CHAPTER 19

DATABASE SYSTEMS AND THE WEB

CHAPTER OBJECTIVES

- Comprehend the enormous significance of the merging of database and Web technologies
- Gain a broad understanding of the interaction between the two technologies
- Initially, have a quick review of the Internet and the World Wide Web
- Examine the motivation for union of database systems and the Web
- Recognize the advantages and shortcomings of Web-database integration
- Review various integration approaches
- Consider a few security options

What is the most dominant phenomenon in computing and communications that arose in the 1990s? Undoubtedly, it is the Internet with its unique information delivery mechanism, the World Wide Web. In just a little more than 10 years since its origin in 1989, the World Wide Web has become the most powerful data content repository and information delivery system. It is estimated that at the beginning of the year 2000, about 50 million households worldwide were using the Internet. In five short years, by the end of 2005, this number is expected to grow 10-fold.

The Web changes everything, as they say. It has changed the way people shop, run their business, collect information, read the news headlines, make travel arrangements, invest in stocks, buy a new home, chat with others, study, and entertain themselves. The earlier websites began as storage places for web pages that could be served up to anyone who would request them. These static pages are no

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4

longer adequate for today's environment. Web users need to react to the information these pages present and use the data content interactively. Mere storage of information as rigid web pages does not serve the purposes of today's users. Information content has to be stored in databases managed by powerful DBMSs to provide interactive access in a fast and reliable manner. Electronic commerce has made the database the centerpiece of its operations over the Internet. The productive integration of Web and database technologies has emerged.

In this chapter, we will concentrate particularly on the combined application of Web and database technologies. We will consider the types of computing architecture that promote the use of database systems in delivering information on the Web. We will specifically review some of the common approaches that make the integration of the two technologies happen. Although we will begin with a brief overview of the Internet and the Web, this is just background information, not the main focus. The primary emphasis is how database and Web technologies are made to work together.

WEB TECHNOLOGY: A REFRESHER

Let us first set the stage for our coverage of database systems in the Web environment. How do Web and database technologies merge and work together? How is a user interacting with a website able to retrieve web pages and input parameters and to access a database that drives the applications? What type of system architecture supports access of web pages and data from a database at the same time?

Our main discussion is concerned about such questions. But to put those questions in proper perspective, let us begin with an examination of a few basic concepts of the Internet and the Web. For information technology professionals in today's environment, this has to be just a brief review. After beginning with the basics, we will quickly walk through the languages for the Web and the data communications protocol for moving web pages around.

The Internet and the Web

Very simply stated, the Internet is a collection of interconnected networks separately owned by government, commercial, industrial, educational, and research organizations. In addition to these organizations, internet service providers (ISPs) are also connected to the network. With this interconnection, the Internet is able to provide a number of services including e-mail, transfer of files, conferencing, and chat with others with the same interests. The Internet had its experimental beginning in the late 1960s. Here is a very brief history.

- **Late 1960s:** U.S. Department of Defense project ARPANET (Advanced Research Projects Agency NETwork) set up to build networks that would stay up even if some sections of the network go down because of any catastrophe.
- **1982:** TCP/IP (Transmission Control Protocol and Internet Protocol) adopted as the standard communications protocol for ARPANET. TCP ensures correct delivery of messages. IP manages sending and receiving information packets based on a four-byte destination address (IP number).

- **1986:** NSFNET (National Science Foundation NETwork) created after responsibility for the technology was transferred to NSF.
- **1995:** NSFNET no longer the backbone for the Internet. A complete commercial system of backbones replaced it.

Strength and Simplicity of the Web Consider the hundreds of thousands of computer systems connected over the Internet. Think of all the information residing in those systems in the form of pages spread across as connected pieces of information. The World Wide Web is a simple mechanism for exploring this huge volume of information scattered across the Internet.

As you know, information on the Web is presented in the form of pages containing multimedia components such as graphics, video and audio clips, and pictures in addition to text. A web page may also contain connections or hyperlinks to other pages. This is a clever way of navigating through the Web and browsing information on pages irrespective of the computer systems in which they reside.

When you examine the computer systems connected on the Web, you note the systems performing one of two roles. Servers are the computer systems storing information in the form of web pages; clients, usually known as browsers, are those requesting information in the form of web pages. The common language used for creating Web documents is known as HyperText Markup Language (HTML). Browsers on the Web interpret HTML and display the requested web page. The set of rules or protocols governing the transfer of Web documents between browsers and servers is known as HyperText Transfer Protocol (HTTP). A Web document or its component is identified by an address stipulated as a Uniform Resource Locator (URL). We will briefly discuss HTTP, URL, and HTML. We will also cover the usage of Web technology in intranets and extranets.

But first refer to Figure 19-1 illustrating the fundamental layout of the Web environment. Note the components and how the Internet provides the network connection.

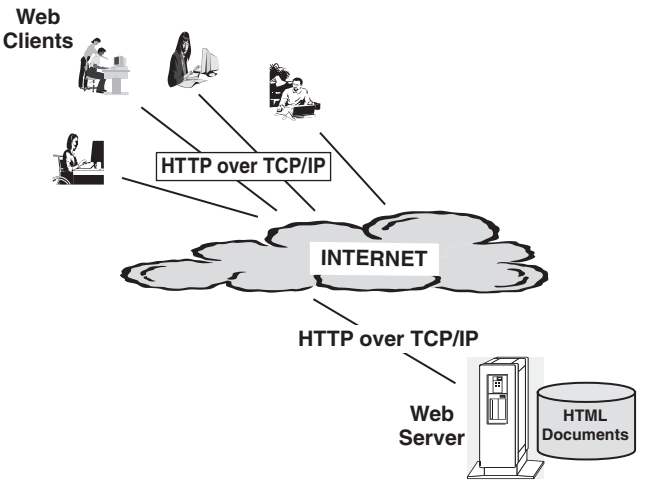

Figure 19-1 The Web environment.

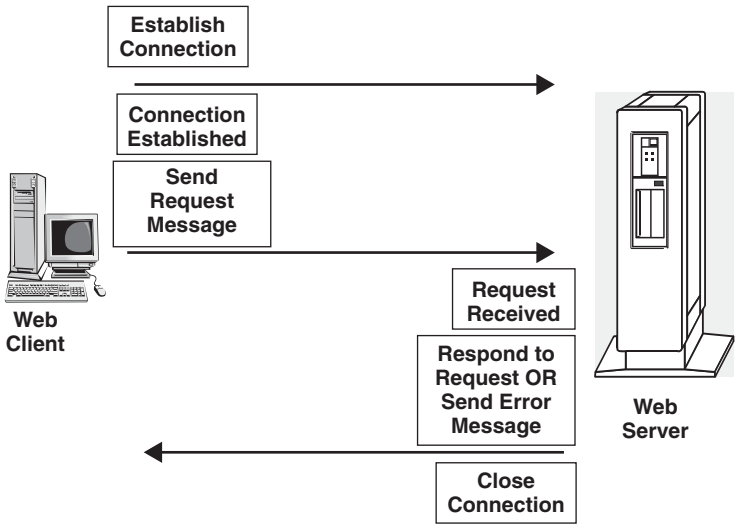

Figure 19-2 Stages of an HTTP transaction.

HyperText Transfer Protocol (HTTP)

HTTP is a simple, object-oriented protocol governing the exchange of hypertexts written in HTML on the Web. The protocol stipulates how clients and servers can communicate and exchange information. This protocol was developed especially for the Web. Its simplicity is its strength. HTTP may be thought of as a great equalizer; it allows you to create HTML documents on any computing system and deliver them in a consistent way to any other system. If a server can deliver the document, any browser can view it the way you created it.

With HTTP, exchange of information takes places in four stages or phases. An HTTP transaction operates on a request-response method. Figure 19-2 presents the stages of an HTTP transaction.

Here is a brief description of the actions in the four stages:

Connection. The client establishes connection with the identified Web server. You may note the status line of most browsers indicating this action in the status line as "connecting to . . . ." If client cannot establish the connection, nothing more happens. The connection attempt times out.

Request. If the connection is established, the client sends a request message to the Web server. The request includes information about the transmission protocol being used with the version number, the object requested, and how the server must respond to the request.

Response. The Web server sends the requested HTML document to the client indicating the protocol being used. If the server cannot fulfill the request, it sends back an error message. As soon as the document is received, the browser launches the appropriate software to display its text and the multimedia components.

Close. The server closes the connection.

The simplicity of the protocol mainly comes from its property of being stateless. What does this mean? As soon as the server closes the connection, nothing is retained in the server about the transaction. The server does not remember any prior requests. Once the connection is closed, the memory about the transaction is completely erased. When you request a document, the server does not know who you are, whether this is your first request, or whether you are at the website for the fiftieth time. This property, however, requires very simple logic. There is no need for elaborate software, and the clients and servers can run lean without holding extra memory and storage for old requests.

A big challenge in using HTTP is its inability to keep track of the state of your interaction with the server. The state indicates information about who you are and about your request or visit. Maintaining state means remembering prior information as the user at the client system moves from page to page at the website at the server system.

Being stateless, HTTP, by itself, cannot apply to database transactions. Database transactions need the concept of a session during which interactions between the client and the database can take place. Some schemes are available for maintaining state information: store it in cookies, encode it in the links, send it in the form of hidden variables, and store it on the Web server. However, these schemes have limitations. As you will see later, server functionality must be extended for database transactions to execute on the Web.

Uniform Resource Locator (URL)

URLs act like addresses to any definable resource on the Internet. They are all-purpose reference mechanisms for the Internet. URLs are the basis for HTTP and the Web. They are used to define addresses on the Web where documents or resources can be found based on their location. Any line from one document to another is coded in the form of a URL.

A URL is written with a simple syntax consisting of the following parts:

Protocol or scheme. The way or method to exchange data on the Internet (HTTP, FTP, Telnet, etc.)

