Chapter 5

Protecting Your Data

In This Chapter

▶ Understanding MySQL data security

▶ Adding new MySQL accounts

▶ Modifying existing accounts

▶ Changing passwords

▶ Making backups

▶ Repairing data

▶ Restoring data

Your data is essential to your Web database application. You have spent valuable time developing your database, and it contains important information entered by you or by your users. You need to protect it. In this chapter, I show you how.

Controlling Access to Your Data

You need to control access to the information in your database. You need to decide who can see the data and who can change it. Imagine what would happen if your competitors could change the information in your online product catalog or copy your list of customers — you'd be out of business in no time flat. Clearly, you need to guard your data.

MySQL provides a security system for protecting your data. No one can access the data in your database without an account. Each MySQL account has the following attributes:

- A name
- A *hostname* — the machine from which the account can access the MySQL server
- A password
- A set of privileges

To access your data, someone must use a valid account name and know the password associated with that account. In addition, that person must be connecting from a computer that's permitted to connect to your database via that specific account.

After the user is granted access to the database, what he or she can do to the data depends on what privileges have been set for the account. Each account is either allowed or not allowed to perform an operation in your database, such as `SELECT`, `DELETE`, `INSERT`, `CREATE`, or `DROP`. The settings that specify what an account can do are *privileges,* or *permissions.* You can set up an account with all privileges, no privileges, or anything in between. For instance, for an online product catalog, you want the customer to be able to see the information in the catalog but not be able to change it.

When a user attempts to connect to MySQL and execute a query, MySQL controls access to the data in two stages:

- **Connection verification:** MySQL checks the validity of the account name and password and checks whether the connection is coming from a host that's allowed to connect to the MySQL server by using the specified account. If everything checks out, MySQL accepts the connection.
- **Request verification:** After MySQL accepts the connection, it checks whether the account has the necessary privileges to execute the specified query. If it does, MySQL executes the query.

Any query that you send to MySQL can fail either because the connection is rejected in the first step or because the query is not permitted in the second step. An error message is returned to help you identify the source of the problem.

In the following few sections, I describe accounts and privileges in detail.

Understanding account names and hostnames

Together, the account name and *hostname* (the name of the computer that is authorized to connect to the database) identify a unique account. Two accounts with the same name but different hostnames can exist and can have different passwords and privileges. However, you *cannot* have two accounts with the same name *and* the same hostname.

REMEMBER

The MySQL account name is completely unrelated in any way to the Unix, Linux, or Windows username (also sometimes called the *login name*). If you're using an administrative MySQL account named `root`, it is not related to the Unix or Linux `root` login name. Changing the MySQL login name does not affect the Unix, Linux, or Windows login name, and vice versa.

MySQL account names and hostnames are defined as follows:

- **An account name can be up to 16 characters long.** You can use special characters in account names, such as a space or a hyphen (-). However, you cannot use wildcards in the account name.
- **An account name can be blank.** If an account exists in MySQL with a blank account name, any account name will be valid for that account. A user could use any account name to connect to your database, given that the user is connecting from a hostname that's allowed to connect to the blank account name and uses the correct password, if required. You can use an account with a blank name to allow anonymous users to connect to your database.
- **The hostname can be a name or an IP address.** For example, it can be a name such as `thor.mycompany.com` or an IP (Internet protocol) address such as `192.163.2.33`. The machine on which the MySQL server is installed is `localhost`. The hostname can contain a wildcard, such as `%`, which means any host, or can be blank, which also allows the account to connect from any host.

When MySQL is installed with XAMPP, it automatically installs an account `root@localhost`. Thus, you can access your MySQL server from the computer on which it's installed, and from no other computer. This account is okay for a development account on your local computer.

When you open an account with a Web hosting company, the name and hostname of your database is provided to you. The hostname you use to access the database from your Web site is often `localhost`, but it might be something else. If you don't receive this information, you need to ask for it.

Finding out about passwords

A password is set up for every account. If no password is provided for the account, the password is blank, which means that no password is required. MySQL doesn't have any limit for the length of a password, but sometimes other software on your system limits the length to eight characters. If so, any characters after eight are dropped.

For extra security, MySQL encrypts passwords before it stores them. That means passwords are not stored in the recognizable characters that you entered. This security measure ensures that no one can look at the stored passwords and see what they are.

