INTRODUCTION TO DATABASES

One of the most useful features of PHP is the ability to access a database. Databases store and manage large collections of information. PHP pages can be used to make this information available to the users who visit your Web site.

Instead of storing information in text files or static Web pages, a PHP page can be set up to retrieve, format and display data from a database. When a user accesses the PHP page, the information in the PHP page will be created from the current information in the database. A PHP page can also allow users to manipulate the data in a database.

Using databases to store information and then using PHP pages to access that information is an efficient method of displaying up-to-date information in a Web site.

DATABASE STRUCTURE

Information in a database is made up of two components—raw data and meta-data. The raw data component of a database is the actual information being stored, such as the phone number of a client. The meta-data component of a database determines the way the data is structured and organized. For example, data in a database is usually organized into tables, which consist of fields and records. A field is a specific category of information in a table, such as the first names of all your clients. A record is a collection of information about one person, place or thing, such as the name and address of a client. More sophisticated databases would also contain meta-data such as data types and the relationship between tables.

DATABASE MANAGEMENT SYSTEM (DBMS)

To access and modify the data in a database, you need a program or set of programs called a DataBase Management System (DBMS). Depending on the type of database you are using, the DBMS can be very simple or very sophisticated. For example, you can write your own DBMS in PHP for a flat file database. You can also obtain a DBMS, such as MySQL or Oracle, which is capable of handling more complex databases.

ACCESSING A DATABASE

The way you access a database depends on the type of database you are using. You can use PHP's Filesystem functions to directly access a flat file database. For other types of databases, you must create a connection to the database before you can access the database. To create a connection to a database, you use the set of PHP functions specifically written to work with the database you are using. For example, if you are using a MySQL database, you would use PHP's MySQL functions. If the database you are using does not have its own set of functions, you can use PHP's Unified ODBC functions to connect to the database. ODBC, or Open DataBase Connectivity, is a standard supported by many databases.

TYPES OF DATABASES

The type of database you are using determines the structure of the database and the way data is organized in the database. A database with a simple structure allows you to access data quickly, but manipulating the data may be difficult. A database with a more complex structure allows you to easily manipulate data, but requires more resources. When selecting a database, you should consider the amount of data that will be stored and how the data is going to change over time. PHP is commonly used with flat file, hierarchical and relational databases.

Flat File Databases

Flat file databases are the most basic database type. Flat file databases have minimal structure and usually store information in one large table, similar to a tab-delimited text file. A flat file database is very easy to set up and is ideal for creating simple Web applications that use small amounts of data, such as an application that keeps track of user preferences.

When working with larger applications, a flat file database can be inefficient and difficult to maintain. The simple structure of a flat file database makes it inadequate for storing complex relationships, resulting in the duplication of information. Also, if your data storage requirements change, a flat file database can be very difficult to modify.

Hierarchical Databases

A hierarchical database organizes data into different levels, similar to directories and sub-directories. This organization makes data access fast and easy. You would typically use a hierarchical database to store large amounts of information that seldom need to be changed, such as contact lists or user directories. Some of the hierarchical databases supported by PHP use the Lightweight Directory Access Protocol (LDAP). This protocol was designed specifically for reading data quickly. You can use PHP's LDAP functions to access these types of databases.

Due to its rigid structure, manipulating data in a hierarchical database can be very slow. Data redundancy can also be problem. For example, if you organized customers by method of payment in a hierarchical database, a customer who pays by both cash and credit card would have two accounts. Data redundancy makes hierarchical databases inefficient for certain applications.

Relational Databases

Relational databases store information in separate tables. Each record in a table has a unique identifier, or primary key, which can be used to form relationships with entries in other tables. Using relationships to bring together information from different tables eliminates data redundancy. Relational databases are powerful, flexible and effectively store large amounts of information. A relational database is also faster and easier to maintain than other types of databases. You would typically use a relational database when building dynamic database-driven Web sites with PHP.

PLAN A DATABASE

You should take the time to properly design a database, especially when working with a relational database management system. A good database design ensures that you will be able to perform tasks efficiently and accurately. As you add information to a database, the database becomes larger and more complex. A database that is designed properly will be easier to modify and work with as it grows. Good planning can also make it easier for other users to work with a database you create.

Determine the Purpose of the Database

Decide what you want the database to do and how you plan to use the information. If other people will be using the database, you should consult with them and consider their needs. This can help you determine what information you need to include to make the database complete.

Security and Reliability

You should determine the level of security you will need to protect the information the database will store. If the database will store sensitive information such as credit card numbers, you need to select a database system that is secure and reliable. To ensure maximum security, you may even want to create a separate database to store sensitive information.

