JĘZYK SQL

PRZYJAZNY PODRĘCZNIK

WYDANIE II

LARRY ROCKOFF

Tytuł oryginału: The Language of SQL, Second Edition

Tłumaczenie: Beata Błaszczyk

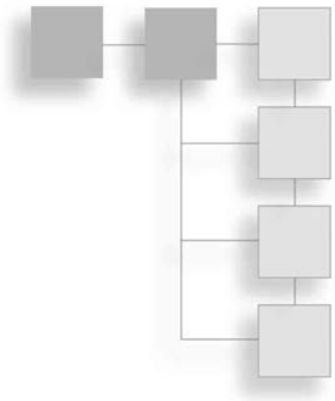
ISBN: 978-83-283-3190-7

Authorized translation from the English language edition, entitled: THE LANGUAGE OF SQL, Second Edition; ISBN 0134658256; Larry Rockoff; published by Pearson Education, Inc, publishing as Addison Wesley Professional.

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

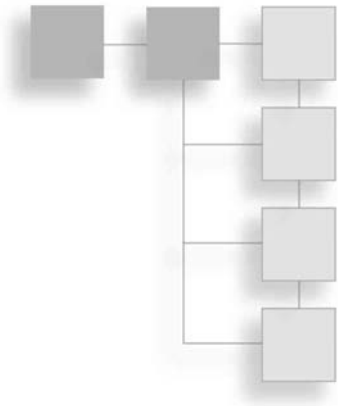
Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/jsqlp2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.



Spis treści

ROZDZIAŁ 1



Relacyjne bazy danych i SQL

Jak wspomniano we wprowadzeniu, SQL jest najpowszechniej stosowanym narzędziem umożliwiającym interakcję z danymi znajdującymi się w relacyjnych bazach danych. Aby jednak do niej doszło, konieczne jest uwzględnienie zarówno aspektów natury językowej, jak i elementów logiki. Z perspektywy językowej SQL wykorzystuje unikalną składnię z wieloma angielskimi słowami, takimi jak `WHERE`, `FROM` czy `HAVING`. Natomiast jeśli chodzi o wymiar logiczny, umożliwia określenie warunków wyszukiwania danych znajdujących się w relacyjnej bazie danych bądź sposobu ich aktualizacji.

Ta książka adresuje oba powyższe aspekty, a odzwierciedlenie tej dwoistości znajdziesz, drogi Czytelniku, w objaśnieniach każdego elementu języka SQL. Jak wiadomo, we wszystkich językach, bez względu na to, czy chodzi o te mówione, czy te służące do programowania, występują słowa, które trzeba poznać i zapamiętać. Dlatego też w tej książce omawiane będą przy zachowaniu logicznego porządku poszczególne słowa kluczowe występujące w języku SQL. Wraz z każdym kolejnym rozdziałem do poznanych już przez Ciebie pojęć dochodzić będą kolejne terminy, dając Ci coraz większe możliwości interakcji z bazą danych.

Jak wiadomo, oprócz słów ważna jest również logika, z jaką są one wypowiadane lub zapisywane. Nie inaczej jest w przypadku języka SQL, w którym każde z nich ma określone znaczenie i służy do osiągnięcia konkretnego celu. Stosowanie zasad logiki w instrukcjach SQL jest wobec tego tak samo ważne jak korzystanie w nich z odpowiedniej terminologii. Jest to szczególnie istotne z uwagi na fakt, że tak jak i w przypadku innych języków programowania często pożądany efekt można osiągnąć na wiele różnych sposobów. Diabeł tkwi jednak w szczegółach, a w tym przypadku w odpowiednim doborze słownictwa przy jednoczesnym uwzględnieniu zasad logiki.

Przyjrzyjmy się wobec tego najpierw aspektom związanym z językiem.

Po zapoznaniu się ze składnią języka SQL nie oprzesz się wrażeniu, że jego polecenia są analogiczne do zdań w języku angielskim i również mają pewne ekspresyjne znaczenie.

Porównaj na przykład poniższe zdanie:

```
Poproszę na wynos hamburgera i frytki z promocyjnego menu.
```

z instrukcją SQL:

```
Select miasto, województwo[1]
from klienci
order by województwo
```

Nie wchodząc na razie w szczegóły, powyższa instrukcja SQL oznacza, że chcemy wyświetlić pola z informacjami o mieście i województwie z tabeli zawierającej dane o klientach, znajdującej się w naszej bazie danych. Ponadto chcemy posortować wyniki alfabetycznie według województwa.

W obu przypadkach określamy interesujące nas elementy (hamburger/frytki lub miasto/województwo), wskazujemy, skąd chcemy je pozyskać (promocyjne menu lub tabela z informacjami o klientach), oraz zamieszczamy dodatkowe instrukcje (przygotowanie zamówienia na wynos lub sortowanie wyników według województwa).

