CSc 460 — Database Design
Spring 2017 (McCann)

http://www.cs.arizona.edu/classes/cs460/spring17

Program #4: Database-driven Web Application

Due Dates:

Team Members:	April 19th, 2017, at the beginning of class
E-R Diagram:	April 24th, 2017, at the beginning of class
Final Product:	May 1st, 2017, at the beginning of class

Designed by *Yawen Chen and Jacob Combs*

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 lastly implement this application as a web-based system.

Assignment: In this assignment you are to implement a three-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 ER 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. **The business logic and data processing layer**, which is the middle tier that runs on an application server, which, in this assignment, will be `lectura.cs.arizona.edu` running the Tomcat web server. This layer sits in the middle, receives requests from client application and generates response back to client application. The response generation may involve accessing the back-end database you have created. Though there are many server-side techniques available for use, in this assignment we are requiring that you use Java and JavaServer Pages (JSP).
3. **Web Front-End**, which is the client user-interface. You need to design web pages appropriately to handle all the required functionalities. Your client application can run in any machine within the CS department with a web browser installed.

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

A supermarket management system needs to keep information about products, resupply orders, users (customers, employees, and managers), and orders placed online by customers for pickup in-store.

The system should provide a login window and check that the information corresponds to a valid user. There are three types of users, each of which should be given different options upon logging in: Customers should be provided an interface to order products, or view past order information, such as product name and pickup date; Employees and Managers should be able to perform customer check-outs, and get information about past online orders and the stores current stock. Managers should also be able to query past supply orders, place new supply orders, update the supermarkets stock information accordingly, and add or delete system users.

Make your own user table and product table, and keep them in the database. In your table of users, create at least two users of each type (customers, employees, and managers). In your table of products, create as many records as you need to demonstrate the correctness of the queries executed by your application (but at least 15).

(Continued...)

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

We realize that you are not all supermarket managers, and as such are not knowledgeable about all aspects of supermarket operation. Neither are we! However, you've all probably visited a supermarket; use those experiences, and some Googling, to help you. We're far more interested in your DB design and implementation than in how a real supermarket operates.

Required functionalities: Within the context and restrictions of the provided application domain, your system is expected to perform all of the following operations:

1. *Record insertion*: Your application should support inserting a new data record via web interface.
2. *Record deletion*: Your application should support deleting an existing data record via web interface.
3. *Record update*: Your application should support updating an existing data record via web interface.
4. *Record query*: Your application should support querying your database via the web interface for the problem description given above. You are required to implement at least <u>five</u> different queries, each of your own design, but with the following restrictions: At least one must be constructed using at least one piece of information gathered from the user. At least one must involve at least two relations. And, there should no trivial queries (for example, simply selecting everything from a table); your queries should be constructed to answer questions that real users would be expected to ask.

For each table you create, you need to populate a reasonable number of tuples in order to test your queries. Some basic amounts of data 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.)

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 homework provides such an opportunity for teamwork. At the same time, we realize that upper-division classes often require a team project, leaving some students hoping for something that they can implement on their own. Therefore, we are accepting team sizes of 1-4 members. Given the scope of this assignment, working solo is not recommended, but is acceptable.

You will need to agree on a reasonable work-load distribution scheme. More importantly, you need to come up with a well-formed design at the beginning. This will minimize conflicts and debugging effort in the actual implementation.

Hand In: There are three due dates for this assignment:

1. *Team Composition*: By the first due date (see the top of the front page of this handout), one member of your team must use the Google Form at `https://goo.gl/EIciRI` to record your team information Failure to do so will cost your team the points listed in the Grading section (below).
2. *E–R Diagram*: As stated above, in the Assignment section, your team will need to create an E–R diagram that describes your database design. On or before the second due date, a member of your team will need to submit your E–R diagram **and** schedule a meeting with one of the TAs to discuss your design. 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 a TA, the better. In about a week, we'll let you know how to sign up for a discussion time.

(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) A directory called "ROOT", which contains all the source code for the application. The structure of it should follow exactly as what you will see in the simple demo application (see below).
   (b) A directory called "doc", containing a PDF document including these sections in this order:
      i. *Conceptual database design*: Final ER diagram along with your design rationale and any necessary high-level text description of the data model (e.g., constraints or anything not able to show in the ER diagram but is necessary to help people understand your database design).
      ii. *Logical database design*: Converting an ER schema into a relational database schema. Show the schemas of the tables resulted in this step.
      iii. *Normalization analysis*: Show the FDs of all your tables and justify why your design adheres to 3NF.
      iv. *Query description*: Describe your five queries; what questions they are answering?
   (c) A `ReadMe.txt` describing how the TA can operate your website to see the required functionalities, and the work-load distribution among team members (that is, who was responsible for what?).

   In addition, each team should schedule a time slot (10 minutes) to meet with a TA and demonstrate your system. Closer to the third due date, We will let you know how to sign up.

A Simple Demo Application: To speed up the development, we've put a simple demo application under `/home/cs460/spring17/2017p4/`, please read the `HowTo.txt` within it to see how to run the demo. The demo contains a simple web page with a button you can click. By clicking the button, it will retrieve all the records from table `yawenchen.product` and display the content on another web page.

You should run this demo soon because (1) it will let you install the Tomcat web server under your account which is needed for your application to run, and (2) it will help you get familiar with the techniques you are going to use for this assignment.

Grading: Total: 100

1. Team Composition (1st due date): 5
2. E–R Diagram (2nd due date): 10
3. Final Product:
   (a) Coding style: 10
   (b) Database design: 15
      - Final E–R diagram: 5
      - Normalization analysis: 10
   (c) Implementation: 60
      - Record insertion: 12
      - Record deletion: 12
      - Record update: 12
      - Record query: 12
      - web front-end: 12

Note: We won't put much weight at all on the look of the pages. The main point of the assignment is the DB design, the web side is a nice bonus. You should put your focus on web page functionality. Don't worry if your web pages don't look "nice".

(Continued...)

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. (Justification: The TAs need to get grading done soon after the due date, and you need time to study for your final exams.)

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 to get their project materials submitted.