John L. Viescas
Michael J. Hernandez

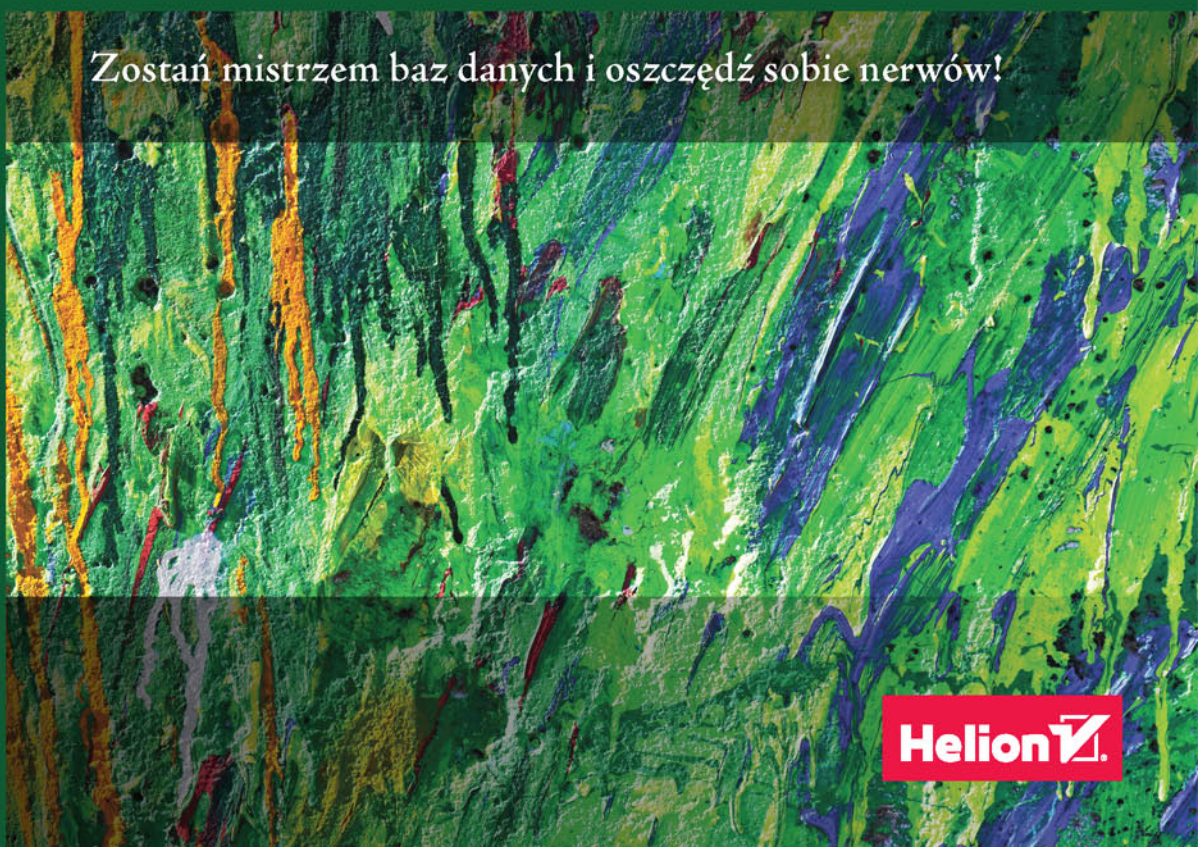
Słowo wstępne: Keith W. Hare
wiceprezes amerykańskiej komisji ds. norm SQL

ZAPYTANIA W SQL

PRZYJAZNY PRZEWODNIK

WYDANIE III

Zostań mistrzem baz danych i oszczędź sobie nerwów!



Helion

Tytuł oryginału: SQL Queries for Mere Mortals®: A Hands-On Guide to Data Manipulation in SQL, Third Edition

Tłumaczenie: Piotr Cieślak

ISBN: 978-83-283-1364-4

Authorized translation from the English language edition, entitled: SQL QUERIES FOR MERE MORTALS: A HANDS-ON GUIDE TO DATA MANIPULATION IN SQL, Third Edition; ISBN: 0321992474; by John L. Viescas; and by Michael J. Hernandez; published by Pearson Education, Inc, publishing as Addison Wesley.

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

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/sqldkp.zip

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

Printed in Poland.

Spis treści

4
Tworzenie prostego zapytania

Myśl jak mędrzec, ale komunikuj się językiem ludu.
— William Butler Yeats

Zagadnienia omówione w tym rozdziale

- SELECT — wprowadzenie
- Instrukcja SELECT
- Krótka dygresja: dane a informacje
- Przekładanie żądania na SQL
- Eliminowanie powtarzających się wierszy
- Sortowanie informacji
- Zapisywanie pracy
- Przykładowe instrukcje
- Podsumowanie
- Zagadnienia do samodzielnego rozwiązania

Teraz gdy poznałeś już szczyptę historii SQL, nadszedł najwyższy czas na naukę samego języka. Jak już wspomnieliśmy we „Wstępie", większość tej książki jest poświęcona tym aspektom SQL, które służą do przetwarzania danych. Zaczniemy od prawdziwego wołu roboczego SQL — instrukcji `SELECT`.

SELECT — wprowadzenie

Najważniejsze spośród wszystkich słów kluczowych, SELECT, stanowi samo serce SQL. Jest to fundament najpotężniejszych i najbardziej złożonych instrukcji w całym języku, a zarazem narzędzie do pozyskiwania informacji znajdujących się w tabelach bazy danych. SELECT używa się w połączeniu z różnymi słowami kluczowymi i klauzulami, pozwalającymi na wyszukiwanie i wyświetlanie informacji na niemal nieskończenie wiele sposobów. Za pomocą SELECT można odpowiedzieć na prawie każde pytanie dotyczące tego „kto?", „co?", „gdzie?", „kiedy?", a nawet „co by było, gdyby?" i „jak?". Jeśli tylko poprawnie zaprojektowałeś bazę i zgromadziłeś właściwe dane, to możesz w ten sposób pozyskać odpowiedzi ułatwiające podejmowanie ważnych decyzji w organizacji. W części V „Modyfikowanie zbiorów danych" przekonasz się, że wiele technik opanowanych podczas poznawania SELECT przyda Ci się do nauki innych instrukcji, takich jak UPDATE, INSERT i DELETE.

Operację SELECT w SQL można podzielić na trzy mniejsze działania, do których będziemy się odwoływać jako do instrukcji SELECT, wyrażenia SELECT i zapytania SELECT. (Rozdzielenie operacji SELECT w taki sposób zdecydowanie ułatwia jej zrozumienie i docenienie jej złożoności). W ramach każdej z tych operacji można używać osobnego zbioru słów kluczowych i klauzul, co zapewnia elastyczność niezbędną do utworzenia finalnej instrukcji SQL, zgodnej z zapytaniem, które chcesz przekazać do bazy danych. W następnych rozdziałach dowiesz się, że operacje te można na wiele sposobów łączyć, by uzyskiwać odpowiedzi nawet na bardzo skomplikowane pytania.

W tym rozdziale rozpoczniemy omawianie instrukcji SELECT i pokrótce przyjrzymy się zapytaniu SELECT. Instrukcję SELECT przeanalizujemy bardziej szczegółowo w dalszej części książki — w rozdziale 5. „Nie tylko zwykłe kolumny" i rozdziale 6. „Filtrowanie danych".

Uwaga. W innych książkach poświęconych relacyjnym bazom danych słowem *relacja* niekiedy określa się *tabelę*, *wiersz* nazywa się *krotką* albo *rekordem*, zaś kolumnę — *atrybutem* albo *polem*. Ale w standardzie SQL wymienione elementy struktury bazy danych zostały nazwane odpowiednio: *tabelą*, *wierszem* oraz *kolumną*. Pozostaniemy wierni standardowi SQL i w dalszej części książki będziemy posługiwać się właśnie tymi terminami.

Instrukcja SELECT

Instrukcja SELECT stanowi podstawę każdego zapytania do bazy danych. Utworzenie i wykonanie instrukcji SELECT tak naprawdę oznacza przekazanie zapytania do bazy. (Zdajemy sobie sprawę, że zapewne nie musimy tego tłumaczyć, ale chcemy mieć pewność, iż każdy czytający te słowa zacznie od tego samego punktu wyjścia). Wiele programów RDBMS umożliwia zapisywanie instrukcji SELECT w postaci *zapytania*, *widoku*, *funkcji* albo *procedury składowanej*. Za każdym razem, gdy ktoś mówi, że chce przekazać zapytanie do bazy danych,

możesz być pewny, że zamierza on w jakiejś formie wykonać instrukcję SELECT. W zależności od programu RDBMS instrukcje SELECT mogą być wykonywane bezpośrednio w oknie wiersza poleceń, za pośrednictwem interaktywnego szablonu typu QBE (*Query by Example*) albo z poziomu kodu. Niezależnie od sposobu zdefiniowania i wykonania instrukcji SELECT jej składnia pozostaje bez zmian.

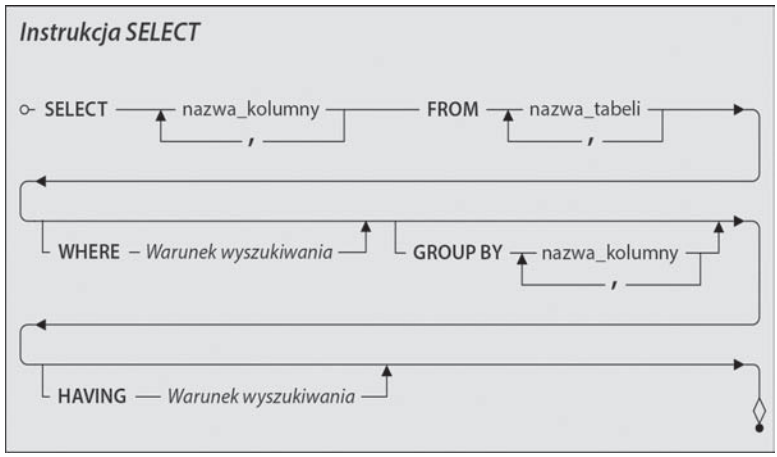
Uwaga. Wiele systemów baz danych jest wyposażonych w rozszerzenia standardu SQL umożliwiające tworzenie skomplikowanych konstrukcji programistycznych (w rodzaju `If ... Then ... Else`) w ramach funkcji i składowanych procedur, ale składnia tych konstrukcji jest inna w przypadku każdego z tych produktów. Omówienie choćby jednego lub dwóch takich języków programowania — na przykład Transact-SQL z systemu Microsoft SQL Server albo PL/SQL firmy Oracle — zdecydowanie wykracza poza zakres materiału tej książki. (Jednak w rozdziale 19. „Operacje warunkowe" omówiliśmy podstawową konstrukcję `If ... Then ... Else [CASE]`, przewidzianą w standardzie SQL). Podczas pisania funkcji i procedur składowanych dla konkretnego systemu baz danych wciąż używa się podstawowej formy instrukcji SELECT. W dalszej części tej książki używaliśmy pojęcia *widok* do nazywania zapisanych instrukcji SQL, nawet jeśli własną instrukcję SQL umieścisz w ramach funkcji albo procedury.

