CHAPTER

Working with Databases

Connecting PHP to databases on the server is a natural, and this chapter discusses that connection. You're going to see how to connect PHP to database tables and the like in this chapter, updating data as you want in that database.

PHP comes with a lot of support for databases built in, and that's good, because one of the most popular ways to work with PHP is to handle databases on the server. You can see the databases that PHP supports in Table 10-1.

If you want to use the built-in PHP support for the various database servers in Table 10-1, you can find the manuals for them at www.php.net/*dbname*, where *dbname* is the database name, like mysql, sybase, mssql, and so on. For ODBC, use the name uodbc; for Oracle, oci8.

By far, the most frequently used database system with PHP is MySQL, so this chapter focuses on MySQL, which you can get for free from www.mysql.com, although other database servers are discussed as well. All these databases support PHP, and the only difference between them, for the most part, is how you connect to them—once you know how to work with MySQL, the process is easily generalized to other databases.

Adabas	Ingres	Oracle
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro	mSQL	Solid
Hyperwave Direct	MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

TABLE 10-1 PHP-Supported Databases

What Is a Database?

So, just what is a database? We'll take a look at what makes a database here briefly (and if you are already familiar with databases, tables, and so on, you can naturally skip this intro).

Databases organize data for easy access and use by programs. The most popular database construct is the table, and we'll take a look at tables here. Say, for example, that you're teaching PHP to a class of students and want to keep track of their scores. You might create a table with two columns, Name and Grade:

```
Name  Grade
```

You can store the name of the first student in the Name column, like this:

```
Name  Grade
Ann
```

This creates a table entry for Ann—that is, a new row. Each row in a database table is a *record*, and this record is for the student named Ann. Each column in a record is called a *field*, and you've given the Name field the value "Ann". Similarly, you can give Ann a grade in the Grade field:

```
Name  Grade
Ann   C
```

And you can add records for other students as well:

```
Name  Grade
Ann   C
Mark  B
Ed    A
Frank A
Ted   A
Mabel B
Ralph B
Tom   B
```

What, then, is a database? In its most conventional form, a database is just a collection of one or more tables. And to access the data in those tables, you use SQL in PHP, which is coming up in the next topic.

Some Essential SQL

To interact with databases in PHP, you use Structured Query Language, SQL. We're going to take a look at some SQL here (note that the full SQL language is beyond the score of this book, which focuses on connecting to databases and so on in PHP—the SQL you use to work with your database is up to you).

For example, say that you had a database table named fruit, and you wanted to recover the records from that table. Here's what might be in that table at present:

Name	Number
apples	1020
oranges	3329
bananas	8582
pears	235

To work with this table in your code, you can execute an SQL statement, called an *SQL query*, on the table. This query will return all the records in the table:

```
SELECT * FROM FRUIT
```

Executing this query gives you a record set containing all matching records from the fruit table. Because you specified that you wanted to match the wildcard * here, all records in the fruit table are returned by this query. In PHP, that record set is returned as an array, and you can loop over that array in ways you're going to see in this chapter.

How's this look in PHP? How do you actually execute an SQL query in PHP? To interact with MySQL, you'd use the mysql_query function to execute that SQL query, something like this, which returns the entire fruit table and stores it in $result:

```
$query = "SELECT * FROM fruit";
$result = mysql_query($query) or die("Query failed: " . mysql_error());
```

Now $result holds an array with the records for the fruit table, and you can access the fields in each record by name.

How about some more SQL? You can also select specific fields from a table like this, where we're selecting the name and number fields from the fruit table:

```
SELECT name, number FROM fruit
```

Using the WHERE clause, you can set up selection criteria that the records in the record set generated by the query must meet. For example, to select all the records in the fruit table where the name field equals apples, you can execute this statement: SELECT * FROM fruit WHERE name= "apples".

You don't have to use an equal sign here; you can test fields using these operators:

- < (less than)
- <= (less than or equal to)
- > (greater than)
- >= (greater than or equal to)

You can use an IN clause to specify a set of values that fields can match. For example, here's how you can retrieve records that have values in the name field that match apples or oranges:

```
SELECT * FROM fruit WHERE name IN ("apples", "oranges")
```

You can also use logical operations on the clauses in your SQL statements. Here's an example where we're specifying two criteria: the name field must hold either "apples" or "oranges", and there must be some value in the number field. You use the NULL keyword to test if there's anything in a field:

```
SELECT * FROM fruit WHERE name NOT IN ("apples", "oranges") AND number IS NOT NULL
```

You can use these logical operators to connect clauses: AND, OR, and NOT. Using AND means that both clauses must be true, using OR means either one can be true, and using NOT flips the value of a clause from TRUE to FALSE or FALSE to TRUE.

As you might expect, you can also order the records in the record set produced by an SQL statement. Here's an example where we're ordering the records in the fruit table using the name field:

```
SELECT * FROM fruit ORDER BY name
```

You can also sort records in descending order with the DESC keyword:

```
SELECT * FROM fruit ORDER BY name DESC
```

You can use the DELETE statement to delete records like this, where we're removing all records from the fruit table that have name values that are not apples or oranges:

```
DELETE * FROM fruit WHERE name NOT IN ("apples", "oranges")
```

You use the UPDATE statement to update a database when you want to make changes. For example, here's how to change the value of the number field in the record that contains the number of apples:

```
UPDATE fruit SET number = "2006" WHERE name = "apples"
```

You can also insert new data into a table. Here's an example that inserts a new row into the fruit table:

```
INSERT INTO fruit (name, number) VALUES('apricots', '203')
```

Okay, we've gotten as much SQL under our belts as we'll need here. The next step is all about creating a database to work with in PHP.

