MySQL. Darmowa baza danych. Ćwiczenia praktyczne

Autor: Marcin Lis
ISBN: 83-246-0600-9
Format: A5, stron: 184

Wykorzystaj w swoich projektach bazę MySQL

MySQL to system zarządzania bazami danych stworzony i rozwijany przez szwedzką firmę MySQL AB. Jest bardzo szybki i wydajny, dostępny dla różnych systemów operacyjnych i, co najważniejsze, dystrybuowany na licencji GPL, więc korzystanie z niego nie wymaga wnoszenia żadnych opłat. MySQL wykorzystywany jest coraz powszechniej – nie tylko jako zaplecze bazodanowe witryn WWW, ale także jako magazyn danych dla złożonych systemów informatycznych operujących na setkach tysięcy rekordów.

„MySQL. Darmowa baza danych. Ćwiczenia praktyczne" to zbiór krótkich ćwiczeń, dzięki którym poznasz podstawy pracy z MySQL. Dowiesz się, jak zainstalować i skonfigurować serwer MySQL w systemach Windows i Linux, nauczysz się tworzyć bazy danych i zakładać konta użytkowników. Poznasz typy danych stosowane w MySQL, zaprojektujesz i utworzysz tabele w bazie danych oraz wykorzystasz język SQL do wstawiania, wybierania i modyfikowania danych.

- Instalacja MySQL
- Uruchamianie i zatrzymywanie serwera bazy danych
- Tworzenie kont użytkowników i nadawanie uprawnień
- Uruchamianie poleceń zapisanych w plikach zewnętrznych
- Tworzenie struktury bazy danych
- Wprowadzanie danych do tabel
- Pobieranie danych
- Złożone zapytania

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Tworzenie struktury bazy danych

Ogólna postać instrukcji CREATE

Dane w bazie przechowywane są w tabelach. Pojęcie tabeli poznaliśmy w rozdziale 3., „Koncepcja relacyjnych baz danych". Czas więc dowiedzieć się, w jaki sposób można tworzyć tabele. Służy do tego instrukcja `CREATE TABLE` o schematycznej postaci:

```
CREATE TABLE nazwa_tabeli
(
  nazwa_kolumny_1 typ_kolumny_1 [atrybuty],
  nazwa_kolumny_2 typ_kolumny_2 [atrybuty],
  ...
  nazwa_kolumny_n typ_kolumny_n [atrybuty],
)
```

Nazwa tabeli może zawierać dowolne znaki, jakie może zawierać nazwa pliku w systemie operacyjnym, na którym działa serwer MySQL, z wyjątkiem `/`, `\` i `.`. Maksymalna długość nazwy to 64 bajty. Począwszy od wersji 3.23.6, nazwa tabeli może być nazwą słowa zastrzeżonego dla konstrukcji języka (np. `SELECT`, `CREATE`), w takim wypadku musi być jednak zawsze ujęta w znaki `‘`, np. `‘SELECT‘`. Nie

może również zawierać znaków o kodach 0 i 255 w standardzie ASCII, a na jej końcu nie powinny się znajdować tzw. białe znaki (czyli spacje, tabulatory itp.).

Nazwa kolumny może zawierać dowolne znaki, jednak jej długość jest również ograniczona do 64 bajtów i także nie powinna być zakończona białym znakiem. Typ kolumny określa rodzaj danych, które dana kolumna będzie mogła przechowywać, np. łańcuch znaków, liczby, itp. Występujące w MySQL typy danych zostaną omówione w następnej sekcji.

W nazwach tabel (oraz baz danych) mogą występować zarówno małe, jak i duże litery, jednak to, czy będą rozróżniane, zależy od systemu plików systemu operacyjnego, na którym został zainstalowany MySQL. I tak w większości odmian Uniksa wielkie i małe litery są rozróżniane, natomiast w systemach Windows — nie. W systemach Mac OS rozróżnianie wielkości liter zależy od tego, czy wykorzystywany jest system plików HFS (nie są rozróżniane), czy USF (są rozróżniane). W związku z tym, o ile to możliwe, najwygodniej przyjąć po prostu zasadę, że nazwy baz i tabel zawsze są pisane małymi literami (najlepiej alfabetu łacińskiego).

Nazwy kolumn również mogą zawierać małe i duże litery, jednak w tym wypadku nie są one rozróżniane, niezależnie od wersji systemu operacyjnego czy systemu plików. Począwszy od wersji 4.1, wszystkie identyfikatory i nazwy odnoszące się do definicji zawartości tabel są zapisywane w standardzie Unicode. Należy również pamiętać, że jeśli identyfikator (np. nazwa kolumny) zawiera znaki spoza standardowego zestawu ASCII (np. polskie litery), należy go ująć w lewe apostrofy (podobnie jak w sytuacji, kiedy identyfikator jest nazwą zastrzeżoną dla konstrukcji języka SQL).

