helion.pl

Idź do

- Spis treści
- Przykładowy rozdział

Katalog książek

- Katalog online
- Zamów drukowany katalog

Twój koszyk

- Dodaj do koszyka

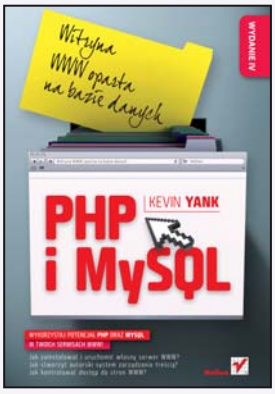
Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

Czytelnia

- Fragmenty książek online

PHP i MySQL. Witryna WWW oparta na bazie danych. Wydanie IV



Autor: Kevin Yank
Tłumaczenie: Daniel Kaczmarek
ISBN: 978-83-246-2580-2
Tytuł oryginału: Build Your Own Database Driven Web Site Using PHP & MySQL
Format: 158×235, stron: 424

Wykorzystaj potencjał PHP oraz MySQL w Twoich serwisach WWW!

- Jak zainstalować i uruchomić własny serwer WWW?
- Jak stworzyć autorski system zarządzania treścią?
- Jak kontrolować dostęp do stron WWW?

PHP i MySQL to najpopularniejszy tandem webmasterski. Ilość serwisów opartych na tym połączeniu jest nie do ogarnięcia. Skąd taka popularność? Niezależnie od innych zalet atutem tego rozwiązania jest prostota. Już kilka chwil wystarczy, żeby rozpocząć przygodę z profesjonalnym tworzeniem serwisów WWW. A jeżeli poświęcisz trochę więcej czasu, poznasz i wykorzystasz jeszcze więcej możliwości PHP i MySQL.
Ale czy rzeczy proste nie mogą być jeszcze prostsze?

Mogą. Z książką „PHP i MySQL. Witryna WWW oparta na bazie danych. Wydanie IV" błyskawicznie przebrniesz przez proces instalacji wszystkich niezbędnych komponentów – niezależnie od systemu, którego używasz. Autor opisuje tu sposób instalacji na platformach Windows, MacOS X oraz Linux. Po udanej instalacji napiszesz swój pierwszy skrypt, a następnie przejdziesz do kolejnych, coraz bardziej zaawansowanych tematów. Wśród nich znajdziesz opisy takich zagadnień, jak: język SQL, składnia PHP, nawiązywanie połączenia z bazą, publikowanie treści znajdujących się w bazie. Ponadto zdobędziesz wiedzę z zakresu administrowania bazą MySQL, systemów zarządzania treścią, zarządzania sesją czy też wykorzystania wyrażeń regularnych. Książka ta jest nieocenioną pomocą dla wszystkich osób zaczynających przygodę ze stronami WWW oraz językami PHP i SQL.

- Instalacja systemu na platformach Windows, Linux oraz MacOS X
- MySQL – podstawowe zagadnienia
- PHP – składnia, podstawowe polecenia i struktury
- Nawiązywanie połączenia z bazą danych z poziomu PHP
- Publikacja treści zawartych w bazie MySQL
- Zasady projektowania relacyjnej bazy danych
- Tworzenie systemu zarządzania treścią
- Wykorzystanie wyrażeń regularnych
- Kontrola dostępu do stron
- Zarządzanie sesją oraz „ciasteczkami"
- Zadania administracyjne w bazie MySQL
- Tworzenie zaawansowanych zapytań MySQL
- Przechowywanie i wykorzystanie danych binarnych

Twórz dynamiczne, bogate w treści i łatwe w zarządzaniu witryny WWW!

Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2010

Spis treści

Rozdział 10.

Administrowanie bazą MySQL

Sercem niemal każdej dobrze zaprojektowanej witryny tematycznej jest relacyjna baza danych. W tej książce utworzyliśmy bazę danych przy użyciu systemu relacyjnych baz danych MySQL. Programiści witryn internetowych chętnie wybierają MySQL, ponieważ jest darmowy, a także ze względu na łatwość uruchomienia. Jak się przekonaliśmy w rozdziale 1., „Instalacja", nowy użytkownik, uzbrojony w odpowiednie instrukcje, może zainstalować i uruchomić serwer MySQL w czasie krótszym niż 30 minut — a przy niewielkim doświadczeniu nawet w krótszym niż dziesięć!

Ten, kto zamierza wykorzystać serwer MySQL tylko do wypróbowania przykładów i poeksperymentowania, prawdopodobnie nie potrzebuje niczego poza opisem procesu instalacji, który omówiliśmy w rozdziale 1., „Instalacja". Natomiast czytelnik, który chce zbudować bazę danych będącą podstawą dla prawdziwej witryny WWW — na przykład witryny firmowej — powinien zapoznać się jeszcze z kilkoma zagadnieniami, zanim zacznie polegać na serwerze MySQL w codziennej pracy.

Najpierw, zgodnie z obietnicą, którą złożyłem w rozdziale 2., pokażę, jak skonfigurować narzędzie *phpMyAdmin*, aby przeglądać i edytować bazy danych oraz administrować nimi z poziomu przeglądarki internetowej. Korzystanie z *phpMyAdmin* jest generalnie o wiele łatwiejsze niż zmaganie się z wierszem poleceń serwera MySQL, zwłaszcza gdy serwer ten działa na innym komputerze.

Później zajmiemy się kopiami zapasowymi. Tworzenie kopii zapasowych danych ważnych dla nas albo naszej firmy powinno mieć wysoką pozycję na liście priorytetów administratora. Niestety, konfigurowanie wykonywania kopii zapasowych nie należy do najciekawszych obowiązków administratora, więc zazwyczaj taka procedura tworzona jest jednorazowo, z konieczności i uznawana za „wystarczająco dobrą" dla wszystkich zastosowań. Każdy, kto dotychczas na pytanie: „Czy powinniśmy zrobić kopie zapasowe naszych baz danych?" odpowiadał: „W porządku, zrobimy kopie razem z całą

resztą", powinien uważnie przeczytać ten rozdział. Zademonstrujemy, dlaczego standardowe rozwiązanie tworzenia kopii zapasowych jest niewystarczające przy wielu instalacjach MySQL oraz pokażemy właściwą metodę wykonywania i przywracania kopii zapasowej bazy MySQL.

W rozdziale 1., „Instalacja", skonfigurowaliśmy serwer MySQL tak, aby pozwalał na łączenie się jako specjalny użytkownik *root* z dowolnie wybranym hasłem. Użytkownik *root* w MySQL (który zresztą nie ma nic wspólnego z użytkownikiem *root* w Linuksie i podobnych mu systemach) ma prawa odczytu i zapisu do wszystkich baz i tabel. W wielu organizacjach konieczne jest tworzenie użytkowników mających dostęp tylko do poszczególnych baz i tabel oraz ograniczanie w jakiś sposób tego dostępu (na przykład dostęp do pewnej tabeli z prawem tylko do odczytu). W tym rozdziale dowiemy się również, jak umożliwić takie obostrzenia przy użyciu dwóch nowych poleceń MySQL, czyli `GRANT` i `REVOKE`.

W pewnych sytuacjach, takich jak zaniki prądu, bazy MySQL mogą ulec uszkodzeniu. W takim przypadku można się jednak odwołać do sporządzonych wcześniej kopii zapasowych. Zakończymy przegląd administrowania bazami MySQL, poznając narzędzie do sprawdzania i naprawiania prostych uszkodzeń baz.

phpMyAdmin

Podobnie jak większość kodu prezentowanego w książce, phpMyAdmin[1] to skrypt PHP przeznaczony do komunikowania się z serwerem MySQL za pośrednictwem stron WWW generowanych „w locie". Jednak zamiast prezentować użytkownikom strony miłe dla oka, phpMyAdmin ma za zadanie udostępnienie interfejsu WWW, za pomocą którego będzie możliwe administrowanie serwerem MySQL.

Aplikacja phpMyAdmin pozwala na wykonanie niemal każdej czynności, dostępnej z poziomu wiersza poleceń serwera MySQL. Jednak w phpMyAdmin używa się do tego celu myszy, a nie wpisywanych ręcznie zapytań języka SQL. Oczywiście, jeżeli musisz wykonać zadanie, które można wyrazić wyłącznie za pomocą ręcznie napisanego kodu SQL, phpMyAdmin również na to pozwoli, ponieważ udostępnia formularz służący do wpisywania kodu SQL. Kod ten zostanie wykonany, a wynik zostanie wyświetlony w przeglądarce.

