

Helion 

JOE CELKO
PRAKTYKI
MISTRZA
SQL

PROGRAMOWANIE
ZAAWANSOWANE

MK
MORGAN KAUFMANN

Tytuł oryginału: Joe Celko's SQL for Smartie: Advanced SQL Programming, 5th Edition

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-283-2530-2

Copyright © 2015, 2011, 2005, 2000, 1995 Elsevier Inc.
All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

This edition of Joe Celko's SQL for Smartie: Advanced SQL Programming, 5th Edition by Joe Celko is published by arrangement with ELSEVIER INC., a Delaware corporation having its principal place of business at 360 Park Avenue South, New York, NY 10010, USA.

Translation copyright © 2016 Helion SA

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

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

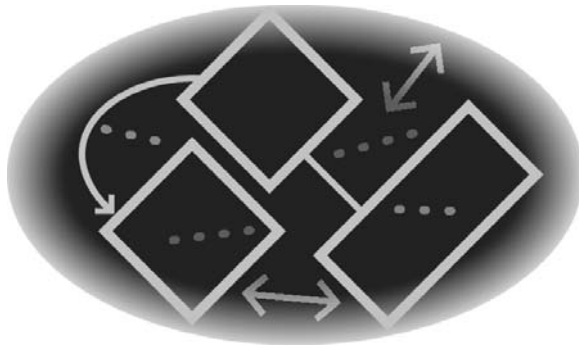
Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/pmsqlp.zip>

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

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)



Spis treści

Wprowadzenie do wydania piątego	13
Część I. Mechanizmy związane z deklarowaniem danych	15
1. Bazy danych a systemy plików	17
1.1. Instrukcja Schema	19
1.2. Tabele jako encje	24
1.3. Tabele jako relacje	24
1.4. Wiersze a rekordy	26
1.5. Kolumny a pola	27
2. Transakcje i kontrola współbieżności	31
2.1. Sesje	32
2.2. Transakcje i model ACID	32
2.3. Kontrola współbieżności	34
2.4. Poziomy izolacji	36
2.5. Pesymistyczna kontrola współbieżności	38
2.6. Optymistyczna kontrola współbieżności za pomocą izolacji snapshotów	39
2.7. Logiczna kontrola współbieżności	42
2.8. Twierdzenie CAP	42
2.9. Model BASE	43
2.10. Spójność po stronie serwera	45
2.11. Obsługa błędów	46
2.12. Pasywne i aktywne zakleszczenie	46



3. Tabele	49
3.1. Instrukcja CREATE TABLE	50
3.2. Definicje kolumn	52
3.3. Kolumny obliczane	63
3.4. Ograniczenia [NOT] DEFERRABLE	64
3.5. Instrukcje CREATE DOMAIN i CREATE SEQUENCE	65
3.6. Konstrukcje związane z zestawem znaków	72
4. Klucze, lokalizatory i generowane wartości	75
4.1. Typy kluczy	76
4.2. Praktyczne wskazówki związane z denormalizacją	79
5. Normalizacja	85
5.1. Zależności funkcyjne i wielowartościowe	88
5.2. Pierwsza postać normalna (1NF)	88
5.3. Druga postać normalna (2NF)	93
5.4. Trzecia postać normalna (3NF)	94
5.5. Postać normalna z kluczem podstawowym (EKNF)	95
5.6. Postać normalna Boyce'a-Codda (BCNF)	96
5.7. Czwarta postać normalna (4NF)	98
5.8. Piąta postać normalna (5NF)	99
5.9. Postać normalna z kluczem dziedziny (DKNF)	101
5.10. Praktyczne wskazówki z zakresu normalizacji	108
5.11. Nadmiarowość, gdy nie jest używana postać normalna	109
6. Widoki, tabele pochodne i inne tabele wirtualne	115
6.1. Widoki w kwerendach	116
6.2. Widoki modyfikowalne i tylko do odczytu	117
6.3. Typy widoków	118
6.4. Obsługa widoków w silnikach bazodanowych	124
6.5. Klauzula WITH CHECK OPTION	128
6.6. Usuwanie widoków	133
6.7. Tabele ze zmaterializowanymi wynikami kwerend	134
7. Tabele pomocnicze	137
7.1. Tabela Series	138
7.2. Pomocnicze tabele wyszukiwania	143
7.3. Zaawansowane tabele pomocnicze z funkcjami	153
7.4. Globalne tabele stałych	164
7.5. Uwagi na temat przekształcania kodu proceduralnego na tabele	171
8. Inne obiekty schematu	177
8.1. Instrukcja CREATE SCHEMA	179
8.2. Tabele schematu	180
8.3. Tabele tymczasowe	180
8.4. Instrukcja CREATE ASSERTION	180
8.5. Instrukcja CREATE DOMAIN	181



8.6. Instrukcja CREATE COLLATION	182
8.7. Instrukcja CREATE TRANSLATION	183
8.8. Instrukcja CREATE PROCEDURE	183
8.9. Wyzwalacze	183
8.10. Model działania wyzwalaczy	184
9. Błędy, których należy unikać w instrukcjach DDL	193
9.1. Stosowanie skrótu „tbl” i powiązane błędy	193
9.2. Podział atrybutów	196
9.3. Problemy z przeciążaniem projektu	197
9.4. Nadmiarowość wynikająca z braku postaci normalnej	201
Część II. Typy danych	205
10. Dane liczbowe w SQL-u	207
10.1. Dokładne liczbowe typy danych	208
10.2. Przybliżone liczbowe typy danych	213
10.3. Konwersje typów liczbowych	215
10.4. Arytmetyka oparta na czterech funkcjach	217
10.5. Przekształcanie wartości na NULL i dokonywanie odwrotnych konwersji	219
10.6. Funkcje matematyczne	222
10.7. Adresy IP	226
11. Znakowe typy danych w SQL-u	229
11.1. Problemy z łańcuchami znaków w SQL-u	230
11.2. Standardowe funkcje znakowe	232
11.3. Często dostępne rozszerzenia od producentów	234
11.4. Tablice Cuttera	242
12. SQL-owe typy danych związane z czasem	243
12.1. Uwagi na temat standardów związanych z kalendarzem	244
12.2. Natura modeli danych związanych z czasem	247
12.3. SQL-owe typy danych związane z czasem	248
12.4. Typy danych z rodziny INTERVAL	255
12.5. Kwerendy z operacjami arytmetycznymi na datach	256
12.6. Stosowanie wartości NULL do oznaczania „wieczności”	257
12.7. Predykat OVERLAPS()	258
12.8. Ograniczenia związane ze zmianami stanu	260
12.9. Tabele z kalendarzem	265
13. Wielokolumnowe elementy danych	269
13.1. Elementy danych w postaci wektorów i współrzędnych	269
13.2. Hierarchiczne elementy danych	276
14. Wartości NULL — brakujące dane w SQL-u	283
14.1. Puste i brakujące tabele	285
14.2. Brakujące wartości w kolumnach	285



14.3. Kontekst i brakujące wartości	287
14.4. Porównywanie wartości NULL	288
14.5. Wartości NULL i logika	289
14.6. Wartości NULL a obliczenia matematyczne	293
14.7. Funkcje związane z wartościami NULL	294
14.8. Wartości NULL a języki główne	294
14.9. Wskazówki projektowe związane z wartościami NULL	295
14.10. Uwagi na temat różnych rodzajów wartości NULL	298
15. Operacje na tabelach	301
15.1. Instrukcja DELETE FROM	301
15.2. Instrukcja INSERT INTO	307
15.3. Instrukcja UPDATE	309
15.4. Uwaga na temat błędów w popularnych rozszerzeniach producentów	317
15.5. Instrukcja MERGE	318
16. Operacje na zbiorach	321
16.1. UNION i UNION ALL	322
16.2. Operatory INTERSECT i EXCEPT	325
16.3. Uwagi na temat opcji ALL i SELECT DISTINCT	330
16.4. Równość i podzbiory właściwe	331
Część III. Mechanizmy z poziomu wierszy i kolumn	333
17. Operatory porównywania (operatory theta)	335
17.1. Konwersja typów danych	336
17.2. Porównywanie wierszy w SQL-u	338
17.3. Operator IS [NOT] DISTINCT FROM	340
17.4. Operatory monadyczne	341
18. Predykaty dotyczące wyników podkwerend	345
18.1. Predykat UNIQUE	345
18.2. Predykat [NOT] IN()	347
18.3. Predykat [NOT] EXISTS()	357
18.4. <operator porównania> [SOME ANY] <podkwerenda>	365
18.5. <operator porównania> ALL <podkwerenda>	365
19. Predykaty BETWEEN i OVERLAPS	369
19.1. Predykat BETWEEN	369
19.2. Predykat OVERLAPS	372
20. Rodzina wyrażeń CASE	383
20.1. Wyrażenie CASE	383
20.2. Wyrażenia w postaci podkwerend i stałe	391
21. Predykaty LIKE i SIMILAR TO	393
21.1. Sztuczki związane ze wzorcami	394
21.2. Wyniki dla wartości NULL i pustych łańcuchów znaków	396



