CHAPTER 45

Accessing External Databases

In This Chapter

Understanding databases

The Web-to-database interface

Options for accessing database data

External database access without programming

External database access with SQL

The Web-to-database interface is the best thing to happen to the Web since inline graphics. So many ways exist to connect your Web page to a database that this chapter can't begin to discuss all the options available today. Instead, this chapter discusses how the Web-to-database interface works, what software types are available for this interface, and how they work. Finally, the chapter focuses on four of the technologies that work well.

Understanding Databases

Databases have been around for a long time. Properly used, databases are an efficient way to store data so redundancy is removed. On UNIX, most databases are text files with indexes. This chapter should be of particular interest to you if your organization already has data in a database.

Regardless of how much data is in your database, the most important things you should know about that data is how to get at it (meaning the unique keys to the data) and how to modify it (again using the unique keys).

Tables

A database is made up of *tables*. Tables hold *rows* of data. Rows of data in a database are similar to rows of data in a spreadsheet. Tables also have *fields*. Each field in a database is similar to a column of data in a spreadsheet. If one field is for e-mail addresses, then for every row of data, either an e-mail address will be in that field or it will be blank. Tables are related to each other with keys.

Keys

A *key* is a value associated with one row in a table. Usually, a key is unique, meaning a key can only be associated with one row. A social security number would be a unique key. While a last name could be used as a key, it would not always be a unique key. Frequently, a database assigns its own unique value to each row to ensure that all keys are, indeed, unique.

Relationships

So far, nothing about databases makes them any more powerful than spreadsheets. It is the relationships you can create between tables that makes them powerful. A *normalized* database consists of multiple tables related to each other by keys so the same data isn't stored more than once, such as name and address information.

Common databases

The most common enterprise databases are Oracle, Sybase, and SQL server. The most common desktop database is Microsoft Access.

Definition

Enterprise Database. An *enterprise database* is one that resides on a server and can be accessed from multiple clients and multiple applications, concurrently.

Definition

Desktop Database. A *desktop database* is one that resides on the user's desktop. It is only intended for use by an individual or from the Web, depending on the interface, for a low volume of transactions.

ODBC

Databases can talk to each other and to Web servers using the Open Database Connectivity Standard (ODBC). *ODBC* is a set of rules databases agree to obey. Most databases, and software that interacts with databases, know this language and can communicate with each other using this language. You needn't worry about how it works; you only need to know whether your database is ODBC-compliant and whether your Web server or database-to-Web engine is ODBC-compliant. If they both are, then you have the tools you need to start building a Web-to-database interface.

If your database or your database-to-Web engine are not ODBC-compliant (such as File Maker Pro), then you must use the proprietary system this database has available for communicating with the Web. In the case of File Maker Pro, there is one. Not every non-ODBC-compliant database makes such a tool available. Even if your non-ODBC-compliant database has a way to get data to and from the Web, it will probably not be a scalable, enterprising solution. The most robust solutions — the ones that can handle the most traffic — tend to be ODBC-compliant.

The Web-to-Database Interface

To understand how the Web-to-database interface works, it helps to review how data is normally delivered to a client workstation, in a browser, via the Web (using the HTTP protocol, to be specific). Figure 45-1 shows the client workstation (on the left) requesting a page (using the HTTP protocol) from a Web server (on the right). The Web server receives the request and, if it can find the page requested and if the client workstation meets any security restrictions that may exist on the page, the server delivers the page to the client workstation (using the HTTP protocol).