Host. The host computer or the server on which the data resource is located

Port. Port number if the requested service is not located at the default port of the host system

Path. Path name on the host where the resource is to found

A URL, therefore, is a string of characters composed of these components. Here are a few examples of URLs:

http://wiley.com/author/format.html
ftp://raritanval.edu/documents/dbdesign/syllabus.txt
telnet://iitf.doc.gov

HyperText Markup Language (HTML)

HTML is a simple language for marking up and tagging a document so that it can be published, that is, made available on the Web. It is known as a markup language because you can add marks to regular text; these marks carry special meanings for a Web browser indicating how the document must be formatted and presented. The flexibility and strength of HTML come from its being platform-independent.

HTML uses tags to mark the structure of a document. The label of a tag is enclosed with the "<"and">" characters. For example, <HEAD> is a tag indicating the start of the page heading. Each structure begins and ends with a tag. The heading structure begins with the <HEAD> tag and ends with the </HEAD> tag.

Parts of an HTML Document Without going into too much detail, let us quickly review the parts of an HTML document. The document consists of elements that form its structure.

Head Elements These are used to indicate the properties of the whole document, such as title, links denoting connections to other documents, and the base URL of the document. This element is not displayed to the user as part of the document but contains information about the document that is used by browsers.
Example: Selected Notes of Databases and the Web

Body Elements These are used to indicate the text in the body of the document. Unlike the head elements, the body elements are displayed by browsers. There are many different body elements. Just a few samples of body elements follow.

Headings

Standard HTML supports up to six levels of headings.
Example: <H2>DBMS-Web Integration</H2>

Anchors

These are used to mark the start and end of hypertext links.
Example: <A HREF=http://www.wiley.com/author/instructions.html> </A>

Paragraph marks

These are used to define paragraph breaks.
Example: The paragraph mark defines a page break. It is placed at the end of a paragraph. <P>

Line breaks

These are used to indicate the start of a new line.
Example: The HTML School
100 Main Street
Anytown, XX 12345

```
<HTML>
<HEAD>
<TITLE> Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals </TITLE>
</HEAD>
<BODY> background=serenity.jpg
<H2> Welcome to the Home Page of our data warehousing textbook. Review the table of contents. Look at the sample chapters. Get
a complete inside into the unique presentation. Above all, get practical and useful knowledge that can propel your career into new
heights. For further information, please contact John Wiley & Sons, world-renowned publisher of superb technical books. <BR>
<A HREF="http://wiley.com/dw001/book/toc.html">TABLE OF CONTENTS <BR>
</A>< HREF="http://wiley.com/dw001/book/chapter1.html">1. The Compelling Need for Data Warehousing <BR>
</A>< HREF="http://wiley.com/dw001/book/chapter2.html">2. Data Warehouse: The Building Blocks<BR>
</A>< HREF="http://wiley.com/dw001/book/chapter3.html">3. Trends in Data Warehousing<BR>
</A>< HREF="http://wiley.com/dw001/book/chapter4.html">4. Planning and Project Management<BR>
</A></P>
<P>We would love to hear your comments. Happy to serve you.
</BODY>
</HTML>
```

Figure 19-3 Sample HTML coding.

Graphic Elements All modern browsers display in-line graphics embedded in an HTML page. The source of the image is indicated by the term "SRC" in the coding for the graphic element. Browsers not capable of handling graphics will display a text string based on the "ALT" attribute.

Example: <IMG SRC="ABCD.logo.gif" ALT="ABCD Logo">ABCD Company

Figure 19-3 shows a sample coding for an HTML document. Note how the various elements are combined to create the document.

Static and Dynamic Web Pages An HTML document is stored in a file on the server and displayed by the browser. Every time you request for the web page, you will see the same information. The content of the page stays intact; it does not change. If the information on a particular web page must be changed, then the document must be recoded and the revised version must be stored in the file on the server. Such web pages are static web pages.

When HTML was first introduced, its features included text and graphics as well as the embedding of hypertext links. This itself was a dramatic achievement for distributing and presenting information over the Internet. Even though the web pages were static, HTML revolutionized the use of the Internet. However, as the use of the Web expanded, the limitations of static web pages became apparent. Users want exchange of information to be interactive. Users need the ability of servers to respond to user inputs and produce web pages according to the inputs. Users need dynamic web pages.

A dynamic web page changes every time it is accessed. Dynamic HTML is a term applied to a collection of technologies that make HTML documents more dynamic and interactive. In addition to having the standard features of a static web page, a dynamic web page can return the results of a database query. For example, you can complete a form and send it as an input to the server. The server must then return results based on your input. Furthermore, a dynamic web page may be customized on the basis of user preferences.

To implement dynamic web pages, servers will have to generate hypertext based on the requests received. Consider database transactions. These transactions such as insert, update, and delete effect changes in the database. A database is dynamic—changing. Similarly, as database technology is integrated with Web technology, dynamic web pages become a requirement. In the later sections of this chapter, we will explore some of the major methods for making web pages dynamic.

Beyond HTML

Even with this brief introduction of HTML, you must have noted the striking simplicity of HTML. It is a simple language. Its simplicity has made Web publishing so widespread within a short period. HTML follows straightforward rules so that it can be easily incorporated in editing tools. A programmer can generate hypertext through these tools even without a full knowledge of HTML. The ease of use of HTML is probably one of the chief reasons for the phenomenal growth of the Web.

A language meant to be simple must necessarily adopt a small set of rules. HTML does this. The simplicity of HTML is both its strength and weakness. Consider the following list of inadequacies:

- HTML follows a set of restrictive rules.
- HTML is an application of the Standard Generalized Markup Language (SGML) that is based on a set of restrictive rules.
- HTML lacks flexibility. It cannot be used for simple and complex documents with equal ease.
- HTML is not able to describe complex documents.
- HTML is more focused on content presentation than on content description. HTML tags stipulate how the content must be presented; they say little about the meaning of the content.
- HTML is not a descriptive language.
- With just head and body elements, different types of documents such as abstracts, summaries, chapters, reference sections, and bibliographies cannot all be described adequately.

What is the solution? Perhaps more types of tags must be added to HTML. But how many new types of tags would satisfy all the numerous requirements? A method of defining user tags may be a solution. What was needed is a language with extended capabilities, a language beyond HTML—a more flexible and extensible language. Such a language is XML (eXtensible Markup Language). XML focuses on content description. Let us look at the major benefits. We are not getting into the details of XML components here. That is not the purpose of this chapter. For more information on XML, you may refer to a wide assortment of good books on the subject.

Major beneficial features of XML include:

Flexible. Capable of handling a simple home page or a large document such as *War and Peace*.

Extensible. Extended capabilities to enable you to define your own XML elements. You can define your own tags to suit your particular needs in the type of document you are publishing.

Structured. Although flexible and extensible, XML's structured approach helps conformance to rules—not a free-for-all method.

Descriptive. XML elements concentrate on meaning of the contents, not on their presentation. The elements can describe what they contain, allowing intelligent handling of the content by other programs.

Portable. When you define your own tags, how are you going to pass on the definitions and syntax of those tags to others? XML provides simple facilities to produce files capturing the rules of your elements and tags so that your documents can be read properly.

Intranets and Extranets

Consider the ease with which you can publish web pages and exchange e-mails over the Internet. The Internet provides low-cost transmission of information. The concept of exchanging information using Internet and Web technologies need not necessarily be confined to a specific set of users. All you need is the set of components: a server, a browser, and a set of applet-based applications for exchanging information. An organization needs the ability to exchange information among internal users; it also needs information to be made available to a selected set of outsiders such as distributors, suppliers, and customers.

So why cannot the same paradigm for exchanging information with the public over the Internet be extended to exchanging information with internal employees and external business partners? Intranets and extranets are the solution options.

Intranet From the time the term "intranet" was coined in 1995, this concept of a private network has gripped the corporate world. An intranet is a private computer network based on the data communications standards of the public Internet. An intranet may be thought of as a set of websites belonging to an organization and accessible only to internal users. Information exchange happens within the firewall and, therefore, is more secure. You can have all the benefits of the popular Web technology and still manage security better on the intranet. Intranets are less expensive to build and manage.

Extranet The Internet and the intranet have been quickly followed by the extranet. An extranet is not completely open like the Internet, nor it is restricted to internal use like an intranet. An extranet is an intranet that is open to selective access by outsiders. From your extranet, you can look outward to your customers, suppliers, and other business partners. For the purpose of security, access over the extranet must be controlled through security mechanisms such as valid user names and passwords. The Internet itself serves as the communications infrastructure. Web technology provides the means for publishing information. Tracking packages at the FedEx site and checking balances at your bank's site are examples of extranet applications.

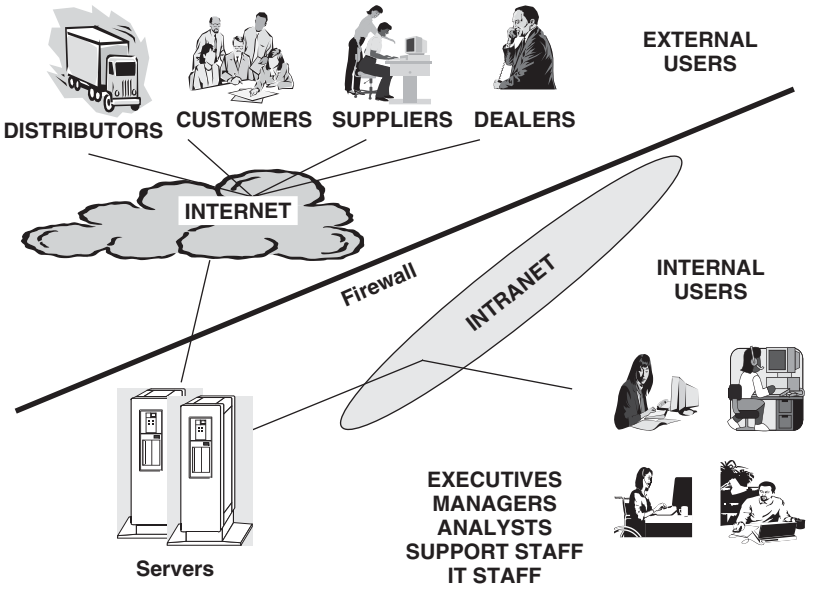

Figure 19-4 Intranet and extranet.

