

Dan Sullivan



NOSQL

PRZYJAZNY PRZEWODNIK



Helion 

Tytuł oryginału: NoSQL for Mere Mortals®

Tłumaczenie: Jakub Hubisz

ISBN: 978-83-283-2488-6

Authorized translation from the English language edition, entitled: NOSQL FOR MERE MORTALS; ISBN 0134023218; by Dan Sullivan; published by Pearson Education, Inc, publishing as Addison-Wesley Professional.

Copyright © 2015 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Polish language edition published by HELION S.A. Copyright © 2016.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek 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/nosqlp.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/nosqlp>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

O autorze	15
Przedmowa	17
Wprowadzenie	19
Kto powinien przeczytać tę książkę?	20
Cel tej książki	21
Jak czytać tę książkę	22
Jak zorganizowana jest ta książka	22
Część I: „Wprowadzenie”	22
Część II: „Bazy klucz-wartość”	22
Część III: „Bazy dokumentów”	23
Część IV: „Bazy rodziny kolumn”	23
Część V: „Bazy grafowe”	23
Część VI: „Wybór bazy danych dla Twojej aplikacji”	24
Część VII: „Dodatki”	24
 Część I Wprowadzenie	 25
Rozdział 1. Różne bazy do różnych zastosowań	27
Projekt bazy relacyjnej	28
Aplikacja e-commerce	28
Wczesne systemy zarządzania bazami danych	29
Systemy oparte na plikach płaskich	29
Systemy z hierarchicznym modelem danych	33
Systemy z sieciowym modelem danych	35
Podsumowanie wczesnych systemów baz danych	37
Rewolucja baz relacyjnych	38
Relacyjne systemy zarządzania danymi	39
Przyczyny powstania baz NoSQL	45
Skalowalność	46
Koszt	47

Elastyczność	47
Dostępność	48
Podsumowanie	48
Studium przypadku	50
Pytania kontrolne	50
Odniesienia	51
Bibliografia	51
Rozdział 2. Różnorodność baz NoSQL	53
Zarządzanie danymi w bazach rozproszonych	54
Przechowywanie danych w sposób trwały	55
Utrzymanie spójności danych	56
Zapewnienie dostępności danych	57
Zrównoważenie czasów reakcji, spójności i trwałości	60
Spójność, dostępność i partycjonowanie: teoria CAP	62
ACID i BASE	64
ACID: atomowość, spójność, izolacja, trwałość	64
BASE: zasadnicza dostępność, miękki stan, ostateczna spójność	65
Rodzaje ostatecznej spójności	66
Cztery typy baz NoSQL	68
Bazy par klucz-wartość	68
Bazy dokumentów	73
Bazy rodziny kolumn	75
Bazy grafowe	77
Podsumowanie	79
Pytania kontrolne	80
Odniesienia	81
Bibliografia	81
Część II Bazy klucz-wartość	83
Rozdział 3. Wprowadzenie do baz klucz-wartość	85
Od tablic do baz klucz-wartość	86
Tablice: baza klucz-wartość z kółkami treningowymi	86
Tablice asocjacyjne: zdjęcie kółek treningowych	87
Pamięć podręczna: dodanie do roweru biegów	88
Bazy klucz-wartość w pamięci i na dysku: od rowerów do pojazdów zmotoryzowanych	91

Podstawowe funkcjonalności baz klucz-wartość	92
Prostota: komu w ogóle potrzebne są skomplikowane modele danych?	93
Szybkość: nie ma czegoś takiego jak „za szybko”	94
Skalowalność: nadążaj za wyścigiem	95
Klucze: więcej niż tylko nic nieznaczące identyfikatory	101
Jak konstruować klucze	101
Używanie kluczy do odnajdywania wartości	102
Wartości: przechowywanie prawie dowolnych danych	106
Wartości nie wymagają silnego typowania	106
Ograniczenia w wyszukiwaniu wartości	107
Podsumowanie	108
Pytania kontrolne	109
Odniesienia	109
Bibliografia	110
Rozdział 4. Terminologia baz klucz-wartość	111
Terminy związane z modelowaniem danych w bazach klucz-wartość	112
Klucz	114
Wartość	116
Przestrzeń nazw	117
Partycja	118
Klucz partycjonowania	120
Brak schematu	121
Terminy związane z architekturą baz klucz-wartość	122
Klaster	122
Pierścień	124
Replikacja	124
Terminy związane z implementacją baz klucz-wartość	126
Funkcje haszujące	126
Kolizja	127
Kompresja	128
Podsumowanie	129
Pytania kontrolne	129
Odniesienia	130
Rozdział 5. Projektowanie baz klucz-wartość	131
Projektowanie kluczy i partycjonowanie	132
Klucze powinny posiadać konwencję nazewniczą	132
Dobrze zaprojektowane klucze wymagają mniej kodu	133

Praca z zakresami wartości	134
Klucze muszą brać pod uwagę ograniczenia	135
Wykorzystanie kluczy do partycjonowania	136
Projektowanie struktury wartości	137
Typy posiadające strukturę pomagają skrócić czas oczekiwania	138
Duże wartości mogą prowadzić do mało wydajnych operacji odczytu i zapisu	140
Ograniczenia baz klucz-wartość	142
Pobieranie wartości wyłącznie za pomocą klucza	143
Bazy klucz-wartość nie wspierają przeszukiwania zakresów	144
Brak standardowego języka zapytań podobnego do SQL dla baz relacyjnych	144
Wzorce projektowe dla baz klucz-wartość	145
Klucze z ograniczonym czasem życia	145
Emulowanie tabel	147
Agregacje	148
Agregacje atomowe	150
Indeksy	151
Podsumowanie	153
Studium przypadku: bazy klucz-wartość dla konfiguracji aplikacji mobilnej	153
Pytania kontrolne	156
Odniesienia	156

Część III Bazy dokumentów 157

Rozdział 6. Wprowadzenie do baz dokumentów 159

Czym jest dokument?	160
Dokumenty nie są wcale takie proste	160
Dokumenty a pary klucz-wartość	163
Zarządzanie wieloma dokumentami w kolekcji	164
Unikaj jawnego definiowania schematu	172
Podstawowe operacje na bazach dokumentów	173
Wstawianie dokumentów do kolekcji	175
Usuwanie dokumentów z kolekcji	176
Aktualizacja dokumentów w kolekcji	177
Pobieranie dokumentów z kolekcji	178
Podsumowanie	180
Pytania kontrolne	180
Odniesienia	181

Rozdział 7. Terminologia baz dokumentów 183

Terminy dotyczące dokumentów i kolekcji	184
Dokument	184
Kolekcja	186
Dokument osadzony	187
Brak schematu	188
Schemat polimorficzny	190
Typy partycji	191
Partycjonowanie pionowe	192
Partycjonowanie poziome, czyli sharding	194
Modelowanie danych i przetwarzanie zapytań	198
Normalizacja	198
Denormalizacja	200
Procesor zapytań	200
Podsumowanie	201
Pytania kontrolne	201
Odniesienia	201

Rozdział 8. Projektowanie baz dokumentów 203

Normalizacja, denormalizacja i poszukiwanie równowagi	204
Relacja jeden-do-wielu	206
Relacja wiele-do-wielu	206
Potrzeba złączeń	206
Wykonywanie złączeń: podnoszenie ciężarów dla baz relacyjnych	208
Co zrobiłby projektant bazy dokumentów?	210
Planowanie z uwzględnieniem dokumentów zmiennych	215
Unikanie przenoszenia dużych dokumentów	218
Strefa Złotowłosej w indeksach	218
Aplikacje zorientowane na odczyt	218
Aplikacje zorientowane na zapis	219
Modelowanie powszechnych relacji	221
Relacja jeden-do-wielu w bazach dokumentów	221
Relacja wiele-do-wielu w bazach dokumentów	222
Modelowanie hierarchii w bazach dokumentów	223
Podsumowanie	225
Studium przypadku: manifesty użytkowników	226
Osadzać czy nie?	227
Wybór indeksów	228
Osobne kolekcje dla typów?	228

Pytania kontrolne	229
Odniesienia	229

Część IV Bazy rodziny kolumn 231

Rozdział 9. Wprowadzenie do baz rodziny kolumn 233

Na początku było Google BigTable	234
Wykorzystanie dynamicznej kontroli nad kolumnami	236
Indeksowanie po rekordzie, nazwie kolumny i stemplu czasowym	236
Kontrolowanie lokalizacji danych	237
Odczyt i zapis wierszy atomowych	237
Utrzymywanie posortowanych wierszy	238
Podobieństwa i różnice między bazami rodziny kolumn	
a bazami klucz-wartość i bazami dokumentów	240
Cechy baz rodziny kolumn	240
Podobieństwa i różnice między bazami rodziny kolumn	
i bazami dokumentów	241
Bazy rodziny kolumn kontra bazy relacyjne	242
Architektura baz rodziny kolumn	245
Architektura HBase: różnorodność węzłów	245
Architektura Cassandra: peer-to-peer	247
Rozgłaszanie: protokół plotki	248
Termodynamika i bazy rozproszone: po co nam antyentropia	250
Przechowaj to dla mnie: przekazanie ze wskazaniem	251
Kiedy korzystać z baz rodziny kolumn	252
Podsumowanie	254
Pytania kontrolne	254
Odniesienia	255

Rozdział 10. Terminologia baz rodziny kolumn 257

Podstawowe komponenty baz rodziny kolumn	258
Przestrzeń kluczy	258
Klucz wiersza	258
Kolumna	259
Rodziny kolumn	260
Struktury i procesy: implementacja baz rodziny kolumn	261
Wewnętrzne struktury i parametry konfiguracyjne	
baz rodziny kolumn	261
Starzy znajomi: klastry i partycje	262
Rzut oka pod maskę: inne komponenty baz rodziny kolumn	264

