CSc 460 — Database Design
Spring 2023 (McCann)

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

Program #4: Database Design and Implementation

Due Dates:

Team Members:	April 18th, 2023, at the beginning of class
Draft E–R Diagram:	April 25th, 2023, at the beginning of class
Final Product:	May 2nd, 2023, at the beginning of class

Designed by *Tanner Finken and Aayush Pinto*

Overview: In this assignment, you will build a database–driven 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:

> With all the traveling that people are doing nowadays. Aayush and Tanner think it is a good idea to invest in building a new luxury hotel here in Tucson called "Motel 460." The difference between a hotel and motel is that motels are generally located along major highways. While some are relatively isolated, others are located close to other businesses that cater to travelers, including restaurants, service stations, and convenience stores. Hotels, on the other hand, are usually located in city centers or near airports. The owners wanted to stand out, so Motel 460 will be a brand new hospitality service provider located in the middle of the city that will operate a single hotel with multiple restaurants and amenities. The owners will need a database that keeps track of operations at their motel. Your task is to design a database and an associated manipulation/querying application for Motel 460.
>
> The hotel will allow customers to book a room for any available dates. The customers will be given different categories based on what or how they are booking. College students get discounts on bookings and services provided. If the customer is signed up as a subscribed member of "Club 460," they will get a variety of perks including discounts, free use of amenities, and the ability to earn points which can be redeemed for free stays. Certain credit cards will also offer discounts, and travel rewards. The hotel offers a variety of amenities. The initial amenities for the hotel include restaurants, bar, parking, gym, and spa. However, new amenities may be added later.

(Continued...)

The hotel would also like to keep track of how the business is running with respect to its employees and paid services. Thus, the hotel will need a system that keeps track of when employees are scheduled to work and what duties they will be performing. Finally, when a customer is finished with their stay, they will be billed for the room they were in, any amenities they used (yes, they all cost extra), and monetary tips they'd like to include. While checking out, the customer will also be able to provide a rating for all amenities used, because the owners would like to know which amenities are most liked, so they can add more like them.

Note: Your E–R diagram must include at least one E–E–R concept.

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 four 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 customer, employee, bookings, ratings, etc. Update different service details of an employee. When updating, the user is allowed to update everything except the IDs of the customers, employees, bookings, ratings, etc. When deleting any record, the entire record needs to be deleted.
2. While inserting or updating a booking, it must not overlap with existing bookings. You also need to take care of any other obvious logical inconsistencies with using amenities or forming a schedule for employees.

(Continued...)

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

1. Print a current bill (total $) for a customer for their stay and all unpaid amenities.
2. Given a certain date, output the customers that are currently staying at the hotel along with their room numbers. Order by room numbers and group by category of customer.
3. Print the schedule of staff given a week (input the start date of the week by the user). A schedule contains the list of staff members working that week and a staff member's working hours (start and stop times).
4. Print the average ratings of different amenities recorded within the range of two dates(input by the user) and sort them in descending order.
5. A query of your choice, subject to 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. Therefore, we are requiring team sizes of between two and four members (inclusive). More would be unwieldy, and fewer wouldn't be much of a team!

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.

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.

(Continued...)

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 ($\sim$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.

Grading Criteria: Total: 100 points

1. Team Composition (1st due date): 5
2. Complete E–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–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.