

Marcin Lis



ĆWICZENIA

PRAKTYCZNE

MySQL

Darmowa baza danych

Postaw na MySQL — na pewno Ci się przyda!

Rzut oka na SQL i MySQL, czyli jak porozumiec się z bazą danych i zacząć z nią pracować
Najważniejsze elementy MySQL, czyli co trzeba wiedzieć, żeby sprawnie poruszać się w bazie danych
Skomplikowane operacje na danych, czyli jak najbardziej efektywnie współpracować z MySQL



Wydanie II

Helion



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

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

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

Redaktor prowadzący: Ewelina Burska
Projekt okładki: Maciej Pasek

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

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)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie?cwmsq2>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzje.

ISBN: 978-83-246-6857-1

Copyright © Helion 2013

Printed in Poland.

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

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

Spis treści

	Wstęp	5
Rozdział 1.	Instalacja i konfiguracja	9
	Instalacja w systemie Windows	9
	Wstępna konfiguracja w systemie Windows	24
	Uruchamianie i zatrzymywanie serwera w systemie Windows	27
	Instalacja w systemie Linux	32
	Wstępna konfiguracja w systemie Linux	36
	Uruchamianie i zatrzymywanie serwera w systemie Linux	37
Rozdział 2.	Praca z serwerem	43
	Łączenie z serwerem	43
	Tworzenie i usuwanie baz danych	46
	Wybór bazy danych	47
	Tworzenie i usuwanie kont użytkowników	49
	Uprawnienia użytkowników	53
	Odbieranie uprawnień	61
	Użytkownicy anonimowi	63
	Systemy kodowania znaków	65
	Wczytywanie poleceń z plików zewnętrznych	72
	Lista dostępnych baz danych	73

Rozdział 3. Koncepcja relacyjnych baz danych	77
Tabele	77
Klucze	78
Relacje	80
Zasady projektowania tabel	85
Rozdział 4. Tworzenie struktury bazy danych	93
Ogólna postać instrukcji CREATE	93
Typy tabel	99
Typy danych	100
Atrybuty kolumn	113
Indeksy	118
Kodowanie znaków dla tabel i kolumn	122
Pobieranie struktury tabel	124
Modyfikowanie tabel	127
Usuwanie tabel	134
Tabele w praktyce	135
Rozdział 5. Wprowadzanie, modyfikacja i usuwanie danych	145
Wprowadzanie danych	145
Pobieranie danych	154
Modyfikacja danych	171
Usuwanie danych	174
Modyfikacja istniejącego klucza głównego	176
Rozdział 6. Złożone instrukcje SQL	179
Pobieranie danych z kilku tabel	179
Typy złączeń	183
Grupowanie danych	189
Perspektywy (widoki)	201
Klucze obce	204
Transakcje	208



4

Tworzenie struktury bazy danych

Ogólna postać instrukcji CREATE

Dane w bazie przechowywane są w tabelach. Pojęcie to zostało przedstawione w rozdziale 3. „Koncepcja relacyjnych baz danych”. Czas więc dowiedzieć się, w jaki sposób można tworzyć tabele. Służy do tego instrukcja `CREATE TABLE`, która w uproszczonej, schematycznej postaci wygląda następująco:

```
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].
);
```

Nazwy tabel i kolumn konwertowane są na standard Unicode i w wersji podstawowej mogą zawierać dowolne litery, cyfry, znaki \$ (dolar), znaki _ (podkreślenie), znaki o kodach od U+0080 do U+FFFF. Nie mogą jednak składać się z samych cyfr ani być słowem kluczowym (zastrzeżonym dla konstrukcji języka, np. `SELECT`, `CREATE`). Takie nazwy (ogólniej — identyfikatory) mogą być stosowane bezpośrednio.

W wersji rozszerzonej nazwy (identyfikatory) mogą zawierać praktycznie wszystkie znaki Unicode z zestawu podstawowego (tzw. podstawowy zestaw znaków wielonarodowych, ang. *Basic Multilingual Plane*, obejmuje znaki o kodach od U+0000 do U+FFFF) z wyjątkiem znaku o kodzie 0 (U+0000). Nazwy mogą się wtedy też składać z samych cyfr, a także ze słów kluczowych. W praktyce oznacza to możliwość stosowania dowolnych znaków specjalnych. Takie nazwy zawsze muszą być jednak ujmowane w znaki ` (znak umieszczony na klawiaturze pod znakiem tyldy, lewy apostrof, grawis, ang. *backtick, grave accent*), np. `SELECT`¹.

Uwaga: niezależnie od sposobu zapisu nazwa nie może kończyć się spacją.

