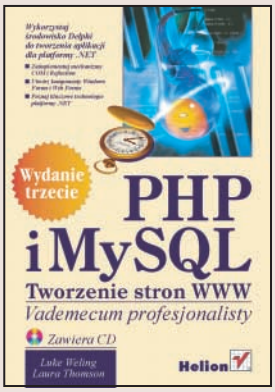
PHP i MySQL. Tworzenie stron WWW. Vademecum profesjonalisty. Wydanie trzecie



Autorzy: Luke Welling, Laura Thomson
Tłumaczenie: Paweł Gonera, Daniel Kaczmarek
ISBN: 83-7361-784-1
Tytuł oryginału: PHP and MySQL Web Development, 3rd Edition
Format: B5, stron: 912

Tandem PHP i MySQL to niewątpliwie najpopularniejsze i najpowszechniej rozpoznawane narzędzia do tworzenia dynamicznych witryn WWW i aplikacji internetowych. Ich popularność jest wynikiem nie tylko ogromnych możliwości, ale także bezpłatnego dostępu do obu narzędzi. Za pomocą PHP i MySQL tworzone są proste skrypty, mechanizmy zarządzania treścią serwisów WWW, sklepy internetowe i tysiące innych aplikacji. Społeczność programistów korzystających z PHP i MySQL stale się powiększa, a producenci tych narzędzi nieustannie pracują nad ich udoskonalaniem.

„PHP i MySQL. Tworzenie stron WWW. Vademecum profesjonalisty. Wydanie trzecie" to kolejne wydanie bestsellerowego podręcznika opisującego wszystkie aspekty projektowania stron i aplikacji internetowych z wykorzystaniem PHP i MySQL. W trzecim wydaniu książki znaleźć można szczegółowe omówienie możliwości najnowszych wersji obu narzędzi – programowania obiektowego, obsługi wyjątków, biblioteki SimpleXML oraz procedur składowanych. Książka zawiera przykłady demonstrujące wykorzystanie PHP i MySQL do realizacji różnych zadań związanych z funkcjonowaniem dynamicznych witryn WWW. Może to być uwierzytelnianie użytkowników, tworzenie koszyka na zakupy, dynamiczne generowanie obrazków oraz dokumentów w formacie PDF, wysyłanie poczty elektronicznej i zarządzanie nią oraz łączenie się z usługami WWW za pomocą XML-a.

- Podstawy języka PHP w wersji 5.0
- Programowanie obiektowe w PHP
- Praca z MySQL
- Projektowanie bazy danych
- Operacje na danych zgromadzonych w tabelach
- Administrowanie bazą danych
- Projektowanie komercyjnych witryn WWW
- Bezpieczeństwo witryn WWW
- Metody uwierzytelniania użytkowników
- Interakcja aplikacji z serwerem
- Mechanizmy kontroli sesji
- Dynamiczne generowanie grafiki i plików PDF
- Personalizacja witryny
- Korzystanie z usług sieciowych za pomocą protokołu SOAP

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

Spis treści

Rozdział 9.

Tworzenie internetowej bazy danych

W rozdziale tym przedstawimy sposób tworzenia bazy danych MySQL, przeznaczonej do udostępniania na stronach WWW.

W tym rozdziale zostaną poruszone następujące zagadnienia:

- tworzenie bazy danych,
- definiowanie użytkowników i przywilejów,
- wprowadzenie do systemu przywilejów,
- tworzenie tabel bazy danych,
- tworzenie indeksów,
- wybieranie typów kolumn w MySQL.

Również i w tym rozdziale posłużymy się znanym już przykładem księgarni internetowej „Książkorama". Przypomnijmy schemat jej bazy danych:

```
Klienci(KlientID, Nazwisko, Adres, Miejscowosc)

Zamowienia (ZamowienieID, KlientID, Wartosc, Data)

Ksiazki (ISBN, Autor, Tytul, Cena)

Pozycje_zamowione (ZamowienieID, ISBN, Ilosc)

Recenzje_ksiazek (ISBN, Recenzja)
```

Należy również przypomnieć, iż nazwy pól będących kluczami podstawowymi są podkreślone linią ciągłą, a nazwy pól będących kluczami obcymi — linią przerywaną.

W celu przeanalizowania materiału zawartego w tym rozdziale należy zapewnić sobie dostęp do serwera MySQL, co oznacza, że:

- Trzeba przeprowadzić instalację MySQL na serwerze WWW, obejmującą następujące czynności:
  - instalację plików,
  - rejestrację użytkownika na serwerze MySQL,

- zdefiniowanie odpowiedniej ścieżki dostępu,
- w razie konieczności uruchomienie aplikacji `mysql_install_db`,
- określenie hasła administratora,
- usunięcie użytkownika anonimowego,
- uruchomienie serwera MySQL po raz pierwszy i skonfigurowanie go w taki sposób, by uruchamiał się automatycznie.

Po wykonaniu wymienionych czynności można przystąpić do lektury tego rozdziału. Jeśli jednak instalacja nie powiodła się, należy kierować się instrukcjami zawartymi w dodatku A.

Ewentualne błędy, które mogą wystąpić w trakcie wykonywania czynności opisywanych w tym rozdziale, spowodowane będą prawdopodobnie nieprawidłową pracą serwera MySQL. W takim przypadku należy ponownie przeanalizować czynności opisane powyżej oraz przedstawione w dodatku A w celu upewnienia się, iż MySQL został prawidłowo zainstalowany.

- Należy zapewnić sobie dostęp do serwera MySQL zainstalowanego na komputerze znajdującym się w miejscu pracy czytelnika, udostępnianego przez dostawcę usług internetowych na zdalnym komputerze itp.

  W takim przypadku wykonanie zawartych w tym rozdziale przykładów lub utworzenie własnej bazy danych wymaga zarejestrowania przez administratora odpowiedniego użytkownika oraz znajomości nadanego mu identyfikatora, hasła dostępu i nazwy przypisanej mu bazy danych.

  Nie jest konieczne zapoznanie się z treścią podrozdziałów dotyczących rejestrowania użytkowników i przydzielania im bazy danych, chyba że istnieje potrzeba bardziej precyzyjnego wyjaśnienia administratorowi przedstawionych wcześniej wymagań. Zwykli użytkownicy nie będą wszakże mogli samodzielnie tworzyć własnych użytkowników i baz danych.

Wszystkie przykłady zawarte w tym rozdziale były uruchamiane na serwerze MySQL w wersji 5.0, która w trakcie pisania tej książki była wersją najnowszą. Niektóre wcześniejsze wersje serwera nie posiadają pewnych możliwości, dlatego też przed rozpoczęciem lektury należy zastąpić starszą wersję nową instalacją MySQL. Najnowszą wersję serwera MySQL można pobrać ze strony *http://mysql.com*.

W książce komunikację z serwerem MySQL prowadzimy przy użyciu klienta wiersza poleceń o nazwie MySQL monitor, stanowiącego składnik każdej instalacji. Nic jednak nie stoi na przeszkodzie, by używać innego klienta. Jeśli na przykład używany serwer MySQL znajduje się na dzierżawionym serwerze WWW, administratorzy systemu zwykle udostępniają użytkownikom działający przez przeglądarkę interfejs phpMyAdmin. Różne klienty GUI działają nieco inaczej, niż zostanie to przedstawione w tym rozdziale, lecz odpowiednie zaadaptowanie przedstawionych instrukcji nie powinno sprawić większego kłopotu.

Użytkowanie monitora MySQL

W przykładach przedstawionych w dalszej części tego rozdziału oraz w rozdziale następnym każda komenda jest zakończona znakiem średnika (;). Informuje on serwer MySQL o konieczności wykonania zadanego polecenia. Opuszczenie średnika spowoduje brak reakcji serwera, o czym często zapominają użytkownicy, zwłaszcza niedoświadczeni.

Opuszczenie znaku średnika spowoduje, że nastąpi przejście do nowego wiersza przed zakończeniem wpisywania tekstu komendy. Pozwoliło to nam na zwiększenie czytelności przykładów. Odnalezienie miejsca, w którym nastąpiło przejście do nowego wiersza, nie powinno sprawić żadnych problemów — MySQL wyświetli tam znak kontynuacji w formie strzałki przedstawionej poniżej:

```
mysql> grant select
    ->
```

Symbol ten oznacza, że MySQL czeka na wprowadzenie dalszej części polecenia. Za każdym razem, gdy zostanie naciśnięty klawisz *Enter* bez wcześniejszego wpisania znaku średnika, nastąpi przejście do nowego wiersza i wyświetlenie symbolu strzałki.

