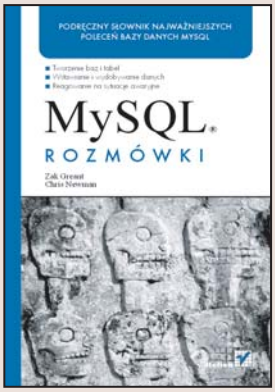
MySQL. Rozmówki



Autorzy: Zak Greant, Chris Newman
Tłumaczenie: Przemysław Szeremiota
ISBN: 83-246-0550-9
Tytuł oryginału: MySQL Phrasebook
Format: B6, stron: 216

Podręczny słownik najważniejszych poleceń bazy danych MySQL

- Tworzenie baz i tabel
- Wstawianie i wydobywanie danych
- Reagowanie na sytuacje awaryjne

MySQL to jeden z najbardziej popularnych systemów zarządzania bazami danych. Jego ogromne możliwości idą w parze z łatwością obsługi, ale nie jest to jego jedyna zaleta. Ważne jest również to, że zarówno sama aplikacja, jak i jej kod źródłowy dostępne są nieodpłatnie. MySQL jest wykorzystywany nie tylko w roli zaplecza bazodanowego dla witryn WWW, ale również w rozbudowanych systemach informatycznych wymagających stabilnego źródła danych. Zestaw poleceń MySQL to nie tylko instrukcje związane z manipulowaniem danymi, ale także wiele innych wskazówek przeznaczonych dla administratorów bazy.

Książka „MySQL. Rozmówki" to zbiór najbardziej przydatnych i najczęściej wykorzystywanych poleceń MySQL. Zawiera prezentacje sposobów realizacji najbardziej typowych zadań, przed jakimi stają użytkownicy baz danych MySQL, uzupełnione licznymi wskazówkami i przykładami. Czytając ją, poznasz sposoby zakładania tabel i wypełniania ich danymi, pobierania danych z bazy oraz przetwarzania ich za pomocą odpowiednio sformułowanych zapytań. Nauczysz się administrować kontami użytkowników, archiwizować dane i odtwarzać je w przypadku awarii bazy.

- Tworzenie nowej bazy oraz tabel
- Zakładanie indeksów
- Wprowadzanie danych do tabel
- Proste zapytania
- Przetwarzanie danych w zapytaniach
- Tworzenie kont użytkowników i nadawanie uprawnień
- Korzystanie z interfejsów programistycznych MySQL
- Archiwizowanie danych

Jeśli często korzystasz z MySQL, a dziesiątki opasłych tomów nie mieszczą się już na Twoim biurku, sięgnij po tę książkę

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

2

Tworzenie baz danych, tabel i indeksów i zarządzanie nimi

Polecenia prezentowane w tym rozdziale wykorzystuje się do tworzenia i zarządzania strukturami MySQL, które (z logicznego punktu widzenia) umożliwiają podgląd i składowanie danych i organizują dane przechowywane na serwerze MySQL.

Nie wszyscy Czytelnicy są zaznajomieni z podstawami modelowania danych przechowywanych na serwerach MySQL, więc na początek proponujemy pięciominutową powtórkę z podstaw dotyczących baz danych, tabel, kolumn i indeksów.

Bazy danych, tabele, kolumny i indeksy

Podstawowy model składowania danych, przyjęty w MySQL, jest całkiem prosty, ale pewnie tylko dla tych, którzy już go znają. Przedstawia się tak:

- Serwer MySQL zawiera pewną liczbę baz danych (przynajmniej jedną).
- Każda baza danych posiada nazwę i zawiera pewną liczbę tabel (przynajmniej jedną).
- Każda tabela posiada nazwę, definicję i pewną liczbę wierszy (tabela może być pusta).
- Definicja każdej tabeli określa jedną lub więcej kolumn tabeli i zero lub więcej indeksów.
- Każda kolumna posiada nazwę i typ.
- Typy kolumn wybiera się z zestawu predefiniowanych typów MySQL. Każdy z nich posiada nazwę i definiuje zestaw wartości, które można umieścić w kolumnie danego typu.
- Każdy wiersz tabeli zawiera po jednej wartości dla każdej kolumny wymienionej w definicji tabeli.
- Indeksy redukują narzuty czasowe typowe dla serwerów baz danych, a związane z wyszukiwaniem konkretnych wartości w kolumnie (zbiorze kolumn) tabeli.