Figure 19-4 illustrates the concepts of intranets and extranets. Note the firewall and how information exchange takes places within and outside the firewall. Note also how the Internet provides the means for data communications.

Let us summarize a few major benefits of intranets and extranets:

- With a universal browser, users will have a single point of entry of information.
- Minimal training is required to access information. Users are already familiar with the use of a browser.
- Universal browsers run on any computer system.
- Web technology opens up multiple information formats such as text, images, graphics, charts, audio, and video.
- It is easy to keep the data content updated for access on the intranet and the extranet. Usually, there will be one source for a particular piece of information.
- Opening up limited corporate information on the extranets fosters better business partnerships and improves customer service.
- Deployment and maintenance costs are low compared to implementing communications networks run on proprietary protocols. The Internet and the Web are standardized.

WEB-DATABASE INTEGRATION

The Web is the cornerstone of electronic commerce. All you have to do is sign on to the Internet and search for any product or service. You will get a huge list of businesses offering that product or service. It is hard to believe that, not long ago, any

type of commercial activity was strictly forbidden on the Internet. Now the Web is where you go to shop for thousands of products and services. There are also many special interest business sites.

Every airline, every bank, every auction house, every computer vendor, every car dealer, every database company—the list goes on and on—has its presence on the Web. Think of all the companies in the entertainment business, all those in the travel business—these alone can fill up page after page when you search for them.

How do these companies operate their businesses? How do you look for prices? When you place an order and make a payment, where are the details of your order stored? What is behind all of the commercial activity on the Web? Databases form the underlying foundation for electronic commerce. Database and Web technologies must mesh together to make electronic commerce possible. Let us now examine this integration of the two technologies.

Motivation for Integration

Imagine that you wish to shop for books on French cooking on the Internet. You browse the various sites and find an electronic bookstore that has a wide selection of books on that topic. You ask for a list of books. You narrow down your choice to three books. You want to look at what the reviewers have said about these books. You check the prices and any available discounts. You add them to your electronic shopping cart. You indicate your preferred method of shipping. The website adds shipping charge and sales tax, completes the calculations, and displays the total amount due. You pay with your credit card. The transaction is complete; you have placed your order.

How does the transaction get processed? When the website displays the list of books on French cooking, from where does it get the data? When the site shows you the prices and shipping charges, from where does it display this data? When you finish your order, where do the details of your order get stored? That merchant's database.

It makes the greatest sense to take the information stored in an organization's database and selectively allow access to the information to those visiting its website on the Internet, intranet, or extranet. Here are a few obvious but compelling reasons for integration of an organization's database with its website:

- Fixed or static Web documents do not allow customized information display. Every time data are updated, the static document becomes outdated.
- Trying to reenter data stored in an organization's database on HTML documents constitutes an enormous waste of time.
- The database can be used to create web pages to reflect the current data values.
- Customers and internal users can get up-to-date information from the database.
- Business partners can receive up-to-date information on product data such as prices and availability directly from the database.
- Order entry and fulfillment transactions can interact directly with the database. The older method of buying products on the Web required taking an order on

a form, sending the form by e-mail to a person in the organization, and that person keying in the data into the database through a separate application.

Requisites for Integration

How can Web-database integration be brought about? How can the Web be made a platform to provide users with interfaces into the database? Naturally, database vendors are working on proprietary solutions, but organizations are looking for generic solutions without restricting themselves to single-vendor options.

Review the following list of requisites of organizations for integrating their databases with their Web presence.

- Overcome the stateless nature of a Web transaction and provide continuous sessions for users to interact with the database.
- Provide secure access to the database.
- Allow session-based and application-based authentication of users.
- Protect the database from a transaction destroying the database updates made by another concurrent transaction over the Internet.
- Provide connectivity to the database, independent of the DBMS.
- Provide connectivity to the database, independent of proprietary Web browsers or servers.
- Offer a solution that utilizes all the standard features of DBMSs.
- Present an open architecture for the database system to be operable by a variety of technologies.
- Provide an interface to the database that is scalable as the volume of Web transactions rises.
- Enable access to the database guaranteeing acceptable performance.
- Offer the ability to administer the interface efficiently in a cost-effective manner.
- Allow the interface to work with productivity tools.

Architecture for Integration

Think of a simple database query. A customer picks a product sold over the Internet and wants to know the price. At one end is the browser, where the customer is establishing contact with the website. At the other end is the company's database, where the price for the selected product is stored. Make the connection between the browser at one end and the database server at the other end.

The browser and the Web server exchange information over the Internet with HTTP. The server transmits a form to the browser, and the user selects a product on the form and indicates a request for the price. The transmission of this request from the browser to the Web server takes place. What must the next link be in the communication chain?

For the request to be fulfilled by the DBMS, it must be translated from HTML into SQL for relational databases. The request in SQL must be transmitted to the DBMS at the database server to retrieve the price for the product, possibly from the

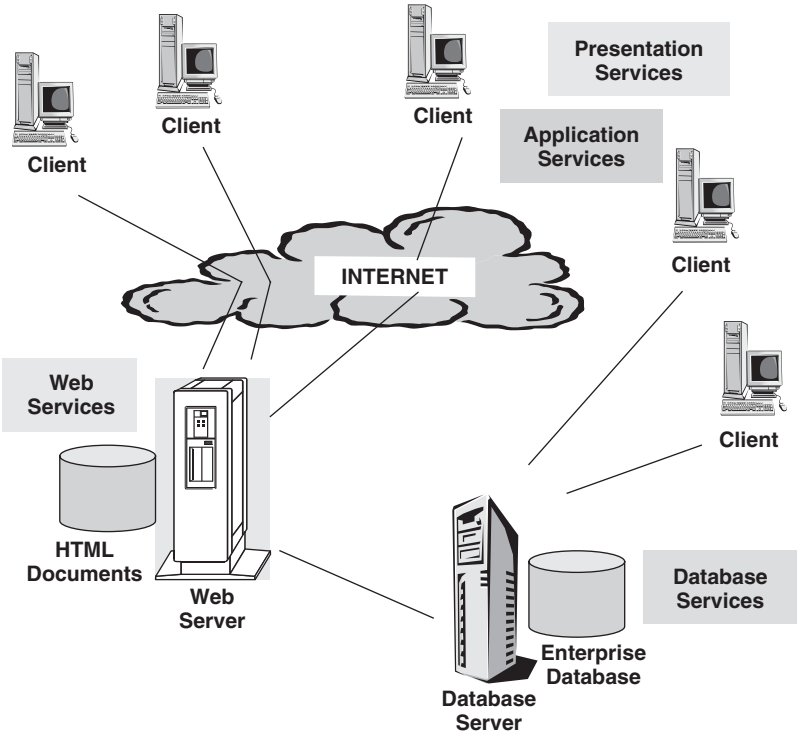

Figure 19-5 Three-tier architecture for Web database access.

product table in the relational database. The product information must be incorporated into an HTML document for the server to transmit to the browser. This has to be done in a secure environment. Importantly, it must be done as though it is part of a single session. This is because the user will probably complete the order and transmit it back to the server for further processing and recording in the database.

Let us design a suitable architecture for the type of processing needed for the simplest of queries described here. Consider the traditional two-tier client/server architecture with the client at the first tier and the database server at the second tier. In this architecture, the client handles the presentation services and the database server performs the database services. Although for a simple query, the two-tier architecture may be satisfactory for access over the local network, we need at least one more tier to accommodate the Web server for access over the Internet. Figure 19-5 shows such a three-tier architecture.

When the complexity of application services and logic increases, the need for a separate application server becomes apparent. The browser interacts with the Web server based on HTTP; if static HTML pages are requested, the Web server delivers these to the browser. If dynamic web pages with database content are needed, the Web server passes on the request to an application server. The application server interacts with one or more data sources to assemble a response. If there is a request for database services, it is sent to the database server. When the responses to the data sources are completed, the application server assembles a consolidated response in the form of web pages and responds to the browser.

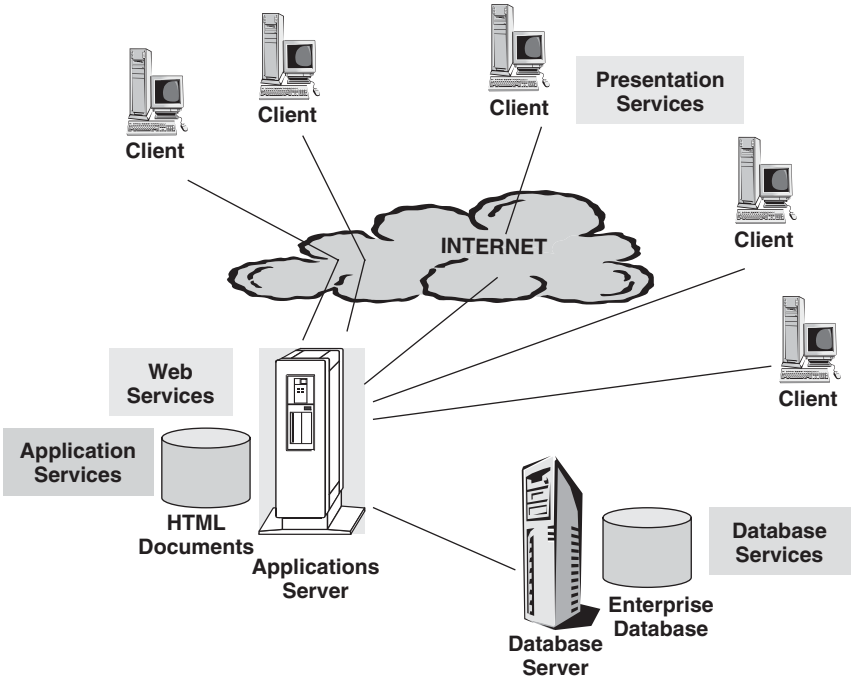

Figure 19-6 Architecture with application server.

Figure 19-6 shows this architecture with the inclusion of the application server.