21.3. Operator LIKE nie oznacza równości	396
21.4. Rozbudowywanie predykatu LIKE za pomocą złączenia	396
21.5. Wyrażenia CASE i predykaty LIKE	397
21.6. Predykat SIMILAR TO	397
21.7. Sztuczki związane z łańcuchami znaków	399
22. Podstawowa instrukcja SELECT	403
22.1. Wyrażenia CTE	404
22.2. Klauzula FROM	405
22.3. Klauzula WHERE	405
22.4. Klauzula GROUP BY	406
22.5. Klauzula HAVING	406
22.6. Klauzula SELECT	407
22.7. Klauzula ORDER BY	407
22.8. Wyrażenie z zagnieżdżonymi kwerendami a ortogonalność	407
23. Podstawowe funkcje agregujące	409
23.1. Funkcje z rodziny COUNT()	410
23.2. Funkcja SUM()	414
23.3. Funkcja AVG()	415
23.4. Funkcje zwracające ekstrema	419
23.5. Funkcja agregująca LIST()	427
23.6. Funkcja agregująca wyznaczająca dominantę	429
23.7. Funkcja agregująca zwracająca medianę	430
23.8. Funkcja agregująca PRD()	432
24. Zaawansowane statystyki opisowe	437
24.1. Funkcje dla tabel dwukolumnowych	437
24.2. Korelacja	439
25. Używanie SQL-a do agregacji w systemach OLAP	441
25.1. Kwerendy a raporty	441
25.2. Operatory grupowania	442
25.3. Klauzula okna	445
25.4. Funkcje agregujące oparte na oknie	449
25.5. Funkcje porządkowe	449
25.6. Rozszerzenia udostępniane przez producentów	452
25.7. Kartka z historii	456
26. Zaawansowane instrukcje SELECT	459
26.1. Podkwerendy skorelowane	459
26.2. Wrostkowe złączenia wewnętrzne	464
26.3. Złączenia zewnętrzne	465
26.4. Operator UNION JOIN	477
26.5. Skalarne wyrażenia SELECT	479
26.6. Dawna i nowa składnia złączeń	480
26.7. Złączenia z ograniczeniami	481
26.8. Złączenia T dr. Codda	490



26.9. Brakujące wartości w danych	496
26.10. Brakujące i mieszane dane w wierszach	501

Część IV. Struktury danych w SQL-u 503

27. Grafy w SQL-u 505

27.1. Podstawowe cechy grafów	506
27.2. Ścieżki w grafie	510
27.3. Grafy acykliczne jako zbiory zagnieżdżone	517
27.4. Model wykorzystujący macierz sąsiedztwa	519
27.5. Punkty w wielokącie	520
27.6. Geometria taksówkowa	521
27.7. Klasy równoważności i klikli	524
27.8. Podsumowanie	532

28. Drzewa i hierarchie w SQL-u 533

28.1. Listy sąsiedztwa	534
28.2. Znajdowanie korzenia	535
28.3. Znajdowanie liści	536
28.4. Określanie poziomów w drzewie	536
28.5. Operacje na drzewach	537
28.6. Model oparty na zbiorach zagnieżdżonych	538
28.7. Znajdowanie korzenia i liści	541
28.8. Wyszukiwanie poddrzew	541
28.9. Wyszukiwanie poziomów i ścieżek w drzewie	542
28.10. Funkcje w modelu wykorzystującym zbiory zagnieżdżone	545
28.11. Usuwanie wierzchołków i poddrzew	546
28.12. Przegląd funkcji dotyczących drzew	548
28.13. Wstawianie i aktualizowanie elementów drzew	554
28.14. Przekształcanie list sąsiedztwa na zbiory zagnieżdżone	558
28.15. Przekształcanie zbiorów zagnieżdżonych na listy sąsiedztwa	558
28.16. Porównywanie wierzchołków i struktur	559

29. Kolejki 563

29.1. Podstawowe instrukcje DDL	563
29.2. Procedury Enqueue, Dequeue i Empty	564
29.3. Zmienianie uporządkowania	565
29.4. Kolejki i matematyka	566
29.5. Kolejki priorytetowe	567
29.6. Kolejki FIFO i LIFO	568

30. Macierze w SQL-u 573

30.1. Tablice oparte na nazwanych kolumnach	574
30.2. Tablice oparte na kolumnach z indeksem	577
30.3. Operacje na macierzach w SQL-u	579
30.4. Spłaszczanie tabeli do postaci tablicy	583



30.5. Porównywanie tablic zapisanych jako tabele	585
30.6. Inne operacje na macierzach	587

Część V. Typowe kwerendy 589

31. Partycjonowanie i agregowanie danych w kwerendach 591

31.1. Pokrycie i partycje	591
31.2. Zaawansowane grupowanie, agregowanie z uwzględnieniem okien i funkcje OLAP w SQL-u	596

32. Podsekwencje, obszary, serie, luki i wyspy 613

32.1. Znajdowanie podobszarów o wielkości (n)	614
32.2. Numerowanie obszarów	615
32.3. Wyszukiwanie obszarów o maksymalnej wielkości	617
32.4. Kwerendy z granicami	621
32.5. Kwerendy dotyczące serii i sekwencji	621
32.6. Sumowanie ręcznie generowanych sekwencji wartości	626
32.7. Przesławianie i przesuwawanie wartości na liście	629
32.8. Eliminowanie luk na listach liczb	630
32.9. „Zawijawanie” listy liczb	630
32.10. Pokrycia	631
32.11. Klasy równoważności i kliki	635

33. Aukcje 645

33.1. Rodzaje ofert	645
33.2. Typy aukcji	646
33.3. Stosowanie modeli LIFO i FIFO w magazynach	648
33.4. Problem pakowania	653

34. Dzielenie relacji 655

34.1. Dzielenie z resztą	656
34.2. Dzielenie bez reszty	658
34.3. Uwaga na temat wydajności	659
34.4. Dzielenie Todda	659
34.5. Dzielenie z użyciem złączeń	662
34.6. Dzielenie z wykorzystaniem operatorów zbiorów	662
34.7. Dzielenie Romleya	663
34.8. Wyrażawia logiczne w dzieleniu relacji	666

35. Kwerendy związane z czasem 669

35.1. Obliczenia matematyczne na czasie	670
35.2. Kalendarze	674
35.3. Szeregi czasowe	677
35.4. Daty w formacie juliańskim	691
35.5. Inne funkcje związane z czasem	694
35.6. Okresy obejmujące wiele dni	694
35.7. Reprezentowanie czasu w tabelach	698



35.8. Funkcje LEAD() i LAG()	700
35.9. Problemy z rokiem 2000 — przegląd historyczny	701

Część VI. Kwestie związane z implementacją i kodowaniem 707

36. Programowanie proceduralne, mieszane i deklaratywne w SQL-u 709

36.1. Słowa mają znaczenie	710
36.2. Porządkowanie kodu	711

37. Poziomy zagnieżdżenia w SQL-u 717

37.1. Tabele pochodne	718
37.2. Reguły tworzenia nazw kolumn	719
37.3. Reguły określania zasięgu	720
37.4. Dostępne nazwy tabel	722
37.5. Wyrażenia CTE	723
37.6. Tabele z klauzulą LATERAL	723
37.7. Wskazówki programistyczne	725

38. Zagnieżdżony SQL, dynamiczny SQL z interfejsem CLI oraz SQL/PSM 727

38.1. Zagnieżdżony SQL	728
38.2. SQL/CLI	729
38.3. Dynamiczny SQL	729
38.4. Historia standardu SQL/PSM	729
38.5. Parametry w formacie CSV	738

Skorowdz 744



Błędy, których należy unikać w instrukcjach DDL

POZNAŁEŚ JUŻ podstawowe narzędzia do budowania instrukcji DDL potrzebnych w schemacie. Jednak sama znajomość narzędzi nie oznacza jeszcze, że wiesz, jak się nimi posługiwać. Możesz zapytać o to każdego, kto próbował samodzielnie zajmować się hydrauliką.

9.1. Stosowanie skrótu „tbl” i powiązane błędy

W czasach kart dziurkowanych i taśm magnetycznych identyfikatory były krótkie i musiały obejmować metadane dla prostych kompilatorów i systemów wykonujących powtarzalne instrukcje. Klasycznym przykładem były języki FORTRAN I i FORTRAN II. Były one oparte na algebrze i używały instrukcji GOTO do zarządzania przepływem sterowania. Występowały w nich tylko dwa liczbowe typy danych — INTEGER i REAL. Nazwy zmiennych miały do sześciu znaków, musiały rozpoczynać się literą i być unikatowe w programie. Deklarowano je, stosując je w instrukcji, a kompilator musiał wywnioskować typ danych na podstawie nazwy. Zmienne rozpoczynające się od liter od I do N reprezentowały liczby całkowite (typ INTEGER), wszystkie pozostałe nazwy oznaczały liczby zmiennoprzecinkowe (typ REAL).



Metadane w przedrostkach nazw elementów danych występowały także w systemach operacyjnych, gdzie sprzęt (na przykład napęd taśmowy lub dyskowy) miał zapisane na stałe nazwy w postaci skrótów literowych lub liczbowych. Nie istniały wtedy nowoczesne kompilatory i nie rozdzielano abstrakcyjnej postaci programu od jego implementacji.

Jednak także obecnie stosowane są przedrostki określające typy danych, na przykład `vch_` (typ `VARCHAR(n)`), `int_` (typ `INTEGER`) itd. Angielskie określenie *tibble* (czyli stosowanie skrótu `tbl`) pochodzi od przedrostka `tbl_` w nazwach tabel. Czasem pojawiają się też przedrostki `vw_` (od *Volkswagen Views*, czyli widoki firmy Volkswagen) i jeszcze gorsze nazwy.