Procesy i protokoły	268
Replikacja	268
Antyentropia	268
Protokół plotki	269
Przekazanie ze wskazaniem	270
Podsumowanie	271
Pytania kontrolne	271
Odniesienia	272
Rozdział 11. Projektowanie baz rodziny kolumn	273
Wskazówki dotyczące projektowania tabel	275
Denormalizuj, zamiast łączyć	276
Wykorzystuj kolumny bez wartości	276
Używaj zarówno nazwy kolumny, jak i wartości kolumn do przechowywania danych	277
Modeluj encje za pomocą pojedynczego wiersza	278
Unikaj punktów zapalnych w kluczach wierszy	279
Utrzymuj odpowiednią liczbę wersji wartości kolumn	280
Unikaj rozbudowanych struktur danych w wartościach kolumn	281
Wskazówki dotyczące indeksowania	282
Kiedy korzystać z indeksów pomocniczych zarządzanych przez system bazy rodziny kolumn	282
Kiedy tworzyć indeksy pomocnicze i zarządzać nimi za pomocą tabeli	286
Narzędzia do pracy z bazami Big Data	288
Ekstrakcja, transformacja i ładowanie danych Big Data	289
Analizowanie danych Big Data	290
Narzędzia do monitorowania Big Data	293
Podsumowanie	294
Studium przypadku: analiza danych klienta	294
Zrozumienie potrzeb użytkownika	295
Pytania kontrolne	296
Odniesienia	297
Część V Bazy grafowe	299
Rozdział 12. Wprowadzenie do baz grafowych	301
Czym jest graf?	301
Modelowanie grafów i sieci	302
Modelowanie lokalizacji geograficznych	303
Modelowanie chorób zakaźnych	303

Modelowanie encji abstrakcyjnych i konkretnych	305
Modelowanie mediów społecznościowych	307
Zalety baz grafowych	308
Szybsze wykonywanie zapytań dzięki unikaniu złączeń	308
Upraszczenie modelowania	310
Wiele relacji pomiędzy encjami	310
Podsumowanie	311
Pytania kontrolne	311
Odniesienia	312
Rozdział 13. Terminologia baz grafowych	313
Elementy grafów	314
Wierzchołek	314
Krawędź	315
Ścieżka	317
Pętla	317
Operacje na grafach	318
Unia grafów	318
Przecięcie grafów	319
Przeszukiwanie grafu	320
Właściwości grafów i krawędzi	320
Izomorfizm	321
Rząd i rozmiar	321
Stopień	322
Bliskość	322
Pośrednictwo	322
Typy grafów	323
Grafy skierowane i nieskierowane	324
Sieć przepływowa	324
Grafy dwudzielne	325
Multigraf	325
Graf ważony	326
Podsumowanie	327
Pytania kontrolne	327
Odniesienia	327
Rozdział 14. Projektowanie baz grafowych	329
Początki projektowania grafów	329
Projektowanie bazy grafowej sieci społecznościowej	331
Projektowanie sterowane przez zapytania (znowu)	334

Odpytywanie grafu	336
Cypher: zapytania deklaratywne	336
Gremlin: zapytania przez trawersowanie grafu	337
Wskazówki i sztuczki przydatne w projektowaniu baz grafowych	341
Użyj indeksów do poprawienia czasu pobierania danych	342
Używaj krawędzi odpowiedniego rodzaju	342
Podczas przeszukiwania grafu uważaj na cykle	343
Weź pod uwagę skalowalność swojej bazy grafowej	344
Podsumowanie	345
Studium przypadku: optymalizacja tras transportowych	345
Zrozumieć potrzeby użytkownika	345
Projektowanie rozwiązania polegającego na analizie grafu	346
Pytania kontrolne	347
Odniesienia	347

Część VI Wybór bazy danych dla Twojej aplikacji 349

Rozdział 15. Wytyczne do wyboru bazy danych	351
Wybór bazy danych NoSQL	352
Przypadki użycia i kryteria wyboru baz klucz-wartość	353
Przypadki użycia i kryteria wyboru baz dokumentów	354
Przypadki użycia i kryteria wyboru baz rodziny kolumn	354
Przypadki użycia i kryteria wyboru baz grafowych	356
Używanie baz NoSQL i baz relacyjnych razem	357
Podsumowanie	358
Pytania kontrolne	358
Odniesienia	359

Dodatki 361

Dodatek A Odpowiedzi do pytań kontrolnych	363
Dodatek B Lista baz NoSQL	389
Dodatek C Słowniczek	393
Skorowidz	401

7

Terminologia baz dokumentów

*Musimy mieć odwagę myśleć o tym, co „nie do pomyślenia”.
Musimy nauczyć się eksplorować wszystkie opcje i możliwości,
które spotykają nas w skomplikowanym i gwałtownie zmieniającym się świecie.*
— J. William Fulbright, były senator USA

Tematy omawiane w tym rozdziale:

Terminy dotyczące dokumentów i kolekcji

Typy partycji

Modelowanie danych i przetwarzanie zapytań

W poprzednim rozdziale omówiliśmy podstawowe koncepcje związane z bazami dokumentów. W tym rozdziale skupimy się na zdefiniowaniu terminów powszechnie używanych w teorii i praktyce baz dokumentów.

Uwaga. Podobnie jak w przypadku innych rozdziałów dotyczących terminologii celem jest zdefiniowanie dokładnych opisów głównych terminów używanych w kontekście baz dokumentów. Część terminologii dotyczy również innych baz rozproszonych, podczas gdy niektóre pojęcia dotyczą wyłącznie baz dokumentów.

Pierwszy zestaw terminów jest związany z podstawowymi strukturami danych w bazie dokumentów. Powinieneś znać je z poprzedniego rozdziału, ale tutaj zostaną przedstawione w sposób bardziej formalny.

W drugim podrozdziale zdefiniowane zostaną terminy potrzebne podczas nauki architektury baz dokumentów. Terminy zdefiniowane w tym podrozdziale są używane

do ogólnego opisu baz dokumentów, ze szczególnym uwzględnieniem skalowania dużych baz. Będziesz się z nimi często spotykał podczas pracy z bazami NoSQL.

Ostatni podrozdział jest najbardziej heterogeniczny. Zawiera terminy związane z modelowaniem dokumentów i operacji. Terminy związane z modelowaniem są ogólnymi koncepcjami. W następnym rozdziale przedstawione zostaną bardziej szczegółowe wzorce projektowe. Na zakończenie podrozdziału omówię kilka ogólnych, ale ważnych terminów, które powinienś znać, pracując z bazą dokumentów.

Terminy dotyczące dokumentów i kolekcji

Dokumenty i kolekcje to podstawowe struktury bazy dokumentów. Są analogiczne do rekordów i tabel w bazach relacyjnych. Nieformalne wprowadzenie do tych terminów w rozdziale 6. „Wprowadzenie do baz dokumentów” było wystarczające, aby przekazać podstawowe koncepcje baz dokumentów. Czas teraz na bardziej formalne definicje, które dokładniej opiszą zasady rządzące tymi strukturami.

Zdefiniowane zostaną następujące terminy:

- dokument,
- kolekcja,
- dokument osadzony,
- brak schematu,
- schemat polimorficzny.

Na końcu tego podrozdziału powinienś dobrze rozumieć, w jaki sposób dokumenty są organizowane w kolekcje. Powinieneś również rozumieć ważne właściwości organizacji bazy dokumentów, które pozwalają na tworzenie elastycznych projektów. Jest to jeden z głównych powodów, dla których bazy dokumentów zyskały popularność pośród programistów.

Dokument

Dokument jest zestawem uporządkowanych par klucz-wartość. **Klucz-wartość** to struktura, która składa się z dwóch elementów, co oczywiste, klucza i wartości.

Uwaga. Dla osób, które opuściły rozdziały poświęcone bazom klucz-wartość: **klucz** jest unikatowym identyfikatorem używanym do odnalezienia wartości, **wartość** jest instancją dowolnego wspieranego typu danych, takiego jak ciąg znaków, numer, tablica lub lista.

Dokumenty: uporządkowany zestaw par klucz-wartość

Ponieważ dokument jest zestawem, posiada po jednej instancji każdego elementu. Elementy to pary klucz-wartość. Przykładowo poniżej znajduje się zestaw z trzema elementami: 'foo': 'a', 'bar': 'b' i 'baz': 'c':

```
{ 'foo': 'a', 'bar': 'b', 'baz': 'c' }
```

Wystarczy drobna zmiana, aby zestaw przestał być zestawem (i stał się *torbą*):

```
{ 'foo': 'a', 'bar': 'b', 'baz': 'c', 'foo': 'a' }
```

Ta lista nie jest zestawem, ponieważ występują dwie instancje pary 'foo': 'a'.

Zestawy nie są rozróżniane na podstawie kolejności. Poniższy zestaw:

```
{ 'foo': 'a', 'bar': 'b', 'baz': 'c' }
```

jest równoważny z:

```
{ 'baz': 'c', 'foo': 'a', 'bar': 'b' }
```

Jednak z punktu widzenia projektowania baz dokumentów są to różne dokumenty. Kolejność par klucz-wartość ma znaczenie w określaniu identyfikacji dokumentu.

Dokument { 'foo': 'a', 'bar': 'b', 'baz': 'c' } nie jest równoznaczny z dokumentem { 'baz': 'c', 'foo': 'a', 'bar': 'b' }.

Klucze i typy wartości

Klucze to ciągi znaków. Niektóre bazy klucz-wartość wspierają bardziej rozbudowany zestaw typów danych dla kluczy, więc bazy dokumentów mogłyby również wspierać wiele typów danych dla kluczy.

Wartości mogą być różnego typu. Jak możesz się spodziewać, bazy dokumentów wspierają wartości liczbowe i ciągi znaków. Wspierają również bardziej strukturalne typy danych, takie jak tablice i inne dokumenty.

Tablice są użyteczne, kiedy musisz śledzić wiele instancji wartości, a wszystkie wartości są tego samego typu. Jeżeli na przykład musisz zamodelować pracownika i listę jego projektów, możesz skorzystać z dokumentu takiego jak poniżej:

```
{ 'nazwaPracownika' : 'Julia Kowalska',  
  'dzial' : 'Wytwarzanie oprogramowania',  
  'dataRozpoczecia' : '10/02/2010',  
  'kodyZakonczonychProjektow' : [ 189847, 187731, 176533, 154812 ]  
}
```

Klucz kodyZakonczonychProjektow zawiera listę numerów projektów. Wszystkie kody projektów to numery, dlatego użycie tablicy jest poprawne.