Creating a MySQL Database

We're using MySQL databases in this chapter, and to have something to work with in code, we're going to create a database in MySQL now. You can get MySQL for free from www.mysql.com—and in fact, your system might already have it installed. To check if it's already installed, try this from a command prompt (remember, % stands for the generic command-line prompt in this book):

```
%mysql
```

If you see a response from MySQL, congratulations, you've already got it installed. Otherwise, you'll need to download and install it. In fact, MySQL used to come with PHP, but it no longer does because of licensing issues.

Depending on your system and MySQL version, you might have to start the MySQL server before working with MySQL. You can start the MySQL server with this command line:

```
$mysqld --console
```

On some systems, such as Windows with recent versions of MySQL, you don't need to start the MySQL server at all—it's already running. In that case, you'll get an error if you try to start the MySQL server a second time; the error will concern shared resources.

Next, you should start a MySQL session that's going to connect to the server, and which you can use to create your own database. You need a username and password to work with MySQL; say your username is "user" and your password is "password". You can start MySQL like this at the command prompt:

```
$mysql -u user -p
```

This will ask you to enter your password:

```
$mysql -u user -p
Enter password: ********
```

Once you've started a session, you'll see a response something like this:

```
$mysql -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
mysql>
```

The last line, mysql>, is a prompt for you to enter your commands.

If you don't have a username and password, enter **mysql -u root**, or just **mysql**, to start:

```
$mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
mysql>
```

To get started, enter **SELECT VERSION(), CURRENT_DATE;** to confirm that MySQL is working:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----------+--------------+
| VERSION() | CURRENT_DATE |
+-----------+--------------+
| 5.0.19-nt | 2007-05-10   |
+-----------+--------------+
1 row in set (0.01 sec)
```

This command gives you the MySQL version and the current date. Note that MySQL commands like this end with a semicolon.

MySQL comes with some databases built in—which you can check with the SHOW DATABASES; command:

```
mysql> SHOW DATABASES;
+----------------------+
| Database             |
+----------------------+
| mysql                |
| test                 |
+----------------------+
2 rows in set (0.08 sec)
```

These databases already exists in MySQL—mysql is a database that MySQL uses for internal administration, and test is a database for test purposes.

Database tables are stored inside databases, so the first step is to create a database. You might use this database to contain information about various fruits and vegetables, for example, and so you can create a database named, say, produce. You can create the produce database with the CREATE DATABASE command in the MySQL monitor:

```
mysql> CREATE DATABASE produce;
Query OK, 1 row affected (0.01 sec)
```

That creates a new, empty database, which you can see with the SHOW DATABASES; command:

```
mysql> CREATE DATABASE produce;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+----------------------+
| Database             |
+----------------------+
| mysql                |
| test                 |
| produce              |
+----------------------+
3 rows in set (0.08 sec)
```

Now make the produce database the default database with the USE command like this in MySQL:

```
mysql> CREATE DATABASE produce;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+----------------------+
| Database             |
+----------------------+
| mysql                |
| test                 |
| produce              |
+----------------------+
3 rows in set (0.08 sec)
```

```
mysql> USE produce
Database changed
```

Are there any tables in this new database? You can use the SHOW TABLES command to check that out:

```
mysql> USE produce
Database changed

mysql> SHOW TABLES;
Empty set (0.01 sec)
```

This response—"Empty set"—tells you that this database doesn't contain any tables yet.

Creating a New Table

You can change that by creating a new table, say fruits, to contain various fruits. To create a database table, you have to create the various fields in that table, which means setting their data format. Here are a few of the most popular data formats:

- **VARCHAR(*length*)** Creates a variable-length string
- **INT** Creates an integer
- **DECIMAL(totaldigits, decimalplaces)** Creates a decimal value
- **DATETIME** Creates a date and time object, such as 2008-11-15 20:00:00

Records in the fruit table will contain two 20-character strings—the name of a fruit, and the number of that fruit on hand. Here's how you create the fruit table in the MySQL monitor:

```
mysql> CREATE TABLE fruit (name VARCHAR(20), number VARCHAR(20));
Query OK, 0 rows affected (0.13 sec)
```

That creates the fruit table, which you can check with the SHOW TABLES; command like this:

```
mysql> CREATE TABLE fruit (name VARCHAR(20), number VARCHAR(20));
Query OK, 0 rows affected (0.13 sec)

mysql> SHOW TABLES;
+------------------+
| Tables_in_produce |
+------------------+
| fruit            |
+------------------+
1 row in set (0.00 sec)
```

In fact, you can check on this table, getting the format of its fields with the DESCRIBE SQL command like this:

```
mysql> CREATE TABLE fruit (name VARCHAR(20), number VARCHAR(20));
Query OK, 0 rows affected (0.13 sec)

mysql> SHOW TABLES;
+-------------------+
| Tables_in_produce |
+-------------------+
| fruit             |
+-------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE fruit;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| name   | varchar(20) | YES  |     | NULL    |       |
| number | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Cool—now you've got a new database table, fruits. Time to stock it with some data.

