub 2016. Zgodnie z założeniem podczas projektowania tabeli **tWpłyty** liczby te oznaczają odpowiednio lata akademickie: 2014/2015, 2015/2016, 2016/2017.

Dalej wszędzie traktujemy jako rok akademicki zakres czasu od 1 października do 30 czerwca.

Wpłaty

Wpłaty za zamieszkanie w Domu Studenta

ID	<input type="text"/>
Nazwisko	Kulesza
Imiona	Paweł
Album	34567

**Wpłaty**

ID_Wpłaty	ID	Pokwitowanie	Data wpłaty	Wpłata	Za rok akademicki
1	1	UJ-1234567	2014-10-20	280.00 zł	2014
2	1	UJ-3452345	2015-02-19	280.00 zł	2014
3	1	AS16788987	2015-10-20	310.00 zł	2015
4	1	DF89773001	2016-11-23	300.00 zł	2016
*	(Nowy)			0.00 zł	2016

Rekord: ▲ ▼ 1 z 4 ► ►► □ Bez filtru Wyszukaj

ID_Zakwaterowanie	Data zakwaterowania	Nr pokoju	Data wykwaterowania	Wydział
1	2014-10-01	101	2015-06-30	Mechaniczny
2	2015-09-25	202	2016-06-30	Mechaniczny
3	2016-10-01	411		Mechaniczny
*	(Nowy)			Mechaniczny

Rekord: ▲ ▼ 1 z 3 ► ►► □ Bez filtru Wyszukaj

Rys. 5.27. Formularz **Wpłaty** po wprowadzeniu danych testowych o wpłatach osoby, której numer ID jest równy 1

## 5.7. Przykład wprowadzenia zmian do projektu tabeli

Relacyjne bazy danych mają wiele zalet, ale największą ich wadą jest trudność wprowadzenia zmian do ich struktury. Oczywiście chodzi tu o działającą bazę danych, która jest obsługiwana jedną lub wieloma aplikacjami. Może to wiązać się z poniesieniem dużych kosztów modyfikacji aplikacji.

Wykonaliśmy pięć tabel w najprostszej postaci, z niedużą ilością wprowadzonych danych, co oznacza, że jest to bardzo prosta baza danych. Jednak gdybyśmy na obecnym etapie spróbowali modyfikować pola, na których utworzono relacje, to część naszych danych mogłyby okazać się nieprawidłowa. Zatem nie jest wskazana modyfikacja tabel już istniejącej bazy danych. Tym niemniej narzędzia do takich modyfikacji istnieją w programie: w oknie projektu tabeli można znaleźć ikonki do dodania i usunięcia pól. Możemy posługiwać się nimi w trakcie tworzenia nowej tabeli, ale także możemy dodawać pola do tabel w istniejącej bazie danych. Właśnie taki przykład dalej rozważymy.

Wykonaj zadanie 5\_14 w celu dodania do tabeli **tOsoba** tekstowego pola *NIP*.

### Zadanie 5\_14

---

Otwórz tabelę **tOsoba** w widoku projektu. Za pomocą polecen przedstawionych na zakładce **Projektowanie** wstażki:

- wstaw między polami *PESEL* a *Imię\_ojca* nowe pole tekstowe o nazwie *NIP*;
- zdefiniuj maskę wprowadzania pola *NIP*.

W widoku arkusza danych wprowadź wartości *NIP* do dwóch wierszy tabeli **tOsoba**. Zamknij tabelę.

---

## II. Kwerenda – podstawowe narzędzie bazy danych

Relacyjna baza danych jest zbiorem połączonych ze sobą tabel. Relacyjny system zarządzania bazami danych zawsze dysponuje narzędziami wyszukiwania danych. Proces pozyskiwania informacji z bazy danych polega na wyświetlaniu zawartości pewnych pól z określonych rekordów jednej lub kilku tabel. Oznacza to, że baza danych „oczekuje” od użytkownika zdefiniowania pewnego kryterium wyszukiwania oraz konkretnego polecenia. Użytkownik powinien dysponować szczegółową wiedzą na temat sposobów znajdowania rekordów w danym systemie baz danych, tj. umiejętnością definiowania kryteriów wyszukiwania.

Głównym narzędziem wyszukiwania danych w tabelach, a także ich dodawania, usunięcia oraz aktualizacji są kwerendy. Można je tworzyć zarówno za pomocą narzędzi graficznych, jak i języka SQL. Przy tym użytkownik powinien dysponować wiedzą o tworzeniu kryteriów, tzn. różnego rodzaju wyrażeń. Najłatwiej jest nauczyć się tego, korzystając ze środowiska graficznego programu **Microsoft Access**. Uzyskana wiedza okaże się bardzo przydatna przy nauce języka **SQL**.

W części II podręcznika będą zaprezentowane liczne możliwości graficznego interfejsu programu **Microsoft Access** stosowane do manipulowania danymi za pośrednictwem kwerend oraz innych narzędzi.

### 6. Proste techniki znajdowania danych

W niniejszym rozdziale przedstawiono przykłady prostych rozwiązań programu **Microsoft Access 2016**, przy czym nacisk został położony na zasady tworzenia filtrów danych, w tym tych zaawansowanych. Centralne miejsce zajmuje nauka tworzenia kryteriów wyszukiwania, bez czego nie jest możliwe opanowanie najważniejszych narzędzi relacyjnych baz danych.

## 6.1. Przykładowa baza danych Northwind

W celu zapoznania się z możliwościami znajdowania potrzebnych informacji w relacyjnej bazie danych skorzystamy z przykładowej bazy danych firmy Microsoft o nazwie **Northwind**.

Program **Microsoft Access 2016** zawiera profesjonalnie zaprojektowane szablony ułatwiające tworzenie baz danych. Każdy szablon może zostać użyty bez modyfikacji lub dostosowany do potrzeb użytkownika.

### Zadanie 6\_1

Skorzystaj z szablonu **Northwind** do utworzenia bazy danych **Northwind**. Zaloguj się do bazy danych. Obejrzyj okno główne aplikacji przeznaczonej do pracy z tą bazą danych.

W celu wykonania zadania postępuj według następującego algorytmu:

- Uruchom program *Microsoft Access 2016*.
- Wśród ikonek szablonów w prawej części ekranu wybierz ikonę



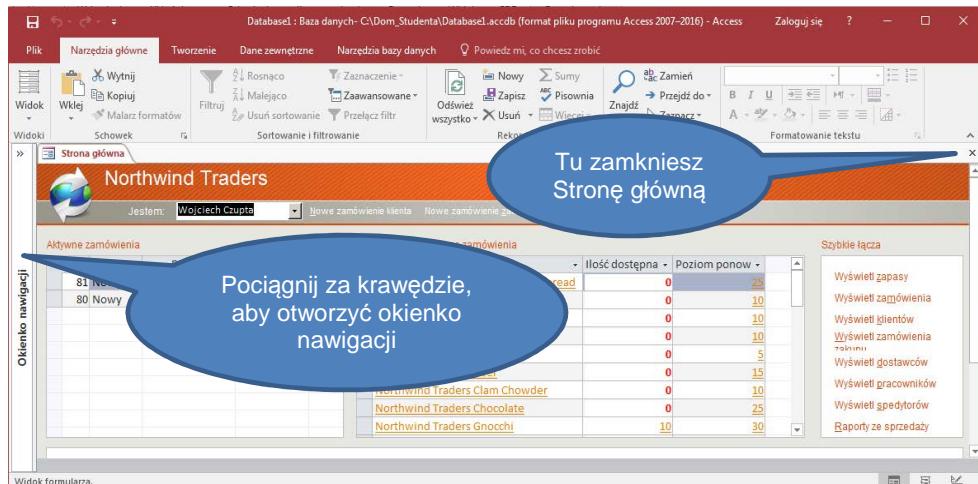
**Northwind**.

- W okienku dialogowym za pomocą ikony obok pola **Nazwa pliku** wybierz lokalizację (swój folder na dysku lokalnym).
- Naciśnij przycisk **Utwórz**. Za chwilę nowa baza danych będzie gotowa do użytku.
- Pierwsze okno, które się wyświetli, jest oknem logowania (rys. 6.1). Zaloguj się do bazy danych: wystarczy wybrać dowolne nazwisko pracownika z listy rozwijanej i nacisnąć przycisk **Zaloguj**.



Rys. 6.1. Okno logowania do aplikacji bazodanowej **Northwind**

**Northwind** jest systemem śledzenia zamówień. Jego okno główne (rys. 6.2) stanowi formularz zawierający informacje na temat aktywnych zamówień produktów. Są tu widoczne także szybkie łącza do innych formularzy.



Rys. 6.2. Strona główna systemu **Northwind**

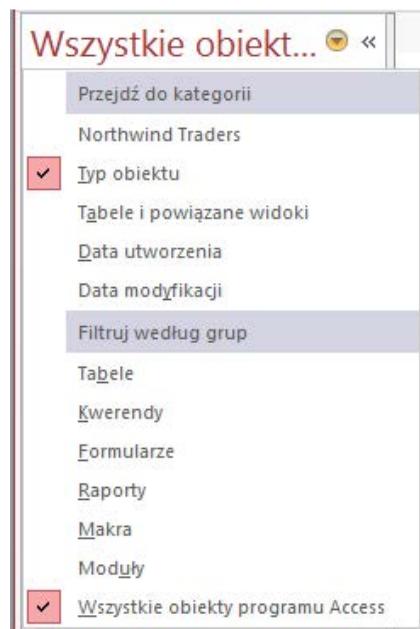
## Zadanie 6\_2

Zapoznaj się ze wszystkimi tabelami bazy danych **Northwind**. Przejrzyj każdą w **widoku projektu** i w **widoku arkusza danych**. Zapoznaj się z nowymi typami danych, które nie były używane w tabelach bazy danych Dom Studenta.

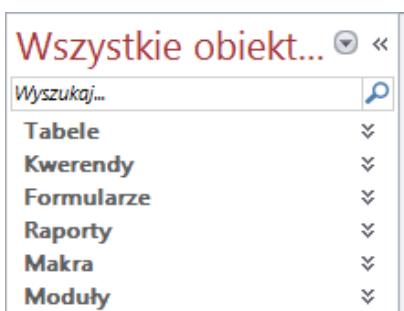
### Wykonanie

- Zamknij stronę główną aplikacji.
- Otwórz okienko nawigacji.
- Kliknij ikonkę obok nagłówka okienka nawigacji. Pojawi się menu okienka nawigacji (rys. 6.3).
- Zaznacz w menu polecenia **Typ obiektu** oraz **Wszystkie obiekty programu Access**, jak pokazano na rysunku 6.3. Wówczas widok okienka nawigacji zmieni się (rys. 6.4).
- Rozwiń węzeł **Tabele** (rys. 6.5) i przejrzyj każdą tabelę w widoku projektu oraz w widoku arkusza danych. Znajdź nowe typy danych, odmienne od tych używanych w tabelach bazy danych Dom Studenta.

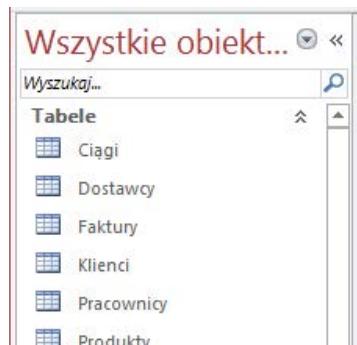
Dalsze ćwiczenia będą dotyczyć tabel z listy przedstawionej na rysunku 6.5.



Rys. 6.3. Menu okienka nawigacji



Rys. 6.4. Typy obiektów w okienku nawigacji

Rys. 6.5. Lista tabel bazy danych **Northwind** w okienku nawigacji

## 6.2. Najprostsze metody wyszukiwania danych w tabeli

Gdy warunek wyszukiwania zależy od danych tylko jednego pola, wówczas stosujemy najprostsze narzędzia wyszukiwania. Rozważmy zastosowanie takich metod na przykładach.

## *Wyszukiwanie danych za pomocą pola Wyszukaj*

Zastosowanie pola **Wyszukaj** jest najprostszą metodą znajdowania wartości w tabeli. Pole jest ulokowane na dolnym pasku tabeli w **widoku arkusza danych** (rys. 6.6).



Rys. 6.6. Widok pola **Wyszukaj**

### **Zadanie 6\_3**

W okienku nawigacji otwórz tabelę **Klienci** w **widoku arkusza danych**. Wyszukaj wiersze zawierające nazwę *lubelskie*.

---

#### **Wykonanie**

- a. W polu **Wyszukaj** (rys. 6.6) wpisz: *lubelskie* (rys. 6.7).



Rys. 6.7. Widok pola **Wyszukaj** po wprowadzeniu tekstu - *lubelskie*

- b. Naciśnij klawisz **Enter** – kurSOR ustawi się w komórce zawierającej zadaną tekst.
- c. Następne naciśnięcie na klawisz **Enter** ustawi kurSOR w kolejnym wierszu zawierającym daną nazwę.

## *Wyszukiwanie danych za pomocą polecenia Znajdź*

Polecenie **Znajdź** działa tak samo jak w innych programach w systemie Windows. Można się o tym przekonać wykonując następujące zadanie.

### **Zadanie 6\_4**

Zastosuj polecenie **Znajdź** do wyszukiwania w tabeli **Klienci** wartości zadanej nazwy firmy. Przykładowo należy odnaleźć wartość „Firma G”.

---

#### **Wykonanie**

- a. Otwórz tabelę **Klienci**.
- b. Zaznacz kolumnę *Firma*.

- c. Na karcie **Narzędzia główne** w grupie **Znajdowanie** wybierz ikonkę Zostanie wyświetcone okno dialogowe **Znajdowanie i zamienianie**.
- d. W polu **Znajdź** wpisz wartość: **Firma G** - jest to kryterium wyszukiwania.

### Filtry wspólne

Filtrowanie pozwala wyszukać grupę wierszy, które spełniają określone przez użytkownika warunki.

**Filtry wspólne** – to polecenia menu **Filtruj**. Ikonka znajdziemy na wstążce, w grupie **Sortowanie i filtrowanie**. Ikonka jest dostępna, jeśli jest otwarta którakolwiek tabela w **widoku arkusza danych**.

Po wybraniu polecenia **Filtruj** użytkownikowi udostępniana jest lista filtrów zależna od typu danych i wartości zaznaczonego pola. Zapoznamy się z nimi, wykonując kolejne zadanie.

## Zadanie 6\_5

Wyszukaj w tabeli **Zamówienia** wiersze dotyczące zamówień firmy o nazwie „Firma BB”. Po obejrzeniu wynikowych wierszy ponownie wyświetl wszystkie rekordy tabeli.

### Wykonanie

- Otwórz tabelę **Zamówienia** w **widoku arkusza danych**.
- Zaznacz pole *Klient*.
- Na karcie **Narzędzia główne** w grupie **Sortowanie i filtrowanie** kliknij ikonkę **Filtruj**.
- W okienku, które się wyświetli, kliknij w **Zaznacz wszystko** (aby skasować tę opcję), a następnie wybierz wartość „**Firma BB**”.

W wyniku filtrowania zostaną ukryte wszystkie wiersze, które nie zawierają nazwy tego klienta.

Aby skasować filtr i ponownie wyświetlić wszystkie rekordy, wybierz na zakładce **Narzędzia główne** wstążki ikonkę .

Sposób wyszukiwania rekordów za pomocą filtrów wspólnych zależy od typu zaznaczonego pola.

## Filtr oparty na zaznaczeniu

W przypadku gdy zaistnieje potrzeba wyświetlenia rekordów zawierających w jednym z pól pewną wartość, można skorzystać z możliwości filtrowania rekordów opartego na zaznaczeniu. Wykonaj następujące zadania w celu zapoznania się z tą możliwością.

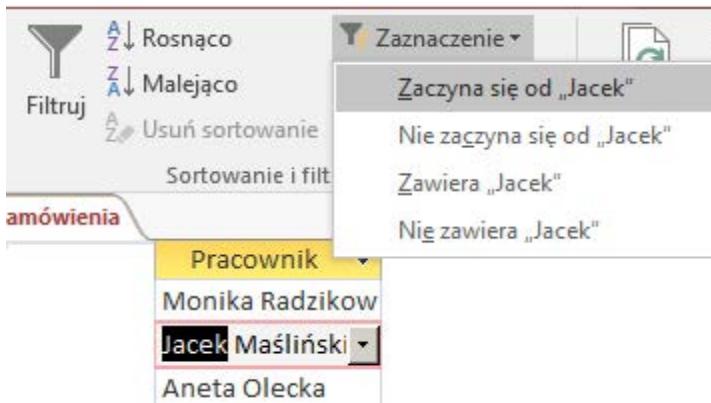
### Zadanie 6\_6

Wyświetl dane z tabeli **Zamówienia** dotyczące zamówień wykonanych przez pracowników o imieniu „Jacek”.

---

#### Wykonanie

- Otwórz tabelę **Zamówienia** w **widoku arkusza danych**.
- W polu *Pracownik* zaznacz imię „Jacek”.
- Wybierz **Zaznaczenie**, otwórz listę poleceń i wybierz **Zaczyna się od „Jacek”** (rys. 6.8).



Rys. 6.8. Zastosowanie filtru opartego na zaznaczeniu

Aby skasować filtr i wyświetlić wszystkie rekordy, wybierz na zakładce **Narzędzia główne** wstążki ikonkę **Przełącz filtr**.

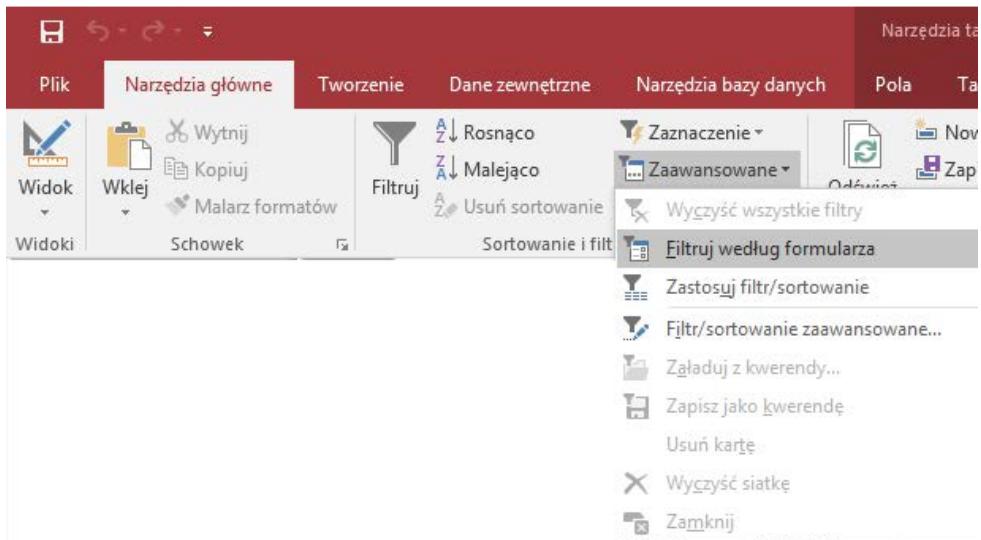
Warto zauważyć, że polecenia „**Zaczyna się**”, „**Nie zaczyna się**”, „**Zawiera**”, „**Nie zawiera**” są dostępne również w menu kontekstowym pola, widocznym po jego kliknięciu prawym przyciskiem myszy.

## 6.3. Metoda filtrowania danych według formularza

Ten sposób filtrowania rekordów jest stosowany, gdy warunek przeszukiwania jest złożony i bazuje na dwóch lub więcej polach tabeli.

Polecenie **Filtruj według formularza** jest umieszczone na zakładce **Narzędzia główne** wstążki w grupie **Sortowanie i filtrowanie** wśród innych poleceń menu **Zaawansowane** (rys. 6.9). Jest ono dostępne, gdy którakolwiek tabela jest otwarta w **widoku arkusza danych**.

W przypadku tego rodzaju filtrowania użytkownik powinien wypełnić pola zaproponowanego przez program formularza, wprowadzając do nich kryteria filtrowania. Kryterium filtrowania jest pewnym wyrażeniem. Jego postać zależy od typu danych pola.



Rys. 6.9. Polecenie **Filtruj według formularza**

### Zadanie 6\_7

Wykonaj filtrowanie tabeli **Klienci** metodą **filtruj według formularza**: wyświetl rekordy, których pole **Tytuł zawodowy** zawiera tekst „**Właściciel**”, a pole **Województwo** zawiera tekst „**mazowieckie**”.

#### Wykonanie

- Otwórz tabelę **Klienci**.

- b. Na zakładce **Narzędzia główne** w grupie **Sortowanie i filtrowanie** wybierz **Zaawansowane**, a potem polecenie **Filtruj według formularza** (rys. 6.9) – zostanie wyświetlony pusty arkusz danych.
- c. Wprowadź wartości kryteriów do komórek (tzn. pół formularza), jak to przedstawiono na rysunku 6.10.

Tytuł zawodowy	Telefon służby	Telefon domowy	Telefon komórkowy	Numer faksu	Adres	Miasto	Województwo
"Właściciel"							"Mazowieckie"

Rys. 6.10. Okienko **filtrowania według formularza**

- d. Wybierz z listy rozwijanej **Zaawansowane** (rys. 6.9) polecenie **Zastosuj filtr/sortowanie** – program wyświetli wszystkie rekordy zgodnie ze zdefiniowanym kryterium.
- e. Aby usunąć filtr i wyświetlić wszystkie rekordy, wybierz polecenie .

Aby zmodyfikować filtr, należy wybrać ponownie polecenie **Filtruj według formularza**. Zostanie wyświetlony zestaw kryteriów bieżącego filtru.

Należy pamiętać, że zestaw wyrażeń umieszczonych w kilku kolumnach jednej strony arkusza danych jest traktowany przez program jak jedno wyrażenie, w którym wszystkie kryteria są połączone za pomocą logicznej funkcji **AND**.

**Filtrowanie według formularza** może być zrealizowane za pomocą jednego, a wielu formularzy – każdy z nich będzie ulokowany na innej zakładce. Ta możliwość jest wykorzystywana w sytuacji, gdy pewne kryteria trzeba połączyć logiczną funkcją **LUB**. Rozważmy dalej odpowiednie zadanie.

## Zadanie 6\_8

Wykonaj filtrowanie tabeli **Klienci** metodą **filtrowania według formularza**, umieszczając kryterium na dwóch zakładkach. Wyświetl rekordy dotyczące właścicieli firm mających na imię „Karolina” lub „Janusz”.

### Wykonanie

- a. Otwórz tabelę **Klienci** w **widoku arkusza danych**.

Klienci: Filtruj według formularza

ID	Firma	Nazwisko	Imię	Adres e-mail	Tytuł zawodowy
			"Karolina"		"Właściciel!"

Wyszukaj Lub

Rys. 6.11. Utworzenie kryterium na dwóch zakładkach – tu przedstawiono widok pierwszej z nich

- b. Wybierz polecenie **Filtruj według formularza**.
- c. Wyczyść siatkę formularza: wybierz z listy **Zaawansowane** polecenie **Wyczyszczyć siatkę**.
- d. Wprowadź kryteria, jak pokazano na rysunku 6.11.
- e. Kliknij myszką zakładkę **Lub** i zdefiniuj drugą część kryterium, jak to przedstawiono na rysunku 6.12.

Klienci: Filtruj według formularza

ID	Firma	Nazwisko	Imię	Adres e-mail	Tytuł zawodowy
			"Janusz"		"Właściciel!"

Rys. 6.12. Utworzenie kryterium na dwóch zakładkach – tu przedstawiono widok drugiej z nich

## 6.4. Informacje na temat kryteriów filtrowania

W przedstawionych powyżej przykładach zaprezentowano metody tworzenia najprostszych filtrów. Jednak najważniejsza jest wiedza na temat tworzenia złożonych kryteriów filtrowania. Będzie dalej również potrzebna przy wykonaniu kwerend jak w środowisku graficznym, tak i za pomocą języka SQL.

Kryterium filtrowania może się składać z wielu wyrażeń zapisanych w polach **arkusza danych**. Stąd wynika potrzeba poznania sposobu tworzenia tych wyrażeń.

W dokumentacji do programu **Access** można znaleźć szczegółowy opis technik tworzenia wyrażeń, a także tabele z wykazem obsługiwanych przez program operatorów. Tabele te zostały zamieszczone w tekście poniżej.

**Operatory arytmetyczne** (tab. 6.1) służą do obliczania wartości z dwóch lub więcej liczb albo do zmiany znaku liczby z plus na minus.

Tab. 6.1. Operatory arytmetyczne

Operator	Przeznaczenie	Przykład
+	Suma dwóch liczb	[Wpłata] + [Dodatek]
-	Różnica dwóch liczb lub oznaczenie ujemnej wartości liczby	[Wpłata] - [Rabat]
*	Iloczyn dwóch liczb	[Wpłata]*0.25
/	Iloraz dwóch liczb	[Suma]/[Liczba]
\	Zaokrąglenie obu liczb do wartości całkowitych, a następnie podzielenie pierwszej liczby przez drugą i obcięcie wyniku do liczby całkowitej	[Kwota]\[Liczba]
Mod	Reszta z dzielenia pierwszej liczby przez drugą	[Zakwaterowane] Mod [Pokoje]
^	Podniesienie liczby do potęgi podanej w wykładniku	Liczba ^ Wykładnik

**Operatory porównywania** (tab. 6.2) służą do porównywania wartości i zwracania wyniku w postaci „prawda”, „fałsz” lub „Null”.

Tab. 6.2. Operatory porównywania

Operator	Przeznaczenie
<	Określenie, czy pierwsza wartość jest mniejsza niż druga
<=	Określenie, czy pierwsza wartość jest mniejsza niż druga lub jej równa
>	Określenie, czy pierwsza wartość jest większa niż druga
>=	Określenie, czy pierwsza wartość jest większa niż druga lub jej równa
=	Określenie, czy pierwsza wartość jest równa drugiej
<>	Określenie, czy pierwsza wartość nie jest równa drugiej

We wszystkich przypadkach, jeśli jedna wartość jest pusta (Null), wynik jest również pusty (Null). Ponieważ wartość pusta oznacza wartość nieznaną, wynik porównania z taką wartością jest również nieznany.

**Operatory logiczne** (tab. 6.3) służą do łączenia dwóch wartości i zwracania wyniku w postaci „prawda”, „fałsz” lub „Null”.

Tab. 6.3. Operatory logiczne

Operator	Zastosowanie	Opis
And	Wyr1 And Wyr2	Zwraca wartość „prawda”, jeśli Wyr1 i Wyr2 mają wartość „prawda”
Or	Wyr1 Or Wyr2	Zwraca wartość „prawda”, jeśli Wyr1 lub Wyr2 ma wartość „prawda”
Eqv	Wyr1 Eqv Wyr2	Zwraca wartość „prawda”, jeśli Wyr1 i Wyr2 mają wartość „prawda” lub Wyr1 i Wyr2 mają wartość „fałsz”
Not	Not Wyr	Zwraca wartość „prawda”, jeśli Wyr nie jest „prawda” lub zwraca „fałsz”, gdy Wyr jest „prawda”
Xor	Wyr1 Xor Wyr2	Zwraca wartość „prawda”, jeśli jedno z wyrażeń Wyr1 lub Wyr2 ma wartość „prawda”, ale nie oba

**Operatory łączenia tekstów** (tab. 6.4) służą do sklejania dwóch wartości tekstowych w jedną.

Tab. 6.4. Operatory łączenia tekstów

Operator	Zastosowanie	Opis
&	ciag1 & ciag2	Łączy dwa ciągi w jeden
+	ciag1 + ciag2	Łączy dwa ciągi w jeden, informując o wartościach pustych

**Operatory specjalne** przedstawiono w tabeli 6.5.

Tab. 6.5. Operatory specjalne

Operator	Opis
Is Null lub Is Not Null	Określa, czy wartość jest pusta ( <i>Null</i> ), czy też nie jest pusta ( <i>Is Not Null</i> )
Like "wzorzec"	Określa zgodność wartości i „wzorca” z uwzględnieniem symboli wieloznacznych we „wzorcu”.
Between wartość1 And wartość2	Określa, czy wartość liczbową lub wartość daty mieści się w określonym zakresie
In(ciag1, ciag2...)	Określa, czy dana wartość należy do przedstawionego zbioru wartości

Przykłady wyrażeń przedstawiono także w tabelach 6.6 - 6.8. Zamieszczone w nich wzory wyrażeń będą użyteczne przy wykonywaniu kolejnych zadań z tego i z kolejnych rozdziałów.

### Przykłady kryteriów dla pól liczbowych

Przykłady wyrażeń zbudowanych na danych liczbowych przedstawiono w tabeli 6.6.

Tab. 6.6. Przykłady wyrażeń dla danych liczbowych

Przykład wyrażenia dla danych liczbowych	Objaśnienie
120	Wartość jest równa 120
= 120	
<> 120	Wartość nie jest równa 120
Not 120	
> 120	Wartość jest większa od 120
< 120	Wartość jest mniejsza od 120
>= 120	Wartość jest większa <b>lub</b> równa 120
<= 120	Wartość jest mniejsza <b>lub</b> równa 120
120 Or 220	
In (120; 220)	Wartość jest równa 120 <b>lub</b> 220
(<>120) And (<>200)	
Not(120 Or 220)	Wartość jest nie równa 120 <b>ani</b> 220
<>(120 Or 220)	
>= 120 And <= 220	Wartość jest z zakresu <120; 220>, tzn. jest większa lub równa 120 <b>i</b> jednocześnie jest mniejsza lub równa 220
Between120 And 220	
> 120 And < 220	Wartość jest większa od 120 <b>oraz</b> mniejsza od 220
< 120 Or > 220	Wartość jest mniejsza od 120 <b>albo</b> większa od 220 (tzn. jest spoza zakresu <120; 220>)
Not(>= 120 And <= 220)	
< 120 Or >= 220	Wartość jest mniejsza od 120 <b>albo</b> równa lub większa od 220
<= 120 Or >220	Wartość jest mniejsza lub równa 120 <b>albo</b> większa od 220
<= 120 Or 220	Wartość jest równa lub mniejsza od 120 <b>albo</b> równa 220
>= 120 And < 220	Wartość jest większa lub równa 120 <b>oraz</b> mniejsza od 220
> 120 And <= 220	Wartość jest większa od 120 <b>oraz</b> mniejsza lub równa 220

### Przykłady kryteriów dla pól zawierających daty

W wyrażeniu, które posłuży jako kryterium filtrowania pól zawierających daty, mogą być użyte operatory porównania, logiczne i specjalne. W tabeli 6.7 przedstawiono przykłady takich wyrażeń.

Zapis daty w środowisku programu **Access** różni się od danych innych typów tym, że każda data w wyrażeniu powinna zaczynać się i kończyć znakiem #. Między znakiem # i liczbą nie wolno używać spacji.

Tab. 6.7. Przykłady wyrażeń dla danych typu **Data/Godzina**

Przykład wyrażenia dla danych typu <b>Data/Godzina</b> (format <b>Data krótka</b> )	Objaśnienie
#2001-10-25#	Data jest równa 2001-10-25
=#2001-10-25#	
Not #2001-10-25#	Wszystkie daty oprócz dnia 2001-10-25
<>#2001-10-25#	
< #1996-1-1#	Daty sprzed roku 1996
<= #1995-12-31#	
>= #1991-01-01#	Daty po roku 1991
> #1990-12-31#	
>= #2014-10-01# And <= #2015-01-01#	Daty z zakresu czasu <2014-10-01; 2015-01-01>
Between #2014-10-01# And #2015-01-01#	
<#2014-10-01# OR >#2015-01-01#	
Not (>= #2014-10-01# AND <= #2015-01-01#)	Daty spoza zakresu czasu <2014-10-01; 2015-01-01>
Not Between #2014-10-01# And #2015-01-01#	
> #1996-12-31# And < #1998-1-1#	
>= #1997-1-1# And <= #1997-12-31#	Daty z roku 1997
Between #1997-1-1# And #1997-12-31#	
Between #1995-1-5# And #1995-1-10#	Dowolna data z zakresu dni od 5 stycznia 1995 roku do 10 stycznia 1995 roku
>= #1995-1-5# And <= #1995-1-10#	
< #1997-1-1# Or > #2000-12-31#	Dowolna data spoza zakresu lat <1997 ; 2000>
In(#2001-10-12#; #2001-11-22#)	Data jest równa jednej z dwóch: 2001-10-12 lub 2001-11-22 Można zastosować <b>In</b> do dowolnej skończonej liczby dat
Not In(#2001-10-12#; #2001-11-22#)	Nie jest to data 2001-10-12 ani 2001-11-22

### Przykłady kryteriów dla pól zawierających dane tekstowe

Każdą wartość tekstową w wyrażeniu należy ujmować w znaki cudzysłowu. W prostych wyrażeniach program dodaje je automatycznie.

Mogimy używać w wyrażeniu także symboli wieloznacznych. Są to: \* (**gwiazdka**) oraz ? (**znak zapytania**).

Symbol wieloznaczny \* w wyrażeniu oznacza dowolną liczbę dowolnych symboli.

Symbol wieloznaczny ? w wyrażeniu oznacza dowolny pojedynczy znak.

Aby nie wymieniać kolejnej litery alfabetu, można stosować zapis w postaci **[Litera1-Litera2]**. Taki zapis oznacza kolejne litery alfabetu, zaczynając od **Litera1**, a kończąc **Litera2**. Na przykład wyrażenie **Like "[C-E]"** można wykorzystać do przedstawienia liter **C**, **D** lub **E**.

W tabeli 6.8. przedstawiono przykłady wyrażeń dla danych tekstowych.

Tab. 6.8. Przykłady wyrażeń dla danych tekstowych

Przykład wyrażenia dla danych tekstowych	Objaśnienie
<i>Like "Adam"</i>	Wartość tekstowa jest równa "Adam"
"Adam"	
<i>Like "Piotr" Or "Marek"</i>	Wartość tekstowa jest równa "Piotr" lub "Marek"
"Piotr" Or "Marek"	
<i>In("Adam"; "Piotr"; "Marek")</i>	Wartość tekstowa jest równa "Adam" lub "Piotr", lub "Marek"
<i>Not In("Adam"; "Piotr"; "Marek")</i>	Wartość tekstowa nie jest równa "Adam" ani "Piotr", ani "Marek"
<i>Not "Mechaniczny"</i>	Dowolny tekst oprócz "Mechaniczny"
<i>Not Like "Mechaniczny"</i>	
<i>Like "D*"</i>	Wartość tekstowa zaczyna się od litery "D"
<i>Not Like "D*"</i>	Wartość tekstowa nie zaczyna się od litery "D"
<i>Not "D*"</i>	
<i>Like "*d"</i>	Wartość tekstowa kończy się literą "D"
<i>Like "An*"</i>	Wartość tekstowa zaczyna się od "An"
<i>Like "A*" Or "D*"</i>	Wartość tekstowa zaczyna się od litery "A" lub od litery "D"
<i>Like "[A-D]*"</i>	Wartość tekstowa zaczyna się od litery "A" lub "B", lub "C", lub "D"
<i>Like "Biał? "</i>	Wartości tekstowe, które mają dokładnie 5 liter, pierwsze 4 to są "Biały", a ostatnia dowolna
<i>Like "K?wal*"</i>	Wartości tekstowe, zaczynające się od litery "K", których druga litera jest dowolna, następnie są litery "wal" i dowolny ciąg znaków.

Prostym przykładem zastosowania operatora **Or** jest filtr utworzony w kolejnym zadaniu.

## Zadanie 6\_9

Wykonaj filtrowanie tabeli **Klienci** metodą **filtruj według formularza**, umieszczając kryterium na jednej zakładce. Wyświetl rekordy dotyczące właścicieli firm mających na imię „Karolina” lub „Janusz”.

Na rysunku 6.13 przedstawiono filtr w postaci jednego wyrażenia zawierającego logiczny operator **OR**.

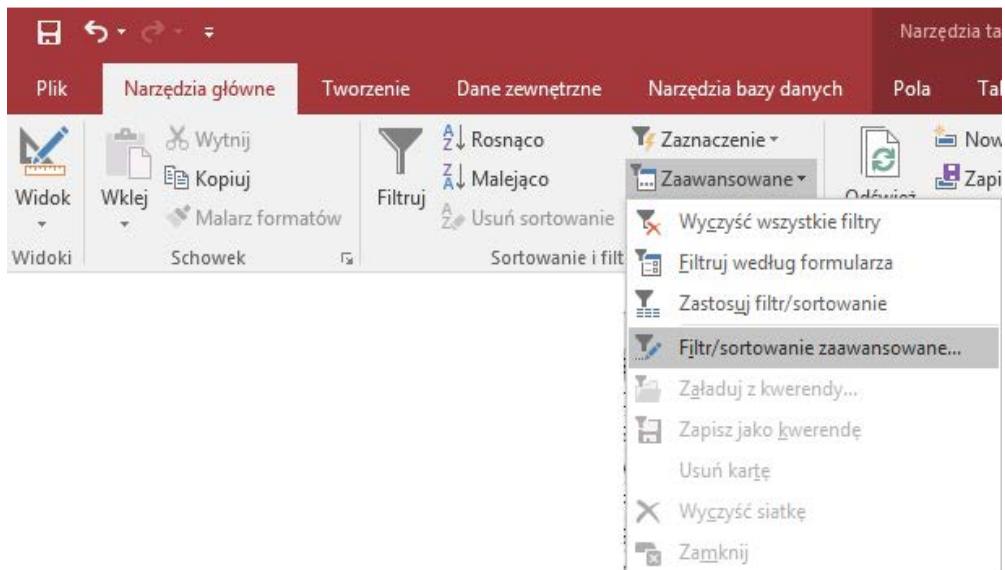
ID	Firma	Nazwisko	Imię	Adres e-mail
			"Karolina" Or "Janusz"	

Rys. 6.13. Zdefiniowanie kryterium za pomocą operatora **OR**

W komórkach formularza do filtrowania możemy utworzyć bardziej złożone wyrażenia niż pokazane w zadaniach 6.7 - 6.9. Gdy jednak zachodzi potrzeba utworzenia rozbudowanego filtra, najlepiej zastosować filtr zaawansowany.

## 6.5. Filtr zaawansowany

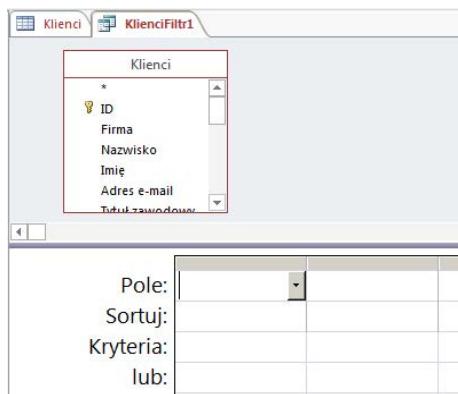
Wywołujemy filtr zaawansowany dla tabeli otwartej w widoku arkusza danych, klikając w zakładce **Narzędzia główne**, w grupie **Sortowanie i filtrowanie**, przycisk **Zaawansowane**, a następnie wybierając polecenie **Filtr/sortowanie** (rys. 6.14).



Rys. 6.14. Wywołanie filtru zaawansowanego

Przy definiowaniu **filtru zaawansowanego**, oprócz sformułowania złożonego kryterium wyszukiwania, możemy także wymusić na danych sortowanie rekordów (rosnąco lub malejąco).

Polecenie **Filtr/sortowanie zaawansowane** powoduje wyświetlenie na ekranie pustej siatki przeznaczonej do budowy filtru (rys. 6.15), w której widzimy okienko z listą pól tabeli. Na rysunku 6.15 przedstawiono pustą siatkę **filtru zaawansowanego** tabeli **Klienci**. Siatka projektu jest przeznaczona do wprowadzenia kryteriów filtrowania. Po wypełnieniu siatki uruchamiamy filtr, wybierając w menu **Zaawansowane** polecenie **Zastosuj filtr/sortowanie** (rys. 6.14).



Rys. 6.15. Pusta siatka projektu filtru zaawansowanego

Aby wrócić po filtrowaniu do wszystkich rekordów, należy skorzystać z polecenia **Przełącz filtr**.

Za pomocą polecenia **Filtr/sortowanie zaawansowane** można ponownie otworzyć projekt ostatnio wykonanego **filtru zaawansowanego**.

Przed utworzeniem kolejnego filtru należy, podobnie jak przy filtrowaniu według formularza, wyczyścić siatkę za pomocą polecenia **Wyczyść siatkę**.

Budując filtr, należy brać pod uwagę, że kryteria umieszczone w kolumnach łączy logiczna funkcja **AND**. Gdy w jednej kolumnie ulokowano więcej niż jedno wyrażenie, to łączy je logiczna funkcja **OR**.

Zapoznaj się z tym rodzajem filtrowania, wykonując kolejne zadanie.

## Zadanie 6\_10

Wykonaj filtrowanie tabeli **Klienci** za pomocą filtru zaawansowanego: wyświetl rekordy dotyczące właścicieli firm mających na imię „Anna”, „Karolina” lub „Janusz”.

Posortuj dane rosnąco według pola *Województwo*.

## Wykonanie

- Otwórz tabelę **Klienci** w **widoku arkusza danych**.
- Upewnij się, że tabela nie jest odfiltrowana. W tym celu na pasku nawigatora rekordów sprawdź, czy jest wyświetlany przycisk **Filtrowane**. Jeśli tak, to w menu **Zaawansowane** wybierz polecenie **Wyczść wszystkie filtry** (rys. 6.14). Jeśli polecenie **Wyczść wszystkie filtry** jest wygaszone, oznacza to, że nie działają żadne filtry.
- Wybierz w menu **Zaawansowane** polecenie **Filtr/sortowanie zaawansowane** (rys. 6.14). Na ekranie natychmiast pojawi się okno filtrowania (rys. 6.15).
- W okienku pól tabeli **Klienci** kliknij dwukrotnie nazwę każdego pola, które należy umieścić w polach siatki. Polami tymi są: *Województwo*, *Tytuł zawodowy*, *Imię*.
- W kolumnie *Województwo* w wierszu **Sortuj** ustaw opcję **Rosnąco** (rys. 6.16).
- W kolumnie *Tytuł zawodowy* w wierszu **Kryteria** wprowadź: "Właściciel" (rys. 6.16).
- W kolumnie *Imię* w wierszu **Kryteria** wprowadź: "Anna" Or "Karolina" Or "Janusz" (rys. 6.16).
- Kliknij przycisk **Zastosuj filtr/sortowanie**, aby wyświetlić filtrowane wiersze.

Pole:	Województwo	Tytuł zawodowy	Imię
Sortuj:	Rosnąco		
Kryteria:		"Właściciel"	"Anna" Or "Karolina" Or "Janusz"
lub:			

Rys. 6.16. Siatka projektu filtru zaawansowanego do **Zadania 6\_10**

Należy zauważyć, że nie możemy w kolumnie *Imię* zamiast wyrażenia logicznego (**OR**) w jednym wierszu zapisać kryterium w trzech wierszach, jeżeli równocześnie nie zapiszemy także trzykrotnie kryterium w kolumnie *Tytuł zawodowy*.

Drugi sposób wypełnienia siatki projektu do zadania 6\_10 prezentuje rysunek 6.17.

Pole:	Województwo	Tytuł zawodowy	Imię
Sortuj:	Rosnąco		
Kryteria:		"Właściciel"	"Anna"
lub:		"Właściciel"	"Karolina"
		"Właściciel"	"Janusz"

Rys. 6.17. Siatka projektu filtru zaawansowanego do **Zadania 6\_10** - drugi sposób.

W dalszej części ćwiczeń baza danych **Northwind** nie będzie wykorzystywana, więc **należy ją zamknąć**.

## 6.6. Przykłady złożonych kryteriów filtrowania

Kryterium filtrowania może stanowić wyrażenie zawierające różne operatory. Ponieważ będziemy pracować z bazą danych Dom Studenta, należy ją otworzyć. Wykonane w niej tabele i wprowadzone dane będą służyć jako dane testowe do dalszych ćwiczeń.

Na początku należy nadmienić, że każde z kolejnych zadań powinno być wykonane na konkretnych danych. Filtrowanie oznacza wyszukiwanie, więc w naszych tabelach powinniśmy mieć pewne dane, aby móc je wyświetlić za pomocą filtrowania.

Wszystkie następne zadania, w tym i w kolejnych rozdziałach, będą dotyczyć bazy danych Dom Studenta. W każdym zadaniu zostaną wymienione konkretne wartości, których może nie być w twoich tabelach. W celu prawidłowej pracy z bazą danych należy dopasować każde zadanie do swoich danych.

Należy wykonywać zadania na danych, dbając o zgodność z warunkami zadania. Nie trzeba za każdym razem zmieniać danych w tabelach, lecz wykonywać zadania o zmienionych warunkach. Podczas wykonania zadania 6\_11 zostanie zaprezentowane, jak należy postępować.

### Zadanie 6\_11

---

Wykonaj filtrowanie według formularza tabeli **tOsoba** z bazy danych Dom Studenta. Wyświetl wiersze z osobami o jednym z dwóch zadanych imion (przykładowo: „Marek” lub „Irena”), których numer albumu jest liczbą ze zdefiniowanego zakresu (przykładowo, <80000; 90000>). Po sprawdzeniu działania skasuj filtr.

---

#### Wykonanie

- a. Otwórz bazę danych Dom Studenta.
- b. Otwórz tabelę **tOsoba** w widoku arkusza danych.
- c. Wybierz polecenie **Filtruj według formularza**.
- d. Wprowadź kryteria, jak przedstawiono na rysunku 6.18, jeśli w tabeli **tOsoba** są takie wartości, jak podano w zadaniu.

Jeżeli tabela **tOsoba** nie zawiera takich wierszy, nie wpisz w formularzu wartości „Marek”, „Irena” itp. Zamiast nich wpisz takie, które występują w Twojej tabeli **tOsoba**.

tOsoba: Filtruj według formularza				
ID	Nazwisko	Imiona	Album	Płeć
	"Marek" Or "Irena"		>=80000 And <=90000	

Rys. 6.18. Filtr według formularza do wykonania **Zadania 6\_11**

Kolejne zadanie 6\_12 jest przykładem filtrowania zaawansowanego o złożonym kryterium.

## Zadanie 6\_12

Zbuduj filtr zaawansowany dla tabeli **tOsoba** z bazy danych Dom Studenta w celu wyświetlenia rekordów dotyczących osób spoza województwa podlaskiego, których PESEL zaczyna się od liczb 94 lub 96, a numer albumu jest większy od 85555. Dane należy uporządkować rosnąco według nazwiska i imienia.

Po sprawdzeniu działania skasuj filtr.

**Zadanie 6\_12** należy wykonywać według tego samego algorytmu co i **Zadanie 6\_10**, inne natomiast będą kryteria filtrowania.

### Wykonanie

- Sprawdź, czy w twojej tabeli są wymienione w zadaniu wartości. Jeśli nie, to załóż, których danych zamiast wymaganych będziesz używać przy filtrowaniu.
- Utwórz filtr zaawansowany, jak przedstawiono na rysunku 6.19.

Pole:	Nazwisko	Imiona	Województwo	PESEL	Album
Sortuj:	Rosnąco	Rosnąco			
Kryteria:			Not Like "podlaskie"	Like "94*" Or Like "96*"	>85555
lub:					

Rys. 6.19. Filtr zaawansowany do wykonania **Zadania 6\_12**

## Zadania do samodzielnego wykonania

## Zadanie 6\_13

Zastosuj **filtrowanie według formularza** do wyświetlania tych wierszy z tabeli **tZakwaterowanie**, w których wartość w polu

Data\_zakwaterowania jest większa od daty 15-10-2018 i dotyczy Wydziału Mechanicznego. Skasuj filtr.

---

### Zadanie 6\_14

Zastosuj **filtrowanie według formularza** do wyświetlania tych wierszy z tabeli **tZakwaterowanie**, w których wartość w polu Data\_zakwaterowania jest z zakresu od 01-10-2017 do 10-10-2017. Skasuj filtr.

---

### Zadanie 6\_15

Zastosuj **filtrowanie według formularza** do wyświetlania wierszy z tabeli **tZakwaterowanie**, w których wartość w polu Data\_zakwaterowania nie dotyczy dni 01-10-2017 i 08-10-2017. Skasuj filtr.

---

### Zadanie 6\_16

Zastosuj **filtrowanie według formularza** do wyświetlania z tabeli **tOsoba** tych wierszy, które zawierają dane o wszystkich osobach płci męskiej i nazwiskach zaczynających się od litery „A” lub „B”. Skasuj filtr.

---

### Zadanie 6\_17

Zastosuj **filtrowanie według formularza** do wyświetlania z tabeli **tOsoba** tych wierszy, które zawierają dane o wszystkich osobach o nazwiskach kończących się na „cz” lub na „i”, a numer albumu jest liczbą z zakresu <82333; 92333>. Skasuj filtr.

---

### Zadanie 6\_18

Zastosuj **filtr zaawansowany** do wyświetlania tych wierszy z tabeli **tZakwaterowanie**, w których pole Data\_wykwaterowania jest puste (kryterium: *Is Null*), a data zakwaterowania jest z roku 2018. Rekordy należy posortować rosnąco według pola Data\_zakwaterowania. Skasuj filtr.

---

## Zadanie 6\_19

Zastosuj **filtr zaawansowany** do wyświetlania wierszy z tabeli **tOsoba** uporządkowanych rosnąco według numerów PESEL.

Wiersze powinny dotyczyć osób:

- o mających na imię „Elżbieta” lub „Małgorzata”, lub „Marek”;
- o których nazwisko zaczyna się od liter „M” lub „T”;
- o zamieszkałych na stałe w Białymstoku.

Skasuj filtr.

## Zadanie 6\_20

Zastosuj **filtr zaawansowany** do wyświetlania wierszy z tabeli **tOsoba** uporządkowanych według numerów albumów. Wiersze powinny spełniać następujące warunki:

- o pole *PESEL* wskazuje na urodzenie osoby w miesiącu kwietniu lub w miesiącu czerwcu;
- o pole *NIP* jest puste;
- o wartość pola *Album* jest liczbą spoza zakresu <70000, 72000>.

Skasuj filtr.

## Zadanie 6\_21

Zastosuj **filtr zaawansowany** do wyświetlania wierszy z tabeli **tOsoba** uporządkowanych według kraju, województwa, miasta. Wiersze powinny spełniać następujące warunki:

- o wartość numeru albumu jest wartością mniejszą od 50000 lub z zakresu <56000; 57000>;
- o nazwisko osoby powinno kończyć się na literę „C” lub „D”, lub „E”.

Skasuj filtr.

## Zadanie 6\_22

Zastosuj **filtr zaawansowany** do wyświetlania wierszy z tabeli **tZakwaterowanie** uporządkowanych według daty zakwaterowania.

Wiersze powinny spełniać następujące warunki:

- o data zakwaterowania powinna być z roku 2016 lub 2018;
- o pole *Wydział* nie może zawierać wartości „Mechaniczny”.

Skasuj filtr.

## 7. Kwerenda wybierająca

### 7.1. Wprowadzenie

Kwerendy są najważniejszym narzędziem wykorzystywany w pracy z bazami danych. Służą do przeglądania, zmiany i analizy danych. Można ich również używać jako źródła rekordów dla innych obiektów bazy danych.

Kwerendę można utworzyć, stosując język SQL, a także za pomocą narzędzi interfejsu graficznego. W tym rozdziale omawiana jest metoda zbudowania kwerendy w postaci graficznej. Przy tym zaproponowano tworzenie kwerendy „od podstaw”, tzn. w **widoku projektu**.

Najczęściej używanym typem kwerendy jest kwerenda wybierająca. Kwerenda wybierająca pobiera dane z jednej tabeli lub kilku tabel przy użyciu kryteriów określonych przez użytkownika, a następnie wyświetla je w żądanym porządku. W wyniku wykonania kwerendy wybierającej otrzymuje się dynamiczny zestaw danych wynikowych, który nie jest zapamiętywany na stałe w bazie danych. Wyświetlany jest on w postaci arkusza danych.

Kwerendę wybierającą budujemy w oknie projektu według następującego schematu:

- w polu wstążki w zakładce **Tworzenie** wybieramy ikonkę **Projekt kwerendy**;
- w okienku, które się pojawi, określamy tabele lub/ oraz kwerendy, których pola będą wyświetlane lub będą potrzebne do generowania kryteriów;
- umieszczamy te pola w siatce projektu w pewnej kolejności;
- definiujemy kryteria wyszukiwania;
- określamy pola, według których ewentualne dane będą uporządkowane;
- zaznaczamy, które pola spośród obecnych w siatce będą pokazywane, bądź też nie;
- uruchamiamy kwerendę.

Kryterium kwerendy definiujemy na tych samych zasadach co i dla filtra zaawansowanego. W przedstawionych dalej przykładach zaprezentowano tworzenie kwerend wybierających o złożonych kryteriach, zawierających oprócz już znanych konstrukcji funkcje wbudowane programu **Microsoft Access**.

## 7.2. Kwerenda wybierająca zbudowana na jednej tabeli

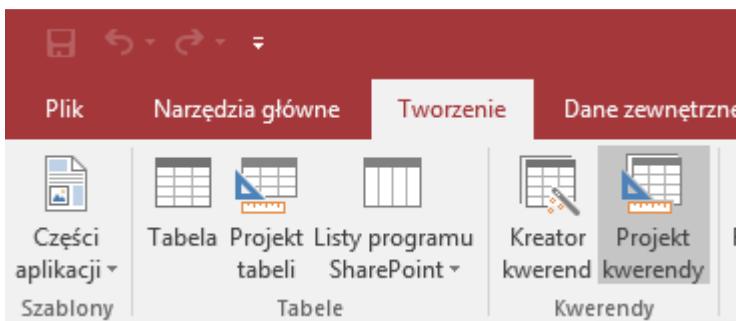
Rozważmy przykłady tworzenia kwerend przeznaczonych do wyszukiwania danych w jednej tabeli. W tym celu wykonaj **Zadania 7\_1 – 7\_8**.

### Zadanie 7\_1

Zbuduj kwerendę do wyświetlenia z tabeli **tOsoba** nazwiska, imienia oraz numeru albumu każdej osoby o nazwisku zaczynającym się od litery „K” lub „M”, zamieszkałej w Białymstoku, której pole numeru NIP nie jest puste. Wyświetlane dane należy posortować alfabetycznie – według nazwiska i imienia. Zapisz kwerendę, nadając jej nazwę **k7\_1**.

### Wykonanie

- a. W oknie bazy danych Dom Studenta wybierz na wstążce kartę **Tworzenie**, a na niej ikonkę **Projekt kwerendy** (rys. 7.1). Wówczas zostanie wyświetlone okno dialogowe **Pokazywanie tabeli** (rys. 7.2).

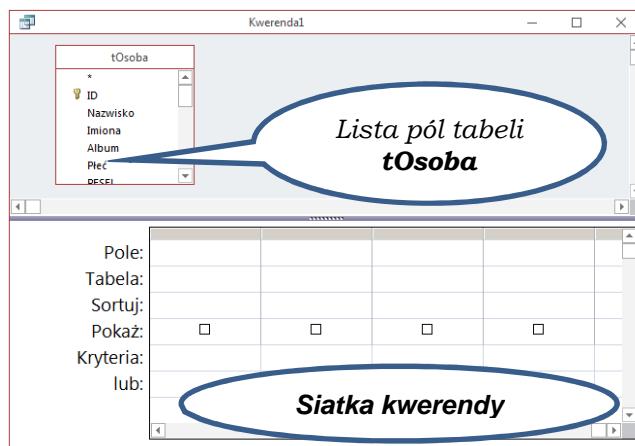


Rys. 7.1. Widok ikonki **Projekt kwerendy** w grupie **Kwerendy** na karcie **Tworzenie**

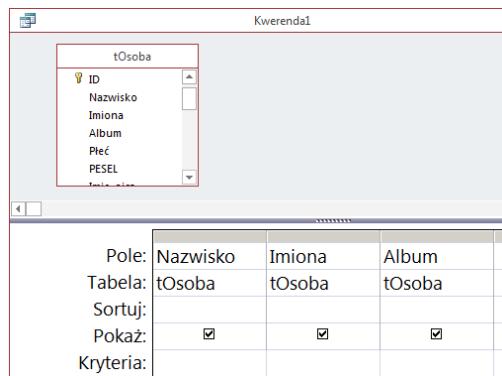


Rys. 7.2. Okienko **Pokazywanie tabeli**

- b. W oknie **Pokazywanie tabeli** (rys. 7.2) na karcie **Tabele** kliknij nazwę **tOsoba**, a potem **Dodaj** i **Zamknij** – zostanie wyświetcone okno projektu kwerendy. Składa się ono z listy pól tabeli **tOsoba** oraz siatki przeznaczonej do zdefiniowania kryteriów wyświetlania danych z tej tabeli (rys. 7.3).
- c. Następnie należy umieścić w siatce kwerendy nazwy wszystkich pól, których wartości mają być wyświetlane, tzn. pola *Nazwisko*, *Imiona*, *Album* (rys. 7.4), zostawiając zaznaczenie  w wierszu **Pokaż**. Aby przenieść pole do siatki, kliknij dwukrotnie jego nazwę w okienku z listą pól tabeli (można również przeciągnąć nazwę pola wskaźnikiem myszy lub wybrać z listy rozwijanej **Pole** w siatce kwerendy).
- d. W siatce kwerendy w wierszu **Sortuj** przy polach *Nazwisko* oraz *Imię* z listy rozwijanej wybierz opcję **Rosnąco** (rys. 7.5).



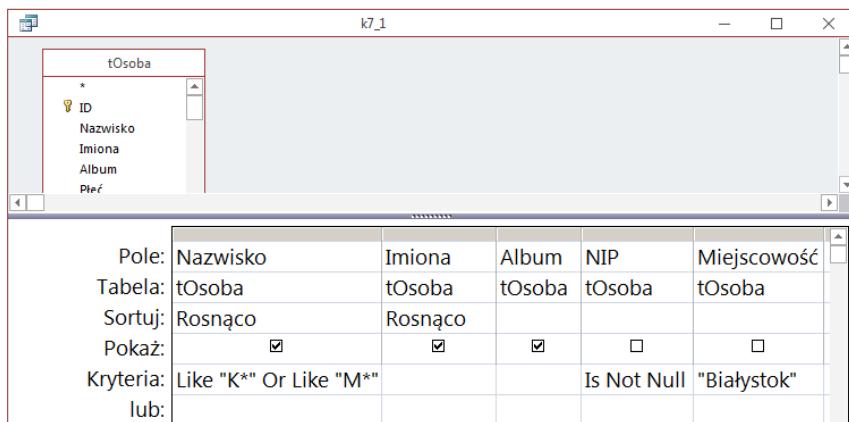
Rys. 7.3. Okno pustego projektu kwerendy



Rys. 7.4. Okno projektu kwerendy zawierającego trzy pola do wyświetlania

- e. Zdefiniuj kryteria wyszukiwania. W tym celu należy umieścić w siatce pola *NIP* i *Miejscowość* (rys. 7.5). Ponieważ nie jest wymagane, aby wartości tych pól były wyświetlane, skasuj zaznaczenie  w wierszu **Pokaż** dla tych pól.

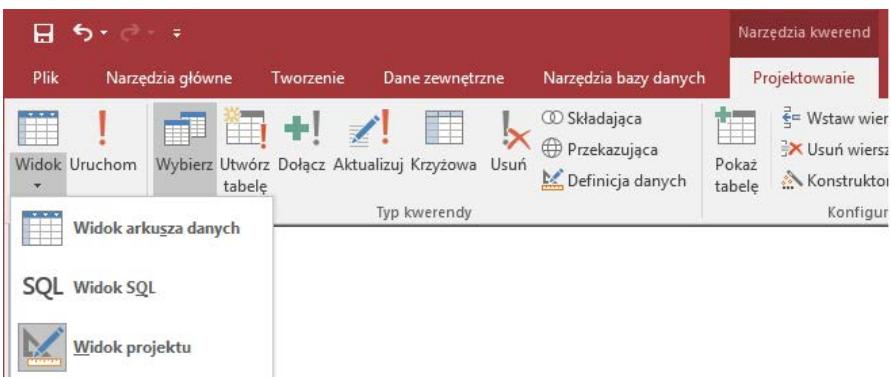
Kryteria wprowadzamy tak samo jak w oknie **filtra zaawansowanego**. Warto zaznaczyć, że kryterium w polu *Nazwisko* tworzymy bez słowa **Like** oraz znaków cudzysłowu: K\* Or W\* - wystarczy umieścić kurSOR w innej kolumnie, a wyrażenie zostanie uzupełnione automatycznie.



Rys. 7.5. Projekt kwerendy **k7\_1**

- f. Zapisz kwerendę, nadając jej nazwę **k7\_1**.

- g. Uruchom kwerendę za pomocą przycisku **Uruchom**, który jest umieszczony na wstążce w zakładce **Projektowanie**.



Rys. 7.6. Wybór polecenia **Widok projektu** kwerendy

Przeanalizuj wyniki działania kwerendy. Jeśli po uruchomieniu kwerendy postać wyświetlonych danych różni się od oczekiwanej, należy ponownie otworzyć kwerendę w widoku projektu (rys. 7.6) i wprowadzić niezbędne zmiany.

## Zadanie 7\_2

Utwórz **kwerendę wybierającą** do wyszukiwania wszystkich danych z tabeli **tOsoba** dotyczących kobiet, których wartości numerów albumu są z zakresu  $<80000 ; 90000>$ . Dane należy posortować rosnąco według numerów albumu. Zapisz kwerendę, nadając jej nazwę **k7\_2**.

### Wykonanie

- Otwórz okno projektu kwerendy (postępuj tak samo jak przy wykonaniu punktów a. i b. **Zadania 7\_1**).
- Umieść wszystkie pola tabeli **tOsoba** w siatce kwerendy. W tym celu kliknij dwa razy na nagłówku okienka tabeli – wszystkie pola będą zaznaczone. Teraz można je przeciągnąć wskaźnikiem myszki do wiersza **Pole**. Siatkę okna projektu kwerendy (pierwszych pięć pól) przedstawiono na rysunku 7.7.
- W wierszu **Kryteria** kolumny *Płeć* wprowadź: „Kobieta”.
- W wierszu **Kryteria** kolumny *Album* wprowadź:  
 $>=80000 \text{ AND } <=90000$
- W wierszu **Sortuj** kolumny *Album* ustaw: **Rosnąco**.
- Zapisz kwerendę, nadając jej nazwę **k7\_2**.
- Uruchom kwerendę i przeanalizuj wyniki.

Pole:	ID	Nazwisko	Imiona	Album	Płeć
Tabela:	tOsoba	tOsoba	tOsoba	tOsoba	tOsoba
Sortuj:	Rosnąco				
Pokaż:	<input checked="" type="checkbox"/>				
Kryteria:	>=80000 And <=90000 "Kobieta"				

Rys. 7.7. Projekt kwerendy **k7\_2**

## Zadanie 7\_3

Utwórz **kwerendę wybierającą** do wyświetlenia wszystkich danych z tabeli **tOsoba** o osobach, których adres e-mail kończy się na „@onet.pl” lub na „@gmail.com”. Dane należy posortować rosnąco według pól *Nazwisko*, *Imiona* oraz *Miejscowość*. Zapisz kwerendę, nadając jej nazwę **k7\_3**.

Budując tę kwerendę, możesz od razu umieścić wszystkie pola tabeli **tOsoba** w siatce projektu, wybierając w okienku pól tabeli gwiazdkę (\*). Następnie skonstruuj kryterium. Każde pole w siatce projektu powinno występować tylko raz, więc umieszczając w siatce kwerendy pola niezbędne do zdefiniowania kryteriów oraz sortowania, skasuj w wierszu **Pokaż** znaczenie tych pól . Widok projektu kwerendy prezentuje rysunek 7.8.

Pole:	tOsoba.*	Nazwisko	Imiona	Miejscowość	e-mail
Tabela:	tOsoba	tOsoba	tOsoba	tOsoba	tOsoba
Sortuj:		Rosnąco	Rosnąco	Rosnąco	
Pokaż:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Kryteria:					Like "*@onet.pl" Or Like "*@gmail.com"

Rys. 7.8. Projekt kwerendy **k7\_3**

## Zadanie 7\_4

Utwórz **kwerendę wybierającą** do wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie**, których wartość pola *Data\_zakwaterowania* jest z października 2018 roku, a pole *Data\_wykwaterowania* jest puste. Dane należy posortować rosnąco według pola *ID*. Zapisz kwerendę, nadając jej nazwę **k7\_4**.

Kwerendę budujemy tak samo jak przy wykonaniu poprzednich zadań.

Kryterium wyszukiwania dla pola *Data\_zakwaterowania*:  
 $>= \#2018-10-01\#$  AND  $<= \#2018-10-31\#$ .

Kryterium wyszukiwania dla pola *Data\_wykwaterowania*: Is Null.

**Microsoft Access**, jak każdy system baz danych, posiada funkcje wbudowane. **Zadanie 7\_4** można również wykonać, używając funkcji wbudowanych: **Year** oraz **Month**, które należą do kategorii **Data/Godzina**.

Funkcja **Month** zwraca wartość określającą liczbę całkowitą z zakresu od 1 do 12, reprezentującą miesiąc roku. Na przykład, w wyniku wykonania funkcji **Month(#2018-04-16#)** zostanie wyświetlony numer miesiąca – 4 (tzn. kwiecień).

Funkcja **Year** zwraca liczbę całkowitą reprezentującą rok. Przykładowo w wyniku działania funkcji **Year (#2018-04-16#)** zostanie wybrany rok 2018. Kryterium wyszukiwania w postaci **Year([Data\_zakwaterowania])** oznacza, że chcemy w kolumnie *Data\_zakwaterowania* wybrać z każdej daty tylko rok, a nie całą datę.

Kolejny przykład przedstawia krok po kroku konstruowanie złożonego kryterium kwerendy, zawierającego wymienione funkcje.

## Zadanie 7\_5

---

Utwórz **kwerendę wybierającą** w celu wyświetlenia danych z wszystkich pól tabeli **tZakwaterowanie** z tych rekordów, których wartość pola *Data\_zakwaterowania* jest z października 2016 roku. Dane wynikowe należy posortować rosnąco według pola *ID*.

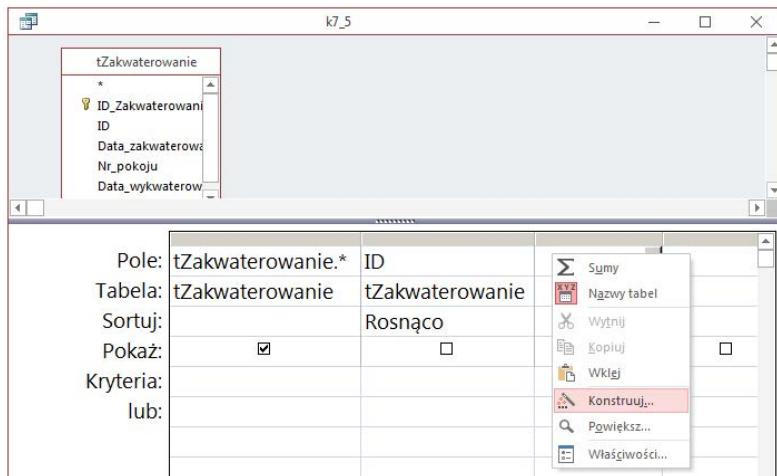
Do zdefiniowania kryteriów kwerendy zastosuj **Konstruktora wyrażeń**, a także funkcje wbudowane **Year** i **Month**.

Zapisz kwerendę, nadając jej nazwę **k7\_5**.

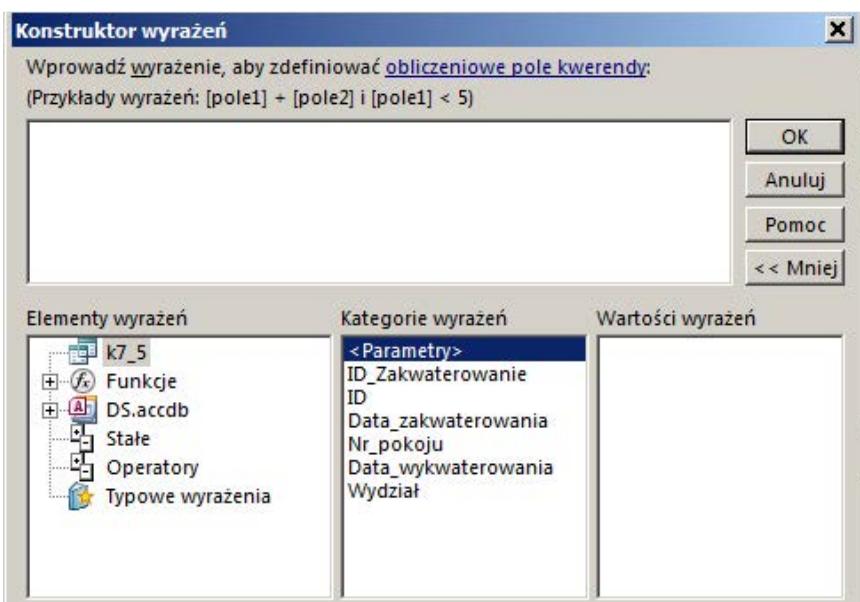
---

### Wykonanie

- a. Utwórz pustą kwerendę w widoku projektu.
- b. Umieść w siatce kwerendy wszystkie pola tabeli **tZakwaterowanie**.
- c. Dodaj do siatki pole *ID*, w wierszu **Sortuj** ustaw: **Rosnąco**, a w wierszu **Pokaż** skasuj zaznaczenie (rys. 7.9).
- d. Koniecznie zapisz kwerendę. Jest to niezbędne, jeśli mamy zamiar skorzystać z **Konstruktora wyrażeń**.
- e. Wywołaj **Konstruktora wyrażeń**. W tym celu w pustej kolumnie ustaw w wierszu **Pole** (tzn. w górnym wierszu siatki) wskaźnik myszy i naciśnij jej prawy klawisz. W wyświetlonym menu podręcznym kliknij polecenie **Konstruuj...** (rys. 7.9).
- f. Na ekranie pojawi się okno **Konstruktora wyrażeń** (rys. 7.10). W pierwszej kolumnie widzimy zaznaczoną nazwę kwerendy. Druga kolumna zawiera listę nazw jej pól (tylko gdy kwerenda jest zapisana).



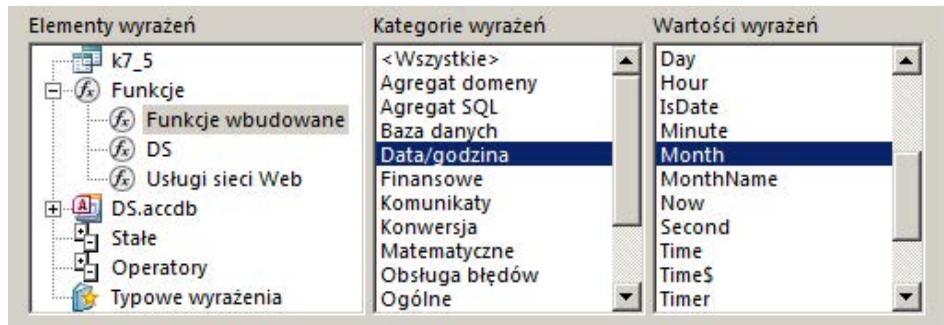
Rys. 7.9. Menu podręczne kwerendy **K7\_5** z zaznaczonym poleceniem do wywołania **Konstruktora wyrażeń**



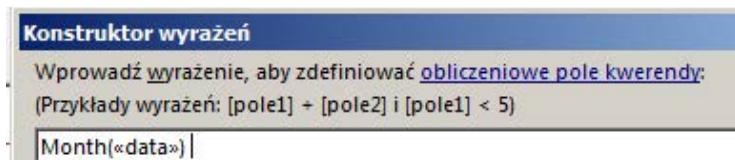
Rys. 7.10. Widok **Konstruktora wyrażeń** w oknie projektu kwerendy **k7\_5**

- g. W oknie **Konstruktora wyrażeń** kliknij lewym klawiszem myszki na znaku „+” obok **Funkcje**, wybierz **Funkcje wbudowane**, w drugiej kolumnie wybierz kategorię **Data/Godzina**, w trzeciej dwukrotnie kliknij **Month** (rys. 7.11). W górnej części okna **Konstruktora wyrażeń** poja-

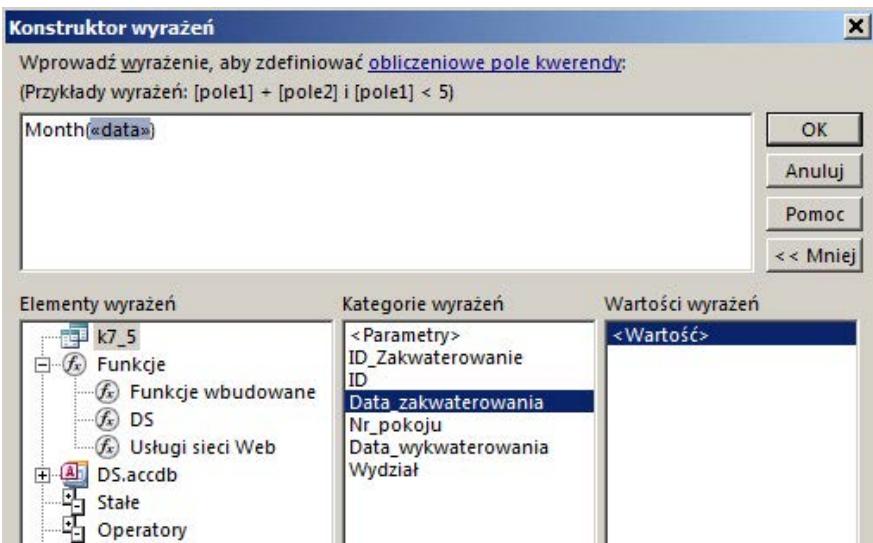
wi się nazwa funkcji **Month** oraz dołączony w nawiasach okrągłych szablon (rys. 7.12). Szablon ten podpowiada, że w miejscu <>data>> należy zapisać wartość typu **Data/Godzina**.



