SQL

Przewodnik dla początkujących

JAK ZACZĄĆ EFEKTYWNĄ PRACĘ Z DANYMI

Walter Shields

Helion

Tytuł oryginału: SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL

Tłumaczenie: Agnieszka Górczyńska

ISBN: 978-83-8322-657-6

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

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

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

Printed in Poland.

Spis treści

CZĘŚĆ II TWORZENIE POLECEŃ SQL

1

Struktura bazy danych

Oto tematy, które zostały omówione w tym rozdziale:

- język używany przez bazy danych;
- sposób działania relacyjnej bazy danych;
- typy danych;
- systemy zarządzania relacyjnymi bazami danych;
- SQLite.

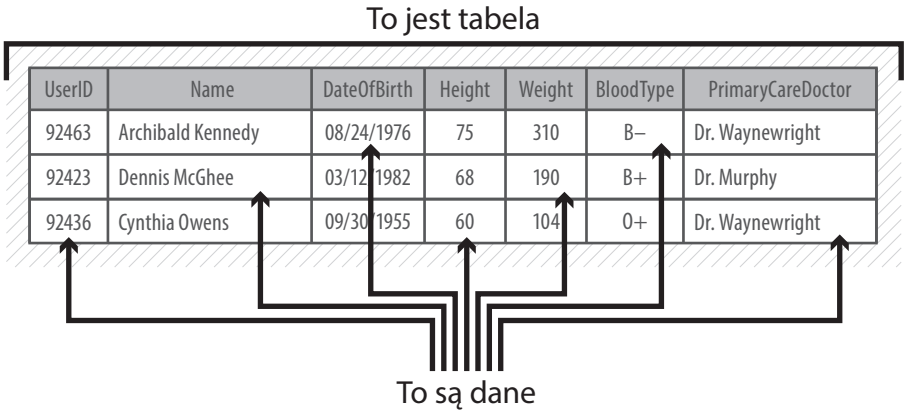
Aby rozpocząć zdobywanie nowych umiejętności technicznych, konieczne jest opanowanie słownictwa z danej dziedziny. Pod tym względem zamierzam zachować właściwą równowagę: wyjaśnię podstawowe pojęcia i koncepcje, których będziesz potrzebować do zrozumienia materiału zamieszczonego w książce, przy czym postaram się uniknąć niepotrzebnego żargonu i koncepcji zaawansowanych. Z tego rozdziału dowiesz się, czym jest relacyjna baza danych oraz jakie typy danych mogą być w niej przechowywane. Ponadto przedstawię podstawowe zapytanie SQL: `SELECT`.

Podstawowe pojęcia

Wprawdzie dane pojawiają się wszędzie i znajdują się praktycznie we wszystkim, ale z praktycznego punktu widzenia pojęcie „dane" odnosi się do informacji możliwych do zapisania lub do zarejestrowania. Jednym z najprostszych narzędzi używanych do zapisywania i wizualizowania jest ***tabela***, czyli dwuwymiarowa siatka składająca się z rekordów (wierszy) i kolumn.

Podczas pracy w środowisku baz danych tabela może być określana mianem „base relvar". W książce pozostanę przy określeniu „tabela". Graficzne podsumowanie pojęć znajdziesz w tabeli 1.1.

Rysunek 1.1.
Przykładowa tabela bazy danych



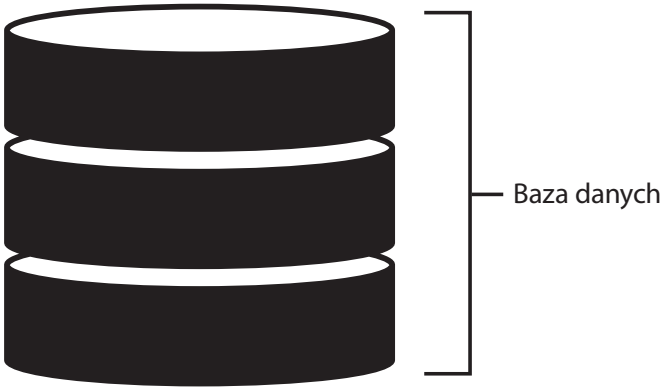
Jak możesz zobaczyć na rysunku 1.1, tabela przechowuje dane różnych typów. Dane mogą mieć postać imion i nazwisk, liczb, dat, znaków (takich jak „+” i „–”), a także wiele innych postaci. Dane można najprościej określić słowem „informacje”. Dlatego podczas ich przetwarzania należałoby je odpowiednio ograniczyć. Można odnieść wrażenie, że tabela pokazana na rysunku 1.1 przechowuje pewne informacje o grupie pacjentów. Te dane zostały zdefiniowane w różnych formatach. Mamy więc liczby, imiona i nazwiska oraz daty, w kolumnie `BloodType` zaś przechowywany jest ciąg tekstowy składający się z dwóch znaków (jednym z nich jest „+” lub „–”). Formaty używane do przedstawiania informacji nie są przypadkowe. Wszystkie bazy danych zawierają tzw. ***metadane***, czyli zbiór danych opisujących strukturę i formatowanie samych danych, czyli innymi słowy „dane dotyczące danych”. Na przykład `DateOfBirth` może zawierać metadane ograniczające format przechowywanych informacji do mm/dd/rrrr. Z kolei metadane w kolumnie `Height` mogą ograniczać dane na przykład do dwóch liczb i wymagać, aby wartość była podana w calach.

Pojęcie ***baza danych*** może oznaczać zbiór danych zorganizowanych w sposób ułatwiający oraz przyśpieszający ich wyszukiwanie i pobieranie przez komputer. W sposób graficzny baza danych jest często przedstawiana w postaci ikony pokazanej na rysunku 1.2, która ma symbolizować stos dysków tworzących centrum pamięci masowej o dużej pojemności.

Dane w bazie danych są zwykle przechowywane w postaci zbioru tabel, z których każda zawiera określony zbiór danych. Te dane mogą być ze sobą powiązane oraz odwoływać się do danych znajdujących się w innych tabelach bazy danych.