9.1.1. Standard ISO-11179

Obecnie dostępne są standard ISO-11179 i standardy rozwijane przez jednostkę DM32.8 Meta Data w organizacji *INCITS*. Zgodnie z tymi standardami należy *nazywać rzeczy zgodnie z ich naturą*. Oto co to oznacza:

1. Nie należy podawać typów danych (czyli sposobu fizycznego zapisu danych).
2. Nie należy stosować przedrostków `pk_` ani `fk_` określających sposób używania danego elementu w tabeli (jako klucza głównego lub obcego).
3. Nie należy stosować nazw w formacie „tabela-kolumna”, określających lokalizację danego wystąpienia elementu danych.
4. Tabele to zbiory, dlatego jako ich nazwy należy stosować rzeczowniki zbiorowe lub w liczbie mnogiej. To oznacza, że nazwa `Pracownik` jest zła. `Pracownicy` to lepsza wersja, natomiast właściwe określenie to `Personel`. Podobnie `Drzewo` to zły wybór, `Drzewa` to lepsze rozwiązanie, a `Las` to poprawna forma. Zapewne już rozumiesz ten wzorzec.
5. Nazwa elementu danych może obejmować nazwę atrybutu, podkreślenie i cechę atrybutu. Jeśli ten sam element występuje w jednej tabeli wielokrotnie, jego nazwę można poprzedzić nazwą roli.

Zgodnie ze standardem ISO-11179 nazwy elementów danych powinny mieć format `[<rola>_]<atrybut>_<cecha>`. W przykładzie z pierwotnego dokumentu opisano atrybut `tree` z cechami takimi jak `tree_diameter`, `tree_species` i tak dalej. Niektóre cechy nie dotyczą niektórych atrybutów. Na przykład nie trzeba reprezentować cechy `employee_diameter`, a cecha `employee_species` może okazać się obraźliwa. Cechy atrybutów związane z wartościami ze skal potencjalnie można wykorzystać w tabelach wyszukiwania. Poniżej znajdziesz listę i definicje podstawowych przyrostków, które opisałem w książce *SQL Programming Style*.

- ◆ Przyrostek `_id` oznacza identyfikator. Jest on unikatowy w schemacie i wszędzie tam, gdzie występuje w schemacie, określa jedną *encję*. W tabeli wyszukiwania występują atrybuty i ich wartości, a nie encje, dlatego z definicji takie identyfikatory nie występują w tego typu tabelach. To dlatego określenia takie jak `_category_id` czy `_type_id` to bardzo złe nazwy. Nigdy nie stosuj określenia `<nazwa_tabeli>_id`, które jest nazwą



opartą na lokalizacji i informuje, że kolumna prawdopodobnie nie zawiera prawdziwych kluczy. Samo `id` to zbyt niejasna nazwa, aby była dla kogokolwiek przydatna. Takie nazwy zanieczyszczają słownik danych, gdy trzeba sprawdzać mnóstwo takich samych nazw oznaczających różne elementy danych (często o identycznym, za dużym typie).

- ◆ Przyrostek `_date` (lub `dt`) oznacza wymiar czasowy. Dane oznaczają wtedy datę czegoś — zatrudnienia, urodzin, zwolnienia itd. Nie należy tworzyć kolumn o nazwie w postaci samego `date`.
- ◆ Przyrostek `_nbr` (lub `num`) oznacza numer identyfikacyjny. Jest to łańcuch cyfr (lub znaków alfanumerycznych) określający nazwę czegoś. Nie stosuj przyrostka `_no`, ponieważ wygląda on jak wartość logiczna (`yes` lub `no`). Ja chętniej stosuję `nbr` niż `num`, ponieważ jest to skrót często stosowany w kilku językach europejskich.
- ◆ Przyrostek `_name` lub `_nm` to nazwa składająca się z liter i nie wymaga objaśnień. Czasem zbiór takich nazw jest nazywany skalą nominalną.
- ◆ Przyrostek `_code` lub `_cd` oznacza standardowy kod opracowany przez zaufane źródło poza firmą. Na przykład kod pocztowy jest określany przez Poczta Polską. Kod jest zrozumiały w określonym kontekście, dlatego nie zawsze trzeba go przekształcać, gdy jest wyświetlany ludziom.
- ◆ Przyrostek `_size` związany jest ze standardem branżowym lub firmowym i służy do określania wielkości towarów (na przykład ubrań, butów, kopert lub śrub). Zwykle istnieje definiujący rozmiary prototyp przechowywany przez zaufane źródło.
- ◆ Przyrostek `_seq` oznacza sekwencję (z numerami porządkowymi). Nie jest tym samym co numer identyfikacyjny, ponieważ nie może obejmować luk. Określone są też reguły dotyczące następników w sekwencji.
- ◆ Przyrostek `_status` określa wewnętrzne kodowanie opisujące stan, który może wynikać z wielu czynników. Na przykład kolumna `credit_status` może być obliczana na podstawie różnych źródeł. Słowo „status” pochodzi od słowa „state”. Można założyć, że istnieją określone dozwolone zmiany stanu. Na przykład stan cywilny może się zmienić na „rozwiędziony” tylko wtedy, gdy obecny stan to „żonaty”.
- ◆ Przyrostek `_cat` określa kategorię, czyli kodowanie oparte na zewnętrznym źródle z odrębnymi grupami encji. Do ustalania przynależności do kategorii powinny służyć bardzo wyraźne formalne kryteria. Przykładem jest klasyfikacja królestw w biologii.
- ◆ Przyrostek `_class` oznacza wewnętrzne kodowanie bez zewnętrznego źródła, opisujące podkategorię encji. Klasyfikacja powinna odbywać się na podstawie wyraźnych kryteriów formalnych. Przykładem jest tu klasyfikacja roślin w biologii.
- ◆ Przyrostek `_type` określa kodowanie, które wewnętrznie i zewnętrznie ma to samo znaczenie. Typy są zwykle mniej formalne od klas i mogą się pokrywać. Na przykład prawo jazdy można otrzymać na motor, samochód osobowy, taksówkę, ciężarówkę itd.



Różnice między typami, klasami i kategoriami dotyczą rosnącej ścisłości algorytmu przypisującego encje do danego typu albo określonej klasy lub kategorii.

Kategorie są bardzo jednoznaczne. Rzadko trzeba zgadywać, czy coś jest zwierzęciem, warzywem, czy minerałem, by przypisać dany obiekt do jednej z tych kategorii.

Klasa to zestaw obiektów o pewnych cechach wspólnych. Istnieją reguły pozwalające stwierdzić, że zwierzę jest ssakiem lub gadem. W niektórych sytuacjach reguły nie są oczywiste (na przykład w przypadku składających jaja ssaków australijskich), przy czym wyjątki często są zaliczane do odrębnej kategorii (tu są nią stekowce).

Typy są najmniej ściśle spośród trzech omawianych sposobów klasyfikowania i mogą wymagać subiektywnej oceny. Na przykład w niektórych stanach trójkołowce są uznawane za motocykle, natomiast w innych — za samochody. W jeszcze innych stanach trójkołowce są traktowane jak samochody tylko wtedy, jeśli mają bieg wsteczny.

W praktyce wszystkie trzy pojęcia często stosuje się w różnych kontekstach. Na przykład określanie wartości kolumny `blood_type` (z grupą krwi) odbywa się w wyniku procedury laboratoryjnej i ma prowadzić do jednoznacznego ustalenia wartości ze zbioru {A, B, AB, 0}. Zachowaj zgodność ze standardem branżowym, nawet jeśli narusza to przedstawione wcześniej definicje.

9.1.2. Błędy związane z typami danych i ograniczeniami

Pomyśl o używanym typie danych. Jeśli element danych jest używany w obliczeniach, musi być typu liczbowego. Nie powinien być łańcuchem znaków ani typem związanym z czasem. Podobnie dla danych reprezentujących czas stosuj typy związane z czasem, a nie łańcuchy znaków. Łańcuchy znaków są odpowiednie dla skal i numerów identyfikacyjnych. Szczegółowe informacje znajdziesz w książce *SQL Programming Style* (Morgan Kaufmann, 2005; ISBN: 978-0120887972).

Najczęstszy błąd dotyczący ograniczeń polega na ich pomijaniu. Ograniczenia są przekazywane do optymalizatora i poprawiają jakość kodu — chronią integralność danych. Po ich utworzeniu są używane dla każdej instrukcji wykonywanej na danym obiekcie zbudowanym za pomocą DDL-a. *Za każdym razem działają w ten sam sposób.*

9.2. Podział atrybutów

Podział atrybutów przyjmuje wiele form. Występuje, gdy używany jest jeden atrybut, ale jego wartości są umieszczane w schemacie w różnych miejscach. Wymaga to ponownego poskładania pierwotnego faktu, aby móc go zastosować. Najczęściej występującą formą podziału atrybutów jest tworzenie dla każdej wartości odrębnych tabel. Inna postać to tworzenie w tej samej tabeli odrębnych wierszy z częściami poszczególnych wartości. Te problemy łatwiej będzie zrozumieć na podstawie przykładów.



9.2.1. Podział atrybutów na poziomie schematu

Ten problem polega na umieszczeniu jednego schematu w wielu bazach. Zamiast tworzyć jedną bazę firmy, dane poszczególnych jednostek są zapisywane w odrębnych bazach. Podzielone bazy trzeba połączyć, by uzyskać kompletny model danych. Sytuacja ta różni się od podziału schematu na partycje. Schemat podzielony na partycje nadal jest jednostką logiczną.

9.2.2. Podział atrybutów na poziomie kolumny

Kolumna powinna być wartością *skalarną*. Nie oznacza to, że musi być wartością *atomową*. Określenie „atomowość” pochodzi od greckiego słowa „a-toma” oznaczającego „bez części”. Skalarność oznacza, że dana wartość jest pomiarem na skali. Pomyśl o lokalizacji geograficznej. Para (długość, szerokość) określa *jedną* lokalizację, ale obejmuje *dwie* kolumny. W celu zlokalizowania obiektu na kuli ziemskiej możesz też zastosować jedną liczbę **HTM** (ang. *Hierarchical Triangular Mesh*).

Jeśli wartość nie jest atomowa, potrzebne są ograniczenia gwarantujące, że jej łączone komponenty są prawidłowe.

