CHAPTER 7

Working with MySQL

Database Concepts

Name	Address	Phone Number
Davis, Michele	7505 N. Linksway FxPnt 53217	414-352-4818
Meyer, Simon	5802 Beard Avenue S 55419	612-925-6897
Phillips, Jon	4204 Zenith Avenue S 55416	612-924-8020
Phillips, Peter	6200 Bayard Avenue HgldPk 55411	651-668-2251

Figure 7-9. Phone book record and fields

Structured Query Language

Creating Tables

Example 7-1. Creating the books and authors tables

```
CREATE TABLE books (
title_id INT NOT NULL AUTO_INCREMENT,
title VARCHAR (150),
pages INT,
PRIMARY KEY (title_id));

CREATE TABLE authors (
author_id INT NOT NULL AUTO_INCREMENT,
title_id INT NOT NULL,
author VARCHAR (125),
PRIMARY KEY (author_id));
```

```
DESCRIBE books;
```

which returns:

```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| title_id | int(11)      | NO   | PRI | NULL    | auto_increment |
| title    | varchar(150) | YES  |     | NULL    |                |
| pages    | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

```
DESCRIBE authors;
```

returns:

```
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| author_id | int(11)      | NO   | PRI | NULL    | auto_increment |
| title_id  | int(11)      | NO   |     |         |                |
| author    | varchar(125) | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

Adding Data to a Table

For example:

```
INSERT INTO books VALUES (1,"Linux in a Nutshell",112);
INSERT INTO authors VALUES (NULL,1,"Ellen Siever");
INSERT INTO authors VALUES (NULL,1,"Aaron Weber");
```

As long as there were no errors, you should get:

```
mysql> INSERT INTO books VALUES (1,"Linux in a Nutshell",112);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO authors VALUES (NULL,1,"Ellen Siever");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO authors VALUES (NULL,1,"Aaron Weber");
Query OK, 1 row affected (0.00 sec)
```

Likewise, we add the other book:

```
INSERT INTO books VALUES (2,"Classic Shell Scripting",256);
INSERT INTO authors VALUES (NULL,2,"Arnold Robbins");
INSERT INTO authors VALUES (NULL,2,"Nelson Beebe");
```

Table Definition Manipulation

Renaming a table

To rename a table, use `ALTER TABLE` *`table`* `RENAME` *`newtable`*. In this example, we are renaming the table from books to publications:

```
ALTER TABLE books RENAME publications;
```

Changing a column's data type

To change a column data type, use `ALTER TABLE` *`table`* `MODIFY` *`column datatype`*. The following syntax modifies the author field so that the column can take 150 characters:

```
ALTER TABLE authors MODIFY author VARCHAR(150);
```

Adding a column

To add a column, use `ALTER TABLE` *`table`* `ADD` *`column datatype`*. Here, we're changing the publications table so a timestamp is automatically added to it.

```
ALTER TABLE publications ADD time TIMESTAMP;
```

Renaming a column

To rename a column, use `ALTER TABLE` *`table new_column_name old_column_name definition new_column`*. Here, we're renaming the author column to author_name. You can also change the definition of the column at the same time. Even if you're not changing the column definition, you still need to include the definition:

```
ALTER TABLE authors CHANGE author author_name varchar(125);
```

Removing a column

If you look at your database tables and decide you don't need a specific column, you can remove it. To remove a column, use `ALTER TABLE` *`table`* `DROP` *`column`*. Here, we're removing the pages column; therefore, we'll no longer know how many pages are in a book listed in the database:

```
ALTER TABLE publications DROP COLUMN pages;
```

Deleting an entire table

Sometimes you may want to completely remove a table. Use the `DROP` command to permanently remove a table and its data:

```
DROP TABLE test_table;
```

Querying the Database

```
SELECT * FROM books;
```

This displays the following:

```
+----------+-------------------------+-------+
| title_id | title                   | pages |
+----------+-------------------------+-------+
|        1 | Linux in a Nutshell     |   112 |
|        2 | Classic Shell Scripting |   256 |
+----------+-------------------------+-------+
2 rows in set (0.01 sec)
```

```
SELECT author_id, title_id, author FROM authors;
```