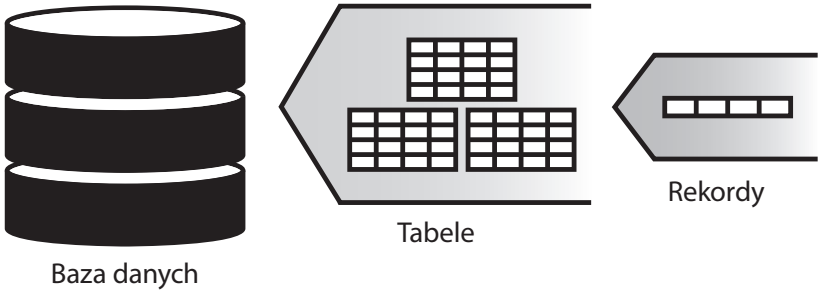
Pokazana na rysunku 1.1 tabela pacjentów to po prostu tabela, a nie baza danych. Jednak ta tabela może znajdować się w bazie danych, razem z innymi tabelami przechowującymi na przykład o wynikach badań laboratoryjnych, przepisanych lekach, historii wizyt, personelu szpitala, lekarzach, specjalnościach, wolnych terminach wizyt itd.

Rysunek 1.2.
Często używana
ikona bazy danych



Celem bazy danych jest ułatwienie gromadzenia, przechowywania i analizowania powiązanych ze sobą danych pochodzących z wielu źródeł. Gdy dane zostają umieszczone w tabelach, które są powiązane z innymi tabelami bazy danych, wówczas możliwe jest osiągnięcie większej wszechstronności (rysunek 1.3)

Rysunek 1.3.
Baza danych składa się
z tabel, które z kolei
zawierają rekordy



Wiersz tabeli jest nazywany ***rekordem***. Możemy go nazwać również *krotką*. ***Kolumna*** tabeli jest nazywana ***polem***, można się spotkać również z określeniem ***atrybut***. Pola/atrybuty to kategorie używane do zdefiniowania danych w rekordzie (wierszu).

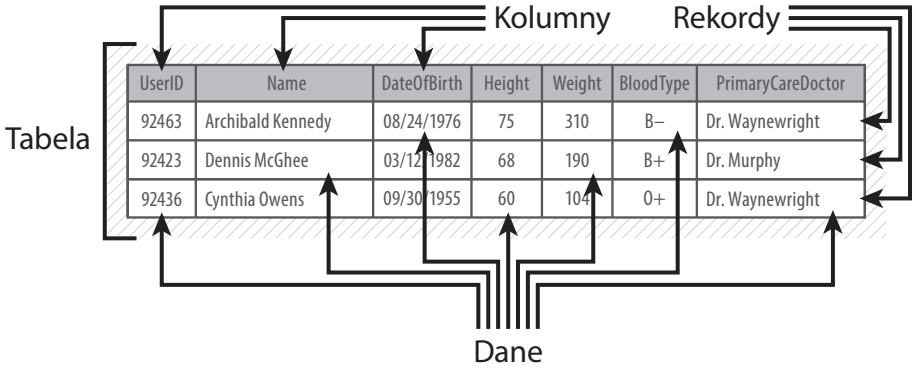
W książce będę używał określenia „rekord" w odniesieniu do wiersza tabeli oraz określenia „kolumna" w odniesieniu do pól. Zapoznaj się z tabelą 1.1, w której zamieściłem podsumowanie terminologii dotyczącej bazy danych.

Tabela 1.1. Podsumowanie terminologii dotyczącej bazy danych

Określenia używane w książce	Określenia, z którymi można się spotkać w innych źródłach
rekord, wiersz	krotka
kolumna, pole	atrybut
tabela	relacja, base relvar

Każdy rekord składa się z kilku kolumn reprezentujących pojedyncze elementy danych opisujące konkretną rzecz. Na przykład tabela pokazana na rysunku 1.4 zawiera informacje o pacjentach, prawdopodobnie konkretnego szpitala, przychodni bądź firmy ubezpieczeniowej. Niezależnie od natury organizacji, jeżeli korzysta ona z informacji przechowywanych w bazie danych, to prawdopodobnie taka baza będzie składała się z wielu tabel. Zrozumienie sposobu, w jaki tabele odwołują się do siebie, to kolejny kluczowy aspekt podczas poznawania architektury bazy danych.

Rysunek 1.4.
Przykładowa tabela bazy danych



Podstawowe elementy relacyjnej bazy danych

Relacyjna baza danych to baza danych zaprojektowana w postaci opracowanej w 1969 roku przez Edgara F. Codda, inżyniera firmy IBM. Rok później Codd opublikował ten projekt w artykule zatytułowanym *A Relational Model of Data for Large Shared Data Banks*[1]. Dziewięć lat później kilka wielkich firm technologicznych, m.in. IBM i Relational Software (później przekształcona w Oracle), zaczęło stosować relacyjne bazy danych w rozwiązaniach komercyjnych. Kilka dekad później model relacyjny wciąż jest najczęściej wykorzystywanym projektem bazy danych.

1 *https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf*

Aby ogólnie zrozumieć sposób działania relacyjnej bazy danych, konieczne jest poznanie roli kolumn tzw. kluczy (rysunek 1.5).

