Ian Lizarda, Donovan Moini
Homework 1

Problems from Chapter 1

- Ch 1, #13: *Give examples of systems in which it may make sense to use traditional file processing instead of a database approach.*
    - A small single user application like a Calculator app which only requires calculations locally
    - Something similar to the macOS “Finder” utility, which only needs to locate files
    - Offline games that are stored locally, like Minesweeper
    - Microsoft Word, Excel, and PowerPoint with local files

- Ch 1, #14: *Consider Figure 1.2.*
    a. *If the name of the 'CS' [Computer Science] Department changes to 'CSSE' [Computer Science and Software Engineering] Department and the corresponding prefix for the course number also changes, identify the columns in the database that would need to be updated.*
        i.

Table	**Column**
STUDENT	Major
COURSE	Course_number Department
SECTION	Course_number
PREREQUISITE	Course_number Prerequisite_number

    b. *Can you restructure the columns in the COURSE, SECTION, and PREREQUISITE tables so that only one column will need to be updated?*
        i.

Table	**Column**	**New Columns**
COURSE	Course_number	Course_number & Course_department
SECTION	Course_number	Course_number & Course_department
PREREQUISITE	Course_number	Course_number & Course_department

PREREQUISITE	Prerequisite_number	Prereq_number & Prereq_department

Problems from Chapter 2

- Ch 2, #3: *What is the difference between a database schema and a database state?*
  - The database schema is the **description** of the database, which is specified during database design and is not expected to change a lot. A displayed schema is called a **schema diagram**. A database state, however, refers to the **data in the database at a particular moment.** It is also known as the *current* set of **occurrences** or **instances** in the database.
- Ch 2, #7: *Discuss the different types of user-friendly interfaces and the types of users who typically use each.*
  - Menu-based Interfaces for Web Clients/Browsing
    - This interface presents the user with a list of options (a menu), which are popularly used in web-based and browsing interfaces. Users who use these often look through the contents of a database in an exploratory and unstructured manner.
  - Forms-Based Interfaces
    - These interfaces displays a form to each user, which they can fill out the form entries to insert new data and fill out certain entries. Forms are usually designed and programmed for naive users as interfaces to canned transactions.
  - Graphical User Interfaces (GUI)
    - This interface displays the schema to users as a diagrammatic form and allows users to specify queries by manipulating the diagram. Most use pointing devices (mouse, stylus, etc…) to select specific parts of the diagram. Most users end up interacting through a GUI nowadays unless using specific instances of a command-line interface.
  - Natural Language Interfaces
    - These interfaces accept requests written in English or another language and attempt to *understand* them. A NL interface usually has its own schema and a dictionary of important words. For example, search engines accept strings and match them to web pages.
  - Speech Input and Output
    - Typically used for applications with limited vocabularies including inquiries for credit card information and telephone directories. A library of predefined words are used to detect speech input and set up parameters supplied to queries, with outputs are supplied in a similar conversion.
  - Interfaces for Parametric Users
    - Parametric users have a small set of operations they must perform repeatedly, such as a single function key that invokes routine and

        - repetitive transactions like balance inquiries. The goal is to minimize the number of keystrokes required for each request.
    - Interfaces for the DBA
        - Includes commands that can only be used by DBA staff, such as creating accounts, setting system parameters, granting account authorization, changing schema, and reorganizing the storage structures of a database.

- Ch 2, #14: *If you were designing a Web-based system to make airline reservations and sell airline tickets, which DBMS architecture would you choose from Section 2.5? Why? Why would the other architectures not be a good choice?*
    - The three-tier client/server architecture for web application is the best choice. Users would interact with the client and send requests to the server for airline inquiries. The web server, which contains the application logic, would handle all the rules and regulations related to the reservation process and issuing tickets to the user, The database server contains the DBMS for the users, tickets, flights, etc.
    - The centralized DBMS architecture would not work because the user interface and database server would exist on different machines for a web-based system.
    - The basic client/server and two-tier client/server architectures could work, but having the business logic exist in the DBMS server would be extremely taxing and put a burden on the server.

Problems from Chapter 3

- Ch 3, #2: *Why are tuples in a relation not ordered?*
    - A relation is defined as a set of tuples. Elements of a set have no order among them, therefore tuples in a relation don't have any particular order.
- Ch 3, #5: *Why do we designate one of the candidate keys of a relation to be the primary key?*
    - We commonly designate one of the candidate keys as the primary key of the relation, which is used to uniquely identify tuples in the relation. Deciding which candidate key becomes the designated primary key is random, but it is better to choose a primary key with a single attribute or a small number of attributes.
- Ch 3, #9: *Define foreign key. What is this concept used for?*
    - A foreign key is used to link two tables in a database by choosing a key (usually a primary key) in one table and maintaining it as an attribute in another table.
- Ch 3, #13: *Consider the relation CLASS(Course#, Univ_Section#, Instructor_name, Semester, Building_code, Room#, Time_period, Weekdays, Credit_hours). This represents classes taught in a university, with unique Univ_section#s. Identify what you think should be various candidate keys, and write in your own words the conditions or assumptions under which each candidate key would be valid.*
    - **Univ_Section#:** This would make for a good candidate key with the condition that the section is unique across all semesters.

- **(Course#, Univ_Section#, Semester):** This candidate key would be a good candidate key assuming that the Univ_Section# key is not unique across all semesters.
- **(Weekdays, Semester, BuildingCode, Room#, Time_period):** This would be a good candidate key assuming that only one class is taught at a time for every room.

- <u>Ch 3, #20 [part c only]</u>: *Recent changes in privacy laws have disallowed organizations from using Social Security numbers to identify individuals unless certain restrictions are satisfied. As a result, most U.S. universities cannot use SSNs as primary keys [except for financial data]. In practice, Student_id, a unique identifier assigned to every student, is likely to be used as the primary key rather than SSN since Student_id can be used throughout the system.*
  - c. *What are the advantages and disadvantages of using generated [surrogate] keys?*
    - i. Advantages
      1. Student IDs are unique among students at the same university
      2. Surrogate keys are typically integers, meaning they only need 4 bytes of memory, so primary key index will be small. This makes certain operations quicker.
    - ii. Disadvantages
      1. Two schools could use the same student ID for a student at each university
      2. Have no meaning, so usually not useful when searching for data
      3. Since usually 32-bit integer, limited to 2147483647 possible student IDs