Na większości komercyjnych serwerów WWW dostępne są już prekonfigurowane wersje phpMyAdmin, stanowiące element konsoli administracyjnej. Poza tym, jeżeli samodzielnie zainstalujesz serwer MySQL pochodzący z rozwiązania pakietowego, takiego jak WampServer czy MAMP, szybko zauważysz, że tam również udostępniana jest aplikacja phpMyAdmin. Wówczas konieczne jednak będzie odpowiednie skonfigurowanie aplikacji przez podanie hasła użytkownika *root* serwera MySQL, które zostało zdefiniowane w rozdziale 1.

[1] *http://www.phpmyadmin.net/*

Jeżeli na serwerze, którego używasz, phpMyAdmin jest niedostępna, możesz zainstalować aplikację samodzielnie, co nie jest trudnym zadaniem. W tym celu należy przejść na stronę pobierania phpMyAdmin[2] i pobrać najnowszą zalecaną wersję (w trakcie powstawania tej książki była to wersja 3.1.3.2) w preferowanym formacie (*.zip* dla systemu Windows lub Mac OS X bądź też *.tar.gz* dla Linuksa). Plik trzeba rozpakować, a po rozpakowaniu okaże się, że archiwum zawiera katalog *phpMyAdmin-**wersja-językowa***. Nazwę tego katalogu trzeba zmienić na *phpMyAdmin* i przenieść do głównego katalogu dokumentów WWW serwera internetowego.

Następnie w edytorze tekstu trzeba utworzyć nowy plik o nazwie *config.inc.php* i zapisać go w katalogu *phpMyAdmin*. W pliku należy wpisać następujący fragment kodu:

```
<?php
$cfg['blowfish_secret'] = 'bhvhbv3577h3qguw83qdh37b2fnqe1inbq38qhg';

$cfg['Servers'][1]['auth_type'] = 'cookie';
?>
```

Elementowi `$cfg['blowfish_secret']` można przypisać dowolną wartość złożoną z liter i cyfr. Nie trzeba nawet zapamiętywać wpisanej wartości, ponieważ wystarczy, by była jak najtrudniejsza do zgadnięcia przez potencjalnego hakera. Dlatego im bardziej ciąg znaków będzie przypominał ciąg wygenerowany losowo, tym lepiej.

Drugi wiersz kodu konfiguruje phpMyAdmin w taki sposób, by aplikacja łączyła się z serwerem MySQL znajdującym się na tym samym komputerze oraz serwer baz danych wymagał podania nazwy użytkownika i hasła. Jeżeli wykorzystywany serwer MySQL działa na innym komputerze albo chcesz, by phpMyAdmin logował się do serwera baz danych automatycznie, trzeba wprowadzić pewne zmiany, które najlepiej są opisane w dokumentacji aplikacji. Wystarczy w przeglądarce internetowej otworzyć plik *Documentation.html*, który znajduje się w katalogu *phpMyAdmin*.

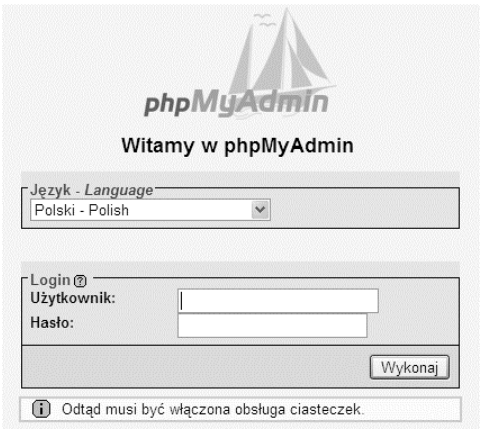
Gdy plik konfiguracyjny jest już gotowy, należy otworzyć przeglądarkę i wywołać adres *http://localhost/phpMyAdmin/* (lub inny, który wskazuje na serwerze WWW katalog *phpMyAdmin*). W odpowiedzi w przeglądarce powinna pojawić się strona logowania, widoczna na rysunku 10.1.

Jeżeli na stronie logowania pojawi się ostrzeżenie dotyczące braku możliwości załadowania rozszerzenia mcrypt, można je z czystym sumieniem zignorować. Na większości serwerów rozszerzenie mcrypt jest komponentem opcjonalnym, którego zainstalowanie dość znacząco powinno zwiększyć wydajność działania phpMyAdmin, lecz i bez tego rozszerzenia aplikacja będzie sobie radzić bez problemu[3].

---

[2] *http://www.phpmyadmin.net/home_page/downloads.php*

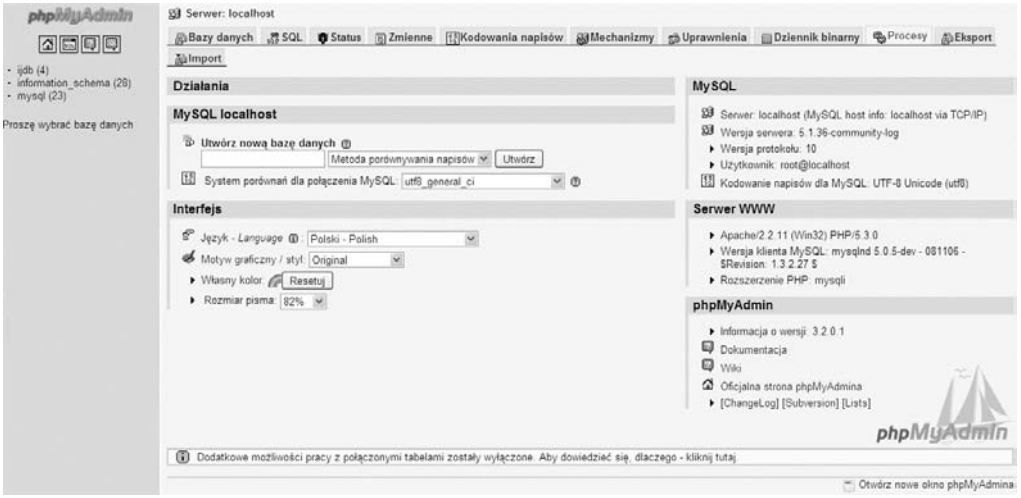
[3] Rozszerzenie mcrypt jest wymagane w niektórych 64-bitowych systemach operacyjnych. Aplikacja phpMyAdmin wyświetli odpowiedni komunikat, jeśli okaże się, że nie może obyć się bez tego rozszerzenia. W takim przypadku trzeba będzie dodać rozszerzenie do PHP. W systemie Windows wystarczy w tym celu jedynie odkomentować wiersz `extension=php_mcrypt.dll` pliku `php.ini`, lecz w systemach Mac OS X i Linux konieczne jest zainstalowanie biblioteki programistycznej libmcrypt i wykonanie rekompilacji PHP. Trudno powiedzieć, by był to najlepszy sposób spędzania popołudnia!



Rysunek 10.1. *Ekran logowania do aplikacji phpMyAdmin*

W polu *Użytkownik* należy wpisać `root`, natomiast w polu *Hasło* trzeba podać hasło użytkownika *root* na serwerze MySQL. Alternatywnie, jeżeli nie znasz hasła użytkownika *root* na serwerze baz danych, z którym się łączysz, wpisz nazwę użytkownika i hasło, jakie znasz. Aplikacja phpMyAdmin zacznie wówczas pracę z takimi samymi uprawnieniami, jakie posiada zastosowany użytkownik MySQL.

Po zalogowaniu się w przeglądarce powinna pojawić się strona podobna do widocznej na rysunku 10.2.