Rysunek 1.5. Kolumny kluczy tabeli patient_info bazy danych

Kolumna klucza podstawowego

Kolumna klucza zewnętrznego

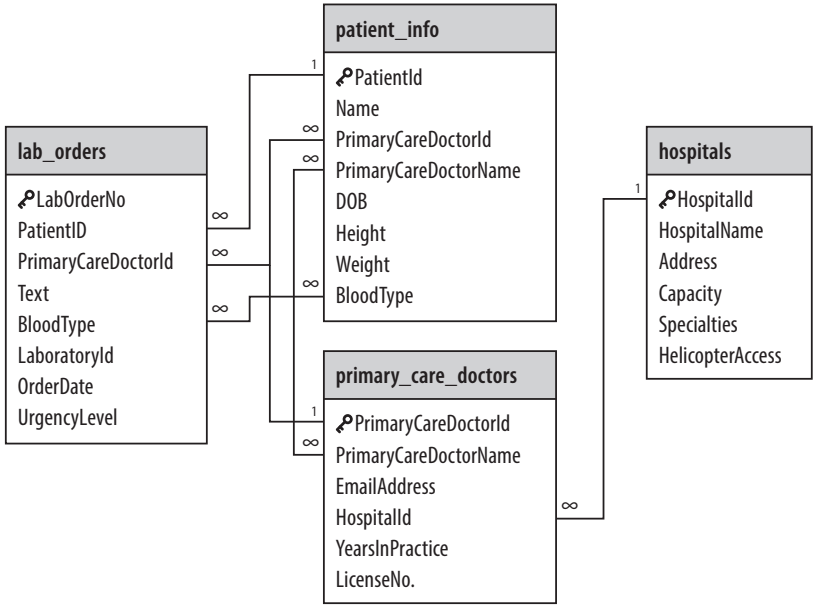
PatientID	PatientName	PrimaryCareDoctorID	PrimaryCareDoctorName	DateOfBirth	Height	Weight	BloodType
92463	Archibald Kennedy	106547	Dr. Waynewright	8/24/1976	75	310	B–
92425	Dennis McGhee	106474	Dr. Murphy	3/12/1982	68	190	B+
92443	Cynthis Owens	106547	Dr. Waynewright	9/30/1955	60	104	O+
92478	William Hampton	106437	Dr. Salazar	6/5/1973	73	175	AB–
92392	Hilda Bass	106783	Dr. Dean	6/10/1997	68	152	B+
92436	Frankie Stone	106437	Dr. Salazar	5/28/1979	68	106	O+
92403	Verna Sullivan	106984	Dr. Conner	7/17/2010	66	125	O+
92398	Merle Doyle	106439	Dr. Frank	1/8/1962	65	143	B–
92442	Ruth Swanson	106954	Dr. Hines	2/15/1970	61	160	O–
92384	Johnathan Singleton	106474	Dr. Murphy	6/2/1970	61	232	AB+
92405	WM Patrick	106439	Dr. Frank	6/11/1955	69	196	O+
92376	Mona Norris	106984	Dr. Conner	10/15/1932	60	98	B+
92399	Rick Gordon	106366	Dr. Hart	1/25/2002	68	149	B+
92408	Don Rivera	106437	Dr. Salazar	7/26/1954	72	185	A–
92389	Sheri Griffin	106211	Dr. Harvey	12/16/1987	78	132	AB–
92466	Guillermo Lawrence	106954	Dr. Hines	2/8/1978	60	219	O+
92310	Felipe Parker	106474	Dr. Murphy	12/10/1998	61	165	O–
92413	Brandi Carlson	106399	Dr. Flowers	11/20/2000	66	112	B+
92398	Floyd Casey	106783	Dr. Dean	12/14/1986	61	203	A–
92439	Patrick Walton	106366	Dr. Hart	8/11/1973	76	189	O+
92421	Vicki Klein	106954	Dr. Hines	11/28/1980	65	98	O+
92381	Cathy Harrison	106474	Dr. Murphy	11/16/1946	78	203	AB–
92393	Ann Guerrero	106783	Dr. Dean	6/25/1974	61	142	B–
92437	Gustavo Bates	106399	Dr. Flowers	2/25/2001	78	165	A–

Relacyjna baza danych zawiera wiele tabel podobnych do pokazanej na rysunku 1.5 tabeli `patient_info`. Te tabele są powiązane ze sobą za pomocą kolumn kluczy. W omawianej tabeli zwróć uwagę na kolumny klucza podstawowego i zewnętrznego. Za najlepszą praktykę uznaje się, że każda tabela relacyjnej bazy danych powinna zawierać ***klucz podstawowy***, który jest unikatowym identyfikatorem rekordu w tabeli. Klucze podstawowe rekordów muszą być unikatowe i nie mogą być puste (czyli nie mogą mieć wartości `null`). Spójrz na kolumnę `PatientId` w tabeli `patient_info`. Skoro ta kolumna jest uznawana za klucz podstawowy tabeli, każdy rekord musi zawierać unikatowe dane w tej kolumnie. Innymi słowy dwa rekordy nie mogą mieć takiej samej wartości `PatientId`.

Wprawdzie wartość kolumny klucza podstawowego (w omawianym przykładzie jest to `PatientId`) musi być unikatowa, ale pozostałe kolumny mogą zawierać dane powtarzające się w wielu rekordach. Na przykład spójrz na kolumnę `PrimaryCareDoctorId`. Jeżeli dr Waynewright (identyfikator 106547, zobacz pierwszy rekord w tabeli pokazanej na rysunku 1.5) opiekuje się wieloma pacjentami, to jego nazwisko i identyfikator będą się pojawiały w wielu rekordach.

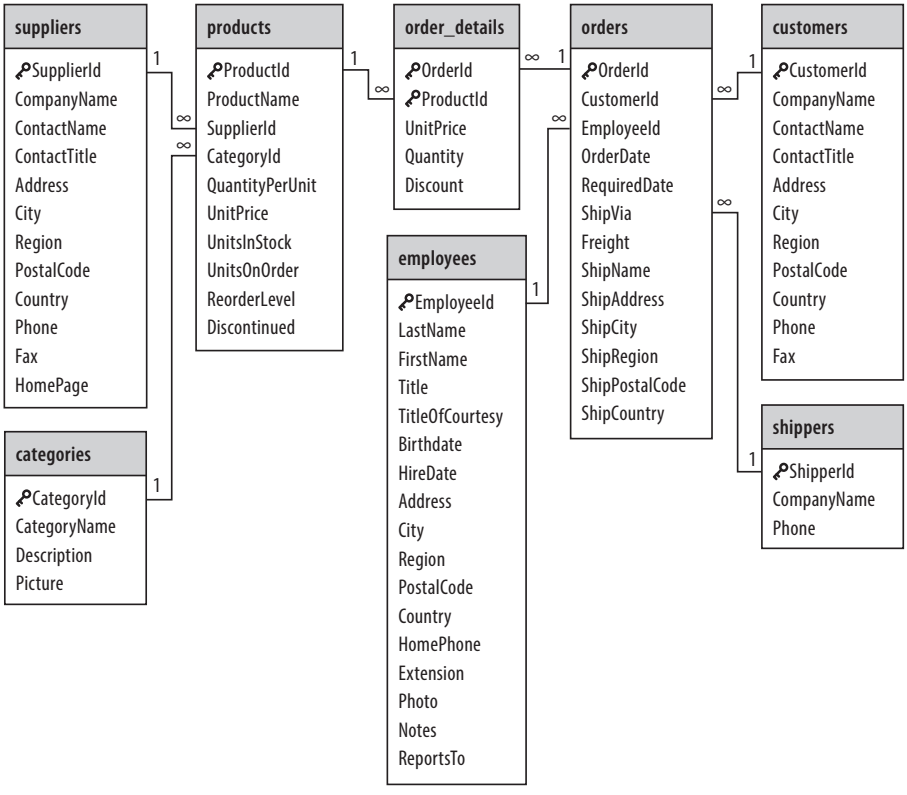
W relacyjnej bazie danych tabela są często określane mianem „relacji", ponieważ zawiera zbiór rekordów (wierszy) powiązanych z różnymi kolumnami (polami). Jednak w tej książce pozostanę przy określeniu „tabela". Zapoznaj się z terminologią zamieszczoną w tabeli 1.1.