Typ kolumny określa rodzaj danych, które dana kolumna będzie mogła przechowywać, np. daty, liczby itp. Występujące w MySQL typy danych zostaną omówione w kolejnym podrozdziale.

W nazwach tabel (oraz baz danych) mogą występować zarówno małe, jak i wielkie litery, jednak to, czy będą rozróżniane, zależy od systemu plików w systemie operacyjnym, w 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 UFS (są rozróżniane).

Nazwy kolumn również mogą zawierać małe i duże litery, jednak w tym wypadku nie są one rozróżniane i to niezależnie od wersji systemu operacyjnego czy systemu plików. Należy jednak pamiętać, że jeśli w obrębie pojedynczej instrukcji SQL jedna nazwa (nawet gdy wielkość liter nie jest rozróżniana) występuje kilkakrotnie, w każdym wystąpieniu powinna mieć taką samą postać.

W książce została przyjęta konwencja, że nazwy tabel i kolumn (oraz innych struktur) będą rozpoczynane wielką literą, choć ma to znaczenie wyłącznie estetyczne i pozostaje bez wpływu na działanie instrukcji języka. W nazwach tabel i kolumn będą także używane polskie znaki, należy więc zadbać o odpowiednie ustawienia związane z klientem i serwerem (było to opisane w rozdziale 3.). W przypadku wystąpienia problemów pomoc może ujęcie tych nazw w znaki ` (można

¹ Po włączeniu na serwerze trybu ANSI_QUOTES jako wyróżniki identyfikatora mogą być także stosowane znaki cudzysłowu prostego.

też zrezygnować ze stosowania znaków narodowych i ograniczyć się do alfabetu łacińskiego).

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)).

Ć W I C Z E N I E

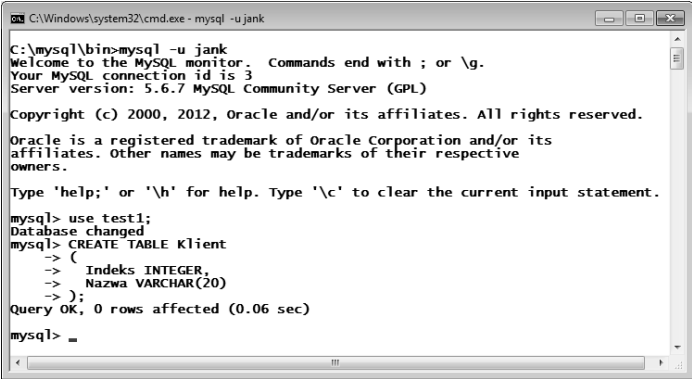
4.1 Utworzenie prostej tabeli

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. test1), tak jak było to opisywane we wcześniejszych rozdziałach. Po wykonaniu wymienionych czynności w oknie konsoli pojawi się widok zaprezentowany na rysunku 4.1.



```
C:\Windows\system32\cmd.exe - mysql -u jank
C:\mysql\bin>mysql -u jank
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.7 MySQL Community Server (GPL)
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use test1;
Database changed
mysql> CREATE TABLE Klient
-> (
->   Indeks INTEGER,
->   Nazwa VARCHAR(20)
-> );
Query OK, 0 rows affected (0.06 sec)
mysql> _
```

Rysunek 4.1. Zalogowanie do serwera i utworzenie tabeli Klient w bazie test1

Za pomocą instrukcji `CREATE TABLE` można utworzyć tabelę w dowolnej bazie danych, do której ma się odpowiednie prawa, a nie tylko w aktualnie wybranej (jak w ćwiczeniu 4.1). W takim wypadku oprócz nazwy tabeli należy podać nazwę bazy:

```
nazwa_bazy.nazwa_tabeli
```

Gdyby nazwy te miały zawierać jakieś znaki specjalne, należy je zgodnie z podanymi wyżej informacjami ująć w znaki ```. Często jednak w celu ujednolicenia kodu znaki te stosuje się zawsze, niezależnie od postaci nazwy.

Ć W I C Z E N I E

4.2 Tworzenie tabeli w konkretnej bazie danych

Utwórz przykładową tabelę bez wcześniejszego wybierania bazy danych. Tabela powinna mieć nazwę `SELECT` i znajdować się w bazie o nazwie `CREATE`.