Warto również zapamiętać, że MySQL nie rozróżnia małych i wielkich liter w poleceniach języka SQL, może natomiast je rozróżniać w nazwach baz danych i tabel (kwestia ta zostanie poruszona w dalszej części rozdziału).

Logowanie się do serwera MySQL

W celu zalogowania się do serwera MySQL należy przejść do wiersza poleceń systemu operacyjnego i wpisać następującą komendę:

```
> mysql -h nazwa_komputera -u identyfikator_uzytkownika -p
```

Polecenie `mysql` wywołuje monitor MySQL, mający postać wiersza poleceń łączącego klienta z serwerem.

Opcji `-h` używa się w celu wskazania komputera, do którego ma nastąpić połączenie (na maszynie tej pracuje serwer MySQL). Jeśli polecenie to ma być wykonane na tym samym komputerze, na którym znajduje się MySQL, to opcja `-h` oraz parametr *nazwa_komputera* mogą zostać pominięte. W przeciwnym wypadku konieczne jest wpisanie w miejsce parametru *nazwa_komputera* nazwy maszyny, na której uruchomiony jest serwer bazy danych.

Opcja `-u` jest stosowana w celu wskazania identyfikatora użytkownika, na którego następuje logowanie. Jeśli identyfikator ten nie zostanie podany, wówczas MySQL domyślnie użyje identyfikatora tego użytkownika, który jest aktualnie zalogowany do systemu operacyjnego.

Jeśli MySQL został zainstalowany na komputerze lub serwerze czytelnika, będzie on zmuszony zalogować się jako użytkownik `root` oraz samodzielnie utworzyć bazę danych, którą będzie wykorzystywał w trakcie uruchamiania zawartych w tym rozdziale przykładów. Przy pierwszym uruchomieniu serwera użytkownik `root` jest jedynym zarejestrowanym

użytkownikiem. W trakcie pracy na serwerze zainstalowanym na komputerze administrowanym przez inną osobę należy przy logowaniu użyć identyfikatora użytkownika, podanego przez administratora.

Opcja `-p` informuje serwer o logowaniu się z użyciem hasła. Może ona zostać pominięta, jeśli dla danego użytkownika hasło nie zostało ustanowione.

Jeżeli czytelnik loguje się jako użytkownik `root` bez konieczności podania hasła, zalecane jest określenie hasła dostępu zgodnie z instrukcjami zawartymi w dodatku A. Jego brak powoduje, iż system będzie niedostatecznie zabezpieczony.

Podawanie hasła w tym samym wierszu co polecenie `mysql` nie jest obowiązkowe. Jeżeli nie zostanie ono podane, to serwer MySQL sam o nie „zapyta". Tak naprawdę pominięcie hasła jest właściwie lepszym rozwiązaniem, gdyż wpisane w wierszu poleceń będzie miało formę zwykłego tekstu, co umożliwi osobom postronnym jego wykrycie.

Po wpisaniu komendy podanej na początku podrozdziału wyświetlona zostanie odpowiedź w następującej formie:

```
Enter password:
```

(jeżeli polecenie to nie zadziała, trzeba sprawdzić, czy serwer MySQL jest uruchomiony, a komenda `mysql` może być zrealizowana w bieżącej ścieżce dostępu).

Następnie należy podać hasło dostępu. Jeśli wszystko przebiegnie prawidłowo, powinien zostać wyświetlony komunikat podobny do poniższego:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.0-alpha-max-debug

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Jeżeli czytelnik pracuje na własnym komputerze i komunikat nie będzie przypominał powyższego, wówczas należy upewnić się, czy uruchomiona jest aplikacja `mysql_ install_db` (jeżeli jest wymagana) i czy zostało określone i wpisane prawidłowo hasło dostępu dla użytkownika `root`. Jeśli praca odbywa się na zdalnej maszynie, trzeba upewnić się, iż podane hasło jest prawidłowe.

Po zalogowaniu się do serwera kursor powinien znajdować się przy znaku zachęty, umożliwiając stworzenie nowej bazy danych. Czytelnik pracujący na własnym komputerze powinien stosować się do instrukcji podanych w dalszej części rozdziału. Czytelnik korzystający z maszyny administrowanej przez inną osobę powinien mieć utworzoną i przypisaną mu bazę danych, może więc od razu przejść do lektury podrozdziału „Używanie odpowiedniej bazy danych". Można oczywiście zapoznać się z poprzedzającymi go podrozdziałami, nie będzie jednak możliwości (a przynajmniej nie powinno być) uruchamiania wyszczególnionych w nich poleceń.

Tworzenie baz i rejestrowanie użytkowników

System baz danych MySQL jest w stanie obsługiwać wiele baz danych jednocześnie. Zazwyczaj jedna aplikacja współpracuje z jedną bazą danych. Dla księgarni „Książkorama" baza ta nosi nazwę `Ksiazki`.

Tworzenie bazy danych

Tworzenie bazy danych to czynność najprostsza. W wierszu poleceń MySQL należy wpisać:

```
mysql> create database nazwa_bazy;
```

W miejsce *nazwa_bazy* należy podać nazwę bazy danych, która ma zostać utworzona. W naszym przykładzie jest to baza o nazwie `Ksiazki`.

I to wszystko! Otrzymana odpowiedź powinna wyglądać mniej więcej tak:

```
Query OK, 1 row affected (0.06 sec)
```

Oznacza to, że operacja została wykonana bez błędów. Jeśli taki komunikat się nie pojawi, należy upewnić się, że na końcu komendy został wpisany znak średnika. Informuje on serwer MySQL o zakończeniu wpisywania polecenia i o konieczności jego wykonania.

Definiowanie użytkowników i przywilejów

System MySQL może obsługiwać wielu użytkowników. Użytkownik `root` powinien być wykorzystywany w zasadzie tylko do celów administracyjnych, co jest podyktowane względami bezpieczeństwa. Dla każdego użytkownika, który będzie pracował w systemie, trzeba utworzyć konto i nadać mu hasło dostępu. Nie muszą one być takie same, jak identyfikator użytkownika i hasło wykorzystywane do zalogowania się do systemu operacyjnego. Ta sama zasada odnosi się do użytkownika `root`. Pożądane jest stosowanie różnych haseł dostępu w razie logowania się do systemu operacyjnego i serwera MySQL, szczególnie w przypadku użytkownika `root`.

Nadawanie hasła nowym użytkownikom nie jest obowiązkowe, jednak ze względów bezpieczeństwa pożądane jest, aby każdy użytkownik posiadał własne hasło dostępu. W przypadku tworzenia internetowej bazy danych wskazane jest utworzenie co najmniej jednego użytkownika, który będzie wykorzystywany tylko przez daną aplikację bazodanową. Można by zadać pytanie o cel tej operacji. Odpowiedź związana jest z systemem przywilejów.

Wprowadzenie do systemu przywilejów MySQL

Jedną z najważniejszych cech MySQL jest obsługa wyrafinowanego systemu przywilejów. *Przywilej* to inaczej posiadane przez określonego użytkownika prawo do wykonania określonego polecenia na określonym obiekcie. Idea przywilejów jest zbliżona do koncepcji praw dostępu do plików. Rejestrując nowego użytkownika MySQL, należy

nadać mu odpowiednie przywileje w celu wyszczególnienia czynności, które będzie on mógł wykonać w systemie.

Zasada najmniejszego przywileju

Zasada najmniejszego przywileju jest stosowana w celu zwiększenia bezpieczeństwa systemu komputerowego. Jest to podstawowa, ale bardzo ważna zasada, o której niestety często się zapomina. Brzmi ona następująco:

> Użytkownik (lub proces) powinien posiadać minimalny zbiór przywilejów potrzebnych do wykonania przypisanego mu zadania.

Zasadę tę należy stosować wszędzie, nie tylko w odniesieniu do MySQL. Na przykład do wysłania zapytania do bazy danych ze strony WWW użytkownik nie będzie potrzebował wszystkich przywilejów, które posiada użytkownik `root`. Należy więc utworzyć nowego użytkownika i nadać mu tylko przywileje niezbędne do uzyskania dostępu do bazy danych.