Klucz zewnętrzny to kolumna tabeli będąca kluczem podstawowym w innej tabeli bazy danych. Załóżmy, że oprócz przedstawionej już tabeli `patient_info` baza danych zawiera jeszcze tabelę `primary_care_doctors`, w której kluczem podstawowym jest kolumna `PrimaryCareDoctorId`. W tabeli `primary_care_doctors` dr Waynewright o identyfikatorze 106547 będzie pojawiał się tylko w jednym rekordzie. Pojawianie się różnych kluczy w tabelach prowadzi do powstania ważnych zależności (relacji) w bazie danych trafnie określanej jako relacyjna. Te relacje są często przedstawiane za pomocą tzw. ***schematu*** bazy danych (można się spotkać także z określeniem ***entity relationship diagram*** (***ERD***)), który pokazuje strukturę bazy danych. Na rysunku 1.6 pokazałem schemat przykładowej bazy danych.



Rysunek 1.6. Schemat przykładowej bazy danych

Na razie nie zastanawiaj się nad znaczeniem znaków `1` i ∞ na rysunku 1.6, powrócę do nich za chwilę. Spróbuj przeanalizować schemat i znajdź relacje. Ten schemat bazy danych zawiera tylko cztery tabele, połączone ze sobą za pomocą co najmniej jednej kolumny. Kolumna `PatientId` to klucz podstawowy tabeli `patient_info`, a jednocześnie klucz zewnętrzny tabeli `lab_orders`. Podobnie `HospitalId` to klucz podstawowy tabeli `hospitals` i jednocześnie klucz zewnętrzny tabeli `primary_cate_doctors`. To jest całkiem proste, prawda? Spójrz teraz na rysunek 1.7, pokazujący zupełnie inny schemat bazy danych.



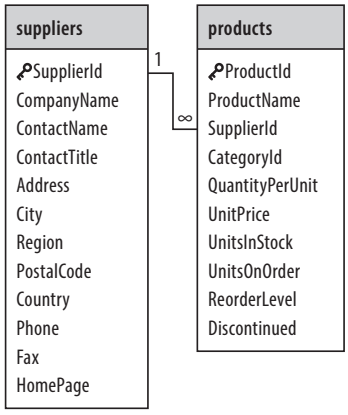
GRAFIKA

Rysunek 1.7.
Schemat innej bazy danych

Schemat zamieszczony na rysunku 1.7 opisuje bazę danych przeznaczoną do obsługi zamówień składanych przez klientów. Powrócę teraz do znaków `1` i ∞ widocznych na końcach linii znajdujących się na schemacie z rysunku 1.6: wskazują one naturę interaktywności między tabelami. Gdy na jednym końcu linii znajduje się znak `1`, a na drugim znak ∞, taka linia przedstawia relację „jeden do wielu" zachodzącą między kolumnami tabel.

Przyjrzyj się teraz pokazanej na rysunku 1.8 tabeli `products`. Nie ulega wątpliwości, że znajdujące się w niej dane dotyczą różnych produktów i ich atrybutów.

Rysunek 1.8.
Tabele suppliers i products



Kolumna `ProductId` jest kluczem podstawowym w tej tabeli, na co wskazuje ikona przedstawiająca klucz. Każdy rekord tabeli będzie zawierał unikatowy numer identyfikacyjny produktu. W rzeczywistości to jest cel istnienia tej tabeli — katalogowanie atrybutów różnych produktów, o których informacje są przechowywane w bazie danych.

Przejdźmy teraz do relacji między tabelami `suppliers` (rysunek 1.9) i `products` (rysunek 1.10).

Rysunek 1.9.
Tabela suppliers

Suppliers

SupplierId	CompanyName	ContactName	ContactTitle	Address	City	etc.
S-101	Van Eck Industries	Bruce Davidson	Vp Operations	2158 Del Dew Drive	Temple Hills	...
S-102	Wright & Gate Co	Wilma Joy	Supply Chain Supervisor	291 Creekside Lane	Ventura	...
S-103	Olivias Supply	Brad Pence	Site Manager, Baton Rouge	4353 Locust View Drive	Baton Rouge	...
S-104	Cantor Corporation	Orville Bedford	President	2811 West Drive	Chicago	...
S-105	Bellagio Finland	Wallace Grim	Distributions Superviser	4939 Breezewood Court	Chanute	...
S-106	Decks Materials	John Tuck	VP Operations	4529 Counts Lane	Lextington	...
S-107	Lennor Co	Rachel Durst	Site Manager, Jackson	2216 Rhapsody Street	Gainesville	...

Rysunek 1.10.
Tabela products

Products

ProductId	ProductName	SupplierId	CategoryId	QuantityPerUnit	UnitPrice	etc.
P001	Welding goggles	S102	SA-432	1	$12.99	...
P002	Welding helmet	S102	SA-432	1	$41.49	...
P003	Stick electrodes	S104	WE-214	40	$7.00	...
P004	Magnetic clamp	S101	WE-220	1	$11.86	...
P005	Heat resistant blanket	S104	WE-212	1	$3.73	...
P006	Work table	S105	GE-100	1	$1,386.67	...
P007	Replacement plates	S105	GE-100	1	$396.00	...
P008	Welding wire	S104	WE-214	1	$112.86	...
P009	Welding coveralls	S102	SA-435	1	$60.27	...
P010	Welding nozzle	S103	WE-214	1	$141.65	...
P011	Gas regulator	S106	AU-100	1	$166.25	...
P012	Welding hoods	S102	SA-432	1	$42.37	...
P013	Spot welding electrode	S104	WE-212	1	$2.35	...
P014	Plasma cutter	S107	PL-100	1	$1,645.91	...
P015	Plasma cutter cutting tip	S107	PL-100	1	$9.27	...

Między tabelami `suppliers` i `products` istnieje relacja typu „jeden do wielu", bazująca na kolumnie `SupplierId`. W tabeli `suppielrs` każdy rekord zawiera unikatowy numer identyfikacyjny poszczególnych dostawców (ang. *suppliers*), podczas gdy w tabeli produktów (ang. *products*) może istnieć wiele rekordów z takim samym numerem identyfikacyjnym dostawcy.

Przedstawiająca klucz ikona obok kolumny `SupplierId` w tabeli `suppliers` wskazuje, że ta kolumna jest kluczem podstawowym w tabeli. Zdecydowanie możemy mieć wiele różnych produktów (każdy z unikatowym identyfikatorem) pochodzących od tego samego dostawcy i skatalogowanych w tabeli `products`. Dla porównania w tabeli `suppliers` nie mogą znajdować się dwa lub więcej rekordów zawierających ten identyfikator dostawcy (rysunek 1.11).

