

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Bazy danych i MySQL. Od podstaw

Autorzy: Richard Stones, Neil Matthew

Tłumaczenie: Paweł Gonera

ISBN: 83-7197-728-X

Tytuł oryginału: [Beginning Databases with MySQL](#)

Format: B5, stron: 560

[Przykłady na ftp: 45 kB](#)



MySQL zajmuje szczególną pozycję wśród systemów obsługi relacyjnych baz danych. Dzięki dużej wydajności, prostocie obsługi i dostępności interfejsów programistycznych dla wielu języków programowania, osiągnął ogromną popularność w zastosowaniach internetowych. Jeśli dysponujesz ograniczonym budżetem, powinieneś rozważyć użycie darmowego systemu, który pod wieloma względami może konkurować z drogimi systemami komercyjnymi. MySQL rozwija się zresztą bardzo dynamicznie, a jego kolejne wersje dysponują coraz to bardziej wyrafinowanymi funkcjami.

Do kogo adresowana jest ta książka?

„Bazy danych i MySQL. Od podstaw” to pozycja zarówno dla początkujących, jak i dla zaawansowanych użytkowników MySQL-a. Początkujący dowiedzą się jak instalować i konfigurować system MySQL, nauczą się projektowania wydajnych baz danych i zgłębią tajemnice języka SQL. Doświadczeni programiści będą mogli poznać interfejs pozwalający na korzystanie z MySQL-a z poziomu ich ulubionego języka programowania (książka omawia C, C++, Perla, PHP i Javę). Zainteresować ich powinny także rozdziały poświęcone wzorcom projektowym stosowanym przy tworzeniu aplikacji bazodanowych i omówienie mechanizmu transakcji, zaimplementowanego w najnowszych wersjach MySQL-a.

Co zawiera ta książka?

- Przystępne omówienie zagadnień związanych z projektowaniem baz danych
- Opis kompilacji, instalacji, konfiguracji i obsługi systemu MySQL
- Kompletny kurs języka SQL w wersji implementowanej w MySQL
- Prezentację narzędzi ułatwiających pracę z MySQL
- Omówienie zagadnień związanych z administracją systemem bazodanowym (replikacja, tworzenie kopii zapasowych, odtwarzanie danych)
- Opis interfejsów umożliwiających pisanie aplikacji bazodanowych w językach C, C++, PHP, Perl i Java
- Dodatki, zawierające m.in. skorowidz poleceń SQL i informacje na temat obsługi dużych obiektów (pola BLOB)



Spis treści

| | |
|--|-----------|
| O Autorach | 13 |
| Wstęp | 15 |
| Rozdział 1. Wprowadzenie do MySQL | 19 |
| Zastosowanie danych przy programowaniu | 19 |
| Bazy danych korzystające z płaskich plików | 21 |
| Czym jest baza danych | 23 |
| Typy baz danych | 23 |
| Języki zapytań | 27 |
| System zarządzania bazą danych | 30 |
| Czym jest MySQL | 31 |
| Krótka historia MySQL | 31 |
| Architektura MySQL | 32 |
| Licencja open source | 33 |
| Zasoby | 34 |
| Rozdział 2. Podstawy relacyjnych baz danych | 35 |
| Arkusze kalkulacyjne | 35 |
| Terminologia | 36 |
| Ograniczenia arkuszy | 37 |
| Różnice pomiędzy arkuszem i bazą danych | 38 |
| Wybór kolumn | 39 |
| Wybór typów danych dla kolumn | 39 |
| Jednoznaczne identyfikowanie wierszy | 40 |
| Kolejność wierszy | 41 |
| Umieszczanie danych w bazie danych | 41 |
| Dostęp przez sieć | 41 |
| Danych cięcie i gięcie | 43 |
| Podstawy projektowania baz danych z wykorzystaniem wielu tabel | 45 |
| Relacje między tabelami | 46 |
| Projektowanie tabel | 49 |
| Kilka podstawowych zasad | 49 |
| Baza danych „Klienci i zamówienia” | 51 |
| Rozszerzamy bazę danych | 52 |
| Uzupełnianie projektu bazy danych | 54 |

| | |
|---|------------|
| Podstawowe typy danych | 57 |
| NULL | 57 |
| Przykładowa baza danych | 59 |
| Podsumowanie | 60 |
| Rozdział 3. Instalowanie i uruchamianie MySQL | 61 |
| Instalacja czy uaktualnienie? | 61 |
| Instalowanie MySQL z pakietu binarnego dla Linuksa lub Uniksa | 63 |
| Instalowanie MySQL z kodu źródłowego | 67 |
| Instalowanie MySQL w systemie Windows | 82 |
| Instalowanie MySQL z pakietu binarnego dla Windows | 82 |
| Podsumowanie | 88 |
| Rozdział 4. Dostęp do danych | 89 |
| Wykorzystanie narzędzia mysql | 90 |
| Proste wyrażenia SELECT | 93 |
| Przesłanie nazw kolumn | 96 |
| Ustalanie kolejności wierszy | 96 |
| Eliminowanie duplikatów | 98 |
| Wykonywanie obliczeń | 101 |
| Wybór wierszy | 102 |
| Bardziej skomplikowane wyrażenia | 105 |
| Dopasowanie wzorców | 107 |
| Ograniczanie wyniku | 108 |
| Porównanie różnych typów danych | 109 |
| Operacje na danych dotyczących dat i czasu | 112 |
| Funkcje daty i czasu | 113 |
| Wielokrotne złączenia tabel | 117 |
| Łączenie dwóch tabel | 117 |
| Synonimy nazw tabel | 122 |
| Łączenie trzech tabel | 123 |
| Podsumowanie | 127 |
| Rozdział 5. Narzędzia graficzne dla MySQL | 129 |
| mysql | 130 |
| Uruchamianie mysql | 130 |
| Polecenia w mysql | 130 |
| Historia poleceń | 131 |
| Skrypty w mysql | 131 |
| Poznawanie bazy danych | 133 |
| Parametry wiersza poleceń | 133 |
| Polecenia wewnętrzne | 135 |
| KSql i KMySQL | 135 |
| Przeglądarka tabel | 137 |
| Eksport HTML | 138 |
| Edytor formularzy | 140 |
| MySQL GUI | 141 |
| ODBC | 143 |
| Microsoft Access | 148 |
| Tabele połączone | 149 |
| Wprowadzanie danych | 151 |
| Raporty | 153 |

| | |
|---|------------|
| Microsoft Excel | 153 |
| Zasoby | 157 |
| Podsumowanie | 158 |
| Rozdział 6. Modyfikacja danych..... | 159 |
| Dodawanie danych do bazy danych | 159 |
| Instrukcja INSERT | 160 |
| Wstawianie danych do kolumn typu AUTO_INCREMENT | 164 |
| Wstawianie wartości NULL | 167 |
| Polecenie LOAD DATA | 169 |
| Ładowanie danych za pomocą mysqlimport | 172 |
| Bezpośrednie ładowanie danych z innych aplikacji | 174 |
| Aktualizacja bazy danych za pomocą instrukcji UPDATE | 176 |
| Usuwanie wierszy za pomocą instrukcji DELETE | 179 |
| Podsumowanie | 181 |
| Rozdział 7. Zaawansowana selekcja danych | 183 |
| Funkcje agregujące | 184 |
| COUNT() | 184 |
| Funkcja MIN() | 192 |
| Funkcja MAX() | 193 |
| Funkcja SUM() | 194 |
| Funkcja AVG() | 194 |
| Złączenia UNION | 195 |
| Złączenia własne | 197 |
| Złączenia zewnętrzne | 199 |
| Podzapytania | 202 |
| Typy podzapytań | 203 |
| Podzapytania skojarzone | 205 |
| Zastępowanie podzapytań | 207 |
| Podsumowanie | 210 |
| Rozdział 8. Definiowanie danych i manipulacja nimi | 213 |
| Typy danych | 214 |
| Typ logiczny | 214 |
| Typy znakowe | 215 |
| Typy numeryczne | 219 |
| Data oraz czas | 221 |
| Operatory | 222 |
| Kolejność i łączenie operatorów | 223 |
| Konwersje typów | 229 |
| Zmienne specjalne | 230 |
| Funkcje wbudowane | 231 |
| Operacje na tabelach | 235 |
| Tworzenie tabel | 235 |
| Typy tabel MySQL | 241 |
| Modyfikacja tabel | 242 |
| Usuwanie tabel | 246 |
| Tabele tymczasowe | 246 |
| Klucze obce | 247 |
| Deklarowanie kluczy obcych | 249 |
| Opcje kluczy obcych | 253 |
| Podsumowanie | 254 |

| | |
|--|------------|
| Rozdział 9. Transakcje i blokowanie | 255 |
| Czym są transakcje? | 256 |
| Zasady ACID | 259 |
| Obsługa transakcji dla jednego użytkownika | 260 |
| Obsługa transakcji dla wielu użytkowników | 261 |
| Poziomy izolacji ANSI | 261 |
| Definicje ANSI/ISO | 266 |
| Ograniczenia transakcji | 266 |
| MySQL i transakcje | 267 |
| MySQL i tabele MyISAM | 267 |
| MySQL i tabele InnoDB | 271 |
| Zakleszczenia | 276 |
| Podsumowanie | 278 |
| | |
| Rozdział 10. Administrowanie MySQL | 279 |
| Uruchamianie i zatrzymywanie serwera bazy danych | 279 |
| Windows | 280 |
| Linux | 282 |
| Wersje serwera | 284 |
| Dodawanie i usuwanie baz danych | 285 |
| Konfigurowanie użytkowników | 285 |
| Konto administratora bazy danych — root | 285 |
| Zarządzanie uprawnieniami użytkowników | 290 |
| Tworzenie kont użytkowników | 290 |
| Odbieranie uprawnień | 295 |
| Plik śladu serwera | 297 |
| Pliki konfiguracyjne | 297 |
| Ustawienia serwera | 297 |
| Pierwotne pliki konfiguracyjne | 298 |
| Pliki InnoDB | 301 |
| Przeglądanie baz danych | 303 |
| Kopie zapasowe | 303 |
| Podsumowanie | 307 |
| | |
| Rozdział 11. Projektowanie bazy danych..... | 309 |
| Rozpoznanie problemu | 309 |
| Cechy dobrego projektu bazy danych | 310 |
| Przechowywanie wymaganych danych | 311 |
| Tworzenie wymaganych relacji | 311 |
| Rozwiązywanie problemów | 311 |
| Wymuszenie integralności danych | 312 |
| Efektywny dostęp do danych | 312 |
| Rozszerzalność | 313 |
| Etapy projektowania | 313 |
| Zbieranie informacji | 313 |
| Projekt logiczny | 314 |
| Określanie relacji i liczebności | 319 |
| Konwersja do modelu fizycznego | 324 |
| Tworzenie kluczy głównych | 324 |
| Tworzenie kluczy obcych | 326 |

| | |
|---|------------|
| Ustalanie typów danych | 328 |
| Dokończenie definicji tabel | 330 |
| Sprawdzenie projektu | 331 |
| Postacie normalne | 332 |
| Pierwsza postać normalna | 332 |
| Druga postać normalna | 333 |
| Trzecia postać normalna | 333 |
| Często stosowane wzorce projektowe | 334 |
| Relacje wiele-do-wielu | 334 |
| Hierarchia | 335 |
| Relacje rekurencyjne | 335 |
| Zasoby | 337 |
| Podsumowanie | 338 |
| Rozdział 12. Dostęp do MySQL z programów napisanych w językach C i C++ | 339 |
| Użycie biblioteki libmysqlclient | 340 |
| Połączenie z bazą danych | 341 |
| Kontrola błędów | 346 |
| Wykorzystanie pliku Makefile | 346 |
| Dalsze informacje | 347 |
| Zarządzanie połączeniami do serwera | 348 |
| Wykonywanie zapytań SQL za pomocą libmysqlclient | 350 |
| Transakcje | 355 |
| Pobieranie danych zwracanych przez zapytania | 356 |
| Przeglądanie zbioru wynikowego | 361 |
| Operacje bez użycia kursorów | 362 |
| Podsumowanie funkcji API | 363 |
| Dostęp do MySQL z programów napisanych w języku C++ | 365 |
| Mysql++ | 366 |
| Instalacja Mysql++ | 366 |
| Kompilowanie programów z użyciem Mysql++ | 367 |
| Obiekty klasy Connection | 368 |
| Wyjątki | 371 |
| Zapytania | 371 |
| Zbiory wynikowe | 373 |
| Obiekty Query | 377 |
| Szablony zapytań | 378 |
| Zasoby | 379 |
| Podsumowanie | 379 |
| Rozdział 13. Dostęp do MySQL z programów napisanych w języku PHP | 381 |
| Dołączenie do PHP modułu wsparcia dla MySQL | 382 |
| Używanie PHP API z MySQL | 383 |
| Połączenia z bazą danych | 386 |
| Operacje na zbiorze wynikowym | 390 |
| Obsługa błędów | 404 |
| Używanie interfejsu Database Abstraction w PEAR | 412 |
| Podsumowanie | 419 |