Ewentualnie, jeżeli wraz z danymi pracowników chcesz przechować lub osadzić więcej informacji o projektach, możesz w ramach dokumentu pracownika zamieścić inny dokument. Taki dokument mógłby wyglądać tak:


```
{ 'nazwaPracownika' : 'Julia Kowalska',
  'dzial' : 'Wytwarzanie oprogramowania',
  'dataRozpoczecia' : '10/02/2010',
  'zakonczoneProjekty' : {
    { 'kodProjektu' : 189847,
      'nazwaProjektu' : 'System rekomendacji produktów',
      'menadzerProjektu' : 'Jagoda Dulska' },
    { 'kodProjektu' : 187731,
      'nazwaProjektu' : 'Wymiana danych finansowych, wersja 3',
      'menadzerProjektu' : 'Jakub Rosiewicz'},
    { 'kodProjektu': 176533,
      'nazwaProjektu' : 'Uwierzytelnianie klientów',
      'menadzerProjektu' : 'Michał Krawiec'},
    { 'kodProjektu': 154812,
      'nazwaProjektu' : 'Miesięczny raport sprzedaży',
      'menadzerProjektu': 'Bożena Rondeł' }
  }
}
```

Podsumowując: dokumenty to uporządkowane zestawy par klucz-wartość. Klucze są wykorzystywane do odnajdywania konkretnych wartości. Wartości mogą być albo typu prostego, albo strukturalnego.

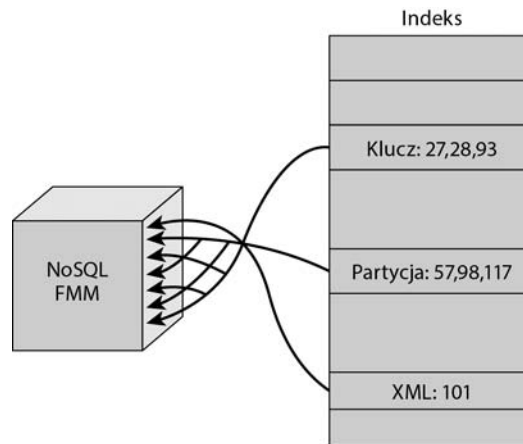
Kolekcja

Kolekcja to grupa dokumentów. Dokumenty w ramach kolekcji są zazwyczaj spokrewnione z tą samą encją obiektu, taką jak pracownicy, produkty, zalogowane zdarzenia lub profile użytkowników. Możliwe jest przechowywanie w kolekcji niezwiązanych ze sobą dokumentów, ale nie jest to polecane.

Mówiąc najbardziej ogólnie: kolekcje pozwalają operować na grupach spokrewnionych dokumentów. Jeżeli utrzymujesz kolekcję informacji o pracownikach, możesz przejrzeć wszystkie rekordy w kolekcji i odnaleźć konkretnych pracowników, na przykład takich, którzy rozpoczęli pracę przed 1 stycznia 2011 roku. Jeżeli masz dużą liczbę pracowników, takie przeszukiwanie może być niewydajne, ponieważ musiałbyś porównać dane każdego pracownika z kryteriami wyszukiwania.

Oprócz możliwości łatwego wykonywania operacji na grupach dokumentów kolekcje wspierają dodatkowe struktury, które sprawiają, że te operacje są bardziej wydajne. Przykładowo bardziej wydajne podejście do skanowania wszystkich dokumentów w kolekcji to wykorzystanie indeksów. Indeksy w kolekcjach są jak indeksy na końcu książki: zestaw informacji mapujących jeden atrybut, taki jak hasło **klucz**, do powiązanych z nim informacji, takich jak lista stron (patrz rysunek 7.1).

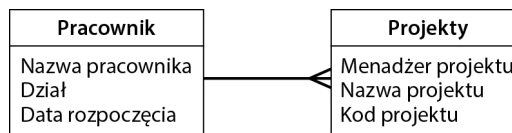
Kolekcje to grupy podobnych dokumentów, dzięki którym łatwo dotrzeć do wszystkich dokumentów w grupie lub wykonywać na nich operacje. Kolekcje wspierają dodatkowe struktury, takie jak indeksy, które poprawiają wydajność operacji na tych grupach dokumentów.



Rysunek 7.1. Indeksy mapują atrybuty, takie jak hasła, do powiązanych z nimi informacji, takich jak numery stron. Korzystanie z indeksu jest szybsze niż skanowanie całej książki

Dokument osadzony

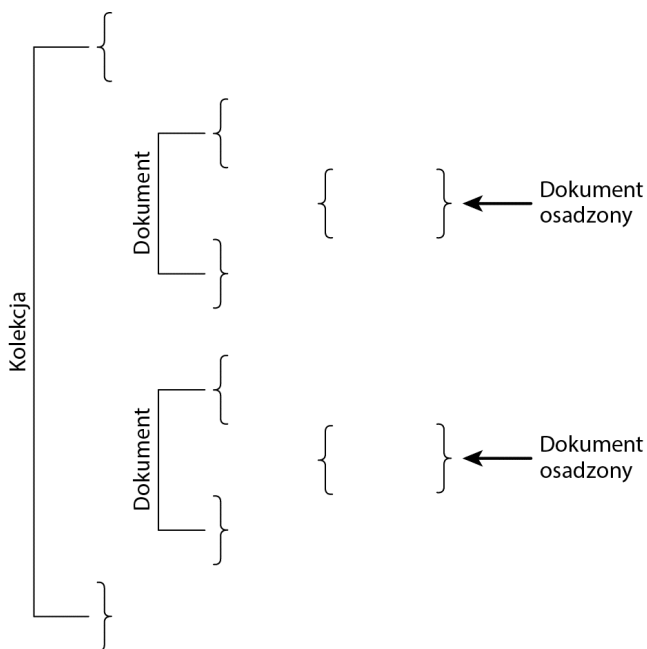
Jedną z zalet baz dokumentów jest to, że pozwalają programistom przechowywać spokrewnione dane w sposób bardziej elastyczny niż w przypadku baz relacyjnych. Gdybyś miał zamodelować pracowników i projekty, nad którymi pracują, prawdopodobnie stworzyłbyś dwie tabele: jedną przechowującą informacje o pracownikach i drugą przechowującą informacje o projektach (patrz rysunek 7.2).



Rysunek 7.2. Modele relacyjne dzielą dane różnych encji na osobne tabele. Wymaga to sprawdzania informacji w obu tabelach z wykorzystaniem procesu złączania

Dokument osadzony pozwala użytkownikom bazy dokumentów na przechowywanie spokrewnionych danych w jednym dokumencie. Dzięki temu baza dokumentów może uniknąć procesu złączania, polegającego na tym, że dane z jednej tabeli są używane do odnalezienia danych w innej tabeli (klucz obcy).

Złączenie dwóch dużych tabel może potencjalnie być czasochłonne i wymagać znacznej liczby odczytów z dysku. Dokumenty osadzone pozwalają na wspólne przechowywanie spokrewnionych danych. Kiedy dokument jest odczytywany z dysku, zarówno dane główne, jak i dane powiązane są odczytywane bez potrzeby ich złączania. Rysunek 7.3 przedstawia dokumenty osadzone w innym dokumencie.



Rysunek 7.3. Dokumenty osadzone to dokumenty zapisane w innym dokumencie. Osadzanie jest używane do efektywnego przechowywania i pobierania danych, które często są wykorzystywane razem

Dokumenty osadzone to dokumenty zapisane w innym dokumencie. Są używane do podnoszenia wydajności bazy przez wspólne przechowywanie danych, które często są wykorzystywane razem.

Brak schematu

Bazy dokumentów nie wymagają od osób modelujących dane formalnej specyfikacji struktury dokumentów. Formalna specyfikacja struktury nazywana jest **schematem**. Bazy relacyjne wymagają schematu. Schemat zawiera zazwyczaj specyfikacje:

- tabel,
- kolumn,
- kluczy głównych,
- kluczy obcych,
- ograniczeń.

Pozwalają one systemowi zarządzania bazą relacyjną na zarządzanie danymi w bazie. Pomagają również wykrywać błędy podczas dodawania danych do bazy. Jeżeli na przykład ktoś spróbuje dodać ciąg znaków w miejscu, w którym wymagana jest liczba, system zarządzania bazą danych zgłosi błąd.

Ograniczenia to reguły opisujące, jakie dane lub relacje pomiędzy danymi są dozwolone. W schemacie można wskazać, że kolumna zawsze musi posiadać wartość i nigdy nie może być pusta.

Wskazówka. Pusta kolumna w bazie relacyjnej przyjmuje wartość NULL.

Bazy dokumentów nie wymagają specyfikacji przed rozpoczęciem dodawania dokumentów do bazy. Z tego powodu bazy dokumentów są nazywane bazami bez schematu. Bazy bez schematu różnią się od baz relacyjnych w dwóch zasadniczych sprawach:

- dają większą elastyczność,
- wymagają większej odpowiedzialności.

Brak schematu oznacza większą elastyczność

W bazie bez schematu programiści i aplikacje mogą do dokumentów dodawać nowe pary klucz-wartość w dowolnym momencie. Po stworzeniu kolekcji możesz dodawać do niej dokumenty. Nie ma potrzeby informowania bazy dokumentów o strukturze dokumentów. Sama struktura dokumentów będzie często różna w różnych dokumentach w kolekcji. Poniższe dwa dokumenty:

```
{ 'nazwaPracownika' : 'Julia Kowalska',  
  'dzial' : 'Wytwarzanie oprogramowania',  
  'dataRozpoczecia' : '10/02/2010',  
  'kodyZakonczonejProjektow' : [ 189847, 187731, 176533, 154812 ]  
}  
  
i  
  
{ 'nazwaPracownika' : 'Robert Lucek',  
  'dzial' : 'Finanse',  
  'dataRozpoczecia' : '21/05/2009',  
  'certyfikaty' : 'CPA'  
}
```

opisują pracowników, ale pierwszy jest przystosowany dla osoby z działu wytwarzania oprogramowania, a drugi został zaprojektowany z myślą o pracowniku działu finansów.

Te dokumenty i ich wariacje mogą być po prostu dodawane do kolekcji. Nie ma potrzeby specyfikowania, że niektóre dokumenty będą posiadały klucz kodyZakonczonejProjektow, a inne klucz certyfikaty. Nie ma również potrzeby określania, że niektóre wartości będą ciągami znaków, a inne tablicami.

Uwaga. Baza dokumentów czerpie potrzebne jej informacje ze struktury dokumentów w kolekcji, nie z osobnej specyfikacji struktury.