Zanim jednak przejdziemy dalej, zatrzymajmy się na chwilę i zastanówmy nad jedną, drobną kwestią, mianowicie jak właściwie należy wymawiać słowo SQL? Tak naprawdę są dwie możliwości. Jedną z opcji jest wymówienie po prostu pojedynczych liter, czyli powiedzenie „S-Q-L”. Inna możliwość to wymówienie tego jako pojedynczego słowa „siquel”. Ta wersja składa się tylko z dwóch sylab i o wiele łatwiej ją wymówić, niemniej kwestia tego, która z podanych powyżej opcji jest poprawna, nie jest raczej przedmiotem sporu. Jest to w zasadzie związane z osobistymi preferencjami.

Jeżeli chodzi o znaczenie liter S-Q-L, większość zgadza się, że jest to „strukturalny język zapytań”. Są jednak również tacy, którzy twierdzą, że skrótu SQL nie można rozwinąć, gdyż język ten wywodzi się ze stworzonego przez IBM starego języka o nazwie sequel, która *nie* oznaczała strukturalnego języka zapytań. W tej kwestii zatem również brak jednomyślności.

[1] W przykładach zawartych w niniejszej książce zarówno w nazwach tabel, jak i kolumn zastosowano polskie znaki diakrytyczne. Nie jest to jednak zalecane w przypadku realizacji rzeczywistych projektów, m.in. ze względu na możliwe problemy z kodowaniem tego rodzaju znaków w realnie istniejących bazach danych — *przyp. tłum.*

Definicja SQL

Czym więc jest SQL? W skrócie, SQL jest standardowym językiem programowania wykorzystywanym w celu utrzymywania danych zawartych w relacyjnych bazach danych i korzystania z nich. Mówiąc prościej, SQL to język, który pozwala użytkownikom na interakcję z relacyjnymi bazami danych. Począwszy od 1970 roku, przez wiele lat był rozwijany przez różne organizacje. W 1986 roku *Amerykański Instytut Normalizacyjny* (ang. *ANSI — American National Standards Institute*) opublikował swój pierwszy zestaw norm dotyczących języka SQL i od tego czasu wielokrotnie je aktualizował.

Ogólnie rzecz biorąc, język SQL składa się z trzech głównych elementów. Pierwszy z nich nosi nazwę *DML* lub *języka manipulowania danymi* (ang. *Data Manipulation Language*). Obejmuje on zestaw instrukcji wykorzystywanych do pobierania, aktualizacji, dodawania i usuwania danych z bazy danych. Drugi element to *DDL* bądź *język definicji danych* (ang. *Data Definition Language).* Umożliwia on tworzenie lub modyfikowanie struktur bazy danych. Na przykład w ramach języka definicji danych występuje instrukcja `ALTER`, która pozwala modyfikować tabele w bazie danych. Wreszcie trzeci komponent języka SQL to *DCL* lub *język kontroli danych* (ang. *Data Control Language*), pozwalający zarządzać bezpieczeństwem dostępu do obiektów bazy danych.

Główni producenci oprogramowania, tacy jak Microsoft i Oracle, dostosowali standard SQL do własnych potrzeb i dodali do niego liczne rozszerzenia i modyfikacje. Jednak mimo że każdy dostawca w wyjątkowy sposób interpretuje SQL, podstawy tego języka pozostają niezmienione i wspólne dla wszystkich producentów oprogramowania. Ten właśnie zakres zostanie omówiony w tej książce.

Jako język programowania, SQL różni się od pozostałych języków, takich jak Visual Basic lub C++, które być może znasz. Inne języki mają zazwyczaj charakter proceduralny. Oznacza to, że umożliwiają określenie pewnych procedur w celu osiągnięcia pożądanych wyników. SQL jest raczej językiem deklaratywnym, w którym cel do osiągnięcia zazwyczaj deklarowany jest za pomocą pojedynczej instrukcji. W SQL możliwe jest zastosowanie prostszej struktury, ponieważ jest on wykorzystywany jedynie w kontekście relacyjnych baz danych, a nie szeroko rozumianych systemów komputerowych.