Instrukcja SELECT składa się z kilku szczególnych słów kluczowych, zwanych *klauzulami*. Instrukcję SELECT definiuje się poprzez zastosowanie różnych konfiguracji owych klauzul, mających na celu pozyskanie żądanych informacji. Niektóre klauzule są wymagane, inne — opcjonalne. Ponadto każda klauzula jest powiązana z jednym albo kilkoma słowami kluczowymi, które odzwierciedlają wymagane albo opcjonalne wartości. Wartości podane w klauzulach pomagają w uzyskaniu informacji żądanych przez całą instrukcję SELECT. Rysunek 4.1 przedstawia diagram ilustrujący instrukcję SELECT i jej klauzule.

Uwaga. Diagram składni pokazany na rysunku 4.1 przedstawia podstawowy wariant instrukcji SELECT. Diagram ten będziemy aktualizować i modyfikować w miarę wprowadzania i wyjaśniania nowych słów kluczowych i klauzul. Tych czytelników, którzy mają już pewne doświadczenie w konstruowaniu instrukcji SQL, prosimy o cierpliwość i wyrozumiałość.

Oto krótkie podsumowanie klauzul w instrukcji SELECT.

- SELECT. Jest to podstawowa i zarazem bezwzględnie wymagana klauzula instrukcji SELECT. Używa się jej do określenia kolumn, które mają być uwzględnione w zbiorze wyników zapytania. Kolumny te znajdują się w tabeli lub widoku określonym



Rysunek 4.1. Diagram instrukcji SELECT

w klauzuli `FROM`. (Istnieje możliwość jednoczesnego odwołania się do kolumn z różnych tabel, ale tą kwestią zajmiemy się później — w części III „Praca z wieloma tabelami"). W ramach tej klauzuli można też użyć funkcji agregujących, takich jak `SUM(PrzepracowaneGodziny)` albo wyrażeń matematycznych, jak `Ilosc * Cena`.

- `FROM`. Jest to druga najważniejsza klauzula w instrukcji `SELECT`, także wymagana. Klauzuli `FROM` używa się do określenia tabel albo widoków, z których mają być zaczerpnięte kolumny podane w klauzuli `SELECT`. Istnieje możliwość wykorzystania tej klauzuli w bardziej złożony sposób, ale o tym opowiemy w kolejnych rozdziałach.

- `WHERE`. Jest to klauzula opcjonalna, której używa się do przefiltrowania wierszy zwróconych przez klauzulę `FROM`. Po słowie kluczowym `WHERE` następuje wyrażenie, fachowo nazywane *predykatem*, które zwraca logiczną prawdę (*true*), fałsz (*false*) lub wynik nieokreślony. Rezultatu tego wyrażenia można użyć w konstrukcjach ze zwykłymi operatorami porównania, operatorami boolowskimi lub operatorami specjalnymi. O wszystkich tych aspektach klauzuli `WHERE` przeczytasz w rozdziale 6.

- `GROUP BY`. Jeśli użyjesz funkcji agregujących w klauzuli `SELECT` z myślą o uzyskaniu informacji zbiorczych, możesz użyć klauzuli `GROUP BY` do podzielenia tych informacji na grupy. Twój system bazodanowy użyje kolumny (lub listy kolumn) podanej po słowie kluczowym `GROUP BY` do pogrupowania otrzymanych informacji. Klauzula `GROUP BY` jest opcjonalna, a jej działaniu przyjrzymy się uważnie w rozdziale 13. „Grupowanie danych".

- `HAVING`. Klauzula `HAVING` filtruje rezultaty funkcji agregujących w ramach pogrupowanych informacji. Jest podobna do klauzuli `WHERE` pod tym względem, że po słowie kluczowym `HAVING` należy podać wyrażenie zwracające logiczną prawdę, fałsz lub wynik nieokreślony. Rezultatu tego wyrażenia można użyć w konstrukcjach

ze zwykłymi operatorami porównania, operatorami boolowskimi lub operatorami specjalnymi. HAVING także jest klauzulą opcjonalną, której działanie omówimy szczegółowo w rozdziale 14. „Filtrowanie zgrupowanych danych".

Najpierw zajmiemy się bardzo prostą wersją instrukcji SELECT, w której skupimy się na klauzulach SELECT i FROM. W kolejnych rozdziałach będziemy stopniowo, po jednej, wprowadzać kolejne klauzule umożliwiające konstruowanie bardziej złożonych instrukcji SELECT.

Krótka dygresja: dane a informacje

Zanim wyślemy do bazy danych pierwsze zapytanie, musimy wyjaśnić sobie jedną ważną rzecz, a mianowicie istotną różnicę między *danymi* a *informacją*. Pokrótce: dane są tym, co jest przechowywane w bazie, zaś informacje to to, co się z tej bazy pozyskuje. Tę różnicę warto sobie uświadomić, ponieważ ułatwia ona spojrzenie na omawiane kwestie z odpowiedniej perspektywy. Pamiętaj, że baza danych jest zaprojektowana tak, by dostarczała informacje cenne z perspektywy członków Twojej organizacji. Takie informacje można jednak z niej pozyskać jedynie wówczas, gdy w bazie znajdują się właściwe dane, a sama baza została opracowana w sposób adekwatny do rodzaju tych danych. Przeanalizujmy te pojęcia bardziej szczegółowo.

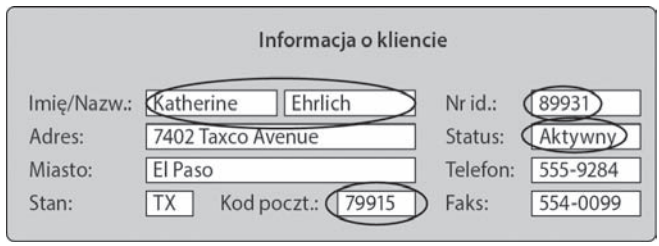
Wartości przechowywane w bazie danych to dane. Dane są statyczne w tym sensie, że pozostają w tym samym stanie, aż je zmodyfikujesz — ręcznie lub wskutek uruchomienia automatycznego procesu. Rysunek 4.2 przedstawia przykładowe dane.

Katherine	Ehrlich	89931	Aktywny	79915

Rysunek 4.2. Przykładowe, zwykłe dane

W tej postaci podane dane są pozbawione znaczenia. Nie da się na przykład łatwo określić, co oznacza wartość 89931. Czy to kod pocztowy? Numer jakiegoś podzespołu? Nawet jeśli wiesz, że jest to numer identyfikacyjny klienta, to czy rzeczywiście jest on powiązany z Katherine Ehrlich? Tego się nie dowiemy, dopóki dane nie zostaną przetworzone. Po przetworzeniu danych tak, by nabrały konkretnego znaczenia i były zrozumiałe oraz przydatne w celach poglądowych, albo nadawały się do wykorzystania podczas dalszej pracy, dane te stają się informacjami. Informacje są dynamiczne w tym sensie, że ulegają nieustannym zmianom względem danych przechowywanych w bazie, a także w tym sensie, że można je przetwarzać i prezentować na nieskończoną liczbę sposobów. Otrzymane informacje możesz wyświetlić bezpośrednio jako rezultat wydania instrukcji SELECT, umieścić je w formularzu na ekranie komputera lub wydrukować w postaci raportu. Najważniejsza sprawa do zapamiętania to że dane należy przetworzyć w sposób umożliwiający pozyskanie z nich informacji mających określone znaczenie.

Rysunek 4.3 przedstawia dane z poprzedniego przykładu przekształcone w informacje wyświetlone na ekranie klienta. Ten rysunek ilustruje możliwość wykorzystania danych w taki sposób, że staną się one zrozumiałe dla każdego, kto je przegląda.



Rysunek 4.3. Przykład danych przetworzonych na informacje

Przy pracy z instrukcjami SELECT poszczególne klauzule służą do przetwarzania *danych*, ale sama instrukcja zwraca *informacje*. Rozumiesz już, na czym to polega?

Jest jeszcze jedna kwestia, którą należy poruszyć. Otóż instrukcja SELECT zwykle zwraca jeden lub kilka wierszy informacji — dokładna ich liczba zależy od sposobu sformułowania tej instrukcji. Te wiersze łącznie są nazywane *zbiorem rezultatów* i właśnie takim pojęciem będziemy się posługiwać w dalszej części tej książki. Ta nazwa jest wyjątkowo trafna, ponieważ podczas korzystania z relacyjnej bazy danych zawsze pracuje się ze zbiorami danych. (Przypominamy, że model relacyjny w pewnym stopniu opiera się na teorii zbiorów). Informacje w zbiorze rezultatów można łatwo przeglądać, a w wielu przypadkach także modyfikować. Ponownie jednak wszystko to zależy od sposobu skonstruowania instrukcji SELECT.

Zabierzmy się zatem do pracy i zacznijmy używać instrukcji SELECT.

Przekładanie żądania na SQL

Gdy chcesz pozyskać z bazy danych jakieś informacje, to takie żądanie zwykle ma formę pytania lub zdania implikującego pytanie. Żądanie można sformułować na przykład tak:

W jakich miastach mieszkają nasi klienci?

Pokaż mi bieżącą listę pracowników wraz z ich numerami telefonów.

Jakiego rodzaju zajęcia obecnie oferujemy?

Podaj mi nazwiska ludzi z naszego zespołu i daty ich zatrudnienia.