Dla treningu spróbujmy teraz utworzyć prostą tabelę `klient`, która będzie zawierała dwie kolumny. Pierwsza — o nazwie `Indeks` — będzie przechowywała liczby całkowite (typ danych `INTEGER`), druga — o nazwie `Nazwa` — będzie przechowywała ciągi maksymalnie 20 znaków (typ `VARCHAR(20)`).

ĆWICZENIE

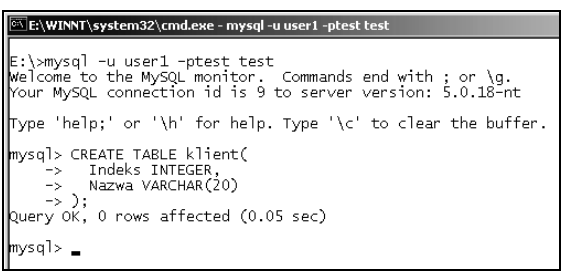
4.1 Utworzenie prostej tabeli

Utwórz tabelę o nazwie `klient` zawierającą dwie kolumny — pierwszą o nazwie `Indeks` typu `INTEGER`, drugą o nazwie `Nazwa` typu `VARCHAR(20)`.

Utworzenie takiej tabeli osiągniemy po wydaniu polecenia w postaci:

```
CREATE TABLE klient
(
  Indeks INTEGER,
  Nazwa VARCHAR(20)
);
```

Oczywiście najpierw należy uruchomić klienta `mysql`, zalogować się do serwera i wybrać bazę danych (np. `test`), tak jak było to opisywane we wcześniejszych rozdziałach. Po wykonaniu opisanych czynności w oknie konsoli zobaczymy widok zaprezentowany na rysunku 4.1.



Rysunek 4.1. *Zalogowanie do serwera i utworzenie tabeli klient w bazie test*

Co się jednak stanie, jeśli spróbujemy utworzyć tabelę o nazwie, która już istnieje w bazie? W takiej sytuacji zostanie zgłoszony błąd widoczny na rysunku 4.2. Jest to całkiem zrozumiałe zachowanie systemu. Czasem jednak chcielibyśmy utworzyć tabelę o zadanej nazwie tylko wtedy, gdy nie istnieje ona w bazie, a gdyby istniała — nie podejmować żadnego działania. W takiej sytuacji powinniśmy skorzystać z dodatkowej konstrukcji `IF NOT EXISTS` w ogólnej postaci:

```
CREATE TABLE IF NOT EXISTS nazwa_tabeli
(
  definicje kolumn
)
```

którą możemy przetłumaczyć jako: utwórz tabelę *nazwa_tabeli*, o ile nie istnieje ona jeszcze w bazie.

Rysunek 4.2. *Próba utworzenia już istniejącej tabeli*

```
mysql> CREATE TABLE klient(
    ->   Indeks INTEGER,
    ->   Nazwa VARCHAR(20)
    -> );
ERROR 1050 (42S01): Table 'klient' already exists
mysql>
```

ĆWICZENIE

4.2 Utworzenie tabeli, o ile nie istnieje ona już w bazie

Napisz instrukcję tworzącą tabelę `klient` (taką jak w ćwiczeniu 4.1), która nie spowoduje wystąpienia błędu w sytuacji, jeśli tabela o takiej nazwie będzie już istniała w bazie.

```
CREATE TABLE IF NOT EXISTS klient
(
  Indeks INTEGER,
  Nazwa VARCHAR(20)
);
```

Tworzona tabela może być również tymczasową, czyli taką, która zostanie automatycznie usunięta po zakończeniu połączenia. Co więcej, taka tabela jest powiązana wyłącznie z połączeniem, w którym została utworzona, tak więc dwóch użytkowników może w jednym czasie w jednej bazie utworzyć różne tabele o takiej samej nazwie. Tymczasowość tabeli zapewnia słowo `TEMPORARY` umieszczone za `CREATE`, czyli instrukcja w ogólnej postaci:

```
CREATE TEMPORARY TABLE nazwa_tabeli
(
  definicje kolumn
)
```

ĆWICZENIE

4.3 Tabela tymczasowa

Utwórz tymczasową tabelę o dwóch dowolnych kolumnach.

```
CREATE TEMPORARY TABLE test
(
  id INTEGER,
  wartosc INTEGER
);
```

Istnieje także możliwość utworzenia nowej tabeli na bazie już istniejącej. Stosujemy w tym celu instrukcję CREATE w postaci:

```
CREATE TABLE nowa_tabela LIKE istniejąca_tabela;
```

która oznacza: utwórz tabelę o nazwie *nowa_tabela* o strukturze takiej jak *istniejąca_tabela*.