This displays the following:

```
+-----------+----------+------------------+
| author_id | title_id | author           |
+-----------+----------+------------------+
|         1 |        1 | Ellen Siever     |
|         2 |        1 | Aaron Weber      |
|         3 |        2 | Arnold Robbins   |
|         4 |        2 | Nelson Beebe     |
+-----------+----------+------------------+
4 rows in set (0.01 sec)
```

Limit results with WHERE

```
SELECT * FROM books WHERE title = "Classic Shell Scripting";
```

This returns:

```
+----------+-------------------------+-------+
| title_id | title                   | pages |
+----------+-------------------------+-------+
|        2 | Classic Shell Scripting |   256 |
+----------+-------------------------+-------+
1 row in set (0.00 sec)
```

You can also list out just the columns you're interested in from a table by using:

```
SELECT books.pages FROM books WHERE title = "Classic Shell Scripting";
```

This returns:

```
+-------+
| pages |
+-------+
|   256 |
+-------+

1 row in set (0.00 sec)
```

Specifying the order

```
SELECT * FROM authors ORDER BY author;
```

This displays:

```
+-----------+----------+--------------------+
| author_id | title_id | author             |
+-----------+----------+--------------------+
|         2 |        1 | Aaron Weber        |
|         5 |        9 | Alex Martelli      |
|         3 |        2 | Arnold Robbins     |
|         1 |        1 | Ellen Siever       |
|         4 |        2 | Nelson Beebe       |
+-----------+----------+--------------------+
```

Joining tables together

Example 7-3. The SQL to create and populate a purchases table that links user_ids and title_ids to a purchase_id

```
CREATE TABLE purchases (
purchase_id int NOT NULL AUTO_INCREMENT,
user_id varchar(10) NOT NULL,
title_id int(11) NOT NULL,
purchased timestamp NOT NULL default CURRENT_TIMESTAMP,
PRIMARY KEY  (purchase_id));
INSERT INTO `purchases` VALUES (1, 'mdavis', 2, '2005-11-26 17:04:29');
INSERT INTO `purchases` VALUES (2, 'mdavis', 1, '2005-11-26 17:05:58');
SELECT * FROM purchases;
+-------------+---------+----------+---------------------+
| purchase_id | user_id | title_id | purchased           |
+-------------+---------+----------+---------------------+
|           1 | mdavis  |        2 | 2005-11-26 17:04:29 |
|           2 | mdavis  |        1 | 2005-11-26 17:05:58 |
+-------------+---------+----------+---------------------+
2 rows in set (0.00 sec)
```

```
SELECT books.*, author FROM books, authors WHERE books.title_id = authors.title_id;
```

which produces:

```
+----------+-------------------------+-------+-------------------+
| title_id | title                   | pages | author            |
+----------+-------------------------+-------+-------------------+
|        1 | Linux in a Nutshell     |   112 | Ellen Siever      |
|        1 | Linux in a Nutshell     |   112 | Aaron Weber       |
|        2 | Classic Shell Scripting |   256 | Arnold Robbins    |
|        2 | Classic Shell Scripting |   256 | Nelson Beebe      |
+----------+-------------------------+-------+-------------------+
4 rows in set (0.00 sec)
```

Aliases

```
SELECT * FROM books AS b,authors AS a WHERE b.title_id = a.title_id;
```

results in the following:

```
+----------+-------------------------+-------+-----------+----------+-------------+
| title_id | title                   | pages | author_id | title_id | author
        |
+----------+-------------------------+-------+-----------+----------+-------------+
|        1 | Linux in a Nutshell     |   112 |         1 |        1 | Ellen Siever
|        1 | Linux in a Nutshell     |   112 |         2 |        1 | Aaron Weber
|        2 | Classic Shell Scripting |   256 |         3 |        2 | Arnold Robbins
|        2 | Classic Shell Scripting |   256 |         4 |        2 | Nelson Beebe
+----------+-------------------------+-------+-----------+----------+-------------+
4 rows in set (0.00 sec)
```

