CSc 460 — Database Design
Fall 2021 (McCann)

http://u.arizona.edu/~mccann/classes/460

Program #4: Database Design and Implementation

Due Dates:

Team Members:	November 22nd, 2021, at the beginning of class
Draft E–R Diagram:	November 29th, 2021, at the beginning of class
Final Product:	December 6th, 2021, at the beginning of class

Designed by *Sourav Mangla and Justin Do*

Overview: In this assignment, you will build a database–driven web information management system from ground up. We will give you an application domain to work on, your goal is to design the underlying database and define the application functionalities you will provide with the database, and implement this application using Oracle within a text–based JDBC program.

Assignment: In this assignment you are to implement a two–tier client–server architecture.

1. **Database Back-End**, which runs the Oracle DBMS on `aloe.cs.arizona.edu`. Your job is to design the database relational schema, create tables and populate your tables with some initial data. We are requiring that you create an E–R diagram, analyze the FDs of each table and apply table normalization techniques to your schema to justify that your schema satisfies 3NF, and, if possible, BCNF.
2. **JDBC Front–End**, which is the client's user interface. You need to design a text–based application that appropriately handles all the required functionalities. Your client application will run on lectura.

Application Domain: The problem description for the project is as follows:

When people have basic licensing and identification needs, such as driver's licenses, their local Department of Motor Vehicles (DMV) office is the place to go. Your task is to design a database and associated manipulation/querying application for a DMV.

The DMV provides multiple services like Driver License, State Id, Vehicle Registration, Permit, etc. To get a service, you need to book an appointment with the DMV. Appointments can be multiple types, as a new license or renewing the expired license. Users can have multiple appointments. Appointments for a single person cannot overlap in time.

Of course, the DMV has employees to handle patron's needs. Appointments are assumed to always lead to some sort of transaction in which the employee that handles the transaction will be associated with it. Every transaction has logs that hold the service amount paid by the user. For logs, we are only focusing on the service fee. Charges for the available services are as follows:

1. Permit: $7
2. Licence: $25
3. Vehicle Registration: $100
4. State ID: $12

A user can have just one license and just one state ID, but a user can have multiple vehicle registrations if they have more than one vehicle (one registration for each). The DMV needs to maintain a date associated with each ID because the ID is valid for a limited time.

(Continued...)

We are focusing on four services in this DMV: license, vehicle registration, state ID, and permit. Each service has a table detailing the documents. An ID's issued date is equal to the appointment date if an appointment is successful. The expiry date is as follows:

1. Permit: One year from the date of issue
2. Licence: 12 years from the date of issue
3. Vehicle Registration: One year from the date of issue
4. State ID: 20 years from the date of issue

Each employee belongs to one department, each service is handled by a separate department; thus, there are four departments. Each employee has a designation or job type such as supervisor, front desk, security, etc. A job has a salary, job title, job id, etc. Every employee has basic details, such as name, department, etc.

This description does not describe every detail. These are the essentials; we expect that your team will create logical and conceptual designs that incorporate all of these features, at minimum. You are free to add additional details that you feel are appropriate.

For each table you create, you need to populate a reasonable number of tuples in order to test your queries adequately. Some data basics are provided in the application domain description; the rest are left for you to determine, based on your needs. (What is 'reasonable' is difficult to define; a few dozen tuples per relation certainly would be; just a handful per relation may not provide sufficient variety.)

We realize that you are not an expert in this domain, but you have dealt with similar organizations in your life. Hopefully, you have enough experience that this problem description makes sense. If you have questions, please ask, and the TAs will help you clear things up.

Required functionalities: Within the framework provided above, your system is expected to perform examples of the following operations:

1. *Record insertion*: Your application should support inserting a new data record via a JDBC interface.
2. *Record deletion*: Your application should support deleting an existing data record via a JDBC interface.
3. *Record update*: Your application should support updating an existing data record via a JDBC interface.
4. *Queries*: Your application should support querying your database via a JDBC interface for the problem description given above. You are required to implement the three provided queries as well as at least one query of your own design. Details are provided below.

Specifically, the JDBC application's interface should enable users to:

1. Add, update or delete a client, employee, appointment, and service. When adding, updating, the user is allowed to update everything except the ID. When deleting, the entire row needs to be deleted.While inserting or updating an appointment, appointment time must not overlap with any existing appointment.
2. Add or update different service details of a client. When adding licence details: id, issued date and expiry date cannot be empty. Also, the issued date cannot be ahead of the expiry date. A user can not have multiple licences.

(Continued...)

