CHAPTER 9

Application Development

Solutions for the Practice Exercises of Chapter 9

Practice Exercises

9.1

Answer:

The CGI interface starts a new process to service each request, which has a significant operating system overhead. On the other hand, servlets are run as threads of an existing process, avoiding this overhead. Further, the process running threads could be the web server process itself, avoiding interprocess communication, which can be expensive. Thus, for small to moderate-sized tasks, the overhead of Java is less than the overhead saved by avoiding process creation and communication.

For tasks involving a lot of CPU activity, this may not be the case, and using CGI with a C or C++ program may give better performance.

9.2

Answer:

Most computers have limits on the number of simultaneous connections they can accept. With connectionless protocols, connections are broken as soon as the request is satisfied, and therefore other clients can open connections. Thus more clients can be served at the same time. A request can be routed to any one of a number of different servers to balance load, and if a server crashes, another can take over without the client noticing any problem.

The drawback of connectionless protocols is that a connection has to be reestablished every time a request is sent. Also, session information has to be sent each time in the form of cookies or hidden fields. This makes them slower than the protocols which maintain connections in case state information is required.

9.3

Answer:
A hacker can edit the HTML source code of the web page and replace the value of the hidden variable price with another value, use the modified web page to place an order. The web application would then use the user-modified value as the price of the product.

9.4

Answer:
Although the link to the page is shown only to authorized users, an unauthorized user may somehow come to know of the existence of the link (for example, from an unauthorized user, or via web proxy logs). The user may then log in to the system and access the unauthorized page by entering its URL in the browser. If the check for user authorization was inadvertently left out from that page, the user will be able to see the result of the page.
The HTTP referer attribute can be used to block a naive attempt to exploit such loopholes by ensuring the referer value is from a valid page of the web site. However, the referer attribute is set by the browser and can be spoofed, so a malicious user can easily work around the referer check.

9.5

Answer:
This ensures connections are closed properly, and you will not run out of database connections.

9.6

Answer:
Caching can be used to improve performance by exploiting the commonalities between transactions.

a. If the application code for servicing each request needs to open a connection to the database, which is time consuming, then a pool of open connections may be created beforehand, and each request uses one from those.

b. The results of a query generated by a request can be cached. If the same request comes again, or generates the same query, then the cached result can be used instead of connecting to the database again.

c. The final web page generated in response to a request can be cached. If the same request comes again, then the cached page can be outputed.

9.7

Answer:

The tester should run netstat to find all connections open to the machine/socket used by the database. (If the application server is separate from the database server, the command may be executed at either of the machines). Then the web page being tested should be accessed repeatedly (this can be automated by using tools such as JMeter to generate page accesses). The number of connections to the database would go from 0 to some value (depending on the number of connections retained in the pool), but after some time the number of connections should stop increasing. If the number keeps increasing, the code underlying the web page is clearly not closing connections or returning the connection to the pool.

9.8

Answer:

a. One approach is to enter a string containing a single quote in each of the input text boxes of each of the forms provided by the application to see if the application correctly saves the value. If it does not save the value correctly and/or gives an error message, it is vulnerable to SQL injection.

b. Yes, SQL injection can even occur with selection inputs such as drop-down menus, by modifying the value sent back to the server when the input value is chosen—for example by editing the page directly, or in the browser's DOM tree. Most modern browsers provide a way for users to edit the DOM tree. This feature can be able to modify the values sent to the application, inserting a single quote into the value.

9.9

Answer:

It is not possible in general to index on an encrypted value, unless all occurrences of the value encrypt to the same value (and even in this case, only equality predicates would be supported). However, mapping all occurrences of a value to the same encrypted value is risky, since statistical analysis can be used to reveal common values, even without decryption; techniques based on adding random "salt" bits are used to prevent such analysis, but they make indexing impossible. One possible workaround is to store the index unencrypted, but then the index can be used to leak values. Another option is to keep the index encrypted, but then the database system should know the decryption key, to decrypt required parts of the index on the fly. Since this requires modifying large parts of the database system code, databases typically do not support this option.

The primary-key constraint has to be checked by the database when tuples are inserted, and if the values are encrypted as above, the database system will

not be able to detect primary-key violations. Therefore, database systems that support encryption of specified attributes do not allow primary-key attributes, or for that matter foreign-key attributes, to be encrypted.

9.10

Answer:

When the entire database is encrypted, it is easy for the database to perform decryption as data are fetched from disk into memory, so in-memory storage is unencrypted. With this option, everything in the database, including indices, is encrypted when on disk, but unencrypted in memory. As a result, only the data access layer of the database system code needs to be modified to perform encryption, leaving other layers untouched. Thus, indices can be used unchanged, and primary-key and foreign-key constraints enforced without any change to the corresponding layers of the database system code.

9.11

Answer:

The key problem with digital certificates (when used offline, without contacting the certificate issuer) is that there is no way to withdraw them.

For instance (this actually happened, but names of the parties have been changed) person C claims to be an employee of company X and gets a new public key certified by the certifying authority A. Suppose the authority A incorrectly believed that C was acting on behalf of company X, and it gave C a certificate *cert*. Now C can communicate with person Y, who checks the certificate *cert* presented by C and believes the public key contained in *cert* really belongs to X. C can communicate with Y using the public key, and Y trusts the communication is from company X.

Person Y may now reveal confidential information to C or accept a purchase order from C or execute programs certified by C, based on the public key, thinking he is actually communicating with company X. In each case there is potential for harm to Y.

Even if A detects the impersonation, as long as Y does not check with A (the protocol does not require this check), there is no way for Y to find out that the certificate is forged.

If X was a certification authority itself, further levels of fake certificates could be created. But certificates that are not part of this chain would not be affected.

9.12

Answer:

A scheme for storing passwords would be to encrypt each password (after adding randomly generated "salt" bits to prevent dictionary attacks), and then use a hash index on the user-id to store/access the encrypted password. The password being used in a login attempt is then encrypted (if randomly gener-

ated "salt" bits were used initially, these bits should be stored with the user-id and used when encrypting the user-supplied password). The encrypted value is then compared with the stored encrypted value of the correct password. An advantage of this scheme is that passwords are not stored in clear text, and the code for decryption need not even exist. Thus, "one-way" encryption functions, such as secure hashing functions, which do not support decryption can be used for this task. The secure hashing algorithm SHA-1 is widely used for such one-way encryption.