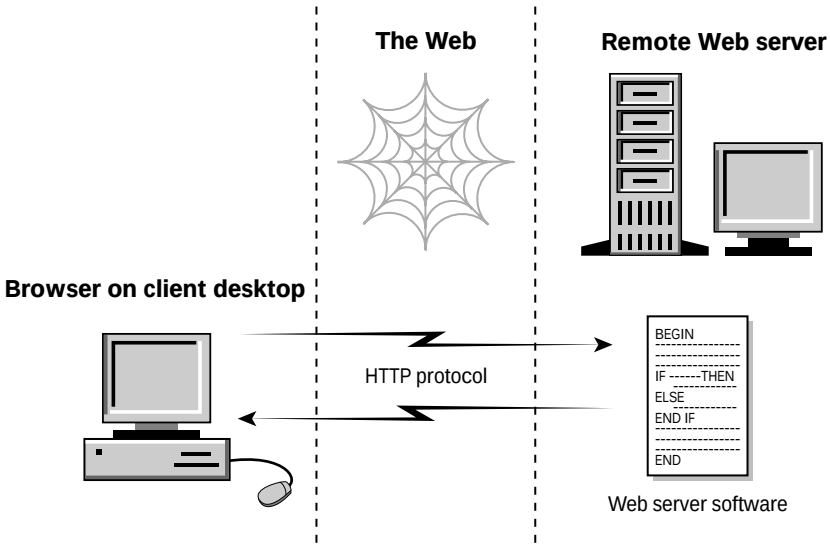

Figure 45-1: Delivering a page with no database interaction

Requesting data

When database interaction is required, a few more steps need to take place on the server. Figure 45-2 shows the client workstation (on the left) requesting a page (using the HTTP protocol) from a Web server (on the right). In this case, the page has a file extension of something other than .HTML (or .HTM), indicating to the Web server that some special type of action is required. The Web server determines whether it can handle the processing by itself or whether it needs to pass this page onto another software package. How does it know whether to pass on the page and where to pass the page? By checking the MIME type of the page requested.