Modifying Database Data

For example, this is how you'd update the books table:

```
UPDATE books SET pages = 476 WHERE title = "Linux in a Nutshell";
```

The example returns:

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

SELECT * FROM books;
```

This returns:

```
+----------+-------------------------+-------+
| title_id | title                   | pages |
+----------+-------------------------+-------+
|        1 | Linux in a Nutshell     |   476 |
|        2 | Classic Shell Scripting |   256 |
+----------+-------------------------+-------+
2 rows in set (0.00 sec)
```

Deleting Database Data

In this example, only Ellen Siever's book is deleted from the database:

```
DELETE FROM books WHERE author_id = 1;
```

Search Functions

For example, to do a general search, you would use the following syntax:

```
SELECT * FROM authors WHERE author LIKE "%b%";
```

This statement returns:

```
+-----------+----------+-------------------+
| author_id | title_id | author            |
+-----------+----------+-------------------+
|         2 |        1 | Aaron Weber       |
|         3 |        2 | Arnold Robbins    |
|         4 |        2 | Nelson Beebe      |
+-----------+----------+-------------------+
3 rows in set (0.00 sec)
```

Another wildcard character is the _ character. It will match exactly one character. Following is how to use a literal wildcard character in your searches:

```
SELECT * FROM authors WHERE author LIKE "Aaron Webe_"
```

Logical Operators

You can use AND, OR, and NOT in your query's WHERE clause:

```
SELECT * FROM authors WHERE NOT (author = "Ellen Siever" );
```

This query returns book and author information from the following code:

```
SELECT *
  FROM books, authors
 WHERE title = "Linux in a Nutshell"
   AND author = "Aaron Weber"
   AND books.title_id = authors.title_id;
```

This query returns all records with author names of either Aaron Weber or Ellen Siever:

