JĘZYK SQL

PRZYJAZNY PODRĘCZNIK

WYDANIE III

LARRY ROCKOFF

Tytuł oryginału: The Language of SQL, 3rd Edition

Tłumaczenie: Piotr Cieślak na podstawie książki "Język SQL. Przyjazny podręcznik. Wydanie II" w przekładzie Beaty Błaszczyk

ISBN: 978-83-283-9264-9

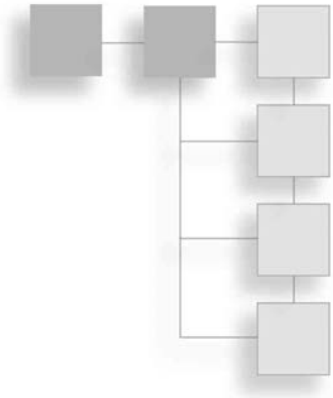
Authorized translation from the English language edition, entitled The Language of SQL, 3rd Edition by Larry Rockoff, published by Pearson Education, Inc, publishing as Addison Wesley Professional, Copyright © 2022 by Pearson Education, Inc.

Polish language edition published by Helion S.A., Copyright © 2022.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

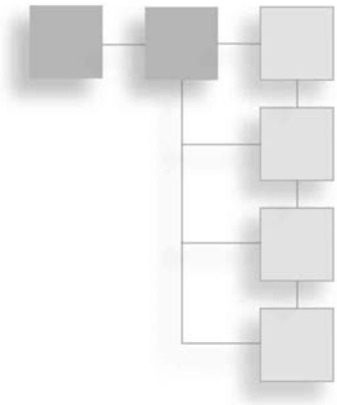
Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/jsqlp3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.



Spis treści

ROZDZIAŁ 1



Relacyjne bazy danych i SQL

Jak wspomnieliśmy 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 unikatową 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.

Biorąc pod uwagę tę dwoistość, przy prezentowaniu poszczególnych zagadnień składających się na SQL staraliśmy się uwzględnić obydwa powyższe aspekty — język oraz komponenty logiczne. 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, co da 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 trudno oprzeć się wrażeniu, że jego polecenia są analogiczne do zdań w języku angielskim i mają konkretne znaczenie.

Porównaj na przykład poniższe zdanie:

```
Poproszę o muffinkę jagodową z menu deserów i podgrzanie jej przed podaniem.
```