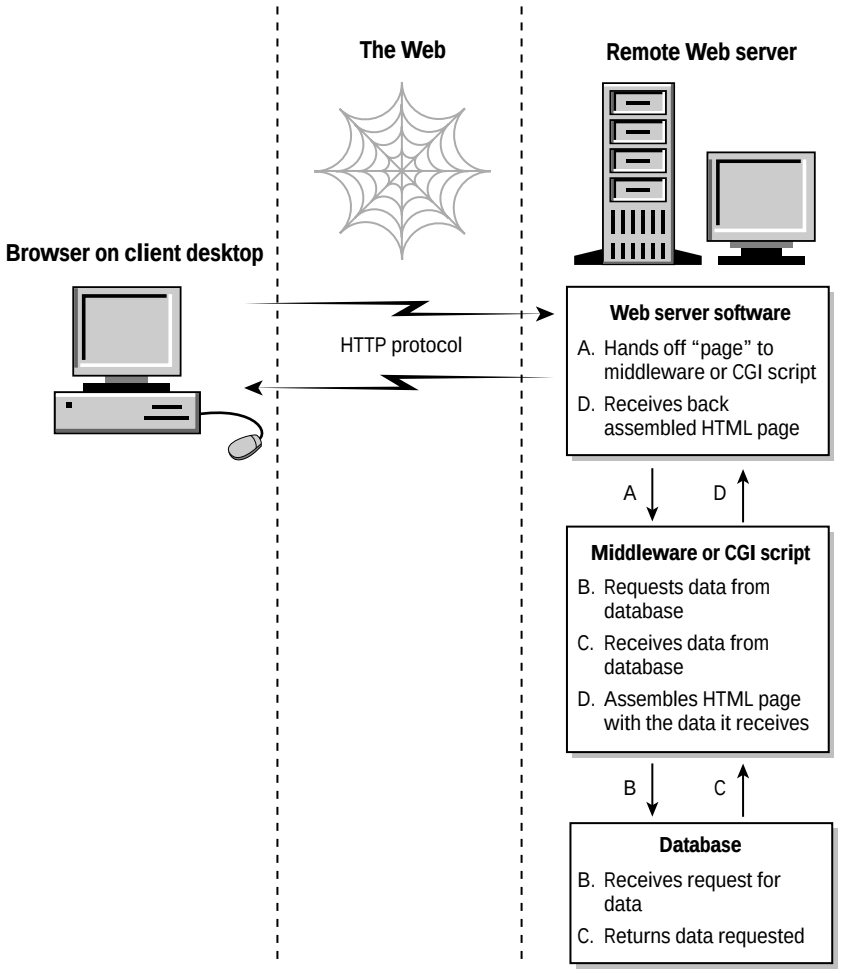

Figure 45-2: Delivering a page with database interaction

Communicating with the database

If the MIME type of the page requested is not supported by the Web server software, the page is handed off to the appropriate application, referred to as the Web-to-Database Engine, based on the MIME type. This application communicates with the database (usually in ODBC) to perform one of three actions:

1. **Search for data.** From the Web page, the visitor can indicate search criteria and only receive back data that matches those criteria.
2. **Add new data to the database.** From the Web page, the visitor can enter new data that will be added to the database.

3. **Modify existing data in the database.** From the Web page, the visitor can modify existing data — say, updating his address or phone number, or changing the inventory by making a purchase.

If the MIME type of the page requested is supported by the Web server, then it processes the request and communicates with the database directly. The choice of actions will still be one from the preceding list.

Returning results

Finally, the Web-to-Database Engine has the results of its communications with the database and can format those results into HTML, which is easily understood by the Web server (again, this can all take place within the Web-server software, if the Web-server software supports the MIME type of the page requested). The Web server delivers the results of the action on the database to the client workstation.

Conclusions from the Web-to-database interaction

Important to note about this entire transaction are the following points:

- The workstation doesn't perform any extra processing.
- The Web server, or the combination of the Web server and Web-to-Database Engine, communicates transparently with the database.
- The database can actually reside on a different computer than the Web server and Web-to-Database Engine.

That last observation is the most salient. You can leave your database wherever it normally resides and communicate with that database — via ODBC — across a network. This enormously expands your options for Web-to-database interaction.

A significant performance penalty can occur when the Web server software resides on a different physical machine from the database. You must do extensive testing before implementing this approach. One way around this slow-down is to copy (nightly or more often, if necessary) the data from the remotely located database to a mirror copy of the database located on your Web-server machine. How well this actually works depends on the location of the database, the size of the database, and the processing power of both machines.

Options for Accessing Database Data

Now that you understand how the interface between the Web server and the database works, you should understand what kind of choices you have for delivering this service on your own Web server. Because the Web-server side of the processing has three components — the Web server, the Web-to-Database Engine, and

the database — you can purchase three different combinations of products to meet these needs.

1. Three stand-alone components: stand-alone Web server, stand-alone Web-to-Database Engine, and stand-alone database
2. Dual-purpose Web server and stand-alone database
3. Stand-alone Web server and dual-purpose database

Three stand-alone components

Using three stand-alone components gives you the greatest flexibility in creating your applications. Chances are, your Web server has already been selected for you and is running on your server. For most people, this is half the equation in selecting complementary products. The other half of the equation is the database, which you probably already have in place, populated with your business data. This leaves you with the decision of which solution to use to connect your Web server to your database.

If your database is ODBC-compliant, then the hands-down best middleware on the market is Cold Fusion, by Allaire (`www.allaire.com`). Using Cold Fusion, you can script interaction with the database right in your Web pages. The scripting language is Cold Fusion Markup Language (CFML) and looks like HTML. The Cold Fusion Web-to-Database Engine processes the CFML tags and returns the data from the database to the Web server, as in Figure 45-2.

Cold Fusion is available in both a workgroup edition, which works with desktop databases, such as Fox Pro and Access, and an Enterprise version, which works with enterprise databases such as Oracle, Sybase, Paradox, and SQL Server. Cold Fusion for Workgroups runs on Windows NT/2000 Server. Cold Fusion Enterprise Edition runs on both Windows NT/2000 Server and UNIX servers.