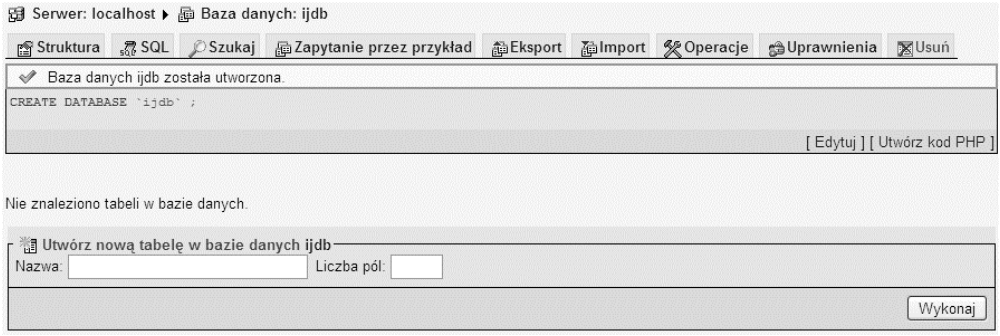


Rysunek 10.2. *Interfejs aplikacji phpMyAdmin może być trochę zniechęcający, ale nie poddawaj się!*

Nie warto stresować się przytłaczającą liczbą opcji udostępnianych przez phpMyAdmin. Aplikacja rzeczywiście jest bardzo złożona, lecz gdy tylko poznasz ją trochę bliżej, praca z nią przestanie być jakimkolwiek problemem.

Na początek warto spojrzeć na listę dostępnych baz danych, widoczną po lewej stronie. Jeżeli phpMyAdmin łączy się z serwerem MySQL, na którym implementowałeś przykłady opisywane w tej książce, powinieneś wśród dostępnych baz danych ujrzeć między innymi bazę `ijdb`. Jeśli tak nie jest, możesz ją utworzyć choćby teraz.

Jeżeli posiadasz uprawnienia do tworzenia baz danych na serwerze MySQL, skorzystaj z formularza *Utwórz nową bazę danych* widocznego mniej więcej na środku ekranu, aby zbudować nową bazę danych `ijdb`. Na liście rozwijanej *System porównań dla połączenia MySQL* można pozostawić wartość domyślną. Gdy klikniesz przycisk *Utwórz*, phpMyAdmin utworzy nową, pustą bazę danych `ijdb` i wyświetli komunikat „Nie znaleziono tabeli w bazie danych", widoczny na rysunku 10.3.



Rysunek 10.3. *Nie znaleziono tabeli w bazie danych*

Jeżeli natomiast możesz korzystać tylko z jednej bazy danych, którą specjalnie utworzył administrator (na przykład administrator dostawcy usług internetowych), kliknij nazwę tej bazy w menu po lewej stronie. Również w tym przypadku w przeglądarce wyświetlona zostanie lista tabel lub komunikat „Nie znaleziono tabeli w bazie danych", jeżeli baza danych będzie pusta.

Teraz za pomocą phpMyAdmin możesz utworzyć wszystkie tabele niezbędne do odpowiedniego funkcjonowania bazy danych witryny zarządzania kawałami. W tym celu kliknij zakładkę *Import* u góry strony, a następnie w sekcji *Plik do importu* wskaż plik archiwum kodów źródłowych książki o nazwie *Rozdzial09/sql/ijdb.sql*. Kliknięcie przycisku *Wykonaj* u dołu strony spowoduje, że aplikacja phpMyAdmin prześle zawartość wskazanego pliku do serwera MySQL i na tej podstawie powstaną tabele i przykładowe dane bazy do zarządzania dowcipami.

Nazwy utworzonych tabel powinny się pojawić z lewej strony ekranu przeglądarki. Po kliknięciu myszą nazwy którejś z tabel będzie można przejrzeć jej zawartość; warto również przejrzeć wszystkie zakładki dostępne w aplikacji phpMyAdmin. Większość dostępnych opcji nie powinna budzić wątpliwości, jednak przeznaczenie niektórych zakładek może wydać się niejasne. Jeżeli jesteś ciekaw, do czego służy dana funkcja, po prostu uruchom ją i zobacz, jaki będzie wynik (oczywiście, na serwerze MySQL, który nie zawiera żadnych danych!). Szczególną ostrożność należy zachować przy przeglądaniu zakładki *Usuń*, która usuwa całą bazę danych.

Niewielu programistów PHP zna każdą funkcję tak złożonego narzędzia, jakim jest phpMyAdmin. Jednak nawet zrozumienie tylko podstawowych funkcji aplikacji sprawi, że narzędzie szybko stanie się nieodzowne do administrowania bazami danych na serwerze MySQL.

Kopie zapasowe baz danych MySQL

Podobnie jak w przypadku serwerów WWW, większość serwerów MySQL musi działać 24 godziny na dobę, siedem dni w tygodniu. Powoduje to, że tworzenie kopii zapasowych plików bazy MySQL jest problematyczne. Serwer MySQL używa pamięci podręcznej i buforowania, aby zwiększyć efektywność uaktualniania plików bazy przechowywanych na dysku, co sprawia, że w każdej chwili pliki te mogą się znajdować w stanie niespójności. Jako że standardowe procedury wykonywania kopii zapasowych zakładają tylko kopiowanie plików systemowych i zawierających dane, kopie plików danych MySQL nie są wiarygodne. Nie ma bowiem żadnej gwarancji, że skopiowane pliki będą w stanie nadającym się do użycia w zastępstwie plików straconych w awarii.

Wiele baz danych otrzymuje nowe informacje przez cały czas, zaś standardowe kopie zapasowe zawierają tylko stan plików z daty ostatniego utworzenia kopii. Jeśli z jakiegoś powodu pliki danych bazy MySQL zostaną zniszczone albo nie da się ich odczytać, wszelkie informacje, które zmieniły się w bazie od ostatniej kopii, przepadną bezpowrotnie. W wielu sytuacjach, na przykład jeśli serwer MySQL służy do śledzenia zamówień klientów w sklepie internetowym, taka strata jest nie do przyjęcia.

MySQL zawiera mechanizmy służące do tworzenia aktualnych kopii zapasowych, na które nie powinna mieć wpływu aktywność serwera podczas archiwizacji. Niestety, ich użycie wymaga zaprojektowania specjalnej procedury wykonywania kopii zapasowych dla danych MySQL, całkowicie oddzielnej od wszelkich systemów archiwizacji, które mogły powstać dla innych danych. Jednak podobnie jak w przypadku każdego dobrego systemu kopii zapasowych, docenimy go, kiedy przyjdzie czas go wykorzystać.

Wykonywanie kopii za pomocą programu mysqldump

Oprócz `mysql`, klienta MySQL, pakiet instalacyjny MySQL zawiera wiele przydatnych programów narzędziowych. Poznaliśmy już na przykład program `mysqladmin`, odpowiedzialny za kontrolę i pobieranie informacji o uruchomionym serwerze MySQL.

Program `mysqldump` jest kolejnym z tych narzędzi. Po uruchomieniu łączy się on z serwerem MySQL (w podobny sposób, jak program `mysql` albo skrypt PHP) i pobiera pełną zawartość wskazanej bazy danych. Pobrane dane są następnie wypisywane w postaci szeregu poleceń SQL `CREATE TABLE` i `INSERT`, które, wykonane na serwerze MySQL, odtworzą bazę danych (lub bazy danych) i zapełnią ją takimi samymi danymi, jakie zawierał oryginał.

Jeśli przekierujemy wynik działania `mysqldump` do pliku, możemy zachować uzyskany zrzut zawartości bazy jako kopię zapasową. Poniższe polecenie (zapisane w jednej linii)

łączy się z serwerem MySQL uruchomionym na komputerze lokalnym, podając użytkownika *root* i hasło `password`, a następnie zapisuje kopię zapasową wszystkich baz danych do pliku *full_backup.sql*[4].

```
mysqldump -u root -ppassword --all-databases > full_backup.sql
```

Poniższe komendy odtworzą bazę po awarii:

```
mysql -u root -ppassword < full_backup.sql
```

Przytoczone polecenie łączy się z serwerem MySQL przy użyciu programu `mysql` i podaje do wykonania listę komend z utworzonego uprzednio pliku. Jeżeli wolisz korzystać z klienta wiersza poleceń serwera MySQL, możesz wykonać polecenie `source`, aby wykonać wszystkie polecenia zawarte w pliku *full_backup.sql*:

```
mysql> source full_backup.sql
```


Uwaga

Polecenie source nie jest poleceniem języka SQL

Polecenie `source` jest słabo udokumentowanym[5] poleceniem obsługiwanym bezpośrednio przez program klienta *mysql* i nie jest to w żadnym razie polecenie języka SQL, takie jak na przykład `CREATE DATABASE`. Dlatego na końcu polecenia nie należy wstawiać znaku średnika, ponieważ znak ten uniemożliwi poprawne wykonanie polecenia.

