ECE356
JAVA/SQL Project

Department of Electrical and Computer Engineering
University of Waterloo

Abstract

The project involves design and implementation of a Management Information System (MIS) using Java and SQL database technologies. Our computing environment provides basic Java development environment, MySQL database engine, and some GUI tools.

1. Purpose

The purpose of this project is to introduce you to database design and writing of database applications. In most 'real' developments, not all resources are spent on the SQL query writing. The formal design of the database (based on user requirements) and the specification, design and implementation of the user visible interactions with the system also consume a major portion of the development effort. Hence this project attempts to provide exposure to all aspects of developing a major database application.

2. Overview

For this project, you will develop a portable and concurrent application in Java and use JDBC (a Java to RDBMS API) to connect to a relational database for data storage and query execution. The application will use the conceptual model of a hospital database environment which stores various patient records. You will be required to design the database schemes and the necessary graphical user interfaces (GUI). You can consider the following description to be a first cut requirement specification for your application.

3. Details

Consider that the members of your group are the MIS administrators for a hospital, and have been given the opportunity of developing a prototype for the next generation information system for the hospital. After discussions with the various hospital groups (Legal, Finance, Doctors, Staff and Patients) that will be using this system, you have the following initial requirements.

3.1 Legal Department

The legal department has indicated the following requirements should be achievable:

- require that all records of every visit, length of visit, every procedure, etc. be available.
- adding information to an existing record should create a new record which contains this additional information, rather than modifying the original.
- financial officers should be able see a summary of a particular doctor, but should not be able to look up patient's names from id numbers.
- doctors should only be able to access visitation records of their own patients.
- doctors can grant/revoke permission to another doctor to look at one of their patient's visitation records.

- staff (with permission granted) should be able to access the visitation records for the doctor(s) they work for only.
- patients should only be able to look at their own records.
- patient information records should contain at least the name, address, phone number, Health Card (assume all patients from Ontario), social insurance number, number of visits, default doctor, indication of current health (e.g. healthy, diagnosis in progress, prescription currently being taken, receiving treatment, scheduled for surgery, transferred to another hospital...) - ONE per patient.
- patient visitation records record the details about a particular visit (diagnosis, prescriptions, scheduling of treatment, etc) - probably one per visit.

3.2 Financial Department:

- ability to monitor any doctor to determine how many patients he/she saw in a given time period, how many times a given patient was seen, what the diagnosis/result was, any drugs prescribed, etc.

3.3 Doctors:

- need the capability to look up patient names from "list of current patients" or all patients. Searchable by several criteria (name, patient #, last visit date, ...).
- past records of a patient available when looking at current record (i.e. no need to search to find past records of patient currently onscreen).
- find past records by date, patient name, diagnosis, keyword in freeform comments, prescriptions given, surgery performed, etc.
- enter a patient visitation record, including prescription(s), diagnosis, etc.
- want to record "freeform" comments (not visible to patients).

3.4 Staff:

- enter appointments for a patient for a given doctor, check for scheduling conflicts with that doctor (manually or automatically).
- delete appointments or reschedule them.
- enter new patients.
- update patient information.
- assign patients to a doctor.
- review patient records for a doctor that is served by this staff person.

3.5 Patients:

- view my past appointments, prescriptions, diagnoses, etc.
- the records should be confidential (need password of some kind).
- update my address, phone number, etc.

Note that you are not being asked to deal with things like scheduling of facilities (X-rays, operating rooms, therapy, etc). You are only being asked to design the patient record portion of the system.

You must keep track of certain other data to allow this system to work: which patients are assigned to a particular doctor, which doctors are allowed to view patient records, a list of legal prescriptions (a hierarchy would probably be useful, since there are many valid prescriptions), outcomes of diagnosis, which doctor a particular visit was with (since patients can switch doctors), etc.

As an example, consider a GUI for doctors. A doctor might have to enter his/her name and password to access the database. Once the password has been verified, then, depending on your design, the doctor could i) enter a patient's name and examine the records for that patient (provided access is permitted), ii) enter a range of dates and see all patients treated between those dates, iii) enter some other criteria and see portions of the patients records, and/or perhaps other operations. Facilities must exist for entering new patient data as well as updating current patient records, etc. The types and ranges of queries must obviously be limited since the goal is not to produce, for example, a general purpose QBE type of interface. The types of SQL queries (joins, selections, projections, etc.) must be to some extent limited and fixed depending on the GUI active at any given time.