Rejestrowanie użytkowników: polecenie GRANT

Polecenia `GRANT` i `REVOKE` służą do nadawania i odbierania użytkownikom MySQL praw na czterech poziomach uprzywilejowania. Wyróżniamy następujące poziomy przywilejów:

- globalny,
- baza danych,
- tabela,
- kolumna.

W dalszej części rozdziału wyjaśnimy, który z nich i kiedy należy stosować.

Polecenie `GRANT` służy do tworzenia nowych użytkowników i nadawania im przywilejów. Składnia tego polecenia jest następująca:

```
GRANT przywileje [kolumny]
ON obiekt
TO identyfikator_uzytkownika [IDENTIFIED BY 'haslo']
[REQUIRE opcje_ssl]
[WITH [GRANT OPTION | ograniczenia] ]
```

Klauzule w nawiasach kwadratowych są opcjonalne. Poniżej zostaną omówione poszczególne parametry tego polecenia. Pierwszy z nich, *przywileje*, ma postać listy przywilejów oddzielonych przecinkami. MySQL posiada liczny zbiór przywilejów, opisanych w następnym podrozdziale.

Parametr *kolumny* jest opcjonalny. Używa się go w celu wskazania kolumn, do których podane przywileje zostaną zastosowane. Można podać nazwę pojedynczej kolumny lub listę nazw kolumn oddzielonych przecinkami.

Parametr *obiekt* wskazuje bazę lub tabelę, do której zostaną zastosowane podane przywileje. W celu nadania przywilejów na wszystkie bazy danych w systemie *obiekt* powinien przyjąć wartość `*.*`. Wówczas przywileje zostały nadane na poziomie *globalnym*.

Ten sam efekt można osiągnąć, przypisując parametrowi *obiekt* wartość *, jeśli nie jest używana żadna baza danych. Częściej jednak wskazuje się wszystkie tabele w bazie, wpisując *nazwa_bazy*.*, konkretną tabelę (*nazwa_bazy.nazwa_tabeli*) lub pojedyncze kolumny w danej tabeli poprzez wpisanie *nazwa_bazy.nazwa_tabeli* i nadanie odpowiedniej wartości parametrowi *kolumny*. Za pomocą tych metod nadaje się przywileje na pozostałych trzech poziomach uprzywilejowania, odpowiednio: *baza danych*, *tabela* i *kolumna*. Jeżeli w trakcie wykonywania tego polecenia używana jest konkretna baza danych, to podanie samej nazwy tabeli będzie zinterpretowane jako nadanie przywilejów tabeli o tej nazwie, znajdującej się w używanej bazie danych.

Parametr *identyfikator_uzytkownika* powinien wskazywać identyfikator, za pomocą którego użytkownik loguje się do serwera MySQL. Oczywiście nie musi być to ten sam, który jest używany przy logowaniu się do systemu operacyjnego. Wartość tego parametru może również zawierać nazwę komputera. Możliwość tę wykorzystuje się na przykład do odróżnienia użytkownika `laura` (traktowanego jako `laura@localhost`) od użytkownika `laura@gdzieindziej.com`. Rozwiązanie to jest szczególnie użyteczne w przypadku, gdy użytkownicy z różnych domen mają te same identyfikatory. Poza tym metoda ta zwiększa poziom bezpieczeństwa systemu, gdyż pozwala określić, z jakiej domeny konkretny użytkownik może się zalogować, a nawet do jakich tabel lub baz ma on dostęp z określonej lokalizacji.

Wartością parametru *haslo* jest hasło dostępu, którym powinien posługiwać się wskazany użytkownik w trakcie logowania do serwera. Stosuje się tutaj standardowe zasady doboru haseł. Kwestie bezpieczeństwa zostaną omówione w dalszej części książki, jednak należy zaznaczyć, że hasło nie powinno być łatwe do odgadnięcia. Niewskazane jest używanie słów, które można znaleźć w słowniku ani nie powinno być takie samo, jak identyfikator użytkownika. Najlepszym rozwiązaniem jest nadanie hasła składającego się z liter małych i wielkich oraz znaków nie będących literami.

Klauzula `REQUIRE` pozwala wskazać, że użytkownik musi łączyć się poprzez protokół Secure Sockets Layer (SSL), a także wskazać opcje SSL. Więcej informacji na temat połączeń z serwerem MySQL za pośrednictwem SSL można znaleźć w podręczniku MySQL.

Dodanie opcji `WITH GRANT OPTION` spowoduje, że wskazany użytkownik będzie mógł nadawać innym użytkownikom takie przywileje, jakie sam posiada.

Zamiast niej można użyć klauzuli `WITH` w postaci

```
MAX_QUERIES_PER_HOUR n
```

lub

```
MAX_UPDATES_PER_HOUR n
```

lub

```
MAX_CONNECTIONS_PER_HOUR n
```

Dzięki tym klauzulom można ograniczyć liczbę zapytań, uaktualnień lub połączeń, jakie jeden użytkownik może zrealizować w ciągu godziny. Rozwiązanie to może przydać się do ograniczania obciążenia systemów współużytkowanych generowanego przez jednego użytkownika.

Informacje dotyczące nadanych przywilejów zapamiętywane są w czterech tabelach systemowych, znajdujących się w bazie danych o nazwie `mysql`. Tabele te noszą nazwy: `mysql.user`, `mysql.db`, `mysql.host`, `mysql.tables_priv` oraz `mysql.columns_priv`. Zamiast wykorzystywać polecenie `GRANT`, można zmieniać dane bezpośrednio w wymienionych tabelach. Szczegółowe informacje na temat sposobu działania tych tabel oraz metod wprowadzania zmian bezpośrednio w nich zostaną przedstawione w rozdziale 12.

Typy i poziomy przywilejów

MySQL wykorzystuje trzy typy przywilejów:

- przywileje nadawane zwykłym użytkownikom,
- przywileje dla administratorów,
- przywileje specjalne.

Każdemu użytkownikowi można przyporządkować przywileje dowolnego typu, najrozsądniejsze jednak jest nadawanie przywilejów dla administratorów tylko administratorom systemu, czyli zgodnie ze wspomnianą wcześniej zasadą najmniejszego przywileju.

Użytkownikom należy nadawać takie przywileje, które umożliwią korzystanie tylko z potrzebnych im baz i tabel. Nikomu, z wyjątkiem administratora, nie należy udostępniać systemowej bazy `mysql`, gdyż w niej właśnie przechowywane są identyfikatory wszystkich użytkowników, ich hasła dostępu itp. (szerzej zagadnienie to jest opisane w rozdziale 12.).

Przywileje przeznaczone dla zwykłych użytkowników odwołują się bezpośrednio do konkretnych rodzajów poleceń SQL i określają, czy dany użytkownik może wykonywać polecenia tego typu. Polecenia języka SQL zostaną szerzej omówione w następnym rozdziale, teraz ograniczymy się tylko do krótkiego opisu (tabela 9.1). Kolumna „Zastosowanie" wskazuje obiekty, którym może być nadany określony typ przywilejów.

Tabela 9.1. *Przywileje dla użytkowników*

Przywilej	Zastosowanie	Opis
`SELECT`	Tabele, kolumny	Pozwala na wyszukiwanie wierszy (rekordów) z tabel
`INSERT`	Tabele, kolumny	Pozwala na wstawianie nowych wierszy do tabel
`UPDATE`	Tabele, kolumny	Pozwala na zmianę wartości wierszy zapisanych w tabeli
`DELETE`	Tabele	Pozwala na usuwanie z tabeli istniejących wierszy
`INDEX`	Tabele	Pozwala na tworzenie i usuwanie indeksów w poszczególnych tabelach
`ALTER`	Tabele	Pozwala na dokonywanie zmian w strukturze istniejących tabel, np. dodawanie nowych kolumn, zmianę nazw kolumn lub tabel, zmianę typu danych istniejących kolumn
`CREATE`	Bazy danych, tabele	Pozwala na tworzenie nowych tabel i baz danych. Jeżeli w ramach polecenia `GRANT` podano nazwę konkretnej tabeli lub kolumny, to użytkownik ma prawo utworzyć tabelę lub kolumnę tylko o tej nazwie, a więc najpierw będzie zmuszony ją usunąć
`DROP`	Bazy danych, tabele	Pozwala na usuwanie baz lub tabel

Większość przywilejów przeznaczonych dla zwykłych użytkowników nie powoduje zmniejszenia bezpieczeństwa systemu. Przywilej ALTER może być wykorzystywany do obchodzenia systemu przywilejów poprzez zmianę nazw tabel, jednak jest on potrzebny większości użytkowników. Ochrona systemu jest zawsze rezultatem kompromisu między jego użytecznością a poziomem zabezpieczeń. Administrator powinien samodzielnie podejmować decyzję o nadaniu, lub nie, przywileju ALTER zwykłym użytkownikom. Zazwyczaj jednak przywilej ten jest nadawany.