Kiedy już będziesz wiedział, o co chcesz zapytać, możesz nadać swojemu żądaniu bardziej oficjalną formę. Przekładu należy dokonać na podstawie poniższego wzoru:

Wybierz <obiekt> z <źródło>

Zacznij od przeanalizowania swojego żądania i zastąp słowa lub frazy takie jak „zrób listę", „pokaż mi", „którzy" albo „kto" słowem „wybierz". Następnie wyszukaj w żądaniu rzeczowniki i sprawdź, czy dany rzeczownik odpowiada obiektowi, który chciałbyś wyświetlić, czy raczej nazwie tabeli, w której ów obiekt może się znajdować. Jeśli rzeczownik jest obiektem, zastąp nim pozycję <obiekt> we wzorze przekładu. Jeśli jest to nazwa tabeli, zastąp nią element <źródło>. Jeżeli według tego wzoru przetłumaczysz na przykład pierwsze pytanie z poprzedniej listy, to otrzymana instrukcja będzie wyglądała mniej więcej tak:

Wybierz miasto z tabeli klienci

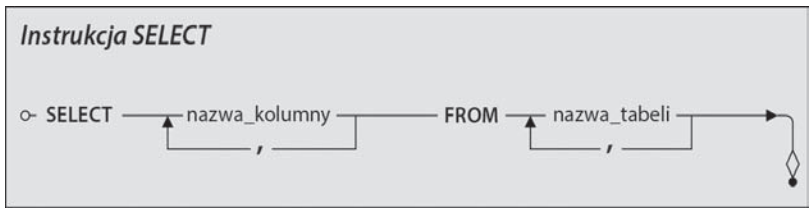
Po dokonaniu przekładu na postać przypominającą zapytanie otrzymane zdanie trzeba przekształcić na instrukcję `SELECT` z prawdziwego zdarzenia, korzystając ze składni SQL pokazanej na rysunku 4.4. Pierwszy krok powinien polegać na uporządkowaniu wstępnego przekładu. Aby to zrobić, wystarczy wykreślić wszystkie słowa, które nie są rzeczownikami odpowiadającymi nazwie kolumny lub tabeli, a także wszelkie słowa nienależące do składni SQL. Oto przykład wcześniejszego zdania po takim oczyszczeniu:

Wybierz miasto z ~~tabeli~~ `Klienci`

Po usunięciu wykreślonych słów i zastąpieniu słów kluczowych odpowiednikami z SQL w języku angielskim otrzymujemy gotową instrukcję `SELECT`:

```
SELECT Miasto FROM Klienci
```

Słowniczek	wybierz, wybrane etc. — `SELECT` z — `FROM`



Rysunek 4.4. Składnia prostej instrukcji SELECT

Uwaga. Autorzy książki przyjęli koncepcję etapowego „tłumaczenia" zwykłych zdań na zapytania SQL. W języku polskim wiąże się to z dodatkową trudnością, wynikającą z konieczności przełożenia słów i określeń używanych w mowie potocznej na odpowiedniki w postaci poszczególnych instrukcji i klauzul SQL. Aby to ułatwić, przy wprowadzaniu kolejnych elementów języka w książce zostały zawarte słowniczki podobne do zamieszczonego powyżej. W dodatku E „Słowniczek" znajduje się lista najważniejszych zwrotów.

Przedstawionego przed chwilą trzyetapowego procesu „tłumaczenia" można użyć w odniesieniu do dowolnego żądania, jakie zamierzasz przekazać do bazy danych. Co więcej, techniki tej używamy przez większą część tej książki i podczas nauki konstruowania tego rodzaju instrukcji zachęcamy Cię do posługiwania się nią. Z upływem czasu, gdy opanujesz już konstruowanie instrukcji `SELECT`, nauczysz się intuicyjnie łączyć wszystkie wymienione czynności w jedną operację.

Pamiętaj, że na początku nauki posługiwania się SQL będziesz pracował głównie z kolumnami i tabelami. Dobrze pokazuje to diagram składni przedstawiony na rysunku 4.4, na którym po klauzuli `SELECT` następuje tylko pozycja `nazwa_kolumny`, zaś po klauzuli `FROM` — pozycja `nazwa_tabeli`. W następnym rozdziale dowiesz się, w jaki sposób używać innych wyrażeń w ramach tych klauzul, by konstruować bardziej złożone instrukcje `SELECT`.

Zapewne zwróciłeś uwagę na to, że żądanie omówione w poprzednim przykładzie jest stosunkowo proste. Łatwo było przetłumaczyć je na instrukcję i określić nazwy kolumny i tabeli, które powinny w niej wystąpić. Ale co zrobić, jeśli żądanie nie jest tak proste do przetłumaczenia i trudno jest w nim zidentyfikować kolumny, które należy podać w instrukcji `SELECT`? Najprostsze wyjście polega na dopracowaniu i skonkretyzowaniu go. Na przykład żądanie w rodzaju *Wyświetl mi informacje o naszych klientach* można sprecyzować i sformułować tak: *Wyświetl nazwiska, miasta zamieszkania i numery telefonów wszystkich naszych klientów*. Jeśli doprecyzowanie żądania nie rozwiązuje problemu, nadal pozostają Ci dwa wyjścia. Pierwsze polega na sprawdzeniu, czy w tabeli podanej w klauzuli `FROM` instrukcji `SELECT` znajdują się jakieś kolumny, które pozwoliłyby na sprecyzowanie żądania i w ten sposób ułatwiły przekształcenie go na instrukcję. Druga opcja polega na uważniejszym przeanalizowaniu żądania pod kątem tego, czy jakieś zawarte w nim słowo lub fraza *implikują* konkretne nazwy kolumn. Możliwość zastosowania jednego lub obydwu tych wyjść zależy od samego żądania. Warto po prostu o nich pamiętać, gdy zetkniesz się z pytaniem, które trudno Ci będzie przełożyć na instrukcję. Przyjrzyjmy się przykładom zastosowania obydwu wymienionych technik i sposobowi ich zastosowania w typowej sytuacji.

Aby zilustrować pierwszą technikę, przypuśćmy, że chcesz przełożyć na instrukcję następujące żądanie:

Potrzebuję imion, nazwisk i adresów wszystkich naszych pracowników.

Pozornie to żądanie wygląda na bardzo proste. Ale jeśli uważnie mu się przyjrzysz, dostrzeżesz jeden drobny problem: choć da się na jego podstawie określić nazwę docelowej tabeli (Pracownicy), to nic w tym żądaniu nie wskazuje na kolumny, jakie należy w tej instrukcji uwzględnić. Choć w żądaniu pojawiają się słowa takie jak *nazwiska* i *adresy*, są to pojęcia dość ogólne. Ten problem można rozwiązać poprzez zapoznanie się ze zidentyfikowaną przed chwilą tabelą i określenie, czy zawiera ona jakieś kolumny, które można byłoby podstawić pod niejednoznaczne określenia. Jeśli tak, użyj nazw tych kolumn w docelowej instrukcji. (Jeżeli pomoże Ci to w lepszym wyobrażeniu sobie instrukcji, to w przetłumaczonym żądaniu możesz użyć uogólnionych nazw kolumn, pamiętaj jedynie, że w docelowej składni SQL będziesz musiał zastosować te prawdziwe). W tym przypadku w tabeli Pracownicy pokazanej na rysunku 4.5 należy poszukać takich nazw kolumn, którymi można byłoby zastąpić słowa „imiona", „nazwiska" i „adresy".

Rysunek 4.5. Struktura tabeli Pracownicy

Aby w pełni zrealizować żądanie dotyczące „imion, nazwisk i adresów", trzeba użyć sześciu kolumn z podanej tabeli. Kolumny `ImiePracownika` i `NazwiskoPracownika` rozwiążą kwestię „imion i nazwisk", zaś `AdresPracownika`, `MiastoPracownika`, `StanZamPracownika` i `KodPocztowyPracownika` spełnią wymagania związane z podaniem „adresów". Teraz można zastosować opisany wcześniej proces tłumaczenia żądania, które dla ułatwienia powtórzymy raz jeszcze. (W przetłumaczonym żądaniu użyjemy ogólnych nazw kolumn, a w instrukcji SQL — już tych prawdziwych).

Potrzebuję imion, nazwisk i adresów wszystkich naszych pracowników.

Tłumaczenie	Wybierz imię, nazwisko, adres, miasto, stan oraz kod pocztowy z tabeli Pracownicy
Porządkowanie	Wybierz imię, nazwisko, adres, miasto, stan, ~~oraz~~ kod pocztowy z ~~tabeli~~ Pracownicy
SQL	`SELECT ImiePracownika, NazwiskoPracownika, AdresPracownika, MiastoPracownika, StanZamPracownika, KodPocztowyPracownika FROM Pracownicy`

Uwaga. Powyższy przykład przystępnie ilustruje możliwość zastosowania kilku kolumn w klauzuli `SELECT`. Możliwość tę omówimy bardziej szczegółowo w dalszej części tego rozdziału.

Kolejny przykład ilustruje drugą z wymienionych wcześniej metod, która polega na sprawdzeniu, czy jakieś słowa w żądaniu implikują nazwy kolumn. Przypuśćmy, że chcemy poddać tłumaczeniu następujące żądanie:

Jakiego rodzaju zajęcia obecnie oferujemy?

Na pierwszy rzut oka przetłumaczenie tego żądania na instrukcję wydaje się trudne. W żądaniu nie padają nazwy kolumn, a skoro nie mamy żadnego obiektu do wybrania, to nie możemy skonstruować kompletnej instrukcji. Co zrobić w takiej sytuacji? Przyjrzyj

się każdemu słowu w żądaniu i zastanów się, czy któreś z nich *implikuje* nazwę kolumny w tabeli Zajecia. Zanim przystąpisz do dalszego czytania, poświęć chwilę na przemyślenie tego żądania. Czy dostrzegasz takie słowo?

W tym przypadku słowem mogącym sugerować nazwę kolumny w tabeli Zajecia jest „rodzaj". Dlaczego? Ponieważ przy założeniu, że mamy do czynienia z bazą danych szkoły lub uczelni, rodzaj zajęć można zinterpretować jako „przedmiot" lub „dziedzina". Jeśli w tabeli Zajecia istnieje kolumna o nazwie Przedmiot, to znaczy, że właśnie tej kolumny potrzebujesz, by skompletować przekład żądania, a w rezultacie otrzymać instrukcję `SELECT`. Przypuśćmy, że w tabeli Zajecia rzeczywiście istnieje kolumna o nazwie Przedmiot, i na tej podstawie ponownie przeprowadźmy opisany wcześniej trzyetapowy proces translacji.

Jakiego rodzaju zajęcia obecnie oferujemy?

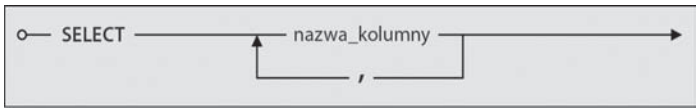
Tłumaczenie	Wybierz przedmiot z tabeli zajęcia
Porządkowanie	Wybierz przedmiot z ~~tabeli~~ zajęcia
SQL	`SELECT Przedmiot` `FROM Zajecia`

Jak wynika z tego przykładu, omawiana technika polega na umiejętnym wyszukiwaniu synonimów, którymi można zastąpić określone słowa lub frazy w żądaniu. Jeśli znajdziesz słowo albo frazę, które mogą sugerować nazwę kolumny, spróbuj zastąpić je synonimem. Może się okazać, że trafnie dobrany synonim rzeczywiście będzie pasował do jednej z kolumn w bazie danych. Jeżeli pierwszy synonim, jaki przyjdzie Ci do głowy, nie zadziała — wypróbuj inny. Kontynuuj ten proces, aż znajdziesz synonim pasujący do jednej z kolumn bądź do chwili, gdy uznasz, że ani oryginalnego określenia, ani żadnego z określeń bliskoznacznych nie da się dopasować do kolumn w tabeli.

Uwaga. Jeśli nie zostało podane inaczej, nazwy kolumn i tabel użyte w składni SQL instrukcji zostały zaczerpnięte z przykładowych baz danych, przedstawionych w dodatku B „Schematy przykładowych baz danych". Ta zasada dotyczy wszystkich przykładów w dalszej części książki (z niewielkimi odstępstwami związanymi z koniecznością dopasowania poziomu skomplikowania zapytań do bieżącego etapu nauki).