WSKAZÓWKA

Omawiane wyżej pojęcia ilustruje rysunek „Terminologia tabel MySQL" w rozdziale 1. („Mapy MySQL").

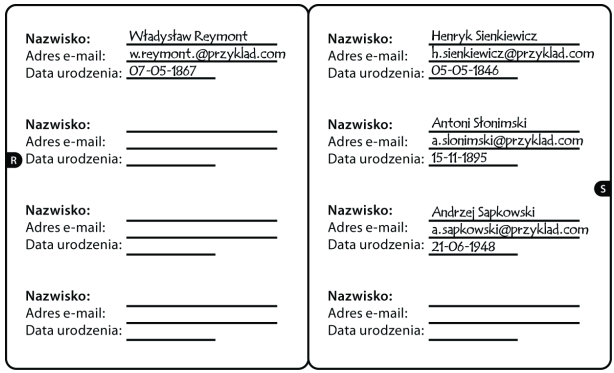
Bazy danych łatwo zrozumieć; są najzwyklejszymi pojemnikami na dane. Nieco gorzej z tabelami. Do ich ogarnięcia przyda się prosta metafora.

Tabele metaforycznie

Dobrym, choć uproszczonym modelem bazy danych jest znany każdemu mały, czarny notesik adresowy. W owej pierwotnej, prostej wersji książki adresowej informacje o znajomych i kolegach zapisywało się na kolejnych kartkach. Wpisy dotyczące poszczególnych osób były rozmieszczane w specjalnych szablonach, z polami do wypełnienia, opatrzonymi etykietami: Imię i nazwisko, Adres czy Data urodzenia.

Do tego bywało, że strony były oznaczane zakładkami indeksowymi, pomocnymi w odszukiwaniu konkretnych osób. Bez takich zakładek wyszukiwanie wymagało przeglądania kolejnych kartek aż do momentu natrafienia na szukaną osobę.

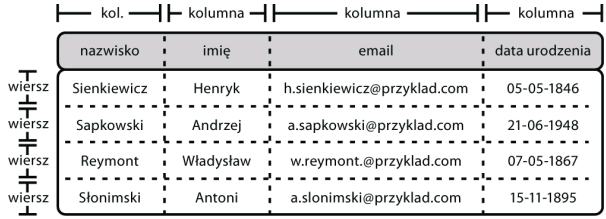
Książeczka, o której mówimy, mogłaby wyglądać jak na rysunku 2.1.



Rysunek 2.1. Osobista książka adresowa

Taki notesik można traktować jako tabelę złożoną z wierszy i kolumn. Każdy wiersz odnosi się do pojedynczej osoby, podczas gdy kolumny grupują atrybuty poszczególnych osób (nazwiska, adresy poczty elektronicznej, daty urodzenia).

Zawartość naszego notesika w formie tabelarycznej przedstawiałaby się tak, jak na rysunku 2.2.

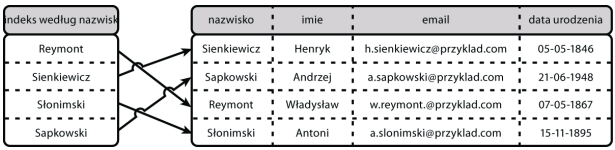


Rysunek 2.2. Książeczka adresowa jako tabela bazy danych

W papierowej książeczce adresowej wpisy poszczególnych osób umieszczone były na stronach z odpowiednią zakładką indeksową, pasującą do pierwszej litery nazwiska. Na przykład Antoni Słonimski został zapisany na stronie z zakładką indeksową S.