Rys. 7.11. Wybór funkcji wbudowanej **Month** w oknie **Konstruktora wyrażeń**

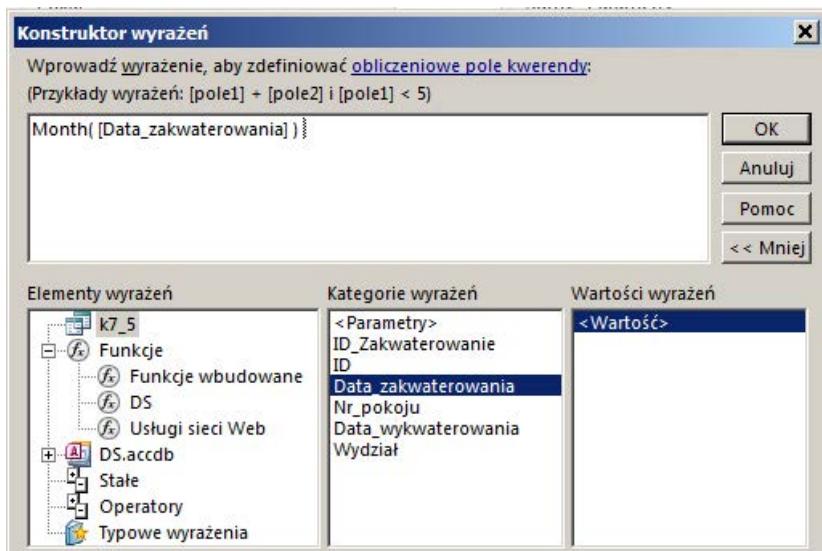


Rys. 7.12. Szablon funkcji **Month** w oknie **Konstruktora wyrażeń**



Rys. 7.13. Przygotowania do zastąpienia szablonu <>data<> nazwą pola *Data\_zakwaterowania*

- h. Kliknij lewym wskaźnikiem myszy kolejno:
- o szablon <>data>>,
  - o nazwę kwerendy **k7\_5** w pierwszej kolumnie,
  - o nazwę pola *Data\_zakwaterowania* (w drugiej kolumnie).
- W tym momencie okno **Konstruktora** będzie miało postać jak na rysunku 7.13.
- i. Kliknij dwukrotnie słowo **Wartość**, wówczas szablon zostanie zastąpiony nazwą pola *Data\_zakwaterowania* (rys. 7.14). Należy zauważyć, że wygenerowane przez program wyrażenie zawiera nawiasy okrągłe (bo jest to argument funkcji) oraz nawiasy kwadratowe dodawane zawsze do nazwy pola.



Rys. 7.14. Wyrażenie utworzone w oknie **Konstruktora wyrażeń**

- j. Aby zatwierdzić pracę w oknie **Konstruktora wyrażeń**, kliknij przycisk **OK** (rys. 7.14) – okno zostanie zamknięte, a w siatce kwerendy w wierszu **Pole** pojawi się zapis: *Wyr1: Month([Data\_zakwaterowania])*. Siatkę kwerendy przedstawiono na rys. 7.15.

Pole:	tZakwaterowanie.*	ID	Wyr1: Month([Data_zakwaterowania])
Tabela:	tZakwaterowanie	tZakwaterowanie	
Sortuj:		Rosnąco	
Pokaż:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Kryteria:			

Rys. 7.15. Siatka projektu kwerendy **k7\_5** po skonstruowaniu wyrażenia **Month([Data\_zakwaterowania])**

- k. W siatce kwerendy zamiast **Wyr1** wpisz słowo „Miesiąc”, a w wierszu **Kryteria** wprowadź wartość – 10 (jest to numer miesiąca - rys. 7.16).

Pole:	tZakwaterowanie.*	ID	Miesiąc: Month([Data_zakwaterowania])
Tabela:	tZakwaterowanie	tZakwaterowanie	
Sortuj:		Rosnąco	
Pokaż:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Kryteria:			10

Rys. 7.16. Siatka projektu kwerendy **k7\_5** po dodaniu kryterium dla pola Miesiąc

1. Ponownie zapisz kwerendę.
- m. Postępując analogicznie do zaleceń w punktach e - k, skorzystaj z funkcji wbudowanej **Year** i skonstruuj w siatce projektu odpowiednie wyrażenie oraz kryterium wyszukiwania rekordów z zadanego roku (rys. 7.17).

Miesiąc: Month([Data_zakwaterowania])	Rok: Year([Data_zakwaterowania])
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10	2016

Rys. 7.17. Zastosowanie funkcji **Month** i **Year** przy tworzeniu kwerendy **k7\_5**

- n. Zapisz kwerendę, a następnie ją uruchom i przeanalizuj wyniki.

Rozważmy teraz przykład zastosowania funkcji tekstowej **Mid**, która zwraca określoną liczbę znaków z ciągu. Funkcja ma postać:

**Mid(ciąg; początek [;długość ])**,

gdzie:

**ciąg** – jest wyrażeniem tekstowym, z którego wybieramy znaki;

**początek** – numer pozycji znaku w ciągu, od której zaczyna się część ciągu do pobrania;

**długość** – liczba znaków do zwrócenia (parametr ten może nie występować, wskazują na to nawiasy kwadratowe).

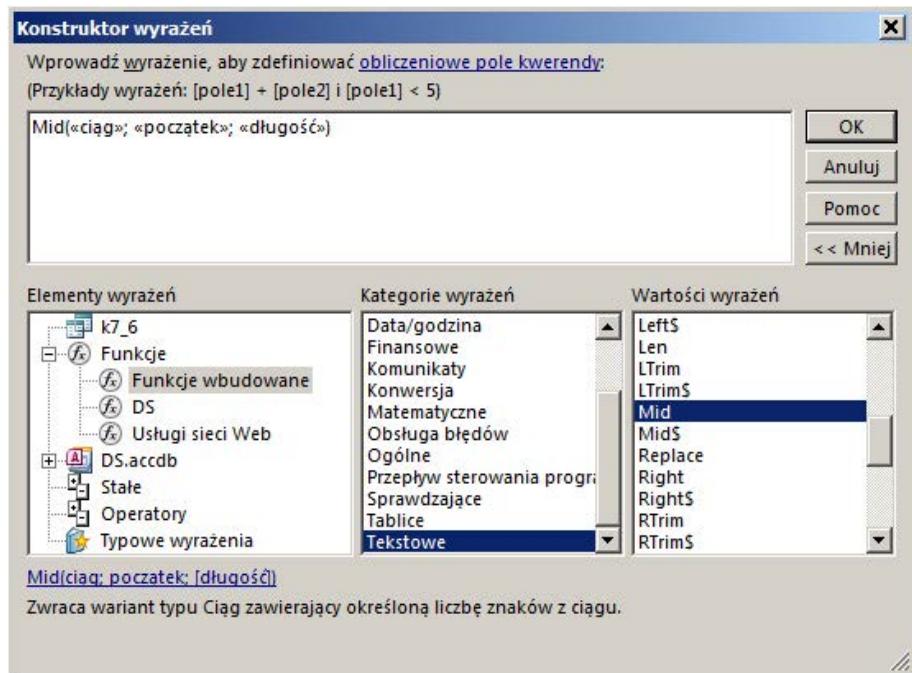
Wiadomo, że piąta oraz szósta pozycja numeru PESEL zawierają numer dnia miesiąca narodzin osoby. Wyodrębnimy go za pomocą funkcji **Mid**, wykonując następujące zadanie.

## Zadanie 7\_6

Utwórz kwerendę do wyświetlenia wartości pól: *ID*, *Nazwisko*, *Imiona*, *Album*, *Płeć*, *PESEL* z tabeli **tOsoba**. Wyświetlone dane powinny dotyczyć osoby urodzonej w 1 lub 3, lub 5 dniu miesiąca. Dane należy posortować według nazwiska i imienia. Zapisz kwerendę, nadając jej nazwę **k7\_6**.

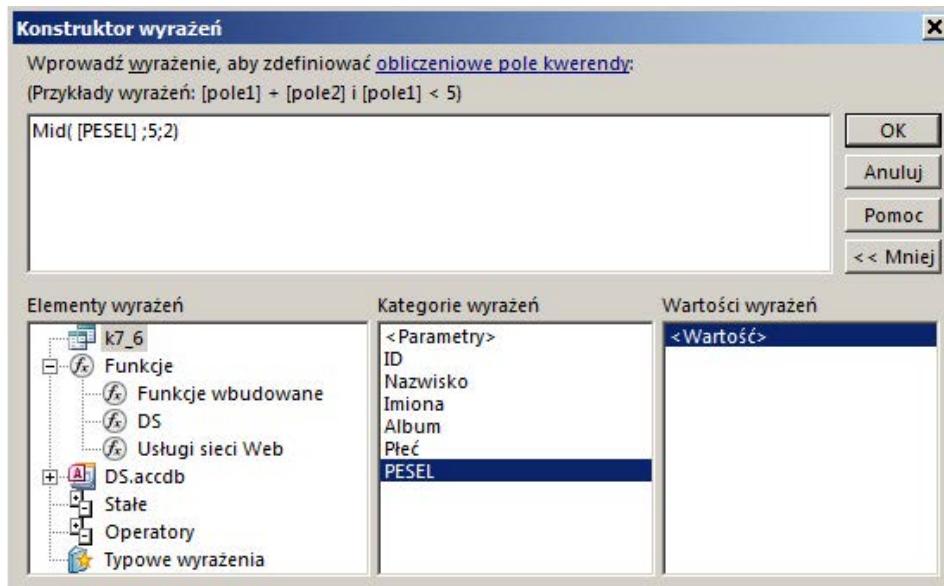
### Wykonanie

- Utwórz kwerendę wybierającą do wyświetlenia z tabeli **tOsoba** pól: *ID*, *Nazwisko*, *Imiona*, *Album*, *Płeć*, *PESEL*.
- Zapisz kwerendę.
- Wywołaj **Konstruktora wyrażeń** i w jego oknie wybierz funkcję **Mid**, jak przedstawiono na rysunku 7.18.



Rys. 7.18. Okienko **Konstruktora wyrażeń** przy tworzeniu kwerendy **k7\_6**

- Zastąp szablony `<>ciąg>`, `<>początek>` oraz `<>długość>` w sposób pokazany na rys. 7.19.
- Wybierz **OK**.
- W siatce kwerendy zamień **Wyr1** na **Dzień urodzenia** i dodaj kryterium (rys. 7.20).



Rys. 7.19. Wyrażenie do wyodrębnienia numeru miesiąca urodzenia z numeru PESEL

Pole:	ID	Nazwisko	Imiona	Album	Płeć	PESEL	Dzień urodzenia: Mid([PESEL];5;2)
Tabela:	tOsoba	tOsoba	tOsoba	tOsoba	tOsoba	tOsoba	
Sortuj:							
Pokaż:	<input checked="" type="checkbox"/>						
Kryteria:							"01" Or "03" Or "05"

Rys. 7.20. Siatka projektu kwerendy **k7\_6**

g. Zapisz i uruchom kwerendę.

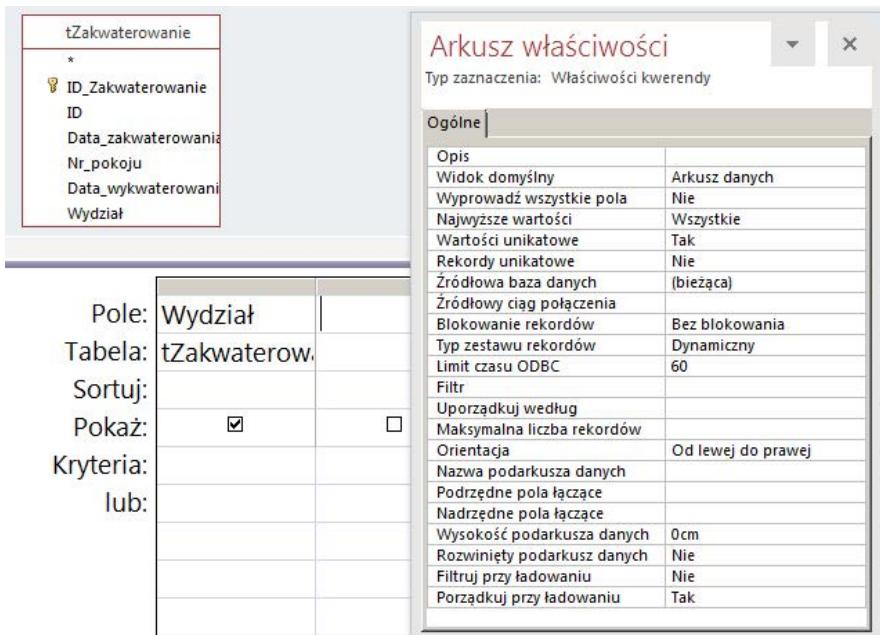
W niektórych zagadnieniach należy wyświetlić listę wszystkich wartości pola i przedstawić każdą wartość na liście tylko jeden raz. W kolejnym zadaniu zostanie rozważona taka sytuacja.

## Zadanie 7\_7

Utwórz kwerendę do przedstawienia wartości z pola *Wydział* tabeli **tZakwaterowanie**. Wymagane jest, aby każda nazwa wydziału występowała na liście tylko jeden raz. Zapisz kwerendę, nadając jej nazwę **k7\_7**.

## Wykonanie

- Utwórz kwerendę do wyświetlenia z tabeli **tZakwaterowanie** pola *Wydział*.
- Kliknij górny wiersz pustej kolumny siatki kwerendy, aby wywołać **Arkusz właściwości** kwerendy (rys. 7.20).
- W wierszu **Wartości unikatowe** wybierz **Tak**.



Rys. 7.21. Siatka projektu razem z arkuszem właściwości kwerendy **k7\_7**

W **Zadaniu 7.5** została użyta funkcja **Month**, która zwraca numer miesiąca. Wbudowana funkcja **MonthName** pozwala wyświetlić nazwę miesiąca. Kolejne zadanie przedstawia przykład zastosowania tej funkcji.

## Zadanie 7\_8

Utwórz kopię kwerendy **k7\_5**, nazwij nową kwerendę **k7\_8**. Dodaj do siatki kwerendy kolumnę w celu wyświetlenia nazwy miesiąca.

Wykonując dane zadanie, należy skorzystać z funkcji wbudowanej **MonthName**. Argumentem funkcji **MonthName** jest numer miesiąca.

Przykładowo funkcja **MonthName (2)** zwróci tekst: „luty” (tu argumentem funkcji **MonthName** jest liczba 2).

W tworzonej kwerendzie argumentem ma być numer miesiąca wybrany z daty zakwaterowania, więc argumentem funkcji **MonthName** będzie wynik funkcji **Month(Data\_zakwaterowania)**. Na rysunku 7.22 przedstawiono fragment siatki kwerendy **k7\_8** z funkcją **MonthName**.

Nazwa miesiąca: MonthName(Month([Data\_zakwaterowania]))

"październik"

Rys. 7.22. Zastosowanie funkcji **MonthName** w projekcie kwerendy **k7\_8**

### 7.3. Kwerenda wybierająca oparta na wielu tabelach

Kwerendy wybierającej najczęściej używamy w celu jednoczesnego wyświetlenia danych z wielu połączonych tabel. Taką kwerendę tworzymy na tych samych zasadach co i kwerendę opartą na jednej tabeli. Różnica polega na początkowym wskazaniu nazwy tabeli, na bazie których kwerenda ma być zbudowana.

#### Zadanie 7\_9

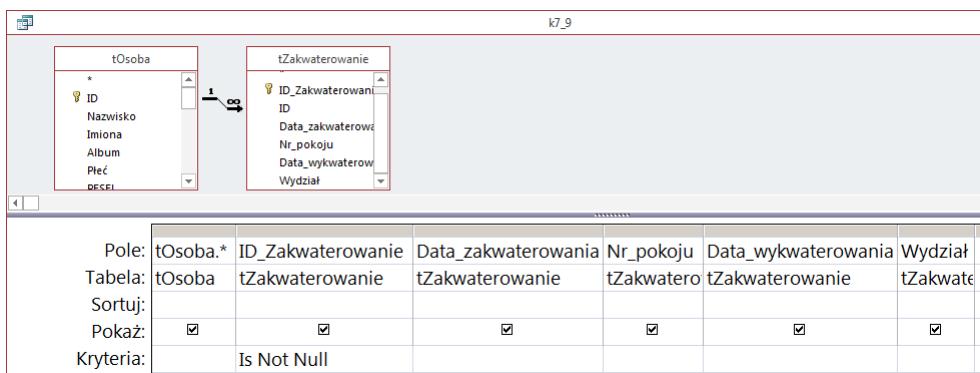
Zbuduj kwerendę **k7\_9** do wyświetlania danych z tabel **tOsoba** oraz **tZakwaterowanie**. Kwerenda powinna zwracać rekordy z danymi tylko tych osób, dla których zostały wpisane dane o zakwaterowaniu.

---

#### Wykonanie

- a. W oknie bazy danych **Dom studenta** wybierz na wstążce przycisk **Projekt kwerendy**.
- b. W wyświetlonym oknie dialogowym **Pokazywanie tabeli** wybierz (z przyciskiem *Ctrl*) po kolej nazwy **tOsoba** i **tZakwaterowanie**, a następnie **Dodaj i Zamknij** – zostanie wyświetlane okno projektu kwerendy.
- c. Umieść w siatce kwerendy wszystkie pola tabeli **tOsoba** (rys. 7.23).

- d. Umieść w siatce kwerendy wszystkie, oprócz pola *ID*, pola tabeli **tZakwaterowanie**.
- e. Dodaj kryterium dla pola *ID\_Zakwaterowanie* (rys. 7.23).
- f. Zapisz kwerendę, nadając jej nazwę **k7\_9**.



Rys. 7.23. Projekt kwerendy **k7\_9** opartej na dwóch tabelach

## 7.4 Używanie kwerendy jako źródła rekordów formularza

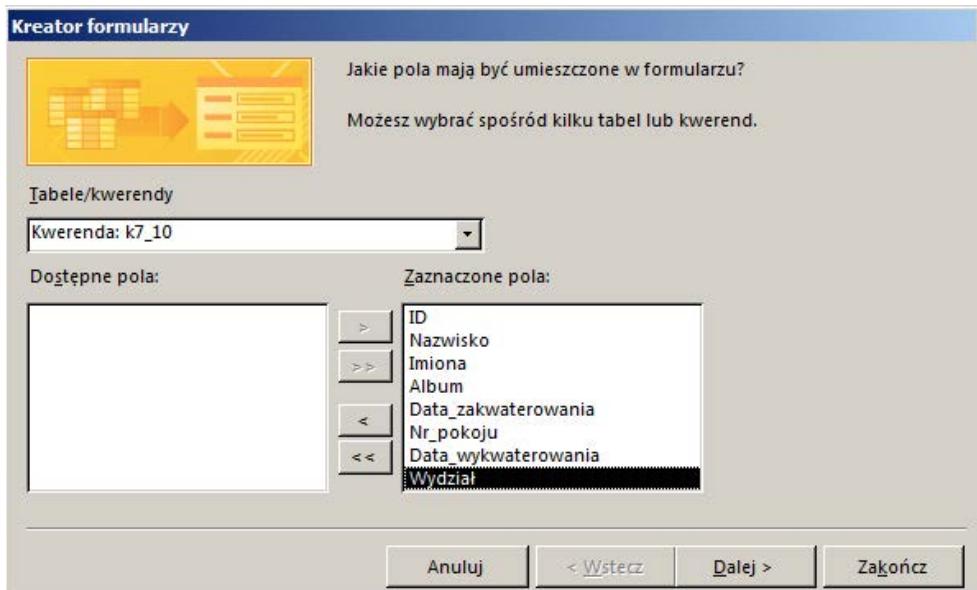
Rozważane wcześniej formularze zawsze były oparte na jednej tabeli. Jednak często do skonstruowania formularza nie wystarcza danych z jednej tabeli. W takim przypadku należy stworzyć odpowiednią kwerendę, a następnie użyć jej jako źródła danych.

W celu zapoznania się z tą możliwością należy samodzielnie wykonać kolejne zadanie, w wyniku czego powstanie formularz oparty na kwerendzie.

### Zadanie 7\_10

Zbuduj kwerendę **k7\_10** do wyświetlenia danych z tabel **tOsoba** oraz **tZakwaterowanie** w postaci: *ID*, *Nazwisko*, *Imiona*, *Album*, *Data\_zakwaterowania*, *Data\_wykwaterowania*, *Nr\_pokoju*, *Wydział*. Zbuduj formularz tabelaryczny o nazwie **fk7\_10**. Skorzystaj z kreatora formularzy. Źródłem rekordów formularza powinna być kwerenda **k7\_10**. Po utworzeniu formularza przejdź do widoku układu i popraw jego wygląd, aby wszystkie pola były dobrze widoczne.

Jako podpowiedź do wykonania formularza może posłużyć rysunek 7.24, na którym wyświetlono pierwsze okno kreatora.



Rys. 7.24. Pierwsze okno **Kreatora formularzy** podczas wykonania zadania 7\_10

## 7.5. Przykład zastosowania w kwerendzie funkcji IIF

Jedną z często używanych funkcji wbudowanych systemu **Access** jest funkcja **IIF**, która w oknie **Konstruktora wyrażeń** należy do kategorii **Przepływ sterowania programu**. Funkcję **IIF** stosujemy, jeżeli nie możemy jednoznacznie określić pewnej wartości, gdyż zależy ona od innej wartości (wyrażenia). W zależności od tego, czy wynikiem wyrażenia jest logiczna stała **PRAWDA** (True) czy **FALSZ** (False), dokonuje się różnych czynności.

Funkcję **IIF** można odnaleźć w oknie **Kreatora wyrażeń** wśród funkcji wbudowanych. Funkcję zapisuje się w postaci:

**IIF(<<wyrażenie>>; <<jeśli\_prawda>>; <<jeśli\_fałsz>>).**

W miejscu symbolicznie oznaczonym jako <<wyrażenie>> umieszczamy wyrażenie, którego wynikiem może być logiczne znaczenie **PRAWDA** lub **FALSZ**. Program sprawdza to wyrażenie i gdy wynik ma znaczenie **PRAWDA**, to wykonuje polecenie oznaczone jako <<jeśli\_prawda>>. W przeciwnym przypadku wykonywane jest drugie polecenie, oznaczone jako <<jeśli\_fałsz>>.

Następne **zadanie 7\_11** jest przykładem wyjaśniającym działanie tej funkcji.

## Zadanie 7\_11

Utwórz kwerendę **k7\_11** do wyświetlenia pól: *Nazwisko*, *Imiona*, *Album*, *PESEL* tabeli **tOsoba** oraz pól *Data\_zakwaterowania* i *Data\_wykwaterowania* tabeli **tZakwaterowanie**.

Należy dodać do kwerendy jeszcze jedno pole i umieścić w nim tekst „*Zamieszkuje*”, jeśli pole *Data\_wykwaterowania* jest puste.

Aby wykonać to zadanie, należy umieścić w projekcie kwerendy pola: *Nazwisko*, *Imiona*, *Album*, *PESEL*, *Data\_zakwaterowania* i *Data\_wykwaterowania*. Następnie należy zapisać kwerendę, a potem w oknie **Konstruktora wyrażeń** zbudować wyrażenie:

**IIF([Data\_wykwaterowania] Is Not Null;"";"Zamieszkuje")**

Tu rozważane jest wyrażenie: *[Data\_wykwaterowania] Is Not Null*. Jeśli wynikiem jest **PRAWDA**, wówczas wyświetlany jest pusty ciąg znaków, w przeciwnym przypadku zostanie wyświetlony tekst: „*Zamieszkuje*”.

## Zadania do samodzielnego wykonania

### Zadanie 7\_12

Zbuduj kwerendę **k7\_12** do wyświetlania danych o wszystkich osobach, które zamieszkali w Domu Studenta w bieżącym roku akademickim i jeszcze się nie wyprowadziły. Wyświetl pola: *ID*, *Nazwisko*, *Imiona*, *Album*, *Dzień urodzenia*, *Płeć*, *Data\_zakwaterowania*, *Data\_wykwaterowania*, *Nr\_pokoju*, *Wydział*. Przy czym pole *Dzień urodzenia* należy utworzyć, wykorzystując funkcję wbudowaną. Posortuj dane rosnaco według nazwiska, imienia, albumu. Zbuduj za pomocą kreatora formularz tabelaryczny o nazwie **fk7\_12** oparty na utworzonej kwerendzie. Po utworzeniu dostosuj w widoku układu formularza szerokości wszystkich powstałych pól.

### Zadanie 7\_13

Zbuduj kwerendę o nazwie **k7\_13** do wyświetlania wszystkich danych z tabeli **tOsoba** o studentach z województwa lubelskiego lub podlaskiego, których trzecią i czwartą pozycję pola *PESEL*

jest „03”, a numer albumu jest spoza zakresu <61000; 61100>. Dane należy posortować malejąco według numeru albumu.

---

### Zadanie 7\_14

---

Zbuduj kwerendę o nazwie **k7\_14** do wyświetlania wszystkich danych z tabeli **tOsoba** o mężczyznach, których pierwsza litera nazwiska jest „D”, drugą literą jest litera nieznana, natomiast pozostała część nazwiska to „browski”, a ich imię zaczyna się od liter: „L”, „M”, „N” lub „O”.

---

### Zadanie 7\_15

---

Zbuduj kwerendę o nazwie **k7\_15** do wyświetlania wszystkich danych o jedno- i dwuosobowych pokojach z 4, 5, 6, 7 piętra Domu Studenta. Dane należy posortować rosnąco według piętra i numeru pokoju.

---

### Zadanie 7\_16

---

Utwórz kwerendę wybierającą o nazwie **k7\_16** do wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie**, których wartość w polu *Data\_wykwaterowania* jest z 21, 23, 27 lub 30 dnia miesiąca, a w polu *Wydział* zawarty jest tekst: „Elektryczny”. Uporządkuj rekordy według dnia wykwaterowania oraz numeru *ID* malejąco.

---

*Podpowiedź do wykonania zadania 7\_16:*

Zastosuj funkcję wbudowaną **Day**, która zwraca numer dnia miesiąca od 1 do 31.

### Zadanie 7\_17

---

Utwórz kwerendę wybierającą o nazwie **k7\_17** do wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie**, których wartość pola *Data\_zakwaterowania* jest z lat 2014 lub 2016, a nazwa wydziału – „Mechaniczny”. Dane posortuj według daty zakwaterowania oraz numeru *ID* rosnąco.

---

## Zadanie 7\_18

Utwórz kwerendę wybierającą o nazwie **k7\_18** do wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie**, których dzień określony w polu *Data\_zakwaterowania* jest poniedziałkiem lub środą. Uporządkuj dane rosnąco według numeru dnia tygodnia oraz daty zakwaterowania.

*Podpowiedź do wykonania zadania 7\_16.*

Zastosuj funkcję **WeekDay**, która zwraca numer dnia tygodnia, zaczynając od 1 (niedziela) i kończąc na 7 (sobota).

## Zadanie 7\_19

Zbuduj kwerendę o nazwie **k7\_19** do wyświetlania danych w postaci: *ID, Nazwisko, Imiona, Album, PESEL, Płeć, Data\_zakwaterowania, Data\_wykwaterowania, Nr\_pokoju, Piętro, Wydział* o osobach, które zakwaterowały się w 2017 roku, a wykwaterowały w roku 2018 i zamieszkiwały na 2, 4 lub 6 piętrze. Dane należy posortować malejąco według *Data\_wykwaterowania* oraz rosnąco według albumu.

## Zadanie 7\_20

Zbuduj kwerendę o nazwie **k7\_20** do wyświetlenia wszystkich danych o osobach zameldowanych na stałe w Augustowie, które zamieszczały w 2018 roku w dwuosobowych pokojach i nie wykwaterowały się. Dane należy posortować rosnąco według daty zakwaterowania, numeru pokoju oraz nazwiska.

## Zadanie 7\_21

Zbuduj kwerendę o nazwie **k7\_21** do wyświetlania danych w postaci: *ID, Nazwisko, Imiona, Album, Płeć, e-mail, Data\_zakwaterowania, Data\_wykwaterowania, Nr\_pokoju, Wydział* o osobach urodzonych w roku 1995 lub 1997, których adres e-mail kończy się literami „@wp.pl”, a pole *Foto* nie jest puste. Dane należy posortować rosnąco według imienia, nazwiska oraz daty zakwaterowania.

## 8. Obliczenia za pomocą kwerend

Kwerenda wybierająca pozwala nie tylko na wyświetlanie danych umieszczonych w tabelach bazy danych, ale za jej pomocą również wykonać obliczenia na tych danych. Operacje matematyczne mogą być przeprowadzone zarówno na oddzielnych rekordach tabeli, jak i na grupach tych rekordów. W dalszej części rozdziału zostały przedstawione przykłady podobnych działań.

### 8.1. Pole obliczeniowe w kwerendzie

Postępując ściśle z zasadami relacyjnego modelu danych, nie zamieszczamy w tabeli pól, których wartości można obliczyć na podstawie innych pól tabeli. W celu wykonania obliczeń na danych tworzymy kwerendy zawierające tak zwane pola obliczeniowe. Są to wyrażenia zbudowane za pomocą operatorów i funkcji wbudowanych na wartościach pól tabeli. Obliczenia są wykonywane przy każdym uruchomieniu kwerendy, dzięki czemu wartości w polach obliczeniowych zmieniają się w zależności od zmian w polach tabeli. Kolejne zadanie posłuży jako przykład tworzenia pola obliczeniowego w kwerendzie.

#### Zadanie 8\_1

---

Utwórz kwerendę o nazwie **k8\_1** do wyświetlenia danych ze wszystkich pól tabeli **tKoszty\_zakwaterowania** oraz dwóch pól obliczeniowych:

- o pola *Opłata\_dodatkowa*, zawierającego wartość 5% wartości z pola *Kwota\_mies\_opłaty*;
  - o pola *Nowa\_kwota*, zawierającego sumę wartości z pól *Kwota\_mies\_opłaty* oraz *Opłata\_dodatkowa*.
- 

#### Wykonanie

- a. Utwórz projekt kwerendy wybierającej opartej na tabeli **tKoszty\_zakwaterowania**.

- Umieść w siatce projektu wszystkie pola tabeli.
- Zapisz kwerendę, nadając jej nazwę **k8\_1**.
- Kliknij prawym przyciskiem myszki w pustej komórce wiersza **Pole** i wybierz polecenie **Konstruuj**.
- W oknie **Konstruktora wyrażeń** wpisz wyrażenie: *Kwota\_mies\_opłaty \* 0,05* (rys. 8.1). Wybierz **OK**.
- W oknie projektu kwerendy zamień nazwę wyrażenia **Wyr1** na **Opłata\_dodatkowa** (rys. 8.2).
- Ponownie zapisz kwerendę.
- Kolejny raz skorzystaj z **Konstruktora wyrażeń** i zbuduj jeszcze jedno pole obliczeniowe (rys. 8.3).



Rys. 8.1. Tworzenie pola obliczeniowego w **Konstruktore wyrażeń**

Pole:	tKoszty_zakwaterowania.*	Opłata_dodatkowa: [Kwota_mies_opłaty]*0,05
Tabela:	tKoszty_zakwaterowania	
Sortuj:		
Pokaż:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kryteria:		

Rys. 8.2. Siatka projektu kwerendy **k8\_1** po utworzeniu pierwszego pola obliczeniowego

- i. W oknie projektu kwerendy zamień nazwę **Wyr1** na **Nowa\_kwota**. W wyniku tych działań pola obliczeniowe w siatce projektu kwerendy będą miały postać jak na rysunku 8.4.
- j. Uruchom kwerendę.

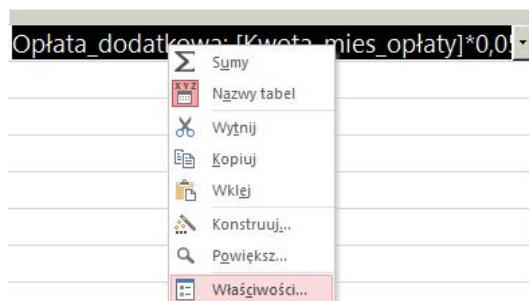


Rys. 8.3. Drugie pole obliczeniowe kwerendy **k8\_1** w oknie **Konstruktora wyrażeń**

Opłata_dodatkowa: [Kwota_mies_opłaty]*0,05	Nowa_kwota: [Kwota_mies_opłaty]+[Opłata_dodatkowa]
--	--

Rys. 8.4. Utworzono pola obliczeniowe w kwerendzie **k8\_1**

Jeżeli obie kolumny, bądź jedna z nich, nie zawierają nazwy waluty (zł), w oknie projektu kliknij na tej kolumnie prawym przyciskiem myszki, wybierz **Właściwości** (rys. 8.5) i zdefiniuj format: **Walutowy**.



Rys. 8.5. Wybór właściwości pola **Opłata\_dodatkowa**

## 8.2. Wiersz podsumowujący w arkuszu danych kwerendy wybierającej

Gdy zachodzi konieczność podsumowania danych w kolumnie tabeli, można wówczas stworzyć na podstawie tej tabeli kwerendę wybierającą i dodać w jej arkuszu danych **wiersz podsumowujący**, gdzie oprócz sumy można zastosować także dowolną funkcję agregującą.

Rozważmy tę możliwość na przykładzie.

### Zadanie 8\_2

Utwórz kwerendę wybierającą **k8\_2** z wierszem podsumowującym w celu wyświetlenia danych ze wszystkich pól tabeli **tPokoje**. W wierszu podsumowującym wyświetl liczbę miejsc do zamieszkania na drugim i trzecim piętrze Domu Studenta.

#### Wykonanie

- Utwórz kwerendę wybierającą do wyświetlenia wszystkich danych z tabeli **tPokoje** o pokojach na 2 i 3 piętrze.
- Zapisz kwerendę, nadając jej nazwę **k8\_2**.
- Uruchom kwerendę – wówczas zostanie wyświetlony arkusz danych.
- Na zakładce **Narzędzia główne** w grupie **Rekordy** kliknij przycisk **Sumy** (rys. 8.6). Poniżej wiersza oznaczonego gwiazdką pojawi się nowy wiersz **Suma** (rys. 8.7).

Nr Pokoju	Pietro	Liczba miejsc	Uwagi
202	2	3	
310	3	3	
210	2	1	

Rys. 8.6. Wybór przycisku **Sumy** z karty **Narzędzia główne** w oknie arkusza kwerendy wybierającej

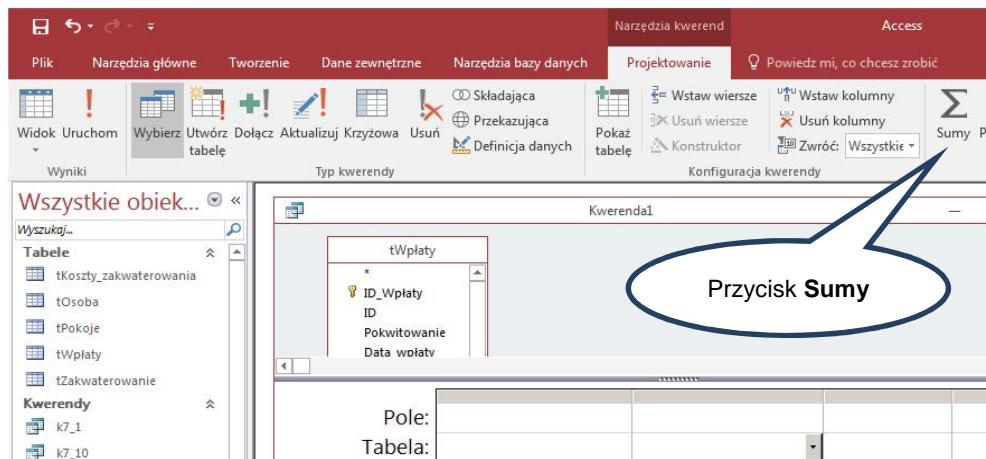
- e. W wierszu **Suma** zaznacz kolumnę **Liczba miejsc**, a następnie z listy wybierz pozycję **Suma**, tak jak pokazano na rysunku 8.7. Wówczas liczby w kolumnie zostaną zsumowane.
- f. Przeanalizuj wynik działania kwerendy i zapisz ją ponownie.

The screenshot shows a Microsoft Access query design grid. The columns are labeled 'Nr Pokoju', 'Pietro', 'Liczba miejsc', and 'Uwagi'. The 'Liczba miejsc' column is highlighted with a yellow background. A context menu is open over this column, with the 'Suma' option selected. Other options in the menu include 'Brak', 'Średnia', 'Liczba', 'Maksimum', 'Minimum', 'Odchylenie standardowe', and 'Wariancja'.

Rys. 8.7. Dodawanie sumowania liczby miejsc w arkuszu danych kwerendy

### 8.3. Funkcje agregujące w projekcie kwerendy wybierającej

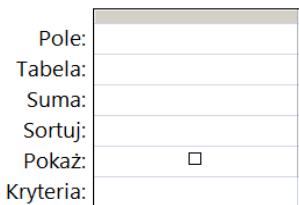
W relacyjnych bazach danych istnieje możliwość wykonywania operacji na całej tabeli lub jej części za pomocą tak zwanych funkcji agregujących. Możemy z nich skorzystać w oknie projektu kwerendy wybierającej.



Rys. 8.8. Przycisk **Sumy** ulokowany na karcie **Projektowanie** w oknie projektu kwerendy wybierającej

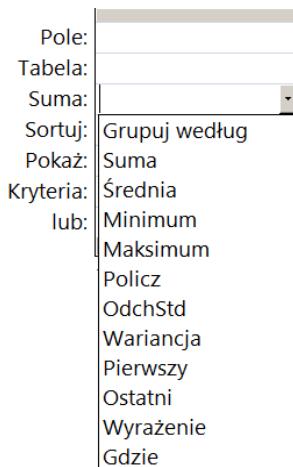
Na rysunku 8.8 pokazano okno pustego projektu kwerendy. W tym oknie na karcie **Projektowanie**, tak samo jak w arkuszu danych, znajduje się ikona **Sumy**, która jednak – choć nosi tę samą nazwę – powoduje inne działania, niż zostało to przedstawione w poprzednim przykładzie.

Po kliknięciu ikonki **Sumy** siatka projektu będzie uzupełniona o dodatkowy wiersz o nazwie *Suma* (rys. 8.9).



Rys. 8.9. Wiersze siatki kwerendy wybierającej po dodaniu wiersza **Suma**

W wierszu **Suma** można otworzyć listę rozwijaną (rys. 8.10) i wybrać jedną z funkcji agregujących: **Suma**, **Średnia**, **Minimum**, **Maksimum**, **Policz**, **Pierwszy**, **Ostatni**, **Wariancja**, **OdchStd**.



Rys. 8.10. Lista rozwijana w wierszu **Suma** w projekcie kwerendy wybierającej

Kwerenda, której projekt zawiera wiersz **Suma**, często jest nazywana kwerendą podsumowującą. Opcje **Grupuj według**, **Wyrażenie**, **Gdzie** z listy rozwijanej wykorzystywane są przy jej budowie.

Przykłady kwerend zawierających funkcje agregujące podzielone zostały w dalszej części na dwie grupy ze względu na sposób wykorzystania funkcji agregujących w oknie projektu kwerendy: oddzielnie rozważano

obliczenia na jednej oraz na więcej niż jednej grupie rekordów. W tym drugim przypadku kwerenda grupuje dane i wyświetla wynik dla każdej grupy.

### *Obliczenia na jednej grupie rekordów*

Rozważmy przykłady kwerend stworzonych w celu wykonania obliczeń na wszystkich rekordach tabeli za pomocą funkcji agregujących bez grupowania. Kwerendy te zawsze zawierają co najmniej jedną kolumnę z funkcją agregującą, a także mogą mieć dowolną liczbę kolumn definiujących kryteria działania tych funkcji. Żadna kolumna projektu takiej kwerendy nie zawiera opcji **Grupuj według**.

## **Zadanie 8\_3**

---

Utwórz kwerendę o nazwie **k8\_3** do obliczenia sumy wartości wpłat dokonanych za rok akademicki 2016/2017.

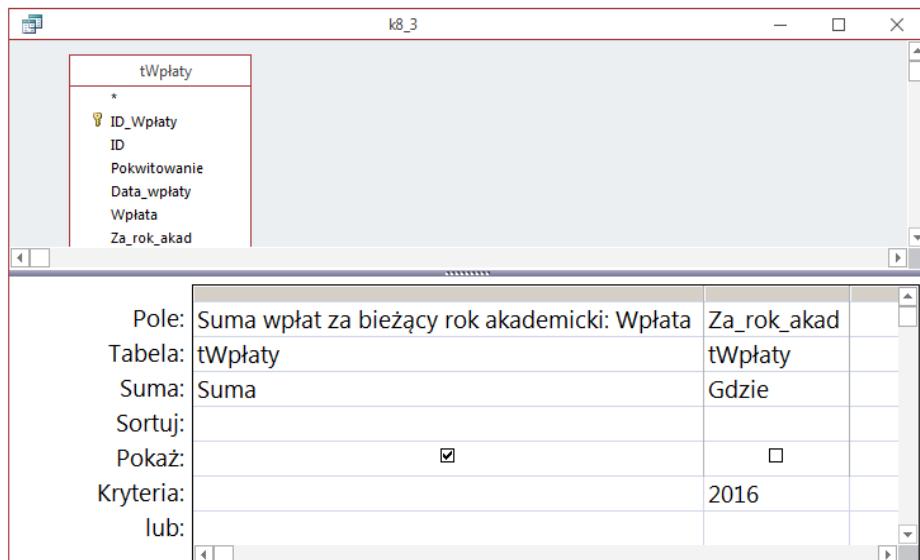
---

### **Wykonanie**

- a. Utwórz pusty projekt kwerendy wybierającej na podstawie tabeli **tWpłaty**.
- b. Zapisz kwerendę, nadając jej nazwę **k8\_3**.
- c. W oknie projektu na zakładce **Projektowanie** kliknij przycisk **Sumy** (rys. 8.8).
- d. W siatce projektu kwerendy wypełnij pierwszą kolumnę (rys. 8.11):
  - umieść w siatce kwerendy pole **Wpłata**,
  - w wierszu **Pole** wpisz przed nazwą **Wpłata** tekst:

#### **Suma wpłat za bieżący rok akademicki:**

- w wierszu **Suma** wybierz z listy rozwijanej nazwę funkcji **Suma**,
- w wierszu **Pokaż** pozostaw zaznaczenie, gdyż pole jest pokazywane.
- e. W siatce projektu kwerendy wypełnij drugą kolumnę (rys. 8.11):
  - umieść w siatce kwerendy pole **Za\_rok\_akad**,
  - w wierszu **Suma** wybierz z listy rozwijanej opcję **Gdzie**,
  - w wierszu **Kryteria** umieść: 2016 (co ma oznaczać rok akademicki 2016/2017),
  - w wierszu **Pokaż** zaznaczenie należy skasować (w takich kwerendach nigdy nie pokazujemy pola kryteriów).
- f. Zapisz i uruchom kwerendę – prawidłowy wynik zawiera tylko jedną liczbę. Przykładową postać wyniku kwerendy przedstawia rysunek 8.12.



Rys. 8.11. Zastosowanie funkcji agregującej **Suma** w kwerendzie **k8\_3**

k8_3	
Suma wpłat za bieżący rok akademicki	▼
1 170,00 zł	

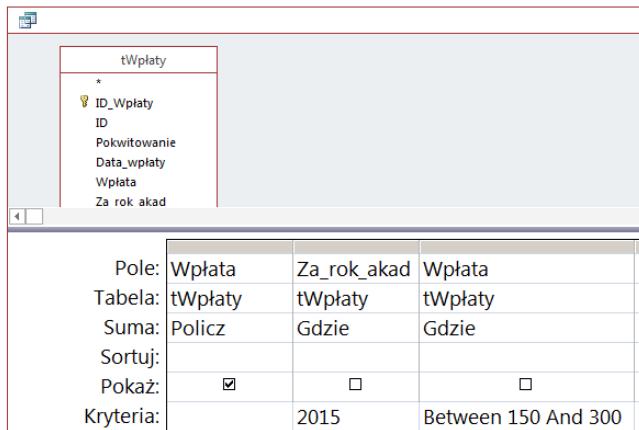
Rys. 8.12. Przykładowy wynik wykonania kwerendy **k8\_3**

## Zadanie 8\_4

Utwórz kwerendę **k8\_4** do wyświetlenia liczby wpłat z tabeli **tWpłaty** dotyczących roku akademickiego 2015/2016. Kwota wpłaty powinna mieścić się w zakresie <150, 300>.

### Wykonanie

Daną kwerendę wykonuje się tak samo jak poprzednią. Różni się ona tylko zastosowaną funkcją. W tym przypadku należy wykorzystać funkcję **Policz** (rys. 8.13).



Rys. 8.13. Przykład zastosowania funkcji **Policz**

## Zadanie 8\_5

Utwórz kwerendę **k8\_5** do wyświetlenia liczby rekordów tabeli **tZakwaterowanie**, których pole *Data\_wykwaterowania* jest puste, a wartość tekstowa w polu *Wydział* nie jest ani „Elektryczny”, ani „Budownictwo”.

## Wykonanie

Przy wyliczeniu liczby rekordów najlepiej jest ulokować funkcję **Policz** w polu klucza podstawowego tabeli. Siatka projektu kwerendy jest przedstawiona na rysunku 8.14.

Pole:	ID_Zakwaterowanie	Data_wykwaterowania	Wydział
Tabela:	tZakwaterowanie	tZakwaterowanie	tZakwaterowanie
Suma:	Policz	Gdzie	Gdzie
Sortuj:			
Pokaż:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Kryteria:		Is Null	Not In ("Budownictwo";"Elektryczny")

Rys. 8.14. Przykład wyliczenia liczby wierszy tabeli spełniających zadane kryterium (kwerenda **k8\_5**)

## Zadanie 8\_6

Utwórz kwerendę o nazwie **k8\_6** do wyświetlenia średniej wartości kwot miesięcznych opłat z tabeli **tKoszty\_zakwaterowania**, przyznawanych studentom od października 2016 roku.

## Wykonanie

Wypełnioną siatkę projektu kwerendy przedstawia rysunek 8.15.

Pole:	Kwota_mies_opłaty	Wyr1: Month([tKoszty_zakwaterowania]![Od_kiedy])	Wyr2: Year([tKoszty_zakwaterowania]![Od_kiedy])
Tabela:	tKoszty_zakwaterowania	Gdzie	Gdzie
Suma:	Średnia		
Sortuj:			
Pokaz:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Kryteria:	10		2016

Rys. 8.15. Okno projektu kwerendy **k8\_6**

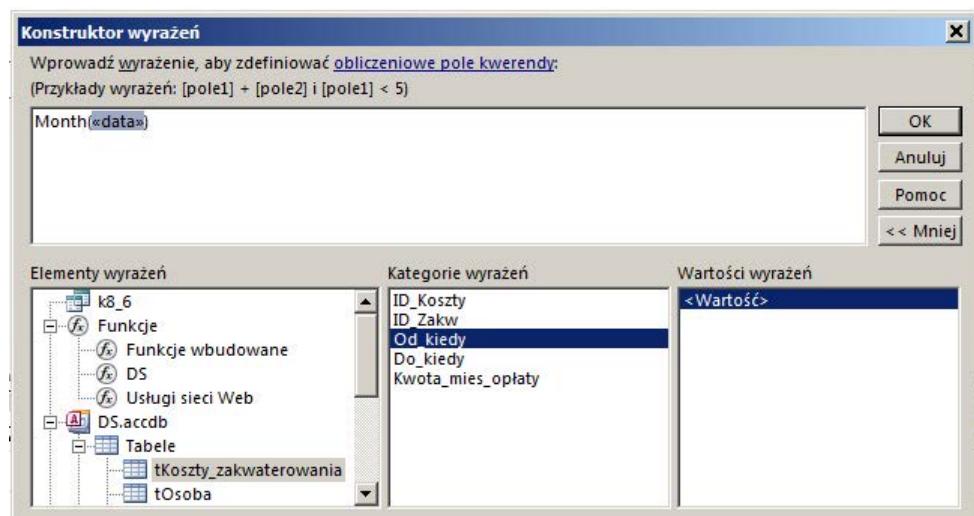
Przy tworzeniu warunku wyszukiwania danych zastosowano funkcje wbudowane **Month** oraz **Year**. W siatce projektu widzimy dwa wyrażenia, które zostały zbudowane za pomocą konstruktora wyrażeń:

**Month([tKoszty\_zakwaterowania]![Od\_kiedy])**

oraz

**Year([tKoszty\_zakwaterowania]![Od\_kiedy]).**

Znak „!” oddziela nazwę tabeli od nazwy pola. Na rysunku 8.16 przedstawiono sposób wybierania nazwy pola tabeli w oknie konstruktora wyrażeń. Na tym rysunku zostały przedstawione działania w bazie danych o nazwie DS.



Rys. 8.16. Budowa wyrażenia **Month([tKoszty\_zakwaterowania]![Od\_kiedy])**

## Obliczenia na więcej niż jednej grupie rekordów

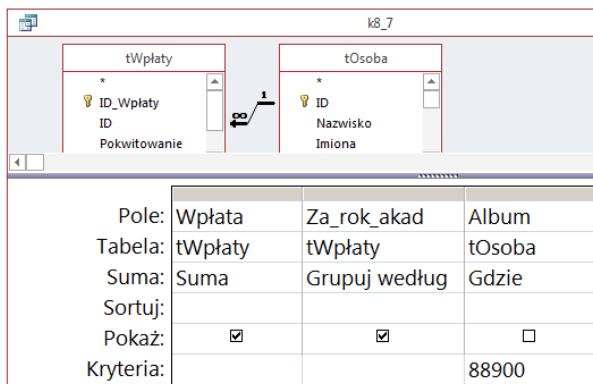
Funkcję agregującą można zastosować do więcej niż jednej grupy rekordów. W takim przypadku kwerenda zawiera dodatkową (lub dodatkowe) kolumnę (kolumny), której wartością w wierszu **Podsumowanie** jest opcja **Grupuj według**. W odróżnieniu od kolumny zawierającej opcję **Gdzie** kolumna zawierająca **Grupuj według** jest pokazywana w wyniku wykonania kwerendy.

### Zadanie 8\_7

Utwórz kwerendę o nazwie **k8\_7** do wyliczenia sumy wartości wpłat za każdy rok akademicki dokonanych przez studenta, którego numer albumu jest równy 88900. Dane wynikowe mają być posortowane rosnaco według roku akademickiego.

### Wykonanie

Projekt kwerendy **k8\_7** zaprezentowano na rysunku 8.17. Po uruchomieniu kwerendy będą wyświetlane dwie kolumny wynikowe. Przykładowy wynik działania kwerendy przedstawiono na rysunku 8.18.



Rys. 8.17. Projekt kwerendy **k8\_7**, w którym dokonano agregowania oraz grupowania

SumaOfWpłata	Za rok akademicki
280,00 zł	2014
930,00 zł	2015

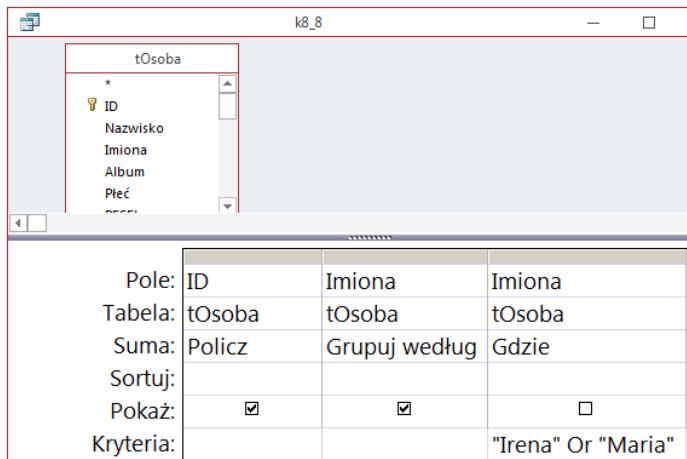
Rys. 8.18. Przykładowy wynik wykonania kwerendy **k8\_7**

## Zadanie 8\_8

Utwórz kwerendę o nazwie **k8\_8** do ustalenia liczby zapisanych do tabeli **tOsoba** osób noszących imię Maria oraz liczby osób o imieniu Irena.

### Wykonanie

Siatkę projektu kwerendy **k8\_8** przedstawiono na rysunku 8.19. Po uruchomieniu kwerendy zobaczymy dwa wiersze wynikowe (zakładając, że w tabeli **tOsoba** wymienione imiona zostały wpisane przynajmniej jeden raz). W tym zadaniu nie wymaga się uporządkowania wyników alfabetycznie, jednak często taka potrzeba zachodzi. Należy wówczas w wierszu **Sortuj** wybrać odpowiednią opcję: „Rosnąco” bądź „Malejąco”. W kolejnym przykładzie właśnie ta opcja zostanie wykorzystana.



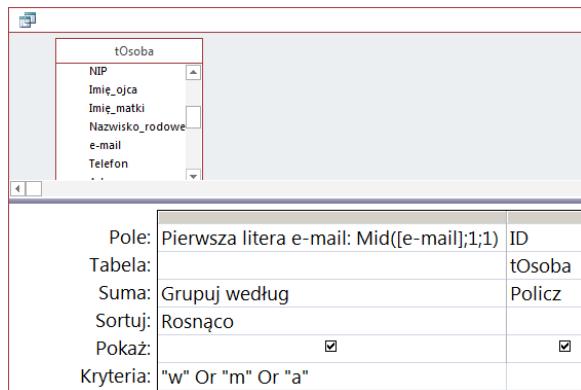
Rys. 8.19. Siatka projektu kwerendy **k8\_8**

## Zadanie 8\_9

Utwórz kwerendę o nazwie **k8\_9** do wyświetlenia liczby osób, których e-mail zaczyna się od litery „w”, „m” lub „a”. Należy podać wynik dla każdej litery oddzielnie oraz uporządkować wynikowe dane alfabetycznie.

### Wykonanie

Jeśli skorzystamy z funkcji **MID**, to siatka projektu kwerendy **k8\_9** będzie miała postać przedstawioną na rysunku 8.20.



Rys. 8.20. Siatka projektu kwerendy **k8\_9**

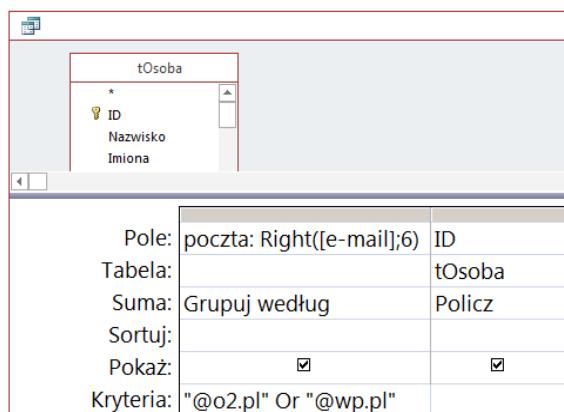
W kolejnym zadaniu rozważymy między innymi zastosowanie funkcji wbudowanej **RIGHT**, która tak samo jak i funkcja **MID** wybiera w tekście zadaną liczbę znaków, przy czym wybiera je od końca ciągu.

## Zadanie 8\_10

Utwórz kwerendę o nazwie **k8\_10** w celu wyświetlenia liczby osób, których e-mail ma końcówkę „@o2.pl”, oraz liczby osób, których e-mail kończy się na „@wp.pl”.

### Wykonanie

Sposób wykonania pokazano na rysunku 8. 21.



Rys. 8. 21. Siatka projektu kwerendy **k8\_10**

## 8.4. Kwerenda krzyżowa

Zastosowanie kwerendy krzyżowej jest bardzo użyteczne wówczas, gdy konieczne jest grupowanie treści tabeli w taki sposób, aby jej kolumny zostały wierszami, a wiersze kolumnami. Oprócz grupowania wykonywane są również funkcje agregowania. Rozważmy ten rodzaj kwerendy na przykładzie.

### Zadanie 8\_11

Utwórz kwerendę krzyżową o nazwie **k8\_11** na bazie tabeli **tWpłaty** do przedstawienia sum wpłat trzech studentów za każdy rok akademicki. Numery *ID* tych studentów wynoszą: 1, 3 oraz 5.

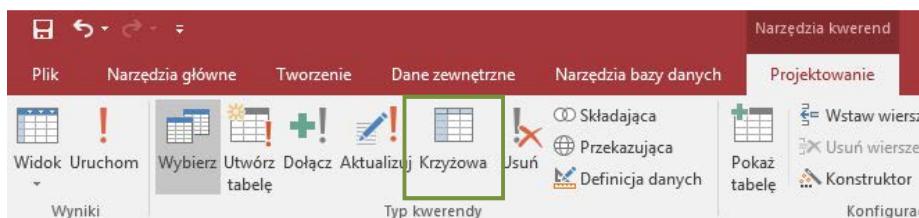
### Wykonanie

Przed wykonaniem kwerendy należy przygotować dane testowe. Dla każdej osoby z tabeli **tOsoba** powinny być co najmniej dwa wpisy z różnych lat akademickich w tabeli **tZakwaterowanie**, odpowiednie rekordy w tabeli **tKoszty\_zakwaterowania**, natomiast w tabeli **tWpłaty** powinny się znaleźć dane na temat wpłat tej osoby w wybranych latach akademickich.

Po przygotowaniu danych testowych należy się zastanowić, w jaki sposób chcemy wyświetlać dane. Zgodnie z treścią zadania będziemy dążyć do tego, aby wyświetlane dane miały postać tabeli, której nagłówki kolumn to lata, nagłówki wierszy – numery *ID*, a każda komórka będzie zawierała sumę wpłat studenta za rok akademicki.

Postępuj zgodnie z podanym niżej algorytmem.

- Utwórz w widoku projektu kwerendę wybierającą na bazie **tWpłaty**.
- W siatce projektu umieść pola: *ID*, *Za\_rok\_akad*, *Wpłata*.
- W oknie kwerendy wybierz na wstążce zakładkę **Projektowanie**, a następnie przycisk **Krzyżowa** (rys. 8.22) – w siatce kwerendy pojawią się dwa nowe wiersze: **Suma** oraz **Krzyżowe** (rys. 8.23).



Rys. 8.22. Karta **Projektowanie** z zaznaczoną ikonką do tworzenia kwerendy krzyżowej

d. W kolumnie *ID* (rys. 8.23):

- o w wierszu **Krzyżowe** wybierz z listy opcje **Nagłówek wiersza**,
- o w wierszu **Sortuj** wybierz z listy opcję **Rosnąco**,
- o w wierszu **Kryteria** wprowadź: 1 Or 3 Or 5.

Pole:	<b>ID</b>	<b>Za_rok_akad</b>	<b>Wpłata</b>
Tabela:	tWpłaty	tWpłaty	tWpłaty
Suma:	Grupuj według	Grupuj według	Suma
Krzyżowe:	Nagłówek wiersza	Nagłówek kolumny	Wartość
Sortuj:	Rosnąco	Rosnąco	
Kryteria:	1 Or 3 Or 5		

Rys. 8.23. Siatka projektu kwerendy **k8\_11**

e. W kolumnie *Za\_rok\_akad* (rys. 8.23):

- o w wierszu **Krzyżowe** wybierz opcję **Nagłówek kolumny**,
- o w wierszu **Sortuj** ustaw **Rosnąco**.

f. W kolumnie *Wpłata* (rys. 8.23):

- o w wierszu **Krzyżowe** wybierz opcję **Wartość**,
- o w wierszu **Podsumowanie** wybierz opcję **Suma**.

g. Zapisz kwerendę, nadając jej nazwę **k8\_11**.

Przykład wyniku pracy kwerendy przedstawia rysunek 8.24.

ID	2014	2015	2016
1	560,00 zł	300,00 zł	300,00 zł
3	280,00 zł	930,00 zł	
5			300,00 zł

Rys. 8.24. Widok wyniku pracy kwerendy **k8\_11** na przykładowych danych

Kwerenda krzyżowa pozwala na podsumowanie wynikowych danych w kolumnach. Założymy, że jednocześnie z już uzyskanymi danymi chcemy wyświetlić ogólne sumy wpłat każdego studenta. Wtedy należy dodać do siatki projektu kwerendy jeszcze jedną kolumnę (rys. 8.25).

Pole:	<b>ID</b>	<b>Za_rok_akad</b>	<b>Wpłata</b>	<b>Ogólna suma: Wpłata</b>
Tabela:	tWpłaty	tWpłaty	tWpłaty	tWpłaty
Suma:	Grupuj według	Grupuj według	Suma	Suma
Krzyżowe:	Nagłówek wiersza	Nagłówek kolumny	Wartość	Nagłówek wiersza
Sortuj:	Rosnąco	Rosnąco		
Kryteria:	1 Or 3 Or 5			

Rys. 8.25. Siatka projektu kwerendy krzyżowej

## Zadania do samodzielnego wykonania

### Zadanie 8\_12

Utwórz kwerendę do wyświetlania danych w postaci: *Nazwisko, Imiona, Album, Pokwitowanie, Data\_wpłaty, Wpłata, Na\_remont*, gdzie:

- *Nazwisko, Imiona, Album* – pola tabeli **tOsoba**;
- *Pokwitowanie, Data\_wpłaty, Wpłata* – pola tabeli **tWpłaty**;
- *Na\_remont* – pole obliczeniowe, które stanowi 10% kwoty zawartej w polu *Wpłata*.

Kwerenda powinna wyświetlać dane o wpłatach za rok akademicki 2017/2018, dokonanych w listopadzie przez osoby, których numery albumu są większe od 60000. Dane należy posortować rosnaco według nazwiska, imienia oraz numeru albumu.

### Zadanie 8\_13

Utwórz kwerendę do wyświetlenia z tabeli **tOsoba** wszystkich danych o mężczyznach. W arkuszu danych kwerendy należy również wyświetlić liczbę wierszy.

### Zadanie 8\_14

Utwórz kwerendę do wybierania danych o wpłatach za Dom Studenta dokonanych przez studentów, których nazwisko zaczyna się od litery „A” lub „C”. Należy również wyświetlić w arkuszu danych kwerendy wiersz podsumowujący, a w nim datę najwcześniejszej wpłaty.

### Zadanie 8\_15

Na podstawie tabeli **tOsoba** policz liczbę osób urodzonych w 1995 roku. Zastosuj funkcję agregującą.

### Zadanie 8\_16

Policz liczbę wierszy w tabeli **tZakwaterowanie** dotyczących zamieszkania osób, których nazwisko zaczyna się na literę „D” lub „K”, a imię kończy się na literę „ł”. Zastosuj funkcję agregującą.

### Zadanie 8\_17

---

Utwórz kwerendę do wyświetlenia minimalnej kwoty miesięcznej opłaty przyznawanej osobom płci męskiej urodzonym w maju. Zastosuj funkcję agregującą.

---

### Zadanie 8\_18

---

Utwórz kwerendę do wyświetlania najwcześniejszych dat wpłat za rok akademicki 2017/2018 wszystkich studentów skierowanych do Domu Studenta przez Wydział Mechaniczny. Wynik kwerendy powinien zawierać dwie kolumny, z których pierwsza to album, a druga – data wpłaty. Zastosuj funkcję agregującą.

---

### Zadanie 8\_19

---

Utwórz kwerendę do wyświetlenia, w ilu wierszach tabeli **tOsoba** zawarte są dane o kobietach, a w ilu – o mężczyznach. Zastosuj funkcję agregującą.

---

### Zadanie 8\_20

---

Utwórz kwerendę do wyliczenia liczby osób, które zostały zakwaterowane w dniu 2017-10-15 w pokojach ulokowanych na 2 piętrze. Zastosuj funkcję agregującą.

---

### Zadanie 8\_21

---

Utwórz kwerendę do wyświetlenia, ile wpłat za zamieszkanie w Domu Studenta w roku akademickim 2017/2018 zostało dokonanych przez mężczyzn, a ile przez kobiety. Zastosuj funkcję agregującą.

---

### Zadanie 8\_22

---

Utwórz kwerendę do wyliczenia, ile osób skierowanych przez Wydział Mechaniczny zamieszkiwało i nadal zamieszuje w Domu Studenta.

---

*Podpowiedź do wykonania zadania 8\_22.*

W celu wykonania zadania utwórz kolejno dwie kwerendy. Pierwsza z nich to zwykła kwerenda wybierająca na bazie tabeli **tZakwaterowanie**, która wyświetla pola *Wydział* oraz *ID*. Za pomocą **Właściwości kwerendy** należy wyświetlać tylko wartości unikatowe. Druga kwerenda powinna być zbudowana w oparciu o pierwszą kwerendę.

### Zadanie 8\_23

---

Wykonaj **zadanie 8\_11** za pomocą **kreatora kwerend** (odpowiedni przycisk znajdziesz w karcie wstążki **Tworzenie**).

---

### Zadanie 8\_24

---

Utwórz kwerendę krzyżową na bazie tabeli **tWpłaty** do przedstawienia średniej wartości wpłat każdego studenta z tabeli **tWpłaty** za każdy rok akademicki.

---

### Zadanie 8\_25

---

Utwórz kwerendę krzyżową na bazie tabeli **tZakwaterowanie** do przedstawienia, ile spośród zakwaterowanych osób było skierowanych do zamieszkania w każdym roku przez Wydział Mechaniczny, a ile przez Wydział Zarządzania.

---

## 9. Kwerendy funkcjonalne

Kwerendy funkcjonalne są przeznaczone do dodawania, usuwania i aktualizacji danych w wielu rekordach tabeli jednocześnie.

Możemy wyróżnić następujące kwerendy funkcjonalne: tworzącą, dołączającą, usuwającą oraz aktualizującą. Sposób zbudowania każdej z tych kwerend jest inny, jednak aktualna zostaje już nabyta wiedza o tworzeniu kryteriów kwerendy.

Podczas wykonywania zadań z tego rozdziału baza danych zostanie zmieniona, a część wierszy usunięta. Dlatego należy utworzyć kopię swojej bazy danych Dom Studenta i wszystkie zadania wykonywać w nowym pliku.

## 9.1. Kwerenda usuwająca

Kwerenda usuwająca pozwala na usunięcie grupy rekordów z jednej lub kilku tabel. Użycie tej kwerendy powoduje usunięcie całych rekordów,



nie zaś wybranych pól w rekordach. Ikonka kwerendy usuwającej jest umieszczona na wstążce, w zakładce **Projektowanie**. W kolejnym zadaniu zapoznamy się z tą kwerendą.

### Zadanie 9\_1

W bazie danych, która jest kopią bazy danych Dom Studenta, utwórz kwerendę **k9\_1** do usunięcia rekordów z tabeli **tKoszty\_zakwaterowania**. Należy usunąć tylko te wiersze, których wartość w polu *ID\_Koszty* jest wartością większą lub równą 4.

#### Wykonanie

- Utwórz nową pustą kwerendę wybierającą w oparciu o tabelę **tKoszty\_zakwaterowania**. Zapisz ją, nadając jej nazwę **k9\_1**.

**Plik** Narzędzia główne Tworzenie Dane zewnętrzne Narzędzia bazy danych **Projektowanie**

Widok Uruchom Wybierz Utwórz Dołącz Aktualizuj Krzyżowa Usuń Składająca Przekazująca Definicja danych Wstaw Usuń w Pokaż tabelę Konstruuj

Wyniki Typ kwerendy

**Kwerenda1**

**tKoszty\_zakwaterowania**

- \* ID\_Koszty
- ID\_Zakw
- Od\_kiedy
- Do\_kiedy
- Kwota\_mies\_opłaty

Pole:	ID_Koszty
Tabela:	tKoszty_zakwaterowania
Usuwanie:	Gdzie
Kryteria:	>=4

Rys. 9.1. Widok projektu kwerendy usuwającej **k9\_1**



- b. W oknie projektu kwerendy kliknij na wstążce ikonkę – postać siatki projektu zmieni się i pojawi się wiersz **Usuwanie** (rys. 9.1), a w nim – słowo **Skąd**.
- c. Aby określić kryterium usuwania rekordów, umieść w siatce projektu kwerendy pole *ID\_Koszty*. Wpisz w polu **Kryteria**:  $\geq 4$  (rys. 9.1).



- d. W celu wykonania kwerendy kliknij przycisk : Uruchom. Natychmiast pojawi się okienko dialogowe z pytaniem o potwierdzenie usunięcia rekordów. Potwierdź polecenie, wybierając opcję **Tak**.

## 9.2. Kwerenda aktualizująca

Kwerendę aktualizującą wykorzystuje się w celu dokonania globalnych zmian na grupie rekordów jednej lub kilku tabel. Oznacza to, że podając pewne kryterium, możemy zmienić dane w wielu wierszach tabeli jednocześnie. Tak samo wykorzystujemy kwerendę do modyfikacji jednej wartości określonego przez nas pola. Wtedy należy w kryterium umieścić wartość klucza podstawowego lub jednoznacznego.



Ikonka kwerendy aktualizującej jest umieszczona na wstążce, w zakładce **Projektowanie**.

### Zadanie 9\_2

Utwórz kwerendę o nazwie **k9\_2** do zmiany wartości pola *Kwota\_mies\_opłaty* tabeli **tKoszty\_zakwaterowania**. Za pomocą kwerendy należy zmniejszyć o 5 % wartości pola *Kwota\_mies\_opłaty* w tych rekordach, które w polu *Od\_kiedy* zawierają daty z poniedziałku lub środy.

### Wykonanie

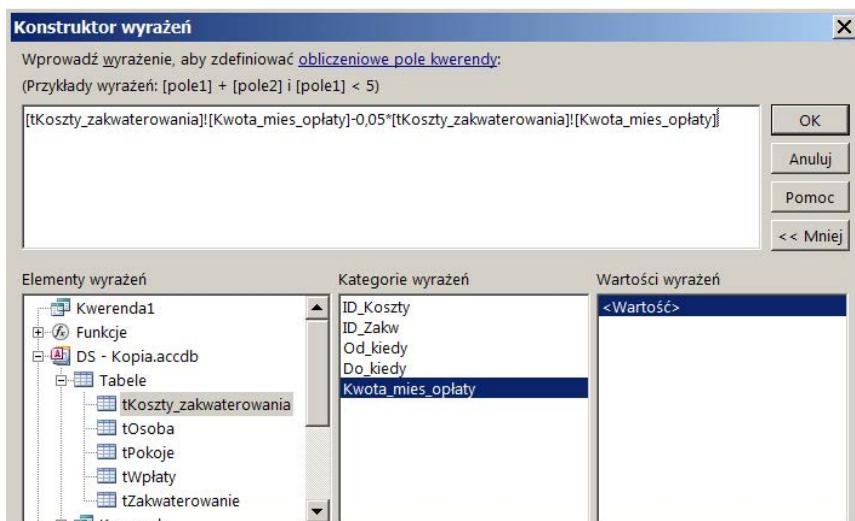
- a. Utwórz nową pustą kwerendę wybierającą w oparciu o tabelę **tKoszty\_zakwaterowania**. Zapisz ją, nadając jej nazwę **k9\_2**.
- b. W oknie projektu kwerendy kliknij na wstążce ikonkę – postać siatki projektu zmieni się i pojawi się wiersz **Aktualizacja do**.



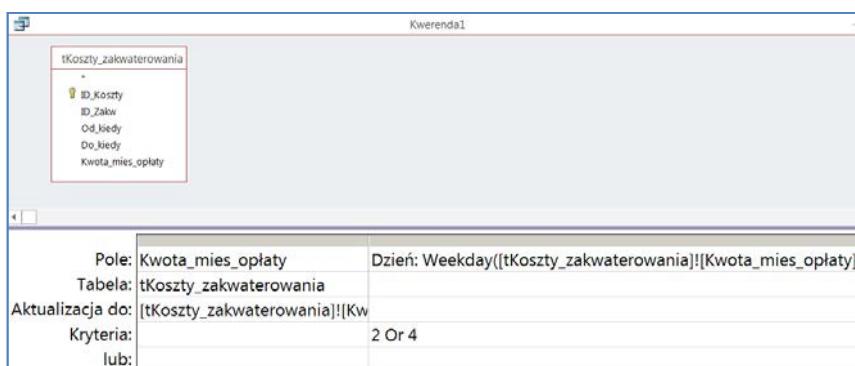
- c. Umieść w siatce kwerendy pole *Kwota\_mies\_opłaty*.  
d. W tej samej kolumnie w wierszu **Aktualizacja do** zbuduj wyrażenie przeznaczone do zmniejszenia kwoty miesięcznej opłaty o 5%. Może ono mieć postać:

**[tKoszty\_zakwaterowania]![Kwota\_mies\_opłaty] – 0.05\* [tKoszty\_zakwaterowania]![Kwota\_mies\_opłaty].**

Skorzystaj z kreatora wyrażeń (rys. 9.2).



Rys. 9.2. Widok konstruktora wyrażeń w momencie budowy wyrażenia dla kwerendy **k9\_2**



Rys. 9.3. Widok projektu kwerendy aktualizującej **k9\_2**

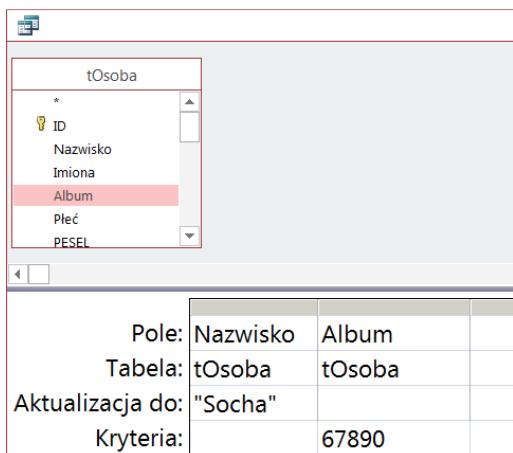
- e. Dodaj jeszcze jedną kolumnę w celu określenia kryterium. Widok projektu kwerendy przedstawia rysunek 9.3.

## Zadanie 9\_3

Utwórz kwerendę o nazwie **k9\_3** do zmiany wartości pola *Nazwisko* tabeli **tOsoba**. Za pomocą kwerendy należy zamienić nazwisko osoby, której numer albumu jest znany (np. 67890) na nazwisko „Socha”.

### Wykonanie

Projekt kwerendy **k9\_3** prezentuje rysunek 9.4.



Rys. 9.4. Widok projektu kwerendy aktualizującej **k9\_3**

## 9.3. Kwerenda tworząca tabelę

Kwerenda tego typu tworzy nową tabelę z częścią danych lub z wszystkich danych znajdujących się w jednej lub kilku tabelach.



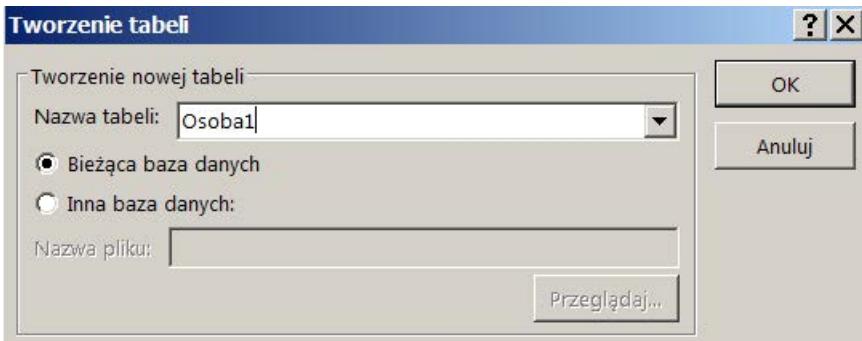
Ikonka kwerendy tworzącej jest umieszczona na wstążce, w zakładce **Projektowanie**.

## Zadanie 9\_4

Utwórz kwerendę o nazwie **k9\_4** do utworzenia tabeli o nazwie **Osoba1**, która będzie składać się z wybranych pól tabeli **tOsoba**: *ID, Nazwisko, Imiona, Album, PESEL* oraz pól zawierających dane dotyczące adresu stałego zameldowania studenta.