Unfortunately, some bad people out there might try to access your data by guessing your password. They use software that tries to connect rapidly in succession using different passwords — a practice called *cracking.* The following are some recommendations for choosing a password that is as difficult to crack as possible:

- Use six to eight characters.
- Include one or more of each of the following — uppercase letter, lowercase letter, number, and punctuation mark.
- Do not use your account name or any variation of your account name.
- Do not include any word in a dictionary, including foreign language dictionaries.
- Do not include a name.
- Do not use a word that might be easily identified as related to you, such as a pet's name, the street you live on, and so forth.
- Do not use a phone number or a date.

TIP

A good password is hard to guess and easy to remember. If it's too hard to remember, you might need to write it down, which defeats the purpose of having a password. One way to create a good password is to use the first characters of a favorite phrase. For instance, you could use the phrase "All for one! One for all!" to make this password:

```
Afo!Ofa!
```

This password doesn't include any numbers, but you can fix that by using the numeral *4* instead of the letter *f.* Then your password is

```
A4o!O4a!
```

Or you could use the number *1* instead of the letter *o* to represent one. Then the password is

```
A41!14a!
```

This password is definitely hard to guess. Other ways to incorporate numbers into your passwords include substituting *1* (one) for the letter *l* or substituting *0* (zero) for the letter *o.*

When MySQL is installed with XAMPP, the `root@localhost` account is installed with no password, meaning that no password is required to access the database using this account. Because no one can access the database from any other machine, having no password is probably fine. However, if others have access to your local computer, you might want to add a password to this account.

When you obtain your Web hosting account, you're provided with a MySQL account and password. This information should be provided to you at that time.

Taking a look at account privileges

MySQL uses account privileges to specify who can do what. Anyone using a valid account can connect to the MySQL server, but he or she can do only the things that are allowed by the privileges for the account. For example, an account might be set up so that users can select data but cannot insert or update data.

Privileges can be granted for particular databases, tables, or columns. For instance, an account can be set up that allows the user to select data from all the tables in the database, but insert data into only one table and update only a single column in a specific table.

Privileges can be granted or removed individually or all at once. Table 5-1 lists some privileges that you might want to assign or remove.

Table 5-1 MySQL Account Privileges

Privilege	*Description*
`ALL`	All privileges
`ALTER`	Can alter the structure of tables
`CREATE`	Can create new databases or tables
`DELETE`	Can delete rows in tables
`DROP`	Can drop databases or tables
`FILE`	Can read and write files on the server
`GRANT`	Can change the privileges on a MySQL account
`INSERT`	Can insert new rows into tables
`SELECT`	Can read data from tables
`SHUTDOWN`	Can shut down the MySQL server
`UPDATE`	Can change data in a table
`USAGE`	No privileges

WARNING!

Granting `ALL` is not a good idea because it includes privileges for administrative operations, such as shutting down the MySQL server. You're unlikely to want anyone other than yourself to have such sweeping privileges.

Setting Up MySQL Accounts

An *account* is identified by the account name and the name of the computer allowed to access MySQL using this account. You have one account that you can use to administer your MySQL databases. This account is shown on the phpMyAdmin main page. On your local computer, it's probably `root@localhost`. This is the only account you need for your development site because no one needs to access it from the outside — only from your computer.

On your Web hosting account, the account may be `domain@localhost` or something else. Web hosting companies use different naming conventions. However, you don't need to worry about the hostname. Your Web host handles that. You can see the account and hostname on the phpMyAdmin main page. If you're using a company Web site, your company IT staff provides you with an account name and hostname.

In this book, you're discovering how to write PHP scripts that interact with your database. The script might retrieve data from the database to display on a Web page or store data from a form into the database or both. The script uses a MySQL account in a code statement to access the database. For security reasons, you don't want the account used by the script to have any more privileges than necessary. If the account used by the script has only `SELECT` privileges, you don't have to worry about a bad guy using the script to delete or change data or for other unintended purposes.

You need to create at least one account with limited privileges to use on your Web site in PHP scripts that access the database. When you create a new account, you can specify a password when you create the account or you can add a password later. You can set up privileges when you create the account or add/remove privileges later.

REMEMBER

You don't need to create a restricted account for your PHP scripts on your local computer, where no one can access the scripts from outside. You need to create only the new account for the PHP scripts that are accessed by visitors to your Web site.