Observe the services offered at each tier. However, note that the separation into a distinct application server may just be logical so that it may be deemed to reflect additional functionality of the Web server itself. Therefore, in our further discussions, we will construe the application server functionality to be part of the Web server itself.

Advantages and Disadvantages

Before proceeding to explore options for integrating Web and database technologies, let us summarize the advantages and disadvantages of such integration. This discussion will further strengthen your appreciation of the need for the integration.

Advantages Web documents are files. In a sense, storing information in Web documents tends to be a file-oriented approach. Contrast this with storing information in databases and using databases for providing information. All the advantages of a database approach become available when you integrate database technology in the Web environment.

More specifically, here are few significant advantages.

Popular interface. Using and accessing information through a web page has become widespread. Everyone knows how to access and use a web page. It is a familiar and intuitive method for information access and navigation.

Simple interface. HTML pages do not present complexity or difficulty for either users and developers. Most pages provide a simple interface to the database.

Accessibility. With browsers being platform-independent, data from the database may be provided in the form of a dynamic web page for access to anyone using any type of client system.

Standardization. A database application providing information through web pages adheres to HTML standards. Therefore, a user needs very little training for switching among different database applications.

Single interface. The browser becomes the single interface mechanism for the user to access data stored in a database or information stored in static web pages.

Network transparency. In the Web environment, the network is basically transparent to the user, simplifying database access.

Easily upgradeable. By separating out logic and database services from the client machines, upgrading and administration of multiple platforms in an organization become easier.

Innovative services. Organizations are able to provide services and information in innovative ways to reach potential customers.

Disadvantages Web-database integration is not without a few disadvantages. Although strong reasons drive the effort to make the database available over the Web, a few disadvantages give rise to some concerns.

Here is a list of some difficulties that need to be overcome.

Security. Security concerns top the list. Most of the users of a company's website are anonymous, and therefore proper user authentication and secure data transmission require special attention.

Reliability. The Internet, compared to an enterprise's own network, is slow and unreliable. You cannot be sure whether or not the requested information is delivered. Organizations continue to depend on their intranets for critical data.

HTML easy, but limited. Although easy and simple, HTML is limited when it comes to supporting any large and intricate database application.

HTTP limitation. The stateless nature of the protocol requires elaborate methods to provide the continuous sessions needed in a database application.

Performance. Poor performance and slow responses could become real problems because many of the scripts used in the Web environment are written in interpreted, not compiled, languages.

Inadequate tools. The tools for developing database applications to run in the Web environment are immature and lack many significant functionalities.

INTEGRATION APPROACHES

We have reviewed the motivation for integrating Web and database technologies. We have explored architectural options. We have considered the advantages and disadvantages. Now that you are convinced of the importance of the integration, let us look at the feasibility of such an integration. We will now examine the leading approaches for Web-database integration.

Common Gateway Interface (CGI)

You are now familiar with the basic functionalities of the browser and the Web server. The browser can request for Web documents stored in files, and the server can dispense these precoded HTML documents to the browser. The Web server can read files but cannot write to them. The Web server is intended to read HTML documents but not data from multiple types of sources. Apart from providing static web pages, the Web server cannot customize web pages or satisfy a request for data from a database.

Consider a simple Web application. A simple form is presented to the browser for the user to fill in the name and address. All that is required is for the name and address to be stored at the website and a confirmation message sent back to the user addressing him or her by the first name. The Web server by itself is not equipped to complete even such a simple transaction.

Some other interface is needed to receive the name and address details from the browser, store those details in a database, embed the user's first name in a web page, and send back the page to the user. A program is needed to interface with the database and store the name and address. The Common Gateway Interface (CGI) is such an interface and a specification for programs to accomplish such tasks.

What is CGI? CGI is a standard method for interfacing Web applications to sources of data. Programs written under CGI specification can retrieve web pages, access database records, build web pages on-the-fly, send e-mails, and so on. A CGI program can read and write data files; the Web server can only read them. A browser sends information to the server, and the server passes on the information to a CGI program that sends its output to the server, which in turn sends the requested information to the browser. A CGI program does not communicate with the browser directly but through the Web server. CGI is a standard method for a Web server and a CGI program to communicate. CGI defines the standard for a browser to communicate through the server with a CGI program executing on the server machine.

Figure 19-7 illustrates the CGI methodology. Note how CGI provides a generic, standard method to interface with multiple types of data sources.

A CGI program may be written in any of the different programming languages such as C/C++ or Perl. With regard to processing logic, a CGI program is just like any other program. The essential difference lies in the way a CGI program receives information from the Web server and sends information back to the server. CGI programs are generally referred to as scripts.

When a URL from the browser points to a HTML page in a file, the Web server fetches that page and sends it to the browser. On the other hand, if a URL points to a program, the server starts that program. Let us say that the URL from the

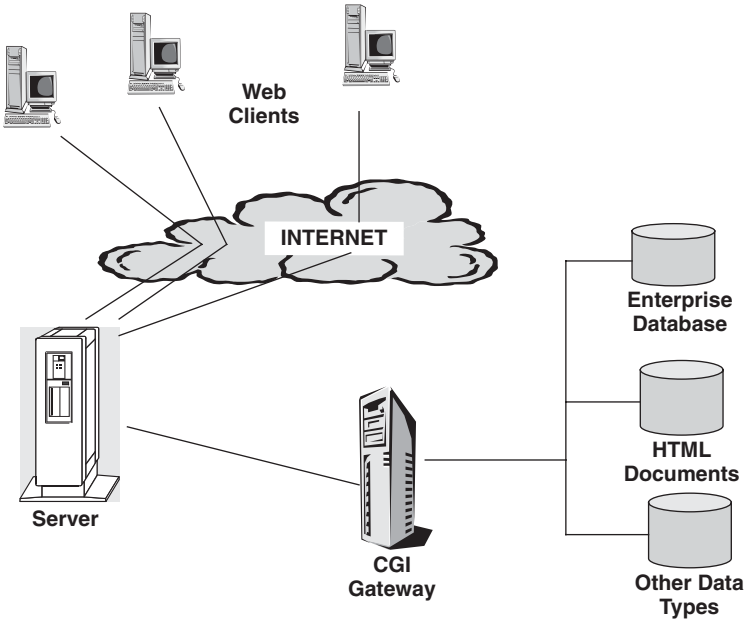

Figure 19-7 The CGI methodology.

browser points to a CGI script, then the Web server prepares certain environment variables to indicate the state of the server, directs the browser input to the script's STDIN, and launches the script. At the completion of its execution, the script places its output in its STDOUT. The server passes on the output in the script's STDOUT to the browser. STDIN and STDOUT are mnemonics for standard input and standard output, two predefined data stream or file handlers for each process. The CGI specification calls for the Web server to pass information to a script through its STDIN and to receive information from a script through its STDOUT.

Just one more detail about the information sent back from the execution of the CGI script is worthwhile to note. The server does not know what type of output is being sent to the browser from the execution of the CGI script. The CGI script could have created a plain text file, a text file with graphics, an image file, and so on. This information about the nature of the output must be communicated to the browser. So a CGI script includes a header with information about the type of page contents.

How CGI works Let us consider how CGI applies to a Web environment and trace the execution of a Web transaction.
Here are the broad steps:

1. The browser displays an HTML page containing an input form.
2. The user enters data into the form and clicks the submit button.
3. Sometimes, a script in the browser itself may perform client-side editing and validation of the data filled into the form.
4. The browser decodes the URL and contacts the server.

5. The browser requests the file containing the HTML page from the server.
6. The server translates the URL into a path and file name.
7. The server notes that the URL points to a program and not a static file.
8. The server prepares the environment, creates the environment variables, places user input in STDIN, and launches the script.
9. The script begins execution, reads the environment variables, and gets input from STDIN.
10. At the conclusion of the execution, the script sends proper headers to STDOUT.
11. The script places its output in STDOUT.
12. The script terminates.
13. The Web server recognizes the termination of the script and closes the connection to the browser.
14. The browser displays the output from the script.

Advantages of CGI CGI has become the de facto standard for communication of external applications with Web servers. The following is a listing of advantages offered by CGI.

Generic interface. Provides a generic interface between Web servers and user-designed applications accessing various types of data sources.

Simplicity. The only requirement for any program to be CGI-compliant is to conform to the method for passing information to and from the script.

Coding in any language. CGI programs may be coded with any of the several programming languages.

Server independence. A CGI program can be written in such a way that it will execute without modification on any server running on the same operating system.

Platform independence. A CGI program can be kept portable by using a commonly available language and avoiding platform-specific code.

Problems with CGI CGI scripts tend to impose a heavy burden on Web servers. The scripts are separate programs. The server process must initiate a new task for every CGI script launched. Consider an enormously popular website. The Web server will be launching numerous scripts for execution. Each task initiated by every script uses system resources such as memory, disk space, and slices of processor time. When many scripts have to execute almost simultaneously, the server can be quickly bogged down. When scripts become large and complex, they take longer to load and execute.

The Web server may also become a serious communications bottleneck. The communication between a client and the database server must always go through the Web server. The problem gets worse as the number of simultaneously executing

scripts increases. Every time a request is made by a Web client to a database server and every time a result is conveyed by the database server to the client, the Web server must convert an HTML document into data or vice versa.

A CGI-based approach lacks efficiency and does not provide transaction support. Essentially, this approach is not able to shake off the limitation of statelessness of HTTP. Even repeated requests from the same user require logon and logout every time.

CGI and Web-Database Integration To what extent does CGI enable the integration of the database into the Web environment? How exactly does CGI support database applications on the Web?

Take the example of a user filling in a form on a web page at the client side. The browser transfers the information to the Web server, which in turn passes on the information to the CGI script while launching it. Suppose the scenario is like this: The Web server sends the blank web page containing the form to the browser to display the blank form to the user. The transaction is completed, and the server ends the connection. The form is expected to be filled in and sent back to the Web server so that the server can pass on the information to the CGI script for further action. However, if the user does not complete the mandatory fields on the form, the CGI script is unable to display a warning box and refuse to accept the incomplete input on the form. Applications of this nature require extending interaction with the users. The difficulties caused by the statelessness of HTTP persist even when adopting CGI.

