# SQL Programming Project

CS-6360 Database Design

Due: 11:59pm, October 12, 2018

## Description

This SQL programming project involves the creation of a database host application that interfaces with a backend SQL database implementing a Contact List application. This is an individual (not group) project. All work, design, and coding must be done individually by you.

## Programming Language(s) and Frameworks

Your host application should have a GUI interface. You may implement either a native GUI application or a web GUI interface.

Your application GUI may be programmed with Java, Ruby, Python, Javascript, or PHP.

- You ***may*** use a web programming framework like Django (Python) or Node.js (Javascript).
- You ***may*** use an ORM (object-relational mapping) framework, like Hibernate, provided that your SQL schema is compatible with the requirements.
- You ***may not*** use a NoSQL database such as JSON, JDOM, XML, etc.
- Approved SQL databases are MySQL, PostgreSQL, MS SQL Server, and SQLite.

If you would like to use any other language or framework not specifically listed above, you *must* obtain prior approval from the TA that she is able to effectively evaluate the language of your submission.

## Functional Requirements

### 1) Graphical User Interface (GUI) and Overall Design [20 points]

All interface with the database (queries, updates, deletes, etc.) must be done from a graphical user interface of your original design. Your GUI application will interface with the Contact List SQL database via an appropriate SQL connector. Initial database creation and population may be done from command line or other admin tool, like a Workbench. Overall design will be judged on usability first, look-and-feel secondary.

Required information to display in a Contact Display Window includes:

- First, Middle, and Last name
- Address List – must accomodate a variable number of addresses, each of which should contain the following:
  - Type of address (e.g. home, work, other, etc.)
  - Street Address
  - City
  - State
  - Zip code (i.e. postal code)
- Phone Number List - must accomodate a variable number of phone number, each of which should contain the following:
  - Type of number (e.g. home, work, fax, etc.)
  - Area Code (3-digits)
  - Number
- Date List – must accomodate a variable number of dates, each of which should contain the following:
  - Type of date (e.g. birthday, anniversary, etc.)
  - Calendar date
- An option to modify or delete an entry from the Contact Display Window.

## 2) Contact Search [20 points]

Using your GUI, be able to search for a contact, given any combination of Name components, Address component(s), Phone number components. All searches should be done via a ***single*** search field (like Google or Bing). The result of your query should display a list of all contacts that match the search. A search results should include a list of "hits" that displays

- Full name
- An option to modify or delete an entry from the search results.

## 3) Adding New Contacts [20 points]

Your GUI should provide a menu item or button that brings up a New Contact Entry form. The New Contact Entry form should allow for entry of all data, including the ability to add a new Addresss, new Phone number, or new Date to the respective lists.