Jeszcze jedna kwestia wymaga w tym miejscu wyjaśnienia, a mianowicie fakt, że język SQL jest czasami utożsamiany z określonym rodzajem baz danych. Istnieje wiele firm komputerowych sprzedających oprogramowanie dla systemów zarządzania bazami danych (ang. *DBMS — Database Management Systems*). Powszechnie bazy danych w tego rodzaju pakietach oprogramowania są często określane jako *bazy danych SQL* (ang. *SQL databases*), jako że język SQL jest podstawowym narzędziem służącym do zarządzania samymi bazami danych i dostępem do danych w nich przechowywanych. W nazwie baz danych niektórych producentów występuje nawet słowo „SQL”. Na przykład firma Microsoft swój najnowszy system zarządzania bazą danych nazwała *SQL Server 2016*. Jednak w gruncie

rzeczy SQL jest raczej językiem. Nie jest to baza danych. Dlatego też kanwą niniejszej książki będzie scharakteryzowanie języka SQL, nie zaś konkretnej bazy danych.

Microsoft SQL Server, Oracle i MySQL

Choć celem tej książki — jak wspomniano powyżej — jest przedstawienie podstawowych elementów języka SQL, które są wykorzystywane podczas wszystkich wdrożeń systemów opartych na relacyjnej bazie danych, to zaprezentowane zostaną także konkretne przykłady składni SQL. Mając jednak na uwadze fakt, że składnia SQL ustanawiana przez poszczególnych dostawców nieco się od siebie różni, zdecydowaliśmy się skupić na składni tego języka wykorzystywanej przez następujące trzy bazy danych:

- Microsoft SQL Server,
- Oracle Database,
- MySQL.

W większości przypadków składnia poleceń SQL stosowanych w powyższych bazach danych niczym się nie różni, jednak zdarzają się odstępstwa od tej reguły. Gdy takie różnice będą miały miejsce, w tekście książki przedstawiona zostanie składnia stosowana w Microsoft SQL Server. Wszelkie różnice w składni występujące w przypadku MySQL oraz Oracle zostaną oznaczone taką ramką i zatytułowane „Różnice w ramach innych baz danych", tak jak poniżej:

RÓŻNICE W RAMACH INNYCH BAZ DANYCH

> Tego typu ramka pojawi się zawsze , gdy prezentowane będą różnice w składni w przypadku bazy danych Oracle lub MySQL. Składnia dla Microsoft SQL Server zostanie umieszczona w tekście głównym.

Microsoft SQL Server jest dostępny w kilku wersjach i edycjach. Najnowsza wersja to *Microsoft SQL Server 2016*. Producent oferuje edycje od podstawowej, o nazwie Express, do edycji Enterprise, zawierającej pełny zakres funkcjonalności. Wersja Express jest darmowa, ale zawiera mnóstwo funkcji, które pozwalają rozpocząć przygodę z budowaniem bazy danych. Wersja Enterprise ma wiele wyrafinowanych funkcji służących do zarządzania bazą danych, a także zaawansowane komponenty do analiz business intelligence.

Jeśli chodzi o bazę danych MySQL, to mimo że obecnie jest ona własnością firmy Oracle, pozostaje bazą danych typu open source. Oznacza to, że nie jest własnością żadnej organizacji. MySQL jest dostępna na wielu platformach innych niż Windows, takich jak Mac OS X i Linux. Community Edition to baza danych MySQL, którą można pobrać za darmo. Najnowsza wersja tej bazy danych to MySQL 5.7.

Ostatnia baza danych na naszej liście to Oracle. Jest ona dostępna w wielu wersjach. Najnowsza wersja to *Oracle Database 12c*. Bezpłatna edycja tej bazy danych nosi nazwę Express.

Zaczynając pracę z bazami danych, czasami warto wcześniej pobrać wybraną bazę danych, aby móc poćwiczyć budowanie instrukcji SQL z wykorzystaniem znajdujących się w niej tabel. Jednak w celu przyswojenia wiedzy zawartej w niniejszej książce nie musisz tego robić. Została ona napisana w taki sposób, abyś nauczył się posługiwania językiem SQL w trakcie jej czytania. W tekście zamieszczę ponadto wystarczającą ilość danych, które umożliwią Ci zrozumienie wyników różnych instrukcji SQL bez konieczności pobierania oprogramowania i samodzielnego wpisywania i wykonywania prezentowanych instrukcji.

Niemniej jednak, jeśli chciałbyś pobrać darmowe wersje którejkolwiek z wymienionych baz danych, na końcu tej książki znajdziesz trzy dodatki zawierające kilka przydatnych wskazówek i porad, jak to zrobić. Dodatek A przedstawia kompletne informacje o tym, jak rozpocząć pracę z Microsoft SQL Server. Zawarta w nim instrukcja prezentuje szczegółowe informacje na temat instalowania oprogramowania i wykonywania poleceń SQL. Analogicznie, dodatek B dotyczy bazy danych MySQL, zaś dodatek C objaśnia sposób postępowania w przypadku bazy danych Oracle.