Anticipate Future Needs

Decide on a naming convention for databases, tables and fields. Make sure names are concise and descriptive to prevent confusion in case changes need to be made in the future. If many people will work with the database, you should also decide on a method for documenting changes made to the database. For example, you can use a spreadsheet program to keep track of the dates and times changes were made and the name of the person who made the changes.

Determine the Tables You Need

Gather all the information you want to store in the database and then divide the information into separate tables. A table should contain related information about one subject only. The same information should not appear in more than one table in a database. You can work more efficiently and reduce errors if you need to update information in only one table.

Consider the Fields You Need

Each field should relate directly to the subject of the table. When adding fields, make sure you break down information into its smallest parts. For example, break down names into two fields called firstName and lastName.

Try to keep the number of fields in a table to a minimum. For example, do not include a field containing data you can calculate from other fields. Tables with many fields increase the time it takes to process information in a database.

Determine the Relationship Between Tables

A relationship tells a relational database how to bring together related information stored in separate tables. You can use the primary key to form a relationship between tables. A primary key is one or more fields that uniquely identifies each record in a table. For example, the primary key for a table of employees could be the social security number of each employee.

SELECT A DATABASE MANAGEMENT SYSTEM

PHP supports a wide variety of DataBase Management Systems (DBMS) that you can use to store and manage information you want to make available in your PHP pages. When choosing a DBMS, you should consider a number of factors, such as your budget and the volume of information the database will be required to handle. You should also consider the ease with which you will be able to move data to a more robust system if the database grows beyond the capabilities of the current system. If the DBMS you use does not support accepted standards, moving data to a new system might be difficult.

MySQL

MySQL is the database program most commonly used to develop database-driven PHP Web sites. MySQL is a fast, efficient program that is available for use on Unix and Windows computers. MySQL is suitable for small to medium-sized projects and requires very few system resources to run. MySQL is also an Open Source product, which means you can use it free of charge. Although MySQL is very powerful, it is relatively easy to install and manage. This makes it a good program for new developers to use when learning to work with databases using PHP. More information about MySQL is available at the www.mysql.com Web site.

PostgreSQL

PostgreSQL is another popular DBMS used with PHP. Like MySQL, PostgreSQL is an Open Source product that is freely available for personal and commercial use, but PostgreSQL is a more advanced system that supports almost all SQL features. PostgreSQL is available for use only on Unix systems and is usually included with the RedHat distribution of the Linux operating system. For more information about PostgreSQL, you can visit the www.postgresql.org Web site.

Oracle

Oracle is a powerful database program that is typically used with high-end Web servers on large commercial Web sites, such as sites that offer online shopping. Oracle is reliable, provides advanced security, is highly scalable and includes powerful tools that you can use to manage your database. Oracle is available for both Unix and Windows computers. For more information about the Oracle database program, you can visit the www.oracle.com Web site.

Microsoft SQL Server

Microsoft SQL Server is an industrial-strength DBMS that offers features that are comparable to Oracle. SQL Server includes advanced features, such as OnLine Analytical Processing (OLAP) and data mining, which allow you to work more efficiently with information in a large database. SQL Server is only available for use with Windows systems. For more information about SQL Server, you can visit the www.microsoft.com/sql Web site.

Microsoft Access

Microsoft Access is a database program that is useful for small to medium-sized applications. Access is available as a stand-alone product or as part of the Microsoft Office package. There are currently no built-in PHP functions that allow you to connect to an Access database from a PHP page, so you must use PHP's Unified ODBC functions to connect. For more information about PHP's Unified ODBC functions, you can visit the www.php.net/manual/en/ref.odbc.php Web site. Information about Access is available at the www.microsoft.com/access Web site.

INTRODUCTION TO STRUCTURED QUERY LANGUAGE

Relational database management systems, such as MySQL and Oracle, are usually used for building dynamic, database-driven Web sites with PHP. In order for a PHP script to work with data in a relational database, the script must be able to communicate with the database. Structured Query Language (SQL) is the language used in a PHP script to communicate with a database.

Originally developed by IBM as a database query language for use with mainframe computers, SQL was soon adopted by many vendors for use with their relational database management systems. Most relational database management systems that have a client/server structure now support SQL.

Standardization

SQL is the industry standard language for managing and manipulating data in a database. SQL can be used to work with many types of databases, which makes it easy to upgrade from one database management system to another. For example, a small Web site might start out using MySQL but then grow large enough to require a database created using Oracle. You have to learn only one language to have your PHP scripts communicate with both types of databases.

Ease of Use

SQL uses many easy-to-understand commands, which makes it a very simple language to work with. For example, SQL uses the `INSERT` statement to add information to a database and the `DELETE` statement to remove information. These plain-language commands make it easy for you to read and determine the purpose of SQL code.