The following sections describe how to create accounts, add or change passwords, and add/remove account privileges on your Web hosting account. If your Web site is hosted on a company Web site, you need to discuss adding accounts with the IT staff at your company.

Adding accounts

The preferred way to access MySQL from PHP is to set up an account specifically for this purpose with only the privileges that are needed. Some Web hosts don't allow you to create a new account. If you can't create a new account on your Web hosting account, perhaps your Web host will create a new account for you, with limited privileges.

One way to create accounts is to send SQL queries, such as `INSERT` or `UPDATE`, directly to the `mysql` database that stores the account information. This is a database that's created when MySQL is installed. However, most Web hosts do not give you access to this database, either to send direct SQL queries to affect this database or through your phpMyAdmin interface. Efforts to interact with the `mysql` database generally produce error messages, such as

```
Access denied for user 'me'@'localhost' to database
          'mysql'
```

Instead of allowing you access to the `mysql` database directly, most Web hosts provide a page specifically for the purpose of creating and managing accounts. You need to look at your control panel icons to find the icon for creating new MySQL accounts. Because they are MySQL accounts, the icon is probably in the database section of your control panel. It may be the same icon you use to create a new MySQL database. If you can't figure out where it is, read the documentation provided by your Web host or ask tech support at your Web hosting company.

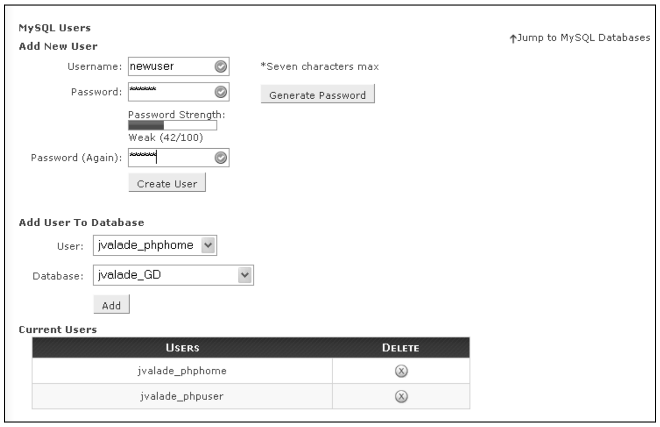
The following steps show how to create a new account on cPanel, a popular control panel used by many Web hosting companies:

1. **Open cPanel on your Web hosting account.**
2. **Find and click the icon for MySQL databases.**

   In cPanel, the icon is located in the section labeled Databases. The icon says MySQL Databases.

   A MySQL databases page opens. Notice that the page lists all the current databases, along with the account names of the accounts allowed to access the database.
3. **Click Jump to MySQL Users in the upper-right corner or scroll down to the MySQL Users section.**

   Figure 5-1 shows the MySQL Users section of the page. The section lists all the current accounts.


MySQL Users
Add New User
Username: newuser *Seven characters max
Password: ****** Generate Password
Password Strength:
Weak (42/100)
Password (Again): ******
Create User
Add User To Database
User: jvalade_phphome
Database: jvalade_GD
Add
Current Users

Users	Delete
jvalade_phphome	⊗
jvalade_phpuser	⊗

Figure 5-1: The MySQL Users section of the MySQL Database page.

4. **Type the new account name into the Username field.**
5. **Type a password into the Password field.**

   Notice the field underneath the password labeled Password Strength. A bar in the field shows how strong the password is. This password isn't very strong, less than 50 percent. Factors that add to password strength are length; making sure it's not a word in the dictionary; and using characters, numbers, and punctuation.

   Notice the Generate Password button. I guarantee the password generated by clicking the button will be 100 percent strong, but I also guarantee that it will be impossible to remember.
6. **Type the same password into the Password (Again) field.**

   This repetition is to ensure you typed the password correctly.
7. **Click the Create User button.**

   A page displays, showing your new account and password.
8. **Click Go Back to return to the MySQL database page.**

   The new account you just created is now listed on the MySQL page as one of the current users. However, if you scroll up to the list of databases, you won't see the new account listed for any of the databases. At this point, the account exists but can't access any databases. You must specifically allow it to access one or more databases, as shown in the next section.

Allowing access to a database

If you use the procedure described in the preceding section, no account has automatic access to any database. You must specifically give the account access to each database. You can give the account access to as many databases as you want the account to use.