For simple direct queries against the database, CGI seems to be useful. CGI provides a useful interface to the database server. For example, if you want to know the current status of the parcel you sent through a shipping company, you can log on to the company's website, type in your tracking number and send it to the Web server. The tracking number is supplied to the CGI script, and the script is launched. The CGI program retrieves the current status of the parcel from a database and sends the information to the browser through the Web server. If the transaction is meant to end here without any further interaction, then CGI is helpful. CGI provides a generic way to interface with the database server and send back the information to the browser. The Web browser, just by itself, could not accomplish this. To get around fundamental problems arising out of the stateless nature of HTTP, some methods other than CGI must be explored.

As you have already noted, performance could be a major concern while working with CGI. Each request requires an additional process to be created. This places an enormous overhead on the server when the number of on-line requests peaks during the day.

Application Programming Interface (API)

To overcome the performance problems encountered while adopting CGI, one approach is to extend the functionality of Web servers. Servers can be changed and enhanced to provide application programming interfaces. Such enhancements or extensions are known as non-CGI gateways. An application programming interface (API) offers an interfacing technique between the Web server and a database application at the back end based on shared objects and dynamic linking. This technique

avoids the need to create a separate process for each execution of a script, as in the case of CGI.

Here are the major advantages of server extensions through APIs:

- Scripts performing database operations are loaded as part of the server; therefore, back-end applications can fully utilize the input and output functions of the server.
- Only one copy of the application script needs to be loaded and shared among multiple requests to the server from browsers.
- Through an authentication layer asking for user ID and password, in addition to the browser's own security schemes, the web page or website may be protected.
- Passes information out to browsers in additional ways, not possible by unextended servers.
- Tracks more information by logging incoming and outgoing activities at the server.

However, the API approach poses certain difficulties. It is a lot more complex than the CGI method. Using APIs effectively calls for special programming knowledge and skills. Adding proprietary APIs may introduce potential risks. The normal functioning of the server is being changed, and that could introduce unknown problems. The most practical difficulty relates to portability. APIs are completely proprietary and therefore reduce the portability of the server using such APIs.

Web-Database Integration: CGI vs. API Both CGI and API extend the functionality of the Web server so that back-end database applications may communicate with the browser through the server. Although these approaches tend to produce similar results, they essentially differ in the way each approach makes the communication possible and in the efficiency with which they accomplish it. Let us summarize these two points.

Communication Method CGI establishes communication in a restrictive manner. A CGI program can communicate with the server according to specifications through one or more variables. It executes when the Web server interprets a request from the browser and launches it. After execution, the CGI script returns the results to the server. That is the way a CGI program can communicate—take information from the Web server and return information to the server. Only the server can send the information to the browser.

Contrast this to how communication is carried out in the API approach. An application program can intercept information sent by the browser to the server before the server even reads the information. The program can revise the intercepted information and send the information, as revised, back to the browser without the direct involvement of the browser. Additionally, an API-based program can also perform operations based on requests from the server just as a CGI script can.

Performance API-based extensions run as part of the server; they are loaded into the same address space as the Web server. As you know, for each execution of a

CGI script, a separate process needs to be created on the server. The API approach consumes a great deal less memory and performs a whole lot better than running CGI scripts.

Server-Side Includes (SSI)

When a browser requests for a file, the Web server just fetches the file based on the URL and simply sends the file over to the browser. Normally, the server just verifies that the user is authorized to receive the file and simply hands over the file without looking at it.

Typically, graphics are included in the web page and sent to the browser. It is the browser that scans the web page received, scans and interprets the contents, and takes action. This process of scanning and interpretation, known as parsing, usually takes place on the client side of the system.

If the server can be made to do the parsing of the web page before it sends it on to the browser, then we can make the server take action based on what the server finds on the web page. Instead of the server blindly passing on a web page to the browser, the server can parse and interpret the page first. This process of the server doing the parsing is called server-side include (SSI).

Initially, SSI was used to include other files in addition to the one requested in the web page sent to the browser. This information to include other files is embedded in the web page. The server parses the web page first, notes the indication to include other files, and then includes the other files as part of the information sent back to the browser. Under SSI, generally, the end result conveyed to the browser is in the form of text.

SSI commands are embedded like comments within regular HTML pages. Therefore, like comments, all SSI commands begin with the characters <!—# and end with the characters — >. The following command directs the inclusion of the file containing the current address in a web page:

<!—# include file = “currentaddress.htm — >.

Embedding SSI commands as comments makes it easy to implement SSI. HTML pages with SSI commands are still portable. Any server that does not support SSI does not parse and interpret the commands; it simply passes on the commands to the browser; the browser simply ignores the commands as though they are comments.

Each vendor of Web servers may implement SSI in a proprietary fashion. There is no standard governing SSI; each vendor is free to implement SSI as it chooses. However, most vendors follow the basic SSI specifications outlined by NCSA, which maintain these general rules.

As you have noted, with SSI, you can include files in a Web document. But SSI can accomplish many other actions. You can include special commands to report the size of a file or the date when it was last modified. You may require the current date to be presented. You may embed a special command to send an e-mail to a given address.

Although you accomplish a number of different actions with SSI, what we are interested is database integration with the Web and how SSI enables this integra-

tion. A special SSI command relates to the execution of a subprogram similar to a CGI script. On parsing and recognizing this command, a Web server can initiate the execution of a subprogram and include its result in the web page to be delivered to the browser. And this subprogram can be a script to access a database, perform database operations, and produce results from the database.

Cookies

As we have mentioned several times in our previous discussions, HTTP is a stateless protocol. But for interactive applications such as database applications, the application program must know whether a transaction is part of an interactive session. As you know, under HTTP, the Web server maintains no state information. That is, when a user contacts a website through an URL submitted at the client site using a browser, the Web server does not know whether this is the user's first visit or thirtieth visit.

For example, in an electronic shopping encounter, the application program must know which shopping cart belongs to whom. Let us say that the database table representing a shopping cart with three items so far stores the user's name and password that came in as part of the registration. Now the user returns to the website through the browser to add a fourth item. Because of the stateless nature of HTTP, the server has no idea whether the user has returned for a fourth time or this is a brand new first visit. Any CGI script interfacing with the database server cannot receive any state information from the Web server.

Let us stop the discussion right here. Imagine that when the user returns to the website, the CGI script is able to request the browser for a file stored at the browser site containing user registration information. Suppose from the registration file received, the script is able to determine that the user is the same person with a virtual shopping cart with three items. Then this file stored at the browser end somehow is able to maintain state to the extent required by the shopping cart application. This small text file stored on the Web client system is known as a cookie (just called cookie for no valid reason). More formally, these files are referred to as persistent client state HTTP cookies. A cookie may persist at a client site for many months or even years.

How Cookies Work The use of cookies makes CGI scripts more interactive. A cookie is a small file created by a CGI script and stored at the browser site. In its simplest form, a cookie consists of a name-value pair. A developer may choose any such pair to create a cookie. More advanced implementations include expiration dates and restrictions on which Web pages can see the cookie as part of the cookie definition.

In our electronic shopping example, when the user visits the website to begin shopping, the CGI script creates a cookie, perhaps consisting of name and password as the name-value pair, and sends it to the user's browser. The browser receives the cookie and stores it on the hard drive at the client site. At a later time, when the user goes back to add items to the shopping cart, the CGI script requests for the cookie from the client side and compares with the registration information stored in the database during the previous visit. Based on the results of the comparison, the interaction proceeds further.

Some Limitations Usually, a special cookie file stores cookies on the user's computer system. Therefore, cookies are vulnerable to the whims of the user's actions. Accidentally or willfully, the user may delete the cookie file. Again, if the cookie file is not large enough or if the browser imposes restrictions on the size and number of cookie files, older cookies may be overwritten with later ones. Also, if you switch browsers, the cookies stored by the earlier browser will be lost.

Another drawback relates to name and password information stored in a cookie file. This information is usually not encrypted and therefore could cause serious problems. As a user, you are not aware of the cookies being stored on your computer's hard drive. Some browsers may be configured to alert users when cookies are being stored. Some browsers allow you to prevent any cookies from being set on your computer system at all.

Use of Java Applications

As most IT professionals know, Java is a proprietary programming language developed by Sun Microsystems. Java has been designed and developed to be completely portable. It is a "write-once, run anywhere" language. A few attributes of Java are that it is simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, and dynamic.

Java's portability represents its greatest strength. It is machine-independent. For most programming languages, first you need to establish the processor and operating system on which the completed program will be running. After compiling, the linker must prepare the executable object file with components from the operating system libraries. The whole process is different for a Java program.

The Java compiler generates a file of bytecodes, called a class file. These are instructions of the Java Virtual Machine (JVM). JVM does not physically exist; however, JVM (interpreter and runtime system) may be ported to any kind of computer. You can load a class file to any computer with JVM. The Java program will run on the JVM of that computer. Therefore, a Java program is totally portable. Figure 19-8 presents the components of the Java programming environment.

Well, what has the portability of Java to do with Web computing? Currently, all available browsers have embedded JVM. Therefore, a Java program can run any computer with a browser. That includes an enormous number of and many types of computer systems. Java is generally accepted as the preferred programming language for applications on the Web. Using Java, you can build Web applications, called applets, and server applications, known as servlets.

Applets. These are Java programs that are embedded in an HTML page. A Java-enabled browser executes an applet. The browser applies its own JVM interpreter to execute the bytecode of an applet. Applets are generally written to enhance the user interface and add dynamic content to HTML pages.

Servlets. These add functionality to Web servers. Whereas an applet executes on the browser, a servlet runs on the server. A servlet may connect to database servers, process the application, and return the results in a web page.