Putting Data into the New Database

Say that you're keeping track of the fruit inventory at your local grocery store. You count this many fruits:

- apples 1020
- oranges 3329
- bananas 8582
- pears 235

Alright, how about storing that in the new fruit database table? You've got two fields in each record in the fruit table, as you can see in the MySQL description of the table:

```
mysql> DESCRIBE fruit;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| name   | varchar(20) | YES  |     | NULL    |       |
| number | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

So now you can insert two strings into the record for each kind of fruit—the name of the fruit, and the number of that fruit on hand. You can create records in the fruit table with the SQL INSERT command. For example, to insert a record for the number of apples on hand, you'd execute this command:

```
mysql> INSERT INTO fruit VALUES ('apples', '1020');
Query OK, 1 row affected (0.00 sec)
```

Great, that creates a new record for apples. You can check that new record with the SELECT * FROM fruit; command:

```
mysql> INSERT INTO fruit VALUES ('apples', '1020');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM fruit;
+--------+--------+
| name   | number |
+--------+--------+
| apples | 1020   |
+--------+--------+
1 row in set (0.00 sec)
```

As you can see, there is a new record for apples here.

Here's how you add the other fruits with INSERT statements:

```
mysql> INSERT INTO fruit VALUES ('apples', '1020');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO fruit VALUES ('oranges', '3329');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO fruit VALUES ('bananas', '8582');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO fruit VALUES ('pears', '235');
Query OK, 1 row affected (0.00 sec)
```

And you can check the new fruit table and its contents with SELECT * FROM fruit; like this:

```
mysql> SELECT * FROM fruit;
+---------+--------+
| name    | number |
+---------+--------+
| apples  | 1020   |
| oranges | 3329   |
| bananas | 8582   |
| pears   | 235    |
+---------+--------+
4 rows in set (0.00 sec)
```

That creates a database, produce, and a table in that database, fruit. You can quit the MySQL monitor like this:

```
mysql> SELECT * FROM fruit;
+---------+--------+
| name    | number |
+---------+--------+
| apples  | 1020   |
| oranges | 3329   |
| bananas | 8582   |
| pears   | 235    |
+---------+--------+
4 rows in set (0.00 sec)

mysql>quit
```

That creates your database—it's time to access it in PHP.

Accessing the Database in PHP

When you install PHP, you can select from a number of extensions. To install support for MySQL, click the Extensions node in the installer to open that node and select the MySQL node to install that extension. Your PHP installation may have already been installed with MySQL support—most installations on Web servers are.

The MySQL support in PHP consists of a number of functions you can call to interact with MySQL, and here they are:

- **mysql_affected_rows** Get the number of rows affected by the previous MySQL operation.
- **mysql_change_user** Change the logged-in user.
- **mysql_client_encoding** Return the name of the current character set.
- **mysql_close** Close a MySQL connection.
- **mysql_connect** Open a connection to a MySQL Server.
- **mysql_create_db** Create a MySQL database.
- **mysql_data_seek** Seek data in the database.
- **mysql_db_name** Get the name of the database.
- **mysql_db_query** Send a MySQL query.
- **mysql_drop_db** Drop (that is, delete) a MySQL database.
- **mysql_error** Return the text of the error message from the previous MySQL operation.
- **mysql_fetch_array** Fetch a result row as an associative array, a numeric array, or both.
- **mysql_fetch_assoc** Fetch a result row as an associative array.

- **mysql_fetch_row** Get a result row as an enumerated array.
- **mysql_field_len** Return the length of a given field.
- **mysql_field_name** Get the name of the given field in a result.
- **mysql_field_seek** Seek to a given field offset.
- **mysql_field_table** Get the name of the table the given field is in.
- **mysql_field_type** Get the type of the given field in a result.
- **mysql_get_server_info** Get MySQL server info.
- **mysql_info** Get information about the most recent query.
- **mysql_list_dbs** List databases available on a MySQL server.
- **mysql_list_fields** List MySQL table fields.
- **mysql_list_tables** List the tables in a MySQL database.
- **mysql_num_fields** Get the number of fields in result.
- **mysql_num_rows** Get the number of rows in result.
- **mysql_pconnect** Open a persistent connection to a MySQL server.
- **mysql_query** Send a MySQL query.
- **mysql_result** Get result data.
- **mysql_select_db** Select a MySQL database.
- **mysql_tablename** Get the table name of a field.

We're going to put these functions to work in this chapter. For example, you might put together an example, phpdatatable.php, which reads in and displays the fruit table from the produce database.

Connecting to the Database Server

PHP connects to databases using *connection* objects. To create a connection object for MySQL, use mysql_connect:

```
mysql_connect ( [server [, username [, password [, new_link [, client_flags]]]]] )
```

Here, *server* is the MySQL server, which can be URLs, port numbers, and so on. The *username* and *password* arguments are the MySQL username and password.

The new_link argument, if set to TRUE, forces PHP to establish a new link to the database, even if it already has such a link. Otherwise, if you try to open a second link to the database by calling mysql_connect a second time, PHP may use its already-established link instead.

The client_flags parameter can be a combination (created by Oring values together with the OR operator, |) of the following constants: MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE, or MYSQL_CLIENT_INTERACTIVE.

The mysql_connect function returns a connection object if successful, FALSE otherwise.

Okay, let's connect to MySQL from PHP using mysql_connect. In this case, MySQL and PHP are on the same machine, so the server is just "localhost". Here's how to create the connection object (fill in your own username and password, of course):

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    .
    .
    .
```

Let's augment this a little, making it display a message if there was an error, and quit, using the die function:

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    or die ("Couldn't connect to server");
    .
    .
    .
```

Connecting to the Database

Okay, we're connected to MySQL; the next step is to select the database you want to use. You select the database with the mysql_select_db, which works like this:

```
mysql_select_db (database_name [, link_identifier] )
```

Here, *database_name* is the name of the database, which is "produce" here, and *link_identifier* is the connection object. Here's how to put mysql_select_db to work in phpdatatable.php:

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce", $connection)
    or die ("Couldn't select database");
    .
    .
    .
?>
```

Great—you've connected to the database server, and you've selected the database you want to work with. Selecting the database is crucial—if you don't select a database before proceeding, you'll get errors.