Obie podane nazwy są fragmentami instrukcji SQL, a zatem aby można było ich użyć jako identyfikatorów, konieczne jest ujęcie wskazanych ciągów w znaki ```. Ponieważ tworzenie tabeli ma się odbywać bez wyboru bazy (bez wskazania bazy przy uruchamianiu klienta i bez użycia instrukcji `use`), to dodatkowo konieczne będzie wskazanie nazwy bazy w instrukcji `CREATE`. Zakładając więc, że w tabeli będzie tylko jedna kolumna o nazwie `Indeks` i typie `INTEGER` (oraz że wskazanej bazy nie ma jeszcze na serwerze), polecenie zawarte w ćwiczeniu można wykonać za pomocą dwóch następujących instrukcji (rysunek 4.2):

```
CREATE DATABASE `SELECT`;
CREATE TABLE `SELECT`.`CREATE`
(
  Indeks INTEGER
);
```

Rysunek 4.2.

Użycie słów kluczowych w nazwie bazy i tabeli

```
mysql> CREATE DATABASE `SELECT`;
Query OK, 1 row affected (0.01 sec)

mysql> CREATE TABLE `SELECT`.`CREATE`
-> (
->   Indeks INTEGER
-> );
Query OK, 0 rows affected (0.39 sec)

mysql> _
```


Co się jednak stanie, jeśli spróbujemy utworzyć tabelę o nazwie, która już istnieje w bazie? W takiej sytuacji zostanie zgłoszony błąd widoczny na rysunku 4.3 (przy bliższym przyjrzeniu się widać również, że w użytej wersji serwera nie jest rozpoznawana wielkość liter w nazwie tabeli). 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. Jeśli oczekujemy takiego zachowania od serwera, powinniśmy skorzystać z dodatkowej konstrukcji IF NOT EXISTS w ogólnej postaci:

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

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

Rysunek 4.3.

*Próba utworzenia
już istniejącej tabeli*

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

Ć W I C Z E N I E

4.3 Tworzenie 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 tymczasowa, czyli taka, która zostanie automatycznie usunięta po zakończeniu połączenia. Co więcej, tabela jest wtedy powiązana wyłącznie z połączeniem, w którym została utworzona, tak więc dwóch użytkowników może w jednym czasie w jednej bazie utworzyć różne tabele tymczasowe o takiej samej

nazwie. Tymczasowość tabeli zapewnia słowo `TEMPORARY` umieszczone za `CREATE`, czyli instrukcja w ogólnej postaci:

```
CREATE TEMPORARY TABLE nazwa_tabeli  
(  
    definicje_kolumn  
);
```

Ć W I C Z E N I E

4.4 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;
```

Oznacza ona: utwórz tabelę o nazwie *nowa_tabela* i o strukturze takiej jak *istniejąca_tabela*.

Ć W I C Z E N I E

4.5 Tworzenie jednej tabeli na podstawie innej

Utwórz tabelę `Klient2` o strukturze pobranej z tabeli `Klient`.

W celu wykonania ćwiczenia należy użyć instrukcji:

```
CREATE TABLE Klient2 LIKE Klient;
```

Typy tabel

MySQL obsługuje kilka rodzajów typów tabel. Domyślnie wykorzystywany jest typ InnoDB (od wersji 5.5) lub MyISAM (do wersji 5.1)². Zmianę typu domyślnego można osiągnąć, modyfikując parametr `default-storage-engine` w pliku konfiguracyjnym `my.ini` (`my.cnf`). MyISAM charakteryzuje się dużą szybkością, nie oferuje jednak wsparcia dla transakcji i kluczy obcych (rozdział 6.); obsługuje za to takie możliwości jak przeszukiwanie pełnotekstowe (ang. *full text search*) czy spakowane indeksy (ang. *packed indexes*). InnoDB jest domyślny w najnowszych wersjach bazy, nowocześniejszy i obsługuje transakcje (ang. *transactions*) oraz tzw. blokowanie wierszy (ang. *row locking*). W obecnych implementacjach okazuje się również bardzo wydajny.

Każda tabela bazy danych może mieć swój własny typ. Aby określić go podczas tworzenia tabeli, do instrukcji `CREATE TABLE` należy dodać ciąg³:

```
ENGINE = 'typ'
```

Jeśli zatem chcemy, aby tabela miała np. typ MyISAM, należy zastosować instrukcję:

```
CREATE TABLE nazwa_tabeli  
(  
    definicja_kolumn  
) ENGINE = 'MyISAM'
```

W przypadku już istniejącej tabeli jej typ można zmienić za pomocą instrukcji `ALTER TABLE` o schematycznej postaci:

```
ALTER TABLE nazwa_tabeli ENGINE='typ'
```

na przykład:

```
ALTER TABLE Klienci ENGINE='InnoDB'
```

² Nie są to jedyne rodzaje tabel. Opis pozostałych typów można znaleźć w dokumentacji MySQL, w sekcji „Storage Engines”.

³ W wersjach serwera starszych niż 4.0.18 powinien być to ciąg `TYPE = 'typ'`.

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 można podzielić na dwa rodzaje — typy całkowitoliczbowe (ang. *integer types*) oraz typy zmiennoprzecinkowe (ang. *floating point types*). 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` — był już używany 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`.

Na przykład 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 lewej 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 boolowskiego.
BOOLEAN	-	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 boolowskiego.
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 -32 768 (-2^{15}) do 32 767 ($2^{15} - 1$) dla liczb ze znakiem i od 0 do 65 535 ($2^{16} - 1$) dla liczb bez znaku.	2	Reprezentacja małych wartości całkowitoliczbowych.
MEDIUMINT	Od -8 388 608 (-2^{23}) do 8 388 607 ($2^{23} - 1$) dla liczb ze znakiem i od 0 do 16 777 215 ($2^{24} - 1$) dla liczb bez znaku.	3	Reprezentacja średnich wartości całkowitoliczbowych.

Tabela 4.1. Typy całkowitoliczbowe — ciąg dalszy

Typ	Zakres wartości	Liczba zajmowanych bajtów	Opis
INT	Od $-2\,147\,483\,648$ (-2^{31}) do $2\,147\,483\,647$ ($2^{31} - 1$) dla liczb ze znakiem i od 0 do $4\,294\,967\,295$ ($2^{32} - 1$) dla liczb bez znaku.	4	Reprezentacja zwykłych wartości całkowitoliczbowych.
INTEGER	Od $-2\,147\,483\,648$ (-2^{31}) do $2\,147\,483\,647$ ($2^{31} - 1$) dla liczb ze znakiem i od 0 do $4\,294\,967\,295$ ($2^{32} - 1$) dla liczb bez znaku.	4	Synonim dla INT.
BIGINT	Od $-9\,223\,372\,036\,854\,775\,808$ (-2^{63}) do $9\,223\,372\,036\,854\,775\,807$ ($2^{63} - 1$) dla liczb ze znakiem i od 0 do $18\,446\,744\,073\,709\,551\,615$ ($2^{64} - 1$) dla liczb bez znaku.	8	Reprezentacja dużych wartości całkowitoliczbowych.

Ć W I C Z E N I E

4.6 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
);
```

Ć W I C Z E N I E

4.7 Tabela z kolumnami typu INTEGER i dodatkowymi atrybutami

Utwórz tabelę, która będzie zawierała kolumnę typu INTEGER przechowującą wyłącznie dodatnie wartości z przedziału 0 – 65 535 i w której liczba wyświetlanych znaków będzie zawsze równa 5, a wolne miejsca dla 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.

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.

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 wartości 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 $-1.7976931348623157E+308$ do $1.7976931348623157E+308$	8	Liczby zmiennoprzecinkowe podwójnej precyzji.
DOUBLE PRECISION	jw.	jw.	Synonim dla DOUBLE.
REAL	jw.	jw.	Synonim dla DOUBLE.
DECIMAL	zmienny	zmienna	Wartości z separatorem dziesiętnym. W wersjach przed 5.0.3 przechowywana jako łańcuch znaków. Zarówno całkowita maksymalna liczba znaków, jak 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.

Ć W I C Z E N I E

4.8 Tabela z kolumną przechowującą wartości rzeczywiste

Utwórz tabelę zawierającą kolumnę przechowującą wartości rzeczywiste pojedynczej precyzji.

Przykładowa instrukcja tworząca taką tabelę może mieć postać:

```
CREATE TABLE Test
(
    Id INTEGER,
    Wartość FLOAT
);
```

Po użyciu powyższej instrukcji tabela Test będzie zawierała dwie kolumny, pierwszą o nazwie Id typu INTEGER i drugą o nazwie Wartość typu FLOAT.

Ć W I C Z E N I E

4.9 Kolumna z wartościami rzeczywistymi o określonej precyzji

Utwórz tabelę zawierającą kolumnę przechowującą wartości z separatorem dziesiętnym, z szerokością wyświetlania określoną na sześć cyfr znaczących, z trzema miejscami po przecinku.

```
CREATE TABLE Test
(
    Wartość DECIMAL(6, 3)
);
```

Typy daty i czasu

Typy pozwalające na reprezentację daty i czasu zostały zebrane w tabeli 4.3. Dane tych typów będą wyświetlane w formatach przedstawionych w kolumnie *Opis* tabeli, mogą być natomiast zapisywane w bazie przy użyciu różnych formatów. W przypadku typów DATE, DATETIME i TIMESTAMP dopuszczalne są formaty:

- ❑ Ciąg znaków RRRR-MM-DD GG:MM:SS i RR-MM-DD GG:MM:SS. Pomiedzy składowymi daty oraz pomiędzy składowymi czasu mogą występować dowolne znaki przestankowe. Prawidłowe są zatem zapisy: 2014-05-20 20:12:55, 2014.05.20 20-12-55, 2014*05*20 20%12%55.