Rysunek 1.11. Relacja zachodząca między tabelami products i suppliers

Products

ProductId	ProductName	SupplierId	CategoryId	QuantityPerUnit	UnitPrice	etc.
P001	Welding goggles	S102	SA-432	1	$12.99	...
P002	Welding helmet	S102	SA-432	1	$41.49	...
P003	Stick electrodes	S104	WE-214	40	$7.00	...
P004	Magnetic clamp	S101	WE-220	1	$11.86	...
P005	Heat resistant blanket	S104	WE-212	1	$3.73	...
P006	Work table	S105	GE-100	1	$1,386.67	...
P007	Replacement plates	S105	GE-100	1	$396.00	...
P008	Welding wire	S104	WE-214	1	$112.86	...
P009	Welding coveralls	S102	SA-435	1	$60.27	...
P010	Welding nozzle	S103	WE-214	1	$141.65	...
P011	Gas regulator	S106	AU-100	1	$166.25	...
P012	Welding hoods	S102	SA-432	1	$42.37	...
P013	Spot welding electrode	S104	WE-212	1	$2.35	...
P014	Plasma cutter	S107	PL-100	1	$1,645.91	...
P015	Plasma cutter cutting tip	S107	PL-100	1	$9.27	...

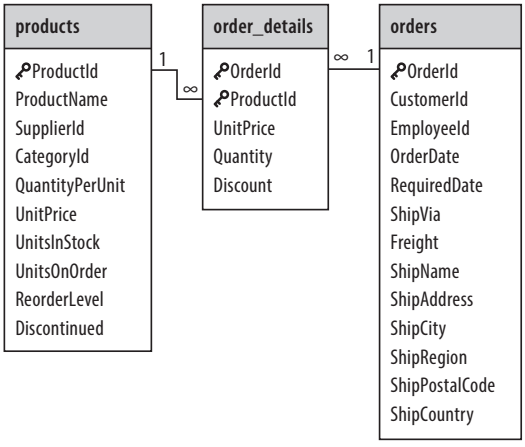
Suppliers

SupplierId	CompanyName	ContactName	ContactTitle	Address	City	etc.
S-101	Van Eck Industries	Bruce Davidson	Vp Operations	2158 Del Dew Drive	Temple Hills	...
S-102	Wright & Gate Co	Wilma Joy	Supply Chain Supervisor	291 Creekside Lane	Ventura	...
S-103	Olivias Supply	Brad Pence	Site Manager, Baton Rouge	4353 Locust View Drive	Baton Rouge	...
S-104	Cantor Corporation	Orville Bedford	President	2811 West Drive	Chicago	...
S-105	Bellagio Finland	Wallace Grim	Distributions Superviser	4939 Breezewood Court	Chanute	...
S-106	Decks Materials	John Tuck	VP Operations	4529 Counts Lane	Lextington	...
S-107	Lennor Co	Rachel Durst	Site Manager, Jackson	2216 Rhapsody Street	Gainesville	...

Identyczne dane `SupplierId` mogą pojawiać się w wielu rekordach tabeli *products*, ale już nie w tabeli *suppliers*.

Na rysunku 1.12 pokazałem relacje zachodzące między tabelami `products`, `order_details` i `orders`.

Rysunek 1.12. Relacje zachodzące między tabelami products, order_details i orders



Wydaje się, że tabela `order_details` zawiera dwa klucze podstawowe, na co wskazują dwie ikony przedstawiające klucz. W takich przypadkach mówimy o istnieniu ***złożonego klucza podstawowego***, czyli sytuacji, w której do zdefiniowania klucza podstawowego zostały użyte co najmniej dwie kolumny (rysunek 1.13). Wprawdzie formalnie rzecz biorąc mamy tutaj dwa klucze podstawowe, ale lepiej jest traktować je jako jeden — *klucz podstawowy*.

Rysunek 1.13. Przykład złożonego klucza podstawowego

Złożony klucz podstawowy

OrderId	ProductId	UnitPrice	Quantity	Discount
101	P006	$1,386.67	1	NULL
101	P003	$7.00	3	NULL
101	P005	$3.73	1	10%
102	P011	$166.23	1	NULL
102	P013	$2.35	1	NULL
103	P014	$1,645.91	1	NULL
104	P001	$12.99	3	NULL
104	P012	$42.37	3	NULL
104	P011	$166.23	2	10%
104	P003	$7.00	5	NULL
105	P010	$141.65	1	NULL
105	P004	$11.86	3	NULL
105	P003	$7.00	2	NULL
106	P014	$1,645.91	1	NULL

Połączenie danych znajdujących się w kolumnach użytych do zdefiniowania złożonego klucza podstawowego stanowi unikatowy identyfikator dla wszystkich rekordów tabeli. Innymi słowy jeśli wartością `OrderId` w tabeli `order_details` jest `101`, a wartość `ProductId` dla tego samego rekordu wynosi `P006`, wówczas można przyjąć złożenie, że żaden inny rekord w tabeli nie będzie miał takiego samego połączenia danych w obu wymienionych kolumnach. Może być wiele innych rekordów z kolumną `OrderId` o wartości `101` i wiele rekordów z kolumną `ProductId` o wartości `P006`, ale tylko jeden rekord będzie miał kolumny `OrderId` o wartości `101` *i* `ProductId` o wartości `P006`. Takie połączenie danych między kolumnami działa jako klucz podstawowy, który podobnie jak każdy inny klucz podstawowy stanowi unikatowy identyfikator rekordów tabeli.

Być może dostrzegasz, że standardowy klucz podstawowy w tabeli zwykle reprezentuje element „jeden" w relacji typu „jeden do wielu". Na przykład w przypadku tabeli `orders` można dostrzec, że klucz podstawowy, kolumna `OrderId`, dostarcza unikatowy identyfikator dla poszczególnych rekordów tabeli. Z kolei element „wiele" relacji można znaleźć w tabeli `order_details`. Jak sądzisz, dlaczego tak jest?

Zastanówmy się nad tym przez chwilę. Można wysnuć wniosek, że celem tabeli *order_details* jest dostarczenie informacji o różnych produktach zamawianych przez klientów. Tym samym można uznać, że dany produkt może być zamawiany wielokrotnie przez wielu różnych klientów w wielu odmiennych sytuacjach oraz w wielu różnych cenach itd. Dlatego też sama wartość `ProductId` nie może być kluczem podstawowym tabeli `order_details`. Kolejnym założeniem jest, że dane zamówienie może zawierać wiele produktów, i jeśli przyjrzymy się innym kolumnom w tabeli `order_details` — `UnitPrice`, `Quantity` i `Discount` — widać wyraźnie, że dotyczą one właściwości pojedynczego produktu, a nie całego zamówienia. Dlatego kolumna `OrderId` nie może być użyta sama jako klucz podstawowy w tabeli `order_detail`. Rozwiązaniem jest połączenie kolumn `ProductId` i `OrderId`, co prowadzi do powstania złożonego klucza podstawowego. Dzięki temu mamy pewność, że dane w kolumnach `UnitPrice`, `Quantity` i `Discount` odpowiadają unikatowej i konkretnej kolejności *oraz* unikatowemu i konkretnemu produktowi w zamówieniu.