Rozszerzanie zakresu działań

Za pomocą instrukcji `SELECT` można odwołać się do kilku kolumn równie łatwo jak do jednej. Wystarczy wymienić żądane kolumny po klauzuli `SELECT`, rozdzielając ich nazwy przecinkami. Na diagramie składni pokazanym na rysunku 4.6 możliwość użycia kilku kolumn została pokazana w postaci linii biegnącej od prawej do lewej strony, pod napisem *nazwa_kolumny*. Przecinek pośrodku linii oznacza, że przed wpisaniem nazwy kolejnej kolumny, do której chcesz się odwołać za pomocą klauzuli `SELECT`, należy wstawić przecinek.



Rysunek 4.6. Składnia klauzuli SELECT umożliwiająca odwołanie się do wielu kolumn

Możliwość podania wielu kolumn w instrukcji SELECT umożliwia znalezienie odpowiedzi na pytania takie jak poniższe:

Zrób aktualną listę pracowników wraz z ich numerami telefonów.

Tłumaczenie	Wybierz nazwisko, imię oraz numer telefonu wszystkich naszych pracowników z tabeli Pracownicy
Porządkowanie	Wybierz nazwisko, imię, ~~oraz~~ numer telefonu ~~wszystkich naszych pracowników~~ z ~~tabeli~~ Pracownicy
SQL	`SELECT NazwiskoPracownika, ImiePracownika, TelefonPracownika FROM Pracownicy`

Jak nazywają się i ile kosztują produkty, które sprzedajemy, i do jakiej kategorii należy każdy z nich?

Tłumaczenie	Wybierz nazwę, cenę oraz kategorię wszystkich produktów z tabeli Produkty
Porządkowanie	Wybierz nazwę, cenę, ~~oraz~~ kategorię ~~wszystkich produktów~~ z ~~tabeli~~ Produkty
SQL	`SELECT NazwaProduktu, CenaDetaliczna, Kategoria FROM Produkty`

Odwołanie się do kilku kolumn w instrukcji SELECT ma tę zaletę, że umożliwia wyświetlenie większego spektrum informacji. Nawiasem mówiąc, kolejność kolumn w klauzuli SELECT nie jest istotna — można wymienić je w dowolnym porządku. To zaś umożliwia wyświetlenie tych samych informacji na różne sposoby.

Przypuśćmy, że masz do dyspozycji tabelę pokazaną na rysunku 4.7 i zostałeś poproszony o skonstruowanie podanego niżej żądania do bazy danych:

Rysunek 4.7. Struktura tabeli Przedmioty

Podaj listę przedmiotów, dziedzinę, do której należy każdy z nich, oraz przypisane im numery kodowe w naszym katalogu. Chciałbym, aby zestawienie rozpoczynało się od przedmiotu, po nim następowała dziedzina, a na końcu kod.

To żądanie także można przekształcić na odpowiednią instrukcję SELECT, pomimo że zgłaszająca je osoba życzy sobie ustawienia kolumn w określonym porządku. Aby uzyskać taki efekt, wystarczy podać nazwy kolumn w oczekiwanej kolejności przy tłumaczeniu żądania na instrukcję. Oto jak powinien wyglądać proces translacji wspomnianego żądania na instrukcję SELECT:

Słowniczek	Pola kluczy często są nazywane z użyciem przedrostka ID, od określenia *identyfikator*. W dalszej części książki w celu skrócenia zapytań skrót ID będzie coraz częściej stosowany już na etapie tłumaczenia.

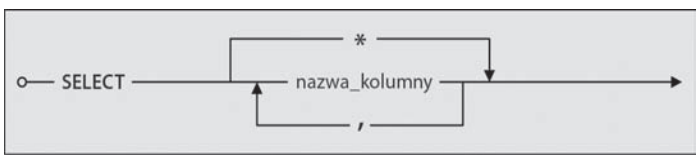
Tłumaczenie	Wybierz przedmiot, identyfikator dziedziny oraz kod przedmiotu z tabeli Przedmioty
Porządkowanie	Wybierz przedmiot, ~~identyfikator~~ ID dziedziny, ~~oraz~~ kod przedmiotu z ~~tabeli~~ Przedmioty
SQL	`SELECT NazwaPrzedmiotu, IDDziedziny, KodPrzedmiotu` `FROM Przedmioty`

Zastosowanie skrótu umożliwiającego odwołanie do wszystkich kolumn

Liczba kolumn w klauzuli SELECT nie jest w żaden sposób ograniczona — w praktyce można w ten sposób odwołać się nawet do wszystkich kolumn z docelowej tabeli. Poniższy przykład pokazuje instrukcję SELECT, której można użyć w celu odwołania się do wszystkich kolumn z tabeli Przedmioty, pokazanej na rysunku 4.7.

```
SQL     SELECT IDPrzedmiotu, IDDziedziny, KodPrzedmiotu, NazwaPrzedmiotu,
        OpisPrzedmiotu
        FROM Przedmioty
```

Jeśli źródłowa tabela zawiera wiele kolumn, to wymienienie ich wszystkich w powyższy sposób wymaga mnóstwa pisania! Na szczęście standard SQL uwzględnia skrót w postaci gwiazdki (asterysku), umożliwiający zdecydowane „odchudzenie" takiej instrukcji. Diagram składni pokazany na rysunku 4.8 ilustruje możliwość użycia gwiazdki jako alternatywy dla listy kolumn po klauzuli SELECT.



Rysunek 4.8. Składnia z uproszczeniem w postaci gwiazdki

Jeśli chcesz odwołać się do wszystkich kolumn w tabeli źródłowej podanej w klauzuli FROM, wystarczy, że umieścisz gwiazdkę od razu po klauzuli SELECT. Na przykład poprzednia instrukcja SELECT zapisana z użyciem gwiazdki wygląda następująco:

```
SQL                SELECT * FROM Przedmioty
```

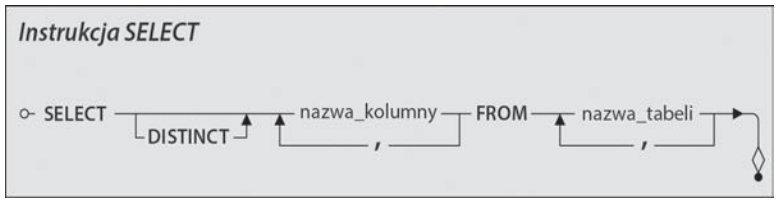
Ta postać instrukcji wymaga zdecydowanie mniej pisania! Konstruowanie instrukcji SELECT w opisany sposób wiąże się jednak z pewną kwestią: otóż gwiazdka odpowiada wszystkim kolumnom *aktualnie* znajdującym się w tabeli źródłowej, co oznacza, że dodanie lub usunięcie kolumn w tej tabeli będzie miało wpływ na zbiór rezultatów otrzymany przy użyciu instrukcji SELECT. (Co ciekawe, w standardzie SQL zostało powiedziane, że dodanie albo usunięcie kolumn *nie powinno* mieć wpływu na zbiór rezultatów). Kwestia ta jest jednak istotna tylko wtedy, gdy zależy Ci, by w zbiorze rezultatów za każdym razem były uwzględnione dokładnie te same kolumny. W przypadku użycia instrukcji SELECT z gwiazdką system bazodanowy nie ostrzeże Cię, jeśli jakieś kolumny zostały usunięte z tabeli źródłowej, ale ostrzeżenie takie pojawi się w przypadku, gdy nie uda mu się znaleźć kolumny, do której odwołałeś się *w sposób jawny*. Choć akurat dla naszych celów nie ma to żadnego znaczenia, to kwestia ta stanie się znacznie bardziej istotna, jeśli zagłębisz się w świat programowania z użyciem SQL. My wychodzimy z następującego założenia: gwiazdki należy używać tylko w przypadku „roboczych" zapytań, aby wyświetlić komplet informacji z danej tabeli. W przeciwnym razie lepiej wymienić w zapytaniu wszystkie potrzebne kolumny. Takie zapytanie zwróci dokładnie te same informacje, na których Ci zależy, ale będzie bardziej oczywiste.

Podane dotychczas przykłady opierają się na prostych żądaniach wymagających odwołania się do kolumn z tylko jednej tabeli. W części III nauczysz się konstruować bardziej skomplikowane zapytania, wymagające sięgnięcia do kolumn z kilku tabel.

Eliminowanie powtarzających się wierszy

Podczas pracy z instrukcjami SELECT bez wątpienia będziesz miał do czynienia ze zbiorami rezultatów, w których niektóre wiersze będą się powtarzały. Otrzymanie takiego zbioru rezultatów nie jest powodem do niepokoju. Aby wyeliminować ze zbioru rezultatów wszelkie powtarzające się wiersze, wystarczy użyć w instrukcji SELECT słowa kluczowego DISTINCT. Rysunek 4.9 przedstawia diagram składni dla słowa kluczowego DISTINCT.

Jak widać na diagramie, DISTINCT jest opcjonalnym słowem kluczowym, poprzedzającym listę kolumn w klauzuli SELECT. Użycie słowa kluczowego DISTINCT oznacza, że oczekujemy od systemu baz danych potraktowania wartości wszystkich wybranych kolumn w danym wierszu *jako całości* i na tej podstawie wyeliminowania powtarzających się wierszy. Pozostałe, unikatowe wiersze są następnie umieszczane w zbiorze rezultatów. Poniższy przykład pokazuje różnicę wynikającą z zastosowania słowa kluczowego DISTINCT w określonych okolicznościach.



Rysunek 4.9. Składnia słowa kluczowego DISTINCT

Przypuśćmy, że chcesz przekazać do bazy danych następujące żądanie:

Z jakich miast pochodzą członkowie naszej ligi kręglarskiej?

Problem wydaje się prosty, poddajesz go więc operacji tłumaczenia.

Tłumaczenie	Wybierz miasto z tabeli kręglarze
Porządkowanie	Wybierz miasto z ~~tabeli~~ kręglarze
SQL	`SELECT Miasto FROM Kreglarze`

Problem polega na tym, że w zbiorze rezultatów dla takiej instrukcji `SELECT` pojawią się *wszystkie wystąpienia* każdej nazwy miasta znajdującej się w tabeli Kreglarze. Na przykład jeśli 20 graczy pochodzi z Bellevue, 7 z Kent, a 14 z Seattle, to w zbiorze rezultatów 20 razy pojawi się słowo Bellevue, 7 razy słowo Kent i 14 razy słowo Seattle. Bezsprzecznie wszystkie te nadmiarowe informacje są niepotrzebne. Wystarczy wyświetlić *jedno* wystąpienie każdej nazwy miasta znajdującej się w tabeli Kreglarze. Ten problem można rozwiązać dzięki zastosowaniu w instrukcji `SELECT` słowa kluczowego `DISTINCT`, które wyeliminuje nadmiarowe informacje.