Power

Although SQL is easy to use, it is a very powerful language. As well as being suitable for retrieving data from a database and performing simple tasks such as adding and deleting records, SQL can be used to perform complicated procedures, such as compiling different types of data from multiple data sources.

Flexibility

The ability to easily manipulate data into many useful forms makes SQL a very effective tool. SQL allows you to easily format data you retrieve from a database in a variety of ways. You can sort data in a specific order, create summaries and combine information from different fields. This saves you from having to expend additional time and resources to format the retrieved data in PHP.

Vendor Specifications

Although many of the basic concepts of SQL, such as selecting and inserting data, are common to most database management systems, certain features may vary depending on the vendor of the system. For example, a database may include a unique set of commands that allow users to access information about a table. These commands may not work for other databases since they are not part of the standard set of SQL statements. The types of information that a database can store may also vary depending on the vendor.

SQL STATEMENTS

SQL is made up of many statements and clauses. In order to work with a database in your PHP scripts, you will need to be familiar with some common SQL statements and how they are used.

CREATE TABLE

The `CREATE TABLE` statement is used to create a new table in a database. You indicate the name of the table you want to create after the `CREATE TABLE` statement and then specify the fields you want to include in the table. You also specify a description of each field that includes the data type of the field and other attributes, such as the maximum number of characters the field can hold. The data types you specify will depend on the database you are using. You should refer to the database documentation for more information.

Example:

```
CREATE TABLE orders
(
      invoiceNumber INT(8),
      orderDetails CHAR(255),
      totalCost FLOAT(9, 2)
)
```

INSERT

The `INSERT` statement allows you to add records to a database. The `INSERT` statement uses the `INTO` clause to specify the name of the table to which you want to add data and the names of the fields that store the data in the table. The `VALUES` clause specifies the values you are adding.

Example:

```
INSERT INTO orders (invoiceNumber, totalCost)
VALUES (12843, 34.56)
```

UPDATE

The `UPDATE` statement is used to modify data in a database. You indicate the table that contains the data you want to modify after the `UPDATE` statement. The `UPDATE` statement uses the `SET` statement to indicate which field needs to be changed and the new value. The `WHERE` clause is then used to specify the data to be modified.

Example:

```
UPDATE orders SET totalCost = 55.66
WHERE invoiceNumber = 12843
```

SELECT

The `SELECT` statement allows you to retrieve data from a database. You specify the names of the fields from which you want to retrieve data after the `SELECT` statement. The `SELECT` statement uses the `FROM` clause to specify the name of the table that stores the data you want to retrieve. The `WHERE` clause specifies exactly which data you want to retrieve.

Example:

```
SELECT invoiceNumber, totalCost FROM orders
WHERE totalCost > 100.00
```

Result Set

The `SELECT` statement returns data in a table format called a result set. The fields you specify make up the columns of the table and the information from each field forms the rows. For example, when you issue a `SELECT` statement that retrieves the invoice numbers for orders totaling less than $100 from a table in a database, the result set will display a table with two columns that show the invoice numbers and the amounts.

The `SELECT` statement allows you to access different groups of data from one or more tables and display the data in a single result set. You can indicate parameters with the `SELECT` statement, such as a specific sorting order, to control the way the data is displayed in the result set.

DELETE

The `DELETE` statement is used to remove data from a database. The `DELETE` statement uses the `FROM` clause to specify the name of the table that stores the data you want to delete. The `WHERE` clause contains information that uniquely identifies the data you want to delete.

Example:

```
DELETE FROM orders WHERE year < 1996
```

USING A DATABASE CLIENT

The most popular relational DataBase Management Systems (DBMS) have a client/server architecture. A client/server architecture allows multiple users to simultaneously connect to a single database. Compared to other database management systems, this type of system is extremely cost effective and offers a high level of performance.

The server component of the DBMS contains the actual database. The server software interprets commands from users and then manages the data in the database accordingly.

The client component of the system provides the interface needed to work with the server. You can issue Structured Query Language (SQL) statements from a text-based client to perform tasks such as displaying a list of the databases on the server, finding a list of the tables in a database or viewing the contents of a table.

You can also use the client software that came with your DBMS to set up a database. While it is possible to perform tasks such as creating the tables for a database by issuing SQL statements from a PHP script, this process is often more difficult and time consuming than using the client software. Typically, client software can also be used to perform administrative duties, such as adding users to the system, provided the correct access privileges have been granted.

When a DBMS is installed on a server, both the server and client software are installed. To access the server from a remote computer, only the client software is required on the computer. The operating system of some client computers may have to be altered in order to work properly with the client software. For more information, consult the documentation that came with your DBMS. Before using the client software to work with a database, you must make sure that both the client and server software are running. It is a common error to launch the client software without launching the server software first.

