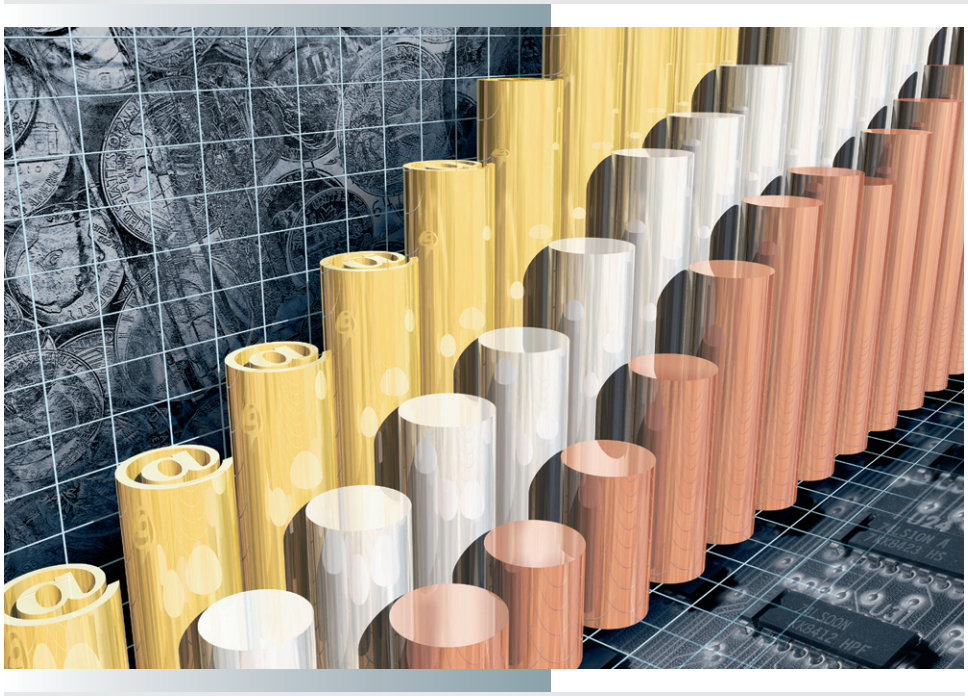


chapter 12

Database Connectivity and Server-Side Scripting

"High definition is the state of being well filled with data."

—Marshall McLuhan, Understanding Media

In this chapter, you will learn how to:

- Define the technologies through which Web servers provide client access to server-side databases.
- Describe the three basic kinds of databases and list the steps involved in designing a relational database.
- Define the purpose of a database connection object and an SQL statement.
- Describe how server-side scripts use loops and logical expressions to make decisions based on dynamic database content.

THE three-tier Web application model consists of the user interface in tier 1, the business object in tier 2, and the back office databases in tier 3. The first two parts of this book concentrated on what happens in the first two tiers. Now it is time to go behind the scenes and understand what happens in the data tier. Accordingly, this chapter defines the technologies through which Web servers provide client access to server-side databases.

Databases can be flat, relational, or object-oriented. After explaining the pluses and minuses of these three kinds of databases, this chapter steps through the process of creating a relational database. You will understand how the business tier queries the database to determine the dynamic content that the user sees onscreen. One of the greatest strengths of a data-driven Web site is its extensibility; you will understand how a properly designed relational database enables you to key new data structures to preexisting data, thereby enabling the business tier to handle new processes as the site expands its service offerings.

The business tier uses server-side programs or scripts to interact with the data tier. This chapter describes how these programs connect to the database, issue commands to query or update the data, and use computer logic to make decisions based on the contents of the database. Through interactions with end-users in the user interface tier, the business tier conducts transactions that update databases in the data tier. Because the database content is dynamic, so are the screens that the business tier presents to the users based on the status of their data records. By the end of this chapter, you will understand how databases power the Internet's data-driven Web sites.

Providing Web Access to Server-Side Databases

Because the Web uses the HTTP protocol, Web sites that provide access to server-side databases are sometimes called HTTP gateways. These gateways enable the business object or script to process forms data received from the user interface tier. While processing this data, the business object opens connections to the appropriate back-end databases in the data tier. Through these connections, the business object can retrieve, update, delete, or insert information in the database.

Common Gateway Interface (CGI)

As you learned in the previous chapter, the first HTTP gateway protocol was the **common gateway interface (CGI)**, which defines the manner in which forms data, cookies, and other kinds of information in a Web request get submitted to the program or script that processes and responds to the request. Any programming language that runs on the server can process the data and respond to the request. The form tag's action parameter tells the server which program to run by providing the HTTP address of the CGI script.

The National Center for Supercomputing Applications (NCSA) invented CGI back in 1993 for use on a UNIX-based Web server, HTTPd, which stands for HTTP daemon. On the server, CGI scripts typically reside in the *cgi-bin* directory, so named because the scripts were binary files. When HTTPd received a request addressed to a CGI script, it stored the forms data in UNIX shell environment variables and launched the CGI program as a separate process. After the script processed the request, HTTPd returned the CGI program's response to the user.

In the beginning, many developers wrote CGI programs in **Perl**, which is a scripting language invented by Larry Wall for people who need to write relatively short programs. More technically inclined programmers write CGI scripts in C, which is the programming language used to develop many popular applications, including Perl. Because CGI is language neutral, however, you can write CGI scripts in any programming language. You can even write CGI scripts in the **Bourne shell language**, which is the original scripting language of the UNIX shell.

The NCSA stopped work on the HTTPd server in 1998, but the code lives on in Apache, which is the most popular Web server on UNIX and Linux systems. For more information about Apache projects, go to www.apache.org.

Server Application Programming Interfaces (SAPIs)

A **Server Application Programming Interface (SAPI)** is a collection of software components used by the business tier to obtain, process, and respond to forms data submitted when end-users interact with the site and make requests through the user interface tier. Instead of running separate CGI scripts to process the forms data, SAPI has integrated libraries of precompiled code containing the software components out of which the Web developer creates the business object. SAPI uses multithreading to enable these components to load once and process multiple user requests, as opposed to CGI scripts, which run out of process, meaning that each incoming request launches a separate code instance. By running in process, SAPI enables the server to handle a higher number of simultaneous users.

Microsoft's brand of SAPI is called **Internet SAPI (ISAPI)**, and Netscape's is called **Netscape SAPI (NSAPI)**. Both brands use dynamic link libraries (DLLs) to contain the precompiled software components that SAPI comprises. Microsoft continues to develop ISAPI, which is a key component of its Web server architecture. Netscape has discontinued work on NSAPI.

FIGURE 12-1 *This source code of an ASP page contains a script that will greet an authenticated user by name. The script is the code between the <% script start and %> script stop tags. Compare this code to Figures 12-2 and 12-3.* ■

Active Server Pages (ASP)

Active Server Pages (ASP) is a Microsoft ISAPI technology that enables Web developers to embed on a Web page server-side scripts written in either the JScript or VBScript programming languages. End-users never see the scripts, which the server executes when the user accesses the page. Instead of seeing an embedded script, end-users view the results of the script's execution. You can understand this by comparing Figures 12-1, 12-2, and 12-3. Figure 12-1 shows the code of an ASP page programmed to greet users when they log in or deny access to unauthorized users. Figure 12-2 shows the HTML source code that the ASP code generates when an authenticated user logs on, and Figure 12-3 shows what the user sees onscreen. If the user tries to see the script via the browser's View | Source option, the user sees only the HTML source code shown in Figure 12-2, which is the result of running the script. Thus, ASP pages provide a way for developers to include scripts on a Web page without the user seeing their code.