Jak wspomniano we wprowadzeniu, na uzupełniającej stronie internetowej zamieszczone zostały dodatkowe materiały w postaci zestawu wszystkich instrukcji SQL, jakie znalazły się w tej książce, z uwzględnieniem składni dedykowanej dla każdej z trzech wymienionych powyżej baz danych. Pobieranie ich oraz zapoznawanie się z zawartością tych plików może jednak wydać Ci się zbędne. Przykłady pokazane w książce są bowiem oczywiste i nie wymagają zaglądania do dodatkowych źródeł wiedzy w celu zrozumienia prezentowanego materiału. Jednak jeżeli masz taką potrzebę, zachęcamy Cię do skorzystania z wspominanych wyżej dodatkowych materiałów.

Warto również wspomnieć, że poza Microsoft SQL Server, MySQL i Oracle istnieje wiele innych powszechnie wykorzystywanych relacyjnych baz danych. Należą do nich między innymi:

- DB2 firmy IBM,
- Informix, również stworzony przez IBM,
- SQL Anywhere firmy Sybase,
- PostgreSQL, która jest bazą danych typu open source,
- Microsoft Access firmy Microsoft.

Z wymienionych powyżej baz danych dość wyjątkowy jest Microsoft Access, gdyż zawiera element graficzny. W istocie, Access jest graficznym interfejsem dla relacyjnych baz

danych. Innymi słowy, umożliwia on utworzenie kwerendy do relacyjnej bazy danych przy wykorzystaniu interfejsu graficznego. Użytecznym aspektem Accessa dla początkujących jest możliwość łatwego utworzenia kwerendy poprzez jej wizualizację, a następnie przełączenie się na widok SQL, aby zobaczyć właśnie utworzoną instrukcję SQL. Innym istotnym wyróżnikiem Accessa jest to, że baza ta jest instalowana lokalnie. Za pomocą tego narzędzia można zatem nie tylko utworzyć bazę danych i zapisać ją na komputerze jako pojedynczy plik, ale również podłączyć się do baz danych utworzonych za pomocą innych narzędzi, takich jak Microsoft SQL Server.

Relacyjne bazy danych

Po przedstawieniu wstępnych informacji przejdźmy teraz do omówienia podstaw relacyjnych baz danych oraz sposobu ich działania. Relacyjna baza danych jest zbiorem danych przechowywanych w dowolnej liczbie tabel. W powszechnym użyciu termin *relacyjne* (ang. *relational*) może być wykorzystywany w celu wskazania, że tabele są ze sobą w pewien sposób powiązane. Chcąc jednak być bardziej precyzyjnym, należy nadmienić, że określenie to odnosi się raczej do matematycznej teorii relacji i wskazuje na sposób logicznego powiązania ze sobą tabel, między którymi występują pewne relacje.

Weźmy pod uwagę prosty przykład bazy danych składającej się tylko z dwóch tabel: `Klienci` i `Zamówienia`. Tabela `Klienci` zawiera po jednym rekordzie dla każdego klienta, który kiedykolwiek złożył zamówienie. Tabela `Zamówienia` zawiera jeden rekord dla każdego złożonego zamówienia. Każda tabela może mieć dowolną liczbę pól, które są używane do przechowywania różnych atrybutów związanych z każdym rekordem. Na przykład tabela `Klient` może zawierać takie pola jak `ImięKlienta` czy `NazwiskoKlienta`.

W tym momencie przydatna może okazać się wizualizacja kilku tabel i zawartych w nich danych. Zwyczajowo tabele przedstawia się jako siatkę składającą się z wierszy i kolumn. Każdy wiersz oznacza rekord, natomiast każda kolumna reprezentuje pole w tabeli. Górny wiersz w nagłówku zazwyczaj zawiera nazwy pól. W pozostałych wierszach znajdują się właściwe dane.

W terminologii SQL rekordy i pola w rzeczywistości noszą nazwę *wierszy* (ang. *rows*) i *kolumn* (ang. *columns*), odpowiadając ich wizualnej reprezentacji. Odtąd zatem w celu opisania budowy tabel w relacyjnych bazach danych będziemy używali terminów *wiersze* i *kolumny* zamiast *rekordy* i *pola*.

Spójrzmy na przykład najprostszej możliwej relacyjnej bazy danych, w której znajdują się tylko dwie tabele, `Klienci` i `Zamówienia`. Oto jak mogłaby wyglądać tabela `Klienci`:

IDKlienta	ImięKlienta	NazwiskoKlienta
1	**Jan**	Kowalski
2	**Andrzej**	Nowak
3	**Anna**	Kwiatkowska

Tabela `Zamówienia` mogłaby mieć następującą postać:

IDZamówienia	IDKlienta	KwotaZamówienia
1	1	50,00
z	1	60,00
3	2	33,50
4	3	20,00

W powyższym przykładzie tabela `Klienci` zawiera trzy kolumny: `IDKlienta`, `ImięKlienta` i `NazwiskoKlienta`. Każdy z trzech wierszy tabeli prezentuje dane trzech różnych osób: Jana Kowalskiego, Andrzeja Nowaka i Anny Kwiatkowskiej. Każdy wiersz reprezentuje innego klienta, a każda kolumna zawiera inny fragment informacji o nim. Podobnie, w tabeli `Zamówienia` znajdują się trzy kolumny, ale cztery wiersze. Oznacza to, że w bazie danych istnieją cztery zamówienia i trzy przypisane do nich atrybuty.