To allow access, follow these steps:

1. **Go to the MySQL User section of the MySQL database page.**

   You can see this section in Figure 5-1, shown earlier.

   The list of users should contain all your accounts, including any new account you just created.

2. **In the Add User to Database section, select a user from the User drop-down list.**

   The drop-down list contains all your existing accounts.

3. **Select a database from the Database drop-down list.**

   All your current databases are included in the drop-down list.

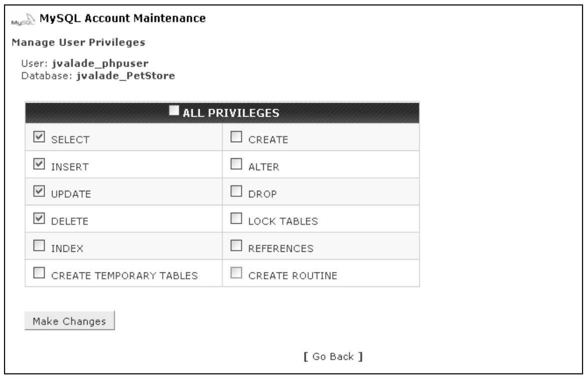
4. **Click the Add button.**

   The selected user is given access to the selected database.

   The Manage User Privileges page opens showing the privileges given the account for the selected database. Because you're just giving this account access to the database for the first time, the account currently has no privileges. You undoubtedly want to select some privileges, if only `SELECT`.

5. **Select the check boxes next to the privileges you want for this account on this database.**

   Figure 5-2 shows the Manage User Privileges page after you have selected some privileges. You can change the privileges at any time, as shown in the next section.


MySQL Account Maintenance

Manage User Privileges

User: jvalade_phpuser
Database: jvalade_PetStore

☐ ALL PRIVILEGES	
☑ SELECT	☐ CREATE
☑ INSERT	☐ ALTER
☑ UPDATE	☐ DROP
☑ DELETE	☐ LOCK TABLES
☐ INDEX	☐ REFERENCES
☐ CREATE TEMPORARY TABLES	☐ CREATE ROUTINE

Make Changes

[Go Back]

Figure 5-2: The Manage User Privileges page.

6. **Click the Make Changes button.**

   A page displays showing that the changes were successful.

7. **Return to the Database page.**

   The account is now listed next to the database name in the list of databases, showing that the account now has access to the database.

Changing privileges

The privileges that you can give an account on a database are listed and explained earlier in this chapter. Accounts should be given only the privileges needed. The previous section explained how to set privileges when creating a new account. In this section, you see how to change the privileges for an existing account.

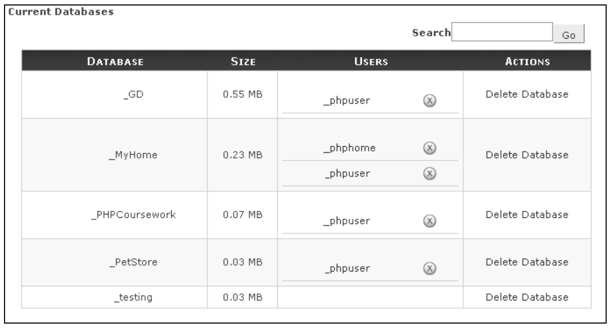
To change an account's privileges, follow these steps:

1. **Open cPanel on your Web hosting account.**
2. **Find and click the icon for MySQL databases.**

   The MySQL databases page opens.

3. **Scroll down to the Current Databases section of the page.**

   You can see a list of your current MySQL databases, as shown in Figure 5-3.



Current Databases

Search [ ] Go

Database	Size	Users	Actions
_GD	0.55 MB	_phpuser ⊗	Delete Database
_MyHome	0.23 MB	_phphome ⊗ _phpuser ⊗	Delete Database
_PHPCoursework	0.07 MB	_phpuser ⊗	Delete Database
_PetStore	0.03 MB	_phpuser ⊗	Delete Database
_testing	0.03 MB		Delete Database

Figure 5-3: The list of your MySQL databases.

   In the database list, each database name starts a row. The third column contains the account names that are allowed to access the database. More than one account can access a database.

4. **Find the row for the database you want to change privileges for.**

   If the account you want to modify is not listed as able to access the database, you must first add it to the list of accounts that are allowed to access this database. To add the account, follow the instructions in the previous section, "Allowing access to a database."