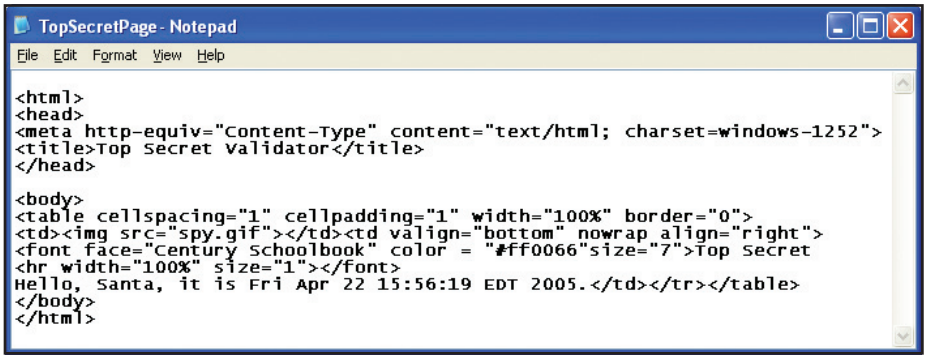


FIGURE 12-2 *This is the HTML code that the server returns when an authenticated user visits the page illustrated in Figure 12-1. Notice how the result of running the script appears in place of the script's source code. Thus, end users never see the server-side code on an Active Server Page.* ■

FIGURE 12-3 *The browser displays this screen when an authenticated user logs on and the Active Server Page illustrated in Figure 12-1 responds. Notice how the script displays the user's name onscreen. If the user pulls down the browser's View menu and chooses Source, the browser displays the HTML code shown in Figure 12-2. Thus, end-users never see the server-side code of the ASP page in Figure 12-1.* ■

Java Servlets and Java Server Pages (JSP)

Sun's Java is a hot technology in spite of the legal battles waged between Sun and Microsoft regarding whether Microsoft has the right to create its own version of Java, and whether Microsoft must include the Java Virtual Machine (JVM) as part of Windows. As this book goes to press, the understanding is that yes, Microsoft can create its own version of Java, and yes, Microsoft will include Sun's Java Virtual Machine (JVM) in future versions of Windows. As a developer, I believe this agreement is beneficial for both sides.

Java is an object-oriented programming language that developers can use to create almost any kind of software. Java code compiles into an intermediary language that executes on any platform running the JVM. Versions of the JVM exist for practically every operating system, including UNIX, Linux, Macintosh, and Windows. Thus, Java code is machine independent.

Earlier in this book, you learned how Java applets can download as part of a Web page and execute on the client in the browser window. On the server side, the most well-known uses of Java are for creating Java servlets and Java Server Pages. A **servlet** is a Java applet that runs on the server instead of in the browser; hence the name *servlet*. The servlet runs in the JVM under a multithreaded environment that can listen for Internet requests to come in and serve multiple users from the same instance of the code.

Java Server Pages (JSP) is an active Web page technology that is Sun's equivalent to Microsoft's ASP. In the midst of the page, the Web developer can write Java code to be run when a user accesses the page. At runtime, when a user hits the JSP, the server executes the code and sends the user the resulting page. JSP is therefore similar to ASP, although JSP runs in the JVM, while ASP runs in Microsoft's ISAPI.

PHP Hypertext Preprocessor (PHP)

The **PHP Hypertext Preprocessor (PHP)** is another active page technology that enables the Web developer to include code on the page that will run on the server, which executes the code before sending the completed page to the user. The command syntax is like that of C and Perl. As an open source Apache project, PHP runs primarily with Apache on UNIX and Linux servers, although versions of PHP are also available for Windows. For more on PHP, go to us3.php.net.

ColdFusion

A product of Macromedia, **ColdFusion** is an active scripting technology that uses its own proprietary scripting language, the ColdFusion Markup Language. Web developers can include in their HTML pages ColdFusion tags, which begin with the letters *CF* for ColdFusion. ColdFusion pages have the filename extension *.cfm*, which stands for ColdFusion markup. When a Web server that is running ColdFusion encounters a *.cfm* page, the server executes the Cold Fusion tags and replaces them with the output generated by the server in executing that code. Thus, end-users never see the CF tags, just as PHP, JSP, and ASP pages strip the server-side script before presenting the page to the user. Macromedia markets ColdFusion components for Web servers running on Windows, UNIX, Linux, Macintosh, HP, and IBM operating systems. For more on ColdFusion, go to www.macromedia.com/software/coldfusion.

ASP.NET

Microsoft's **ASP.NET** is much more than a new version of ASP. Besides letting you include code on a Web page, ASP.NET lets you create code behind the Web page, on so-called *code-behind pages*. These code-behind pages can be part of complete applications with which the user interacts from the browser window, which becomes the Web application's display surface. From my personal experience developing the Serf instructional management system in ASP.NET, I can attest to the .NET framework's elegance and power. Serf consists of several dozen code-behind pages and C# classes that the Serf name space comprises. More information about Serf is at www.serfsoft.com.

ASP continues to be a popular choice for developers who are not quite ready to dive into ASP.NET. My *Advanced Web Design* textbook (ISBN 0-07-256594-2) contains tutorials in both ASP and ASP.NET, offering students a choice of where to jump in. When compared to the other SAPI technologies described in this chapter, however, ASP is more like JSP, PHP, and ColdFusion. The ASP.NET environment is a totally new platform, written from the ground up and offering a choice of programming languages including VBScript, JScript, C++, C#, and J#, which is Microsoft's version of Java. The source code compiles into the Microsoft Intermediate Language (MSIL) that executes in the common language runtime (CLR), which is the execution layer of the .NET framework. For more on the .NET framework, go to www.microsoft.com/net.

Understanding Databases

The three basic kinds of databases are called (1) flat file, (2) relational, and (3) object-oriented. A **flat file database** keeps all of the records in a single file, in which the data elements are separated by a break character such as a comma or a tab. The terms **comma-delimited data** or **tab-delimited data** refer to data stored in this manner. Flat file databases typically contain just one **data table**, which is a database structure that organizes the data into rows and columns. Each row contains one record, and the columns contain the data fields that the record comprises. An example of a flat file database appears in Figure 12-4, which contains the data table illustrated in Figure 12-5. By comparing these two figures, you can see how the commas in the file delimit the fields in the data table.

A **relational database** is a data structure that comprises multiple tables containing primary key columns through which the records in one table can relate (i.e., key) to the data in another table. A data table's **primary key** is a column in which every entry is unique—for example, the UserID column that appears in Figure 12-5. Notice how every user has a unique UserID. In a relational database, other tables can refer to those users by their unique ID. Figure 12-6 shows how a log table, for example, can keep track of the history of each user's visits to the site. In the log table, the UserID column is called a **foreign key**, which is a data field that relates the record to the table in which that same column occurs as a primary key. The color-coding in Figure 12-6 helps you see this relationship, from which the term

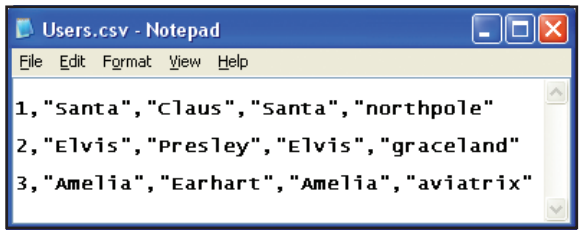


FIGURE 12-4 *A flat file database contains data records typically delimited by commas or tab characters. The database illustrated here consists of comma-delimited data. Compare this database to Figure 12-5, which shows the data table in a columnar format, which is how a script interprets it.* ■

Users : Table

UserID	FirstName	LastName	UserName	Password
1	Santa	Claus	Santa	northpole
2	Elvis	Presley	Elvis	graceland
3	Amelia	Earhart	Amelia	aviatrix

FIGURE 12-5 *In a data table, each row is called a record, and each column contains one of the data fields in that record. In this example, the table contains the UserIDs, names, and passwords of the people who are permitted to visit a Web site. Compare this table to Figure 12-4, which shows this table in a flat file database.* ■

