Homework Assignment 4
Java Database Connectivity
ECE 155B – Spring 2012
Demos – Wednesday, May 23

In this assignment you will modify your Healthcare System that supports patients, doctors and pharmacists. You will store all information at the doctor in an SQL database, rather than in an XML file.

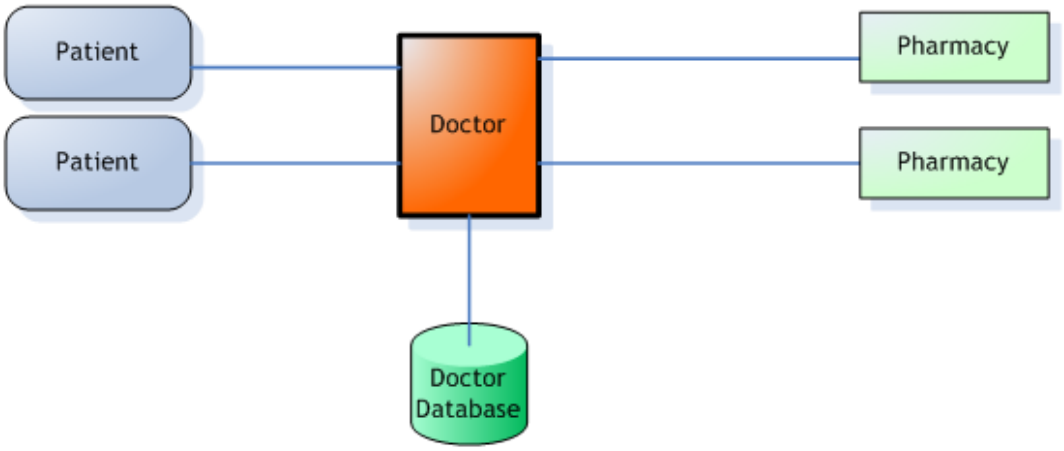
For this assignment, your Healthcare System will be implemented as a three-tier distributed application, using Java Database Connectivity (JDBC) and the MySQL database system. You will be provided MySql accounts that you will use for this project.

Please contact the TA for a MySql account, username and password for the MySql account. Provide your desired username/password in your email to the TA.

Requirements

This assignment requires that you modify your Healthcare System to run as a three-tier distributed application with a database in the back-end (third) tier at the doctor. The Doctor application in the middle tier will use the JDBC APIs to encapsulate the SQL commands to the MySQL database system in the third tier.

The Patient in the first tier will have a Graphical User Interfaces (GUI) and provide the same functionality as in the previous project. The Pharmacy application will also be the same as in the previous project (unless you choose to use a database instead of an XML file at the pharmacist). The Doctor application will insert, update and retrieve data from the database in the third tier at the doctor.



Your Database in the third tier will consist of the following tables:

- **Doctor**
- **Patient**
- **Doctor Patient** *(*relational table, referencing both Patient and Doctor)
- **Appointment** (relational table referencing both Patient and Doctor)
- **Prescription** (relational table referencing Appointment)
- **Notes** (relational table referencing Appointment).

- ***Optional [10 points]*:**
  - Provide a database for the Pharmacy with (at least) the following table:
    - **Prescription**

Example Doctor, Patient and Appointment table definitions.