Tabela 4.3. Typy daty i czasu

Typ	Dopuszczalne wartości	Liczba zajmowanych bajtów	Opis
DATE	Od 1000-01-01 do 9999-12-31.	3	Typ przeznaczony do reprezentacji daty. Wartości będą pobierane z bazy i wyświetlane w formacie RRRR-MM-DD.
DATETIME	Od 1000-01-01 00:00:00 do 9999-12-31 23:59:59.	8	Typ przeznaczony do reprezentacji daty i czasu. Wartości będą pobierane z bazy i wyświetlane w formacie RRRR-MM-DD GG:MM:SS.
TIMESTAMP	Zależne od dodatkowych opcji.	4	Typ przeznaczony do reprezentacji znacznika czasu.
TIME	Od -838:59:59 do 838:59:59.	3	Typ przeznaczony do reprezentacji czasu. Wartości będą pobierane z bazy i wyświetlane w formacie GG:MM:SS lub GGG:MM:SS.
YEAR	Od 1901 do 2155.	1	Typ przeznaczony do reprezentacji lat. Wartości będą pobierane z bazy i wyświetlane w formacie RRRR. Wartości tego typu są zapisywane na jednym bajcie.

- ❑ Ciąg znaków RRRR-MM-DD i YY-MM-DD. Pomiędzy składowymi daty mogą występować dowolne znaki przestankowe. Prawidłowe są zatem zapisy: 2014-05-20, 2014.05.20, 14*05*20.
- ❑ Ciąg znaków RRRRMMDDGGMMSS i RRMMDDGGMMSS. Pomiędzy składowymi nie mogą występować żadne znaki przestankowe, cały ciąg musi zaś reprezentować poprawną datę i czas. Prawidłowe są zatem zapisy: 20140520201255, 140520201255 — oba interpretowane jako 2014-05-20 20:12:55.
- ❑ Ciąg znaków RRRRRMDD i RRMMDD. Pomiędzy składowymi nie mogą występować żadne znaki przestankowe, cały ciąg musi zaś reprezentować poprawną datę. Prawidłowe są zatem zapisy: 20140520, 140520, oba interpretowane jako 2014-05-20.
- ❑ Wartość liczbowa zapisana jako RRRRMMDDGGMMSS, RRMMDDGGMMSS, RRRRMMDD lub RRMMDD, o ile reprezentuje poprawną datę i (lub) czas.

W przypadków typu TIME dopuszczalne są następujące formaty:

- ❑ Ciąg znaków D GG:MM:SS. Ciąg D reprezentuje dni i może przyjmować wartości od 0 do 34. Możliwe są również warianty skrócone w następujących postaciach: GG:MM:SS, GG:MM, D GG:MM, D GG i SS. Poprawne są zatem zapisy: 12:52:24, 12:52, 24.
- ❑ Ciąg znaków GGMMSS. Pomiędzy składowymi nie mogą występować żadne znaki przestankowe, cały ciąg musi zaś reprezentować poprawny czas. Poprawne są zatem zapisy: 125224, 182931.
- ❑ Wartość liczbowa zapisana jako GGMMSS, o ile reprezentuje poprawny czas. Możliwe są również alternatywne zapisy w postaci: MMSS, SS.

W przypadku typu YEAR dopuszczalne są następujące formaty:

- ❑ Ciąg znaków w formacie RRRR. Dopuszczalny zakres to 1901 – 2155.
- ❑ Ciąg znaków w formacie RR. Dopuszczalny zakres to 00 – 99. Ciągi od 00 do 69 są interpretowane jako lata 2000 – 2069, natomiast ciągi od 70 do 99 jako lata 1970 – 1999.
- ❑ Wartość liczbowa w formacie RRRR. Dopuszczalny zakres to 1901 – 2155.
- ❑ Wartość liczbowa w formacie RR. Dopuszczalny zakres to 1 – 99. Wartości od 1 do 69 są interpretowane jako lata 2001 – 2069, natomiast ciągi od 70 do 99 jako lata 1970 – 1999.