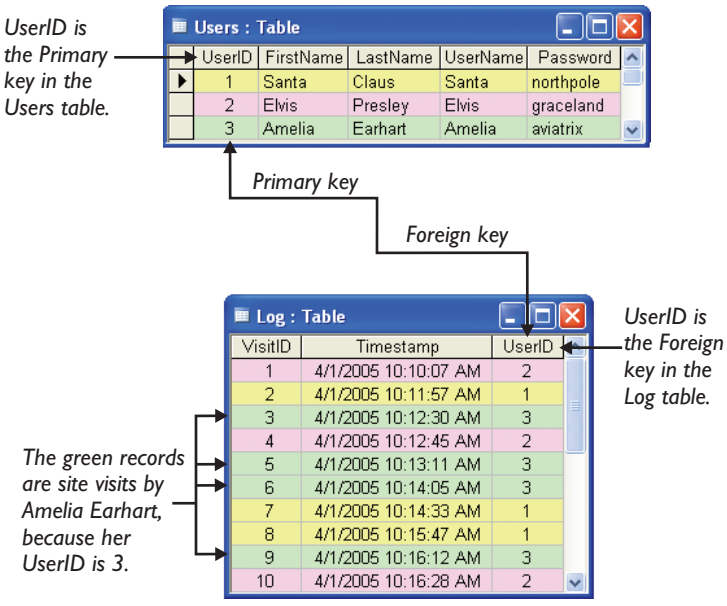


FIGURE 12-6 *In a relational database, the primary key of one table relates its data to the records in another table containing a foreign key column consisting of values from the first table's primary key column. In this example, study the color-coding to see how the log table keeps track of the dates and times when people in the Users table visited the site.* ■

relational database arises. The technical term for the kind of software that powers this kind of database is RDBMS, which stands for **relational database management system (RDBMS)**.

The third and final kind of database is an **object-oriented database**, in which programmers writing code in object-oriented languages can create complex data structures in which one data type can build upon, or inherit, properties from another. The technical term for this kind of database is **object-oriented database management system (ODBMS)**. It is beyond the scope of this book to teach object-oriented programming. This chapter teaches you how to design a relational database, which is the most popular kind of database. By studying this process, you will understand that the simple concept of a key enables the creation of data structures that are infinitely expandable and thereby capable of powering new features as your enterprise grows and offers new services to your users.

Designing a Relational Database

I use an eight-step process to design a relational database. You begin by defining the purpose of the database and creating the data tables. Then you specify the data columns that will contain the data. After defining the relationships among the primary and foreign key columns, you take an imaginary walk through your database. You think about how a typical user will navigate through your application. Then you write a little essay, called a *walkthrough*, describing what will happen in the database as the user walks through the Web site. Writing such an essay helps ensure you have not omitted any essential tables or fields in the design of the database. Considering how you will retrieve the data, make decisions about it, and report results helps ensure that you have included all of the necessary keys that relate the data tables to each other. If a table contains data that cannot be retrieved in the context you need, you can supply the missing foreign key. If a table of items purchased, for example, does not identify the buyer who purchased them, you can add a buyerID column as a foreign key that you can use to indicate who made the purchase. The steps you follow in creating a database in this manner are as follows:

1. Write a paragraph describing the purpose of the database.
2. Make a list of the tables that the database will comprise.
3. List the fields (i.e., data columns) each table will comprise.
4. Indicate the kind of data (i.e., data type) that each column will hold.
5. Indicate which data columns are primary keys. Remember that a primary key field cannot contain any duplicate values; each value in a primary key column must be unique.
6. Indicate which data columns are foreign keys and state the name of the table and data column in which each foreign key is a primary key.
7. Write a walkthrough to make sure you haven't missed something important. Describe how the typical user will enter your site and

navigate its pages. State what will happen in the database as the user submits information. Explain how the data will be used onscreen to create pages whose contents vary depending on the contents of the database. If anything is absent from your database design, writing a complete walkthrough helps you identify the missing elements.

8. If you have any data tables with no keys, ask yourself whether the data really stands alone. If not, add the necessary foreign key column to key the data to the primary key column of the data table to which it relates.

Normalizing a Database

Some database designs are more efficient than others. If a design is inefficient, the database requires more computing resources to process. This slows down the Web site, and the delays frustrate users.

In an efficient design, each table plays one role in the database. In an inefficient design, on the other hand, a table takes on too many roles. **Normalization** is the process of separating a large table fulfilling multiple roles into smaller tables that increase efficiency by serving smaller roles that relate through keys to other tables in the database. A database that has not been normalized has wide tables (i.e., more columns) that require more time to query, sort, and retrieve records. A normalized database contains data tables that are narrower (i.e., fewer columns) and serve efficiently a single purpose that, when related through keys to other tables, enable the database to accomplish its goal more quickly.

Indexing a Database

If a lot of users begin interacting with a database-driven Web site, the data tables can grow quite large. The larger your tables get, the more time it takes the server to search them and return the results of your queries. As the data tables grow in size, queries take longer because the computer has more data to search.

To increase database performance, you can create indexes. An **index** is a database column or collection of columns that the database engine uses to keep the data presorted in the order in which you plan to query it. In the Users table, for example, if you plan to use queries that look up users alphabetically, you create a two-column index based on last name, first name. I can attest from first-hand experience that the performance boost gained from indexing is phenomenal.

Database Design Principles

When designing a database, keep the following principles in mind:

1. Each table should have a column containing a unique row ID number. This enables the column to serve as the table's primary key.

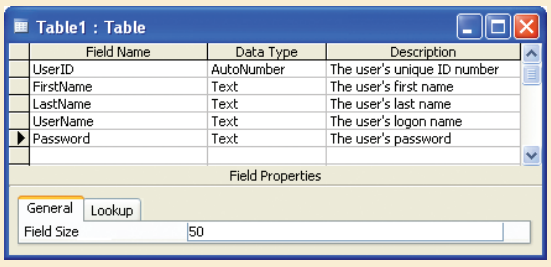
2. A table should store data for a single type of entity. Attempting to store too many different kinds of information in a single table can slow down the database and make it inefficient.
3. A table should avoid columns that are allowed to contain null values. Although you may use null values in isolated cases, they require special handling in the database and should be avoided if possible. If you must have empty values, consider using an empty string, for example, instead of a null value. If you must have null values in a data column, put that column in a separate table so the design of the main table can be simple.
4. A table should not have repeating values. If you need to keep a list of values for one of the items in a table, create another table to hold the list and link it to the item's primary key.
5. A field should have the same meaning in each row of a table. If the meaning of the data column is not consistent, you should create another table to encode the data with a different meaning.
6. Multiple instances of an entity should not be represented as multiple columns. In the Log table, for example, it would be a mistake to limit the number of visits by creating data columns to hold each hit. Suppose you created five columns called Visit1, Visit2, Visit3, Visit4, and Visit5. What if someone wanted to visit the site a sixth time? Such a design accounts for only five visits.

Try This!

Creating a TopSecret Database

In this exercise, you will use Microsoft Access to create a database named TopSecret, which will consist of the user names and passwords of users permitted to access the secret pages at your Web site. Because everyone else will be denied access, the pages will be top secret—hence the name of the database. To create the TopSecret database, follow these steps:

1. When you create a database for use at a Web site, you put it in a separate folder that is not in Web space to prevent unauthorized users from downloading the database. Use the Windows Explorer or My Computer to create a folder named *TopSecret* that is not in Web space. In this example, create the *TopSecret* folder on the root of your hard drive so that its location will be *c:\TopSecret*.
2. Start Microsoft Access, pull down its File menu, and choose the option to create a new blank database. When the File New Database dialog appears, name the database *TopSecret.mdb*, and save it in your *TopSecret* folder. Thus, the complete path/filename of your database is *c:\TopSecret\TopSecret.mdb*.
3. Double-click the option to create a table in Design view. When the Table window opens, create the following four fields. After you type each Field Name, press the Tab key, and a pull-down menu appears in the Data Type column. Pull down that menu and make the field's data type match the ones illustrated here. Also type the following descriptions into each field's description field:



Try This!
continued

4. Pull down the File menu and choose Save; when the Save dialog appears, name the table Users. When Access asks if you want a primary key, say yes. Access will make UserID the primary key, because the autonumber data type makes each user have a unique UserID.
5. Close the table by clicking its close icon. Then double-click the name of the table to open it in data entry mode. Type the following records into the table. You may substitute names of your own for the fictitious entries shown here. At any time, you can repeat this step to modify, add, or delete users from the table. Begin by typing the first name of the first user; Access will populate the UserID column automatically, since it is an autonumber:

Users : Table

	UserID	FirstName	LastName	UserName	Password
	1	Santa	Claus	Santa	northpole
	2	Elvis	Presley	Elvis	graceland
	3	Amelia	Earhart	Amelia	aviatrix
*	(AutoNumber)				

Record: 3 of 3

6. This completes the creation of the Users table. Click its close icon to close it, and say yes when Access asks if you want to save the data you entered in the previous step. You will use this table again in the next Try This! exercise later in this chapter.