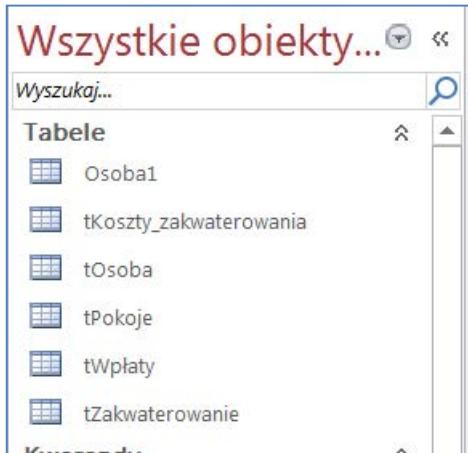
### Wykonanie

- Utwórz nową pustą kwerendę wybierającą w oparciu o tabelę **tOsoba**. Zapisz kwerendę, nadając jej nazwę **k9\_4**.
- Umieść w siatce kwerendy pola: *ID, Nazwisko, Imiona, Album, PESEL, Kod\_pocztowy, Kraj, Województwo, Miejscowość, Adres*.
- W oknie projektu kwerendy kliknij na wstążce ikonkę – na ekranie pojawi się okno dialogowe **Tworzenie tabeli** (rys. 9.5).
- W polu **Nazwa tabeli** wprowadź nazwę tabeli, która ma być utworzona: **Osoba1**.
- Pozostaw bez zmiany opcje **Bieżąca baza danych** i zatwierdź przyciskiem **OK**.



Rys. 9.5. Okno dialogowe wyświetlane przy konstruowaniu kwerendy tworzącej **k9\_3**

- Aby obejrzeć treść nowej tabeli przed jej utworzeniem, kliknij ikonkę **Widok** i wybierz **Widok arkusza danych**.
- Powróć do widoku projektu kwerendy i uruchom kwerendę.
- W okienku dialogowym, które się pojawi, potwierdź polecenie, wybierając opcję **Tak**.
- Zapisz kwerendę, otwórz okienko nawigacji i sprawdź, czy nowa tabela istnieje (rys. 9.6).



Rys. 9.6. Lista tabel w okienku nawigacji po wykonaniu kwerendy **k9\_4**

## 9.4. Kwerenda dołączająca

Kwerenda dołączająca dodaje grupę rekordów do tabeli. Rekordy te mieszczą się w innej tabeli – tabeli źródłowej. Struktura tabeli źródłowej powinna być taka sama jak struktura tabeli, do której dodajemy rekordy, czyli powinna zawierać takie same nazwy, typy i właściwości pól.

Kwerenda dołączająca może być użyteczna, kiedy na przykład dwie osoby oddzielnie wprowadzają dane do pewnej tabeli w pliku bazy danych wykonanym w **Microsoft Access**. Następnie, jeżeli wszystkie wprowadzone dane należy zgromadzić w jednym pliku, to można importować tabelę z jednej bazy do drugiej (ze zmianą nazwy tabeli), a potem skorzystać z kwerendy dołączającej. W przeciwieństwie do zwykłego kopiowania za pomocą **Kopiuj/Wklej** w tym przypadku można zdefiniować kryteria dołączania.

Bardziej skomplikowaną operacją jest realizacja pewnego algorytmu z wykorzystaniem makr i kodu języka VBA w aplikacji, która przewiduje uzupełnienie danych tabel przez użytkownika.



Ikonka kwerendy dołączającej jest umieszczona na wstążce, w zakładce **Projektowanie**.

## Zadanie 9\_5

Utwórz kwerendę o nazwie **k9\_5** w celu dołączania dwóch nowych rekordów do tabeli **tPokoje**.

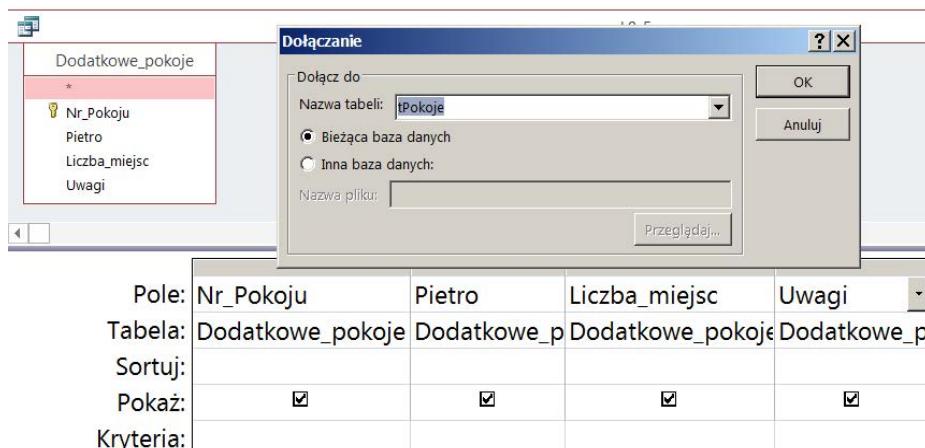
## Wykonanie

- Najpierw należy utworzyć tabelę z dwoma rekordami. W tym celu:
- Wykonaj kopię tabeli **tPokoje** (wybierz w okienku nawigacji tabelę **tPokoje** i w jej menu podręcznym kliknij **Kopiuj**, a następnie **Wklej**). Wprowadź nazwę nowej tabeli: **Dodatkowe\_pokoje**. Tabela ta zawiera takie same rekordy jak tabela **tPokoje**, a jej rekordy zawierają wartości klucza podstawowego, jakie umieszczone w rekordach tabeli **tPokoje** – dlatego tych rekordów nie możemy dołączyć do tabeli **tPokoje**.
  - o Usuń wszystkie rekordy z tabeli **Dodatkowe\_pokoje**, a następnie wprowadź do tej tabeli dwa nowe rekordy, które powinny różnić się od tych usuniętych.
- Utwórz kwerendę wybierającą w widoku projektu na bazie tabeli **Dodatkowe\_pokoje**.
- Umieść w siatce projektu kwerendy wszystkie pola tabeli **Pokoje\_dodatek**.
- Zapisz kwerendę, nadając jej nazwę **k9\_5**.



**Działacz**

- W oknie projektu kwerendy wybierz na wstążce ikonkę – wyświetli się okno dialogowe **Działczanie**.



Rys. 9.7. Widok projektu kwerendy dołączającej **k9\_5**

- Otwórz listę w polu **Nazwa tabeli** i wybierz: **tPokoje** (rys. 9.7).
- Zostaw bez zmiany opcję **Bieżąca baza danych** i kliknij przycisk **OK**.
- Aby obejrzeć rekordy przed ich dołączeniem, kliknij ikonkę **Widok** i wybierz **Widok arkusza danych**.
- Powróć do widoku projektu kwerendy i uruchom kwerendę.

- k. W okienku dialogowym, które się pojawi, potwierdź polecenie, wybierając: **Tak**.
1. Otwórz tabelę **tPokoje** i sprawdź, czy nowe wiersze zostały dołączone.

## 9.5. Kwerenda parametryczna

Kwerenda parametryczna umożliwia definiowanie kryteriów na etapie wykonania. Oznacza to, że po jej uruchomieniu na ekranie pojawiają się okna dialogowe, do których użytkownik powinien wprowadzić kryteria. Jeżeli użytkownik nie poda żadnej wartości, kwerenda parametryczna zinterpretuje wartość wejściową jako pusty ciąg.

W siatce kwerendy należy umieścić tak zwany parametr. Zawsze jest on ujęty w nawiasy kwadratowe.

Nie znajdziemy żadnej specjalnej ikonki dla kwerendy parametrycznej, ponieważ parametry może zawierać dowolna kwerenda, tak wybierająca, jak i funkcjonalna. Rozważmy dalej przykłady utworzenia kwerend parametrycznych.

### Zadanie 9\_6

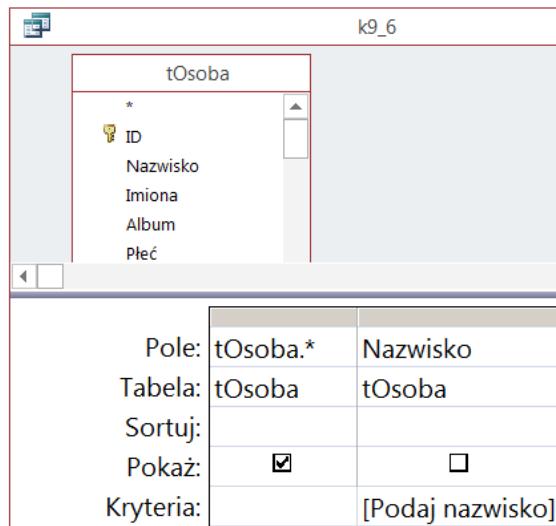
Utwórz kwerendę o nazwie **k9\_6** do wyszukiwania z tabeli **tOsoba** wszystkich informacji o dowolnej osobie według jej nazwiska. Nazwisko będzie podane przez użytkownika podczas wykonania kwerendy.

#### Wykonanie

Z treści zadania wynika, że mamy zbudować kwerendę parametryczną. Parametrem będzie nazwisko szukanej osoby. Pole zawierające nazwisko ma typ tekstowy, czyli naszym zadaniem jest utworzyć kwerendę wybierającą o jednym parametrze tekstowym. Zatem:

- a. Utwórz kwerendę wybierającą na bazie tabeli **tOsoba**.
- b. Umieść wszystkie pola tabeli **tOsoba** w siatce projektu kwerendy.
- c. Zapisz kwerendę, nadając jej nazwę **k9\_6**.
- d. W komórce **Kryteria** kolumny *Nazwisko* wpisz w nawiasach kwadratowych tekst, który zostanie wyświetlony po uruchomieniu kwerendy. Na rysunku 9.8 takim tekstem jest: „Podaj nazwisko”. Właśnie ten ciąg będzie przyjmowany jako parametr.
- e. Wybierz na wstążce, w zakładce **Projektowanie**, w grupie **Pokazywanie/ukrywanie** ikonkę





Rys. 9.8. Widok projektu kwerendy parametrycznej **k9\_6**

- f. W oknie dialogowym **Parametry kwerendy**, w kolumnie **Parametr** wprowadź powtórnie dokładnie taki sam tekst jak poprzednio, a w komórce obok wybierz z listy rozwijanej typ parametru. Jest on taki, jak typ pola, do którego parametr jest przypisany. Na rysunku 9.9 przedstawiono listę typów danych do wyboru oraz zaznaczono typ **Tekst**.

Parametry kwerendy		?	X
Parametr	Typ danych	▲	
Podaj nazwisko	Krótki tekst		

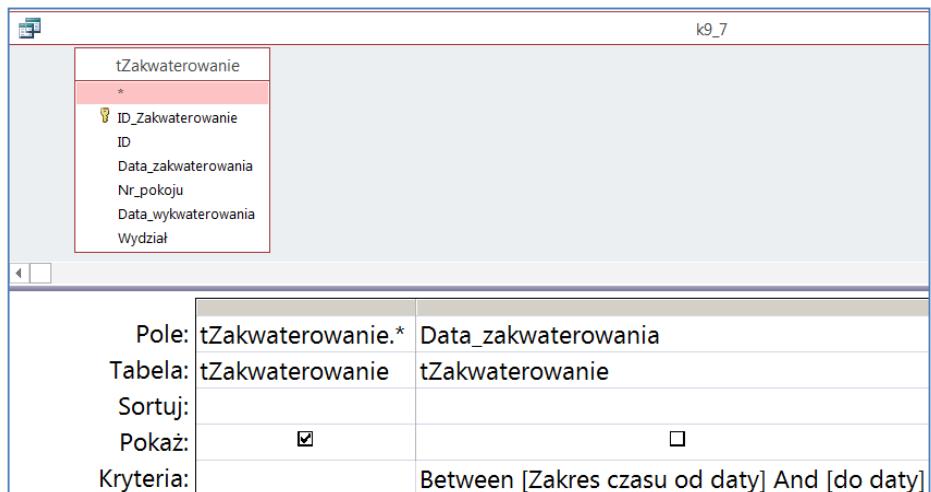
Rys. 9.9. Widok okienka przedstawiającego parametry kwerendy k9\_6

## Zadanie 9\_7

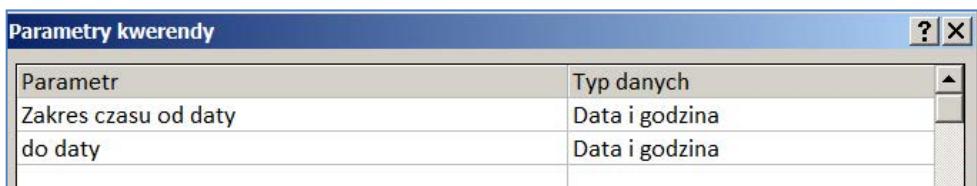
Utwórz kwerendę o nazwie **k9\_7** do wyświetlenia danych z tabeli **tzakwaterowanie**, dotyczących zakwaterowania studentów w zadanych przedziałach czasu.

## Wykonanie

Zgodnie z treścią zadania kwerenda powinna mieć dwa parametry typu **Data/godzina**. Widok projektu kwerendy **k9\_7** prezentuje rysunek 9.10, a parametry kwerendy przedstawiono na rysunku 9.11.



Rys. 9.10. Widok projektu kwerendy **k9\_7**



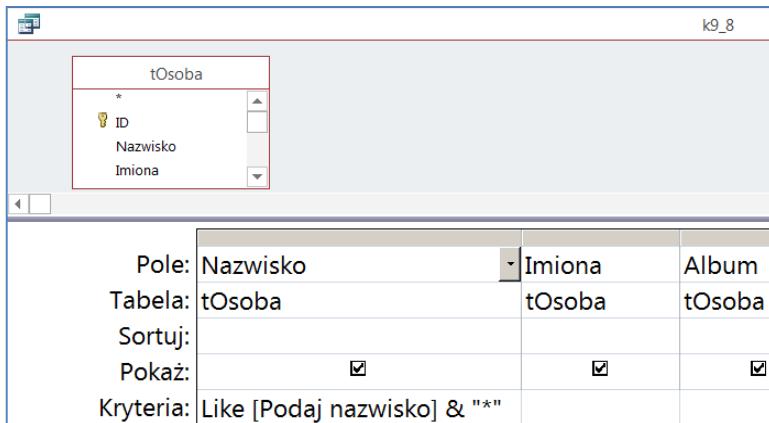
Rys. 9.11. Widok okienka przedstawiającego parametry kwerendy **k9\_7**

## Zadanie 9\_8

Utwórz kwerendę o nazwie **k9\_8** do wyświetlenia pól: *Nazwisko, Imiona, Album* tabeli **tOsoba** po podaniu dowolnej liczby początkowych liter nazwiska.

## Wykonanie

Kwerenda ta ma taki sam parametr jak kwerenda **k9\_6**, ale ma odmienne kryterium, w którym stosujemy operację połączenia ciągów (&). Projekt kwerendy przedstawia rysunek 9.12.



Rys. 9.12. Widok projektu kwerendy **k9\_8**

## Zadania do samodzielnego wykonania

### Zadanie 9\_9

Utwórz kwerendę o nazwie **k9\_9** do usunięcia z tabeli **tWpłaty** wszystkich rekordów dotyczących osoby o numerze *ID* równym 4. Te rekordy mają zawierać dane o wpłatach dokonanych w dniach 30, 3 lub 5 dowolnego miesiąca.

---

### Zadanie 9\_10

Utwórz kwerendę o nazwie **k9\_10** do zmiany wartości pola *Data\_wykwaterowania* w rekordzie, w którym wartość *ID\_zakw* jest równa 2. Po aktualizacji pole *Data\_wykwaterowania* powinno zawierać datę: 30 listopada 2017 roku.

---

### Zadanie 9\_11

Wykonaj kopię tabeli **tPokoje**. Usuń regułę poprawności pola *Liczba\_miejsc*. Utwórz kwerendę o nazwie **k9\_11** do aktualizacji danych tej tabeli dotyczących pokojów nr 4 oraz 6. Liczba miejsc w tych pokojach ma być zwiększoną o 1.

---

## Zadanie 9\_12

Utwórz kwerendę o nazwie **k9\_12** do utworzenia nowej tabeli o nazwie **Wpłaty\_2017** ze wszystkich pól tabeli **tWpłaty** o wpłatach za rok akademicki 2017/2018.

## Zadanie 9\_13

Utwórz kwerendę o nazwie **k9\_13** w celu dołączenia do tabeli **tWpłaty** dwóch nowych wierszy o wpłatach osoby o numerze *ID* równym 2. Sprawdź, czy taki numer znajduje się w tabeli **tOsoba**.

Podpowiedź do wykonania zadania 9\_13.

W siatce projektu kwerendy dołączającej nie umieszczaj pola *ID\_wpłaty*. Postępuj tak, aby w tabeli **tWpłaty** powstał kolejny numer *ID\_wpłaty*.

## Zadanie 9\_14

Utwórz kwerendę parametryczną o nazwie **k9\_14** do wyświetlenia danych z tabeli **tZakwaterowania**, dotyczących zakwaterowania studentów w zadanym przedziale wartości *ID*.

## Zadanie 9\_15

Utwórz kwerendę o nazwie **k9\_15** do wyświetlania wszystkich danych z tabeli **tOsoba** po podaniu dowolnej liczby końcowych liter pola *e\_mail*.

## Zadanie 9\_16

Utwórz kwerendę o nazwie **k9\_16** do aktualizacji danych z tabeli **tKoszty\_zakwaterowania**. Kwota miesięcznej opłaty za zamieszkanie w Domu studenta powinna być zwiększena o 2% w tych rekordach, w których data w polu *Od\_kiedy* jest równa zadanej lub większa .

## III. Manipulowanie danymi za pomocą języka SQL

**SQL** – Strukturalny Język Zapytań (ang. **S**tructured **Q**uery **LSQL** pozwala na wykonanie w bazie danych dowolnych czynności z danymi, takimi jak: wprowadzanie, aktualizacja, usunięcie i wyszukiwanie danych oraz tworzenie i modyfikowanie ich struktur (np. tabel).

Język **SQL** programu **Microsoft Access** jest zasadniczo zgodny ze specyfikacjami *ANSI - 89 Level 1*. Pomimo że powstają coraz nowsze standardy języka **SQL**, rozszerzające jego możliwości i funkcjonalność, to jego podstawowe konstrukcje nie ulegają zmianie – i właśnie takie elementy składowe zostaną omówione w tym rozdziale podręcznika.

Odnosząc do stosowanej terminologii warto zaznaczyć, że powszechnie jest stosowanie kilku terminów o tym samym znaczeniu: „instrukcja SQL”, „kwerenda”, „zapytanie”, „polecenie”, „query”.

Język **SQL** składa się z instrukcji trzech typów. Odpowiednio jest podzielony na trzy części:

- **DCL** (ang. *Data Control Language*) – Język Sterowania Danymi;
- **DDL** (ang. *Data Definition Language* – Język Definicji Danych;
- **DML** (ang. *Data Manipulation Language*) – Język Manipulacji Danymi.

Polecenia Języka Sterowania Danymi (**GRANT**, **DENY**, **REVOKE**) służą do zabezpieczenia danych i używane są przez administratorów systemów baz danych.

Polecenia Języka Definicji Danych (**CREATE**, **DROP**, **ALTER**) pozwalają pracować ze strukturami danych. Przykłady zastosowania takich instrukcji przedstawiono w rozdziale 20.

W tym rozdziale zaprezentowane zostaną przykłady instrukcji języka **DML**. Służą one do wykonania operacji na danych: do umieszczenia ich w bazie, kasowania, przeglądania oraz zmiany.

Do instrukcji **DML** zalicza się następujące polecenia:

- **SELECT** – pobranie danych z tabel,

- **INSERT** – umieszczenie danych w tabeli,
- **UPDATE** – zmiana danych w tabeli,
- **DELETE** – usunięcie danych z tabeli.

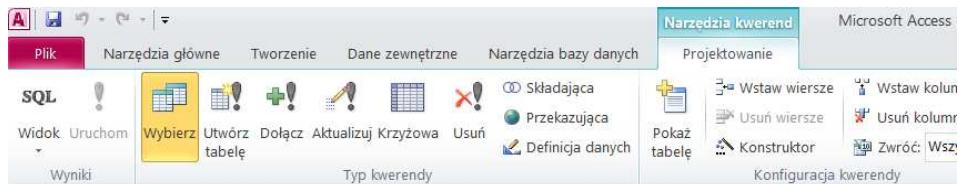
Instrukcja **SELECT** ma złożoną składnię i jest niczym innym, jak znaną z poprzednich rozdziałów kwerendą wybierającą. Wykorzystujemy ją do pobrania i wyświetlenia danych, a także do przeprowadzania obliczeń na danych.

Operacje wprowadzenia, aktualizacji i usuwania danych wykonywaliśmy do tej pory ręcznie lub za pomocą kwerend funkcjonalnych. W dalszej części rozdziału zaprezentowano, jak można realizować te czynności za pomocą instrukcji **INSERT**, **UPDATE**, **DELETE**.

Do tworzenia zapytań **SQL** w programie **Microsoft Access** przeznaczono okno **Widoku SQL** kwerendy. Algorytm wejścia do okna edytora instrukcji **SQL** jest następujący:



- a. Na zakładce wstążki **Tworzenie** wybieramy ikonkę .
- b. W oknie projektu kwerendy zamykamy okienko **Pokazywanie tabeli**, nie zaznaczając żadnej z tabel bazy danych. Wówczas widok wstążki się zmieni i uaktywni się zakładka **Projektowanie** (rys. 10.1).



Rys. 10.1. Wstążka programu z aktywną ikonką **SQL Widok**

- c. Klikamy ikonkę **Widok**, wówczas na ekranie pojawi się okno edytora poleceń SQL, a w nim – początek instrukcji **SELECT** (rys. 10.2).



Rys. 10.2. Okno do zapisu instrukcji **SQL**

- d. Zapisujemy tekst instrukcji **SQL** w okienku, które się pojawi; możemy tu zapisać dowolną instrukcję **języka DML** (nie tylko **SELECT**).
- e. Co istotne, małe i wielkie litery nie są rozróżniane, możemy także dodać dowolną liczbę spacji między standardowymi nazwami.
- f. Na końcu instrukcji zamieszczamy średnik, chociaż nie jest on obowiązkowy.



- g. W celu wykonania instrukcji wybieramy na wstążce ikonkę .

Jeśli po wykonaniu instrukcji uznamy, że działa ona nieprawidłowo, możemy poprawić jej tekst i uruchomić ponownie. Oznacza to, że **Widok SQL** służy zarówno do zapisu nowej instrukcji w języku **SQL**, jak i do jej modyfikacji.

## 10. Pobieranie danych z jednej tabeli

W tym rozdziale przedstawiono przykłady utworzenia instrukcji **SQL** do pobierania i wyświetlenia danych z jednej tabeli.

### 10.1. Instrukcja SELECT do wybierania określonych pól tabeli

Instrukcję **SELECT** wykorzystujemy w celu pobrania i wyświetlenia danych. Możemy także wykonać w niej obliczenia na danych.

Części instrukcji **SQL** nazywane są **klauzulami**. Instrukcja **SELECT** ma dosyć złożoną składnię. Najpierw przedstawimy jej uproszczoną wersję.

W uproszczonej instrukcji **SELECT**, przeznaczonej do wyświetlania danych z jednej tabeli, zazwyczaj trzeba określić:

- z jakich pól (kolumn) mają być wyświetlane dane (klauzula **SELECT**);
- z jakiej tabeli bazy danych mają być sprowadzone dane (klauzula **FROM**);
- jakie kryterium mają spełniać dane (klauzula **WHERE**).

Składnia najprostszej instrukcji **SELECT** ma postać:

```
SELECT      nazwa_pola1, nazwa_pola2, ...
FROM        nazwa_tabeli
WHERE       kryterium;
```

Nazwa pola lub tabeli powinna zostać ujęta w nawiasy kwadratowe. Jeśli nazwa nie zawiera żadnych spacji ani znaków specjalnych (takich jak znaki interpunkcyjne), nawiasy kwadratowe nie są obowiązkowe.

Kryterium wyszukiwania, które umieszczamy w klauzuli **WHERE**, jest pewnym wyrażeniem, które jest dowolną kombinacją operatorów matematycznych i logicznych, stałych, funkcji, nazw pól, formantów i właściwości. Wyrażenie budujemy na tych samych zasadach jak w środowisku graficznym. Jedyną różnicą jest konieczność stosowania w wyrażeniu obok operatorów nazw pól tabeli.

### *Wyświetlenie danych ze wszystkich rekordów tabeli*

Jeżeli próbujemy wyświetlić dane ze wszystkich rekordów (wierszy) tabeli, to wyszukiwanie nie zależy od żadnego kryterium. Należy tylko wymienić nazwy pól (kolumn), w których ulokowane są interesujące nas dane. Wówczas odpowiednia instrukcja **SELECT** nie zawiera klauzuli **WHERE** i ma następującą postać:

```
SELECT      nazwa_pola1, nazwa_pola2,...
FROM        nazwa_tabeli;
```

## Zadanie 10\_1

Utwórz instrukcję **SQL** o nazwie **sql10\_1** w celu wyświetlenia nazwiska oraz imienia z każdego rekordu tabeli **tOsoba**.

### **Wykonanie**

- Otwórz okno do utworzenia instrukcji **SQL**.
- W **widoku SQL** wprowadź tekst instrukcji:

```
SELECT Nazwisko, Imiona
FROM    tOsoba;
```

- Wybierz przycisk **Uruchom** – kwerenda zostanie wykonana.
- Zapisz instrukcję, nadając jej nazwę **sql10\_1**, i zamknij okno.

Wyświetlenie danych ze wszystkich pól tabeli można zrealizować, nie wypisując wszystkich pól tabeli, lecz umieszczając w klauzuli **SELECT** symbol wieloznaczny \*, tak jak zostało to zaprezentowane w kolejnym przykładzie.

## Zadanie 10\_2

Utwórz instrukcję **SQL** o nazwie **sql10\_2** w celu wyświetlenia wszystkich danych z tabeli **tOsoba**.

---

### Wykonanie

Wymagana instrukcja **SELECT** ma postać:

```
SELECT      *
FROM        tOsoba;
```

*Ograniczenie liczby wyświetlanych rekordów  
(klauzula SELECT TOP)*

W przedstawionych powyżej instrukcjach domyślnie przyjęte jest, że wyświetlane będą wszystkie rekordy tabeli. Jednak gdy tabela zawiera dużo danych, można ograniczyć liczbę zwracanych wierszy, używając klauzuli **SELECT TOP n**, co dosłownie oznacza: „*Wyświetl pierwsze n rekordów*”.

## Zadanie 10\_3

Utwórz instrukcję **SQL** o nazwie **sql10\_3** w celu wyświetlenia pierwszych trzech rekordów tabeli **tOsoba**.

---

### Wykonanie

W celu wykonania zadania zapiszemy instrukcję podobną do **sql10\_2** oraz dodamy opcję **TOP 3** (gdzie 3 oznacza liczbę pierwszych trzech wierszy tabeli). Wymagana instrukcja **SELECT** ma postać:

```
SELECT TOP 3 *
FROM      tOsoba;
```

*Alias (AS) – alternatywny nagłówek kolumny*

Zarówno w środowisku graficznym programu **Microsoft Access**, jak i w **języku SQL** istnieje możliwość przypisania do wybranych pól innych nazw (aliasów). W czasie wyświetlania danych pełnią one rolę nagłówków kolumn zamiast oryginalnych nazw kolumn w tabeli. Wykorzystujemy w tym celu słowo kluczowe **AS**.

## Zadanie 10\_4

Utwórz instrukcję **SQL** o nazwie **sql10\_4** w celu wyświetlenia pól tabeli **tOsoba**: *Nazwisko*, *Imiona*, *Album*. Nagłówki wyświetlonych kolumn powinny mieć postać: *Nazwisko studenta*, *Imię studenta*, *Numer albumu*.

### Wykonanie

Instrukcja **sql10\_4** ma postać:

```
SELECT      Imiona AS [Imię Studenta],  
            Nazwisko AS [Nazwisko Studenta],  
            Album AS [Numer Albumu]  
FROM        t Osoba;
```

Zwróć uwagę, że nowe nagłówki kolumn zostały zapisane z użyciem nawiasów kwadratowych. Postępujemy tak zawsze, gdy nazwa nagłówka zawiera spację.

Na rysunku 10.3 przedstawiono wynik wykonania instrukcji **sql10\_4**.

Imię Studenta	Nazwisko Studenta	Numer Albumu
Anna	Malinowska	65555
Adam	Milewski	965555
Ferdynand	Cieplak	911114
Wacław	Lityński	311119
Adam	Lenic	471187
Marek	Socha	565700
Wanda	Matysiak	511111

Rys. 10.3. Wynik wykonania instrukcji **sql10\_4**

### Pole obliczeniowe w instrukcji **SELECT**

Tak jak zaprezentowano w rozdziale 8, kwerenda wybierająca może zawierać pole obliczeniowe. Zatem w instrukcji **SELECT**, będącej kwerendą wybierającą, możemy umieścić dowolne wyrażenie do wyliczenia. Kolejny przykład przedstawia pole obliczeniowe w postaci prostego wyrażenia arytmetycznego.

## Zadanie 10\_5

Utwórz instrukcję **SQL** o nazwie **sql10\_5** do wyświetlania wszystkich danych z tabeli **tWpłaty** oraz wartości pola *Wpłata* powiększonych o 10%.

---

### Wykonanie

Instrukcja **sql10\_5** ma postać:

```
SELECT *, Wpłata * 1.1 AS [Wpłata powiększona o 10%]
FROM tWpłaty;
```

Mogimy zapisać ją także w postaci:

```
SELECT ID_Wpłaty, ID, Pokwitowanie, Data_wpłaty, Wpłata,
       Wpłata + Wpłata * 0.1 AS [Wpłata powiększona o 10%]
FROM tWpłaty;
```

Należy zwrócić uwagę na zastosowanie kropek w liczbach rzeczywistych (w projekcie graficznym kwerendy były używane przecinki).

*Połączenie danych tekstowych z różnych pól  
(zastosowanie operatora konkatenacji &)*

Operator konkatenacji napisów **&** stosujemy do wyświetlenia w jednej kolumnie danych z wielu pól tekstowych.

## Zadanie 10\_6

Utwórz instrukcję **SQL** o nazwie **sql10\_6** do wyświetlania jednej kolumny, która będzie zawierała imiona i nazwiska studentów. Nagłówek tej kolumny powinien mieć postać: „Imię i nazwisko studenta/studentki”.

---

### Wykonanie

Instrukcja **SQL** do realizacji zadania:

```
SELECT Imiona & " " & Nazwisko
      AS [Imię i nazwisko studenta/studentki]
FROM tOsoba;
```

Wynik wykonania kwerendy przedstawiono na rysunku 10.4.

Imię i nazwisko studenta/studentki
Anna Malinowska
Adam Milewski
Ferdynand Cieplak
Wacław Lityński
Adam Lenic
Marek Socha

Rys. 10.4. Wynik wykonania instrukcji **sql10\_6**

### Eliminowanie powtórzeń przy wyświetleniu wartości pola (słowo kluczowe DISTINCT)

Przy wybieraniu danych pewnego pola ta sama wartość może powtarzać się wielokrotnie. Jeśli jest wymagane wyświetlenie każdej wartości tylko jeden raz, należy zastosować operator **DISTINCT**, zapisując go przed nazwą pola.

## Zadanie 10\_7

Za pomocą instrukcji **SQL** o nazwie **sql10\_7** utwórz listę imion z pola *Imiona* wprowadzonych do tabeli **tOsoba**. Imiona nie mogą się powtarzać (tzn. listę wartości unikatowych).

### Wykonanie

```
SELECT DISTINCT Imiona
FROM    tOsoba;
```

## 10.2. Zastosowanie klauzuli WHERE

Rozważmy dalej instrukcję **SELECT** zawierającą klauzulę **WHERE**. Służy ona do zdefiniowania kryterium wyszukiwania danych w tabeli. Kryterium może być bardzo prostym, jak również i złożonym wyrażeniem zawierającym, pomimo nazw pól, specjalne operatory (np. **LIKE**) i funkcje wbudowane. Informacje na temat budowy wyrażeń były podane w rozdziale 6. Pozostają one aktualne również przy tworzeniu instrukcji **SQL**.

Następne zadanie ilustruje strukturę instrukcji **SQL** zawierającą klauzulę **WHERE**.

## Zadanie 10\_8

Utwórz instrukcję **SQL** o nazwie **sql10\_8** do wyświetlenia wszystkich pól rekordu tabeli **tOsoba**, którego wartość w polu **ID** jest równa 4.

---

### Wykonanie

Możemy porównać instrukcję **SQL** z projektem kwerendy wybierającej realizującej postawione zadanie. Na rysunku 10.5 przedstawiono siatkę projektu kwerendy wybierającej (po lewej stronie) oraz instrukcję **sql10\_8** (po prawej stronie).

Pole:	tOsoba.*	ID	
Tabela:	tOsoba	tOsoba	
Sortuj:			
Pokaż:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Kryteria: lub:		4	

```
SELECT *
FROM tOsoba
WHERE ID = 4;
```

Rys. 10.5. Wynik wykonania instrukcji **sql10\_8**

### Operatory *LIKE* oraz *NOT LIKE*

Operator **LIKE** wykorzystuje się do sprawdzania, czy w danym napisie występuje określony wzorzec. Operator **NOT LIKE** jest zaprzeczeniem wyżej opisanego działania. Zazwyczaj stosujemy ten operator do utworzenia kryterium na danych tekstowych.

## Zadanie 10\_9

Utwórz instrukcję **SQL** o nazwie **sql10\_9** do wybierania z tabeli **tOsoba** nazwisk, imion, numerów albumów studentów, których nazwisko zaczyna się na literę „K”.

---

### Wykonanie

```
SELECT Imiona, Nazwisko, Album
FROM tOsoba
WHERE Nazwisko Like "K*";
```

---

## Zadanie 10\_10

Utwórz instrukcję SQL o nazwie **sql10\_10** do wybierania z tabeli **tOsoba** nazwisk, imion, numerów albumów studentów, których nazwiska nie kończą się na literę „z”.

---

### Wykonanie

```
SELECT Imiona, Nazwisko, Album  
FROM tOsoba  
WHERE Nazwisko NOT LIKE "*z";
```

*Operatory IS NULL oraz IS NOT NULL*

Operator **IS NULL** jest używany do sprawdzania, czy rozpatrywane pole jest puste (**NULL**). Odpowiednio operator **IS NOT NULL** służy do sprawdzania, czy wartość pola nie jest pusta.

---

## Zadanie 10\_11

Utwórz instrukcję SQL o nazwie **sql10\_11** do wybierania z tabeli **tOsoba** nazwisk, imion, numerów albumów studentów, których pole *Nazwisko\_rodowe\_matki* jest puste.

---

### Wykonanie

```
SELECT Imiona, Nazwisko, Album  
FROM tOsoba  
WHERE Nazwisko_rodowe_matki IS NULL;
```

---

## Zadanie 10\_12

Utwórz instrukcję SQL o nazwie **sql10\_12** do wybierania z tabeli **tOsoba** rekordów, w których wartość w polu *Strona\_WWW* nie jest pusta.

---

### Wykonanie

```
SELECT *  
FROM tOsoba  
WHERE Strona_WWW IS NOT NULL
```

## Operatory BETWEEN oraz NOT BETWEEN

Operator **BETWEEN** sprawdza, czy dana wartość należy do określonego przedziału wartości. Końce przedziału są wliczane do przedziału. Odpowiednio operator **NOT BETWEEN** sprawdza, czy dana wartość nie należy do określonego przedziału wartości.

### Zadanie 10\_13

---

Utwórz instrukcję SQL o nazwie **sql10\_13** do wyświetlenia rekordów tabeli **tOsoba**, w których wartości *ID* znajdują się w przedziale wartości <2; 4>.

---

#### Wykonanie

```
SELECT *
FROM   tOsoba
WHERE  ID BETWEEN 2 AND 4;
```

### Zadanie 10\_14

---

Utwórz instrukcję SQL o nazwie **sql10\_14** do wyświetlenia rekordów tabeli **tOsoba**, w których wartości *ID* znajdują się poza przedziałem wartości <2; 4>.

---

#### Wykonanie

```
SELECT *
FROM   tOsoba
WHERE  ID NOT BETWEEN 2 AND 4;
```

## Operatory IN oraz NOT IN

Operator **IN** określa, czy wartość wyrażenia jest równa jednej z wartości wymienionych na liście.

Operator **NOT IN** jest zaprzeczeniem działania operatora **IN**.

Lista elementów powinna być ujęta w nawiasy okrągłe. Elementy listy rozdzielamy za pomocą przecinków.

Elementami listy mogą być wartości pól tekstowych, liczbowych lub dat. Nie stosujemy w zapisie elementu listy symboli wieloznacznych.

Kolejne dwa zadania ilustrują sposób zastosowania tych operatorów.

## Zadanie 10\_15

Utwórz instrukcję **SQL** o nazwie **sql10\_15** do wyświetlenia rekordów tabeli **tOsoba**, w których pole *Imiona* zawiera wartości „Adam”, „Michał” lub „Stanisław”.

### Wykonanie

```
SELECT *
FROM tOsoba
WHERE Imiona IN ( "Adam", "Michał", "Stanisław" );
```

## Zadanie 10\_16

Utwórz instrukcję **SQL** o nazwie **sql10\_16** do wyświetlenia rekordów tabeli **tOsoba**, w których pole *ID* nie zawiera wartości 2, 4 lub 6.

### Wykonanie

```
SELECT *
FROM tOsoba
WHERE ID NOT IN (2,4,6);
```

*Wykorzystanie operatorów logicznych  
OR, AND, NOT*

Tworząc kryterium w klauzuli **WHERE**, proste warunki logiczne możemy łączyć spójnikami logicznymi: **OR** (alternatywą „lub”), **AND** (koniunkcją „i”) oraz **NOT** (negacją - „nieprawda, że”). Jeśli kryterium jest złożone i zawiera operator **OR**, a również **AND** lub **NOT**, należy wykorzystywać nawiasy okrągłe. Brak nawiasów może spowodować utworzenie błędnego kryterium, gdyż operacja **NOT** ma z nich najwyższy priorytet, następny ma operacja **AND**, a najsłabszą operacją jest operacja **OR**.

## Zadanie 10\_17

Utwórz instrukcję **SQL** o nazwie **sql10\_17** do wyświetlenia wszystkich danych z tabeli **tOsoba** o studentach, których imię zaczyna się na literę „A” lub „K”, a nazwisko zaczyna się od litery „M” lub „S”.

## Wykonanie

Instrukcję **sql10\_17** z zastosowaniem **AND** i **OR** możemy zapisać w postaci:

```
SELECT      *
FROM        tOsoba
WHERE       ((Imiona Like "A*") OR (Imiona Like "K*"))
            AND ((Nazwisko Like "M*") OR (Nazwisko Like "S*"));
```

## Zadanie 10\_18

---

Utwórz instrukcję SQL o nazwie **sql10\_18** do wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie** dotyczących zakwaterowania studentów wydziałów Mechanicznego lub Zarządzania w dowolnym roku oprócz 2014 lub 2016.

---

## Wykonanie

Można wykonać to zadanie wieloma sposobami. Jeden z nich to zapis:

```
SELECT      *
FROM        tZakwaterowanie
WHERE
    (Data_zakwaterowania NOT BETWEEN #2014-01-01#
        AND #2014-12-31#)
    AND (Data_zakwaterowania NOT BETWEEN #2016-01-01#
        AND #2016-12-31#)
    AND ((Wydział LIKE "Mechaniczny")
        OR (Wydział LIKE "Zarządzania"));
```

A oto inny sposób, bez użycia operatora **BETWEEN**:

```
SELECT      *
FROM        tZakwaterowanie
WHERE
    NOT(
        ((Data_zakwaterowania >= #2014-01-01#) AND
        (Data_zakwaterowania <= #2014-12-31#))
    OR
        ((Data_zakwaterowania >= #2016-01-01#) AND
        (Data_zakwaterowania <= #2016-12-31#)))
    )
AND (Wydział IN ("Mechaniczny", "Zarządzania"));
```

## Wykorzystanie funkcji wbudowanych

Tak samo jak przy graficznym tworzeniu kwerendy wybierającej, możemy w tekście instrukcji **SELECT** umieszczać funkcje wbudowane. Kolejne zadanie przedstawia zastosowanie funkcji **YEAR** (zwraca wartość roku), **MONTH** (zwraca numer miesiąca) oraz **WEEKDAY** (zwraca numer dnia tygodnia) należących do kategorii **Data/Godzina**. W podobny sposób można stosować dowolną z funkcji wbudowanych programu.

### Zadanie 10\_19

Utwórz instrukcję SQL o nazwie **sql10\_19** do wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie** dotyczących zakwaterowania studentów w październiku oraz grudniu 2016 roku, w poniedziałki.

#### Wykonanie

Numery miesięcy października i grudnia – to 10 i 12, a numer domyślnie przypisany do poniedziałku jest 2 (1 – to niedziela, 7 – sobota). Odpowiednia instrukcja **sql10\_19** może być zapisana w postaci:

```
SELECT      *
FROM        tZakwaterowanie
WHERE       (YEAR(Data_zakwaterowania) = 2016) AND
           ((MONTH(Data_zakwaterowania) = 10) OR
            (MONTH(Data_zakwaterowania) = 12)) AND
           (WEEKDAY(Data_zakwaterowania) = 2)
```

### 10.3. Sortowanie danych

Dane wyświetlane za pomocą instrukcji **SELECT** można uporządkować rosnąco lub malejąco. Jeżeli zestaw wynikowy ma być posortowany według jednego pola (przykładowo *pola1*), to stosujemy instrukcję **SELECT** w postaci:

```
SELECT      nazwa_pola1, nazwa_pola2, ...
FROM        nazwa_tabeli
WHERE       kryterium
ORDER BY    nazwa_pola1 specyfikator;
```

Tu *specyfikator* oznacza porządek sortowania wierszy wynikowych – **ASC** (rosnący) lub **DESC** (malejący). Można nie podawać ani **ASC**, ani **DESC**. Wtedy przyjmowany jest specyfikator **ASC**. Domyślnie program **Microsoft Access** sortuje wartości w kolejności rosnącej (A-Z, od najmniejszych do największych).

Dane można sortować według dwóch lub więcej pól. Przykładowo, jeśli zestaw wynikowy ma być uporządkowany według dwóch pól, to składnia klauzuli **ORDER BY** będzie następująca:

```
ORDER BY nazwa pola1 specyfikator1,  
nazwa pola2 specyfikator2;
```

Należy nadmienić, że w tym przypadku sortowanie według drugiego pola odbędzie się wyłącznie w tych rekordach, w których wartości w pierwszym polu są takie same.

## Zadanie 10\_20

---

Utwórz instrukcję **SQL** o nazwie **sql10\_20** do wyświetlenia pól: *Kraj*, *Województwo*, *Miejscowość*, *Album*, *Nazwisko*, *Imiona* z tabeli **tOsoba**, w których numer albumu jest spoza zakresu <60000; 60020>. Należy posortować dane rosnąco według kraju, województwa oraz miejscowości, a także malejąco według pola *Album*.

---

### Wykonanie

```
SELECT Kraj, Województwo, Miejscowość, Album,  
       Nazwisko, Imiona  
FROM   tOsoba  
WHERE NOT ((Album >= 60000) AND (Album <= 60020))  
ORDER BY Kraj, Województwo, Miejscowość ASC, Album DESC;
```

## Zadanie 10\_21

---

Utwórz instrukcję **SELECT** o nazwie **sql10\_21** w celu wyświetlenia wszystkich rekordów z tabeli **tOsoba** spełniających następujące kryteria:

- nazwisko zaczyna się od litery „P” lub „W”,
- imię kończy się na „a”,
- województwo – podlaskie lub mazowieckie,
- dzień urodzenia – 3, 15 lub 18 dzień miesiąca.

Wynikowe dane należy posortować alfabetycznie według nazwiska i imienia oraz malejąco według numeru albumu.

---

## Wykonanie

Przy tworzeniu instrukcji **sql10\_21** zastosujemy funkcje wbudowane **MID** oraz **VAL**.

Funkcja tekstowa **MID** była już używana w zadaniach rozdziału 7. Teraz, przy zastosowaniu jej w instrukcji **SQL**, zamiast średników należy stosować przecinki. Ogólna postać funkcji:

**Mid (ciag, start, długość),**

gdzie:

- **ciag** – wyrażenie tekstowe,
- **start** – wartość liczbową – numer pozycji znaku w ciągu, od której rozpoczyna się część do pobrania;
- **długość** – liczba znaków do zwrócenia (może być pominięta, wówczas wszystkie znaki, zaczynając od pozycji start, będą zwrócone).

Jeśli ciąg zawiera wartość **Null**, zostanie zwrócona wartość **Null**.

Przykładem funkcji konwersji jest funkcja **VAL**, która zwraca liczby zawarte w ciągu jako wartość numeryczną odpowiedniego typu. Funkcja **VAL** nie może konwertować pustego ciągu. Dlatego w naszej bazie należy wyeliminować rekordy, w których pole PESEL jest puste.

Tekst instrukcji **sql10\_21** ma następującą postać:

```
SELECT *
FROM tOsoba
WHERE
    (PESEL IS NOT NULL )
    AND
    ((Nazwisko Like "P*") OR (Nazwisko Like "W*"))
    AND
    (Imiona Like "*a")
    AND
    ((Województwo Like "podlaskie")
     OR (Województwo Like "mazowieckie"))
    AND
    (
        (VAL(MID ([PESEL], 5, 2)) = 3) OR
        (VAL(MID ([PESEL], 5, 2)) = 15) OR
        (VAL(MID ([PESEL], 5, 2)) = 18)
    )
ORDER BY Nazwisko ASC, Imiona ASC, Album DESC;
```

## 10.4. Zapytania parametryczne

W rozdziale 9 zapoznaliśmy z możliwością utworzenia kwerendy parametrycznej. Pozwala ona na zadawanie wartości kryteriów w trakcie wykonania kwerendy. Zapytanie **SQL** także może zawierać kryteria parametryczne. Wystarczy w kryterium ująć jakiś ciąg znaków w nawiasy kwadratowe a będzie traktowany jako parametr. Kolejne zadanie może służyć jako przykład parametrycznego zapytania **SQL**.

### Zadanie 10\_22

---

Utwórz instrukcję **SELECT** o nazwie **sql10\_22** w celu wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie**, które zawierają datę zakwaterowania oraz datę zakwaterowania zadane na etapie wykonania.

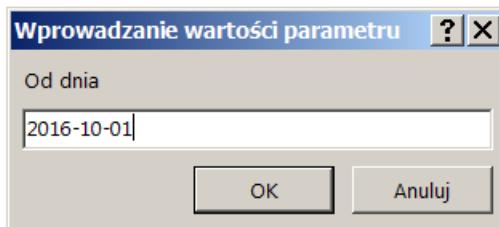
---

#### Wykonanie

Instrukcja **sql10\_22** o dwóch parametrach **Od dnia** oraz **Do dnia** może być zapisana w postaci:

```
SELECT *
FROM tZakwaterowanie
WHERE Data_zakwaterowania BETWEEN [Od dnia] AND [Do dnia];
```

Po uruchomieniu polecenia na ekranie kolejno będą wyświetlane okna dialogowe do wprowadzenia dat. Przykład pierwszego z nich wraz z wprowadzoną wartością przedstawia rysunek 10.6.



Rys. 10.6. Okno dialogowe w trakcie wykonania zapytania **sql10\_22**

Instrukcja **sql10\_22** jest przypadkiem prostego polecenia **SQL**, dla którego program **Microsoft Access** może sam prawidłowo określić typ danych (**DateTime**) wartości podstawianej przez użytkownika. Jednak

w bardziej złożonych poleceniach należy deklarować typy parametrów. Wymienione polecenie **SQL** uzupełnione deklaracją parametrów ma postać:

**PARAMETERS**

```
[Od dnia] DateTime, [Do dnia] DateTime;  
SELECT *  
FROM tZakwaterowanie  
WHERE Data_zakwaterowania BETWEEN [Od dnia] AND [Do dnia];
```

Podsumowując, należy zauważyć, że instrukcje rozważane w kolejnych rozdziałach także mogą zawierać parametry. Definiujemy je na tych samych zasadach, jakie zaprezentowano w przykładzie **Zadania 10\_22**.

## Zadania do samodzielnego wykonania

### Zadanie 10\_23

Utwórz instrukcję **SQL** w celu wybierania wszystkich danych z tabeli **tPokoje**. Wykonaj to zadanie dwukrotnie: z zastosowaniem wieloznacznika (\*) oraz bez niego.

### Zadanie 10\_24

Utwórz instrukcję **SQL** do wyświetlania wszystkich pól tabeli **tKoszty\_zakwaterowania** oraz pola obliczeniowego *Nowa kwota*, które zawiera wartość pola *Kwota\_mies\_opłaty* zmniejszoną o 25 zł.

### Zadanie 10\_25

Utwórz instrukcję **SQL**, w której zastosuj operator konkatenacji w celu wyświetlenia w jednej kolumnie wszystkich danych o miejscu stałego zamieszkania studentów. Poza tą kolumną należy wyświetlić jeszcze jedną - numer *ID*.

### Zadanie 10\_26

Za pomocą instrukcji **SELECT** utwórz listę nazw wydziałów wprowadzonych do tabeli **tZakwaterowanie**. Nazwy nie mogą się powtarzać.

## Zadanie 10\_27

Utwórz instrukcję **SQL** do wyświetlania wszystkich danych z tabeli **tPokoje** o numerach 15 oraz 25.

---

## Zadanie 10\_28

Utwórz instrukcję **SQL** do wyświetlania wszystkich danych z tabeli **tOsoba** o studentach, których imię zaczyna się od dwóch zadanych liter, przykładowo od „Ka”.

---

## Zadanie 10\_29

Utwórz instrukcję **SQL** do wyświetlania z tabeli **tZakwaterowanie** wszystkich rekordów, w których pole *Data\_wykwaterowania* nie jest puste.

---

## Zadanie 10\_30

Utwórz instrukcję **SQL** do wyświetlenia wszystkich rekordów z tabeli **tWpłaty** dotyczących kwot wpłat mieszczących się w przedziale <100zł; 200zł>. (Przy wykonaniu zadania nie należy dopisywać w kryterium „zł” do liczb).

---

## Zadanie 10\_31

Utwórz instrukcję **SQL** do wyświetlenia wszystkich rekordów z tabeli **tWpłaty** dotyczących wpłat dokonanych w dniach od 10 do 30 listopada 2016 roku.

---

## Zadanie 10\_32

Utwórz instrukcję **SQL** do wyświetlenia wszystkich rekordów z tabeli **tKoszty\_zakwaterowania** zawierających w polu *Kwota\_mies\_opłaty* wartości spoza przedziału <150zł; 250zł>, a w polu *Od\_kiedy* - daty z ostatnich pięciu lat.

---

## Zadanie 10\_33

Utwórz dowolną instrukcję **SELECT** w celu sprawdzenia działania operatora **IN** na danych pola *Imiona* tabeli **tOsoba**.

## Zadanie 10\_34

Zapisz instrukcję **SQL** do wyświetlenia wszystkich rekordów z tabeli **tZakwaterowanie** dotyczących osób skierowanych do Domu Studenta przez dowolny wydział uczelni. Wyklucz wydziały: Mechaniczny i Elektryczny.

## Zadanie 10\_35

Utwórz instrukcję **SQL** do wyświetlenia wszystkich danych z tabeli **tOsoba** o studentach zamieszkałych na stałe w województwie podlaskim, w Augustowie lub w Sokółce.

## Zadanie 10\_36

Utwórz instrukcję **SQL** do wyświetlenia wszystkich danych z tabeli **tPokoje** o dwuosobowych pokojach ulokowanych na 3 lub 5 piętrze.

## Zadanie 10\_37

Utwórz instrukcję **SQL** do wyświetlenia wszystkich danych z tabeli **tWpłaty** dotyczących wpłat dokonanych za wszystkie lata, poza rokiem akademickim 2015/2016.

## Zadanie 10\_38

Utwórz instrukcję **SQL** do wyświetlenia wszystkich wierszy z tabeli **tZakwaterowanie**, których wartość *Data\_zakwaterowania* jest z miesiąca listopada lub stycznia, a dzień zakwaterowania jest z przedziału <10; 20>. W instrukcji wykorzystaj funkcję **MONTH** oraz **DAY**.

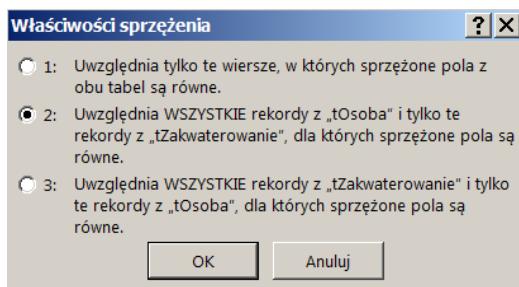
## 11. Pobieranie danych z połączonych tabel

Najczęściej informacje otrzymywane z bazy danych są danymi nie z jednego, lecz z wielu źródeł. W tym rozdziale nauczymy się budować instrukcję **SELECT** opartą na wielu tabelach.

W celu wyświetlenia danych z dwóch tabel w instrukcji **SELECT**, w klauzuli **FROM** używa się operacji sprzężenia tabel. Taką operacją może być jedna z wymienionych poniżej:

- **INNER JOIN** – złączenie wewnętrzne,
- **LEFT JOIN** – złączenie zewnętrzne lewostronne,
- **RIGHT JOIN** – złączenie zewnętrzne prawostronne.

We wcześniejszych rozdziałach, korzystając z narzędzi graficznych, łączyliśmy dwie tabele ze sobą w oknie **Relacje**. Mieliśmy wtedy możliwość wyboru typu sprzężenia. Przykładowo w trakcie połączenia tabel **tOsoba** i **tZakwaterowanie** okienko do wyboru typu złączenia miało postać jak na rysunku 11.1. Tabela **tOsoba** jest tabelą nadrzędną, a tabela **tWpłaty** – podrzędną. Złączenia (sprzężenia) przedstawione na rysunku 11.1 mają odpowiednio nazwy: 1 – **wewnętrzne (INNER JOIN)**, 2 – **zewnętrzne lewostronne (LEFT JOIN)**, 3 – **zewnętrzne prawostronne (RIGHT JOIN)**.



Rys. 11.1. Widok okna **Właściwości sprzężenia**

### 11.1. Złączenie wewnętrzne (INNER JOIN)

Jest to najpopularniejszy rodzaj sprzężenia. Przy **złączeniu wewnętrznym** łączą się rekordy obu tabel wtedy i tylko wtedy, gdy tabela

po lewej stronie ma odpowiadający jej rekord po prawej stronie. Składnię klauzuli **FROM** w przypadku sprzężenia wewnętrznego schematycznie możemy przedstawić następująco:

```
FROM tabela1 INNER JOIN tabela2  
ON tabela1. pole1 operator_porównania tabela2. pole2,
```

gdzie:

- **tabela1, tabela2** – nazwy tabel, z których pochodzą łączone rekordy;
- *pole1, pole2* – nazwy spręganych pól; jeśli nie są to pola numeryczne, muszą mieć ten sam typ danych, ale mogą mieć inne nazwy;
- *operator\_porównania* – dowolny relacyjny operator porównania: „=”, „<”, „>”, „<=”, „>=” lub „<>”.

Mimo możliwości używania różnych operatorów porównania najczęściej używamy „=”.

## Zadanie 11\_1

Utwórz instrukcję SQL o nazwie **sql11\_1** w celu wyświetlenia danych tabel **tOsoba** oraz **tZakwaterowanie** w postaci: *Nazwisko, Imiona, Album, Data\_zakwaterowania, Nr\_pokoju, Data\_wykwaterowania, Wydział*. Dane powinny dotyczyć osób z Łomży. Wynikowe rekordy należy posortować rosnąco według nazwiska, imienia, albumu oraz daty zakwaterowania.

Jeśli utworzymy instrukcję do realizacji sprzężenia wewnętrznego, to wyświetlimy dane tych osób, dla których utworzono rekordy w obu tabelach. Jeśli któraś z osób nie ma żadnego odpowiedniego rekordu w tabeli **tZakwaterowanie**, to jej dane nie będą wyświetlane.

Instrukcja **sql11\_1** zawierająca złączenie wewnętrzne ma postać:

```
SELECT tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album,  
tZakwaterowanie.Data_zakwaterowania,  
tZakwaterowanie.Nr_pokoju,  
tZakwaterowanie.Data_wykwaterowania,  
tZakwaterowanie.Wydział  
FROM tOsoba INNER JOIN tZakwaterowanie  
ON tOsoba.ID = tZakwaterowanie.ID  
WHERE (tOsoba.Miejscowość LIKE "Łomża")  
ORDER BY tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album,  
tZakwaterowanie.Data_zakwaterowania;
```

Warto zaznaczyć, że nie ma potrzeby pisania w poleceniu **SQL** nazw tabel w przypadku, kiedy nazwa pola występuje tylko w jednej z tabel. W takiej sytuacji system uzupełni polecenie automatycznie. Natomiast konieczne jest dodawanie nazwy tabeli do nazwy pola *ID*, ponieważ taka nazwa jest w obu tabelach. Zgodnie z tym instrukcję **sql11\_1** można zapisać następująco:

```
SELECT Nazwisko, Imiona, Album, Data_zakwaterowania,
        Nr_pokoju, Data_wykwaterowania, Wydział
FROM tOsoba INNER JOIN tZakwaterowanie
ON tOsoba.ID = tZakwaterowanie.ID
WHERE (Miejscowość LIKE "Łomża")
ORDER BY Nazwisko, Imiona, Album, Data_zakwaterowania;
```

Instrukcja **SELECT** pozwala na zrealizowanie sprzężenia wewnętrzne-go bez zastosowania operacji **INNER JOIN**. W tym celu należy wykonać zadanie **Z11\_1** inaczej:

```
SELECT tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album,
        tZakwaterowanie.Data_zakwaterowania,
        tZakwaterowanie.Nr_pokoju,
        tZakwaterowanie.Data_wykwaterowania,
        tZakwaterowanie.Wydział
FROM tOsoba, tZakwaterowanie
WHERE (tOsoba.ID = tZakwaterowanie.ID) AND
        (tOsoba.Miejscowość LIKE "Łomża")
ORDER BY tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album;
```

## 11.2. Złączenie zewnętrzne lewostronne (LEFT JOIN)

Terminu **złączenie lewostronne** używamy w celu uzyskania rekordów z lewej tabeli bez względu na to, czy odpowiednie rekordy w prawej tabeli istnieją. Aby utworzyć takie złączenie, postępujemy analogicznie do stoso-wania operatora **INNER JOIN**, lecz zamiast wyrazu **INNER** zapisujemy **LEFT**. Klauzula **FROM** przy sprzężeniu zewnętrznym lewostronnym ma postać:

```
FROM tabela1 LEFT JOIN tabela2
ON tabela1. pole1 operator_porównania tabela2. pole2
```

Jeśli mamy w tabeli **tOsoba** dane osób, dla których nie utworzono od-powiednich rekordów w tabeli **tZakwaterowanie**, to po wykonaniu nastę-pującego zadania na ekranie zostanie wyświetlona tabela zawierająca pu-

ste wartości w polach: *Data\_zakwaterowania*, *Nr\_pokoju*, *Data\_wykwaterowania*, *Wydział*.

## Zadanie 11\_2

Utwórz instrukcję **SQL** o nazwie **sql11\_2** w celu wyświetlenia pól: *Nazwisko*, *Imiona*, *Album* ze wszystkich rekordów tabeli **tOsoba** oraz odpowiednich danych z pól: *Data\_zakwaterowania*, *Nr\_pokoju*, *Data\_wykwaterowania*, *Wydział* tabeli **tZakwaterowanie**. Należy posortować dane malejąco według pola *Album* oraz rosnąco według pola *Data\_zakwaterowania*.

### Wykonanie

```
SELECT tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album,
tZakwaterowanie.Data_zakwaterowania, tZakwaterowanie.
Nr_pokoju, tZakwaterowanie.Data_wykwaterowania,
tZakwaterowanie.Wydział
FROM tOsoba LEFT JOIN tZakwaterowanie
ON tOsoba.ID = tZakwaterowanie.ID
ORDER BY tOsoba.Album DESC,
tZakwaterowanie.Data_zakwaterowania ASC;
```

### 11.3. Złączenie zewnętrzne prawostronne (RIGHT JOIN)

W przypadku gdy chcemy uzyskać rekordy z „prawej” tabeli bez względu na to, czy istnieje odpowiadający im rekord w „lewej” tabeli, stosujemy złączenie prawostronne. Złączenie prawostronne budujemy za pomocą klauzuli **FROM** w postaci:

```
FROM tabela1 RIGHT JOIN tabela2
ON tabela1.pole1 = tabela2.pole2
```

Instrukcja **SQL** do realizacji zadania **Z11\_2** z zastosowaniem operacji **RIGHT JOIN** ma postać:

```
SELECT tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album,
tZakwaterowanie.Data_zakwaterowania,
tZakwaterowanie.Nr_pokoju,
tZakwaterowanie.Data_wykwaterowania,
tZakwaterowanie.Wydział
FROM tZakwaterowanie RIGHT JOIN tOsoba
```

```
ON tOsoba.ID = tZakwaterowanie.ID
ORDER BY tOsoba.Album DESC,
tZakwaterowanie.Data_zakwaterowania ASC;
```

Operacja **RIGHT JOIN** może być bardzo przydatna, gdy chcemy połączyć dwie tabele wypełnione rekordami, definiując jednocześnie więzy integralności referencyjnej. Jeśli tabela podrzędna zawiera wartości klucza obcego, które nie występują w tabeli nadrzędnej, to próba utworzenia relacji skończy się niepowodzeniem. W tym przypadku należy wyświetlić takie rekordy w tabeli podrzędnej w celu ich dalszego usunięcia. Kolejne zadanie ilustruje takie czynności.

### Zadanie 11\_3

---

Utwórz kopię tabeli **tZakwaterowanie**, nadając jej nazwę **Kopia\_tZakwaterowanie**. Dodaj ręcznie jeden wiersz do tej tabeli, wpisując dowolne wartości do wszystkich pól. Do pola *ID* wprowadź wartość, która nie występuje wśród wartości pola *ID* tabeli **tOsoba**.

Utwórz instrukcję **SQL**, która wyświetli rekordy z tabeli **Kopia\_tZakwaterowanie**, które przeszkadzają w połączeniu tabel **tOsoba** oraz **Kopia\_tZakwaterowanie**.

---

### Wykonanie

W celu realizacji zadania utworzymy kolejno dwie instrukcje **SQL**. Pierwsza (o nazwie **sql11\_3a**) wyświetli pola *ID\_Zakwaterowanie* oraz *ID* ze wszystkich rekordów tabeli **Kopia\_tZakwaterowanie**. Jednocześnie zostaną wyświetcone odpowiednie wartości pola *ID* z tabeli **tOsoba**:

```
SELECT Kopia_tZakwaterowanie.ID_zakwaterowanie,
       Kopia_tZakwaterowanie.ID, tOsoba.ID
FROM tOsoba RIGHT JOIN Kopia_tZakwaterowanie
ON tOsoba.ID = Kopia_tZakwaterowanie.ID;
```

Druga instrukcja (o nazwie **sql11\_3b**) zostanie zbudowana na bazie pierwszej i będzie wyświetlała wartości kluczów rekordów tabeli **Kopia\_tZakwaterowanie**, które przeszkadzają w połączeniu tabel **tOsoba** oraz **Kopia\_tZakwaterowanie**.

```
SELECT Kopia_tZakwaterowanie.ID_zakwaterowanie,
       Kopia_tZakwaterowanie.ID
FROM sql11_3a WHERE tOsoba.ID IS Null;
```

## 11.4. Instrukcja SELECT zbudowana na trzech tabelach

Wyświetlenie danych z trzech połączonych ze sobą tabel realizuje się na tych samych zasadach jak w przypadku dwóch tabel. Zostanie to rozpatrzone na poniższym przykładzie.

### Zadanie 11\_4

Utwórz instrukcję SQL o nazwie **sql11\_4** do wyświetlenia danych z tabel **tOsoba**, **tZakwaterowanie** oraz **tKoszty\_zakwaterowania** w postaci: *Nazwisko, Imiona, Album, Data\_zakwaterowania, Od\_kiedy, Do\_kiedy, Kwota\_mies\_opłaty*. Należy wyświetlić wszystkie kiedykolwiek ustalone kwoty miesięcznych opłat studentów skierowanych do domu studenta przez Wydział Mechaniczny. Dane należy posortować rosnąco według nazwiska, imienia, numeru albumu.

### Wykonanie

Instrukcja **sql11\_4** może zostać zapisana w postaci:

```
SELECT tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album,
       tZakwaterowanie.Data_zakwaterowania,
       tKoszty_zakwaterowania.Od_kiedy,
       tKoszty_zakwaterowania.Do_kiedy,
       tKoszty_zakwaterowania.Kwota_mies_opłaty
  FROM (tOsoba LEFT JOIN tZakwaterowanie
        ON tOsoba.ID = tZakwaterowanie.ID)
    LEFT JOIN tKoszty_zakwaterowania
        ON tZakwaterowanie.ID_zakwaterowanie =
       tKoszty_zakwaterowania.ID_zakw
 ORDER BY tOsoba.Nazwisko, tOsoba.Imiona, tOsoba.Album;
```

## 11.5. Pobieranie danych z tabel połączonych za pomocą podzapytania

Podzapytanie umieszczamy w klauzuli **WHERE**, tworząc za jego pomocą złożone kryterium wyszukiwania danych. Często odwołuje się ono do tabeli innej niż ta, która jest wymieniona w klauzuli **FROM**. Zgodnie ze strukturalnym podejściem do rozwiązywania problemów z podzapytań

korzystamy w tych przypadkach, kiedy konieczne jest zagnieżdżanie instrukcji. Według tego podejścia dzielimy problem na podproblemy, rozwiązujemy te podproblemy, a następnie używając tych rozwiązań, kontrolujemy rozwiązanie całego problemu.

Jeżeli wynikiem instrukcji **SELECT-FROM-WHERE** jest jedna wartość, można ją ująć w nawiasy i traktować tak samo jak wartość stałą. Wtedy mamy podzapytanie, które umożliwia obliczenie wartości skalarnej.

## Zadanie 11\_5

---

Utwórz polecenie SQL o nazwie **sql11\_5** w celu wyświetlenia danych z tabeli **tPokoje** o pokoju, którego numer jest ulokowany w rekordzie tabeli **tZakwaterowanie** o znanym kluczu podstawowym (przykładowo numer *ID\_zakwaterowanie* równy jest 3).

---

### Wykonanie

Instrukcja **sql11\_5** ma postać:

```
SELECT Nr_pokoju, Piętro, Liczba_miejsc
FROM tPokoje
WHERE Nr_pokoju =
      (SELECT Nr_pokoju FROM tZakwaterowanie
       WHERE ID_zakwaterowanie = 3);
```

Podzapytanie można utworzyć za pomocą operatora **IN (NOT IN)**. Wtedy mamy podzapytanie, które określa listę wartości.

## Zadanie 11\_6

---

Utwórz polecenie SQL o nazwie **sql11\_6** w celu wyświetlenia nazwiska, imienia i numeru albumu każdego studenta, który mieszka od 1 października 2016 roku w pokoju o numerze 10 i jeszcze się nie wykwarterował.

---

### Wykonanie

Instrukcja **sql11\_6** ma postać:

```
SELECT Nazwisko, Imiona, Album
FROM tOSOBA
WHERE ID IN
      (SELECT ID FROM tZakwaterowanie
```

```
WHERE (Nr_pokoju = 101) AND (Data_zakwaterowania >= #2016-10-01#) AND (Data_wykwaterowania IS NULL));
```

Złączenie wewnętrzne można zbudować bez używania operacji **INNER JOIN**. Zamiast niej można skorzystać z operatorów **EXISTS** (istnieje) oraz **NOT EXISTS** (nie istnieje) sprawdzających, czy podzapytanie daje pusty zbiór wyników.

## Zadanie 11\_7

Utwórz instrukcję **SQL** o nazwie **sql11\_7** do wyświetlenia z tabeli **tOsoba** pól: *ID, Nazwisko, Imiona, Album* z rekordów dotyczących osób, które nie dokonały wpłat w październiku 2016 roku.

### Wykonanie

```
SELECT ID, Nazwisko, Imiona, Album
FROM tOsoba
WHERE NOT EXISTS (SELECT ID_wpłaty FROM tWpłaty
                   WHERE (tOsoba.ID = tWpłaty.ID) AND
                     (MONTH(Data_wpłaty) = 11) AND
                     (YEAR(Data_wpłaty) = 2016));
```

## 11.6. Dodawanie synonimów do nazw tabel

Dodając (za pomocą **AS**) synonim (**alias**) do nazwy tabeli w instrukcji **SELECT**, możemy znacznie skrócić jej tekst i poprawić czytelność.

## Zadanie 11\_8

Wykonaj **Zadanie 11\_1**, używając synonimów w celu skrócenia tekstu instrukcji **SQL**.

### Wykonanie

Na przykład, możemy zastosować synonim „o” dla nazwy „**tOsoba**” oraz synonim „z” dla nazwy „**tZakwaterowanie**”. Wówczas instrukcję **SQL** możemy zapisać w następującej postaci:

```
SELECT o.Nazwisko, o.Imiona, o.Album, z.Data_zakwaterowania,  
z.Nr_pokoju, z.Data_wykwaterowania, z.Wydział  
FROM tOsoba o INNER JOIN tZakwaterowanie z  
ON o.ID = z.ID  
WHERE o.Miejscowość LIKE "Łomża"  
ORDER BY o.Nazwisko, o.Imiona,  
o.Album, z.Data_zakwaterowania;
```

## Zadania do samodzielnego wykonania

### Zadanie 11\_9

Utwórz instrukcję **SQL** do wyświetlenia danych z tabel **tOsoba** oraz **tWpłaty** o wpłatach za ubiegły rok akademicki w postaci: *Nazwisko, Imiona, Album, Wpłata*. Dane powinny dotyczyć osób mających na imię „Jan” lub „Maria”. Dane należy posortować rosnąco według numeru *ID* i malejąco według pola *Data\_wpłaty*.

---

### Zadanie 11\_10

Utwórz instrukcję **SQL** do wyświetlenia danych w postaci: *Nazwisko, Imiona, PESEL, Data\_wpłaty, Za\_rok\_akad, Wpłata*, Dodatkowo o wpłatach za bieżący rok akademicki każdej osoby, której nazwisko zaczyna się od liter [D - F]. W polu o nazwie *Dodatkowo* powinna wyświetlać się wartość stanowiąca 1% wartości zawartej w polu *Wpłata*. Dane należy posortować rosnąco według pól *Data\_wpłaty, Nazwisko, Imię*.

---

### Zadanie 11\_11

Utwórz instrukcję **SQL** do wyświetlenia danych z tabel **tPokoje** oraz **tZakwaterowanie** w postaci: *Piętro, Nr\_pokoju, Liczba\_miejsc, ID, Data\_zakwaterowania, Data\_wykwaterowania, Wydział*. Dane powinny dotyczyć zakwaterowania w 2017 roku.

---

### Zadanie 11\_12

Utwórz instrukcję **SQL** w celu wyświetlenia nazwiska, imienia i numeru albumu każdego studenta, który wykwaterował się w ubiegłym roku akademickim.

---

## Zadanie 11\_13

Utwórz instrukcję **SQL**, używając polecenia **EXISTS** do wyświetlenia nazwiska, imienia i numeru PESEL studentów, którzy zakwaterowali się w Domu Studenta w październiku 2016 roku.

# 12. Instrukcje SQL do agregowania danych

## 12.1. Przykłady zastosowania funkcji agregujących

Przez agregowanie należy rozumieć takie działanie, które z listy wartości zapisanych w kolumnie tworzy jedną wartość. Zastosowanie funkcji agregujących w kwerendzie zbudowanej za pomocą narzędzi graficznych rozpatrywano w rozdziale 8. Funkcje te posiadały nazwy polskie: **Suma**, **Średnia**, **Policz**,.... W instrukcjach **SQL** używamy tych samych funkcji, jednak ich nazwy są skrótami słów w języku angielskim. Najczęściej używane funkcje agregujące to:

- **SUM** (Suma),
- **AVG** (Średnia),
- **MIN** (Minimum),
- **MAX** (Maksimum),
- **COUNT** (Policz),
- **FIRST** (Pierwszy),
- **LAST** (Ostatni).

**Zadanie 12\_1** jest przykładem zastosowania funkcji **SUM** (Suma) w zapytaniu **SQL** opartym o jedną tabelę.

## Zadanie 12\_1

Utwórz instrukcję **SQL** o nazwie **sql12\_1** do wyświetlenia sumy wszystkich wpłat osoby o numerze *ID* równym 5 za rok akademicki 2016/2017.

### Wykonanie

```
SELECT SUM(Wpłata) AS [Suma wpłat ID = 5 rok - 2016/2017]
```

```
FROM tWpłaty
WHERE (ID = 5) AND (Za_rok_akad = 2016);
```

Przykładowy wynik działania kwerendy przedstawiono na rysunku 12.1.

	Suma wpłat ID = 5 rok - 20016/2017	
	600,00 zł	

Rys. 12.1. Widok okna z wynikiem wykonania instrukcji **sql12\_1**

Jeżeli kryterium wykonania instrukcji **SQL** zależy od pól ulokowanych w dwóch różnych tabelach (które są połączone bądź mogą być połączone na skutek istniejącego w jednej z nich klucza obcego), to należy zbudować polecenie w oparciu o dwie tabele.

**Zadanie 12\_2** jest przykładem zastosowania funkcji **SUM** (Suma) w zapytaniu **SQL** opartym o dwie tabele.

## Zadanie 12\_2

---

Utwórz instrukcję **SQL** o nazwie **sql12\_2** do wyświetlenia sumy wszystkich wpłat osoby o numerze albumu równym 56567.

---

### Wykonanie

```
SELECT SUM (Wpłata) AS [Suma wpłat]
FROM tOsoba INNER JOIN tWpłaty
ON tOsoba.ID = tWpłaty.ID
WHERE Album = 56567;
```

## Zadanie 12\_3

---

Utwórz instrukcję **SQL** o nazwie **sql12\_3** do wyświetlenia średniej wartości kwot wpłaconych w 2016 roku.

---

### Wykonanie

```
SELECT AVG(Wpłata) AS [Średnia wartość wpłat]
FROM tWpłaty
WHERE YEAR(Data_wpłaty) = 2016;
```

## Zadanie 12\_4

Utwórz instrukcję **SQL** o nazwie **sql12\_4** w celu wyświetlenia minimalnej wartości numeru albumu w tabeli **tOsoba**, w rekordach dotyczących osób zamieszkałych w województwie podlaskim.

