CHAPTER 8

Application Design and Development

Solutions to Practice Exercises

8.1 The CGI interface starts a new process to service each request, which has a significant operating system overhead. On the other hand, servelets 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 overheads saved by avoiding process creating 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.

8.2 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 form of cookies or hidden fields. This makes them slower than the protocols which maintain connections in case state information is required.

8.3 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 before hand, and each request uses one from those.

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

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

8.4 For inserting into the materialized view *branch_cust* we must set a database trigger on an insert into *depositor* and *account*. We assume that the database system uses *immediate* binding for rule execution. Further, assume that the current version of a relation is denoted by the relation name itself, while the set of newly inserted tuples is denoted by qualifying the relation name with the prefix – **inserted**.

The active rules for this insertion are given below –

```
define trigger insert_into_branch_cust_via_depositor
after insert on depositor
referencing new table as inserted for each statement
insert into branch_cust
    select branch_name, customer_name
    from inserted, account
    where inserted.account_number = account.account_number
```

```
define trigger insert_into_branch_cust_via_account
after insert on account
referencing new table as inserted for each statement
insert into branch_cust
    select branch_name, customer_name
    from depositor, inserted
    where depositor.account_number = inserted.account_number
```

Note that if the execution binding was *deferred* (instead of immediate), then the result of the join of the set of new tuples of *account* with the set of new tuples of *depositor* would have been inserted by *both* active rules, leading to duplication of the corresponding tuples in *branch_cust*.

The deletion of a tuple from *branch_cust* is similar to insertion, except that a deletion from either *depositor* or *account* will cause the natural join of these relations to have a lesser number of tuples. We denote the newly deleted set of tuples by qualifying the relation name with the keyword **deleted**.

```
define trigger delete_from_branch_cust_via_depositor
after delete on depositor
referencing old table as deleted for each statement
delete from branch_cust
    select branch_name, customer_name
    from deleted, account
    where deleted.account_number = account.account_number
```

```
define trigger delete_from_branch_cust_via_account
after delete on account
referencing old table as deleted for each statement
delete from branch_cust
    select branch_name, customer_name
    from depositor, deleted
    where depositor.account_number = deleted.account_number
```

8.5 Query:

```
create trigger check-delete-trigger after delete on  account
referencing old row as orow
for each row
delete from depositor
where depositor.customer_name not in
    ( select customer_name from depositor
    where account_number <> orow.account_number )
end
```

8.6 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 get 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, it gives C a certificate $cert$. Now, C can communicate with person Y, who checks the certificate $cert$ presenetd by C, and believes the public key contained in $cert$ really belongs to X. Now C would 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 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 can be created. But certificates that are not part of this chain would not be affected.

8.7 A scheme for storing passwords would be to encrypt each password, and then use a hash index on the user-id. The user-id can be used to easily access the encrypted password. The password being used in a login attempt is then encrypted and compared with the stored encryption 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!