# 15-415 Databases
# Project #1 (Database Application)

**Assigned:** Thursday, September 8, 2011
**Proposal:** Tuesday, September 13, 2011 [See description below]
**Checkpoint:** Thursday, September 15, 2011 [ERDs, Schemas, Data source/format/massaging]
**Due:** Tuesday, September 27, 2011

## Overview

This assignment asks you to write a database application – from the ground up. You'll do everything from the data massaging to the database design and implementation through the GUI development. It is designed to reinforce your understanding of ER Diagrams, database schema, and SQL.

It is also designed to help you think about using databases to solve real problems for real people. At the heart of this assignment is making an interesting dataset accessible and useful to the end user. We hope you'll pick a problem that is interesting to you – and learn something in that domain as well as about databases.

Of course, there will be plenty of arbitrary complexity. You'll get to exercise your skill in data massaging, programming, debugging, version control (or, ouch, no version control) – and especially time management. Have fun!

## Teams

Teams of up to two people are permitted, with the caveat that each team member must do, at the least, some substantive portion of the ER Diagramming and the SQL coding. This may be assessed by one-on-one or team interviews and code reviews for grading purposes. Be prepared.

## Proposal

By 11:59PM Tuesday, September 13th, you should submit a proposal explaining, at a high level, what you'd like to build. It should be one to two pages long. We'll tell you how to submit it during class on Tuesday.

**Deliverables**

We'll describe the electronic submission process to you a little closer to the deadline. But, for now please know that we will ask you to submit the following –electronically--:

- Report (In .pdf), organized as follows:
    - Original proposal, unrevised
    - Project overview (Proposal revised to reflect reality)
    - Informal use case document (use case name, goal, summary, basic flow)
    - ER Diagrams
    - Lower-level schema documentation
    - Description of your input data file structure
    - Information about what we need to do to run and build your project on the Andrew/GHC Linux systems
    - Requirements checklist and mapping
    - Citations (Where you got your data, libraries you used, etc)
- Source code (in whatever languages you choose to use)
- The application, itself
- The data you used

**Good Questions, Interesting Answers**

A good place to start is to pick something you'd like to learn about – then find some data. Your data can come from one source, or multiple sources. Under "Resources", I've listed some of my favorite sources of data (Socrata, Data Ferret, and FedStats). But, there are a billion more out there. You might even have some from a local research project, yours or someone else's.

Some of the sources of data have good tools for extracting it in CSV files, etc. But, the primary responsibility for the data massaging is yours. Regardless, it might be time to dust off those old 15-123 skills. Scripting is good. Excel is good, too.

You need to have at least three different types of input record, each with more than 3-4 attributes, and from a few hundred to a few tens of thousands of records. As a rule of thumb, we want you to play with enough data, and rich enough data, to have fun and learn something, but not so much data that it becomes a problem with respect to your quota or processing power.

Across all of your data sets, your data must exist in at least three domains. If your data is all of one type, and you can't find another data set of interest to use along side of it, you can do some data massaging and break it into multiple data sets, yourself. If all of your data is in the same domain, you need to look for more or different data.

Ultimately, among other things, your application will need to allow the user to ask interesting questions about this data. So, as you are selecting your data – keep the focus on why it is interesting. In other words, keep in mind, "What can we learn by looking at this data the right way?"

**Software Requirements**

Your application needs to have a convenient, easy-to-use GUI. You can write it in any language, but we think you'll probably find Java AWT or Swing easiest. It can be a Web Application – or a local application. It doesn't need to be sophisticated or super-cool (though we like super-cool). But, it does have to be easy to use. Quick and dirty is fine.

It needs to use SQLite as its back end. You need to access the database by feeding it SQL, not through higher level constructs.

Your application needs to be able to load large volumes of data from CSV files. We might let you use XML files, or some other format, if you prefer – but you'll need to speak with one of us first. We want to keep you out of trouble.

It needs to be able to generate reports. And, by this I mean that it needs to be able to save results files, and results of a non-trivial length exclusively to files.

It needs to provide a way for the user to ask interesting questions and get answers from the data. These questions should involve you making use exercising the following capabilities in, or in conjunction with, SQL:

(a) Representing the input data using tables
(b) Building relations across input data tables
(c) Asking questions answerable from a single table
(d) Asking questions answerable using more than one table (joins)
(e) Using views
(f) Using foreign keys
(g) Summarizing data (Average, mean, median, histogram, etc)

It also should enable the user to mutate the database by deleting data that meets some non-trivial criteria, specifically in at least one case where the delete affects more than one table and is likely to affect multiple rows of at least one table. And, it needs to allow the user to add additional data with the same fields as the original input

**Project overview**

Everyone's project will be different. We want a cheat sheet to let us know what to expect. Revise your original proposal to reflect this.