Note: *Whenever you name a database, data table, or data field, you should avoid typing spaces or special characters. Out on the Web, situations can arise in which spaces and special characters cause problems in data names and filenames. It is best to avoid these problems by forming the habit of avoiding spaces and special characters when naming databases, data tables, and data fields.*

Accessing Server-Side Databases

Figure 12-7 shows the three-tier Web application model, in which the middle tier business object accesses databases in the data tier through server-

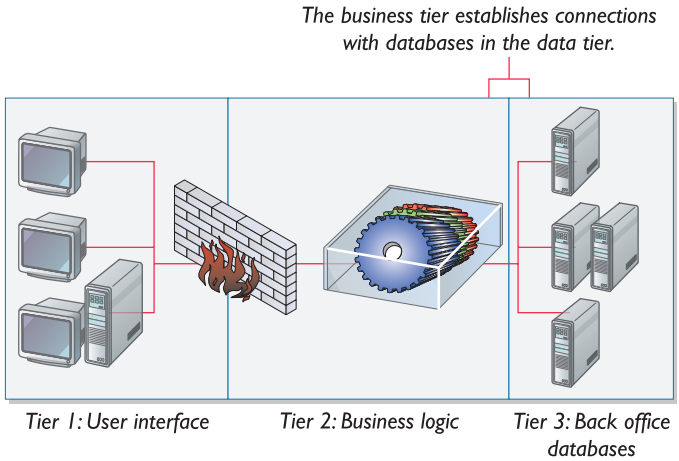


FIGURE 12-7 *The business tier communicates with the data tier by establishing database connections. Depending upon the brand of database and server architecture, these connections may follow OCBC, JDBC, OLE DB, or XML Web Service protocols.* ■

side programs or scripts that open connections to the databases and issue commands to retrieve or manipulate the data. As you might expect, standard protocols exist for making database connections and issuing commands that can retrieve or manipulate the data.

Connectivity Options

The major brands of databases include Microsoft SQL Server, Oracle 9i, Borland Interbase, IBM DB2, and iPlanet Application Server. To make these database products work across a broad range of application development environments and scripting languages, the computer industry has developed standards defining alternate means for connecting with databases. When creating a data-driven application, the Web developer chooses the connection method that is most appropriate for the task at hand. The two primary connection options are:

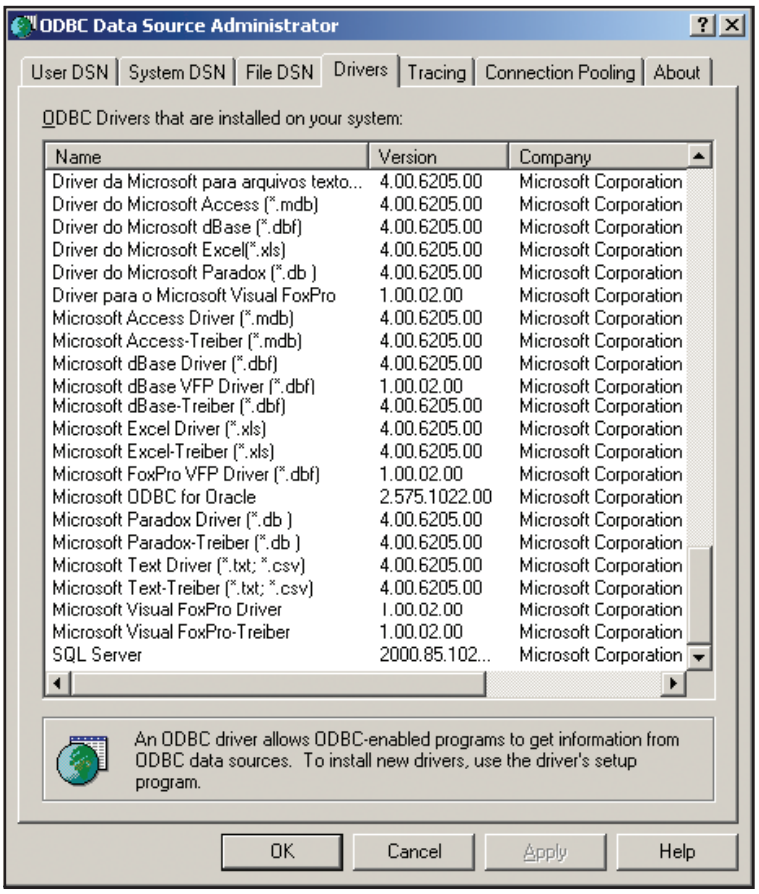


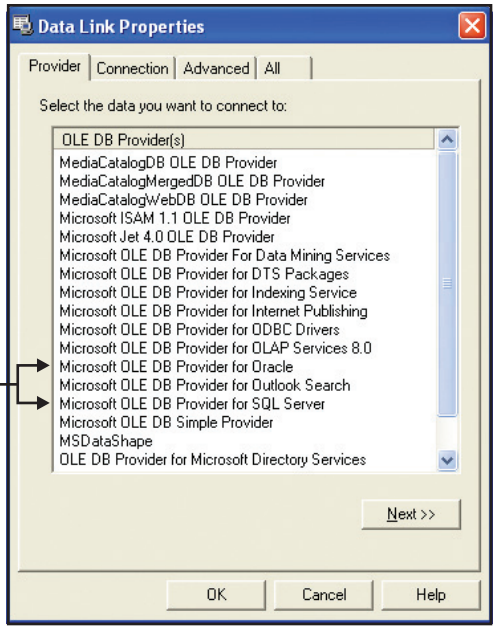
FIGURE 12-8 *Due to the popularity and industry-wide acceptance of the open database connectivity (ODBC) standard, an ODBC driver exists for almost every brand of database. Pictured here are the ODBC drivers that come preinstalled for use with Windows 2003 Server. Notice how Microsoft provides a driver for Oracle, which is the largest competitor to Microsoft's SQL Server enterprise database product.* ■

- ODBC **Open database connectivity (ODBC)** is a standard database access method created by Microsoft based on the Call-Level Interface (CLI) specifications from the X/Open and ISO/IEC database APIs. The goal is to make it possible for any application to access any database, regardless of its vendor. ODBC accomplishes this goal through database drivers that translate queries and commands issued by the application into a format that the database can process. An ODBC driver exists for almost every brand of relational database. Figure 12-8 shows the ODBC Data Source Administrator controls that a Windows server administrator uses to configure a company's ODBC driver for use with a data source name (DSN), through which an application will access the data.

- JDBC Developed by JavaSoft for Sun Microsystems, **Java database connectivity (JDBC)** enables Java programs to communicate with SQL-based database systems. A JDBC-ODBC bridge can translate JCBC calls into ODBC calls, but Java developers are encouraged to use native JDBC drivers that communicate directly with the database, thereby avoiding the overhead introduced by the bridge.