Z kolei w tabeli wiersze są rozmieszczone w dowolnej kolejności. Aby umożliwić wyszukanie odpowiedniego wiersza bez konieczności przeglądania całości tabeli, kolumny tabel uzupełnia się o indeksy. Indeksy takie stanowią jakby listy uporządkowanych wartości kolumn, reprezentujących właściwe pozycje tych wartości w kolumnie.

Sposób powiązania indeksu z właściwymi pozycjami wierszy w tabeli ilustruje rysunek 2.3.



Rysunek 2.3. Pozycje wierszy w tabeli

Tworzenie baz danych i zarządzanie nimi

Bazy danych zasadniczo tworzy się zupełnie prosto, nieskomplikowane jest też zarządzanie nimi — bazy są po prostu pojemnikami na dane, z zaledwie paroma dodatkowymi własnościami. Zajmiemy się teraz tym, co trzeba

wiedzieć, aby skutecznie tworzyć i usuwać bazy i zarządzać już istniejącymi bazami danych.

Wypisanie listy baz danych

```
SHOW DATABASES;
```

Polecenie `SHOW DATABASES` pobiera od serwera MySQL listę baz danych pozostających pod opieką tego serwera.

W świeżej instalacji MySQL polecenie `SHOW DATABASES` powinno zwrócić wypis podobny do poniższego:

```
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)
```

Filtrowanie listy baz danych

```
SHOW DATABASES LIKE 'my%';
```

Jeśli na wypisie mają znaleźć się jedynie takie bazy danych, których nazwy pasują do zadanego słowa lub jego części, należy zastosować operator `LIKE`.

Słowo występujące za operatorem `LIKE` może być zwyczajnym słowem, uzupełnionym o znaki procenta (`%`) i podkreślenia (`_`), tak zwane symbole wieloznaczne.

W kontekście operatora `LIKE` znak procenta (`%`) może reprezentować dowolną sekwencję (również pustą) znaków, podczas gdy znak podkreślenia (_) reprezentuje dowolny, ale pojedynczy znak.

W tym przykładzie polecenia `SHOW DATABASES` na wypisie baz danych znajdą się tylko te bazy danych, których nazwy zaczynają się od `my` (a także ewentualna baza o nazwie `my`).

WSKAZÓWKA

Operator `LIKE` może być wykorzystywany w wielu innych poleceniach (zapytaniach) SQL; poza poleceniem `SHOW` występuje typowo w klauzulach `WHERE` zapytań `SELECT` i `DELETE`.

W przypadku świeżo zainstalowanej kopii oprogramowania MySQL uruchomienie polecenia przykładowego z tak określonym filtrem baz danych powinno dać efekt taki, jak poniżej:

```
+----------+
| Database |
+----------+
| mysql    |
+----------+
1 rows in set (0.00 sec)
```

UWAGA

Zainteresowanych dodatkowymi informacjami o operatorze `LIKE` odsyłamy pod *adres http://mysql.com/LIKE.*

Ustawianie domyślnej bazy danych

```
USE test;
```

Pojedynczy serwer MySQL może pełnić pieczę nad wieloma bazami danych. Zapytanie operujące na bazie danych albo tabeli musi w jakiś sposób zostać skojarzone z odpowiednią bazą danych. Docelową bazę danych można podać jawnie w tekście zapytania. Na przykład poniższe zapytanie odwołuje się do tabeli `książka` w bazie danych `biblioteka`:

```
SELECT tytul FROM biblioteka.książka;
```

Alternatywnie można ustalić domyślną bazę danych, wybieraną automatycznie dla wszystkich kolejnych zapytań. Służy do tego polecenie `USE`:

```
USE biblioteka;
```

W dalszej perspektywie ustawienie domyślnej bazy danych zaoszczędzi nam mnóstwo pisania, bo nie trzeba będzie już podawać w zapytaniach przy każdej tabeli nazwy bazy danych. Zapytania będą dotyczyły tej bazy danych, która poleceniem `USE` została ustawiona jako domyślna.