### Wykonanie

W celu wykonania **zadania 12\_4** użyj instrukcji:

```
SELECT MIN(Album) AS [Minimalna wartość numeru albumu]
FROM tOsoba
WHERE Województwo Like "podlaskie";
```

## Zadanie 12\_5

Utwórz instrukcję **SQL** o nazwie **sql12\_5** do wyświetlenia maksymalnej wartości kwot wpłaconych przez osoby urodzone w 1996 roku.

### Wykonanie

Zapytanie **SQL** może być zapisane następująco:

```
SELECT MAX(Wpłata) AS [Maksymalna wartość wpłaty]
FROM tOsoba INNER JOIN tWpłaty
ON tOsoba.ID = tWpłaty.ID
WHERE tOsoba.PESEL like "96*";
```

## Zadanie 12\_6

Utwórz instrukcję **SQL** o nazwie **sql12\_6** do wyliczenia liczby rekordów tabeli **tOsoba** zawierających dane o kobietach.

### Wykonanie

Instrukcja **SQL** do wyliczenia liczby wierszy tabeli powinna zawierać funkcję **COUNT**. Tę funkcję można zapisać dwójako: 1) wpisując jako argument symbol wieloznaczny „\*” lub 2) jako nazwę pola (najlepiej wybrać pole klucza podstawowego).

## Pierwszy sposób

```
SELECT COUNT(*) AS [Liczba rekordów]
FROM tOsoba
WHERE Płeć LIKE "Kobieta";
```

## Drugi sposób

```
SELECT COUNT(ID) AS [Liczba rekordów]
FROM tOsoba
WHERE Płeć LIKE "Kobieta";
```

## 12.2. Grupowanie

We wszystkich przedstawionych powyżej przykładach w wyniku wykonania instrukcji **SQL** otrzymano jedną wartość, ponieważ zapytanie dotyczyło jakiejś jednej grupy rekordów. Mogły to być rekordy jednej lub dwóch tabel, ale zapytanie odnosiło się do wszystkich wierszy, które wypełniały kryterium zapytania. Jednak często zachodzi konieczność jednoczesnego uzyskania wielu wynikowych wartości dla wielu grup rekordów. W tym celu stosuje się instrukcję **SELECT** z klauzulą **GROUP BY**, którą umieszczaamy od razu po klauzuli **WHERE**. Po słowie kluczowym **GROUP BY** następuje lista atrybutów grupujących.

Na przykład, jeżeli potrzebujemy informacji o liczbie danych w tabeli **tOsoba** o osobach każdej płci, to aby to policzyć, trzeba wiersze tabeli **tOsoba** pogrupować według pola **Płeć**. W wyniku tego działania powinna zostać wyświetlona wynikowa tabela dwukolumnowa. Pierwsza kolumna będzie zawierać nazwy określające płeć, druga – liczby osób.

## Zadanie 12\_7

---

Utwórz instrukcję **SQL** o nazwie **sql12\_7** do wyliczenia liczby rekordów w tabeli **tOsoba** zawierających dane o kobietach oraz liczby rekordów zawierających dane o mężczyznach.

---

## Wykonanie

```
SELECT Płeć, Count(ID) AS [Liczba osób danej płci]
FROM tOsoba
GROUP BY Płeć;
```

W wyniku wykonania instrukcji na ekranie zostanie wyświetcone okno podobne do przedstawionego na rysunku 12.2.

Płeć	Liczba osób danej płci
Kobieta	3
Mężczyzna	5

Rys. 12.2. Widok okna z wynikiem wykonania instrukcji **sql12\_7**

## Zadanie 12\_8

Utwórz instrukcję SQL o nazwie **sql12\_8** do wyświetlenia informacji o liczbie osób z każdego województwa zakwaterowanych w Domu Studenta w dniu dzisiejszym. Nazwy województw umieść alfabetycznie. Do wyznaczenia daty z dnia dzisiejszego należy skorzystać z funkcji wbudowanej **DATE()**.

### Wykonanie

Instrukcję **sql12\_8** można zapisać w postaci:

```
SELECT tOsoba.Województwo, COUNT(*) AS [Liczba studentów]
FROM tOsoba INNER JOIN tZakwaterowanie
ON tZakwaterowanie.ID = tOsoba.ID
WHERE tZakwaterowanie.Data_zakwaterowania = DATE()
GROUP BY tOsoba.Województwo
ORDER BY tOsoba.Województwo;
```

## Zadanie 12\_9

Utwórz instrukcję SQL o nazwie **sql12\_9** do wyświetlenia liczby osób z każdego województwa, które kiedykolwiek zakwaterowały się w Domu Studenta. Nazwy województw ulokuj alfabetycznie.

### Wykonanie

Wykonując dane zadanie, weźmiemy pod uwagę możliwość wielokrotnego zakwaterowania tej samej osoby. Zatem należy najpierw utworzyć instrukcję do zdefiniowania numerów osób, które kiedykolwiek zakwaterowały w Domu Studenta. Odpowiednia instrukcja o nazwie **sql12\_9a** będzie miała postać:

```
SELECT DISTINCT ID AS [ID_osoby]
FROM tZakwaterowanie;
```

Teraz utworzymy zapytanie **SQL** o nazwie **sql12\_9** oparte na tabeli **tOsoba** oraz instrukcji **sql12\_9a**:

```
SELECT tOsoba.Miejscowość, Count(tOsoba.ID) AS [Liczba osób]
FROM tOsoba INNER JOIN sql12_9a
ON tOsoba.ID = sql12_9a.ID_osoby
GROUP BY tOsoba.Miejscowość;
```

### *Klauzula HAVING*

Instrukcja **SELECT**, zawierająca klauzulę **GROUP BY**, może również zawierać klauzulę **HAVING**. Za pomocą klauzuli **HAVING** określa się warunki, które muszą być spełnione przez grupy wymienione w klauzuli **GROUP BY**, tak jak za pomocą klauzuli **WHERE** określa się warunki, które muszą spełniać wiersze wymienione w klauzuli **SELECT**. Przy tym należy umieszczać w klauzuli **HAVING** funkcję agregującą.

## Zadanie 12\_10

---

Utwórz instrukcję **SQL** o nazwie **sql12\_10** w celu wyświetlenia informacji z tabeli **tWpłaty** o sumie wpłat każdej osoby za zamieszkanie w roku akademickim 2016/2017. Przy tym należy wyświetlić tylko takie kwoty, które są mniejsze od 1000 zł. Osoba identyfikuje się swoim numerem *ID*. Numery *ID* należy posortować malejaco.

---

### Wykonanie

Instrukcja do **zadania 12\_10** będzie miała postać:

```
SELECT ID, SUM (Wpłata) AS [SUMA WPŁAT]
FROM tWpłaty
WHERE Za_rok_akad = 2016
GROUP BY ID
HAVING SUM(Wpłata) < 1000
ORDER BY ID DESC;
```

## Zadania do samodzielnego wykonania

### Zadanie 12\_11

Utwórz instrukcję **SQL** do wyliczenia liczby miejsc we wszystkich pokojach Domu Studenta ulokowanych na trzecim oraz piątym piętrze.

### Zadanie 12\_12

Utwórz instrukcję **SQL** do wyliczenia liczby wierszy w tabeli **tZakwaterowanie**, dotyczących zakwaterowania w latach 2014 oraz 2016 osób skierowanych przez Wydział Mechaniczny.

### Zadanie 12\_13

Utwórz instrukcję **SQL** do wyliczenia minimalnej wartości wpłat dokonanych w miesiącu kwietniu lub czerwcu.

### Zadanie 12\_14

Utwórz instrukcję **SQL** do wyliczenia liczby studentów zamieszkałych na stale w Białowieży, którzy mają numery albumu z zakresu <65555; 66000>.

### Zadanie 12\_15

Utwórz polecenie **SQL** w celu wyświetlenia sum wszystkich wpłat dokonanych przez studenta o założonym numerze *ID* za każdy rok akademicki.

### Zadanie 12\_16

Utwórz polecenie **SQL** w celu wyświetlenia liczby zakwaterowań w Domu Studenta w każdym roku.

## Zadanie 12\_17

Utwórz polecenie **SQL** w celu wyświetlenia informacji ile w sumie jest miejsc do zamieszkania na każdym piętrze Domu Studenta.

---

## Zadanie 12\_18

Utwórz polecenie **SQL** w celu wyświetlenia średniej wartości kwot wpłaconych za zamieszkanie w Domu Studenta dla każdego roku wymienionego w kolumnie *Data\_wpłaty*. Skorzystaj z funkcji **Year**.

---

## Zadanie 12\_19

Utwórz instrukcję **SQL** w celu wyświetlenia sumarycznej wartości kwot za zamieszkanie w Domu Studenta wpłaconych w każdym miesiącu 2017 roku. Dane należy pogrupować według miesiąca. Skorzystaj z funkcji **Month**.

---

# 13. Zapytania SQL do wprowadzenia, aktualizacji, usunięcia danych

## 13.1. Wprowadzenie danych do tabeli

Jeśli chcemy dodać do tabeli nowy rekord, należy stosować instrukcję **INSERT**, która ma postać:

```
INSERT INTO tabela_docelowa (pole1, pole2,...)  
VALUES (wartość1, wartość2,...);
```

Tu *pole1, pole2,...* – nazwy pól, do których mają być dołączone dane, a *wartość1, wartość2,...* – wartości, które mają być wstawione w polach nowego rekordu. Każda wartość jest wstawiana w polu odpowiadającym pozycji wartości na liście: *wartość1* jest wstawiana w *polu1* nowego rekordu, *wartość2* — w *polu2* itd. Wartości tekstowe muszą być ujęte w znaki cudzysłowu, a wartości typu **Data/Godzina** powinny być ujęte w znaki #.

W instrukcji **INSERT** możemy nie wymieniać nazw pól po nazwie tabeli docelowej. W tym przypadku po słowie kluczowym **VALUES** należy podać wartości do wszystkich pól tabeli. Taka instrukcja ma postać:

```
INSERT INTO tabela_docelowa  
VALUES (wartość1, wartość2,...);
```

Wartości wprowadzane nie powinny naruszać więzów integralności. Zatem przed utworzeniem instrukcji **INSERT**, która będzie zawierała konkretne wartości, należy obejrzeć projekt tabeli, aby sprawdzić, jakie pola są wymagane, czy jest tam pole typu **Autonumerowanie**, czy są klucze obce.

Do pola typu **Autonumerowanie** nie wprowadzamy żadnych wartości, skoro zrobi to za nas program.

Jeśli pole klucza obcego jest wymagane, to powinno ono zawierać wartość pochodząca z dziedziny klucza podstawowego połączonej tabeli.

## Zadanie 13\_1

Utwórz instrukcję **SQL** o nazwie **sql13\_1** w celu wprowadzenia jednego rekordu danych do tabeli **tKoszty\_zakwaterowania**.

Pole *ID\_historia\_opłat* ma typ **Autonumerowanie**. Do takiego pola nie wprowadzamy żadnych wartości, skoro zrobi to za nas program.

W tabeli **tKoszty\_zakwaterowania** wszystkie pola są wymagane. Pole *ID\_zakw* jest kluczem obcym. Dlatego nową wartością w tym polu może być tylko jedna z wartości obecnych w polu *ID\_zakwaterowanie* tabeli **tZakwaterowanie**. Każda innna wartość będzie odrzucana przez program i wykonanie instrukcji skończy się niepowodzeniem.

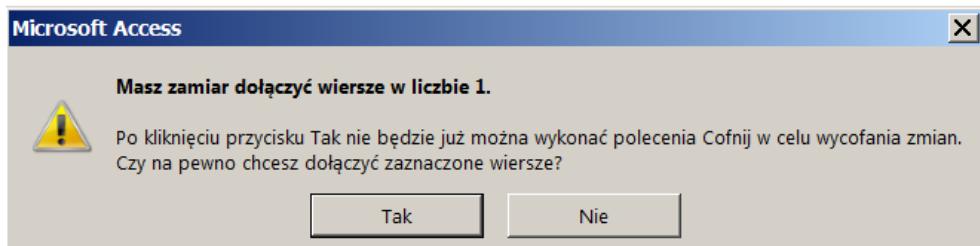
Przykładowa instrukcja **sql13\_1**:

```
INSERT INTO tKoszty_zakwaterowania  
        (ID_zakw, Od_kiedy, Do_kiedy, Kwota_mies_opłaty)  
VALUES (6, #2016-10-01#, #2017-06-30#, 280);
```

Tu wartość liczbową 6 to jedna z wartości pola *ID\_zakwaterowanie* tabeli **tZakwaterowanie**.

Po uruchomieniu tego polecenia na ekranie pojawi się okienko dialogowe (rys. 13.1), które wymaga potwierdzenia lub odwołania wykonania operacji. Podobne okienko pojawia się również w trakcie wykonywania kwerendy dołączającej przy pracy w środowisku graficznym (rozdział 9).

Po wykonaniu polecenia **SQL** otwórz tabelę **tKoszty\_zakwaterowania** i sprawdź, czy w nowy wiersz został dodany.



Rys. 13.1. Okno dialogowe przy próbie dołączenia rekordu

## 13.2. Aktualizacja danych w tabeli

Instrukcja **UPDATE** służy do modyfikacji (aktualizacji) danych: zmienia wartości w zadanych polach określonej tabeli na podstawie wyznaczonych kryteriów.

Schemat instrukcji **UPDATE**:

```
UPDATE tabela_docelowa
SET pole1 = wyrażenie1, pole2 = wyrażenie2, ...
WHERE kryterium;
```

W tabeli docelowej będą zmodyfikowane wszystkie wiersze spełniające podany warunek. Modyfikacja polega na zastosowaniu instrukcji przypisania *kolumna = wyrażenie* do każdej kolumny, której nazwa znajduje się po prawej stronie równości w klauzuli **SET**.

Klauzula **WHERE** jest opcjonalna (nie musi wystąpić). W razie jej braku będą zaktualizowane wszystkie rekordy tabeli.

### Zadanie 13\_2

Utwórz instrukcję **SQL** o nazwie **sql13\_2** do aktualizacji danych osoby o numerze albumu równym 65555: nowe nazwisko to „Głowacka”, nowy adres e-mail – dana.glowacka@wp.pl.

### Wykonanie

W celu realizacji **zadania 13\_2** zapisz polecenie:

```
UPDATE tOsoba
SET Nazwisko = 'Głowacka', [e-mail] = 'dana.glowacka@wp.pl'
WHERE Album = 65555;
```

Tu nazwa pola *e-mail* jest ujęta w nawiasy kwadratowe, ponieważ nazwa pola zawiera niedozwolony w nazwach identyfikatorów symbol „-” (łącznik).

## Zadanie 13\_3

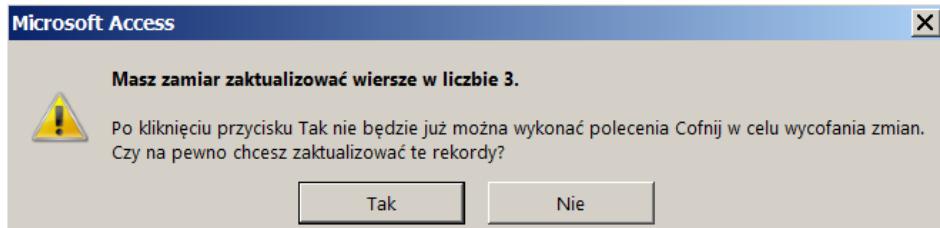
Zapisz instrukcję **SQL** o nazwie **sql13\_3** do aktualizacji danych w tabeli **tPokoje**. Zmniejsz liczbę miejsc do zakwaterowania o jedno w pokojach o numerach 2, 4 oraz 6.

### Wykonanie

Instrukcję **SQL** można zapisać w postaci:

```
UPDATE tPokoje SET Liczba_miejsc = Liczba_miejsc -1  
WHERE (Nr_pokoju = 2) OR (Nr_pokoju = 4) OR (Nr_pokoju = 6);
```

Po uruchomieniu tego polecenia na ekranie pojawi się okienko dialogowe podobne do przedstawionego na rysunku 13.2, które wymaga potwierdzenia lub odwołania aktualizacji danych.



Rys. 13.2. Okno dialogowe przy próbie dołączenia rekordu

### 13.3. Usunięcie danych z tabeli

Instrukcja **DELETE** służy do usuwania wierszy z tabeli. Poniżej przedstawiono składnię instrukcji:

```
DELETE FROM tabela  
WHERE kryterium;
```

Warto tu zauważyć, że jeśli zapomnimy dodać kryterium, to zostaną usunięte wszystkie wierszy tabeli.

Jeśli próbujemy usunąć rekordy z tabeli nadrzędnej i zdefiniowane są więzy integralności oraz kaskadowe usunięcie rekordów, to razem z rekordem w nadrzędnej tabeli zostaną usunięte wszystkie połączone rekordy w tabelach pokrewnych. Jeżeli kaskadowe usunięcie rekordów nie było zdefiniowane, to system nie pozwoli na usunięcie rekordu z tabeli nadrzędnej.

## Zadanie 13\_4

---

Zapisz instrukcję **SQL** o nazwie **sql13\_4** do usunięcia danych z tabeli **tOsoba** o osobie, której numer *ID* jest równy 5.

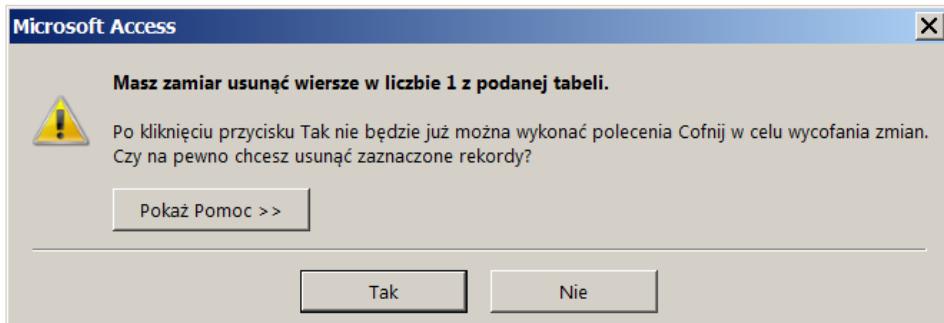
---

### Wykonanie

Instrukcja **SQL** ma postać:

```
DELETE FROM tOsoba  
WHERE ID = 5;
```

Po uruchomieniu kwerendy na ekranie będzie wyświetlane okienko (rys. 13.3) do potwierdzenia lub odwołania usunięcia danych.



Rys. 13.3. Okno dialogowe przy próbie dołączenia rekordu

W tym przypadku oprócz rekordu w tabeli **tOsoba** zostaną usunięte odpowiednie rekordy z tabel **tWpłaty**, **tZakwaterowanie** oraz **tKoszty\_zakwaterowania**.

Jak już wspomniano w rozdziale 10, zapytanie **SQL** może zawierać parametry. Kolejne zadanie przeznaczone jest do utworzenia parametrycznego zapytania **SQL** do usunięcia rekordu w tabeli.

## Zadanie 13\_5

Zapisz instrukcję **SQL** o nazwie **sql13\_5** do usunięcia danych z tabeli **tOsoba** o osobie, której numer *ID* jest podawany przez użytkownika podczas wykonania kwerendy.

### Wykonanie

Oto przykładowa instrukcja zawierająca parametr, która pozwoli podawać numer *ID* podczas jej wykonania:

```
DELETE FROM tOsoba  
WHERE ID = [Podaj numer ID osoby];
```

## Zadania do samodzielnego wykonania

### Zadanie 13\_6

Utwórz instrukcję **SQL** w celu wprowadzenia jednego rekordu dowolnych danych do tabeli **tWpłaty**.

### Zadanie 13\_7

Zapisz instrukcję **SQL** do aktualizacji danych w tabeli **tKoszty\_zakwaterowania**. Zwięksź o 10 % wartość w polu *Kwota\_mies\_opłaty*, w rekordzie, którego wartość *ID\_historia\_oplat* jest równa 5.

### Zadanie 13\_8

Zapisz instrukcję **SQL** do aktualizacji danych w tabeli **tZakwaterowanie**. Wprowadź wartość daty wykwaterowania do jednego z rekordów tabeli, którego wartość *ID\_zakwaterowanie* jest podawana przez użytkownika w trakcie wykonania instrukcji.

## IV. Wprowadzenie do formularzy, raportów i makr

Klasyczne pojęcie relacyjnej bazy danych – to zbiór połączonych ze sobą tabel zbudowanych zgodnie z założeniami modelu relacyjnego. Współczesna baza danych zawsze funkcjonuje w środowisku systemu zarządzania bazami danych i dysponuje o wiele szerszymi możliwościami niż tylko przechowywanie tabel. Zazwyczaj struktura bazy danych jest skomplikowana. Przykładowo do składu bazy danych zawsze wchodzą kwerendy.

Z bazą danych współpracuje aplikacja (lub aplikacje). Taka aplikacja obsługująca bazę danych jest związana z nią nieroziędznie. Do utworzenia aplikacji obsługującej bazę danych możemy stosować różnorodne oprogramowanie.

**Microsoft Access** pozwala szybko utworzyć aplikację bazodanową, ponieważ pozwala obok utworzonych tabel budować na ich podstawie kwerendy, formularze, raporty, makra, korzystając z narzędzi graficznych, w tym z licznych konstruktorów. Dopiero w niestandardowych sytuacjach niezbędne jest napisanie kodu. W tym celu można korzystać z wbudowanego języka *Visual Basic for Applications (VBA)*.

W tej części przedstawiono zarys możliwości narzędzi programu *Microsoft Access 2016*, pozwalających na wykonanie aplikacji bazodanowej bez zastosowania języka VBA.

### 14. Formularze

Formularz jest podstawowym obiektem interfejsu graficznego użytkownika bazy danych. Składa się on ze zbioru formantów, takich jak: pola do wyświetlenia i wprowadzenia danych, listy rozwijane, pola wyboru, teksty, rysunki, wykresy, przyciski poleceń i inne. Formularz może zawierać również podformularze. Każdy podformularz i cały formularz oparte są na tabeli lub kwerendzie.

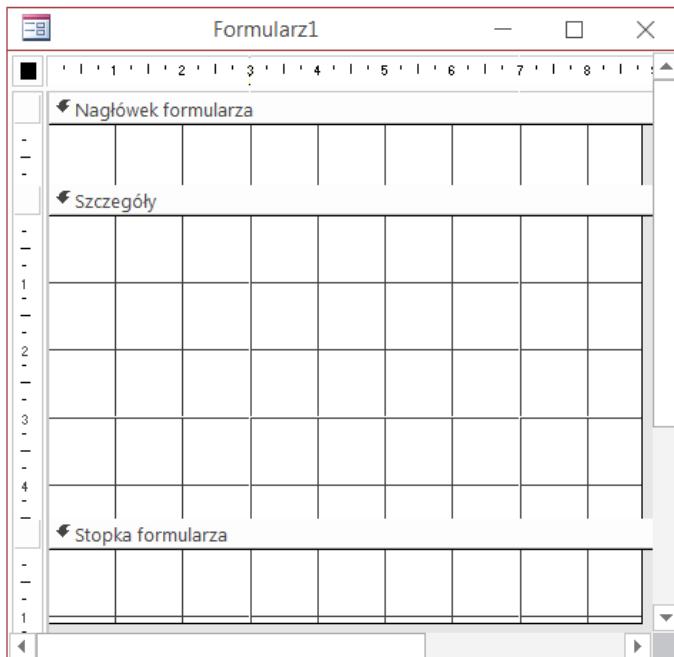
W poprzednich rozdziałach tworzyliśmy formularze, korzystając z kreatorów programu *Microsoft Access*. W tym rozdziale nauczymy się wprowadzać zmiany do gotowych formularzy oraz projektować własne.

## 14.1. Struktura formularza w Widoku projektu

**Widok projektu** formularza przedstawia szczegółową strukturę formularza. Widok ten pozwala wykorzystać w pełni wszystkie narzędzia budowania formularzy programu *Microsoft Access*.

Generalnie projekt formularza składa się z trzech części: 1) nagłówka, 2) szczegółów i 3) stopki. Sekcja szczegółów najczęściej służy do wyświetlania, wprowadzania i modyfikacji danych.

Na rysunku 14.1 przedstawiono projekt pustego formularza.



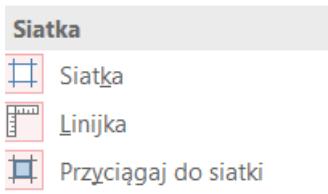
Rys. 14.1. Pusty formularz w **Widoku projektu**

Gdy na ekranie jest wyświetlony formularz w widoku projektu, to najczęściej używane są polecenia z zakładki **Projektowanie**. Jednocześnie bardzo pomocne są polecenia zakładki **Rozmieszczanie**. Na rysunku 14.1 przykładowo widzimy w formularzu siatkę i dwie linijki. Możemy je usunąć

lub ustawić za pomocą przycisków wstążki umieszczonych na karcie **Rozmieszczenie**. Wywołujemy te polecenia, wybierając ikonkę



**Rozmiar/odstęp**, a następnie odpowiednie polecenie z menu, które zostanie wyświetcone. Fragment tego menu przedstawiono na rysunku 14.2. Zatem, jeśli nie chcemy, aby w projekcie formularza była widoczna siatka, należy kliknąć lewym wskaźnikiem myszy na słowie **Siatka**. Gdy powtórzymy kliknięcie, siatka w projekcie formularza pojawi się ponownie.



Rys. 14.2. Fragment menu ikonki **Rozmiar/odstęp**

Każda sekcję projektu, tzn. **Nagłówek formularza**, **Szczegóły**, **Stopkę formularza**, możemy edytować osobno. Aby zaznaczyć sekcję, należy kliknąć wskaźnikiem myszy na odpowiednim pasku. Każda sekcja ma swoje menu podrzędne. Za jego pomocą możemy wprowadzić zmiany do widoku każdej z sekcji.

Aby zapoznać się ze strukturą formularza, utwórz najprostszy formularz, wykonując kolejne zadanie.

## Zadanie 14\_1

Utwórz formularz kolumnowy o nazwie **f14\_1**, przeznaczony do przeglądu i edycji danych tabeli **tOsoba**. Skorzystaj z kreatora formularzy, wybierz pola: *ID*, *Nazwisko*, *Imiona*, *Album*, *Płeć*, *PESEL*. Po utworzeniu formularza przełącz się na **Widok projektu**. Wyłącz pokazywanie siatki, a także zmień kolor tła nagłówka formularza.

## Wykonanie

Zastosuj **Kreatora formularzy**, działając, jak to opisano w poprzednich rozdziałach. Po zakończeniu pracy z konstruktorem na ekranie zostanie wyświetlony wygenerowany formularz w **Widoku formularza** (rys. 14.3).

f14\_1

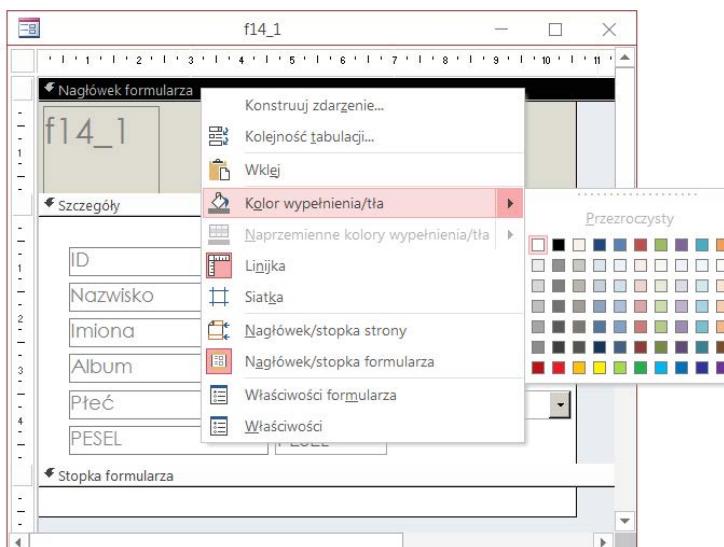
ID	1
Nazwisko	Kulesza
Imiona	Paweł
Album	34567
Płeć	Mężczyzna
PESEL	9011221234

Rekord: ▶◀ 1 z 9 ▶▶ Bez filtra Wyszukaj

Rys. 14.3. Wygenerowany formularz f14\_1 w Widoku formularza

Przełącz się na **Widok projektu**. Wybierz na zakładce **Rozmieszczenie ikonki Rozmiar/odstęp**, a potem polecenie **Siatka**.

Kliknij prawym przyciskiem myszy na pasku **Nagłówek** formularza i wybierz z menu podręcznego (rys. 14.4) polecenie **Kolor wypełnienia tła** i konkretny kolor (na rys. 14.3 wybrano kolor biały.) Zapisz formularz, i w celu jego przejrzenia ponownie przejdź do **Widoku formularza**.



Rys. 14.4. Menu podręczne sekcji Nagłówek formularza

**Nagłówek i stopkę formularza** można usunąć z projektu, a potem dodać ponownie. Kolejny przykład ilustruje te możliwości.

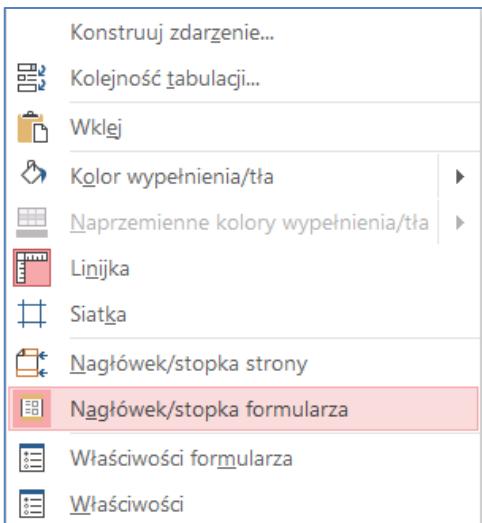
## Zadanie 14\_2

Usuń nagłówek i stopkę z projektu formularza **f14\_1**, a potem ponownie go dodaj. Wprowadź do nagłówka formularza tekst: „Dane osobowe”. Zmień kolor i rozmiar czcionki tekstu.

### Wykonanie

Kliknij prawym przyciskiem myszy na dowolnym pustym obszarze projektu formularza i w menu podręcznym wybierz polecenie **Nagłówek/Stopka formularza** (rys. 14.4). Nagłówek i stopka formularza znikną z projektu.

Ponownie wywołaj menu podręczne formularza i ponownie wybierz polecenie **Nagłówek/Stopka formularza** (rys. 14.5) – nagłówek i stopka zostaną ponownie widoczne w projekcie formularza, jednak poprzedni tekst w nagłówku zniknie.



Rys. 14.5. Menu podręczne sekcji **Nagłówek formularza**

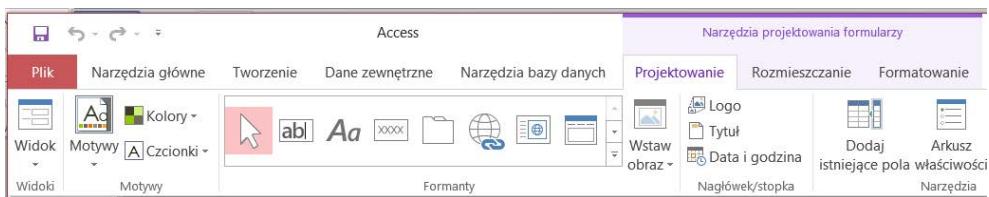
## Zadanie 14\_3

Utwórz w formularzu **f14\_1** tytuł: „Dane osobowe”.

## Wykonanie

Po usunięciu nagłówka formularza **f14\_1** jednocześnie został usunięty jego tytuł. Aby utworzyć tytuł, należy umieścić w nagłówku formularza formant o nazwie **Etykieta**. Wybierz zatem w zakładce **Projektowanie**

ikonkę **Aa** (rys. 14.6). Kliknij na niej lewym wskaźnikiem myszy, a potem w obrębie nagłówka formularza określ wskaźnikiem myszy prostokąt w celu oznaczenia miejsca dla tekstu. Dalej już tylko wprowadź tekst, jak pokazano na rysunku 14.7.



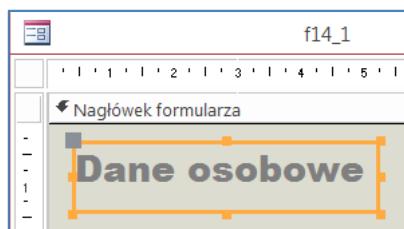
Rys. 14.6. Widok paska narzędzi projektowania formularzy



Rys. 14.7. Wprowadzenie tekstu do etykiety w nagłówku formularza

Zawsze, gdy chcemy zmienić postać etykiety, należy najpierw ją zaznaczyć: kliknąć na jej ramce. Na ramce zaznaczonej etykiety pojawią się kwadraciki. Teraz można zmienić kolor i rozmiar czcionki, korzystając z przycisków umieszczonych w zakładce **Narzędzia główne**.

Na rys. 14.8 widać zaznaczoną etyktę w momencie zmiany czcionki.



Rys. 14.8. Zaznaczona etykieta w nagłówku formularza

Lewy górny kwadracik na ramce pozwala przeciągnąć etykietę do innego miejsca w nagłówku, a pozostałe kwadraciki służą do zmiany rozmiaru etykiety. Ta sama zasada obowiązuje w odniesieniu do pozostałych formantów przedstawionych na wstążce.

## 14.2. Kontrolki (formanty) oraz ich arkusze właściwości

Obiekty umieszczane w formularzu nazywane są **kontrolkami** (lub **formantami**). Służą do wyświetlania danych i wykonywania akcji, a także zwiększały funkcjonalność interfejsu użytkownika.

Zastosowany w nagłówku formant **Etykieta** jest przykładem **kontrolki niepowiązanej**, tzn. takiej, do której nie przypisano źródła danych (jak pole lub wyrażenie). Kontrolek niepowiązanych można używać do wyświetlania informacji, linii, prostokątów i obrazów.

W sekcji **Szczegóły** widzimy ramki z nazwami pól tabeli. Są to **kontrolki powiązane**. Kontrolka powiązana służy do wyświetlania wartości pól bazy danych. Takimi wartościami mogą być dane tekstowe, daty, liczby, wartości **Tak/Nie**, obrazy lub wykresy.

W formularzu **Z14\_1** w sekcji **Szczegóły** powstały formanty powiązane, dla których źródłem danych są pola tabeli **tOsoba**. Jako przykład na rysunku 14.9 przedstawiono formant powiązany, służący do pokazywania wartości w polu *Nazwisko*.



Rys. 14.9. Przykład kontrolki powiązanej

Każdy formant powiązany składa się z dwóch części. Pierwsza zawsze jest etykietą, druga zawiera dane. W danym formularzu większość elementów powiązanych to **Pola tekstowe**. Dowiadujemy się o tym z **Arkusza właściwości** formantu. Przedstawiony na rys. 14.9 formant także jest polem tekstem.

Każda kontrolka, każda sekcja, cały formularz rozważane są przez program jako obiekty bazy danych. Każdy taki obiekt ma swoje odmienne właściwości.

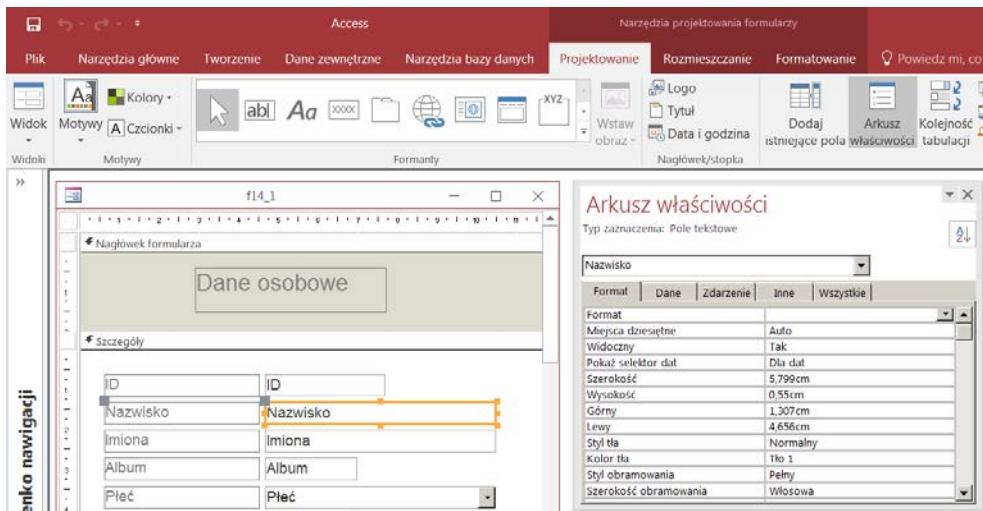
Aby obejrzeć listę właściwości dowolnego obiektu, wywołaj jego menu podrzczne (klikając na nim prawym przyciskiem myszy) i wybierz polecenie **Właściwości**. Można również otworzyć **Arkusz właściwości** dowolnego



Arkusz

obiektu, jeśli po jego zaznaczeniu wybierze się ikonkę **właściwości** na wstążce (rys. 14.6) (w zakładce **Projektowanie**, w grupie **Narzędzia**). Na rysunku

14.10 przedstawiono widok formularza **f14\_1** w **Widoku projektu** oraz **Arkusz właściwości** zaznaczonego formantu **Nazwisko**.



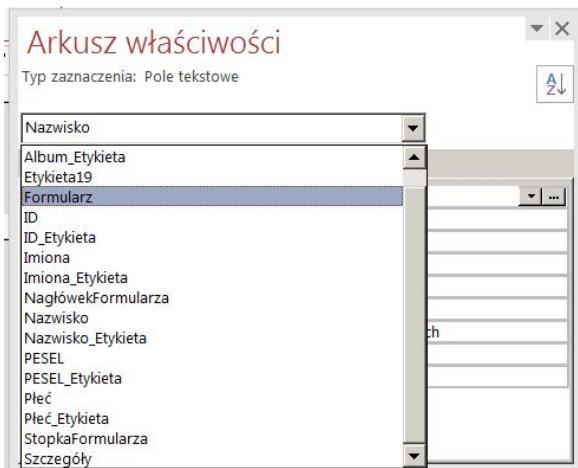
Rys. 14.10. Widok projektu formularza **f14\_1** oraz **Arkusza właściwości** pola tekstowego **Nazwisko**

Dla porównania na rysunku 14.11 przedstawiono **Arkusz właściwości nagłówka formularza** (wywoływany przez kliknięcie na pasku **Nagłówek formularza**).



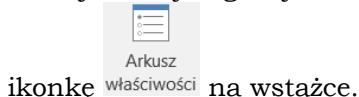
Rys. 14.11. Widok **Arkusza właściwości** nagłówka formularza

Możemy w **Arkuszu właściwości** dowolnego obiektu rozwinąć ulokowaną na górze listę i wybrać nazwę dowolnego innego obiektu. Takim obiektem może być cały formularz. Na rysunku 14.12 przedstawiono listę w momencie zaznaczenia nazwy **Formularz**.

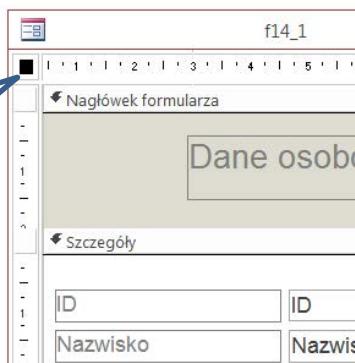


Rys. 14.12. Wybór nazwy **Formularz** w celu obejrzenia jego właściwości

Wyświetlić **Arkusz właściwości** formularza można również w inny sposób. W tym celu należy zaznaczyć formularz, klikając lewym przyciskiem myszy na selektorze formularza. Ma on postać dużej kropki, umieszczonej w lewym górnym rogu formularza (rys.14.13). Potem należy wybrać

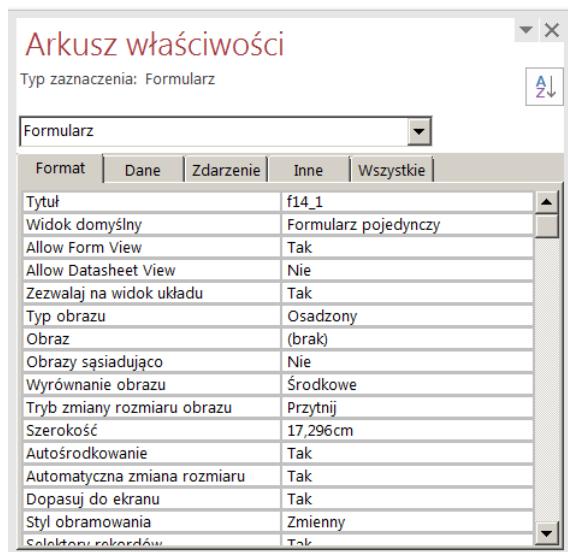


ikonkę **Arkusz właściwości** na wstążce.



Rys. 14.13. Zaznaczenie formularza w celu wywołania jego **Arkusza właściwości**

Na rysunku 14.14 zaprezentowano widok **Arkusza właściwości** formularza **f14\_1**.



Rys. 14.13. Widok **Arkusza właściwości** formularza

## Zadanie 14\_4

W celu sprawdzania zrozumienia nowych informacji otwórz **Arkusz właściwości** formularza **f14\_1** i odpowiedz na następujące pytania:

- ❖ Jaka jest szerokość wykonanego przez Ciebie formularza (zakładka **Format**)?
- ❖ Czy przyciski nawigacyjne mają być w nim pokazywane (zakładka **Format**)?
- ❖ Co jest źródłem rekordów formularza (zakładka **Dane**)?
- ❖ Czy dodawanie danych jest dozwolone (zakładka **Dane**)?
- ❖ Czy usuwanie rekordów jest dozwolone (zakładka **Dane**)?
- ❖ Czy edycja rekordów jest dozwolona (zakładka **Dane**)?

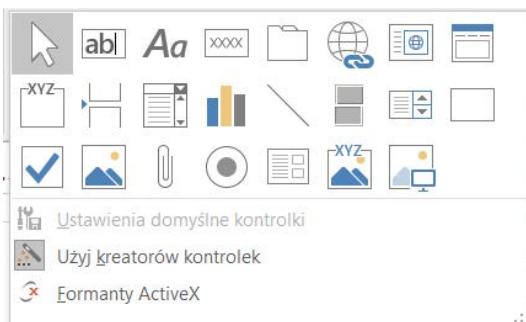
### 14.3. Formant obliczeniowy

W formularzu można utworzyć kontrolkę obliczeniową (formant obliczeniowy). Różni się ona od zwykłych kontrolek niepowiązanych tym, że jej źródłem danych jest wyrażenie. Wyrażenie jest kombinacją operatorów,

nazw formantów, nazw pól, funkcji zwracających jedną wartość i wartości stałych. Wyrażenia omawiane są przy tworzeniu kwerend. W przypadku formularza buduje się je bardzo podobnie.

Do tworzenia formantu obliczeniowego służy kontrolka powiązana

**Pole tekstowe**. Odpowiednią ikonkę znajdujemy na wstążce, na zakładce **Projektowanie**. Na rysunku 14.15 zaprezentowano formanty powiązane i niepowiązane widoczne po rozwinięciu listy na zakładce **Projektowanie**.



Rys. 14.15. Lista kontrolek na zakładce **Projektowanie**

## Zadanie 14\_5

Utwórz w formularzu **f14\_1** trzy pola obliczeniowe do wyświetlenia dnia, miesiąca urodzin osoby. Formularz powinien mieć postać podobną do przedstawionej na rysunku 14.23.

### Wykonanie

- Przejdź do widoku projektu formularza.
- W celu uzyskania miejsca na formularzu do umieszczenia formantu obliczeniowego przeciągnij w dół pasek, oddzielający **stopkę formularza od szczegółów**.
- Umieść w formularzu kontrolkę **Pole tekstowe**, pobierając ją ze wstążki. Postać formularza ma zostać podobna do przedstawionej na rysunku 14.16.
- Zaznacz za pomocą myszki wszystkie formanty (tzn. określ wskaźnikiem myszki prostokąt zawierający wszystkie formanty), wybierz na wstążce kartę Rozmieszczenie, a w niej ikonkę Stosowy. W wyniku postać projektu zmieni się, jak na rysunku 4.17.

The screenshot shows the Microsoft Access Form Designer window titled 'f14\_1'. The main area is labeled 'Szczegóły' (Details). It contains several text boxes paired with dropdowns or other controls. A new text box control, 'Tekst13', has been placed in the layout, positioned below the 'Płeć' and 'PESEL' controls. The 'Tekst13' control is highlighted with a yellow selection border.

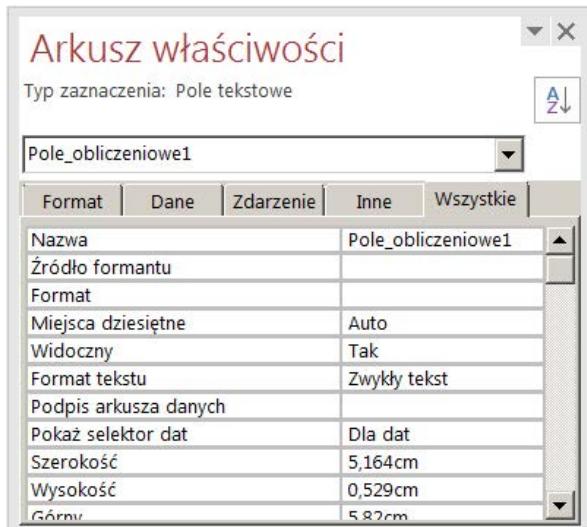
Rys. 14.16. Widok projektu formularza **f14\_1** po umieszczeniu na nim pustej kontrolki **Pole tekstowe**

This screenshot shows the same form designer window as above, but now all the controls (text boxes and dropdowns) are enclosed within a single large orange rectangular formant. This indicates that they have been grouped together as a single formant object.

Rys. 14.17. Widok projektu formularza **f14\_1** z formantami w układzie stosowym

- Formant umieszczony na formularzu ma domyślną nazwę. Na rysunkach 14.16 oraz 14.17 widzimy nazwę **Tekst13** (w twojej bazie danych może być inna nazwa). Zmień nazwę formantu. Przykładowo niech to będzie: „**Pole\_obliczeniowe1**”. W tym celu:

- zaznacz prawą część formantu i wybierz na wstążce, w zakładce **Projektowanie**, w grupie **Narzędzia** polecenie **Arkusz właściwości**;
- w **Arkuszu właściwości** wybierz zakładkę **Wszystkie**;
- wprowadź nową nazwę w wierszu **Nazwa** (rys. 14.18).



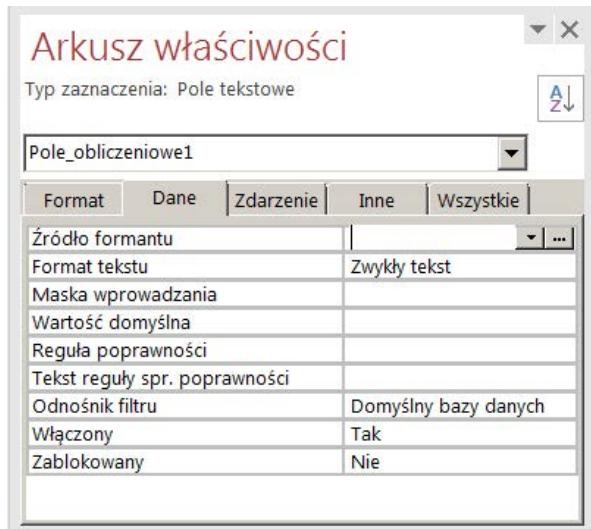
Rys. 14.18. Widok **Arkusza danych** formantu obliczeniowego po zmianie jego nazwy

- f. Kliknij myszką w lewej części formantu **Pole\_obliczeniowe1** i wpisz tekst etykiety: „Dzień urodzenia” (rys. 14.19).



Rys. 14.19. Widok formantu **Pole tekstowe** po wprowadzeniu tekstu etykiety

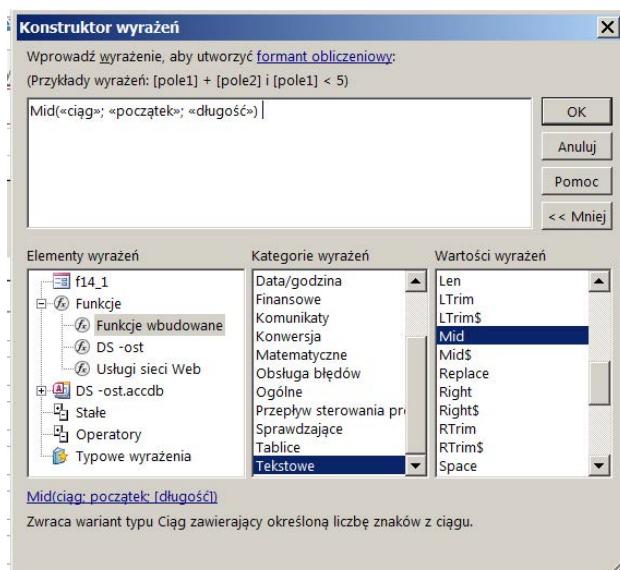
- g. W prawej części formantu **Pole\_obliczeniowe1** zbuduj za pomocą **kreatora wyrażeń** funkcję do zdefiniowania dnia urodzenia. Postępuj następująco:
- zaznacz prawą część formantu i wybierz na wstążce, na zakładce **Projektowanie**, w grupie **Narzędzia**, polecenie **Arkusz właściwości**;
  - wybierz w **Arkuszu właściwości** zakładkę **Dane**;
  - w **Arkuszu Właściwości** na stronie **Dane**, w wierszu **Źródło formantu** kliknij przycisk (rys. 14.20), a otworzysz okno **Konstruktora wyrażeń**;



Rys. 14.20. Zakładka **Dane** arkusza właściwości formantu **Pole\_obliczeniowe1**

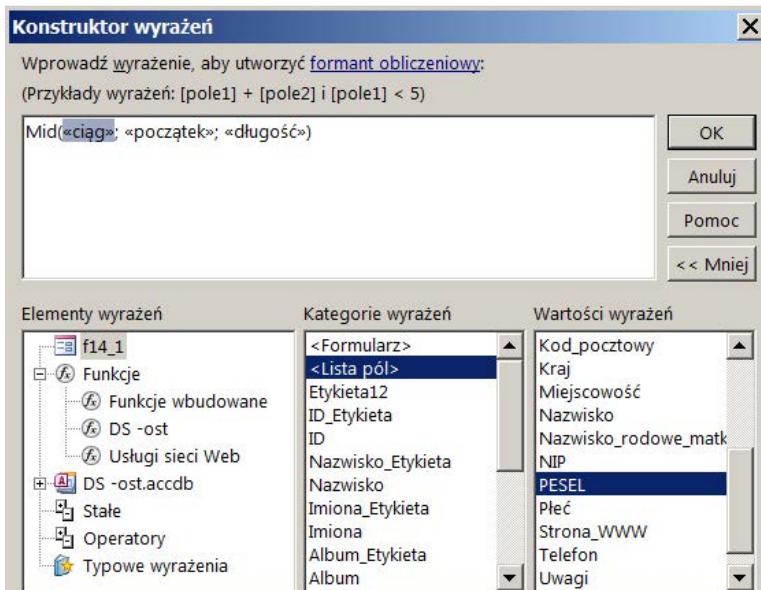
- w oknie **Konstruktora wyrażeń**

- ✓ wybierz (rys. 14.21) szablon funkcji **Mid** (wybierając kolejno w kolumnach konstruktora nazwy: **Funkcje**, **Funkcje Wbudowane**, **Tekstowe**, **Mid**),



Rys. 14.21. Widok szablonu funkcji **Mid** w oknie **Konstruktora wyrażeń**

- ✓ zaznacz pierwsze pole w szablonie – «ciąg»;
- ✓ zaznacz nazwę formularza **f14\_1** (w kolumnie **Elementy wyrażeń**),
- ✓ w sąsiedniej kolumnie kliknij polecenie **Lista pól**,
- ✓ w prawej kolumnie wybierz nazwę pola *PESEL* (rys. 14.22);



Rys. 14.22. Zdefiniowanie ciągu w funkcji **Mid**

- kliknij dwukrotnie – nazwa pola zostanie dodana do wyrażenia; dodane razem z nazwą nawiasy kwadratowe służą w programie **ACCESS** do przedstawienia nazw pól i tabel;
- zamiast <<początek>> wpisz numer pozycji PESEL, od której zaczyna się numer miesiąca, tzn. 5, a zamiast <<długość>> wpisz 2 – wyrażenie będzie miało postać:

**= Mid ([PESEL];5;2)**

(dodane razem z nazwą nawiasy kwadratowe służą w programie **ACCESS** do przedstawienia nazw obiektów, ale tutaj nawiasy kwadratowe nie są konieczne);

- wybierz **OK** – okno kreatora zostanie zamknięte;
  - naciśnij klawisz **ENTER**;
  - zamknij okno **Arkusza właściwości**.
- h. Przejdz do widoku formularza i przekonaj się, że utworzone pole wyliczane działa prawidłowo.

- i. Utwórz drugie pole obliczeniowe samodzielnie. Nadaj mu nazwę „Pole\_obliczeniowe2”. W oknie konstruktora wyrażeń w celu zdefiniowania źródła rekordów wprowadź funkcję:

```
= MonthName(Mid([PESEL];3;2)).
```

- j. Otwórz arkusz właściwości formularza i w zakładce **Format** zmień właściwość **Tytuł**, wpisując tekst: „Formularz zawierający pola obliczeniowe” (rys. 14.23).

ID	<input type="text" value="1"/>
Nazwisko	<input type="text" value="Kulesza"/>
Imiona	<input type="text" value="Paweł"/>
Album	<input type="text" value="34567"/>
Płeć	<input type="button" value="Mężczyzna"/>
PESEL	<input type="text" value="90112212315"/>
Dzień urodzenia	<input type="text" value="22"/>
Miesiąc urodzenia	<input type="text" value="listopad"/>

Rekord: 1 z 9

Rys. 14.23. Formularz f14\_1 w widoku formularza

#### 14.4. Przykłady utworzenia formularzy w widoku projektu

Wszystkie formularze zbudowane do tej pory były utworzone automatycznie lub za pomocą kreatora. Dopiero w poprzednim paragrafie wprowadziliśmy pewne zmiany do utworzonego wcześniej formularza. Teraz zapoznamy się z technikami potrzebnymi do tego, aby wykonać formularz od podstaw w **Widoku projektu**.

Następny przykład przedstawia proces utworzenia formularza kolumnowego. W takim formularzu wyświetlany jest zawsze jeden rekord danych. Pola kolejnego rekordu pokazują się w wyniku przewijania rekordów.

## Zadanie 14\_6

Utwórz w **Widoku projektu** formularz o nazwie **f14\_2** do wyświetlenia danych z tabeli **wpłaty**.

W formularzu:

- ❖ umieść wszystkie pola tabeli **wpłaty**;
- ❖ utwórz formant obliczeniowy do wyliczenia wartości w wysokości 5% od wartości pola *Wpłaty*;
- ❖ utwórz formant obliczeniowy, który na bazie wartości pola *Za\_rok\_akad* przedstawi rok akademicki w zwykłej postaci (tzn. dwóch kolejnych lat rozdzielonych kreską); po wykonaniu tego formantu pole *Za\_rok\_akad* zróbi niewidocznym;
- ❖ każdy formant sekcji **Szczegóły** wyświetli, ustawiając w **Atrykuszu Właściwości** opcję **Efekt specjalny - Cieniowany**;
- ❖ umieść logo oraz bieżącą datę i godzinę;
- ❖ w nagłówku wpisz tekst: „Przegląd danych tabeli Wpłaty”;
- ❖ na stopce umieść swoje imię i nazwisko.

Formularz powinien być podobny do przedstawionego na rys. 14.24.

## Wykonanie

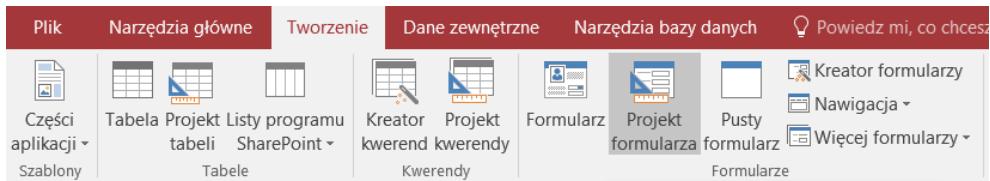
The screenshot shows a Microsoft Access form titled "Formularz o dwóch formantach obliczeniowych". The form contains the following data:

ID Wpłaty	1
ID	1
Pokwitowanie	UJ-1234567
Data wpłaty	2014-10-20
Wpłata	280,00 zł
5% wpłaty	14,00 zł
Za rok akademicki	2014/2015

Below the form, the status bar displays: "Wykonała Mariola Socha". At the bottom, there is a navigation bar with buttons for record navigation, search, and filter options.

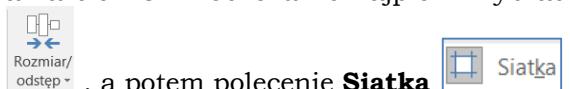
Rys. 14.24. Widok formularza **f14\_2**

- a. Wybierz na wstążce karte **Tworzenie**, a potem ikonkę **Projekt formularza** (rys. 14.25) – wyświetli się pusty formularz w widoku projektu.



Rys. 14.25. Widok wstążki z wybranym poleceniem **Projekt formularza**

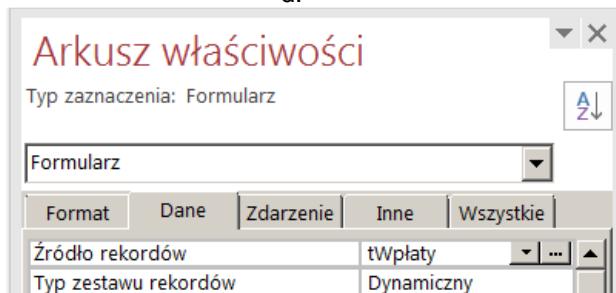
- b. Dodaj siatkę, która pomoże przy wyrównaniu formantów. Siatka zostanie dodana, jeśli na karcie **Rozmieszczenie** najpierw wybrać ikonę



**kę Rozmiar/odstęp** , a potem polecenie **Siatka** .

- c. Zaznacz formularz i otwórz **Arkusz właściwości**, w którym zdefiniuj źródło rekordów formularza. W tym celu na zakładce **Dane**, w wierszu **Źródło rekordów** wybierz z listy rozwijanej nazwę tabeli **tWpłaty** (rys. 14.26).

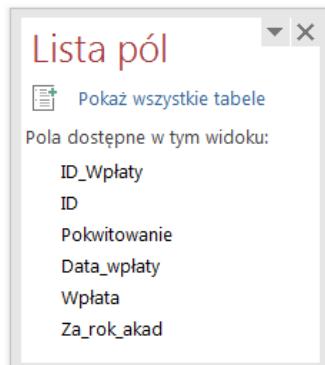
d.



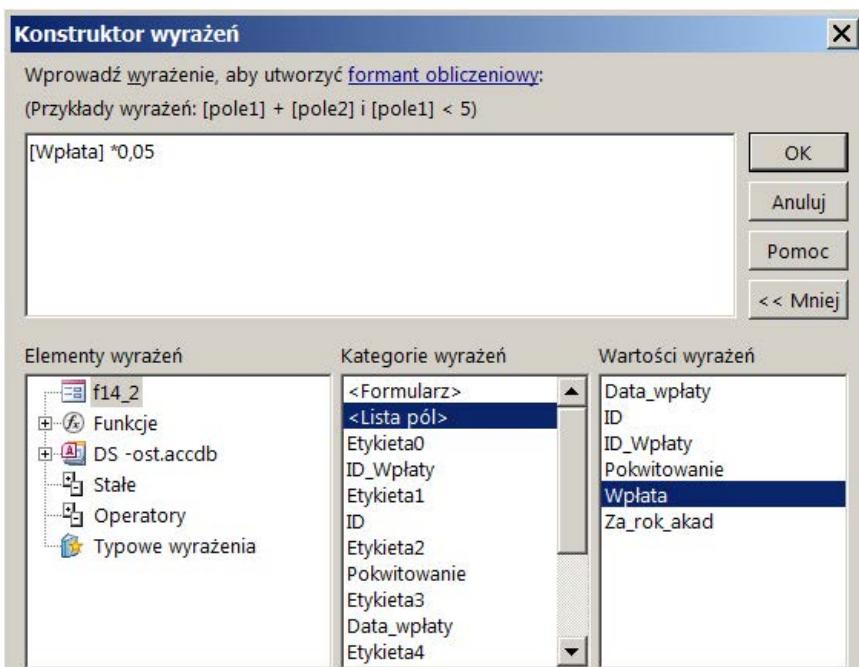
Rys. 14.26. Widok **Arkusza właściwości** formularza ze zdefiniowanym źródłem rekordów

- e. Wybierz na wstążce, na zakładce **Projektowanie**, w grupie **Narzędzia** przycisk **Dodaj istniejące pola** – wyświetli się lista pól, które można umieścić na formularzu (rys. 14.27).
- f. Utrzymując wciśnięty klawisz **SHIFT**, zaznacz na liście pól wszystkie pola tabeli **tWpłaty** i przeciągnij je myszką do formularza.
- g. Zapisz formularz, nadając mu nazwę **f14\_2**.
- h. W sekcji **Szczegóły** utwórz formant obliczeniowy do wyświetlenia 5% od wartości w polu *Wpłata*:
- o umieść w formularzu kontrolkę **Pole tekstowe**;

- o zaznacz go i w **Arkuszu właściwości** na zakładce **Dane** wybierz w wierszu **Źródło formantu** przycisk ..., wywołując **Konstruktor wyrażeń**;
- o w oknie **Konstruktora wyrażeń** wprowadź wyrażenie: **= [Wpłata]\*0,05** (rys. 14.28);



Rys 14.27. Lista pól dostępnych do umieszczenia w formularzu



Rys 14.28. Wyrażenie w oknie Konstruktora wyrażeń

- wybierz **OK**;
- w oknie **Arkusz właściwości** tworzonego pola obliczeniowego na zakładce **Format** w polu **Format** wybierz **Walutowy**;
- wprowadź do lewej części pola tekstowego: „5% wpłaty”. Teraz formant będzie miał postać jak na rysunku 14.29.

5% wpłaty	=[Wpłata]*0,05
-----------	----------------

Rys 14.29. Widok pierwszego pola obliczeniowego w projekcie formularza

- i. W sekcji **Szczegóły** utwórz drugi formant obliczeniowy, który wyświetli ciąg znaków przedstawiający rok akademicki (rys. 14.30). W tym celu najpierw umieść kontrolkę **Pole tekstowe** w formularzu. Następnie do lewej części pola tekstowego wpisz: „Za rok akademicki”. Do utworzenia wyrażenia w prawej części formantu skorzystaj z operatora **&** (służy do łączenia ciągów). Źródłem formantu będzie wyrażenie:

```
= [Za_rok_akad] & "/" & ([Za_rok_akad]+1).
```

Należy tu zauważyć, że wartość liczbową pola *Za\_rok\_akad* będzie automatycznie przekonwertowana przez program do typu tekstowego.

Za rok akademicki	=[Za_rok_akad] & "/" & ([Za_rok_akad]+1)
-------------------	--

Rys 14.30. Widok drugiego pola obliczeniowego w projekcie formularza

- j. Nowo utworzony formant umieść w miejscu pola tabeli *Za\_rok\_akad*, które po prostu usuń.
- k. Zaznacz wszystkie formanty w sekcji **Szczegóły**, otwórz **Arkusz właściwości**, na zakładce **Format** w wierszu **Efekt specjalny** wybierz z listy rozwijanej opcję **Cieniowany**.
- l. Zastosuj do formantów sekcji **Szczegóły** układ stosowy.
- m. Dodaj nagłówek i stopkę. Na stopce umieść etykietę, a w niej swoje imię i nazwisko, jak na rysunku 14.24.
- n. Z wykorzystaniem poleceń paska **Nagłówek/Stopka** wstaw w nagłówku bieżącą datę i logo.
- o. Zmień właściwość **Tytuł** formularza. Wprowadź tekst: „Formularz o dwóch formantach obliczeniowych”.
- p. Zapisz formularz.

Kolejny przykład prezentuje utworzenie formularza od podstaw i zastosowanie przy tym kontrolek **Karta**, **Pole kombi**, **Podformularz/podraport** oraz **Przycisk**.

## Zadanie 14\_7

Utwórz w **Widoku projektu** formularz **f14\_3** o trzech zakładkach. Przy wyborze konkretnego rekordu w zakładkach powinny się znaleźć dane o osobie, jej zakwaterowaniach oraz wpłatach. Na pierwszej zakładce należy ulokować dane z tabeli **tOsoba**, na drugiej – dane tabeli **tZakwaterowanie**, na trzeciej – dane tabeli **tWpłaty**.

Na pierwszej stronie umieść także listę rozwijaną, zawierającą albumy osób. Będzie ona wspomagać wyszukiwanie danych według pola *Album* tabeli **tOsoba**. Obok listy ulokuj przyciski do nawigowania pomiędzy rekordami, a pasek z przyciskami nawigacyjnymi zrób niewidocznym.

Formularz powinien być podobny do przedstawionego na rysunku 14.31.

### Wykonanie

Przykład zastosowania kontrolek KARTA, PRZYCISK, POLE KOMBI, PODFORMULARZ/PODRAPORT

Dane osobowe    Zakwaterowanie    Wpłaty

ID	2	Adres	ul. Sienkiewicza 48 m 5
Nazwisko	Poniatowska	Miejscowość	Białystok
Imiona	Maria	Województwo	podlaskie
Album	87890	Kod pocztowy	15-345
Płeć	Kobieta	Kraj	Polska
PESEL	94090989766	Foto	
NIP	234-567-89-01		
Imię ojca	Bogumił		
Imię matki	Fryderyka	Dok	
Nazwisko rodowe matki	Iwaszkiewicz	Uwagi	
e-mail	poniatowska@o2.pl		
Telefon	567-244-667	Strona WWW	
Wybierz album:	87890	▶	◀ ▶ □

Rys 14.31. Formularz **f14\_3**

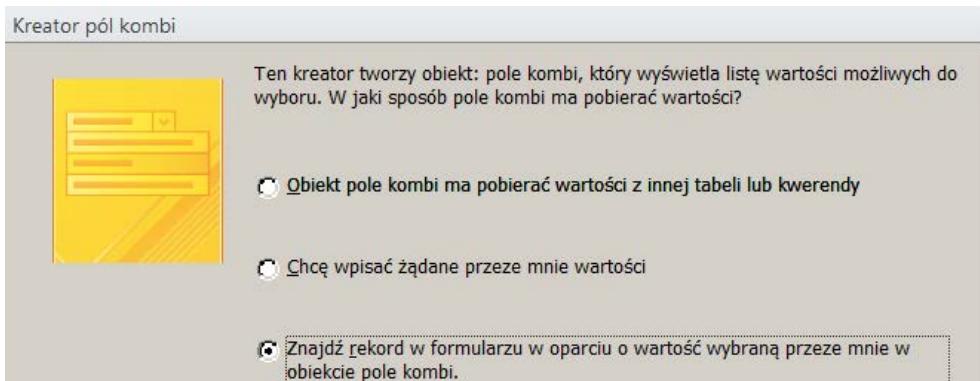
Pierwsza strona formularza

- Utwórz pusty formularz w **Widoku projektu**.

- b. Otwórz **Arkusz właściwości** formularza i zdefiniuj **Źródło rekordów** – wybierz tabelę **tOsoba**.
- c. Przejdz na zakładkę **Format** w **Arkusza właściwości formularza** i w wierszu **Przyciski nawigacyjne** wybierz **Nie**.
- d. Zapisz formularz, nadając mu nazwę **f14\_3**.
- e. Pobierz na wstążce i umieść w formularzu formant **Karta**, rozciągając go za pomocą wskaźnika myszy na cały obszar sekcji **Szczegóły**.
- f. Kliknij na słowie **Strona1**, a następnie wybierz **Arkusz właściwości** na wstążce i w zakładce **Format** w wierszu **Tytuł** wpisz: „**Dane osobowe**” (przy zapisie tytułu cudzysłowu nie stosujemy).
- g. Tytuł zakładki **Strona2** zamień na „**Zakwaterowanie**” (rys. 14.31).
- h. Dodaj jeszcze jedną stronę: kliknij prawym przyciskiem myszy na zakładce formularza i wybierz polecenie **Wstaw stronę**.
- i. Napis zakładki **Strona3** zamień na „**Wpłaty**”.
- j. Przejdz do widoku projektu i umieść na pierwszej stronie formularza wszystkie pola tabeli **tOsoba**, przeciągając je za pomocą wskaźnika myszy z **Listy pól**. Można umieścić je dowolnie. Na rysunku 14.32 przedstawiono zakładkę **Dane osobowe** w widoku projektu po ulokowaniu pól tabeli **tOsoba** i wprowadzeniu zmian do wyglądu formantów.

Rys 14.32. Widok formularza **f14\_3** po umieszczeniu w nim pól tabeli **tOsoba**

- k. Przeciągnij kontrolkę **Pole kombi**  w dolną część formularza – natychmiast włączy się **Kreator pola kombi** (w przypadku, gdy się jego okno nie pojawi, należy kliknąć na poleceniu wstążki **Użyj kreatorów kontrolek**). Dalej postępuj następująco:
- w pierwszym oknie **Kreatora pola kombi** (rys. 14.33) wybierz opcję **Znajdź rekord w formularzu w oparciu o wartość wybraną przeze mnie w obiekcie pole kombi**;
  - w kolejnym oknie **Kreatora pól kombi** wśród dostępnych pól wybierz *Album*;
  - wybierz **Dalej**;
  - w kolejnym oknie wybierz **Dalej**;
  - wpisz tekst etykiety: „Wybierz album”;
  - **Zakończ**. Widok listy rozwijanej w widoku projektu będzie podobny do przedstawionego na rysunku 14.34.



Rys 14.33. Wybór sposobu pobierania wartości przez pole kombi

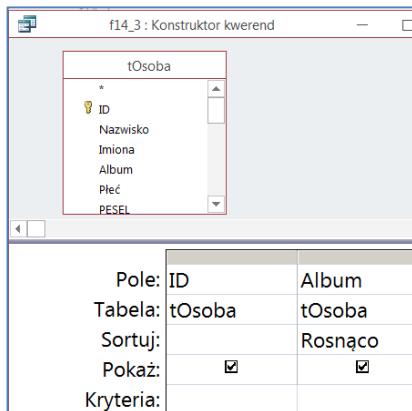


Rys. 14.34. Widok pola kombi w projekcie formularza

- l. Uporządkuj wartości w polu kombi. W tym celu zaznacz kontrolkę i otwórz jej **Arkusz właściwości**. W zakładce **Dane** w wierszu **Źródło wierszy** ma być instrukcja **SQL**:

`SELECT tOsoba.ID, tOsoba.Album FROM tOsoba.`

- m. Kliknij na przycisku po prawej  i wprowadź zmiany do projektu kwerendy, dodając sortowanie według albumu, jak pokazano na rys. 14.35.



Rys. 14.35. Kwerenda będąca źródłem wierszy pola kombi

- n. Umieść obok listy rozwijanej także przyciski do nawigowania między rekordami.

Dla ulokowania pierwszego z przycisków nawigowania postępuj następująco:

- o wybierz w przyborniku kontrolkę **Przycisk** i umieść go na formularzu – wówczas zostanie wyświetcone okno **Kreatora przycisków poleceń**; wybierz kategorię **Nawigowanie pomiędzy rekordami**, akcję **Przejdź do następnego rekordu**;
- o **Dalej**;
- o **Dalej**;
- o **Zakończ**;
- o dodaj jeszcze trzy przyciski do nawigowania pomiędzy rekordami, korzystając z pomocy **Kreatora przycisków poleceń** (w celu zmiany rozmiarów i wyglądu przycisku najlepiej korzystać z poleceń zakładki **Format** jego **Arkusza właściwości**).

- p. Zapisz formularz i zamknij go.

## Druga strona formularza

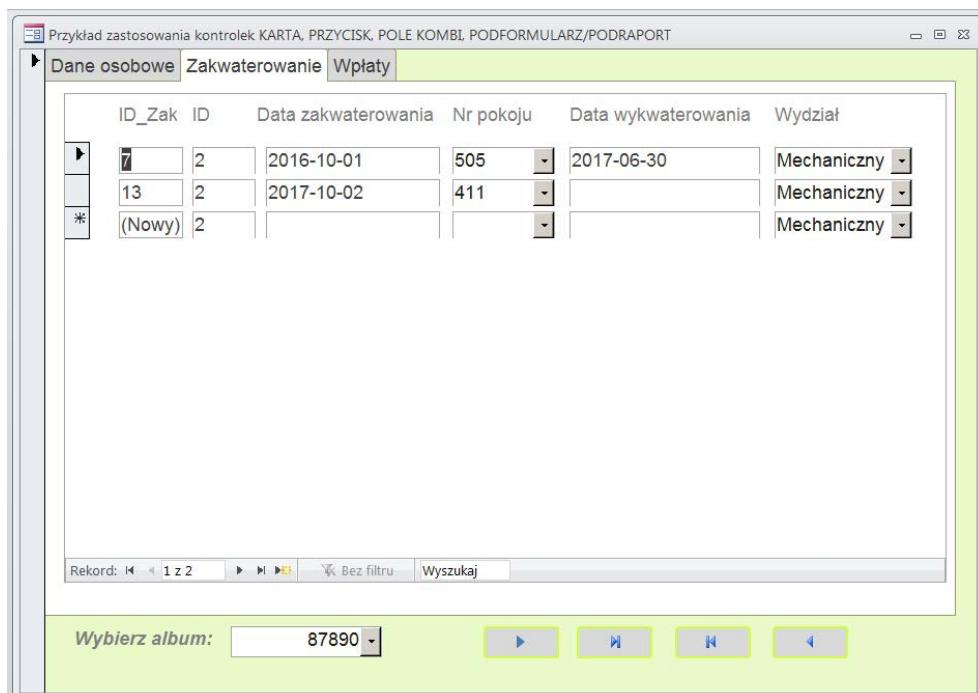
Pierwsza strona formularza zawiera pola tabeli **tOsoba**. Druga strona powinna zawierać dane wybranej osoby o jej zakwaterowaniach. W celu zademonstrowania stosowania kontrolki **Podformularz/podraport** dalej utworzymy dodatkowy formularz w oparciu o tabelę **tZakwaterowanie** i umieścimy go na drugiej stronie formularza **f14\_3**. Działaj dalej według poniższego algorytmu:

- a. Utwórz formularz na bazie tabeli **tZakwaterowanie** w postaci tabelarycznej, zapisz go, nadając mu nazwę **f14\_3\_Zakwaterowanie** i zamknij.
- b. Otwórz ponownie formularz **Z\_8-3** w widoku projektu.