OLE DB Data Sources

A vast amount of information exists outside the traditional DBMS data sources accessed through ODBC and JDBC connections. Microsoft invented a data source technology called **object linking and embedding database (OLE DB)** to enable Windows applications to open a wide variety of connections to data stored not only in ODBC data sources, but also in file systems, media catalogs, e-mail, spreadsheets, active directories, project management tools, and XML files. Figure 12-9 shows the wide range of data source connections that OLE DB enables. Through these connections, the application can issue queries as if the data source were a database. Hence the name, OLE DB.



Oracle is the largest competitor to SQL Server, which is Microsoft's premier enterprise RDBMS.

FIGURE 12-9 *The business tier communicates with the data tier by establishing database connections. Depending upon the brand of database and server architecture, these connections may follow OCBC, JDBC, OLE DB, or XML Web Service protocols.* ■

ActiveX Data Objects (ADO)

On the Windows operating system, **ActiveX Data Objects (ADO)**) is an API that enables business objects to make connections and issue commands against many different kinds of data sources. Because Windows is a prevalent operating system, I am going to show you examples of ADO programming to give you an idea of what it is like to create a server-side script.

Creating an ADO Connection Object

True to its name, the **connection object** is the ADO component that connects you to the database. Once you have the connection open, you can execute queries that insert, update, retrieve, or delete records from the database. The following code opens the connection in the two popular ASP programming languages, namely, JScript and VBScript.

JScript version

L 12-1
```
connection = Server.CreateObject("ADODB.Connection");
connection.mode = 3;  //read-write mode
sConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;"
    + "Data Source=YourDrive:/YourFolder/YourDatabase.mdb";
connection.Open(sConnectionString);
```

The name of your folder and your database file go here.

VBScript version

L 12-2
```
connection = Server.CreateObject("ADODB.Connection")
connection.mode = 3  //read-write mode
sConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;"_
     & "Data Source=YourDrive:/YourFolder/YourDatabase.mdb"
connection.Open(sConnectionString)
```

As you can see, the JScript and VBScript coding is very similar because both languages use the same ADO components, which many other Microsoft programming languages can also use. The connection object is very powerful, because you can write connection strings to open almost any kind of data source from any database vendor. Here are some examples of different kinds of connection strings:

Open an Access database on a local drive:

L 12-3
```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:/YourFolder/YourDatabase.mdb
```

Open an SQL Server database:

L 12-4
```
server=YourServerName;database=YourDatabaseName;uid=userid;pwd=password
```

Open an Oracle database:

L 12-5
```
Provider=msdaora;Data Source=YourOracleServer;USER ID=userid;PASSWORD=password
```

Open an Excel spreadsheet on a local drive:

L 12-6
```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:/YourFolder/YourExcel.xls
```

Access plaintext files in a local folder:

L 12-7
```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:/YourFolder/;Extended
Properties="text;HDR=Yes;FMT=Delimited"
```

Open a data store from a data source name (DSN):

L 12-8
```
DSN=yourDSN;uid=userid;pwd=password
```

Using the Structured Query Language (SQL)

Once you have the ADO connection established, you can issue SQL commands against the database. The **Structured Query Language (SQL)** is an international standard that defines the syntax for issuing commands that can query, update, insert, or delete records in a database. SQL is a rich language that contains dozens of commands. While it is beyond the purpose of this book to teach SQL commands, you can get a feeling for how SQL works by studying the functions of the SELECT command, which you use to retrieve records from a data table. The simplest form of the SELECT statement is:

** is a wildcard that selects everything.*

L 12-9
```
SELECT * FROM TableName
```

Replace `TableName` with the actual name of the data table.

Instead of using * to select everything, you can specify the names of the columns you want to retrieve. To select a single column, the command syntax is

L 12-10
```
SELECT ColumnName FROM TableName
```

Replace `ColumnName` with the name of the column you want to retrieve.

Replace `TableName` with the name of the data table containing that column.

To select more than one specific column, you type a comma-separated list of the names of the columns you want to retrieve, as in:

There is no limit to the number of columns you can specify; to add another column, type a comma here followed by the name of the column.

L 12-11
```
SELECT ColumnName1, ColumnName2, ColumnName3 FROM TableName
```

Replace `ColumnName1`, `ColumnName2`, and `ColumnName3` with the names of the columns you want to retrieve.

Unless you specify otherwise, SQL commands return results in the order in which they were stored in the database. To change the order in which the results are returned, you can add an ORDER BY clause to the SELECT command. The syntax is

L 12-12
```
SELECT * FROM TableName ORDER BY ColumnName
```

*You can replace the * with a comma-separated list of specific column names.*

Replace `ColumnName` with the name of the column that will control the ordering.

If more than one column is involved in the ordering, you can specify a comma-separated list of columns to order by. The sorting will be based on the columns you specify, with the leftmost column sorted first. The syntax is

Replace `ColumnName1` with the name of the column you want sorted first.

Replace `ColumnName3` by the name of the column you want sorted third.

L 12-13
```
SELECT * FROM TableName ORDER BY ColumnName1, ColumnName2, ColumnName3
```

`ColumnName2` is the name of the column that will be sorted.

There is no limit to the number of columns you can specify; to add another column, type a comma here followed by the name of the column.

One of the most important features of a database is its ability to find information. In the SQL language, this kind of searching is provided by the WHERE clause. By filtering out unwanted information, the WHERE clause lets you focus on the data you are looking for. It is easy to add a WHERE clause to the SELECT statement. The syntax is

L 12-14
```
SELECT * FROM TableName WHERE ColumnName = DataValue
```

The WHERE clause acts like a filter that lets you focus on the data you are looking for.

You can replace the = with other operators including < for less than, > for greater than, and <> for not equal.

In the Try This! exercise that follows, you encounter a situation where you need to make a WHERE clause to handle more than one condition. You can accomplish that with an AND, which you use to specify another condition for the search. The syntax is

L 12-15
```
SELECT * FROM TableName WHERE ColumnName = DataValue AND ColumnName2 = DataValue2
```

Besides AND, you can use OR and AND NOT and OR NOT.

Try This!

Selecting TopSecret Users

In the first Try This! exercise earlier in this chapter, you created the TopSecret database consisting of the names of users who will be permitted to access your top secret pages. The TopSecret database is a good place to practice creating SQL queries. Follow these steps:

1. Start Microsoft Access. Pull down the File menu and open the TopSecret database containing the data you entered in the Try This! exercise earlier in this chapter.
2. Under Objects, click Queries, or pull down the View menu and choose Database Objects | Queries.
3. Double-click the option to Create query in Design view. The Show Table window opens.
4. You do not need to use the Show Table window because you will be typing the SQL command manually. Close the Show Table window.
5. Pull down the View menu and choose SQL view; the SQL view appears with a query partially started.
6. Replace the partially started query with the command you want to practice. In this example, type the query:

   ```
   SELECT * FROM Users ORDER BY LastName, FirstName
   ```

7. Click the run icon to execute the query; the results of issuing the query appear onscreen.
8. Pull down the View menu and choose SQL view again. Modify the query to read as follows:

   ```
   SELECT * FROM Users where UserName = 'Santa' AND Password = 'northpole'
   ```

9. Click the run icon to execute the query; the results of issuing the modified query appear onscreen. If a user in your table is named Santa and has the password northpole, that user's data appears onscreen. If no user was listed under that name and password, no data would be returned. You will use that strategy later in this chapter's final Try This! exercise, in which you will create a script that will deny access to users who do not appear in this table.
10. Repeat the last two steps to practice this process of typing an SQL command and viewing the results. Here are some additional queries you can try:

    ```
    SELECT LastName, FirstName FROM Users where UserID < 3
    SELECT * from Users where UserName = 'anybody' and password = 'anything'
    ```

Creating Data-Driven Web Pages

A **data-driven Web page** is an HTML document in which part or all of the content derives from or depends on records or relationships in one or more databases. In the typical ASP page, the Web developer includes a server-side script in the HTML body of a page. At runtime, the server executes the script, which uses the ADO connection object to open a database connection. Through this connection, the script executes SQL statements that query the database.