| | |
|--|------------|
| Rozdział 14. Dostęp do MySQL z programów napisanych w języku Perl | 421 |
| Perl DBI | 422 |
| Instalowanie DBI oraz MySQL DBD | 423 |
| Korzystanie z DBI | 425 |
| Wykorzystanie DBIx::Easy | 453 |
| Podsumowanie | 455 |
| Rozdział 15. Dostęp do MySQL z programów napisanych w języku Java | 457 |
| Wstęp do JDBC | 458 |
| Sterowniki JDBC | 458 |
| Typ 1 — Mostek JDBC-ODBC | 459 |
| Typ 2 — Połączenie własnego API z kodem Java | 459 |
| Typ 3 — Realizacja protokołu sieciowego przy pomocy kodu Java | 459 |
| Typ 4 — Kod języka Java realizujący protokół systemu bazy danych | 460 |
| Kompilowanie sterownika JDBC dla MySQL | 460 |
| Interfejsy DriverManager oraz Driver | 461 |
| java.sql.DriverManager | 461 |
| java.sql.Driver | 464 |
| Połączenia | 466 |
| Tworzenie wyrażeń | 466 |
| Obsługa transakcji | 467 |
| Metadane | 468 |
| Zbiory wynikowe JDBC | 470 |
| Typy zbiorów wynikowych oraz współbieżność | 470 |
| Przeglądanie zbiorów wynikowych | 471 |
| Dostęp do danych wynikowych | 473 |
| Zbiory wynikowe podlegające aktualizacji | 474 |
| Inne ważne metody | 476 |
| Instrukcje JDBC | 476 |
| Instrukcje | 477 |
| Instrukcje przygotowywane | 481 |
| Wyjątki i ostrzeżenia SQL | 485 |
| Aplikacja graficzna korzystająca z JDBC | 485 |
| Diagram klas | 486 |
| Interakcja z systemem | 488 |
| Podsumowanie | 499 |
| Rozdział 16. Dodatkowe informacje i zasoby | 501 |
| Nierelacyjne źródła danych | 501 |
| Terminologia baz danych | 502 |
| Zasoby | 504 |
| Zasoby sieci | 505 |
| Narzędzia uniwersalne | 505 |
| Książki | 506 |
| Podsumowanie | 507 |
| Dodatek A Ograniczenia bazy danych MySQL | 509 |
| Rozmiar bazy danych — bez ograniczeń | 510 |
| Rozmiar tabeli — 64 TB – 16 PB | 510 |
| Ilość wierszy w tabeli — 2^{64} | 511 |
| Indeksy tabeli — 32 | 511 |

| | |
|--|------------|
| Rozmiar kolumny — 16 MB – 4 GB | 511 |
| Ilość kolumn w tabeli — 1000 | 511 |
| Rozmiar wiersza — 4 GB | 511 |
| Ograniczenia innego rodzaju | 512 |
| Dodatek B Typy danych w MySQL | 513 |
| Typy numeryczne — dokładne | 513 |
| Typy numeryczne — przybliżone | 514 |
| Typy reprezentujące daty i czas | 515 |
| Typy znakowe | 515 |
| Inne typy | 516 |
| Dodatek C Składnia MySQL | 517 |
| Polecenia SQL w MySQL | 517 |
| Składnia SQL w MySQL | 517 |
| Dodatek D Informator mysql | 529 |
| Opcje programu mysql wpisywane w wierszu poleceń | 529 |
| Polecenia wewnętrzne | 531 |
| Dodatek E Schemat bazy danych i definicje tabel | 533 |
| Dodatek F Obsługa dużych obiektów | 537 |
| Obiekty BLOB | 537 |
| Dane binarne w polach BLOB | 538 |
| Definicja BLOB w MyODBC | 538 |
| Użycie GROUP BY na polach BLOB | 538 |
| Programowanie pól BLOB | 539 |
| Skorowidz | 541 |

10

Administrowanie MySQL

W tym rozdziale zajmiemy się zarządzaniem serwerem MySQL. Sposób instalacji serwera został już opisany w rozdziale 3. — „Instalowanie i uruchamianie MySQL”, ale konfigurację serwera opisaliśmy tam tylko pobieżnie.

W tym rozdziale poruszymy następujące tematy:

- Uruchamianie i zatrzymywanie serwera bazy danych
- Konfigurowanie użytkowników
- Zarządzanie uprawnieniami użytkowników
- Pliki konfiguracyjne
- Kopie zapasowe

Administrowanie bazą danych jest złożonym przedsięwzięciem; w tym rozdziale omówimy jedynie podstawy administracji serwerem MySQL.

Uruchamianie i zatrzymywanie serwera bazy danych

Pomiędzy programem korzystającym z bazy danych a danymi zawartymi w tej bazie znajduje się kilka procesów zarządzających dostępem do danych. Zanim program kliencki będzie mógł uzyskać dostęp do danych, uruchomiony musi zostać proces serwera, co opisaliśmy w rozdziale 3. Wyłączając komputer musisz zapewnić, że proces ten zostanie prawidłowo zatrzymany. W takim przypadku proces serwera może odpowiednio zamknąć bazę danych, zapisując wszystkie dane znajdujące się w buforze. Jeżeli nie pozwolisz prawidłowo zamknąć bazy danych, ryzykujesz uszkodzeniem zapisanych w niej danych.

Zapewnienie prawidłowego zamknięcia bazy danych jest jedną z najważniejszych czynności, o których należy pamiętać w pracy z serwerem MySQL.

Najszybszą metodą sprawdzenia, czy lokalny serwer działa, jest uruchomienie programu *mysqladmin* z opcją *ping*:

```
[rick@gw1 rick]$ mysqladmin ping
mysqld is alive
[rick@gw1 rick]$
```

Aby uzyskać bardziej wyczerpujący raport, można zamiast opcji *ping* użyć *version*, co spowoduje wyświetlenie informacji o wersji uruchomionego serwera, czasie jego pracy oraz innych.

Gdy już wiemy, jak sprawdzić, czy serwer działa, zapoznamy się ze sposobami jego uruchamiania. Metody uruchamiania i zatrzymywania serwera zależą od rodzaju systemu operacyjnego, pod kontrolą którego pracuje serwer. Na początek zajmiemy się systemem Windows, a później Linux i Unix.

Windows

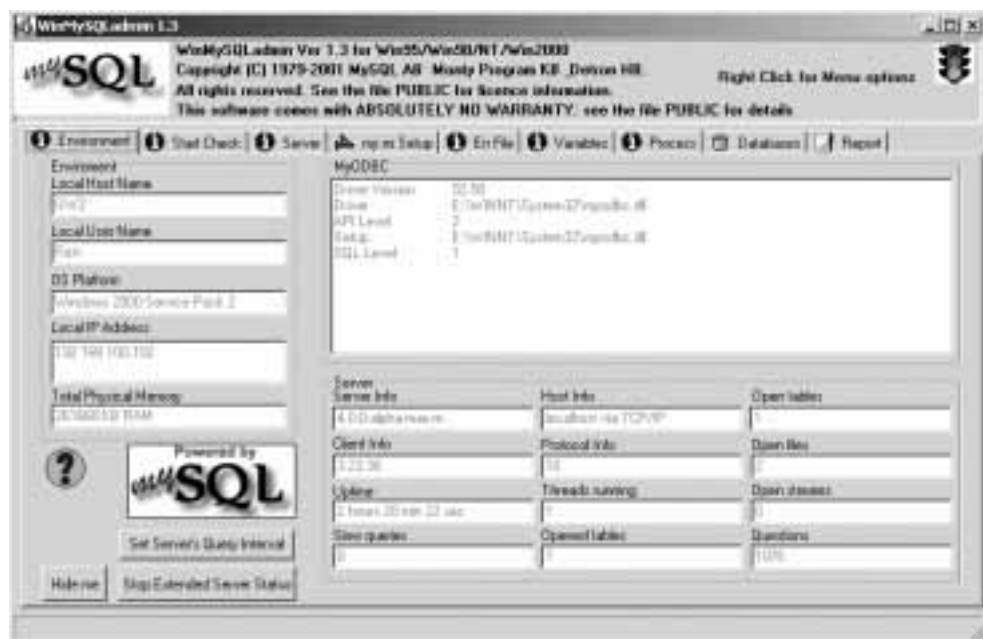
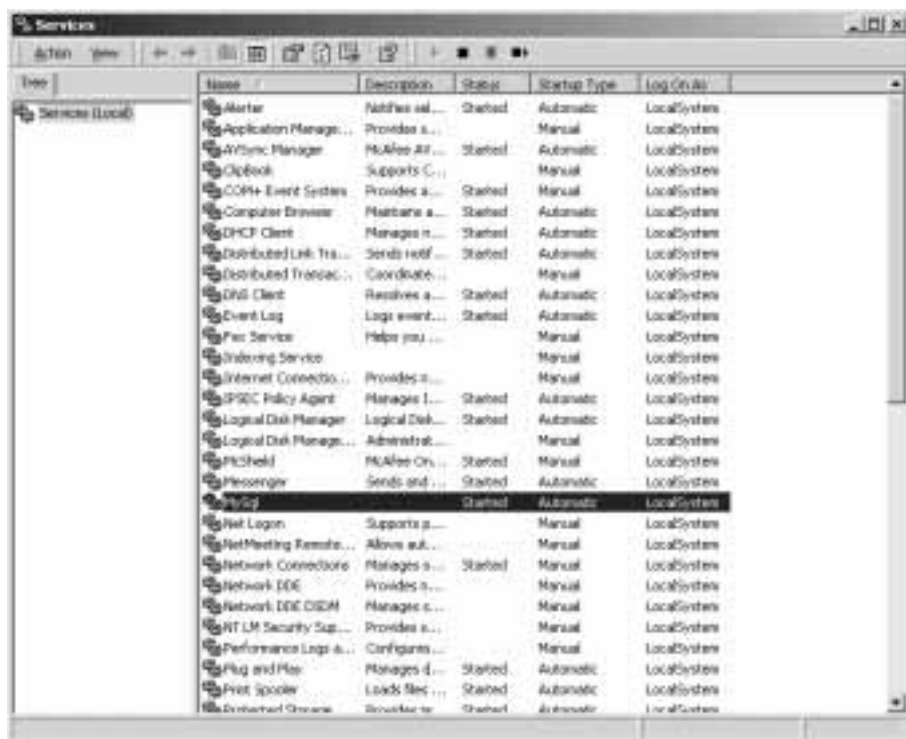
Jeżeli serwer MySQL działa w systemie Windows, najprawdopodobniej po zainstalowaniu programu i ponownym uruchomieniu komputera serwer zostanie uruchomiony, a podczas zamykania systemu automatycznie zatrzymany. Działanie takie zostało osiągnięte dzięki temu, że w Windows NT, 2000 i XP serwer MySQL jest domyślnie instalowany jako standardowa usługa systemu. Można to zobaczyć, przeglądając zawartość okna *Usługi* dostępnego w *Panelu sterowania* (dokładna lokalizacja zależy od wersji Windows). Jak widać, usługa MySQL jest uruchamiana automatycznie. Na górnym rysunku na następnej stronie pokazana jest lista usług dostępnych w systemie Windows 2000.