- c. Kliknij na zakładkę **Zakwaterowanie** formularza.
- d. Wybierz na wstążce, na zakładce **Projektowanie**, formant **Podformularz/podraport**  i umieść go na zakładce **Zakwaterowanie**, rozciągając go myszką na cały obszar strony. W wyniku tego działania wyświetli się pierwsze okno **Kreatora podformularzy**. Dalej postępuj następująco:
- o wybierz opcję **Użyj istniejącego formularza** i wybierz w okienku poniżej nazwę formularza **f14\_3\_Zakwaterowanie**;
  - o kliknij **Dalej**;
  - o **Zakończ**.
1. Przejdz do widoku układu formularza i wprowadź niezbędne zmiany do postaci formularza, a także posortuj rosnąco dane według pola **Data\_zakwaterowania**;
- m. Zapisz formularz.

Na rys. 14.36 przedstawiono widok drugiej strony formularza.

Działając analogicznie do tworzenia zakładki **Zakwaterowanie**, umieść niezbędne formanty w zakładce **Wpłaty**. Zmień tytuł formularza i zapisz go.



ID_Zak	ID	Data zakwaterowania	Nr pokoju	Data wykwaterowania	Wydział
7	2	2016-10-01	505	2017-06-30	Mechaniczny
13	2	2017-10-02	411		Mechaniczny
(Nowy)	2				Mechaniczny

Rys 14.36. Druga strona formularza **f14\_3**

## 14.5. Formularz nawigacji

Program **Microsoft Access 2016** oferuje możliwość tworzenia tak zwanego formularza nawigacji. Ten formularz też zawiera zakładki, tak jak formularz **f14\_3**, lecz można wybrać szablon, co znacznie ułatwia wykonanie. Należy jednak zauważyć, że dane umieszczone na jednej karcie tego formularza nie mają połączenia z innymi kartami.

Są do wyboru różne postacie formularza nawigacji. Aby wybrać jedną z nich, wystarczy na wstążce wybrać kartę **Tworzenie**, a w niej menu **Nawigacja**, które proponuje szablon do wykonania formularza.

Formularz nawigacji dobrze nadaje się na pierwsze okno aplikacji.

W celu zapoznania się z tym rodzajem formularzy wykonaj następujące zadanie.

### Zadanie 14\_8

---

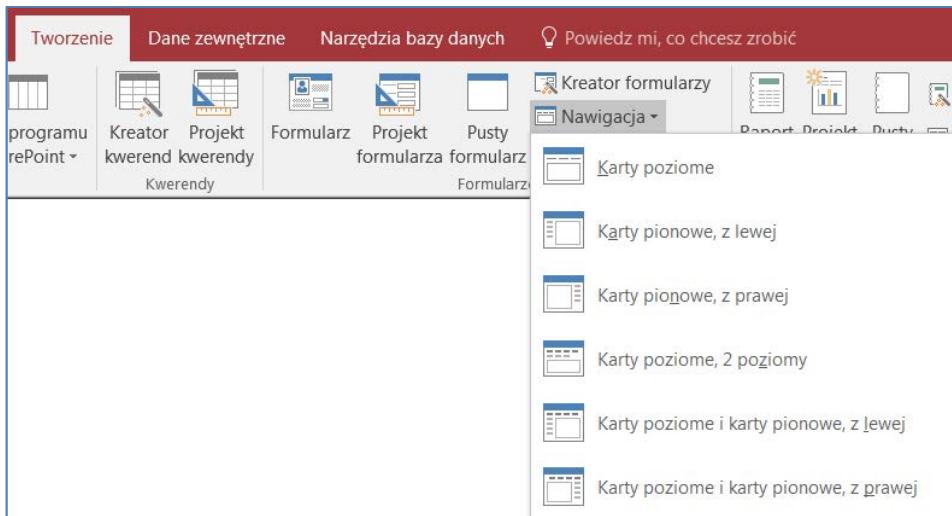
Utwórz formularz nawigacji o nazwie **f14\_4** do przedstawienia danych o osobach i pokojach. Formularz nawigacji ma być zbudowany na podstawie szablonu **Karty poziome**.

---

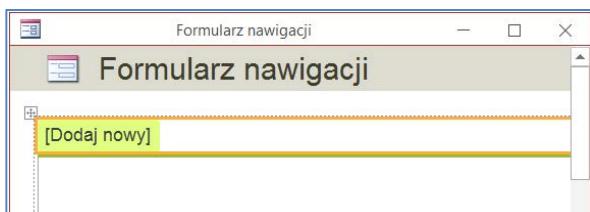
#### Wykonanie

- Wykonaj formularz kolumnowy o nazwie **f14\_4\_Osoby** na bazie tabeli **tOsoba**.
- Wykonaj formularz tabelaryczny w oparciu o tabele **tPokoje**.
- Wybierz z menu **Nawigacja** polecenie **Karty poziome** (rys. 14.37). Na ekranie zostanie wyświetlony szablon formularza (rys. 14.38).
- Zaznacz w okienku nawigacji nazwę formularza **f14\_4\_Osoby** i przeciągnij go wskaźnikiem myszy do przycisku nawigacji [Dodaj nowy]. Wówczas otrzymasz pierwszą kartę. Zmień tytuł karty, np. wpisz: „Osoba”.
- Działając podobnie, utwórz drugą kartę formularza w oparciu o formularz **f14\_4\_Pokoje**. Zmień tytuł karty, np. wpisz: „Pokoje”.

Na rys. 14.39 wyświetlono taki formularz utworzony automatycznie. Dalej można go upiększać według swojego gustu.



Rys 14.37. Menu **Nawigacja**



Rys 14.38. Szablon formularza nawigacji o kartach poziomanych

The screenshot shows the 'Formularz nawigacji' (Navigation Form) window with a horizontal card layout. The title bar says 'Formularz nawigacji'. There are two tabs at the top: 'Osoba' (active) and 'Pokoje'. The 'Osoba' tab displays four data entries:

Osoba	Pokoje		
ID	<input type="text" value="1"/>	Adres	<input type="text" value="ul. Stud..."/>
Nazwisko	<input type="text" value="Kulesza"/>	Miejscowość	<input type="text" value="Łomża"/>
Imiona	<input type="text" value="Paweł"/>	Województwo	<input type="text" value="podlasie"/>
Album	<input type="text" value="34567"/>	Kod pocztowy	<input type="text" value="16-456"/>

Rys 14.39. Formularz nawigacji **f14\_4**

## Zadanie do samodzielnego wykonania

### Zadanie 14\_9

Utwórz formularz o nazwie **f14\_5** o trzech kartach do wyświetlenia danych o zamieszkaniu i wpłatach w roku akademickim 2016/2017.

Wymagania do formularza:

- ❖ na pierwszej zakładce należy wyświetlić wszystkie dane osoby; mają się one pokazywać po podaniu przez Użytkownika w okienku dialogowym numeru albumu;
  - ❖ na drugiej - wydobyć wszystkie dane o zakwaterowaniu tej osoby w roku akademickim 2016/2017, przy tym zakładka ma wyświetlać wszystkie dane o zakwaterowaniu w postaci tabelarycznej;
  - ❖ na trzeciej - wyświetlić wszystkie dane o wpłatach zadanej osoby, też w postaci tabelarycznej.
- 

*Podpowiedź do wykonania zadania 14\_9.*

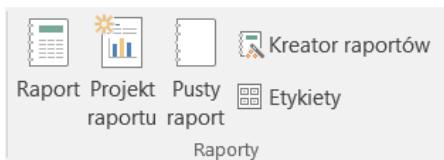
Wykonaj kopię formularza **f14\_3** i na niej zbuduj formularz **f14\_5**: zmień źródło rekordów (teraz ma być kwerenda parametryczna), usuń przyciski nawigacji oraz listę wyboru album.

## 15. Raporty

Raport jest obiektem specjalnie dostosowanym do potrzeb prezentacji i podsumowywania informacji o danych w postaci dokumentu przygotowanego do druku. Najczęściej służy on do prezentacji informacji pochodzących z bazy danych, tzn. do przedstawienia danych z określonych pól jednej tabeli lub wielu tabel.

*Microsoft Access* dysponuje różnymi narzędziami do tworzenia raportów. O pierwszych dwóch (narzędziu **Raport** oraz **Kreatorze raportów**) już wspomniano w rozdziale 4. Przy ich stosowaniu raport generuje się automatycznie. Jednak raport można projektować samodzielnie w sposób podobny do projektowania formularza. W tym celu należy wykorzystać okno projektu raportu.

Na karcie **Tworzenie** wstążki znajduje się grupa ikonek, które możemy stosować do wykonania raportu.



Rys. 15.1. Fragment wstążki z ikonami do tworzenia raportów

## 15.1. Generowanie raportu przy użyciu narzędzia RAPORT

Narzędzie **RAPORT** (rys. 15.1), którego ikonkę widzimy na karcie **Tworzenie**, umożliwia szybkie utworzenie raportu, a operacja generowania go nie wymaga podawania dodatkowych informacji. W zaproponowanym przykładzie raport jest generowany na podstawie jednej tabeli. W ten sam sposób może być utworzony na podstawie kwerendy, która wyświetla dane z wielu tabel.

### Zadanie 15\_1

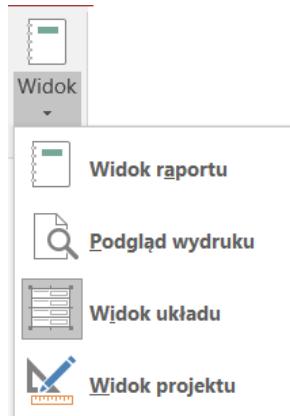
Utwórz raport o nazwie **r15\_1** do wyświetlenia danych o pokojach w Domu Studenta.

#### Wykonanie

- Kliknij w okienku nawigacji tabelę **tPokoje**, aby zdefiniować ją jako źródło raportu.
- Na karcie **Tworzenie** w grupie **Raporty** (rys. 15.1) kliknij przycisk **Raport** – program utworzy raport i wyświetli go w widoku układu.
- Zapisz raport, nadając mu nazwę **r15\_1**.

Po wykonaniu opisanych czynności utworzony raport przedstawiony jest w **widoku układu**, aktywna jest karta **Projektowanie**. W **Widoku układu** po wyświetleniu rzeczywistych danych raportu można dopasować szerokość oraz zmienić układ kolumn, a także dodać poziomy grupowania i sumy. Można umieszczać pola w projekcie raportu oraz ustawać właściwości raportu i jego formantów.

Przycisk umieszczony skrajnie po lewej stronie wstążki (rys. 15.2) pozwala także na obejrzenie raportu w oknie **Podglądu wydruku**.



Rys. 15.2. Przyciski do zmiany widoku raportu

Przełącz się do widoku **Podglądu wydruku**, a po tym zamknij go za pomocą przycisku **Zamknij podgląd wydruku**.

## Zadanie 15\_2

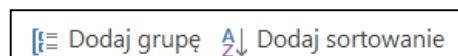
---

W widoku układu raportu **r15\_1** umieść kolumnę z numerami pięter jako pierwszą, a jako drugą – numery pokojów. Pogrupuj i posortuj dane według numeru piętra, a także posortuj numery pokoi w każdej grupie. Scenariusz dane. Zapisz zmieniony raport, nadając mu nazwę **r15\_2**.

---

### Wykonanie

- Umieść kursor myszy wewnętrz jednej z wartości pola *Piętro*, kliknij na kartę **Rozmieszczenie** i wybierz ikonkę **Zaznacz kolumnę**. Przeciągnij kolumnę za pomocą wskaźnika myszy w lewo, aby umieścić przed kolumną **Pokoje**.
- W celu uporządkowania numerów piętra zaznacz kolumnę *Piętro*, wybierz kartę **Projektowanie** i kliknij na ikonę **Grupuj i sortuj**. Poniżej tabeli pojawią się ikonki do grupowania i sortowania danych (rys. 15.3).



Rys. 15.3. Ikonki grupowania i sortowania danych w widoku projektu raportu

- c. Kliknij na ikonę **Dodaj grupę** i wybierz z listy rozwijającej się nazwę **Piętro**. Poniżej tabeli zobaczymy wybrane opcje w postaci przedstawionej na rysunku 15.4. Dane kolumny *Pietro* domyślnie zostaną posortowane rosnąco.



Rys. 15.4. Widok opcji wybranych w celu grupowania danych według pola Pietro tabeli **Pokoje**

- e. Centruj dane w kolumnach. W tym celu zaznacz tabelę, klikając na małym kwadraciku w lewym górnym rogu tabeli, a potem wybierz na karcie **Narzędzia główne** ikonkę centrowania .  
 f. Zmień tytuł raportu: zamiast „**tPokoje**” pozostaw „**Pokoje**”.  
 g. Pozostaw lub usuń ustawione domyślnie datę i godzinę utworzenia raportu.  
 h. Usuń domyślne logo obok tytułu raportu.  
 i. Przejdz do **Widoku raportu**. Zbudowany raport będzie wyglądał podobnie do zaprezentowanego na rysunku 15.4.

Piętro	Nr Pokoju	Liczba miejsc	Uwagi
1	120	2	
	115	2	
	101	1	
2	210	1	
	202	2	

Rys. 15.5. Raport **r15\_2** w widoku układu

Po utworzeniu raportu należy go dalej redagować w celu wydruku. Zmiany te wnosimy w **widoku projektu**, a potem w **widoku podglądu wydruku**.

## 15.2. Tworzenie raportu przy użyciu narzędzia KREATOR RAPORTÓW

Tworząc raport za pomocą oprogramowania **Microsoft Access**, wychodząmy z następującego założenia:

- aby wyświetlać dane ze wszystkich rekordów wybranych pól tabel, możemy skorzystać z kreatora raportów;
- jeżeli interesują nas rekordy spełniające pewne warunki, to przed generowaniem raportu niezbędne jest utworzenie odpowiedniej kwerendy.

Rozważmy przykład raportu, który wyświetla dane z niektórych pól wybranych rekordów dwóch tabel.

### Zadanie 15\_3

---

Utwórz za pomocą kreatora raport o nazwie **r15\_3** do wyświetlenia danych o zakwaterowaniach w zakresie czasu od 1 września 2016 roku do 1 czerwca 2017 roku w postaci danych z pól: *Data\_zakwaterowania, Nazwisko, Imiona, Album, Data\_wykwaterowania, Nr\_pokoju*.

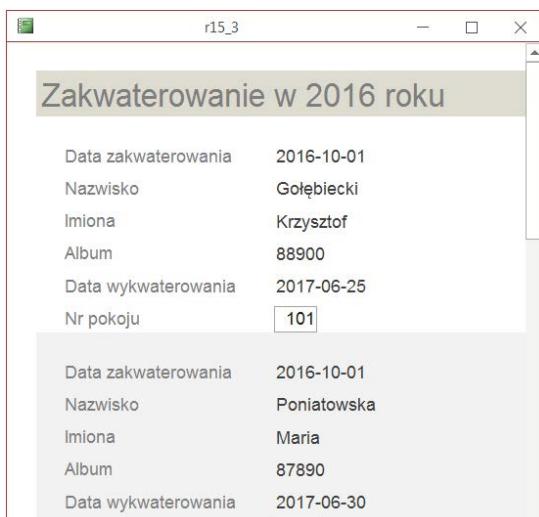
---

#### Wykonanie

- a. Utwórz kwerendę wybierającą o nazwie **k15\_1** opartą na tabelach **tOsoba** oraz **tZakwaterowanie** do wyświetlenia pól *Data\_zakwaterowania, Nazwisko, Imiona, Album, Data\_wykwaterowania, Nr\_pokoju*.
- b. Zdefiniuj w projekcie kwerendy kryterium dla pola *Data\_zakwaterowania*:  
 $>=\#2016-09-01\# \text{ AND } <= \#2017-06-01\#$ .
- c. Wybierz na zakładce **Tworzenie** w grupie **Raporty** polecenie **Kreator raportów** (rys. 15.1).
- d. W oknie **Kreatora raportów** wybierz z listy **Tabele/kwerendy** nazwę kwerendy **k15\_1**.
- e. Wybierz wszystkie pola kwerendy, a następnie wybierz **Dalej**.
- f. Wybierz sposób wyświetlania danych: **przez Zakwaterowanie**, a następnie **Dalej**.
- g. Wybierz **Dalej**.
- h. Wybierz sortowanie według *Data\_zakwaterowania, Nazwisko, Imiona, Album*
- i. **Dalej**.
- j. Wybierz **Układ – Kolumnowy, Orientacja – Pionowa, Dalej**.

- k. **Jaki ma być tytuł raportu?** Wpisz: „Zakwaterowanie w 2016 roku” (bez znaków cudzysłowu).
- l. **Zakończ.**
- m. Przejdź do **Widoku układu** i wprowadź do raportu niezbędne zmiany.  
Aby zaznaczyć za pomocą myszy całą kolumnę, należy przytrzymać klawisz SHIFT, klikając na każdym polu. Przesunięcia formantów najłatwiej dokonywać, korzystając z klawiszy <, >, ^, v.  
Aby zmienić rozmiar formantu, wystarczy pociągnąć wskaźnikiem myszy za prowadnicę; do formatowania tekstu najlepiej jest używać ikonki z karty **Narzędzia główne**.

Na rysunku 15.6 przedstawiono raport **r15\_3** w **Widoku raportu**.



Rys. 15.6. Raport **r15\_3** w **widoku raportu**

Trzeba mieć na uwadze, że przed wydrukiem raportu należy dopasować opcje druku do swoich potrzeb w oknie **Widoku podglądu wydruku**. Przy tym najczęstszym błędem użytkowników jest przekroczenie dopuszczalnej szerokości raportu względem szerokości karty papieru.

### 15.3. Tworzenie raportu w Widoku projektu

Tworzenie raportu w **Widoku projektu** zawsze należy zaczynać od zdefiniowania źródła jego rekordów w **Arkuszu właściwości** raportu.

Jeżeli wszystkie dane znajdują się w jednej tabeli, to źródłem rekordów może być właśnie ta tabela. W innym przypadku należy utworzyć kwerendę do wyświetlenia danych ze wszystkich tabel zawierających potrzebne informacje, a stworzona kwerenda będzie źródłem rekordów raportu.

Po przełączeniu widoku dowolnego raportu na **widok projektu** widzimy, że jest on bardzo podobny do **widoku formularza**. Przykładowo na rys. 15.7 przedstawiono raport **r15\_1** w widoku projektu.

W widoku projektu widać, że raport jest podzielony na sekcje:

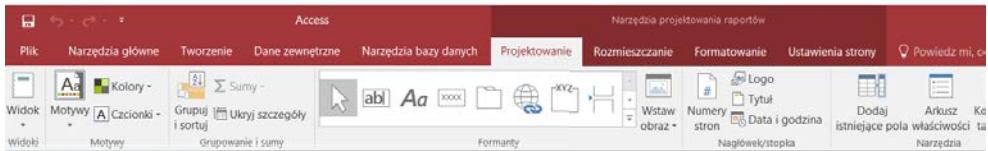
- **nagłówek raportu** (ta sekcja jest drukowana tylko raz – na początku raportu – i służy do umieszczenia informacji, które zwykle są drukowane na stronie tytułowej, takich jak: logo, tytuł lub data);
- **nagłówek strony** (ta sekcja jest drukowana u góry każdej strony);
- **szczegóły** (ta sekcja jest drukowana jednorazowo dla każdego wiersza w źródle rekordów; zawiera ona formanty tworzące główną treść raportu);
- **stopka strony** (ta sekcja jest drukowana na końcu każdej strony i służy do drukowania numerów stron oraz pozostałych informacji dotyczących całych stron);
- **stopka raportu** (ta sekcja jest drukowana tylko raz – na końcu raportu i służy do drukowania sum oraz pozostałych informacji podsumowujących cały raport).

Rys. 15.7. Raport **r15\_1** w **widoku projektu**

Oprócz wymienionych powyżej sekcji w raporcie mogą występować:

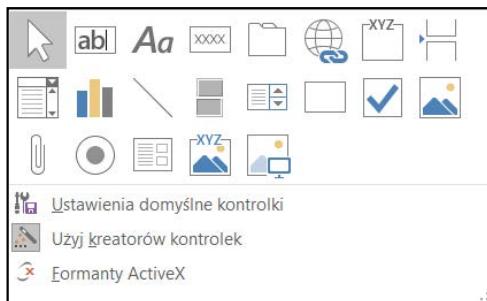
- **nagłówek grupy** (ta sekcja jest drukowana na początku każdej grupy rekordów i umożliwia drukowanie nazwy grupy);
- **stopka grupy** (ta sekcja jest drukowana na końcu każdej grupy rekordów i umożliwia drukowanie informacji podsumowujących dla grup).

W **widoku projektu** raportu na wstążce domyślnie jest otwarta karta **Projektowanie** (rys. 15.8).



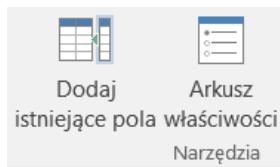
Rys. 15.8. Karta **Projektowanie** wstążki w widoku projektu raportu

Do raportu możemy dodawać formanty, wybierając je na zakładce **Projektowanie**. Są one takie same jak używane do budowy formularzy (rys. 15.9).



Rys. 15.9. Widok formantów do projektowania raportu na karcie **Projektowanie**

Podstawowe narzędzia w oknie projektu raportu to ikonki **Arkusz właściwości** oraz **Dodaj istniejące pola** (rys. 15.10).



Rys. 15.10. Ikonki na karcie **Projektowanie** do zdefiniowania źródła rekordów raportu oraz wybrania pól

Tak samo jak i przy tworzeniu formularza, w projekcie raportu najpierw należy zdefiniować źródło rekordów, zapisując go na karcie **Dane** w **Arkuszu właściwości** raportu. Potem można umieścić w projekcie pola, korzystając z ikonki **Dodaj istniejące pola**.

Najczęściej raport tworzy się za pomocą kreatora, a potem dostosowuje się go w widoku układu oraz widoku projektu.

## Zadanie 15\_4

Utwórz raport o nazwie **r15\_4** do wyświetlenia danych o wpłatach każdego studenta za rok akademicki 2016/2017. Dane należy wyświetlić w postaci: *Lp.*, *Album*, *Nazwisko*, *Imiona*, *Data\_wpłaty*, *Wpłata*, gdzie kolumna *Lp.* będzie zawierała liczby porządkowe. Wyświetli w raporcie sumy wpłat każdej osoby.

---

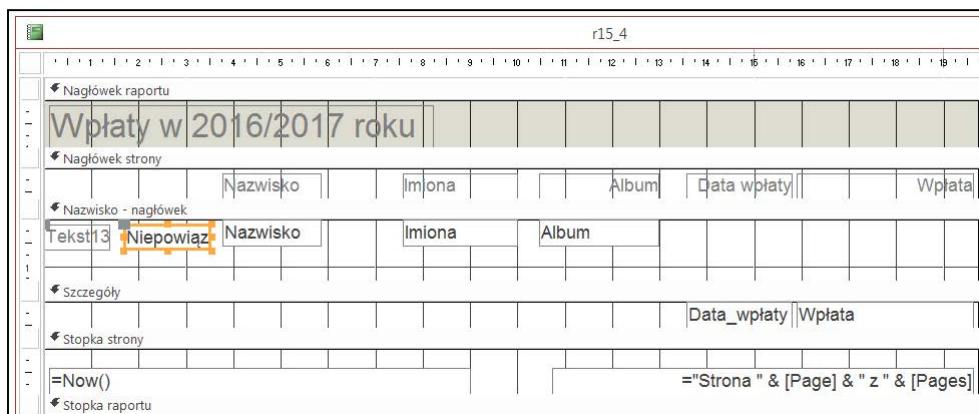
### Wykonanie

- a. Utwórz kwerendę wybierającą o nazwie **k15\_2** na podstawie tabel **tOsoba** i **tWpłaty**, zamieszczając w kwerendzie pola: *Nazwisko*, *Imiona*, *Album*, *Data\_wpłaty*, *Za\_rok\_akad*, *Wpłata* oraz definiując kryterium dla pola *Za\_rok\_akad*.
- b. Za pomocą kreatora raportów utwórz raport na bazie kwerendy **k15\_2**:
  - o umieść w raporcie pola kwerendy: *Nazwisko*, *Imiona*, *Album*, *Data\_wpłaty*;
  - o wybierz **Dalej; Dalej; Dalej**;
  - o ustaw sortowanie według pola *Data\_wpłaty*;
  - o wybierz **Dalej**;
  - o wybierz układ **Krokowy**;
  - o **Orientacja – Pionowa**;
  - o wybierz **Dalej**;
  - o zapisz raport, nadając mu nazwę: „Wpłaty w roku 2016/2017”;
  - o **Zakończ**.
- c. Wykonany za pomocą kreatora raport będzie wyświetlony w **Widoku podglądu wydruku** (rys. 15.11). Zamknij raport i w okienku nawigacji zmień nazwę raportu, ustawiając ją jako **r15\_4**.
- d. Otwórz raport w widoku układu.
- e. Zwolnij miejsce dla liczby porządkowej – można, na przykład wykonać to tak: najpierw zaznaczyć kolumnę *Nazwisko*, zweźć ją, a następnie za pomocą strzałki klawiatury > przemieścić całą kolumnę w prawo; tak samo można postąpić z kolumną *Imiona*.
- f. Przełącz się do widoku projektu.
- g. Dodaj pole obliczeniowe do wyświetlania liczby porządkowej każdej osoby. W tym celu:
  - o odciagnij w dół pasek **Szczegóły** i umieść w projekcie formant **Pole tekstowe**; w przykładzie przedstawionym na rysunku 15.12 jest to pole tekstowe o nazwie **Tekst 13**;

The screenshot shows a Microsoft Access report titled "Wpłaty w 2016/2017 roku". The report displays payment details for two individuals: Krzysztof Gołecki and Paweł Kulesza. The data is presented in a table format with columns for Name, Surname, Album, Date of payment, and Payment amount.

Nazwisko	Imiona	Album	Data wpłaty	Wpłata
Gołecki	Krzysztof	88900	2016-10-03 2016-11-05 2016-12-12	350,00 zł 350,00 zł 350,00 zł
Kulesza	Paweł	34567	2016-10-22 2016-11-23 2016-12-20	300,00 zł 300,00 zł 300,00 zł

Rys. 15.11. Pierwsza wersja raportu **r15\_4** wykonana za pomocą **kreatora raportów**



Rys. 15.12. Projekt raportu **r15\_4** po umieszczeniu w nim pola tekstowego

- w polu etykiety pola tekstu zamiast **Tekst 13** wprowadź: „Lp.” (bez znaku cudzysłowu);
- przemieśc tę etykietę do sekcji **Nagłówek strony**, jak to pokazano na rysunku 15.12 (aby oddzielić etykietę, kliknij na niej dwukrotnie i przeciągnij, utrzymując za lewy górny róg);
- przemieśc pasek **Szczegóły** na stare miejsce;
- zaznacz część **Niepowiązany** pola tekstu **Tekst 13**, a następnie otwórz jego **Arkusz właściwości**;
- kliknij kartę **Dane**;
- w wierszu **Źródło formantu** wpisz: **=1**;
- w polu **Suma bieżąca** wybierz wartość: **W grupie**.
- kliknij kartę **Format**;

- o w wierszu **Format** wpisz: **#.** – spowoduje to sformatowanie numeru wiersza oraz dodanie po nim kropki;
- o zamknij arkusz właściwości; odpowiednia część projektu raportu będzie miała postać jak na rysunku 15.13;

<b>Nagłówek strony</b>						
Lp.		Nazwisko	Imiona	Album	Data wpłaty	Wpłata

Rys. 15.13. Widok sekcji **Nagłówek strony** raportu **r15\_4**

<b>Nazwisko - nagłówek</b>			
=1	Nazwisko	Imiona	Album

Rys. 15.14. Fragment projektu raportu **r15\_4**

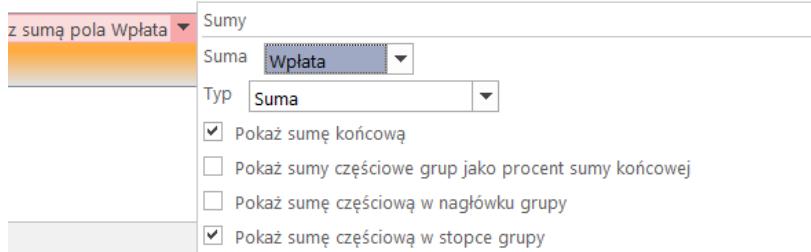
- o na karcie **Format** w wierszu **Styl obramowania** wybierz **Przezroczysty**;
- o popracuj nad wyglądem raportu: poprzesuwaj formanty, niektóre z nich scentruj, zmień styl tekstu w etykietach umieszczonych w sekcji **Nagłówek strony**; raport będzie miał postać podobną do przedstawionej na rysunku 15.15.

Wpłaty w 2016/2017 roku						
Lp.	Nazwisko	Imiona	Album	Data wpłaty	Wpłata	
1.	Golebiecki	Krzysztof	88900	2016-10-03	350,00 zł	
				2016-11-05	350,00 zł	
				2016-12-12	350,00 zł	
2.	Kulesza	Paweł	34567	2016-10-22	300,00 zł	
				2016-11-23	300,00 zł	

Rys. 15.15. Raport **r15\_4** w widoku **raportu** po umieszczeniu w nim kolumny **Lp.**

- #### h. Dodaj w raporcie sumy wpłat każdej osoby. W tym celu:
- o otwórz raport **r15\_4** w **widoku projektu**;
  - o wybierz na karcie **Projektowanie** ikonkę **Grupuj i sortuj** – w dolnej części okna projektu zostanie wyświetlona lista opcji grupowania;
  - o kliknij przycisk **Więcej**;

- o rozwiń listę **bez sum** i ustaw opcję, jak to pokazano na rysunku 15.15.
- o ustaw opcję **zachowaj całą grupę na jednej stronie**.



Rys. 15.16. Zdefiniowanie opcji sumowania

- i. Umieść **Etykietę „Suma”** przed sumą częściową, a **etykietę „Razem”** – przed sumą końcową (rys. 15.16).
- j. Wybierz wśród narzędzi projektowania formant **Linia** i umieść go, utrzymując klawisz SHIFT, na stopce przed sumą częściową w celu podkreślenia sumy wpłat jednej osoby (rys. 15.17).
- k. Zapisz raport.

Na rysunku 15.18 przedstawiono raport **r15\_4** w widoku raportu.

Rys. 15.17. Projekt raportu **r15\_4**

Wpłaty w 2016/2017 roku					
Lp.	Nazwisko	Imiona	Album	Data wpłaty	Wpłata
1.	Gołeckiecki	Krzysztof	88900	2016-10-03	350,00 zł
				2016-11-05	350,00 zł
				2016-12-12	350,00 zł
					Suma: 1 050,00 zł
2.	Kulesza	Paweł	34567	2016-10-22	300,00 zł
				2016-11-23	300,00 zł
				2016-12-20	300,00 zł
					Suma: 900,00 zł

Rys. 15.18. Raport **r15\_4** w widoku raportu

## Zadania do samodzielnego wykonania

### Zadanie 15\_5

Utwórz raport do wyświetlenia danych o zakwaterowaniach każdego studenta. Dane należy wyświetlić w postaci: *Lp.*, *Album*, *Nazwisko*, *Imię*, *Data zakwaterowania*, *Data wykwaterowania*, *Wydział* i pogrupować tak samo, jak zrobiono to w zadaniu **15\_4**.

---

### Zadanie 15\_6

Utwórz raport do wyświetlenia danych o zakwaterowaniach studenta o zadanym w okienku dialogowym numerze albumu. Dane należy wyświetlić w postaci: *Lp.*, *Nazwisko*, *Imię*, *Album*, *Data zakwaterowania*, *Data wykwaterowania*, *Wydział*.

---

# 16. Makra

Makra służą do tworzenia aplikacji bazodanowych. Prosta aplikacja kliencka często może być zbudowana z zastosowaniem wyłącznie narzędzi graficznych programu (w tym różnych kreatorów) oraz makr. W tym przypadku język **VBA** może wcale nie być stosowany, ponieważ makra realizują część z jego instrukcji. Przy tym stworzenie makra często jest o wiele prostszym zadaniem dla użytkownika niżeli napisanie kodu **VBA**.

Ze względu na funkcjonalność możemy podzielić makra na takie, które działają w tabelach (makra danych), oraz na takie, które wykorzystywane są w formularzach i raportach (makra interfejsu użytkownika).

## 16.1. Makra interfejsu użytkownika

Najłatwiej przedstawić sobie makro interfejsu użytkownika jako zbiór poleceń, które można uruchomić jednym kliknięciem.

Makra interfejsu użytkownika mogą być częściami obiektów lub kontrolek. Takie makra nazywane są osadzonymi. Drugi typ makr, nazywanych autonomicznymi, to makra występujące jako samodzielne obiekty. W odróżnieniu od osadzonych autonomiczne makra przedstawione są w okienku nawigacji.

Każde makro składa się z co najmniej jednej akcji makra. Lista możliwych akcji makra jest w programie **Microsoft Access** z góry ustalona i widoczna w oknie projektu makra.

### *Makra osadzone*

Makro osadzone zawsze jest częścią formularza lub raportu. Możemy samodzielnie je utworzyć. Wtedy działamy na tych samych zasadach co i przy tworzeniu makra autonomicznego, jedynie umieszczamy go we właściwościach zdarzenia obiektu, a nie w okienku nawigacji.

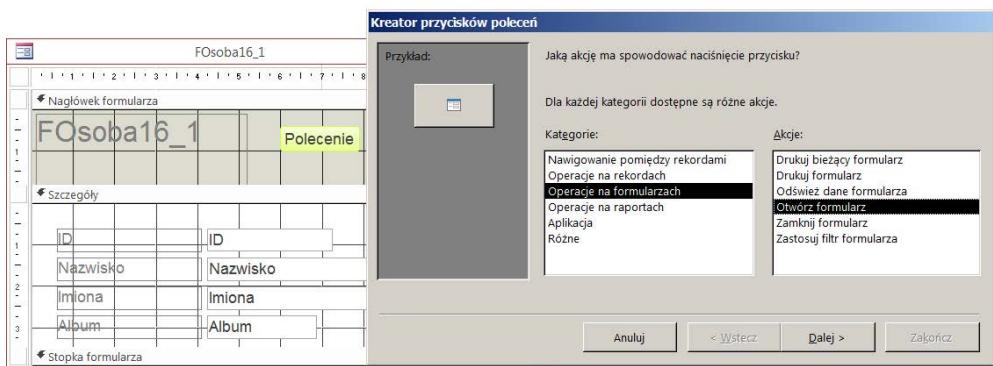
Makro osadzone często generowano w tle przez program, gdy używano różnego rodzaju kreatorów. Kolejny przykład przedstawia takie makro osadzone, które powstaje przy umieszczeniu w formularzu przycisku polecenia przeznaczonego do otwierania innego formularza.

## Zadanie 16\_1

Utwórz formularz kolumnowy o nazwie **fOsoba16\_1** do wyświetlenia danych z pól *ID*, *Nazwisko*, *Imiona*, *Album* tabeli *tOsoba*. Utwórz formularz kolumnowy o nazwie **fZakwaterowanie16\_1** do wyświetlenia wszystkich pól z tabeli *tZakwaterowanie*. Umieść w nagłówku formularza **fOsoba16\_1** formant **Przycisk**. Powinien on służyć do otwierania formularza **fZakwaterowanie16\_1** i wyświetlenia danych o zakwaterowaniu tej osoby, której dane osobowe aktualnie przedstawia formularz **fOsoba16\_1**. Obejrzyj utworzone makro osadzone w **Arkuszu właściwości przycisku** (zdarzenie **Przy kliknięciu**).

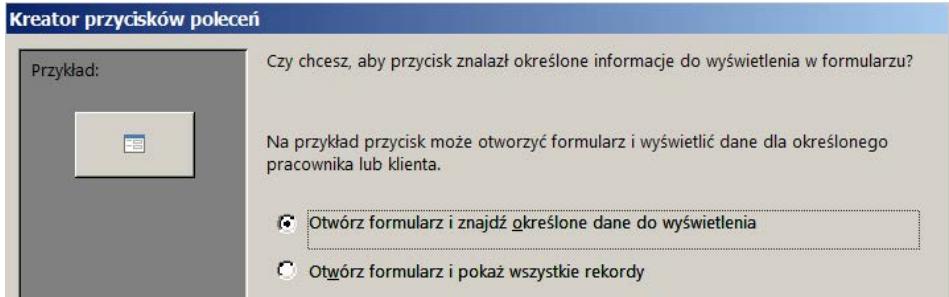
### Wykonanie

- Zbuduj formularze **fOsoba16\_1** oraz **fZakwaterowanie16\_1** w oparciu o wiedzę nabycą w rozdziałach 3, 5, 14.
- Umieść formant **Przycisk** w nagłówku formularza **fOsoba16\_1**. Przy tym w oknie **Kreatora przycisków poleceń** wybierz kategorię **Operacje na formularzach** oraz akcję **Otwórz formularz** (rys. 16.1).

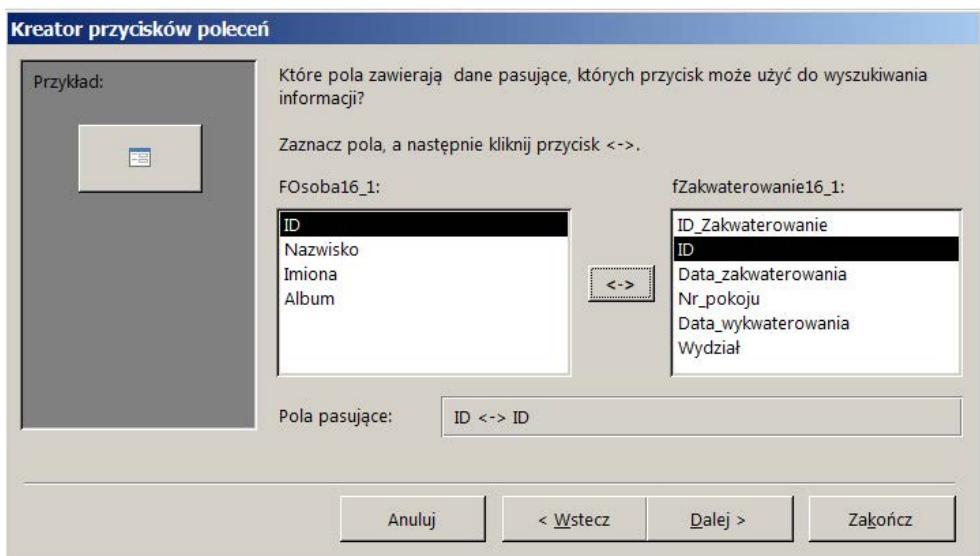


Rys. 16.1. Określenie kategorii i akcji w oknie **Kreatora przycisków poleceń**

- W kolejnym oknie **Kreatora** wybierz nazwę formularza **fZakwaterowanie16\_1**.
- Dalej możemy zarządzić, aby otwierany za pomocą przycisku formularz wyświetlał nie wszystkie dane z tabeli *tZakwaterowanie*, a tylko dotyczące konkretnej osoby. W tym celu zaznaczamy opcję **Otwórz formularz i znajdź określone dane do wyświetlenia** (rys. 16.2).
- Kolejne okno pozwala na oznaczenie tego, że pole *ID* powinno służyć do połączenia tabel (rys. 16.3). Potem zostanie tylko zdecydować, jak będzie wyglądał przycisk w oknie formularza, i zakończyć pracę z **Kreatorem**.



Rys. 16.2. Definiowanie zasady wyświetlania danych w formularzu



Rys. 16.3. Okno **Kreatora przycisków poleceń** w zaznaczeniu pasujących do siebie pól przy nawiązywaniu połączenia między tabelami

f. Przejdz do widoku formularza i przetestuj działanie utworzonego przycisku.

Po sprawdzeniu działania przycisku obejrzyj makro osadzone utworzone w tle działania **Kreatora**. W tym celu:

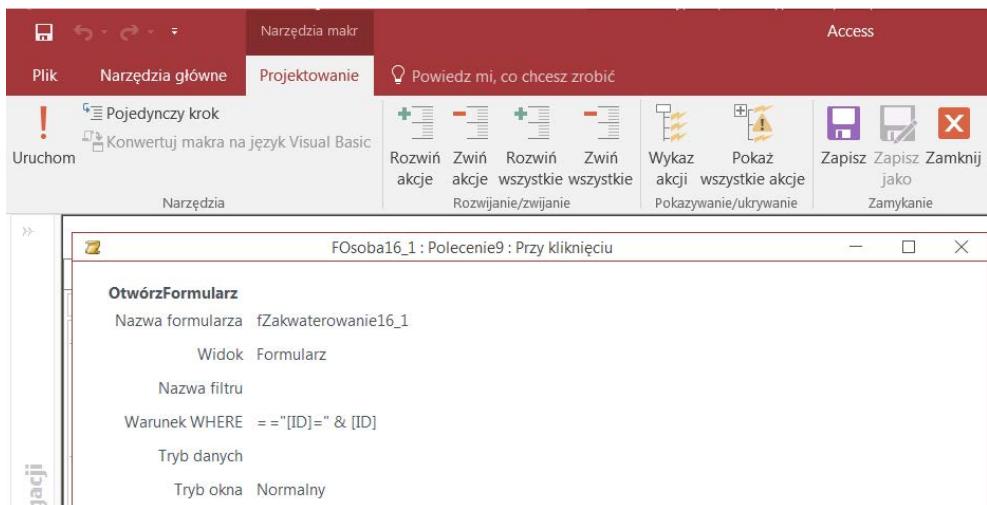
- o w oknie projektu formularza **fOsoba16\_1** zaznacz formant **Przycisk** i otwórz jego **Akusz właściwości**;
- o przejdź do zakładki **Zdarzenie** – obok zdarzenia **Przy kliknięciu** zobaczysz wpis: **[Makro osadzone]** (rys. 16.4),

- o kliknij na przycisku oznaczonym trzema kropkami, a zostanie otwarte okno projektu utworzonego makra osadzonego. Treść makra przedstawia rysunek 16.5.



Rys. 16.4. Widok zakładki **Zdarzenie** w Arkuszu właściwości formantu **Przycisk**

Nazwa formantu **Przycisk** także była utworzona przez **Kreatora**. Na rysunku 16.5 widzimy ją w nagłówku makra: **Polecenie 9**. Dobrą praktyką jest nadanie formantowi własnej nazwy zamiast tej nic nie znaczącej.



Rys. 16.5. Okno projektu makra, a w nim makro osadzone wygenerowane przez **Kreator przycisków poleceń**

### Makra autonomiczne

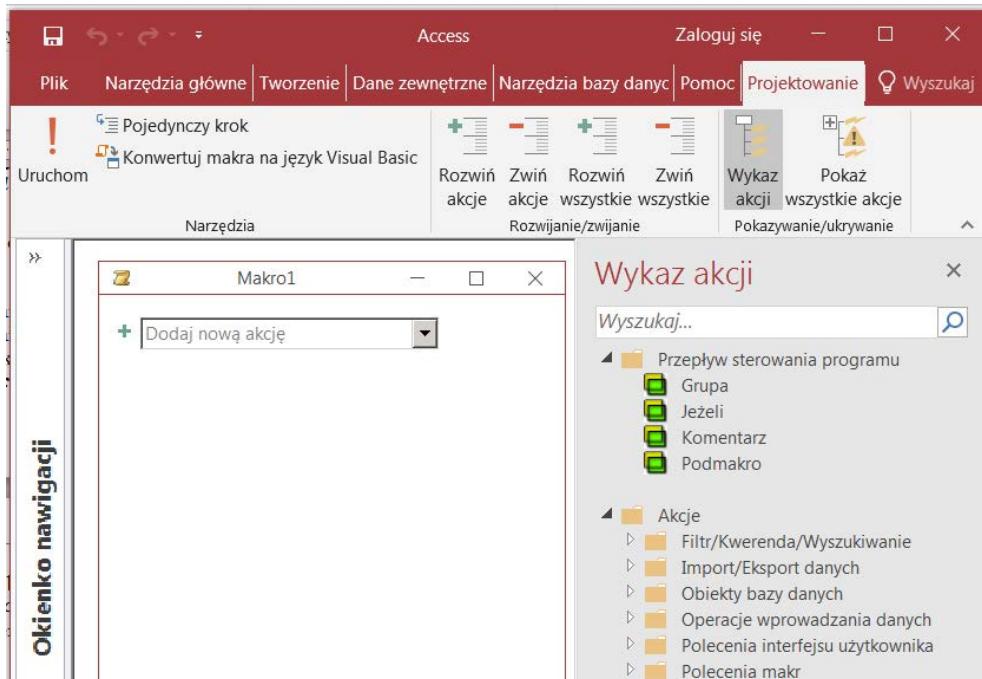
Tworzenie makra autonomicznego zaczynamy od wybierania ikonki



w karcie **Tworzenie** wstążki. Zostanie otwarte okno projektu makra (rys. 16.6).

Dalej należy wybrać akcję makra do zrealizowania postawionego zadania. Akcje możemy wybierać, klikając na odpowiedniej nazwie w okienku z **Wykazem akcji** lub bezpośrednio z listy rozwijanej pola kombi, w którym widzimy napis: *Dodaj nową akcję*.

Proces tworzenia makra autonomicznego najlepiej zaprezentować na przykładzie.



Rys. 16.6. Okno projektowania makra

## Zadanie 16\_2

Utwórz za pomocą kreatora formularz tabelaryczny w oparciu o pola: *ID*, *Nazwisko*, *Imiona*, *Album* tabeli **tOsoba**. Nazwij formularz **fOsoba16\_2**.

Zbuduj makro o nazwie **M1** do filtrowania danych: wyświetlenia wszystkich rekordów tabeli **tOsoba** zawierających nazwiskaaczynające się od litery lub liter wskazanych przez użytkownika. Na rysunku 16.7 przedstawiono taki formularz w momencie wprowadzenia danych przez Użytkownika.

Umieść w formularzu przycisk **Filtruj według nazwiska** do uruchomienia makra **M1**.

Utwórz również makro o nazwie **M2** do skasowania filtru i wyświetlenia danych ze wszystkich rekordów tabeli **tOsoba**.

Umieść w formularzu przycisk **Skasuj filtr** do uruchomienia makra **M2**.

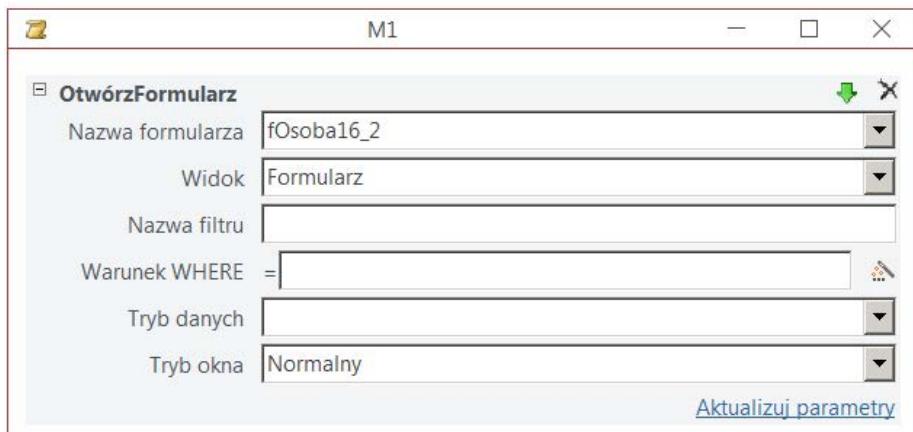
## Wykonanie

ID	Nazwisko	Imiona	Album
1	Kulesza	Paweł	34567
2	Poniatowski		
3	Gołeć		
4	Kowalewski		
5	Socha		
6	Molski	Adam	90919
7	Kowalski	Adam	234567

Widok formularza fOsoba16\_2

Rys. 16.7. Widok formularza fOsoba16\_2

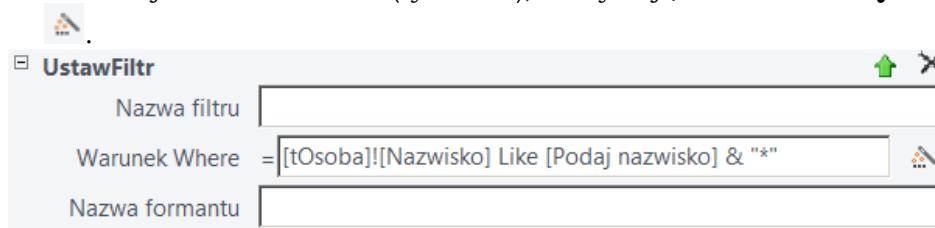
- Wybierz w zakładce **Tworzenie** ikonkę **Makro** – otworzy się okno projektu makra (rys. 16.6).
- Rozwiń listę **Dodaj nową akcję** i wybierz polecenie **Otwórz formularz**.
- Wybierz nazwę formularza – **fOsoba16\_2** (rys. 16.8).



Rys. 16.8. Okno projektu makra M1, akcja OtwórzFormularz

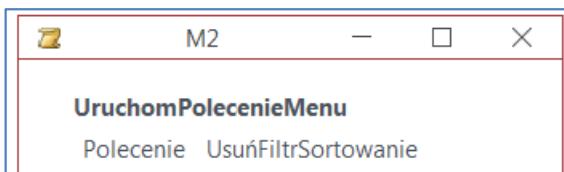
- Zdefiniuj kolejną akcję, wybierając **Ustaw filtr**.

- e. Zdefiniuj warunek **Where** (rys. 16.9), korzystając z **Kreatora wyrażeń**



Rys. 16.9. Okno projektu makra M1, akcja **UstawFiltr**

- f. Zapisz makro, nadając mu nazwę **M1**.
- g. Otwórz formularz **fOsoba16\_2** w widoku projektu.
- h. Umieść w nim formant **Przycisk**, wybierając w oknie **Kreatora poleceń: Kategoria – Różne, Akcja – Uruchom makro**, a potem nazwę makra – **M1**.
- i. Utwórz makro **M2** do skasowania filtru i wyświetlenia wszystkich rekordów. Projekt makra przedstawia rysunek 16.10.



Rys. 16.10. Okno projektu makra M2

- j. Umieść w formularzu drugi przycisk, który będzie służył do uruchomienia makra **M2**.

Kolejne zadanie prezentuje wykonanie makra do wyświetlenia wybranych fragmentów raportu.

### Zadanie 16\_3

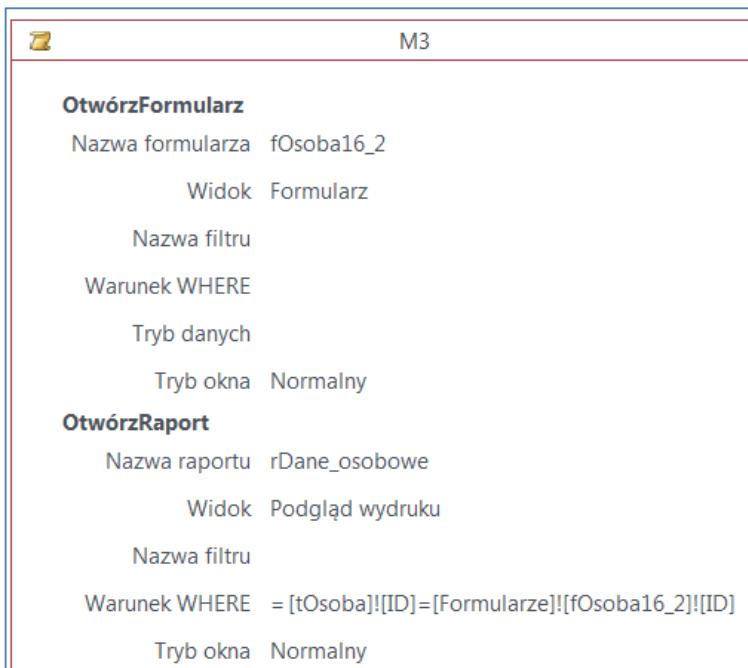
Utwórz za pomocą kreatora raport kolumnowy do wyświetlenia z tabeli **tOsoba** wszystkich pól oprócz pola *Dok.* Nazwij raport **rDane\_osobowe**.

Zbuduj makro o nazwie **M3** do otwierania raportu **rDaneOsobowe** w widoku podglądu wydruku. Raport powinien wyświetlać dane wyłącznie tej osoby, której dane w tym momencie rozważane w formularzu.

Umieść w formularzu **fOsoba16\_2** jeszcze jeden przycisk, który po kliknięciu na nim uruchomi makro **M3**.

## Wykonanie

To zadanie wykonujemy, działając analogicznie do poprzednich zadań, lecz makro **M3** będzie różnić się od makr **M1** i **M2**. Widok projektu makra **M3** prezentuje rysunek 16.11.



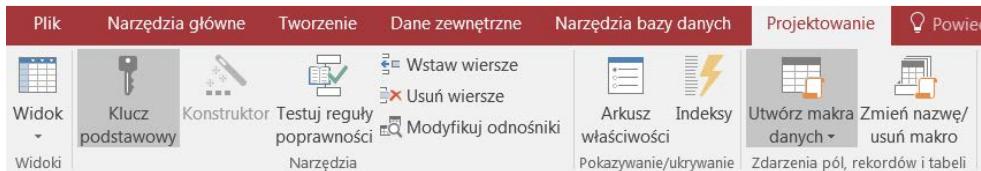
Rys. 16.11. Okno projektu makra **M3**

## 16.2. Makra danych

Makra danych nie są widoczne w obszarze **Makra** okienka nawigacji. Takie makra zawsze tworzone są dla pewnej tabeli. Odpowiednio narzędzia do ich utworzenia – ikonki na wstążce – stają się dostępne dopiero po otworzeniu tej tabeli. Przy tym jeśli tabela jest otwarta w widoku projektu, to na wstążce wybieramy ikonkę **Otwórz makra danych** (rys. 16.12), a potem odpowiednie polecenie. Gdy tabela jest otwarta w arkuszu danych, to wszystkie polecenia przedstawione są oddzielnymi ikonkami (rys. 16.13).

Rozróżniane są **makra danych wywoływane przez zdarzenia** występujące w tabelach (takie jak dodawanie, aktualizowanie i usuwanie danych) oraz makra uruchomiane w odpowiedzi na podanie ich nazwy

**(nazwane makra danych).** Niżej podano przykłady utworzenia makr obu typów.



Rys. 16.12. Widok ikonki do tworzenia makr danych w zakładce **Projektowanie** na wstążce



Rys. 16.13. Widok ikonek do tworzenia makr danych w zakładce **Tabela** na wstążce

### Tworzenie makra danych uruchomianego przez zdarzenie

Według dokumentacji programu *Microsoft Access* wystąpienie zdarzenia w tabeli jest związane z dodawaniem, aktualizacją lub usuwaniem danych w tej tabeli. Makro danych można zaprogramować tak, aby było uruchamiane natychmiast po dowolnym z tych trzech zdarzeń lub bezpośrednio przed zdarzeniem usunięcia bądź zmiany.

Jeśli otworzymy dowolną tabelę w widoku arkusza danych i wybierzymy na wstążce kartę **Tabela**, to zobaczymy między innymi dwie grupy ikon: **Zdarzenia przed** oraz **Zdarzenia po** (rys. 16.12). Każda z ikon służy do opracowania jednego z wymienionych powyżej zdarzeń. Na przykład ikonka **Po usunięciu** może być wykorzystana przy tworzeniu makra danych, które będzie uruchamiane po usunięciu rekordu z tabeli.

W celu zapoznania się z makrem danych zaproponowano do wykonania **Zadanie 16\_4**. Wykonując je, zrozumiesz, jak zbudowane jest makro reagujące na wprowadzenie wartości do jednego z pól już istniejącego rekordu tabeli.

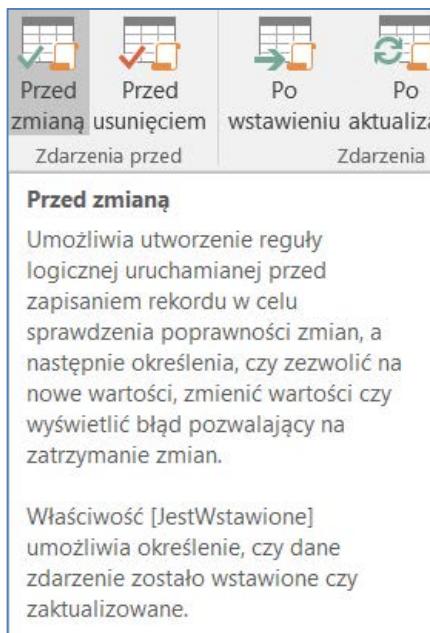
## Zadanie 16\_4

Utwórz makro danych, które będzie działać w trakcie aktualizacji rekordu tabeli **tZakwaterowanie**, gdy wprowadzana jest

wartość do pola *Data\_wykwaterowania* w już istniejącym rekordzie. Makro powinno sprawdzać, czy wprowadzana wartość do pola *Data\_wykwaterowania* tabeli **tZakwaterowanie** nie jest mniejsza lub równa wartości w polu *Data\_zakwaterowania* tegoż rekordu. Makro powinno nie pozwolić na wprowadzenie takiej wartości, a jednocześnie wywołać odpowiedni komunikat o błędzie.

## Wykonanie

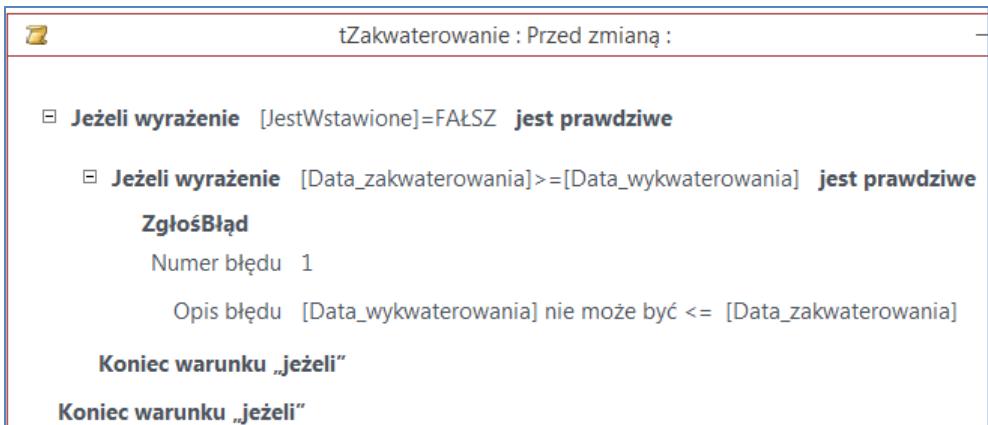
- Otwórz tabelę **tZakwaterowanie** w widoku arkusza danych. Przejdź na zakładkę **Tabela** (rys. 16.13).
- Wybierz ikonę **Przed zmianą**. Utrzymując na niej wskaźnik myszy, wywołasz krótki opis tego polecenia (rys. 16.14).



Rys. 16.14. Widok opisu polecenia **Przed zmianą**

Zwrócić uwagę na opis właściwości **JestWstawione**, która służy dla określenia, czy dodawany jest nowy rekord danych, czy aktualizujemy wartość w już istniejącym rekordzie.

- Po kliknięciu na ikonę **Przed zmianą** zostanie otworzone okno konstruktora makr, do którego wprowadź tekst makra, tak jak pokazano to na rysunku 16.15.



Rys. 16.15. Okno konstruktora makr wywołane poleceniem **Przed zmianą**

W przedstawionym na rysunku 16.15 makrze zawarte są dwa zagnieżdżone polecenia **Jeżeli wyrażenie**.

W pierwszym poleceniu **Jeżeli wyrażenie** sprawdzane jest, czy odbywa się proces aktualizacji tabeli **tZakwaterowanie**, gdy właściwość **JestWstawiono** powinna mieć wartość FAŁSZ.

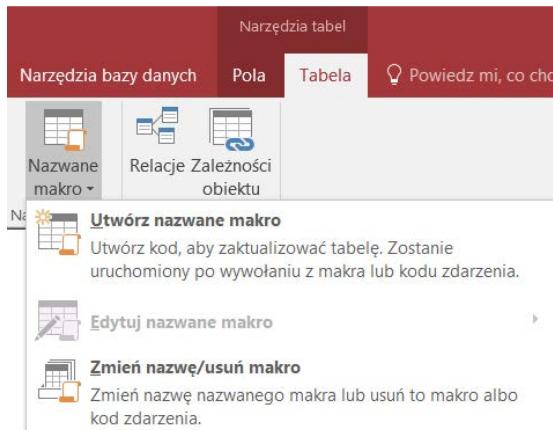
Kolejne polecenie **Jeżeli wyrażenie** sprawdza, czy wprowadzona przez użytkownika wartość w polu *Data\_wykwaterowania* jest większa lub równa wartości *Data\_zakwaterowania*. Jeśli tak nie jest, wykonywane jest polecenie **ZgłośBłąd**, w którym numer błędu jest dowolnie wymyślony przez twórcę makra. W polu **Opis błędu** dodano tekst wyświetlanego komunikatu.

### *Tworzenie nazwanego makra danych*

Makro danych może być wywołane nie tylko przez zdarzenie. Nazwane (lub „autonomiczne”) makro jest skojarzone z określoną tabelą i można je wywoływać z poziomu dowolnego innego makra danych lub makra standardowego.

Do pracy z nazwanymi makrami służy ikonka **Nazwane makro** (rys. 16.16), która jest ulokowana na karcie **Tabela** w sytuacji, gdy jakąś tabelę jest otwarta w widoku arkusza danych. Odpowiednie polecenie można znaleźć na wstążce także w widoku projektu tabeli.

Wykonując **zadanie 16\_5**, zrozumiesz mechanizm działania nazwanego makra.



Rys. 16.16. Ikonka do tworzenia ***nazwanego makra*** w zakładce ***Tabela*** na wstążce

## Zadanie 16\_5

Utwórz tabelę **tNazwiska** do przechowywania starych i nowych nazwisk tych osób, które zmieniły swoje nazwisko.  
 Utwórz nazwane makro danych, które przy zmianie wartości w polu *Nazwisko* tabeli **tOsoba** będzie zapisywało stare i nowe nazwiska osoby do tabeli **tNazwiska**.

### Wykonanie

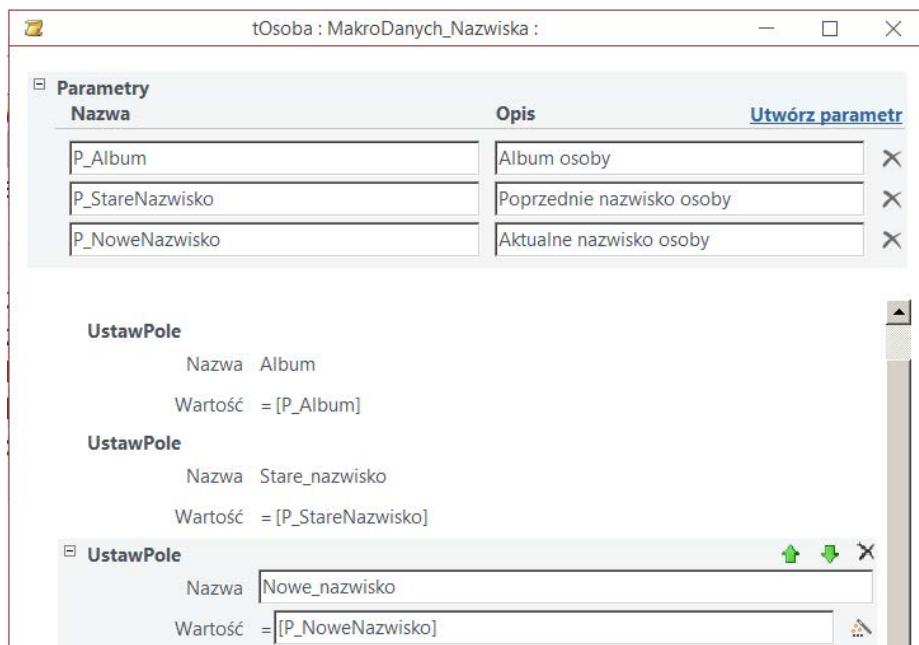
- Utwórz tabelę o nazwie **tNazwiska** w postaci przedstawionej na rysunku 16.17.

		Nazwa pola	Typ danych
	ID_nazwiska		Autonumerowanie
	Album		Liczba
	Stare_nazwisko		Krótki tekst
	Nowe_nazwisko		Krótki tekst

Rys. 16.17. Tabela ***tNazwiska*** w widoku projektu

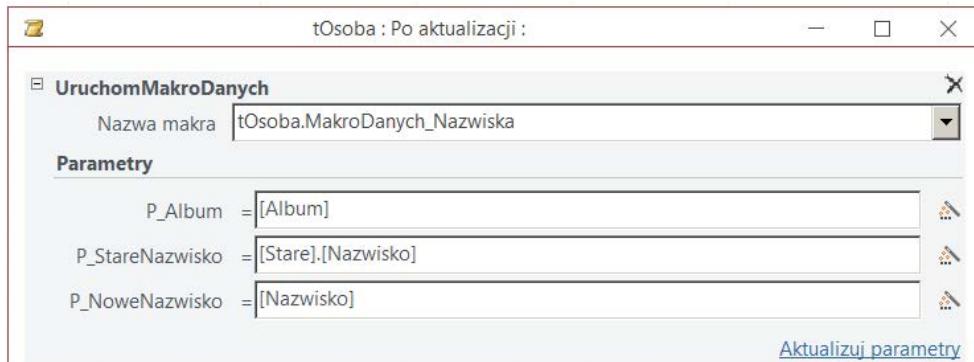
- Kliknij dwukrotnie na nazwie tabeli **tOsoba**.
- Na karcie **Tabela** wstążki kliknij na ikonę **Nazwane makro** i wybierz polecenie **Utwórz nazwane makro**.

- d. Kliknij na napisie **Utwórz parametr** i wprowadź parametr *P\_Album* (rys. 16.18). Jest niezbędny w celu jednoznacznej identyfikacji osoby w tabeli **tNazwisko**.
- e. Analogicznie stwórz jeszcze dwa parametry *P\_StareNazwisko* oraz *P\_NoweNazwisko* do przechowywania poprzedniego i kolejnego nazwiska osoby. Przez to, że tabela **tNazwiska** zawiera pole *ID\_nazwiska*, można będzie zapisywać zmiany nazwisk tej samej osoby niejednokrotnie.
- f. Dalej w makrze należy umieścić polecenie **Utwórz rekord** w celu utworzenia odpowiedniego rekordu w tabeli **tNazwisko** oraz polecenia **UstawPole** (rys. 16.18).
- g. Wybierz na wstążce ikonkę **Zapisz**. Domyślna nazwa to **MakroDanych1**. Można ją zmienić na **MakroDanych\_Nazwiska**.
- h. Zamknij okno makra.



Rys. 16.18. Projekt nazwanego makra danych **MakroDanych\_Nazwiska**

- i. Wybierz na wstążce w karcie **Tabela** ikonkę **Po aktualizacji**. Utwórz makro do uruchomienia makra danych tak, jak przedstawia rysunek 16.19.  
Przy wpisaniu wartości parametru *P\_StareNazwisko* należy skorzystać z właściwości *[Stare]*, którą dysponuje oprogramowanie.



Rys. 16.19. Makro danych do uruchomienia nazwanego makra danych o nazwie **MakroDanych\_Nazwiska**

- j. Zapisz i zamknij makro. Przetestuj jego działanie. W tym celu zmień jedno z nazwisk w tabeli **tOsoba**. Jeśli makro działa prawidłowo, to w tabeli **tNazwiska** pojawi się odpowiedni rekord (na rys. 16.20 wyświetlono arkusz danych tabeli **tNazwisko** po zamianie nazwiska „Walewska” na „Sikorska”).

TNazwiska				
ID_nazwiska	Album	Stare_nazwisko	Nowe_nazwisko	
1	921213	Walewska	Sikorska	
*	(Nowy)			

Rys. 16.20. Makro danych **Po aktualizacji** tabeli **tOsoba**

### 16.3. Makro AutoExec

Program **Access** nadaje możliwość automatycznego uruchomienia makra o nazwie **AutoExec** w momencie utworzenia bazy danych. Jest to zwykłe makro autonomiczne, które zostaje wykonane, zanim zostaną uruchomione jakiekolwiek inne makra i kod języka **VBA**. Więc jeśli chcemy, aby jakieś makro wykonywało się przy uruchomieniu pliku bazy danych, wystarczy nadać temu makru nazwę **AutoExec**.

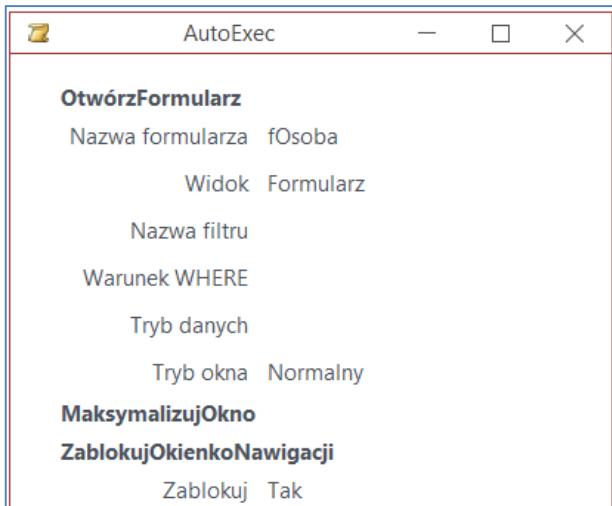
### Zadanie 16\_6

W bazie danych **Dom Studenta** utwórz makro o nazwie **AutoExec**, które otworzy przy uruchomieniu programu jeden ze zbudowanych wcześniej formularzy, przykładowo **fOsoba**. Powiększ maksymal-

nie rozmiar okna formularza. Jednocześnie zablokuj **Okienko Nawigacji**, aby nie było możliwości usunięcia z niego któregokolwiek z obiektów.

## Wykonanie

- a. Utwórz makro autonomiczne, widok projektu którego przedstawia rysunek 16.21.



Rys. 16.21. Przykład makra **AutoExec**

- b. Zapisz makro, nadając mu nazwę **AutoExec**.  
c. Zamknij i otwórz ponownie bazę danych w celu testowania wykonanego makra.

Należy tu zauważyć, że makro nie będzie uruchamiane automatycznie, jeśli przy kliknięciu na nazwie bazy danych w celu jej otwarcia naciśnięty jest także klawisz SHIFT.

## Zadania do samodzielnego wykonania

### Z a d a n i e 1 6 \_ 7

Wykorzystując wykonane formularze i raporty lub tylko wzorując się na nich, utwórz przyjazną dla użytkownika aplikację klienta bazy danych Dom Studenta. Powinna ona pozwalać na

wprowadzenie, aktualizację, usunięcie oraz przegląd danych, a także na wykonanie raportów o zamieszkujących w bieżącym roku osobach i ich wpłatach za zamieszkanie.

---

### Zadanie 16\_8

Utwórz makro danych, które przy dodaniu nowego rekordu do tabeli **tOsoba** nie pozwoli wprowadzić wartości albumu mniejszej od 3000.

---

### Zadanie 16\_9

Dodaj do tabeli **tNazwisko** pole *Data* i uzupełnij makro danych oraz makro **Po aktualizacji** w celu przechowywania daty zmiany nazwiska w bazie danych. W tym celu skorzystaj z funkcji wbudowanej **DATE()**.

---

## V. Pierwsze kroki z programem Microsoft SQL Server Management Studio

*Microsoft SQL Server (MS SQL)* – to relacyjny system zarządzania bazami danych. Jest głównym produktem bazodanowym firmy *Microsoft*. Używany w nim dialekt języka zapytań ma nazwę **Transact SQL (T-SQL)**.

W systemie **MS SQL** można tworzyć, modyfikować i usuwać bazy danych za pomocą języka **Transact SQL**, jak również z poziomu interfejsu programu **Microsoft SQL Server Management Studio**, który jest najważniejszy wśród narzędzi systemu.

Rozważmy dalej przykłady operacji wykonywanych w środowisku jednej z wersji **MS SQL** – oprogramowania **SQL Server 2014 Express With Advanced Services With Service Pack 1 32/64-bit (English)**.

Zakładamy, że oprogramowanie zostało zainstalowane w trybie autentyczacji *Windows (Windows Authentication)*, a wszystkie zaproponowane operacje będą wykonywane przez użytkownika z prawami administratora.

### 17. Utworzenie najprostszej bazy danych przy zastosowaniu narzędzi graficznych

#### 17.1. Uruchomienie programu MS SQL Server Management Studio

##### Zadanie 17\_1

Uruchom program **Microsoft SQL Server Management Studio**, a następnie połącz się z SQL serwerem zainstalowanym na twoim komputerze.

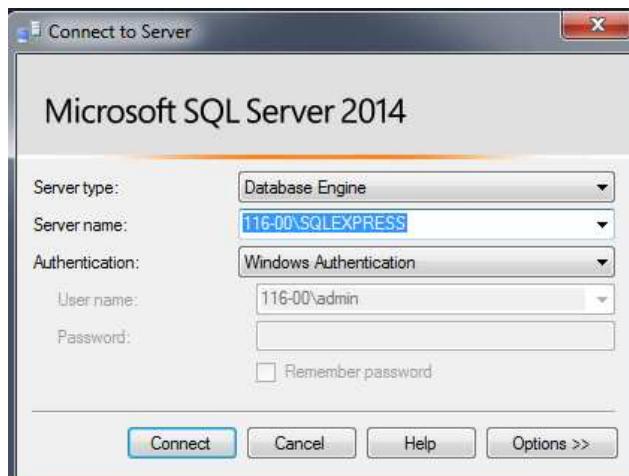
## Wykonanie

- a. Wybierz w menu **Start** kolejne polecenia: **Programy -> Microsoft SQL Server 2014 -> SQL Server 2014 Management Studio**. Na ekranie pojawi się pierwsze okno programu (rys. 17.1).

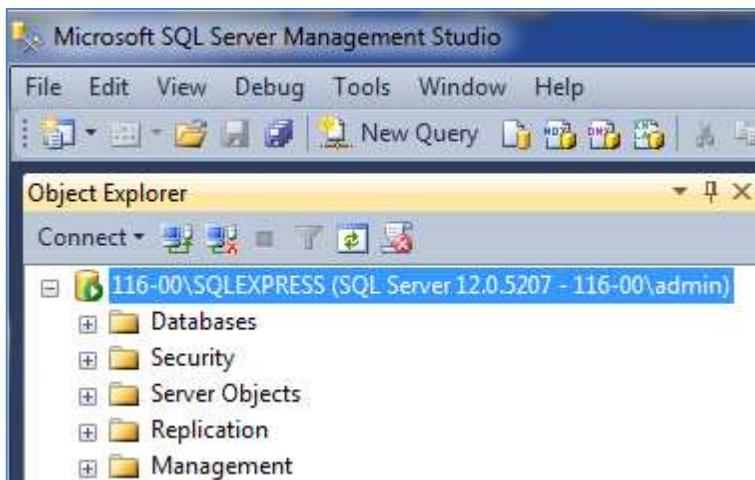


Rys. 17.1. Pierwsze okno programu

- b. Po pierwszym oknie na ekranie pojawia się drugie. Jest to okno dialogowe do połączenia z serwerem. Na rysunku 17.2 wyświetlono przykład takiego okna. W oknie dialogowym wybierz nazwę serwera w polu **Server name** i nacisnij przycisk połączenia **Connect** (na rysunku 17.2 nazwa serwera to 116-00\SQLEXPRESS, na twoim komputerze będzie inną nazwą).  
c. Jeśli połączenie zostanie wykonane pomyślnie, to na ekranie wyświetli się główne okno programu **Microsoft SQL Server Management Studio** (rys. 17.3).