Oczywiście, jest to bardzo prosty przykład, mający za zadanie pokazać, jaki typ danych może być przechowywany w rzeczywistej bazie danych. Na przykład tabela `Klienci` zazwyczaj zawiera wiele dodatkowych kolumn, opisujących inne atrybuty klienta, takie jak miasto, województwo, kod pocztowy i telefon. Podobnie tabela `Zamówienia` zazwyczaj zawiera kolumny opisujące dodatkowe atrybuty zamówienia, takie jak data zamówienia, wartość podatku obrotowego oraz informacje o sprzedawcy, który przyjął zamówienie.

Klucze główne i obce

Zwróć uwagę na pierwszą kolumnę w każdej tabeli: `IDKlienta` w tabeli `Klienci` i `IDZamó` ↪`wienia` w tabeli `Zamówienia`. Kolumny te są zwykle określane jako *klucze główne* (ang. *primary keys*). Klucze główne są przydatne, a wręcz niezbędne z dwóch powodów. Po pierwsze, pozwalają one jednoznacznie zidentyfikować pojedynczy wiersz w tabeli. Na przykład, gdybyś chciał pobrać wiersz dla Jana Kowalskiego, wystarczy po prostu użyć kolumny `IDKlienta`, aby wyświetlić takie dane. Klucze główne zapewniają również unikalność. Oznaczenie kolumny `IDKlienta` jako klucza głównego gwarantuje, że w tej kolumnie znajdować się będzie unikalna wartość dla każdego wiersza w tabeli. Nawet jeśli w bazie danych znajdą się dwie osoby o takim samym imieniu i nazwisku, na przykład Jan Kowalski, w obu tych wierszach w kolumnie `IDKlienta` będą występowały różne wartości.

W powyższym przykładzie wartości w kolumnach z kluczem głównym nie oznaczają niczego szczególnego. W tabeli `Klienci` kolumna `IDKlienta` zawiera wartości `1`, `2` i `3` dla trzech wierszy w tabeli. Często bowiem tabele w bazie danych są zaprojektowane w taki sposób, aby kolumny z kluczem głównym wraz z dodawaniem kolejnych wierszy wypełniane były automatycznie generowanymi numerami sekwencyjnymi. Ta cecha projektowa zwykle jest określana jako *automatyczny przyrost* (ang. *auto-increment*).

Drugim powodem zastosowania kluczy głównych jest to, że pozwalają w łatwy sposób połączyć jedną tabelę z inną. W naszym przykładzie kolumna `IDKlienta` w tabeli `Zamówienia` wskazuje na odpowiadający jej wiersz w tabeli `Klienci`. Patrząc na czwarty wiersz tabeli `Zamówienia`, można zauważyć, że w kolumnie `IDKlienta` występuje wartość `3`. Oznacza to, że to zamówienie odnosi się do klienta z `IDKlienta` o numerze `3`, czyli do Anny Kwiatkowskiej. Wykorzystanie wspólnych kolumn między tabelami jest istotnym elementem projektowania w relacyjnych bazach danych.

Oprócz jedynie wskazania tabeli `Klienci`, kolumna `IDKlienta` w tabeli `Zamówienia` może zostać oznaczona jako *klucz obcy* (ang. *foreign key*). Zagadnienie kluczy obcych zostanie szczegółowo omówione w rozdziale 18., zatytułowanym „Utrzymanie tabel”. Na razie po prostu zapamiętaj, że klucze obce mogą być zdefiniowane w celu zapewnienia, że kolumna ma poprawną wartość. Przykładem może być sytuacja, gdy nie chcesz, aby w kolumnie `IDKlienta` w tabeli `Zamówienia` znalazła się określona wartość, dla której nie istnieje odpowiednik w kolumnie `IDKlienta` w tabeli `Klienci`. Takie ograniczenie jest możliwe dzięki oznaczeniu kolumny jako klucza obcego.

Typy danych

Za pomocą kluczy głównych i obcych tworzona jest struktura tabel bazy danych. Dzięki nim tabele są ze sobą poprawnie powiązane, a także możliwy jest dostęp do wszystkich tabel w bazie danych. Inną ważną cechą każdej kolumny w tabeli jest typ przechowywanych w niej danych.