```
SELECT *
  FROM books, authors
 WHERE (author = "Aaron Weber"
    OR  author = "Ellen Siever")
   AND books.title_id=authors.title_id
```

CHAPTER 9

Getting PHP to Talk to MySQL

Querying the Database with PHP Functions

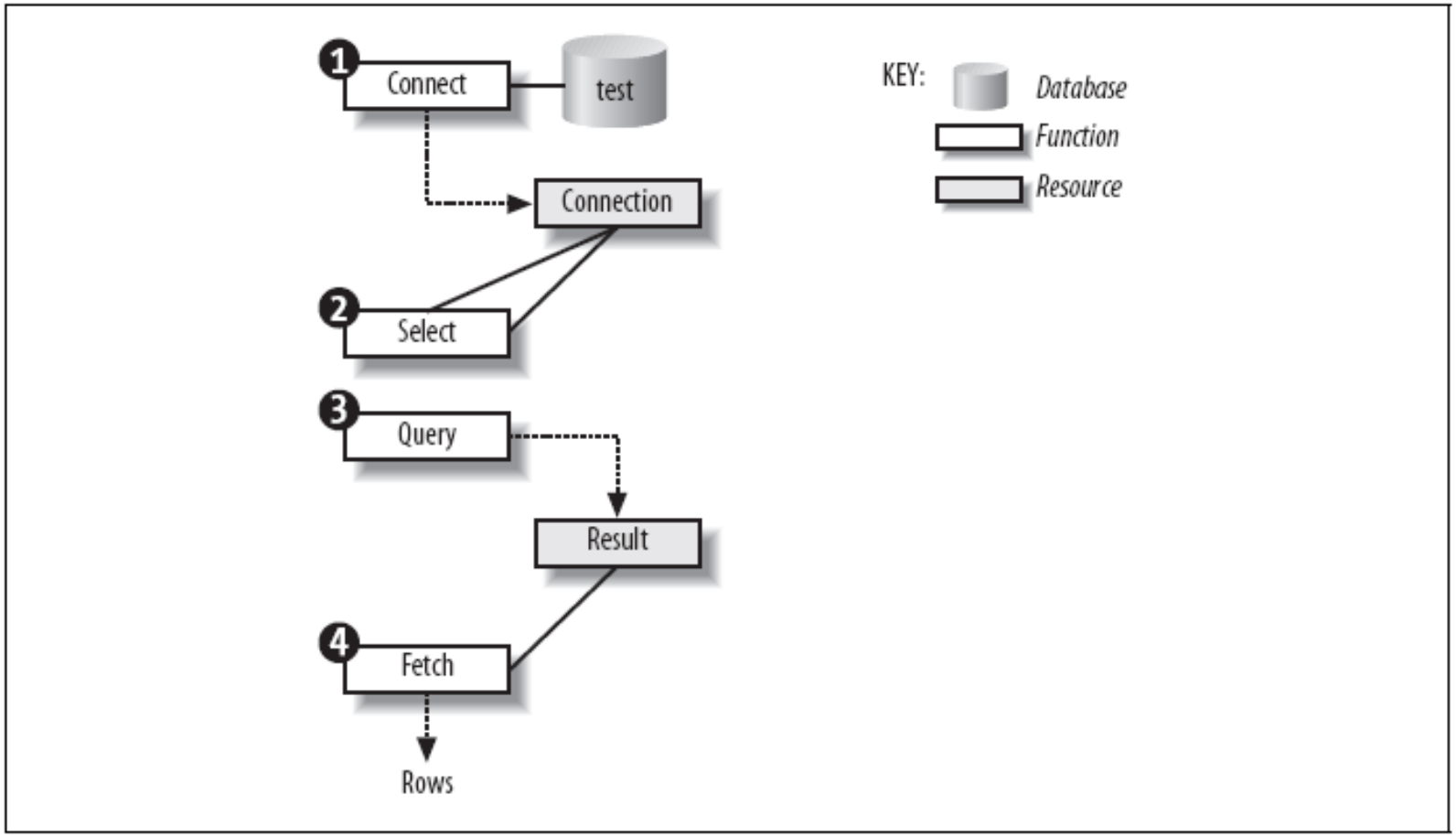

Figure 9-1. The interaction between functions and resources when using the database

Including Database Login Details

Example 9-1. A template for setting database login settings

```
<?php
$db_host='hostname of database server';
$db_database='database name';
$db_username='username';
$db_password='password';
?>
```

Example 9-2. The db_login.php file with sample values filled in