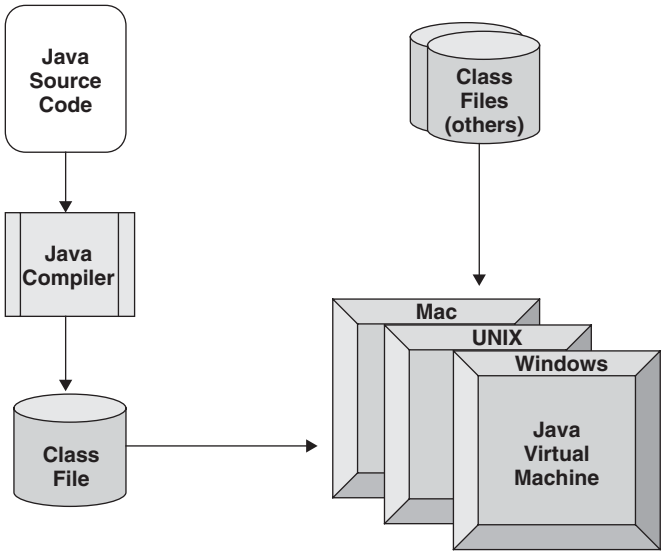

Figure 19-8 Java programming environment.

JDBC JDBC (Java Database Connectivity) bases its structure on ODBC (Open Database Connectivity), commonly used to provide interfaces between application programs and any type of ODBC-compliant database systems. JDBC is the most effective approach for Java programs to access relational database systems. JDBC allows you to write Java programs that would work with any database system that has a JDBC driver. The driver provides a uniform interface to a Java program irrespective of the type of database system. Most database vendors offer JDBC drivers for use with their products.

The JDBC method involves two components for the API—one on the side of the program and the other on the side of the database server. A Java program can use the JDBC API to establish a connection to the database server through a pure Java JDBC driver at the database server side.

Here are the common options for the connectivity:

Driver with Direct Connection In this method, the pure Java JDBC driver establishes a direct connection with the database server. JDBC calls are converted into the network protocol used by the DBMS, enabling a direct call from the client machine to the database server. This seems to be a practical approach for database access on the organization's intranet.

Driver Connecting Through Middleware In this method, JDBC calls are converted into the protocol of the middleware vendor and subsequently translated into the DBMS protocol by the middleware server. Because the middleware provides connectivity to a number of database systems, this method has flexibility.

The JDBC-ODBC Bridge This method provides JDBC access through ODBC drivers. ODBC code is required to be loaded on a client machine that adopts this

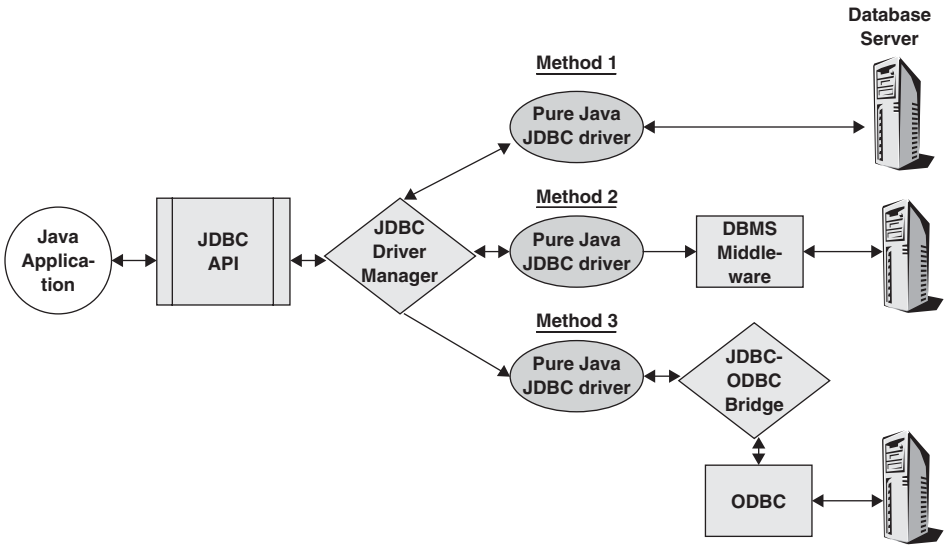

Figure 19-9 JDBC approaches.

approach. Because ODBC has become the de facto standard for database access, this is a popular method.

Figure 19-9 illustrates these three approaches of using JDBC. In each case, note the layers through which interface between a Java application program and a database system is established.

Use of Scripting Languages

This is another method for providing database functionality to browsers and servers. JavaScript from Netscape and JScript from Microsoft are the commonly used scripting languages. JavaScript resembles Java in its structure. Another popular scripting language is VBScript from Microsoft. The syntax of VBScript is more like that of Visual Basic than Java.

A script written in any of these languages may be embedded in the HTML and downloaded every time the page is accessed. The scripting languages are interpreted, not compiled. Therefore, they are well suited for small applications.

Scripts executing on the browser have objectives similar to applets written in the Java language. Figure 19-10 presents a comparison between client-side JavaScript and Java applets.

JavaScript on the Server Side On the server side, JavaScript can perform the functions related to database operations, as listed below:

- Establish connection to a database.
- Begin database query or transaction.
- Retrieve data requested in the query or perform database operations included in the transaction such as inserts, updates, or deletes.

	Java Applets	Java Script
Code Generation	Compiled on server	Interpreted by client
HTML integration	Separate – accessed from HTML documents	Integrated and embedded in HTML documents
Data typing	Strong typing	Loose typing
Binding	Static binding	Dynamic binding
Object technology	Object-oriented – with classes and inheritance	Object-based – no classes or inheritance
Updates	No automatic write to hard disk	No automatic write to hard disk

Figure 19-10 Java applets and JavaScript: comparison.

- Commit or abort and roll back a transaction according to the conditions.
- Display results of a query.
- Where necessary, access multimedia content from the database in the form of BLOBs (binary large objects).
- Terminate connection from the database.

Database Tools

We have examined various techniques of integrating databases into the Web environment. We covered different techniques for extending browser and server functionalities to access databases on the Web. Let us now briefly consider a few third-party tools that enable the creation of Web applications and linking them to databases. Only a few tools work with any type of database systems. Some of them are proprietary, geared toward specific DBMSs. Therefore, the tool you would want to use depends on a few significant factors such as:

- The database you are using
- The platform where your Web server and database server are running
- The technique for database access—CGI or proprietary API
- Cost considerations
- Maturity of the tool
- Vendor stability

Now, let us selectively review a few tools without getting into details. The discussion is simply meant to pique your interest and prompt you to be open to the possibility of using such tools in your database environment.

ColdFusion Created by Allaire, ColdFusion enables the writing of scripts within an HTML document. ColdFusion executes the scripts and returns results within the HTML document. The tool works with every Web server running on Windows NT and interfaces with every SQL engine using ODBC.

Figure 19-11 presents the working of ColdFusion.

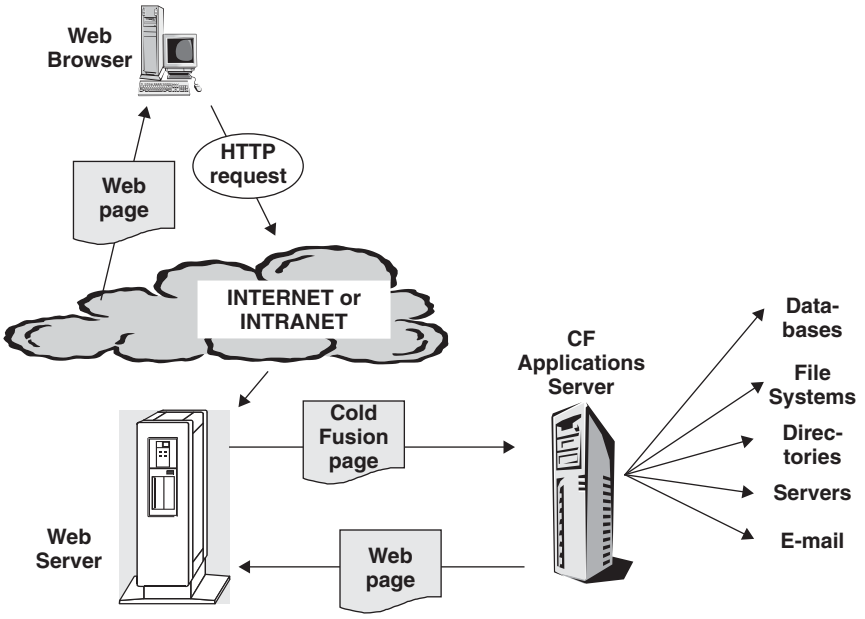

Figure 19-11 How ColdFusion works.

Active Server Pages The Active Server Platform is Microsoft's response to server-side application development. You can develop on the server side with HTML, scripts, and software modules such as Java applets and ActiveX server components. Several Microsoft server technologies, including Transaction Server and Proxy Server, are available to implement applications.

You can develop Active Server Pages (ASP) with the same set of components used for client-side scripting. This means that you can combine Java applets and Active Server components within your ASP script using scripting languages such as JavaScript, Jscript, or VBScript.

Browser Capabilities Component (BCC) and Active Data Object (ADO) are two important precreated and tested components that are used widely. With BCC you can detect the type of browser and create the kind of HTML pages the browser can handle. ADO, a more useful component, works with your server-side scripts and enables you to access databases and embed the query results in a dynamic web page.

The following indicates the sequence of actions:

Server-side Active Server Page script communicates an SQL query to ADO.

ADO passes query to database server.

Database server processes query.

Database server sends query results to ADO.

ADO returns query results to the ASP script.

Script uses query results to generate dynamic web page and sends it to browser.

Other Tools Here are few other tools that are also available for Web-database integration. The names of the tools and the website addresses are indicated below.

PHP	http://www.php.net/
W3-mSQL	http://www.hughes.com.au/software/w3-msql.htm
MsqlPerl	ftp://Bond.edu.au/pub/Minerva/msql/Contrib/
MsqlJava	http://mama.minmet.uq.oz.au/msqljava/
WDB	http://arch-http.hq.eso.org/wdb/html/wdb.html
Web/Genera	http://gdbdoc.gdb.org/letovsky/genera/
MORE	http://rbse.jsc.nasa.gov:81/DEMO/
DBI	http://www.hermetica.com/technologia/DBI/
DBGateway	http://fcim1.csdc.com/

SECURITY OPTIONS

In Chapter 16, we discussed database security in detail. We covered security administration. You are aware of the security risks and the general methods of protecting the organization's database system. Now, when your database is integrated with the Web, the security concerns intensify. The Internet is traditionally an open network. The underlying protocols, TCP/IP and HTTP, are not designed for security. Any packet-sniffing software can easily retrieve sensitive data from the open network.