Porównajmy poniższe zapytania. Zapytania pierwsze i trzecie zwracają te same wyniki, ale w pierwszym docelowa baza danych jest określana jawnie, a w trzecim polegamy na ustawieniu domyślnej bazy danych, wykonanym w zapytaniu numer dwa:

```
SELECT tytul FROM biblioteka.książka
USE biblioteka;
SELECT tytul FROM książka;
```

WSKAZÓWKA

Mimo ustalenia domyślnej bazy danych poleceniem `USE`, wciąż można jawnie podawać docelowe bazy danych w zapytaniach (jak w pierwszym z trzech powyższych zapytań), co pozwala na wybiórcze przełączanie baz danych na potrzeby pojedynczych zapytań.

Tworzenie bazy danych

```
CREATE DATABASE biblioteka;
```

Powyższe polecenie tworzy nową bazę danych o nazwie `biblioteka`. Samo w sobie nie jest wielce użyteczne, ale w podrozdziale „Tworzenie tabel i zarządzanie nimi" wypełnimy nową, pustą bazę danych treścią.

Usuwanie bazy danych

```
DROP DATABASE biblioteka;
```

Polecenie `DROP DATABASE` usuwa wskazaną bazę danych wraz ze wszystkimi jej tabelami.

OSTRZEŻENIE

Polecenie `DROP DATABASE` należy stosować z zachowaniem najwyższej ostrożności — nie da się łatwo wycofać niszczycielskiego efektu tego polecenia po jego wykonaniu.

Zmiana nazwy bazy danych

Operacja zmiany nazwy bazy danych jest rzadkością. Częściowo dlatego, że jest to operacja nieco karkołomna. Do tego w MySQL nie da się jej załatwić pojedynczym poleceniem SQL.

W archaicznych wersjach MySQL zmiana nazwy bazy danych sprowadzała się do:

1. Zatrzymania serwera MySQL.
2. Zmiany nazwy katalogu reprezentującego bazę danych.
3. Ponownego uruchomienia serwera MySQL.

We współczesnych wersjach MySQL taka procedura może spowodować, że serwer przestanie rozpoznawać pewne rodzaje tabel przechowywanych w bazie danych. Aby bezpiecznie zmienić nazwę bazy danych, należałoby wykonać poniższą procedurę:

1. Upewnić się, że żadna z tabel bazy danych nie uczestniczy w odwołaniu. Najprościej zapewnić to odebraniem uprawnień do używania bazy danych za pomocą narzędzia MySQL Administrator. Szczegóły zostaną podane w podrozdziałach „Regulowanie uprawnień użytkowników" i „Blokowanie dostępu" w rozdziale 6. („Konta użytkowników i bezpieczeństwo").
2. Utworzyć nową bazę danych. Nadać jej pożądaną nazwę.

3. Dla każdej tabeli w pierwotnej bazie danych wykonać polecenie `SHOW TABLES` i `RENAME TABLE` w celu przeniesienia tabel do nowej bazy danych.
4. Nadać użytkownikom pierwotnej bazy danych uprawnienia dostępu do nowej bazy danych.
5. Sprawdzić, czy wszystko działa.
6. Usunąć pierwotną bazę danych.

Kod realizujący migrację tabel `książka`, `czytelnik` i `wypożyczenie` z bazy danych `książki` do bazy danych `biblioteka` mógłby wyglądać tak:

```
-- Czasowe zablokowanie dostępu
CREATE DATABASE biblioteka;
RENAME TABLE książki.książka TO biblioteka.książka;
RENAME TABLE książki.czytelnik TO biblioteka.czytelnik;
RENAME TABLE książki.wypożyczenie TO
biblioteka.wypożyczenie;
-- Przeniesienie uprawnień użytkowników
-- Aktywacja uprawnień
```

Tworzenie tabel i zarządzanie nimi

Tabele to struktury nieco bardziej skomplikowane od baz danych, nic dziwnego więc, że ich tworzenie jest trudniejsze, tak samo jak zarządzanie tabelami.