ĆWICZENIE

4.4 Tworzenie jednej tabeli na bazie innej

Utwórz tabelę klient2 o strukturze pobranej z tabeli klient.

```
CREATE TABLE klient2 LIKE klient;
```

Typy danych

Każda kolumna tabeli w bazie danych ma przypisany typ, który określa rodzaj danych, jakie mogą być w niej przechowywane. Występujące w MySQL typy danych można podzielić na trzy grupy:

- liczbowe,
- daty i czasu,
- łańcuchowe.

Typy liczbowe

Typy liczbowe możemy podzielić na dwa rodzaje — typy całkowitoliczbowe oraz typy zmiennoprzecinkowe. Zgodnie z nazwami służą one do reprezentacji wartości całkowitych oraz zmiennoprzecinkowych (zmiennopozycyjnych, rzeczywistych). Typy całkowitoliczbowe zostały przedstawione w tabeli 4.1. Jeden z wymienionych w niej typów — INTEGER — wykorzystywaliśmy już przy tworzeniu przykładowej tabeli klient. W każdym z wymienionych przypadków z wyjątkiem BOOL i BOOLEAN można zastosować dodatkowy modyfikator określający maksymalną szerokość wyświetlania w sytuacji, kiedy liczba znaków wartości jest mniejsza niż maksymalna. Definicja typu ma wtedy postać:

```
nazwa_typu(ile)
```

Dozwolone są także modyfikatory UNSIGNED oraz ZEROFILL. Pierwszy z nich oznacza, że wartość ma być traktowana jako liczba bez znaku (czyli niedopuszczalne są wartości ujemne). Drugi powoduje, że jeżeli liczba cyfr w danej wartości jest mniejsza od maksymalnej liczby wyświetlanych znaków, wolne miejsca zostaną dopełnione zerami. Zastosowanie atrybutu ZEROFILL powoduje, że automatycznie zostanie również zastosowany atrybut UNSIGNED.

Przykładowo, jeżeli zostanie zastosowany typ TINYINT UNSIGNED, w poszczególnych wierszach kolumny będzie można zapisywać wartości od 0 do 255. Jeżeli natomiast zostanie zastosowany typ TINYINT(4) ZEROFILL, w poszczególnych wierszach kolumny również będzie można zapisywać jedynie wartości od 0 do 255, ale będą one wyświetlane zawsze w postaci czteroznakowej, w której wolne miejsca z prawej strony zostały wypełnione zerami. Oznacza to, że wartość 2 będzie wyświetlana jako 0002, wartość 64 jako 0064, a 128 jako 0128.

Tabela 4.1. *Typy całkowitoliczbowe*

Typ	Zakres wartości	Liczba zajmowanych bajtów	Opis
BIT	-	zmienna	W wersjach od 5.0.3 reprezentuje pola bitowe od 1 do 64 bitów, w wersjach wcześniejszych synonim dla TINYINT(1).
BOOL	-	1	Synonim dla TINYINT(1). Wartość 0 jest interpretowana jako false, wartość różna od 0 jako true. W przyszłości ma zostać wprowadzona pełna obsługa typu BOOLEAN.

Tabela 4.1. *Typy całkowitoliczbowe (ciąg dalszy)*

Typ	Zakres wartości	Liczba zajmowanych bajtów	Opis
BOOLEAN	-	1	Synonim dla TINYINT(1). Wartość 0 jest interpretowana jako false, wartość różna od 0 jako true. Wprowadzony w wersji 4.1.0. W przyszłości ma zostać wprowadzona pełna obsługa typu BOOLEAN.
TINYINT	Od -128 do 127 dla liczb ze znakiem i od 0 do 255 dla liczb bez znaku.	1	Reprezentacja bardzo małych wartości całkowitoliczbowych.
SMALLINT	Od -32768 (-2^{15}) do 32767 ($2^{15}-1$) dla liczb ze znakiem i od 0 do 65535 ($2^{16}-1$) dla liczb bez znaku.	2	Reprezentacja małych wartości całkowitoliczbowych.
MEDIUMINT	Od -8388608 (-2^{23}) do 8388607 ($2^{23}-1$) dla liczb ze znakiem i od 0 do 16777215 ($2^{24}-1$) dla liczb bez znaku.	3	Reprezentacja średnich wartości całkowitoliczbowych.
INT	Od -2147483648 (-2^{31}) do 2147483647 ($2^{31}-1$) dla liczb ze znakiem i od 0 do 4294967295 ($2^{32}-1$) dla liczb bez znaku.	4	Reprezentacja zwykłych wartości całkowitoliczbowych.