Poddajmy to żądanie ponownemu procesowi tłumaczenia, ale tym razem z użyciem słowa kluczowego `DISTINCT`. Zwróć uwagę, że słowo to pojawia się już na etapie tłumaczenia i pozostaje po uporządkowaniu zapytania.

Słowniczek	jednokrotnie — `DISTINCT`

Tłumaczenie	Wybierz jednokrotnie miasto z tabeli kręglarze
Porządkowanie	Wybierz jednokrotnie miasto z ~~tabeli~~ kręglarze
SQL	`SELECT DISTINCT Miasto FROM Kreglarze`

Zbiór rezultatów dla tej instrukcji `SELECT` będzie zawierał dokładnie to, czego oczekujesz — czyli po jednym wystąpieniu nazwy każdego miasta z tabeli `Kreglarze`.

Słowa kluczowego `DISTINCT` można użyć także w odniesieniu do wielu kolumn. Zmodyfikujmy poprzedni przykład tak, by odczytać z tabeli `Kreglarze` nie tylko miasto, ale też stan. Nowa instrukcja `SELECT` wygląda teraz następująco:

```
SELECT DISTINCT Miasto, Stan FROM Kreglarze
```

Ta instrukcja SELECT zwróci zbiór rezultatów zawierający unikatowe rekordy i uwzględni różnice między miastami o tych samych nazwach, ale znajdującymi się w różnych stanach USA. Jako odrębne zostaną potraktowane na przykład rezultaty „Portland, ME" i „Portland, OR", a także „Hollywood, CA" i „Hollywood, FL". Warto wiedzieć, że większość systemów baz danych sortuje rezultaty zgodnie z kolejnością kolumn, podane wyżej wartości będą więc posortowane następująco: „Hollywood, CA", „Hollywood, FL", „Portland, ME", „Portland, OR". Standard SQL nie przewiduje jednak sortowania rezultatów w taki sposób. Jeśli chcesz mieć pewność co do kolejności sortowania, zapoznaj się z działaniem klauzuli ORDER BY, opisanej w następnej części tego rozdziału.

Słowo kluczowe DISTINCT we właściwych okolicznościach jest bardzo przydatnym narzędziem. Powinieneś jednak stosować je tylko wtedy, gdy chcesz, aby wartości w zbiorze rezultatów się nie powtarzały.

Ostrzeżenie. W systemach baz danych wyposażonych w interfejs graficzny zwykle można zażyczyć sobie, by zbiór rezultatów zapytania został wyświetlony w postaci siatki wierszy i kolumn, której zawartość można aktualizować. W kolumnie albo wierszu można wpisać nową wartość, a system zaktualizuje wartość źródłową z danej tabeli. (Tak naprawdę system baz danych wykonuje wówczas „za kulisami" zapytanie typu UPDATE, o którym przeczytasz w rozdziale 15. „Aktualizowanie zbiorów danych).

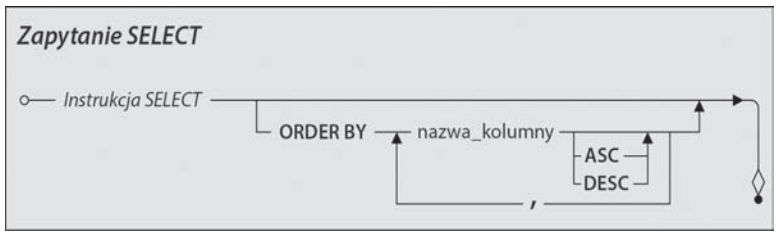
Jeśli jednak w zapytaniu zostanie zawarte słowo kluczowe DISTINCT, to we wszystkich systemach baz danych, z którymi mieliśmy do czynienia, zwróconego zbioru wartości nie będzie można aktualizować w opisany sposób. Aby możliwa była zmiana wartości w danej kolumnie i wierszu, system bazodanowy musi jednoznacznie zidentyfikować kolumnę i wiersz, w których chcesz wprowadzić zmiany. Tymczasem w przypadku użycia słowa DISTINCT wartości wyświetlane w poszczególnych wierszach są rezultatem przetworzenia na przykład dziesiątków powtarzających się wierszy. Jeśli spróbujesz zaktualizować jedną z kolumn, baza danych nie będzie wiedziała, którego wiersza ma dotyczyć zmiana. System nie wie też, czy przypadkiem nie chciałeś w ten sposób zmienić wszystkich wierszy, w których dana wartość się powtarza.

Sortowanie informacji

Na początku tego rozdziału napisaliśmy, że operację SELECT można podzielić na trzy mniejsze: instrukcję SELECT, wyrażenie SELECT oraz zapytanie SELECT. Stwierdziliśmy też, że wymienione operacje można łączyć na różne sposoby, aby móc realizować złożone żądania. Operacje te należy połączyć także w celu posortowania wierszy w zbiorze rezultatów.

Z definicji wiersze w zbiorze rezultatów zwróconym przez instrukcję SELECT są nieuporządkowane. Kolejność, w jakiej zostaną wyświetlone, zwykle odpowiada

ich fizycznemu położeniu w tabeli. (Konkretna sekwencja wierszy często jest generowana dynamicznie, w zależności od tego, jaką sekwencję system baz danych uzna za najbardziej efektywną z perspektywy Twojego zapytania). Jedyny sposób na posortowanie zbioru rezultatów polega na umieszczeniu instrukcji SELECT w zapytaniu SELECT, tak jak ilustruje to rysunek 4.10. Zapytanie SELECT z instrukcją SELECT definiujemy przy użyciu klauzuli ORDER BY. Klauzula ORDER BY zapytania SELECT umożliwia określenie kolejności wierszy w otrzymanym zbiorze rezultatów. W kolejnych rozdziałach przeczytasz o tym, że instrukcję SELECT można zawrzeć w innej instrukcji SELECT albo w wyrażeniu SELECT w celu uzyskania odpowiedzi na bardzo złożone żądania. Jednak zapytanie SELECT można uznać za „najwyższy poziom" omawianej operacji — nie da się go zagnieździć w innych konstrukcjach.



Rysunek 4.10. Diagram składni zapytania SELECT

Uwaga. W całej książce posługujemy się tymi samymi pojęciami, które definiuje standard SQL bądź które są powszechnie wykorzystywane w większości systemów baz danych. Starsze wersje standardu SQL definiowały klauzulę ORDER BY jako element *wskaźnika* (ang. *cursor*; obiektu definiowanego wewnątrz aplikacji), element *tablicy* (listy wartości tworzących logiczną tabelę, taką jak rezultat *podzapytań*, omówionych w rozdziale 11. „Podzapytania") bądź jako element *podzapytania skalarnego* (podzapytania zwracającego tylko jedną wartość). Dogłębne omówienie wskaźników i tablic wykracza poza ramy tej książki.

Ponieważ niemal wszystkie implementacje SQL umożliwiają umieszczenie klauzuli ORDER BY na końcu instrukcji SELECT, którą można zapisać w postaci widoku, postanowiliśmy stworzyć pojęcie *zapytanie SELECT*, które opisywałoby ten rodzaj instrukcji. To określenie umożliwiło nam też omówienie koncepcji sortowania wyników zapytania w celu wyświetlenia rezultatów online lub użycia ich w raporcie. Najnowsza norma SQL z roku 2011 definiuje pojęcie <*specyfikacja zapytania*> (ang. <*query specification*>) odpowiadające naszej *instrukcji SELECT* oraz pojęcie <*wyrażenie zapytania*> (ang. <*query expression*>) odpowiadające konstrukcji, którą my nazwaliśmy *zapytaniem SELECT*. W tym jednym przypadku pozwoliliśmy sobie odbiec od nazewnictwa określonego w standardzie i zastosowaliśmy naszą terminologię.

Klauzula ORDER BY umożliwia posortowanie zbioru rezultatów danej instrukcji SELECT na podstawie jednej kolumny lub większej ich liczby, a także pozwala na określenie kolejności sortowania dla każdej z tych kolumn: rosnącej albo malejącej. Jedyne kolumny, jakich można użyć w klauzuli ORDER BY, to te, które zostały wymienione w klauzuli SELECT. (Choć wymóg ten jest określony w standardzie SQL, to w niektórych implementacjach języka istnieje możliwość zignorowania go i użycia dowolnej kolumny z dowolnej tabeli wymienionej w klauzuli FROM. My jednak będziemy przestrzegać tego zalecenia we wszystkich przykładach podanych w książce). Jeśli w klauzuli ORDER BY wymienisz dwie lub więcej kolumn, rozdziel ich nazwy przecinkami. Zapytanie SELECT zwróci ostateczny zbiór rezultatów dopiero po zakończeniu sortowania.

Uwaga. Klauzula ORDER BY *nie ma wpływu* na fizyczną kolejność wierszy w tabeli. Jeśli zależy Ci na zmianie fizycznej kolejności wierszy, poszukaj odpowiedniej procedury w dokumentacji Twojego systemu baz danych.

Zacznijmy od podstaw: kolejność sortowania

Zanim przyjrzymy się przykładom zastosowania zapytania SELECT, musimy napisać kilka słów na temat kolejności sortowania. Otóż to, jak klauzula ORDER BY sortuje informacje, jest uzależnione od sekwencji sortowania wykorzystywanej przez Twój program do obsługi baz danych. Sekwencja sortowania decyduje o kolejności wszystkich znaków zawartych w puli znaków dla bieżącego języka ustawionego w systemie operacyjnym. Sekwencja określa na przykład, czy małe litery będą miały pierwszeństwo przed wielkimi albo czy wielkość znaków w ogóle będzie miała jakiekolwiek znaczenie. Sprawdź informacje na ten temat w dokumentacji systemu baz danych lub skonsultuj się z administratorem bazy danych, aby zapoznać się z domyślną sekwencją sortowania w Twojej bazie. Więcej informacji o sekwencjach sortowania znajdziesz w sekcji „Uwaga dotycząca porównywania ciągów znaków" w rozdziale 6.

Przywołajmy wyniki do porządku

Dzięki zastosowaniu klauzuli ORDER BY informacje zaczerpnięte z bazy danych można wyświetlić w bardziej przemyślany sposób. Odnosi się to zarówno do prostych żądań, jak i tych skomplikowanych. Żądania możesz sformułować tak, by definiowały także sposób sortowania. Na przykład pytanie w rodzaju *Zajęcia o jakiej tematyce obecnie oferujemy?* można zadać następująco: *Podaj tematykę zajęć, jakie oferujemy, i wymień je w kolejności alfabetycznej.*

Zanim przystąpimy do pracy nad zapytaniami SELECT, musimy nieco zmodyfikować sposób tłumaczenia instrukcji. Będzie to polegało na dodaniu nowej sekcji na końcu przetłumaczonej instrukcji, uwzględniającej metodę sortowania określoną w żądaniu. Nowy szablon translacji żądania wygląda następująco:

Wybierz <obiekt> z <źródło> **i uporządkuj według <kolumn(y)>**

Jeśli żądanie będzie zawierało sformułowania takie jak „posortuj rezultaty według miasta", „wyświetl je w kolejności chronologicznej" bądź „ułóż według nazwiska i imienia", przyjrzyj się mu uważnie, aby określić, która kolumna albo kolumny będą konieczne w celu odpowiedniego posortowania wyników. Jest to o tyle proste, że większość ludzi naturalnie posługuje się podanymi sformułowaniami, a wybór kolumn, na podstawie których zostaną posortowane wyniki, zwykle jest oczywisty. Po zidentyfikowaniu właściwej kolumny albo kolumn zastąp ich nazwami określenie <kolumn(y)> w szablonie translacji. Przyjrzyjmy się prostemu żądaniu, aby się przekonać, jak to działa:

Podaj tematykę zajęć, jakie oferujemy, i wymień je w kolejności alfabetycznej.