```
<?php
$db_host='localhost';
$db_database='test';
$db_username='test';
$db_password='yourpass';
?>
```

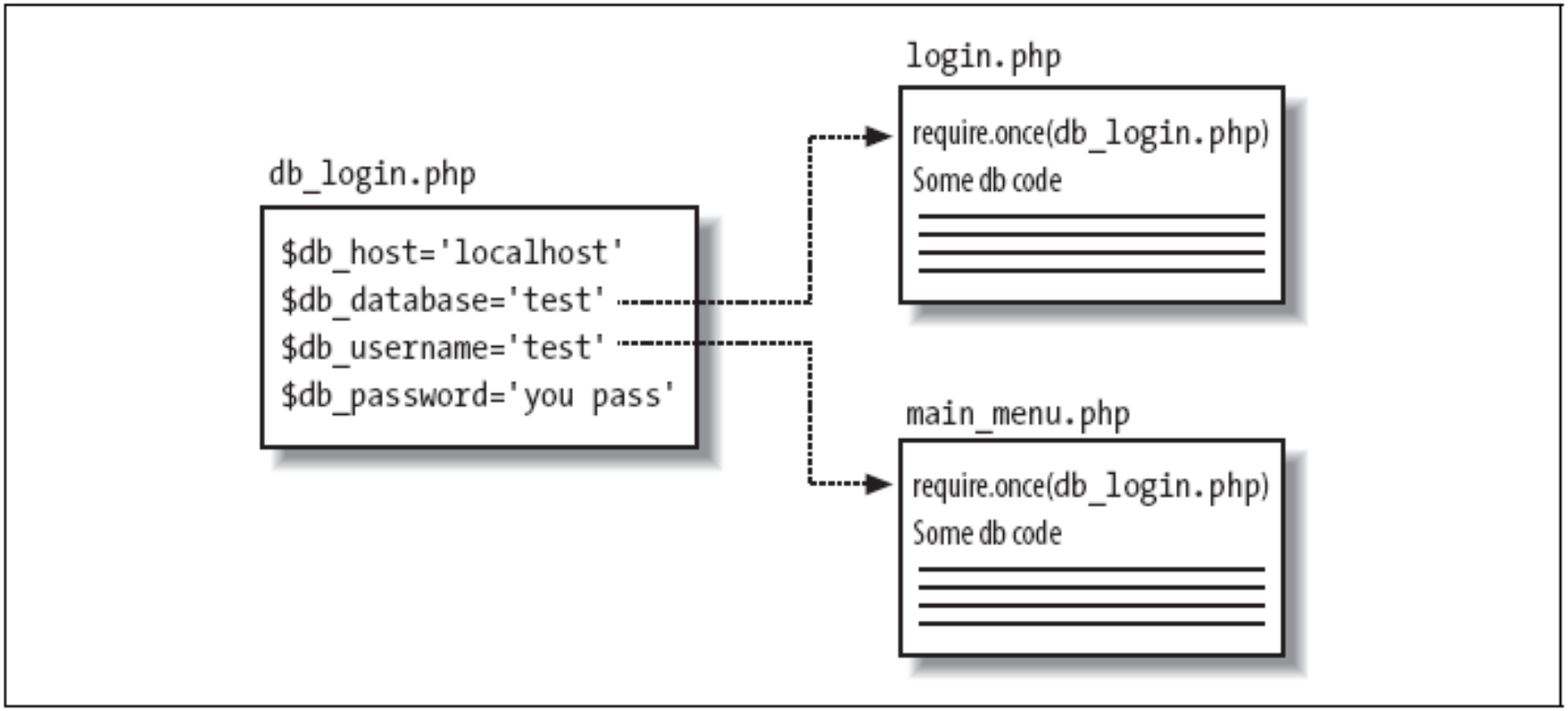

Figure 9-2. Reusing the login details in multiple files

Example 9-5. Displaying the books and authors

```
<?php
// Include our login information
include('db_login.php');
// Connect
$connection = mysql_connect( $db_host, $db_username, $db_password );
if (!$connection){
   die ("Could not connect to the database: <br />". mysql_error( ));
}
// Select the database
$db_select=mysql_select_db($db_database);
if (!$db_select){
   die ("Could not select the database: <br />". mysql_error( ));
}
```

Example 9-5. Displaying the books and authors (continued)

```
// Assign the query
$query = "SELECT * FROM books NATURAL JOIN authors";
// Execute the query
$result = mysql_query( $query );
if (!$result){
   die ("Could not query the database: <br />". mysql_error( ));
}

// Fetch and display the results
while ($result_row = mysql_fetch_row(($result))){
       echo 'Title: '.$result_row[1] . '<br />';
       echo 'Author: '.$result_row[4] . '<br /> ';
       echo 'Pages: '.$result_row[2] . '<br /><br />';
}
/ /Close the connection
mysql_close($connection);
?>
```

Here's HTML markup output from Example 9-5:

```
Title: Linux in a Nutshell<br />Author: Ellen Siever<br /> Pages: 476<br />
<br />Title: Linux in a Nutshell<br />Author: Aaron Weber<br /> Pages: 476<br />
<br />Title: Classic Shell Scripting<br />Author: Arnold Robbins<br /> Pages: 256<br />
<br />Title: Classic Shell Scripting<br />Author: Nelson H.F. Beebe<br /> Pages:
256<br /><br />
```

This displays in your browser as in Figure 9-3.

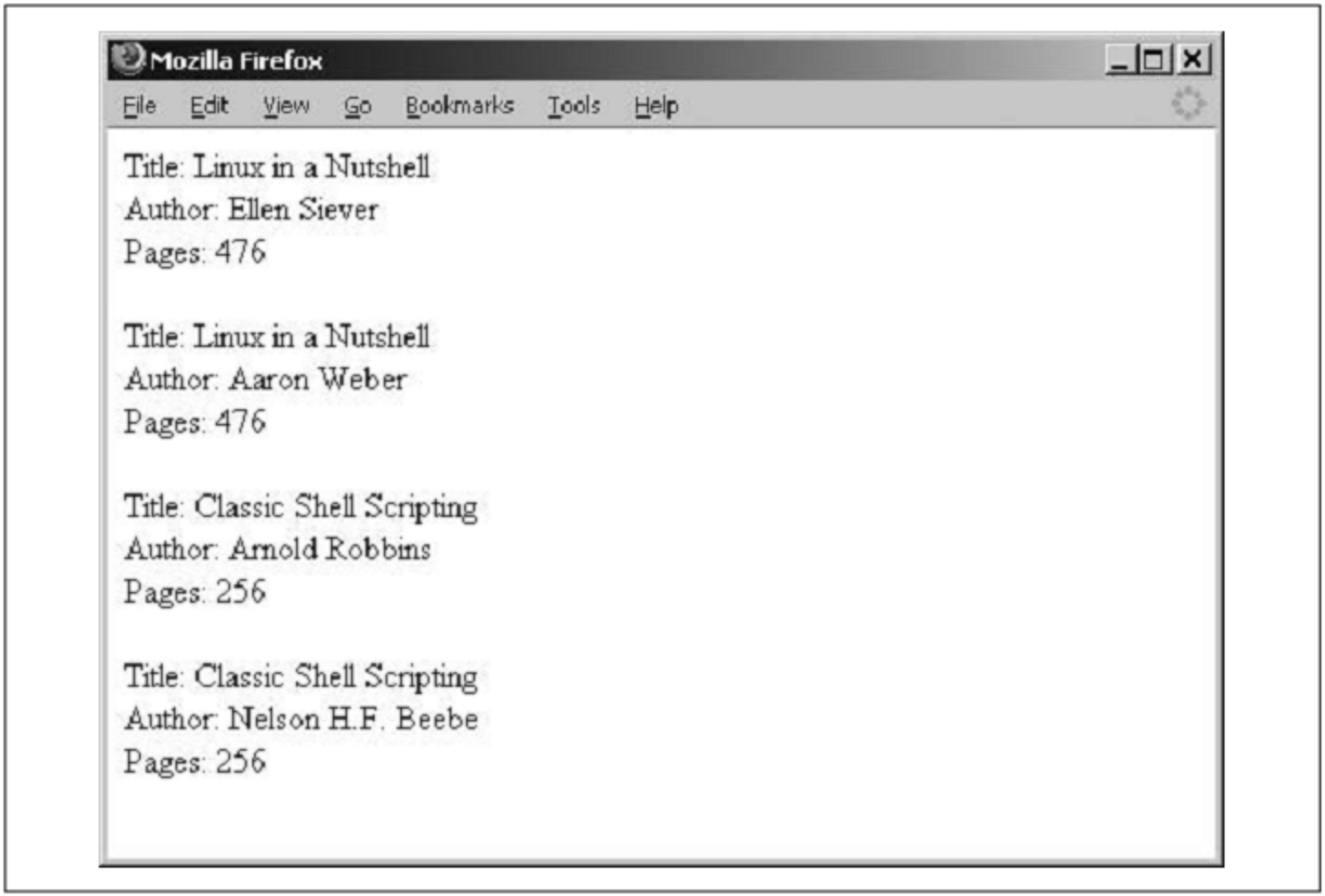

Figure 9-3. How Example 9-5 displays in the browser

Example 9-6. Displaying the results of a query in an HTML table

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Displaying in an HTML table</title>
</head>
<body>
<table border="1">
    <tr>
        <th>Title</th>
        <th>Author</th>
    <th>Pages</th>
</tr>
```

```php
<?php
//Include our login information
include('db_login.php');
// Connect
$connection = mysql_connect($db_host, $db_username, $db_password);
if (!$connection){
    die("Could not connect to the database: <br />". mysql_error( ));
}
// Select the database
$db_select = mysql_select_db($db_database);
if (!$db_select){
    die ("Could not select the database: <br />". mysql_error( ));
}
// Assign the query
$query = "SELECT * FROM books NATURAL JOIN authors";
// Execute the query
$result = mysql_query($query);
if (!$result){
    die ("Could not query the database: <br />". mysql_error( ));
}
```

```
// Fetch and display the results
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)){
    $title = $row["title"];
    $author = $row["author"];
    $pages = $row["pages"];
    echo "<tr>";
    echo "<td>$title</td>";
    echo "<td>$author</td>";
    echo "<td>$pages</td>";
    echo "</tr>";
}

// Close the connection
mysql_close($connection);
?>
</table>
</body>
</html>
```