Oprócz przywilejów opisanych w tabeli 9.1 istnieje również przywilej REFERENCES (obecnie nie jest on praktycznie używany) oraz przywilej GRANT, nadawany częściej poprzez dodanie opcji WITH GRANT OPTION niż przez dołączanie go do listy wartości parametru *przywileje*.

W tabeli 9.2 opisano przywileje przeznaczone dla administratorów.

Tabela 9.2. *Przywileje dla administratorów*

Przywilej	Opis
CREATE TEMPORARY TABLES	Pozwala administratorowi na używanie słowa kluczowego TEMPORARY w instrukcjach CREATE TABLE
FILE	Pozwala na wczytywanie danych z plików do tabel i odwrotnie
LOCK TABLES	Pozwala na jawne używanie instrukcji LOCK TABLES
PROCESS	Pozwala na śledzenie procesów wykonywanych przez serwer i ich przerywanie
RELOAD	Pozwala na powtórne załadowanie tabel zawierających informacje na temat praw dostępu oraz na odświeżenie przywilejów, listy nazw łączących się komputerów, dziennika zdarzeń i tabel
REPLICATION CLIENT	Pozwala na używanie instrukcji SHOW STATUS na nadawcach i odbiorcach replikacji. Mechanizm replikacji zostanie opisany w rozdziale 12
REPLICATION SLAVE	Pozwala serwerom będącym odbiorcami replikacji na łączenie się z serwerem-nadawcą. Mechanizm replikacji zostanie opisany w rozdziale 12
SHOW DATABASES	Pozwala na odczytywanie listy wszystkich baz danych przy użyciu instrukcji SHOW DATABASES. Użytkownicy, którzy nie mają tego uprawnienia, będą widzieć wyłącznie bazy, do których przydzielono im dostęp
SHUTDOWN	Umożliwia zakończenie pracy serwera MySQL
SUPER	Pozwala administratorowi na zabijanie wątków należących do dowolnego użytkownika

Istnieje możliwość nadania opisanych przywilejów użytkownikom nie mającym statusu administratora, jednak należy tego dokonywać z zachowaniem najwyższej ostrożności.

Z przywilejem FILE sprawa wygląda nieco inaczej. Jest on bardzo przydatny dla użytkowników, gdyż możliwość wstawiania danych bezpośrednio z plików do tabel pozwala zaoszczędzić czas, konieczny na ręczne wpisywanie kolejnych wartości. Z drugiej strony jednak może zostać wykorzystany do załadowania wszelkich plików, które widzi serwer MySQL, np. pliku bazy należącej do innego użytkownika czy pliku zawierającego hasła dostępu. Przywilej ten powinien więc być nadawany z rozwagą lub też administrator powinien sam dokonywać ładowania danych z pliku do wskazanej przez użytkownika tabeli.

MySQL posiada również dwa specjalne przywileje, przedstawione w tabeli 9.3.

Tabela 9.3. *Przywileje specjalne*

Przywilej	Opis
ALL	Nadaje wszystkie przywileje opisane w tabelach 9.1 i 9.2. Jest on równoważny przywilejowi ALL PRIVILEGES
USAGE	Nie nadaje żadnych przywilejów. Powoduje zarejestrowanie użytkownika i pozwala mu na zalogowanie się, lecz jakiekolwiek inne czynności są dla niego niedostępne. Odpowiednie przywileje nadawane są zwykle później

Polecenie REVOKE

Przeciwieństwem GRANT jest polecenie REVOKE, używane w celu odebrania użytkownikowi określonych przywilejów. Jego składnia jest bardzo podobna do składni polecenia GRANT:

```
REVOKE przywileje [kolumny]
ON obiekt
FROM identyfikator_uzytkownika
```

Jeżeli do polecenia GRANT dołączono opcję WITH GRANT OPTION, przywilej ten można cofnąć (podobnie jak wszystkie inne przywileje) w następujący sposób:

```
REVOKE ALL PRIVILEGES, GRANT
FROM identyfikator_uzytkownika
```

Przykłady użycia poleceń GRANT i REVOKE

Aby zarejestrować użytkownika mającego status administratora, należy wpisać:

```
mysql> grant all
    -> on *
    -> to fred identified by 'mnb123'
-> with grant option;
```

Polecenie to spowoduje nadanie wszystkich przywilejów na wszystkie bazy danych użytkownikowi o identyfikatorze Fred, posługującemu się hasłem mnb123, oraz umożliwi nadanie dowolnego przywileju innym użytkownikom.

Niekiedy konieczne okazuje się wyeliminowanie tego użytkownika. Można to zrobić w następujący sposób:

```
mysql> revoke all privileges, grant
    -> from fred;
```

Zarejestrujmy teraz użytkownika, który nie będzie posiadał żadnych przywilejów:

```
mysql> grant usage
    -> on ksiazki.*
    -> to zosia identified by 'magic123';
```

Po rozmowie z Zosią wiadomo już, w jaki sposób chce korzystać z bazy, można więc nadać jej odpowiednie przywileje:

```
mysql> grant select, insert, update, delete, index, alter, create, drop
    -> on ksiazki.*
    -> to zosia;
```

Jak widać, nie ma już potrzeby podawania hasła dostępu, jakim posługuje się Zosia.

Jeżeli administrator dojdzie do wniosku, że Zosia wykonała już część swoich zadań, może ograniczyć nadane jej przywileje:

```
mysql> revoke alter, create, drop
    -> on ksiazki.*
    -> from zosia;
```

Kiedy Zosia nie będzie już potrzebowała dostępu do bazy danych, można odebrać jej wszystkie pozostałe przywileje:

```
mysql> revoke all
    -> on ksiazki.*
    -> from zosia;
```

Rejestrowanie użytkownika łączącego się z Internetu

Konieczne jest zarejestrowanie użytkownika, który łączy się z bazą danych poprzez stronę WWW zawierającą skrypty PHP. Również w tym przypadku należy zastosować zasadę najmniejszego przywileju, trzeba więc rozważyć, jakie operacje ma wykonywać skrypt PHP.

W większości przypadków wystarczą przywileje `SELECT`, `INSERT`, `DELETE` i `UPDATE`. Nadaje się je w następujący sposób:

```
mysql> grant select, insert, delete, update
    -> on ksiazki.*
    -> to ksiazkorama identified by 'ksiazkorama';
```

Oczywiście ze względów bezpieczeństwa hasło powinno być bardziej skomplikowane niż podane w przykładzie.

Użytkownik korzystający z usług zewnętrznej firmy internetowej otrzyma zapewne szersze przywileje na pracę z przydzieloną mu bazą danych. Identyfikator użytkownika i jego hasło dostępu będą prawdopodobnie umożliwiały zarówno wydawanie poleceń serwerowi MySQL (tworzenie tabel itp.), jak i łączenie się z bazą z poziomu strony WWW (np. wyszukiwanie danych w tabeli). Użycie takiego samego identyfikatora użytkownika i hasła mogłoby obniżyć nieznacznie stopień zabezpieczenia systemu, zalecane jest więc zarejestrowanie nowego użytkownika z następującym zbiorem przywilejów:

```
mysql> grant select, insert, update, delete, index, alter, create, drop
    -> on ksiazki.*
    -> to ksiazkorama identified by 'ksiazkorama';
```

Utwórz więc drugą wersję użytkownika, gdyż będziemy jej potrzebować w następnych sekcjach.

Wylogowanie się użytkownika root

Wylogowanie się z serwera MySQL jest dokonywane za pomocą polecenia `quit`. Zalecane jest ponowne zalogowanie się jako użytkownik internetowy (zarejestrowany w poprzednim punkcie) i sprawdzenie poprawności działania bazy danych. Jeżeli wpisana instrukcja `GRANT` została wykonana, lecz nadal nie można się zalogować, zwykle oznacza to, że w trakcie procesu instalacji nie usunięto anonimowego użytkownika. Z powrotem więc trzeba zalogować się jako użytkownik `root` i wykonać instrukcje zawarte w Dodatku A, zmierzające do usunięcia kont anonimowych. Po wykonaniu tej operacji nie powinno już być problemów z zalogowaniem się jako użytkownik internetowy.