**Informal Use Case Document**

We want you to document your software's use cases, just to make sure that you know what you are building before you start to build it. But, we don't' want you to invest a dramatic amount of time on formal, fully-dressed use cases. For those of you who happen to be familiar with use case documentation, here's what we need: use case name, goal, summary, and basic flow.

For those who don't happen to be familiar with formal use case documents, that's fine. We'd like you to provide us with a list of your software's features by name. And, for each named feature we'd like you to provide us with a couple to very small few sentences summarizing what the feature does, followed by a list that briefly enumerates the steps a user takes to exercise the feature.

**ER Diagrams**

Using a notation similar to the one used in class and in your textbook, please diagram your database using the ER Model. We strongly suggest using software for this rather than crayons and a scanner. Please.

**Lower-Level Schema**

We'd like you to distill your database design's ER diagrams into a terse schema. For each relation, whether representing an entity or a relationship, please document the structure of the corresponding table as follows:

TableName(field$_1$:domain$_1$, …, field$_n$ :domain$_n$ )

For example:

```
Retiree(fname:varchar(20), lname:varchar(20), dob:date, balance:money, update:datetime)
```

**Description of Your Input File Structure**

We'll eventually probably poke at your raw input data. Please briefly and tersely describe for us the meaning of each field and its representation. This goes for the raw data, as you get it, and any post-processed data files that you may create from it via scripts, etc, prior to feeding your database. For example:

Field 1: Retiree's first name, text
Field 2: Retiree's last name, text
Field 3: Retiree's data of birth MM/DD/YYYY or MM-DD-YYYY
Field 4: Retiree's pension balance in dollars and cents, without commas, e.g. $25673.09

**Information We Need to Run and Build Your Project**

We're going to take your source code and plop it into a directory. We're going to take your data and plop it into another directory. We're going to take SQLite and plop it into a third directory. Then, we are going to twirl our magic wands in the air twice and then tap them on the keyboard three times. At this point, we expect that your project will build. Next, we are going to sprinkle some pixie dust on our keyboards, clap our hands, and expect that your project will start to run.

If this is not a viable technique for causing your project to build and run, please provide us with brief instructions for a better approach. You may assume that we have only the software installed on the Andrew/GHC Linux systems, plus anything you give us. Even better would be a really brief instruction about using and ant script, shell script, make file, Eclipse project, etc.

**Citations**

Please let us know any third party libraries or tools you used, where you got your data, and anything or anyone else that contributed to the success of your project and how.

**Requirements Checklist**

The bullet points below are intended to summarize the details of the requirements of this project. But, they are only a summary. The details in the other sections are definitive.

- Report (In .pdf), organized as follows:
  - Original proposal, unrevised
  - Project overview (Proposal revised to reflect reality)
  - Informal use case document (use case name, goal, summary, basic flow)
  - ER Diagrams
  - Lower-level schema documentation
  - Description of your input data file structure
  - Information about what we need to do to run and build your project on the Andrew/GHC Linux systems
  - Requirements checklist and mapping
  - Citations (Where you got your data, libraries you used, etc)
- Source code (in whatever languages you choose to use)

**Requirements Checklist, *cont.***

- The application, itself
  - SQLite back end
  - Coded using SQL, not higher-level primitives
  - Mass load feature
  - Report feature
  - Features that exhibit the following abilities in, or in conjunction with, SQL:
    - Represent the input data using tables
    - Build relations across input data tables
    - Ask questions answerable from a single table
    - Ask questions answerable using more than one table (joins)
    - Use views
    - Use foreign keys
    - Summarizing data (Average, mean, median, histogram, etc)
    - Mutate the database by deleting data that meets some non-trivial criteria, specifically in at least one case where the delete affects more than one table and is likely to affect multiple rows of at least one table.
    - Allow the user to add additional data with the same fields as the original input
- The data you used
  - Few hundred to few thousand input records
  - 3-4 fields per input record
  - At least three different types of input records
  - Comma-delimited (or permission received)
- This checklist, itself
  - Explain the status of each incomplete item
  - For each of the "Features that exhibit the following abilities…", please identify at least one user-visible feature. If you missed any required bullet points, please clearly identify this.

**Some Resources**

| | |
|---|---|
| SQLite | (http://www.sqlite.org/) |
| SQLiteJDBC | (http://www.zentus.com/sqlitejdbc/) |
| Swing Tutorial: | (http://download.oracle.com/javase/tutorial/uiswing/) |
| AWT Tutorial: | (http://java.sun.com/developer/onlineTraining/awt/contents.html) |
| Socrata Data: | (http://www.socrata.com/solution/socrata-vs-dataset-directories) |
| Data Ferret: | (http://dataferrett.census.gov/index.html) |
| FedStats: | (http://www.fedstats.gov/toolkit.html) |