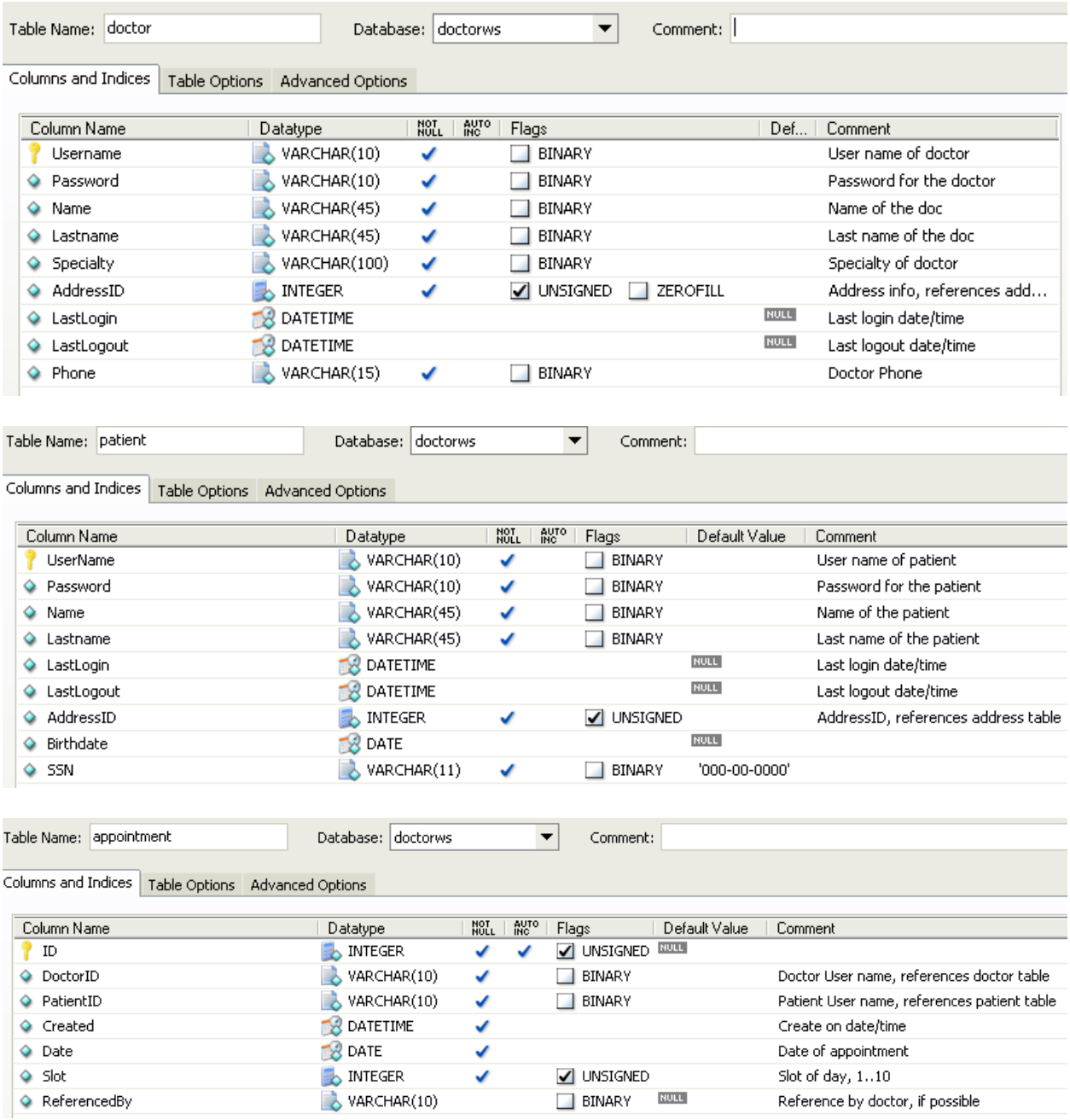


Table Name: doctor Database: doctorws Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Def...	Comment
Username	VARCHAR(10)	✓		BINARY		User name of doctor
Password	VARCHAR(10)	✓		BINARY		Password for the doctor
Name	VARCHAR(45)	✓		BINARY		Name of the doc
Lastname	VARCHAR(45)	✓		BINARY		Last name of the doc
Specialty	VARCHAR(100)	✓		BINARY		Specialty of doctor
AddressID	INTEGER	✓		UNSIGNED ZEROFILL		Address info, references add...
LastLogin	DATETIME				NULL	Last login date/time
LastLogout	DATETIME				NULL	Last logout date/time
Phone	VARCHAR(15)	✓		BINARY		Doctor Phone

