CMPUT 391

Project Report

Randy Wong

Marcin Pietrasik

Xin Ying Wang

The overall system is built from a mix of servlets and scriplets within jsp. The general flow of the site is Login → IndexFile → Specialized Pages for Modules.

Login Module:

The original login page is a .jsp that has a check to see if you are currently logged in, if you are you will be redirected to the index page. Otherwise, you are given the chance to enter a valid username and password. When entered a servlet “Authenticate” is called which checks the given username and password with those within the Users table. Specifically the query is below, and then the password is checked. If the passwords match, session attributes are created that include the class and person\_id of the user for use in other modules.

select password, class, person\_id from users u where u.user\_name = (entered\_username)

Next, there is an “Update Personal Information” page that allows users to update all attributes in their persons row except for person\_id, and they can also update their password. Here two servlets are used to deal separately with the password update and the information update. The information update is the same servlet that is used by the Updater in the User Management Module, and will be discussed later. The password update is a simple servlet that takes the entered new password, and retrieves person\_id from the session to perform the following update statement.

Update users set password = (entered\_password) where person\_id=(session\_pid)

User Management Module:

This module is accessible only by admin accounts, and allows updates/adds to users, persons, and doctors. When this module is accessed the user is given 6 options for each of the above cases, and those will then each call individualized servlets. The servlets create submission pages that allow the user to fill in the appropriate information, and then passes it to another servlet to perform. Here the “updateXsubmit” servlets are the ones which actually enter the information. The SQL statements are all similar in the following form.

Insert into persons values (all attributes)

OR

update persons set (first\_name = (entered\_firstname) … ) where person\_id=(entered\_pid)

Attributes in the above are only added to the SQL statement if specifically updated from previous page, empty fields were ignored.

Report Generating Module:

This module is also done through a servlet which simply calls the following SQL statement given fully correct parameters, and prints a table of the results.

SELECT p1.first\_name, p1.last\_name, p1.address, p1.phone, r1.test\_date

from persons p1, radiology\_record r2

where p1.person\_id = r1.patient\_id

and r1.diagnosis like ‘(diagnosis)’

and r1.prescribing\_date between to\_date('(from date)’,'MM-DD-YYYY') and to\_date('(to date)','MM-DD-YYYY')

and r1.test\_date IN

(select min(r2.test\_date) from radiology\_record r2

where r2.patient\_id = p1.person\_id

and r2.prescribing\_date between to\_date('(from date)','MM-DD-YYYY') and to\_date('to date','MM-DD-YYYY')

and r2.diagnosis like '(diagnosis)' )

Search Module:

This module is entirely done in a scriplet on itself. It builds the SQLstatement by examining all entered parameters. It generates small pieces of the statement depending on each entered value and then puts them together at the end. It generates a small piece depending on the choice for ordering, (r.test\_date desc for newest, r.test\_date asc for oldest, rank desc for rank). It then does this for keywords if necessary, separating keywords by space and checking singular keywords against all 4 fields of firstname, lastname, diagnosis, and description. An example SQL statement for 2 keywords and a date is shown below.

Select r.\*, 6\*score(1) + 6\*score(2) + 3\*score(3) + score(4)

+6\*score(5) + 6\*score(6) + 3\*score(7) + score(8) as rank

from radiology\_record r full join persons p on r.patient\_id = p.person\_id

where r.radiologist\_id = (session\_pid)

and r.test\_date between to\_date(entered\_date1) and to\_date(entered\_date2)

and contains(p.first\_name, (keyword1), 1) > 0

or contains(p.last\_name, (keyword1), 2) > 0

or contains(p.diagnosis, (keyword1), 3) > 0

or contains(p.description, (keyword1), 4) > 0

or contains(p.first\_name, (keyword2), 5) > 0

or contains(p.last\_name, (keyword2), 6) > 0

or contains(p.diagnosis, (keyword2), 7) > 0

or contains(p.description, (keyword2), 8) > 0

order by rank desc

Analysis Module:

This module is entirely done in a scriplet on itself. The module allows you to generate and olap report based on any combination of the following parameters: start and end dates, patient name, test type. The user may (but doesn’t have to) also select a level of hierarchy to display the results. After pressing submit, the module will call itself with the inputed parameters. An sql statement is generated based on the inputs of the user. This is done through a series of if statements to determine the case, the sorting hierarchy is added dynamically based on the inputs of the user.. After this, the sql statement is executed and a table is printed based on the users inputs. All rows with a sum of zero are omitted to reduce clutter. The sql, while all different, follow a *general* pattern:

select (parameters selected), [if hierarchy inputed → trunc(test\_date,(hierarchy))] f

from (relevant tables)

where (all statements to join table) [if date is inputed → test\_date between (from date) and (to date)]

group by cube((parameters selected)