Design Documentation

Section 1

The goal of this project is to create a primitive database management system based on relational algebra. We are to develop the basic management system engine code, develop a parser to interpret commands that are to be fed into the engine, then later create an application for the system as well as fine tune the parser and engine.

Management systems are essential to solve many of today’s problems, and the project’s goal is to show us the fine tunings on how such an engine works by creating a basic example. Through this assignment, not only do we get to know the depth of such an advanced piece of software, but we adopt the skills of working with other coders on the same project and the management involved in it.

As far as the database management engine goes itself, there are several concepts that we have to adopt such as the design and functionality of the engine, the grammar of the system to interpret and parse inputs to allow the essential functions that any end-use of a database management system would need, and the application itself which handles the user’s input and output. After fulfilling these requirements, we should have a basic database management system fully functioning.

Section 2

High Level Entities - Description (from bottom to top)

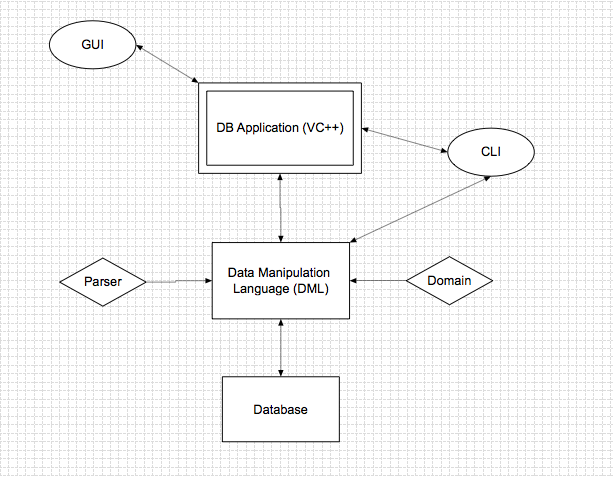
Database: Represents a collection of files (.txt, .csv, .xml,…) that each identify a relation table. A table can identify attributes for a specific entity and/or relationships between many entities.

Data Manipulation Language (DML): Language to respond to database queries. These consist of: open, close, write, exit, show, create table, insert into, update, delete, select, project, +, -, and \*. More generally, the implemented functions (for which a CLI will be provided) will allow us to update the database for long-term usage.

* As mentioned above, a command line interface (CLI) will be provided to make the DML function calls to the database
* A parser will be implemented for our domain-specific DML for our queries
* The domain for our database will also be defined

DB Application: written in c++ to interact with the DML and provide the user with a way to use the database. This application can provide the user with direct access to the stored data and/or integrate the data to the back-end for a more complex application

* a CLI will be an easy implementation to use the functionality of the DB
* a GUI will be a more difficult implementation that will result in a more user-friendly experience



Section 3

**Usage**

The objects we will be using are stored within a database. The user will be able to create the object, edit the object, or remove the object from the database. There will be external access to the database that will allow the user to draw information from it and edit it as desired. Since the objects will be stored in a database we will keep the information secure by having the external sources accessing it with the necessary permissions. Since we will keep users from having direct access to the database, we can perform necessary checks to insure that the data will remain uncorrupted. This will reduce the risk of having the database crash from user error.

**Configuration**

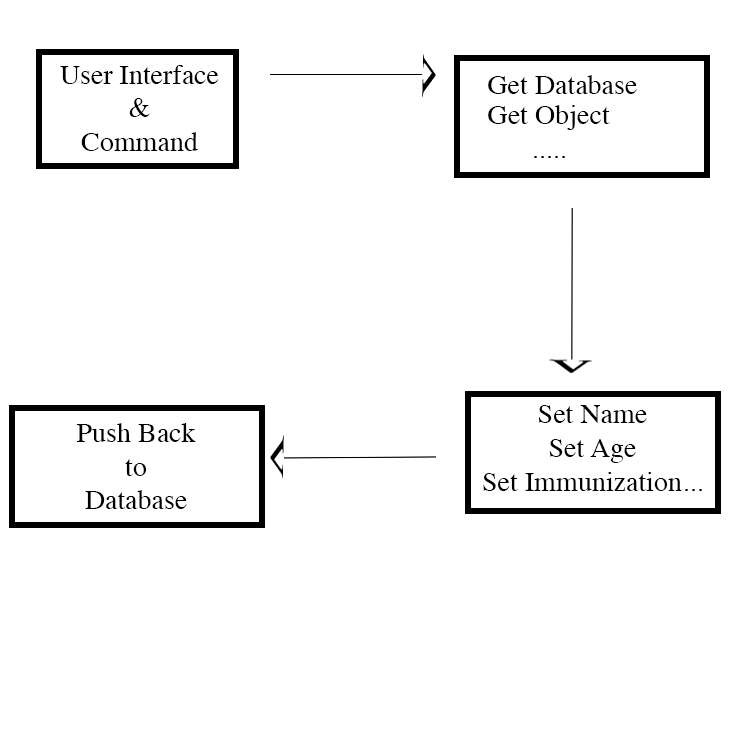
The object we will be implementing stores different types of information. The user must correctly define all of these different variables, before the object can be created and stored within the database. The proper functions must also be called if the user wishes to edit any object or remove it from the database.

**Model**

Model 1 depicts the interaction of the object with the user and the database. There is some pseudo code the shows some of the characteristics that the user will be required to input when creating an object. The pseudo code also shows the relation that the user has when accessing the object. The information will be pulled, edited, and then ‘pushed back’.

**Interaction**

Since the interaction of the object is pretty simple and straight forward, there is no diagram for this section. The user will access the database, view the information, retrieve the required data, manipulate the data, and push the changes to the database. The object itself will store the information that the user can edit so there may be implementation of reading the objects and pulling those that meet a certain condition (Such as Age >= 4) but the data of one object will not influence that of another.



Section 4

The main benefits of the database management system are the reduction of corruption in data, it keeps the information secure from any possible intruders trying to obtain specific information, and it will allow for each object to be handled separately. Another benefit of the design, not involving the database management system, is the freedom with input when the parsing function is being implemented. The main issues that will be faced with the implementation of the design is making sure each part in the design does what it is supposed to do and making sure that the parser integrates properly with the database management system. The only major risk with the design that will be faced, because programming will be separated amongst other coders, is that each part in the design will need to be compatible to several parts in the database management system.