Reading the Table

The goal of the phpdatatable.php example is to read and display the database table named fruit. To get that table returned from the MySQL server, you can send that server an SQL query using the mysql_query function:

```
mysql_query (query [, link_identifier] )
```


Here, *query* is the SQL query you want to send to the database server, and *link_identifier* is the connection object, representing the connection to that server; if you only have one connection open, PHP will use that connection.

For SELECT, SHOW, DESCRIBE, EXPLAIN, and other statements returning table data, mysql_query returns a resource on success, or FALSE on error. For other type of SQL statements, UPDATE, DELETE, DROP, and so on, mysql_query returns TRUE on success or FALSE on error. The returned result resource should be passed to mysql_fetch_array, and other functions for dealing with result tables, to access the returned data. And you can use mysql_num_rows to find out how many rows were returned for a SELECT statement.

The SQL for getting all records from the fruit table is "SELECT * FROM fruit", so that'll be our query:

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "SELECT * FROM fruit";
    .
    .
    .
?>
```

Now you can get a returned data table full of rows from the database like this with mysql_query:

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "SELECT * FROM fruit";

  $result = mysql_query($query)
    .
    .
    .
?>
```

Note that if the SQL query failed, you can end the application and display an error using the mysql_error function:

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    or die ("Couldn't connect to server");
```

```
$db = mysql_select_db("produce",$connection)
  or die ("Couldn't select database");

$query = "SELECT * FROM fruit";

$result = mysql_query($query)
  or die("Query failed: " . mysql_error());


?>
```

Alright, you've made progress. Now it's time to decipher the data you've recovered from the database.

Displaying the Table Data

You can display the data from the fruit table in an HTML table in the Web page returned by this example. The two fields in the fruit table are name and number, so you can start by creating the HTML table with table headers Name and Number:

```
<?php
  $connection = mysql connect("localhost","root","**********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "SELECT * FROM fruit";
  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  echo "<table border='1'>";
  echo "<tr>";
  echo "<th>Name</th><th>Number</th>";
  echo "</tr>";


?>
```

Now it's time to start dealing with the result you got back from your SQL query. That's a PHP resource; it's the fruit table in code form. To extract the records from that table, you can use the mysql_fetch_array function—this function returns an array corresponding to the current record (that is, the current row) in the data table, and you can loop over the records using loops. Here's how to use mysql_fetch_array:

```
mysql_fetch_array (result [  result type] )
```

Here, *result* is the resource returned by the mysql_query function—that's the data table you've recovered from the database. And *result_type* is the type of array that you want.

This value is a constant and can take the following values: MYSQL_ASSOC, MYSQL_NUM, and the default value of MYSQL_BOTH.

You can use mysql_fetch_array to fetch rows from the fruit table, and you might loop over those rows with a while loop. Here's how that looks, where you store each row from the table in a variable named $row:

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "SELECT * FROM fruit";
  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  echo "<table border='1'>";
  echo "<tr>";
  echo "<th>Name</th><th>Number</th>";
  echo "</tr>";

  while ($row = mysql_fetch_array($result))
  {
    .
    .
    .
  }
?>
```

This assigns the current row from the fruit table to $row. You can access the name field in that row like this: $row['name'], and the number field in the row as $row['number'].

That means you can display the data in the current row's two fields, name and number, like this in phpdatatable.php:

```
<?php
  $connection = mysql_connect("localhost","root","**********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "SELECT * FROM fruit";
  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  echo "<table border='1'>";
  echo "<tr>";
  echo "<th>Name</th><th>Number</th>";
  echo "</tr>";
```

You can see how this works in phpdataupdate.php. First, you get a connection to the database server (put in your own username and password, of course):

```
<?php
   $connection = mysql_connect("localhost","root","**********")
     or die ("Couldn't connect to server");
   .
   .
   .
```

Then you select the produce database:

```
<?php
   $connection = mysql_connect("localhost","root","**********")
     or die ("Couldn't connect to server");

   $db = mysql_select_db("produce",$connection)
     or die ("Couldn't select database");
   .
   .
   .
```

So how do you update the number of pears? You can use the UPDATE SQL statement "UPDATE fruit SET number = 234 WHERE name = 'pears'" like this:

```
<?php
   $connection = mysql_connect("localhost","root","**********")
     or die ("Couldn't connect to server");

   $db = mysql_select_db("produce",$connection)
     or die ("Couldn't select database");

   $query = "UPDATE fruit SET number = 234 WHERE name = 'pears'";
   .
   .
   .
```

Now you can execute this SQL with mysql_query:

```
<?php
   $connection = mysql_connect("localhost","root","*********")
     or die ("Couldn't connect to server");

   $db = mysql_select_db("produce",$connection)
     or die ("Couldn't select database");

   $query = "UPDATE fruit SET number = 234 WHERE name = 'pears'";

   $result = mysql_query($query)
     or die("Query failed: ".mysql_error());
   .
   .
```