The queries return data in a set of records called a **recordset**. The script reads the data in the recordset. Through computer logic known as IF-THEN statements, the script makes decisions based on the content of the data. If a user's bank account contains a negative balance, for example, the script can deny a pending sale and respond to the user with a message explaining the account is overdrawn. At runtime, when the user accesses the ASP page, the server replaces the script with the result of the script's execution. If the user inspects the source code of the page via the browser's View | Source menu, the user will see the HTML code of the insufficient funds message in place of the JScript or VBScript code that created the message.

Without getting overly technical, the last few pages of this chapter walk you through some ASP code that reads data from a recordset and makes decisions based on the status of this data. In the concluding Try This! exercise, you have the opportunity to try this code yourself.

Reading Data from a Recordset

The ADO recordset object contains methods that enable a script to move through its contents, record by record, and obtain information to display directly onscreen or to use in making decisions that result in deciding what to display onscreen. The following script, for example, reads the names of registered users from a Users table and prints the names onscreen. The script contains documentation explaining how it works. In an ASP script, each line of documentation begins with the symbol //. Reading this documentation gives you a sense of the underlying concepts that programmers use in creating these kinds of scripts. If you want to learn more about ASP scripting, my textbook *Advanced Web Design* teaches JScript and VBScript in more detail.

JScript version

L 12-16

```
//Create and open the database connection
connection = Server.CreateObject("ADODB.Connection");
connection.mode = 1; //read only mode
sConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;"
    + "Data Source=c:/TopSecret/TopSecret.mdb";
connection.Open(sConnectionString);
//Create and execute the SQL query
sQuery = "SELECT * FROM Users ORDER BY LastName, FirstName";
rsResults = connection.Execute(sQuery);
//While the recordset is not (!) at the end of file (EOF)
while (! rsResults.EOF)
```

```
{
     sFirstName = rsResults("FirstName");
     sLastName = rsResults("LastName");
     Response.Write(sLastName + ", " + sFirstName + "<br>");
     //Move to the next record in the recordset
     rsResults.MoveNext;
}
connection.Close();
%>
```

VBScript version

L 12-17

```
//Create and open the database connection
Set connection = Server.CreateObject("ADODB.Connection")
connection.mode = 1 //read only mode
sConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" _
    & "Data Source=c:/TopSecret/TopSecret.mdb"
connection.Open(sConnectionString)
//Create and execute the SQL query
sQuery = "SELECT * FROM Users ORDER BY LastName, FirstName"
Set rsResults = connection.Execute(sQuery)
//While the recordset is not at the end of file (EOF)
Do Until rsResults.EOF
     sFirstName = rsResults("FirstName")
     sLastName = rsResults("LastName")
     Response.Write(sLastName & ", " & sFirstName & "<br>")
     //Move to the next record in the recordset
     rsResults.MoveNext
Loop
//Close the connection
connection.Close()
```

Using Logic to Make Decisions

The true power of data-driven Web pages comes from the script's ability to make decisions based on the current contents of the database. One of the most important decisions is to decide whether to permit or deny access to a page. Imagine a situation in which a Login form posts to a script the user name and password of someone attempting to access your site. You want only members of the Users table in the TopSecret database to have access to the page. Think about the steps a script must take to decide whether a user should be allowed in. Consider it for a moment, and then study the following steps, which describe what the script would do:

1. Retrieve the user name and password from the incoming form data.
2. Open a connection to the TopSecret database.
3. Issue an SQL command to query the database. This query asks the database to retrieve the record containing the user name and password that the user entered on the Login form.

4. Use an IF-THEN statement to make the following decision based on the contents of the recordset that the query in step 3 returns:
   a. If the recordset is empty, this user is not valid, so you deny access.
   b. If the recordset contains the requested record, the user is allowed in. Set the authentication cookie and send the user to the welcome screen.

A script that accomplishes these four steps follows. Study the documentation (i.e., the green lines beginning with the // symbol) to get a sense of how the script works. If the user is allowed in, the script sets an authentication cookie that subsequent pages can check to decide whether the user should be permitted access. In the Try This! exercise that concludes this section, you learn how to put a one-line script on any page that you want to make top secret by denying access to users who do not belong to the TopSecret database. This exercise does not require you to write any database scripts. Rather, you follow the step-by-step instructions to download the scripts from the book's Web site and use them to make selected pages top secret.

JScript version

```
//Get the data coming in from the Login form
sUsername = Request.Form("Username");
sPassword = Request.Form("password");
//Create the database connection object
connection = Server.CreateObject("ADODB.Connection");
connection.mode = 1; //read only mode
sConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;"
    + "Data Source=c:/TopSecret/TopSecret.mdb";
connection.Open(sConnectionString);
//Create the SQL query
sQuery = "SELECT * FROM Users WHERE UserName = '" + sUsername
   + "' AND Password = '" + sPassword + "'";
//Execute the query and get its results into a recordset
rsResults = connection.Execute(sQuery);
//Decide whether the user is allowed in
if (rsResults.EOF)
{
     Response.Write("<br>You are not a top secret user.");
     //Negate any validation cookie
     Response.Cookies("Validation") = "-1";
}
else
{
     sUserID = rsResults("UserID");
     //Set the validation cookie
     Response.Cookies("Validation") = sUserID;
     //Send the user to the Welcome page
     Response.Redirect("Welcome.html");
}
connection.Close();
```

VBScript version

```
//Get the data coming in from the Login form
sUsername = Request.Form("Username")
sPassword = Request.Form("password")
//Create the database connection object
Set connection = Server.CreateObject("ADODB.Connection")
connection.mode = 1 //read only mode
sConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" _
    & "Data Source="c:/TopSecret/TopSecret.mdb"
connection.Open(sConnectionString)
//Create the SQL query
sQuery = "SELECT * FROM Users WHERE UserName = '" & sUsername _
   & "' AND Password = '" & sPassword & "'"
//Execute the query and get its results into a recordset
Set rsResults = connection.Execute(sQuery)
//Decide whether the user is allowed in
if (rsResults.EOF) then
Response.Write("<br>You are not a top secret user.")
     //Negate any validation cookie
     Response.Cookies("Validation") = "-1"
else
     sUserID = rsResults("UserID")
     //Set the validation cookie
     Response.Cookies("Validation") = sUserID
     //Send the user to the Welcome page
     Response.Redirect("Welcome.html")
end if
connection.Close()
```

Try This!

Creating Top Secret Web Pages

This exercise teaches you to use ASP to make Web pages deny access to users who are not in the TopSecret database you created in an earlier Try This! exercise. To run these pages, however, you must have the IIS Web server running on your computer. If you do not have IIS, you can read through this exercise to see how it works, but you will not be able to run it on your computer. If you have Windows NT Workstation, Windows 2000 Pro, Windows 2003, Windows XP Pro, or Windows XP Media Edition, you can install IIS from the Add/Remove Windows Components option under Add or Remove Programs in the Windows Control Panel. Once you have IIS running on your computer, you can create the ASP page by following these steps:

1. Use My Computer or the Windows Explorer to move the *website* folder into your computer's Web root folder, which is probably located at c:\inetpub\wwwroot.
2. Into your *website* folder, download from this book's Web site the files *login.html, TopSecret.asp*, and *TopSecretValidator.js*. Follow the onscreen instructions explaining how to download instead of run the executable files.
3. Use Notepad to open the HTML of any page you want to protect. In this example, open the file *hello.html*, *mountains.html*, or *resume.html*. Immediately after the `<body>` tag, paste the following code:

   ```
   <script src="TopSecretValidator.js" language=javascript></script>
   ```

4. After saving the file you modified in the previous step, use your browser to open it. Because you have not yet logged on, the script will redirect you to the login page, which prompts you to type your user name and password. If you respond with the user name and password of a user in your TopSecret database, you will get in. If you respond otherwise, the script will deny access until you log on as a TopSecret user.
5. Some pretty nifty code runs behind the scenes of this exercise. First, the *TopSecretValidator.js* script checks whether the user has a validated authentication cookie. If not, it redirects the user to the *Login.html* page, which prompts the user for a user name and password. When the user clicks the Login button, the form submits its data to the *TopSecret.asp* page.
6. The *TopSecret.asp* page uses a script that creates an ADO connection object, through which it queries the TopSecret database. The code on the *TopSecret.asp* page is self-documenting. To study how the code works, use Notepad to open the *TopSecret.asp* file, and read the documentation contained in the script.