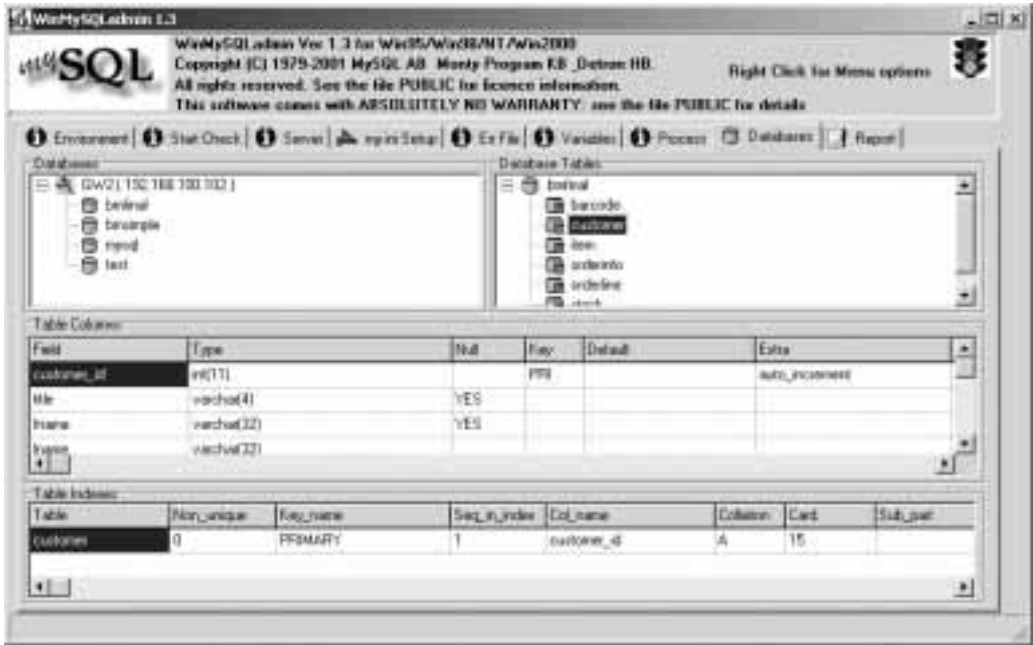
Dzięki takiemu ustawieniu operacje uruchamiania i zatrzymywania serwera są automatycznie wykonywane przez system operacyjny.

Program *WinMySQLManager*, znajdujący się w podkatalogu *bin* katalogu z instalacją MySQL, pozwala na sprawowanie kontroli nad serwerem MySQL na dosyć wysokim poziomie. Zapewnia on wygodny interfejs do zarządzania serwerem. Po uruchomieniu tego programu zauważysz, że po wyświetleniu okna wskazującego na start programu właściwie niewiele więcej się stało. Jeżeli jednak przyjrzyj się dokładniej, odszukasz małą ikonę przedstawiającą sygnalizator świateł drogowych, umieszczoną w zasobniku systemowym. Zielone światło sygnalizuje działanie serwera MySQL. Jeżeli klikniesz tę ikonę prawym klawiszem myszy i wybierzesz pozycję menu *Show me*, na ekranie pokaże się pełne okno programu *WinMySQLAdmin* (patrz dolny rysunek na następnej stronie).

Okno to zawiera podstawowe informacje na temat platformy i wersji serwera. Na kolejnych zakładkach znajdziesz wiele szczegółowych informacji na temat statusu i konfiguracji serwera. Na przykład na zakładce *Report* można wygenerować raport zawierający wiele informacji na temat działającego serwera. Jest to szczególnie użyteczne w przypadku, gdy występują kłopoty z prawidłowym działaniem serwera.

Bardzo przydatną zakładką jest *Databases*. Pozwala ona na przeglądanie informacji na temat baz danych, tabel w nich zawartych, a nawet definicji kolumn. Na rysunku na stronie 282 pokazana została definicja tabeli *customer* w bazie danych *bmsimple*.





Klikając prawym klawiszem myszy na ikonie świateł drogowych można dostać się do dodatkowych funkcji, dostępnych w podręcznym menu. Opcje te pozwalają na ręczne uruchomienie i zatrzymanie usługi; po zatrzymaniu usługi można usunąć usługę MySQL.

Linux

Nieco inaczej sprawuje się kontrolę nad serwerem MySQL w systemie Linux. Metoda zależy od tego, czy zainstalowałeś serwer korzystając z plików źródłowych, czy z gotowego pakietu, na przykład RPM dla systemów Red Hat lub SuSE Linux.

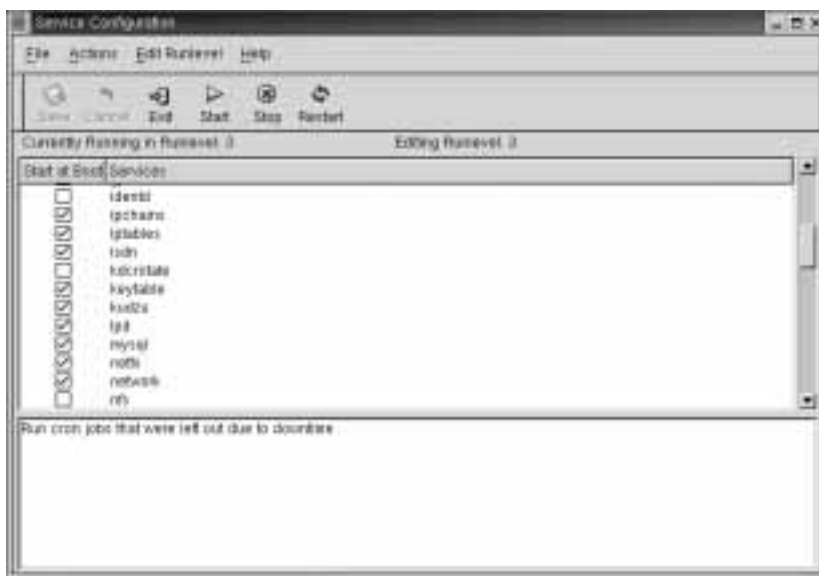
Najłatwiejszy sposób kontroli serwera jest dostępny w przypadku instalacji z pakietu instalacyjnego. Jeżeli wykonałeś taką instalację, najprawdopodobniej zainstalowane zostały odpowiednie skrypty służące do uruchamiania i zatrzymywania serwera. Dokładna lokalizacja tych skryptów zależy od dystrybucji. W przypadku Red Hat Linux jest to katalog */etc/rc.d/init.d*.

Zwykle system Linux posiada zdefiniowane odpowiednie poziomy działania (*runlevel*). Najczęściej numerem 3 oznacza się poziom umożliwiający pracę w środowisku tekstowym, z obsługą wielu użytkowników i sieci, natomiast numerem 5 poziom z uruchomionym środowiskiem graficznym. W większości systemów ustawienia domyślne są określone w pliku */etc/inittab*. Gdy system przechodzi na nowy poziom (lub opuszcza go), wykonywany jest zestaw skryptów przypisanych do tego poziomu. Zwykle są one umieszczone w katalogu */etc/rc.d/rcN.d*, gdzie *N* jest numerem poziomu. Skrypty te są najczęściej dołączeniami symbolicznymi do skryptów z katalogu */etc/rc.d/init.d*. Nazwy skryptów wykonywanych w celu uruchomienia poziomu zaczynają się od litery *S*, po której znajduje się liczba wskazująca na kolejność wykonywania skryptów. Pełny opis poziomów działania oraz skryptów startowych

wykracza poza ramy tej książki. Więcej informacji możesz znaleźć na stronie podręcznika *man* dla programu *init*, w dokumentacji *HOWTO* lub w książkach poświęconych administracji systemem Linux.

Najprostszą metodą ustalenia poziomu działania, na którym będzie uruchamiany serwer MySQL jest użycie odpowiedniego narzędzia graficznego. W systemie Red Hat 7.2 można skorzystać z programu *Serviceconf*, znajdującego się w menu *Programs, System, Serviceconf* dla GNOME lub *Red Hat, System, Serviceconf* dla KDE. Po uruchomieniu programu widzisz listę dostępnych usług z polami wyboru pozwalającymi określić, czy dany program będzie uruchamiany na wybranym poziomie.

Na poniższym rysunku przedstawiono wygląd okna *Service Configuration* w systemie działającym na poziomie 3., z wybranym serwerem MySQL.



Aby potwierdzić, że MySQL działa, można napisać w wierszu poleceń: `ps -e | grep mysqld` i sprawdzić, czy jest uruchomiony program *mysqld*.

Jeżeli instalując MySQL korzystałeś z plików źródłowych, musisz wykonać nieco więcej czynności. Na początek musisz odszukać skrypt *mysql_start_stop*, który powinien znajdować się w podkatalogu *bin* katalogu bazowego programu MySQL. Możesz użyć go do stworzenia własnego skryptu kontrolnego; w wielu przypadkach nie musisz go już potem modyfikować.

Skopiuj ten skrypt do katalogu ze skryptami wykonywanymi w momencie startu systemu (w Red Hat Linux jest to */etc/rc.d/init.d*), a następnie korzystając z Twojego ulubionego narzędzia zmień konfigurację tak, aby skrypt startowy był uruchamiany na wymaganym poziomie. Możesz w tym celu wykorzystać wspomniany wcześniej program *serviceconf*, a jeżeli wolisz narzędzia tekstowe, możesz użyć programu *chkconfig*, który pozwala na wygodne zarządzanie programami uruchamianymi na określonych poziomach pracy.

Jeżeli chcesz dodać skrypt *mysql* do programów uruchamianych w chwili przechodzenia na poziom 3, możesz skopiować ten skrypt do katalogu */etc/rc.d/init.d*, a następnie — za pomocą polecenia `chkconfig --level 3 mysql` — spowodować, że będzie on uruchamiany na poziomie 3. Więcej informacji na ten temat możesz znaleźć na stronie podręcznika *man chkconfig*.

Możesz oczywiście ręcznie uruchamiać i zatrzymywać serwer, ale uruchamianie automatyczne zmniejsza ryzyko przypadkowego wyłączenia komputera bez zatrzymania serwera bazy danych.

Wersje serwera

W przypadku systemu Linux, instalując serwer MySQL musiałeś wybrać odpowiedni pakiet do instalacji. Do wyboru miałeś standardowy pakiet serwera oraz opcjonalny pakiet *max*. Pakiet *max* jest wymagany w przypadku, gdy oprócz standardowych tabel MyISAM chcesz korzystać z tabel typu InnoDB lub BDB. Zalecamy zainstalowanie standardowego pakietu serwera oraz pakietu *max*.

Jeżeli samodzielnie kompilowałeś MySQL, korzystając z plików źródłowych, w trakcie konfigurowania kompilacji decydujesz, które opcje serwera będą Ci potrzebne (patrz rozdział 3.). Począwszy od MySQL 4.0, domyślnie dołączana jest obsługa tabel InnoDB. Po wykonaniu programów *configure*, *make* oraz *make install* zostaje zainstalowana odpowiednia wersja programu serwera.

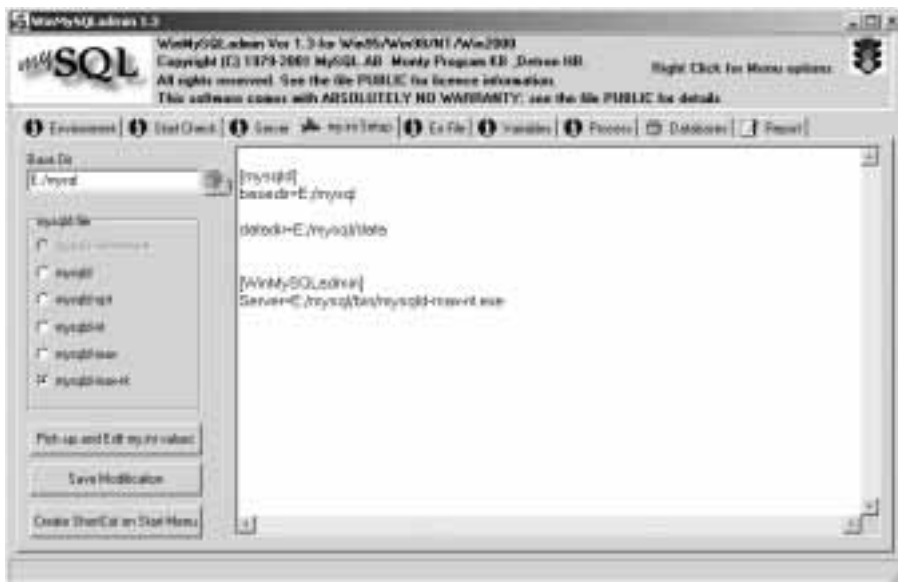
W przypadku Windows sprawa nieco się komplikuje, ponieważ zwykle w pakiecie instalacyjnym znajduje się kilka wersji serwera, a użytkownik musi wybrać odpowiadającą mu wersję. Operacje tę można wykonać za pomocą programu *WinMySQLAdmin*, korzystając z opcji na zakładce *my.ini Setup* (patrz rysunek na następnej stronie).