9.2.3. Podział atrybutów na poziomie tabeli

Gdybym zamierzał utworzyć bazę danych z osobnymi tabelami z męskimi i żeńskimi pracownikami, natychmiast zauważyłbyś, że należy zbudować tylko jedną tabelę z kolumną z kodem płci. Tabelę można potem podzielić na podstawie tej kolumny (`sex_code`). Ta sytuacja jest oczywista, jednak zdarzają się też bardziej subtelne przypadki.

9.3. Problemy z przeciążaniem projektu

Niewłaściwy podział atrybutu skutkuje rozbiciem faktu na fragmenty w tabeli. Przeladowywanie to odwrotny problem — polega na umieszczaniu zbyt wielu elementów w schematach, tabelach i kolumnach.

9.3.1. Przeciążanie na poziomie schematu

Wymyśliłem zwrot „samochód, kalmar i Lady GaGa”, aby opisać umieszczenie w jednostce logicznej pomieszanych i niepowiązanych danych. Na poziomie schematu oznacza to obciążanie jednej bazy danych zbyt wieloma zadaniami. Może tak się stać, gdy dwa systemy wyglądają podobnie, dlatego programista rozszerza istniejący model danych.



9.3.2. Przeciążanie na poziomie tabeli

Choć table można przeglądać na wiele sposobów, koszmarem, który często się powtarza, są table OTLT (ang. *One True Look-up Table*). Związane jest to z umieszczeniem WSZYSTKICH kodowań w jednej dużej tabeli, gdy należy utworzyć osobne table dla poszczególnych kodowań. Myślę, że to Paul Keister był pierwszą osobą, która wymyśliła zwrot OTLT. Don Paterson (<http://www.sqlservercentral.com>) w jednym ze swoich artykułów nazwał tę technikę MUCK (ang. *Massively Unified Code-Key*).

Argumentem na rzecz takich tabel jest to, że potrzebna jest tylko jedna procedura do zarządzania *wszystkimi* kodowaniami i jedna generyczna funkcja do ich wywoływania (funkcja typu „samochody, kalmary i Lady GaGa”). Ta technika pojawia się nieustannie, warto jednak uhonorować Paula Keistera jako pierwszego, który nadał jej nazwę. Pomysł polega na tym, by utworzyć jedną tabelę do wyszukiwania wszystkich kodów ze schematu. Taka tabela zwykle wygląda mniej więcej tak:

```
CREATE TABLE OTLT -- Generyczne_Wyszukiwania?
(generic_code_type CHAR(10) NOT NULL, -- Okropne nazwy!
 generic_code_value VARCHAR(255) NOT NULL, -- Zwróć uwagę na wielkość!
 generic_description VARCHAR(255) NOT NULL, -- Zwróć uwagę na wielkość!
 PRIMARY KEY (generic_code_value, generic_code_type));
```

Elementy danych są teraz traktowane jak metadane, dlatego mają okropne nazwy. Kolumny nie reprezentują tu nic konkretnego — są „magicznymi” generycznymi miejscami na dowolne wartości. Dlatego jeśli schemat obejmuje znane z bibliotek kody DDC (ang. *Dewey Decimal Classification*), kody chorób w systemie ICD (ang. *International Classification of Diseases*) i dwuliterowe kody państw w standardzie ISO-3166, wszystkie te dane będą znajdować się w jednej wielkiej tabeli.

Zacznijmy od problemów z instrukcjami DDL, a następnie przejdźmy do okropnych kwerend, które trzeba pisać (lub ukrywać w widokach), gdy stosowana jest taka tabela. Wróćmy do pierwotnych instrukcji DDL i dodajmy ograniczenie CHECK() do kolumny code_type. W przeciwnym razie w wyniku literówki użytkownik może „wymyślić” nowy system kodowania.

Kody DDC i ICD są liczbowe i mają ten sam format. Składają się z trzech cyfr, kropki i dalszych cyfr (zwykle trzech). Jednak kody ISO-3166 są literowe. Ups — potrzebne będzie następne ograniczenie CHECK, które sprawdzi wartość kolumny code_type i ustali, czy łańcuch znaków ma właściwy format. Teraz tabela będzie wyglądać tak (jeśli oczywiście ktoś spróbuje zbudować ją w prawidłowy sposób, co zdarza się rzadko):

```
CREATE TABLE OTLT
(generic_code_type CHAR(10) NOT NULL
 CHECK(generic_code_type IN ('DDC', 'ICD', 'ISO3166', ..),
 generic_code_value VARCHAR(255) NOT NULL,
 CONSTRAINT Valid_Generic_Code_Type
```




```
CHECK (CASE WHEN generic_code_type = 'DDC'
            AND generic_code_value
            LIKE '[0-9][0-9][0-9].[0-9][0-9][0-9]'
            THEN 'T'
            WHEN generic_code_type = 'ICD'
            AND generic_code_value
            LIKE '[0-9][0-9][0-9].[0-9][0-9][0-9]'
            THEN 'T'
            WHEN generic_code_type = 'IS03166'
            AND generic_code_value LIKE '[A-Z][A-Z]'
            THEN 'T' ELSE 'F' END = 'T'),
generic_description VARCHAR(255) NOT NULL,
PRIMARY KEY (generic_code_value, generic_code_type));
```

Ponieważ typowa baza danych w aplikacji może obejmować dziesiątki kodów, należy stosować się do przedstawionego wzorca tak długo, jak to konieczne. Efekt nie jest zachęcający, prawda? Pomyśl teraz, co się stanie po dodaniu nowych wierszy do tabeli OTLT.

```
INSERT INTO OTLT
(generic_code_type, generic_code_value, generic_description)
VALUES ('ICD', 259.0, 'Inadequate Genitalia after Puberty'),
('DDC', 259.0, 'Christian Pastoral Practices & Religious
Orders');
```

Jeśli popełnisz błąd w kolumnie `generic_code_type` w trakcie wstawiania, aktualizowania lub usuwania danych, otrzymasz niewłaściwą wartość. Jeżeli w kwerendzie użyjesz błędnej wartości kolumny `generic_code_type`, efekty mogą być zaskakujące. Znalezienie takiego błędu może okazać się naprawdę trudne, gdy w jednym ze schematów o podobnych strukturach występują nieużywane kody.

Następną rzeczą, jaką warto zauważyć w tej tabeli, jest to, że używane są długie kolumny typu `VARCHAR(n)` lub, co jeszcze gorsze, typu `NVARCHAR(n)`. Ustawiana długość łańcuchów znaków (n) jest często jedną z największych dozwolonych w danym produkcie SQL-owym.

Ponieważ nie wiesz, jakie dane znajdują się w tabeli, nie da się przewidzieć i uwzględnić w projekcie bezpiecznej i rozsądnej wielkości maksymalnej. Ograniczenie wielkości trzeba umieścić w klauzuli `WHEN` w drugim ograniczeniu `CHECK()` dotyczącym kolumn `code_type` i `code_value`. Możesz też pogodzić się z istnieniem kodów o stałej długości większej niż konieczna.

Duża pojemność kolumny sprzyja wprowadzaniu błędnych danych. Jeśli udostępnisz kolumnę typu `VARCHAR(n)`, możesz otrzymać łańcuch znaków z wieloma spacjami i dziwnym znakiem na końcu. Jeżeli utworzysz kolumnę typu `NVARCHAR(255)`, ktoś może zapisać w niej buddyjską sutrę chińskimi znakami w kodowaniu Unicode.

Pomyśl teraz o problemach związanych z używaniem opisanej tabeli OTLT w kwerendach. Zawsze konieczne jest podanie wartości kolumny `generic_code_type`, a także szukanej wartości.



```
SELECT P1.ssn, P1.lastname, .., L1.generic_description
FROM OTLT AS L1, Personnel AS P1
WHERE L1.generic_code_type = 'ICD'
      AND L1.generic_code_value = P1.disease_code
      AND ..;
```

W tej przykładowej kwerendzie trzeba znać wartość kolumny `generic_code_type` powiązaną z kolumną `disease_code` z tabeli `Personnel`, a także wartości innych kolumn z kodami. Jeśli jednak wpiszesz błędną wartość kolumny `generic_code_type`, i tak możesz otrzymać jakiś wynik.

Ponadto trzeba uwzględnić dodatkowe zasoby na konwersję typów. Dla niektórych kodów bardziej naturalny może okazać się typ liczbowy niż `VARCHAR(n)`, aby zapewnić poprawne sortowanie danych. Dopelnianie łańcuchów cyfr zerami na początku wymaga dodatkowych zasobów i jest ryzykowne, jeśli programiści nie uzgodnili, ilu zer należy używać.

Gdy kwerenda jest wykonywana, trzeba pobrać całą tabelę wyszukiwania — nawet jeśli używanych jest tylko kilka kodów. Jeżeli jeden kod znajduje się na początku obszaru pamięci fizycznej, a drugi — na końcu, potrzebnych może być wiele operacji zmiany strony i zapisu danych w pamięci podręcznej. W trakcie aktualizowania tabeli `OTLT` trzeba zablokować ją do czasu wykonania zadania. Przypomina to noszenie ze sobą wielu tomów encyklopedii, gdy potrzebny jest tylko jeden artykuł z czasopisma.