Jeśli w którymkolwiek z wymienionych przypadków zostanie podana wartość, która nie może zostać zinterpretowana jako poprawny argument danego typu, w bazie będzie ona interpretowana jako wartość specjalna:

- ❑ dla typu DATE — 0000-00-00,
- ❑ dla typu DATETIME — 0000-00-00 00:00:00,
- ❑ dla typu TIMESTAMP — 0000000000000000,
- ❑ dla typu TIME — 00:00:00,
- ❑ dla typu YEAR — 0000.

Ć W I C Z E N I E

4.10 Kolumna przechowująca dane o dacie i czasie

Utwórz tabelę zawierającą kolumnę przechowującą wartości określające jednocześnie datę i czas.

```
CREATE TABLE Test
(
  Dataiczas DATETIME
);
```

Typy łańcuchowe

Typy łańcuchowe służą do przechowywania zarówno ciągów znaków, jak i danych binarnych. Można je podzielić na cztery grupy:

- CHAR i VARCHAR,
- BINARY i VARBINARY,
- BLOB i TEXT,
- ENUM i SET.

Typy CHAR i VARCHAR

Typy CHAR i VARCHAR służą do przechowywania łańcuchów znakowych, czyli tekstów. Oba wymagają podania długości łańcucha za nazwą typu w nawiasie okrągłym, czyli:

```
CHAR(długość)
```

i

```
VARCHAR(długość)
```

gdzie *długość* oznacza liczbę znaków⁴. Na przykład jeśli chcemy utworzyć kolumnę, która będzie mogła przechowywać do 20 znaków, należy zastosować konstrukcję:

```
CHAR(20)
```

lub

```
VARCHAR(20)
```

⁴ W wersjach wcześniejszych niż 4.1 — liczbę bajtów.

W przypadku typu CHAR cała kolumna w bazie danych będzie miała długość wskazaną parametrem *długość*. Jeśli zapisywane dane będą miały mniej znaków, pozostałe miejsca zostaną uzupełnione spacjami z prawej strony. Spacje te, jak i te znajdujące się na początku tekstu (!), będą usuwane podczas pobierania danych. Parametr *długość* może przyjmować wartości od 0 do 255⁵.

W przypadku typu VARCHAR każdy wiersz kolumny ma zmienną długość wynikającą z liczby znaków zapisywanego łańcucha (plus 1 lub 2 bajty niezbędne do zapisania liczby znaków łańcucha). Parametr *długość* może przyjmować następujące wartości: od 1 do 255 w wersjach przed 4.0.2, od 0 do 255 w wersjach od 4.0.2 oraz od 0 do 65 535, począwszy od wersji 5.0.3. Należy jednocześnie zwrócić uwagę, że maksymalna długość jednego wiersza danych to również 65 535 bajtów, zatem kolumna typu VARCHAR o maksymalnej długości musiałaby być jedyną kolumną w wierszu i zawierać wyłącznie znaki jednobajtowe w liczbie nie większej niż 65 533 (ze względu na konieczność zapisania na dwóch bajtach liczby znaków). W wersjach przed 5.0.3 podczas zapisywania danych do bazy usuwane są spacje z początku i końca tekstu. Począwszy od wersji 5.0.3, spacje te nie są usuwane — ani podczas zapisu, ani podczas odczytu.

W przypadku próby zapisania w wierszu kolumny większej liczby znaków, niż wynika to z wartości parametru *długość*, nadmiarowa liczba bajtów zostanie obcięta i wygenerowane będzie ostrzeżenie.

Ć W I C Z E N I E

4.11 Tabela z kolumnami przechowującymi krótkie dane tekstowe

Utwórz tabelę, która będzie zawierała następujące kolumny: Id — zawierającą hipotetyczny identyfikator osoby, Imię — przechowującą imiona (o długości nie większej niż 25 znaków), Nazwisko — przechowującą nazwiska (o długości nie większej niż 35 znaków).

```
CREATE TABLE Osoby
(
  Id INTEGER,
  Imię VARCHAR(25),
  Nazwisko VARCHAR(35)
);
```

⁵ W wersjach wcześniejszych niż 3.23 — od 1 do 255.

Typy BINARY i VARBINARY

Typy BINARY i VARBINARY są podobne do CHAR i VARCHAR, z tą różnicą, że przechowują łańcuchy bajtów, a nie znaków. Typ BINARY definiuje się w postaci:

BINARY(*długość*)

natomiast typ VARBINARY w postaci:

VARBINARY(*długość*)

Pozostałe właściwości są analogiczne. Należy jedynie zwrócić uwagę, że parametr *długość* w tym przypadku oznacza liczbę bajtów, a nie znaków.