Po lewej stronie okna przedstawionego na rysunku na następnej stronie znajdują się nazwy dostępnych serwerów. Aby zmienić wersję serwera, należy wykonać następujące kroki:

- Wybierz nowy typ serwera.
- Zatrzymaj usługę.
- Usuń usługę.
- Zainstaluj usługę.
- Uruchom usługę.

Można to wykonać korzystając z wiersza poleceń, uruchamiając stary program z parametrem `--remove`, a następnie nową wersję programu serwera z opcją `--install`, co spowoduje zainstalowanie podanego w poleceniu pliku jako nowej usługi MySQL. Więcej informacji na ten temat znajduje się w podręczniku MySQL. Sugerujemy użycie wersji serwera *mysqld-max-NT*, jak jest to pokazane na rysunku na następnej stronie.

Jeżeli chcesz korzystać z tabel InnoDB, których używaliśmy w rozdziale 9. — „Transakcje i blokowanie”, przy omawianiu transakcji oraz kluczy obcych, zalecamy korzystanie z wersji *max* lub *max-nt*.



Dodawanie i usuwanie baz danych

Dodawanie i usuwanie baz danych jest dosyć proste. Aby stworzyć bazę danych, należy wykonać instrukcję:

```
CREATE DATABASE nazwa_bazy;
```

Aby usunąć bazę danych, należy wykonać:

```
DROP DATABASE nazwa_bazy;
```

Aby wykonać te operacje, użytkownik podłączony do serwera musi posiadać odpowiednie uprawnienia, o których napiszemy w kolejnym podrozdziale.

Konfigurowanie użytkowników

Teraz, gdy nasz serwer bazy danych jest automatycznie uruchamiany i — co ważniejsze — automatycznie zatrzymywany, możemy zająć się opisem konfigurowania użytkowników.

Konto administratora bazy danych — root

Podczas instalowania serwera MySQL tworzone są dwa konta użytkowników — domyślne konto administratora oraz konto anonimowe. Konto administratora nosi nazwę *root*, ale nie jest to konto związane w jakikolwiek sposób z kontem w systemie Linux o tej samej nazwie.

Domyślnie użytkownik *root* nie ma hasła oraz posiada pełną kontrolę nad bazą danych. Pierwszą czynnością powinno być przydzielenie hasła dla tego użytkownika.

Istnieje kilka sposobów ustawienia hasła użytkownika. Opiszemy tutaj dwa z nich: za pomocą programu *mysqladmin* oraz poprzez bezpośrednią zmianę danych użytkownika w bazie danych.

Ustawienie hasła użytkownika *root* (administratora) za pomocą programu *mysqladmin* wygląda następująco:

```
mysqladmin --user=root password nowe-hasło
```

Na przykład:

```
[rick@gw1 bmysql]$ mysqladmin --user=root password secret
[rick@gw1 bmysql]$
```

Po ustawieniu hasła można je zmieniać za pomocą programu *mysqladmin*, ale w takim przypadku należy podać poprzednie hasło za pomocą opcji `--password=stare_hasło`.

Sposób tworzenia domyślnych użytkowników powoduje, że dla każdego użytkownika występują zwykle dwie pozycje — jedna do logowania lokalnego oraz druga do logowania przez sieć. Program ustawia hasło dla wykorzystywanej metody logowania, co może spowodować, że baza danych nie będzie zabezpieczona tak dobrze, jak się tego mogłeś spodziewać. Z tego powodu zalecamy wykorzystanie innej metody zmiany hasła, która wymaga zalogowania się do bazy danych.

Jeżeli nie ustawiałeś jeszcze hasła użytkownika *root*, możesz zalogować się jako administrator w następujący sposób:

```
[rick@gw1 rick]$ mysql --user=root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 4.0.1-alpha

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

mysql>
```

Jeżeli skorzystałeś już z programu *mysqladmin* do ustawienia hasła, musisz podać hasło, dodając parametr `--password=hasło`, lub lepiej (nikt nie będzie mógł wtedy podejrzeć hasła) parametr `-p`, co spowoduje, że program *mysql* zapyta o hasło:

```
[rick@gw1 rick]$ mysql --user=root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 4.0.1-alpha

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

mysql>
```

Ponieważ jesteśmy podłączeni do serwera bazy danych jako użytkownik *root*, mamy uprawnienia do wykonania dowolnego polecenia we wszystkich bazach danych. Wybieramy bazę danych *mysql*, która zawiera tabele systemowe, w tym tabele wykorzystywane przy sprawdzaniu uprawnień użytkowników:

```
mysql> use mysql
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
mysql>
```

Teraz zmieniamy hasło dla użytkownika *root*, poprzez zmianę wartości w tabeli *user*:

```
mysql> UPDATE user SET password=PASSWORD('secret-password')
-> WHERE user='root';
```

```
Query OK, 2 rows affected (0.02 sec)
Rows matched: 2 Changed: 2 Warnings: 0
```

```
mysql>
```

Jest to zwykła instrukcja UPDATE, wykorzystująca funkcję PASSWORD(). Instrukcja ta wymaga podania nowego hasła (ciągu znaków) i zwraca je w postaci zaszyfrowanej. Zwróć uwagę na to, że zmienione zostały dwa wiersze; powód tego przedstawimy za chwilę. Zaszyfrowane hasło jest zapisywane w tabeli *user*. Korzystając z tej metody możemy ustawić hasło dla dowolnego użytkownika, ponieważ administrator posiada wszystkie prawa do tej tabeli.

Aby wprowadzone zmiany zaczęły obowiązywać, musimy wykonać instrukcję FLUSH PRIVILEGES:

```
mysql> FLUSH PRIVILEGES;
```

```
Query OK, 0 rows affected (0.09 sec)
```

```
mysql>
```

Instrukcja ta powoduje, że MySQL ponownie wczytuje uprawnienia użytkowników do wewnętrznych buforów.

Popatrzmy na fragment tabeli *user*:

```
mysql> SELECT user, host, password FROM user;
```

```
+-----+-----+-----+
| user | host      | password      |
+-----+-----+-----+
| root | localhost | 15e8e3984da94ac2 |
| root | gw1       | 15e8e3984da94ac2 |
|      | localhost |                |
|      | gw1       |                |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

```
mysql>
```

Jak widać, użytkownik *root* posiada ustawione hasło, ale występują dla tego użytkownika dwie pozycje. Obie zawierają to samo zaszyfrowane hasło (zignoruj na razie pozycje bez nazwy użytkownika, omówimy je nieco później). Powodem powstania podwójnych wpisów jest praca w sieci. Hasło wykorzystywane do połączenia się z MySQL może być inne dla połączeń realizowanych z lokalnego komputera i inne dla komputera lub grupy komputerów w sieci. Nasz komputer posiada nazwę *gw1*, ale podczas instalacji serwera MySQL dodany

został wpis dla komputera *localhost* (nazwa lokalnego połączenia). W niektórych instalacjach zamiast *localhost* może znajdować się tu adres IP *127.0.0.1*. Możliwe jest stworzenie kont użytkowników, którzy posiadają prawo do zalogowania się do bazy danych z lokalnego komputera, ale nie mogą się zalogować z innych komputerów podłączonych do sieci.

Dobrym pomysłem jest pozwolenie na logowanie się użytkownika *root* jedynie z komputera z serwerem MySQL. Zmniejsza to ryzyko zdobycia dostępu do serwera przez nieautoryzowanych użytkowników.

Przywracanie hasła administratora

Może zdarzyć się sytuacja, że konto administratora zostanie usunięte z serwera — albo z powodu uszkodzenia danych, albo przez błąd człowieka.

Na szczęście istnieje kilka sposobów wyjścia z tej sytuacji. Jeżeli jesteś nadal zalogowany jako *root*, problem można łatwo naprawić. Przełącz się do bazy danych *mysql* i wstaw nowy wiersz dla użytkownika *root*, komputera *localhost*, bez hasła i ze znakami *Y* w pozostałych kolumnach. Następnie ustaw hasło administratora w przedstawiony wcześniej sposób:

```
UPDATE user SET password=PASSWORD('secret-password')WHERE user='root'
```

Jeżeli nie jesteś zalogowany do bazy, istnieje kilka sposobów na naprawę sytuacji. Na początek musisz zatrzymać pracę serwera. W systemie Windows możesz do zatrzymania serwera użyć programu *WinMySQLAdmin*; możesz także zatrzymać usługę *mysql* ręcznie, korzystając z konta użytkownika o prawach administratora. Również w systemie Linux do uruchomienia skryptu zatrzymującego serwer musisz posiadać prawa administratora.

Najlepszą metodą przywracania konta administratora jest ręczne uruchomienie procesu serwera ze specjalnym parametrem `--skip-grant-tables`. Powoduje to, że serwer ignoruje tabele uprawnień i pozwala na zalogowanie się jako użytkownik *root* bez konieczności podawania hasła. Dodatkowo musisz podać inne parametry, takie jak położenie katalogu *data*. Na przykład po zainstalowaniu MySQL w katalogu */usr/local/mysql* w systemie Linux powinieneś wydać następujące polecenie:

```
/usr/local/mysql/bin/mysqld_safe --skip-grant-tables --datadir=/usr/local/mysql/var  
--pid-file=/usr/local/mysql/var/gw1.pid
```

W systemie Windows można uruchomić serwer z wieszka poleceń w następujący sposób:

```
mysqld-nt --standalone --skip-grant-tables
```

Jeżeli w systemie Linux instalowałeś serwer korzystając z plików źródłowych, istnieje jeszcze jeden sposób przywrócenia konta użytkownika *root* (choć nie jest on zalecany). W czasie instalacji musiałeś wykonać skrypt o nazwie *mysql_install_db*. Skrypt ten tworzy tabele uprawnień, a jego powtórne wykonanie spowoduje utworzenie domyślnych tabel. Powoduje to jednak utratę wszystkich kont użytkowników utworzonych w trakcie pracy z bazą danych.

Jeżeli jest to tylko możliwe, do odtwarzania konta administratora zalecamy korzystanie z opcji `--skip-grant-tables`.

Możemy teraz uruchomić program *mysql* w zwykły sposób, podłączyć się do serwera bazy danych jako *root* bez konieczności podawania hasła i ponownie wykonać instrukcje, które odtworzą konto administratora. Następnie należy zatrzymać serwer i uruchomić go już w normalnym trybie, po czym w opisywany wcześniej sposób ustawić hasło użytkownika *root*.

Usuwanie domyślnego użytkownika

Bieżąca wersja MySQL instaluje domyślnego użytkownika, który może podłączyć się do serwera z lokalnego komputera i korzystać z baz danych, których nazwy zaczynają się od *test*. Jest to wygodne w przypadku, gdy pierwszy raz instalujesz MySQL i chcesz sprawdzić, czy inni niż *root* użytkownicy mogą podłączyć się do bazy. Zawsze istnieje jednak ryzyko, że na serwerze produkcyjnym pozostanie aktywne konto domyślnego użytkownika, utworzone w czasie instalacji. Z tego powodu sugerujemy usunięcie takiego konta natychmiast po stwierdzeniu, że baza danych działa prawidłowo. Aby to zrobić, zaloguj się do bazy jako *root* i uważnie wykonaj następujące instrukcje:

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
mysql> DELETE FROM user WHERE user = '';
Query OK, 2 rows affected (0.00 sec)
```

Jeżeli pomylisz się w poleceniu DELETE, możesz usunąć z bazy danych swoje konto użytkownika. Sprawdź, czy usunięte zostały tylko dwa wiersze. Jeżeli przypadkowo usunąłeś własne konto, poszukaj w dalszej części tego rozdziału przykładu tworzenia nowego konta administratora i wykonaj podany w nim kod.