Brak schematu oznacza większą odpowiedzialność

Bazy bez schematu są swego rodzaju bronią obosieczną. Z jednej strony elastyczność pracy bez schematu pozwala łatwo radzić sobie z różnicami w strukturach dokumentów. Z drugiej jednak strony baza dokumentów nie może narzucać reguł opartych na strukturze. Ponieważ nie ma możliwości określenia, że para klucz-wartość powinna zawsze istnieć w dokumencie, system zarządzania bazą dokumentów nie będzie tego sprawdzał.

Jeżeli system zarządzania bazą nie zapewnia reguł dotyczących danych, co ma się tym zająć? Odpowiedź brzmi — Twój kod.

Wskazówka. Wyjątkiem od tej reguły jest wykorzystanie unikatowych identyfikatorów. Jeżeli spróbujesz wstawić dokument bez unikatowego identyfikatora, baza dokumentów prawdopodobnie go uzupełni. Szczegółowe informacje na ten temat znajdziesz w dokumentacji swojej bazy.

Część kodu Twojej aplikacji powinna być dedykowana sprawdzaniu reguł dotyczących struktury danych. Jeżeli w dokumencie pracownika zawsze powinna być zawarta jego nazwa, Twój kod dodający pracowników powinien to sprawdzać podczas umieszczania dokumentów w bazie. Jest to prosty przykład walidacji danych, jednak nie wszystkie przypadki są tak proste.

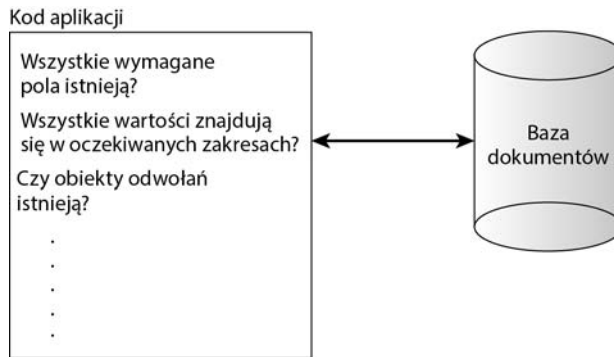
Z czasem klucze i wartości przechowywane w dokumentach w kolekcji mogą się zmieniać. Być może zacząłeś zbierać informacje o certyfikatach pracowników w zeszłym roku. Wszyscy pracownicy, którzy od tamtego czasu zostali dodani do bazy, mogą mieć klucz dotyczący certyfikatów. Pracownicy, którzy zostali zaktualizowani, również mogą mieć klucz i wartość dla certyfikatów. Pozostałe dokumenty nie będą miały tego klucza.

W tej sytuacji to kod, który używa i przetwarza dokumenty pracowników (nie kod dodający dokumenty), musi sprawdzać poprawność struktur danych albo przynajmniej obsłużyć przypadek, w którym oczekiwany klucz nie istnieje (patrz rysunek 7.4).

Bazy bez schematów nie wymagają formalnej specyfikacji struktury. Potrzebne informacje są pozyskiwane z dokumentów w kolekcji. To pozwala uzyskać większą elastyczność niż w bazach wymagających schematu, ale daje mniejsze możliwości automatycznego sprawdzania danych i integralności dokumentów.

Schemat polimorficzny

Innym terminem, z jakim możesz się spotkać w kontekście baz dokumentów, jest schemat polimorficzny. Może się wydawać dziwne, że bazy mogą nie mieć schematu (**brak schematu**) i jednocześnie mieć wiele schematów (**schemat polimorficzny**). Jest to jednak całkiem



Rysunek 7.4. Kod walidujący dane i obsługujący błędy jest używany w aplikacjach po to, aby zastąpić automatyczną, opartą na schemacie walidację

logiczne, jeżeli dostrzeżesz różnicę pomiędzy formalną specyfikacją struktury i strukturą wynikającą z dokumentów w kolekcji.

Wskazówka. Jeszcze raz. Bazy dokumentów nie mają schematu, ponieważ nie musisz specyfikować formalnej definicji struktury dokumentów, kluczy i wartości.

Baza dokumentów jest polimorficzna, ponieważ dokumenty istniejące w kolekcjach mogą mieć wiele różnych form (patrz rysunek 7.5).

Typy partycji

Partycjonowanie to słowo, które bardzo często jest używane w świecie NoSQL — być może za często.

W rozdziale 2. omówiliśmy teorię CAP — jak być może pamiętasz, teoria ta opisuje ograniczenia spójności, dostępności i tolerancji dla partycjonowania. W tym kontekście słowo **partycjonowanie** odnosi się do partycjonowania lub separowania sieci na odrębne części, które są niedostępne dla siebie nawzajem.

Jest to ważna koncepcja dla wszystkich rozproszonych baz danych, ale w kontekście baz dokumentów partycjonowanie oznacza coś innego. Kiedy podczas dyskusji na temat baz dokumentów ludzie używają pojęcia **partycjonowanie**, prawdopodobnie chodzi im o dzielenie bazy i dystrybuowanie różnych jej części na różne serwery.

Polimorfizm

```

{
    { 'a' : 1,
      'b' : 2,
      'c' : 3
    }

    { 'a' : 7,
      'b' : 8
      'd' : 10
    }

    { 'a' : 20,
      'e' : 30,
      'f' : 40,
      'g' : 50
    }
}

```

Rysunek 7.5. Brak schematu oznacza, że nie ma formalnej definicji struktury.

Schemat polimorficzny oznacza, że istnieje wiele struktur dokumentów wynikających z dokumentów występujących w kolekcji

Są dwa typy partycjonowania bazy: partycjonowanie pionowe i partycjonowanie poziome.

Należy różnicować znaczenie pojęcia **partycjonowanie** na podstawie kontekstu, w jakim zostanie użyte (patrz rysunek 7.6).

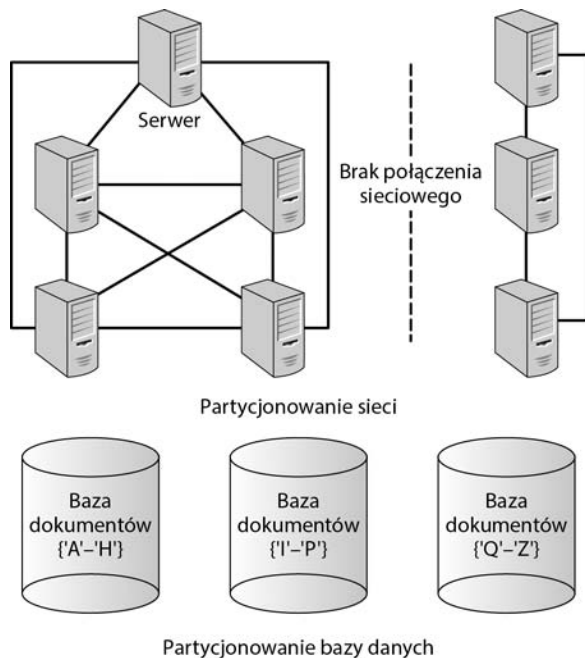
Partycjonowanie pionowe

Partycjonowanie pionowe jest techniką poprawiania wydajności bazy danych przez rozdzielenie kolumn tabeli relacyjnej do kilku odrębnych tabel (patrz rysunek 7.7).

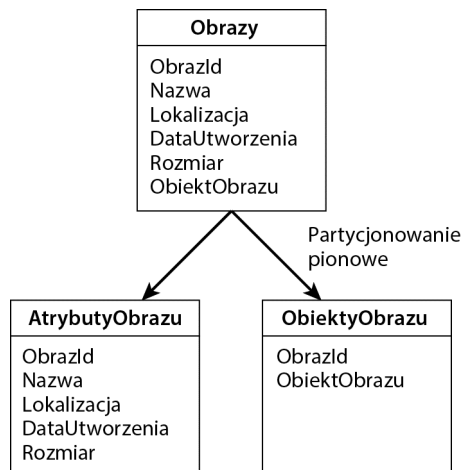
Ta technika jest szczególnie użyteczna, kiedy masz kolumny, które są często pobierane, podczas gdy inne są pobierane rzadziej. Rozważmy tabelę z obrazami i atrybutami tych obrazów, takimi jak nazwa, lokalizacja czy data utworzenia obrazu. Tabela może być wykorzystywana w aplikacji pozwalającej wyszukiwać obrazy na podstawie ich charakterystyk.

Ktoś może poszukiwać zdjęć Paryża zrobionych w ciągu ostatnich trzech miesięcy. System zarządzania bazą danych do odnalezienia rekordów spełniających kryteria wyszukiwania najprawdopodobniej wykorzystałby indeks. Jeżeli aplikacja w wynikach zwraca tylko listę atrybutów i przed wyświetleniem obrazu czeka, aż użytkownik wybierze konkretny rekord, nie ma potrzeby, aby z bazy pobierać obraz razem z atrybutami.

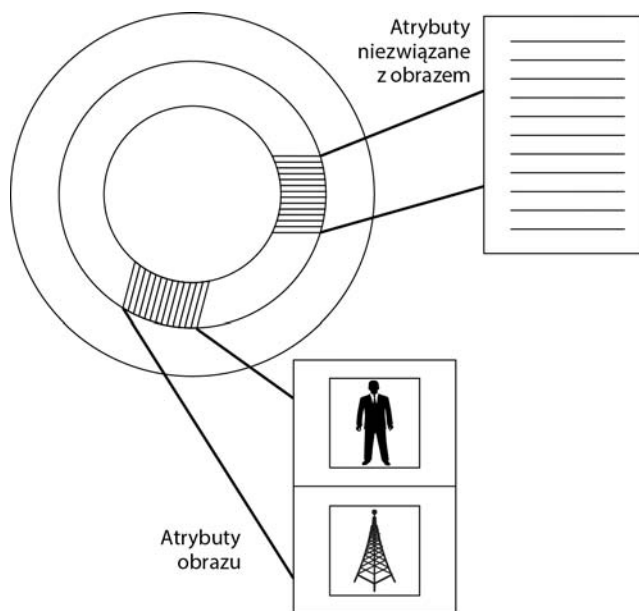
Jeżeli atrybuty obrazu i sam obraz byłyby przechowywane w tej samej tabeli, to z powodu rozkładu danych na dysku próba odczytania atrybutów mogłaby się skończyć odczytaniem także obrazu. Dzięki podzieleniu tabeli obrazu na tabelę zawierającą atrybuty i tabelę zawierającą obraz baza może efektywnie pobierać dane dla aplikacji (patrz rysunek 7.8).