z instrukcją SQL:

```
Select miasto, województwo1
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 o nazwie `Klienci`, znajdującej się w naszej bazie danych. Ponadto chcemy posortować wyniki alfabetycznie według województwa.

W obu przypadkach określamy interesujące nas elementy (muffinka lub miasto/województwo), wskazujemy, skąd chcemy je pozyskać (menu deserów lub tabela *Klienci*), oraz zamieszczamy dodatkowe instrukcje (podgrzanie dania lub posortowanie 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. Jedna polega na wymówieniu pojedynczych liter, czyli powiedzeniu „S-Q-L". Preferowaną przez autora alternatywą jest wymówienie akronimu jako pojedynczego słowa „siquel". Ta wersja składa się tylko z dwóch sylab i 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.

[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.*

Czym jest 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 podstawowy zakres zostanie omówiony w tej książce.

Jako język programowania SQL różni się od pozostałych języków, takich jak Python 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 2019*. 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 naszym celem było omówienie podstaw SQL odnoszących się do wszystkich jego implementacji, musieliśmy przedstawić konkretne przykłady składni zapytań 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 systemy baz danych:

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

W większości przypadków składnia poleceń SQL stosowanych w powyższych bazach danych jest taka sama, jednak zdarzają się odstępstwa od tej reguły. Gdy takie różnice będą miały miejsce, w tekście książki zostanie przedstawiona składnia stosowana w Microsoft SQL Server. Wszelkie różnice w składni występujące w przypadku MySQL oraz Oracle zostaną omówione w ramce zatytułowanej „Różnice w ramach innych baz danych", takiej jak poniższa.

Różnice w ramach innych baz danych

> W tego typu ramkach omawiane są wszelkie charakterystyczne cechy składni różniące systemy MySQL i Oracle od Microsoft SQL Server. Składnia tego ostatniego jest przedstawiona w głównej części tekstu.

Microsoft SQL Server jest dostępny w kilku wersjach i edycjach. Jego najnowsze wydanie w chwili, gdy pisaliśmy te słowa, nosiło oznaczenie *Microsoft SQL Server 2019*. 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 jej rozwój nie jest uzależniony od tylko jednej organizacji. MySQL jest dostępna na wielu platformach innych niż Windows, takich jak macOS i Linux. Community Edition to baza danych MySQL, którą można pobrać za darmo. Najnowsza wersja tej bazy danych to MySQL 8.0.x.

Ostatnia baza danych na naszej liście to Oracle. Jest ona dostępna w wielu wersjach. Najnowsza to *Oracle Database 18c*. Bezpłatna edycja tej bazy danych nosi nazwę Express Edition (XE).

Zaczynając pracę z bazami danych, czasami warto wcześniej pobrać wybrany system bazodanowy, aby móc z nim poeksperymentować. 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 zamieścimy ponadto wystarczającą ilość danych, które umożliwią Ci zrozumienie wyników różnych instrukcji SQL bez konieczności pobierania oprogramowania i samodzielnego wpisywania oraz wykonywania prezentowanych instrukcji.

Jeśli jednak 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. Zamieszczona 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.

Zgodnie z tym, o czym pisaliśmy we wstępie, na stronie internetowej z materiałami pomocniczymi znajdziesz archiwum ze wszystkimi instrukcjami SQL, jakie zostały zamieszczone w tej książce, z uwzględnieniem składni właściwej 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 ze wspominanych wyżej dodatkowych materiałów.

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

- DB2 firmy IBM,
- MongoDB (system baz danych open source),
- PostgreSQL (system baz danych open source),
- Microsoft Access firmy Microsoft.

Z wymienionych powyżej baz danych dość wyjątkowy jest Microsoft Access, przy czym wyjątkowość ta dotyczy głównie aspektu wizualnego. Access jest bowiem w istocie graficznym interfejsem dla relacyjnych baz danych. Innymi słowy, umożliwia on tworzenie zapytań (zwanych w nim kwerendami) przy użyciu interaktywnych narzędzi wizualnych. Z perspektywy

początkującego zaletą Accessa jest możliwość łatwego, wizualnego projektowania zapytań i przełączania się na widok SQL w celu wyświetlenia treści utworzonej instrukcji. Innym istotnym wyróżnikiem Accessa jest to, że baza ta jest instalowana lokalnie. Za pomocą tego narzędzia można 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 przejdziemy teraz do omówienia podstaw relacyjnych baz danych oraz sposobu ich działania. Relacyjna baza danych jest zbiorem danych przechowywanych w dowolnej liczbie tabel. W powszechnym znaczeniu termin *relacyjne* (ang. *relational*) oznacza, 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 odzwierciedla logiczne właściwości decydujące o sposobie powiązania tabel.

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 się okazać 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 (główka tabeli) zazwyczaj zawiera nazwy pól. W pozostałych wierszach znajdują się właściwe dane.

W terminologii SQL rekordy i pola noszą nazwę *wierszy* (ang. *rows*) i *kolumn* (ang. *columns*), zgodnie z ich wizualną reprezentacją. 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
4	Sylwia	Bóbr

Tabela Zamówienia mogłaby mieć następującą postać:

IDZamówienia	IDKlienta	DataZamówienia	KwotaZamówienia
1	1	2021-09-01	10,00
2	2	2021-09-02	12,50
3	2	2021-09-03	18,00
4	3	2021-09-15	20,00

W powyższym przykładzie tabela Klienci zawiera trzy kolumny: IDKlienta, ImięKlienta i NazwiskoKlienta. Każdy z czterech wierszy tabeli prezentuje dane jednej osoby: Jana Kowalskiego, Andrzeja Nowaka, Anny Kwiatkowskiej i Sylwii Bóbr. Każdy wiersz reprezentuje innego klienta, a każda kolumna zawiera inny fragment informacji o nim. Na podobnej zasadzie w tabeli Zamówienia znajdują się cztery kolumny i cztery wiersze. Oznacza to, że w bazie danych istnieją cztery zamówienia i cztery 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, e-mail i telefon. Podobnie tabela Zamówienia zazwyczaj zawiera kolumny opisujące dodatkowe atrybuty zamówienia, takie jak data, wartość podatku oraz informacje o sprzedawcy, który przyjął zamówienie.