Okay, now let's confirm that the change was made to the fruit table, updating the number of pears. You can display the new contents of the fruit table this way in phpdataupdate.php:

```
<html>
<head>
  <title>
    Updating databases
  </title>
</head>

<body>
  <h1>Updating databases</h1>

  <?php
     $connection = mysql_connect("localhost","root","*********")
       or die ("Couldn't connect to server");

     $db = mysql_select_db("produce",$connection)
       or die ("Couldn't select database");

     $query = "UPDATE fruit SET number = 234 WHERE name = 'pears'";

     $result = mysql_query($query)
       or die("Query failed: ".mysql_error());

     $query = "SELECT * FROM fruit";

     $result = mysql_query($query)
       or die("Query failed: " . mysql_error());

     echo "<table border='1'>";
     echo "<tr>";
     echo "<th>Name</th><th>Number</th>";
     echo "</tr>";

     while ($row = mysql_fetch_array($result))
     {
       echo "<tr>";
       echo "<td>", $row['name'], "</td><td>", $row['number'], "</td>";
       echo "</tr>";
     }

     echo "</table>";

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

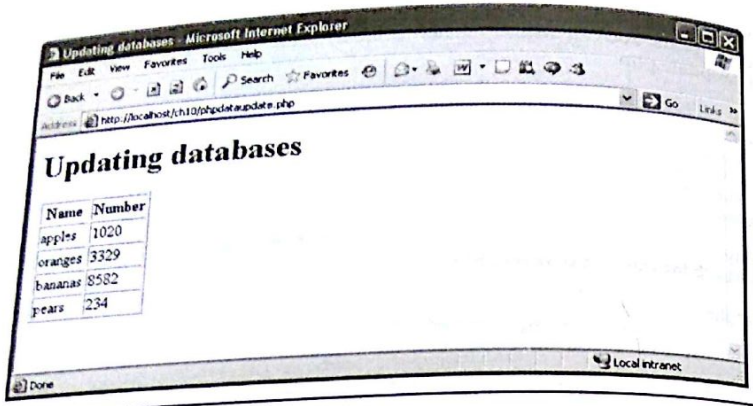


FIGURE 10-2 Updating the fruit table

You can see the results in Figure 10-2, where the number of pears has been updated to 234. Very nice.

Okay, so far so good. Now you've been able to read data from a database, and update that data. What about inserting some new data?

Inserting New Data Items into a Database

Business is good, and you decide to start adding more fruit to your Web site, starting with apricots. Your first shipment is 203 apricots, so you might add that to your stock of fruit:

- apples 1020
- oranges 3329
- bananas 8582
- pears 234
- **apricots 203**

But there's no record for apricots, so it's time to add one, which you can do with the INSERT SQL statement in a new example, phpdatainsert.php.

First, you connect to the database server:

```
<?php
  $connection = mysql_connect("localhost","root","stoic8888")
    or die ("Couldn't connect to server");
    .
    .
    .
```


Then select the fruit database:

```
<?php
  $connection = mysql_connect("localhost","root","stoic8888")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce", $connection)
    or die ("Couldn't select database");
    .
    .
    .
```

You can insert a new record into a table with the SQL INSERT statement. To insert a new record for apricots into the fruit table, you can execute this SQL statement: "INSERT INTO fruit (name, number) VALUES('apricots', '203')":

```
<?php
  $connection = mysql_connect("localhost","root","stoic8888")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce", $connection)
    or die ("Couldn't select database");

  $query = "INSERT INTO fruit (name, number) VALUES('apricots', '203')";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());
    .
    .
    .
```

If successful, that inserts a new record into the database table fruit. You can confirm that by displaying the fruit table in phpdatainsert.php:

```
<html>
 <head>
  <title>
    Inserting new data
  </title>
 </head>

 <body>
  <h1>Inserting new data</h1>

  <?php
    $connection = mysql_connect("localhost","root","stoic8888")
      or die ("Couldn't connect to server");

    $db = mysql_select_db("produce",$connection)
      or die ("Couldn't select database");

    $query = "INSERT INTO fruit (name, number) VALUES('apricots', '203')";
```

```
    $result = mysql_query($query)
      or die("Query failed: " . mysql_error());

    $query = "SELECT * FROM fruit";

    $result = mysql_query($query)
      or die("Query failed: " . mysql_error());

    echo "<table border='1'>";
    echo "<tr>";
    echo "<th>Name</TH><TH>Number</th>";
    echo "</tr>";

    while ($row = mysql_fetch_array($result))
    {
      echo "<TR>";
      echo "<td>", $row['name'], "</td><td>", $row['number'], "</td>";
      echo "</tr>";
    }

    echo "</TABLE>";

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

And you can see the results in Figure 10-3, where a new record has been added for apricots. Cool.

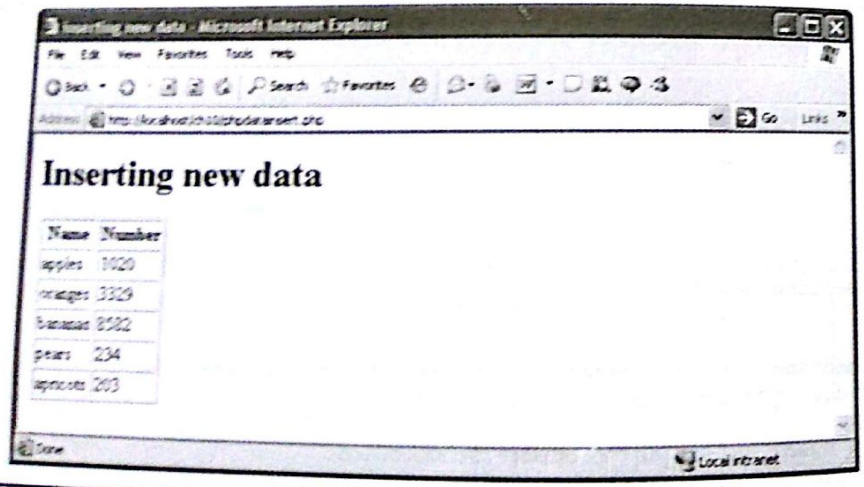


Figure 10-3 Insert a new record into the fruit table

Deleting Records

Now say that your supply of apricots has dried up, so you need to remove them from being offered on your Web site. How do you delete a record? You can use the SQL DELETE statement. Here's an example, phpdatadelete.php, which removes the apricots record from the fruit table. You start as usual, by connecting to the database server and selecting the produce database this way:

```
<?php
  $connection = mysql_connect("localhost","root","")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");
    .
    .
    .
```