Słowniczek	uporządkuj według — `ORDER BY`

Tłumaczenie	Wybierz dziedzinę z tabeli zajęcia i uporządkuj według dziedziny
Porządkowanie	Wybierz dziedzinę z ~~tabeli~~ zajęcia ~~i~~ uporządkuj według dziedziny

SQL	`SELECT Dziedzina`
	`FROM Zajecia`
	`ORDER BY Dziedzina`

W tym przykładzie musimy założyć, że do sortowania trzeba użyć kolumny `Dziedzina`, ponieważ jest to jedyna kolumna podana w żądaniu. Musimy też przyjąć, że sortowanie powinno się odbyć w kolejności rosnącej, ponieważ nic nie wskazuje, by miało być inaczej. Jest to bezpieczne założenie. Zgodnie ze standardem SQL, jeśli kolejność sortowania nie zostanie jawnie określona, to baza powinna automatycznie przyjąć kolejność rosnącą. Jeśli chciałbyś wyrazić to oczekiwanie w sposób dosłowny, to po nazwie kolumny `Dziedzina` w klauzuli `ORDER BY` wstaw słowo `ASC`.

W poniższym, kolejnym żądaniu kolumna, według której ma się odbyć sortowanie, została jasno sprecyzowana:

Wyświetl listę nazw dostawców uporządkowaną według numeru kodu pocztowego.

Tłumaczenie	Wybierz nazwę dostawcy oraz kod pocztowy z tabeli dostawcy i uporządkuj je według kodu pocztowego
Porządkowanie	Wybierz nazwę dostawcy, ~~oraz~~ kod pocztowy z ~~tabeli~~ dostawcy ~~i~~ uporządkuj je według kodu pocztowego

SQL	`SELECT NazwaDostawcy, KodPocztowyDostawcy`
	`FROM Dostawcy`
	`ORDER BY KodPocztowyDostawcy`

Ludzie, którym zależy na posortowaniu informacji w kolejności malejącej, na ogół to po prostu podkreślają. W sytuacji, gdy będziesz potrzebował wyświetlić zbiór rezultatów w odwrotnej kolejności, wstaw słowo kluczowe `DESC` po nazwie odpowiedniej kolumny

w klauzuli ORDER BY. Na przykład tak należałoby zmodyfikować instrukcję SELECT z poprzedniego przykładu, aby wyświetlić informacje posortowane według kodu pocztowego w kolejności malejącej.

Słowniczek	malejąco — DESC rosnąco — ASC

SQL
```
SELECT NazwaDostawcy, KodPocztowyDostawcy
FROM Dostawcy
ORDER BY KodPocztowyDostawcy DESC
```

Następny przykład ilustruje bardziej złożone żądanie, wymagające sortowania względem kilku kolumn. Jedyna różnica między tym przykładem a dwoma poprzednimi polega na tym, że tym razem w klauzuli ORDER BY jest podana większa liczba kolumn. Zwróć uwagę, że nazwy kolumn są rozdzielone przecinkami, co jest zgodne z diagramem składni pokazanym wcześniej na rysunku 4.10.

Wyświetl imiona i nazwiska pracowników, ich numery telefonów oraz identyfikatory i uporządkuj listę alfabetycznie według nazwiska i imienia.

Tłumaczenie Wybierz nazwisko, imię, numer telefonu oraz identyfikator pracownika z tabeli pracownicy i uporządkuj według nazwiska i imienia

Porządkowanie Wybierz nazwisko, imię, numer telefonu, ~~oraz identyfikator~~ ID pracownika z ~~tabeli~~ pracownicy ~~i~~ uporządkuj według nazwiska, ~~i~~ imienia

SQL
```
SELECT NazwiskoPracownika, ImiePracownika, TelefonPracownika,
IDPracownika
FROM Pracownicy
ORDER BY NazwiskoPracownika, ImiePracownika
```

Jedna z interesujących rzeczy, jakie można zrobić z kolumnami w klauzuli ORDER BY, polega na określeniu różnego sposobu sortowania dla każdej z nich. W poprzednim przykładzie rezultaty można posortować malejąco według kolumny z nazwiskiem i rosnąco według kolumny z imieniem. Tak wyglądałaby instrukcja SELECT po wprowadzeniu odpowiednich modyfikacji.

SQL
```
SELECT NazwiskoPracownika, ImiePracownika, TelefonPracownika,
IDPracownika
FROM Pracownicy
ORDER BY NazwiskoPracownika DESC, ImiePracownika ASC
```

Choć użycie słowa kluczowego ASC nie jest obowiązkowe, działanie instrukcji będzie bardziej oczywiste, jeśli je uwzględnisz.

Poprzedni przykład może skłaniać do postawienia interesującego pytania: czy kolejność kolumn w klauzuli ORDER BY jest istotna? Odpowiedź brzmi „Tak!". Kolejność ta jest istotna, ponieważ system baz danych przetwarza kolumny w klauzuli ORDER BY od strony lewej do prawej. Znaczenie właściwego uporządkowania kolumn rośnie wprost proporcjonalnie do ich liczby. Kolumny w klauzuli ORDER BY zawsze należy wymieniać w odpowiedniej kolejności, by rezultaty zostały prawidłowo posortowane.

Uwaga. Systemy baz danych firmy Microsoft (Microsoft Office, Access i Microsoft SQL Server) zawierają interesujące rozszerzenie, umożliwiające zażądanie tylko pewnego podzbioru wierszy uporządkowanych przez klauzulę ORDER BY. Można to zrobić za pomocą słowa kluczowego TOP w klauzuli SELECT. W ten sposób da się na przykład wyszukać pięć najdroższych produktów w bazie danych Zamówienia:

```
SELECT TOP 5 NazwaProduktu, CenaDetaliczna
   FROM Produkty
   ORDER BY CenaDetaliczna DESC
```

Baza danych posortuje wszystkie wiersze z tabeli Produkty malejąco względem ceny, a następnie zwróci tylko pięć pierwszych wierszy. Obydwa wymienione systemy baz danych umożliwiają ponadto określenie liczby zwracanych w ten sposób wierszy w postaci procentu całego zbioru wierszy. Za pomocą poniższego zapytania można na przykład wyświetlić 10% najdroższych produktów:

```
SELECT TOP 10 PERCENT NazwaProduktu, CenaDetaliczna
   FROM Produkty
   ORDER BY CenaDetaliczna DESC
```

Co więcej, jeśli w systemie SQL Server chcesz utworzyć widok z użyciem klauzuli ORDER BY, to musisz użyć słowa kluczowego TOP. Jeżeli chciałbyś, aby widok zawierał wszystkie wiersze, powinieneś zawrzeć w zapytaniu słowo kluczowe TOP 100 PERCENT. Z tego względu wszystkie widoki w przykładach dla systemu SQL Server, skonstruowane z użyciem klauzuli ORDER BY, zawierają też warunek TOP 100 PERCENT. Zapis ten nie jest wymagany w programie Microsoft Access.

Zapisywanie pracy

Zapisuj instrukcje SELECT — każdy popularny program do obsługi baz danych umożliwia ich zapisywanie! Zapisywanie instrukcji pozwala uniknąć ponownego pisania ich za każdym razem, gdy chcesz skierować do bazy danych to samo pytanie. Podczas zapisywania instrukcji SELECT nadaj jej logiczną nazwę, która ułatwi Ci zapamiętanie, jakiego rodzaju informacje ona zwraca. A jeśli Twój system baz danych na to pozwala, stwórz zwięzły opis celu istnienia danej instrukcji. Taki opis docenisz zwłaszcza po dłuższym czasie nieużywania danej instrukcji SELECT, gdy będziesz chciał sobie przypomnieć, po co ją napisałeś.

Zapisane zapytanie SELECT w niektórych programach do obsługi baz danych jest nazywane po prostu zapytaniem, w innych zaś nosi nazwę widoku, funkcji albo procedury składowanej. Niezależnie od nazwy każdy program do obsługi baz danych umożliwia wykonywanie („uruchamianie") zapisanych instrukcji i pracę z otrzymanym zbiorem rezultatów.

Uwaga. W dalszej części książki będziemy nazywać zapisaną instrukcję SELECT *zapytaniem*, a operację uruchamiania go — *wykonywaniem*.

Zasadniczo można wyróżnić dwie metody wykonywania zapytań. Pierwsza polega na zastosowaniu jakiegoś rodzaju interaktywnego narzędzia (na przykład polecenia na pasku narzędzi albo szablonu zapytania), zaś druga na uruchomieniu pewnego fragmentu kodu. Najczęściej będziesz używał pierwszej metody. Nad drugą nie musisz się szczególnie zastanawiać, dopóki nie zaczniesz używać wewnętrznego języka programowania Twojego systemu do obsługi baz danych. Naszym zadaniem jest nauczenie Cię tworzenia instrukcji SQL i posługiwania się nimi, zaś Twoim — opanowanie tworzenia, zapisywania i wykonywania ich w używanym przez Ciebie systemie baz danych.

Przykładowe instrukcje

Po przedstawieniu podstawowych właściwości instrukcji SELECT i zapytania SELECT przyjrzyjmy się wybranym przykładom zastosowania tych operacji w różnych sytuacjach. Zostały one opracowane na podstawie wszystkich przykładowych baz danych i ilustrują użycie instrukcji SELECT, zapytania SELECT oraz dwóch opisanych wcześniej metod ułatwiających wybieranie kolumn przy tłumaczeniu instrukcji. W ramach każdego przykładu pod tabelą ze składnią SQL zamieściliśmy zbiory rezultatów, które powinny zostać zwrócone przez daną operację. Tytuły podane tuż nad zbiorem rezultatów pełnią dwojaką funkcję: identyfikują ten zbiór i są zarazem nazwą, jaką nadaliśmy instrukcji SQL w danym przykładzie.