Rys. 17.2. Przykładowe okno połączenia z **SQL Server 2014 Express**



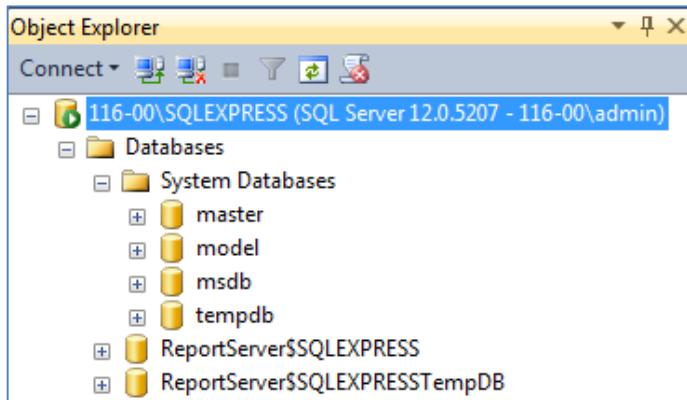
Rys. 17.3. Widok okna programu **Microsoft SQL Server Management Studio**

W górnej części okna programu **Microsoft SQL Server Management Studio** (rys. 17.3) widzimy pasek menu wraz z ikonami niektórych poleceń, a poniżej ulokowano okienko eksploratora obiektów **Object Explorer**. W nim przedstawiono ikonkę z nazwą serwera oraz obiekty serwera, a wśród nich węzeł **Databases**, którego elementy będą właśnie dalej rozwijane.

## 17.2. Utworzenie pustej bazy danych

System **SQL Server** może obsługiwać jednocześnie nie jedną, a wiele baz danych. Wszystkie zgrupowane są w węźle **Databases**. Systemowe bazy danych, które zawierają podstawowe pliki oprogramowania, zawarte są w węźle **System Databases**.

Na rysunku 17.4 w węźle **Databases** nie widać żadnej bazy danych stworzonej przez użytkownika. Są tylko systemowe bazy. Wśród nich **master** jest bazą główną, ponieważ zawiera całą konfigurację instancji serwera oraz informacje o pozostałych bazach oraz użytkownikach. Systemowa baza danych **model** służy jako wzorzec do tworzenia nowych baz danych. Każda nowa baza danych tworzona jest poprzez tworzenie kopii bazy **model**. **Tempdb** – tymczasowa baza danych – służy do tworzenia tymczasowych tabel i procedur. Wszystkie obiekty tej bazy żyją tylko do momentu ponownego uruchomienia serwera.



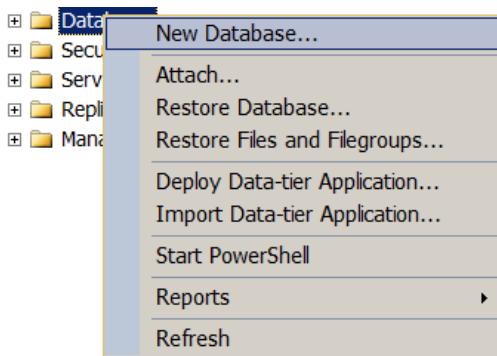
Rys. 17.4. Lista baz danych ulokowanych na serwerze

## Zadanie 17\_2

Stwórz nową pustą bazę danych o nazwie **MariaLisiecka**.

### Wykonanie

- Kliknij prawym wskaźnikiem myszki na węźle **Databases**, wywołując tym samym jego menu podrzczne (rys. 17.5).
- Wybierz w menu podrzcznym wezła **Databases** polecenie **New Database** – zostanie otwarte okno **New Database**.

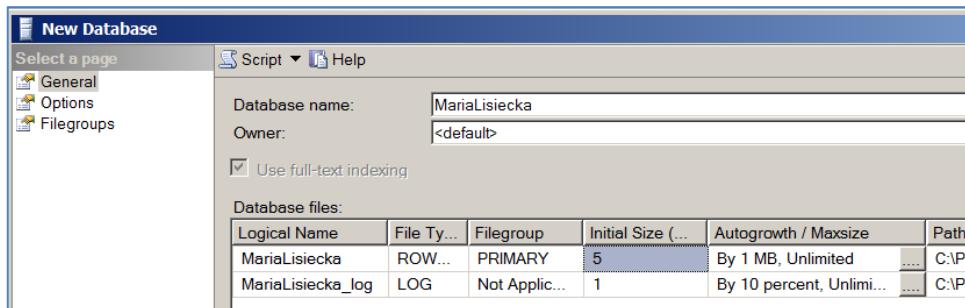


Rys. 17.5. Menu podrzczne węzła **Databases**

- Wpisz nazwę bazy danych w oknie **New Database** w wierszu **Database name** (rys. 17.6) i wybierz **OK**.
- Ponownie otwórz menu podrzczne węzła **Databases** (rys. 17.5).

e. Wybierz polecenie **Refresh** (tzn. **Odśwież**).

Na rysunku 17.6 widzimy opis plików, które składają się na bazę danych o nazwie **MariaLisiecka**. Nazwy i ścieżki do plików tworzą się automatycznie. Można je zmienić.



Rys. 17.6. Okno nowej bazy danych

Każda baza danych w **MS SQL Server** składa się co najmniej z dwóch plików. Musi w niej istnieć co najmniej jeden plik danych i co najmniej jeden plik dziennika transakcji. Wśród plików danych rozróżniamy podstawowy plik (**primary**) z rozszerzeniem **MDF** oraz pliki pomocnicze (**secondary**), które zwykle mają rozszerzenie **NDF**. Na początku wszystkie obiekty trzymane są w pliku podstawowym. W każdej bazie może istnieć tylko jeden plik podstawowy. Pliki pomocnicze zawierają wszystkie obiekty, które nie mieścią się w podstawowym pliku bazy danych. W każdej bazie danych może istnieć wiele plików pomocniczych, jednak w niektórych bazach nie ma potrzeby ich stosowania.

Oprócz plików **MDF** i **NDF** baza danych zawiera pliki dzienników mające rozszerzenie **LDF**. Każda baza danych ma przynajmniej jeden plik dziennika. Zawiera on informacje niezbędne do odzyskiwania wszystkich transakcji bazy danych. Pliki bazy danych można obejrzeć po wywołaniu okna jej właściwości (**Properties**).

### 17.3. Stworzenie przykładowych tabel za pomocą narzędzi graficznych

Na początku prac związanych z utworzeniem i modyfikacją baz danych zdefiniujemy jedną z opcji programu **Microsoft SQL Management Studio**.

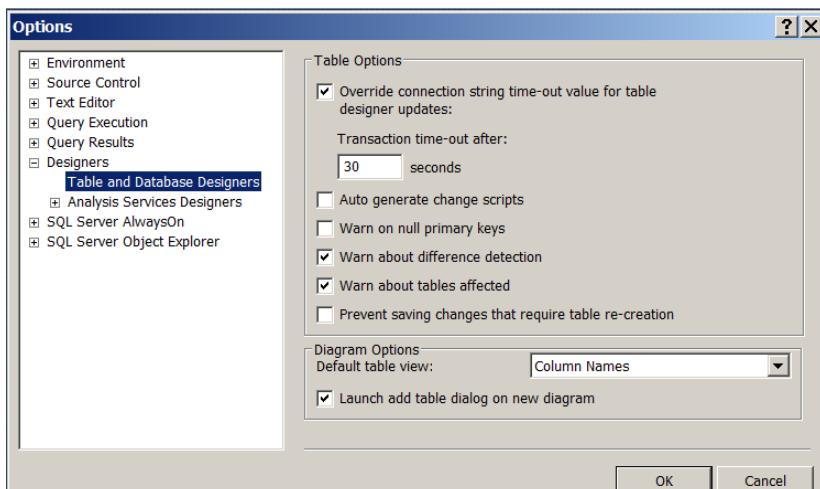
## Zadanie 17\_3

Wprowadź zmiany do opcji programu **Microsoft SQL Management Studio** w celu uzyskania możliwości modyfikacji tabel i baz danych.

---

### Wykonanie

- a. Wybierz polecenie **Options** w menu **Tools**.
- b. Wybierz opcję **Designers**, a potem **Table and Database Designers**.
- c. Odnacz zaznaczenie przy opcji **Prevent saving changes that require table re-creation**, jak przedstawiono na rysunku 17.7. Wybierz **OK**.



Rys. 17.7. Okno **Options** menu **Tools** programu **Microsoft SQL Management Studio**

Wykonując przedstawione poniżej **zadania 17\_4 – 17\_11**, nauczymy się budować tabele za pomocą narzędzi graficznych programu.

Baza danych będzie zawierać pięć tabel o nazwach: **Osoba**, **Kraj**, **Miasto**, **Wojew**, **Datki**. Będą w nich zgromadzone dane o członkach pewnego stowarzyszenia. Pierwsze cztery tabele posłużą do przechowywania danych osobowych oraz adresów osób. Tabela **Datki** będzie zawierała kwoty wpłacone przez osoby na konto tej organizacji.

W pierwszej kolejności będziemy budować puste tabele, następnie utworzymy odpowiednie połączenia między nimi, a w końcu wypełnimy tabele danymi.

## Zadanie 17\_4

W utworzonej bazie danych (o nazwie **MariaLisiecka**) utwórz tabelę **Kraj** o dwóch polach: *id\_kraju*, *kraj*. Typy danych mają być takie same jak na rysunku 17.8.

### Wykonanie

	Column Name	Data Type	Allow Nulls
!	<i>id_kraju</i>	int	<input type="checkbox"/>
	<i>kraj</i>	nvarchar(50)	<input type="checkbox"/>

Rys. 17.8. Tabela **Kraj** w widoku projektu

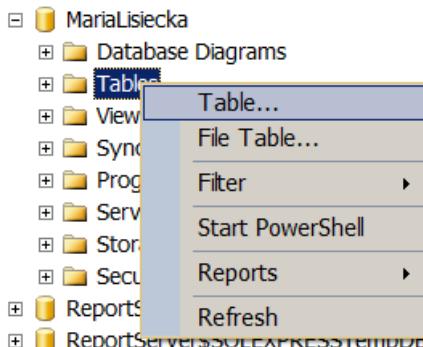
Na rysunku 17.8 przedstawiono widok tabeli w oknie projektu. Kolumna **Column Name** zawsze zawiera nazwy pól, **Data Type** – typy pól, a kolumna **Allow Nulls** zawiera informację dla każdego pola, czy jest ono wymagane.

Pole *id\_kraju* ma typ **int** (liczba całkowita), a pole *kraj* – typ **nvarchar** (tekst zmiennej długości, która nie przekracza zadanej w nawiasach liczb; litera „n” wskazuje na znaki *Unicode*).

W danej tabeli każde pole jest wymagane, tzn. **NOT NULL** (na to wskazuje brak zaznaczenia w kolumnie **Allow Nulls**).

Wykonaj następujące czynności:

- Kliknij dwukrotnie na nazwie bazy danych.
- Prawym przyciskiem myszki jednokrotnie kliknij na węźle **Tables** i wybierz polecenie **Table...** (rys. 17.9).
- W nowo otwartym oknie pustej tabeli wprowadź nazwy pól (*id\_kraju*, *kraj*) oraz wybierz typy danych, jak pokazano na rysunku 17.8.



Rys. 17.9. Polecenie **Table** w menu podręcznym węzła **Tables**

- d. Zaznacz pole *id\_kraju*, klikając na szarej krawędzi po lewej stronie od nazwy pola, i wybierz ikonkę na pasku narzędzi lub w menu podręcznym pola wybierz polecenie **Set Primary Key**.
- e. Zapisz tabelę za pomocą ikonki lub za pomocą polecenia **Save Kraj** z menu **File**.
- f. Zamknij okno projektu.
- g. Otwórz menu podręczne węzła **Tables** i wybierz polecenie **Refresh** w celu odświeżania danych.

## Zadanie 17\_5

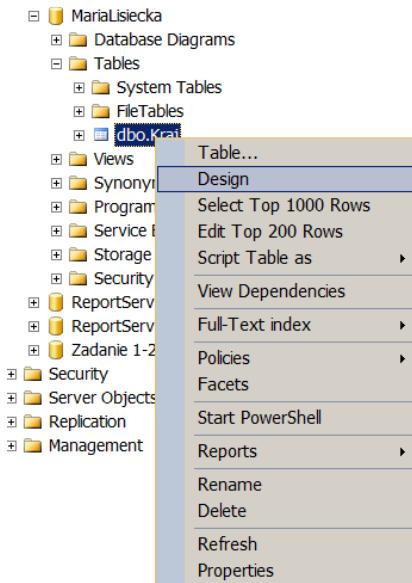
Otwórz projekt tabeli **Kraj**. Zmodyfikuj właściwości pola *id\_kraju*: to pole powinno być automatycznie uzupełniane.

### Wykonanie

W systemie **MS Access** do automatycznego uzupełniania danych służy typ danych **Autonumerowanie**. W **Microsoft SQL Server** do tego służy opcja **Identity**, którą dodajemy do typu danych (typ ma definiować liczbę całkowitą).

Wykonaj następujące czynności:

- a. Otwórz okno projektu tabeli **Kraj** za pomocą polecenia **Design** (rys. 17.10).



Rys. 17.10. Menu podręczne zaznaczonej tabeli

- Zaznacz w oknie projektu (rys. 17.11) pole *id\_kraju*.
- W dolnej części okienka projektu kliknij na znak + po lewej od **Identity Specification**.
- Wybierz **Yes** w wierszu (**Is Identity**), jak przedstawia rysunek 17.11.

The screenshot shows the Microsoft Access Table Designer. At the top, there is a table structure with two columns: 'Column Name' and 'Data Type'. The first row has 'id\_kraju' in 'Column Name' and 'int' in 'Data Type'. The second row has 'kraj' in 'Column Name' and 'nvarchar(50)' in 'Data Type'. Below this, the 'Allow Nulls' checkbox is unchecked for both rows.

Below the table structure is the 'Column Properties' window. It contains several sections:

- (General)**: Shows '(Name)' as 'id\_kraju', 'Allow Nulls' as 'No', 'Data Type' as 'int', and 'Default Value or Binding' as empty.
- Table Designer**: Shows 'Collation' as '<database default>', 'Computed Column Specifica' as empty, 'Condensed Data Type' as 'int', 'Description' as empty, 'Deterministic' as 'Yes', 'DTS-published' as 'No', 'Full-text Specification' as 'No', 'Has Non-SQL Server Subscr' as 'No', and 'Identity Specification' as 'Yes'.
  - (Is Identity)** is selected and set to 'Yes'.
  - 'Identity Increment' is set to '1'.
  - 'Identity Seed' is set to '1'.

Rys. 17.11. Definiowanie właściwości **Identity** dla pola *id\_kraju*

## Zadanie 17\_6

Wzorując się na tabeli **Kraj**, stwórz tabelę **Miasto**, której projekt wyświetlono na rysunku 17.12. Pole *id\_miasta* zdefiniuj jako **Identity**.

### Wykonanie

The screenshot shows the Microsoft Access Table Designer for the 'Miasto' table. It has two columns: 'Column Name' and 'Data Type'. The first row has 'id\_miasta' in 'Column Name' and 'int' in 'Data Type'. The second row has 'miasto' in 'Column Name' and 'nvarchar(50)' in 'Data Type'. Below this, the 'Allow Nulls' checkbox is unchecked for both rows.

Rys. 17.12. Struktura tabeli **Miasta**

## Zadanie 17\_7

Wzorując się na tabeli **Kraj**, stwórz tabelę **Wojew** (rys. 17.13) do przechowywania nazw województw. Pole *id\_wojew* zdefiniuj jako **Identity**.

---

### Wykonanie

	Column Name	Data Type	Allow Nulls
	<i>id_wojew</i>	int	<input type="checkbox"/>
	wojew	nvarchar(50)	<input type="checkbox"/>

Rys. 17.13. Struktura tabeli **Wojew**

## Zadanie 17\_8

Stwórz tabelę o nazwie **Osoba** (rys. 17.14). Pole *id\_osoba* zdefiniuj jako **Identity** (tu pola *id\_m*, *id\_w*, *id\_k* przeznaczone są do przechowywania wartości z pól *id\_miasta*, *id\_wojew* oraz *id\_kraju* utworzonych wcześniej tabel, tzn. są kluczami obcymi).

---

### Wykonanie

	Column Name	Data Type	Allow Nulls
	<i>id_osoba</i>	int	<input type="checkbox"/>
	nazwisko	nvarchar(50)	<input type="checkbox"/>
	imiona	nvarchar(50)	<input type="checkbox"/>
	pleć	nvarchar(9)	<input checked="" type="checkbox"/>
	pesel	char(11)	<input checked="" type="checkbox"/>
	<i>id_m</i>	int	<input checked="" type="checkbox"/>
	<i>id_w</i>	int	<input checked="" type="checkbox"/>
	<i>id_k</i>	int	<input checked="" type="checkbox"/>
	adres	nvarchar(50)	<input checked="" type="checkbox"/>
	kod_pocztowy	varchar(10)	<input checked="" type="checkbox"/>

Rys. 17.14. Struktura tabeli **Osoba**

## Zadanie 17\_9

Utwórz tabelę **Datki** (rys. 17.15). Zdefiniuj pole *id\_datki* tak, aby było uzupełniane automatycznie (tu pole *id\_osoba* jest przeznaczone do połączenia z tabelą **Osoba**, tzn. jest kluczem obcym).

---

## Wykonanie

	Column Name	Data Type	Allow Nulls
1	id_datki	int	<input type="checkbox"/>
2	data	date	<input type="checkbox"/>
3	kwota	money	<input type="checkbox"/>
4	id_osoba	int	<input type="checkbox"/>

Rys. 17.15. Struktura tabeli **Datki**

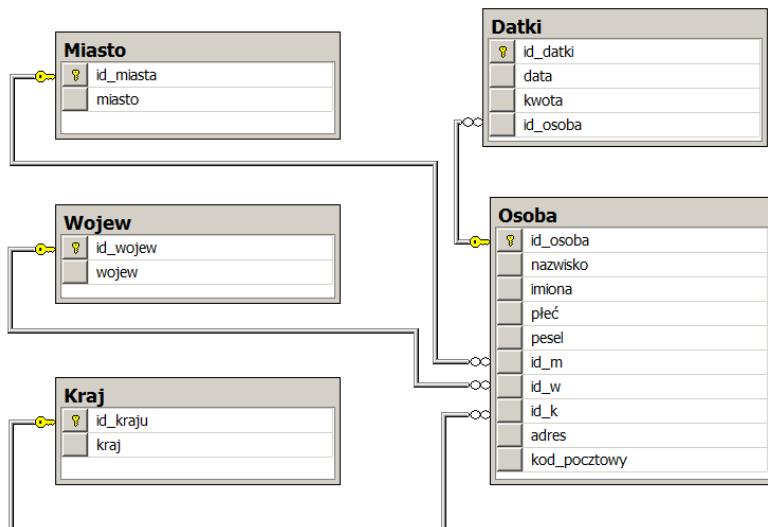
## 17.4. Diagram bazy danych

Po utworzeniu tabel należy je połączyć. Najłatwiej to zrobić, budując diagram bazy danych.

### Zadanie 17\_10

Utwórz diagram wykonanej ostatnio bazy danych. Przykład diagramu przedstawia rysunek 17.16.

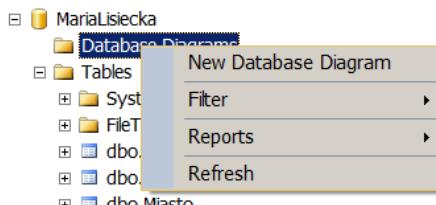
## Wykonanie



Rys. 17.16. Diagram bazy danych

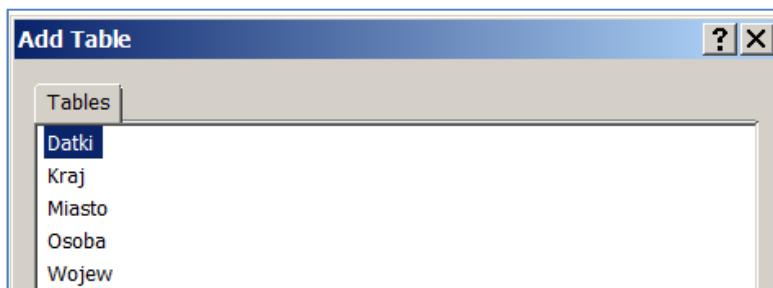
- a. Kliknij dwukrotnie lewym przyciskiem myszy na nazwie bazy danych, a potem w menu podręcznym węzła **Database Diagrams** wybierz polecenie **New Database Diagram** (rys. 17.17).

Należy tu zauważyć, że jeśli to ma być pierwszy diagram bazy danych, to pojawi się okienko dialogowe, w którym należy wybrać **Yes**. Potem trzeba ponownie otworzyć menu podręczne węzła **Database Diagram** i wybrać polecenie **New Database Diagram**.



Rys. 17.17. Menu podręczne węzła **Database Diagrams**

- b. Budowa diagramu zaczyna się od wyboru tabel, które będą w nim umieszczone. Pojawi się okno dialogowe z listą nazw tabel bazy danych **Add Table** (rys. 17.18). Kliknij kolejno na każdej nazwie oraz na przycisku **Add**, dodając nazwy tabel: **Kraj**, **Wojew**, **Miasto**, **Osoba**, **Datki** i wybierz **Close**.



Rys. 17.18. Okno dialogowe **Add Table**

- c. Zapisz diagram.  
d. Połącz tabelę **Kraj** i **Osoba** za pomocą wskaźnika myszy, postępując analogicznie do tego, jak w oknie **Relacje** programu Microsoft Access. W tym celu zaznacz pole *id\_kraju* w tabeli **Kraj** i przeciągnij go dokładnie do wiersza *id\_k* tabeli **Osoba**. Otworzy się okno dialogowe przedstawione na rysunku 17.19. Wybierz w nim **OK**.  
e. W kolejnym oknie **Foreign Key Relationship** możemy definiować, co się stanie przy próbach usunięcia tabeli nadzędnej bądź jej aktualizacji. Ale w tym przypadku nic nie zmieniamy, więc wybierz w nim **OK**.



Rys. 17.19. Okno połączenia tabel **Osoba** i **Kraj**

- f. Zapisz diagram, wybierając w kolejnym okienku **Yes**. Jeśli pojawi się komunikat o niemożliwości zapisania diagramu, należy jeszcze raz przejrzeć struktury tabel: czy typy pól, które łączymy, są takie same, czy jest zdefiniowany klucz podstawowy. Po naprawie błędu próbujemy zapisać diagram ponownie.
- g. Działając podobnie, połącz ze sobą pozostałe tabele, jak pokazano na rysunku 17.16.
- h. Zapisz diagram i odśwież, wybierając polecenie **Refresh**.

Diagram bazy danych pozwala na wiele korzystnych operacji. Przykładowo w oknie diagramu można wywołać menu podręczne tabeli i wybrać polecenie **Design** w celu jej modyfikacji. Po zmianie tabeli należy ją zapisać, zamknąć i kontynuować pracę z diagramem.

## 17.5. Wprowadzenie danych do tabel

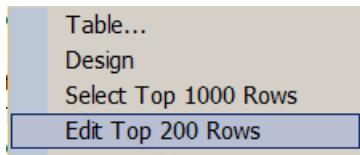
### Zadanie 17\_11

Wprowadź przykładowe dane do każdej z utworzonych tabel – co najmniej jeden przykładowy wiersz do każdej z nich. Na końcu odśwież bazę danych.

### Wykonanie

Należy zachować prawidłową kolejność wypełniania tabel. Najpierw wprowadzamy dane do tabel **Kraj**, **Wojew**, **Miasto**, które nie mają kluczy obcych. Następnie wprowadzamy dane do tabeli **Osoba**. Do pól *id\_k*, *id\_m*, *id\_w* wstawiamy odpowiednio wartości z tabel **Kraj**, **Miasto**, **Wojew**. Na końcu wprowadzamy dane do tabeli **Datki**. Do pól zdefiniowanych jako **Identity** danych nie wprowadzamy.

Aby wprowadzić dane do tabeli, można skorzystać z menu podręcznego, wybierając w nim polecenie **Edit Top 200 Rows** (rys. 17.20).



Rys. 17.20. Polecenie **Edit Top 200 Rows** w menu podręcznym tabeli

W trakcie wypełnienia wiersza zobaczymy czerwone wykrzykniki. Nie zwracając na nie uwagi, wypełniamy wiersz do końca i klikamy myszką w następnym wierszu. Wówczas ten znak zniknie.

Gdy chcemy skasować niezakończony wiersz, naciskamy na ikonkę z czerwonym wykrzyknikiem ! .

## 17.6. Modyfikacja projektu tabeli za pomocą narzędzi graficznych

Korzystając z menu podręcznego tabeli i wybierając odpowiednie polecenia, wprowadzimy zmiany do struktury tabeli. W ten sposób zmieniamy typy danych, dodajemy do pól wartości domyślne, definiujemy reguły poprawności oraz budujemy indeksy.

### Zadanie 17\_12

---

W tabeli **Datki** zdefiniuj dla kolumny data wartość domyślną w postaci funkcji wbudowanej **getdate()**, która powoduje zapisywanie do pola aktualnej daty, jeśli po wprowadzeniu rekordu danych pole pozostanie puste.

Dodaj jeden wiersz danych do tabeli **Datki**, nie wpisując daty, w celu sprawdzenia sposobu działania wartości domyślnej.

---

### Wykonanie

- a. Otwórz tabelę **Datki** w oknie projektu.
- b. Zaznacz pole *data*.
- c. Wprowadź w zakładce **Column Properties** wartość domyślną **getdate ()** w wierszu **Default Value or Binding**, jak pokazano na rysunku 17.21.
- d. Zamknij okno projektu, otwórz tabelę w widoku arkusza danych.

- e. Wpisz przykładowy wiersz danych, wypełniając tylko pola *kwota* oraz *id\_osoba*, przejdź do kolejnego wiersza i naciśnij na pasku narzędzi przycisk 



Rys. 17.21. Definiowanie wartości domyślnej pola *data*

## Zadanie 17\_13

W tabeli **Osoba** dodaj regułę poprawności (**CHECK**) do pola *pesel* wymagającą, aby pole miało długość równą 11.

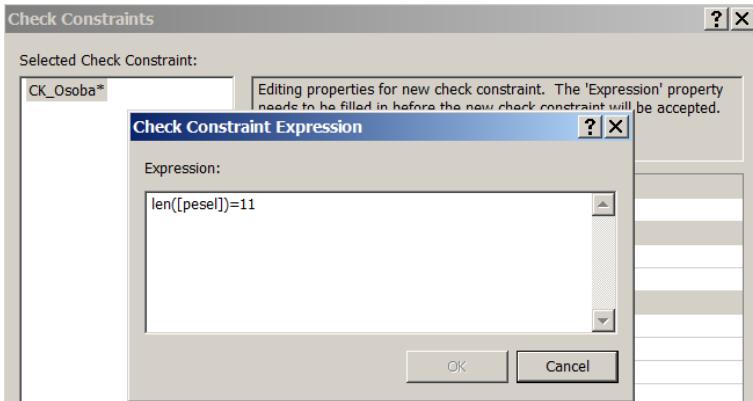
### Wykonanie

- Otwórz menu podręczne węzła **Constraints** tabeli **Osoba**.
- Wybierz polecenie **New Constraint** (rys. 17.22).



Rys. 17.22. Utworzenie ograniczenia **CHECK**

- W okienku **Check Constraints** w wierszu **Expression** wybierz przycisk .
- W okienku dialogowym **Check Constraint Expression** (rys. 17.23) wpisz wyrażenie: **len([pesel]) = 11**, gdzie **len** – funkcja wbudowana, która zwraca liczbę znaków ciągu.
- Wybierz **OK, Close**.
- Zapisz tabelę **Osoba**.
- Odśwież węzeł **Constraints** tabeli **Osoba**.



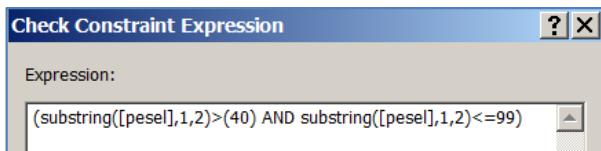
Rys. 17.23. Utworzenie ograniczenia **CHECK** dla pola pesel

## Zadanie 17\_14

W tabeli **Osoba** dodaj regułę poprawności (**CHECK**) do pola **pesel**, aby data urodzenia była z zakresu lat <1941; 1999>.

### Wykonanie

- Postępuj jak przy wykonaniu **Zadania 17\_13** (kroki a-c).
- Zapisz wyrażenie **Check**, stosując funkcję **substring** (analog funkcji **Mid** w **MS Access**) w postaci: **substring ([pesel],1,2)** dla określenia dwóch ostatnich cyfr roku (rys. 17.24).



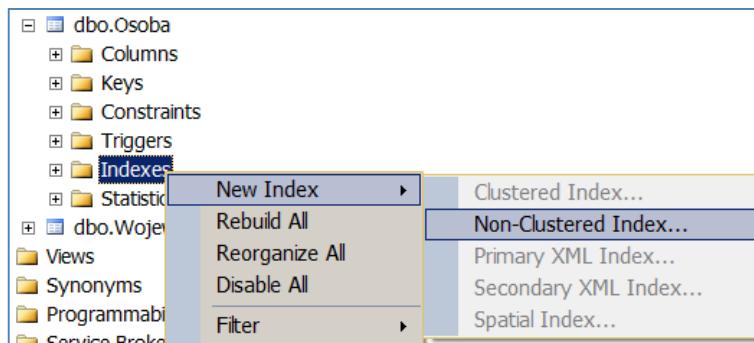
Rys. 17.24. Ograniczenie **CHECK** dla pola pesel

## Zadanie 17\_15

W tabeli **Osoba** utwórz indeksy dla pól *nazwisko* oraz *pesel*.

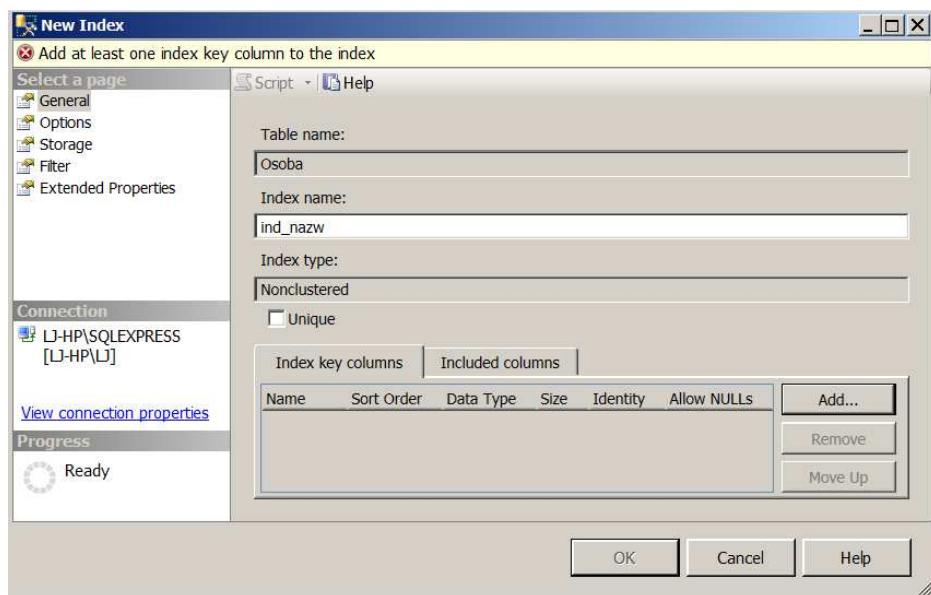
### Wykonanie

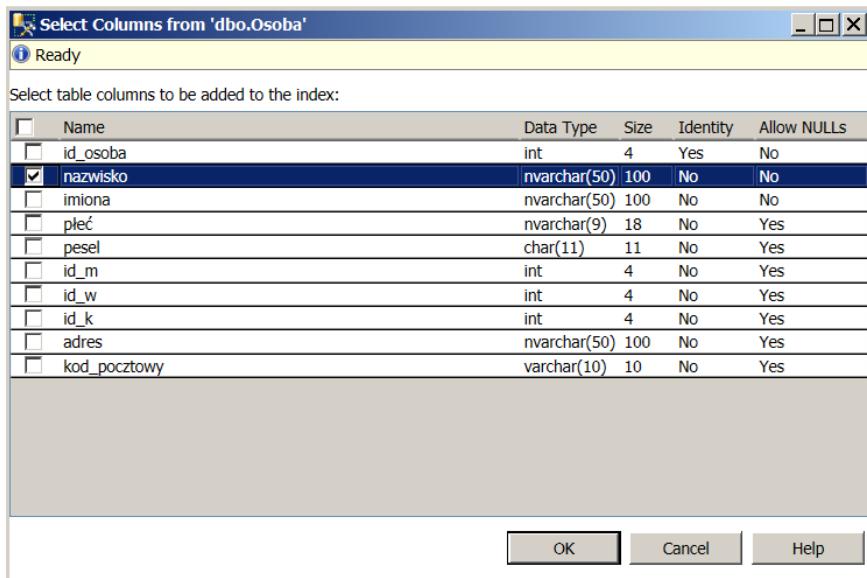
- Otwórz menu podręczne węzła **Indexes** tabeli **Osoba** i wybierz polecenie **New Index** (rys. 17.25).



Rys. 17.25. Polecenie do utworzenia indeksu

- W oknie dialogowym **New Index** wpisz nazwę indeksu (przykładowo: *ind\_nazw*, jak na rys. 17.26) i naciśnij na przycisk **Add**;
- W okienku **Select Columns from 'dbo.Osoba'** zaznacz pole *nazwisko* (rys. 17.27) i naciśnij **OK**.
- W oknie **New Index** wybierz **OK**.
- Analogicznie postępuj przy tworzeniu indeksu dla pola *pesel*, lecz zaznacz opcję  **Unique**.

Rys. 17.26. Okno dialogowe **New Index**



Rys. 17.27. Utworzenia indeksu dla pola nazwisko

## 17.7. Modyfikacja opcji połączenia tabel

Tak samo jak możemy modyfikować strukturę pojedynczej tabeli, możemy też edytować utworzone relacje między tabelami. Przy wykonaniu połączenia między wykonanymi wcześniej tabelami domyślnie ustawiona jest opcja **NO ACTION**, związana z usunięciem rekordu nadziednego. Taką samą opcję program ustalił i dla aktualizacji rekordu nadziednego.

W systemie **MS SQL** możliwe są cztery rozwiązania przy usunięciu rekordu nadziednego, który posiada rekordy podrzędne. Odpowiednie opcje:

- **NO ACTION** – próba usunięcia kończy się błędem;
- **CASCADE** – zostają usunięte wszystkie rekordy podrzędne;
- **SET NULL** – wartości kluczy obcych w rekordach podrzędnych zostają przestawione na nieokreślone;
- **SET DEFAULT** – wartości kluczy obcych w rekordach podrzędnych zostają przestawione na wartość domyślną.

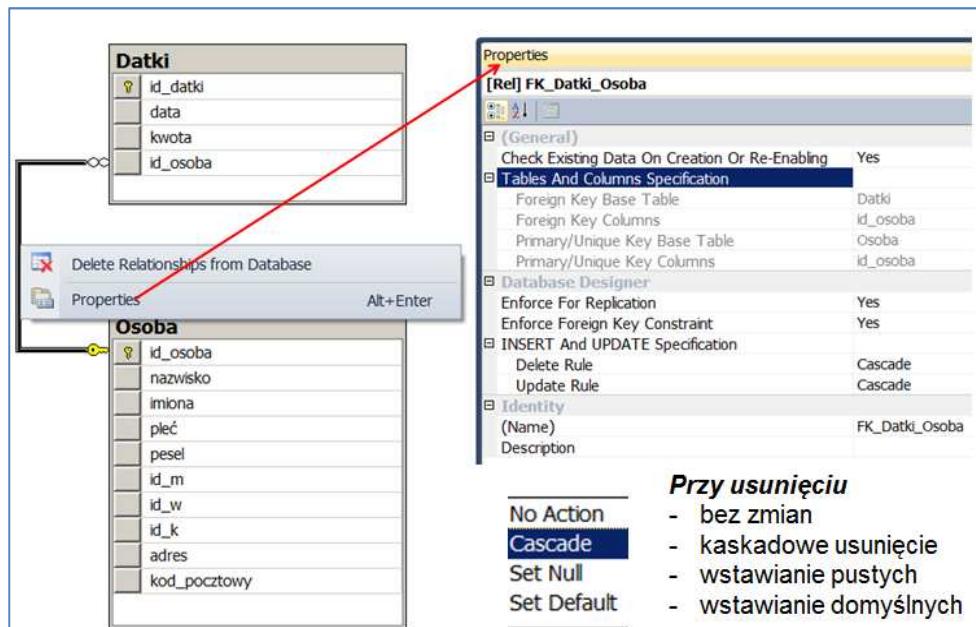
Podobne opcje można ustawić i dla próby aktualizacji rekordu nadziednego.

## Zadanie 17\_16

Modyfikuj opcję połączenia między tabelami **Datki** i **Osoba** w celu możliwości automatycznego usunięcia odpowiednich rekordów z tabeli **Datki** przy usunięciu rekordów z tabeli **Osoba**. Wprowadź po jednym rekordzie do tabeli **Osoba** i **Datki**. Przetestuj wprowadzone zmiany.

### Wykonanie

- Otwórz diagram bazy danych **MariaLisiecka**.
- Kliknij prawym przyciskiem myszy na linii łączącej tabele **Osoba** i **Datki** (rys. 17.28). Wybierz polecenie **Properties** – zostanie wyświetcone okno właściwości relacji.
- Kliknij na znaku + przy **Tables And Columns Specification**.
- Kliknij na znaku + przy **INSERT AND UPDATE Specification**.
- W wierszu **Delete Rule** wybierz opcję **Cascade**.
- Zapisz i zamknij diagram i sprawdź, jaki wpływ ma aktualizacja relacji na rekordy tabeli **Datki** w przypadku usunięcia rekordów z tabeli **Osoba**.



Rys. 17.28. Okno właściwości połączenia tabel **Datki** i **Osoba** z bazy danych **MariaLisiecka**

## Zadania do samodzielnego wykonania

### Zadanie 17\_17

Utwórz indeks nieunikatowy dla pola `id_m` tabeli **Osoba** w bazie danych **MariaLisiecka**.

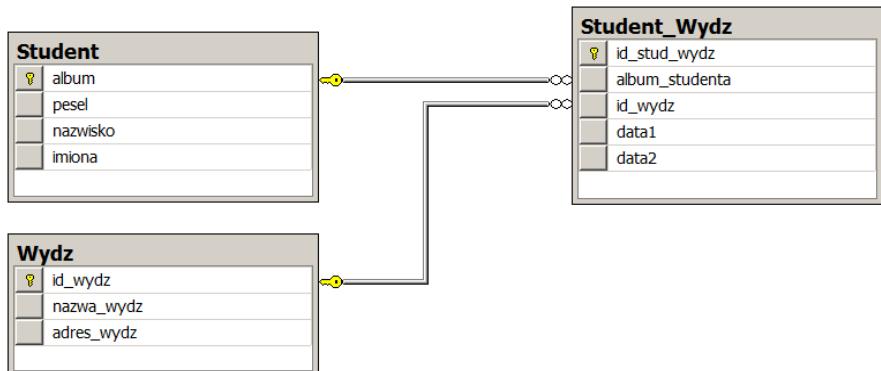
### Zadanie 17\_18

Utwórz w tabeli **Osoba** bazy danych **MariaLisiecka** ograniczenie **Check** dla pola `kwota`: wartość w polu powinna być większa od 0.

### Zadanie 17\_19

Utwórz bazę danych o nazwie **Zadanie17\_19**, której diagram przedstawiono na rysunku 17.29. Baza powinna zawierać dane o studentach, wydziałach uczelni oraz datach studiowania studentów na tych wydziałach. W tabeli **Student\_Wydz** pola `data1` i `data2` zawierają datę początku oraz datę ukończenia studiowania na danym wydziale.

- ❖ Zdefiniuj właściwe typy danych dla wszystkich pól. Pole `id_wydz` oraz pole `id_student_wydz` mają być **Identity**.
- ❖ Wszystkie pola w tych tabelach mają być wymagane, oprócz pola `data2` w tabeli **Student\_Wydz** oraz pola `adres_wydz`,
- ❖ Utwórz indeks dla pola `pesel` tabeli **Student**.



Rys. 17.29. Diagram bazy danych **Zadanie17\_19**

- ❖ Połącz tabele tak, jak przedstawiono na rysunku 17.29.

- ❖ Zdefiniuj właściwości połączenia tabel w celu reagowania systemu na usunięcie rekordów z tabeli **Student**: po usunięciu rekordu danych z tabeli **Student** mają być usunięte odpowiednie dane o jego studiowaniu na wydziałach. Przetestuj wprowadzone zmiany.
- ❖ Utwórz regułę poprawności dla pola **data1**: data nie może być sprzed 2010 roku (zastosuj funkcję wbudowaną **year**).
- ❖ Wprowadź dane do tabel. Do tabeli **Student** - 3 rekordy, do tabeli **Wydz** - 4 rekordy, do tabeli **Student\_Wydz** tyle rekordów, aby przedstawić następującą informację:
  - o pierwszy student studiował tylko na jednym wydziale;
  - o drugi student studiował na trzech wydziałach, na jednym z nich jeszcze studiuje;
  - o trzeci student studiuje w danej chwili na dwóch wydziałach.

## 18. Wprowadzenie do technik przeniesienia bazy danych

Wykonując zadania z rozdziału 17, utworzyłeś dwie przykładowe bazy danych (**MariaLisiecka** oraz **Zadanie 17\_19**). Powstaje pytanie, jak zrobić kopię bazy danych w celu jej przechowania? Są na to różne sposoby i w tym rozdziale będą one przedstawione. Trzeba jednak wiedzieć, że są różne techniki wykonania kopii zapasowych i przy kopiowaniu dużych baz danych (jak jest w rzeczywistości) należy zapoznać się z wieloma dodatkowymi informacjami na ten temat.

### 18.1. Skrypt bazy danych

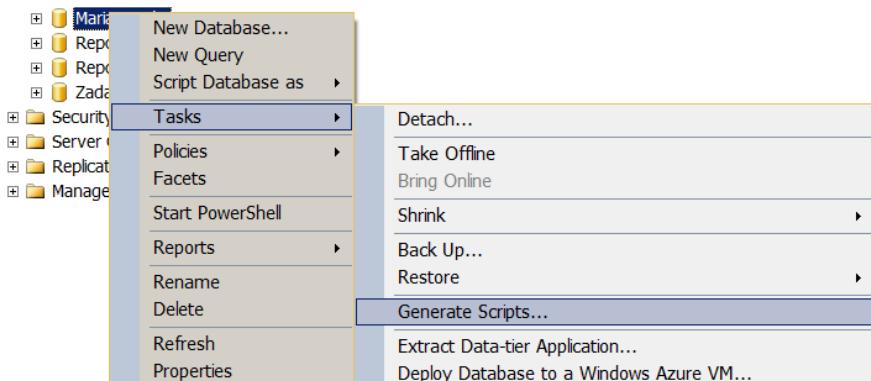
#### Z a d a n i e 18\_1

Wygeneruj za pomocą kreatora skrypt bazy danych **MariaLisiecka**. Skrypt powinien zawierać strukturę i dane. Zapisz skrypt na dysku lokalnym.

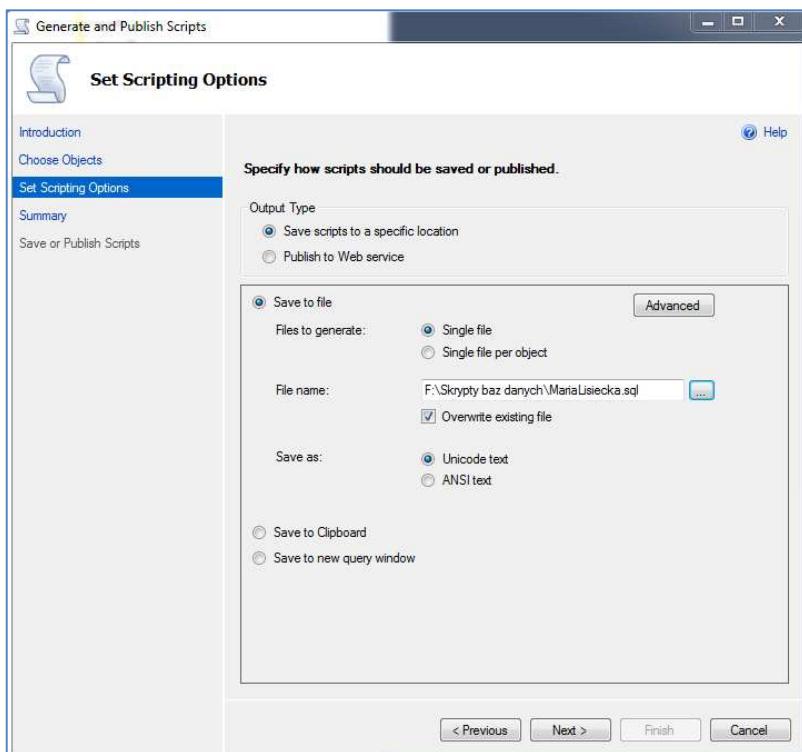
#### Wykonanie

- a. Otwórz program **Microsoft SQL Management Studio**.

- b. W oknie **Object Explorer** otwórz menu podręczne bazy **danych MariaLisiecka** i wybierz w menu podręcznym polecenie **Tasks, Generate Scripts...** (rys. 18.1).



Rys. 18.1. Zastosowanie polecenia **Tasks** do generowania skryptu bazy danych



Rys. 18.2. Okno definiowania opcji generowania skryptu bazy danych

- a. W kolejnych oknach wybierz: **Next; Next.**
- b. Zdefiniuj lokalizację pliku o rozszerzeniu **.sql** w swoim folderze na dysku lokalnym i wybierz przycisk **Advanced** (rys. 18.2).
- c. W oknie **Advanced Scripting Options** w wierszu **Types of data to script** wybierz: **Schema and Data**, a potem **OK**.
- d. W kolejnych oknach wybierz **Next, Next, Finish.**

W wyniku powyższych działań w wybranym przez ciebie folderze pojawi się plik skryptu bazy danych o rozszerzeniu **.sql**. Teraz bazę danych można będzie utworzyć na innym komputerze, wystarczy uruchomić skrypt w okienku **New Query** (zasady pracy w tym okienku rozważane są w rozdziale 19).

## 18.2. Utworzenie kopii bazy danych za pomocą operacji eksportowania

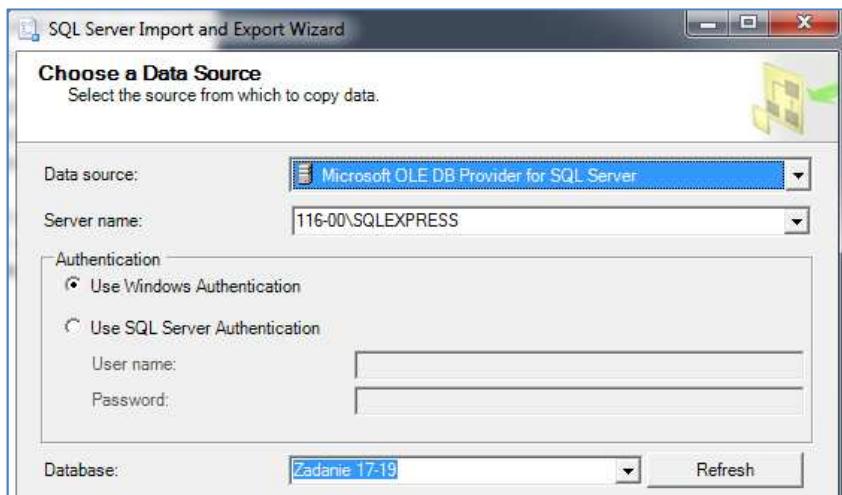
Menu podrzczne każdej bazy danych w oknie **Object Explorer** zawiera, oprócz polecenia **Generate Script**, także inne komendy. Wśród nich – polecenia exportu oraz importu bazy danych. Możemy zastosować operację exportu do utworzenia kopii bazy danych.

### Zadanie 18\_2

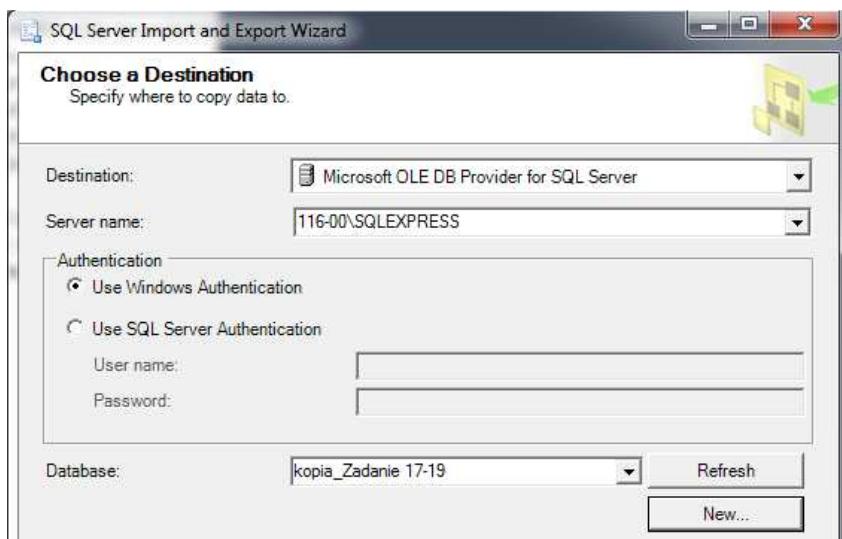
Stwórz kopię bazy danych **Zadanie 17\_19** za pomocą eksportowania. Nadaj jej nazwę **kopia\_Zadanie 17\_19**.

#### Wykonanie

- a. Utwórz nową pustą bazę danych o nazwie **kopia\_Zadanie17\_19**.
- b. Otwórz menu podrzczne bazy danych **Zadanie17\_19**, wybierz polecenie **Tasks, a potem Export data...** – zostanie otwarte okno kreatora **SQL Server Import and Export Wizard**. Wybierz **Next**.
- c. W kolejnym oknie zdefiniuj źródło danych do kopiowania (rys. 18.3), wybierając:
  - **Microsoft OLE DB Provider for SQL Server**,
  - nazwę serwera (na rys. 18.3 to 116-00\SQLEXPRESS),
  - nazwę bazy danych, której dane będą kopowane – **Zadanie 17\_29**. Wybierz **Next**.
- d. W kolejnym oknie (rys. 18.4) podaj dane na temat miejsca docelowego – bazy danych **kopia\_Zadanie17\_19**. Wybierz **Next**.



Rys. 18.3. Definiowanie źródła danych do kopирования



Rys. 18.4. Definiowanie lokalizacji docelowej bazy danych

- e. W kolejnym oknie wybierz **Next**.
- f. W kolejnym oknie zaznacz wszystkie tabele, wybierz **Next**.
- g. W następnych oknach kreatora nic nie zmieniaj, tylko wybieraj **Next**, a w ostatnim – **Finish**.

- h. Obejrzyj utworzone tabele. Możliwe jest, że nie będzie w nich kluczy podstawowych. W tym przypadku należy zdefiniować je ponownie, a potem zbudować diagram w celu połączenia tabel.

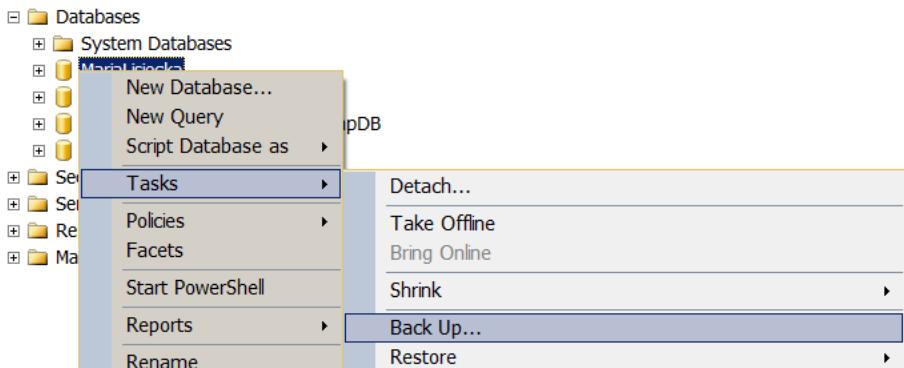
### 18.3. Utworzenie kopii zapasowej bazy danych (backup)

#### Zadanie 18\_3

Utwórz kopię zapasową (backup) bazy danych **MariaLisiecka**.

#### Wykonanie

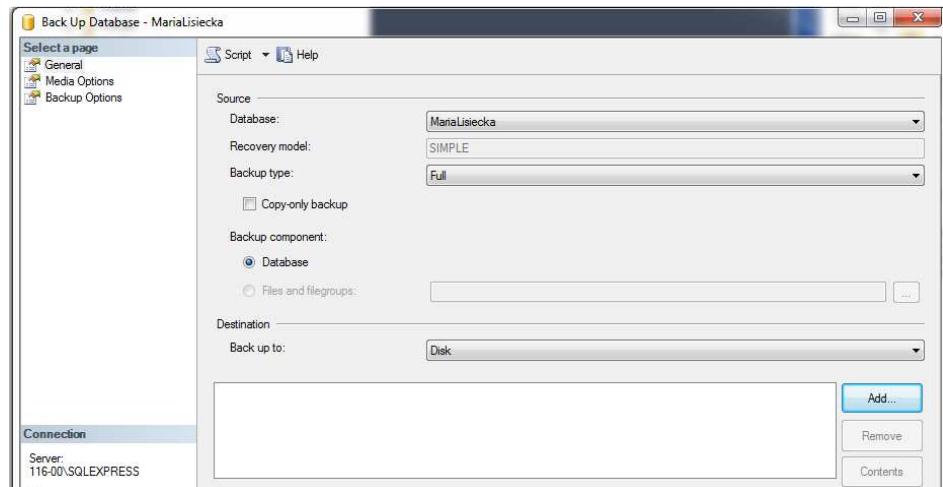
- Otwórz program **Microsoft SQL Management Studio**.
- W oknie **Object Explorer** otwórz menu podręczne bazy danych **MariaLisiecka** i wybierz w menu podrzędnym polecenie **Tasks, Back Up...** (rys. 18.5).



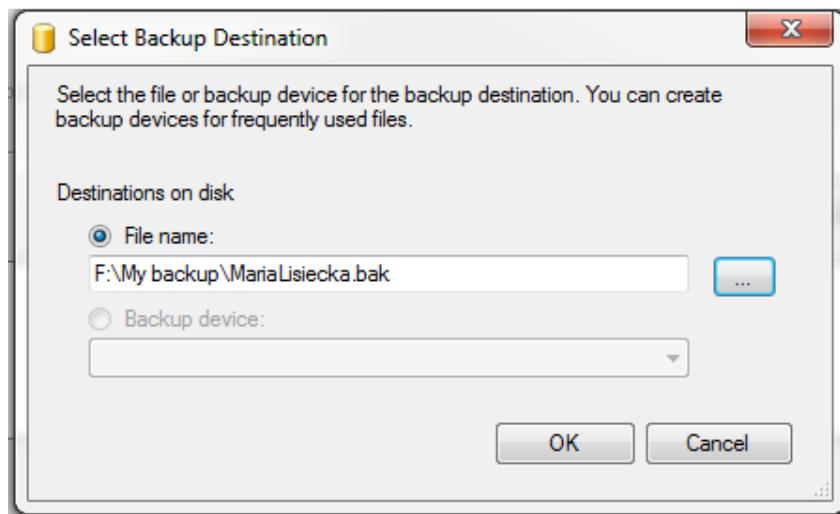
Rys. 18.5. Zastosowanie polecenia **Tasks** do utworzenia kopii zapasowej bazy danych

- W wyświetlonym oknie **Back Up Database – MariaLisiecka** zostaną przedstawione opcje domyślne, które pozostawiamy bez zmiany. Należy wybrać przycisk **Add** (jeśli jakaś kopia już była wykonywana, może być widoczna ścieżka do pliku, a wtedy należy najpierw nacisnąć **Remove**).
- W kolejnym oknie (rys. 18.7) należy wybrać przycisk i zdefiniować ścieżkę do pliku kopii bazy danych. Jako nazwę pliku najlepiej wpisać nazwę bazy danych i koniecznie dodać rozszerzenie **.bak**.

(np. F:\My Backup\MariaLisiecka.bak). Można nic nie zmieniać, wtedy kopią bazy danych zostanie utworzona w folderze domyślnym.

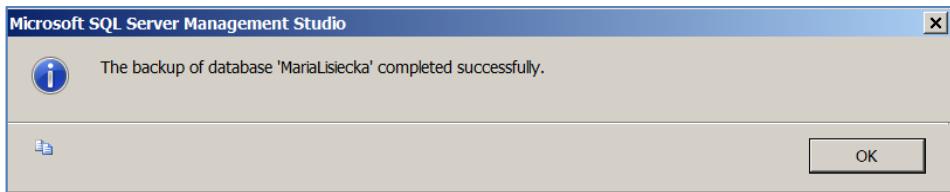


Rys. 18.6. Okno właściwości kopii zapasowej bazy danych



Rys. 18.7. Okno do określenia miejsca przechowywania pliku kopii zapasowej

- e. Wybierz w oknie **Select Backup Destination** przycisk **OK**, a zobaczyś ponownie okno definiowania opcji kopii zapasowej bazy danych. Wybierz w nim **OK** – na ekranie pojawi się komunikat o pomyślnym utworzeniu kopii zapasowej (rys. 18.8).



Rys. 18.8. Komunikat o utworzeniu kopii zapasowej bazy danych

## 18.4. Odtworzenie bazy danych z kopii zapasowej

Kopię zapasową bazy danych możemy odtworzyć na tym samym serwerze SQL lub na innym komputerze, gdzie zainstalowano analogiczne oprogramowanie.

Aby odtworzyć bazę danych i mieć możliwość z nią pracować, należy zastosować polecenie **RESTORE**.

Jeśli chcemy odtworzyć bazę danych na innym komputerze, należy po odzyskaniu bazy danych za pomocą polecenia **RESTORE** zdefiniować właściciela bazy danych, podając swój login. Bez tego nie będziemy mieli do tej bazy danych dostępu.

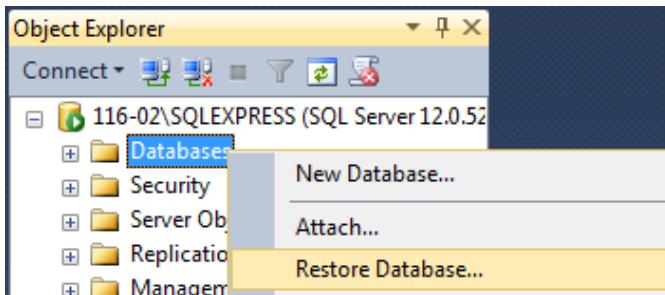
### Zadanie 18\_4

Na dysku **F:** jest folder **My backup**, a w nim plik **MariaLisiecka.bak**, który został stworzony w środowisku **MS SQL Server 2014 Express** i zawiera kopię zapasową bazy danych o nazwie **MariaLisiecka**. Należy odtworzyć bazę danych na innym komputerze w środowisku **MS SQL Server 2014 Express** – pod warunkiem, że w nim nie istnieje baza danych o takiej samej nazwie lub o plikach mających takie same nazwy jak pliki bazy danych w kopii zapasowej.

### Wykonanie

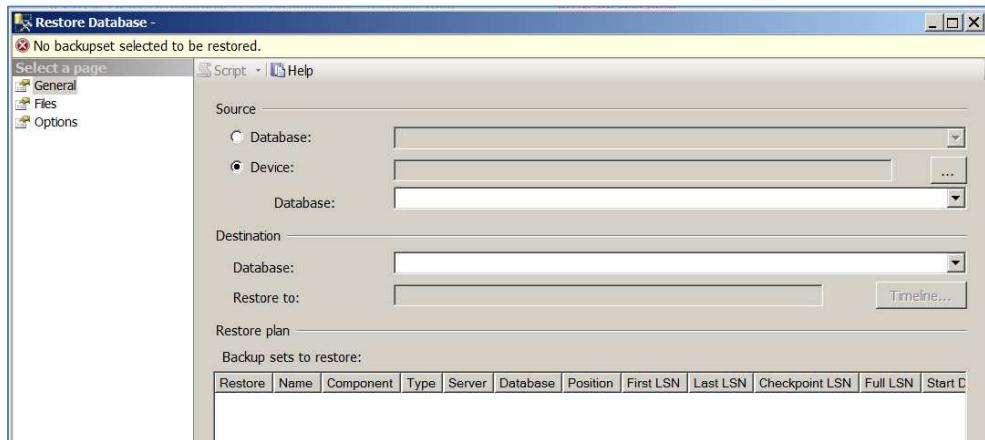
#### Zastosowanie polecenia **RESTORE**

- a. Otwórz program **Microsoft SQL Management Studio**.
- b. W oknie **Object Explorer** otwórz menu podręczne węzła **Databases** i wybierz w nim polecenie **Restore Database...** (rys. 18.9).



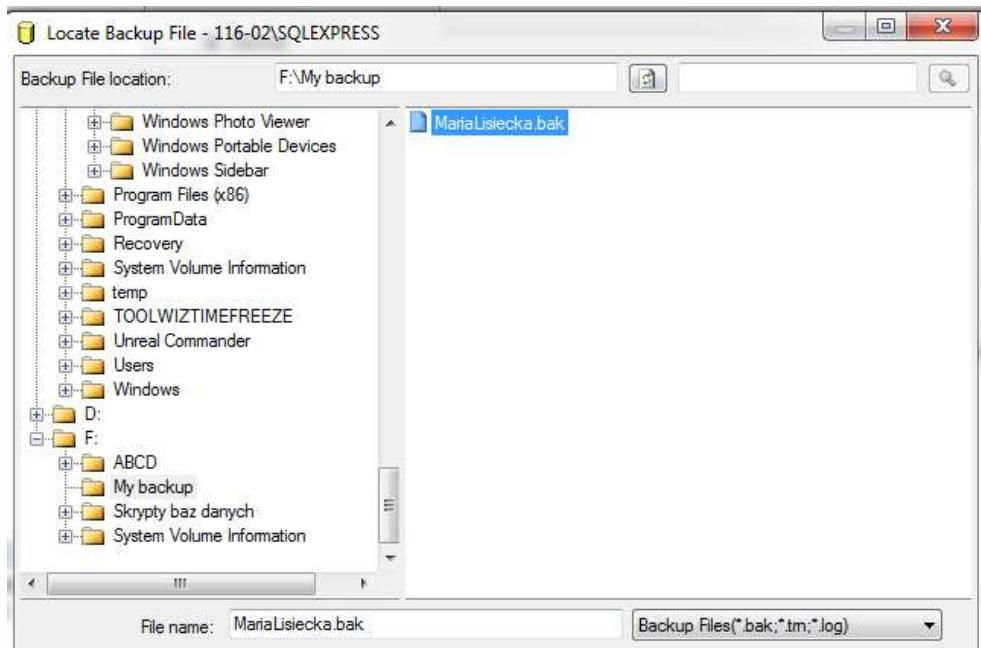
Rys. 18.9. Menu podręczne węzła **Databases**

- c. W oknie **Restore Database** (rys. 18.10) wybierz polecenie **Device**, a potem kliknij na przycisk .

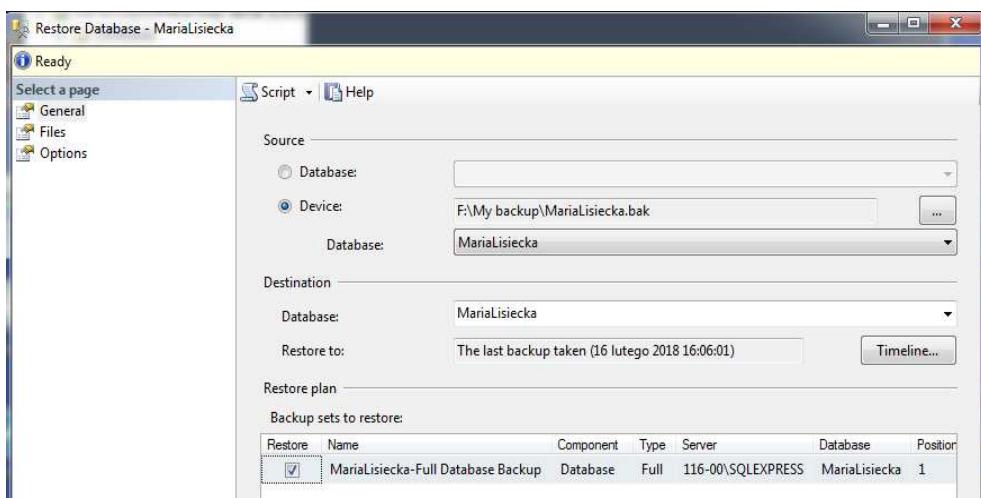


Rys. 18.10. Wybór opcji **Device** w oknie **Restore Database**

- d. W oknie dialogowym **Locate Backup File** (rys. 18.11) wskaż plik kopii zapasowej i wybierz **OK**.  
e. W oknie **Select Backup Devices** wybierz **OK** – zostanie wyświetlane okno odtworzenia bazy danych z wprowadzonymi danymi o lokalizacji pliku (rys. 18.12). Wybierz **OK** – zostanie wyświetlony komunikat o pomyslnym odtworzeniu bazy danych.  
Odśwież węzeł **Databases**. Po rozwinięciu go zobaczysz nazwę odtworzonej bazy danych.



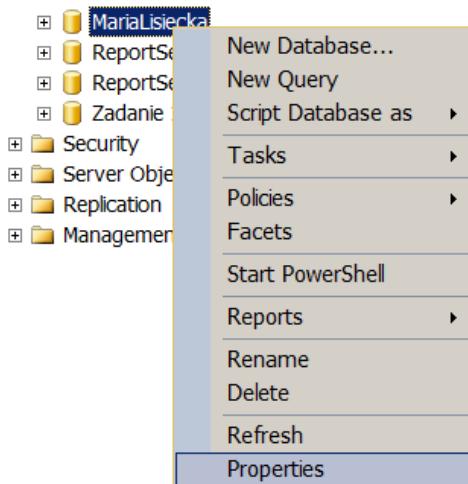
Rys. 18.11. Okno dialogowe **Locate Backup File** ze zdefiniowanym plikiem kopii zapasowej



Rys. 18.12. Okno z opcjami do odtworzenia bazy danych **MariaLisiecka** z kopii zapasowej

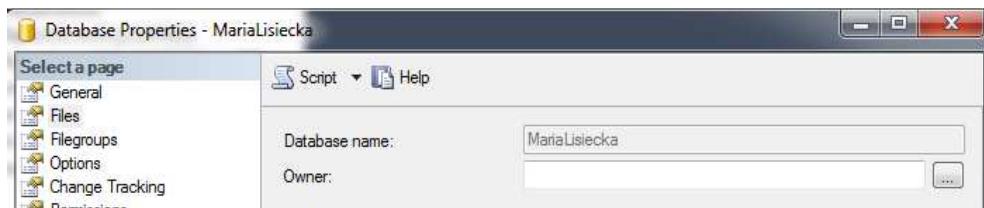
## Definiowanie właściciela bazy danych

- f. W menu podręcznym bazy danych **MariaLisiecka** wybierz polecenie **Properties** (rys. 18.13). Zostanie wyświetcone okno właściwości bazy danych.



Rys. 18.13. Wybór polecenia **Properties** w menu podręcznym bazy danych

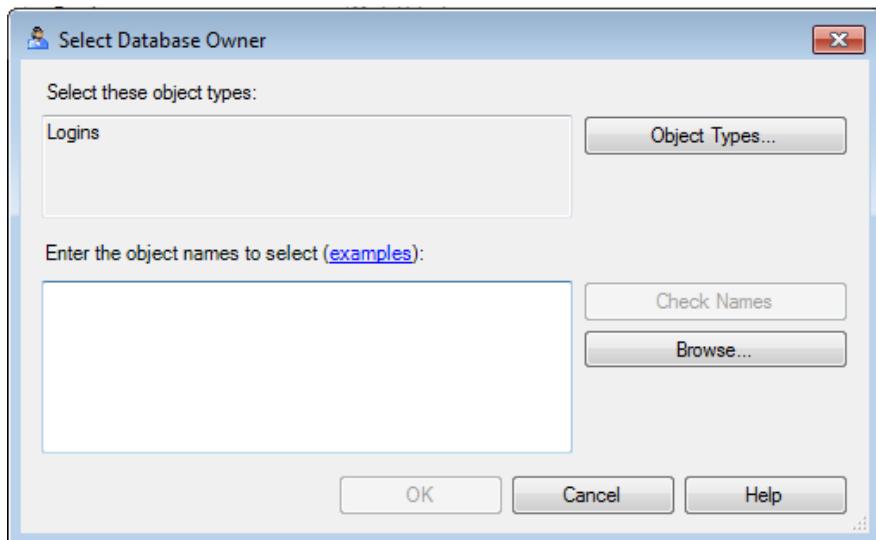
- g. W oknie właściwości bazy danych otwórz zakładkę **Files** (rys. 18.14) i w celu definiowania właściciela bazy danych wywołaj kolejne okno poprzez wybór przycisku obok pola tekstowego **Owner**.



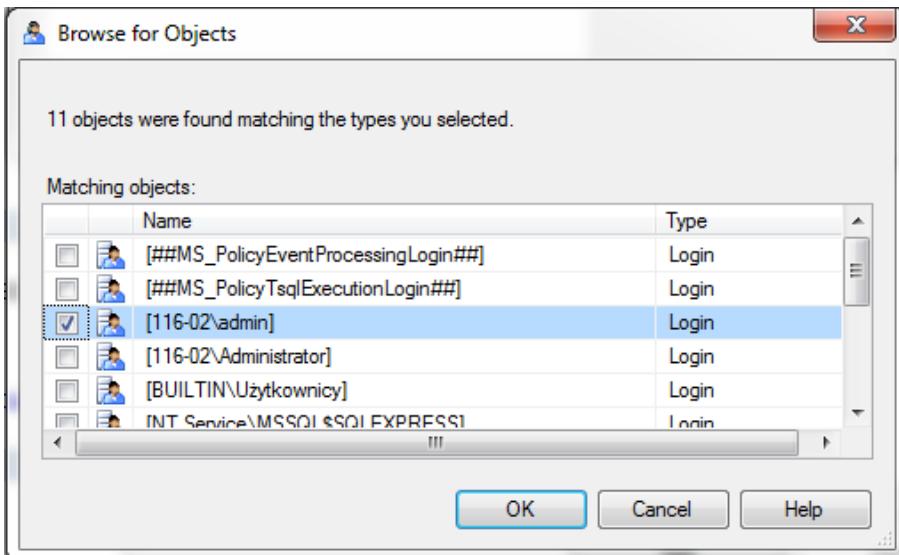
Rys. 18.14. Zakładka **Files** okna właściwości bazy danych

- h. W oknie dialogowym **Select Database Owner** (rys. 18.15) wybierz **Browse** w celu wyszukiwania identyfikatora użytkownika (a faktycznie obiektu potocznie nazywanego *loginem*), który zostanie właścicielem bazy danych.

- i. W kolejnym oknie wybierz login właściciela bazy danych. Na przykład wiadomo, że identyfikator (login) to [116-02\admin]. Wtedy widok okna z zaznaczonym loginem przedstawia rysunek 18.16. Wybierz OK.

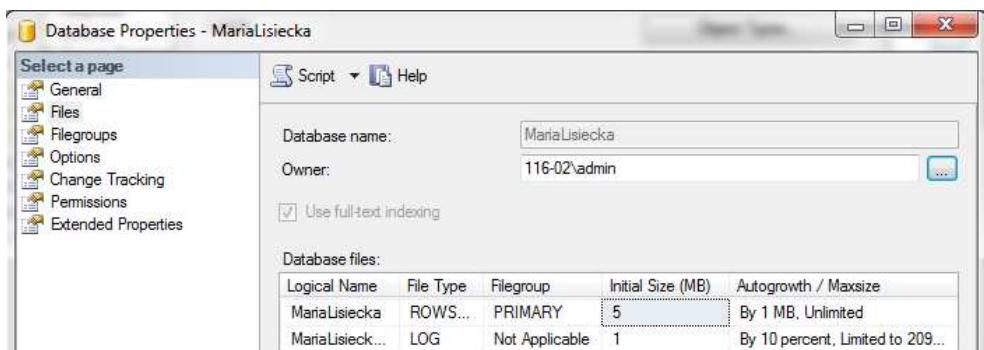


Rys. 18.15. Okno dialogowe **Select Database Owner**



Rys. 18.16. Okno z zaznaczonym loginem właściciela bazy danych

- j. W kolejnym oknie wybierz **OK** w celu zatwierdzenia wybranego loginu. W wyniku zostanie ponownie wyświetlona zakładka **Files** okna właściwości bazy danych **MariaLisiecka**, której właściciel (**Owner**) już będzie zdefiniowany (rys. 18.17).



Rys. 18.17. Okno właściwości bazy danych po zdefiniowaniu jej właściciela

## 18.5. Odłączenie i dołączenie bazy danych

W systemie **SQL Server** jest możliwość odłączenia bazy danych (**Detach**), a potem ponownego jej dołączenia (**Attach**) lub w ogóle umieszczenie jej na innym serwerze. Po odłączeniu baza danych przestaje być widoczna w węźle **Databases**, ale jej pliki pozostają na dysku (domyślnie w folderze **DATA** serwera).

Należy tu zauważyć, że przed kopiowaniem plików bazy danych za pomocą polecenia **Kopij** systemu **Windows** konieczne jest odłączenie bazy danych. Inaczej taka kopia będzie bezużyteczna.

### Zadanie 18\_5

Odłącz bazę danych **MariaLisiecka**, stosując opcje domyślne.