Rozważ teraz dodatkowe koszty związane ze stosowaniem w tabeli dwuczłowego klucza obcego:

```
CREATE TABLE EmployeeAbsences
(..
  generic_code_type CHAR(3) -- Niezbędna minimalna długość
  DEFAULT 'ICD' NOT NULL
  CHECK (generic_code_type = 'ICD'),
  generic_code_value CHAR(7) NOT NULL, -- Niezbędna minimalna długość
  FOREIGN KEY (generic_code_type, generic_code_value)
  REFERENCES OTLT (generic_code_type, generic_code_value),
..);
```

Teraz trzeba przekształcić typy znakowe, co oznacza dodatkowe koszty. Co gorsza, kody ICD mają naturalną wartość domyślną (000.000 oznacza brak diagnozy), natomiast dla kodów DDC taka wartość nie istnieje. W starszych systemach kodowania często stosowano same dziewiątki, aby przedstawić wartość „różne”. Dzięki temu takie dane po sortowaniu trafiały na koniec raportów w platformach w COBOL-u. Podobnie jak nie istnieje „magiczny i uniwersalny” identyfikator, tak nie ma też „magicznej i uniwersalnej” wartości domyślnej. Opisanie podejście oznacza więc utratę jednego z najważniejszych mechanizmów SQL-a.

Podejrzewam, że to rozwiązanie zostało wymyślone przez programistów obiektowych, którzy traktują je jak rodzaj polimorfizmu w SQL-u. Dla nich tabela jest klasą (choć jest to nieprawdą), dlatego powinna działać polimorficznie (choć tak się nie dzieje).



9.3.3. Przeciążenie na poziomie kolumn

Jeśli nazwa kolumny wymaga wielu członów, jest źle sformułowana. Przyjrzyj się tabeli reprezentującej arkusz z rejestrem przyjąć i wyjść.

```
CREATE TABLE Clipboard
(emp_name CHAR(35) NOT NULL,
 signature_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
 signature_type CHAR(3) DEFAULT 'IN' NOT NULL
CHECK (signature_type IN ('IN', 'OUT')),
PRIMARY KEY (emp_name, signature_type));
```

Rzeczywistą nazwą kolumny `signature_timestamp` powinno być `in_or_out_signature_timestamp`. Gdy chce się wykonać dowolną prostą kwerendę, trzeba podać dwa wiersze w złączeniu zewnętrznym typu *self-join*, aby otrzymać parę z czasem przyścia i czasem wyjścia każdego pracownika.

Oto prawidłowy projekt:

```
CREATE TABLE Clipboard
(emp_name CHAR(35) NOT NULL,
 sign_in_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
 sign_out_time TIMESTAMP, -- NULL oznacza bieżący czas
PRIMARY KEY (emp_name, sign_in_time));
```

Jeden atrybut — czas pracy — trzeba zamodelować w standardowym SQL-u za pomocą *dwóch kolumn*. Tu jednak rozbito go na wiersze identyfikowane za pomocą kodu z metadanymi, informującego, który kraniec przedziału czasowego reprezentuje każda wartość. Gdyby wartościami były długość i szerokość geograficzna, natychmiast dostrzegłbyś problem i umieścił obie części jednego atrybutu (lokalizacji geograficznej) w tym samym wierszu.

W skrajnym przypadku w tym błędnym podejściu używana jest opcja z metadanymi do określania zawartości kolumny, która obejmuje zupełnie różne i niepowiązane atrybuty.

9.4. Nadmiarowość wynikająca z braku postaci normalnej

Przyjrzyj się tabeli używanej do obliczania odległości w linii prostej między podanymi na liście miastami.

```
CREATE TABLE City_Coords
(city_name VARCHAR(25) NOT NULL PRIMARY KEY,
 city_latitude FLOAT NOT NULL
CHECK (city_latitude BETWEEN -1.57079633 AND + 1.57079633)
 city_longitude FLOAT NOT NULL,
CHECK (city_longitude BETWEEN -3.14159265 AND +3.14159265));
```



Możesz użyć kwerendy ze wzorem na dystans między punktami na sferze, aby otrzymać odległość. Ta tabela jest bardzo zwięzła, jednak obliczenia matematyczne dla każdej pary miast generują pewne koszty.

Rozważ teraz tabelę o tym samym przeznaczeniu, która wyszukuje odległość między dowolnymi dwoma miastami z listy.

```
CREATE TABLE City_Distances
(origin_city_name VARCHAR(25) NOT NULL,
 destination_city_name VARCHAR(25) NOT NULL,
 PRIMARY KEY (origin_city_name, destination_city_name),
 CHECK (origin_city_name < destination_city_name)
 linear_distance DECIMAL (7,2) NOT NULL
 CHECK (linear_distance >= 0));
```

Teraz można posortować nazwy dwóch miast i sprawdzić odległość między nimi. Dla (n) miast tabela zawiera $\sim(n^2/2)$ wierszy. Można stosować argumenty związane z wydajnością i innymi aspektami, ważne jest jednak to, że obie tabeli wykonują w schemacie to samo zadanie. Obie są znormalizowane, *jednak jedna z nich jest nadmiarowa*. W branży informatycznej mówi się, że programista, który nosi dwa zegarki, nigdy nie wie, jaka jest godzina. Dlatego posługują się zasadą „jeden fakt, jeden sposób, jedno miejsce, jeden czas” do definiowania normalizacji.

9.4.1. Sprzeczne relacje

Dalej opisana jest reguła biznesowa, która wydaje się sensowna, póki jej się bliżej nie przyjrzeć. Zespół sprzedawców odpowiada za każdego klienta, do którego przypisany jest jeden z członków danego zespołu (sprzedawca). Za pozostałych klientów zespół nie odpowiada. Dlatego dla danego klienta należy poszukać przypisanego do niego sprzedawcy, a gdy ten jest niedostępny, trzeba zgłosić się do zespołu. W takiej sytuacji występują dwie możliwości pozbycia się nadmiarowej relacji.

Nadmiarowość można wyeliminować, usuwając relację „jest odpowiedzialny za” między zespołem a klientem. Uzyskany model zawiera tyle samo informacji co przed wyeliminowaniem nadmiarowości.

```
-- To najprostsza tabela w schemacie
CREATE TABLE SalesTeams
(sales_team_id INTEGER NOT NULL PRIMARY KEY,
 sales_team_name CHAR(10) NOT NULL);

CREATE TABLE Salespersons
(sales_person_id INTEGER NOT NULL PRIMARY KEY,
 sales_person_name CHAR(15) NOT NULL,
 sales_team_id INTEGER NOT NULL
 REFERENCES SalesTeams(sales_team_id)
 ON UPDATE CASCADE);
```



Co zrobić, jeśli przypisany do klienta sprzedawca jest niedostępny? Potrzebna jest możliwość ustawienia wartości NULL i przejścia na poziom zespołu, by pomóc klientowi.

```
CREATE TABLE Customer_Assignments
(customer_id INTEGER NOT NULL PRIMARY KEY,
responsible_sales_team_id INTEGER NOT NULL
REFERENCES SalesPerson(sales_team_id)
ON UPDATE CASCADE,
responsible_sales_person_id INTEGER
REFERENCES Salespersons(sales_person_id)
ON UPDATE CASCADE);
```

Druga możliwość polega na kontrolowaniu nadmiarowości za pomocą operacji DRI. Ponieważ dwa klucze obce wymagające synchronizacji znajdują się w tym samym wierszu, potrzebna jest tylko jedna aktualizacja. Zespół o danym **identyfikatorze** może (ale nie musi) obejmować sprzedawców, ale sprzedawca musi być przydzielony do jakiegoś zespołu.

```
-- To ta sama prosta tabela
CREATE TABLE SalesTeams
(sales_team_id INTEGER NOT NULL PRIMARY KEY,
sales_team_name CHAR(10) NOT NULL);

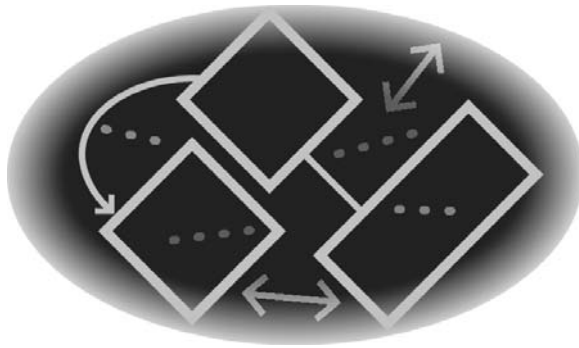
CREATE TABLE Salespersons
(sales_person_id INTEGER NOT NULL PRIMARY KEY,
sales_person_name CHAR(15) NOT NULL,
sales_team_id INTEGER NOT NULL
REFERENCES SalesTeams(sales_team_id)
ON UPDATE CASCADE,
UNIQUE (sales_person_id, sales_team_id));

CREATE TABLE Customer_Assignments
(customer_id INTEGER NOT NULL PRIMARY KEY,
sales_team_id INTEGER NOT NULL,
sales_person_id INTEGER
REFERENCES Salespersons(sales_person_id)
ON UPDATE CASCADE
ON DELETE SET NULL,
FOREIGN KEY (sales_person_id, sales_team_id)
REFERENCES Salespersons (sales_person_id, sales_team_id)
ON UPDATE CASCADE);
```





Typy danych



Skorowidz

A

ACID, 32
adres
 IP, 226
 IPv6, 274
agregacja, 110, 591, 596
 w systemach OLAP, 441
aktualizowanie, 312
 stanu magazynu, 651
 wierzchołków, 554
algebra boolowska, 289
algorytm
 Metaphone, 238
 NYSIIS, 241
 Soundex, 235
aliasy tabel, 638
alokacja pamięci, 186
ANSI SQL, 40
arytmetyka, 217
atomowość, 33
atrybuty, 112, 196
aukcje, 645
 angielskie, 646
 holenderskie, 646
 japońskie, 646
 w modelu Vickrey, 647

