Radiology Information System

CMPUT 391 Database Management Systems (Winter, 2014)

Submitted by: Jessica Surya, Qamar Ali

Department of Computing Science

University of Alberta

Professor: Li-Yan Yuan

March 31, 2014

System Architecture

Major Modules

Login

This module is used by all users to login to the system with proper privileges, and to modify their personal information and/or the password. The graphic user interface to this module is login.jsp. When a username and password is entered in their respective fields, the request is sent to and handled by UserLoginServlet.java which verifies the user's credentials by calling the verifyUser() method in the entities.Db.java class.

The verifyUser() method queries the database with the following prepared statement:

> *SELECT * FROM users WHERE user_name = ? AND password = ?*

The method returns a boolean (true or false) if the user credentials are valid or not. If the login fails, UserLoginServlet.java returns an error message which is displayed in login.jsp. If the login is valid, the user is redirected to index.jsp which displays a search box for accessing the search module.

User Management

This module allows a system administrator to manage (to enter or update) the user information, i.e., the information stored in tables *users, persons, family_doctor*. The graphic user interface to this module is user_management.jsp. It can be accessed through the navigation bar by clicking the "User Management" button. In the module, users with administrator privileges can create or view users, and manage patients tables by clicking their respective links.

Clicking "Create New User" will redirect users to user-form.jsp which allows administrators to add new user accounts by filling in the respective fields required in tables *users* and *persons*. When the "Save" button is clicked, the request is sent and handled by UserRegistrationServet.java. User input is captured, the database is queried with getNextID() to get the next available *person_id*, and then an instance of the User class is created to hold the attributes. getNextID() uses the following the SQL statement pattern to retrieve the next free *person_id*, *record_id*, or *image_id*:

> *SELECT MAX(* fieldID*) FROM* table
>
> where "fieldID" is a *person_id*, *record_id*, or *image_id* and "table" is *persons*, *radiology_record*, or *pacs_images*

The ID number obtained by getNextID() is checked by checkValidity() method to ensure the Oracle unique constraint will not be violated when new entries are added to the database. checkValidity() method uses the following SQL statement pattern to verify the ID number's uniqueness:

> *SELECT COUNT(*) FROM* table *WHERE* fieldID = num
>
> where "fieldID" is a *person_id*, *record_id*, or *image_id*, "table" is *persons*, *radiology_record*, or *pacs_images*, and "num" is the number queried by getNextID()

New user accounts are created with the createUserAccount() method which passes the instance of User to the insertUserAccount() in the Db class. insertUserAccount() enters data for a new user with the following prepared statements:

> *INSERT INTO persons (person_id, first_name, last_name, address, email, phone) VALUES (?, ?, ?, ?, ?, ?)*
>
> *INSERT INTO users (user_name, password, class, person_id, date_registered) VALUES (?, ?, ?, ?, CURRENT_DATE)*

The first statement creates an entry in the *persons* table and the second statement creates an entry in the *users* table.

Clicking “View Users” will redirect users to user-list.jsp which displays a table containing all registered users in the system. This data is obtained by user-list.jsp using the method getUserAccounts() in the Db class getUserAccounts() uses the following prepared statement to obtain the list of registered users:

> *SELECT p.person_id, u.user_name, p.first_name, p.last_name, u.class, p.address, p.email, p.phone FROM persons p, users u WHERE p.person_id = u.person_id ORDER BY last_name DESC*

Users can click “Edit” to change or update the information of a registered user in the user-form.jsp interface. The *person_id* is stored as a hidden value in user-form.jsp which is obtained by UserRegistrationServlet.java creates an instance of the User class with the inputs in the form, assigns the existing *person_id* to the User, then passes the instance to updateUserAccount() in the Db class. updateUserAccount() queries the database with an UPDATE statement in the following prepared statements:

> *UPDATE persons SET first_name = ?, last_name = ?, address = ?, email = ?, phone = ? WHERE person_id = ?*
>
> *UPDATE users SET user_name = ?, password = ?, class = ? WHERE person_id = ?*

The first statement updates the entry in the *persons* table and the second statement updates the entry in the *users* table.

Clicking “Manage Patients” will redirect users to patient_list.jsp which displays tables of patients grouped by their family doctor. Patient rows can be selected with the checkbox which correspond to *patient_id*s which are submitted to UserManagementServlet.java along with the *doctor_id*. UserManagementServlet.java handles the request from patient_list.jsp with removePatients() method which processes the input and calls removePatient() in the Db class. removePatient() deletes individual entries from *family_doctor* table using the following prepared statement:

> *DELETE FROM family_doctor WHERE patient_id = ? AND doctor_id = ?*
>
> *INSERT INTO family_doctor (patient_id, doctor_id, radiologist_id, test_type, prescribing_date, test_date, diagnosis, description) VALUES (?, ?, ?, ?, ?, ?, ?, ?)*

After the transaction is completed, the user is redirected to the same page and the updated state of the database is shown in patient_list.jsp.