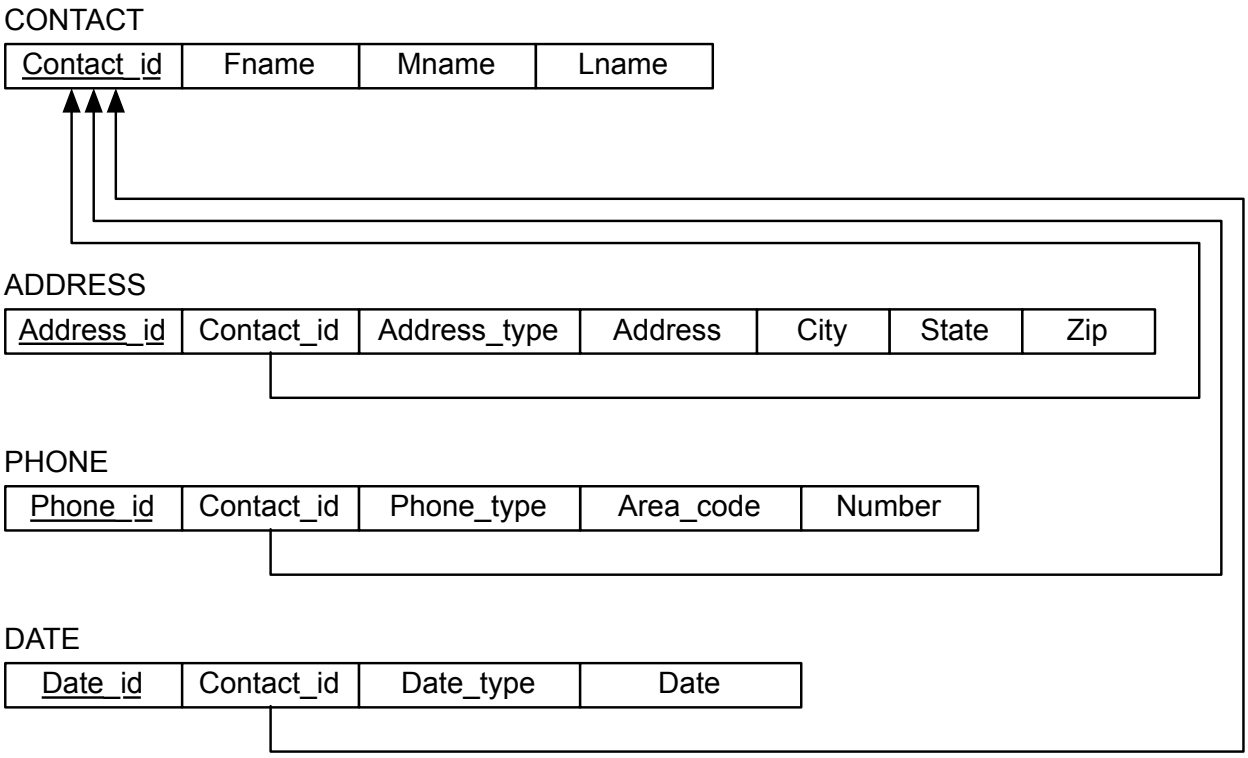
## 4) Modifying Existing Contacts [20 points]

Your GUI should provide a menu item or button that brings up a Modify Contact Entry form. The New Contact Entry form should allow for entry of all data, including the ability to add a new Addresss, new Phone number, or new Date to the respective lists.

## Schema

The schema for the minimum required SQL schema requirements are pictured below. You are permitted to modify or augment this schema provided that your system (a) is backwards compatible with the given schema, and (b) adheres to the written requirements. As long as your system supports the documented functionality, you may add any features you deem useful.

Note that every record has a primary key that should be auto-generated. It is suggested that these keys be integers that are created with the SQL AUTOINCREMENT option.



## Data

- Baseline data to initialize your database is provided in the eLearning programming assignment folder.
- All data is provided in plain text CSV files. Note that there is ***not*** a one-to-one, file-to-table correspondence. Part of your system design task to map (i.e. normalize) these data onto your schema and tables.
- All book id's are 10-character ISBN numbers (i.e. some contain alpha characters). Note that some may contain leading zeroes. These are part of ISBN and should ***not*** be truncated!

## Submission

You will be required to submit the following files:

- A design document that describes your system architecture including design decisions and assumptions. (1.5-3 pages 12 point font, ***not*** including any schema diagrams or system architecture figures). Format should be PDF.
- A Quick Start user guide for librarian system users (1-2 pages).
- A `readme.txt` file that describes how to compile, build, and install your application. It should include any technical dependencies (language, frameworks, platform, OS, software libraries, software versions, etc.).
- All application source code, including any build files (e.g. make, ant, maven, etc.).
- All files must be zipped together into a single file. This file should be named `<netid>_cs6360.zip`, where `<net-id>` is your Net ID.
  Example: `cid021000_cs6360.zip`

## Grading

You will be required to demonstrate your application for the TA. If you are unable to bring a laptop computer to demonstrate your application, please let me know as soon as possible so I can make alternate arrangements.

An online sign-up mechanism will be made available to reserve a specific time with the TA for evaluation and grading. As a courtesy to the TAs and those waiting behind you, **please be on time for your scheduled slot** and have your application already launched and ready to go.

Each student will have 10-15 minutes to demonstrate their system and execute the test cases provided at grading time. Test cases will be designed to validate use cases from the published requirements.