Rysunek 7.6. Termin partycjonowanie ma wiele znaczeń, które są zależne od kontekstu, innego na przykład w dyskusji na temat sieci i w dyskusji na temat baz danych



Rysunek 7.7. Partycjonowanie pionowe jest zazwyczaj wykorzystywane w tabelach relacyjnych, ponieważ mają one ustaloną strukturę



Rysunek 7.8. Rozdzielanie kolumn na różne tabele może poprawić efektywność odczytów, ponieważ dane, które nie będą potrzebne (na przykład obiekt obrazu), nie są pobierane wraz z danymi, które są potrzebne (na przykład atrybuty obrazu)

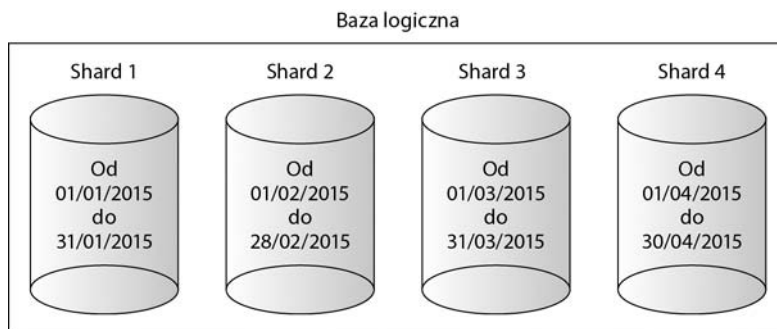
Partycjonowanie pionowe jest częściej wykorzystywane w bazach relacyjnych niż w bazach dokumentów. Są metody implementowania partycjonowania pionowego w bazach nierelacyjnych, ale partycjonowanie poziome (lub sharding) jest bardziej powszechne.

Uwaga. Przykład shardingu znajdziesz w artykule autorstwa J. Kaura i innych *A New and Improved Vertical Partitioning Scheme for Non-Relational Databases Using Greedy Method*, opublikowanym w numerze 8 „International Journal of Advanced Research in Computer and Communication Engineering” (sierpień 2013).

Partycjonowanie poziome, czyli sharding

Partycjonowanie poziome to proces dzielenia bazy danych na poziome dokumentów w bazie dokumentów lub na poziomie wierszy w bazie relacyjnej.

Takie części bazy danych, nazywane shardami, są przechowywane na odrębnych serwerach (poziome partycjonowanie bazy dokumentów jest też nazywane **shardingiem**). Jeżeli baza zostanie skonfigurowana do replikowania danych, pojedynczy shard może być przechowywany na wielu serwerach. Niezależnie od tego, czy dane są replikowane, czy nie, pojedynczy serwer w klastrze będzie przechowywał tylko jeden shard (patrz rysunek 7.9).



Rysunek 7.9. Partycjonowanie poziome dzieli bazę na poziomie dokumentów lub wierszy i dystrybuje części bazy, nazywane shardami, na różne serwery. Kiedy klastrer zaimplementuje replikację, pojedynczy shard będzie dostępny na wielu serwerach

Sharding ma wiele zalet w przypadku implementacji dużych baz dokumentów. Znaczna liczba użytkowników lub inne pracochłonne zadania mogą mocno obciążać dostępne procesory, pamięć i łącze. Jednym ze sposobów radzenia sobie z taką sytuacją jest wykorzystanie większego serwera, z większą liczbą procesorów i ilością pamięci oraz lepszym łączem.

To rozwiązanie, nazywane skalowaniem pionowym, może wymagać znacznie większych nakładów finansowych i większej ilości czasu niż sharding. Dodatkowe serwery mogą być dodawane do klastra wraz z powiększaniem się bazy dokumentów. Istniejące serwery nie są zastępowane i pozostają w ciągłym użytku.

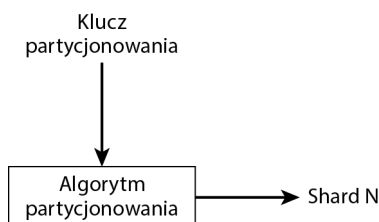
Aby zaimplementować sharding, projektanci baz danych muszą wybrać klucz i metodę partycjonowania. Tematy te zostaną omówione w następnych podrozdziałach.

Dzielenie danych za pomocą klucza partycjonowania

Klucz partycjonowania to jeden klucz lub więcej kluczy istniejących we wszystkich dokumentach w kolekcji i wykorzystywanych do podzielenia dokumentów. Klucz partycjonowania może być dowolnym atomowym polem dokumentu:

- unikatowym identyfikatorem dokumentu,
- nazwą,
- datą (na przykład datą utworzenia),
- kategorią lub typem,
- regionem geograficznym.

Klucz partycjonowania określa, jakie wartości mają być używane podczas grupowania dokumentów w shardy. Algorytm partycjonowania wykorzystuje klucz partycjonowania jako wsad i determinuje odpowiedni shard dla klucza (patrz rysunek 7.10).



Rysunek 7.10. Klucze partycjonowania to dane wsadowe dla algorytmu partycjonowania zwracającego shard

Uwaga. Pomimo tego, że bazy dokumentów nie mają schematu, niektóre elementy tych baz są podobne do schematu baz relacyjnych. Wykorzystanie indeksów jest jednym z takich podobieństw. Indeksy są częścią fizycznego modelu danych bazy relacyjnej, co oznacza, że w bazie danych istnieje struktura implementująca indeks. Indeksy są częścią schematu baz relacyjnych. W bazach bez schematu, takich jak bazy dokumentów, mogą występować obiekty podobne do obiektów schematu, takie jak indeksy. Indeksy pomagają poprawić szybkość operacji odczytu i są użyteczne podczas implementowania shardingu. Ponieważ wszystkie dokumenty w kolekcji muszą być umieszczone w shardzie, ważne jest, aby wszystkie miały klucz partycjonowania.

Dystrybuowanie danych przy pomocy algorytmu partycjonowania

Jest wiele sposobów na poziome partycjonowanie danych, włączając w to:

- zakresy,
- hasze,
- listy.

Partycjonowanie zakresami jest użyteczne, kiedy masz uporządkowany zestaw wartości dla kluczy partycjonowania, taki jak daty lub liczby. Przykładowo jeżeli wszystkie dokumenty w kolekcji mają pole z datą utworzenia, można wykorzystać to pole do partycjonowania danych na miesięczne shardy. Dokumenty stworzone pomiędzy 1 stycznia 2015 roku i 31 stycznia 2015 roku zostałyby zapisane w jednym shardzie, podczas gdy dokumenty stworzone pomiędzy 1 lutego 2015 roku i 28 lutego 2015 roku zostałyby zapisane w innym.

Uwaga. Systemy analizy biznesowej, które tworzą raporty dla określonego zakresu dat — na przykład raport porównujący sprzedaż z tego miesiąca ze sprzedażą z poprzedniego miesiąca — zazwyczaj wykorzystują partycjonowanie na podstawie czasu.

Partycjonowanie za pomocą haszy wykorzystuje funkcję haszującą do określenia, gdzie umieścić dokument. Funkcje haszujące zostały zaprojektowane tak, aby generowały wartości równomiernie rozłożone w ramach zakresu wartości funkcji haszującej. Jeżeli na przykład masz klaster złożony z ośmiu serwerów, a Twoja funkcja haszująca generuje wartości pomiędzy 1 a 8, powinieneś mieć prawie równą liczbę dokumentów na każdym z ośmiu serwerów.

Partycjonowanie na podstawie list wykorzystuje zestaw wartości do określenia lokalizacji danych. Możesz wyobrazić sobie bazę produktów z kilkoma typami, między innymi elektronika, drobne urządzenia, artykuły AGD, książki i ubrania. Te typy produktów mogą być wykorzystane jako klucz partycjonowania.

Jeżeli potrzebujesz więcej partycji, możesz połączyć typy produktów z innym polem, takim jak na przykład region sprzedaży, który jako wartości może przyjmować: północny wschód, południowy wschód, środkowy zachód, północny zachód i południowy zachód. Każdy z pięciu typów produktów może być wykorzystany z każdym z pięciu regionów sprzedaży, co daje dwadzieścia pięć możliwych partycji, w tym:

- elektronika-północny-wschód,
- elektronika-południowy-wschód,
- elektronika-środkowy-zachód,
- elektronika-południowy-zachód,
- elektronika-północny-zachód,
- urządzenia-północny-wschód,
- urządzenia-południowy-wschód,
- urządzenia-środkowy-zachód,
- i tak dalej...

Partycjonowanie to podstawowy proces pozwalający na skalowanie baz dokumentów wraz z rosnącym zapotrzebowaniem aplikacji, które mają dużą liczbę użytkowników lub wykonują dużo operacji. Partycjonowanie pionowe jest możliwe w przypadku baz dokumentów. Partycjonowanie poziome jest często stosowane.

Programiści korzystający z baz dokumentów mogą wybierać klucze używane podczas partycjonowania. Ale to programiści systemów zarządzania bazami dokumentów wybierają algorytmy partycjonowania wykorzystywane w bazie.

Ostatni podrozdział tego rozdziału wprowadza kilka nowych terminów niepasujących do żadnej z poprzednich kategorii. Powinieneś jednak zrozumieć te terminy, zanim przejdziemy do omawiania modelowania baz dokumentów w rozdziale 8.

Modelowanie danych i przetwarzanie zapytań

Bazy dokumentów są elastyczne. Mogą przechowywać szeroki wachlarz typów dokumentów i ich wariacji w ramach kolekcji dokumentów. Gdybyś miał teraz usiąść i zacząć projektować bazę dokumentów, prawdopodobnie zacząłbyś od listy zapytań, które chciałbyś uruchamiać w bazie (a przynajmniej powinieneś od tego zacząć). Gdybyś projektował bazę relacyjną, zacząłbyś prawdopodobnie od przemyślenia encji, które masz zamodelować, i ich wzajemnych relacji.

Kiedy już z grubsza zrozumiałbyś encje i relacje, zająłbyś się prawdopodobnie czymś, co nazywa się normalizacją. Jeżeli napotkasz problemy wydajnościowe z bazą, możesz zająć się procesem o nazwie denormalizacja. Ten proces byłby kierowany (przynajmniej do pewnego stopnia) informacjami uzyskanymi na podstawie niewydajnych zapytań z procesora zapytań.