Typy BLOB i TEXT

Typy BLOB i TEXT służą do przechowywania dużej ilości danych. Typ BLOB (z ang. *Binary Large Object*) służy do przechowywania ciągów binarnych, natomiast TEXT — tekstowych. Oba typy dzielą się na cztery podtypy różniące się od siebie maksymalną wielkością danych, które mogą być za ich pomocą zapisane. Zobrazowano to w tabelach 4.4 i 4.5. Dane tych typów nie są zapisywane bezpośrednio w wierszu tabeli (jak w przypadku typów CHAR i VARCHAR), ale w osobnym obszarze pamięci. Dzięki temu w jednym wierszu może się znaleźć wiele kolumn przechowujących dane tych typów.

Tabela 4.4. Typy BLOB

Typ	Maksymalny rozmiar danych	Opis
TINYBLOB	255 ($2^8 - 1$) bajtów	Niewielki obiekt binarny
BLOB	65 535 ($2^{16} - 1$) bajtów	Zwykły obiekt binarny
MEDIUMBLOB	16 777 215 ($2^{24} - 1$)	Obiekt binarny średniej wielkości
LONGBLOB	4 294 967 295 ($2^{32} - 1$)	Duży obiekt binarny

Tabela 4.5. Typy TEXT

Typ	Maksymalny rozmiar danych	Opis
TINYTEXT	255 ($2^8 - 1$) bajtów	Niewielki obiekt tekstowy
TEXT	65 535 ($2^{16} - 1$) bajtów	Zwykły obiekt tekstowy
MEDIUMTEXT	16 777 215 ($2^{24} - 1$)	Obiekt tekstowy średniej wielkości
LONGTEXT	4 294 967 295 ($2^{32} - 1$)	Duży obiekt tekstowy

Ć W I C Z E N I E

4.12 Tabela z danymi typu TEXT

Utwórz tabelę zawierającą kolumny: Id typu INTEGER oraz Opis typu tekstowego; kolumna Opis powinna mieć możliwość przechowywania tekstu w ilości do 64 KiB⁶.

```
CREATE TABLE Test
(
  Id INTEGER,
  Opis TEXT
);
```

Typy ENUM i SET

Typ ENUM jest typem wyliczeniowym pozwalającym ograniczyć zbiór wartości, który będzie mógł być przechowywany w danej kolumnie. Dopuszczalne wartości definiuje się w nawiasie okrągłym za nazwą typu, oddzielając je od siebie znakami przecinka, schematycznie:

```
ENUM('wartość1', 'wartość2', ..., 'wartośćN')
```

W tak określonej kolumnie w pojedynczym wierszu będzie mogła się znaleźć tylko jedna z zadeklarowanych wartości (a także wartość NULL i pusty ciąg znaków zapisywany jako ''). Maksymalna liczba wartości w typie ENUM to 65 535. Przykładowo w kolumnie zdefiniowanej jako:

```
ENUM('jeden', 'dwa', 'trzy')
```

⁶ KiB (kibibyte) czyli $2^{10} = 1024$ bajty. Jednostka zgodna z normą IEC 60027-2 definiującą używane w informatyce jednostki miar. Pozwala to na uniknięcie niejednoznaczności powstających w przypadku stosowania oznaczeń SI, gdzie zapis KB może być rozumiany zarówno jako 1000 bajtów, jak i 1024 bajty.

będą dopuszczalne jedynie ciągi znaków: jeden, dwa i trzy. Każda wartość umieszczona w definicji typu otrzymuje swój indeks (pierwszy element otrzymuje wartość 1), który jest następnie używany w tabelach. Dzięki temu można zachować czytelną formę zapisywanych wartości przy jednoczesnej oszczędności miejsca w bazie. Uwaga: z tego też powodu nie zaleca się używania wartości, które mogą być interpretowane jako numeryczne. Przykładowy poniższy zapis jest prawidłowy:

```
ENUM('0', '1', '2')
```

ale może prowadzić do niejednoznaczności, lepiej więc unikać takich konstrukcji.

Ć W I C Z E N I E

4.13 Kolumna z typem wyliczeniowym ENUM

Utwórz tabelę zawierającą kolumnę, która będzie mogła przechowywać jedną z wartości: zielony, czerwony, niebieski.

```
CREATE TABLE Test
(
  Kolor ENUM('zielony', 'czerwony', 'niebieski')
);
```

Typ SET jest również typem wyliczeniowym, który definiowany jest w analogiczny sposób jak typ ENUM, czyli:

```
SET('wartość1', 'wartość2', ..., 'wartośćN')
```

W tym jednak przypadku każdy wiersz kolumny będzie mógł zawierać dowolny podzbiór zdefiniowanych wartości oddzielonych od siebie znakami przecinka. Czyli po zdefiniowaniu kolumny jako:

```
SET('jeden', 'dwa')
```

w każdym wierszu będą mogły być zarówno wartości jeden i dwa, jak i jednocześnie jeden, dwa. Maksymalna liczba wartości możliwa do zadeklarowania w typie SET to 64.