Aby upewnić się, że wszystko zostało wykonane prawidłowo, sprawdźmy, czy w tabeli *user* znajduje się użytkownik *root*, który może się logować z komputera *localhost*:

```
mysql> SELECT user, host FROM user WHERE user = 'root';
+-----+-----+
| user | host      |
+-----+-----+
| root | gw1       |
| root | localhost |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

W wyniku tego zapytania powinien znajdować się co najmniej jeden wiersz zawierający wpisy *root* i *localhost*. Jeżeli wszystko zostało wykonane prawidłowo, możemy zażądać przeładowania uprawnień przez serwer MySQL:

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.09 sec)

mysql>
```

Konto domyślnego użytkownika zostało usunięte.

Zarządzanie uprawnieniami użytkowników

Po zabezpieczeniu konta *root* możemy zająć się tworzeniem kont zwykłych użytkowników, nadawaniem uprawnień użytkownikom oraz — gdy to konieczne — usuwaniem ich kont.

Tworzenie kont użytkowników

Starsze wersje MySQL posiadały program *mysqlaccess*, który był przeznaczony do tworzenia kont użytkowników w bazie danych. Jest on nadal dostępny, ale zalecamy korzystanie z nowego polecenia GRANT, które zostało wprowadzone w wersji 3.2.1. Omówimy tu tworzenie kont użytkowników za pomocą polecenia GRANT dostępnego w wierszu poleceń *mysql*. Istnieją niewielkie różnice w działaniu polecenia GRANT w wersjach 3.2.1 i 4.0 MySQL. W MySQL 3.2.1 należało jawnie nadawać użytkownikom uprawnienia do dostępu do bazy danych (nadanie prawa dostępu do tabeli nie powodowało automatycznego nadania prawa dostępu do bazy danych). W wersji 4.0 nadanie prawa dostępu do co najmniej jednej tabeli w bazie danych powoduje umożliwienie dostępu do bazy danych. Instrukcja GRANT ma dosyć prostą składnię, ale posiada wiele możliwych opcji:

```
GRANT prawo [(lista kolumn)] ON baza_lub_tabela TO użytkownik
  [IDENTIFIED BY 'hasło'] [WITH GRANT OPTION]
```

Zapoznamy się teraz z opcjami instrukcji GRANT i ich praktycznym zastosowaniem. Popatrzymy w tym celu na listę dostępnych uprawnień (patrz górna tabela na następnej stronie); jest ona dosyć długa:

Kolejną częścią instrukcji GRANT jest opcjonalna lista kolumn, która pozwala na określenie, do których kolumn będą stosowane nadane uprawnienia. Jest to użyteczne jedynie w przypadku, gdy nadawane są uprawnienia do konkretnej tabeli; pozwala to jednak na bardzo precyzyjne określanie uprawnień do różnych części bazy danych. W dalszej części rozdziału pokażemy przykład praktycznego wykorzystania instrukcji GRANT z listą kolumn.

Następną częścią polecenia jest wyrażenie *baza_lub_tabela*. Pozwala ono określić jeden z trzech poziomów dostępu, przy wykorzystaniu składni *baza_danych.tabela*, gdzie znak *** służy jako znak specjalny. Przykład przedstawiono w dolnej tabeli na następnej stronie.

Kolejną częścią instrukcji jest nazwa użytkownika. Jak wcześniej wspomnieliśmy, możemy nadać różne uprawnienia dla użytkownika, w zależności od komputera, z którego podłącza się do bazy danych. Realizujemy to, korzystając z tej właśnie części instrukcji GRANT oraz odpowiednio wypełniając pole *host* w tabeli *user* znajdującej się w bazie danych *mysql*.

Nazwa użytkownika składa się z trzech części — nazwy podawanej przy logowaniu się do bazy, symbolu @ oraz wyrażenia określającego komputer. Jeżeli chcesz nadać użytkownikowi uprawnienia obowiązujące jedynie w przypadku logowania realizowanego na komputerze, na którym działa proces serwera, podaną nazwą komputera musi być *localhost*. Jeżeli chcesz nadać uprawnienia dla wszystkich komputerów, powinieneś użyć symbolu *'%'* (wraz z apostrofami). Trzeba pamiętać o podaniu nazwy komputera w pojedynczych apostrofach, ponieważ bez nich uprawnienia nie zostaną nadane.

| Słowo kluczowe | Znaczenie |
|----------------|--|
| ALL | Nadaje użytkownikowi wszystkie wymienione dalej uprawnienia |
| ALL PRIVILEGES | Takie samo jak dla ALL |
| ALTER | Pozwala na wykonanie polecenia ALTER tabela |
| CREATE | Pozwala na tworzenie baz danych i tabel |
| DELETE | Pozwala na usuwanie danych z tabel |
| DROP | Pozwala na usuwanie baz danych i tabel |
| FILE | Pozwala na dostęp do plików zapisanych na serwerze |
| INDEX | Pozwala na zarządzanie indeksami |
| INSERT | Pozwala na wstawianie danych do tabel |
| PROCESS | Pozwala na przeglądanie informacji o procesie serwera |
| REFERENCES | Jest to zarezerwowane słowo kluczowe; na razie nie ma przypisanego żadnego działania |
| RELOAD | Pozwala na powtórne załadowanie informacji z tabel uprawnień do serwera |
| SELECT | Pozwala na pobieranie danych z tabel |
| SHUTDOWN | Pozwala na zatrzymanie bazy danych |
| UPDATE | Pozwala na zmianę informacji w tabelach |
| USAGE | Pozwala na utworzenie konta użytkownika bez żadnych uprawnień |

| Poziom dostępu | Znaczenie |
|------------------------------|---|
| Nazwa bazy danych lub tabeli | Zasięg nadawanego uprawnienia |
| *.* | Wszystkie tabele we wszystkich bazach danych, w tym prawo do tworzenia nowych baz danych |
| bmsimple.* | Wszystkie tabele w bazie <i>bmsimple</i> , w tym prawo do utworzenia bazy danych o podanej nazwie |
| bmsimple.customer | Tabela <i>customer</i> w bazie danych <i>bmsimple</i> |

Można również określić dowolny komputer z domeny; na przykład aby nadać uprawnienia użytkownikowi *Deborah* z dowolnego komputera w domenie *docbox.co.uk*, nazwę użytkownika określamy jako *deborah%docbox.co.uk*. Można również określać zakres komputerów, podając adres IP podsieci — na przykład *deborah@192.168.100.%* — co umożliwi logowanie się z dowolnego komputera z podsieci klasy C o adresach rozpoczynających się od 192.168.100.

Następna część instrukcji — IDENTIFIED BY 'hasło' — jest opcjonalna i używana jedynie w czasie tworzenia konta nowego użytkownika; pozwala ona określić przypisane doń hasło. Jeżeli nadajesz prawa dla nieistniejącego użytkownika i nie podasz opcji IDENTIFIED BY, utworzony zostanie nowy użytkownik bez hasła, co nie jest właściwe ze względu na bezpieczeństwo serwera.

Ostatnią częścią instrukcji jest `WITH GRANT OPTION`. Część ta jest również opcjonalna. Jeżeli zostanie podana, nowy użytkownik będzie mógł przekazywać innym użytkownikom uprawnienia, które posiada. Może być to przydatne w przypadku dużej ilości użytkowników, kiedy administrator przekazuje część odpowiedzialności za zarządzanie określonymi bazami danych innym użytkownikom. Operacja taka musi być przemyślana.

Działanie instrukcji `GRANT` kumuluje się, więc można wykonać tę instrukcję kilkakrotnie, nadając użytkownikowi kolejne uprawnienia. Pamiętaj, że aby zadziałały nowe uprawnienia, należy wykonać instrukcję `FLUSH PRIVILEGES`.

Instrukcja GRANT

Utwórzmy konto użytkownika *Deborah*, który posiada uprawnienia do połączenia się z komputera z domeny `'docbox.co.uk'` i dostęp tylko do tabel w bazie danych *bmsimple*:

```
mysql> GRANT ALL ON bmsimple.* TO deborah@'% .docbox.co.uk '
-> IDENTIFIED BY 'secret';
```

```
Query OK, 0 rows affected (0.65 sec)
```

```
mysql>
```

Zauważ, że określenie komputera, z którego może się logować użytkownik zostało ujęte w apostrofy, ponieważ zawiera znak specjalny `%`, używany do wybrania wszystkich komputerów z domeny *docbox.co.uk*. Ponieważ nadaliśmy użytkownikowi *Deborah* wszystkie uprawnienia, może on wykonywać wszystkie operacje na tabelach w bazie danych — w tym również tworzyć nowe tabele. Jeżeli chcielibyśmy dać mu możliwość korzystania z dowolnego komputera w sieci, powinniśmy skorzystać z instrukcji:

```
GRANT ALL ON bmsimple.* TO deborah@'% ' IDENTIFIED BY 'secret'
```

Jeżeli chcesz pozwolić użytkownikowi *Deborah* na logowanie się z komputera, na którym działa serwer bazy danych, powinieneś wykonać osobną instrukcję `GRANT`, podając przy nazwie użytkownika nazwę lokalnego komputera, czyli *localhost*:

```
mysql> GRANT ALL ON bmsimple.* TO deborah@localhost IDENTIFIED BY 'secret';
```

```
Query OK, 0 rows affected (0.04 sec)
```

```
mysql>
```

Po tych operacjach w tabeli *mysql.user* pojawiły się dwa wiersze:

```
mysql> SELECT user, host, password FROM user WHERE user='deborah';
```

```
+-----+-----+-----+
| user  | host          | password          |
+-----+-----+-----+
| deborah | localhost     | 428567f408994404 |
| deborah | %.docbox.co.uk | 428567f408994404 |
+-----+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql>
```

Kolejnym przykładem będzie utworzenie konta użytkownika *manager1*, który będzie miał bardzo ograniczone uprawnienia. Założmy, że użytkownik *manager1* będzie mógł pobierać dane z niektórych kolumn tabeli *customer*, zapisanej w bazie danych *bmsimple*. Dodatkowo będzie mógł się zalogować tylko z komputera, na którym znajduje się baza danych. Aby zrealizować te założenia, skorzystamy z prawa `SELECT`, które umożliwia jedynie odczytywanie danych. Dodatkowo wymienimy nazwy kolumn, do których nadamy te uprawnienia. Korzystając z tych ograniczonych uprawnień użytkownik *manager1* może się podłączyć do bazy jako lokalny użytkownik.

```
mysql> GRANT SELECT (title, fname, lname) on bmsimple.customer TO manager1@localhost
IDENTIFIED BY 'foo';
```

```
Query OK, 0 rows affected (0.71 sec)
```

```
mysql>
```

Sprawdźmy, czy nadane uprawnienia działają prawidłowo:

```
[rick@gw1 rick]$ mysql --user=manager -p
```

```
Enter password: ***
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

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

```
mysql> use mysql;
```

```
ERROR 1044: Access denied for user: 'manager1@localhost' to database 'mysql'
```

```
mysql> use bmsimple;
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
```

```
mysql>
```

Użytkownik *manager1* może korzystać jedynie z bazy danych *bmsimple*. Nie ma natomiast praw do tabeli *item*, na której będzie pracował:

```
mysql> SELECT * FROM item;
```

```
ERROR 1142: select command denied to user: 'manager1@localhost' for table 'item'
mysql>
```

Spróbujmy teraz wykonać zapytanie `SELECT` na tabeli *customer*. Zwróć uwagę na to, że w tym przypadku zwrócony jest nieco inny komunikat błędu, ponieważ tym razem użytkownik może korzystać z tabeli, ale użyty w poleceniu znak `*` powoduje próbę dostępu do kolumn, do których nie posiada dostępu:

```
mysql> SELECT * FROM customer;
```

```
ERROR 1143: select command denied to user: 'manager1@localhost' for column
'customer_id' in table 'customer'
mysql>
```

Jak widać, udało się nam nadać uprawnienia do poszczególnych pól tabeli. Wypróbujmy działanie innych kolumn z tabeli *customer*:

```
mysql> SELECT title, fname, lname FROM customer WHERE lname = 'Jones';
+-----+-----+-----+
| title | fname | lname |
+-----+-----+-----+
| Mr    | Dave  | Jones |
+-----+-----+-----+
1 row in set (0.02 sec)
```