5. **Click the name of the account you want to modify in the row for the database you want to change privileges for.**

   The Manage Account Privileges page opens, as shown earlier in Figure 5-2. The page shows the current privileges that this account has for the database.

6. **Select the check boxes for the privileges you want to add or remove.**
7. **Click the Make Changes button.**

   If you don't click this button, the changes won't be saved.

   A results page displays, showing that the privileges were updated.

Adding and changing passwords

When you create an account, you can add a password or not. You can change a password or add a password to an existing account; you don't need to add the password when the account is created.

To change the password, add the account again. That is, use the same steps you used to create the account. In the Add New User section, type the account name that you want to change the password for and type the new password into the Password and Password (Again) fields. Click the Add User button. The account is added again with the new password. Any existing privileges for any databases remain the same.

In addition, MySQL provides an SQL query specifically for creating a password that looks like this:

```
SET PASSWORD FOR username@hostname = PASSWORD('password')
```

However, most Web hosts do not allow you to use this SQL query. You see the access denied error message, such as

```
Access denied for user 'me'@'localhost' to database
          'mysql'
```

Removing accounts

When you look at the list of usernames on the database page, you see a column named Delete and a red x displayed for each username. To remove any account, click the red x by the account name.

If you look at the list of databases, you see a red x by each username in the User Name column. You can remove access to the database for any username by clicking the red x by the username. The database is not affected, but the username removed can no longer access the specified database. However, the username can still access any other databases for which it has access.

Backing Up Your Data

You need to have at least one copy of your valuable database. Disasters occur rarely, but they do occur. The computer where your database is stored can break down and lose your data, the computer file can become corrupted, the building can burn down, and so on. Backup copies of your database guard against data loss from such disasters.

If your Web site is housed at a Web hosting company or on your company computer, other people are responsible for backing up the Web site, including the database. The administrators of the computers will have backup procedures in place. At least, you can assume they have such procedures. However, it's best to be sure. Talk to your Web hosting company staff or your company IT department about its backup procedures. Be sure it performs backups that make you feel secure about your data and that allow rapid replacement of a damaged database.

Even if you're happy with the backup procedures in place at your Web hosting company, you probably want to back up your database to your local computer. By doing so, you make doubly sure that you have a backup and speed up the process of replacing a damaged file. You can back up your database as often as you consider necessary.

In addition, if your Web site collects data from users, you can install the backup from your Web site on your local computer. Thus, when you're developing and testing on your local development site, you're using the actual database, making your testing more reliable.

REMEMBER

In general, you need to have two backup copies of your data: one copy in a handy location where it can be quickly replaced and another copy in a different physical location from the Web site location for that remote chance that the building burns down. The Web host probably stores the backups on a different computer than the computer that hosts the Web site and/or database. The Web host may also store a copy of its backups offsite. If you back up the database regularly to your local computer, your backup is both convenient and offsite.

You should not copy the actual data files from one computer, such as the Web host computer, to another computer, such as your local computer, exactly as they are. However, you can move the data using features of phpMyAdmin. In the following sections, I use the example of backing up (moving) the data from your Web host to your local computer as an example. You can use the same procedure to move the data from any MySQL database to another.

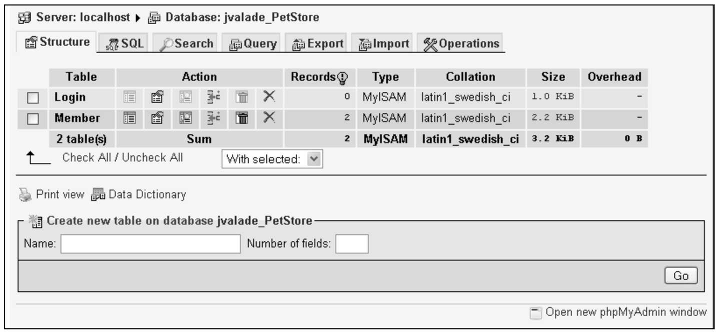
First, you export the database from your Web host. The export procedure saves a text file on your local computer that contains all the SQL queries needed to re-create your database. Then you use the import feature of phpMyAdmin on your local computer to execute the SQL queries in the text file, which builds the database.

Exporting your data with phpMyAdmin

Follow these steps to make a backup copy of the database on your Web hosting company using phpMyAdmin.