W ten sposób możemy używać `mysqldump` do tworzenia kopii zapasowych naszych baz danych. Program `mysqldump` nie kopiuje bezpośrednio plików znajdujących się w katalogu danych MySQL, tylko pobiera dane, łącząc się z serwerem. Dzięki temu kopia, którą wykonuje, jest właściwą i pewną kopią bazy, a nie zrzutem plików, które mogą być niepełne i nieaktualne, dopóki serwer MySQL jest uruchomiony.

Co zrobić z dziurą między kopiami zapasowymi, aby mieć pewność, że zarchiwizowane dane z bazy są zawsze aktualne? Rozwiązanie jest proste: należy skonfigurować serwer tak, aby prowadził binarny dziennik aktualizacji.

Kopie przyrostowe w binarnym dzienniku aktualizacji

Jak już wspomniano, w wielu zastosowaniach, w których używa się serwera MySQL, utrata danych — jakichkolwiek danych — jest nie do zaakceptowania. W takich przypadkach trzeba znaleźć sposób na utrzymywanie aktualności danych również pomiędzy kopiami wykonywanymi przy użyciu `mysqldump`. Rozwiązaniem tego problemu jest skonfigurowanie serwera MySQL, tak aby prowadził binarny dziennik aktualizacji. Binarny dziennik aktualizacji jest zapisem wszystkich otrzymanych przez bazę zapytań SQL, które w jakiś sposób zmieniły zawartość bazy. Zaliczają się do nich między innymi polecenia `INSERT`, `UPDATE` i `CREATE TABLE`, natomiast `SELECT` już nie.

[4] Aby uruchomić `mysqldump` i inne programy narzędziowe MySQL, trzeba znajdować się w katalogu *bin* serwera MySQL lub katalog ten musi znajdować się w ścieżce systemowej. Jeżeli wykonałeś instrukcje instalacji zawarte w rozdziale 1., ścieżka systemowa powinna już zawierać odpowiednie wpisy.

[5] *http://dev.mysql.com/doc/refman/5.1/en/batch-commands.html*

Podstawowym celem jest umożliwienie odzyskania zawartości bazy dokładnie z chwili wystąpienia awarii. Można to osiągnąć przez przywrócenie kopii zapasowej (wykonanej za pomocą programu `mysqldump`) i zastosowanie fragmentu binarnego dziennika aktualizacji, powstałego po utworzeniu ostatniej kopii zapasowej.

Można także edytować binarny dziennik aktualizacji, aby cofnąć ewentualne błędy. Jeśli na przykład współpracownik przypadkowo wyda polecenie `DROP TABLE`, możemy wyeksportować binarny dziennik aktualizacji do postaci tekstowej i z pliku tekstowego usunąć szkodliwy wpis. Następnie można przywrócić bazę danych z ostatniej kopii zapasowej i wykonać zmodyfikowany binarny dziennik aktualizacji. W ten sposób zachowujemy nawet te aktualizacje w innych tabelach, do których doszło już *po* wydaniu omyłkowego polecenia. A w ramach przeciwdziałania powinniśmy też zapewne odebrać współpracownikowi prawa do polecenia `DROP` (w następnej części dowiemy się, jak to zrobić).

Uruchamiając serwer MySQL z wiersza poleceń, możemy użyć przełącznika `--log-bin`, aby tworzył binarny dziennik aktualizacji, na przykład w systemie Mac OS X może to wyglądać tak:

```
Komputer:~ użytkownik$ sudo mysqld_safe --log-bin=binlog
```

Powyższe polecenie uruchamia serwer MySQL i instruuje go, aby tworzył w katalogu danych serwera (*/usr/local/mysql/data* na komputerach z Mac OS X i Linuksem, jeśli serwer jest skonfigurowany zgodnie z opisem w rozdziale 1., „Instalacja") pliki *binlog.000001*, *binlog.000002* itd. Nowy plik będzie powstawał, ilekroć serwer zapisze zawartość dziennika na dysk; w praktyce następuje to za każdym uruchomieniem serwera. Jeśli chcemy przechowywać binarny dziennik aktualizacji w innym miejscu (co jest zazwyczaj dobrym pomysłem — jeśli zepsuje się dysk zawierający katalog z danymi, lepiej, żeby kopie zapasowe, mimo wszystko, przetrwały ten kataklizm!), możemy podać pełną ścieżkę do niego.

Jeśli serwer MySQL ma działać stale, to prawdopodobnie system operacyjny odpowiada za jego uruchamianie przy starcie. W takim przypadku dodawanie opcji w linii komend może być utrudnione. Prostszą metodą utworzenia dziennika aktualizacji jest ustawienie odpowiedniej opcji w pliku konfiguracyjnym MySQL o nazwie *my.cnf* lub *my.ini*, jeżeli używany jest system Windows.

Podobnie jak plik *php.ini*, który definiuje konfigurację języka PHP na serwerze, tak samo plik *my.cnf* lub *my.ini* jest zwykłym plikiem tekstowym, który zawiera listę opcji sterujących pracą serwera MySQL. Domyślnie MySQL jest instalowany bez żadnego pliku konfiguracyjnego i działa z wykorzystaniem ustawień domyślnych. Aby włączyć obsługę binarnych dzienników aktualizacji, należy utworzyć plik *my.cnf* lub *my.ini* i ustawić odpowiednie opcje.

W systemie Windows użyj Notatnika lub innego edytora tekstu, aby utworzyć plik o nazwie *my.ini* i zapisać go w katalogu instalacyjnym serwera MySQL (na przykład w *C:\Program Files\MySQL\MySQL Server 5.x*).

W systemie Mac OS X lub Linux należy użyć wybranego edytora tekstu i utworzyć plik o nazwie *my.cnf* z odpowiednimi ustawieniami konfiguracyjnymi; potem plik trzeba przenieść do katalogu instalacyjnego serwera MySQL (*/usr/local/mysql*). Aby to wykonać, prawdopodobnie potrzebne będą uprawnienia administratora.


Uwaga

Instalacje pakietowe

W dalszej części punktu zakładam, że samodzielnie zainstalowałeś serwer MySQL od podstaw. Jest to szczególnie dobre rozwiązanie, jeżeli instalujesz serwer produkcyjny.

Rozwiązania pakietowe, takie jak WampServer czy MAMP, zawierają wbudowany, predefiniowany plik konfiguracyjny serwera MySQL. Oczywiście, można zmienić zawartość tego pliku, aby odpowiednio dostosować konfigurację serwera MySQL i włączyć obsługę binarnych dzienników aktualizacji, lecz najlepiej cofnąć się o krok i samodzielnie w całości skonfigurować sposób działania serwera.

Włączenie mechanizmu tworzenia binarnych dzienników aktualizacji na serwerze rozwojowym jest tylko niepotrzebnym utrudnieniem. Jeżeli chcesz mieć kopie zapasowe serwera rozwojowego, po prostu go wyłącz i wykonaj kopię zapasową plików danych serwera MySQL.

Utworzony plik konfiguracyjny powinien zawierać następujący wpis:

```
[mysqld]
log-bin=/tmp/binlog
```

Tak sformułowany wpis nakazuje serwerowi, by pliki binarnego dziennika aktualizacji były przechowywane w katalogu */tmp*. W środowiskach produkcyjnych zwykle wskazuje się bardziej sensowną lokalizację (na przykład dysk zapasowy). Na serwerach Mac OS X i Linux trzeba się przede wszystkim upewnić, że wskazana lokalizacja jest dostępna do zapisu dla konta użytkownika `mysql`, na którym pracuje serwer MySQL.

Ponieważ nowy plik konfiguracyjny jest już gotowy, należy zrestartować serwer MySQL. Od teraz serwer będzie zachowywał się tak, jakby uruchomiono go w wierszu poleceń z opcją `--log-bin`. Aby upewnić się, że wszystko działa prawidłowo, warto we wskazanej lokalizacji sprawdzić, czy w momencie uruchomienia serwera powstał w niej nowy plik dziennika.