#### Wykonanie

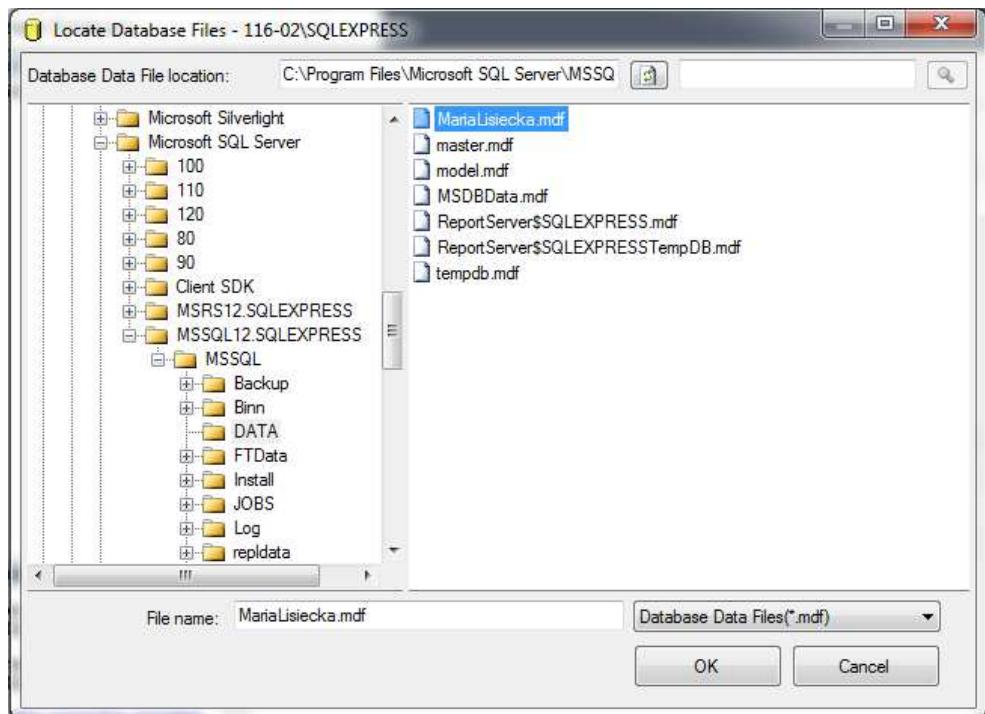
- W menu podręcznym bazy danych wybierz polecenie **Tasks**, a potem **Detach...**
- W oknie **Detach Database** wybierz **OK**.

## Zadanie 18\_6

Dołącz ponownie bazę danych **MariaLisiecka**, która znajduje się w domyślnym folderze serwera.

### Wykonanie

- W menu podręcznym bazy danych wybierz polecenie **Tasks, Attach**.
- W oknie **Attach Database** wybierz **Add**.
- W oknie **Locate Database Files** (rys. 18.18) wybierz nazwę odłączonej bazy danych (domyślna ścieżka do pliku:  
C:\Program Files\Microsoft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\DATA);



Rys. 18.18. Wybór pliku bazy danych w celu dołączenia bazy danych

## Zadania do samodzielnego wykonania

## Zadanie 18\_7

Wygeneruj za pomocą kreatora skrypt bazy danych **Zadanie 17\_19**.

## Zadanie 18\_8

Stwórz kopię bazy danych **MariaLisiecka** za pomocą eksportowania. Nadaj jej nazwę **kopia\_MariaLisiecka**.

---

## Zadanie 18\_9

Stwórz kopię zapasową (**backup**) bazy danych **Zadanie 17\_19**.

---

## Zadanie 18\_10

Stwórz kopię zapasową (**backup**) bazy danych **Zadanie 17\_19**.

---

## Zadanie 18\_11

Odłącz bazę danych **Zadanie 17\_19**. Skopiuj pliki tej bazy danych i zapisz kopię na innym nośniku.

---

## Zadanie 18\_12

Ponownie dołącz bazę danych **Zadanie 17\_19**.

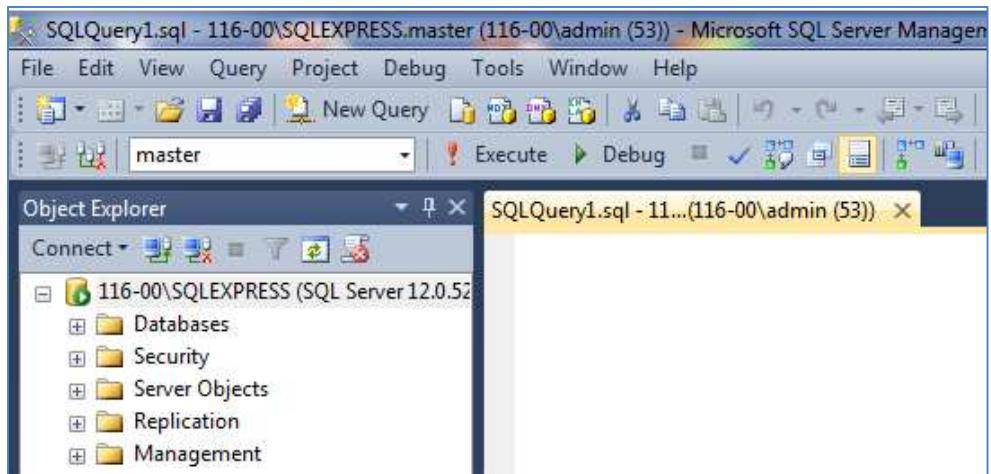
---

# 19. Zastosowanie konstruktora Query Designer do utworzenia instrukcji Transact SQL

Dialekt języka **SQL** zrealizowany w systemie **MS SQL Server** ma nazwę **Transact SQL**. Większość elementów języka **SQL** wykorzystywanego w programie **Microsoft Access** akceptowane są również w **MS SQL**.

Instrukcje **SQL** możemy zapisać w oknie edytora instrukcji **SQL**, które się wyświetli po wyborze w menu programu ikonki  (rys. 19.1).

Jeżeli instrukcja **SQL** jest przeznaczona do pracy z danymi pewnej bazy, to lepiej wybierać polecenie **New Query** z menu podręcznego tej bazy danych. Dalej będzie to pokazane na przykładach.



Rys. 19.1. Widok programu **MS SQL Management studio** z otwartym oknem, w którym zapisujemy polecenia **SQL**

## 19.1. Konstruowanie instrukcji SELECT do wyświetlenia danych

Po zapoznaniu się z dokumentacją **Transact SQL** przychodzimy do wniosku, że znane z programu **Microsoft Access** reguły utworzenia instrukcji **SELECT** możemy również stosować przy zapisywaniu instrukcji w języku **Transact SQL**. Instrukcje **SQL**, przedstawione w rozdziałach 10-13, można wykonać również w programie **Microsoft SQL Management Studio** po zamianie symboli wieloznacznych i niektórych funkcji wbudowanych (np. funkcja wbudowana **Microsoft Access** o nazwie **Mid** w **Transact SQL** ma nazwę **Substring**).

W tabeli 19.1 przedstawiono symbole wieloznaczne stosowane w **Transact SQL**.

Tab. 19.1. Symbole wieloznaczne rozpoznawane przez **SQL Server**

Symbol	Wyszukuje
%	Dowolny ciąg złożony z zera lub kilku znaków
_	Dowolny jeden znak
[ ]	Dowolny jeden znak z określonego przedziału (na przykład [b-e]) lub określonego zbioru znaków (na przykład [b, c, d, e])
[^]	Dowolny jeden znak nie znajdujący się w przedziale (na przykład [^b-e]) lub nie znajdujący się w zbiorze znaków (na przykład [^b, c, d, e])

Program **Microsoft SQL Management Studio** dysponuje konstruktorem o nazwie **Query Designer**, który wspomaga utworzenie instrukcji **Transact SQL**. Zastosowanie tego konstruktora znacznie przyspiesza proces napisania zapytań **SQL**. Wykonanie poniższych zadań ma na celu wyjaśnienie podstawowych zasad pracy w oknie konstruktora instrukcji **Transact SQL**.

## Zadanie 19\_1

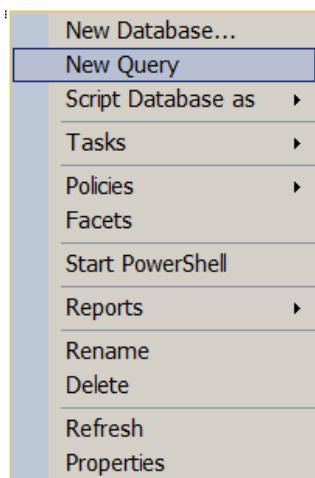
---

W oknie programu **Microsoft SQL Management Studio** otwórz okno konstruktora instrukcji **SQL Query Designer** i utwórz zapytanie **Transact SQL** do wyświetlenia wszystkich danych o studentach z bazy danych o nazwie **Zadanie 17\_19**.

---

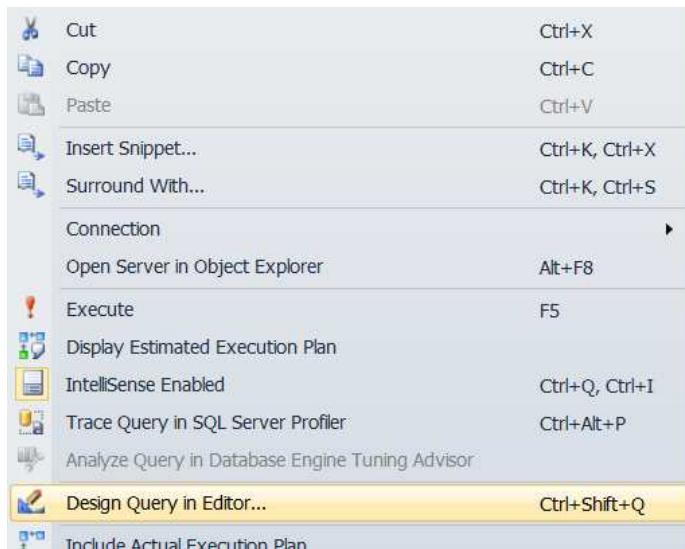
### Wykonanie

- Otwórz menu podręczne bazy danych **Zadanie 17\_19** i wybierz polecenie **New Query** (rys. 19.2). Wówczas po prawej stronie pojawi się puste okienko edytora instrukcji **SQL**.



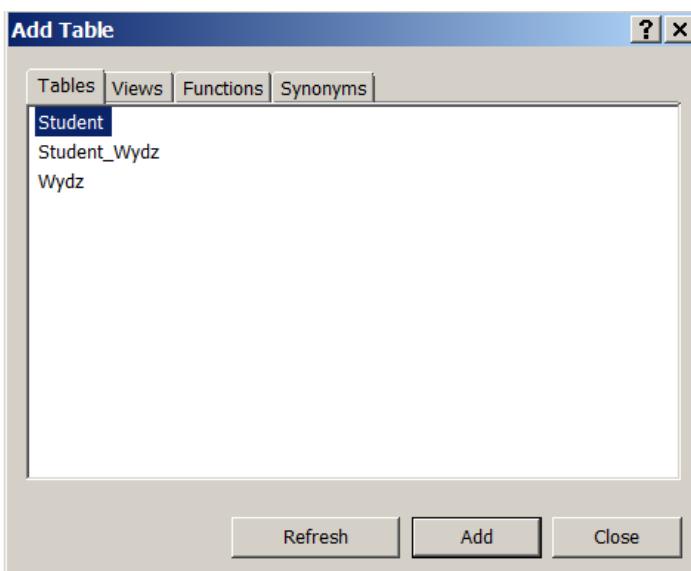
Rys. 19.2. Polecenie **New Query** w menu podręcznym bazy danych

- Kliknij prawym przyciskiem myszy w miejscu utworzenia instrukcji – pojawi się menu podręczne okna edytora instrukcji **SQL**. Wybierz w nim polecenie **Design Query in Editor** (rys. 19.3).



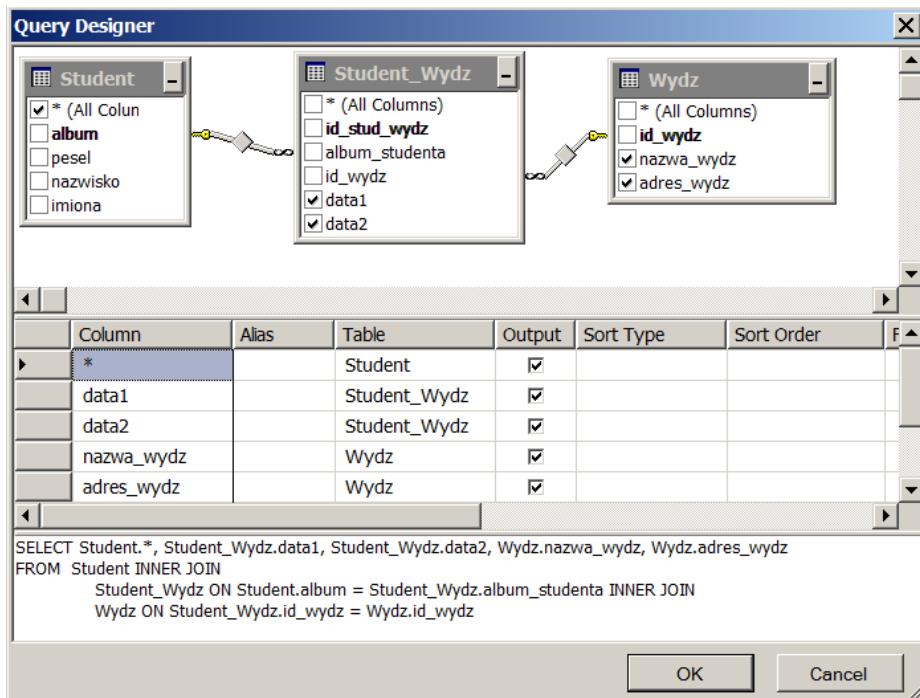
Rys. 19.3. Menu podręczne okna **New Query** z zaznaczonym poleceniem do konstruowania instrukcji **SQL**

- c. Wybierz za pomocą przycisku **Add** w oknie **Add Table** po kolejem wszystkie trzy tabele (rys. 19.4) i na końcu wybierz **OK**.



Rys. 19.4. Widok okna **Add Table**

- d. Wybierz tabele bazy danych oraz pola do wyświetlenia. Można to zrobić tak, jak to pokazano na rysunku 19.5.



Rys. 19.5. Utworzenie w oknie **Query Designer** instrukcji **SQL** do **Zadania 19\_1**

- e. Po wyborze przycisku **OK** okno **Query Designer** zostanie zamknięte, a w oknie **New Query** pojawi się wygenerowane polecenie **SQL**.
- f. Uruchom polecenie za pomocą ikonki **Execute**.
- g. Zapisz plik o rozszerzeniu **.sql**.
- h. Zamknij okno edytora **SQL**.

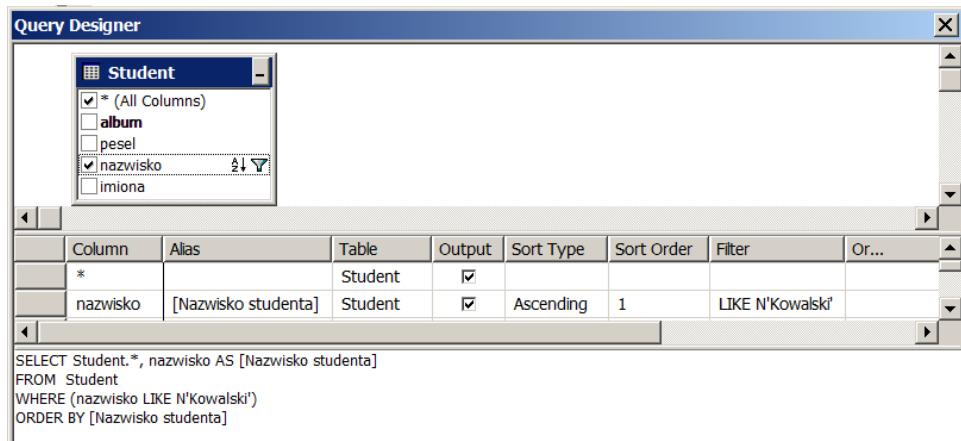
## Zadanie 19\_2

Utwórz za pomocą **Query Designer** instrukcję **Transact SQL** w celu wyszukiwania danych z bazy **Zadanie 17\_19**, a mianowicie do wyświetlenia wszystkich danych z tabeli **Student** o osobach, których nazwisko jest „Kowalski”. Wynikowe dane powinny przy wyświetleniu być posortowane według pola *imiona*.

Wykonaj instrukcję **SQL**. W razie pomyślnego wykonania zapisz instrukcję w postaci pliku o rozszerzeniu **\*.sql**. Zamknij okienko instrukcji **SQL**.

## Wykonanie

- Działając jak w poprzednim zadaniu, otwórz okno **New Query**, a potem **Query Designer**.
- Dodaj tabelę **Student (Add, Close)**.
- Pozostałe czynności (rys. 19.6):
  - o wybierz w tabeli **Student** symbol „\*” do wyświetlenia wszystkich pól;
  - o umieść w kolumnie **Column** pole *nazwisko*;
  - o zapisz kolumnie **Alias** pseudonim do pola *nazwisko*: „Nazwisko studenta”;
  - o w kolumnie **Output** odznacz opcję wyświetlenia pola *nazwisko*;
  - o w kolumnie **Filter** wprowadź kryterium dla pola *nazwisko*: **LIKE 'Kowalski'** (gdy klikniemy w innym polu, kryterium natychmiast zmieni postać przez dopisywanie litery N; tak dzieje się, gdy pole jest przeznaczone do przechowywania symboli Unicodu).
- Wykonaj i zapisz utworzone polecenie **Transact SQL**.



Rys. 19.6. Projekt instrukcji **Transact SQL** utworzonej przy wykonania **Zadania 19\_2**

## Zadanie 19\_3

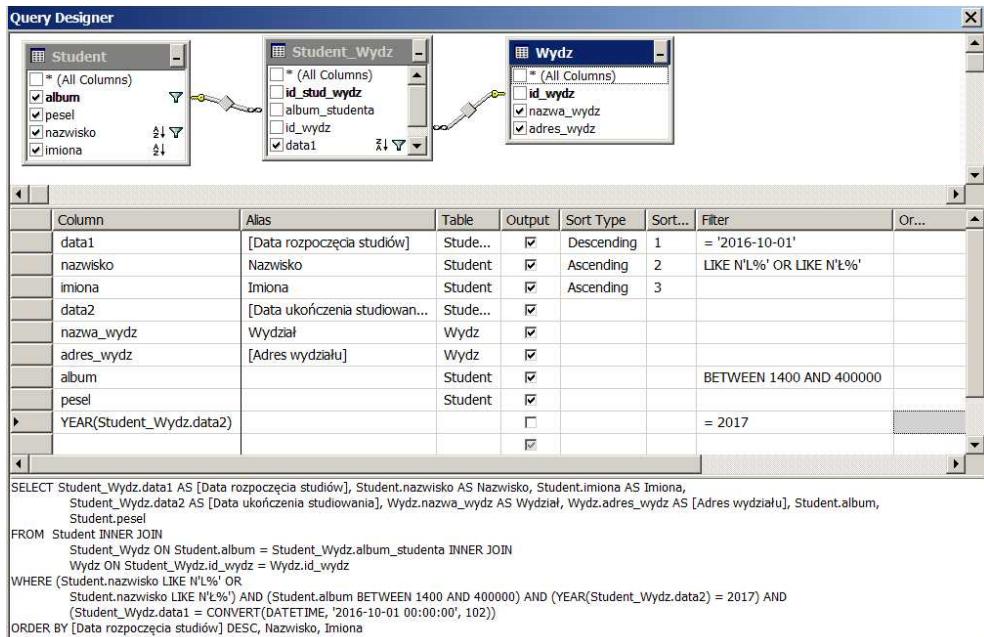
Utwórz za pomocą **Query Designer** instrukcję **SQL** w celu wyszukiwania danych z bazy **Zadanie 17\_19**, a mianowicie do wyświetlenia wszystkich danych dotyczących studentów spełniających następujące kryteria:

- ❖ nazwisko zaczyna się od litery „L” lub „Ł”,
- ❖ numer albumu jest z zakresu <14000; 400000>,
- ❖ data rozpoczęcia studiów to 2016/10/01,

- ❖ data końca studiowania jest z 2017 roku.  
 Dane należy posortować malejąco według pola *data1* oraz rosnąco według pól *nazwisko* i *imiona*.

## Wykonanie

Projekt instrukcji **SQL** przedstawia rysunek 19.7.



Rys. 19.7. Projekt instrukcji **Transact SQL** utworzonej przy wykonaniu  
**Zadania 19\_3**

## 19.2. Konstruowanie zapytań agregujących

Kwerendy zawierające funkcje agregujące oraz grupowanie zostały przedstawione w widoku projektu w **Rozdziale 8**, a w widoku **SQL** – w **Rozdziale 12**. Takie same zapytania możemy zapisać w języku **Transact SQL** z wykorzystaniem konstruktora **Query Designer**, jak zaprezentowano to w poniższych przykładach. W **Zadaniu 19\_4** zapisujemy instrukcję **Transact SQL** zawierającą funkcję agregującą, a w **Zadaniu 19\_5**, pomimo funkcji agregującej, wykonujemy także grupowanie.

## Zadanie 19\_4

Skonstruuj instrukcję **Transact SQL** w celu wyświetlenia danych z bazy **MariaLisiecka** na temat: ile osób w tabeli **Osoba** jest z Warszawy.

### Wykonanie

- W okienku **Query Designer** dodaj tabele **Osoba** i **Miasto**.
- Kliknij prawym przyciskiem myszki w oknie **Query Designer** i wybierz **Add Group By** (rys. 19.8) – w siatce okna konstruktora powstanie dodatkowa kolumna o nazwie **Group By** (rys. 19.9).



Rys. 19.8. Wybór opcji **Add Group By** z menu podręcznego okna **Query Designer**

Query Designer							
Osoba		Miasto					
<input type="checkbox"/> * (All Columns)	<input type="checkbox"/> id_osoba	<input type="checkbox"/> nazwisko	<input type="checkbox"/> imiona	<input type="checkbox"/> pleć	<input type="checkbox"/> * (All Columns)	<input type="checkbox"/> id_miasta	<input type="checkbox"/> miasto
<input type="checkbox"/> id_osoba      Σ <input type="checkbox"/> miasto		<input type="checkbox"/> nazwisko <input type="checkbox"/> imiona <input type="checkbox"/> pleć		<input type="checkbox"/> * (All Columns) <input type="checkbox"/> id_miasta <input type="checkbox"/> miasto			

Column	Alias	Table	Output	Sort T...	Sort Order	Group By	Filter
id_osoba	[Liczba osób]	Osoba	<input checked="" type="checkbox"/>			Count	
miasto		Miasto	<input type="checkbox"/>			Where	LIKE N'Warszawa'

```

SELECT COUNT(Osoba.id_osoba) AS [Liczba osób]
FROM Osoba INNER JOIN
      Miasto ON Osoba.id_m = Miasto.id_miasta
WHERE (Miasto.miasto LIKE N'Warszawa')
  
```

Rys. 19.9. Widok instrukcji **SQL** zawierającej funkcję agregującą **COUNT** (Zadanie 19\_4)

- Wypełnij siatkę, jak to przedstawiono na rysunku 19.9. Zwróć uwagę, że tak samo jak w kwerendach grupujących **Microsoft Access** nie zaznaczaliśmy do pokazywania pola uczestniczącego w tworzeniu kryterium (była wybierana opcja **Gdzie**), tak również i teraz nie zaznaczamy do pokazywania w kolumnie **Output** pola *miasto*.

## Zadanie 19\_5

Skonstruuj instrukcję **Transact SQL** w celu wyświetlenia z bazy danych **MariaLisiecka** listy nazw miejscowości, z których pochodzą darczyńcy. Obok każdej nazwy miejscowości należy wyświetlić liczbę osób z nią związanych. Nazwy miejscowości należy uporządkować rosnąco. Przy wykonaniu należy uwzględnić możliwość istnienia takich samych nazw miejscowości należących do różnych województw.

### Wykonanie

Przykład wykonania zadania przedstawia rysunek 19.10.

```

SELECT Miasto.miasto, Miasto.id_miasta, COUNT(Osoba.id_osoba) AS [Liczba osób]
FROM Osoba INNER JOIN
      Miasto ON Osoba.id_m = Miasto.id_miasta
GROUP BY Miasto.miasto, Miasto.id_miasta
ORDER BY Miasto.miasto, Miasto.id_miasta
  
```

Rys. 19.10. Widok okna **Query Designer** do **Zadania 19\_5**

### 19.3. Konstruowanie instrukcji INSERT, UPDATE, DELETE

Schematy instrukcji **Transact SQL** do manipulowania danymi, tzn. do dodania, usunięcia oraz aktualizacji rekordów, są takie same jak rozważane w rozdziale 13. Poniżej zaprezentowano utworzenie takich instrukcji za pomocą narzędzi graficznych programu **Microsoft SQL Management Studio**.

## Dodawanie danych

Schemat instrukcji dodania rekordu danych do tabeli ma postać:

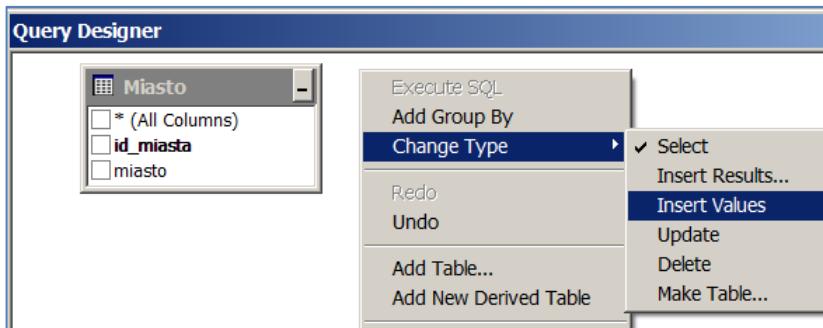
```
INSERT INTO nazwa_tabeli (pole,...) VALUES (wartość,...);
```

### Zadanie 19\_6

Za pomocą narzędzi graficznych programu **Microsoft SQL Management Studio** utwórz instrukcję **Transact SQL** wprowadzenia jednego rekordu danych do tabeli **Miasto** w bazie danych **MariaLisiecka**.

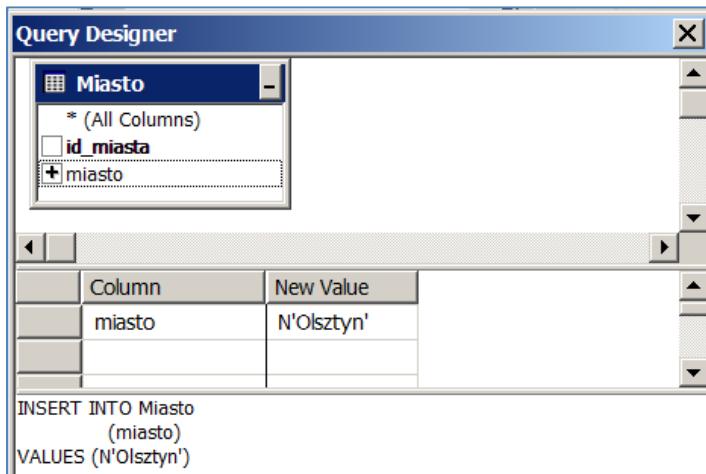
#### Wykonanie

- Zaznacz w węźle **Databases** nazwę bazy danych i otwórz okienko **New Query**.
- Otwórz okno konstruktora zapytań SQL **Design Query in Editor...**
- W oknie **Add Table** wybierz tabelę **Miasto**.
- W menu podręcznym okna **Query Designer** wybierz **Change type, Insert Values**, jak pokazano na rysunku 19.11 – okno **Query Designer** zostanie przygotowane do konstruowania instrukcji **INSERT**.



Rys. 19.11. Wybór polecenia **Insert Values**

- Zaznacz pole *miasto* tabeli **Miasto**. Nie zaznaczaj pola *Id\_miasta*, ponieważ dane do niego są wprowadzane automatycznie.
- Wprowadź wiersz przykładowych danych do siatki. Na rys. 19.12 wyświetlono widok okna **Query Designer** po wpisaniu wartości do pola *miasto*. Apostrofy, jak również litera 'N' zostały dodane automatycznie.



Rys. 19.12. Instrukcja **INSERT** w oknie **Query Designer**

- g. Naciśnij **OK**. Spowoduje to zamknięcie okna **Query Designer**, a wygenerowane polecenie **INSERT** pojawi się w oknie **New Query**. Wykonaj go, wybierając w menu programu ikonę **Execute**.
- h. Zamknij okno **New Query**.
- i. Otwórz tabelę **Miasto** i przekonaj się, że nowy rekord został dodany.

### Aktualizacja danych

Schemat instrukcji aktualizacji danych tabeli ma postać:

**UPDATE nazwa\_tabeli SET pole = wyrażenie, ... WHERE kryterium;**

## Zadanie 19\_7

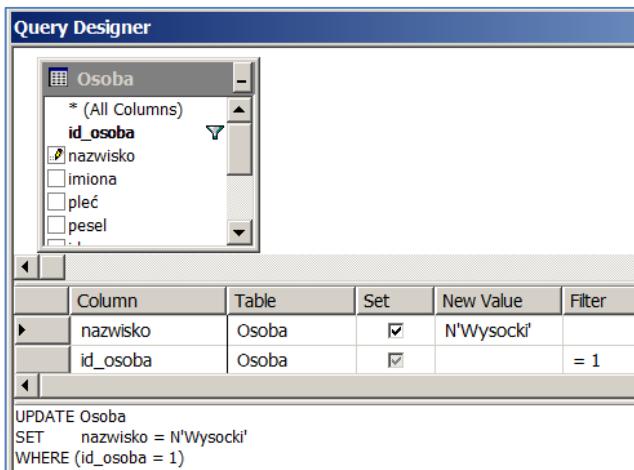
Utwórz i wykonaj instrukcję **Transact SQL** do zmiany rekordu tabeli **Osoba** w bazie danych **MariaLisiecka**. Zmień wartość pola **Nazwisko** w rekordzie, w którym wartość **idOsoba** jest równa 1. Wprowadź nowe nazwisko - „Wysocki”. Zastosuj do utworzenia polecenia narzędzia graficzne programu.

### Wykonanie

- a. Na początku postępuj jak przy wykonaniu **Zadania 19\_6**, lecz wybierz polecenie **UPDATE**.
- b. W siatce konstruktora w kolumnie **Column** wybierz z listy rozwijanej pole **nazwisko**, a w następnym wierszu – pole **id\_osoba**.

- W kolumnie **New Value** wprowadź nową wartość pola *nazwisko*.
- Dodaj w kolumnie **Filter** kryterium dla pola *id\_osoba*.

Na rysunku 19.13 przedstawiono widok wykonanej instrukcji w oknie **Query Designer**.



Rys. 19.13. Instrukcja **UPDATE** w oknie **Query Designer**

### Usunięcie rekordów

Schemat instrukcji do usunięcia rekordów tabeli ma postać:

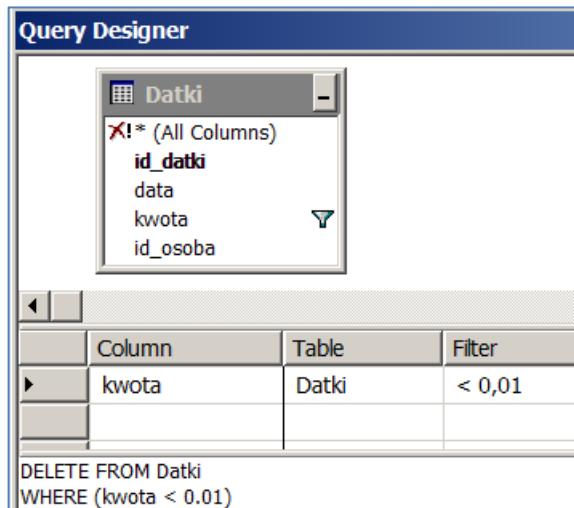
**DELETE FROM nazwa\_tabeli WHERE kryterium**

## Zadanie 19\_8

Utwórz i wykonaj instrukcję **Transact SQL** do usunięcia z tabeli **Datki** (w bazie danych **MariaLisiecka**) wszystkich rekordów, w których pole kwota zawiera wartość mniejszą od 0,01.

### Wykonanie

- Na początku postępuj jak przy wykonaniu **Zadania 19\_6**, lecz wybierz polecenie **DELETE**.
- Dalej wypełnij siatkę konstruktora, tak jak na rysunku 19.14.



Rys. 19.14. Instrukcja **DELETE** w oknie **Query Designer**

## Zadania do samodzielnego wykonania

### Zadanie 19\_9

Utwórz za pomocą **Query Designer** instrukcję **Transact SQL** w celu wyszukiwania danych z bazy **Zadanie 17\_19**, a mianowicie do wyświetlenia wszystkich danych z tabeli **Student** na temat studentek, których e-mail zaczyna się od litery „k”, a kończy się na „@gmail” lub „@o2.pl” Dane należy posortować malejąco według pola **e-mail** oraz rosnąco według **nazwiska** i **imienia**.

### Zadanie 19\_10

Utwórz za pomocą **Query Designer** instrukcję **SQL** w celu wyszukiwania danych z bazy **Zadanie 17\_19**, a mianowicie do wyświetlenia wszystkich danych na temat tych osób, które zaczynały studia w październiku i jeszcze studiują. Dane należy posortować rosnąco według pól **data1**, **album**.

*Podpowiedź do wykonania zadania 19\_10*

Skorzystaj z funkcji **Month**, która działa tak samo jak w programie **Microsoft Access**.

## Zadanie 19\_11

Utwórz instrukcję **SQL** do wyświetlenia wszystkich danych z tabeli **Student** bazy **Zadanie 17\_19** o osobach, których druga litera nazwiska to „o” lub „u”. Dane należy posortować rosnąco według pól *nazwisko, imiona*.

## Zadanie 19\_12

Utwórz za pomocą **Query Designer** instrukcję **SQL**, działającą w bazie danych **MariaLisiecka**, w celu wyświetlenia sumy pieniędzy ofiarowanych przez darczyńców w 2017 roku.

# 20. Elementy wiedzy na temat instrukcji CREATE, ALTER, DROP

W rozdziale tym zapoznamy się z instrukcjami **Transact SQL** przeznaczonymi do definicji danych, tzn. do utworzenia, aktualizacji i usunięcia struktur danych. Są to instrukcje **CREATE**, **ALTER** i **DROP**. Niżej zaprezentowano przykłady poleceń, które pozwalają na utworzenie, aktualizację, usunięcie baz danych, tabel, indeksów i widoków.

## 20.1. Nazwa obiektu w systemie SQL Server

Przy tworzeniu instrukcji języka **Transact SQL** mamy do czynienia z identyfikatorami obiektów **SQL Server**. Każdy obiekt w systemie ma identyfikator. Obiektowi nadaje się identyfikator w momencie utworzenia. Identyfikator jest niezbędny dla pracy z tym obiektem. Maksymalna długość identyfikatora to 128 znaków. Standardowy identyfikator powinien zaczywać się od litery w standardzie Unicode lub od znaku podkreślenia (\_) albo znaku (@), czy też znaku (#). Kolejnymi znakami mogą być następujące:

- litery zdefiniowane w standardzie Unicode;
- cyfry dziesiętne z alfabetu łacińskiego lub innych alfabetów narodowych;
- znak (@), znak dolara (\$), znak (#) lub podkreślenie.

Identyfikator, który rozpoczyna się od znaku @ lub #, ma specjalnie przeznaczenie. Przykłady takich identyfikatorów będą podane dalej.

Identyfikator, który zawiera znaki niedozwolone, tzn. jest niestandardowy, powinien być ujęty w nawiasy kwadratowe bądź w cudzysłów.

Pełna nazwa obiektu zawiera:

- identyfikator serwera,
- identyfikator bazy danych,
- identyfikator schematu,
- identyfikator obiektu.

Na przykład, jeśli takim obiektem jest jedna z wcześniej utworzonych tabel, np. **Osoba**, to identyfikatorem obiektu jest **Osoba**, identyfikatorem serwera jest **[116-00\SQLEXPRESS]**. Identyfikator bazy danych to **MariaLisiecka**.

A czym zatem jest schemat? Schematy łączą w sobie różne obiekty bazy danych. Baza danych zawiera schematy, a schematy zawierają obiekty (jednym z przykładów obiektów są tabele). Schematy wymyślono na potrzeby usprawnienia mechanizmu zabezpieczenia danych.

Nazwa schematu obiektu jest zapisana w oknie jego właściwości, które możemy otworzyć, wybierając w menu podręcznym tabeli polecenie **Properties**. W oknie właściwości tabeli **Osoba** na zakładce **General** zobaczymy nazwę schematu: **dbo (database owner)**. Jest to schemat wbudowany, został przypisany naszej tabeli **Osoba** automatycznie. Zatem pełna nazwa utworzonej wcześniej tabeli **Osoba** to:

**[116-00\SQLEXPRESS].MariaLisiecka.dbo.Osoba.**

Dalej będziemy powszechnie stosować nazwy obiektów, więc należy zaznaczyć, że w instrukcjach **Transact SQL** nie jest konieczne stosowanie pełnych nazw obiektów. Obowiązują następujące zasady:

- jeżeli w nazwie obiektu brakuje identyfikatora serwera, to domyślnym serwerem jest lokalny serwer;
- jeśli brakuje nazwy bazy danych, to domyślną bazą danych jest bieżąca baza danych;
- domyślnym schematem jest schemat aktualnego użytkownika bazy danych.

## 20.2. Przykłady zastosowania instrukcji Transact SQL do utworzenia, modyfikacji oraz usunięcia bazy danych

Do utworzenia bazy danych stosujemy instrukcję **CREATE DATABASE**. Zapoznać się z jej bardzo złożoną strukturą możemy w dokumentacji języka **Transact SQL**. Najprostsze polecenie do utworzenia bazy danych ma postać:

**CREATE DATABASE nazwa\_bazy\_danych.**

W przypadku takiej instrukcji wszystkie niezbędne opcje będą zdefiniowane domyślnie. Lokalizacja utworzonej bazy danych także będzie domyślna (folder **DATA**).

W celu utworzenia bazy danych we własnym, a nie domyślnym folderze, w instrukcji **Transact SQL** należy podać nazwy plików bazy danych i ścieżki do nich, jak to prezentuje zaproponowane dalej rozwiązanie **Zadania 20\_1**.

## Zadanie 20\_1

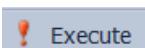
Zapisz instrukcję **Transact SQL** do utworzenia bazy danych o nazwie **MY\_BASE** w folderze **MY\_FOLDER**, na dysku **D**. Wykonaj polecenie. Odśwież węzeł **Databases**.

### Wykonanie

- Kliknij w menu programu na ikonę **New Query** i w pustym oknie, które pojawi się po prawej stronie od okna **Object Explorer**, zapisz instrukcję **T-SQL** tak, jak to przedstawiono na rysunku 20.1. Tu **my\_basedat.mdf** – podstawowy plik bazy danych, **my\_baselog.ldf** – plik dziennika transakcji.

```
CREATE DATABASE My_Base
ON
  (NAME = My_Base_dat,
   FILENAME='D:\My_Folder\My_basedat.mdf')
LOG ON
  (NAME = 'My_Base_log',
   FILENAME = 'D:\My_Folder\My_baselog.ldf')
```

Rys. 20.1. Przykładowa instrukcja **Transact SQL** do utworzenia bazy danych

- Wykonaj instrukcję, wybierając w menu ikonkę  . Po pozytwnym wykonaniu instrukcji pojawi się komunikat: **Command(s) completed successfully**.
- Odśwież węzeł **Databases**, wybierając w menu podręcznym polecenie **Refresh**, a zobaczysz nazwę utworzonej bazy danych.
- Zapisz instrukcję SQL w pliku o rozszerzeniu **sql** i zamknij okno **New Query**.

W przedstawionej na rysunku 20.1 instrukcji nazwy plików bazy danych (po słowie **NAME**) zapisano dwoma sposobami: z zastosowaniem apostrofów i bez nich w celu wskazania na taką możliwość.

W kolejnym zadaniu zaproponowano wykonać bazę danych o określonych opcjach **SIZE**, **MAXSIZE** oraz **FILEGROWTH**.

## Zadanie 20\_2

---

Zapisz instrukcję **Transact SQL** do utworzenia bazy danych o nazwie **lisiecka\_sql** na dysku lokalnym **C** w folderze **LISIECKA**. Utworzona baza danych powinna spełniać następujące wymagania:

- ❖ początkowy rozmiar pliku podstawowego o rozszerzeniu **.mdf**  
– 10 MB;
  - ❖ maksymalny rozmiar pliku podstawowego – 50 MB;
  - ❖ krok powiększania pliku podstawowego – 5 MB;
  - ❖ początkowy rozmiar pliku dziennika transakcji rozszerzeniu **.ldf** – 5 MB;
  - ❖ maksymalny rozmiar pliku dziennika transakcji – 25 MB;
  - ❖ krok powiększania pliku dziennika transakcji – 5 MB.
- 

### Wykonanie

```
|CREATE DATABASE lisecka_sql ON
|  (NAME =lisecka_sql_dat,
|   FILENAME='C:\LISIECKA\lisecka_sql_dat.mdf',
|   SIZE=10MB,
|   MAXSIZE=50MB,
|   FILEGROWTH=5MB)
| LOG ON
|  (NAME='lisecka_sql_log.ldf',
|   FILENAME='C:\LISIECKA\lisecka_sql_log.ldf',
|   SIZE=5MB,
|   MAXSIZE=25MB,
|   FILEGROWTH=5MB)
```

Rys. 20.2. Instrukcja do utworzenia bazy danych (**Zadanie 20\_2**)

Polecenie do zmiany bazy danych jest bardzo rozbudowane i zawsze zaczyna się od słów **ALTER DATABASE**. Niżej przedstawiono tylko jedną z wielu możliwości tej instrukcji – zmianę nazwy bazy danych.

## Zadanie 20\_3

---

Zapisz i wykonaj instrukcję **SQL** do zmiany nazwy bazy danych **My\_Base**. Nowa nazwa – to **Moja\_Baza**.

---

## Wykonanie

Instrukcje do wykonania zadania przedstawia rysunek 20.3.

```
ALTER DATABASE My_Base  
MODIFY NAME = Moja_Baza
```

Rys. 20.3. Instrukcja do zmiany nazwy bazy danych **My\_Base** (**Zadanie 20\_3**)

Do usunięcia baz danych przeznaczona jest instrukcja **DROP**.

## Zadanie 20\_4

Zapisz i wykonaj instrukcję do usunięcia bazy danych o nazwie **Moja\_Baza** (utworzonej przy wykonaniu **Zadania 20\_1**).

## Wykonanie

Instrukcje do wykonania **Zadania 20\_4** przedstawia rysunek 20.4.

```
DROP DATABASE Moja_Baza
```

Rys. 20.4. Instrukcja SQL do usunięcia bazy danych o nazwie **Moja\_Baza**

## 20.3. Przykłady zastosowania instrukcji Transact SQL do utworzenia, modyfikacji oraz usunięcia tabel i indeksów

Do utworzenia, modyfikacji oraz usunięcia tabeli przeznaczone są instrukcje **CREATE TABLE**, **ALTER TABLE** oraz **DROP TABLE**. W celu prezentacji ich działania, a także innych poleceń **Transact SQL** proponuje się utworzyć dalej prostą bazę danych o dwóch tabelach.

*Opis tabel, które mają być utworzone*

Naszym celem będzie zrealizowanie bazy danych **lisiecka\_sql**, zawierającej dane o parkingu firmy: o pracownikach oraz o samochodach firmowych przypisanych do tych pracowników.

### Założenia

- Każdy samochód jest przeznaczony do wykorzystania tylko przez jednego pracownika,

- Każdy pracownik może być przypisany tylko do jednego z samochodów firmy.
- Samochód zawsze jest do kogoś przypisany.
- Lista pracowników może zawierać osoby, które w dany czas nie są odpowiedzialne za któryś z samochodów.

### Dane o pracowniku

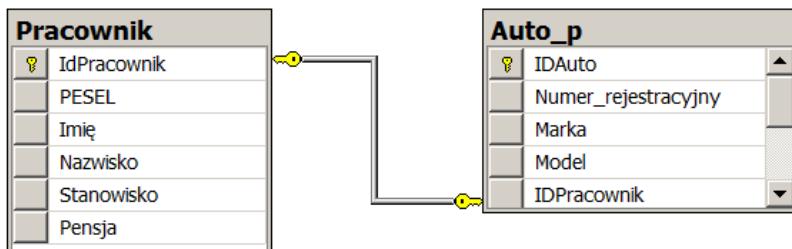
*PESEL, imię, nazwisko, stanowisko, pensja.*

### Dane o samochodzie

*numer\_rejestracyjny, marka, model, rocznik,  
jest\_właściwością\_firmy\_od\_daty, foto\_samochodu.*

Rozważając dziedzinę zagadnienia zgodnie z zasadami projektowania relacyjnych baz danych, przychodzimy do wniosku, że mają być wykonane dwie tabele, między którymi zachodzi związek jedno-jednoznaczny (tzn. 1:1).

Na rysunku 20.5 przedstawiono diagram, który zbudujemy po utworzeniu tych tabel w programie **Microsoft SQL Management Studio**.



Rys. 20.5.Tabele, które należy utworzyć za pomocą instrukcji **SQL**

Przy wykonaniu poniższych zadań będą zaproponowane różne sposoby utworzenia tych tabel, a także ich modyfikacji. Jednocześnie będą przedstawiane przykłady zapytań **Transact SQL** do usunięcia struktur danych.

W celu zaprezentowania instrukcji dodawania pól do tabeli na początku w tabeli **Auto\_p** będą utworzone nie wszystkie pola. Pola *Rocznik* i *Od\_daty*, *Foto* zostaną dodane później.

Każdą z utworzonych poniżej instrukcji **Transact SQL** należy po poważnym wykonaniu zapisać do pliku o rozszerzeniu **\*.sql**.

## Zadanie 20\_5

Zapisz i wykonaj instrukcję **Transact SQL** do utworzenia w bazie danych **lisiecka\_sql** tabeli **Pracownik** bez zdefiniowanego klucza podstawowego. Tabela powinna zawierać pola: *IdPracownik* (*Identity*), *PESEL*, *Imię*, *Nazwisko*, *Stanowisko*, *Pensja*. Zdefiniuj właściwe typy danych. Określ wszystkie pola, oprócz pola *Stanowisko*, jako wymagane.

### Wykonanie

Skrypt **Transact SQL** do zrealizowania **Zadania 20\_5** może mieć postać taką jak na rysunku 20.6.

```
CREATE TABLE [lisiecka_sql].[dbo].[Pracownik]
(IdPracownik int Identity NOT NULL,
PESEL char (11) NOT NULL,
Imię nvarchar(50) NOT NULL,
Nazwisko nvarchar(50) NOT NULL,
Stanowisko nvarchar(50) NULL,
Pensja Money NOT NULL
)
```

Rys. 20.6. Instrukcja **Transact SQL** do utworzenia tabeli *Pracownik* (**Zadanie 20\_5**)

Po słowach kluczowych **CREATE TABLE** w tej instrukcji podano nazwę bazy danych, w której będzie utworzona tabela, potem nazwę właściciela tabeli (**dbo**), a następnie już nazwę tabeli.

Dla pól *Imię*, *Nazwisko*, *Stanowisko* wybrano typ danych **nvarchar**, którego używamy dla danych tekstowych wielojęzycznych o zmiennej długości z wyznaczoną maksymalną liczbą znaków (używamy go specjalnie dla nazw w języku polskim).

Dla pola *PESEL* wybrano typ **char**, używany do danych tekstowych stałej długości.

Typ walutowy w **Transact SQL** to **money**.

Do opisu pól wymaganych dodajemy opcję **NOT NULL**, a niewymaganych – **NULL**.

Na rysunku 20.7 zaprezentowano inny skrypt, który także można zastosować do wykonania **Zadania 20\_5**. Dyrektywa **USE** służy tu do ukazania nazwy bazy danych, w której będzie stworzona tabela **Pracownik**.

Po dyrektywie **USE** zawsze występuje dyrektywa **GO**, która nie jest operatorem języka **Transact SQL**, ale jest rozpoznawalna przez system.

Po słowach kluczowych **CREATE TABLE** nie występuje już nazwa [**dbo**] – będzie ona dodawana przez system domyślnie.

```
USE [lisiecka_sql]
GO
CREATE TABLE [Pracownik]
(IdPracownik int Identity NOT NULL,
PESEL char (11) NOT NULL,
Imię nvarchar(50) NOT NULL,
Nazwisko nvarchar(50) NOT NULL,
Stanowisko nvarchar(50) NULL,
Pensja Money NOT NULL
)
```

Rys. 20.7. Instrukcja **Transact SQL** do utworzenia tabeli **Pracownik** zapisana razem z dyrektywami USE i GO (**Zadanie 20\_5**)

## Zadanie 20\_6

---

Zapisz instrukcję do usunięcia tabeli **Pracownik**.

---

### Wykonanie

Instrukcja **Transact SQL** do wykonania **Zadania 20\_6** może być zapisana w postaci przedstawionej na rysunku 20.8.

```
DROP TABLE [lisiecka_sql].dbo.[Pracownik]
```

Rys. 20.8. Instrukcja **Transact SQL** do usunięcia tabeli (**Zadanie 20\_6**)

## Zadanie 20\_7

---

Ponownie stwórz tabelę **Pracownik**. Wzoruj się na instrukcji przedstawionej na rysunku 20.7.

---

Po utworzeniu tabeli za pomocą polecenia **CREATE TABLE** wszystkich dalszych zmian jej struktury dokonujemy, stosując instrukcję **ALTER TABLE**.

## Zadanie 20\_8

---

Zapisz i wykonaj instrukcję **Transact SQL** do utworzenia w tabeli **Pracownik** klucza podstawowego.

---

## Wykonanie

Instrukcja **Transact SQL** do zrealizowania **Zadania 20\_8** może mieć postać taką, jak na rys. 20.9, gdzie **Klucz podstawowy Pracownik** – jest to nazwa ograniczenia integralnościowego (tzn. **CONSTRAINT**), jakim jest klucz podstawowy.

```
ALTER TABLE [lisiecka_sql].[dbo].[Pracownik]
ADD CONSTRAINT Klucz_podstawowy_Pracownik PRIMARY KEY (IdPracownika)
```

Rys. 20.9. Instrukcja **Transact SQL** do zdefiniowania klucza podstawowego tabeli (**Zadanie 20\_8**)

## Zadanie 20\_9

Zapisz i wykonaj instrukcję **Transact SQL** do utworzenia indeksu unikatowego dla pola *PESEL*. Skorzystaj z polecenia **CREATE**.

## Wykonanie

Po utworzeniu indeksu unikatowego dla pola *PESEL* nie będzie możliwości wprowadzenia do pola *PESEL* dwóch takich samych wartości.

Na rysunku 20.10 przedstawiono jedno z możliwych rozwiązań tego zagadnienia.

```
CREATE UNIQUE INDEX indeks_PESEL
ON [lisiecka_sql].[dbo].[Pracownik] (PESEL)
```

Rys. 20.10. Instrukcja **Transact SQL** do utworzenia indeksu unikatowego (**Zadanie 20\_9**)

## Zadanie 20\_10

W tabeli **Pracownik** usuń utworzony indeks unikatowy o nazwie *index\_PESEL*.

## Wykonanie

Na rysunku 20.11 zaprezentowano skrypt do wykonaniu zadania.

```
USE [lisiecka_sql]
GO
DROP INDEX [Pracownik].[indeks_PESEL]
```

Rys. 20.11. Skrypt **Transact SQL** do usunięcia indeksu (**Zadanie 20\_10**)

## Zadanie 20\_11

Skorzystaj z polecenia **ALTER TABLE** i utwórz ponownie w tabeli **Pracownik** indeks unikatowy o nazwie *index\_PESEL* dla pola **PESEL**.

---

### Wykonanie

```
USE [lisiecka_sql]
GO
```

```
ALTER TABLE Pracownik ADD CONSTRAINT indeks_PESEL UNIQUE (PESEL)
```

Rys. 20.12. Skrypt **Transact SQL** do utworzenia indeksu unikatowego  
**(Zadanie 20\_11)**

## Zadanie 20\_12

Za pomocą instrukcji **SQL** utwórz tabelę **Auto\_p** o polach: *IDAuto*, *Numer\_rejestracyjny*, *Marka*, *Model*, *IDPracownik*. Instrukcja powinna również definiować pole *IDAuto* jako klucz podstawowy, a także dodawać indeks unikatowy dla pola *IDPracownik*.

---

### Wykonanie

Indeks unikatowy jest potrzebny w celu spełnienia jednego z założeń projektu, że każdy samochód jest przeznaczony do wykorzystania tylko przez jednego pracownika.

Na rysunku 20.13 przedstawiono jeszcze jedną możliwą wersję skryptu do utworzenia tabeli: mamy w tym samym poleceniu **CREATE TABLE** opcje do definiowania klucza podstawowego oraz indeksu dla jednego z pól.

```
USE lisiecka_sql
GO
CREATE TABLE Auto_p
(
    IDAuto int Identity (1,1) NOT NULL CONSTRAINT Klucz_podst_Auto_p PRIMARY KEY,
    Numer_rejestracyjny varchar(50) NOT NULL,
    Marka nvarchar (50) NOT NULL,
    Model nvarchar (50) NOT NULL,
    IDPracownik int NOT NULL CONSTRAINT indeks_pracownik UNIQUE
)
```

Rys. 20.13. Instrukcja **Transcat SQL** do utworzenia tabeli **Auto\_p**  
**(Zadanie 20\_12)**

## Zadanie 20\_13

Zapisz i wykonaj instrukcję **Transact SQL** do utworzenia indeksu nieunikatowego dla pola *Marka* tabeli **Auto\_p**.

### Wykonanie

Na rysunku 20.14 zaprezentowano jedno z możliwych rozwiązań tego zadania.

```
USE [lisiecka_sql]
GO
CREATE INDEX indeks_marka ON Auto_p (Marka)
```

Rys. 20.14. Instrukcja do utworzenia indeksu nieunikatowego (**Zadanie 20\_13**)

## Zadanie 20\_14

za pomocą instrukcji **Transact SQL** utwórz połaczenie między tabelami **Pracownik** i **Auto\_p**, oznaczając pole *IDPracownik* tabeli *Auto\_p* jako klucz obcy. Odśwież bazę danych.

### Wykonanie

Na rysunku 20.15 zaprezentowano jedno z możliwych rozwiązań tego zadania.

```
USE lisiecka_sql
GO
ALTER TABLE Auto_p ADD CONSTRAINT Relacja_Pracownik_Auto_p
FOREIGN KEY (IdPracownik) REFERENCES Pracownik (IdPracownik)
```

Rys. 20.15. Instrukcja **Transact SQL** do zdefiniowania klucza obcego w tabeli **Auto\_p**

## Zadanie 20\_15

Zapisz i wykonaj instrukcję **Transact SQL** do zmiany rozmiaru pola *Stanowisko*, aby jego maksymalna długość była 40.

### Wykonanie

W tym przypadku faktycznie chodzi o zmianę typu danych pola. Na rysunku 20.16 przedstawiono instrukcję do zrealizowania zadania.

```
USE lisiecka_sql
GO
ALTER TABLE [Pracownik] ALTER COLUMN [Stanowisko] nvarchar(40)
```

Rys. 20.16. Instrukcja **SQL** do zmiany typu danych pola (**Zadanie 20\_15**)

## Zadanie 20\_16

---

Zapisz polecenie **Transact SQL** do zmiany ograniczenia **NOT NULL** pola *Pensja*: powinno ono pozwalać na wartości puste.

---

### Wykonanie

Instrukcję SQL przedstawiono na rys. 20.17

```
USE lisiecka_sql
GO
ALTER TABLE [Pracownik] ALTER COLUMN [Pensja] money NULL
```

Rys. 20.17. Instrukcja **SQL** do zdefiniowania opcji **NULL** (**Zadanie 20\_16**)

## Zadanie 20\_17

---

Zapisz i wykonaj instrukcję **Transact SQL** do dodania pola *Rocznik* do tabeli *Auto\_p* bazy danych *lisiecka\_sql*.

---

### Wykonanie

Skrypt **SQL** przedstawiono na rysunku 20.18.

```
USE lisiecka_sql
GO
ALTER TABLE [Auto_p] ADD [Rocznik] int NULL
```

Rys. 20.18. Skrypt **Transact SQL** do dodania nowego pola w tabeli (**Zadanie 20\_17**)

Należy tu zauważyć, że jeśli tabela już ma wprowadzone dane, to przy dodawaniu nowego pola powinniśmy zrobić go niewymagany (tzn. ustawić opcję **NULL**). Zmianę na **NOT NULL** (w razie takiej potrzeby) dokonujemy dopiero po wypełnieniu tego pola danymi.

Do usunięcia kolumny tabeli też stosujemy instrukcję **ALTER TABLE**, lecz po nazwie tabeli dodajemy **DROP COLUMN**.

## Zadanie 20\_18

Zapisz i wykonaj instrukcję **Transact SQL** do usunięcia pola **Rocznik** w tabeli **Auto\_p** bazy danych **lisiecka\_sql**.

### Wykonanie

Skrypt **SQL** przedstawiono na rysunku 20.19.

```
USE lisiecka_sql
GO
ALTER TABLE [Auto_p] DROP COLUMN [Rocznik]
```

Rys. 20.19. Instrukcja **SQL** do usunięcia pola w tabeli (**Zadanie 20\_18**)

## Zadanie 20\_19

Zapisz i wykonaj instrukcję **Transact SQL** do dodania nowego pola do tabeli **Auto\_p**. W polu będą przechowywane daty, w których samochody uzyskały zezwolenie przebywania na parkingu. Nazwa nowego pola - *Od\_daty*, powinno ono być wymagane, wartość domyślna pola - 2018/01/01.

### Wykonanie

Na rysunku 20.20 przedstawiono przykładowy skrypt **Transact SQL**, w którym nowe pole zadeklarowane jest jako wymagane, ponieważ tabela **Auto\_p** jest na razie pusta. Określenie wartości domyślnej zrealizowano za pomocą oddzielnej instrukcji. Takie rozwiązanie pozwala na definicję właściwej nazwy ograniczenia, jakim jest wartość domyślna.

```
USE Lisiecka_sql
GO
ALTER TABLE [Auto_p] ADD [Od_daty] date NOT NULL
ALTER TABLE [Auto_p] ADD CONSTRAINT [wart_domniemana_Od_daty]
    DEFAULT '2018-01-01' FOR [Od_daty]
```

Rys. 20.20. Instrukcja **SQL** do usunięcia pola w tabeli (**Zadanie 20\_19**)

Teraz, gdy nazwa ograniczenia integralnościowego **DEFAULT** jest znana, można go w razie potrzeby zmienić lub usunąć. W celu ilustracji pracy z tym ograniczeniem zaproponowano do wykonania kolejne zadanie.

## Zadanie 20\_20

Utwórz i wykonaj instrukcję **Transact SQL** do usunięcia wartości domyślnej pola *Od\_daty* w tabeli **Auto\_p**.

---

### Wykonanie

Instrukcja zamieszczona na rysunku 20.21 przedstawia operację usunięcia ograniczenia, jakim jest wartość domyślna pola *Od\_daty*.

```
USE Lisiecka_sql  
GO  
ALTER TABLE [Auto_p] DROP CONSTRAINT [wart_domniemana_Od_daty]
```

Rys. 20.21. Skrypt **Transact SQL** do usunięcia wartości domniemanej pola  
**(Zadanie 20\_20)**

Kolejne **Zadanie 20\_21** należy wykonać, aby tabela **Auto\_p** zawierała pola *Rocznik* i *Foto*, które będą potrzebne do dalszych ćwiczeń.

## Zadanie 20\_21

Utwórz samodzielnie i wykonaj instrukcję **Transact SQL** do dodania pola *Foto* typu **nvarchar(100)** w tabeli **Auto\_p**. W tym polu będziemy zapisywać ścieżki do plików przechowujących zdjęcia samochodów. Pole ma być niewymagane. Utwórz ponownie pole *Rocznik*, wykonując instrukcję z **zadania 20\_17**.

---

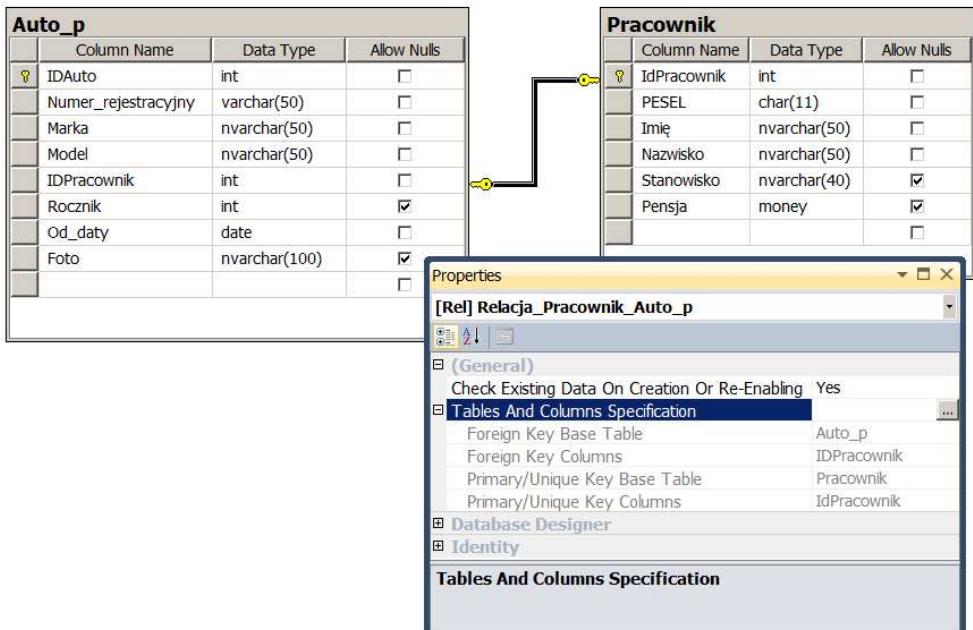
## Zadanie 20\_22

Za pomocą narzędzi graficznych zbuduj diagram bazy danych **lisiecka\_sql**. Zmień domyślną postać tabel w diagramie, wybierając w menu podręcznym każdej tabeli **Table view, Standard**. Otwórz okno właściwości relacji i przekonaj się, że dokonano właściwego połączenia: między polami o nazwie *IDPracownik*.

---

### Wykonanie

Na rysunku 20.22 zaprezentowano utworzony diagram, na którym widoczne są projekty tabel **Pracownik** i **Auto\_p**, a także okno właściwości utworzonej w **zadaniu 20\_14** relacji między tabelami.



Rys. 20.22. Diagram bazy danych lisiecka\_sql

Po rozpatrzeniu przykładów instrukcji do tworzenia i modyfikacji tabeli warto wspomnieć o instrukcji usunięcia tabeli. Czytelnik może sam przećwiczyć, jak ona działa po utworzeniu jakiejś dodatkowej tabeli. Jeśli na przykład w bazie danych **lisiecka\_sql** zostanie utworzona tabela o nazwie **Tabela**, to skrypt do jej usunięcia może być zapisany tak, jak na rysunku 20.23.

```
USE lisiecka_sql
GO
DROP TABLE Tabela
```

Rys. 20.23. Przykładowy skrypt do usunięcia tabeli

## 20.4. Przykłady utworzenia widoków

W relacyjnych systemach baz danych zawsze jest zrealizowana możliwość utworzenia tak zwanych widoków. Synonimy tego terminu – to przedstawienie, perspektywa lub angielski wyraz „view”. Widok jest obiektem bazy danych w postaci wirtualnej tabeli zdefiniowanej w skrypcie w języku **Transact SQL**, którego główną częścią jest instrukcja **SELECT**.

Widok jest przeznaczony do wykorzystania w zapytaniach **SQL** w tej samej roli, co i zwykłe tabele. Umieszczona w widoku instrukcja **SELECT** tworzy wirtualną tabelę. Treść widoku zmienia się dynamicznie na podstawie danych ulokowanych w tabelach, na których zostało zbudowane zapytanie **SELECT**.

Jest wiele powodów do zastosowania widoków. Jeden z nich to udostępnienie użytkownikowi nie wszystkich, a tylko niektórych danych tabeli.

Schemat instrukcji **SQL** do utworzenia standardowego widoku ma postać:

```
CREATE VIEW nazwa_widoku  
AS  
Instrukcja SELECT
```

Po utworzeniu widok przechowuje się w bazie danych. Potem może być wykorzystany w jakimś poleceniu **SQL**:

```
SELECT * FROM nazwa_widoku.
```

W systemie **SQL Server** istnieje rozbudowany interfejs graficzny dla pracy z widokami. Węzeł **Views** jest jednym z węzłów każdej bazy danych w oknie **Object Explorer**. W celu zrozumienia pojęcia widoku wykonaj kolejny przykład za pomocą narzędzi graficznych.

## Zadanie 20\_23

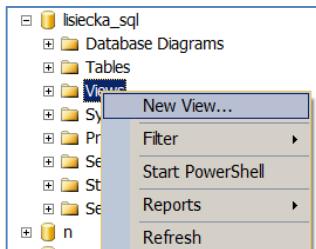
---

Utwórz widok w bazie danych **lisiecka\_sql** o nazwie **W\_Pracownik\_Auto** do wyświetlenia wszystkich danych z tabel **Pracownik** oraz danych ze wszystkich pól tabeli **Auto\_p** oprócz **IDPracownik**.

---

### Wykonanie

- a. W oknie **Object Explorer** rozwiń węzeł bazy danych **lisiecka\_sql**.
- b. W menu podręcznym węzła **Views** wybierz polecenie **New View...** (rys. 20.24).
- c. Skorzystaj z graficznych narzędzi, działając jak wcześniej przy generowaniu instrukcji **SELECT** (patrz **Rozdział 19**). Otrzymasz wynik w postaci przedstawionej na rysunku 20.25.



Rys. 20.24. Polecenie **New View** w menu podręcznym węzła **Views**

- d. Wybierz w menu ikonkę **Execute SQL** w celu sprawdzenia widoku pod względem poprawności składniowej i semantycznej.
- e. Po pomyślnym wykonaniu widoku zapisz go, nadając mu nazwę **W\_Pracownik\_Auto**, i zamknij okno.
- f. Odśwież węzeł **View**.

Ten sam widok można utworzyć, jeśli bezpośrednio w oknie **New Query** umieścimy i uruchomimy skrypt przedstawiony na rysunku 20.26.

Należy tu zauważyć, że jeśli po wykonaniu i odświeżeniu widoku **W\_Pracownik\_Auto** wygenerujemy jego skrypt za pomocą polecenia z menu podręcznego, to zobaczymy w nim dodatkowe opcje dodane automatycznie.

Column	Alias	Table	Output	Sort Type
IDPracownik		Pracownik	<input checked="" type="checkbox"/>	
PESEL		Pracownik	<input checked="" type="checkbox"/>	
Imię		Pracownik	<input type="checkbox"/>	

```

SELECT dbo.Pracownik.IDPracownik, dbo.Pracownik.PESEL, dbo.Pracownik.Imię,
       dbo.Pracownik.Nazwisko, dbo.Pracownik.Stanowisko, dbo.Pracownik.Pensja,
       dbo.Auto_p.IDAuto, dbo.Auto_p.Numer_rejestracyjny, dbo.Auto_p.Marka,
       dbo.Auto_p.Model, dbo.Auto_p.Rocznik, dbo.Auto_p.Od_daty, dbo.Auto_p.Foto
FROM dbo.Auto_p INNER JOIN
      dbo.Pracownik ON dbo.Auto_p.IDPracownik = dbo.Pracownik.IDPracownik
  
```

Rys. 20.25. Polecenie **New View** w menu podręcznym węzła **Views**

```
use lisiecka_sql
GO
CREATE VIEW [dbo].[W_Pracownik_Auto]
AS
SELECT dbo.Pracownik.* , dbo.Auto_p.IDAuto,
       dbo.Auto_p.Numer_rejestracyjny, dbo.Auto_p.Marka,
       dbo.Auto_p.Model, dbo.Auto_p.Rocznik,
       dbo.Auto_p.Od_dat�, dbo.Auto_p.Foto
FROM dbo.Auto_p INNER JOIN dbo.Pracownik
ON dbo.Auto_p.IDPracownik = dbo.Pracownik.IDPracownik
```

Rys. 20.26. Skrypt do utworzenia widoku **W\_Pracownik\_Auto**

W kolejnym zadaniu proponuje się wykonać widok, który dalej (w **Rozdziale 21**) wykorzystywany jest w aplikacji klienckiej przy dodawaniu nowego rekordu do tabeli **Auto\_p**. Zgodnie z założeniem pole *IDPracownika* w tej tabeli zawiera wartości unikatowe.

## Zadanie 20\_24

W bazie danych **lisiecka\_sql** utwórz widok do wyświetlenia danych z tabeli **Pracownik** w postaci: *IDPracownik*, *PESEL*, *Imię*, *Nazwisko*, *Osoba*. Tu *Osoba* – pole obliczeniowe, w którym wyświetlane są razem *Nazwisko*, spacja, *Imię*.

Należy wyświetlić dane tylko z tych rekordów, które dotyczą pracowników nie przypisanych do żadnego auta w tabeli **Auto\_p**.

### Wykonanie

Na rysunku 20.27 przedstawiono skrypt do utworzenia widoku, w którym skorzystano z możliwości dodawania synonimów do nazw tabel. Przy definiowaniu kryterium zastosowano podzapytanie (patrz **Rozdział 11**).

```
USE lisiecka_sql
GO
CREATE VIEW [dbo].[Pracownik_bez_auta]
AS
SELECT p.IDPracownik, p.PESEL, p.Imię, p.Nazwisko,
       p.Nazwisko + ' ' + p.Imię AS Osoba
FROM dbo.Pracownik AS p LEFT OUTER JOIN dbo.Auto_p AS a
  ON p.IDPracownik = a.IDPracownik
 WHERE (NOT (p.IDPracownik IN (SELECT IDPracownik FROM dbo.Auto_p)))
```

Rys. 20.27. Skrypt do utworzenia widoku **Pracownik\_bez\_auta**

Modyfikacji i usunięcia widoków wykonuje się za pomocą instrukcji **ALTER VIEW** oraz **DROP VIEW**.

## Zadania do samodzielnego wykonania

### Zadanie 20\_25

Zapisz i wykonaj instrukcję do utworzenia nowej bazy danych o nazwie **Wpłaty\_klientów**.

### Zadanie 20\_26

Zapisz i wykonaj skrypt do utworzenia w bazie danych **Wpłaty\_klientów** tabeli o nazwie **Klient** o polach: *ID\_Klient*, *Nazwisko*, *Imiona*, *Plec*, *PESEL*.

Pole *ID\_Klient* powinno zawierać liczby całkowite i wypełniać się automatycznie. Jest to klucz podstawowy tabeli. Wszystkie pola należy określić jako wymagane. Dla pola *Płeć* należy określić wartość domniemaną.

### Zadanie 20\_27

Zapisz i wykonaj instrukcję do utworzenia indeksu unikatowego dla pola *PESEL* w tabeli **Klient**.

### Zadanie 20\_28

Zapisz i wykonaj instrukcję do utworzenia indeksu nieunikatowego dla pola *Nazwisko* w tabeli **Klient**.

### Zadanie 20\_29

Zapisz i wykonaj wsad do utworzenia tabeli o nazwie **Kwoty** w bazie danych **Wpłaty\_klientów**. Tabela ma zawierać pola: *ID*, *Kwota*, *Data\_wpłaty*, *ID\_Klienta*, *Komentarz*. Wszystkie pola, oprócz *Komentarz*, mają być wymagane.

### Zadanie 20\_30

Zapisz i wykonaj polecenie **SQL** do zmiany tabeli **Kwoty** w celu zdefiniowania pola *ID\_Klienta* jako klucza obcego tabeli służącego do połączenia z tabelą **Klient**.

## 21. Zastosowanie programu Access do utworzenia aplikacji klienta bazy danych

Jest wiele języków programowania, które dysponują rozbudowanymi narzędziami do utworzenia aplikacji klienta bazy danych. Jednak w niektórych przypadkach warto skorzystać z możliwości oprogramowania **Microsoft Access** i utworzyć aplikację kliencką, wcale nie pisząc kodu. Można również utworzyć złożoną aplikację bazodanową klient-serwer, stosując wbudowany w **Access** język **Visual Basic for Application**. W tym przypadku również można wiele operacji zrealizować za pomocą kreatorów oraz makr programu **Microsoft Access**.

Wykonajmy dalej prostą aplikację kliencką przeznaczoną do wprowadzenia, modyfikacji, usunięcia oraz przeglądania danych w tabelach **Pracownik** oraz **Auto\_p** bazy danych **lisiecka\_sql**.

### 21.1. Połączenie programu Microsoft Access 2016 z bazą danych ulokowaną w systemie Microsoft SQL Server 2014

Utworzmy dalej pustą bazę danych w systemie **Microsoft Access 2016** i połączymy ją z jedną z baz danych ulokowanych w systemie **SQL Serwer**. W tym celu skorzystamy z protokołu **ODBC** (ang. **Open DataBase Connectivity** - otwarte łącze baz danych).

#### Zadanie 21\_1

---

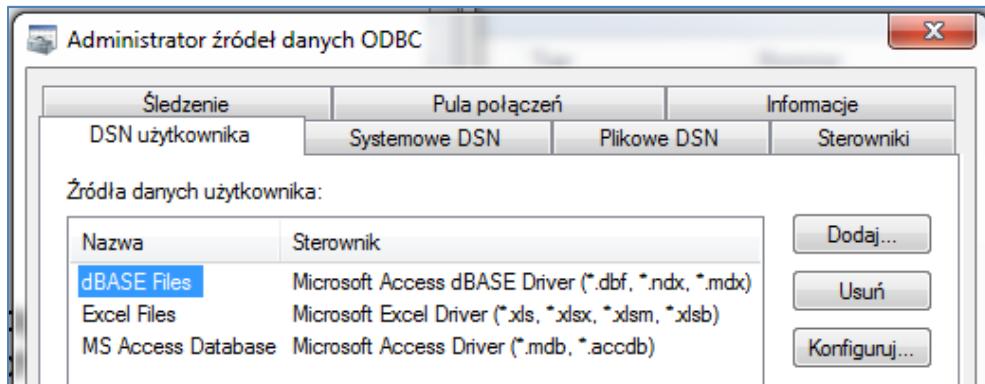
Otwórz w **Panelu sterowania** okno dialogowe **Administrator źródeł danych ODBC** i zdefiniuj nowe źródło danych – bazę danych **lisiecka\_sql** umieszczoną w systemie **Microsoft SQL Server 2014**.