Powinieneś znać zagadnienia normalizacji i denormalizacji, ponieważ najprawdopodobniej napotkasz je podczas pracy z bazami dokumentów. W przypadku baz dokumentów proces jest mniej formalny niż w przypadku baz relacyjnych, dlatego poniższe wyjaśnienia będą znacznie prostsze niż te, które znajdziesz w książkach poświęconych bazom relacyjnym.

Bazy dokumentów również implementują procesory zapytań, które próbują znaleźć optymalną sekwencję kroków prowadzących do pobrania szukanych danych.

Normalizacja

Normalizacja baz danych jest procesem organizacji danych w tabelach w taki sposób, aby ograniczyć potencjalne anomalie danych. **Anomalia** to niespójność danych. Załóżmy na przykład, że w swojej bazie przechowujesz tabelę przedstawioną w tabeli 7.1. Użytkownik odpytuje tę tabelę, szukając adresu klienta o nazwie Jolanta Wasilewska. Jakie adresy powinny zostać zwrócone?

Tabela 7.1. Adresy klientów

Numer zamówienia	Nazwa klienta	Adres klienta
9837373	Jolanta Wasilewska	ul. Parkowa 50, 81-549 Gdynia
9837374	Robert Daniel	os. Młodych, 58-200 Dzierżoniów
9837375	Tomasz Kowal	ul. Kościuszki 12/1, 63-900 Rawicz
9837376	Jolanta Wasilewska	ul. Stanisława Maczka 60/2, 81-417 Gdynia

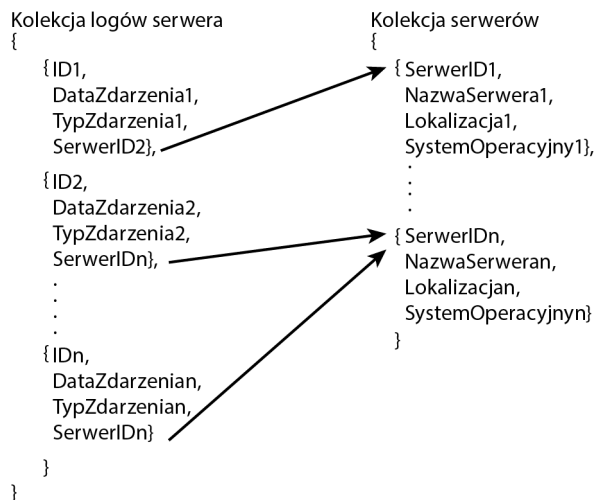
Zapytanie może zwrócić wartość *ul. Parkowa 50, 81-549 Gdynia* lub *ul. Stanisława Maczka 60/2, 81-417 Gdynia*, lub obie naraz. To możliwe, że Jolanta Wasilewska mieszka pod obydwooma adresami, ale również możliwe jest, że jeden z adresów jest aktualny, a drugi nie. W tabeli 7.1 nie ma na ten temat informacji.

Normalizacja redukuje ilość nadmiarowych danych w bazie. Nazwy klientów i ich adresy nie byłyby powtarzane przy każdym zamówieniu, tylko zostałyby umieszczone w osobnej tabeli. Zarówno do klienta, jak i do adresu mogłyby zostać przypisane dodatkowe atrybuty. W szczególności tabela z adresem mogłaby mieć flagę aktywnego adresu, pokazującą, który z wielu adresów jest aktualny.

Jest kilka zasad normalizacji baz danych. Bazy mogą występować w różnych formach normalnych, w zależności od tego, ile zasad normalizacyjnych jest przestrzeganych. Najczęściej projektanci dążą do tak zwanej trzeciej postaci normalnej, która oznacza spełnienie pierwszych trzech zasad normalizacji¹.

Normalizacja jest czasami używana do opisu sposobu projektowania dokumentów. Kiedy projektanci używają wielu kolekcji do przechowywania powiązanych ze sobą danych, mówi się, że dane są znormalizowane.

Dokumenty znormalizowane implikują to, że będziesz miał referencje do innych dokumentów, w których zapisane są dodatkowe dokumenty. Przykładowo dokument z logami serwera może posiadać pole z identyfikatorem serwera, który wygenerował te dane. Kolekcja dokumentów dla serwerów posiadałaby dodatkowe informacje o każdym serwerze, dzięki czemu te dane nie musiałyby być powtarzane w każdym dokumencie logów (patrz rysunek 7.11).



Rysunek 7.11. Dokumenty znormalizowane redukują nadmiarowość danych, przechowując referencje do pojedynczej kopii danych, zamiast powtarzać je w każdym dokumencie

¹ Wprowadzenie do normalizacji znajdziesz w artykule *A Simple Guide to Five Normal Forms in Relational Database Theory* Williama Kenta z września 1982 roku, <http://www.bkent.net/Doc/simple5.htm>.

Denormalizacja

Normalizacja pomaga uniknąć anomalii danych, ale może powodować problemy wydajnościowe. Może się tak zdarzyć, kiedy pobierasz dane z dwóch tabel lub więcej. Proces ten jest nazywany **złączaniem** i jest podstawową operacją w bazach relacyjnych. Dużo wysiłku zostało włożone w stworzenie efektywnych sposobów łączenia danych. Administratorzy baz danych i osoby modelujące dane mogą spędzać dużo czasu, próbując poprawić wydajność operacji złączania. Nie zawsze to się udaje.

Projektowanie baz danych wymaga kompromisów. Możesz zaprojektować wysoko znormalizowaną bazę danych bez żadnych zduplikowanych danych, ale za to mało wydajną. W takim wypadku wielu projektantów zwraca się w kierunku denormalizacji.

Jak sugeruje nazwa, denormalizacja jest odwrotnością normalizacji — wprowadza duplikację danych. Możesz się zastanawiać, po co wprowadzać nadmiarowe dane. Może to powodować anomalie danych, takie jak przedstawione w tabeli 7.1. Oczywiście wymaga to też większej powierzchni przechowywania. Ryzykuje się występowanie anomalii danych dlatego, że denormalizacja może znacząco poprawić wydajność.

Kiedy dane są zdenormalizowane, nie ma potrzeby odczytywania danych z wielu tabel i wykonywania złączeń z wielu kolekcji. Dane są pobierane z jednej kolekcji lub dokumentu. Może to być znacznie szybsze niż pobieranie z wielu kolekcji, w szczególności jeżeli dostępne są indeksy.

Procesor zapytań

Pobieranie danych z bazy dokumentów jest bardziej skomplikowane niż pobieranie ich z bazy klucz-wartość. Pamiętaj, że z bazy klucz-wartość, jeżeli masz klucz, możesz pobrać przypisaną do niego wartość.

Bazy dokumentów oferują większe możliwości pobierania danych. Przykładowo możesz pobrać dokumenty stworzone przed wskazaną datą, dokumenty, które są określonego typu, dokumenty zawierające ciąg znaków **bieganie długodystansowe** w opisie produktu, lub nawet kombinację wszystkich powyższych.

Procesor zapytań jest ważną częścią systemu zarządzania bazą danych. Przyjmuje zapytania oraz dane o dokumentach i kolekcjach w bazie, a następnie zwraca sekwencję operacji pobierających wybrane dane.

Bazy klucz-wartość nie potrzebują procesora zapytań; pobierają wartości za pomocą klucza. Nie ma potrzeby analizowania poleceń logicznych, takich jak poniżej:

```
(dataUtworzenia > '01/01/2015') AND (typProduktu = 'elektronika')
```

Jeżeli istnieje kilka warunków pobierania danych, procesor zapytań musi podjąć decyzję, na przykład które kryteria powinny zostać zaaplikowane jako pierwsze. Czy najpierw powinien znaleźć dokumenty z datą utworzenia po 1 stycznia 2015 roku czy wszystkie produkty elektroniczne?

Jeżeli dokumentów utworzonych po 1 stycznia 2015 roku jest mniej niż dokumentów dla urządzeń elektronicznych, sensowne będzie pobranie dokumentów po dacie utworzenia,

ponieważ wyników będzie mniej. To oznacza, że drugie kryterium będzie aplikowane do mniejszego zbioru dokumentów.

Jest to prosty przykład decyzji, jakie musi podejmować procesor zapytań podczas budowania planu zapytania.

Podsumowanie

Bazy dokumentów posiadają terminologię specyficzną dla tego typu bazy NoSQL, ale współdzielą również część pojęć z innymi bazami NoSQL i bazami relacyjnymi. Dokumenty są równoznaczne z wierszami w bazach relacyjnych, a kolekcje z tabelami. Partycjonowanie, w szczególności sharding, jest używane w bazach dokumentów do poprawiania wydajności i polega na dzieleniu dużych baz danych na wiele serwerów. Normalizacja, denormalizacja i procesory zapytań odgrywają znaczącą rolę w kontekście wydajności bazy dokumentów.

W rozdziale 8. zagłębimy się w problemy związane z projektowaniem konkretnych baz dokumentów.

Pytania kontrolne

1. Opisz, w jaki sposób dokumenty są analogiczne do wierszy w bazach relacyjnych.
2. Opisz, w jaki sposób kolekcje są analogiczne do tabel w bazach relacyjnych.
3. Zdefiniuj schemat.
4. Dlaczego bazy dokumentów są uznawane za nieposiadające schematu?
5. Dlaczego bazy dokumentów są uznawane za polimorficzne?
6. Czym partycjonowanie pionowe różni się od partycjonowania poziomego?
7. Czym jest klucz partycjonowania?
8. Jaki jest cel algorytmu partycjonowania w shardingu?
9. Czym jest normalizacja?
10. Po co denormalizować kolekcje w bazie dokumentów?

Odniesienia

Brewer Eric, *CAP Twelve Years Later: How the 'Rules' Have Changed*, „Computer”,

2 (luty 2012), s. 23 – 29.

Brewer Eric, *Towards Robust Distributed Systems*, „PODC”, 7, 2000.

Chodorow Kristina, *50 Tips and Tricks for MongoDB Developers*, O'Reilly Media, Inc., Sebastopol 2011.

Chodorow Kristina, *MongoDB: The Definitive Guide*, O'Reilly Media, Inc., Sebastopol 2013.

Copeland Rick, *MongoDB Applied Design Patterns*, O'Reilly Media, Inc., Sebastopol 2013.