Oczywiście, pliki binarnych dzienników aktualizacji mogą zajmować znaczną ilość miejsca zwłaszcza wtedy, kiedy serwer jest intensywnie używany. Z tego względu trzeba nakazać serwerowi MySQL, aby za każdym razem, gdy wykonywana jest pełna kopia zapasowa przy użyciu `mysqldump`, serwer usuwał wszystkie przestarzałe pliki binarnego dziennika aktualizacji.

```
mysqldump -u root -ppassword --all-databases --flush-logs --master-data=2
↪--delete-master-logs > backup.sql
```

Opcja `--flush-logs` nakazuje serwerowi MySQL, by bieżący plik binarnego dziennika aktualizacji zamknąć oraz utworzyć nowy plik, tak jakby serwer został wyłączony i ponownie uruchomiony. Opcja `--master-data=2` wskazuje programowi `mysqldump`, że na końcu pliku *ijdb_backup.sql* trzeba umieścić komentarz zawierający nazwę nowego pliku binarnego dziennika aktualizacji. Plik ten będzie zawierał wpisy dotyczące pierw-

szych zmian dokonanych w bazie danych bezpośrednio po wykonaniu pełnej kopii zapasowej. Ostatnia opcja `--delete-master-logs` nakazuje programowi `mysqldump` usunięcie tych plików binarnego dziennika aktualizacji, które po utworzeniu pełnej kopii zapasowej nie są już potrzebne.

W przypadku poważnej awarii posiadanie pełnej kopii zapasowej oraz plików binarnego dziennika aktualizacji wygenerowanych po utworzeniu pełnej kopii pozwoli na stosunkowo bezproblemowe przywrócenie bazy danych. W tym celu trzeba będzie zainstalować i skonfigurować nową, pustą wersję serwera MySQL i odtworzyć na nim pełną kopię zapasową, zgodnie z opisem z poprzedniego punktu. Potem zostanie już tylko zaaplikowanie plików binarnego dziennika aktualizacji przy użyciu programu narzędziowego *mysqlbinlog*, wchodzącego w skład instalacji serwera MySQL.

Zadanie programu narzędziowego *mysqlbinlog* polega na przekształceniu formatu danych, w jakim zapisywane są binarne dzienniki aktualizacji, w polecenia języka SQL, które należy wykonać na bazie danych. Powiedzmy, że istnieją dwa pliki binarnego dziennika aktualizacji, które trzeba odtworzyć po przywróceniu bazy danych z ostatniej pełnej kopii zapasowej. Za pomocą programu *mysqlbinlog* z dwóch plików binarnego dziennika można wygenerować tekstowy plik z odpowiednimi poleceniami języka SQL, a następnie wykonać zawartość utworzonego pliku na serwerze MySQL tak, jakby był to plik wygenerowany przez narzędzie *mysqldump*:

```
mysqlbinlog binlog.000041 binlog.000042 > binlog.sql
mysql -u root -ppassword < binlog.sql
```

Kontrola dostępu w MySQL

W rozdziale 2., „Wprowadzenie do systemu MySQL”, wspomniałem o bazie `mysql`, która występuje na każdym serwerze MySQL i służy do przechowywania informacji o użytkownikach, ich hasłach i o tym, co wolno im robić. Dotychczas jednak zawsze łączyliśmy się z serwerem jako użytkownik *root*, co dawało nam dostęp do wszystkich baz i tabel.

Konto *root* MySQL może wystarczyć do naszych celów, jeśli do uruchomionego przez nas serwera uzyskają dostęp tylko skrypty PHP i zachowamy ostrożność w podawaniu hasła do konta. Jednak w sytuacjach, w których do serwera MySQL ma dostęp wielu użytkowników (na przykład jeśli firma utrzymująca naszą witrynę WWW przechowuje bazy danych klientów na jednym serwerze), warto skonfigurować konta użytkowników z bardziej ograniczonymi prawami dostępu.

System kontroli dostępu w MySQL jest w pełni udokumentowany w rozdziale 6. podręcznika MySQL[6]. W skrócie: prawa dostępu użytkowników regulowane są przez zawartość pięciu tabel w bazie `mysql`: `user`, `db`, `host`, `tables_priv` i `columns_priv`. Gdybyśmy chcieli edytować je ręcznie przy użyciu poleceń `INSERT`, `UPDATE` i `DELETE`, powinniśmy się najpierw zapoznać z odpowiednim fragmentem podręcznika. MySQL

[6] *http://dev.mysql.com/doc/mysql/en/privilege_system.html*

dostarcza jednak prostszą metodę zarządzania prawami dostępu, przeznaczoną dla nas, zwykłych śmiertelników. Używając poleceń GRANT i REVOKE — niestandardowych komend MySQL — możemy tworzyć użytkowników i przydzielać im prawa bez zastanawiania się nad detalami i nad sposobem reprezentacji tego, co stworzymy we wspomnianych tabelach.

Nadawanie uprawnień za pomocą polecenia GRANT

Polecenie GRANT, służące do tworzenia nowych użytkowników, przydzielania haseł i przypisywania praw, jest następujące:

```
mysql>GRANT uprawnienie [(kolumny)] ON do_czego
    ->TO użytkownikowi [IDENTIFIED BY 'hasło']
    ->[WITH GRANT OPTION];
```

A zatem polecenie to ma wiele pustych pól do wypełnienia. Opiszemy każde z nich, a potem obejrzymy kilka przykładów, aby się przekonać, jakie efekty dają razem.

Pole *uprawnienie* definiuje uprawnienia, które chcemy przydzielić tym poleceniem. Uprawnienia, które można nadać, dzielą się na trzy grupy:

- Uprawnienia do baz, tabel lub kolumn

ALTER	Modyfikowanie istniejących tabel (na przykład dodawanie albo usuwanie kolumn) i indeksów.
CREATE	Tworzenie nowych baz i tabel.
DELETE	Usuwanie pozycji z tabeli.
DROP	Usuwanie tabel lub baz.
INDEX	Tworzenie lub usuwanie indeksów.
INSERT	Dodawanie nowych pozycji w tabeli.
LOCK TABLES	Blokowanie tabel, do których użytkownik ma prawo SELECT (zob. rozdział 11., *Zaawansowane zapytania SQL*).
SELECT	Przeglądanie i przeszukiwanie zawartości tabeli.
SHOW DATABASES	Przeglądanie listy dostępnych baz danych.
UPDATE	Modyfikowanie istniejących pozycji w tabeli.

- Globalne uprawnienia administracyjne

FILE	Czytanie i zapisywanie plików na komputerze, na którym działa serwer MySQL.
PROCESS	Oglądanie lub niszczenie wątków należących do innych użytkowników.
RELOAD	Przeładowywanie tabel kontroli dostępu, zapisywanie dziennika na dysku itd.
SHUTDOWN	Zatrzymywanie serwera MySQL.

- Uprawnienia specjalne

`ALL`	Uprawnienie do wszelkich czynności (podobnie jak w przypadku użytkownika *root*), z wyjątkiem prawa do nadawania praw.
`USAGE`	Prawo do zalogowania się i nic ponadto.

Niektóre z tych uprawnień odnoszą się do właściwości MySQL, których jeszcze nie poznaliśmy, ale większość nie powinna być obca czytelnikom.

Pole *do_czego* określa, do których części serwera baz danych odnoszą się nadawane prawa.

- Zapis *.* oznacza, że prawa dotyczą wszystkich baz i tabel.
- *nazwaBazy*.* oznacza, że prawa obejmują wszystkie tabele w bazie `nazwaBazy`.
- *nazwaBazy.nazwaTabeli* oznacza natomiast, że prawa dotyczą tylko tabeli `nazwaTabeli` w bazie `nazwaBazy`.

Można nawet określić prawa do pojedynczych kolumn w tabeli — wystarczy wstawić listę kolumn w nawias następujący po nazwie przydzielanych praw (zob. przykład poniżej).

