PHP MySQL Create Database and Tables

A database holds one or multiple tables.

Create a Database

The CREATE DATABASE statement is used to create a database in MySQL.

Syntax

```
CREATE DATABASE database_name
```

To get PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

```
<?php
$con = mysql_connect("localhost","root"," ");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

if (mysql_query("CREATE DATABASE my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }

mysql_close($con);
?>
```

Create a Table

The CREATE TABLE statement is used to create a table in MySQL.

Syntax

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
)
```

We must add the CREATE TABLE statement to the mysql_query() function to execute the command.

Example

The following example creates a table named "Persons", with three columns. The column names will be "FirstName", "LastName" and "Age":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// Create database
if (mysql_query("CREATE DATABASE my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }

// Create table
mysql_select_db("my_db", $con);
$sql = "CREATE TABLE Persons
(
FirstName varchar(15),
LastName varchar(15),
Age int
)";

// Execute query
mysql_query($sql,$con);

mysql_close($con);
?>
```

Primary Keys and Auto Increment Fields

Each table should have a primary key field.

A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The following example sets the personID field as the primary key field. The primary key field is often an ID number, and is often used with the AUTO_INCREMENT setting. AUTO_INCREMENT automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the NOT NULL setting to the field.

Example

```
$sql = "CREATE TABLE Persons
(
personID int NOT NULL AUTO_INCREMENT,
PRIMARY KEY(personID),
FirstName varchar(15),
LastName varchar(15),
Age int
)";

mysql_query($sql,$con);
```

Foreign key

The ORDER BY Keyword

The ORDER BY keyword is used to sort the data in a recordset.

The ORDER BY keyword sort the records in ascending order by default.

If you want to sort the records in a descending order, you can use the DESC keyword.

Syntax

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

Example

The following example selects all the data stored in the "Persons" table, and sorts the result by the "Age" column:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
```

```
mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons ORDER BY age");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'];
  echo " " . $row['LastName'];
  echo " " . $row['Age'];
  echo "<br />";
  }
mysql_close($con);
?>
```

The output of the code above will be:

```
Glenn Quagmire 33
Peter Griffin 35
```

PHP My SQL Insert into Data.

The INSERT INTO statement is used to insert new records in a table.

Insert Data Into a Database Table

The INSERT INTO statement is used to add new records to a database table.

Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

To get PHP to execute the statements above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example

In the previous chapter we created a table named "Persons", with three columns; "Firstname", "Lastname" and "Age". We will use the same table in this example. The following example adds two new records to the "Persons" table:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Peter', 'Griffin',35)");

mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Glenn', 'Quagmire',33)");

mysql_close($con);
?>
```

Insert Data from a Form into a Database

Now we will create an HTML form that can be used to add new records to the "Persons" table.

Here is the HTML form:

```
<html>
<body>

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php".

The "insert.php" file connects to a database, and retrieves the values from the form with the PHP $_POST variables.

Then, the mysql_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the "insert.php" page:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
```

```
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$sql="INSERT INTO Persons (FirstName, LastName, Age)
VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";

if (!mysql_query($sql,$con))
  {
  die('Error: ' . mysql_error());
  }
echo "1 record added";

mysql_close($con);
?>
```

Delete Data In a Database

The DELETE FROM statement is used to delete records from a database table.

Syntax

```
DELETE FROM table_name
WHERE some_column = some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

To get PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example

Look at the following "Persons" table:

FirstName	LastName	Age
Peter	Griffin	35
Glenn	Quagmire	33

The following example deletes all the records in the "Persons" table where LastName='Griffin':

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
```

```
mysql_select_db("my_db", $con);

mysql_query("DELETE FROM Persons WHERE LastName='Griffin'");

mysql_close($con);
?>
```

After the deletion, the table will look like this:

FirstName	**LastName**	**Age**
Glenn	Quagmire	33

Update Data in a Database

The UPDATE statement is used to update existing records in a table.