Report Generation

This module is used by a system administrator to get the list of all patients with a specified diagnosis for a given time period. The graphic user interface, reports.jsp, contains search fields in which users can specify the search terms are obtained by ReportServlet.java getDiagnosisReports() parses the search terms/dates and calls getDiagnosisReports() in the Db class which uses the following prepared statements to query the database:

> *SELECT * FROM radiology_record r WHERE LOWER(r.diagnosis) LIKE LOWER(?) AND r.test_date BETWEEN ? AND ? ORDER BY r.test_date ASC*

getDiagnosisReports() in Db.java returns an ArrayList of Record instances to ReportServlet.java which then binds User instances for the patients of the radiology record and returns an ArrayList of RadiologyRecord to patient_list.jsp to display in a table below the search fields. This table displays the name, address and phone number of each patient returned by the search, obtained from the User instance, and testing date of the first radiology record that contains the specified diagnosis. getUser() method in Db.java and obtain with existing information on the user in the database and returns an instance of User to encapsulate the data on a user account.

getUser() is an overloaded method, so it can query for users given either the *user_name* string or *person_id* int. The queries used by getUser() are as follows:

> *SELECT p.person_id, u.user_name, p.first_name, p.last_name, u.class, u.password, p.address, p.email, p.phone FROM persons p, users u WHERE p.person_id = u.person_id AND p.person_id = ?*

> *SELECT p.person_id, u.user_name, p.first_name, p.last_name, u.class, u.password, p.address, p.email, p.phone FROM persons p, users u WHERE p.person_id = u.person_id AND u.user_name = ?*

The user is shown an error message if no reports are found with the specific diagnosis entered for a given time period.

Uploading

This module is used by radiologists enter a radiology record, and upload medical images into the radiology record from the user's local file system to the database. The graphic user interface, upload_records.jsp, contains a form where radiologist class users can enter the required data to create a radiology record in the *radiology_record* table. The request from upload_records.jsp is handled by UploadRecordServlet.java which creates an instance of the Record class which is then passed to Db class. A radiology record is then added by insertRadiologyRecord() method which executes the following prepared statement:

> *INSERT INTO radiology_record (record_id, patient_id, doctor_id, radiologist_id, test_type, prescribing_date, test_date, diagnosis, description) VALUES (?, ?, ?, ?, ?, ?, ?, ? ,?)*

getRecords() in Db.java returns an ArrayList of Record instances to UploadRecordServlet.java which then binds User instances for the patients of the radiology record and returns an ArrayList of UploadRecord to patient_list.jsp to display in a table below the search fields. This table displays the record id, patient id,doctor id,radiologist id,test type, prescription date, test date, diagnosis and description for all of the patients. getRecords() uses the following prepared statement to obtain the list of patients with their corresponding radiology records in the database:

*"SELECT * FROM radiology_record WHERE patient_id = ?*

Included images are uploaded with the method insertPacRecord() in the Db class with the following prepared statements:

INSERT INTO pacs_images VALUES(record_id = ? . image_id = ? , empty_blob(), empty_blob(), empty_blob())

*SELECT * FROM pacs_images WHERE record_id = ? AND image_id = ? FOR UPDATE*

getImage() in Db.java retrieves images by the *record_id* it is associated with using the following prepared statementL

*"SELECT * FROM pacs_images WHERE record_id = ? AND image_id = ?*

When the records are uploaded and saved successfully than the page is redirected to upload_record.jsp with the list of all uploaded records together with their related images. If the records or images fail to upload, an error message will be shown to the user.

Search

This module is used by all the registered users to search the database for a list of relevant radiology records and to view medical images with the zoom-in facility. Records can be searched by a list of keywords, and/or specified time periods. The graphic user interface, advanced_search .jsp, contains search fields that is keywords and time periods in which users can specify the search terms that are obtained by SearchServlet.java. searchRecords() parses the search terms/dates and calls getResultsByDateAndKeywords() in the Db class which uses the following query to obtain the search results according to the keywords and dates entered. By default the search results are ordered by Most-Recent-First:

*SELECT score(1)*6 + score(2)*3 + score(3) AS score, record_id FROM radiology_record r, persons p WHERE p.person_id = r.patient_id AND ((r.test_date BETWEEN* 'fDate*' AND* 'tDate*') AND (contains(p.first_name* 'keywords**[i]***', 1) > 0) OR (contains(p.last_name,* 'keywords**[i]***'', 1) > 0) OR (contains(r.diagnosis,* 'keywords**[i]***', 2) > 0) OR (contains(r.description,* 'keywords**[i]***', 3) > 0))* order**;**

where "fDate" and "tDate" are the constraining dates in specified date range, "keywords[i]" is a keyword in the list of keywords which is entered by the user, "order"

getImage() and getRecord() in Db class retrieves images with and associated records. getRecord() uses the following prepared statement to obtain a record by *record_id*.

*SELECT * FROM radiology_record WHERE record_id = ?*

Security

The security module guarantees proper accesses to radiology records. This is done entirely with session variables and verifications when viewing pages.

Data Analysis

This module is used by the system administrator to generate and display an OLAP report for data analysis. A user of this module may choose to display the number of images for each patient, test type, and/or period of time.