Typy danych są po prostu sposobem definiowania rodzaju danych zawartych w kolumnie. Typ danych trzeba określić dla każdej kolumny w każdej tabeli. Niestety, w ramach różnych relacyjnych baz danych dozwolone jest użycie zróżnicowanych typów danych, które mają określone znaczenie. Na przykład każda z relacyjnych baz danych — Microsoft SQL Server, MySQL i Oracle — ma ponad 30 różnych dozwolonych typów danych.

Omówienie każdego dostępnego typu danych z uwzględnieniem wszelkich związanych z nim niuansów byłoby niemożliwe, nawet jeśli mowa jest tylko o trzech bazach danych podanych powyżej. W tej książce dokonamy jednak pewnego streszczenia tego tematu, charakteryzując główne kategorie typów danych, które występują w większości baz danych.

Gdy tylko zapoznasz się z istotnymi typami danych w tych kategoriach, nie będziesz miał większych problemów z innymi, z którymi możesz się zetknąć w przyszłości. Ogólnie rzecz ujmując, istnieją trzy fundamentalne typy danych: liczbowy, znakowy oraz daty i czasu.

Typy danych liczbowych (ang. *numeric datatypes*) występują pod różnymi postaciami — w formie bitów, liczb całkowitych, dziesiętnych i rzeczywistych. *Bity* (ang. *bits*) są typami danych liczbowych, które pozwalają na określenie tylko dwóch wartości — 0 i 1. Są one często używane w celu określenia, że dany atrybut ma przyjmować wyłącznie wartości typu prawda lub fałsz. Typ danych określający *liczby całkowite* (ang. *integers*) wskazuje na liczby bez miejsc po przecinku, natomiast typy danych dla *liczb dziesiętnych* (ang. *decimals*) mogą zawierać wartości dziesiętne po przecinku. W odróżnieniu od bitów, liczb całkowitych i dziesiętnych, wartości *liczb rzeczywistych* (ang. *real*) są podawane jedynie w przybliżeniu, według wewnętrznie ustalonych zasad. Jedną wyróżniającą cechą wszystkich typów danych liczbowych jest to, że mogą one być uwzględnione w obliczeniach arytmetycznych. Oto kilka reprezentatywnych przykładów typów danych liczbowych z Microsoft SQL Server, MySQL i Oracle.

Ogólny opis	Typ danych w Microsoft SQL Server	Typ danych w MySQL	Typ danych w Oracle	Przykład
bit (ang. *bit*)	`bit`	`bit`	(brak)	1
liczba całkowita (ang. *integer*)	`int`	`int`	`number`	43
liczba dziesiętna (ang. *decimal*)	`decimal`	`decimal`	`number`	58,63
liczba rzeczywista (ang. *real*)	`float`	`float`	`number`	80,62345

Typy danych *znakowych* (ang. *character*) są czasem określane jako *łańcuchy znaków* (ang. *strings*) lub *ciągi znaków* (ang. *character strings*). W odróżnieniu od typów danych liczbowych, znakowe typy danych nie ograniczają się do liczb. Mogą zawierać jakiekolwiek litery lub cyfry, a nawet znaki specjalne, takie jak gwiazdki. Gdy za pomocą instrukcji SQL uzupełniana jest wartość w kolumnie o typie znakowym, zawsze musi być podawana w pojedynczym cudzysłowie. W przypadku typów danych liczbowych nigdy nie należy używać cudzysłowu. Poniżej znajduje się kilka przykładów prezentujących typy danych znakowych.

Ogólny opis	Typ danych w Microsoft SQL Server	Typ danych w MySQL	Typ danych w Oracle	Przykład
zmienna długość (ang. *variable length*)	`varchar`	`varchar`	`varchar2`	'Walt Disney'
stała długość (ang. *fixed length*)	`char`	`char`	`char`	'60601'

Wygląda na to, że drugi przykład (60601) to prawdopodobnie kod pocztowy[2]. Na pierwszy rzut oka może jednak wydawać się, że jest to liczba, ponieważ składa się wyłącznie z cyfr. Nie jest to nic niezwykłego. Mimo że kody pocztowe w Stanach Zjednoczonych składają się jedynie z cyfr, zazwyczaj definiowane są jako znakowe typy danych, ponieważ nigdy nie ma potrzeby wykonywania obliczeń arytmetycznych z ich udziałem.