W najbliższych punktach skupimy się na absolutnych podstawach, niezbędnych do przetrwania w obcym środowisku. Przy tworzeniu tabel mamy bowiem do dyspozycji

dosłownie setki opcji, a przy realizacji bardziej złożonych zadań — na przykład przy definiowaniu domyślnego porządku sortowania dla kolumn czy tworzeniu tabel mających przechowywać terabajty danych — trzeba korzystać z dodatkowych zasobów.

Wypisywanie wszystkich albo wybranych tabel bazy danych

```
SHOW TABLES;
SHOW TABLES IN nazwa_bazy_danych;
SHOW_TABLES LIKE 'słowo%'
SHOW TABLES IN nazwa_bazy_danych LIKE 'słowo%'
```

Pierwsza wersja polecenia `SHOW TABLES` wypisuje wszystkie tabele zdefiniowane w domyślnej bazie danych; postać druga wypisuje komplet tabel z bazy danych wskazanej przez nazwę. Trzecia i czwarta wersja tego samego polecenia wykorzystują operator `LIKE` do filtrowania nazw tabel.

WSKAZÓWKA

Podobnie jak w przypadku polecenia `SHOW DATABASES`, operator `LIKE` służy do ograniczania wykazu tabel bazy danych wypisywanych poleceniem `SHOW TABLES`. Więcej o operatorze LIKE dowiesz się z podrozdziału „Filtrowanie listy baz danych".

Tworzenie tabel

```
CREATE TABLE książka (
  id    SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
  tytul VARCHAR(255) NOT NULL,
  autor VARCHAR(255) NOT NULL,
  stan  ENUM('podniszczona', 'dobra', 'idealna',
 'nowa') NOT NULL,
        PRIMARY KEY (id_książki),
};
```

Tworzenie tabel jest tak skomplikowane z dwóch powodów: po pierwsze, z racji złożoności składni zapytań tworzących tabele, obejmującej wiele elementów. Druga przyczyna to konieczność wyboru układu tabeli, co jest zadaniem jeszcze trudniejszym.

Zamiast zajmować się szczegółowo oboma aspektami, posłużymy się przykładem, przypominającym to, co trzeba wiedzieć (albo się dowiedzieć) o tworzeniu tabel. Pełnej składni podawanych poleceń należy szukać w dokumentacji MySQL, publikowanej w sieci WWW.

Zmiana nazw tabel

```
RENAME TABLE pierwotna_nazwa TO nowa_nazwa;
```

Składnia zmiany nazwy tabeli jest wyjątkowo nieskomplikowana.

WSKAZÓWKA

Przy zmienianiu nazw tabel należy pamiętać o obecności użytkowników i aplikacji, którzy będą oczekiwać obecności tabel o znanych im nazwach i odwoływać się do nich przez ich pierwotne nazwy. Możliwe w takim układzie problemy zostały rozpoznane w podrozdziale „Wyszukiwanie użytkowników bazy danych bądź tabeli" rozdziału 6.

Usuwanie tabel

```
DROP TABLE nazwa_tabeli;
```

Usunięcie tabeli oznacza, że (poza ewentualnymi kopiami zapasowymi) tabela znika na dobre. Dlatego polecenie to należy stosować ostrożnie.

WSKAZÓWKA

Przy braku ograniczeń odnośnie do przestrzeni dyskowej, zamiast usuwać tabele, lepiej zmieniać im nazwy. I dopiero po pewnym okresie próbnym (na przykład po tygodniu), kiedy wiadomo na pewno, że nieobecność tabeli nie wywołała żadnych problemów, można ją z czystym sumieniem usunąć.

Kopiowanie tabel

```
CREATE TABLE nowa_tabela LIKE pierwotna_tabela;
INSERT nowa_tabela SELECT * FROM pierwotna_tabela;
```

Kopie istniejących tabel przydają się do testowania zapytań potencjalnie destrukcyjnych, instalowania nowego egzemplarza aplikacji bazodanowej i tym podobnych operacji.