B

bajty, 208
BASE, 43
bazy danych
 OLAP, 596
 OLTP, 596
BCD, Binary-Coded Decimal,
 208
BCNF, 96
bity, 208
blok
 <definicja asercji>, 21
 <definicja kolejności
 sortowania>, 22
 <definicja tłumaczenia>, 22
 <definicja zestawu
 znaków>, 21
 <instrukcja grant>, 21
 <lista kolumn widoku>, 124
błędy, 46, 193
 w rozszerzeniach
 producentów, 317
BNF, Backus-Naur Form, 67
brakujące
 liczby, 623
 odczyty, 498

okresy, 684
wartości, 287
 w danych, 496, 501
 w kolumnach, 285
brudny
 odczyt, dirty read, 35
 zapis, dirty write, 35

C

CAP, 43
cechy grafów, 506
ciągłość okresów, 680
CSV, Comma Separated
 Values, 739
CTE, Common Table
 Expression, 404, 723
cykle, 140
czas, 243, 669, 670
 końcowy, 690
 początkowy, 690
 trwania, 699
czwarta postać normalna, 4NF,
 98

**D**

dane liczbowe, 207
data, 670
 format juliański, 691
DCL, Data Control Language,
 20
DDC, Dewey Decimal
 Classification, 150, 198
DDL, Data Declaration
 Language, 20, 107
definicje kolumn, 52
deklarowanie łańcuchów
 znaków, 400
denormalizacja, 79
diagram zmian stanu, 260
diagramy ER, 24, 25
DKNF, 101
długości ścieżek, 510
długość geograficzna, 270
DML, Data Manipulation
 Language, 86
DNS, Domain Name System, 44
dodawanie
 elementów, 531
 macierzy, 580
dominanta, 429
dopasowywanie fonetyczne, 235
dostępność, 43
DRI, Declarative Referential
 Integrity, 25
druga postać normalna, 2NF, 93
drzewa, 533
 aktualizowanie
 wierzchołków, 554
 funkcje, 548
 liść, 536
 określanie poziomów, 536
 określanie wysokości, 542
 przenoszenie poddrzewa,
 555
 usuwanie poddrzewa, 537,
 546
 usuwanie wierzchołków, 546
 wstawianie poddrzew, 538
 wyszukiwanie poddrzew, 541
 wyszukiwanie poziomów,
 542
 wyszukiwanie ścieżek, 542
 zapełnianie luk, 547
 znajdowanie korzenia, 535
duplikaty w tabeli, 305
dynamiczny SQL, 729
dzielenie
 bez reszty, 658
 relacji, 655
 Romleya, 663
 Todda, 659
 z resztą, 656
 z użyciem złączeń, 662
 z wykorzystaniem
 operatorów zbiorów, 662

E

EKNF, 95
ekstrema, 419
elementy danych, 269, 276
eliminowanie luk, 630
encje, 24
EOF, end of file, 22, 178
ER, Entity-Relationship, 24

F

fantomowe dane, phantom, 35
FIFO, first-in, first-out, 568,
 648
FIPS, 338
format
 CSV, 738
 juliański, 691
formaty
 dat i czasu, 249, 250, 337, 670
 wyświetlania danych, 338
funkcja
 AVG(), 415, 606
 BIT_LENGTH, 233
 CAST, 217, 399, 467
 CEILING(), 225
 CHAR_LENGTH, 233
 COALESCE(), 220, 386,
 473, 713, 714
 CORR, 438, 439
 COUNT, 410, 659
 COVAR_POP, 438

COVAR_SAMP, 438
CUME_DIST, 450, 605
DENSE_RANK(), 450, 605
EXP(), 225
FIRST, 455, 609
FLOOR(), 225, 692
GREATEST(), 225, 424
LAG, 453, 608, 700
LAST, 455, 609
LEAD, 453, 608, 700
LEAST(), 225, 424
LIST(), 427, 428
LN(), 225, 434
LOWER, 231
MAKEDATE, 688
MAX, 419
MOD(), 222
NEXT VALUE FOR, 68
NTILE, 611
NULLIF(), 219, 386
obliczająca IRR, 162
OCTET_LENGTH, 233
PERCENT_RANK(), 450,
 605
POSITION, 233
POWER, 228
PRD(), 432, 433, 434
RANK(), 449, 605
REGR_AVGX, 438
REGR_AVGY, 438
REGR_COUNT, 438
REGR_INTERCEPT, 438
REGR_R2, 438
REGR_SLOPE, 438
REGR_SXX, 438
REGR_SXY, 438
REGR_SYY, 438
REPLACE, 234
REPLICATE, 234
ROUND(), 225
ROW_NUMBER(), 431
SIGN(), 222
SPACE, 234
SQRT(), 224
STDDEV_POP, 438
STDDEV_SAMP, 438
SUBSTRING, 233
SUM(), 414



funkcja

TRIM, 233
TRUNCATE(), 225
UPPER, 231
VAR_POP, 437
VAR_SAMP, 437

funkcje

agregujące, 409, 429, 449,
476, 604
dla tabel
dwukolumnowych, 437
matematyczne, 222
odwrotne, 155
OLAP, 600
porządkowe, 449, 604
wykładnicze, 224
z algorytmem Soundex, 235
zdefiniowane przez
użytkownika, 709
znakowe, 232
związane z czasem, 670, 694
związane z wartościami
NULL, 294
związane ze skalą, 225
zwracające ekstrema, 419,
421, 424

G

GAAP, 266
generalnie dostępny, basically
available, 44
geometria taksówkowa, 521, 523
globalne tabele stałych, 164
główna tabela wyszukiwania,
151
GMT, Greenwich Mean Time,
246
graf skierowany, 533
grafy, 505, 637
acykliczne, 517
dla relacji równoważności,
525
krawędzie, 507
stopień wchodzący, 508
stopień wychodzący, 508
ścieżki, 507, 510
wierzchołki, 506, 508

wyświetlanie ścieżek, 513
wyznaczanie ścieżek, 511
grupowanie, 442
łańcuchów znaków, 232
zaawansowane, 596

H

hierarchiczne
elementy danych, 276
tabele pomocnicze, 150
HTM, Hierarchical Triangular
Mesh, 111

I

IANA, 274
implementacja, 707
implikacja logiczna, 388
indeksowanie, 127
informacje o czasie, 252
instrukcja, 730
CASE, 733
CLOSE, 190
COALESCE(), 172
COMMIT, 33
CREATE ASSERTION, 28,
180
CREATE CHARACTER
SET, 72
CREATE COLLATION, 73,
182
CREATE DOMAIN, 65,
181
CREATE FUNCTION, 736
CREATE PROCEDURE,
183, 736
CREATE SCHEMA, 179
CREATE SEQUENCE, 66,
67
CREATE TABLE, 50, 134
CREATE TRANSLATION,
73, 183
CUBE, 444, 599
DEALLOCATE CURSOR,
190
DECLARE CURSOR, 186

DELETE FROM, 22, 301,
305
FETCH, 38, 188
FOR, 738
GET DIAGNOSTICS, 36
GOTO, 193
GROUP BY GROUPING
SET, 442, 598
IF, 733
INSERT INTO, 307
LEAVE, 735
LOOP, 735
MERGE, 312, 318
OPEN, 188
PRINT, 735
REFRESH TABLE, 135
REPEAT, 735
RESIGNAL, 732
ROLLBACK, 33, 35
ROLLUP, 443, 599
Schema, 19
SELECT, 403–408, 459
SET TRANSACTION, 36
SIGNAL, 732
UNION, 322, 466
UNION ALL, 71, 322, 323
UPDATE, 309, 315, 686

instrukcje

CALL, 736
DDL, 193, 563
warunkowe, 733

interfejs

CLI, 728
WIMP, 144

interpolacja, 162

IRR, Internal Rate of Return,
153

iteracyjne wyznaczanie ścieżek,
511

izolacja, 34, 36
snapshotów, 39

J

jawne lokalizatory fizyczne, 78
jednostki
mieszane, 280
niedziesiątne, 280



język

- DCL, 20
- DDL, 20
- DML, 86

JPNF, 99, 100

K

kalendarze, 244, 674

- osobiste, 676

klasy równoważności, 524, 635

klauzula

- AUTHORIZATION, 21
- CHECK, 28, 132, 388
- DEFAULT, 52
- DELETE FROM, 302
- FOR, 738
- FROM, 405
- GROUP BY, 120, 387, 406, 598
- HAVING, 406
- INSERT INTO, 307
- LATERAL, 723
- NOT MATCHED, 319
- okna, 445, 600, 604
- ORDER BY, 407, 445, 601
- OVER, 68, 600
- PARTITION BY, 600
- REFERENCES, 55, 56
- SELECT, 407, 479
- SET, 310
- UNTIL, 735
- UPDATE, 310
- WHERE, 302, 312, 405, 682
- WITH CHECK OPTION, 128, 132

kliki, 524, 529, 641

klucz, 60, 75

- główny, 61

klucze

- naturalne, 77
- sztuczne, 77
- zagnieżdżone, 60
- zastępcze, 76

kod

- proceduralny, 171

kodowanie, 707

kody

- DDC, 198
- HTM, 270
- ICD, 198
- TZD, 254

kolejki, 563

- FIFO, 568
- LIFO, 568
- priorytetowe, 567
- zmienianie

- uporządkowania, 565

kolejkowanie, 564

kolejność wykonywania operacji, 324

kolumna, 27, 52

- typu CHAR(32), 274
- typu INTEGER, 272
- typu SMALLINT, 273
- typu VARCHAR(15), 272

kolumny

- konwersji, 119
- nazwane, 574
- obliczane, 63, 119
- wzajemnie zależne od siebie, 281
- z indeksem, 577

kontekst, 287

kontrola współbieżności, 34

- logiczna, 42
- optymistyczna, 39
- pesymistyczna, 38

konwencje tworzenia

- diagramów, 26

konwersje

- odwrotne, 219
- typów danych, 336
- typów liczbowych, 215

korelacja, 439

korzeń, 535

krawędzie grafu, 507

kształty w geometrii taksówkowej, 523

kubelki o stałym rozmiarze, 141

kursor, 190

- do odczytu, 189
- z agregacjami, 190

kwantyfikatory, 362

kwerendy, 441, 589

- agregowanie danych, 591
- dotyczące sekwencji, 621
- dotyczące serii, 621
- partycjonowanie danych, 591
- skalarne, 355
- z granicami, 621
- związane z czasem, 669