Chapter 12 Review

Chapter Summary

After reading this chapter and completing the step-by-step tutorials and Try This! exercises, you should understand the following facts about the Internet:

Providing Web Access to Server-Side Databases

- The first HTTP gateway protocol was the Common Gateway Interface (CGI), which defines the manner in which forms data, cookies, and other kinds of information in a Web request get submitted to the program or script that will process and respond to the request. Perl, C, and the UNIX shell were the programming languages of the first generation of CGI scripts.
- A Server Application Programming Interface (SAPI) is a collection of software components used by the business tier to obtain, process, and respond to forms data submitted when end users interact with the site and make requests through the user interface tier. Microsoft's brand of SAPI is Internet SAPI (ISAPI), and Netscape's is Netscape SAPI (NSAPI).
- Active Server Pages (ASP) is a Microsoft ISAPI technology that enables Web developers to embed on a Web page server-side scripts written in either the JScript or VBScript programming languages. End users never see the scripts, which the server executes when the user accesses the page. Instead of seeing an embedded script, end users view the results of the script's execution.
- Java is an object-oriented programming language that developers can use to create almost any kind of software. Java code compiles into an intermediary language that executes on any platform running the JVM. Versions of the JVM exist for practically every operating system, including UNIX, Linux, Macintosh, and Windows. Thus, Java code is machine independent.
- A Java Server Page (JSP) is an active Web page technology that is Sun's equivalent to Microsoft's ASP. In the midst of the page, the Web developer can write Java code to be run when a user accesses the page. At runtime, when a user hits the JSP, the server executes the code and sends the user the resulting page. JSP is therefore similar to ASP, although JSP runs in the JVM, while ASP runs in Microsoft's ISAPI.
- The PHP Hypertext Preprocessor (PHP) is another active page technology. The command syntax is like that of C and Perl. As an open source Apache project, PHP runs primarily with Apache on UNIX and Linux servers, although versions of PHP for Windows are also available.
- ColdFusion is an active scripting technology that uses its own proprietary scripting language called the ColdFusion Markup Language.
- Microsoft's ASP.NET lets you create code behind the Web page, on so-called *code-behind pages*. These code-behind pages can be part of complete applications with which the user interacts from the browser window, which becomes the Web application's display surface.

Understanding Databases

- Three basic kinds of databases are (1) flat file, (2) relational, and (3) object-oriented. A flat file database keeps all of the records in a single file, in which the data elements are separated by a break character such as a comma or a tab.
- A relational database management system (RDBMS) is a data structure that contains multiple tables containing primary key columns through which the records in one table can relate (i.e., key) to the data in another table, in which the related column is called a *foreign key*.
- In an object-oriented database management system (ODBMS), programmers writing code in object-oriented languages can create complex data structures in which one data type can build upon, or inherit, properties from another.
- To design a relational database, you (1) describe its purpose, (2) list its tables, (3) list the fields (i.e., data columns) in each table, (4) define the data types of each column, (5) define the primary keys, (6) identify the foreign key columns, (7) write a walkthrough, and (8) ask yourself if any relations are missing. This design process is not a

standard protocol that is asked on the CIW exam; rather, this process is how I design my databases.

- Normalization is the process of separating a large table fulfilling multiple roles into smaller tables that increase efficiency by serving smaller roles that relate through keys to other tables in the database. A database that has not been normalized has wide tables (i.e., more columns) that require more time to query, sort, and retrieve records. A normalized database has data tables that are narrower (i.e., fewer columns) and serve efficiently a single purpose which, when related through keys to other tables, enables the database to accomplish its goal more quickly.
- An index is a database column or collection of columns that the database engine uses to keep the data presorted in the order in which you plan to query it. Creating an index can lead to a significant boost in database performance.

Accessing Server-Side Databases

- Open database connectivity (ODBC) is a standard database access method created by Microsoft and based on the Call-Level Interface (CLI) specifications from the X/Open and ISO/IEC database APIs. An ODBC driver is available for almost every brand of relational database.
- Developed by JavaSoft for Sun Microsystems, Java database connectivity (JDBC) enables Java programs to communicate with SQL-based database systems.
- Object linking and embedding database (OLE DB) connections enable Windows applications to open a wide variety of connections to data stored not only in ODBC data sources, but also in file systems, media catalogs, e-mail, spreadsheets, active directories, project management tools, and XML files.
- On the Windows operating system, ActiveX Data Objects (ADO) is an API that enables business objects to make connections and issue commands against many different kinds of data sources.
- The connection object is the ADO component that connects you to the database. Once you have the connection open, you can execute SQL commands.
- SQL stands for Structured Query Language, which is an international standard that defines the syntax for issuing commands that can query, update, insert, or delete records in a database.

Creating Data-Driven Web Pages

- SQL queries return data in sets of records called a *recordset*. The script of a data-driven Web page reads the data in the recordset.
- The ADO recordset object contains methods that enable a script to move through its contents, record by record, and obtain information to display directly onscreen or to use in making decisions that will result in deciding what to display onscreen.
- Through computer logic known as IF-THEN statements, the script makes decisions based on the content of the data.
- At runtime, when the user accesses the data-driven Web page, the server replaces the script with the result of the script's execution. If the user inspects the source code of the page via the browser's View | Source menu, the user views the HTML output of the script instead of the server-side code that generated this content.

Key Terms

Active Server Pages (ASP) *(4)*
ActiveX Data Objects (ADO) *(13)*
ASP.NET *(6)*
Bourne shell language *(3)*
ColdFusion *(6)*
comma-delimited data *(7)*
common gateway interface (CGI) *(3)*
connection object *(13)*
data table *(7)*
data-driven Web page *(17)*
flat file database *(7)*
foreign key *(7)*
index *(9)*
Internet SAPI (ISAPI) *(3)*
Java *(5)*
Java database connectivity (JDBC) *(12)*
Java Server Pages (JSP) *(5)*
Netscape SAPI (NSAPI) *(3)*
normalization *(9)*
object-oriented database *(8)*
object-oriented database management system (ODBMS) *(8)*
object linking and embedding database (OLE DB) *(13)*

open database connectivity (ODBC) *(12)*
Perl *(3)*
PHP Hypertext Preprocessor (PHP) *(6)*
primary key *(7)*
recordset *(17)*
relational database *(7)*
relational database management system (RDBMS) *(8)*
Server Application Programming Interface (SAPI) *(3)*
servlet *(5)*
Structured Query Language (SQL) *(14)*
tab-delimited data *(7)*

Key Terms Quiz

1. The first HTTP gateway protocol was the _____, which defines the manner in which forms data, cookies, and other kinds of information in a Web request get submitted to the program or script that will process and respond to the request.
2. Microsoft's brand of SAPI is called _____.
3. _____ is a Microsoft SAPI technology that enables Web developers to embed on a Web page server-side scripts written in either the JScript or VBScript programming languages.
4. Java code compiles into an intermediary language that executes on any platform running the _____.
5. _____ is an active Web page technology that is Sun's equivalent to Microsoft's ASP.
6. _____ is a CGI scripting language invented by Larry Wall for people who need to write relatively short programs.
7. A(n) _____ keeps all of the records in a single file, in which the data elements are separated by a break character such as a comma or a tab.
8. A(n) _____ is a data structure that contains multiple tables containing primary key columns through which the records in one table can relate (i.e., key) to the data in another table.
9. A data table's _____ is a column in which every entry is unique; it keys to the _____ column in the related table.
10. In a(n) _____, programmers writing code in object-oriented languages can create complex data structures in which one data type can build upon, or inherit, properties from another.