Typy danych

We wcześniejszej części rozdziału wprowadziłem koncepcję metadanych, czyli danych opisujących ograniczenia i specyfikację formatowania dla innych danych znajdujących się w bazie danych. Podczas opracowywania bazy danych za pomocą języka SQL konkretne ***typy danych*** muszą być stosowane dla poszczególnych kolumn. Typy danych będą się różniły w zależności od używanej wersji SQL. Ogólnie rzecz biorąc, masz do dyspozycji liczbowe typy danych, znakowe (inaczej tekstowe) typy danych, datę i godzinę oraz wartości boolowskie. Przeanalizujemy pokrótce każdy z tych typów.

Liczbowe typy danych

Obejmują typ liczb całkowitych, czyli wartości pozbawionych części dziesiętnej. Podczas używania typu liczb całkowitych zwykle występuje pewne ograniczenie dotyczące ich maksymalnej wielkości. Przypomnij sobie przedstawioną wcześniej w tym rozdziale tabelę z informacjami o pacjentach. W przypadku kolumny `Weight` można rozważyć użycie typu danych liczb całkowitych z ograniczeniem do trzech cyfr. Dlaczego? Po pierwsze nie ma problemu z zaokrągleniem w dół lub w górę wagi (ang. *weight*) pacjenta do najbliższego kilograma. Po drugie z pewnością nie będziemy potrzebować więcej niż trzech cyfr do zapisania wagi pacjenta. Gdy dane w postaci liczb całkowitych nie wystarczają, potrzebny jest format pozwalający na znacznie dokładniejsze wyrażanie wartości. W takim przypadkach można skorzystać z liczb zmiennoprzecinkowych, które umożliwiają określenie części po przecinku dziesiętnym. Podobnie jak liczby całkowite także liczby zmiennoprzecinkowe mogą mieć ograniczoną wielkość. Przykłady obu typów liczb zamieściłem w tabeli 1.2.

Tabela 1.2. Przykłady liczb całkowitych i zmiennoprzecinkowych

Liczby całkowite	Liczby zmiennoprzecinkowe
5	30,5
6176	14,65
47261	5,634
531	365,1
90	0,437
1	15347,45

Typy danych pozwalające na użycie większej liczby cyfr, znaków itd. będą wymagały większej ilości pamięci masowej. SQL umożliwia również stosowanie walutowych typów danych.

Tekstowe typy danych

Tekstowy typ danych może przechowywać ciągi tekstowe o stałej bądź zmiennej liczbie znaków. Na przykład jeśli jedna z kolumn bazy danych przechowuje standardowe kody pocztowe stosowane w Kanadzie (zawierają zarówno cyfry, jak i litery), to

można użyć tekstowego typu znaków skonfigurowanego do przechowywania ciągu tekstowego o długości sześciu znaków. Jeżeli tworzysz kolumnę przeznaczoną do przechowywania imienia lub nazwiska klienta, to zdecydujesz się na ciąg tekstowy o zmiennej długości, definiując rozsądne wartości minimalną i maksymalną. Przykład takich danych pokazałem w tabeli 1.3.

Tabela 1.3. Przykład użycia tekstowych typów danych

CanadianZipCode	FirstName	LastName
L4K8R3	Ronald	Dalton
V0S0N2	Clara	Abramson
H7L9N0	Joseph	Scalia
L3M0L7	Benjamin	Dreadnaught
E6K5T8	Harold	Mercedes
E7K3C5	James	Rockefeller

Przedstawione dotychczas przykłady obejmowały względnie krótkie dane tekstowe, takie jak imiona, nazwiska i dane adresowe. Wiele baz danych zawiera kolumny pozwalające wstawiać znacznie dłuższe dane tekstowe. Niektóre struktury baz danych umożliwiają przechowywanie wielostronicowych dokumentów, a nawet całych książek.

Data i godzina

Dane w postaci daty i godziny są oczywiście ważne w wielu sytuacjach. SQL pozwala korzystać z różnych formatów takich danych: RRRR-MM-DD, RRRR-MM-DD GG:MM:SS, RR-MM-DD. Kolumna może przechowywać jedynie rok, w postaci czterech lub dwóch cyfr, np. `2019` lub tylko `19`. W tabeli 1.4 przedstawiłem przykłady danych w postaci daty i godziny.

Tabela 1.4. Typy danych w postaci daty i godziny

DateOfBirth	CreditCardExpiration	TimeOfDelivery
01/25/1977	08/2023	2019-04-21 08:25;55
09/30/2003	05/2025	2020-12-05 13:30;15
08/15/1999	01/2023	2020-05-10 22:20;36
02/25/1962	11/2022	2019-01-17 10:20;01
09/12/1998	05/2026	2021-06-29 15:21;59
11/03/1959	03/2023	2022-09-03 16:42;26

Formaty daty i godziny w SQL mają wbudowane wartości liczbowe umożliwiające bazie danych interpretowanie zapytań dotyczących danych wyjściowych w konkretnym porządku chronologicznym. Na przykład jeśli chcesz się dowiedzieć, ilu klientów kupiło określony produkt w okresie od 1 października 2020 roku do 31 grudnia 2020 roku, wówczas SQL pomoże wygenerować takie dane i je posortować.

Wartości boolowskie

Wartość ***boolowska*** jest wyrażana jako prawda (`True`) lub fałsz (`False`). Jeżeli odpowiadasz za tajne operacje dla rządu lub organizacji prywatnej, możesz wykorzystać bazę danych do monitorowania poziomu dostępu pracowników do danych. Jeżeli musisz wyszukać pracowników o dostępie na poziomie A, B lub D, ale niekoniecznie C, to analiza danych boolowskich może znacznie ułatwić pracę. Na rysunku 1.14 pokazałem przykład użycia danych boolowskich.

Różne wersje SQL mają odmienne listy obsługiwanych typów danych. Niektóre wersje SQL, np. SQL Server i MySQL (wspomnę o nich w dalszej części rozdziału), nie oferują użytkownikowi określania danych jako typu „boolowskiego". Zamiast niego dostarczają typ danych bitowych, który można łatwo uznać za odpowiednik typu boolowskiego.