Typy danych związanych z datą i czasem (ang. *date/time*) są wykorzystywane do prezentowania dat i czasu. Podobnie jak typy danych znakowych, muszą być podawane w pojedynczym cudzysłowie. Na tym typie danych możliwe jest wykonywanie specjalnych obliczeń; na przykład można użyć specjalnej funkcji, aby obliczyć liczbę dni pomiędzy dwiema datami zawierającymi zarówno datę, jak i czas. Oto kilka przykładów typów danych związanych z datą i czasem:

Ogólny opis	Typ danych w Microsoft SQL Server	Typ danych w MySQL	Typ danych w Oracle	Przykład
data (ang. *date*)	`date`	`date`	(brak)	'2017-02-15'
data i czas (ang. *date and time*)	`datetime`	`datetime`	`date`	'2017-02-15 08:48:30'

Wartości NULL

Inną ważną cechą poszczególnych kolumn w tabeli jest to, czy kolumna może zawierać wartości `null`. Wartość `null` oznacza, że nie ma danych dla określonego elementu danych. Dosłownie, pole takie nie zawiera żadnych danych. Należy zaznaczyć, że wartości `null` nie są tym samym co spacje i puste pola. Logicznie rzecz ujmując, wartości `null` i spacje są traktowane inaczej. Niuanse związane z pobieraniem danych, które zawierają wartości `null`, zostaną szczegółowo omówione w rozdziale 7., zatytułowanym „Logika Boole'a".

Podczas wyświetlania danych z wartościami `null` wiele baz danych wyświetli słowo `NULL` napisane wielkimi literami. Dzieje się tak po to, aby użytkownik wiedział, że dana kolumna zawiera wartość `null`, a nie spacje. W całej książce będziemy trzymali się tej konwencji i używali pisowni `NULL`, aby podkreślić, że reprezentuje ona unikalny typ wartości.

Klucze główne w bazie danych nie mogą zawierać wartości `NULL`. Jest tak, ponieważ klucze główne, zgodnie z definicją, muszą zawierać unikalne wartości.

[2] Kody pocztowe w USA są zapisywane w postaci ciągu cyfr, bez myślnika, o czym jest mowa w dalszej części akapitu — *przyp. tłum.*

Znaczenie SQL

Zanim zakończymy omawianie tematu relacyjnych baz danych, warto przedstawić krótki rys historyczny rozwoju baz danych do obecnej postaci. Poznając go, docenisz przydatność relacyjnych baz danych i znaczenie SQL.

W latach 60. XX wieku, gdy dziedzina informatyki dopiero zaczynała się rozwijać, dane zazwyczaj przechowywane były na taśmie magnetycznej lub w plikach na dyskach. Programy komputerowe, napisane w językach takich jak FORTRAN i COBOL, zwykle odczytywały dane za pośrednictwem plików wejściowych i dokonywały przetwarzania w trybie jednego rekordu naraz, na koniec przenosząc dane do plików wyjściowych. Przetwarzanie było zawsze złożone, ponieważ procedury musiały być dzielone na wiele pojedynczych etapów, obejmujących tabele tymczasowe, sortowanie i wielokrotne przetwarzanie danych do momentu otrzymania prawidłowego wyniku.

W latach 70., wraz z pojawieniem się hierarchicznych i sieciowych baz danych i rozpoczęciem korzystania z nich, dokonał się postęp. Dzięki skomplikowanemu systemowi wewnętrznych wskaźników bazy danych nowszego typu ułatwiły odczytywanie danych. Na przykład program mógł odczytać rekord z informacjami o kliencie, automatycznie wskazując wszystkie zamówienia danego klienta, a następnie szczegóły każdego z tych zamówień. Jednak w zasadzie odbywało się to nadal zgodnie z zasadą przetwarzania w danym momencie tylko jednego rekordu.

Przed pojawieniem się relacyjnych baz danych głównym problemem nie było to, w jaki sposób dane były przechowywane, ale jak wyglądał dostęp do nich. Prawdziwy przełom w związku z relacyjnymi bazami danych nadszedł wraz z pojawieniem się języka SQL, ponieważ umożliwił on dostęp do danych w zupełnie inny sposób.

W przeciwieństwie do wcześniej stosowanych metod pobierania danych, SQL umożliwiał użytkownikowi dostęp do dużego zbioru danych w tym samym momencie. Za pomocą jednej instrukcji polecenie SQL mogło pobrać lub zaktualizować tysiące rekordów w wielu tabelach. W ten sposób proces ten stał się dużo mniej złożony. Nie były już potrzebne programy komputerowe do odczytywania jednego rekordu naraz w specjalnej sekwencji przy jednoczesnym podejmowaniu decyzji o tym, co należy zrobić z każdym rekordem. To, co do tej pory wymagało setek linii kodu programowania, mogło być teraz realizowane za pomocą zaledwie kilku linii określonej logiki.

