

CS3431: Project Description
B-Term, 2013
Building a Database Application
Phase 3: Complete Hospital Application

Release Date: Dec. 2, 2013.

Due Date: Dec. 14, 2013 (11:59PM)

Teams: The project is done in teams of two.

Description:

In this phase you will complete the design of the *Hospital System Database* by ensuring that the application requirements and constraints are all met.

Part 1 (50 Points): Database Views (10 Points each requirement)

- Create a database view named ***CriticalCases*** that selects the patients who have been admitted to Intensive Care Unit (ICU) at least 2 times. The view columns should be: *Patient_SSN, firstName, lastName, numberOfAdmissionsToICU*.
 - ***Hint:*** ICU is a service that is stored in table 'RoomService'
- Create a database view named ***DoctorsLoad*** that reports for each doctor whether this doctor has an overload or not. A doctor has an overload if (s)he has more than 10 distinct admission cases, otherwise the doctor has an underload. Notice that if a doctor examined a patient multiple times in the same admission, that still counts as one admission case. The view columns should be: *DoctorID, gender, load*.
 - The *load* column should have either of these two values 'Overloaded', or 'Underloaded' according to the definition above.
- Use the views created above (you may need the original tables as well) to report the critical-case patients with number of admissions to ICU greater than 4.
- Use the views created above (you may need the original tables as well) to report the female overloaded doctors. You should report the doctor ID, firstName, and lastName.
- Use the views created above (you may need the original tables as well) to report the comments inserted by underloaded doctors when examining critical-case patients. You should report the doctor Id, patient SSN, and the comment.

Part 2 (70 Points): Database Triggers [10 Points each requirement]

Given the following requirements, create one or more triggers that ensure that these requirements are always satisfied. Think of the three events *Insert*, *Update*, and *Delete* because you may need triggers on one or more of these events to meet the requirements below.

Hint: *You are allowed to combine several requirements into one trigger*

- Any room in the hospital cannot offer more than three services.
- The insurance payment should be calculated automatically as 70% of the total payment. If the total payment changes then the insurance amount should also change.
 - If in your DB you store the insurance payment as a percent, then it should be always set to 70%.
- Ensure that regular employees (with rank 0) must have their supervisors as division managers (with rank 1). Also each regular employee must have a supervisor at all times.
- Similarly, division managers (with rank 1) must have their supervisors as general managers (with rank 2). Division managers must have supervisors at all times.
- When a patient is admitted to ICU room on date *D*, the futureVisitDate should be automatically set to 3 months after that date, i.e., *D* + 3 months. The futureVisitDate may be manually changed later, but when the ICU admission happens, the date should be set to default as mentioned above.
- If an equipment of type 'MRI', then the purchase year must be not null and after 2005.
- When a patient is admitted to the hospital, i.e., a new record is inserted into the *Admission* table; the system should print out the names of the doctors who previously examined this patient (if any).
 - Hint: Use function `dbms_output.put_line()` also make sure to run the following line so you can see the output lines.

```
Sql> set serveroutput on;
```

Hint: *If you faced with error "Table xxxx is mutating...", then try to change either the trigger timing "Before \leftrightarrow After" and/or the trigger level "Row \leftrightarrow Statement levels".*

Part 3 (50 Points): Access DB using JDBC [10 Points each requirement]

You are required to write a java program that access the database, performs some insertions and updates, executes some queries, and prints results on the screen.

Your program will be the interface to perform some simple functionality over the database. Your database should already contain data.

You are required to do the following:

Assume your program is called "Reporting.java"

(0) When your program is executed without any additional arguments, e.g.,

> java Reporting

Then the program should output the following options, and then terminates:

- 1- Report Patients Basic Information
- 2- Report Doctors Basic Information
- 3- Report Admissions Information
- 4- Update Admissions Payment

(1) When the program is executed with an argument 1 as follows:

> java Reporting 1

Then, we are now in the "Reporting Patients Basic Info." mode. The program should print out the following line:

Enter Patient SSN: <and wait for user's input>

When the user enters a SSN, the program should execute a query over the patient table and prints on the screen the patient's information as follows:

**Patient SSN: ...
Patient First Name: ...
Patient Last Name: ...
Patient Address: ...**

Then the program terminates.

(2) When the program is executed with an argument 2 as follows:

> java Reporting 2

Then, we are now in the “Reporting Doctors Basic Info.” mode. The program should print out the following line:

```
Enter Doctor ID: <and wait for user's input>
```

When the user enters an ID, the program should query the doctor table and print on the screen the following information, and then terminates:

```
Doctor ID: ...  
Doctor First Name: ...  
Doctor Last Name: ...  
Doctor Gender: ...
```

(3) When the program is executed with an argument 3 as follows:

> java Reporting 3

Then, we are now in the “Reporting Admissions Info.” mode. The program should print out the following line:

```
Enter Admission Number: <and wait for user's input>
```

When the user enters a number, the program should query the admission table and print on the screen the following information, and then terminates:

```
Admission Number: ....  
Patient SSN: ...  
Doctor ID: ...  
Admission date (start date): .....  
Total Payment: ...  
Rooms:  
  RoomNum: ...  FromDate:....   ToDate:.....  
  RoomNum:...   FromDate:....   ToDate: ...  
  ...
```

Notice that we need to print the room(s) that the patient has stayed in during this admission.

(4) When the program is executed with an argument 3 as follows:

> java Reporting 4

Then, we are now in the “Updating Admission Payment” mode. The program should print out the following line:

<p>Enter Admission Number: <and wait for user’s input> Enter the new total payment: <and wait for user’s input></p>

Then your program should update the total payment value for the specified admission number in the database.

//Now if you execute option 3 again, you should get the new payment value.

Grading:

The maximum grade is 170 Points. Late submissions follow the rules stated on the website.

Deliverables:

Each team should deliver three files as follows:

- 1) Text file (.sql) that contains all SQL commands from Part1 and Part2 above. Also ensure that the file includes the creation of all needed tables (copy it from the submission of Phase 2). The file must be executable from SQL using command:
SQL > @<fileName>

The file must run correctly, creates all tables, and then the views and triggers. If the file's syntax was not correct (and the file did not run), you will loose 20 points in addition to any other deductions.

- 2) A java file containing Part 3. Submit the code (.java) file. The TA will compile your file on a CCC machine. So make sure your file complies on these machines.
- 3) Any comments or assumptions that you have, you can write them to a separate .doc or .pdf file.

Important: In this file include for each of the requirements in Part 2, what trigger(s) did you create to enforce this requirement. You need to state the event type (Before or After), the operation (Insert, Update, or Delete), and the granularity (For each row, or For each statement) and on which table. No code should be included here.

- 4) Put the three files in a single zip file that will be submitted as below.

Submission (Each team give one submission):

- Submit electronically by the due date via blackboard.wpi.edu website. Make sure your three files (the text file .sql, the java file .java, and your report .doc or .pdf) are zipped, and you upload one file.
- Each team submits one copy (from either of the team members).
- ***No hard copy submissions***