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 how a database management system works, 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

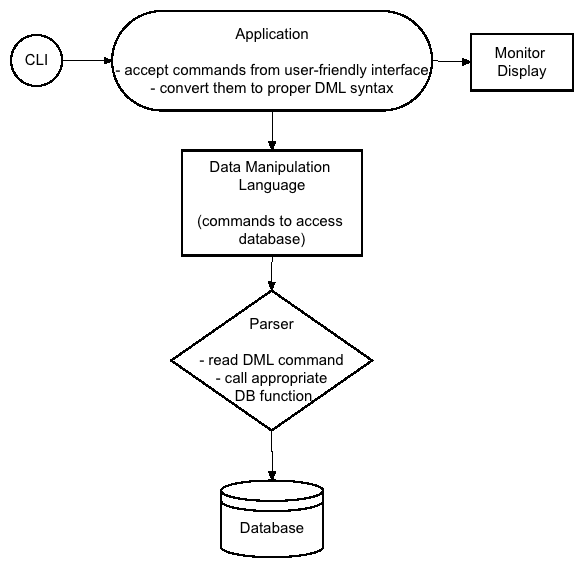
**Database:** Underlying data storage for our pet store, implemented in C++. Modifications will be made to this static class by the Parser (more detail is given in Section 3 – Model)

**Parser:** The commands mentioned above will be read by a parser which will interpret their meaning according to the provided phrase-grammar structure. The parser can call Table and Database functions to make appropriate modifications and accesses.

**Data Manipulation Language (DML):** Language to respond to database queries. Ultimately, users will not have access to these commands. They consist of:

* Queries
* Commands (open, close, write, exit, show, create table, insert into, update, delete, select, project)
* 2-Set Manipulations (union, difference, product, and natural-join)

**Application:** A C++ program that allows a user to use the data stored inside of the Database in an interactive way. Users do not have direct access to the data itself, nor the Data Manipulation Language. Commands will be simplified and user friendly, but converted to DML which is read by the Parser to communicate with the database. A command line interface is provided for this functionality.



Section 3

**Usage**

Objects:

Database - The purpose of the database is to store tables in it. With the tables being stored within the database, we can call on certain tables to perform actions to them individually or as a group. We defined this variable as global to allow access to this data type across all of our functions. It is generally not good practice to create globals, but this implementation allows ease of use when performing certain actions in our program.

Table – The purpose of the table is to store values for column names, entries, and fields. We will implement function that will edit certain values of the table or use tables to define new tables. We chose to create an object table made up of entries (rows of fields) because this is an easily manipulative method. This method also follows the basic structure of a table so it is easy to comprehend when determining how you want to manipulate the data.

Entry - The table will be made up of rows of entries. The entries will store multiple fields that will correlate with the columns. The fields stored within the entries can be manipulated or scanned to perform comparisons between entries. (Therefore allowing us to compare tables to each other) We chose this kind of implementation because it follows the basic structure of a table and will be easily manageable.

**Configuration**

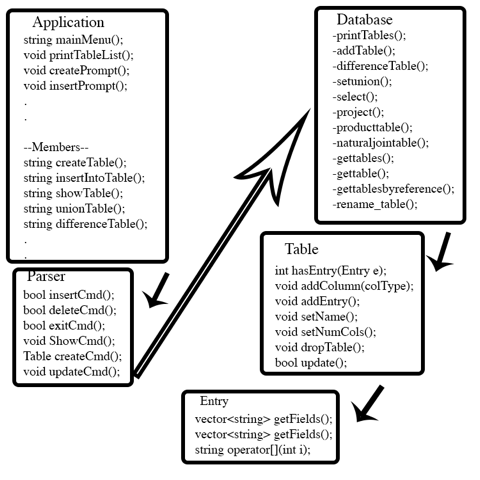
The specific way to create and manipulate this data is determine by the parser. A list of commands available to the user will be displayed to the user. These commands must be typed in the proper format to allow for successful creation and manipulation of the database.

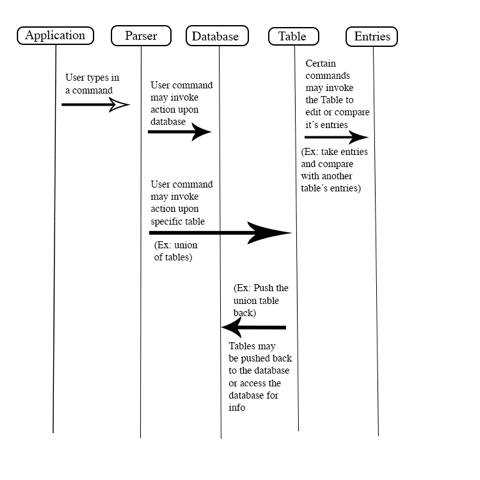
**Model**

In the model the starting point is the application. The chain of occurrences will follow down through the parser, database, table, and entries. The database will contain the data that the functions manage and invoke action upon. The application will let the user create the tables with their own data or allow the user to call for information from a file. The rest of the commands will follow down the list of the hierarchy and are used by user commands.

**Interaction**

The objects will be able to interact with other objects. Model #2 shows that interaction of the objects with each other. Some instances there is a one-way relation, while in other cases there may be a two way relation. (Or a relation to another object of the same kind). Tables can communicate with other tables so that a comparison may occur and a new table is returned. In other instances, a Table may call upon entries, but entries can not call upon a table. (or any other object) All of these examples are show in the diagram.





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 are 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. Other issues that were faced regarding the implementation of the design was understanding other peoples code when the group could not meet up. The critical set back was that the understanding of some parts to some people was different then the actual implementation, which forced them to go back to the old functions and redo them. 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.