Pole `user` wyznacza użytkownika, którego mają dotyczyć prawa. W MySQL użytkownik jest określany przez nazwę użytkownika podaną przy logowaniu oraz nazwę hosta lub adres IP komputera, z którego się łączy. Wartości te rozdziela znak `@` (tj. `nazwa_`
`↪użytkownika@nazwa_hosta`). Obie mogą zawierać znak globalny `%`, ale każdą zawierającą go wartość trzeba ująć w cudzysłów (na przykład `kevin@"%"` pozwoli użytkownikowi o nazwie `kevin` zalogować się z dowolnego hosta i użyć podanych uprawnień).

Pole *hasło* definiuje hasło wymagane od użytkownika przy łączeniu się z serwerem MySQL. Nawiasy kwadratowe oznaczają, że część `IDENTIFIED BY` *'hasło'* polecenia `GRANT` jest opcjonalna. Każde podane w niej hasło zamieni dotychczasowe hasło dla danego użytkownika. Jeśli nie podamy hasła dla nowo tworzonego użytkownika, serwer nie będzie od niego wymagał podania hasła przy połączeniu.

Opcjonalna część `WITH GRANT OPTION` oznacza, że użytkownik ma prawo używać poleceń `GRANT` i `REVOKE` do nadawania innym użytkownikom takich samych uprawnień, jakie sam otrzymał. Tę opcję należy przydzielać ostrożnie — następstwa nie zawsze są oczywiste! Użytkownik założony z opcją `WITH GRANT OPTION` może przekazywać tę opcję innym użytkownikom, aby wymieniać się z nimi prawami dostępu.

Rozważmy kilka przykładów. Poniższe polecenie `GRANT` tworzy użytkownika o nazwie `dbmgr`, mogącego logować się z komputera `server.host.net` z hasłem `managedb` i mającego pełny dostęp tylko do bazy danych o nazwie `ijdb` (z prawem przydzielania innym użytkownikom praw dostępu do tej bazy włącznie).

```
mysql>GRANT ALL ON ijdb.*
    ->TO dbmgr@example.com
    ->IDENTIFIED BY 'managedb'
    ->WITH GRANT OPTION;
```

Następnie, aby zmienić hasło tego użytkownika na funkychicken, użyjemy następującego polecenia:

```
mysql>GRANT USAGE ON *.*
    ->TO dbmgr@example.com
    ->IDENTIFIED BY 'funkychicken';
```

Zauważmy, że nie nadaliśmy żadnych dodatkowych uprawnień (uprawnienie USAGE pozwala użytkownikowi jedynie na zalogowanie się), ale dotychczasowe przywileje użytkownika pozostały niezmienione.

Następnie załóżmy istnienie nowego użytkownika o nazwie jess, który będzie się łączyć z różnych komputerów w domenie example.com. Powiedzmy, że użytkowniczka Jess jest odpowiedzialna za aktualizowanie personaliów i adresów e-mail autorów, ale może też czasem przeglądać pozostałe informacje w bazie. W związku z tym uzyska dostęp tylko do odczytu (tj. SELECT) do bazy ijdb, ale będzie też mogła wykonywać UPDATE na kolumnach nazwa i email w tabeli autor. Oto odpowiednie polecenia:

```
mysql>GRANT SELECT ON ijdb.*
    ->TO jess@"%.example.com"
    ->IDENTIFIED BY "jessrules";
mysql>GRANT UPDATE (nazwa, email) ON ijdb.autor
    ->TO jess@"%.example.com";
```

Zauważmy, że w pierwszym poleceniu użyliśmy w nazwie hosta znaku globalnego %, aby określić, z jakich komputerów będzie się mogła łączyć Jess. Nie daliśmy jej także prawa do przekazywania uprawnień innym użytkownikom, ponieważ nie dodaliśmy klauzuli WITH GRANT OPTION na końcu linii. Drugie polecenie pokazuje, jak przydzielać uprawnienia do określonych kolumn w tabeli — należy umieszczać rozdzielone przecinkami kolumny w nawiasie za nazwą nadawanego uprawnienia.

Aby sprawdzić, jakie uprawnienia nadano konkretnemu użytkownikowi, należy wykonać polecenie SHOW GRANTS:

```
mysql> SHOW GRANTS FOR jess@"%.example.com"
```

Polecenie zwróci listę poleceń GRANT, których należy użyć, aby na nowo utworzyć wskazanego użytkownika z identycznymi uprawnieniami.

Odbieranie uprawnień przy użyciu polecenia REVOKE

Polecenie REVOKE, jak można się domyślić, służy do odbierania użytkownikowi poprzednio nadanych uprawnień dostępu. Składnia polecenia jest następująca:

```
mysql>REVOKE uprawnienie [(kolumny)]
    ->ON do_czego FROM użytkownikowi;
```

Wszystkie pola w poleceniu mają to samo działanie, co w przypadku polecenia GRANT.

Aby odebrać uprawnienia do używania polecenia DROP współpracownikowi Jess o nazwie idiot (jeśli, na przykład, ma on zwyczaj sporadycznego omyłkowego usuwania za pomocą tego polecenia tabel i baz danych), użylibyśmy następującego polecenia:

```
mysql>REVOKE DROP ON *.* FROM idiot@"%.example.com";
```

Cofnięcie prawa użytkownika do zalogowania się jest jedyną rzeczą, jakiej nie można osiągnąć przy użyciu poleceń `GRANT` i `REVOKE`. Te polecenia definitywnie odbiorą użytkownikowi wszelką możliwość jakichkolwiek działań w bazie poza zalogowaniem się:

```
mysql>REVOKE ALL PRIVILEGES ON *.* FROM idiot@"%.example.com";
mysql>REVOKE GRANT OPTION ON *.* FROM idiot@"%.example.com";
```

Jednak aby całkowicie usunąć użytkownika, trzeba wykonać polecenie `DROP USER`:

```
mysql>DROP USER idiot@"%.example.com"
```

Porady na temat kontroli dostępu

Sposób, w jaki działa system kontroli dostępu w MySQL, ma pewne idiosynkrazje, o których trzeba wiedzieć przed rozpoczęciem zakładania kont użytkowników.

Jeśli tworzymy użytkowników mogących się zalogować do serwera MySQL tylko z komputera, na którym działa serwer (na przykład wymagamy, aby logowali się do komputera i uruchamiali stamtąd klienta MySQL, ewentualnie porozumiewali się z serwerem przy użyciu skryptów serwerowych w rodzaju PHP), możemy się zastanawiać, jaka powinna być część polecenia `GRANT` dotycząca użytkownika. Załóżmy, że serwer jest uruchomiony na komputerze *www.example.com*. Jak powinna wyglądać nazwa użytkownika: *`nazwa_użytkownika`*`@example.com` czy *`nazwa_użytkownika`*`@localhost`?

Odpowiedź brzmi: nie można polegać na żadnej z nich, jeśli chcemy obsłużyć wszystkie połączenia. Teoretycznie, jeżeli łączący się użytkownik poda nazwę hosta w programie klienckim `mysql` albo w funkcji PHP `mysqli_connect`, nazwa ta zostanie porównana z wpisem w systemie kontroli dostępu. Ponieważ jednak raczej nie chcemy zmuszać użytkowników do podawania określonej nazwy hosta (ci używający klienta `mysql` nie będą zapewne chcieli podawać nazwy hosta w ogóle), lepiej uciec się do rozwiązania zastępczego.

Dla użytkowników potrzebujących możliwości łączenia się z komputera, na którym jest uruchomiony serwer MySQL, najlepiej stworzyć dwa wpisy w systemie dostępu MySQL: jeden z rzeczywistą nazwą komputera (na przykład *`nazwa_użytkownika`*`@example.com`), drugi z nazwą `localhost` (na przykład *`nazwa_użytkownika`*`@localhost`). Oczywiście, wymaga to oddzielnego nadawania i usuwania uprawnień dla każdej z nazw, ale tylko na takim rozwiązaniu można naprawdę polegać.

Kolejnym problemem, często napotykanym przez administratorów MySQL, jest to, że nazwy użytkowników zawierające nazwy wieloznaczne (na przykład przytaczana `jess@%.example.com`) mogą nie zadziałać. Niepowodzenia następują zazwyczaj w wyniku sposobu, w jaki MySQL nadaje priorytety wpisom w systemie kontroli dostępu. Wpisy są szeregowane w takiej kolejności, aby bardziej jednoznaczne nazwy hosta następowały jako pierwsze (na przykład `www.example.com` jest nazwą całkowicie jednoznaczną, `%.example.com` — mniej jednoznaczną, a `%` — całkiem niejednoznaczną).