L

liczbowe typy danych, 208, 213

liczby

- Fibonacciego, 167
- pierwsze, 164
- podawane w losowej kolejności, 168
- wymierne, 279

LIFO, last-in, first-out, 568, 648

lista

- eliminowanie luk, 630
- kolumn widoku, 124
- parametrów, 738
- przestawianie wartości, 629
- przesuwanie wartości, 629
- przyrostków, 144
- sąsiedztwa, 534, 558
- z wyliczeniem, 139
- zawijanie, 630

liść, 536

logarytm, 434

logika, 289

- trójwartościowa, 363

lokalizatory fizyczne, 78

luki, 613, 630

- w szeregach czasowych, 677

Ł

łańcuch znaków, 230, 400

łączenie operatorów, 324

M

macierze, 573

- dodawanie, 580
- mnożenie, 580



macierze

- równość, 579
- ścisłości, 519
- transponowanie, 582
- wyznacznik, 587

magazyn, 651

masowe wczytywanie, 309

materializowanie widoków, 124

mechanizm DRI, 28

mediana, 430

- ważona, 431

metadane, 194

mieszane dane, 501

milenium, 703

mnożenie macierzy, 580

model

- ACID, 32
- BASE, 43
- działania wyzwalaczy, 184

modyfikator

- ALL, 321
- DISTINCT, 321

MQT, Materialized Query

Table, 134

MUCK, 151, 198

N

nadmiarowość, 109, 201

- całej tabeli, 110
- na poziomie agregacji, 110
- związana ze ścieżką dostępu, 111

narzędzia CASE, 108

nazwy

- kolumn, 144, 719
- kolumn obliczanych, 63
- tabel, 722

niepewny stan, soft state, 44

niepowtarzalny odczyt,

- nonrepeatable read, 35

normalizacja, 85, 108

notacja BNF, 67

NPV, Net Present Value, 153

numerowanie

- obszarów, 615
- wierszy, 449, 604

O

obiekty schematu, 177

obliczanie

- kosztu, 649
- średniej, 416

obliczenia matematyczne, 293, 670

obsługa

- aukcji, 647
- błędów, 46
- widoków, 124
- znaków, 72

obszary, 613

- o maksymalnej wielkości, 617

odczyt z pliku, 22

odległość

- euklidesowa, 522
- taksówkowa, 522

odporność partycji, 43

odwzorowywanie sekwencji

- na cykl, 140

oferty, 645

ograniczenia

- [NOT] DEFERRABLE, 64
- dotyczące jednej tabeli, 119
- referencyjne, 354, 361
- zagnieżdżone unikatowości, 57
- związane ze zmianami stanu, 260

ograniczenie

- CHECK(), 54
- NOT NULL, 53
- NOT NULL UNIQUE, 59
- PRIMARY KEY, 54
- UNIQUE, 54, 501

okresy, 372

- do raportów, 697
- wielodniowe, 694

określanie zasięgu, 720

OLAP, Online Analytic

- Processing, 596

OLTP, Online Transaction

- Processing, 441, 596

opcja

- ALL, 330
- CASCADE, 56, 117

CHECKPOINT, 33

DISTINCT, 413

GLOBAL, 51

IDENTITY, 66, 68

LOCAL, 51

NO ACTION, 57

NO CYCLE, 68

PERSISTENT, 63

SAVEPOINT, 33

SELECT DISTINCT, 330

SET DEFAULT, 57

SET NULL, 57

operacje

- arytmetyczne, 218
- na datach, 256
- na drzewach, 537
- na macierzach, 579
- na tabelach, 301
- na zbiorach, 321
- związane z referencjami, 56

operator

- BETWEEN, 715
- EXCEPT, 325, 328, 329
- INTERSECT, 325, 328, 329
- IS [NOT] DISTINCT FROM, 340
- IS NULL, 341
- LIKE, 396
- MOD, 636
- UNION, 121, 322, 334
- UNION ALL, 324
- UNION JOIN, 477

operatory

- arytmetyczne, 217
- grupowania, 442, 598
- monadyczne, 341
- porównywania, 335
- theta, 335
- zbiorów, 662

optymalizowanie

- agregacji, 413
- predykatu IN(), 348

organizacja IANA, 274

ortogonalność, 407

ostateczna spójność, eventual consistency, 44

ostatnia znana wartość, 497

OTLT, One True Lookup Table, 151, 198

**P**

- pakowanie, 487, 653
- parametry, 738
- partycjonowanie danych, 591
- pętla, 734
 - FOR-DO, 737
 - iteracyjna, 142
 - REPEAT, 735
 - WHILE, 734, 735
- pięta postać normalna, 5NF, 99
- pierwsza postać normalna, 1NF, 88
- podjęzyki SQL-a, 713
- podklauzula
 - ORDER BY, 446
 - PARTITION BY, 446
 - ramki okna, 447, 602
- podkwerendy skorelowane, 459
- podobszary o wielkości (n), 614
- podsekwencje, 613
- podzbiory właściwe, 331
- podział atrybutów, 112, 196
 - na poziomie kolumny, 197
 - na poziomie schematu, 197
 - na poziomie tabeli, 197
- podział na partycje
 - na podstawie kolejności, 594
 - na podstawie podklauzuli, 601
 - na podstawie przedziałów, 592
 - w klauzuli okna, 604
 - za pomocą funkcji, 593
- pokrycie, covering, 591, 631
- pola, 27
- porównywanie
 - struktur, 559
 - tablic, 585
 - wartości NULL, 288
 - wierszy, 338
 - wierzchołków, 559
- porządkowanie kodu, 711
- postać normalna
 - Boyce'a-Codda, BCNF, 96
 - z kluczem dziedziny, DKNF, 101
 - z kluczem podstawowym, EKNF, 95
- powtarzające się kolumny, 90
- poziom izolacji, 36, 40, 41
 - CURSOR STABILITY, 38
 - READ COMMITTED, 37
 - REPEATABLE READ, 37
 - SERIALIZABLE, 37
- poziomy
 - spójności, 41
 - zagnieżdżenia, 717
- predykat
 - [NOT] EXISTS(), 357
 - [NOT] IN(), 347
 - ALL, 367
 - BETWEEN, 369, 592
 - EXISTS, 304, 358–362, 552, 619
 - IN, 351–355
 - IS, 342
 - LIKE, 393
 - NOT EXISTS, 361, 657
 - OVERLAPS, 258, 372
 - SIMILAR TO, 397
 - SOME, 365
 - UNIQUE, 345
- predykaty
 - dotyczące wyników podkwerend, 345
 - z wartością logiczną, 293
- problem
 - pakowania, 653
 - z rokiem 2000, 701
- procedura
 - Dequeue, 564
 - Empty, 564
 - Enqueue, 564
- programowanie
 - deklaratywne, 709
 - mieszane, 709
 - proceduralne, 709
- przechodniość, 528, 640
- przechowywanie adresów IP
 - typ CHAR(39), 227
 - w postaci dwójkowej, 227
 - w SQL-u, 271
- przełączanie
 - na poziomie kolumn, 201
 - na poziomie schematu, 197
 - na poziomie tabeli, 198
 - projektu, 197
- przekształcanie
 - kodu na tabele, 171
 - liczb na słowa, 225
 - list sąsiedztwa, 558
 - wartości na NULL, 219
 - zbiorów zagnieżdżonych, 558
 - stopni na radiany, 270
- przesyłanie danych, 309
- przetwarzanie listy, 91
- przycinanie liczb, 215
- przypisania, 732
- przyrostki, 144, 194
- punkty
 - końcowe, 689
 - początkowe, 689
 - w wielokacie, 520
- puste łańcuchy znaków, 396

R

- ramka okna, 602
- raporty, 441
- referencja, 76
 - FOREIGN KEY, 261
- reguła
 - aktualizowania widoków, 86
 - bezpieczeństwa operacji niskopoziomowych, 87
 - fizyczna niezależność danych, 87
 - gwarantowanego dostępu, 86
 - informacyjna, 86
 - kompletności podjęzyka danych, 86
 - logiczna niezależność danych, 87
 - model relacyjny online, 86
 - niezależność ograniczeń integralności, 87
 - określania zasięgu, 720



traktowania wartości
 NULL, 86
 tworzenia nazw kolumn,
 719
 zerowa, 86
 rekordy, 26
 rekurencyjne wyrażenia, 427
 relacje, 24, 85
 reprezentowanie czasu, 698
 rodzaje
 ofert, 645
 wartości NULL, 298
 rok przestępny, 702
 rozbudowywanie predykatu
 LIKE, 396
 rozkład
 Poissona, 567
 t-Studenta, 163
 rozszerzenia
 liczb
 zmiennoprzecinkowych,
 214
 od producentów, 234, 317,
 452, 608
 rozszerzenie ROLLUP, 443
 równość, 331
 rysowanie diagramów ER, 25