First, we will explore the types of security vulnerabilities and the challenges. After that, we will broadly examine a few significant security options.

Significance of Protection

Several times so far we have reviewed examples from electronic shopping. After filling the virtual shopping cart, let us say, the user is ready to pay with a credit card. When the user enters the credit card data and finishes the transaction, data transmission to the server begins. The credit card details are part of the transmission. Any ordinary packet-sniffing software can detect the details and siphon out the information. In the Web environment, protection must be addressed at different levels.

At Each Tier Information transmitted over the public network moves around and halts at each of the tiers in a three-tier architecture. Protection of information must be addressed at the client, at the Web server, and also at the database server. This requires the use of suitable techniques and security products.

Special Concerns at Client Machine As you have seen, in the Web environment, information transmitted to the client may contain executable code. An HTML page transmitted to the browser may have embedded JavaScript or VBScript or one or more Java applets. These are coming from, perhaps, an unknown server and are likely to cause damages including the following on the client machine:

- Corrupt data.
- Delete or reformat entire disks.
- Steal user identification and password, and impersonate user in other computer systems on the client machine.

- Download confidential and sensitive data.
- Disrupt output devices.
- Lock up resources and render them unavailable to authorized users.
- Shut down the system totally.

Information Exchange Challenges Security protection must ensure that sending and receiving information over the Internet conform to the following safeguards:

- Information specifically transmitted from one person to another is not available to anyone else.
- Information is not modified in transmission.
- The receiver of information is sure that it actually came from the sender as identified.
- The sender is sure that the receiver is authentic.
- The sender is unable to deny that he or she sent the information.

Firewalls, Wrappers, and Proxies

This is a set of defense mechanisms to protect data that are vulnerable over the Internet. Let us briefly survey these.

Firewall A firewall may be implemented with software, hardware, or both. It is meant to prevent unauthorized access to a private network. Most commonly, a firewall is used for protecting an organization's intranet from the Internet by keeping unauthorized users outside. All messages from outside the intranet are intercepted and checked.

Figure 19-12 illustrates the placement of firewall in conjunction with an intranet. Note how users are grouped within and outside the firewall.

Let us discuss how firewalls provide protection.

An IP Packet This is the basic unit for moving information across a TCP/IP network. All information exchange consists of a set of IP packets. The address components in each packet that ensure proper delivery are destination IP address, protocol, destination port number, source IP address, and source port number.

A Network Session A session may be thought of as comprising a set of IP packets sent between the initiation and completion of a request. Therefore, a set of IP packets with the same address information represents a session. A given network application may extend to several sessions to perform its service.

Standard Firewall Services Three major services are usually provided:

Access control. Using information filtered about an IP packet or a network session and matching it against the security policy of the organization, the firewall decides to let the packet pass or deny passage.

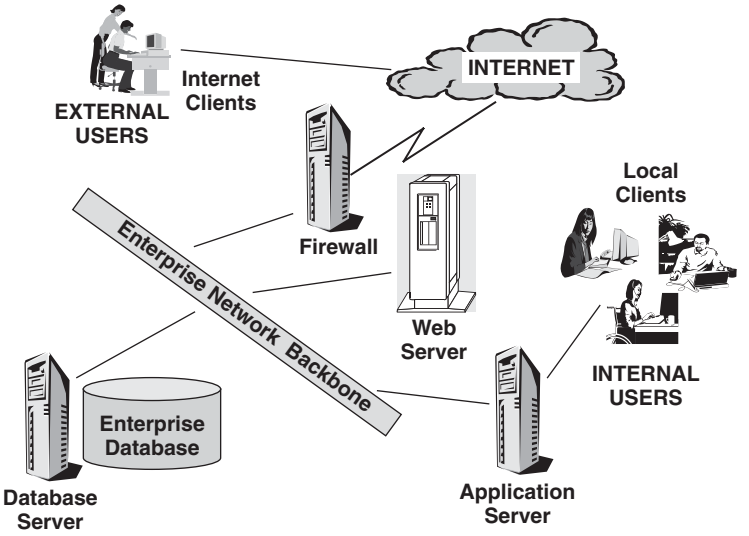

Figure 19-12 Intranet and firewalls.

Activity logging. A good firewall system logs all successful and failed attempts through it. This information is valuable for tracking any intruders and determining the extent of any inflicted damage.

Alarm function. In some firewall systems, the administrator can set an alarm to trigger a special event if an intruder tries to gain access to sensitive data.

Firewall Filtering Techniques The following are common techniques offered:

Packet filter. Matches key fields within an IP packet against defined rules and either passes the packet on or drops it. Dropped packets are discarded. Because packet filters have no concept of the session, this method is vulnerable to spoofing. Also, packet filters are difficult to configure.

Session filter. Retaining information on all active sessions through the firewall, a session filter determines whether packets flowing in the opposite direction belong to an approved connection. A session filter also performs session auditing.

Application-Level Gateway. This is a mechanism to apply interception rules to specific applications such as Telnet or FTP services.

Circuit-Level Gateway. This is a high-level protection mechanism applied when a TCP connection is made. Once the connection is authorized, packets can flow between computer systems without further verifications.

Wrappers Although firewalls are effective in keeping intruders from outside, they can do very little to prevent unauthorized access from within. Wrappers form the

second line of defense. A wrapper runs as a surrounding layer of software, wrapping around other software.

For example, if a user is attempting to do a file transfer to you through FTP, the user will first get a wrapper that would then engage FTP. The wrapper is transparent to the user. Wrappers offer a lot of flexibility. Wrappers can act like firewalls and can refuse access based on the user sign on. Also, wrappers can give indications on how the security system is working because they log all accesses. Another interesting use for wrappers is the creation of blind alleys that can help trap intruders.

Proxy Servers A proxy server is a computer system placed between a browser and a Web server. The proxy server intercepts all requests to the server and determines whether the request can be approved. First, the proxy server makes a determination of whether it could fulfill the request by itself. If not, and if the request is legitimate, the proxy server passes on the request to the Web server.

By saving the results of requests for a certain period, a proxy server is able to fulfill requests without sending them to the regular Web server. In this way, proxy servers can improve performance.

Proxy servers can effectively hide the actual location of data. For example, proxy servers can point to other proxy servers. The actual data can sit far away from the server itself. A local or remote browser connects to a server. But the server can forward the request to another server, and the second server can forward the request to a third server, and so on. The client cannot know where exactly the data come from.

A separate proxy server can be assigned to each major service such as Telnet, FTP, and so on. By doing this, you can route requests accordingly and distribute server loads to different physical machines. In addition to the benefit of data hiding, you also benefit from reduced load on the servers.

Digital Signatures and Digital Certificates

First, try to understand what a message digest algorithm is. Take a message in the form of an arbitrary string and generate a fixed-length string of bits, called a digest of the original message. An algorithm that can generate such a digest is known as a message digest algorithm. No two messages can generate the same digest. The digest gives no clues about the original message. We now proceed to the application of message digests.

Digital Signatures Similar to regular handwritten signatures, digital signatures can be used to verify the origin of a message. A digital signature consists of two parts: a bit string computed from the data that are being signed and the private key of the signatory (an individual user or an organization).

Here are the main features of digital signatures:

- You can verify the authenticity of a digital signature by applying a computation based on the corresponding public key of the user.
- No one can forge a digital signature provided the private key has been kept secret, as required.

- A digital signature is computed for the data that are signed. So a particular digital signature cannot be alleged to be for some other data.
- A signed piece of information cannot be changed. Otherwise, the digital signature will not apply to the piece of information any longer.

Parts of some digital signatures are generated by using message digest algorithms.

Digital Certificates These refer to another security mechanism. A recognized Certificate Authority (CA) issues digital certificates.

Let us say that Mary wants to send a secure electronic message to Bob. She applies to the CA for a digital certificate. The CA issues an encrypted digital certificate containing Mary's public key and additional identification information. The CA's own public key is widely known through printed documents or on the Internet. Mary sends the digital certificate as an attachment to the electronic message to Bob. In this arrangement, the recipient of a message confirms the proper receipt of the message by means of a response.

What happens next?

Bob receives the message and the digital certificate. He uses the CA's public key, decodes the digital certificate, verifies that it was really issued by the CA, and gets the sender's public key and identification encrypted in the certificate. Bob can send back an encrypted response to Mary. In effect, Bob is able to verify that the message that appeared to have been sent by Mary really was from her.

Digital certificates have their application in monetary transactions such as large fund transfers. In this security scheme, you must have noted that the CA plays a key role acting as an intermediary between the sender and the recipient of a message. What if a fraudulent person tries to forge or pose as someone else or tries to bribe the CA? Other additional restrictions exist to thwart the deceptive schemes of the cleverest imposter.

Kerberos In passing, let us note the use of Kerberos in security administration. Kerberos is a secured server keeping user names and password in strictest confidentiality. It provides one central security server for all data and network resources. In a sense, Kerberos provides a function similar to that of the Certificate Authority—authenticate a user. Incidentally, the term Kerberos refers to a three-headed monster in Greek mythology that guards the gate of Hell.

SET and SST

Have you ever provided your credit card information on the Internet? The Secure Electronic Transactions (SET) protocol governs the processing of credit card transactions over the Internet. It is an open, interoperable standard created jointly by Visa, MasterCard, Microsoft, and a few other organizations. The protocol ensures security for credit card transactions and keeps these simple.

To ensure privacy requirements, the actions in a transaction are split in such a way that neither the merchant nor the credit card company has all the pieces of the information about the transaction. SET uses digital certificates for certifying a credit cardholder as well as for confirming the relationship between the merchant and the credit card company.