Zastanawiasz się, po co nazywaliśmy poszczególne instrukcje SQL? Otóż dlatego, że je zapisaliśmy! Na tej samej zasadzie co instrukcje w poniższych przykładach nazwaliśmy i zapisaliśmy wszystkie instrukcje SQL, które pojawiają się w dalszej części książki. Każda z nich jest umieszczona w odpowiedniej bazie danych (zgodnie z opisem przykładu). Nazwy zapytań właściwych dla tego rozdziału poprzedziliśmy przedrostkiem „R04". Jeśli chcesz pobrać przykłady na swój komputer, postępuj zgodnie ze wskazówkami podanymi we „Wstępie". W ten sposób będziesz mógł zobaczyć opisane instrukcje w działaniu, zanim zaczniesz pisać je samodzielnie.

Uwaga. Małe przypomnienie: wszystkie nazwy kolumn i tabel użyte w poniższych przykładach zostały zaczerpnięte z baz danych przedstawionych w dodatku B. Należy ponadto pamiętać, że w przypadku

dowolnych zapytań bez klauzuli `ORDER BY` kolejność zwróconych wierszy jest nieokreślona. W większości sytuacji kolejność wierszy zwróconych przez tego rodzaju zapytanie w dowolnym spośród przykładowych systemów baz danych (Microsoft SQL Server, Microsoft Office Access albo MySQL) powinna być zgodna z kolejnością wierszy podaną w książce. Ale jeśli wypróbujesz przykładowe skrypty SQL w innym systemie baz danych, to o ile liczba zwróconych wierszy oraz zawarte w nich informacje powinny być identyczne, ich kolejność może być inna.

Baza danych Zamówienia

Podaj mi nazwy wszystkich naszych dostawców

Tłumaczenie	Wybierz nazwę dostawcy z tabeli dostawcy
Porządkowanie	Wybierz nazwę dostawcy z ~~tabeli~~ dostawcy
SQL	`SELECT NazwaDostawcy` `FROM Dostawcy`

R04_nazwy_dostawcow (10 wierszy)

NazwaDostawcy
Shinoman, Incorporated
Viscount
Nikoma of America
ProFormance
Kona, Incorporated
Big Sky Mountain Bikes
Dog Ear
Sun Sports Suppliers
Lone Star Bike Supply
Armadillo Brand

Jakie są nazwy i ceny wszystkich produktów, jakie sprzedajemy?

Tłumaczenie	Wybierz nazwę produktu oraz cenę z tabeli produkty
Porządkowanie	Wybierz nazwę produktu, ~~oraz~~ cenę z ~~tabeli~~ produkty
SQL	`SELECT NazwaProduktu, CenaDetaliczna` `FROM Produkty`

R04_lista_cen_produktow (40 wierszy)

NazwaProduktu	CenaDetaliczna
Trek 9000 Mountain Bike	1 200,00 zł
Eagle FS-3 Mountain Bike	1 800,00 zł
Dog Ear Cyclecomputer	75,00 zł
Victoria Pro All Weather Tires	54,95 zł
Dog Ear Helmet Mount Mirrors	7,45 zł
Viscount Mountain Bike	635,00 zł
Viscount C-500 Wireless Bike Computer	49,00 zł
Kryptonite Advanced 2000 U-Lock	50,00 zł
Nikoma Lok-Tight U-Lock	33,00 zł
Viscount Microshell Helmet	36,00 zł
<< kolejne wiersze >>	

Z jakich stanów pochodzą nasi klienci?

Tłumaczenie Wybierz jednokrotnie stany z tabeli klienci

Porządkowanie Wybierz jednokrotnie stany z ~~tabeli~~ klienci

SQL
```
SELECT DISTINCT StanZamKlienta
FROM Klienci
```

R04_stan_zamieszkania_klientow (4 wiersze)

StanZamKlienta
CA
OR
TX
WA

Baza danych Agencja artystyczna

Podaj wszystkich wykonawców oraz miasta, w których znajduje się ich siedziba, i posortuj rezultaty rosnąco według miasta i nazwy wykonawcy.

Tłumaczenie Wybierz miasto oraz nazwę wykonawcy z tabeli wykonawcy i uporządkuj według miasta i nazwy wykonawcy

Porządkowanie Wybierz miasto, ~~oraz~~ nazwę wykonawcy z ~~tabeli~~ wykonawcy ~~i~~ uporządkuj według miasta, ~~i~~ nazwy wykonawcy

SQL
```
SELECT MiastoWykonawcy, NazwaScenicznaWykonawcy
FROM Wykonawcy
ORDER BY MiastoWykonawcy ASC, NazwaScenicznaWykonawcy ASC
```

R04_siedziby_wykonawcow (13 wierszy)

MiastoWykonawcy	NazwaSceniczna Wykonawcy
Auburn	Caroline Coie Cuartet
Auburn	Topazz
Bellevue	Jazz Persuasion
Bellevue	Jim Glynn
Bellevue	Susan McLain
Redmond	Carol Peacock Trio
Redmond	JV & the Deep Six
Seattle	Coldwater Cattle Company
Seattle	Country Feeling
Seattle	Julia Schnebly
<< kolejne wiersze >>	

Podaj mi daty występów, ale tak, żeby się nie powtarzały; liczba występów odbywających się jednego dnia jest dla mnie nieistotna.

Tłumaczenie Wybierz jednokrotnie dni rozpoczęcia występów z tabeli występy

Porządkowanie Wybierz jednokrotnie dni rozpoczęcia występów z ~~tabeli~~ występy

SQL
```
SELECT DISTINCT DzienRozpoczecia
FROM Wystepy
```

R04_daty_imprez (64 wiersze)

DzienRozpoczecia
2012-09-01
2012-09-10
2012-09-11
2012-09-15
2012-09-17
2012-09-18
2012-09-24
2012-09-29
2012-09-30
2012-10-01
<< kolejne wiersze >>

Baza danych Grafik uczelni

Czy możemy otrzymać kompletną informację o zajęciach?

Tłumaczenie Wybierz wszystkie kolumny z tabeli zajęcia

Porządkowanie Wybierz ~~wszystkie kolumny~~ * z ~~tabeli~~ zajęcia

SQL
```
SELECT *
FROM Zajecia
```

R04_informacja_o_zajeciach (132 wiersze)

IDZajec	IDPrzedmiotu	IDSali	Punkty-Edukacyjne	Data-Rozpoczecia	Godzina-Rozpoczecia	Czas-Trwania	<< inne kolumny >>
1000	11	1231	5	2013-9-10	10:00	50	...
1002	12	1619	4	2013-9-9	15:30	110	...
1004	13	1627	4	2013-9-9	08:00	50	...
1006	13	1627	4	2013-9-9	09:00	110	...
1012	14	1627	4	2013-9-9	13:00	170	...
1020	15	3404	4	2013-9-9	13:00	110	...
1030	16	1231	5	2013-9-9	11:00	50	...
1031	16	1231	5	2013-9-9	14:00	50	...
1156	37	3443	5	2013-9-9	16:00	50	...
1162	37	3443	5	2013-9-9	09:00	80	...
<< kolejne wiersze >>							

Podaj mi listę budynków kampusu oraz liczbę pięter w każdym budynku. Posortuj listę według budynków, rosnąco.

Tłumaczenie Wybierz nazwę budynku oraz liczbę kondygnacji z tabeli budynki i uporządkuj je według nazwy budynku

Porządkowanie Wybierz nazwę budynku, ~~oraz~~ liczbę kondygnacji z ~~tabeli~~ budynki ~~i~~ uporządkuj ~~je~~ według nazwy budynku

SQL
```
SELECT NazwaBudynku, LiczbaKondygnacji
FROM Budynki
ORDER BY NazwaBudynku ASC
```

R04_lista_budynkow (6 wierszy)

NazwaBudynku	LiczbaKondygnacji
Biblioteka	2
Budynek główny	3

NazwaBudynku	LiczbaKondygnacji
Centrum dydaktyczne	3
Inżynieria	2
Laboratoria	3
Kompleks sportowy	1

Baza danych Liga kręglarska

Gdzie odbywają się nasze turnieje?

Tłumaczenie Wybierz jednokrotnie nazwy lokalizacji turniejów z tabeli turnieje

Porządkowanie Wybierz jednokrotnie ~~nazwy~~ lokalizacje turniejów z ~~tabeli~~ turnieje

SQL
```
SELECT DISTINCT LokalizacjaTurnieju
FROM Turnieje
```

R04_lokalizacje_turniejow (7 wierszy)

LokalizacjaTurnieju
Acapulco Lanes
Bolero Lanes
Imperial Lanes
Red Rooster Lanes
Sports World Lanes
Thunderbird Lanes
Totem Lanes

Zrób listę wszystkich dat i lokalizacji turniejów. Chciałbym, żeby daty były uporządkowane malejąco, a miejsca w kolejności alfabetycznej.

Tłumaczenie Wybierz daty turniejów oraz lokalizacje turniejów z tabeli turnieje i uporządkuj je według daty malejąco i według lokalizacji rosnąco

Porządkowanie Wybierz daty turniejów, ~~oraz~~ lokalizacje turniejów z ~~tabeli~~ turnieje ~~i~~ uporządkuj je według daty malejąco, ~~i według~~ lokalizacji rosnąco

SQL
```
SELECT DataTurnieju, LokalizacjaTurnieju
FROM Turnieje
ORDER BY DataTurnieju DESC, LokalizacjaTurnieju ASC
```

R04_daty_i_miejsca_turniejow (20 wierszy)

DataTurnieju	LokalizacjaTurnieju
2013-08-16	Totem Lanes

DataTurnieju	LokalizacjaTurnieju
2013-08-09	Imperial Lanes
2013-08-02	Sports World Lanes
2013-07-26	Bolero Lanes
2013-07-19	Thunderbird Lanes
2013-07-12	Red Rooster Lanes
2012-12-04	Acapulco Lanes
2012-11-27	Totem Lanes
2012-11-20	Sports World Lanes
2012-11-13	Imperial Lanes
<< kolejne wiersze >>	

Baza danych Przepisy

Jakie mamy kategorie przepisów i jak nazywają się przepisy znajdujące się w każdej z kategorii? Czy można posortować te informacje według kategorii i nazwy przepisu?

Tłumaczenie — Wybierz identyfikatory kategorii przepisów oraz nazwy przepisów z tabeli przepisy i uporządkuj je według identyfikatora kategorii i nazwy przepisu

Porządkowanie — Wybierz ~~identyfikatory~~ ID kategorii przepisów, ~~oraz~~ nazwy przepisów z ~~tabeli~~ przepisy ~~i~~ uporządkuj je według ~~identyfikatora~~ ID kategorii, ~~i~~ nazwy przepisu

SQL

```
SELECT IDKategoriiPrzepisu, NazwaPrzepisu
FROM Przepisy
ORDER BY IDKategoriiPrzepisu ASC, NazwaPrzepisu ASC
```

R04_kategorie_i_nazwy_przepisow (15 wierszy)

IDKategoriiPrzepisu	NazwaPrzepisu
1	Filety z łososia w pergaminie
1	Gulasz irlandzki
1	Makaron fettuccine Alfredo
1	Pollo z kurczakiem
1	Ryba à la Veracruz
1	Stek wołowy
1	Tourtière (francusko-kanadyjski placek z wieprzowiną)
2	Fasolka z czosnkiem
2	Szparagi

IDKategoriiPrzepisu	NazwaPrzepisu
3	Pudding Yorkshire
<< kolejne wiersze >>	

Wyświetl listę unikatowych identyfikatorów kategorii przepisów z tabeli przepisy.