Now you can delete the apricots record from the fruit table using the SQL DELETE FROM fruit WHERE name = 'apricots'":

```
<?php
  $connection = mysql_connect("localhost","root","")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "DELETE FROM fruit WHERE name = 'apricots'";
    .
    .
    .
?>
```

And you can execute the query on the database:

```
<?php
  $connection = mysql_connect("localhost","root","")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "DELETE FROM fruit WHERE name = 'apricots'";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());
    .
    .
    .
?>
```

And you can display the newly modified table in phpdatadelete.php:

```
<html>
  <head>
    <title>
      Deleting records
    </title>
  </head>

  <body>
    <h1>Deleting records</h1>

    <?php
      $connection = mysql_connect("localhost","root","********")
        or die ("Couldn't connect to server");

      $db = mysql_select_db("produce",$connection)
        or die ("Couldn't select database");

      $query = "DELETE FROM fruit WHERE name = 'apricots'";

      $result = mysql_query($query)
        or die("Query failed: " . mysql_error());

      $query = "SELECT * FROM fruit";

      $result = mysql_query($query)
        or die("Query failed: " . mysql_error());

      echo "<table border='1'>";
      echo "<tr>";
      echo "<th>Name</th><th>Number</th>";
      echo "</tr>";

      while ($row = mysql_fetch_array($result))
      {
        echo "<tr>";
        echo "<td>", $row['name'], "</td><td>", $row['number'], "</td>";
        echo "</tr>";
      }

      echo "</table>";

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

You can see the results in Figure 10-4, where apricots have been deleted. Cool.

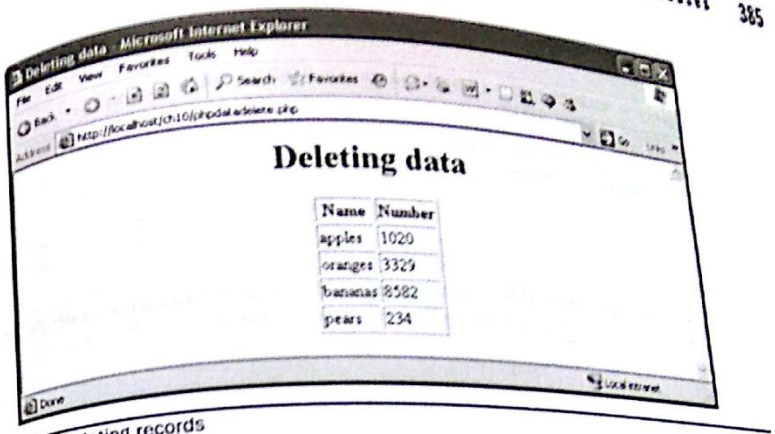


FIGURE 10-4 Deleting records

Creating New Tables

You can also create your own database tables using PHP and SQL. Here's an example, phpcreatetable.php, which creates a new table named vegetables with these records:

Name	Number
corn	2083
spinach	1993
beets	437

You start by connecting to the server, as usual, and by selecting the produce database:

```
<?php
  $connection = mysql_connect("localhost","root","*********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");
    .
    .
    .
?>
```

You can create tables using the SQL CREATE statement. For example, to create a new vegetables table with name and number fields, you could execute the SQL "CREATE TABLE vegetables (name VARCHAR(20), number VARCHAR(20))":

```
<?php
  $connection = mysql_connect("localhost","root","*********")
    or die ("Couldn't connect to server");
```

```
    $db = mysql_select_db("produce",$connection)
      or die ("Couldn't select database");

    $query = "CREATE TABLE vegetables (name VARCHAR(20),
      number VARCHAR(20))";

    $result = mysql_query($query)
      or die("Query failed: " . mysql_error());
      .
      .
      .
?>
```

It's time to stock the new table with some vegetables, which you can do with the SQL INSERT statement. Here's how you might insert a new record for corn, for example:

```
<?php
  $connection = mysql_connect("localhost","root","*********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "CREATE TABLE vegetables (name VARCHAR(20),
    number VARCHAR(20))";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $query = "INSERT INTO vegetables (name, number) VALUES(
    'corn', '2083')";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());
    .
    .
    .
?>
```

And here's how to create the other two records in the vegetables table:

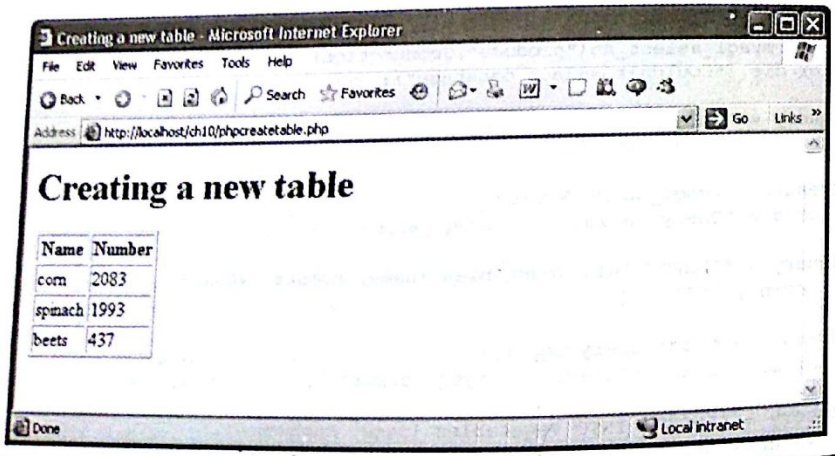
```
<?php
  $connection = mysql_connect("localhost","root","*********")
    or die ("Couldn't connect to server");