Example 9-6 displays in your browser as shown in Figure 9-4.

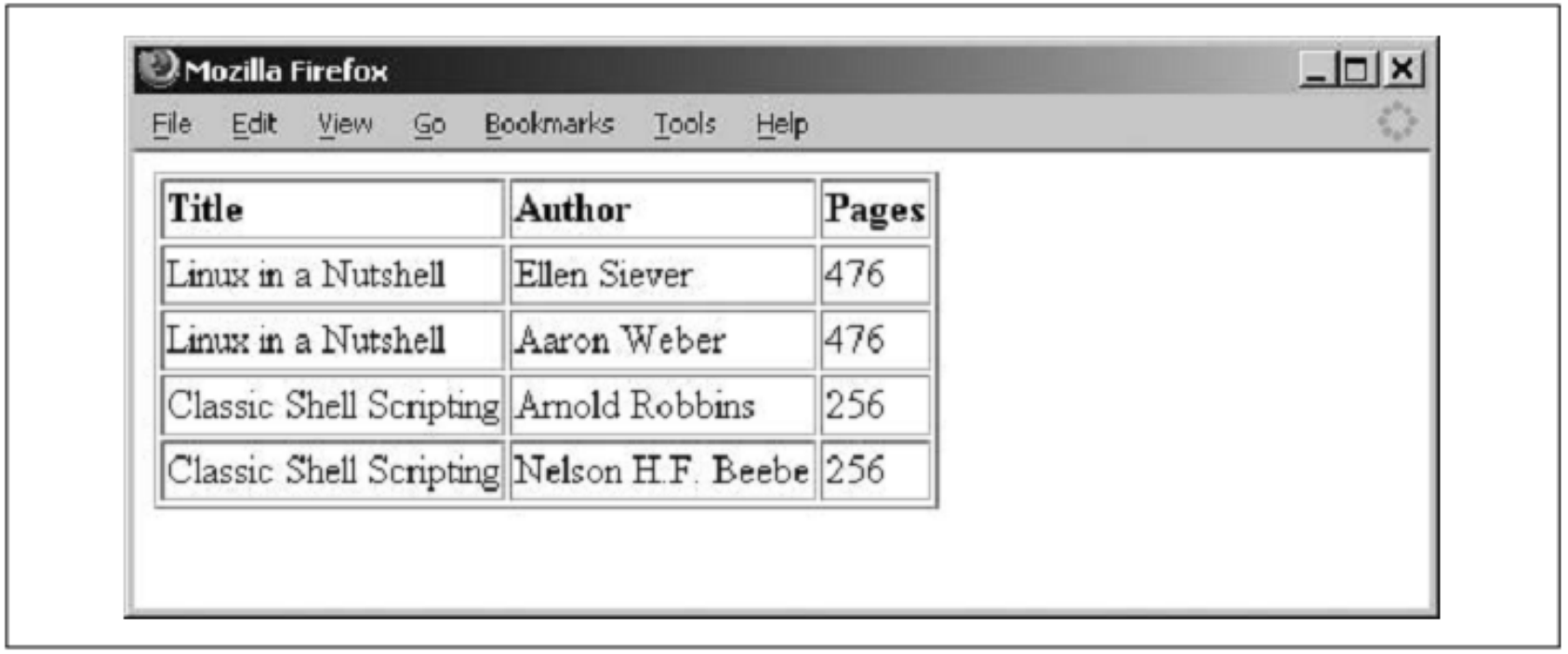

Figure 9-4. The same data in an HTML table

Using PEAR

PEAR is a framework and distribution system for reusable PHP components, creating a collection of add-on functionalities for PHP development. There are many modules

Table 9-1. PEAR modules categories

Authentication	HTML	Processing
Benchmarking	HTTP	Science
Caching	Images	Semantic Web
Configuration	Internationalization	Streams
Console	Logging	Structures

Table 9-1. PEAR modules categories (continued)

Database	Mail	System
Date/Time	Math	Test
Encryption	Networking	Tools and utilities
Event	Numbers	Validate
File formats	Payment	Web services
File system	PEAR	XML
GTK components	PHP	

Rewriting the Books Example with PEAR

Example 9-7. Displaying the books table with PEAR DB

```
<?php

include('db_login.php');
require_once('DB.php');

$connection = DB::connect("mysql://$db_username:$db_password@$db_host/$db_database");

if (DB::isError($connection)){
    die("Could not connect to the database: <br />".DB::errorMessage($connection));
}

$query = "SELECT * FROM books NATURAL JOIN authors";
$result = $connection->query($query);

```

Example 9-7. Displaying the books table with PEAR DB (continued)

```
15 if (DB::isError($result)){
16     die("Could not query the database:<br />$query ".DB::errorMessage($result));
17 }
18
19 echo('<table border="1">');
20 echo '<tr><th>Title</th><th>Author</th><th>Pages</th></tr>';