Ć W I C Z E N I E

4.14 Kolumna z typem wyliczeniowym SET

Utwórz tabelę zawierającą kolumnę, która będzie mogła przechowywać dowolny podzbiór z wartości: zielony, czerwony, niebieski.

```
CREATE TABLE Test
(
    Kolor SET('zielony', 'czerwony', 'niebieski')
);
```

Atrybuty kolumn

Każda kolumna może mieć dodatkowe atrybuty. Najczęściej spotykane, które często przydają się przy pracy z bazą danych, to: PRIMARY KEY, NOT NULL, AUTO_INCREMENT, DEFAULT. Oprócz nich często stosowane są także INDEX i UNIQUE, zostaną jednak opisane dopiero w kolejnym podrozdziale, jako że są związane z tzw. indeksami.

PRIMARY KEY (klucz główny)

Atrybut PRIMARY KEY oznacza, że dana kolumna jest kluczem głównym. Jednocześnie wymusza to indeksowanie tej kolumny (indeksy zostaną opisane w kolejnym podrozdziale; indeksowanie kolumny oznacza, że wewnątrz bazy powstanie specjalna struktura porządkująca dane w kolumnie, co przyspiesza wiele operacji takich jak sortowanie czy przeszukiwanie) oraz zapis w kolejnych wierszach unikatowych wartości (jest to zrozumiałe, skoro klucz podstawowy musi jednoznacznie identyfikować każdy wiersz).

Jeśli kolumna ma być kluczem głównym, za jej podstawową definicją należy umieścić słowa PRIMARY KEY:

```
CREATE TABLE nazwa_tabeli
(
    nazwa_kolumny typ_kolumny PRIMARY KEY,
    definicje_pozostałych_kolumn
);
```

Notatki

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**

MySQL. Darmowa baza danych. ĆWICZENIA PRAKTYCZNE

Wydanie II

Jeden klik – wyszukuj w mig!



MySQL to obecnie jeden z najlepszych, najwygodniejszych w obsłudze i najbardziej stabilnych systemów zarządzania bazami danych, dostępny w wielu wersjach na różne platformy sprzętowe, także w wersji darmowej. Każdy, kto kiedykolwiek zetknął się z problemem przechowywania dużej ilości danych, do których potrzebny jest szybki dostęp, niewątpliwie doceni zalety tego systemu. MySQL jest wydajny, elastyczny, pozwala stosować różne rodzaje zapytań i umożliwia przeszukiwanie bazy danych pod wieloma kątami.

Jeśli chcesz szybko i praktycznie przekonać się, czy MySQL spełnia Twoje wymagania, koniecznie przeczytaj tę książkę. Znajdziesz tu mnóstwo ćwiczeń, które pomogą Ci w lot poznać ważne aspekty pracy z systemem i umieścić w nim własne dane. Dowiesz się, jak zainstalować i skonfigurować MySQL. Nauczysz się tworzyć i usuwać bazy danych, nadawać uprawnienia użytkownikom oraz wzytywać polecenia z plików zewnętrznych. Zorientujesz się, jak zaprojektować najbardziej funkcjonalną bazę danych, manipulować danymi w jej obrębie i maksymalnie wykorzystywać instrukcje SQL. Po prostu bezboleśnie wejdziesz w skomplikowany świat baz danych!

- Instalacja i konfiguracja MySQL
- Praca z serwerem
- Koncepcja relacyjnych baz danych
- Tworzenie struktury bazy danych
- Wprowadzanie, modyfikacja i usuwanie danych
- Złożone instrukcje SQL

helion.pl
księgarnia internetowa



Helion

Nr katalogowy: 13387

Sprawdź najnowsze promocje:

• <http://helion.pl/promocje>

Książki najchętniej czytane:

• <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

• <http://helion.pl/novosci>



Księgarnia internetowa:

<http://helion.pl>



Zamówienia telefoniczne:

0 801 339900



0 601 339900

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>



KOD KORZYŚCI

ISBN 978-83-246-8857-1



Cena 34,90 zł

Informatyka w najlepszym wydaniu

9 788324 668571