Rysunek 1.14.
Przykład użycia danych boolowskich

Dane boolowskie

ClearedForTakeOff	InDefault	ConvictedFelon
True	False	False
False	True	False
False	False	False
True	True	True
False	True	False

Systemy relacyjnych baz danych

Język SQL jest używany w wielu pakietach oprogramowania znanych pod nazwą ***relacyjnych systemów zarządzania bazami danych*** (ang. *relational database management system*, ***RDBMS***). Ułatwiają one używanie SQL w aplikacjach wykonujących zapytania do baz danych. Do popularnych przykładów oprogramowania typu RDBMS zaliczamy Oracle Database, Microsoft SQL Server, MySQL, PostgreSQL, IBM Db2 i SQLite (rysunek 1.15).

Rysunek 1.15.
Logotypy najpopularniejszych systemów typu RDBMS

Dość często zdarza się, że oprogramowanie typu RDBMS jest określane jako baza danych. To jest błędne podejście. Ujmując rzecz dokładniej: oprogramowanie typu RDBMS zapewnia interfejs (zwykle określany mianem przeglądarki SQL) umożliwiający użytkownikowi pracę z danymi, które są przechowywane w bazie danych.

Część oprogramowania typu RDBMS celowo jest wyposażona w graficzny interfejs użytkownika, z kolei inne są bardziej tekstowe. Systemy RDBMS różnią się również w podejściu do języka SQL. O jednej z takich różnic wspomniałem już wcześniej i dotyczyła ona kwestii obsługi wartości boolowskich. Ponadto systemy RDBMS w odmienny sposób mogą przedstawiać informacje przechowywane w bazie danych.

Fakt wskazywania systemowi RDBMS typu informacji przedstawianych użytkownikowi definiuje SQL jako „deklaratywny" język programowania. To odróżnia go od innych języków programowania, z którymi być może już masz doświadczenie, np. C++ i Java. Te języki są bardziej proceduralne pod tym względem, że obsługują tworzenie i uruchomienie programu od początku do końca (alokacja pamięci, dołączenie istniejących plików odniesień itd.). Natomiast w przypadku SQL wszystkie zadania związane z alokacją pamięci oraz z innymi kwestiami proceduralnymi są obsługiwane przez RDBMS.

Zapytanie SELECT

Jak już wcześniej wspomniałem, SQL to strukturalny język zapytań (ang. *structured query language*) będący od kilkudziesięciu lat standardem komunikacji z relacyjnymi bazami danych. Najczęściej wydawanym poleceniem jest niewątpliwie `SELECT`, które dość dokładnie poznasz w rozdziale 4. i będziesz go często używać w pozostałej części książki. Zapytanie SQL zwykle składa się ze słowa kluczowego `SELECT` w połączeniu z innymi słowami kluczowymi SQL oraz odwołaniami do danych w zapytaniu. Podobnie jak w innych językach programowania także w SQL prawidłowa kolejność i wybór słów kluczowych mają istotne znaczenie, aby zapytanie zostało poprawnie zinterpretowane przez przeglądarkę SQL. Struktura, którą należy stosować, jest znana jako ***składnia*** zapytania.

W kolejnym przykładzie zobaczysz, jak składnia może się różnić w poszczególnych implementacjach systemów typu RDBMS. Mamy tutaj dwa bardzo proste zapytania, które wykonują dokładnie to samo zadanie (zwracają pierwsze 10 rekordów tabeli *products*), ale jak widać, ich składnie są nieco odmienne.

W SQL Server trzeba zwykle wykonać następujące zapytanie:

```
SELECT TOP 10 *
FROM
  products;
```

Natomiast w MySQL zapytanie ma taką postać:

```
SELECT *
FROM
  products
LIMIT 10;
```

Jeżeli w MySQL zostanie użyta struktura stosowana na przykład w SQL Server, wówczas przeglądarka SQL wygeneruje tzw. ***błąd składni***, uniemożliwiający wykonanie zapytania. W omawianym przykładzie jedyna różnica między tymi zapytaniami dotyczy sposobu

ograniczenia danych wyjściowych do pierwszych 10 rekordów. Natomiast pozostała część zapytania jest taka sama. Różnice między systemami typu RDBMS są, ogólnie rzecz biorąc, niewielkie, a ilość odmiennego kodu w zapytaniu zazwyczaj wynosi poniżej 10%. Prosta i deklaratywna natura języka SQL jest w miarę spójna w większości systemów RDBMS. Dlatego jeśli chcesz poznać logikę SQL w wybranym systemie RDBMS, tę wiedzę możesz później wykorzystać do szybkiego rozpoczęcia pracy w innym systemie RDBMS.

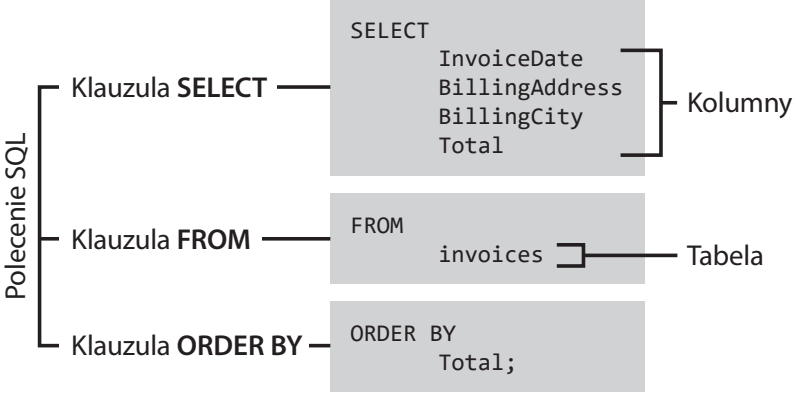
Zapytania, polecenia, klauzule i słowa kluczowe

Jeżeli masz doświadczenie w pracy z SQL, prawdopodobnie wiesz, że te cztery słowa są używane wymiennie: zapytanie, polecenie, klauzula i słowo kluczowe. `SELECT` to specjalne słowo kluczowe w języku SQL, przy czym jest określane jako polecenie `SELECT`, klauzula `SELECT` lub zapytanie `SELECT`. Na czym polega różnica? Zacznę od najobszerniejszego pojęcia, a następnie przejdę do bardziej szczegółowych.