21
22 while ($result_row = $result->fetchRow( )) {
23     echo "<tr><td>";
24     echo $result_row[1] . '</td><td>';
25     echo $result_row[4] . '</td><td>';
26     echo $result_row[2] . '</td></tr>';
27 }
28
29 echo("</table>");
30 $connection->disconnect( );
31
32 ?>
```

Example 9-8. Displaying the books table with PEAR:: MDB2

```
<?php

include('db_login.php');
require_once('MDB2.php');

//Translate our database login information into an array.
$dsn = array(
    'phptype'  => 'mysql',
    'username' => $username,
    'password' => $password,
    'hostspec' => $host,
    'database' => $database
    );

//Create the connection as an MDB2 instance.
$mdb2 = MDB2::factory($dsn);
if (PEAR::isError($mdb2)) {
    die($mdb2->getMessage( ));
}

//Set the fetchmode to field associative.
$mdb2->setFetchMode(MDB2_FETCHMODE_ASSOC);
```

```
$query = "SELECT * FROM books NATURAL JOIN authors";
$result =$mdb2->query($query);
if (PEAR::isError($result)){
    die("Could not query the database:<br />$query ".$result->getMessage( ));
}

//Display the results.
echo('<table border="1">');
echo '<tr><th>Title</th><th>Author</th><th>Pages</th></tr>';

//Loop through the result set.
while ($row = $result->fetchRow( )) {
    echo "<tr><td>";
    echo htmlentities($row['title']) . '</td><td>';
    echo htmlentities($row['author ']) . '</td><td>';
    echo htmlentities($row['pages']) . '</td></tr>';
}

echo("</table>");

//Close the connection.
$result->free( );
?>
```