1. **Open the main phpMyAdmin page.**
2. **Select a database from the list in the left section of the page.**

   The Database page for the selected database appears, as shown in Figure 5-4.



Server: localhost ▸ Database: jvalade_PetStore

Structure | SQL | Search | Query | Export | Import | Operations

	Table	Action	Records	Type	Collation	Size	Overhead
☐	Login		0	MyISAM	latin1_swedish_ci	1.0 KiB	-
☐	Member		2	MyISAM	latin1_swedish_ci	2.2 KiB	-
	2 table(s)	Sum	2	MyISAM	latin1_swedish_ci	3.2 KiB	0 B

Check All / Uncheck All — With selected:

Print view Data Dictionary

Create new table on database jvalade_PetStore

Name: ______ Number of fields: ______

Go

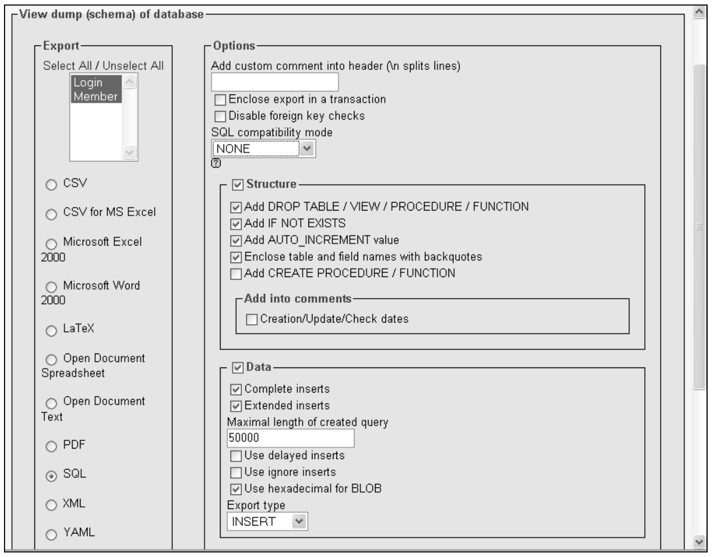
Open new phpMyAdmin window

Figure 5-4: The phpMyAdmin Database page.

   The Database page lists the tables in the database. In this case, the database contains two tables: `Member` and `Login`.

3. **Click the Export tab at the top of the page.**

   The Export page opens, as shown in Figure 5-5.



View dump (schema) of database

Export
Select All / Unselect All
Login
Member

- ○ CSV
- ○ CSV for MS Excel
- ○ Microsoft Excel 2000
- ○ Microsoft Word 2000
- ○ LaTeX
- ○ Open Document Spreadsheet
- ○ Open Document Text
- ○ PDF
- ● SQL
- ○ XML
- ○ YAML

Options
Add custom comment into header (\n splits lines)
- ☐ Enclose export in a transaction
- ☐ Disable foreign key checks

SQL compatibility mode: NONE

☑ Structure
- ☑ Add DROP TABLE / VIEW / PROCEDURE / FUNCTION
- ☑ Add IF NOT EXISTS
- ☑ Add AUTO_INCREMENT value
- ☑ Enclose table and field names with backquotes
- ☐ Add CREATE PROCEDURE / FUNCTION

Add into comments
- ☐ Creation/Update/Check dates

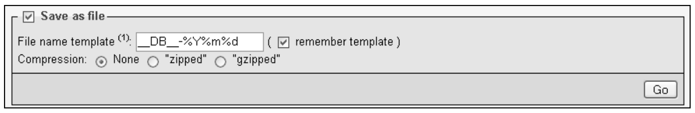
☑ Data
- ☑ Complete inserts
- ☑ Extended inserts

Maximal length of created query: 50000
- ☐ Use delayed inserts
- ☐ Use ignore inserts
- ☑ Use hexadecimal for BLOB

Export type: INSERT

Figure 5-5: The phpMyAdmin Export page.

4. **In the Export section on the left pane of the main panel, in the top list box, select the tables you want to export.**
5. **In the Export section, select the SQL radio button.**
6. **Select the Structure check box and the four check boxes at the top of the Structure section if they aren't already selected.**
7. **Select the Data check box and the Use Hexadecimal for Binary Data check box (or Use Hexadecimal for BLOB check box) if they aren't already selected.**
8. **Scroll down to the File section (see Figure 5-6).**



☑ Save as file

File name template (1): __DB__-%Y%m%d (☑ remember template)

Compression: ● None ○ "zipped" ○ "gzipped"

Figure 5-6: The phpMyAdmin Save as File section of the export page.

