CS430/630 – Homework 4
Released Nov 17, Due Dec 4

Instructions: The homework is due on Fri, Dec 4th, 23:59 hrs. Submission is done in electronic format, create a folder HW4 in your course directory and place source code and compilation scripts there. Have a separate folder for each question (Q1 and Q2). Recall to set the proper permissions for files!

Question 1 (25 points) – Embedded SQL in C

Consider a database schema with three relations:

- **Parts** (pid:integer, pname:string, year:integer, price:integer)
- **Suppliers** (sid:integer, sname: string, state:string, zipcode:string)
- **Orders** (pid:integer, sid:integer, quantity:integer)

The description is as follows: a factory keeps a database with parts that it uses, the suppliers of those parts, and purchase orders. Each part is uniquely identified by pid. Each part has a string description pname, year of fabrication and price per unit. Parts are provided by suppliers, and each supplier is uniquely identified by sid. Each supplier has a name sname, and it operates at the given state and zipcode. The company has a number of orders for parts, and each order contains the pid of the part ordered, the sid of the supplier from which the part is ordered, and the quantity ordered. You are allowed flexibility on the exact attribute types you use for your schema, as long as they reasonably match the specification above (e.g., in terms of number types, string types). Include the schema definition in your submission in a file called `schema.sql`.

Write Embedded SQL programs that perform the following tasks:

(a) Prompt the user for a supplier ID and print the specifications (part number, part name, year and price for all parts ordered from that supplier. Attributes of each part must be printed on a new line (10p).

(b) Prompt the user for the value of a zipcode and a part ID. Output in tabular format (one per line) the supplier ID, supplier name and the dollar amount of the order for the given part for each supplier in the given zipcode that supplied that part. Each line should start with a unique entry number (begin numbering with 1, continue with 2,3, ... etc) which is NOT part of any schema, but uniquely identifies the output line. If there are no matching records, print the message “No Records Found” on the screen. If the result is not empty, prompt the user for an entry number and delete the corresponding record from the Orders table (15p).

Create two separate source files for (a) and (b), and name them Q1a.pc and Q1b.pc.

Question 2 (25 points) - JDBC

You must create a JDBC application for managing the course enrollment at a university. The schema is as follows:

- **Students** (sid:integer, sname:string)
- **Courses** (cid:integer, cname:string, credits:integer)
- **Enrolled** (sid:integer, cid:integer)

The Students relation stores data about students: a unique student id and name. Each course has a course id, name and number of credits). The Enrolled relation stores what courses are taken by which students.
You must create the above schema definition in your submission in a file called `schema.sql`. You are allowed flexibility on the exact attribute types you use for your schema, as long as they reasonably match the specification above (e.g., in terms of number types, string types). Also, you have to create a Java JDBC-based application run by students, with name Student.java. The application must have a command-line interface menu that allows the user to select one option as below. Once that menu function is completed, the program must return to the main menu. For each menu option, you are allowed (and even recommended, if needed) to have multiple steps (or “screens”) to complete the tasks. You will use the DBS2 Oracle instance as DBMS.

**Student Menu:**

Application starts by requesting student’s ID; no authentication is necessary, and the remaining session assumes that student ID is active. If (-1) is introduced, a new student is created, and the user is prompted for all necessary information. The main menu is the following:

L – List: lists all records in the course table

E – Enroll: enrolls the active student in a course; user is prompted for course ID; check for conflicts, i.e., student cannot enroll twice in same course

W – Withdraw: deletes an entry in the Enrolled table corresponding to active student; student is prompted for course ID to be withdrawn from

S – Search: search course based on substring of course name which is given by user; list all matching courses

M – My Classes: lists all classes enrolled in by the active student.

X – Exit: exit application