CSc 460 — Database Design
Spring 2016 (McCann)

http://www.cs.arizona.edu/classes/cs460/spring16

# Program #4: Database-driven Web Application

*Due Date: May 6th, 2016, at the beginning of class*

Designed by *Shuo Yang*

**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 finally 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 webpages 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 this project is as follows:

A University department needs to keep current information about its students, their academic advisors, the student clubs to which they belong, the advisors of the clubs, and the specific activities that those clubs sponsor (such as guest speakers, career nights, etc.).

Each student is assigned to one academic advisor, but of course each advisor advises many students. Academic advisors may be faculty members, but may instead be staff members.

Students can belong to as many clubs as they wish, and clubs can sponsor any number of activities, but each club must have at least one member to exist. Club activities are sponsored by only one club, but the department happily allows multiple club activities per day.

The department has both undergraduate and graduate students. Graduate students may be club members, but they can also be club advisors, as can faculty members or staff members. Graduate students also each have an academic advisor, and may optionally have a thesis advisor, who must be a faculty member.

**Required functionalities:**

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 five different queries, each of your own design, but with the following restrictions: ONE must be constructed using at least one piece of information gathered from the user. ONE must involve at least two relations. There should no trivial queries ( for example, simply selecting everything from a table), you queries need to be able to answer questions that real users are interested in.

More specifically, you should support record insertion and deletion for ALL tables you designed, and support record update for ONLY one table of your own choice. For each table you created, you need to populate a reasonable number of rows in order to test your queries. (50-100 rows seem reasonable, while 5-10 rows may be not)

**Work in Groups:** In industry, projects are usually the work of multiple developers, since 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. Therefore, it is required that groups of 2-4 members should be formed.

You need to come up with a reasonable workload distribution scheme. More importantly, you need to come up with a well-formed design at the beginning. This will save a lot of extra conflicts and debugging efforts in the actual implementation.

*Note*: each team should email the TA (Shuo Yang) at `shuoyang@email.arizona.edu` the members of the team no later than April 25 (Monday).

**Hand In:** You are required to submit a `.tar` file of your well-documented application program file(s) via turnin to the folder `cs460p4`. The tar file should contain the following exactly:

1. 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).
2. A directory called "doc", which contains one PDF document including the sections in the following order:
   (a) *Conceptual database design*: 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).
   (b) *Logical database design*: converting an ER schema into a relational database schema. Show the schemas of the tables resulted in this step.
   (c) *Normalization analysis*: show FDs of all your tables and justify why your design adheres to 3NF.
   (d) *Query description*: describe your five queries, what questions they are answering?
3. A `ReadMe.txt` describing how TA can operate your website to see the required functionalities, and work load distribution among team members (that is, who is responsible for what?). If there are any required functionalities you that you have failed to implement, list them.

Each team should schedule a time slot (10 minutes) to meet with the TA (Shuo Yang) and demonstrate your system. We will let you know how to sign up later.

**A Simple Demo Application:** To speed up the development, I've put a simple demo application under `/home/cs460/spring16/2016p4/`, 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 `mccann.employee` and display the content on another web page.

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

**Grading:** Total: 100

1. Coding style: 5
2. Database design: 50
   - Conceptual database design: 20
   - Logical database design: 10
   - Normalization analysis: 20
3. Implementation: 45
   - Record insertion: 10
   - Record deletion: 10
   - Record update: 5
   - Query: 15
   - web front-end: 5

*Note*: We won't put too much weight on the look of the web pages. The main point of the assignment is the DB design, the web side is a nice bonus. You should put your focus on designing web pages properly (functional) to handle the required functionalities, don't worry if your web pages don't look "nice."

**Late days:** Late days can be used on this assignment; how many a team has to use is determined as follows: Team members total their remaining late days, divide by the number of members in the team (integer division), and that's the number of late days the team has available, to a max of 'two' days. ('Two?' With 'quotes?' We need to get grading done soon after the final, so we can't let any programs be submitted after the start of the final. As the final is at 1:00 p.m. Friday and the due date is at 2:00 p.m. Wednesday, you have 47 hours, not 48. So, 'two' late days max.)

For example, a team whose three members have 1, 1, and 3 late days remaining have (1+1+3) / 3 = 1 late day to use to get their project materials submitted.