9. **Select the Save As File check box.**

10. **Specify the filename.**

    The File Name Template field contains __DB__, which saves the file with the database name. You can add text or special characters to the filename to make a more meaningful filename. In this case, I added %Y%m%d, which adds the current date to the filename of the exported file.

11. **Select the Remember Template check box.**
12. **Next to Compression, select the None radio button.**
13. **Click Go.**

    Your browser's Save File window opens. You see the name of the file being saved.

14. **Select the option to save your file to disk and click OK.**

    The file is saved where your browser saves files. If you have your browser set to ask you where to save files, a window opens, and you can navigate to the directory where you want to save the file.

    In this example, a file named `jvalade_PetStore-20090520` is saved on my local computer.

Now you have a backup copy of your database. You can save the text file on your Web host, on your local computer, on your neighbor's computer, and as many other places that make you feel that your data is safe. You can then re-create your database easily from this file on any computer that has MySQL installed.

Viewing the Export file

The file exported by the phpMyAdmin Export feature is a text file that contains the SQL queries needed to re-create the database, exactly as it was when you exported it. It contains a `CREATE` query for each table in the database. It contains `INSERT` queries for every row of data in the tables.

The following is the export file that contains the queries needed to re-create two tables: `Member` and `Login`.

```
-- phpMyAdmin SQL Dump
-- version 2.11.9.5
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Generation Time: May 22, 2009 at 03:28 PM
-- Server version: 5.1.30
-- PHP Version: 5.2.5
```

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

--
-- Database: `jvalade_PetStore`
--

-- --------------------------------------------------------

--
-- Table structure for table `Login`
--

DROP TABLE IF EXISTS `Login`;
CREATE TABLE IF NOT EXISTS `Login` (
  `loginName` varchar(20) NOT NULL,
  `loginTime` datetime NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `Login`
--


-- --------------------------------------------------------

--
-- Table structure for table `Member`
--

DROP TABLE IF EXISTS `Member`;
CREATE TABLE IF NOT EXISTS `Member` (
  `loginName` varchar(20) NOT NULL,
  `password` varchar(255) NOT NULL,
  `createDate` date NOT NULL,
  `lastName` varchar(50) NOT NULL,
  `firstName` varchar(40) NOT NULL,
  `street` varchar(50) NOT NULL,
  `city` varchar(50) NOT NULL,
  `state` char(2) NOT NULL,
  `zip` char(10) NOT NULL,
  `email` varchar(50) NOT NULL,
  `phone` varchar(15) NOT NULL,
  `fax` varchar(15) NOT NULL,
  PRIMARY KEY (`loginName`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `Member`
--
```

```
INSERT INTO `Member` (`loginName`, `password`,
           `createDate`, `lastName`, `firstName`,
           `street`, `city`, `state`, `zip`, `email`,
           `phone`, `fax`) VALUES
('joey', 'secret', '2009-05-12', 'Customer', 'Joe', '1234
           Oak St', 'Here', 'CA', '12345-1234', 'me@home.
           com', '888-888-8888', ''),
('sammy', 'secret', '2009-05-22', 'Customer', 'Sam', '123
           Pine St', 'New York', 'NY', '54321-4321', 'sam@
           customer.com', '888-888-8888', '');
```

Notice the final section for each table is `Dumping data for table` *`tablename`*. For the first table, `Login`, this section contains no `INSERT` queries because the table is empty. For the `Member` table, the dump section contains an `INSERT` query that inserts three rows.

TIP

If for some reason you're unable to use phpMyAdmin to back up your database, you can create the same text file using the mysqldump program. The mysqldump program was installed automatically when MySQL was installed. Instructions for using the mysqldump program are provided in the MySQL online documentation, such as `http://dev.mysql.com/doc/refman/5.1/en/mysqldump.html` for MySQL 5.1.

Restoring Your Data

In the preceding section, you find out how to create a backup copy of your database. You saved the SQL queries necessary to re-create your database into a text file. You can re-create your database on any computer that has MySQL installed from the backup file you saved. You can replace your database or move your database onto a new computer where it doesn't currently exist.

You may want to replace a database because a table has become damaged and unusable. It's unusual, but it happens. For instance, a hardware problem or an unexpected shutdown of the computer can cause corrupted tables. Sometimes an anomaly in the data that confuses MySQL can cause corrupt tables. In some cases, a corrupt table can cause your MySQL server to shut down.

Here's a typical error message that signals a corrupted table:

```
Incorrect key file for table: 'tablename'.
```

You can replace the corrupted table(s) with the data stored in a backup copy. In some cases, the database might be lost completely. For instance, if the computer where your database resides breaks down and can't be fixed, your current database is lost, but your data isn't gone forever. You can replace the broken computer with a new computer and restore your database from a backup copy.

You may want to re-create the database on a different computer where it doesn't currently exist. For instance, you may want to copy the database from one Web host account to another if you're changing hosting companies. Or, you may want to replace the database on your local development computer with the most recent database from your Web hosting account, so that you're testing your scripts on the latest customer data.

You can use the text file that you created in the preceding section to re-create the database. However, as described previously, you build a database by creating the database and then adding tables to the database. The backup file contains all the SQL statements necessary to rebuild the tables, but it does not contain the statements needed to create the database. Your database must exist before you can re-create the tables from the backup file.

You can re-create the database from the backup file with the `IMPORT` feature of phpMyAdmin by following these steps:

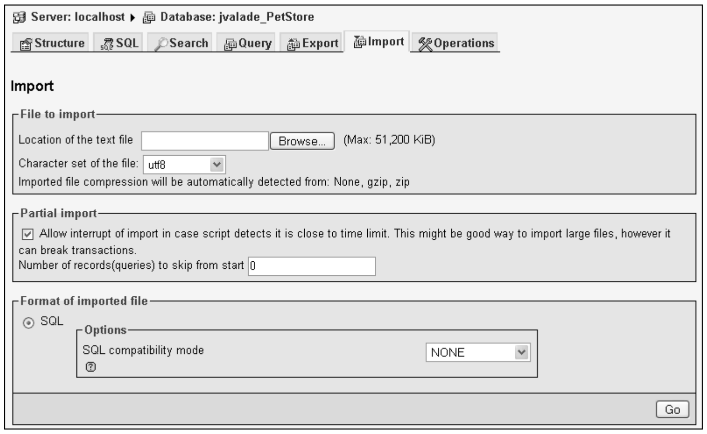
1. **Open the main phpMyAdmin page.**
2. **Click the name of the database you want to re-create.**

   If the database doesn't exist, you need to create it before proceeding. Creating an empty database is described earlier in this chapter.

   The Database page opens, as shown earlier in Figure 5-4.
3. **Click the Import tab at the top of the page.**

   The Import page opens, as shown in Figure 5-7.
4. **Click the Browse button and navigate to the file that you exported.**
5. **In the Format section, select the SQL radio button if it isn't already selected.**
6. **Click the Go button.**

   A new page appears with a message stating that your import was successful.



Server: localhost ▸ Database: jvalade_PetStore

Structure | SQL | Search | Query | Export | Import | Operations

Import

File to import

Location of the text file [ ] Browse... (Max: 51,200 KiB)

Character set of the file: utf8

Imported file compression will be automatically detected from: None, gzip, zip

Partial import

☑ Allow interrupt of import in case script detects it is close to time limit. This might be good way to import large files, however it can break transactions.

Number of records(queries) to skip from start 0

Format of imported file

◉ SQL

Options

SQL compatibility mode NONE

Go

Figure 5-7: The phpMyAdmin Import page.

In some cases, you may want to replace only part of the database. For instance, the backup file created in the previous section contains two tables: `Member` and `Login`. If only the `Login` table is damaged, you want to replace only the `Login` table.

Your database is now restored with all the data that was in it at the time the copy was made. If the data has changed since the copy was made, the changes are lost. For instance, if more data was added after the backup copy was made, the new data is not restored. If you know the changes that were made, you can make them manually in the restored database.

You can control which data is replaced by editing the backup file. Because the backup is a text file, you can edit it with any text editor. Remove any SQL queries that you do not want to execute. For instance, if you do not want to restore the `Member` table, only the `Login` table, remove all the SQL queries from the file that `CREATE` or `INSERT INTO` the `Member` table.