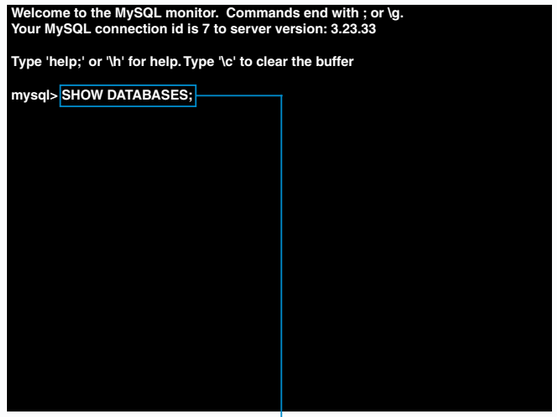
Extra

To access a database server from a remote computer, you must connect to the server through a local area network or Internet connection. To do so, you need to know the address of the server and login information, such as a user name and password. If you are using an Internet Service Provider (ISP) to run your PHP scripts, you will typically have access to a database server, although there may be additional costs for the server's use. You can consult your ISP to obtain more information about connecting to the database server.

You may want to type your SQL statements in a text editor and then copy and paste the statements into the client software. Typing long SQL statements directly into client software can be a slow and tedious process. If you make a mistake that generates an error, you will have to retype the entire statement again. When working in a text editor, you can easily edit your SQL statements and even save the statements for later use.

In addition to text-based client software, most relational database management systems also include a client that utilizes a Graphical User Interface (GUI). You can use GUI client software to modify and administer a database without having to type SQL statements. This type of client software is much easier to use than a text-based client and can help you quickly set up a database.

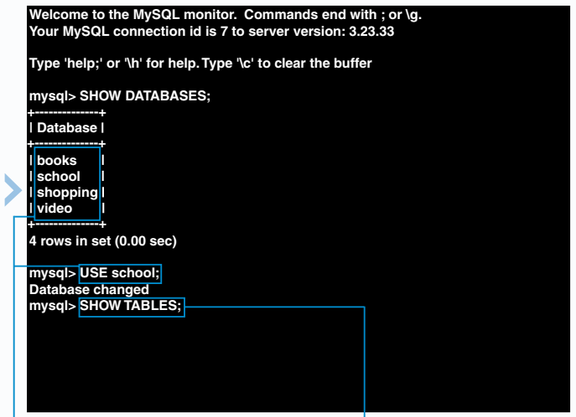
USING A DATABASE CLIENT



1 Locate and start the client software for your database management system.

■ The database server you want to connect to must be running.

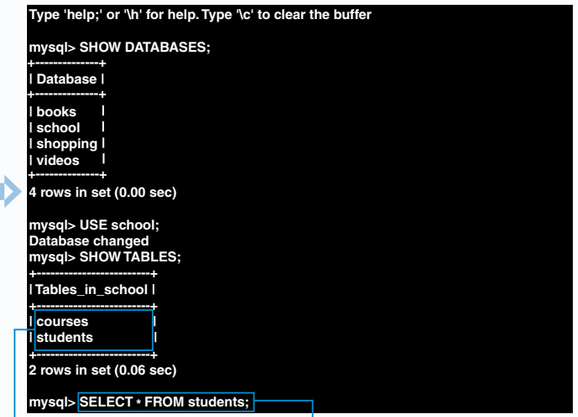
2 To display a list of the databases on the server, type **SHOW DATABASES;** and then press Enter.



■ A list of the databases on the server appears.

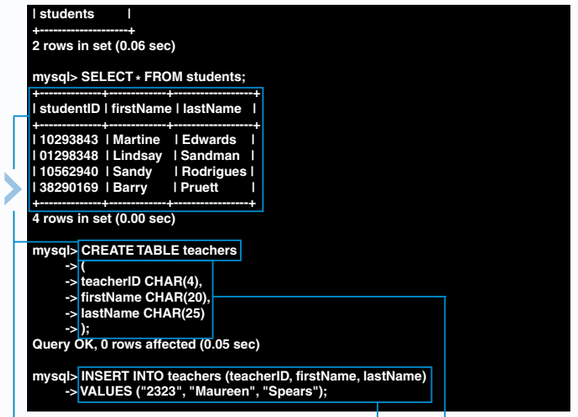
3 To select the database you want to work with, type **USE** followed by the name of the database and a semicolon. Then press Enter.

4 To display a list of the tables in the current database, type **SHOW TABLES;** and then press Enter.



■ A list of the tables in the database appears.

5 To display the contents of a table, type **SELECT * FROM** followed by the name of the table whose contents you want to view. Then type a semicolon and press Enter.



■ The contents of the table appear.

6 To create a new table in the current database, type **CREATE TABLE** followed by the name of the table you want to create. Then press Enter.

7 Type the code that creates fields and records for the table.

■ The new table is created and added to the database.