Używanie odpowiedniej bazy danych

Przed przystąpieniem do lektury tego podrozdziału czytelnik powinien być już zalogowany do serwera MySQL na koncie zwykłego użytkownika, założonym w poprzednim podrozdziale lub też utworzonym przez administratora systemu.

Pierwszą czynnością, jaką należy wykonać, jest wskazanie bazy danych, która zostanie wykorzystana. Służy do tego następujące polecenie:

```
mysql> use nazwa_bazy;
```

gdzie *nazwa_bazy* jest nazwą bazy danych.

Tę samą operację można wykonać, podając nazwę bazy danych w trakcie logowania się do serwera:

```
> mysql -D nazwa_bazy -h nazwa_komputera -u identyfikator_uzytkownika -p
```

W omawianym przykładzie wykorzystywana będzie baza danych `Ksiazki`:

```
mysql> use ksiazki;
```

Po wykonaniu tego polecenia MySQL powinien zwrócić następujący komunikat:

```
Database changed
```

Jeżeli przed przystąpieniem do pracy nie zostanie wybrana żadna baza danych, serwer wyświetli komunikat o błędzie:

```
ERROR 1046 (3D000): No Database Selected
```

Tworzenie tabel bazy danych

Następnym etapem tworzenia bazy danych jest konstrukcja tabel. Służy do tego polecenie języka SQL `CREATE TABLE`, którego składnia przedstawia się następująco:

```
CREATE TABLE nazwa_tabeli(kolumny)
```

Parametr *nazwa_tabeli* powinien określać nazwę tabeli, która ma zostać utworzona. Z kolei parametr *kolumny* powinien mieć postać listy kolumn oddzielonych przecinkami, jakie

będzie zawierać nowa tabela. Każda z kolumn musi być określona przez podanie jej nazwy i typu danych.

Przypomnijmy schemat bazy danych księgarni „Książkorama":

```
Klienci(KlientID, Nazwisko, Adres, Miejscowosc)

Zamowienia (ZamowienieID, KlientID, Wartosc, Data)

Ksiazki (ISBN, Autor, Tytul, Cena)

Pozycje_zamowione (ZamowienieID, ISBN, Ilosc)

Recenzje_ksiazek (ISBN, Recenzja)
```

Na listingu 9.1 przedstawione są polecenia SQL, za pomocą których zostaną utworzone powyższe tabele (przy założeniu, że istnieje już baza danych `Ksiazki`). Kod ten znajduje się również w folderze *rozdzial_09* w pliku o nazwie *ksiazkorama.sql* (przykłady są dostępne na płycie CD dołączonej do książki).

Listing 9.1. *ksiazkorama.sql — kod SQL tworzy tabele w bazie danych księgarni „Książkorama"*

```
create table klienci
( klientid int unsigned not null auto_increment primary key,
  nazwisko char(50) not null,
  adres char(100) not null,
  miejscowosc char(30) not null
);

create table zamowienia
( zamowienieid int unsigned not null auto_increment primary key,
  klientid int unsigned not null,
  wartosc float(6,2),
  data date not null
);

create table ksiazki
( isbn char(13) not null primary key,
  autor char(50),
  tytul char(100),
  cena float(4,2)
);

create table pozycje_zamowione
( zamowienieid int unsigned not null,
  isbn char(13) not null,
  ilosc tinyint unsigned,

  primary key (zamowienieid, isbn)

);

create table recenzje_ksiazek
( isbn char(13) not null primary key,
  recenzja text
);
```

Kod SQL zawarty w pliku *ksiazkorama.sql* zostanie wykonany po wpisaniu następującego polecenia (zakładamy przy tym, że plik *ksiazkorama.sql* ma tę samą lokalizację, co aplikacja `mysql`):

```
> mysql -h nazwa_komputera -u ksiazkorama -D ksiazki -p < ksiazkorama.sql
```

(Pamiętaj, by parametr nazwa_komputera zastąpić nazwą swego komputera).

Mechanizm przekierowania do pliku zawierającego kod SQL jest bardzo poręczny, gdyż pozwala na edycję kodu za pomocą dowolnego edytora tekstowego przed jego wykonaniem.

Pojedyncza tabela jest tworzona za pomocą odrębnego polecenia `CREATE TABLE`. Jak widać, każda z tabel zawiera kolumny wyszczególnione w schemacie bazy danych, utworzonym w poprzednim rozdziale. Ponadto każdej kolumnie nadano nazwę oraz przypisano typ danych. Niektóre kolumny są opisywane przez dodatkowe atrybuty, objaśnione w następnym punkcie.

Znaczenie dodatkowych atrybutów kolumn

`NOT NULL` oznacza, że pole, przy którym ten atrybut stoi, musi w każdym wierszu tabeli mieć nadaną jakąś wartość. Jeżeli przy opisie nowej kolumny atrybut został pominięty, wówczas jej pola mogą być puste (`NULL`).

`AUTO_INCREMENT` jest specjalnym atrybutem nadawanym kolumnom przechowującym wartości całkowitoliczbowe. Jeżeli do tabeli zostanie wpisany nowy rekord, w którym wartość pola z atrybutem `AUTO_INCREMENT` będzie nieokreślona, wówczas serwer MySQL automatycznie wstawi w to miejsce unikalny identyfikator — będzie nim liczba całkowita o wartości równej dotychczasowej maksymalnej wartości w tej kolumnie i powiększonej o jeden. W każdej tabeli może występować najwyżej jedna kolumna z tym atrybutem. Kolumny z atrybutem `AUTO_INCREMENT` muszą być indeksowane.

Atrybut `PRIMARY KEY` oznacza, że wskazana kolumna jest kluczem podstawowym tabeli, a wartości w niej zapisywane muszą być unikalne. MySQL automatycznie indeksuje takie kolumny. Zauważmy, że w pliku *ksiazkorama.sql* wszystkim kolumnom z atrybutem `AUTO_INCREMENT` (np. `KlientID` w tabeli `Klienci`) przypisano również atrybut `PRIMARY KEY`, dzięki czemu został spełniony wymóg indeksowania pól z atrybutem `AUTO_INCREMENT`.

Atrybut `PRIMARY KEY` może być wyszczególniony zaraz za nazwą nowej kolumny wyłącznie w przypadku, gdy tylko ta jedna kolumna jest kluczem podstawowym tabeli. Inny sposób określenia klucza podstawowego został zaprezentowany dla tabeli `Pozycje_zamowione`. Wykorzystanie omawianej metody jest wymuszone tym, że w skład klucza podstawowego tabeli `Pozycje_zamowione` wchodzi nie jedna, ale dwie kolumny jednocześnie. (To prowadzi również do utworzenia indeksu bazującego na obydwóch kolumnach).

Atrybut `UNSIGNED` wpisany za identyfikatorem typu całkowitoliczbowego oznacza, że kolumna może zawierać tylko wartości nieujemne.

Typy kolumn

Rozważmy przykład pierwszej utworzonej tabeli:

```
create table klienci
( klientid int unsigned not null auto_increment primary key,
  nazwisko char(50) not null,
  adres char(100) not null,
  miejscowosc char(30) not null
);
```

Tworząc nową tabelę, należy każdej jej kolumnie przypisywać odpowiedni typ danych.

Według schematu tabela `klienci` składa się z czterech kolumn. Pierwsza z nich, `KlientID`, jest kluczem podstawowym o wartościach typu całkowitoliczbowego (typ `int`). Co więcej, wszystkie pola ID (`KlientID`, `ZamowienieID`) mogą mieć tylko wartości nieujemne. Wykorzystano również cechy atrybutu `AUTO_INCREMENT`, dzięki któremu MySQL sam dba o spełnienie powyższych wymogów — mamy więc jeden kłopot z głowy.

Pozostałe kolumny będą przechowywać dane mające postać łańcuchów znaków — nadano im więc typ `char` o określonej dla każdej kolumny długości łańcucha, zawartej w nawiasach okrągłych. Na przykład w polu `Nazwisko` można zapisywać łańcuchy zawierające nie więcej niż 50 znaków.