  $db = mysql_select_db("produce",$connection)
    or die ("Couldn't select database");

  $query = "CREATE TABLE vegetables (name VARCHAR(20),
    number VARCHAR(20))";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $query = "INSERT INTO vegetables (name, number) VALUES(
    'corn', '2083')";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $query = "INSERT INTO vegetables (name, number)
    VALUES('spinach', '1993')";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $query = "INSERT INTO vegetables (name, number)
    VALUES('beets', '437')";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());
    .
    .
    .
?>
```


Finally, you can display the new table like this in phpcreatetable.php:

```
<html>
 <head>
  <title>
   Creating a new table
  </title>
 </head>

 <body>
  <h1>
   Creating a new table
  </h1>
  <?php
   $connection = mysql_connect("localhost","root","********")
     or die ("Couldn't connect to server");

   $db = mysql_select_db("produce",$connection)
     or die ("Couldn't select database");

   $query = "CREATE TABLE vegetables (name VARCHAR(20),
     number VARCHAR(20))";

   $result = mysql_query($query)
     or die("Query failed: " . mysql_error());

   $query = "INSERT INTO vegetables (name, number) VALUES(
     'corn', '2083')";

   $result = mysql_query($query)
     or die("Query failed: " . mysql_error());

   $query = "INSERT INTO vegetables (name, number)
     VALUES('spinach', '1993')";

   $result = mysql_query($query)
     or die("Query failed: " . mysql_error());
```

```
    $query = "INSERT INTO vegetables (name, number)
      VALUES('beets', '437')";

    $result = mysql_query($query)
      or die("Query failed: " . mysql_error());

    $query = "SELECT * FROM vegetables";

    $result = mysql_query($query)
      or die("Query failed: " . mysql_error());

    echo "<table border='1'>";
    echo "<tr>";
    echo "<th>Name</th><th>Number</th>";
    echo "</tr>";

    while ($row = mysql_fetch_array($result))
    {
      echo "<tr>";
      echo "<td>", $row['name'], "</td><td>",
        $row['number'], "</td>";
      echo "</tr>";
    }

    echo "</table>";

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

You can see the results in Figure 10-5, where the new table appears. Very nice.



E 10-5 Creating a new table

ating a New Database

You can even create entirely new databases, with their own tables, using PHP and SQL. For example, you might create a new database, foods, with a snacks table that has these records:

Name	Number
tacos	218
pizza	193
cheeseburgers	112

Here's how to do that in phpcreatedatabase.php. First, connect to the database server:

```
<?php
 $connection = mysql_connect("localhost","root","********")
  or die ("Couldn't connect to server");

  .
  .
  .
?>
```

Now you can create a whole new database, foods, with the SQL "CREATE DATABASE IF NOT EXISTS foods":

```
<?php
 $connection = mysql_connect("localhost","root","********")
  or die ("Couldn't connect to server");

 $query = "CREATE DATABASE IF NOT EXISTS foods";

 $result = mysql_query($query)
  or die("Query failed: " . mysql_error());
  .
  .
  .
?>
```

Next, you select the foods database:

```
<?php
 $connection = mysql_connect("localhost","root","********")
  or die ("Couldn't connect to server");

 $query = "CREATE DATABASE IF NOT EXISTS foods";

 $result = mysql_query($query)
  or die("Query failed: " . mysql_error());

 $db = mysql_select_db("foods", $connection)
  or die ("Couldn't select database");
  .
  .
  .
?>
```

Then it's time to create the snacks table:

```
<?php
  $connection = mysql_connect("localhost","root","********")
    or die ("Couldn't connect to server");

  $query = "CREATE DATABASE IF NOT EXISTS foods";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $db = mysql_select_db("foods", $connection)
    or die ("Couldn't select database");

  $query = "CREATE TABLE snacks (name VARCHAR(20), number
    VARCHAR(20))";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());
    .
    .
    .
?>
```

And then you stock the snacks table with some snacks:

```
<?php
  $connection = mysql_connect("localhost","root","********")
    or die ("Couldn't connect to server");

  $query = "CREATE DATABASE IF NOT EXISTS foods";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $db = mysql_select_db("foods", $connection)
    or die ("Couldn't select database");

  $query = "CREATE TABLE snacks (name VARCHAR(20), number
    VARCHAR(20))";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $query = "INSERT INTO snacks (name, number)
    VALUES('tacos', '218')";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $query = "INSERT INTO snacks (name, number)
    VALUES('pizza', '193')";
```



```
  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());

  $query = "INSERT INTO snacks (name, number)
    VALUES('cheeseburgers', '112')";

  $result = mysql_query($query)
    or die("Query failed: " . mysql_error());
    .
    .
    .
?>
```