Dokumentacja Couchbase, <http://docs.couchbase.com/>.

Han Jing i in., *Survey on NoSQL Database*, Pervasive computing and applications (ICPCA), VI Międzynarodowa Konferencja nt. IEEE, 2011.

Manual MongoDB 3.2, <http://docs.mongodb.org/manual/>.

O'Higgins Niall, *MongoDB and Python: Patterns and Processes for the Popular Document-Oriented Database*, O'Reilly Media, Inc., Sebastopol 2011.

Skorowidz

A

- ACID, 64
 - atomowość, 64
 - izolacja, 64
 - spójność, 64
 - trwałość, 64
- agregacje, 148
 - atomowe, 150
- aktualizacja dokumentów, 177
- algorytm
 - LRU, 95
 - partycjonowania, 196
- algorytmy kompresji, 129
- analiza
 - danych Big Data, 290
 - danych klienta, 294
 - grafu, 346
- antyentropia, 250, 268
- Apache HBase, 246
- aplikacja e-commerce, 28
- aplikacje
 - mobilne, 153
 - zorientowane na odczyt, 218
 - zorientowane na zapis, 219
- architektura
 - baz rodziny kolumn, 245
 - Cassandry, 247
 - HBase, 245
 - pierścienia, 123
- atomowość, Atomcity, 64
- atrybut, 208

B

- bardzo duża baza, VLDB, 233
- BASE, 64
 - miękki stan, 65
 - ostateczna spójność, 65
 - zasadnicza dostępność, 65
- baza danych, 59
 - Big Data, 288
 - Google BigTable, 234
 - NoSQL, 45
- bazy dokumentów, 54, 73, 157
 - aktualizacja dokumentów, 177
 - brak schematu, 188
 - denormalizacja, 200
 - dokument, 184
 - dokument osadzony, 187
 - dokumenty zmienne, 215
 - jawne definiowanie schematu, 172
 - kolekcja, 186
 - kryteria wyboru, 354
 - modelowanie danych, 198
 - modelowanie hierarchii, 223
 - normalizacja, 198
 - operacje, 173
 - partycjonowanie pionowe, 192
 - partycjonowanie poziome, 194
 - pobieranie dokumentów, 178
 - procesor zapytań, 200
 - projektanci, 210
 - projektowanie, 203
 - przenoszenia dużych dokumentów, 218
 - przetwarzanie zapytań, 198
 - przypadki użycia, 354
 - relacja jeden-do-wielu, 221

- bazy dokumentów
 - relacja wiele-do-wielu, 222
 - schemat polimorficzny, 190
 - usuwanie dokumentów, 176
 - wstawianie dokumentów, 175
 - zarządzanie, 164
- bazy grafowe, 54, 77, 299
 - bliskość, 322
 - cykle, 343
 - graf dwudzielny, 325
 - graf nieskierowany, 324
 - graf skierowany, 324
 - graf ważony, 326
 - izomorfizm, 321
 - krawędź, 315
 - kryteria wyboru, 356
 - mapowanie zapytań, 335
 - modelowanie
 - chorób zakaźnych, 303
 - encji abstrakcyjnych, 305
 - encji konkretnych, 305
 - lokalizacji geograficznych, 303
 - wielu relacji, 310
 - multigraf, 325
 - odpytywanie grafu, 336
 - pętla, 317
 - pośrednictwo, 322
 - projektowanie, 329
 - przecięcie grafów, 319
 - przeszukiwanie grafu, 320
 - przypadki użycia, 356
 - rząd i rozmiar, 321
 - sieci społecznościowe, 331
 - sieć przepływowa, 324
 - skalowalność, 344
 - stopień, 322
 - ścieżka, 317
 - trawersowanie grafu, 338
 - unia grafów, 318
 - unikanie połączeń, 308
 - używanie indeksów, 342
 - używanie krawędzi, 342
 - wierzchołek, 314
 - zalety, 308
- bazy klucz-wartość, 54, 68, 83, 85, 91
 - agregacje, 148
 - agregacje atomowe, 150
 - architektura, 122
 - brak schematu, 121
 - emulowanie tabel, 147
 - funkcje haszujące, 126
 - implementacja, 126
 - indeksy, 151
 - język zapytań, 144
 - klaster, 122
 - klucz, 114
 - klucz partycjonowania, 120
 - kolizja, 127
 - kompresja, 128
 - konfiguracja aplikacji mobilnej, 153
 - kryteria wyboru, 353
 - modelowanie danych, 112
 - ograniczenia, 135, 142
 - partycja, 118
 - pierścień, 124
 - projektowanie, 131
 - projektowanie kluczy, 132
 - prostota, 93
 - przestrzeń nazw, 117
 - przeszukiwanie zakresów, 144
 - przypadki użycia, 353
 - replikacja, 124
 - skalowalność, 95
 - szybkość, 94
 - wartości, 116, 137
 - wzorze projektowe, 145
 - zakres wartości, 134
- bazy relacyjne
 - a bazy dokumentów, 74
 - a bazy grafowe, 78
 - a bazy klucz-wartość, 72
 - a bazy rodziny kolumn, 76
- bazy rodziny kolumn, 54, 75, 231
 - a bazy dokumentów, 241
 - a bazy relacyjne, 242
 - antyentropia, 268
 - architektura, 245
 - cechy, 240
 - dynamiczna kontrola, 236

encje, 278
 filtr Blooma, 265
 implementacja, 261
 indeksowanie, 236, 282
 indeksy pomocnicze, 282, 286
 klaster, 262
 klucz wiersza, 258
 kolumna, 259
 kolumny bez wartości, 276
 komponenty, 258
 kontrolowanie lokalizacji danych, 237
 kryteria wyboru, 354
 log zatwierdzania, 264
 parametry konfiguracyjne, 261
 partycja, 263
 poziom spójności, 267
 projektowanie, 273
 projektowanie tabel, 275
 protokół plotki, 269
 przechowywanie danych, 277
 przekazanie ze wskazaniem, 270
 przestrzeń kluczy, 258
 przypadki użycia, 354
 replikacja, 268
 rodziny kolumn, 260
 rozbudowane struktury danych, 281
 stosowanie, 252
 struktury, 261
 wersje wartości kolumny, 280
 wiersze atomowe, 237
 bazy rozproszone, 250
 zarządzanie danymi, 54
 Big Data, 288
 ekstrakcja, 289
 ładowanie danych, 289
 narzędzia do analizy, 292
 narzędzia do monitorowania, 293
 transformacja, 289
 bitmapy, 115
 bliskość, 322
 brak schematu, 121, 188

C

cechy baz rodziny kolumn, 240
 czas
 oczekiwania, 138
 pobierania danych, 342

D

definicja wzorca projektowego, 132
 denormalizacja, 200, 204, 211, 276
 dokument, 73, 184
 HTML, 160–162
 osadzony, 187
 dokumenty
 a pary klucz-wartość, 163
 o podobnej strukturze, 164
 zmienne, 215
 dostęp
 do danych, 57
 losowy do danych, 31
 dostępność, 48, 62
 w systemach rozproszonych, 59
 duża baza, 233
 dynamiczna kontrola nad kolumnami, 236
 dystrybuowanie danych, 196
 dzielenie danych, 195

E

ekstrakcja Big Data, 289
 elastyczność, 47, 189
 elementy grafów, 314
 emulowanie tabel, 147
 encja, 114, 208, 278
 zdarzenia systemowego, 166
 entropia, 250

F

filtr Blooma, 265
 funkcje haszujące, 126

G

Google BigTable, 240
 graf, 301, 310
 dwudzielny, 325
 nieskierowany, 324
 skierowany, 324
 ważony, 326
 grafy
 analiza, 346
 bliskość, 322
 cykle, 343
 izomorfizm, 321
 krawędź, 315
 pętla, 317
 pośrednictwo, 322
 przecięcie, 319
 przeszukiwanie, 320, 339
 rozmiar, 321
 rząd, 321
 sieć przepływowa, 324
 stopień, 322
 ścieżka, 317
 trawersowanie, 337
 unia, 318
 węzeł, 314
 wierzchołek, 314

H

haszowanie, 103, 115, 136
 HDFS, Hadoop Distributed File System, 246
 hierarchia, 223
 hierarchiczny system zarządzania danymi,
 33

I

implementacja baz rodziny kolumn, 261
 indeksowanie, 282
 po nazwie kolumny, 236
 po rekordzie, 236
 po stemplu czasowym, 236
 indeksy, 42, 151
 pomocnicze, 282, 284, 286

izolacja, Isolation, 65
 izomorfizm, 321

J

jawne definiowanie schematu, 172
 język
 Cypher, 336
 definiowania danych SQL, 42
 deklaratywny, 341
 Gremlin, 340
 manipulowania danymi SQL, 43
 trawersowania, 341
 zapytań, 42
 JSON, JavaScript Object Notation, 73, 162

K

katalog danych, 41
 klaster, 99, 105, 122, 262
 Cassandry, 252
 klucz, 68, 101, 114, 184
 główny, 101
 obcy, 207
 partycjonowania, 120, 195
 wiersza, 258
 z ograniczonym czasem życia, 145, 146
 kod HTML, 161
 kolekcje, 164, 166, 186
 aktualizacja dokumentów, 177
 pobieranie dokumentów, 178
 usuwanie dokumentów, 176
 wstawianie dokumentów, 175
 kolizja, 127
 kolumny, 42, 75, 208, 259
 bez wartości, 276
 komponenty baz rodziny kolumn, 258
 kompresja, 128
 konfiguracja aplikacji mobilnej, 153
 konsorcjum CODASYL, 37
 kontrolowanie lokalizacji danych, 237
 konwencja nazewnictwa kluczy, 132
 koszt licencji, 47
 krawędź, 315

kryteria wyboru

- baz dokumentów, 354
- baz grafowych, 356
- baz klucz-wartość, 353
- baz rodziny kolumn, 354

kubelki, 71

kworum, 61

L

lekke transakcje, 244

leksykograficzne rozmieszczenie wierszy, 279

lista

- baz NoSQL, 389
- przodków, 224

log zatwierdzania, 264

lokalizacja danych, 237

losowy odczyt bloków, 31

Ł

ładowanie danych Big Data, 289

łączenie tabel, 209

M

magazyn, 40

mapowanie zapytań, 335

metoda find(), 178

miękki stan, Soft state, 65

modelowanie

- chorób zakaźnych, 303
- encji abstrakcyjnych, 305
- encji konkretnych, 305
- grafów i sieci, 302
- hierarchii, 223
- lokalizacji geograficznych, 303
- mediów społecznościowych, 307
- relacji, 221
- wielu relacji, 310

monitorowanie Big Data, 293

monotoniczny

- odczyt, 67
- zapis, 67

multigraf, 325

N

napęd taśmowy, 31

narzędzia, 288

do analizy Big Data, 292

do monitorowania Big Data, 293

narzędzie

- Gremlin, 337
- Mahout, 292
- MapReduce, 292
- R, 292
- Spark, 292

normalizacja, 198, 204

NoSQL, 45

- BASE, 64
- bazy dokumentów, 73
- bazy grafowe, 77
- bazy par klucz-wartość, 68
- bazy rodziny kolumn, 75
- dostępność, 48
- elastyczność, 47
- koszt, 47
- lista baz, 389
- skalowalność, 46
- używanie baz relacyjnych, 357