Tłumaczenie Wybierz jednokrotnie identyfikatory kategorii przepisów z tabeli przepisy

Porządkowanie Wybierz jednokrotnie ~~identyfikatory~~ ID kategorii przepisów z ~~tabeli~~ przepisy

SQL
```
SELECT DISTINCT IDKategoriiPrzepisu
FROM Przepisy
```

R04_identyfikatory_kategorii_przepisow (6 wierszy)

IDKategoriiPrzepisu
1
2
3
4
5
6

Podsumowanie

W tym rozdziale opisaliśmy operację SELECT. Dowiedziałeś się, że jest to jedna z czterech operacji w SQL umożliwiających przetwarzanie danych. (Pozostałe to UPDATE, INSERT oraz DELETE, omówione w części V). Napisaliśmy też, że operacja SELECT może zostać podzielona na trzy mniejsze: instrukcję SELECT, wyrażenie SELECT i zapytanie SELECT.

Następnie skupiliśmy się na instrukcji SELECT i przedstawiliśmy klauzule, które wchodzą w jej skład. Zaznaczyliśmy, że klauzule SELECT oraz FROM są podstawowymi klauzulami umożliwiającymi pozyskiwanie informacji z bazy danych, zaś pozostałe — WHERE, GROUP BY i HAVING — służą do warunkowego przetwarzania i filtrowania informacji zwróconych przez klauzulę SELECT.

Przeprowadziliśmy też krótką analizę poświęconą różnicy między danymi a informacjami. Dowiedziałeś się, że wartości przechowywane w bazie to dane, zaś informacje są danymi, które zostały przetworzone w sposób nadający im szczególne znaczenie z perspektywy oglądającej je osoby. Wiesz już też, że wiersze informacji zwracane przez instrukcję SELECT noszą nazwę zbioru rezultatów.

Pozyskiwanie rezultatów było tematem kolejnych rozważań, które zaczęliśmy od przedstawienia podstawowej formy instrukcji SELECT. Dowiedziałeś się, jak skonstruować

prawidłową instrukcję SELECT przy użyciu trzyetapowej metody, kończącej się przekształceniem źródłowego żądania na instrukcję zgodną ze składnią SQL. Wiesz już, że w klauzuli SELECT możesz użyć dwóch lub większej liczby kolumn, aby rozszerzyć zakres informacji pozyskiwanych z bazy danych. Tę część rozdziału zakończyliśmy krótkim omówieniem słowa kluczowego DISTINCT, które umożliwia wyeliminowanie powtarzających się wierszy ze zbioru rezultatów.

Następnie przyjrzeliśmy się zapytaniu SELECT oraz temu, jak można je połączyć z instrukcją SELECT w celu posortowania zbioru rezultatów działania tej instrukcji. Dowiedziałeś się, że jest to konieczne, ponieważ zapytanie SELECT jest jedyną spośród operacji SELECT, która może zawierać klauzulę ORDER BY. Następnie wyjaśniliśmy, że klauzula ORDER BY służy do sortowania informacji na podstawie wybranej kolumny lub większej ich liczby, i że każda z tych kolumn może posłużyć do sortowania w sposób rosnący lub malejący. Przy okazji napomknęliśmy o zapisywaniu instrukcji SELECT i o tym, że instrukcje tego rodzaju można przechowywać w postaci zapytania lub widoku, by ułatwić sobie ich ponowne użycie.

Na koniec podaliśmy kilka przykładów na podstawie różnych tabel z przygotowanych przez nas baz danych. Przykłady te pokazują, w jaki sposób różne koncepcje i techniki zaprezentowane w tym rozdziale są wykorzystywane w praktycznych, typowych sytuacjach i zastosowaniach. W następnym rozdziale przyjrzymy się klauzuli SELECT nieco bliżej i pokażemy, w jaki sposób pozyskać z bazy coś więcej niż zwykłe informacje zaczerpnięte z listy kolumn.

Tymczasem zapraszamy do zapoznania się z kilkoma zadaniami do samodzielnego rozwiązania, zamieszczonymi w dalszej części rozdziału.

Zagadnienia do samodzielnego rozwiązania

Poniżej znajduje się lista żądań wraz z nazwami odpowiadających im zapytań do przykładowych baz danych. Jeśli chcesz trochę poćwiczyć, spróbuj opracować instrukcję SQL dla każdego z żądań, a potem porównaj swoją propozycję z zapytaniem zapisanym w przykładach. Nie przejmuj się, jeśli składnia Twojej instrukcji nie jest stuprocentowo zgodna ze składnią zapisanego zapytania, jeśli tylko otrzymany zbiór rezultatów jest w obu przypadkach taki sam.

Baza danych Zamówienia

1. *Podaj mi wszystkie informacje na temat naszych pracowników.*

   Rozwiązanie znajdziesz w zapytaniu R04_informacje_o_pracownikach (8 wierszy).

2. *Wymień w kolejności alfabetycznej wszystkie miasta, w których znajdują się nasi dostawcy, i dołącz nazwy dostawców, z którymi w danym mieście współpracujemy.*

   Rozwiązanie znajdziesz w zapytaniu R04_siedziby_dostawcow (10 wierszy).

Baza danych Agencja rozrywkowa

1. *Daj mi listę wszystkich naszych agentów wraz z ich numerami telefonów. Lista powinna być uporządkowana alfabetycznie względem nazwisk i imion.*

   Rozwiązanie znajdziesz w zapytaniu R04_lista_telefonow_agentow (9 wierszy).

2. *Podaj mi informacje o wszystkich organizowanych przez nas imprezach.*

   Rozwiązanie znajdziesz w zapytaniu R04_informacje_o_imprezach (111 wierszy).

3. *Potrzebuję listy wszystkich występów wraz z datami ich rozpoczęcia. Posortuj rekordy według daty w porządku malejącym i według występu w porządku rosnącym.*

   Rozwiązanie znajdziesz w zapytaniu R04_zaplanowane_imprezy (111 wierszy).

Baza danych Grafik uczelni

1. *Pokaż mi kompletną listę wykładanych przedmiotów.*

   Rozwiązanie znajdziesz w zapytaniu R04_lista_przedmiotow (56 wierszy).

2. *Jakie tytuły naukowe mają nasi wykładowcy?*

   Rozwiązanie znajdziesz w zapytaniu R04_tytuly_naukowe (3 wiersze).

3. *Podaj imiona, nazwiska i numery telefonów wszystkich naszych pracowników i posortuj je według nazwiska i imienia.*

   Rozwiązanie znajdziesz w zapytaniu R04_lista_pracownikow_z_telefonami (27 wierszy).

Baza danych Liga kręglarska

1. *Zrób zestawienie wszystkich zespołów w kolejności alfabetycznej.*

   Rozwiązanie znajdziesz w zapytaniu R04_lista_zespolow (10 wierszy).

2. *Przedstaw mi listę wyników gier każdego z naszych zawodników, bez imion i nazwisk.*

   Rozwiązanie znajdziesz w zapytaniu R04_zestawienie_wynikow_kreglarzy (1344 wiersze).

3. *Potrzebuję listy zawodników wraz z ich adresami posortowanych w kolejności alfabetycznej.*

   Rozwiązanie znajdziesz w zapytaniu R04_imiona_nazwiska_i_adresy_kreglarzy (32 wiersze).

Baza danych Przepisy

1. *Podaj mi listę wszystkich składników, które aktualnie mamy w bazie.*

   Rozwiązanie znajdziesz w zapytaniu R04_lista_wszystkich_skladnikow (79 wierszy).

2. *Podaj mi wszystkie najważniejsze informacje na temat posiadanych przepisów i posortuj je alfabetycznie według nazwy przepisu.*

   Rozwiązanie znajdziesz w zapytaniu R04_glowne_informacje_o_przepisach (15 wierszy).

Skorowidz

E

F

G

H

I

J

K

R

S

T

U

W

X

Z

Ż

Bazy danych pomagają organizacjom w zarządzaniu i planowaniu działań. Problem w tym, że mało kto potrafi efektywnie używać relacyjnych baz danych, szybko wyszukiwać w nich informacje, porównywać je i filtrować. Jeśli masz już dość mozolnego przedzierania się przez tysiące kolumn w celu sprawdzenia jednej informacji albo porównywania zamówień klientów „na piechotę", koniecznie zapoznaj się z tą książką. To najprzyjemniejszy, najbardziej elastyczny i najprostszy poradnik uczący pisania skutecznych zapytań dla niemal dowolnej nowoczesnej bazy danych.

Krok po kroku przeprowadzi Cię on przez cały proces komunikacji z bazą: dowiesz się, jak wygląda jej optymalna struktura i jak działa SQL. Zobaczysz, jak zbudować proste zapytanie i co trzeba zrobić, by wzbogacić je o wiele kryteriów wyszukiwania. Co więcej, sprawdzisz, do czego warto wykorzystać podzapytania. Nauczysz się filtrować dane w jednej tabeli i porównywać informacje z wielu tabel, a także przeprowadzać operacje na zbiorach danych — to szalenie przydatna funkcja, której użycie pozwoli Ci przejść na wyższy poziom wtajemniczenia. Kolejnym etapem będzie stosowanie złączeń, wykonywanie operacji warunkowych oraz rozwiązywanie problemów. Opanuj do perfekcji korzystanie z relacyjnych baz danych!

- Relacyjne bazy danych i SQL
- Tworzenie prostego zapytania
- Rodzaje wyrażeń i filtrowanie danych
- Myślenie zbiorami
- Złączenia INNER JOIN oraz OUTER JOIN
- Operacja UNION i operacje warunkowe
- Podzapytania i proste zestawienia
- Grupowanie danych i filtrowanie zgrupowanych danych
- Wstawianie i usuwanie zbiorów danych

John L. Viescas jest niezależnym konsultantem baz danych, specjalizującym się w zarządzaniu systemami Microsoft Office Access i Microsoft SQL Server w małych i dużych przedsiębiorstwach. Jest autorem książek: *A Programmers Quick Reference Guide to SQL, Running Microsoft Access, Microsoft Office Access Inside Out* oraz *Building Microsoft Access Applications.*

Michael J. Hernandez jest niezależnym konsultantem i programistą baz danych. Pracował jako szkoleniowiec dla rządu i armii USA oraz sektora prywatnego. Był menedżerem programowym grupy Microsoft Visual Studio .NET. Jest autorem książki *Projektowanie baz danych dla każdego.*

Addison-Wesley

Helion

37665 numer katalogowy

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ

KOD KORZYŚCI

ISBN 978-83-283-1364-4

9 788328 313644

cena: 99,00 zł