```
mysql>
```

Zwróć uwagę na to, że nie możemy nawet utworzyć warunku (`WHERE customer_id = 7`) na kolumnie, do której nie posiadamy uprawnień do odczytu:

```
mysql> SELECT title, fname, lname FROM customer WHERE customer_id = 7;
ERROR 1143: select command denied to user: 'manager1@localhost' for column
'customer_id' in table 'customer'
mysql>
```

Na koniec upewnimy się, że nie posiadamy uprawnień do dodawania nowych danych do tabeli *customer*:

```
mysql> INSERT INTO customer VALUES(NULL, 'Mr', 'Matthew', 'Harvey', '1 the drive',
'Slowtown', 'ST1 4HG', '0121-232343');
ERROR 1142: insert command denied to user: 'manager1@localhost' for table 'customer'
mysql>
```

Zakończmy nasze eksperymenty utworzeniem konta użytkownika, który posiadać będzie wszystkie uprawnienia użytkownika *root*, w tym prawo do nadawania uprawnień innym użytkownikom. Umożliwimy mu ponadto logowanie się do serwera tylko z lokalnego komputera.

Podłącz się do bazy danych jako *root* i wykonaj następującą instrukcję:

```
mysql> GRANT ALL ON *.* TO admin247@localhost IDENTIFIED BY 'password2' WITH GRANT
OPTION;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>
```

Instrukcja ta tworzy nowe konto administratora, o nazwie *admin247*; posiada on te same uprawnienia, co właściwy użytkownik *root*. Może być to niebezpieczne, ale niezbędne w sytuacji, gdy usunąłś konto użytkownika *root*. Sytuacja taka utrudnia zadanie osobie, która usiłuje włamać się do systemu, ponieważ teraz zamiast próbować odgadnąć hasło użytkownika *root*, będzie musiała odgadnąć zarówno nazwę użytkownika administrującego systemem, jak i jego hasło.

Jak to działa?

Instrukcja `GRANT` zmienia dane w różnych tabelach uprawnień zawartych w bazie danych *mysql*. Szczegółowy opis tych danych wykracza poza ramy tej książki, jednak odczytywanie nazwy użytkownika, komputera i hasła wydaje się być przydatne. Można w ten sposób stworzyć czytelną listę użytkowników oraz komputerów, z których mogą się logować.

Odbieranie uprawnień

Odbieranie uprawnień wykonywane jest za pomocą instrukcji `REVOKE`, której składnia jest bardzo podobna do składni instrukcji `GRANT`:

```
REVOKE prawo [(lista_kolumn)] ON baza_lub_tabela FROM użytkownik
```

Wszystkie opcje są identyczne, jak w przypadku instrukcji `GRANT`, można więc jawnie podać listę kolumn i odebrać uprawnienia na poziomie bazy danych lub na poziomie tabeli. Po wykonaniu instrukcji `REVOKE` należy użyć instrukcji:

```
FLUSH PRIVILEGES;
```

Powoduje to ponowne odczytanie tabel z uprawnieniami, dzięki czemu widoczne są efekty działania polecenia.

Można wyświetlić wszystkie uprawnienia użytkownika. W tym celu korzystamy z polecenia:

```
SHOW GRANTS FOR użytkownik;
```

na przykład:

```
mysql> show grants for rick;
```

```
+-----+-----+-----+-----+
| Grants for rick@%                               |
+-----+-----+-----+-----+
| GRANT USAGE ON *.* TO 'rick'@'%' IDENTIFIED BY PASSWORD '5c96ea97620d605c' |
| GRANT ALL PRIVILEGES ON bmsimple.* TO 'rick'@'%' |
+-----+-----+-----+-----+
```

2 rows in set (0.01 sec)

```
mysql>
```

Aby instrukcja ta mogła zadziałać, użytkownik musi mieć nadane uprawnienia.

Odbieranie uprawnień

Rozpoczniemy od utworzenia konta użytkownika *manager2*, z nadanymi uprawnieniami do tabeli *item*:

```
mysql> GRANT SELECT ON bmsimple.item TO manager2@localhost;
```

```
Query OK, 0 rows affected (0.05 sec)
```

```
mysql>
```

Teraz zalogujemy się jako użytkownik *manager2* i sprawdzimy, czy możemy odczytać dane z tabeli *item*:

```
mysql> SELECT * FROM item;
```

```
+-----+-----+-----+-----+
| item_id | description | cost_price | sell_price |
+-----+-----+-----+-----+
|      1 | Wood Puzzle |      15.23 |      21.95 |
|      2 | Rubik Cube  |       7.45 |      11.49 |
```

```

|      3 | Linux CD      |      1.99 |      2.49 |
|      4 | Tissues      |      2.11 |      3.99 |
|      5 | Picture Frame |      7.54 |      9.95 |
|      6 | Fan Small    |      9.23 |     15.75 |
|      7 | Fan Large    |     13.36 |     19.95 |
|      8 | Toothbrush   |      0.75 |      1.45 |
|      9 | Roman Coin   |      2.34 |      2.45 |
|     10 | Carrier Bag  |      0.01 |      0.00 |
|     11 | Speakers     |     19.73 |     25.32 |
+-----+-----+-----+-----+
11 rows in set (0.10 sec)

```

```
mysql>
```

Zaloguj się ponownie jako *root* i odbierz uprawnienia:

```
mysql> REVOKE SELECT ON bmsimple.item FROM manager2@localhost;
Query OK, 0 rows affected (0.09 sec)
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.05 sec)
```

```
mysql>
```

Spróbujmy — znów jako *manager2* — odczytać dane z tabeli *item*:

```
mysql> SELECT * FROM item;
ERROR 1142: select command denied to user: 'manager2@localhost' for table 'item'
```

```
mysql>
```

Jak to działa?

Instrukcja `REVOKE` usuwa z tabel bazy danych *mysql* informacje na temat nadanych uprawnień.

Zawsze sprawdzaj, czy udało się odebrać uprawnienia. W czasie, kiedy powstawała ta książka, implementacje `GRANT` i `REVOKE` były jeszcze nowe. Odbieranie uprawnień domyślnemu użytkownikowi stworzonemu przez MySQL może również działać w nieoczekiwany sposób.

Jeżeli chcesz odebrać jakiemuś użytkownikowi wszystkie uprawnienia, możesz ręcznie usunąć jego konto z tabeli *mysql.user*; operacja daje efekt identyczny z usunięciem go z bazy danych:

```
DELETE FROM user WHERE user='nazwa_uzytkownika_do_usuniecie';
```

Należy rozważyć wykonywać taką operację, ponieważ nie usuwa ona uprawnień nadanych użytkownikowi, lecz nie pozwala mu na podłączenie się do bazy danych. Uprawnienia, jakie posiadał, pozostają niezmienione i jeżeli kiedykolwiek ponownie utworzysz konto tego użytkownika, może się okazać, że posiada on więcej uprawnień niż powinien.

Plik śladu serwera

Czasami występują problemy z pracą serwera. Aby dowiedzieć się, co stanowi przyczynę problemu, musimy przejrzeć plik śladu serwera. Zwykle MySQL tworzy plik śladu o nazwie `<nazwa komputera>.err`, który zawiera informacje o uruchomieniu i zatrzymaniu pracy. W przypadku instalacji z plików źródłowych dla systemu Linux, na naszym komputerze `gw1` ścieżka dostępu do tego pliku to `/usr/local/mysql/var/gw1.err`, zaś sam plik zawiera następujące wpisy:

```
020522 23:47:49 mysqld started
020522 23:47:50 InnoDB started
/usr/local/mysql/libexec/mysqld: ready for connections
020524 0:21:20 /usr/local/mysql/libexec/mysqld: Normal shutdown

020524 0:21:21 InnoDB: Starting shutdown...
020524 0:21:22 InnoDB: Shutdown completed
020524 0:21:23 /usr/local/mysql/libexec/mysqld: Shutdown Complete

020524 0:21:23 MySQL ended

030522 19:47:49 mysqld started
030522 19:47:50 InnoDB started
/usr/local/mysql/libexec/mysqld: ready for connections
```

W przypadku Windows wpisy te są niemal identyczne, ale plik ten zwykle nosi nazwę `mysql.err` i znajduje się w podkatalogu `data` katalogu, w którym zainstalowałeś MySQL.

Pliki konfiguracyjne

MySQL posiada kilka plików pozwalających na ustawienie opcji konfiguracji serwera. Pliki te umożliwiają również określenie domyślnych parametrów dla programu `mysqld` i innych narzędzi dostarczanych wraz z MySQL. Położenie i nazwy tych plików różnią się pomiędzy systemem Linux i Windows, ale dane zawarte w tych plikach i zbiór dostępnych opcji są w obu systemach niemal identyczne. Korzystając z plików konfiguracyjnych można ustawić opcje zarówno dla serwera, jak i dla poszczególnych użytkowników (w systemie Linux).

Ustawienia serwera

Ustawienia konfiguracji znajdują się w następujących plikach:

| Nazwa pliku | Zawartość |
|---|---|
| <code>/etc/my.cnf</code> | Ustawienia konfiguracji w systemie Linux (UNIX) |
| <code>c:\my.cnf</code> | Globalny plik konfiguracyjny w systemie Windows |
| <code><katalog Windows>\my.ini</code> | Dodatkowy lub alternatywny plik konfiguracyjny w systemie Windows |

Należy pamiętać, że w systemie Windows 2000 (i późniejszych wersjach) rozszerzenie *.cnf* jest skojarzone z programem *Speeddial* i z tego powodu może być ukrywane przez system. W Windows, jeżeli korzystasz z obu plików konfiguracyjnych — *c:\my.cnf* i *my.ini* w katalogu Windows, oba pliki będą odczytane, po czym pozycje z pliku *my.cnf* zostaną zastąpione przez odpowiednie opcje z pliku *my.ini*. W programie *WinMySQLAdmin* dostępna jest opcja edycji pliku *my.ini*.

W przypadku Windows można korzystać z dowolnego z tych dwóch plików, poza przypadkiem, gdy zainstalowałeś MySQL w innym katalogu niż *c:\mysql*. Należy wtedy utworzyć plik *c:\my.cnf*, w którym skonfigurowane zostaną niektóre krytyczne dla pracy programu ścieżki.

Pierwotne pliki konfiguracyjne

Po zainstalowaniu MySQL tworzony jest przykładowy plik konfiguracyjny *<Katalog instalacji MySQL>\my-example.cnf*, a w przypadku samodzielnej kompilacji MySQL w katalogu *<katalog ze źródłami>/support_files* znajduje się kilka wariantów plików konfiguracyjnych. Pliki te są przykładami konfiguracji małego, średniego i dużego serwera. Jeżeli nie jesteś pewien, którego pliku powinieneś użyć, sugerujemy skorzystanie z pliku *my-medium.cnf*, który należy skopiować do */etc/my.cnf*.

Format pliku konfiguracyjnego jest podobny do formatu pliku *windows.ini*:

| Opcja | Znaczenie |
|--|--------------------------------------|
| # | Rozpoczyna wiersz komentarza |
| [grupa] | Rozpoczyna grupę opcji |
| Opcja | Ustawia opcję |
| opcja=wartość | Ustawia opcję na określoną wartość |
| set-variable=nazwa_zmiennej = nadawana_wartość | Ustawia zmienną na określoną wartość |

Pierwsza sekcja jest przeznaczona dla programów klienckich i wygląda następująco:

```
[client]
#password=my_password
port=3306
```

Wpisany jest tu domyślny numer portu, który jest używany przez program kliencki do podłączania się do serwera. Nie zalecamy zmiany tego numeru! Można również ustawić domyślne hasło dla programów klienckich, ale nie jest to rozwiązanie zbyt bezpieczne. Zalecamy zatem pozostawienie w tym wierszu komentarza.

Kolejna sekcja jest zwykle konfiguracją serwera i jest dużo bardziej skomplikowana. Jest to grupa *mysqld*:

```
# The MySQL server
[mysqld]
port=3306
#socket=MySQL
```

```

skip-locking
default-character-set=latin1
set-variable = key_buffer=16M
set-variable = max_allowed_packet=1M
set-variable = thread_stack=128K
set-variable = flush_time=1800

```

Na początku grupy *mysqld* ustawiany jest numer portu, na którym nasłuchuje serwer, domyślny zestaw znaków i kilka parametrów konfiguracyjnych. Nie zalecamy zmiany tych ustawień.

Kolejną częścią pliku są ustawienia dotyczące położenia instalacji:

```

# Uncomment the following row if you move the MySQL distribution to another
# location
basedir = e:/mysql/
datadir = e:/mysql/data/

```

Sekcja ta jest niezmiernie ważna w Windows w przypadku, gdy zainstalowałeś serwer w innym katalogu niż domyślny *c:\mysql*. W przykładowym pliku wiersze *basedir* i *datadir* są wyłączone za pomocą komentarza, a w naszym przykładzie pokazaliśmy sposób ich ustawiania w przypadku instalacji serwera w katalogu *e:\mysql*.

Kolejna sekcja pliku to sekcja konfiguracji tabel InnoDB, którą omówimy dokładniej w dalszej części rozdziału. Jeżeli nie chcesz używać tabel typu InnoDB lub korzystasz z minimalnej konfiguracji, możesz usunąć komentarz z wiersza *skip-innodb*, co znacznie zmniejszy ilość pamięci zajmowanej przez serwer:

```

# Uncomment the following row if you are using a Max server and you don't want the
# InnoDB tables
#skip-innodb

```

W pliku tym zawarty jest jeszcze kilka innych sekcji, ale nie powinieneś ich zmieniać.

Ustawienia klienta

Ustawienia programów klienckich powinny się znajdować w pliku *<katalog Windows>/my.ini* w przypadku Windows i *my.cnf* w katalogu domowym w przypadku Linuxa. Pamiętaj, że w systemie Linux (i UNIX) nazwy plików rozpoczynające się od kropki są domyślnie ukryte. Można umieścić ustawienia klienta w globalnym pliku konfiguracyjnym, ale nie jest to najlepsze rozwiązanie. Program *mysql* i inne programy dostępne w instalacji MySQL korzystają z parametrów znajdujących się w sekcji *[client]* plików konfiguracyjnych. Parametry te są odczytywane po parametrach pliku *c:\my.cnf* lub */etc/my.cnf*, więc można przesłonić globalne opcje, podając w lokalnym pliku inne wartości.

Specyfikacja lokalnego pliku konfiguracyjnego jest identyczna ze specyfikacją pliku globalnego. Inny jest tylko typ ustawień, jakie mogą być w nim zawarte. Najbardziej użyteczne wydaje się przechowywanie domyślnych parametrów połączenia z serwerem. Przedstawimy teraz sekcję *client*, która powoduje automatyczne podłączenie się do serwera MySQL uruchomionego na komputerze *gw1*. Zwykle korzystam w tym celu z użytkownika *rick* z hasłem *secret*, a następnie wybieram bazę danych *bmsimple*:

```
[client]
user=rick
password
database=bmsimple
host=gw1
```

Zwróć uwagę na wiersz, w którym wpisane jest słowo *password* bez znaku = (znaku równości) ani wartości. Nie chcemy przechowywać hasła w pliku (choć moglibyśmy tu wpisać *password=secret*); w ten sposób wskazujemy programowi *mysql*, że powinien zapytać o hasło. Jeżeli nie wpiszesz wiersza zawierającego *password*, otrzymamy następujący wynik:

```
c:\mysql\bin>mysql
ERROR 1045: Access denied for user: 'rick@gw1' (Using password: NO)

c:\mysql\bin>
```

Ponieważ *mysql* założył, że hasło nie jest wymagane, otrzymaliśmy komunikat o wystąpieniu błędu. Jeżeli dodamy wiersz *password*, *mysql* spyta o hasło.

Program klienta „widzi” wartości zapisane w tej sekcji, tak jakby były one podane w wierszu poleceń z prefiksem `--`, na przykład:

```
mysql --user=rick
```

Wyłączanie wartości domyślnych

Choć podawanie wartości domyślnych w pliku *my.ini* lub *~/my.cnf* jest bardzo wygodne, istnieją przypadki, gdy niezbędne jest tymczasowe zignorowanie podanych wartości domyślnych. Czasami może wystąpić problem, jeżeli w pliku konfiguracyjnym podamy rzadziej używaną opcję, na przykład `--xml` i uruchomimy program, na przykład *mysqldump*, który nie rozpoznaje takiej opcji. Sytuacja taka jest dosyć myląca, ponieważ uruchamiasz program bez parametrów, a otrzymujesz komunikat o błędzie oznajmiający, że podałeś nieprawidłowy parametr.

Dzieje się tak dlatego, że wszystkie programy *mysql...* odczytują parametry z sekcji *client* i jeżeli program nie zinterpretuje któregoś z parametrów, zwraca komunikat o błędzie. Aby rozwiązać ten problem, należy użyć parametru `--no-defaults`, który powoduje wyłączenie odczytu zmiennych konfiguracyjnych z pliku *my.ini* lub *~/my.cnf*.

Na przykład, pozostawiając niezmieniony plik *my.ini*, który zawiera domyślne połączenie z serwerem na komputerze z Linuksem, dzięki wyłączeniu domyślnych wartości możemy wywołać program *mysql*, aby połączył się z lokalnym serwerem:

```
c:\mysql\bin>mysql --no-defaults --user=rick --password
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 135 to server version: 4.0.1-alpha-max-nt

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

mysql>
```

Pliki InnoDB

We wcześniejszych rozdziałach tej książki wykorzystywaliśmy już typ tabel InnoDB, ponieważ posiada on obsługę transakcji i kluczy obcych, których to opcji nie ma w domyślnym typie tabel MyISAM.

Aby skorzystać z tabel typu InnoDB musisz sprawdzić, czy używany program serwera obsługuje ten typ. Musisz również utworzyć tabelę typu InnoDB, dodając do instrukcji CREATE TABLE klauzulę TYPE=InnoDB lub określając domyślny typ tabel.

Aby ustawić domyślny typ tabel na InnoDB, należy dodać do sekcji [mysqld] w pliku *c:\my.cnf* (w Windows) lub */etc/my.cnf* (w Linuksie) wiersz:

```
default-table-type=innodb
```

Spowoduje to, że od tego momentu wszystkie tworzone tabele będą miały typ InnoDB.

Musisz jeszcze sprawdzić dwie opcje w sekcji [mysqld]. Sprawdź, czy opcje basedir= i datadir= wskazują na istniejące katalogi. Jeżeli wpisy te są nieprawidłowe, proces serwera może nie uruchomić się prawidłowo.

Istnieje wiele parametrów konfiguracyjnych związanych z tabelami InnoDB, zawartych w sekcji [mysqld] pliku konfiguracyjnego, dla których można pozostawić domyślne wartości. Jeżeli posiadasz źródła dla systemu Linux, możesz przejrzeć ustawienia w kilku przykładowych plikach konfiguracyjnych, które mogą dostarczyć Ci informacji niezbędnych do precyzyjnej konfiguracji serwera. Opiszemy tutaj tylko najważniejsze opcje. Pełna ich lista, wraz z dokładnym opisem, znajduje się w podręczniku InnoDB, który można znaleźć pod adresem <http://www.innodb.com/ibman.html>. Domyślne ustawienia tabel InnoDB sprawdzają się bardzo dobrze, więc jeżeli nie jesteś pewien zmian, które wprowadzasz, pozostaw lepiej domyślne wartości opcji.

W tabeli na następnej stronie zamieszczamy krótkie omówienie opcji konfiguracji dla systemu średniej wielkości. Opcje te są zawarte w przykładowych plikach konfiguracyjnych systemu Linux. Instalacja dla Windows posiada własne, podobne do przedstawionych w tym miejscu, domyślne wartości zmiennych konfiguracji.

W czasie pierwszego uruchomienia serwera po zmianie parametrów konfiguracji tabel InnoDB można zauważyć opóźnienie w uruchomieniu, ponieważ tworzone są pliki danych.

Rozwiązywanie problemów

Zdarza się, że w czasie pracy bazy danych występują problemy, na przykład uszkodzenie danych. Jeżeli korzystasz z tabel typu InnoDB, serwer usiłuje automatycznie naprawiać te problemy — o ile jest to możliwe. Jeżeli korzystasz z tabel MyISAM, naprawianie uszkodzeń nie następuje automatycznie, ale można skorzystać z programu *myisamchk*, który umożliwi naprawienie tabel oraz dostarcza informacji na temat wewnętrznych struktur tabeli. Nie powinieneś uruchamiać tego programu, jeżeli nie występują problemy, chyba że użyjesz parametru *-d*, który powoduje wyświetlenie informacji o tabeli. Przed uruchomieniem *myisamchk* należy zatrzymać serwer bazy danych. W opcjach wywołania należy również podać ścieżki do plików, które trzeba naprawić.

| Opcja | Znaczenie |
|--|--|
| <code>innodb_data_file_path = ibdata1:400M</code> | Ustawia nazwę i rozmiar pliku, który będzie wykorzystywany przez InnoDB do przechowywania danych. Plik ten będzie przechowywany w katalogu określonym przez zmienną <code>innodb_data_home_dir</code> i będzie miał wielkość 400 MB. W InnoDB wszystkie tabele są przechowywane w pliku, który nie może być zwiększany; zamiast tego można dodawać kolejne pliki. Jeżeli możesz określić końcową wielkość danych, możesz spróbować podać wielkość pliku, w którym zmieszczą się wszystkie dane |
| <code>innodb_data_home_dir = /usr/local/mysql/var</code> | Ustawia ścieżkę do katalogu z plikami InnoDB. W Windows należy użyć standardowego formatu ścieżki np.: <code>innodb_data_home_dir = c:\ibdata</code> |
| <code>innodb_log_group_home_dir = /usr/local/mysql/var</code> | Ustawia ścieżkę do katalogu, w którym tabela InnoDB będzie przechowywała pliki śladu bazy danych. W Windows należy użyć formatu zgodnego z używanym w tym systemie |
| <code>innodb_log_arch_dir = /usr/local/mysql/var</code> | Ustawia ścieżkę do katalogu z archiwalnymi plikami śladu. Jak poprzednio, w Windows należy użyć formatu zgodnego z używanym w tym systemie |
| <code>set-variable = innodb_mirrored_log_groups=1</code> | Zmienna ta powinna mieć zawsze wartość 1 |
| <code>set-variable = innodb_log_files_in_group=3</code> | Ilość używanych plików śladu. Zaleca się użycie trzech plików |
| <code>set-variable = innodb_log_file_size=5M</code> | Rozmiar pliku śladu. Plik ten powinien mieć co najmniej 1 MB, ale im większy jest jego rozmiar, tym lepszą można osiągnąć wydajność |
| <code>set-variable = innodb_log_buffer_size=8M</code> | Ilość danych transakcji buforowanych przed ich zapisem na dysk. Wartość ta nie powinna przekraczać $(innodb_log_files_in_group * innodb_log_file_size)/2$ i jeżeli nie przewidujesz bardzo dużych i długo trwających transakcji, wystarczy nawet mniejsza wartość |
| <code>innodb_flush_log_at_trx_commit=1</code> | Zmienna powinna mieć wartość 1 |
| <code>innodb_log_archive=0</code> | Zmienna powinna mieć wartość 0 |
| <code>set-variable = innodb_buffer_pool_size=16M</code> | Określa ilość pamięci używanej do buforowania. Wartość ta nie powinna przekraczać 80% ilości zainstalowanej pamięci, nawet na komputerze przeznaczonym jedynie do pracy z MySQL |
| <code>set-variable = innodb_additional_mem_pool_size=2M</code> | Ilość pamięci przeznaczonej do przechowywania słownika danych. W przypadku bardzo skomplikowanych baz danych z dużą ilością tabel należy zwiększyć tę wartość |
| <code>set-variable = innodb_file_io_threads=4</code> | Zmienna powinna mieć wartość 4 |
| <code>set-variable = innodb_lock_wait_timeout=50</code> | Opóźnienie wykrywania zakleszczeń przez procedury InnoDB. Wartość ta nie powinna być zmieniana |

Przy standardowych ustawieniach program ten jest uruchamiany z katalogu z danymi bazy danych, który zwykle znajduje się w lokalizacji `<katalog MySQL>/data/<nazwa bazy danych>`. Najważniejszą jego opcją jest opcja `-r`, powodująca odtworzenie tabeli; trzeba także podać nazwę tabeli lub wpisać `*.MYI`, jeżeli chcemy naprawić wszystkie tabele. Nasz kolejny

przykład demonstruje działanie programu *myisamchk* w systemie Windows. Program wyświetla opis tabeli *customer* w bazie danych *bmsimple*:

```
C:\mysql\data\bmsimple>...\bin\myisamchk -d customer.MYI

MyISAM file:      customer.MYI
Record format:    Packed
Character set:    latin1 (8)
Data records:     15 Deleted blocks:      0
Recordlength:    195

table description:
Key Start Len Index Type
1  2    4  unique long

C:\mysql\data\bmsimple>
```

Istnieje również program *isamchk*, przeznaczony do tabel typu ISAM, które były poprzednim domyślnym typem tabel. Więcej informacji na ich temat możesz znaleźć w podręczniku MySQL.

Lepszą niż użycie *myisamchk* metodą jest wykorzystanie polecenia REPAIR TABLE, dostępnego w programie *mysql*. Polecenie to wymaga podania nazwy tabeli lub rozdzielonej przecinkami listy nazw tabel. Próbuje ono naprawić tabele w identyczny sposób, jak program *myisamchk*:

```
mysql> REPAIR TABLE customer, item;
+-----+-----+-----+-----+
| Table          | Op      | Msg_type | Msg_text |
+-----+-----+-----+-----+
| bmsimple.customer | repair | status   | OK       |
| bmsimple.item    | repair | status   | OK       |
+-----+-----+-----+-----+
2 rows in set (0.04 sec)

mysql>
```

Przeglądanie baz danych

Istnieje kilka sposobów uzyskania informacji na temat konfiguracji; większości z nich użyliśmy już w tej książce. Poza programem *mysqladmin* wszystkie są wewnętrznymi poleceniami programu *mysql* (patrz tabela na następnej stronie), więc aby ich użyć należy się wcześniej zalogować do bazy danych.

Kopie zapasowe

Jeżeli dane przechowywane na dyskach są ważne, należy tworzyć ich kopie zapasowe; identycznie należy postępować w przypadku danych zapisanych w bazie MySQL.

| Polecenie | Przeznaczenie |
|--|---|
| <code>mysqladmin ping</code> | Szybkie sprawdzenie, czy działa serwer bazy danych |
| <code>mysqladmin version</code> | Wyświetla typ serwera, czas pracy i inne informacje |
| <code>STATUS;</code> | Wyświetla nazwę bieżącego użytkownika, bazy danych i informacje na temat serwera |
| <code>SHOW DATABASES;</code> | Wyświetla listę baz danych na bieżącym serwerze |
| <code>SHOW TABLES FROM baza_danych;</code> | Wyświetla listę tabel w podanej bazie danych |
| <code>SHOW TABLE STATUS FROM baza_danych;</code> | Wyświetla informacje na temat tabel w bazie danych — w tym typ, ilość wierszy i datę utworzenia tabel |
| <code>SHOW TABLES;</code> | Wyświetla listę tabel w bieżącej bazie danych |
| <code>SHOW COLUMNS FROM nazwa_tabeli;</code> | Wyświetla informacje na temat podanej tabeli |
| <code>DESCRIBE nazwa_tabeli;</code> | Jak wyżej |
| <code>SHOW CREATE TABLE nazwa_tabeli;</code> | Wyświetla kod SQL potrzebny do utworzenia podanej tabeli |

W przypadku tabel typu MyISAM można tworzyć kopie zapasowe poprzez zatrzymanie bazy danych i skopiowanie wszystkich plików z katalogu bazy danych. W przypadku tabel InnoDB jest to trudniejsze, ale nadal możliwe. Aby prawidłowo zastosować tę metodę tworzenia kopii zapasowej, należy kolejno:

- zatrzymać serwer;
- skopiować pliki `<nazwa_bazy_danych>*.frm`;
- skopiować pliki danych InnoDB;
- skopiować pliki śladu InnoDB;
- uruchomić serwer.

Chociaż zastosowanie tej metody jest możliwe, zaś sama metoda jest stosunkowo wydajna, nie jest jednak zalecana. Dużo lepiej jest użyć programu *mysqldump*, który tworzy czytelniejszy plik wynikowy, a pliki powstałe za jego pomocą pozwalają na przenoszenie danych pomiędzy różnymi systemami, nawet pomiędzy Windows i Linuksem.

Program *mysqldump* wymaga podania w wierszu poleceń kilku argumentów; jako wynik tworzy plik z instrukcjami SQL, na podstawie których można odtworzyć bazę danych. Wspominaliśmy w poprzednich rozdziałach, że program *mysql* może wykonywać skrypty SQL za pomocą polecenia `SOURCE <nazwa_skryptu>` lub `\. <nazwa_skryptu>`. Należy sprawdzić, czy w danych zawartych w tabelach nie występują nieoczekiwane znaki. Może się zdarzyć, że skrypt wygenerowany przez *mysqldump* będzie wymagał zmian przed jego użyciem do odtworzenia bazy danych.

Składnia polecenia *mysqldump* jest następująca:

```
mysqldump [OPCJE] baza_danych | --all-databases [tabela]
```

Normalnie powinniśmy podać nazwę bazy danych i pozwolić, aby *mysqldump* zapisał wszystkie tabele, ale podając dodatkowo nazwy tabel można zapisać jedynie dane z wybranych tabel. Można również skorzystać z opcji `--all_databases`, która powoduje zapisanie w pliku wszystkich tabel z wszystkich baz danych. W wersjach MySQL starszych od 4.0 typ tabel nie był zapisywany przez program *mysqldump*, więc jeżeli ręcznie ustawiłeś typ tabeli na inny niż domyślny, informacja ta była utracona. W bieżących wersjach MySQL *mysqldump* zawsze generuje klauzulę `TYPE=` przy tworzeniu tabeli.

Ilość dostępnych opcji jest dosyć duża, wobec czego zamieścimy tylko główne. Pełna lista opcji znajduje się w podręczniku MySQL.

| Opcja | Znaczenie |
|--------------------------------|--|
| <code>--complete-insert</code> | Zapisuje pełne instrukcje INSERT |
| <code>--compress</code> | Włącza kompresję |
| <code>--extended-insert</code> | Korzysta z nowej składni INSERT, dzięki której odtwarzanie działa szybciej; opcja ta nie mieści się w standardzie SQL |
| <code>--add-drop-table</code> | Dodaje instrukcje DROP TABLE przed tworzeniem każdej tabeli |
| <code>--no-data</code> | Nie zapisuje żadnych danych, tylko strukturę tabel |
| <code>--result-file=</code> | Zapisuje wynik do podanego pliku; podręcznik zaleca używanie tej opcji w systemach opartych na MS-DOS, ponieważ pozwala to uniknąć problemów ze znakami końca wiersza (jeżeli określający je parametr nie zostanie podany, wynik jest wysyłany na standardowe wyjście) |

Ponadto działają parametry `--host`, `--password` i `--user`; program może również odczytać je z plików konfiguracyjnych MySQL.

Skladowanie bazy danych

Program *mysqldump* jest bardzo łatwy w użyciu — nie ma zatem usprawiedliwienia, jeżeli nie tworzysz kopii swoich danych.

Wypróbujmy program *mysqldump* na naszej bazie *bmfinal*, zapisując pełne instrukcje INSERT oraz dodając instrukcje DROP TABLE usuwające istniejące tabele. Wynik zostanie zapisany do pliku *bmfinal.backup*.

```
C:\mysql\data>c:\mysql\bin\mysqldump --complete-insert --add-drop-table
--result-file=bmfinal.backup bmfinal
```

```
C:\mysql\data>
```

Jak to działa?

Jeżeli popatrzymy na wynik działania programu, zobaczymy kompletny skrypt służący do generowania bazy danych, niemal identyczny ze skrypcem używanym na początku książki do utworzenia bazy danych:

```
-- MySQL dump 8.17
--
-- Host: localhost    Database: bmfinal
-----
-- Server version    4.0.1-alpha-max-nt

--
-- Table structure for table 'barcode'
--

DROP TABLE IF EXISTS barcode;
CREATE TABLE barcode (
  barcode_ean char(13) NOT NULL default '',
  item_id int(11) NOT NULL default '0',
  PRIMARY KEY (barcode_ean),
  KEY item_id (item_id)
) TYPE=InnoDB;

--
-- Dumping data for table 'barcode'
--

INSERT INTO barcode (barcode_ean, item_id) VALUES ('6241527836173',1);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('6241574635234',2);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('6241527746363',3);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('6264537836173',3);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('7465743843764',4);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('3453458677628',5);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('6434564564544',6);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('8476736836876',7);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('6241234586487',8);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('9473625532534',8);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('9473627464543',8);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('4587263646878',9);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('2239872376872',11);
INSERT INTO barcode (barcode_ean, item_id) VALUES ('9879879837489',11);

--
-- Table structure for table 'customer'
--

DROP TABLE IF EXISTS customer;
CREATE TABLE customer (
  customer_id int(11) NOT NULL auto_increment,
  title varchar(4) default NULL,
  fname varchar(32) default NULL,
  lname varchar(32) NOT NULL default '',
  addressline varchar(64) default NULL,
  town varchar(32) default NULL,
  zipcode varchar(10) NOT NULL default '',
  phone varchar(16) default NULL,
  PRIMARY KEY (customer_id)
) TYPE=InnoDB;

--
-- Dumping data for table 'customer'
--
```

```
INSERT INTO customer (customer_id, title, fname, lname, addressline, town, zipcode,
phone) VALUES (1, 'Miss', 'Jenny', 'Stones', '27 Rowan Avenue', 'Hightown', 'NT2 1AQ',
'023 9876');
```

Musimy jeszcze zwrócić uwagę na dwie rzeczy. Po pierwsze, w pliku znajdują się instrukcje `DROP TABLE IF EXISTS`, więc uruchomienie tego skryptu na bazie danych, w której istnieją tabele, spowoduje ich usunięcie i powtórne utworzenie. Po drugie, ponieważ w parametrach wiersza poleceń użyliśmy nazwy bazy danych, jej nazwa występuje tylko w komentarzach (wiersze rozpoczynające się od `--`), nie ma jej natomiast we właściwym skrypcie SQL, więc możemy utworzyć kopię bazy *bmfinal* w pod inną nazwą, co może być przydatne w czasie tworzenia aplikacji.

Podsumowanie

W tym rozdziale omawialiśmy cztery główne tematy. Rozpoczęliśmy od podstaw uruchomienia serwera i — co istotniejsze — zapewnienia kontrolowanego zakończenia pracy serwera, co jest niezmiernie ważne dla prawidłowego zabezpieczenia danych.

Następnie omówiliśmy pliki konfiguracyjne, które są zapisane w różnych katalogach w systemie Linux i Windows, choć w obu posiadają podobny format. Pokazaliśmy, w jaki sposób możemy skonfigurować serwer w globalnym pliku konfiguracyjnym i jak ustawić niektóre parametry domyślne dla programów klienckich, co oszczędza nam ciągłego wpisywania tych samych opcji.

Opisaliśmy także polecenia `GRANT` i `REVOKE`, które umożliwiają kontrolowanie dostępu przez użytkowników do poszczególnych elementów danych na serwerze. Korzystając z tych poleceń można dosyć precyzyjnie określać dostęp poszczególnych użytkowników do poszczególnych danych i typów operacji, jakie mogą na tych danych wykonać. Przyjrzelśmy się domyślnemu użytkownikowi, stworzonemu podczas instalacji MySQL i zasugerowaliśmy usunięcie jego konta z baz danych zawierających dane produkcyjne.

Krótko opisaliśmy również program *mysamchk* oraz polecenie `CHECK TABLES`, które są używane do naprawiania uszkodzonych tabel typu MyISAM.

Na koniec przyjrzelśmy się programowi *mysqldump*, który pozwala tworzyć kopie zapasowe danych zapisanych na serwerze.

W tym rozdziale zarysowaliśmy jedynie ogólnie główne zadania administratora serwera MySQL. Zalecamy przeczytanie po zapoznaniu się z tym rozdziałem podręcznika MySQL, w którym znajduje się szczegółowe omówienie przedstawionych tu tematów.