System kontroli dostępu nowo zainstalowanego serwera MySQL zawiera dwa anonimowe wpisy (pozwalające na łączenie się przy użyciu dowolnej nazwy użytkownika z `localhost` i rzeczywistej nazwy lokalnego hosta) i dwa wpisy dla użytkownika *root*. Opisany powyżej problem występuje wówczas, gdy wpisy anonimowe mają pierwszeństwo przed nowo dodanymi użytkownikami ze względu na bardziej jednoznaczną nazwę hosta.

Spójrzmy na uproszczoną zawartość tabeli `user` na *www.example.com*, naszym fikcyjnym serwerze MySQL, po tym, jak dodaliśmy wpis dla `jess`. Wiersze są posortowane w porządku, w jakim przegląda je serwer, kiedy uwierzytelnia połączenie.

```
+-----------------+------+-----------------+
| Host            | User | Password        |
+-----------------+------+-----------------+
| localhost       | root | encrypted value |
| www.example.com | root | encrypted value |
| localhost       |      |                 |
| www.example.com |      |                 |
| %.example.com   | jess | encrypted value |
+-----------------+------+-----------------+
```

Jak widzimy, wpis dla użytkowniczki `jess` ma najmniej jednoznaczną nazwę hosta, zatem pojawia się na liście jako ostatni. Kiedy `jess` próbuje się połączyć spod adresu *www.example.com*, serwer dopasowuje jej próbę do jednego z użytkowników anonimowych (pusta wartość w kolumnie `User` pasuje do każdego użytkownika). Ponieważ wpisy anonimowe nie wymagają hasła, a `jess` zapewne podaje swoje hasło, MySQL odrzuca próbę połączenia. Nawet jeżeli użytkowniczka `jess` zdoła połączyć się bez hasła, uzyska bardzo ograniczone uprawnienia przeznaczone dla użytkowników anonimowych, zamiast praw określonych w jej wpisie w systemie kontroli dostępu.

Istnieją dwa rozwiązania tego problemu. Można rozpocząć administrowanie serwerem MySQL od usunięcia wpisów dla użytkowników anonimowych (`DELETE FROM mysql.user WHERE User=""`) albo dodać po dwa dodatkowe wpisy dla wszystkich użytkowników, którzy potrzebują możliwości łączenia się z adresu `localhost` (tj. wpis dla `localhost` i dla rzeczywistej nazwy serwera).

```
+-----------------+------+-----------------+
| Host            | User | Password        |
+-----------------+------+-----------------+
| localhost       | root | encrypted value |
| www.example.com | root | encrypted value |
| localhost       | jess | encrypted value |
| www.example.com | jess | encrypted value |
| localhost       |      |                 |
| www.example.com |      |                 |
| %. example.com  | jess | encrypted value |
+-----------------+------+-----------------+
```

Trzy wpisy w systemie dla każdego użytkownika stanowią nadmiar, więc zalecane jest usuwanie użytkowników anonimowych, o ile nie ma się dla nich szczególnych zastosowań.

```
+-----------------+------+-----------------+
| Host            | User | Password        |
+-----------------+------+-----------------+
| localhost       | root | encrypted value |
| www.example.com | root | encrypted value |
| %.example.com   | jess | encrypted value |
+-----------------+------+-----------------+
```

Problem braku dostępu

Zapomnienie hasła po trwającym godzinę instalowaniu i konfigurowaniu serwera MySQL jest równie zawstydzające, jak zamknięcie kluczyków w samochodzie. Na szczęście, nie wszystko jest stracone, jeśli mamy uprawnienia administratora na komputerze, na którym działa serwer MySQL, albo możemy się zalogować jako użytkownik, którego skonfigurowaliśmy do uruchamiania serwera (`mysql`, jeżeli serwer został zainstalowany zgodnie z instrukcją instalacji w systemie Linux zamieszczoną w rozdziale 1., „Instalacja"). Poniższa procedura pozwala na odzyskanie kontroli nad serwerem.

Po pierwsze, trzeba wyłączyć serwer MySQL. Zazwyczaj robimy to przy użyciu narzędzia `mysqladmin`, ale ponieważ wymaga ono podania zapomnianego hasła, musimy zamiast tego zakończyć proces serwera. W systemie Windows należy użyć Menedżera zadań, aby znaleźć i zakończyć proces MySQL, lub po prostu zatrzymać usługę MySQL, jeśli została zainstalowana. W systemach Mac OS X i Linux należy użyć polecenia `ps` albo zajrzeć do pliku PID serwera w katalogu danych MySQL, aby określić ID procesu serwera, po czym zatrzymać proces za pomocą następującego polecenia:

```
kill pid
```

pid to identyfikator procesu serwera MySQL.

To powinno wystarczyć do wyłączenia serwera. *Nie należy* używać polecenia `kill -9`, dopóki *nie jest* to absolutnie konieczne, ponieważ może ono spowodować uszkodzenie plików z tabelami. Jeśli jesteśmy do tego zmuszeni, w dalszej części rozdziału znajdują się instrukcje sprawdzania i naprawiania plików z danymi.

Gdy serwer został wyłączony, można uruchomić go ponownie za pomocą polecenia `mysqld_safe` z opcją `--skip-grant-tables`. Oznacza to, że serwer MySQL ma pozwolić na nieograniczony dostęp dla każdego. Oczywiście należy uruchamiać serwer w tym trybie najrzadziej jak to możliwe, ze względu na nieuniknione zagrożenie dla bezpieczeństwa, jakie pociąga za sobą to rozwiązanie.

Po połączeniu się trzeba zmienić hasło użytkownika *root* na takie, którego nie zapomnimy:

```
mysql>UPDATE mysql.user SET Password=PASSWORD("nowehasło")
    ->WHERE User="root";
```

Na koniec rozłączamy się i wydajemy serwerowi polecenie przeładowania tabel z uprawnieniami użytkowników, aby system MySQL ponownie zaczął żądać podawania haseł:

```
mysqladmin flush-privileges
```

Problem rozwiązany — i nikt nie musi wiedzieć, co się wydarzyło. Tak jak w przypadku zamknięcia kluczyków w samochodzie, jesteśmy zdani tylko na siebie.

Sprawdzanie i naprawianie plików danych MySQL

Gdy zanika prąd, w sytuacjach, kiedy wymuszamy zatrzymanie procesu serwera MySQL przy użyciu polecenia `kill -9`, i wtedy, gdy kolega Jess, `idiot@%.example.com`, kopnie wtyczkę zasilania, istnieje ryzyko, że pliki danych MySQL ulegną uszkodzeniu. Może się to zdarzyć, jeśli w czasie wystąpienia jednego z powyższych zakłóceń serwer zmienia zawartość plików; wtedy po zdarzeniu pliki mogą okazać się zniekształcone albo niespójne. Taki rodzaj uszkodzenia potrafi być subtelny i może pozostać niewykryty przez wiele dni, tygodni, a nawet miesięcy. W efekcie, kiedy w końcu odkryjemy problem, wszystkie kopie zapasowe mogą już zawierać to samo uszkodzenie.

W rozdziale 6. podręcznika MySQL[7] znajduje się opis narzędzia `myisamchk`, standardowo dostępnego w instalacji MySQL, oraz instrukcja używania go w celu sprawdzania i naprawy plików danych MySQL. O ile zapoznanie się z tym rozdziałem jest zalecane dla każdego, kto zamierza utworzyć wysoko wydajny, bezpieczny system utrzymania serwera, przedstawimy w tej części podstawowe zagadnienia.

Zanim jednak przejdziemy do dalszych rozważań, warto abyśmy zdali sobie sprawę, że program `myisamchk` oczekuje wyłącznego dostępu do plików danych MySQL, które sprawdza i modyfikuje. Jeśli działający jednocześnie serwer MySQL zapisze coś do pliku, który właśnie jest sprawdzany, `myisamchk` może mylnie uznać to za błąd i próbować go skorygować — co może z kolei spowodować błąd serwera. Aby zatem uniknąć pogorszenia zamiast naprawienia problemu, warto wyłączyć serwer MySQL, jeśli pracujemy na jego plikach z danymi. Alternatywnym rozwiązaniem jest wyłączenie serwera tyko na czas zrobienia kopii plików i praca z kopiami. W takim przypadku po zakończonej pracy trzeba wyłączyć serwer ponownie, aby zamienić stare pliki na nowe i, być może, zastosować wpisy z utworzonego tymczasem binarnego dziennika aktualizacji.