Kopiowanie może się odbywać na wiele sposobów. Dwa zapytania przedstawione powyżej (wykonane łącznie) tworzą kompletną kopię struktury i danych tabeli wraz z jej indeksami, opcjami tabeli i wszelkimi jej atrybutami.

WSKAZÓWKA

Tabele można też kopiować inaczej; alternatywne sposoby bywają groźne, szybsze, nadają się dla szczególnych rodzajów tabel, czy też nie wykonują tak dokładnej kopii.

Z łatwością można skopiować tabelę jednej bazy danych do innej bazy danych, wystarczy przed nazwami tabel podać nazwy baz danych. Oto przykład:

```
CREATE TABLE bd1.tabela LIKE bd2.tabela;
INSERT bd1.tabela SELECT * FROM bd2.tabela;
```

Operator LIKE wykorzystany w powyższym zapytaniu działa inaczej niż ten sam operator LIKE zastosowany w poleceniach SHOW i klauzulach WHERE zapytań.

Tworzenie kolumn tabel i zarządzanie nimi

Kolumny są zazwyczaj tworzone przy okazji tworzenia tabeli. Jeśli potem pojawi się potrzeba zmiany układu czy atrybutów kolumn, można to zrobić — operacja dodania kolumny albo zmiany definicji istniejącej kolumny tabeli nie jest wcale rzadkością.

Nawet drobne modyfikacje istniejących kolumn wymuszają przebudowanie tabeli. Dlatego przed taką zmianą

należy zawsze wykonać kopię zapasową. W przypadku tabel przechowujących gigabajty danych przebudowa może być długotrwała; warto więc rozważyć wprowadzanie takich zmian albo w czasie planowanego przestoju konserwacyjnego, albo po prostu na kopii tabeli.

Dodawanie kolumn

```
ALTER TABLE nazwa_tabeli
      ADD COLUMN [definicja_kolumny];
ALTER TABLE książka
      ADD COLUMN ISBN VARCHAR(10) NOT NULL;
```

Jak widać, kolumny można łatwo dodawać do istniejących już tabel.

Dodanie kolumny polega na przygotowaniu definicji kolumny tak, jak dla zapytania `CREATE TABLE`, i użyciu definicji w zapytaniu `ALTER TABLE` *`nazwa_tabeli`* `ADD COLUMN`.

Pierwsze zapytanie pokazuje postać ogólną zapytania dodającego kolumnę do tabeli; zapytanie drugie to konkretne zapytanie operujące na tabeli przykładowej `książka`, uzupełniające ją o kolumnę `ISBN`.

Zmiana definicji (i nazw) kolumn

```
ALTER TABLE nazwa_tabeli
      CHANGE COLUMN nazwa_kolumny
      [definicja_kolumny];
ALTER TABLE książka
      CHANGE COLUMN ISBN isbn VARCHAR(10) NOT NULL;
```

Zmiana definicji kolumny (z ewentualnym nadaniem jej nowej nazwy) przebiega podobnie do operacji dodawania nowej kolumny do tabeli. Widać jednak drobną różnicę w składni zapytania: zamiast `ALTER TABLE … ADD COLUMN` stosuje się `ALTER TABLE … CHANGE COLUMN`. Definicja kolumny musi przed określeniem typu i atrybutów kolumny podawać jej nazwę, więc przy zmianie samej definicji kolumny jej nazwa musi pojawić się w zapytaniu dwukrotnie.

Pierwsze z powyższych zapytań ilustruje ogólną składnię zapytań zmieniających kolumny; drugie pokazuje sposób zmiany nazwy i szerokości pola kolumny `ISBN` w tabeli `książka`.