S

SAN, Standard Address
 Number, 111
 scalanie taśm, 318
 schemat, 21, 177
 do obsługi aukcji, 647
 sekwencje, 68, 498, 621, 626
 serie, 613, 621
 sesje, 32
 skalarne wyrażenia SELECT,
 479
 skeumorfizm, 710
 składnia złączeń
 dawna, 480
 nowa, 480
 słowo kluczowe
 ANY, 365
 DAY, 670
 DISTINCT, 413
 MINUTE, 670
 MONTH, 670
 OCCURS, 27
 SECOND, 670
 SET, 732
 SOME, 365
 TIMEZONE_HOUR, 670
 TIMEZONE_MINUTE, 670
 YEAR, 670
 sortowanie
 łańcuchów znaków, 231
 szybkie, quicksort, 81
 wierszy, 80
 spłaszczanie tabeli, 583
 spójność, 34, 43
 monotoniczna odczytu, 45
 monotoniczna zapisu, 45
 odczytów własnych
 zapisów, 44
 po stronie serwera, 45
 przyczynowa, 44
 sesji, 44
 sprzeczne relacje, 202
 SQL/CLI, 729
 SQL/PSM, 729
 stabilne małżeństwa, 483
 stałe, 391
 standard
 ANSI SQL, 40
 FIPS, 338
 IEEE, 214
 ISO-11179, 194
 statystyki opisowe, 437
 stopień
 wchodzący, 508
 wychodzący, 508
 stosowanie
 aliasów tabel, 638
 funkcji OLAP, 600
 sekwencji, 68
 tabeli pomocniczej, 157
 wartości NULL, 257
 strefy czasowe, 253
 struktury
 danych, 503
 ze wskaźnikami, 127
 sumowanie sekwencji wartości,
 626
 symetryczność, 527, 639

system
 klasyfikacji dziesiętnej, 150
 Ocelot, 399
 szeregi czasowe, 677
 szerokość geograficzna, 270

Ś

ścieżka najkrótsza, 511
 ścieżki w grafie, 510
 średnia, 416
 dla kilku kolumn, 418
 dla pustych grup, 416
 święta, 675

T

tabela, 49, 585
 pośrednicząca, junction
 table, 57
 Series, 138
 tabele
 bazowe, 50
 brakujące, 285
 tabele
 jako encje, 24
 jako relacje, 24
 krzyżowe, 428
 MQT, 134
 okresów do raportów, 697
 pochodne, 718
 pomocnicze, 137
 do prostych konwersji,
 145
 hierarchiczne, 150
 oparte na przedziale, 148
 oparte na zbiorach, 149
 wyszukiwania, 143
 z funkcjami, 153, 162
 z wieloma konwersjami,
 146
 z wieloma parametrami,
 146
 puste, 285
 schematu, 180
 stałych, 164
 reprezentowanie czasu, 698
 spłaszczanie, 583



tymczasowe, 51, 180
z kalendarzem, 265
z klauzulą LATERAL, 723
z okresami, 267

tablice, 583
Cuttera, 242
oparte na kolumnach
z indeksem, 577
oparte na nazwanych
kolumnach, 574

transakcje, 32

transponowanie macierzy, 582

trwałość, 34

trzecia postać normalna, 3NF, 94

twierdzenie CAP, 42

tworzenie
diagramów, 26
indeksu, 400
kubeków, 141

tygodnie, 695

typ danych, 194, 205
BIGINT, 67, 208
CHAR(32), 274
DATE, 248, 671
DECIMAL, 208
DOUBLE PRECISION, 213
FLOAT, 213
INTEGER, 208, 272
INTERVAL, 255
NUMERIC, 208
REAL, 213
SMALLINT, 208, 227, 273
TIME, 248, 671
TIMESTAMP, 248, 671
TINYINT, 68
VARCHAR(15), 272

typy
aukcji, 646
kluczy, 76
logiczne, 208
widoków, 118

U

UCS, Universal Character Set,
72

UDF, User-Defined Function,
709

uprawnienia użytkownika, 21

ustawienie
NO MAXVALUE, 68
NO MINVALUE, 68

usuwanie
poddzewa, 537, 546
widoków, 133
wierzchołków, 546

UTC, Universal Coordinated
Time, 246

utracona aktualizacja, lost
update, 35

uzupełnianie brakujących liczb,
623

W

wartość NaN, 214

wartość NULL, 28, 218, 283, 454
a języki główne, 294
a obliczenia matematyczne,
293
a predykat IN(), 352
porównywanie, 288
powiązane funkcje, 294
rodzaje, 298
unikanie, 297
w predykatkach, 291
wskazówki projektowe, 295
złączenia zewnętrzne, 471

wartość
QNaN, 214
SNaN, 214
UNKNOWN, 289

wektory, 270
z jednostkami miary, 275
z wartościami, 275

wewnątrzwerszowe rozwijanie
tekstu, 125

wewnętrzna stopa zwrotu, 155

widoki, 649
modyfikowalne, 117
oparte na operatorze
UNION, 121
tylko do odczytu, 117
w kwerendach, 116
zagnieżdżone, 123
zgrupowane, 120

wielokolumnowe
elementy danych, 269

wiersze, 26
reprezentujące
przechodniość, 528, 640
reprezentujące
symetryczność, 527, 639
reprezentujące zwrotność,
526, 638

wierzchołki
grafu, 506
izolowane, 508
ujściowe, 508
wewnętrzne, 508
źródłowe, 508

wrostkowe złączenia
wewnętrzne, 464

współbieżność, 34

współrzędne, 270

wstawianie, 308
poddzew, 538

wydajność, 659

wygładzone wyniki, 498

wrażenia
CTE, 404, 427, 723
logiczne, 666
regularne, 399
w postaci podkwerend, 391

wrażenie
CASE, 155, 315, 383, 650
implikacja logiczna, 388
klauzula CHECK, 388
klauzula GROUP BY, 387
predykaty LIKE, 397
IS [NOT] NORMALIZED,
344
SELECT, 479

wyspy, 613

wyszukiwanie, 400
dat, 688
obszarów, 617
poddzew, 541
poziomów, 542
ścieżek, 542

wyświetlanie ścieżek, 513

wyznaczanie
IRR, 157
mediany, 431
ścieżek, 511



wyznacznik macierzy, 587
 wyzwalacze, 183
 wzorce, 394

Z

- zaawansowane instrukcje
 SELECT, 459
 zagnieżdżanie, 717
 zagnieżdżone kwerendy, 407
 zagnieżdżony SQL, 728
 zakleszczenie, livelock, 32
 aktywne, 46
 pasywne, 46
 zależności
 funkcyjne, 88
 wielowartościowe, 88
 zaokrąglanie liczb, 215
 zapisywanie adresów IPv6, 274
 zarządzanie kolejkami, 565
 zasięg, 720
 zastępowanie
 brakujących odczytów, 498
 operatorów OR, 351
 pętli iteracyjnej, 142
- zbiory, 321
 zagnieżdżone, 517, 538,
 545, 558
 zdarzenia
 ciągłe, 684
 tabeli, 183
 zestaw
 UCS, 72
 znaków, 72
 złączenia
 naturalne, 472
 T dr. Codda, 490
 w widoku, 122
 wewnętrzne, 360, 464
 z ograniczeniami, 481
 pakowanie kulek, 487
 realizowanie zamówień,
 482
 stabilne małżeństwa, 483
 złączenia
 zewnętrzne, 361, 465
 funkcje agregujące, 476
 liczba złączeń, 475
 pełne, 477
- typu self-join, 473
 wartości NULL, 471
 z wyszukiwaniem, 472
- złączenie
 INNER JOIN, 463
 LEFT OUTER JOIN, 658
 NATURAL JOIN, 465
 OUTER JOIN, 465
 self-join, 630
 zmiany stanu, 260
 znajdowanie
 bezpośrednich
 podwładnych, 543
 korzenia, 535, 541
 liści, 536, 541
 najmłodszego
 podwładnego, 543
 podobszarów, 614
 ścieżki, 544
 znakowe typy danych, 229
 zwrotność, 526, 638

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ 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>

Poznaj SQL na poziomie eksperta!

SQL jest narzędziem stworzonym specjalnie na potrzeby baz danych. Nauka tego języka wymaga dogłębnego zrozumienia mechanizmów relacyjnych baz danych. Co więcej, w przypadku dużych zbiorów danych lub zbiorów danych wymagających szczególnego traktowania pozornie nieznaczne usterki kodu SQL mogą prowadzić do istotnych problemów w użytkowaniu aplikacji.

Niniejsza książka jest przeznaczona dla aktywnych programistów SQL. Przedstawia zaawansowane techniki i idiomy programistyczne z obszaru tego języka. Pokazano w niej również, w jaki sposób można rozwiązywać problemy, które często pojawiają się w bazach danych. Prezentowane tu rozwiązania można implementować właściwie we wszystkich systemach bazodanowych korzystających z SQL. To klasyczny podręcznik, systematycznie uaktualniany, który doczekał się prawdziwego uznania wśród specjalistów.

W książce omówiono między innymi:

- transakcje i kontrolę współbieżności
- widoki oraz tabele i normalizację danych
- tabele pomocnicze, dane tymczasowe i operacje na danych, w tym na adresach IP, datach, danych tekstowych
- zaawansowane techniki tworzenia kwerend
- hierarchię w SQL, kolejki i macierze w SQL
- nowości w wydaniu: model z podwójnym zapisem czasu, funkcje agregujące, rozwiązywanie problemów z instrukcjami DDL i wiele innych zagadnień



JOE CELKO

— uznany ekspert w dziedzinie baz danych.

Z jego konsultacji korzystały m.in. służby medyczne, NASA, siły zbrojne USA. Autor wysoko cenionych książek dotyczących SQL. Napisał również ponad 1200 artykułów w prasie informatycznej i akademickiej, w większości poświęconych bazom danych.

Helion

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

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>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-2530-2



9 788328 325302

Informatyka w najlepszym wydaniu

cena: 119,00 zł