---

#### Wykonanie

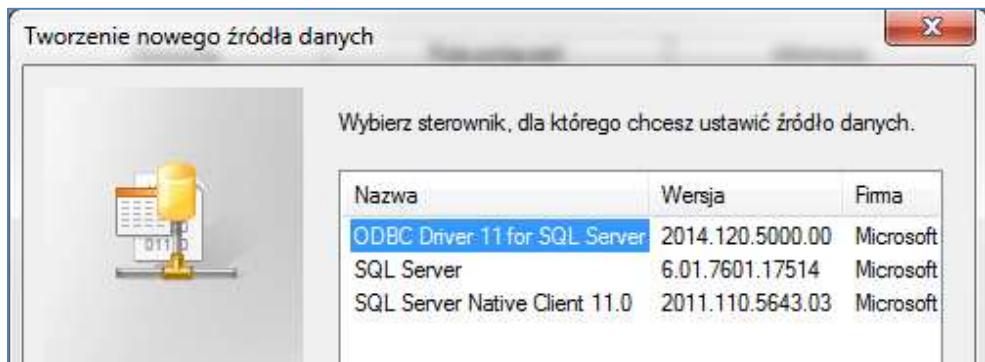
- a. Kliknij przycisk **Start** i wybierz polecenie **Panel sterowania**.

- b. W oknie **Panelu sterowania** wybierz **System i zabezpieczenia**, a potem **Narzędzia administracyjne**.
- c. W oknie dialogowym **Narzędzia administracyjne** kliknij dwukrotnie pozycje **Źródła danych (ODBC)** – zostanie wyświetcone okno dialogowe **Administrator źródeł danych ODBC** (rys. 21.1).



Rys. 21.1. Okno dialogowe **Administrator źródeł danych ODBC**

- d. Na zakładce **DSN użytkownika** wybierz **Dodaj** (rys. 21.1).
- e. W kolejnym oknie wybierz sterownik do połączenia z **Microsoft SQL Server 2014**: kliknij na nazwie **ODBC Driver 11 for SQL Server** (rys. 21.2).

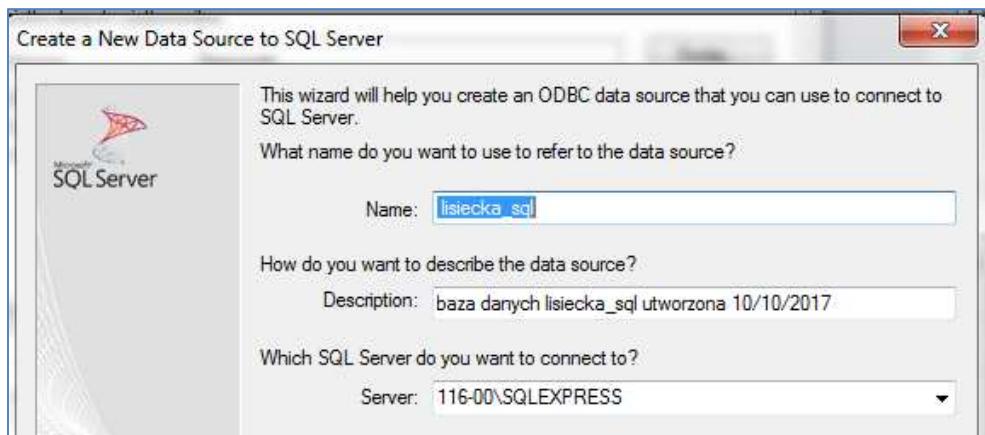


Rys. 21.2. Wybór sterownika

- f. Wybierz **Zakończ**.
- g. W kolejnym oknie dialogowym **Create a New Data Source to SQL Server** wprowadź (rys. 21.3):
  - o w polu **Name** – nazwę nowego źródła danych **ODBC**,

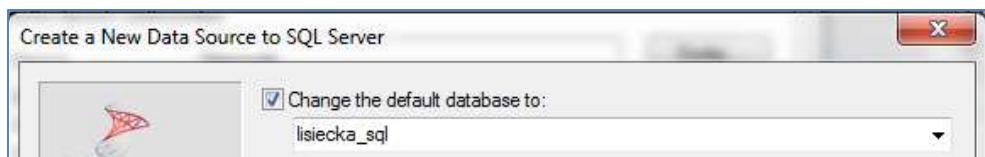
- o w polu **Description** – opis nowego źródła danych **ODBC**. Wybierz z listy nazwę serwera (można ją również wpisać; może być tu także podany numer IP).

h. Wybierz **Dalej**.



Rys. 21.3. Definiowanie nazwy źródła danych oraz nazwy serwera, z którym będzie połączona aplikacja klienta

- i. W kolejnym oknie nie zmieniaj nic i wybierz **Dalej**.
- j. W następnym oknie (rys. 21.4) zdefiniuj domyślną bazę danych. W tym celu kliknij pole **Change the default database to** oraz wybierz z listy rozwijanej nazwę bazy danych **lisiecka\_sql**.



Rys. 21.4. Definiowanie domyślnej bazy danych

- k. Wybierz **Dalej**.
- l. W kolejnym oknie wybierz **Zakończ**.
- m. W oknie **ODBC Microsoft SQL Server Setup** testuj źródło danych za pomocą przycisku **Test Data Source**. W wyniku testowania powinno pojawić się okienko, które zasygnalizuje, że testowanie zakończyło się sukcesem. Wybierz w nim **OK**.
- n. W oknie **Administratora źródeł danych** zobaczysz nowo utworzone DSN. Wybierz **OK**.

## Zadanie 21\_2

Wykonaj samodzielnie:

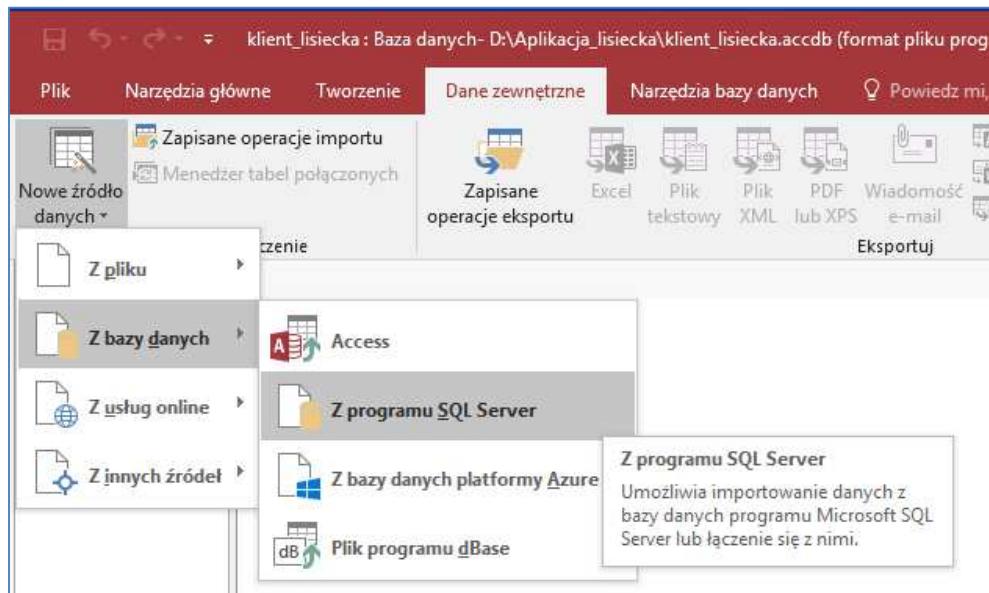
- ❖ uruchom **Microsoft Access 2016** i zdefiniuj folder na dysku lokalnym jako zaufaną lokalizację;
- ❖ utwórz pustą bazę danych **Microsoft Access 2016** w tym folderze (nazwa tej bazy danych – **klient\_lisiecka.accdb**);
- ❖ zamknij pustą tabelę, która otworzy się po stworzeniu bazy danych.

## Zadanie 21\_3

Zastosuj narzędzia importowania danych w celu utworzenia połączenia **Microsoft Access 2016** z bazą danych **lisiecka\_sql** na serwerze SQL.

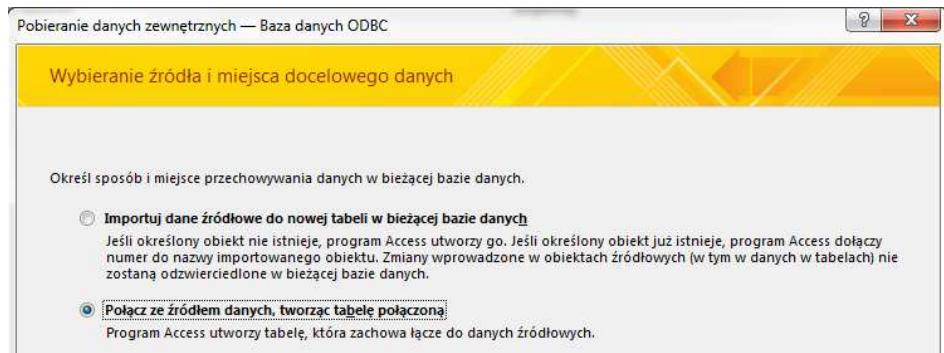
### Wykonanie

- a. Wybierz w karcie **Dane zewnętrzne** wstążki ikonkę **Nowe źródło danych** (rys. 21.5).
- b. Wybierz z listy rozwijającej się tej ikonki opcję **Z bazy danych**, a dalej – **Z programu SQL Server** (rys. 21.5).

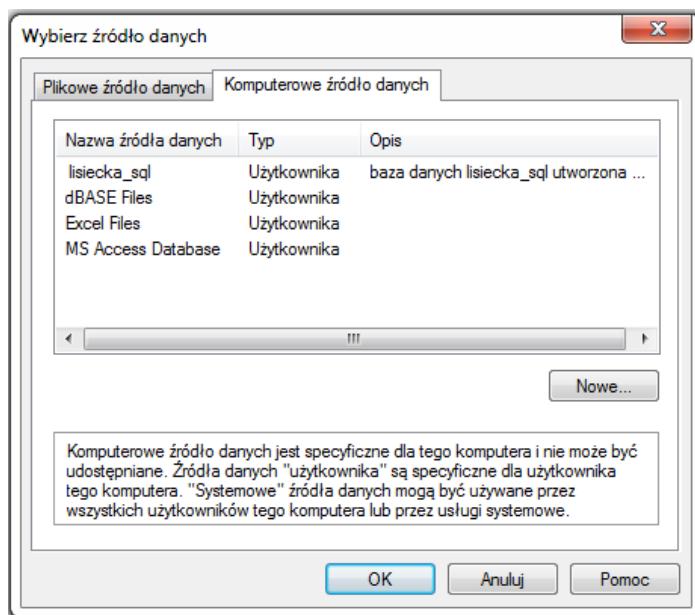


Rys. 21.5. Pierwszy krok przy łączeniu się programu **Access 2016** z bazą danych programu **Microsoft SQL Server**

- c. W oknie **Pobieranie danych zewnętrznych – Baza danych ODBC** (rys. 21.6) wybierz opcję **Połącz ze źródłem danych, tworząc tabelę połączoną**.
- d. W oknie **Wybierz źródło danych** (rys. 21.7) w zakładce **Komputerowe źródło danych** kliknij na nazwie **lisiecka\_sql** i wybierz **OK**.

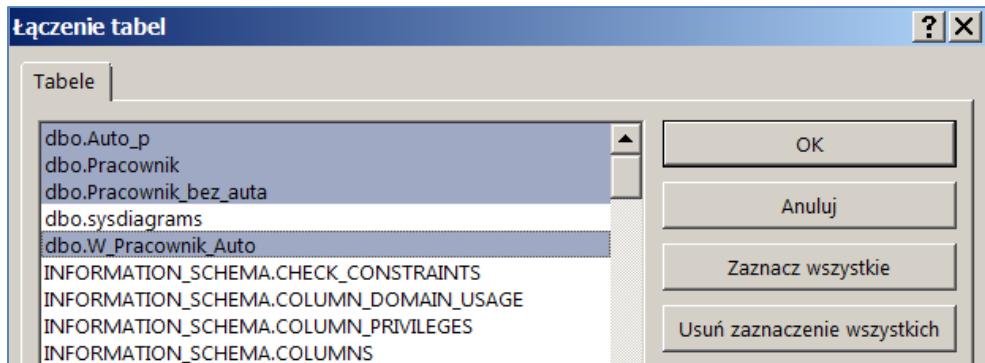


Rys. 21.6. Drugi krok przy łączeniu się programu **Access 2016** z bazą danych programu **Microsoft SQL Server**



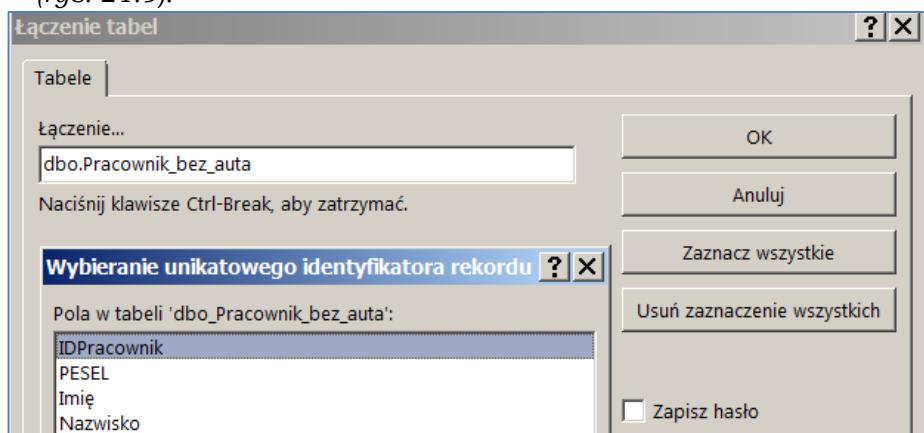
Rys. 21.7. Trzeci krok przy łączeniu się programu **Access 2016** z bazą danych programu **Microsoft SQL Server**

- e. W oknie **Łączenie tabel** (rys. 21.8) zaznacz nazwy tabel **dbo.Pracownik**, **dbo.Auto\_p** oraz **widoków** i wybierz **OK**.



Rys. 21.8 .Okno przedstawiające tabele i widoki bazy danych **lisiecka\_sql**, z którymi można nawiązać połączenie

- f. W oknie, które zostanie otworzone w celu identyfikacji rekordów widoku **Pracownik\_bez\_auta**, należy także wybrać pole **IDPracownik** (rys. 21.9).



Rys. 21.9. Wybieranie identyfikatora dla widoku **Pracownik\_bez\_auta**

- g. Analogicznie należy postąpić w kolejnym oknie otwartym do zidentyfikowania rekordów widoku **W\_Pracownik\_Auto**.  
 h. Teraz, gdy zostało nawiązane połączenie między programem **Microsoft Access** a systemem **SQL Server**, w okienku nawigacji będą wyświetlane tabele i widoki bazy danych **lisiecka\_sql** (rys. 21.10).

Tabele	
»	dbo_Auto_p
»	dbo_Pracownik
»	dbo_Pracownik_bez_auta
»	dbo_W_Pracownik_Auto

Rys. 21.10. Widok tabel i widoków bazy danych **lisiecka\_sql** w oknie programu **Microsoft Access**

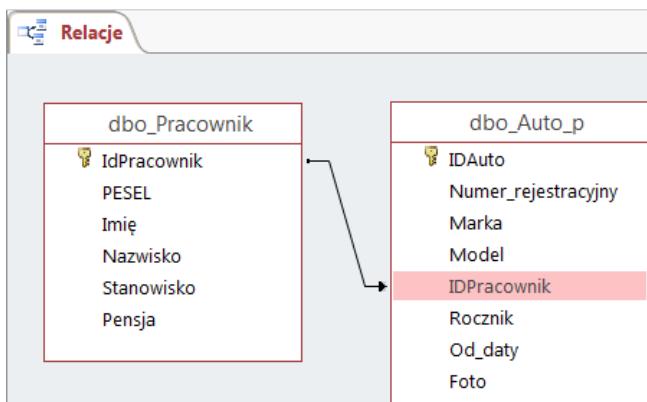
Użytkownik może otworzyć tabelę w widoku arkusza danych w celu ich dodania, usunięcia bądź modyfikacji. Lecz widok projektu tabeli służy tylko do oglądu. Wszystkich zmian dokonujemy wyłącznie w źródłowych tabelach ulokowanych na serwerze.

W okienku **Relacje** należy utworzyć relację między tabelami. Zgodnie z założeniem między tabelami bazy danych **lisiecka\_sql** zachodzi związek binarny jeden-do-jeden.

## Zadanie 21\_4

Utwórz w okienku **Relacje** połączenie między tabelami tak, jak przedstawiono to na rysunku 21.11. Postępuj jak zwykle przy połączeniu tabel w oknie **Relacje** (patrz Rozdział 5).

### Wykonanie



Rys. 21.11. Okienko **Relacje** w oknie **klient\_lisiecka.accdb**

## Zadanie 21\_5

Przetestuj działanie kreatora o nazwie **Menedżer tabel** połączonych, którego ikonka jest umieszczona na wstążce w zakładce **Dane zewnętrzne**.

## 21.2. Wykonanie przykładowej aplikacji klienta bazy danych

Aplikacja klienta bazy danych powinna dysponować narzędziami do przeglądania danych oraz do ich dodawania, usunięcia i modyfikacji.

Czytelnik może zbudować aplikację kliencką dla umieszczonej w systemie **SQL Server** bazy danych **lisiecka\_sql**, wykonując **Zadanie 21\_6**. Widok formularza aplikacji przedstawia rysunek 21.12.



Rys. 21.12. Widok aplikacji klienckiej bazy danych **lisiecka\_sql**

## Zadanie 21\_6

Utwórz aplikację klienta w postaci formularza nawigacji o dwóch zakładkach, jak przedstawiono to na rysunku 21.12. Na zakładkach **Auty** i **Pracownicy** ulokuj odpowiednio formularze, które pozwolą na wykonywanie przeglądania, usunięcia, modyfi-

kacji oraz dodania nowych danych o samochodach i pracownikach do nich przypisanych.

Wyszukiwanie danych o aucie zrealizuj dwoma sposobami:

- 1) za pomocą zwykłego nawigowania między rekordami oraz
- 2) za pomocą filtrowania listy samochodów. Po wyborze pewnego samochodu powinny się wyświetlać dane o nim oraz o pracowniku przypisanym do auta.

Edycja lub dodawanie danych o nowym samochodzie powinny odbywać się w nowym oknie, które zostanie wyświetcone po naciśnięciu odpowiednio na przyciski **Edytuj** lub **Dodaj**. Do pracy z danymi o pracownikach utwórz formularz tabelaryczny.

## Wykonanie

### I. Prace przygotowawcze

- a. Umieść w tym samym folderze, który zawiera plik aplikacji, pliki ze zdjęciami samochodów. Każda nazwa pliku będzie zapisywana do tabeli razem z rozszerzeniem pliku.
- b. Zmień opcje bazy danych. Aby można było sterować rozmiarami okien formularzy, w oknie **Opcje programu Access** w zakładce **Bieżąca baza danych** w grupie **Opcje okna dokumentu** wybierz: **Nakładające się okna**.
- c. Zbuduj kwerendę **kLista\_aut**, która posłuży jako źródło wierszy dla listy samochodów (rys. 21.13).

Pole:	IDAuto	Numer_rejestracyjny	Marka	Model
Tabela:	dbo_Auto_p	dbo_Auto_p	dbo_Auto_p	dbo_Auto_p
Sortuj:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Pokaż:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kryteria:				

Rys. 21.13. Kwerenda **kLista\_aut** w widoku projektu

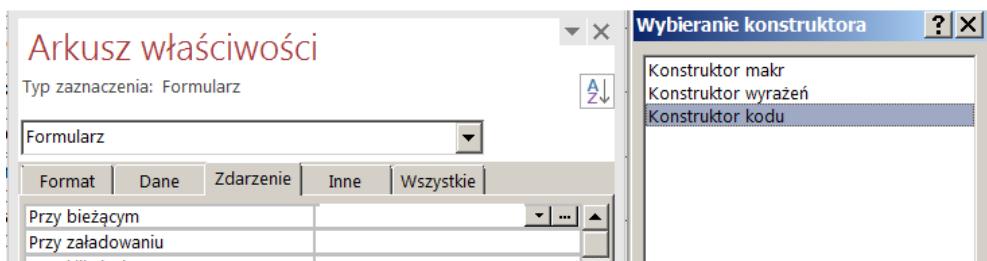
- d. Utwórz w widoku projektu pusty formularz kolumnowy, którego źródłem danych będzie tabela **dbo\_Auto\_p**. Nazwij ten formularz **Fdbo\_Auto\_p**.

- e. Ulokuj pola tabeli **dbo\_Auto\_p** w sekcji **Szczegóły** formularzu, jak pokazano to na rysunku 21.12. Przy tym pole *IDPracownik* zrób niewidocznym, a pole *IDAuto* pozostaw widocznym, lecz ustaw kolor czcionki taki sam jak kolor tła formularza.
- II. Utworzenie formularza do przeglądania danych o samochodach
- f. Dodaj standardowe przyciski do nawigowania między rekordami, korzystając z kreatora przycisków. Dodatkowo można, jak pokazano to na rysunku 21.12, ukryć przyciski nawigacyjne w dolnej części formularza, zaznaczając to w odpowiednim wierszu **Arkusza danych** formularza w zakładce **Format**.
- g. Umieść w formularzu formant **Obraz**, w jego **Arkuszu właściwości** nadaj mu nazwę **Obraz\_autu** (rys. 21.14).



Rys. 21.14. Arkusz właściwości

- h. Dodaj kod VBA do formularza w celu wyświetlenia zdjęć w trakcie przeglądu danych o samochodach. W tym celu dokonaj następujących czynności:
- o otwórz **Arkusz właściwości** formularza **Fdbo\_Auto\_p**;
  - o na zakładce **Zdarzenia** w wierszu **Przy bieżącym** kliknij na przycisku o trzech kropkach;
  - o w wyświetlonym okienku wybierz **Konstruktora kodu** (rys. 21.15);



Rys. 21.15. Wybieranie **Konstruktora kodu**

- o zapisz w oknie, które się otworzy, kod w języku **VBA** przeznaczony do przeglądu zdjęć aut (rys. 21.16).

```

Private Sub Form_Current()
    Dim s_path As String
    On Error GoTo Err_obrazu
    s_path = Application.CurrentProject.Path & "\"
    Me.foto.SetFocus
    If Me.foto <> "" Then
        Me.Obraz_auta.Picture = s_path & Me.foto
    Else
        Me.Obraz_auta.PictureData = Null
    End If
    Exit Sub

Err_obrazu:
    Me.Obraz_auta.PictureData = Null
End Sub

```

Rys. 21.16. Tekst procedury zdarzenia **Przy bieżącym** formularza **Fdbo\_Auto\_p**

Tu:

- **s\_path** – zmienna tekstowa, do której przekazujemy ścieżkę do pliku ze zdjęciem auta (wychodzimy tu z założenia, że plik obrazu jest w folderze zawierającym plik aplikacji);
  - **Me** – oznacza dany formularz;
  - **Me.Foto** – formant *Foto* na formularzu;
  - **Me.Foto.SetFocus** – sprawia, że staje się aktywny formant **Foto** w formularzu;
  - **Me.Obraz\_auta** – oznacza formant **Obraz\_auta**, a **Picture** – jego właściwość wyświetlania właśnie obrazu;
  - programistycznej instrukcji **IF Then Else użyto** do sprawdzania, czy jest podana ścieżka do pliku ze zdjęciem w polu tekstowym **Foto**, czy nie. Jeśli tak, to otwiera się ten plik i w formancie *Obraz* widzimy zdjęcie.
- i. Dodaj kod **VBA** do zdarzenia **po aktualizacji** formantu **Foto**. W tym celu zaznacz w widoku projektu formant **Foto** i dodaj w jego arkuszu właściwości procedurę VBA do zdarzenia **Po aktualizacji** (rys. 21.17). W danym przypadku, aby nie powtarzać tego samego kodu, po prostu odwołujemy się do poprzedniej procedury.

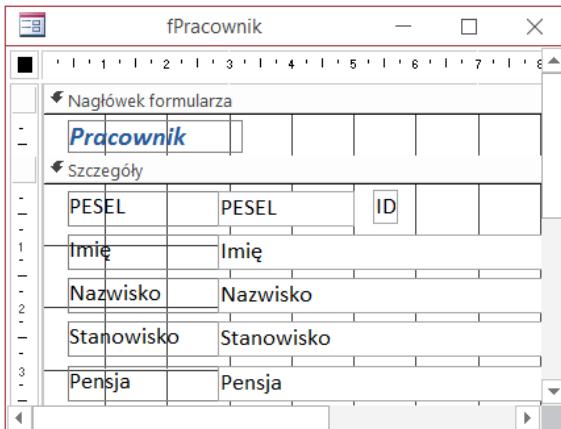
```

Private Sub foto_AfterUpdate()
    Form_Current
End Sub

```

Rys. 21.17. Procedura zdarzenia po aktualizacji formantu **Foto**

- j. W celu oddzielenia wszystkich pól dotyczących auta można, jak to przedstawiono na rysunku 21.12, dodać formant **Prostokąt** i zastosować do niego **Efekt specjalny – Cieniowany**.
- k. Zapisz i zamknij formularz **Fdbo\_Auto\_p**.
- l. Utwórz nowy formularz kolumnowy **fPracownik**, jak przedstawia to rysunek 21.18. Źródłem formularza ustaw tabele **dbo\_Pracownik**. Pole *ID* zróbi niewidoczne, ukryj selektory rekordów i przyciski nawigacyjne, zamknij formularz.



Rys. 21.18. Widok projektu formularza **fPracownik**

- m. Otwórz formularz **Fdbo\_Auto\_p**. Umieść w nim formant **Podformularz/Podraport**. Przy tym skorzystaj z możliwości podformularzy i dopasuj do siebie dane samochodu i pracownika, definiując samodzielnie, które pola łączą główny formularz z podformularzem (tzn. pola o nazwie *IDPracownik*).
- n. Umieść w nagłówku formularza **Fdbo\_Auto\_p** formant **Pole listy**. Przy tym w oknach **Kreatora pól list**:
- o wybierz opcję **Obiekt pole listy ma pobierać wartości z innej tabeli lub kwerendy**;
  - o wybierz: **Kwerenda kLista\_aut**;
  - o wybierz wszystkie pola;
  - o ustaw sortowanie według pól *Numer\_rejestracyjny*, *Marka*, *Model*;
  - o w kolejnych oknach wybieraj **Dalej i Zakoncz**.
- o. Otwórz okno właściwości utworzonej listy i :
- o w zakładce **Inne** wpisz nazwę: **Lista**;
  - o w zakładce **Format**: szerokość pierwszej kolumny – 0 cm;
  - o w zakładce **Format – Nagłówki kolumn Tak**;
  - o pozostałe zmiany kosmetyczne zrealizuj, zmieniając opcje w zakładce **Format**.

- p. Dodaj formant **Przycisk**, umieść go obok listy, nie korzystaj z **Kreatora przycisków poleceń**: wybierz **Anuluj**. W **Arkuszu właściwości** przycisku zdefiniuj:
- o w zakładce **Inne** wpisz nazwę – **pFiltruj**;
  - o w zakładce **Format** w wierszu **Tytuł** wpisz: **Filtruj**;
  - o w zakładce **Zdarzenie** dodaj kod VBA do zdarzenia **Przy kliknięciu** (rys. 21.19).

```
Private Sub PFiltruj_Click()
Dim rm As Variant

Me.FilterOn = False

Me.Lista.SetFocus
If Me.Lista.Column(0) <> "" Then
    rm = Me.Lista.Column(0)
    Me.IDAuto.SetFocus
    Me.Filter = "[IDAuto] = " & rm
    Me.FilterOn = True
End If

End Sub
```

Rys. 21.19. Procedura VBA dla zdarzenia **Przy kliknięciu** formantu **pFiltruj**

- q. Umieść jeszcze jeden przycisk obok listy – do kasowania filtru. Tak samo jak wcześniej, nie korzystaj z **Kreatora przycisków poleceń**. Nadaj formantowi nazwę **pWszystkie**. Dodaj do jego zdarzenia **Przy kliknięciu** nie procedurę zdarzenia, lecz makro, projekt którego przedstawia rysunek 21.20.

#### UruchomPolecenieMenu

Polecenie UsuńFiltrSortowanie

Rys. 21.20. Makro osadzone do zdarzenia **Przy kliknięciu** przycisku **pWszystkie**

- r. Dodaj możliwość filtrowania danych po dwukrotnym kliknięciu na nazwie auta. W tym celu do zdarzenia **Przy kliknięciu dwukrotnym** dodaj procedurę, tekst której podano na rysunku 21.21.

```
Private Sub Lista_DblClick(Cancel As Integer)
PFiltruj_Click
End Sub
```

Rys. 21.21. Opracowanie zdarzenia **Przy kliknięciu dwukrotnym** formantu **Lista**

- s. Umieść jeszcze jeden przycisk **Odśwież dane** i dodaj do jego zdarzenia **Przy kliknięciu** makro, które utwórz w projekcie, wybierając akcję **UruchomPolecenieMenu**, a potem **Odśwież**.

III. Utworzenie formularza wykorzystywanego przy dodaniu i modyfikacji danych o samochodach

- t. Zamknij formularz **Fdbo\_Auto\_p**. Wykonaj nowy formularz kolumnowy o nazwie **fAuto** (rys. 21.22) w oparciu o tabelę **dbo\_Auto\_p**.

IDAuto	6	<b>Cofnij</b>
Numer_rejestracyjny	BI1234V	<b>Zapisz</b>
Marka	Toyota	<b>Powrót</b>
Model	Yaris	
Rocznik	2016	
Od_daty	2017-01-01	
Foto	yaris.png	
IDPracownik	2	

Rys. 21.22. Formularz **fAuto** w widoku formularza

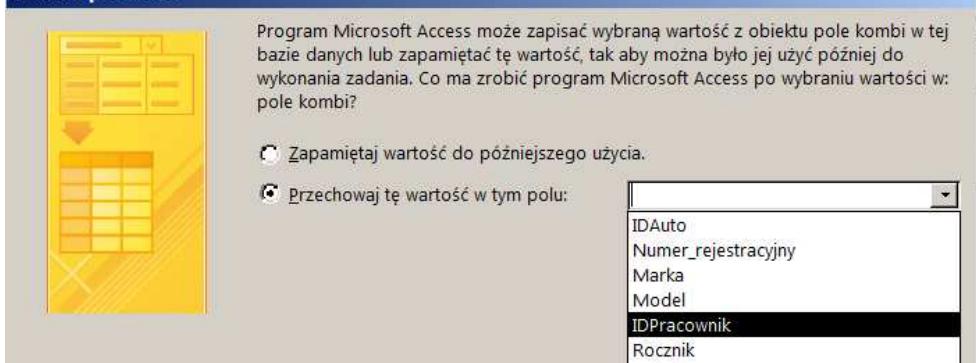
W widoku projektu formularza:

- o umieść w sekcji **Szczegóły** formularza pola tabeli **dbo\_Auto\_p**;
- o w nagłówku formularza umieść formant **Obraz** i dokonaj takich samych czynności jak przy tworzeniu formularza **Fdbo\_Auto\_p** w celu wyświetlenia zdjęcia auta;
- o umieść standardowe przyciski **Cofnij** i **Zapisz** przeznaczone odpowiednio do cofnięcia wpisywanego rekordu lub do zapisywania rekordu, skorzystaj z pomocy **Kreatora przycisków poleceń**.

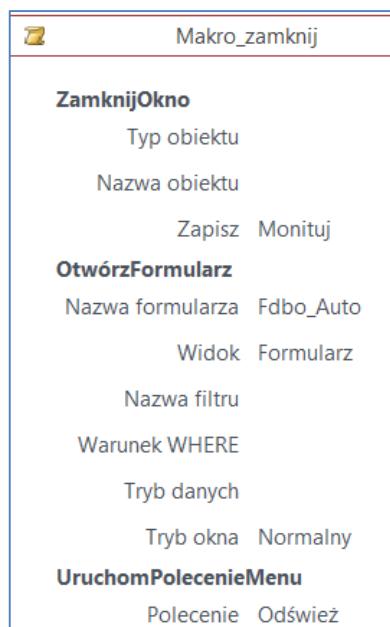
Należy tu zauważyć, że wykorzystanie standardowego makra do zapisywania rekordu nie jest do końca wystarczające. Należałoby tu jeszcze umieścić sprawdzenie, czy zostały wpisane dane do pól wymaganych i opracować sytuacje, gdy takich wartości brak. Inaczej w każdej niewłaściwej sytuacji będą pojawiać się komunikaty programu **Access**. Jednak w celu uproszczenia zagadnienia ograniczamy się do takiego rozwiązania.

- W celu przypisania pracownika do auta umieść w formularzu formant **Pole kombi**. Źródłem jego rekordów mają być rekordy dynamicznej tabeli zbudowanej na ulokowanym na serwerze widoku **dbo.Pracownik\_bez\_autu**. Oznacza to, że utworzona lista nie będzie stała. Jak tylko któryś z pracowników zostanie przypisany do auta, dane tego pracownika znikną z listy.
  - W pierwszym oknie **Kreatora pól kombi** zaznacz opcję **Obiekt pole kombi ma pobierać wartości z innej tabeli lub kwerendy**; wybierz **Dalej**.
  - W kolejnym oknie **Kreatora** w odpowiedzi na pytanie „Która tabela lub kwerenda ma dostarczać wartości do pola kombi?” kliknij **Tabele** i wybierz **Tabela: dbo\_Pracownik\_bez\_autu**, potem **Dalej**.
  - W kolejnym wyświetlonym oknie wybierz pola *IdPracownik* oraz *Osoba*. Będą umieszczone w polu kombi. Wybierz **Dalej**.
  - W oknie dotyczącym sortowania wybierz pole *Osoba*; **Dalej**.
  - W kolejnym oknie zobaczymy treść pola *Osoba*. Można za pomocą wskaźnika myszki zmienić tu szerokość kolumny pola kombi. Wybierz **Dalej**.
  - Kolejny krok jest bardzo ważny, ponieważ pozwala na ustawienie wartości *IDPracownika* poprzez wybór danych osobowych tego pracownika z listy, realizowanej za pomocą pola kombi. Zatem kliknij opcję **Przechowaj tę wartość w tym polu** i wybierz z listy pole *IDPracownik* (rys. 21.23).
  - Dodaj etykietę „Wybierz pracownika” do pola kombi i wybierz **Zakończ**.
- Umieść także w formularzu **fAuto** przycisk **Powrót** (rys.21.22), który zamknie bieżący formularz i wyświetli odświeżony formularz **Fdbo\_auto**. Dodaj do zdarzenia **Przy kliknięciu** tego przycisku makro przedstawione na rysunku 21.24.

### Kreator pól kombi



Rys. 21.23. Okno **Kreatora pól kombi**, w którym uzupełniane jest pole *IDPracownika*



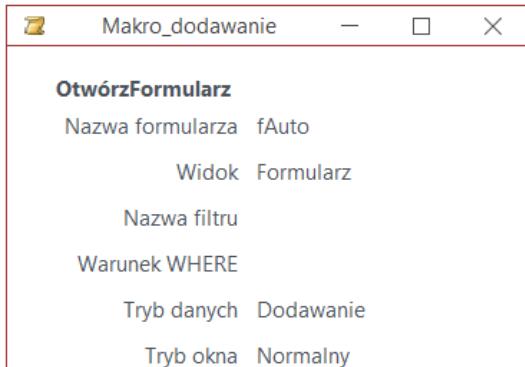
Rys. 21.24. Makro do zdarzenia **Przy kliknięciu** przycisku **Powrót**

#### IV. Nawiązanie połączenia między formularzami **Fdbo\_auto** i **fAuto**

- u. W formularzu **Fdbo\_auto** dla każdego z przycisków **Dodaj**, **Edytuj** oraz **Usuń** utwórz makro, które wykona odpowiednią czynność na rekordach tabeli **dbo\_Auto\_p**.

Makro umieszczamy w zakładce **Arkusza właściwości** przycisku polecenia w wierszu dotyczącym zdarzenia **Przy kliknięciu**.

Na rysunku 21.25 pokazano makro, które wyświetla formularz **fAuto** w trybie dodawania rekordów.



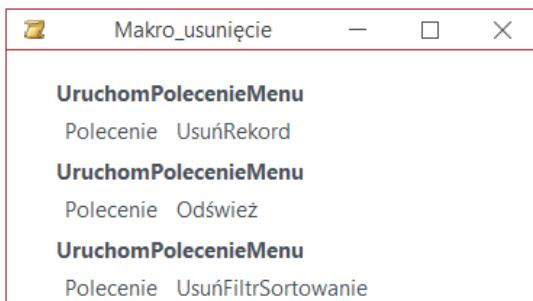
Rys. 21.25. Makro przeznaczone do dodawania nowego rekordu

Na rysunku 21.26 zaprezentowano makro przeznaczone do edycji wybranego rekordu.



Rys. 21.26. Makro przeznaczone do edycji wybranego rekordu tabeli **dbo\_Auto\_p**

Rysunek 21.27 przedstawia makro do usunięcia wybranego rekordu, odświeżania formularza i wyświetlenia wszystkich rekordów, tzn. skasowania filtrowania.



Rys. 21.27. Makro do usunięcia rekordu, odświeżenia formularza oraz wyświetlenia wszystkich rekordów

#### V. Formularz nawigacji

- v. Utwórz formularz nawigacji za pomocą odpowiedniego polecenia na wstążce i umieść na pierwszej zakładce formularza **Fdbo\_auto**.
- w. Wykonaj dowolny formularz tabelaryczny do pracy z danymi tabeli **dbo\_Pracownik**. Nazwij formularz **Fdbo\_Pracownik**. Umieść ten formularz na drugiej zakładce formularza nawigacji.

#### VI. Podsumowanie

Wykonując **zadanie 21\_5**, utworzysz aplikację, która pozwoli na przeglądanie oraz zmianę treści bazy danych. Oczywiście, że funkcjonalność tej aplikacji można rozbudować, dodając na przykład różne metody wyszukiwania danych, wykonania raportów oraz konwertowania danych.

Ten przykład miał zademonstrować możliwość utworzenia aplikacji klienta, która składa się z konstrukcji utworzonych za pomocą kreatorów programu, makr oraz procedur języka **VBA** zastosowanych do opracowania zdarzeń.

## 22. Wybrane konstrukcje programistyczne języka Transact SQL

**Transact SQL** jest proceduralnym językiem. W poprzednich rozdziałach rozważane były zadania, które można wykonać za pomocą pojedynczej instrukcji. W rozdziale tym zaprezentowano przykłady zastosowania wybranych konstrukcji języka **Transact SQL** używanych w bardziej skomplikowanych zadaniach. Dalej będą rozważane następujące pojęcia: wsad

(**batch**), zmienna (**variable**), instrukcje przepływu sterowania (**control-of-flow statements**) oraz procedura składowana (**stored procedure**).

## 22.1. Wsad, zmienna, przypisanie wartości

Zasobem komunikacji między programem klienckim a systemem **SQL Server** jest wsad – jedno lub kilka instrukcji **SQL** wysłanych w tym samym czasie przez pewną aplikację. **SQL Server** kompliluje instrukcję wsadu, tworząc jedyny moduł do wykonania, więc każda omawiana w poprzednich rozdziałach pojedyncza przeznaczona do wykonania instrukcja jest wsadem.

Instrukcje **SQL** we wsadzie można kończyć średnikiem, jednak nie jest to obowiązkowe. Na końcu wsadu umieszczamy dyrektywę **GO**, która sygnalizuje o ukończeniu wsadu.

W **zadaniu 22\_1**, które ilustruje wsad, zaproponowano zapisać wsad do utworzenia przykładowej tymczasowej tabeli. Tabele tymczasowe mają nazwy zaczynające się od symbolu „#”. System lokuje je w systemowej bazie danych **tempdb**. Zobaczmy utworzoną tabelę, jeśli rozwiniemy węzeł **Temporary Tables**. Po restarcie serwera utworzona tabela zniknie.

### Zadanie 22\_1

---

Zapisz wsad do utworzenia tymczasowej tabeli o nazwie **Tabela1**. Tabela powinna zawierać jedno pole typu **Integer**. Do tabeli należy wprowadzić 3 przykładowe rekordy. Po wykonaniu wsadu należy wyświetlić wszystkie wartości tabeli.

---

#### Wykonanie

Na rysunku 22.1 przedstawiono przykładowy wsad do utworzenia tymczasowej tabeli, a następnie – instrukcję do wyświetlenia wszystkich rekordów tabeli.



```
CREATE TABLE #tabela1 (a int) ;
INSERT INTO #tabela1 VALUES (1) ;
INSERT INTO #tabela1 VALUES (2) ;
INSERT INTO #tabela1 VALUES (3) ;
GO
SELECT * FROM #tabela1
GO
```

Rys. 22.1. Przykładowy wsad do wykonania **Zadania 22\_1**

We wsadzie można zadeklarować, a potem wykorzystywać zmienne lokalne. Nazwa zmiennej utworzonej przez użytkownika zawsze zaczyna się znakiem „@”. Na przykład: **@zmienna1**, **@nazwa**. Aby zadeklarować zmienną, należy umieścić słowo **DECLARE** na początku wsadu, a po nim nazwę oraz typ zmiennej (można również umieścić wartość domyślną). Zasięg i widoczność takich zmiennych są ograniczone jedynie do wsadu. Zmiennej można przypisać wartość. W tym celu wykorzystujemy instrukcje **SELECT** oraz **SET** wraz z **operatorem przypisania (=)**.

Deklarowanie oraz wykorzystanie zmiennych wykonuje się nie tylko we wsadzie, a także w procedurach składowanych i innych strukturach języka **Transact SQL**. Najłatwiej jest zapoznać się z nimi, używając ich we wsadach. W tym celu zaproponowano do wykonania kolejne 4 zadania.

## Zadanie 22\_2

Zapisz wsad, w którym zadeklaruj dwie zmienne lokalne i zastosuj je do przedstawienia minimalnej i maksymalnej wartości pola *Rocznik* tabeli **Auto\_p** należącej do bazy danych **lisiecka\_sql**.

### Wykonanie

Przykładowe rozwiązanie tego zadania przedstawia rysunek 22.2.

```

USE lisiecka_sql
GO
DECLARE @max_Rocznik int, @min_Rocznik int
SELECT @max_Rocznik = max(Rocznik),
       @min_Rocznik = min(Rocznik)
FROM Auto_p WHERE Rocznik IS NOT NULL
SELECT 'Maksymalna wartość pola Rocznik' = @max_Rocznik,
       'Minimalna wartość pola Rocznik' = @min_Rocznik
  
```

The screenshot shows the SSMS interface with a query window containing the provided T-SQL script. Below the window, the 'Results' tab is selected, displaying the following output:

	Maksymalna wartość pola Rocznik	Minimalna wartość pola Rocznik
1	2017	2009

Rys. 22.2. Przykładowy wsad do wykonania **Zadania 22\_2**

## Zadanie 22\_3

Zapisz wsad do aktualizacji rekordu tabeli **Pracownik** bazy danych **lisiecka\_sql**. Jeśli wartość pola *IDPracownik* jest rów-

na 2, to wprowadź do pola *Nazwisko* tekst „*Trochimczuk*”. We wsadzie wykorzystaj dwie zmienne lokalne.

## **Wykonanie**

Na rysunku 22.3 przedstawiono przykładowy wsad. W nim zadeklarowano dwie zmienne lokalne w celu przechowywania wartości potrzebnych do aktualizacji. Do przypisania wartości zmiennym zastosowano instrukcję **SET**.

```
USE lisiecka_sql
GO
DECLARE @nazwisko nvarchar(50), @IDPracownik int
SET @nazwisko='Trochimczuk'
SET @IDPracownik=2

UPDATE Pracownik SET Nazwisko=@nazwisko
WHERE IDPracownik=@IDPracownik
```

Rys. 22.3. Przykładowy wsad do wykonania **Zadania 22\_3**

## **Zadanie 22\_4**

Utwórz wsad do wyświetlenia aktualnej daty i czasu. Należy utworzyć dwie zmienne lokalne do ich przechowywania i wyświetlić oddzielnie datę i godzinę.

## **Wykonanie**

Do wyznaczenia aktualnej daty i czasu można zastosować funkcje wbudowaną **getdate()** języka Transact SQL. Przykładowe rozwiązanie tego zadania przedstawia rysunek 22.4. Do wyświetlenia wyników zastosowano dwie różne postaci polecenia **SELECT**.

```
DECLARE @aktualna_data date, @aktualny_czas time
SET @aktualna_data = getdate()
SET @aktualny_czas = getdate()
SELECT @aktualna_data AS 'Aktualna data'
SELECT 'Aktualny czas' = @aktualny_czas
```

Rys. 22.4. Przykładowy wsad do wykonania **Zadania 22\_4**

Oprócz zmiennych użytkownika w skrypcie **Transact SQL** można używać również zmiennych systemowych. Ich nazwy oznaczone są dwoma znakami „@”. Przykładowo często używaną zmienną jest **@@Identity**, która

zawiera ostatnio wygenerowaną wartość **IDENTITY**. W kolejnym zadaniu zaprezentowano zastosowanie takiej zmiennej.

## Zadanie 22\_5

Zapisz wsad do wykonania w bazie danych **lisiecka\_sql**. Wsad będzie przeznaczony do dodania jednego nowego rekordu danych do tabeli **Pracownik** oraz wyświetlenia wygenerowanej wartości pola *IDPracownik*. Wykorzystaj zmienną lokalną.

### Wykonanie

Przykładowy wsad do realizacji zadania przedstawia rysunek 22.5.

The screenshot shows a SQL Server Management Studio interface. In the top pane, there is a query window containing the following T-SQL script:

```
USE lisiecka_sql
GO
DECLARE @Wart_IDPracownik nvarchar(50)
INSERT INTO Pracownik ([PESEL],[Imię],[Nazwisko],[Stanowisko],[Pensja])
VALUES ('80122311115', 'Marek', 'Mostowiak', 'dyrektor', 7000.00)
SET @Wart_IDPracownik = @@IDENTITY
SELECT @Wart_IDPracownik
```

In the bottom pane, there is a results grid with one row of data:

	(No column na...)
1	13

Rys. 22.5. Przykładowy wsad do wykonania **Zadania 22\_5**

## 22.2. Przykłady narzędzi do sterowania przepływem wykonania

**Transact SQL** zawiera podstawowy zestaw narzędzi sterujących przepływem wykonania. Obejmują one instrukcje z warunkami logicznymi (**IF ELSE** oraz **CASE**), pętle (**WHILE** wraz z opcjami **CONTINUE** i **BREAK**), skoki bezwarunkowe (**GOTO**), zwrot wartości stanu do podprogramu wywołującego (**RETURN**), narzędzie do obsługi błędów (**TRY...CATCH**). Zapoznamy się z niektórymi z nich w tym podrozdziale.

### Instrukcja IF ELSE

Konstrukcja **IF ELSE** ma postać:

```
IF <warunek>
    instrukcja1 (lub blok1)
ELSE
    Instrukcja2 (lub blok2)
```

Warunek jest wyrażeniem logicznym. Jeśli *warunek* jest spełniony, to wykonywana jest *instrukcja 1*. Jeżeli *warunek* nie jest spełniony, to wykonywana jest *instrukcja 2*.

Rozważmy konstrukcję **IF ELSE** na przykładzie, wykonując kolejne zadanie.

## Zadanie 22\_6

---

Zapisz wsad do wykonania w bazie danych **lisiecka\_sql**. Należy wyświetlić liczbę samochodów wyprodukowanych w 2010 roku. Jeśli w tabeli **Auto\_p** brakuje rekordów dotyczących takich samochodów, to należy wyświetlić tekst o tym, że nikt nie jeździ samochodem z 2010 roku.

---

### Wykonanie

Na rysunku 22.6 zaproponowano wsad, w którym zastosowano lokalną zmienną **@Liczba\_samochodów** do przechowywania wyniku funkcji agregującej **COUNT**.

```
USE lisiecka_sql
GO
DECLARE @Liczba_samochodów int
SELECT @Liczba_samochodów = COUNT(IDAuto)
FROM Auto_p
WHERE
    Rocznik = 2010
IF @Liczba_samochodów > 0
    SELECT 'Liczba samochodów wyprodukowanych w 2010 roku = ', @Liczba_samochodów
ELSE
    SELECT 'Nikt nie jeździ samochodem z 2010 roku'
```

Rys. 22.6. Przykład zastosowania instrukcji **IF ELSE** (Zadania 22\_6)

### Instrukcja CASE

Instrukcja **CASE** jest uogólnieniem konstrukcji **IF**. W zależności od podanych kryteriów zwraca jedną z możliwych wartości.

Instrukcja **CASE** zaczyna się słowem kluczowym **CASE**, zawiera jedną lub więcej konstrukcji **WHEN THEN**, potem w niej może być ulokowana klauzula **ELSE**, a na końcu zawsze występuje **END**.

Od razu po słowie **CASE** może być ulokowane wyrażenie do sprawdzania, jednak inna odmiana tej instrukcji pozwala na brak jakiegokolwiek wyrażenia przed klauzulą **WHEN THEN**.

Poniżej zaproponowano trzy kolejne zadania, których rozwiązanie pozwala na zrozumienie zasad utworzenia instrukcji **CASE**.

## Zadanie 22\_7

Zapisz wsad do wyświetlenia informacji na temat, jaka jest aktualnie w Polsce meteorologiczna pora roku.

### Wykonanie

Zadanie sprowadza się do wyznaczenia aktualnego numeru miesiąca. Odpowiednie rozwiązanie przedstawia rysunek 22.7. Zadeklarowano w nim zmienną lokalną do przechowywania numeru miesiąca dla aktualnej daty. Aktualną datę zwraca funkcja wbudowana **getdate()**, a numer miesiąca – funkcja **month**.

```
DECLARE @mies int
SET @mies = month(getdate())
SELECT
CASE
WHEN @mies Between 3 AND 5 THEN 'Meteorologiczna wiosna'
WHEN @mies Between 6 AND 8 THEN 'Meteorologiczne lato'
WHEN (@mies Between 9 AND 11) THEN 'Meteorologiczna jesień'
ELSE 'Meteorologiczna zima'
END
```

Rys. 22.7. Przykład instrukcji CASE (Zadanie 22\_7)

Instrukcja **CASE** może być wykorzystana na różne sposoby. Kolejny przykład prezentuje wsad, w którym odbywa się aktualizacja tabeli przy użyciu tej instrukcji.

## Zadanie 22\_8

Zapisz wsad do aktualizacji tabeli **Pracownik** w bazie danych **lisiecka\_sql**: powiększ wartość w polu *Pensja* o 5% w tych rekordach, których pole *Imię* zawiera tekst „Andrzej”, „Marek” lub „Maria”; powiększ wartość w polu *Pensja* o 7% w tych rekordach, których pole *Imię* zawiera tekst „Anna”, „Kinga” lub

„Zbigniew”; w pozostałych rekordach powiększ wartość w polu Pensja o 10%.

---

## Wykonanie

Przykładowy wsad z zastosowaniem instrukcji **CASE** przedstawiono na rysunku 22.8. Oczywiście, że zadanie również można było by wykonać bez instrukcji **CASE**, wykorzystując zamiast tego trzy razy instrukcję **UPDATE**.

```
USE lisiecka_sql
GO
UPDATE dbo.Pracownik
SET Pensja =
(CASE
WHEN Imię IN ('Andrzej', 'Marek', 'Maria') THEN Pensja*1.05
WHEN Imię IN ('Anna', 'Kinga', 'Zbigniew') THEN Pensja* 1.07
ELSE Pensja * 1.1
END
)
```

Rys. 22.8. Przykład zastosowania **CASE** w instrukcji **UPDATE** (Zadanie 22\_8)

W instrukcji **CASE** bezpośrednie po słowie **CASE** może występować wyrażenie. Taki przypadek właśnie ilustruje rozwiązanie kolejnego zadania.

## Zadanie 22\_9

---

Zapisz wsad do wyświetlenia danych z tabeli **Auto\_p** bazy danych **lisiecka\_sql**. Należy wyświetlić kolumny **Numer\_rejestracyjny** oraz **Rocznik**, a także kolumnę, w której trzeba scharakteryzować samochód, zapisując tekst: „Najnowszy”, „Najstarszy” lub „Przeciętny” w zależności od tego, co zawiera pole **Rocznik**. Posortuj dane według pola **Numer\_rejestracyjny**.

---

## Wykonanie

Na rysunku 22.9 zaprezentowano wsad, na który składają się dwie instrukcje – **SELECT** i **ORDER BY**. Klauzula **SELECT** zawiera trzy elementy: pola **Numer\_rejestracyjny** oraz **Rocznik**, a także instrukcję **CASE**, która służy do wyświetlenia odpowiedniego tekstu (po słowie **WHEN** umieszczone podzapytanie, którego wynik porównywany jest z zawartością pola **Rocznik**).

```
USE lisiecka_sql
GO
|SELECT Numer_rejestracyjny, Rocznik,
CASE [Rocznik]
WHEN (SELECT MAX([Rocznik]) FROM dbo.Auto_p) THEN 'Najnowszy'
WHEN (SELECT MIN([Rocznik]) FROM dbo.Auto_p) THEN 'Najstarszy'
ELSE 'Przeciętny'
END
FROM Auto_p
ORDER BY Numer_rejestracyjny
```

Rys. 22.9. Przykładowy wsad do wykonania **Zadania 22\_9**

### Instrukcja WHILE

Jeśli zachodzi konieczność wielokrotnego wykonania jakieś operacji, to stosujemy pętlę. W języku **Transact SQL** przeznaczona jest do tego instrukcja **WHILE**. Jej schemat ma postać:

```
WHILE <kryterium>
BEGIN
<instrukcja1>
<instrukcja2>
...
<instrukcjaN>
END
```

Wykorzystane w instrukcji **WHILE** słowa kluczowe **BEGIN END** pozwalają na umieszczenie wielu instrukcji, tzn. tworzą **blok**. Jeżeli pętla zawiera tylko jedną instrukcję, to z **BEGIN END** można zrezygnować.

Prosty przykład instrukcji **WHILE** przedstawia rozwiązanie kolejnego zadania.

## Zadanie 22\_10

Utwórz 3 wsady: pierwszy – do utworzenia tymczasowej tabeli o jednej kolumnie typu **Integer**; drugi – do wprowadzenia do tabeli wartości od 1 do 10; trzeci – do wyświetlenia rekordów tabeli.

### Wykonanie

Na rysunku 22.10 przedstawiono rozwiązanie zadania. Tu pierwsza instrukcja **SQL** tworzy tabelę o nazwie **#tabela**, która zawiera jedną kolumnę.

```
CREATE TABLE #tabela (a int) ;
GO
DECLARE @I as int = 0

WHILE @I <= 10
BEGIN
    SET @I= @I + 1
    INSERT INTO #tabela VALUES (@I) ;
END
GO
SELECT * FROM #tabela
GO
```

Rys. 22.10. Przykład zastosowania instrukcji WHILE (**Zadanie 22\_10**).

Drugi wsad zaczyna się od deklaracji zmiennej roboczej **@I** typu **Integer**, której przypisano początkowe znaczenie – 0. Zmienna ta służy jako licznik kroków pętli. Po słowie **WHILE** odbywa się sprawdzenie, czy wartość **@I** nie przekroczyła 10. Jeśli nie, to jej wartość powiększa się o 1 i do tabeli zostaje dodany nowy rekord. Po dziesięciu krokach wykonanie pętli się kończy.

Trzeci wsad zawiera polecenie **SELECT** do wyświetlenia treści utworzonej tabeli.

### 22.3. Procedura składowana

Procedura składowana jest zbiorem instrukcji **Transact SQL**, który ma nazwę, jest kompilowany jeden raz, a następnie jest wykorzystywany wiele razy. Stosowanie procedur składowanych zwiększa wydajność aplikacji pracującej z bazą danych.

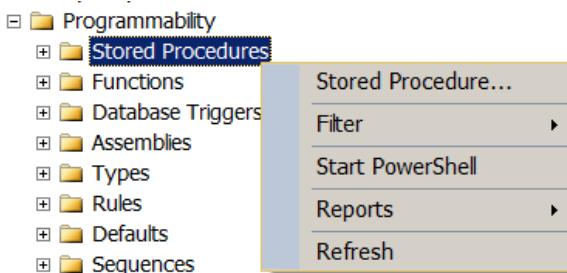
Procedura w języku **Transact SQL** ma te wszystkie cechy co procedury w innych językach programowania: może mieć parametry wejściowe i wyjściowe, zawierać większość z instrukcji języka **Transact SQL**, między innymi wywołanie innych procedur, a także przekazuje wartość stanu (wynikowe znaczenie o pomyślnym lub awaryjnym zakończeniu wykonania).

W systemowej bazie danych **master** możemy obejrzeć systemowe procedury składowane, które służą do pobierania i konfigurowania opcji serwera i baz danych. Ich nazwy zaczynają się od prefiku **sp\_**.

Użytkownik może tworzyć własne procedury. Z reguły w procedurze składowanej umieszcza się instrukcję modyfikujące dane. Przy tym może ona również służyć do wyświetlania danych.

Do utworzenia procedury można zastosować polecenie menu podręcznego węzła **Stored Procedures** bazy danych (rys. 22.11). Wtedy zostanie

otwarte okno z szablonem procedury, który należy uzupełnić, wpisując potrzebne do zrealizowania zadania instrukcje. Na końcu należy procedurę wykonać. Jeśli procedura wykona się pomyślnie, to zostanie ona zapisana w bazie danych.



Rys. 22.11. Menu podręczne węzła **Stored Procedures**

W proponowanym przez system szablonie procedury składowanej jako pierwsza występuje instrukcja **SET NOCOUNT ON**, która służy do rezygnacji wysyłania przez **SQL SERVER** informacji o liczbie opracowanych rekordów. Powoduje to zmniejszenie obciążenia sieci. Domyślnie działa opcja **SET NOCOUNT OFF**, w wyniku czego w przypadku operacji **SELECT**, **INSERT**, **UPDATE**, **DELETE** dodatkowo wysyłane są widomości o liczbie rekordów uczestniczących w operacji.

Procedurę można również utworzyć tak samo jak wsad, w oknie edytora, wybierając w menu programu ikonkę **New Query**. Po zapisaniu tekstu procedury należy ją wykonać, wybierając ikonkę  w menu programu.

Do utworzenia procedury przeznaczona jest instrukcja **CREATE PROCEDURE**, która może mieć zadeklarowane parametry lub wcale nie mieć parametrów.

Gdy zachodzi potrzeba w modyfikacji już istniejącej procedury, należy na początku zamienić słowo kluczowe **CREATE** na **ALTER**, a potem działać jak przy tworzeniu nowej procedury.

Do wykonania procedury służy instrukcja **EXECUTE** (lub **EXEC**).

Tak samo jak we wsadzie, w procedurę mogą być użyte zmienne lokalne, które należy zadeklarować za pomocą słowa kluczowego **DECLARE**.

Najprostszym rodzajem procedury jest **procedura bez parametrów**. Struktura procedury bez parametrów jest następująca:

```
CREATE PROCEDURE nazwa_procedury AS  
BEGIN  
    Instrukcje Transact SQL  
END  
GO
```

Faktycznie po słowie **AS** mamy zapisać wsad. Jeśli w procedurze występują zmienne lokalne, umieszczamy deklarację zmiennych na początku tego wsadu.

Przykładem procedury bez parametrów będzie procedura tworzona przy wykonaniu kolejnego zadania.

## Zadanie 22\_11

---

Utwórz w bazie danych **lisiecka\_sql** procedurę, która wyświetla wszystkie dane utworzonego wcześniej widoku **W\_Pracownik\_Auto**.

### Wykonanie

- W oknie **Object Explorer** zaznacz nazwę bazy danych **lisiecka\_sql**.
- Otwórz okienko **New Query** i zapisz tekst procedury przedstawiony na rysunku 22.12. Procedura służy do wykonania instrukcji **SELECT**. Na początku dodano opcję **NOCOUNT ON**. W wierszu, który zaczyna się od „---”, umieszczono komentarz.

```
CREATE PROCEDURE P_Pracownicy_Auta  
-- przykład procedury bez parametrów  
AS  
BEGIN  
    SET NOCOUNT ON  
    SELECT * FROM W_Pracownik_Auto  
END
```

Rys. 22.12. Przykład procedury bez parametrów (**Zadanie 22\_11**)

- Aby wykonać polecenie i stworzyć procedurę, naciśnij na przycisk **Execute** (lub klawisz funkcyjny **F5**). Na tym etapie zawsze odbywa się sprawdzanie procedury i w razie popełnionych błędów pojawiają się odpowiednie wiadomości.
- Po pomyślnym wykonaniu procedury odśwież węzeł **Programmability** – na liście procedur składowanych (**Stored Procedures**) pojawi się nazwa nowo utworzonej procedury.
- Jak zawsze polecenie można zapisać do pliku o rozszerzeniu **\*.sql**.

## Zadanie 22\_12

Zapisz polecenie Transact SQL do wykonania procedury **P\_Pracownicy\_Auta**.

### Wykonanie

Aby skorzystać z procedury składowanej, należy zastosować instrukcję **EXECUTE** (lub **EXEC**).

- a. Otwórz okno edytora za pomocą ikonki **New Query**.
- b. Wykonaj instrukcję przedstawioną na rysunku 22.13.

```
USE lisiecka_sql  
GO  
EXEC [dbo].[P_Pracownicy_Auta]
```

Rys. 22.13. Instrukcja do wykonania procedury **P\_Pracownicy\_Auta**

Procedura najczęściej zawiera **parametry wejściowe**. Wszystkie parametry należy zadeklarować po nazwie procedury przed słowem **AS**. Deklaracja parametru wejściowego składa się z jego nazwy oraz typu. Może mieć także wartość domyślną. Nazwa parametru zawsze zaczyna się znakiem „@”. Gdy parametrów jest więcej niż jeden, to między ich opisami stawiamy przecinek.

Następujące zadanie pomoże w zrozumieniu, co to jest procedura z parametrami wejściowymi.

## Zadanie 22\_13

W bazie danych **lisiecka\_sql** utwórz i wykonaj procedurę do wyświetlenia danych pracownika według jego numeru **PESEL** (z tabeli **Pracownik**).

### Wykonanie

Zgodnie z wymaganiem zadania należy utworzyć procedurę o jednym parametrze wejściowym, który będzie miał typ danych dokładnie taki sam, jak pole **PESEL**. Na rysunku 22.14 przedstawiono skrypt do tworzenia takiej procedury. Parametr ma nazwę **@param\_PESEL**.

```

CREATE PROCEDURE Pracownik_PESEL
-- Przykład parametru wejściowego
@param_PESEL varchar(11)
AS
BEGIN
SET NOCOUNT ON
SELECT * FROM [dbo].Pracownik WHERE PESEL LIKE @param_PESEL
END

```

Rys. 22.14. Procedura **P\_Pracownik\_Pesel**

Aby wykonać procedurę, należy postępować jak w przypadku procedury bez parametrów i dodatkowo przekazać wartości parametrów. Można dokonać tego za pomocą instrukcji:

```
EXEC nazwa_procedure, nazwa_parametru1 = wartość_parametru1,
nazwa_parametru2 = wartość_parametru2,...
```

## Zadanie 22\_14

---

Zapisz i wykonaj wsad do wykonania procedury **P\_Pracownik\_PESEL** dla konkretnej wartości pola **PESEL**, przykładowo dla **PESEL** równego '89031211111'.

---

### Wykonanie

Na rysunku 22.15 przedstawiono wsad do wykonania **Zadania 22\_14**.

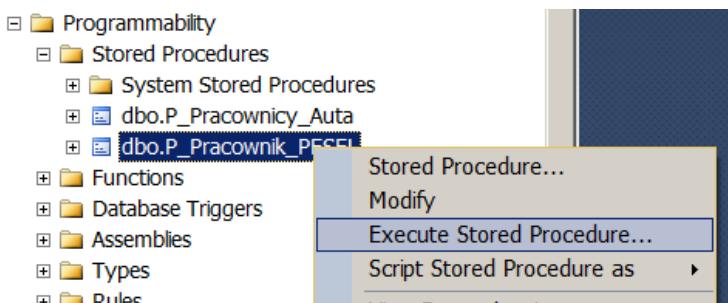
```

USE lisiecka_sql
GO
EXEC [dbo].[P_Pracownik_PESEL]
    @Param_PESEL = '89031211111'

```

Rys. 22.15. Skrypt do wykonania procedury **P\_Pracownik\_PESEL**

Wykonać procedurę składawaną możemy także, korzystając z polecenia **Execute Stored Procedure** menu podręcznego procedury (rys. 22.16). Po wyborze tego polecenia zostanie wyświetlone okno dialogowe, do którego należy wprowadzić parametry procedury.



Rys. 22.16. Menu podręczne procedury **P\_Pracownik\_PESEL**

Kolejne zadanie przedstawia przykład procedury do aktualizacji danych tabeli.

## Zadanie 22\_15

W bazie danych **lisiecka\_sql** utwórz procedurę o nazwie **Zmiana\_nazwiska** przeznaczoną do zmiany wartości pola *Nazwisko* tabeli **Pracownik**, gdy zadana jest wartość klucza podstawowego, a także nowa wartość pola *Nazwisko*.

Wykonaj procedurę, podając wartość pola *IDPracownik* równą 2, a nową wartość pola *Nazwisko* – „Kozłowski”.

### Wykonanie

- W oknie **Object Explorer** zaznacz nazwę bazy danych **lisiecka\_sql**.
- Otwórz okienko **New Query** i zapisz skrypt przedstawiony na rysunku 22.17. Tu dwóm parametrom wejściowym nadano nazwy **@nazwisko** oraz **@idpracownik**. Oczywiście, że nazwę dla parametru zadajemy w dowolny sposób, konieczne jest tylko umieszczenie na początku znaku „@”.

```
CREATE PROCEDURE Zmiana_nazwiska
    -- Parametry wejściowe:
    @nazwisko nvarchar(50), @idpracownik int
    AS
    BEGIN
        SET NOCOUNT ON
        UPDATE Pracownik SET Nazwisko = @nazwisko
        WHERE IDPracownik = @idpracownik
    END
```

Rys. 22.17. Skrypt do utworzenia procedury **Zmiana\_nazwiska** (Zadanie 22\_15)

- c. Zapisz wsad do wykonania procedury w postaci, przedstawionej na rysunku 22.18.

```
USE lisiecka_sql  
GO  
EXEC [Zmiana_nazwiska] @nazwisko = 'Kozłowski', @IDPracownik = 2
```

Rys. 22.18. Skrypt do wykonania procedury **Zmiana\_nazwiska** (Zadanie 22\_15)

Procedura składowana może także przekazywać aplikacji, która ją wywołała, pewne wynikowe wartości. W tym celu w procedurę dodajemy **parametry wyjściowe**. Są deklarowane tak samo jak wejściowe, lecz na końcu deklaracji takiego parametru należy umieścić słowo kluczowe **output**. Odpowiednio przy wykonaniu takiej procedury za pomocą instrukcji **EXEC** też dodaje się taką opcję. Kolejne zadanie prezentuje utworzenie procedury, która zawiera parametry wejściowy i wyjściowy.

## Zadanie 22\_16

---

Umieść w bazie danych **lisiecka\_sql** procedurę składowaną do zwracenia liczby samochodów wyprodukowanych w zadanym roku.

Zapisz wsad do wykonania procedury dla wartości pola *Rocznik* równej 2017.

Wykonaj również procedurę za pomocą narzędzi graficznych programu.

---

### Wykonanie

- W oknie **Object Explorer** zaznacz nazwę bazy danych **lisiecka\_sql**.
- Otwórz okienko **New Query** i zapisz skrypt przedstawiony na rysunku 22.19. Instrukcja **SELECT** w tej procedurze (rys. 22.19) służy do wyliczenia liczby samochodów wyprodukowanych w 2017 roku. Przy tym rekordy, w których wartość pola *Rocznik* jest pusta, nie są rozpatrywane.
- W celu wykonania utworzonej procedury zapisz i wykonaj wsad przedstawiony na rysunku 22.20.
- W celu wykonania procedury za pomocą narzędzi graficznych programu wybierz w menu podręcznym procedury polecenie **Execute Stored Procedure** (rys. 22.16). Wprowadź w oknie **Execute Procedure** (rys. 22.21) wartość parametru wejściowego i wybierz **OK** – okno **New Query** przybierze postać jak na rysunku 22.22.

```

CREATE PROCEDURE [dbo].[Liczba_aut_z_zadanego_roku]
-- Parametr wejściowy:
@rok int,
-- Parametr wyjściowy:
@Liczba_aut int output
AS
BEGIN
SET NOCOUNT ON
SELECT @Liczba_aut = COUNT (IDAuto)
FROM dbo.Auto_p
WHERE (Rocznik is not null) and (Rocznik = @rok)
END

```

Rys. 22.19. Przykład procedury o parametrach wejściowym i wyjściowym

```

USE [lisiecka_sql]
GO

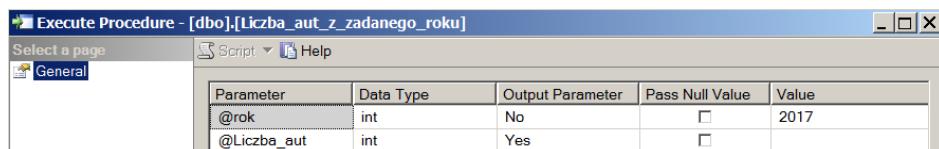
DECLARE @Wynikowa_liczba int

EXEC    [dbo].[Liczba_aut_z_zadanego_roku]
        @rok = 2017,
        @Liczba_aut = @Wynikowa_liczba OUTPUT

SELECT  @Wynikowa_liczba as [Liczba aut z 2017 roku]

```

Rys. 22.20. Wsad do wykonania procedury **Liczba\_aut\_z\_zadanego\_roku**



Rys. 22.21. Okno wykonania procedury **Liczba\_aut\_z\_zadanego\_roku**

Na rysunku 22.22 widzimy zmienną o nazwie **@return\_value**, która stworzył program w celu zapisania w niej kodu zwrotnego procedury. W tym przypadku wartość zmiennej jest równa 0, ponieważ w procedurze nie podano żadnej wartości w instrukcji **RETURN**. Jednak w bardziej zaawansowanych procedurach możemy korzystać z możliwości tej instrukcji. W kolejnym przykładzie przedstawiono właśnie zastosowanie instrukcji **RETURN** w celu zdefiniowania kodu zwrotnego procedury.

```

USE [lisiecka_sql]
GO
DECLARE @return_value int,
        @Liczba_aut int
EXEC    @return_value = [dbo].[Liczba_aut_z_zadanego_roku]
        @rok = 2017,
        @Liczba_aut = @Liczba_aut OUTPUT
SELECT  @Liczba_aut as N'@Liczba_aut'
SELECT  'Return Value' = @return_value
GO
  
```

The screenshot shows the SQL query window with the results pane open. The results are displayed in two tables:

	@Liczba_aut
1	1

	Return Value
1	0

Rys. 22.22. Widok wyników wykonania procedury **Liczba\_aut\_z\_zadanego\_roku** za pomocą narzędzi graficznych

## 22.4. Instrukcja RETURN

W dowolnym miejscu wsadu, procedury lub bloku instrukcji można umieścić instrukcję **RETURN** w celu natychmiastowego zakończenia wykonania tej konstrukcji. Wyrażenia po tej instrukcji nie będą już wykonywane.

W instrukcji można także zwrócić wartość liczbową: **RETURN <wartość>**. Gdy procedura zwraca taką wartość, nazywamy ją kodem powrotnym.

Wykonanie kolejnego zadania przybliży zrozumienie sposobu działania instrukcji **RETURN** razem z procedurą składowaną.

### Zadanie 22\_17

W bazie danych **lisiecka\_sql** utwórz procedurę składowaną do zwrócenia numeru produkcji samochodu po podaniu numeru pracownika, który jest do niego przypisany.

Wykonaj procedurę przy wartości pola *IDPracownik* równej 2.

### Wykonanie

- Zapisz procedurę w postaci przedstawionej na rysunku 22.23.

```

CREATE PROCEDURE [dbo].[Rok_produkcji_auta]
    @IDPracownik int, --parametr wejściowy
    @Rok_produkcji int OUTPUT -- parametr wyjściowy
AS
BEGIN
SET NOCOUNT ON
IF @IDPracownik IS NULL --'Brak numeru ID pracownika'
    RETURN (1)
ELSE
BEGIN
    IF -- 'Pracownik o takim numerze nie jest przypisany do żadnego samochodu'
        (SELECT COUNT(IDPracownik) FROM Auto_p WHERE IDPracownik = @IDPracownik) = 0
        RETURN (2)
    END
SET @Rok_produkcji = (SELECT Rocznik FROM Auto_p WHERE IDPracownik = @IDPracownik)
IF @Rok_produkcji IS NULL -- 'Brak informacji o roku produkcji auta'
    RETURN (3)
ELSE
    RETURN (0) -- Sukcess!
END

```

Rys. 22.23. Procedura składowana do **Zadania 22\_17**

- b. Zapisz wsad do testowania procedury tak jak przedstawia to rysunek 22. 24.
- c. Wykonaj procedurę wielokrotnie w celu sprawdzenia wszystkich możliwych błędnych sytuacji, które zostały opracowane za pomocą instrukcji **RETURN**.

```

USE [lisiecka_sql]
GO
DECLARE @kod_powrotny int,
        @wynik int

EXEC    @kod_powrotny = [dbo].[Rok_produkcji_auta]
        @IDPracownik = 2,
        @Rok_produkcji = @wynik OUTPUT
IF @kod_powrotny = 0
    SELECT  @wynik as [Rok_produkcji]
ELSE
    IF @kod_powrotny = 1
        PRINT 'Brak numeru ID pracownika'
    ELSE
        IF @kod_powrotny = 2
            PRINT 'Pracownik o takim numerze nie jest przypisany do żadnego samochodu'
        ELSE
            PRINT 'Brak informacji o roku produkcji auta'
GO

```

Rys. 22.24. Wsad do wykonania **Zadania 22\_17**

## LITERATURA

1. Ullmann J. D., Widom J.W.: *Podstawowy wykład z systemów baz danych*, Warszawa: WNT, 2000.
2. Elmasri R., Navathe S. B.: *Wprowadzenie do systemów baz danych*, Gliwice: Helion, 2005.
3. Date C. J.: *Relacyjne bazy danych dla praktyków*, Gliwice: Helion, 2006.
4. Allen S.: *Modelowanie danych*, Gliwice: Helion, 2006.
5. Garcia-Molina H., Ullman J. D., Widom J.: *Systemy baz danych: kompletny podręcznik*, Gliwice: Helion, 2011.
6. Beighley L.: *SQL*, Gliwice: Helion, 2011.
7. Hernandez M. J.: *Projektowanie baz danych dla każdego: przewodnik krok po kroku*, Gliwice: Helion, 2014..
8. Michael A., Kusleika R.: *Access 2016 Bible*, Wiley 2015.
9. Rockoff L.: *Język SQL: przyjazny podręcznik*, Gliwice: Helion, 2017.