Usuwanie kolumn

```
ALTER TABLE nazwa_tabeli DROP COLUMN nazwa_kolumny;
```

Usuwanie kolumny z tabeli bazy danych to prosta operacja. Jednak jak zwykle w przypadku poleceń usuwających coś z bazy danych, trzeba pamiętać o zagrożeniach: skoro nie można wycofać operacji usunięcia, trzeba pamiętać o użytkownikach i aplikacjach odwołujących się do usuwanego elementu.

WSKAZÓWKA

Usunięcie kolumny powoduje automatyczne usunięcie wszelkich indeksów obejmujących tę kolumnę.

Identyfikatory

Identyfikatory MySQL, w tym nazwy baz danych i tabel, są zazwyczaj rozpatrywane z uwzględnieniem wielkości poszczególnych liter. Dlatego dla uproszczenia najlepiej definiować identyfikatory z użyciem samych małych liter. Aby wymusić na większości platform spójne zachowanie MySQL pod tym względem, należy ustawić zmienną konfiguracyjną `lower_case_table_names` na `1`. W ten sposób wszystkie identyfikatory, w których ważna jest wielkość liter, będą traktowane jak pisane małymi literami. Więcej informacji o identyfikatorach można znaleźć w podręczniku MySQL pod adresem *http://mysql.com/identifier case sensitivity*.

Tworzenie indeksów i zarządzanie nimi

Indeksy są często najważniejszym czynnikiem określającym szybkość wybierania danych z dużych tabel.

W tym podrozdziale zajmiemy się tworzeniem indeksów i podstawowymi aspektami zarządzania indeksami. Nie będziemy zajmować się zagadnieniami takimi jak optymalny wybór kolumny dla indeksu czy sposobami używania indeksów. Zainteresowanych szczegółami odsyłamy do rozdziału 4., „Wydobywanie danych. Proste zapytania".

Dodawanie indeksu do tabeli

```
CREATE INDEX nazwa_indeksu
       ON nazwa_tabeli (nazwa_kolumny, …);
CREATE INDEX autor ON książka (autor);
CREATE INDEX autor ON książka (autor(16));
CREATE INDEX tytul_autor ON książka (tytul, autor);
```

Aby dodać indeks do zestawu kolumn istniejącej tabeli, należy skorzystać z polecenia `CREATE INDEX` (ewentualnie `ALTER TABLE`, ale polecenie `CREATE INDEX` łatwiej zapamiętać).

Polecenie wymaga określenia nazwy nowego indeksu oraz nazwy tabeli, której indeks ma dotyczyć, wraz z nazwami kolumn, na których indeks ma bazować.

Pierwsze zapytanie ilustruje ogólną składnię polecenia tworzącego indeks, w którym należałoby wstawić własne nazwy i wartości. Pozostałe zapytania pokazują konkretne przykłady tworzenia rozmaitych indeksów dla tabeli `książka`.

Zmiana nazwy indeksu

Zmiana nazwy indeksu to dość rzadka operacja. Indeks jest najczęściej zwyczajnie usuwany, ewentualnie zmienia się jego definicję. Kiedy trzeba zmienić nazwę indeksu, najlepiej po prostu usunąć indeks i utworzyć nowy, o pożądanej nazwie.

Usuwanie indeksu

```
DROP INDEX nazwa_indeksu ON nazwa_tabeli;
DROP INDEX autor ON książka;
```

Składnia usuwania indeksu jest bardzo prosta — wystarczy podać nazwę tabeli i nazwę indeksu, który ma z niej zostać usunięty.

Pierwsze zapytanie prezentuje ogólną postać zapytania usuwającego indeks; drugie to przykład usunięcia indeksu `autor` z tabeli `książka`.

Zobacz również

Zagadnienia opisywane w niniejszym rozdziale są szerzej omawiane w dostępnej on-line dokumentacji MySQL, pod następującymi adresami:

- Zapytania definicji danych — *http://dev.mysql.com/doc/refman/5.0/en/data-definition.html*
- Wykorzystywanie indeksów w MySQL — *http://dev.mysql.com/doc/refman/5.0/en/mysql-indexes.html*