Multiple-Choice Quiz

1. What runs in the JVM under a multithreaded environment that can listen for Internet requests to come in and serve multiple users from the same instance of the code?
   a. Applet
   b. inetd
   c. ISAPI
   d. Servlet
2. What is the open source Apache project's active page technology that enables the Web developer to include code on the page that will run on the server, which executes the code before sending the completed page to the user?
   a. ASP
   b. ColdFusion
   c. JSP
   d. PHP
3. Which active scripting technology uses proprietary tags that begin with the letters CF?
   a. ASP
   b. ColdFusion
   c. JSP
   d. PHP
4. Which programming environment enables you to create code behind the Web page, on so-called code-behind pages, which can be part

of complete applications with which the user interacts via the browser window?
   a. ASP.NET
   b. ColdFusion
   c. Perl
   d. PHP

5. What kind of database typically stores its data in comma-delimited or tab-delimited records?
   a. Flat file
   b. Object-oriented
   c. OLE DB
   d. Relational

6. The process of separating a large table fulfilling multiple roles into smaller tables that increase efficiency by serving smaller roles that relate through keys to other database tables is called
   a. Datamation
   b. Economization
   c. Normalization
   d. Standardization

7. What is a database column or collection of columns that the database engine uses to keep the data presorted?
   a. Catalog
   b. Directory
   c. Index
   d. Query

8. What is a standard database access method created by Microsoft based on the Call-Level Interface (CLI) specifications to make it possible for any application to access any database, regardless of its vendor?
   a. JDBC
   b. ODBC
   c. OLE DB
   d. RDBMS

9. What is the international standard that defines the syntax for issuing commands that can query, update, insert, or delete records in a database?
   a. CFML
   b. PHP
   c. SQL
   d. XML

10. Which clause enables an SQL statement to filter out unwanted information?
   a. FROM
   b. ORDER BY
   c. SELECT
   d. WHERE

Essay Quiz

1. In terms of efficient use of operating system resources, what is the primary advantage of a SAPI over the original CGI?
2. In your own words, explain the fundamental difference between a Java applet and a Java servlet.
3. Explain what ASP, JSP, and PHP have in common in terms of where the scripts reside, where they execute, and what happens at runtime to prevent the end-user from seeing the scripts via the browser's View | Source option.
4. Refer to the summary of database design principles presented in this chapter. Which rule does the following data table's design violate?

Data Table: Children

ParentID	Child1	Child2	Child3	Child4	Child5
1	Fred	Mary Ann	Linda	Bobby	Thomas
2	John	Michael	Susan	Chrystal	Amber
3	Samuel	Jacob	Julian	Alexis	Tommy

5. Suppose you wanted to write a script to judge whether a user answered a test question right or wrong, and you need to decide whether to write the judging code in JavaScript, JScript, or VBScript. Which would be the more secure scripting language(s) in which to write such code? Explain why this language is more secure.

Lab Projects

• Lab Project 12-1: Database Analysis and Design

Imagine that you work for a school or company that needs to create a Web-based discussion forum that will contain confidential discussions by different working groups within your organization. Your employer has assigned you the task of analyzing the needs, specifying the features, and designing the relational database that will contain these discussions. Based on your design, a programming team will write the scripts that power the discussions. Use your word processor to write an essay in which you present your database analysis and design. In formulating this design, consider the following issues:

- **Needs analysis** Before you can design the data tables that the discussion database will comprise, you need to make an outline of the discussion features your coworkers require. Make a preliminary list and consult with your coworkers to make sure the list includes all the discussion features they think are needed.
- **Data table design** Make a list of the tables that the database will comprise. In each table, indicate which column is the primary key, and state which columns in other tables, if any, have a foreign key relationship. In designing these tables, remember that you will need columns to identify the author of each message.
- **Data type definition** Specify the type of data that will reside in each column, such as integer, text, or date/timestamp. Most databases can handle text fields up to 5,000 characters long, which is plenty long for storing discussion messages. If your design permits users to upload file attachments, however, you should plan to store the uploaded files in the server's file system instead of in the database. It is more efficient to store in the database the path/filename of the uploaded file, rather than the file itself.
- **Normalization** Beginners tend to create one or two data tables that are very wide—that is, contain many columns. Relational database engines normally work better when you store data in smaller tables that use primary and foreign key relationships. Tables that are too wide tend to store redundant data. Especially when you find a table repeating data, such as storing the user's first and last name in each discussion record, you should create a secondary table that stores the redundant information more efficiently.
- **Indexing** For optimal database performance, you should specify the sort order for each table and have the database programmers create indexes based on the sort order. Make the sort order be the order in which the discussions will most often retrieve the data. An example of a sort order is DiscussionID, MessageID, ResponseID.
- **Walkthrough** Write a walkthrough to make sure you haven't missed something important. Describe how the typical user will enter the discussion forum and use its options. Explain what will happen in the database as the user posts and responds to messages. If your design permits users to upload file attachments, describe how the database will store them. If the walkthrough uncovers anything missing from your database design, correct the design by supplying the missing elements.
- **Search engines** To ensure you have not missed something important, use Google or Yahoo to search for the following keywords: discussion forum database design. Perusing other database designs can provide ideas for features you may wish to include, and help guard against omitting something important.

If your instructor asked you to hand in the database analysis and design, make sure you put your name at the top of the essay, then copy it onto a disk or follow the other instructions you may have been given for submitting this assignment.

• Lab Project 12-2: Database Backup and Recovery Planning

Without proper database backup and recovery planning, all the hard work and effort you put into creating a data-driven Web site can go down the drain in the event of a hard disk crash, destructive virus attack, or other catastrophic data loss. Imagine that your employer has tasked you with developing a database backup and recovery plan. Use your word processor to write an essay in which you present this plan. In formulating the backup and recovery strategy, consider the following issues:

- **Backup schedule** In the event of catastrophic data loss, you restore the database from the most recent backup. Every transaction recorded since the timestamp of the backup is lost. In deciding how often to back up the database (e.g., hourly, daily, weekly), take into account how frequently the data tends to change and the financial or operational impact of data loss.
- **Off-site backups** Planning where to keep the backup is just as important as deciding how often to make it. A catastrophic fire or explosion, for example, could destroy all of the data in your building, including all of the backups residing there. It is very important, therefore, to keep extra copies of the backup in secure places off site. Large companies, for example, can keep backups in different buildings, cities, states, or countries. Small businesses can keep backups at home as well as in the office.
- **Recovery methods** Create a written procedure explaining how your organization will go about recovering data from the backups in the event of different kinds of failures. Include a range of scenarios from a hard disk crash on the server to total destruction of the site housing the server. Thinking about disasters can be depressing. Remember Murphy's Law: if you have a disaster recovery plan, you will never need it.
- **Emergency hosting** In the event of a catastrophic disaster that destroys the machine room, you will need a plan for hosting the site at a different location on an emergency basis. Regardless of how you feel about Murphy's Law, you must include recovery from catastrophic disaster in your plans.
- **Recovery testing** Unless the recovery procedure works, the backup is of no use. You should recommend that your organization conduct a test in which you format the drives on a spare server, and then rebuild that machine to perform the same function as your production system. Perform tests to make sure the rebuilt system is functioning properly with all the data intact.
- **Best practices** Consider best practice advice you can discover via Google or Yahoo by searching for the following keywords: database backup recovery planning.

If your instructor asked you to hand in the database backup and recovery plan, make sure you put your name at the top of the essay, then copy it onto a disk or follow the other instructions you may have been given for submitting this assignment.