Nawet jeśli nazwisko klienta liczy mniej niż 50 liter wartość pola `Nazwisko`, zawsze będzie zajmować tyle pamięci, ile trzeba do zapisania 50 znaków. MySQL automatycznie doda do nazwiska tyle znaków spacji, ile brakuje do wykorzystania całej przydzielonej polu pamięci. Alternatywą dla typu `char` jest `varchar` — wartości pól tego typu zajmują zawsze o jeden bajt więcej niż trzeba do zapamiętania podanego łańcucha. Stosowanie pól typu `varchar` generalnie oszczędza więc pamięć, może jednak negatywnie wpływać na wydajność całego systemu.

Większość kolumn została zadeklarowana jako `NOT NULL`. Zastosowanie takiego niewielkiego rozwiązania optymalizacyjnego wszędzie tam, gdzie to możliwe, może nieznacznie polepszyć wydajność pracy systemu. Zagadnienia dotyczące optymalizacji pracy serwera MySQL zostaną poruszone w rozdziale 12.

W obrębie niektórych poleceń `CREATE TABLE` zawarte są elementy, które nie zostały jeszcze omówione. Rozważmy kolejny przykład:

```
create table zamowienia
( zamowienieid int unsigned not null auto_increment primary key,
  klientid int unsigned not null,
  wartosc float(6,2),
  data date not null
);
```

Kolumna `Wartosc` jest zadeklarowana jako liczba zmiennoprzecinkowa typu `float`. Dla większości typów liczbowych zmiennoprzecinkowych istnieje możliwość określenia maksymalnej liczby cyfr, jakie będą wyświetlane, oraz ilości miejsc znaczących po przecinku. W przykładzie księgarni internetowej wartość zamówień jest określana w złotówkach, zadeklarowano więc, iż ma ona nie więcej niż sześć cyfr plus dwie cyfry znaczące po przecinku.

Dane przechowywane w kolumnie `Data` mają typ `date`.

W rozważanej tabeli wszystkie kolumny, oprócz kolumny `Wartosc`, są zadeklarowane jako `NOT NULL`. Dlaczego? Otóż wpisując nowe zamówienie należy najpierw zapisać je w tabeli

Zamowienia, następnie dodać odpowiednie rekordy do tabeli Pozycje_zamowione, po czym obliczyć wartość całego zamówienia. W momencie wpisywania nowego zamówienia jego wartość nie zawsze jest znana, dlatego wartości pola Wartosc mogą być nieokreślone.

W podobny sposób jak Zamowienia jest tworzona tabela Ksiazki:

```
create table ksiazki
( isbn char(13) not null primary key,
  autor char(50),
  tytul char(100),
  cena float(4,2)
);
```

W tym przypadku nie ma potrzeby tworzenia klucza podstawowego, ponieważ każda publikacja ma własny, unikalny numer ISBN. Nie licząc kolumny ISBN, wszystkie pozostałe kolumny tej tabeli mogą posiadać wartości NULL. Można bowiem wyobrazić sobie sytuację, w której znany jest tylko numer ISBN książki, zanim jeszcze zostanie ujawniony jej tytuł, autor i cena.

Na przykładzie tabeli Pozycje_zamowione zademonstrowano sposób deklarowania kluczy podstawowych składających się z więcej niż jednej kolumny:

```
create table pozycje_zamowione
( zamowienieid int unsigned not null,
  isbn char(13) not null,
  ilosc tinyint unsigned,

  primary key (zamowienieid, isbn)

);
```

Pole Ilosc, w którym zapamiętywana jest liczba zamówionych egzemplarzy jednej książki, jest typu TINYINT UNSIGNED, a więc może przyjmować wartości liczb całkowitych z zakresu od 0 do 255.

Jak już wspomnieliśmy, klucze podstawowe składające się z więcej niż jednej kolumny należy deklarować za pomocą klauzuli PRIMARY KEY. Przykład jej użycia zaprezentowano powyżej.

Ostatnie polecenie z pliku *ksiazkorama.sql* powoduje utworzenie tabeli Recenzje_ksiazek:

```
create table recenzje_ksiazek
( isbn char(13) not null primary key,
  recenzja text
);
```

Użyto w nim nowego typu danych, który nie został jeszcze omówiony — to typ text. Jest on stosowany w przypadku dłuższych tekstów, np. artykułów prasowych. Istnieje kilka jego wariantów, które zostaną szczegółowo opisane w dalszej części tego rozdziału.

Dla jeszcze lepszego zrozumienia procesu tworzenia tabel podane zostaną zasady nadawania nazw kolumnom oraz ogólne informacje na temat identyfikatorów. Najpierw jednak powrócimy na chwilę do utworzonej bazy danych.

Rzut oka na bazę danych — polecenia SHOW i DESCRIBE

Po zalogowaniu się do serwera MySQL i wybraniu bazy `Ksiazki` można sprawdzić, jakie tabele wchodzą w jej skład, wpisując następujące polecenie:

```
mysql> show tables;
```

Serwer wyświetli wówczas listę wszystkich tabel wybranej bazy danych:

```
+------------------+
| Tables_in_ksiazki |
+------------------+
| klienci          |
| ksiazki          |
| pozycje_zamowione |
| recenzje_ksiazek  |
| zamowienia       |
+------------------+
5 rows in set (0.06 sec)
```

Polecenia `SHOW` można również użyć do wyświetlenia wszystkich baz danych udostępnianych przez serwer:

```
mysql> show databases;
```

Użytkownicy, którzy nie posiadają przywileju `SHOW DATABASES`, zobaczą jedynie listę baz danych, do których udzielono im dostępu.

Szczegółowe informacje na temat konkretnej tabeli są dostępne po użyciu polecenia `DESCRIBE`:

```
mysql> describe ksiazki;
```

MySQL wyświetli wszystkie informacje, które należało podać przy tworzeniu tabeli:

```
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| isbn  | char(13)   |      | PRI |         |       |
| autor | char(50)   | YES  |     | NULL    |       |
| tytul | char(100)  | YES  |     | NULL    |       |
| cena  | float(4,2) | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

Oba polecenia są szczególnie przydatne do przypomnienia sobie np. typu danych przechowywanych w konkretnej kolumnie czy przeanalizowania struktury bazy utworzonej przez innego projektanta.

Tworzenie indeksów

Już wcześniej krótko wspominaliśmy o indeksach, ponieważ definiowanie kluczy głównych jest równoznaczne z utworzeniem indeksów na tych kolumnach.

Jednym z częstych problemów, który sygnalizują niedoświadczeni użytkownicy serwera MySQL, jest to, że narzędzie, które jest uznawane za niezwykle szybkie, na ich bazach danych działa wolno. Przyczyną tego problemu jest po prostu to, że na bazie danych nie utworzono indeksów. (Wszak możliwe jest utworzenie tabel, które nie posiadają kluczy głównych ani indeksów).

Indeksy zostaną utworzone automatycznie na kolumnach, które wskazano jako klucze. Jeśli okaże się, że wiele zapytań wykonywanych jest na kolumnie, która nie jest kluczem, można pokusić się o utworzenie na niej indeksu w celu poprawienia wydajności. Można tego dokonać instrukcją `CREATE INDEX`. Jej ogólna postać jest następująca:

```
CREATE [UNIQUE|FULLTEXT] INDEX nazwa_indeksu
ON nazwa_tabeli (nazwa_kolumny_indeksowanej [(dlugosc)] [ASC|DESC, …])
```

(Indeksy `FULLTEXT`, czyli pełnotekstowe, służą do indeksowania pól tekstowych; więcej na ich temat powiemy w rozdziale 13.).

Opcjonalne pole `dlugosc` służy do wskazania, aby tylko pierwszych `dlugosc` znaków pola podlegała indeksowaniu. Można również wskazać, by indeks był ułożony w porządku rosnącym (`ASC`) lub malejącym (`DESC`). Porządkiem domyślnym jest porządek rosnący.

Uwaga na temat typów tabel

Niektórzy czytelnicy mogą już wiedzieć, że MySQL udostępnia więcej niż jeden typ tabel czy moduł przechowywania danych, w tym również te, które realizują mechanizmy transakcyjne. Więcej na temat typów tabel powiemy w rozdziale 13. Na razie wszystkie tabele znajdujące się w naszych bazach danych będą używały domyślnego modułu przechowywania danych MyISAM.

Identyfikatory MySQL

MySQL rozróżnia pięć typów identyfikatorów: bazy danych, tabele, kolumny i indeksy (wszystkie są już czytelnikowi znane) oraz aliasy (zostaną one szerzej omówione w następnym rozdziale).

Bazy danych w MySQL odwzorowują nazwy katalogów w strukturze plików systemu operacyjnego, natomiast tabele odpowiadają pojedynczym plikom. Fakt ten ma bezpośredni wpływ na zasady nazewnictwa oraz warunkuje rozróżnianie (lub nie) wielkich i małych liter w nazwach baz i tabel. Jeżeli system operacyjny rozróżnia wielkie i małe litery, wówczas czyni to również serwer baz danych (tak jest w przypadku systemu Unix), w przeciwnym razie wielkość liter nie będzie miała znaczenia (np. w przypadku Windows). W nazwach kolumn i aliasów wielkość liter nie odgrywa roli, jednak w obrębie jednego polecenia nazwy te muszą być zapisywane w taki sam sposób.

Warto wspomnieć, że lokalizacja katalogów i plików z danymi jest ustawiana w plikach konfiguracyjnych serwera MySQL. Można ją odnaleźć, wpisując w wierszu poleceń systemu operacyjnego następującą komendę:

```
mysqladmin variables
```

Następnie szukamy zmiennej `datadir`.

Podsumowanie rozważań na temat identyfikatorów znajduje się w tabeli 9.4. Jedynym wyjątkiem, o którym należy wspomnieć, jest fakt, że identyfikatory MySQL nie mogą zawierać znaków o kodzie ASCII równym 0 i 255 (zresztą i tak nie wiadomo, w jakim celu miałyby zostać użyte).

Tabela 9.4. *Identyfikatory MySQL*

Typ identyfikatora	Długość maksymalna	Rozróżnianie wielkości liter	Znaki dozwolone
Baza danych	64	Tak jak system operacyjny	Wszystkie dozwolone przez system operacyjny w nazwie katalogów, z wyjątkiem znaku /
Tabela	64	Tak jak system operacyjny	Wszystkie dozwolone przez system operacyjne w nazwie plików, z wyjątkiem znaków / i .
Kolumna	64	Nie	Wszystkie
Indeks	64	Nie	Wszystkie
Alias	255	Nie	Wszystkie

Podane zasady ulegają jednak częstym zmianom.

Począwszy od wersji 3.23.6 serwera MySQL identyfikatory mogą zawierać słowa zarezerwowane i wszelkiego rodzaju znaki specjalne. Jedynym wymogiem jest wówczas konieczność zamknięcia ich w odwrotnych apostrofach (uzyskiwanych przez wciśnięcie klawisza tyldy w górnym lewym rogu większości klawiatur). Można na przykład napisać:

```
create database `create database`;
```

Wcześniejsze wersje systemu MySQL są pod tym względem bardziej restrykcyjne i nie dozwalają wykonać tak sformułowanego polecenia.

Oczywiście z tak szerokich możliwości należy korzystać z umiarem. To, że baza *może* nosić nazwę `` `create database` ``, wcale nie oznacza, że *powinna* zostać tak nazwana. Stosuje się tu standardową zasadę nazewnictwa, według której identyfikatory powinny mieć sensowne brzmienie.

Wybór typów danych w kolumnach

W MySQL wyróżnia się trzy podstawowe typy danych, które mogą być przechowywane w kolumnach: liczbowy, daty i czasu oraz łańcuchowy. Każda z tych kategorii dzieli się na szereg podtypów. W tym podrozdziale scharakteryzujemy krótko wszystkie dostępne typy danych, natomiast ich wady i zalety omówimy w rozdziale 12.

Wielkość pamięci potrzebnej do przechowania jednej danej ściśle zależy od jej typu. Deklarując typ kolumny, należy kierować się następującą zasadą: zawsze wybiera się taki typ danych, który będzie zajmował najmniej pamięci i jednocześnie pozwoli na zapisanie wszystkich potrzebnych informacji.

W przypadku niektórych typów danych możliwe jest określenie maksymalnej szerokości wyświetlania. W poniższych tabelach wielkość tę zaznaczono literą *M*. Jeżeli jest ona opcjonalna, litera *M* znajduje się w nawiasach kwadratowych. Największa dopuszczalna wartość parametru *M* wynosi 255.

Wszystkie parametry opcjonalne są przedstawione w nawiasach kwadratowych.

Typy liczbowe

Typy liczbowe dzielą się na całkowitoliczbowe i zmiennoprzecinkowe. W przypadku tych ostatnich istnieje możliwość zadeklarowania liczby cyfr znaczących po przecinku — w tabelach wielkość ta jest oznaczona symbolem *D*. Maksymalna wartość parametru *D* jest wyznaczona przez mniejszą spośród dwóch liczb: 30 i $M - 2$ (tj. maksymalna szerokość wyświetlania minus 2 — jeden znak przecinka i jeden znak dla całkowitej części danej liczby).

W przypadku typów całkowitoliczbowych można je zawęzić do typu `UNSIGNED` tak, jak pokazano na listingu 9.1.

Dla wszystkich typów liczbowych można ustawić atrybut `ZEROFILL`. Powoduje on, że przy wyświetlaniu zawartości kolumn posiadających ten atrybut zapisane w nich liczby zostaną poprzedzone zerami. Jeżeli kolumna zostanie zdefiniowana jako `ZEROFILL`, automatycznie będzie ona również mieć typ `UNSIGNED`.

Typy całkowitoliczbowe zostały przedstawione w tabeli 9.5. W drugiej kolumnie tabeli zawarto dopuszczalny zakres danych w przypadku braku atrybutu `UNSIGNED` (pierwszy wiersz) i z ustawionym atrybutem `UNSIGNED` (drugi wiersz).

Tabela 9.5. *Typy całkowitoliczbowe*

Typ	Zakres	Wykorzystanie pamięci (w bajtach)	Opis
`TINYINT[(M)]`	–127..128 lub 0..255	1	Bardzo małe liczby całkowite
`BIT`			Synonim `TINYINT`
`BOOL`			Synonim `TINYINT`
`SMALLINT[(M)]`	–32768..32767 lub 0..65535	2	Małe liczby całkowite
`MEDIUMINT[(M)]`	–8388608..8388607 lub 0..16777215	3	Średnie liczby całkowite
`INT[(M)]`	$-2^{31}..2^{31}-1$ lub $0..2^{32}-1$	4	Zwykłe liczby całkowite
`INTEGER[(M)]`			Synonim typu `INT`
`BIGINT[(M)]`	$-2^{63}..2^{63}-1$ lub $0..2^{64}-1$	8	Duże liczby całkowite

Tabela 9.6 przedstawia typy zmiennoprzecinkowe.

Tabela 9.6. *Typy zmiennoprzecinkowe*

Typ	Zakres	Wykorzystanie pamięci (w bajtach)	Opis
`FLOAT(precyzja)`	Zależnie od precyzji	różny	Używany do deklarowania liczb zmiennoprzecinkowych o pojedynczej lub podwójnej precyzji
`FLOAT[(M, D)]`	±1.175494351E–38 ±3.402823466E+38	4	Liczby zmiennoprzecinkowe o pojedynczej precyzji. Typ ten jest równoznaczny z typem `FLOAT(4)`, pozwala przy tym na określenie szerokości wyświetlania i liczby cyfr znaczących po przecinku
`DOUBLE[(M, D)]`	±1.7976931348623157E–308 ±2.2250738585072014E+308	8	Liczby zmiennoprzecinkowe o podwójnej precyzji. Typ ten jest równoznaczny z typem `FLOAT(8)`, pozwala przy tym na określenie szerokości wyświetlania i liczby cyfr znaczących po przecinku
`DOUBLE PRECISION[(M, D)]`	Jak wyżej		Synonim typu `DOUBLE[(M, D)]`
`REAL[(M,D)]`	Jak wyżej		Synonim typu `DOUBLE[(M, D)]`
`DECIMAL[(M[, D])]`	Różny	*M*+2	Liczba zmiennoprzecinkowa przechowywana jako zmienna typu `CHAR`. Zakres zależy od wartości *M* — szerokości wyświetlania
`NUMERIC[(M, D)]`	Jak wyżej		Synonim typu `DECIMAL`
`DEC[(M, D)]`	Jak wyżej		Synonim typu `DECIMAL`
`FIXED[(M, D)]`	Jak wyżej		Synonim typu `DECIMAL`

Typy daty i czasu

MySQL obsługuje szereg typów daty i czasu — są one zaprezentowane w tabeli 9.7. Każdy z nich pozwala na wpisanie danych w formie liczbowej lub łańcucha znaków. Charakterystyczną cechą typu `TIMESTAMP` jest to, że jeżeli pole tego typu pozostanie niewypełnione, wówczas automatycznie zostaną w nim zapisane czas i data aktualnie wykonywanej operacji. Właściwość ta jest szczególnie przydatna do analizy transakcji.

Tabela 9.8 prezentuje możliwe dostępne formaty wyświetlania wartości typu `TIMESTAMP`.

Tabela 9.7. *Typy daty i czasu*

Typ	Zakres	Opis
DATE	1000-01-01do 9999-12-31	Data wyświetlana w formacie RRRR-MM-DD
TIME	–838:59:59 do 838:59:59	Czas wyświetlany w formacie GG:MM:SS. Zakres typu jest tak szeroki, że zapewne nigdy nie będzie w pełni wykorzystywany
DATETIME	1000-01-01 00:00:00 do 9999-12-31 23:59:59	Data i czas wyświetlane w formacie RRRR-MM-DD GG:MM:SS
TIMESTAMP[(*M*)]	1970-01-01 00:00:00 do roku 2037	Typ szczególnie przydatny do śledzenia transakcji. Format wyświetlania zależy od wartości parametru *M*, a górny zakres typu od systemu operacyjnego
YEAR[(*2* | *4*)]	70 – 69 (czyli 1970 – 2069) lub 1901 – 2155	Rok wyświetlany w formie dwu- lub czterocyfrowej. Jak widać, każdy z nich ma odmienny zakres

Tabela 9.8. *Formaty wyświetlania wartości typu TIMESTAMP*

Podany typ	Format wyświetlania
TIMESTAMP	RRRRMMDDGGMMSS
TIMESTAMP(14)	RRRRMMDDGGMMSS
TIMESTAMP(12)	RRMMDDGGMMSS
TIMESTAMP(10)	RRMMDDGGMM
TIMESTAMP(8)	RRRRMMDD
TIMESTAMP(6)	RRMMDD
TIMESTAMP(4)	RRMM
TIMESTAMP(2)	RR

Typy łańcuchowe

Typy łańcuchowe dzielą się na trzy grupy. Pierwsza z nich to klasyczne (zwykłe) łańcuchy znaków, czyli krótkie fragmenty tekstu. Do tej grupy zaliczamy typy CHAR (łańcuchy o stałej długości) oraz VARCHAR (łańcuchy o zmiennej długości). Dla każdego z nich można określić długość łańcucha. Łańcuchy zapisywane w kolumnach typu CHAR zostaną uzupełnione spacjami w celu pełnego wykorzystania dopuszczalnego limitu ich długości. Natomiast w kolumnach typu VARCHAR łańcuchy są zapisywane w ich pierwotnej formie, dzięki czemu zajmują one tylko tyle pamięci, ile potrzeba (tak więc w przypadku wartości CHAR MySQL usunie końcowe znaki spacji przy ich *pobieraniu* z bazy, natomiast ewentualne znaki spacji znajdujące się na końcu łańcucha VARCHAR zostaną wyeliminowane w momencie *zapisania* go do bazy). Wybór któregoś z tych typów sprowadza się więc do rozstrzygnięcia dylematu między zwiększeniem szybkości działania kosztem używanej pamięci a ograniczeniem zużycia pamięci i spadkiem wydajności systemu. Zagadnienie to zostanie wyczerpująco omówione w rozdziale 12.

Do drugiej grupy należą typy TEXT i BLOB. Wartości obu wymagają zróżnicowanych ilości pamięci. Typy te są przeznaczone do przechowywania, odpowiednio, dłuższych tekstów oraz danych binarnych. Wartości typu BLOB to tzw. *duże obiekty binarne* (ang. *binary large objects*), stąd też wywodzi się jego nazwa. Obiekty te mogą przechowywać każdy rodzaj danych, np. obrazy, dźwięki itp.

W praktyce jedyna różnica pomiędzy oboma typami polega na tym, że w danych typu BLOB rozróżniana jest wielkość liter, typ TEXT natomiast tej właściwości nie posiada. Ponieważ oba typy pozwalają na zapisywanie dużych ilości danych, sposób ich wykorzystania jest nieco bardziej skomplikowany. Zagadnienie to zostanie szerzej przedstawione w rozdziale 12.

Trzecia grupa składa się z dwóch typów specjalnych SET i ENUM. Typ SET jest używany w celu zawężenia wartości danych, które mogą być zapisane w danej kolumnie, do z góry określonego zbioru wartości, przy czym dane zapisane w kolumnie mogą mieć więcej niż jedną wartość z tego zbioru. Podany zbiór wartości może zawierać co najwyżej 64 elementy.

ENUM to inaczej typ wyliczeniowy. Jest on bardzo podobny do typu SET, z jedną różnicą: kolumny typu ENUM mogą zawierać tylko jedną spośród określonego zbioru wartości lub wartość NULL, a zbiór wartości dopuszczalnych może zawierać do 65 535 elementów.

Tabele 9.9, 9.10 i 9.11 zawierają podsumowanie właściwości typów łańcuchowych. Tabela 9.9 przedstawia zwykłe typy łańcuchowe.

Tabela 9.9. *Zwykłe typy łańcuchowe*

Typ	Zakres	Opis
[NATIONAL] CHAR(*M*) [BINARY | ASCII | UNICODE]	1 – 255 znaków	Łańcuch znaków o stałej długości *M*, gdzie *M* może przybierać wartości od 1 do 255. Słowo kluczowe NATIONAL wymusza użycie domyślnego zbioru znaków. Zbiór ten jest i tak domyślnie wykorzystywany przez MySQL, jednak opcja ta została udostępniona jako część standardu ANSI SQL. Słowo kluczowe BINARY wyłącza rozpoznawanie wielkości liter (domyślnie wielkość liter jest rozpoznawana). Słowo kluczowe ASCII oznacza, że w tej kolumnie używany będzie zestaw znaków latin1. Słowo kluczowe UNICODE natomiast wskazuje, że użyty zostanie zestaw znaków ucs
CHAR	1	Synonim typu CHAR(1)
[NATIONAL] VARCHAR(*M*) [BINARY]	1 – 255 znaków	Łańcuch znaków o różnej długości, reszta jak wyżej

Tabela 9.10 prezentuje typy TEXT i BLOB. Maksymalna liczba znaków w polu typu TEXT jest równa maksymalnej wielkości pliku, jaki można by przechowywać w tym polu, mierzonej w bajtach.

Tabela 9.11 prezentuje typy SET i ENUM.

Tabela 9.10. *Typy TEXT i BLOB*

Typ	Maksymalna długość (w znakach)	Opis
TINYBLOB	$2^8 - 1$ (czyli 255)	Mały obiekt BLOB
TINYTEXT	$2^8 - 1$ (czyli 255)	Krótkie pole tekstowe
BLOB	$2^{16} - 1$ (czyli 65535)	Zwykły obiekt BLOB
TEXT	$2^1 - 1$ (czyli 65535)	Pole tekstowe o zwykłej długości
MEDIUMBLOB	$2^{24} - 1$ (czyli 16777215)	Średni obiekt BLOB
MEDIUMTEXT	$2^{24} - 1$ (czyli 16777215)	Pole tekstowe o średniej długości
LONGBLOB	$2^{32} - 1$ (czyli 4294967295)	Duży obiekt BLOB
LONGTEXT	$2^{32} - 1$ (czyli 4294967295)	Długie pole tekstowe

Tabela 9.11. *Typy ENUM i SET*

Typ	Maksymalna ilość wartości w zbiorze	Opis
ENUM('*wartość1*','*wartość2*',…)	65535	W kolumnie tego typu może znajdować się tylko *jedna* wartość ze zbioru wartości dopuszczalnych lub NULL
SET('*wartość1*','*wartość2*',…)	64	W kolumnie tego typu może znajdować się *podzbiór* zbioru wartości dopuszczalnych lub NULL

Propozycje dalszych lektur

Więcej informacji na temat tworzenia bazy danych MySQL znajduje się w podręczniku elektronicznym, dostępnym pod adresem *http://www.mysql.com*.

W następnym rozdziale

Ponieważ znamy już metody rejestrowania użytkowników, zakładania baz danych i tworzenia tabel, możemy przejść do sposobów korzystania z bazy danych. W następnym rozdziale zostaną przedstawione metody zapisywania danych do tabel, ich modyfikacji i usuwania oraz wysyłania zapytań do bazy.