CSc 460 — Database Design
Fall 2023 (McCann)

http://www2.cs.arizona.edu/classes/cs460/fall23/

Program #4: Gym Management Database Design and Implementation

Due Dates:

Team Members:	November 21st, 2023, at the beginning of class
Draft E–R Diagram:	November 28th, 2023, at the beginning of class
Final Product:	December 5th, 2023, at the beginning of class

Designed by Zhenyu Qi and Daniel Bazmandeh

Overview: In this assignment, you will build a database–driven information management system for a fitness center. 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 for the fitness center, create tables, and populate them with initial data. Create an E—R diagram, analyze the FDs of each table, and apply table normalization techniques to ensure 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:

> As people place increasing emphasis on health, fitness centers have become incredibly popular. Recognizing this trend, we have decided to launch a new gym called GYM 460. Our objective is to design a database for GYM 460 and implement relevant query applications to streamline its operations.
>
> GYM 460 is structured with key components crucial for its operation. Under Member Information, we choose a reasonable collection of basic information to store for each member such as name, and telephone number. Members are presented with options to purchase various course packages, which include a junior package with Strength 001 and Yoga 001, a senior package with Strength 002 and Yoga 002, and a strength package combining Strength 001 and Strength 002. The fitness center offers different membership levels, with each level having a minimum spending requirement and a corresponding discount rate; for instance, reaching a total consumption of $1,000 makes one a Gold Member, enabling them to enjoy a 20% discount on future purchases, whereas reaching $500 upgrades a member to Diamond Member status, with a subsequent 10% discount.
>
> The center employs several professional fitness trainers, each with a name and telephone number. Trainers are tasked with teaching multiple courses but must adhere to a schedule that prevents overlapping times; for example, Trainer A cannot teach both from 1–3pm and 2–4pm on Monday, although they can teach consecutively from 1–3pm to 3–5pm. These trainers are akin to tireless superheroes which means they can teach all day!
>
> Our course offerings consist of various fitness classes like Strength 001, Strength 002, Yoga 001, and Yoga 002. Each class is assigned a specific trainer (only one) and has a set weekly class time, duration, start date, end date, and a maximum number of participants. Courses cannot be bought individually but are available through the package deals given above.

(Continued...)

The center is stocked with sports and fitness items such as footballs, basketballs, etc., which members can borrow. We maintain a system to record the check–out time, return time, and quantity of equipment borrowed.

Lastly, the transaction records are meticulously kept for every user interaction, recording details like recharge records, course purchases, and the specifics of each transaction, including the amount, date, and transaction type.

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 likely 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 or delete a member: Adding a member begins with inputting the member's basic information and creating a new member entry. After this initial step, the system presents the contents and price of each available course package for selection. Once a member selects a package, related records are updated to reflect this choice.

   The deletion of a member involves several checks. First, the system verifies if the member has any unreturned equipment. If such equipment exists, it is marked as lost, and the available equipment quantity is updated. Next, the member's transaction history is examined for any unpaid amounts. If unpaid balances are found, they are printed out, and the system prevents the deletion of the member's account. Additionally, the system checks if the member is actively participating in any courses. If so, their course participation records are deleted, and the available spots in the course are updated accordingly.

2. Add or delete a course: Adding a course requires inputting the course's basic information into the system and creating a new course entry.

   Conversely, deleting a course is more complex, especially if the course has not ended and there are members enrolled. In such cases, the names and phone numbers of these members are printed to aid the administrator in notifying them before the course and its corresponding enrollments are deleted. After notifying the members, they can be deleted.

(Continued...)

3. Add, update, or delete a course package: When adding a course package, the system lists all available courses that have not yet ended, allowing the admin to select which to include.

   Updating a course package is done in a way that does not affect members already enrolled in courses; the admin is presented with a list of all course packages and can select one to edit. Within the chosen package, the admin can update, add, or remove a course as needed.

   Deleting a course package also involves a safeguard that ensures the deletion does not impact members' enrolled courses. The admin is provided a list of all course packages and can select one for deletion.

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

1. List all members' names and phone numbers who now have a negative balance (that is, have fees that are not paid off).
2. Check and see a member's class schedule for November.
3. Check and see all trainers' working hours for December.
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. Therefore, we are accepting team sizes of between two and four members (inclusive) 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. For a few basic suggestions on team management, see:

https://www.saintleo.edu/about/stories/blog/7-tips-to-more-effectively-work-on-group-projects

Late days: Late days may 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.

The Clients: Zhenyu and Danial (our TAs) are serving as your clients for this design and implementation assignment. If you have questions about the details of the assignment, direct them to the clients, not to Prof. McCann.

(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 fill out this form with the names and NetIDs of the members of your team:

   https://forms.gle/1xdSvg9kCnnHyrPm9

   (We'll also post it on Piazza.) Failure to submit the form by the start of class on this date will cost your team the corresponding points listed in the Grading Criteria section (see 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 for a meeting.

3. *Final Product*: On or before the third due date, a member of your team must submit a `prog4.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` file describing, at minimum:

      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 final due date, we will let you know the available demo time slots and how to sign up.

(Continued...)

Grading Criteria: Total: 100 points

1. Team Composition (1st due date): 5
2. Draft of the Complete E–R Diagram (2nd due date): 20
3. Final Submission (3rd due date): 75
   (a) Coding / Implementation: 55
      - Documentation 15
      - Style and organization 10
      - Record insertion: 8
      - Record deletion: 8
      - Record update: 4
      - Record query: 10
   (b) Database design: 20
      - Final E–R diagram: 10
      - Normalization analysis: 10

Grading Notes:

- Unless we receive verifiable complaints about inadequate contributions, each member of a team will receive the same score on this assignment.
- 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.