W najprostszej postaci ***zapytanie*** to żądanie, którego wynikiem są zwrócone przez bazę danych informacje w postaci rekordów. Zapytanie może składać się z wielu poleceń SQL (do tego powrócę w rozdziale 8. podczas omawiania podzapytań). ***Polecenie*** SQL to poprawny fragment kodu wykonywany przez system RDBMS. Przykładowe fragmenty kodu, które wcześniej przedstawiłem, to poprawne polecenia SQL (ponieważ są wykonywane przez system RDBMS) i zapytania (ponieważ zwracają zbiór rekordów). ***Klauzula*** to podsekcja zapytania, zawierająca przynajmniej jedno ***słowo kluczowe*** i odpowiednie informacje używane w połączeniu z tym słowem kluczowym (w przypadku odwołań do kolumn i tabel).

Jak widać na rysunku 1.16, polecenie SQL może składać się z wielu klauzul, z których każda zawiera co najmniej jedno słowo kluczowe, a także odwołania do kolumn i tabel.

Rysunek 1.16. Struktura polecenia SQL



Słowa zapisane WIELKIMI LITERAMI to słowa kluczowe SQL.

Podobnie jak we wcześniejszym przykładzie także na rysunku 1.16 mamy pełne polecenie i zapytanie SQL. Zapytanie może zawierać wiele klauzul, z których każda rozpoczyna się słowem kluczowym.

Wprowadzenie do SQLite

Skoro znasz już podstawową architekturę bazy danych i wiesz, jak można z nią pracować, warto zmienić bieg i przejść do bardziej praktycznego podejścia z wykorzystaniem rzeczywistych problemów. Jak wcześniej wspomniałem, istnieje wiele różnych systemów RDBMS. Byłoby wysoce nieefektywne w tym kontekście dokładne ich porównywanie funkcja po funkcji i omawianie niuansów poszczególnych rozwiązań typu RDBMS. Zamiast tego zdecydowałem się na użycie SQLite jako oficjalnego systemu RDBMS w książce. To dość dobry i praktyczny wybór dla początkujących. SQLite to produkt open source, więc można go używać bezpłatnie do dowolnego celu. Praktycznie 99% wiedzy zdobytej podczas poznawania SQLite przydaje się później w pracy z innymi systemami typu RDBMS. SQLite to również jeden z najczęściej używanych systemów RDBMS — znajduje zastosowanie w komputerach, urządzeniach mobilnych, a nawet w pojazdach[2]. Więcej informacji na temat SQLite i dokumentację tego systemu znajdziesz pod adresem *https://www.sqlite.org/index.html*.

Na rysunku 1.17 pokazałem logotypy wybranych z najbardziej znanych firm, które korzystają z SQLite.

Rysunek 1.17. Logotypy kilku najbardziej znanych firm, które używają SQLite

Lite w nazwie SQLite nie odnosi się do mniejszych możliwości oprogramowania, a raczej do faktu, że jest ono *lżejsze* podczas konfiguracji, późniejszej administracji, a także pod względem ilości zużywanych zasobów.

2 „Most Widely Deployed SQL Database Engine — SQLite", *https://www.sqlite.org/ mostdeployed.html*

Podsumowanie

- Tabela to dwuwymiarowa siatka wierszy i kolumn zawierających dane.
- Dane mogą być przechowywane za pomocą wielu różnych typów, takich jak ciągi tekstowe, liczby i znaki specjalne.
- Metadane opisują naturę i format danych, m.in. minimalną i maksymalną wielkość danych, a także wymogi dotyczące użycia liczb, liter i znaków specjalnych.
- Relacyjna baza danych może zawierać wiele tabel. Każda tabela w relacyjnej bazie danych powinna mieć klucz podstawowy służący jako unikatowy identyfikator dla tej tabeli.
- Klucz zewnętrzny to dowolna tabela kolumny, która w innej tabeli jest kluczem podstawowym.
- Relacja między tabelami oraz ich kluczami podstawowymi i zewnętrznymi jest nazywana schematem bazy danych. Graficznie można ją przedstawić za pomocą wykresu ERD, który służy jako matryca dla bazy danych.
- Istnieje wiele różnych relacyjnych systemów zarządzania bazami danych (RDBMS), np. Oracle Database, Microsoft SQL Server, MvSQL, PostgreSQL, IBM Db2 i SQLite. Wprawdzie różnią się one pod wieloma względami, ale używają tego samego strukturalnego języka zapytań.
- Słowo kluczowe `SELECT` to najczęściej stosowane polecenie SQL w zapytaniach SQL.
- Polecenie SQL może zawierać wiele klauzul używających różnych słów kluczowych.
- W książce będziemy używać SQLite. Zdobytą wiedzę można łatwo wykorzystać na innych platformach RDBMS.

SQL. Przepis na big data w bardzo prostych słowach!

Jak wszystko inne, tak i zarządzanie danymi podlega różnym modom i trendom. Mimo to eksperci w tej dziedzinie wciąż korzystają z SQL – języka programowania, który od dziesięcioleci służy do pracy z relacyjnymi bazami danych. Właśnie SQL jest „złotym standardem" pracy ze zbiorami danych, a umiejętność posługiwania się tym językiem to ceniony atut w wielu zawodach technicznych, w tym związanych z projektowaniem i tworzeniem oprogramowania, a także testowaniem i analizą biznesową.

Ten przewodnik sprawi, że nauka zarządzania relacyjną bazą danych będzie łatwiejsza. Szczególnie docenią go czytelnicy myślący o zmianie ścieżki zawodowej i rozpoczęciu pracy z obsługą danych. Opisano tu przede wszystkim podstawowe narzędzia SQL potrzebne do zrozumienia i wyodrębnienia przydatnych informacji z istniejących baz danych. Omówiono też zasady dodawania, modyfikowania i usuwania rekordów z bazy danych i zaprezentowano potrzebne do tego zapytania SQL. Sporo miejsca poświęcono zaawansowanym zagadnieniom tworzenia poleceń SQL, w tym: funkcjom, zapytaniom zagnieżdżonym, widokom i sposobom pobierania danych z wielu tabel równocześnie.

Dzięki książce:

- poznasz zasady działania baz danych
- nauczysz się używać języka SQL do pobierania danych
- poznasz najważniejsze zapytania SQL i dowiesz się, jak je mądrze stosować
- przeanalizujesz przykład profesjonalnej aplikacji SQL
- otrzymasz garść przydatnych porad dotyczących Twojej dalszej kariery

Walter Shields

od ponad dwudziestu lat pracuje z danymi. Jest przedsiębiorcą i autorem książek. Wcześniej był cierpliwym wykładowcą niecierpliwych studentów. Zajmował się danymi m.in. w Target Corporation i NYC Transit Authority. Dziś jego praca polega na pomocy średnim i dużym firmom w profesjonalnej wizualizacji 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 978-83-8322-657-6

9 788383 226576

Cena: 59,00 zł