Here are the queries that your application is to be able to answer:

1. Write a query that displays the user details whose Ids will expire given a date in format MM/DD/YYYY (given by the user). The result should display the user id, user name, id issued date, id expiry date and type of id.
2. To get a licence, the user has to clear some tests, and there are other conditions to other types of ids. So, not every appointment is successful. Your work is to write a query for the previous month count every type of appointment and check how many of them got successful their IDs. For example: in October, DMV got 50 appointments for driving licence and 30 appointments for vehicle registration but 40 got approved for the licence and 25 for registration. Assume all work is done in one day.
3. Write a query that displays the collected fee amount for every department for a given month in the format MM/YYYY (given by the user). The result should display the amount and department information and sort the result on amount in descending order.
4. One additional non–trivial query of your own design, with these restrictions: The question must use more than two relations and must be constructed using at least one piece of information gathered from the user.

Working in Groups: In industry, such a project is usually the work of multiple developers, because it involves several different components. Good communication is a vital key to the success of the project. This assignment provides an opportunity for just this sort of teamwork. At the same time, we realize that upper–division classes often require a team project, leaving some students hoping for the opportunity to work by themselves on such a project for a change. Therefore, we are accepting team sizes of between one and four members (inclusive), but definitely recommend teaming up if possible, due to the scope of the project and the stresses students usually experience at this time of the semester.

Early on, you will need to agree on a reasonable workload distribution plan for your team, with well–defined responsibilities, deliverables, and expected completion dates. Such a plan will minimize conflicts and debugging effort in the actual implementation.

Late days: Late days can be used on this assignment, but only on the third due date. How many a team has to use is determined as follows: Team members total their remaining late days, and divide by the number of members in the team (integer division), producing the number of late days the team has available, **to a max of two days late**. (Why? The TAs need to get grading done soon after the due date, you need time to study for your final exams, and the department has a rule about assignments needing to be due before the start of finals.)

For example, a team whose three members have 1, 1, and 3 late days remaining have $\lfloor \frac{1+1+3}{3} \rfloor = 1$ late day to use, if needed.

(Continued...)

Hand In: Here are the 'deliverables' for each of the assignment's three due dates:

1. *Team Composition*: By the first due date (see the top of the front page of this handout), one member of your team must create a PRIVATE post on Piazza using the "program4" folder with the names and NetIDs of the members of your team. Failure to do so by the start of class on this date will cost your team the corresponding points listed in the Grading Criteria section (below).

2. *E–R Diagram*: As stated in the Assignment section, your team will need to create an E–R diagram that describes your database design. Before the second due date, your team will need to prepare a draft of your E–R diagram **and** a member of your team will need to submit it through `turnin` to the `cs460p4` folder. The purpose of this requirement is to allow the TAs to review your schema and make suggestions for improvement. The sooner you create your design and discuss it with the TAs, the more time you will have to refine your final E–R diagram. If TAs need further explanation of your E–R Diagram, they'll send out an email to make an appointment to have an additional meeting.

3. *Final Product*: On or before the third due date, a member of your team must submit a `.tar` file of your well–documented application program file(s) via turnin to the folder `cs460p4`. The tar file should contain all of the following:
   (a) The source code for your application.
   (b) A PDF file called "design.pdf" containing the following sections in this order:
      i. *Conceptual database design*: Your final E–R diagram along with your design rationale and any necessary high–level text description of the data model (e.g., constraints, or anything you were not able to show in the E–R diagram but that is necessary to help people understand your database design).
      ii. *Logical database design*: The conversion of your E–R schema into a relational database schema. Provide the schemas of the tables resulting from this step.
      iii. *Normalization analysis*: For each of your entity sets (tables), provide all of the FDs of the table and justify why your the table adheres to 3NF / BCNF.
      iv. *Query description*: Describe your self–designed query. Specifically, what question is it answering, and what is the utility of including such a query in the system?
   (c) A `ReadMe.txt` describing:
      i. Compilation and execution instructions, to enable the TAs to execute your application and exercise the required functionalities.
      ii. The workload distribution among team members (that is, which people were responsible for which parts of the project).

   In addition, each team must schedule a time slot (∼15 – 20 minutes) to meet with a TA, demonstrate your system, and perhaps answer some questions about it. Closer to the first due date, we will let you know how to sign up.

(Continued...)

Grading Criteria: Total: 100 points

1. Team Composition (1st due date): 5
2. Complete E–R Diagram Draft (2nd due date): 20
3. Final Submission (3rd due date): 75
   (a) Coding / Implementation: 55
      - Documentation 15
      - Style and organization 10
      - Record insertion: 5
      - Record deletion: 5
      - Record update: 10
      - Record query: 10

   (b) Database design: 20
      - Final E–R diagram: 10
      - Normalization analysis: 10

Grading Notes:

1. Unless we receive verifiable complaints about inadequate contributions, each member of a team will receive the same score on this assignment.
2. We won't put much weight at all on the appearance of the text application; concern yourselves with the application's functionality instead. The main point of the assignment is the DB design.