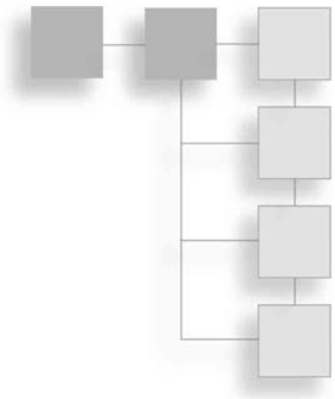
Co dalej?

W tym rozdziale przedstawiono podstawowe informacje o relacyjnych bazach danych. Mając już podstawową wiedzę na ich temat, możemy przejść do głównego zagadnienia, którym będziemy się zajmować, a więc do kwestii związanych z pobieraniem danych z baz danych. Omówiliśmy kilka ważnych cech relacyjnych baz danych, takich jak klucze główne,

klucze obce i typy danych. Wspomnieliśmy również o tym, że możliwe jest wystąpienie w danych wartości `NULL`. Uzupełnimy tę wiedzę w rozdziale 7., zatytułowanym „Logika Boole'a", natomiast w rozdziale 18. „Utrzymanie tabel" powrócimy do ogólnych zagadnień związanych z utrzymaniem bazy danych. Rozdział 19. „Zasady projektowania baz danych" poświęcony jest projektowaniu baz danych.

Dlaczego tak ważny temat, jakim jest projektowanie baz danych, omówiony zostanie w niniejszej książce dopiero kilkanaście rozdziałów dalej? Takie podejście jest jak najbardziej uzasadnione. Krótko mówiąc, warto najpierw zgłębić zagadnienia związane stricte z językiem SQL bez zaprzątania sobie głowy szczegółami dotyczącymi projektowania baz danych, które nosi znamiona zarówno sztuki, jak i nauki. Wówczas, po zapoznaniu się ze szczegółami i niuansami związanymi z pobieraniem danych, być może zasady projektowania baz danych zyskają jeszcze bardziej na znaczeniu. Na razie więc zignorujemy zagadnienia związane z projektowaniem baz danych i w następnym rozdziale przejdziemy od razu do kwestii pobierania danych.

Skorowidz



J

K

L

Ł

M

N

O

P

R

S

T

U

W

Z

Ź

Notatki

Poznaj SQL, a zrozumiesz bazy danych!

Najlepsze relacyjne bazy danych, takie jak Oracle czy MS SQL Server, są nierozłącznie związane z językiem SQL. Stworzono go po to, aby budować i użytkować bazy przechowujące ogromne ilości danych. Bez wątpienia SQL jest dość złożony, obejmuje wiele elementów i funkcji, jednak jego znajomość jest niezwykle ważna dla każdego, kto zajmuje się bazami danych, tworzy je czy nimi administruje. Również te osoby, które korzystają z narzędzi do raportowania w bazach danych, powinny przynajmniej dobrze zrozumieć podstawy tego języka.

Niniejsza książka jest kolejnym wydaniem popularnego podręcznika, dzięki któremu zrozumiesz SQL, jego składnię i najważniejsze aspekty wykorzystywania. Poszczególne tematy zorganizowano w intuicyjny sposób, przedstawiając je w logicznej kolejności. Przykłady zastosowania języka dobrano tak, aby za pomocą małej próbki kodu umożliwić zrozumienie danej instrukcji SQL. W tym wydaniu zaktualizowano informacje o składni SQL stosowanej w Microsoft SQL Server 2016, MySQL 5.7 i Oracle 12c, uzupełniono zagadnienia dotyczące logiki warunkowej, a także przedstawiono kilka nowych tematów, takich jak wspólne wyrażenia tablicowe czy wstawianie komentarzy do zapytań.

Najważniejsze zagadnienia ujęte w książce:

- podstawowe informacje o relacyjnych bazach danych
- składnia instrukcji SELECT i jej możliwości
- agregacje danych i sum częściowych
- stosowanie złączeń, podzapytań, widoków i logiki zbiorów
- procedury składowane, aktualizacja danych i utrzymanie bazy
- projektowanie baz danych i sposoby prezentacji danych

LARRY ROCKOFF jest ekspertem w dziedzinie języka SQL i analityki biznesowej. Specjalizuje się w stosowaniu narzędzi do raportowania w celu analizy danych znajdujących się w złożonych bazach danych. Ukończył studia MBA na Uniwersytecie Chicago. Jest autorem książek poświęconych językowi SQL i zastosowaniu oprogramowania Microsoft Access i Excel. Obecnie odpowiada za rozwój hurtowni danych oraz aplikacji służących do raportowania dla największej sieci aptek w USA.

Helion
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900
Informatyka w najlepszym wydaniu

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

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

Addison-Wesley

ISBN 978-83-283-3190-7
9 788328 331907
cena: 47,00 zł