Now you can display the new snacks table:

```
<html>
<head>
  <title>
    Creating a new database
  </title>
</head>

<body>
  <h1>
    Creating a new database
  </h1>

  <?php
    $connection = mysql_connect("localhost","root","********")
      or die ("Couldn't connect to server");

    $query = "CREATE DATABASE IF NOT EXISTS foods";

    $result = mysql_query($query)
      or die("Query failed: " . mysql_error())

    $db = mysql_select_db("foods", $connection)
      or die ("Couldn't select database");

    $query = "CREATE TABLE snacks (name VARCHAR(20), number
      VARCHAR(20))";

    $result = mysql_query($query)
      or die("Query failed: " . mysql_error());

    $query = "INSERT INTO snacks (name, number)
      VALUES('tacos', '218')";

    $result = mysql_query($query)
      or die("Query failed: " . mysql_error());

    $query = "INSERT INTO snacks (name, number)
      VALUES('pizza', '193')";
```

```
        $result = mysql_query($query)
          or die("Query failed: " . mysql_error());

        $query = "INSERT INTO snacks (name, number)
          VALUES('cheeseburgers', '112')";

        $result = mysql_query($query)
          or die("Query failed: " . mysql_error());
        $query = "SELECT * FROM snacks";

        $result = mysql_query($query)
          or die("Query failed: " . mysql_error());

        echo "<table border='1'>";
        echo "<tr>";
        echo "<th>Name</th><th>Number</th>";
        echo "</tr>";

        while ($row = mysql_fetch_array($result))
        {
          echo "<tr>";
          echo "<td>", $row['name'], "</td><td>",
            $row['number'], "</td>";
          echo "</tr>";
        }
        echo "</table>";

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

You can see the results in Figure 10-6, where the new table in the new database appears.

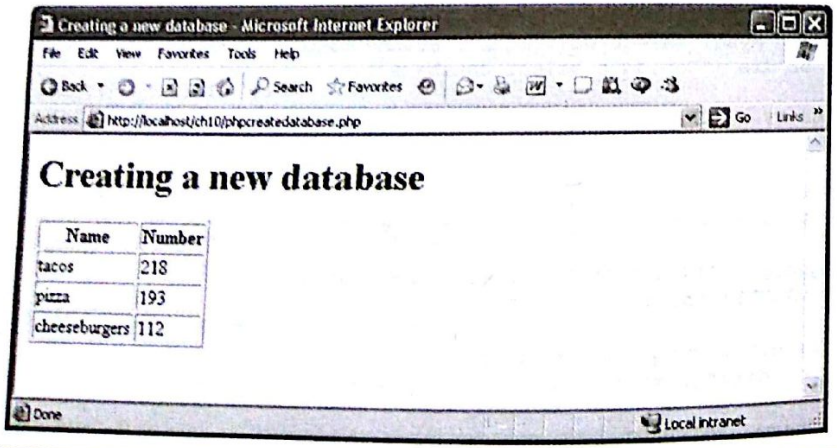


FIGURE 10-6 Creating a new database

Sorting Your Data

You can keep going with your database work using SQL in PHP—anything possible with databases is possible with SQL. Here's another example, phpsortdata.php, that uses SQL to sort the data in the fruit table before displaying it:

```
<html>
 <head>
  <title>
   Sorting your data
  </title>
 </head>

 <body>
  <h1>
   Sorting your data
  </h1>

  <?php
   $connection = mysql_connect("localhost","root","********")
     or die ("Couldn't connect to server");

   $db = mysql_select_db("produce",$connection)
     or die ("Couldn't select database");

   $query = "SELECT * FROM fruit ORDER BY name";

   $result = mysql_query($query)
     or die("Query failed: " . mysql_error());

   echo "<table border='1'>";
   echo "<tr>";
   echo "<th>Name</th><th>Number</th>";
   echo "</TR>";

   while ($row = mysql_fetch_array($result))
   {
     echo "<tr>";
     echo "<td>", $row['name'], "</td><td>",
       $row['number'], "</td>";
     echo "</tr>";
   }

   echo "</table>";

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

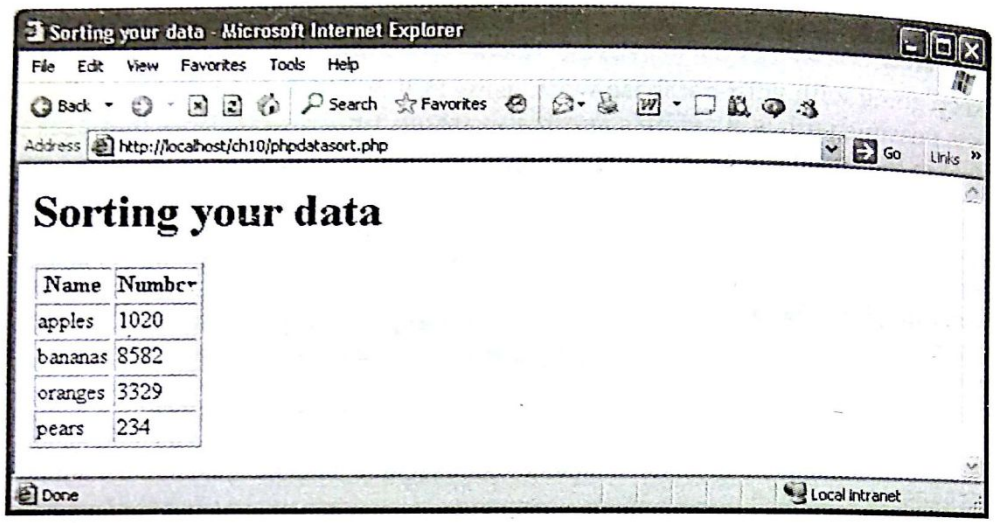


FIGURE 10-7 Sorting your data

You can see the results in Figure 10-7, where your data was sorted.
Now that you can use SQL with PHP, your database capabilities are unlimited.