Klucze główne i obce

Zwróć uwagę na pierwszą kolumnę w opisanych tabelach: IDKlienta w tabeli Klienci i IDZamówienia w tabeli Zamówienia. Kolumny te są zwykle określane jako *klucze główne* (ang. *primary keys*). Klucze główne są przydatne, a wręcz niezbędne z dwóch powodów. Przede wszystkim 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 unikatowość. Oznaczenie kolumny IDKlienta jako klucza głównego gwarantuje, że w tej kolumnie będzie się znajdować unikatowa 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, 3 i 4 dla czterech wierszy w tabeli. Często bowiem tabele w bazie danych są zaprojektowane w taki sposób, aby kolumny z kluczem głównym wraz z dodawaniem kolejnych wierszy były wypełniane automatycznie generowanymi numerami sekwencyjnymi. Ta cecha projektowa jest zwykle 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 tego, że wskazuje ona osobę zamawiającego w 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 „Utrzymywanie 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 każdy z opisywanych w tej książce systemów — 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 dopuszczają tylko dwie 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. Wyróżniającą cechą wszystkich typów danych liczbowych jest to, że mogą one być uwzględnione w obliczeniach arytmetycznych. Oto kilka reprezentatywnych przykładów typów danych liczbowych z systemów 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* (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 się jednak wydawać, 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)	`'2021-12-15'`
data i czas (ang. *date and time*)	`datetime`	`datetime`	`date`	`'2021-12-15 08:48:30'`

Wartości NULL

Inną ważną cechą poszczególnych kolumn w tabeli jest to, czy kolumna może przyjmować wartość `null`. Oznacza ona, że określony element nie zawiera żadnych danych — innymi słowy, jest zupełnie pusty. Należy zaznaczyć, że wartości `null` nie są tożsame z białymi znakami, takimi jak spacje. Pod względem logicznym 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 prezentowania 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 unikatowy typ wartości.

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

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

Krótka historia systemów baz danych

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 były zazwyczaj przechowywane 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 SQL, ponieważ język ten umożliwił dostęp do danych w zupełnie inny sposób. Postęp ten miał swoje korzenie w opublikowanej w 1970 roku przełomowej pracy Edgara F. Codda, informatyka z firmy IBM, w której przedstawił on teoretyczne zarysy relacyjnych baz danych. Trzy lata później teorie te skłoniły dwóch innych informatyków z IBM, Donalda D. Chamberlina i Raymonda F. Boyce'a, do rozpoczęcia prac nad językiem umożliwiającym komunikację z relacyjnymi bazami danych. Pod koniec lat 70. język ten zyskał kształt bardzo zbliżony do tego, co obecnie nazywamy SQL.

W przeciwieństwie do wcześniej stosowanych metod SQL umożliwiał użytkownikowi dostęp do dużego zbioru danych w tym samym momencie. Za pomocą jednej instrukcji SQL można było 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 krótkiej, kilkuwierszowej instrukcji.

Co dalej?

W tym rozdziale przedstawiliśmy ogólne 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., „Utrzymywanie tabel", powrócimy do ogólnych zagadnień związanych z utrzymaniem bazy danych. Rozdział 19., „Zasady projektowania baz danych", jest poświęcony projektowaniu baz danych.

Dlaczego tak ważny temat, jakim jest projektowanie baz danych, zostanie omówiony 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 pobierania danych, zasady projektowania baz danych zyskają jeszcze bardziej na znaczeniu. Na razie więc zignorujemy zagadnienia dotyczące projektowania baz danych i w następnym rozdziale przejdziemy od razu do kwestii pobierania danych.


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Chcesz się nauczyć SQL? Z tą książką zrobisz to bez problemu!

SQL stanowi podstawowe narzędzie komunikowania się z relacyjnymi bazami danych. Jest to dość skomplikowany język o rozbudowanych możliwościach. Pozwala na pobieranie z bazy zestawów danych wyszukiwanych na podstawie wyrafinowanych kryteriów. Umożliwia też modyfikację zarówno danych, jak i struktury bazy, w której są gromadzone. To potężne narzędzie powinni znać nie tylko programiści baz danych, ale również specjaliści zajmujący się ich analizą, jednak zdobycie praktycznych umiejętności posługiwania się SQL często nastręcza trudności.

To trzecie, poprawione i zaktualizowane wydanie praktycznego przewodnika po języku SQL i relacyjnych bazach danych. Przemyślana, uporządkowana struktura podręcznika sprzyja sprawnemu nabywaniu wiedzy i doskonaleniu umiejętności — pracę z nim ułatwiają przystępny sposób prezentowania materiału i czytelne, łatwe do zrozumienia przykłady kodu SQL. Bardzo przydatnym elementem książki są odniesienia do składni (dialektów SQL) stosowanych w trzech najpowszechniejszych bazach danych: Microsoft SQL Server 2019, MySQL 8.0 i Oracle 18c. W tym wydaniu pojawiło się szersze omówienie typowych zadań analitycznych, uzupełniono też informacje o zastosowaniu Excela do wizualnej prezentacji danych, opisano więcej przydatnych funkcji, a także zaktualizowano i ulepszono dodatkowe materiały edukacyjne.

W książce między innymi:

- operacje na danych przy użyciu zapytań SQL
- funkcje i procedury składowane
- grupowanie i agregowanie danych
- projektowanie relacyjnych baz danych
- tworzenie zestawień danych przy użyciu arkuszy kalkulacyjnych
- praca z Microsoft SQL Server, MySQL i Oracle

LARRY ROCKOFF od wielu lat zajmuje się zagadnieniami związanymi z SQL i analizą danych gromadzonych w złożonych bazach danych. Jest autorem cenionych publikacji poświęconych językowi SQL, a także zastosowaniu oprogramowania Microsoft Access i Excel do zaawansowanej analizy danych.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-283-9264-9

Cena: 69,00 zł

Pearson
Addison-Wesley