Dual-purpose Web server and stand-alone database

When would you want to use a dual-purpose Web server? When it is free and already installed on your Web server. Microsoft Internet Information Server (IIS) supports Active Server Pages (.ASP files), which can be processed right in the IIS Web server.

When would you want to use a stand-alone Web server (that is, not use the built-in functionality of the Web server to process some form of database-interaction pages)? When you want to perform more actions than the freeware/middleware included with the Web server will provide.

One function you might want to perform from your database-interaction script is to send a confirmation message to the visitor's e-mail account, confirming certain action has been taken. With IIS, you can't do this without purchasing additional

COM objects, so your free solution is no longer free. With Cold Fusion, which you must purchase up-front, it is included.

Other dual-purpose Web servers include the Netscape Enterprise servers, which process server-side JavaScript, and O'Reilly servers, which process server-side Java, VBScript, JScript, Perl, or Python. If you decide to use either of those technologies, you won't have to purchase separate middleware. Whichever direction you take, you want to make sure the middleware you select is compatible with your database.

Stand-alone Web server and dual-purpose database

Why would you want to use a dual-purpose database that doubles as middleware? When it is built into your database. Netiva is one company that offers such a product. File Maker Pro is another such product. Netiva, unlike most other dual-purpose databases, can handle enterprise-level traffic. Most of the other dual-purpose databases are designed for low volume.

External Database Access without Programming

Today, many ways exist to perform basic interactions with a database without programming. Using FrontPage 2000, Microsoft's Web-development tool, you can create interactive Web pages with Access without programming. If your back-end database is File Maker Pro, Claris HomePage communicates directly with it, in a seamless, drag-and-drop environment. Regardless of the back-end database, as long as it is ODBC-compliant, you can use Tango, which is both the middleware and the editor, to create data-based Web pages without programming.

Only a few of the products on the market meet these needs. The market is quite crowded now. In the next few years, it could winnow down to the few best-of-breed products. For this reason, you want to make sure you select your products carefully.

External Database Access with SQL

To get the maximum flexibility when interacting with your database, you want to get your hands on the code and do the programming or the scripting yourself. Most middleware uses SQL, the structured query language for communicating with a database — not to be confused with the Microsoft product by the same name.

The problem with SQL is it is not standard. Whichever database you select will have its own slight variation on SQL, which you must learn to get data into or out of the database. To perform three tasks on your data, you can use statements similar to the following ones, where the data source name (the ODBC name for the

database) is *Inventory;* the field names are *Partno, Description, Quantity,* and *Cost;* and the unique key is *Partno:*

- **Search for data.**

```
Select * from Inventory
  Where Description = '%description%'
```

- **Add new data to the database.**

```
Insert into Inventory (Partno, Description, Quantity, Cost)
  Values ('Partno', 'Description', Quantity, Cost)
```

- **Modify existing data in the database.**

```
Update Inventory
  Set Description = 'Description'
      Quantity = 'Quantity'
      Cost = 'Cost'
  Where Partno = 'Partno'
```

From Here

Jump to Chapter 48 to learn about JavaScript.

Proceed to Chapter 46 and begin incorporating discussion groups and chats.

Summary

Communicating with a database can be a bit of a hurdle to set up, but nothing enriches your Web pages like access to current, real-time data your visitors want to see. Creating static Web pages that reflect database data accurately is impossible, so if you want your visitors to get at your real-time database data, you must take the time to get this up and running. Once you are operational, the maintenance costs of these products are negligible.

Before you purchase products to get your Web server to communicate with your database, you want to see if your Web server includes middleware and if this middleware is adequate to meet your needs. You also want to confirm that your database can handle the volume of traffic you expect your site to receive.

Once you select your tools, you can determine whether you will be satisfied creating your data-based Web pages with nonprogramming tools or whether you should get your hands dirty with SQL.

✦ ✦ ✦