Table Name: patient Database: doctorws Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
UserName	VARCHAR(10)	✓		BINARY		User name of patient
Password	VARCHAR(10)	✓		BINARY		Password for the patient
Name	VARCHAR(45)	✓		BINARY		Name of the patient
Lastname	VARCHAR(45)	✓		BINARY		Last name of the patient
LastLogin	DATETIME				NULL	Last login date/time
LastLogout	DATETIME				NULL	Last logout date/time
AddressID	INTEGER	✓		UNSIGNED		AddressID, references address table
Birthdate	DATE				NULL	
SSN	VARCHAR(11)	✓		BINARY	'000-00-0000'	

Table Name: appointment Database: doctorws Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
ID	INTEGER	✓	✓	UNSIGNED	NULL	
DoctorID	VARCHAR(10)	✓		BINARY		Doctor User name, references doctor table
PatientID	VARCHAR(10)	✓		BINARY		Patient User name, references patient table
Created	DATETIME	✓				Create on date/time
Date	DATE	✓				Date of appointment
Slot	INTEGER	✓		UNSIGNED		Slot of day, 1..10
ReferencedBy	VARCHAR(10)			BINARY	NULL	Reference by doctor, if possible

You can use the auto increment feature for assigning IDs. Note that, in the Appointment table, the DoctorID references the Doctor table (username) and similarly for the PatientID.

Your Doctor application in the middle tier will communicate with the database in the third tier to store and retrieve information.

In order to process MySQL commands you will need to provide a suitable driver for the Java SQL libraries. You will be provided the necessary JDBC driver. You can then use the Java SQL libraries by importing:

import java.sql.*;

Check out the template available at the class Web site and work with that file.

You might find it useful to use the following applications to manipulate and query the MySql database; links to these applications are also available at the class Web site.

- *MySQL Administrator*
- *MySQL Query Browser*

Design

In database applications, it is good practice to use a layered design. That is, have a class that is totally dedicated to querying and another class that deals with updates (with the help of the other class). For example:

Doctor object:

boolean ***AddPatient*** (Patient)	{... Using **DoctorDB** object, call methods}
boolean ***DelPatient*** (Patient)	{... Using **DoctorDB** object, call methods}
boolean ***UpdatePatient*** (Patient)	{... Using **DoctorDB** object, call methods}
Vector ***ListPatients***()	{... Using **DoctorDB** object, call methods}
Vector ***GetAppointments***()	{... Using **DoctorDB** object, call methods}

DoctorDB object:

boolean ***AddPatient*** (Patient)
boolean ***DelPatient*** (Patient)
boolean ***UpdatePatient*** (Patient)
Vector<Patient> ***ListPatients***()
Vector<Appointment> ***GetAppointments***()

Here the server does not need to know how queries are sent to the database; the implementation details are in the DoctorDB class. When there is a change in the database platform being used (for example, Oracle or DB2 instead of MySQL), the change will affect only the DoctorDB class. Moreover, if you change the design for the server (for example, so that it uses Java Server Pages (JSPs) instead of a Java application), you can still use the DoctorDB class. Again the server does not need to know how to interact with the database, it can achieve the interactions with the help of the DoctorDB class.

If you implement project 4 like this, you will not have much coding related to the database in project 5. For this reason, it is very important to understand and use a layered design.

On the class Web site for this project, you will find a template for the DoctorDB class.

Report

In addition to the program, you will need to submit a small report. The report should be clear, well organized, and concise. It needs to include the following:

1) Database table structures
2) All MYSQL commands *[Table Create, Insert, Delete, Update, Query]*
3) Details of how your implementation works
4) A diagram of the structure of your database, its tables, records and fields.

Submission

You will submit a compressed file *(rar)*. You will need to type "*ant submission*" in order to create your submission file. If there is anything missing in the submission file, edit the *build.xml* file to include the missing information. Make sure you modify the *User* property in the *build.xml* file. The *ant* call will create submission file with the username you provide.

The compressed file should include a folder with all project content necessary to build/run the program (.java, README, Report). The projects are to be sent to ece155b.s12@gmail.com with subject line "Project 04: FirstName LastName".

Grading

Please use the grading sheet as provided online for more grading information.