4. Deliverables

The first stage of your project will be to determine what relational schemes will be necessary. This will involve identification of various entities and relationships, a formal design of the schemes, identification of appropriate functional dependencies in each scheme, all foreign keys and a good BCNF decomposition of the schemes. As well, the concept of maintaining several views of the database (i.e. for doctors, staff, etc.) should be investigated. A diagram clearly showing the data members of the various tables and the relationships between them (primary keys, foreign keys, etc) will be your first deliverable.

The next stage would be to determine what the user interface to the database should be. For example, what GUIs are needed, what SQL queries must be supported within each GUI, etc. This portion will define the functionality of your Java application using JDBC to access the database and perform the requested queries/updates. Some initial data that could be used in the database will be provided as a starting point. Obviously, the GUIs need not support all possible queries that could be considered.

The final submission will include a demo of your system.

5. Resources

You may use the resources available on sunee or any computing resource you have available (home PC, etc.). Most of the implementation can be developed and debugged on a PC running Windows with the JDK Java.(1.1.3 or later) The JDBC interface will most likely use MySQL. Periodically consult the course web page for links to additional project information concerning JDBC, MySQL and Java.

6. Due Dates

- Java-JDBC Project first deliverable: TBA
- Java-JDBC Project Second deliverable: TBA - 23:59:59 hours. Submit code and documentation using CourseBook.
- Demo Date: TBA - Sign up for demo time using CourseBook.

6. Details of Deliverables

A software system without proper documentation is not very useful. Since the project description basically provides only a software requirements overview, you will need to provide the necessary specifications and design documents. The project require two deliverables as discussed below. These deliverables will be followed by a demo of your system.

First deliverable

This is to be a short 10-15 page document providing the following information for your relational database design:

1. What are the Entity-Relationship (E-R) diagrams for the application? What was the initial design of the database schemas (based on the E-R diagrams)? What constraints (i.e. security, flexibility, ease of use, etc.) were considered? What were the superkeys, functional dependencies identified? What decomposition techniques were employed (BCNF, 3NF, etc.). It would be difficult to arrive at some good schema design without being aware of these points.
2. What is the final database design - what are the schema definitions and tables in your design. For this part to minimize your work you can provide the schema definitions using the SQL create table clause. Here, you identify all attributes, the domain for each attribute, the primary key, any foreign keys and any constraints on the values for any of the attributes. Justify any foreign keys.
3. A brief overview of the types of SQL queries that you are prepared to handle. Obviously you do not want to provide the user with complete flexibility to specify any type of query.
4. Describe how you are planning to implement the Audit Trail (History of changes).
5. Describe how you will handle access rights.
6. What development environment will you be using to implement the project? We supply only Java and MySQL. If you plan to use some other environment (e.g. Microsoft J++, ...) or some other SQL server, then how are you planning to demo your final project? Short paragraph.

You are of course not bound to this design for the duration of the project. Depending on your implementation, you may find that you need minor schema redefinition as you progress. This is fine as long as you document any deviations from this document.

Second Deliverable

You do not need to provide hard copies of your Java source files. All source files, Make files and any required executables will be copied at the time of the demo. Your written documentation must be submitted at the time of your demo and should include the following:

1. Outline (briefly) any changes to your first deliverable document - if any.
2. Briefly describe the GUI for your system using a user-interface flow chart.
3. Briefly describe your system's software qualities with respect to portability and concurrency.
4. How did you test your system - any tools used, test drivers, etc.? Short one paragraph.
5. Highlight any part of your overall project that you deem to be significant and perhaps novel.

The above requirements should only serve as a guide as to what is required. You do not need to present them in the order stated - as long as your document contains equivalent information. The first three points would typically comprise the majority of the document. We do not require a long document. Much of the information can be obtained directly from your source files (i.e. class definitions etc.). It is difficult to provide a page count as a goal, but the above information typically should not exceed 15-20 pages - hopefully less. Ideally, it should be written in an incremental fashion as you develop the project to avoid attempting to produce the document in 2 days.

System Demo
We will experiment with the system for some time, entering some values that don't exist for patient names, update which doctor is treating which patient, change a patient's address, etc. During the demo:
1) Your system should not crash.
2) You should be able to justify why we can't do something if we think we should be able to - based on the above requirements.

Demo Signup
Project Demo Date: TBA
Demo Duration: 30 minutes
Use CourseBook to sign up for demo times.

IMPORTANT: Phase 2 deliverable/report is due DURING the demo.