O

obiekty JSON, 162

ochrona partycji, 62

odczyt

- bloku danych, 140
- wierszy atomowych, 237

odczytanie wartości, 139

odnajdywanie wartości, 102

odpowiedzialność, 190

odpytywanie grafu, 336

ograniczenia

- baz klucz-wartość, 142
- w wyszukiwaniu wartości, 107

ograniczony czas życia, TTL, 145

operacje

- atomowe, 239
- na grafach, 318
- odczytu, 139
- zapisu, 140

optymalizacja tras transportowych, 345
 organizacja danych, 30
 ostateczna spójność, Eventually consistent,
 65

P

pamięć podręczna, 88, 89
 para klucz-wartość, 163
 partycja, 118, 263
 partycjonowanie, 64, 132, 136, 191
 pionowe, 192
 poziome, 194
 zakresowe, 136
 peer-to-peer, 247
 pętla, 317
 pierścień, 124
 klastra, 125
 pliki płaskie, 29, 32
 pobieranie
 danych, 55
 dokumentów, 74, 178
 wartości, 143
 podtypy dokumentów, 168
 polecenia
 dotyczące formatowania, 161
 dotyczące treści, 161
 polecenie MATCH, 339
 polubienie, 307
 poszukiwanie równowagi, 204
 pośrednictwo, 322
 poziom spójności, 267
 problemy z zapisem, 104
 procesor zapytań, 200
 programy zarządzające
 magazynem, 39
 pamięcią, 41
 projektowanie
 baz dokumentów, 203
 baz grafowych, 329
 baz klucz-wartość, 131
 baz relacyjnych, 28
 baz rodziny kolumn, 273
 kluczy, 132
 kolekcji, 166

sterowane przez zapytania, 334
 struktury wartości, 137
 tabel, 275

protokół
 Cassandra, 249
 plotki, 248, 269
 przechowywanie danych, 55, 106, 277
 przecięcie grafów, 319
 przekazanie ze wskazaniem, 251, 270
 przenoszenie dużych dokumentów, 218
 przestrzeń
 kluczy, 258
 nazw, 71, 117
 przeszukiwanie
 grafu, 320, 337
 w głąb, 340
 wszerz, 341
 zakresów, 144
 przetwarzanie zapytań, 198
 przypadki użycia
 baz dokumentów, 354
 baz grafowych, 356
 baz klucz-wartość, 353
 baz rodziny kolumn, 354
 punkty zapalne, 279

R

RDBMS, 39, 40
 referencje
 do potomka, 224
 do rodzica, 224
 relacja, 77, 332
 jeden-do-wielu, 206, 221
 wiele-do-wielu, 206, 222
 relacje zawierania, 306
 relacyjne systemy zarządzania danymi, 39
 replikacja, 124, 268
 bez serwera głównego, 97
 master-slave, 96
 rodzaje ostatecznej spójności, 66
 rodziny kolumn, 75, 260
 rozdzielanie kolumn, 194
 rozgłaszanie, 248
 różnorodność węzłów, 245

S

schemat, 41, 172
 polimorficzny, 190
 sharding, 194
 sieciowy system zarządzania danymi, 35
 sieć
 peer-to-peer, 248
 przepływowa, 324
 skalowalność, 46, 95
 bazy grafowej, 344
 z replikacją master-slave, 96
 spójność, Consistency, 62, 65
 danych, 56
 luźna, 66
 sesji, 67
 transakcji bazodanowych, 59
 typu czytaj swój zapis, 66
 w systemach rozproszonych, 59
 SQL, 42
 stopień, 322
 struktura
 dokumentów HTML, 161
 obiektów JSON, 162
 pierścienia, 99
 struktury danych, 41
 studium przypadku, 50
 analiza danych klienta, 294
 konfiguracja aplikacji mobilnej, 153
 manifesty użytkowników, 226
 optymalizacja tras transportowych, 345
 system
 plików Hadoop, 246
 zarządzania danymi, 59
 systemy hierarchiczne, 33
 ograniczenia, 35
 organizacja, 33
 systemy oparte na plikach płaskich, 29
 ograniczenia, 32
 organizacja danych, 30
 systemy relacyjne, 39
 ACID, 64
 ograniczenia, 44
 organizacja, 39
 organizacja aplikacji, 44

systemy rozproszone, 53
 dostępność, 59
 spójność, 59
 systemy sieciowe, 35
 ograniczenia, 37
 organizacja, 35

Ś

ścieżka, 317

T

tabele, 42, 114, 208
 tablice, 86
 asocjacyjne, 87
 taśmy magnetyczne, 30
 teoria CAP, 62
 termodynamika, 250
 transakcje
 bazodanowe, 59
 na wielu wierszach, 243
 transformacja Big Data, 289
 trawersowanie, 337
 trwałość, 62
 trwałość, Durability, 65
 TTL, Time to Live, 145
 tworzenie indeksów pomocniczych, 286
 typy
 baz NoSQL, 68
 dokumentów, 168
 encji, 166
 grafów, 323
 partycji, 191
 wartości, 185

U

uczenie maszynowe, 292
 unia grafów, 318
 unikanie
 podzapytań, 244
 transakcji na wielu wierszach, 243
 złążeń, 308
 uporządkowana lista elementów, 86

- upraszczanie modelowania, 310
- usuwanie dokumentów, 176
- utrzymanie spójności danych, 56
- UUID, Universally Unique Identifier, 175
- używanie
 - baz NoSQL, 357
 - indeksów, 342
 - kluczy, 102
 - krawędzi, 342

V

VLDB, very large database, 233

W

- waga krawędzi, 315
- wartości, 71, 106, 137, 184
- wczesne
 - systemy baz danych, 37
 - systemy zarządzania bazami, 29
- wersje wartości kolumny, 280
- węzły, 77, 314
- widoki, 42
- wiersze
 - atomowe, 237
 - posortowane, 238
- wierzchołek, 314
- więzy, 42
- właściwości
 - grafów, 320
 - krawędzi, 320

- wstawianie dokumentów, 175
- wybór bazy danych, 349–352
- wydajność dostępu do danych, 89
- wykonywanie złączeń, 208, 210
- wzorzec projektowy
 - agregacja, 148
 - agregacja atomowa, 150
 - emulowanie tabeli, 147
 - indeksy, 151
 - klucz z ograniczonym czasem życia, 145

X

XML, Extensible Markup Language, 73

Z

- zakres wartości, 134
- zapis wierszy atomowych, 237
- zapytania
 - deklaratywne, 336
 - przez trawersowanie grafu, 337
- zarządzanie wieloma dokumentami, 164
- zasadnicza dostępność, 65
- zbiory uporządkowane, 115
- złączenia, 200, 206, 209, 214
- zrównoważenie czasów
 - reakcji, 60
 - spójności, 60
 - trwałości, 60

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>

GRUPA WYDAWNICZA



Helion SA

Systemy do zarządzania danymi muszą dziś spełniać o wiele wyższe wymagania niż kiedyś. W wielu przypadkach nierelacyjne bazy danych, zwane NoSQL, są lepszym rozwiązaniem niż dominujące do niedawna bazy relacyjne. Projektant systemu ma więc do dyspozycji dobrze znane bazy relacyjne oraz systemy NoSQL, takie jak bazy klucz – wartość, bazy dokumentów, rodziny kolumn i bazy grafowe. Do rozstrzygnięcia pozostaje problem: którą technologię zarządzania danymi wybrać w danym przypadku.

Niniejsza książka jest przystępnym, pragmatycznym przewodnikiem po nierelacyjnych systemach bazodanowych. Pokazano w niej, czym różnią się NoSQL od baz relacyjnych. Szczególny nacisk położono na wyjaśnienie tych cech i funkcji, które powinny być uwzględniane podczas projektowania aplikacji i wybierania technologii bazodanowych. Przedstawiono wewnętrzne mechanizmy baz NoSQL i wyjaśniono, w jaki sposób zbudować za ich pomocą skalowalne, niezawodne aplikacje. Nie zabrakło przydatnych wskazówek, zasad projektowych i najlepszych praktyk.

W tej książce znajdziesz:

- ▶ podstawy relacyjnych baz danych w odniesieniu do systemów NoSQL
- ▶ bazy klucz – wartość — ich charakterystykę, zasady projektowania i słabe strony
- ▶ bazy dokumentów — koncepcje baz bez schematu, podstawowe operacje, najpowszechniejsze wzorce projektowe
- ▶ bazy rodzin kolumn — aplikacje Big Data, architekturę baz, zasady ich projektowania i wykorzystywania
- ▶ zasady dobierania technologii bazodanowej do konkretnych zastosowań

Dan Sullivan jest naukowcem i architektem danych. Od dwudziestu lat zajmuje się m.in. analizą biznesową, uczeniem maszynowym i data miningiem. Sullivan jest uznanym ekspertem w dziedzinie baz danych, zarówno relacyjnych, jak i NoSQL.

 Addison-Wesley

Helion	
46418	numer katalogowy
księgarnia internetowa	
http://helion.pl	
zamówienia telefoniczne	
	0 801 339900
	0 601 339900
Informatyka w najlepszym wydaniu	

Sprawdź najnowsze promocje:
• <http://helion.pl/promocje>
Książki najchętniej czytane:
• <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
• <http://helion.pl/nawosci>

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

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-2488-6



9 788328 324886

cena: 69,00 zł