Syntax

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Example

Earlier in the tutorial we created a table named "Persons". Here is how it looks:

FirstName	**LastName**	**Age**
Peter	Griffin	35
Glenn	Quagmire	33

The following example updates some data in the "Persons" table:

```
<?php
$con = mysql_connect("localhost","root"," ");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("UPDATE Persons SET Age=36
WHERE FirstName='Peter' AND LastName='Griffin'");

mysql_close($con);
?>
```

After the update, the "Persons" table will look like this:

FirstName	**LastName**	**Age**
Peter	Griffin	36
Glenn	Quagmire	33

MYSQL_FUNCTIONS

mysql_affected_rows

- mysql_affected_rows — Get number of affected rows in previous MySQL operation
- Get the number of affected rows by the last INSERT, UPDATE, REPLACE or DELETE query associated with link_identifier.

```
<?php
  $link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
   if (!$link)
    {
    die('Could not connect: ' . mysql_error());
     }
mysql_select_db('mydb');

/* this should return the correct numbers of deleted records */

mysql_query('DELETE FROM mytable WHERE id < 10');
printf("Records deleted: %d\n", mysql_affected_rows());

/* with a where clause that is never true, it should return 0 */
mysql_query('DELETE FROM mytable WHERE 0');
printf("Records deleted: %d\n", mysql_affected_rows());
?>
```

The above example will output something similar to:

```
Records deleted: 10
Records deleted: 0
```

mysql_fetch_row:

mysql_fetch_row — Get a result row as an enumerated array.

Returns a numerical array that corresponds to the fetched row and moves the internal data pointer ahead.

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
$row = mysql_fetch_row($result);

echo $row[0]; // 42
echo $row[1]; // the email value
?>
```

mysql_fetch_object:

mysql_fetch_object — Fetch a result row as an object

```
<?php
mysql_connect("hostname", "user", "password");
mysql_select_db("mydb");
$result = mysql_query("select * from mytable");
while ($row = mysql_fetch_object($result))
{
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

Performance: Speed-wise, the function is identical to mysql_fetch_array(), and almost as quick as mysql_fetch_row() (the difference is insignificant).

Note: mysql_fetch_object() is similar to mysql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

mysql_fetch_array:

mysql_fetch_array — Fetch a result row as an associative array, a numeric array, or both

Returns an array that corresponds to the fetched row and moves the internal data pointer ahead.

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_NUM)) {
    printf("ID: %s  Name: %s", $row[0], $row[1]);
}

mysql_free_result($result);
?>
```

mysql_fetch_array() with MYSQL_ASSOC

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
   die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
   printf("ID: %s  Name: %s", $row["id"], $row["name"]);
}

mysql_free_result($result);
?>
```

mysql_fetch_array() with MYSQL_BOTH

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
   die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
   printf ("ID: %s  Name: %s", $row[0], $row["name"]);
}

mysql_free_result($result);
?>
```

mysql_fetch_lengths

mysql_fetch_lengths — Get the length of each output in a result

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}
```

```
$row     = mysql_fetch_assoc($result);
$lengths = mysql_fetch_lengths($result);

print_r($row);
print_r($lengths);
?>
```

The above example will output something similar to:

```
Array
(
    [id] => 42
    [email] => user@example.com
)
Array
(
    [0] => 2
    [1] => 16
)
```

mysql_data_seek() Function

→the mysql_data_seek() function moves the internal row pointer.

→The internal row pointer is the current row position in a result returned by the mysql_query() function.

→This function returns TRUE on success, or FALSE on failure.

mysql_data_seek(data, row)

Parameter	**Description**
Data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function
Row	Required. Specifies which record to move to. 0 indicates the first record

```
<?php
$con = mysql_connect("localhost", "root", " ”);
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
$db_selected = mysql_select_db("test_db",$con);
$sql = "SELECT * from Person";
$result = mysql_query($sql,$con);
```

```
print_r(mysql_fetch_row($result));
mysql_data_seek($result,3);
print_r(mysql_fetch_row($result));
mysql_close($con);
?>
Array
(
[0] => Refsnes
[1] => Kai Jim
[2] => Taugata
)

Array
(
[0] => Refsnes
[1] => Ståle
[2] => Sandnes
```

mysql_fetch_lengths() Function

The mysql_fetch_lengths() function returns the length of the contents of each field in a row.

The row is retrieved by mysql_fetch_array(), mysql_fetch_assoc(), mysql_fetch_object(), or mysql_fetch_row().