Visa and Microsoft, major players in the development of SET, have since provided Secure Transaction Technology (SST). This technology enables bank payments over the Internet to be secure.

SSL and S-HTTP

Netscape developed the Secure Sockets Layer (SSL) protocol for transmitting private documents over the Internet in a secured manner. Many websites use this standard to obtain confidential information such as credit card numbers. The protocol prevents tampering, forgery, and eavesdropping by unauthorized persons.

SSL is layered between services like FTP, Telnet, and HTTP and the TCP/IP connection sessions. It guarantees a secure connection between a client and a server. Once a secure connection is established, the server can transmit any amount of confidential information to the client over the safe connection. SSL verifies that the client and server networks are valid, provides data encryption using a private key encryption method, and ensures that the message does not contain any embedded commands or programs.

Secure HTTP (S-HTTP) offers a different method to transmit data securely from the server to the client. This protocol, a revised version of HTTP, allows single messages to be transmitted securely. Enterprise Integration Technologies (EIT) developed S-HTTP.

SSL and S-HTTP both enable secure transmission of confidential information over the Internet—SSL establishes a safe connection, and S-HTTP safeguards individual messages. Therefore, one method complements the other. In effect, both technologies offer the following functions:

- Enable authentication between a browser and a server
- Allow website administrators to control access selectively over specific services
- Permit a browser and a server to share confidential information while keeping others away from the protected transmission
- Ensure reliability of transmitted information, preventing accidental interruption or corruption

Java Security

You know that Java applets, embedded in a web page and transmitted from the Web server to the browser, execute on the client machine. This is unlike a known program executing on your computer. As a user on the client machine, you may not know where the executable code is coming from. Perhaps, if the applet needs local resources to execute, it will ask for permission through the browser. You may grant permission or deny permission to execute. But even if you know the source of the web page containing the applet, you cannot be completely sure. Nevertheless, you must allow a useful applet to execute and still maintain a level of trust. Network applications cannot be operational if you unequivocally deny permission to any outside programs to do useful work on your client machine. This is the crux of Java security.

What can be done? Are there any guidelines to permit applets to execute on the client machine but ensure that no damages are caused? The following are general guidelines:

- List all possible scenarios of malicious actions that could cause damages.
- Analyze and reduce the list into a basic set of wicked actions.
- Design an architecture and language protection mechanism that prevent the basic set of malicious actions.
- Test and prove that the architecture and the language are secure.
- Include flexibility in the design to accommodate any additional type of malicious actions.

Let us try to list the common types of damages that may be caused by malicious actions.

- Applet starts a process that depletes all system resources and brings the client machine to a halt.
- Application locks the browser.
- Applet tampers with the browser's Domain Name Service (DNS).
- Applet destroys other applications trying to load.
- Applet damages or deletes files on the client machine.
- Installs back door access into your network for future unauthorized entry.
- Accesses confidential files and give access privileges to other unauthorized users.
- Assumes your identity and impersonates you for the purpose of attacking other computer systems.

What about the major strength of Java—its portability? If a Java program can run on any computing platform, obviously, Java security cannot rely on security provisions in any operating system. So, safety and security provisions in Java have to be self-contained. Security and safety are, therefore, integral parts of the design of the language. The sandbox concept ensures that no undependable, malicious program can gain access. For this purpose, Java security is implemented using three components: class loader, bytecode verifier, and security manager. Figure 19-13 illustrates Java security implementation.

The three components work together and ensure the following safeguards:

- Only the proper classes are loaded.
- Each class is in the proper format.
- Undependable classes are not permitted to execute dangerous instructions.
- Undependable classes are not allowed to access system resources.

Class Loader A Java-enabled or Java-aware browser invokes the class loader to verify the format and load the incoming applet. The class loader defines a namespace for the particular web page. An executing JVM allows multiple class loaders

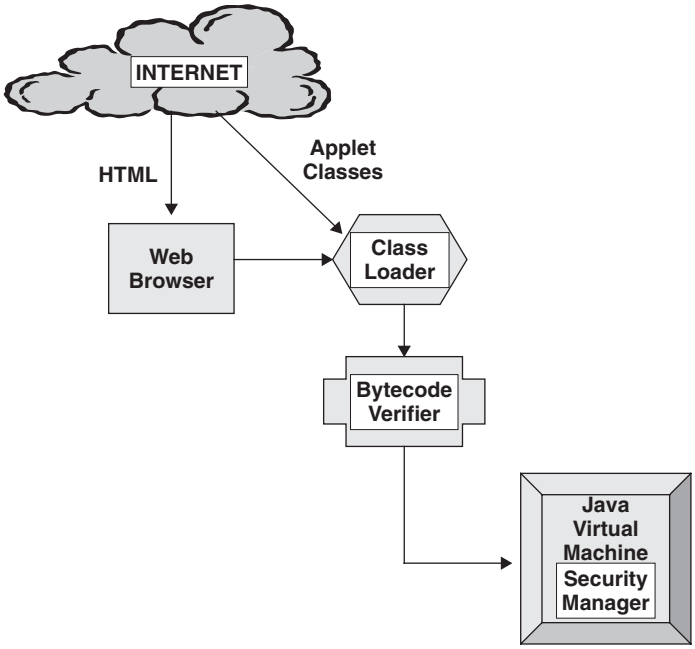

Figure 19-13 Java security.

to be active simultaneously, each with its own namespace. Classes loaded as part of one applet are not allowed to access other classes, although these can access classes in standard Java libraries.

Namespaces are organized in a hierarchical manner, allowing JVM to organize classes based on the origin—local or remote. A class loader does not permit a class from a less protected namespace to replace a class with a more protected namespace. Consider the file system's I/O primitives. These are defined in a local Java class. Therefore, the file system's I/O primitives cannot be invoked or replaced by classes from outside the local machine.

Bytecode Verifier The JVM does not allow an applet to run before the bytecode verifier checks its bytecode completely. The bytecode verifier makes the assumption that the code is designed to crash the system and intends to violate security. From this perspective, the bytecode verifier conducts a series of checks, including the execution of a sophisticated theorem prover to negate its assumption. The theorem prover ensures that the applet does not forge pointers, bypass access restrictions, or perform illegal conversion of objects.

Specifically, verification by the bytecode verifier ensures the following:

- Compiled code is correctly formatted.
- Internal stacks will not overflow or underflow—traditionally how programmers breach security.
- No illegal conversions, such as integer to pointer, will occur.
- Bytecode instructions are correctly typed.

- Every class member access is valid.
- The applet cannot create disasters in its own namespace.

Security Manager The security manager performs the final set of checks. The security manager is responsible to preserve thread integrity—to ensure that code in one group of threads does not interfere with code in another group, even if the two groups belong to the same parent applet.

The security manager watches out for dangerous operations that could cause potential damages. The security manager monitors such operations closely. Every time the applet wants to perform any such operation, it must gain permission from the security manager. Specifically, the security manager must concur with attempts by the applet to access any of the following:

- Network communications
- Protected information including the hard drive or private data
- Programs and processes at the level of the operating system
- The class loader
- Libraries of Java classes

Generally, an applet is prevented from the following actions:

- Loading libraries
- Defining method calls
- Starting other programs on the client
- Reading or writing files on the client machine
- Making network connections to machines other than the one from where the applet emanated

CHAPTER SUMMARY

- Static web pages are no longer adequate for today's information requirements; Web and database technologies had to be integrated.
- The basics: the Internet is a collection of interconnected networks; a URL acts as an address on the Web; you use HTML to create web pages; HTTP is the object-oriented protocol for exchanging web pages.
- A static web page displays the same information every time it is accessed; a dynamic web page customizes and shows changed content based on requests.
- HTML is simple and straightforward but has many inadequacies. XML overcomes the deficiencies and provides a better solution.
- An intranet is a private network based on the communication standards of the public Internet. An extranet is open to selective access by users outside an organization.
- The Web has become the cornerstone of electronic commerce. Databases are the centerpieces of electronic commerce. Web-database integration makes electronic commerce possible.

- CGI is a common approach for integrating Web and database technologies.
- APIs extend the functionality of the Web server and enables Web-database integration more efficiently than CGI.
- With SSI, you can include files in a Web document; SSI can also accomplish other functions including special commands.
- Cookies make CGI scripts more interactive. A cookie is a small file created as a CGI script and stored at the browser site.
- Using Java language, you can build Web applications called applets and server applications known as servlets.
- JDBC is an effective approach for Java programs to access relational databases.
- Security options for Web-based databases: firewalls, wrappers, and proxies. Other security mechanisms: digital signatures, digital certificates, SET, SST, SSL, and S-HTTP.
- Java language offers security through three components: class loader, bytecode verifier, and security manager.

REVIEW QUESTIONS

1. Name the four stages or phases of information exchange under HTTP.
2. Compare and contrast static and dynamic web pages.
3. What are intranets and extranets? List any three of their major benefits.
4. List any six types of routine security functions.
5. State any four reasons for integrating an organization's database with its website.
6. Are there any disadvantages of Web-database integration? If so, list any three.
7. What is CGI? List its main advantages.
8. What is an API? How is it different from CGI?
9. How do scripting languages enable the provision of database functionality to browsers and servers?
10. Describe very briefly the three security components of Java.

EXERCISES

1. Match the columns:

1. URL	A. hide actual location of data
2. SSI	B. Java's security component
3. cookie	C. tool for Web-database integration
4. servlets	D. Internet reference address
5. ColdFusion	E. Web-enabled private network
6. wrappers	F. server parses and interprets a web page
7. proxy servers	G. add functionality to Web servers

8. SSL — H. second line of security mechanism
9. class loader — I. small file stored at browser site
10. intranet — J. guarantees secure connection

2. Describe the major features of HTML. What are its advantages and disadvantages?
3. Describe briefly any three approaches to Web-database integration. Compare their relative merits and shortcomings.
4. What are firewalls? Describe their purpose and functions.
5. A major manufacturing company producing passive electronic components wants to place its database on its extranet for use by its distributors to check stock and place orders. Prepare a general plan for implementing the integration of Web and database technologies at this company. List all the major technologies to be used for the integration.