Struktura katalogu danych MySQL nie jest zbyt skomplikowana. Każdej bazie odpowiada jeden podkatalog, zawierający pliki danych, w których przechowywana jest zawartość tabel w bazie. Każdą tabelę reprezentują trzy pliki o takich samych nazwach, jak nazwa tabeli, ale różnych rozszerzeniach. Plik *nazwaTabeli.frm*, nazywany plikiem formatu tabeli, zawiera definicję tabeli, czyli wchodzące w jej skład kolumny i ich typy danych. Plik *nazwaTabeli.MYD* przechowuje wszystkie dane w tabeli. Plik *nazwaTabeli.MYI* zawiera wszelkie indeksy istniejące dla tabeli (na przykład tabelę przeglądową, która pomaga kolumnie klucza głównego przyspieszać zapytania w tej tabeli).

[7] *http://dev.mysql.com/doc/mysql/en/table-maintenance.html*

Chcąc sprawdzić tabelę na okoliczność błędów, należy uruchomić program `myisamchk` (znajdujący się w katalogu *bin* serwera MySQL) i podać albo ścieżkę do plików i nazwę tabeli, albo nazwę indeksu tabeli[8]:

```
myisamchk /usr/local/mysql/data/nazwaBazy/nazwaTabeli
myisamchk /usr/local/mysql/data/nazwaBazy/nazwaTabeli.MYI
```

Albo, w systemie Windows:

```
myisamchk "C:\"Program Files"\MySQL\data\nazwaBazy\nazwaTabeli"
myisamchk "C:\"Program Files"\MySQL\data\nazwaBazy\nazwaTabeli.MYI"
```

Każda z powyższych komend wykona test podanej tabeli. Aby sprawdzić wszystkie tabele w bazie, należy użyć nazwy wieloznacznej:

```
myisamchk /usr/local/mysql/data/nazwaBazy/*.MYI
```

Natomiast aby sprawdzić wszystkie tabele we wszystkich bazach, trzeba użyć dwóch znaków wieloznacznych:

```
myisamchk /usr/local/mysql/data/*/*.MYI
```

Uruchomiony bez żadnych opcji program `myisamchk` wykonuje standardowy test plików tabel. Jeśli podejrzewamy problem w tabeli, a zwykłe sprawdzenie nie znajduje żadnych błędów, możemy wykonać znacznie bardziej wyczerpujący (ale też znacznie wolniejszy!) test przy użyciu opcji `--extend-check`:

```
myisamchk --extend-check /sciezka/do/tabeli
```

Sprawdzanie pod kątem błędów jest nieszkodliwe, nie trzeba więc martwić się możliwością pogorszenia istniejącego problemu podczas testowania plików danych. Z kolei naprawianie tabel, mimo że zazwyczaj bezpieczne, zmienia pliki danych w sposób nieodwracalny. Z tego powodu bardzo zalecane jest tworzenie kopii uszkodzonych plików tabel przed rozpoczęciem prób naprawiania. Przed kopiowaniem plików z danymi należy także upewnić się, że serwer MySQL jest wyłączony.

Istnieją trzy metody naprawcze, których można użyć, aby rozwiązać problem uszkodzonej tabeli. Należy wypróbowywać je po kolei, za każdym razem na nowej kopii plików z danymi (tj. nie wolno stosować drugiej procedury odzyskiwania danych na zestawie plików, pozostałym po nieudanej próbie pierwszej metody). Jeżeli w którymkolwiek etapie otrzymamy komunikat błędu o niemożności utworzenia pliku tymczasowego, trzeba skasować plik, o którym mówi komunikat, i ponowić próbę — wspomniany plik jest pozostałością po poprzedniej próbie naprawy.

Trzy metody naprawiania plików stosuje się w kolejności ukazanej poniżej:

```
myisamchk --recover --quick /sciezka/do/tabeli
myisamchk --recover /sciezka/do/tabeli
myisamchk --safe-recover /sciezka/do/tabeli
```

[8] Aby uprościć opis, nie wspomniałem już, że w systemach Mac OS X i Linux uzyskanie dostępu do plików danych serwera MySQL będzie wymagać posiadania uprawnień administratora. Aby uruchomić `myisamchk` z uprawnieniami administratora, zamiast polecenia `myisamchk` należy wykonać polecenie `sudo myisamchk`.

Pierwsza z nich jest najszybsza i naprawia najczęściej spotykane błędy; ostatnia jest najwolniejsza i potrafi usunąć kilka problemów, z którymi nie radzą sobie pozostałe metody.

Jeśli wymienione sposoby nie pomogą odzyskać zniszczonej tabeli, można spróbować jeszcze kilku sztuczek:

- Jeśli podejrzewamy, że plik z indeksami tabeli (*nazwaTabeli.MYI*) jest uszkodzony w sposób nieodwracalny albo w ogóle nie istnieje, można go wygenerować od zera i użyć z istniejącymi plikami danych (*nazwaTabeli.MYD*) i formatu tabeli (*nazwaTabeli.frm*). Zaczynamy od skopiowania pliku z danymi tabeli (*nazwaTabeli.MYD*). Następnie uruchamiamy ponownie serwer MySQL, łączymy się z nim i usuwamy zawartość tabeli za pomocą następującego polecenia:

  ```
  mysql>DELETE FROM nazwaTabeli;
  ```

  Powyższe polecenie nie tylko usuwa zawartość tabeli, ale również tworzy dla niej nowy plik indeksów. Wylogowujemy się z serwera i wyłączamy go ponownie, po czym nadpisujemy nowo stworzony, pusty plik z danymi (*nazwaTabeli.MYD*) skopiowanym uprzednio plikiem. Na zakończenie naprawiamy pliki standardową metodą (drugą z wymienionych powyżej), co powoduje, że `myisamchk` generuje na nowo zawartość pliku z indeksami na podstawie zawartości plików danych i formatu.

- Jeśli plik formatu tabeli (*nazwaTabeli.frm*) został usunięty albo bezpowrotnie uszkodzony, ale mamy wystarczającą wiedzę o tabeli, aby odtworzyć polecenie `CREATE TABLE`, za pomocą którego została utworzona, możemy ponownie wygenerować plik *.frm* i użyć go z istniejącymi plikami danych i indeksów. Jeśli plik z indeksami również jest uszkodzony, należy go później odtworzyć metodą podaną powyżej. Najpierw kopiujemy pliki z danymi i indeksami, potem usuwamy oryginały i kasujemy wszelkie ślady po tabeli z katalogu danych serwera.

  Uruchamiamy serwer MySQL i tworzymy nową tabelę przy użyciu tego samego polecenia `CREATE TABLE`. Wylogowujemy się i wyłączamy serwer, po czym kopiujemy dwa uprzednio zachowane pliki w miejsce nowo utworzonych, pustych plików. Nowy plik *.frm* powinien zadziałać, ale na wszelki wypadek wykonujemy też standardową procedurę naprawczą przy użyciu drugiej ze znanych nam metod.

Lepiej się ubezpieczać, niż potem żałować

Przyznaję, że niniejszy rozdział różnił się od poprzednich, urozmaiconych obecnością kodu rozdziałów, do których czytelnicy mogli się już przyzwyczaić. Ale skoncentrowanie się na zagadnieniach tego rozdziału — kopiach zapasowych i przywracaniu danych

MySQL, administrowaniu systemem kontroli dostępu MySQL, sprawdzaniu i naprawianiu tabel — uzbroiło nas w narzędzia, dzięki którym nasz serwer MySQL wytrzyma próbę czasu, nie wspominając już o stałym obciążeniu, któremu będzie musiała sprostać nasza witryna.

W rozdziale 11., „Zaawansowane zapytania SQL”, powrócimy do ciekawszych zagadnień i nauczymy się kilku zaawansowanych technik SQL, dzięki którym serwer relacyjnych baz danych dokona rzeczy, o których się nam nawet nie śniło.