This function returns a numeric array on success, or FALSE on failure or when there are no more rows.

mysql_fetch_lengths(data)

Parameter	**Description**
Data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

```
?php
$con = mysql_connect("localhost", "root", " ");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected = mysql_select_db("test_db",$con);
$sql = "SELECT * from Person WHERE Lastname='Refsnes'";
$result = mysql_query($sql,$con);
print_r(mysql_fetch_row($result));
print_r(mysql_fetch_lengths($result));
```

```
mysql_close($con);
?>
```

The output of the code above could be:

```
Array
(
[0] => Refsnes
[1] => Kai Jim
[2] => Taugata 2
[3] => 22
)

Array
(
[0] => 7
[1] => 7
[2] => 9
[3] => 2
)
```

mysql_field_table() Function

The mysql_field_table() function returns the name of the table where a specified field is located.

Returns the table name of a specified field name on success, or FALSE on failure.

mysql_field_table(data,field_offset)

Parameter	**Description**
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function
field_offset	Required. Specifies which field to start on. 0 indicates the first field

```
<?php
$con = mysql_connect("localhost", "root", " ");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected = mysql_select_db("test_db",$con);

$sql = "SELECT * from Person";
$result = mysql_query($sql,$con);

$table = mysql_field_table($result, 0);

echo $table;
```

```
mysql_close($con);
?>
```

The output of the code above could be:

```
Person
```

mysql_field_name() Function

The mysql_field_name() function returns the name of a field in a recordset.

Returns the field name on success, or FALSE on failure.
mysql_field_name(data,field_offset)

Parameter	**Description**
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function
field_offset	Required. Specifies which field to start returning. 0 indicates the first field

```
<?php
$con = mysql_connect("localhost", "root", " ");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected = mysql_select_db("test_db",$con);

$sql = "SELECT * from Person";
$result = mysql_query($sql,$con);

$name = mysql_field_name($result, 0);

echo $name;

mysql_close($con);
?>
```

The output of the code above could be:

```
LastName
```

mysql_field_type() Function

The mysql_field_type() function returns the type of a field in a recordset.

Returns the type of the specified field on success, or FALSE on failure.

Syntax

mysql_field_type(data,field_offset)

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function
field_offset	Required. Specifies which field to start on. 0 indicates the first field

---

Example

```
<?php
$con = mysql_connect("localhost", " ", " ");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected = mysql_select_db("test_db",$con);

$sql = "SELECT * from Person";
$result = mysql_query($sql,$con);

$type = mysql_field_type($result, 0);

echo $type;

mysql_close($con);
?>
```

The output of the code above could be:

```
string
```

mysql_get_client_info() Function

The mysql_get_client_info() function gets information about the MySQL client.

This function returns the MySQL client version number on success, or FALSE on failure.

Syntax

mysql_get_client_info()

Example

```
<?php
echo "MySQL client info: " . mysql_get_client_info();
?>
```

The output of the code above could be:

```
MySQL client info: 5.0.18
```

mysql_get_host_info() Function

The mysql_get_host_info() function gets information about the MySQL host.

This function returns the type of MySQL connection currently in use, or FALSE on failure.

Syntax

mysql_get_host_info(connection)

Parameter	Description
connection	Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

Example

```
<?php
$con = mysql_connect("localhost", "root", "");
echo "MySQL host info: " . mysql_get_host_info($con);
?>
```

The output of the code above could be:

```
MySQL host info: localhost via TCP/IP
```

mysql_get_server_info() Function

The mysql_get_server_info() function gets information about the MySQL server.

This function returns the MySQL server version on success, or FALSE on failure.

Syntax

mysql_get_server_info(connection)

Parameter	Description
connection	Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

Example

```
<?php
$con = mysql_connect("localhost", " ", " ");
echo "MySQL server info: " . mysql_get_server_info($con);
?>
```

The output of the code above could be:

```
MySQL server info: 5.0.18
```

mysql_info() Function

Definition and Usage

The mysql_info() function returns information about the last query.

This function returns information about the statement on success, or FALSE on failure.

Syntax

```
mysql_info(connection)
```

Parameter	**Description**
connection	Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

Example

```
<?php
$con = mysql_connect("localhost", " ", " ");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected = mysql_select_db("test_db",$con);
$sql = "INSERT INTO person VALUES ('John','Doe','Utah','17')";
$result = mysql_query($sql,$con);

$info = mysql_info($con);
echo $info;

mysql_close($con);
?>
```

The output of the code above could be:

```
String format: Records: 15 Duplicates: 0 Warnings: 0
```