Tabela 4.1. *Typy całkowitoliczbowe (ciąg dalszy)*

Typ	Zakres wartości	Liczba zajmowanych bajtów	Opis
INTEGER	Od -2147483648 (-2^{31}) do 2147483647 ($2^{31}-1$) dla liczb ze znakiem i od 0 do 4294967295 ($2^{32}-1$) dla liczb bez znaku.	4	Synonim dla INT.
BIGINT	Od -9223372036854775808 (-2^{63}) do 9223372036854775807 ($2^{63}-1$) dla liczb ze znakiem i od 0 do 18446744073709551615 ($2^{64}-1$) dla liczb bez znaku.	8	Reprezentacja dużych wartości całkowitoliczbowych.

ĆWICZENIE

4.5 Tabela z kolumnami typu INTEGER

Utwórz tabelę, która będzie zawierała dwie kolumny typu INTEGER, pierwszą o nazwie id i drugą o nazwie znacznik.

```
CREATE TABLE test
(
  id INTEGER,
  znacznik INTEGER
);
```

ĆWICZENIE

4.6 Tabela z kolumnami typu INTEGER z dodatkowymi atrybutami

Utwórz tabelę, która będzie zawierała kolumnę typu INTEGER przechowującą wyłącznie dodatnie wartości z przedziału 0 – 65535, w której liczba wyświetlanych znaków będzie zawsze równa 5, a wolne miejsca wartości krótszych niż 5 znaków będą wypełniane zerami.

```
CREATE TABLE test
(
  id SMALLINT(5) ZEROFILL
);
```

Typy zmiennoprzecinkowe zostały przedstawione w tabeli 4.2. Podobnie jak w przypadku typów całkowitoliczbowych, istnieje możliwość zastosowania modyfikatora określającego szerokość wyświetlania. W przypadku typów FLOAT, DOUBLE i DOUBLE PRECISION występuje on zawsze jednocześnie z modyfikatorem określającym liczbę miejsc po przecinku, ogólnie:

```
nazwa_typu(mod1, mod2)
```

gdzie mod1 określa szerokość wyświetlania (całkowitą liczbę cyfr znaczących), a mod2 liczbę uwzględnianych miejsc po przecinku.

Tabela 4.2. *Typy zmiennoprzecinkowe*

Typ	Zakres wartości	Liczba zajmowanych bajtów	Opis
FLOAT (precyzja)	zmienny	4 lub 8	Parametr precyzja określa precyzję, z jaką będzie reprezentowana dana wartość rzeczywista. W przypadku wartości od 0 do 24 mamy do czynienia z liczbami o pojedynczej precyzji, a w przypadku wartość od 25 do 63 z liczbami o podwójnej precyzji, co odpowiada opisanym niżej typom FLOAT i DOUBLE.
FLOAT	od -3.402823466E+38 do 3.402823466E+38	4	Liczby zmiennoprzecinkowe pojedynczej precyzji.
DOUBLE	od -1.7976931348 623157E+308 do 1.7976931348 623157E+308	8	Liczby zmiennoprzecinkowe podwójnej precyzji.
DOUBLE PRECISION	jw.	jw.	Synonim dla DOUBLE.
REAL	jw.	jw.	Synonim dla DOUBLE.

Tabela 4.2. *Typy zmiennoprzecinkowe (ciąg dalszy)*

Typ	Zakres wartości	Liczba zajmowanych bajtów	Opis
DECIMAL	zmienny	zmienna	Wartości z separatorem dziesiętnym. W wersjach przed 5.0.3 przechowywana jako łańcuch znaków. Całkowita maksymalna liczba znaków i liczba znaków po separatorze dziesiętnym może być określana przez dodatkowe parametry.
DEC	jw.	jw.	Synonim dla DECIMAL.
NUMERIC	jw.	jw.	Synonim dla DECIMAL.
FIXED	jw.	jw.	Synonim dla DECIMAL, dodany w wersji 4.1.0.

W przypadku typu DECIMAL i jego synonimów możliwe jest zastosowanie modyfikatora określającego szerokość wyświetlania bez modyfikatora określającego liczbę miejsc po przecinku, czyli prawidłowa jest zarówno konstrukcja:

```
DECIMAL(mod1)
```

jak i:

```
DECIMAL(mod1, mod2)
```

W stosunku do typów zmiennoprzecinkowych można również stosować modyfikatory ZEROFILL oraz UNSIGNED. Znaczenie pierwszego z nich jest takie samo jak w przypadku typów całkowitoliczbowych. Zastosowanie modyfikatora UNSIGNED powoduje natomiast, że dozwolone będą jedynie wartości nieujemne, nie zmieni się natomiast zakres wartości możliwych do reprezentowania.