Group Report

Waylon Brown

Patrick Sheehan

Elliut Riviera

Garrett Grams

**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**

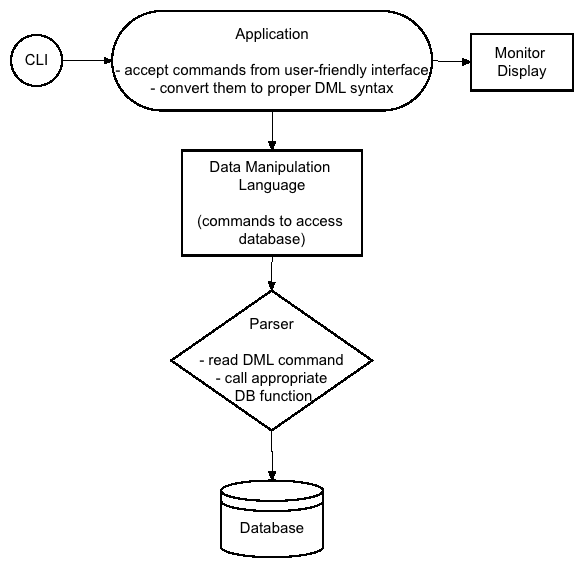
**Database:** Underlying data storage for our pet store, implemented in C++. Modifications will be made to this static class by the Parser (more detail is given in Section 3 – Model)

**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**

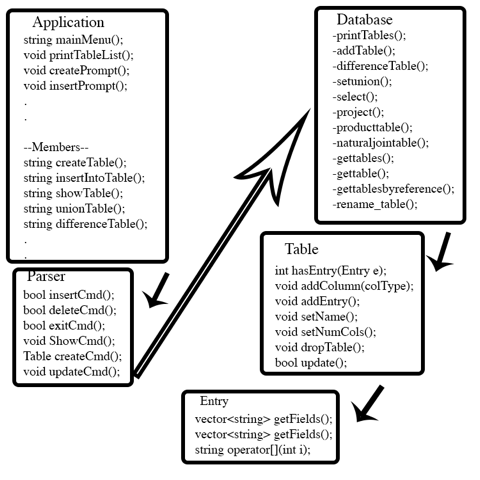
The specific way to create and manipulate this data is determine by the parser. A list of commands available to the user will be displayed to the user. These commands must be typed in the proper format to allow for successful creation and manipulation of the database.

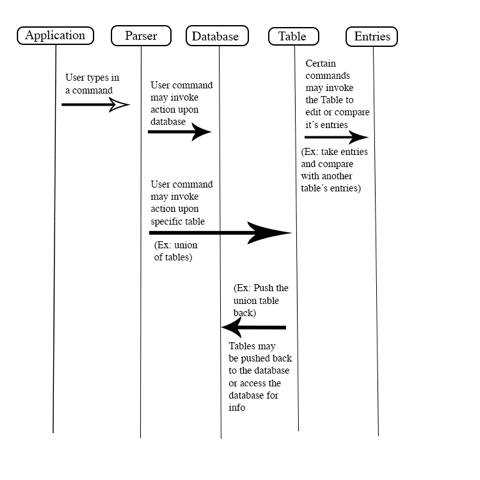
**Model**

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.

**Post-production notes**

**Waylon**

For my part in the Design Documentation, the summary, I now had a better picture of the DBMS as a whole and was able to give a better summary about the system. For the DBMS, there were many changes we each had to make. For an example, one part I had was making the application, and initially I made the menus and they executed functions in the Database. I later had to change this to where they each return a query string that can be run by the parser. The most difficult part of the project for me was collaborating with the partners and getting together to ensure all work was complete from each person. A lesson I have learned is to spend more time with the planning during labs, over coding, as it was ultimately more valuable when we are all together.

**Patrick**

A couple weeks into our project, we realized that our initial design was not thorough enough. We had a general idea of what we were going to do, and jumped right into coding without enough preparation. This caused problems when we realized that most of our code was dependent on each other’s. We had to go back and revise our design to be much more specific in a couple areas. For one, the intercommunication between entities needed a complete architecture. We had to establish how our exactly Data Manipulation Language would be parsed, and understanding the phrase-grammar structure took some time for our group. Subtleties, like return values, needed to be modified frequently near the beginning of our project; they seemed sensible at the beginning, but were not always practical to call when their functionality was needed.

Our main problem was working with GitHub. Learning and understanding it at the beginning took as much or more time than coding did. Obviously it is very valuable and universally used, so I was happy with our time spent there. Similarly, working in a large group was in itself a challenge. Any one of us could have probably done the assignment individually in a month, so this scenario was more educational in terms of collaboration (and GitHub usage).

Test Driven Development was another valuable skill I was happy to learn. It seemed very tedious when we were first introduced to it, but soon we realized the many benefits it has. If done correctly, it can lead you in the right direction when starting to write a function. More commonly, in our case at least, it made it much easier to debug both our own and others’ functions.

**Garrett**

For my part of the assignment, there were a few changes made to my original design/approach.

-Rename was originally designed to take in a list of new attributes that would simply edit the current tables attributes. I was notified that a new table should be returned so I implemented a function called “rename\_table” that took the “rename” function, applied it, and returned a new table object.

-Originally the setunion implementation was more complex. Waylon ended up creating a “has entries” function for his part of the assignment. Upon seeing this new function, I edited my part to accommodate for the use of this new function. This cut down on complexity and lines of code.

-Other than the previously mentioned changes, there was nothing major that was changed from my original idea/design.

Difficulties:

The major challenge that I faced in this assignment was learning GitHub. Trying to use this program cause me more issues and trouble than the actual programming of the assignment did. Other difficulties included conceptualizing how we were each going to implement our parts and get them to work together.

Solutions:

When dealing with GitHub I was able to get the assistance of my teammates to reduce the grief caused by this program. Otherwise, I just tried to look up “work-arounds” or came up with my own ways to handle the issue at hand. The solution to other difficulties was brainstorming with my teammates and bouncing ideas off of each other. This helped generate more ideas and a better understanding of the subject material.

Lessons Learned:

Creating a parser on this scale was a first for me, so this helped me learn about parsing. This includes different methods of parsing, ways to handle input, and different comparisons to help create a parser. I also was able to learn about test methods and debugging. This assignment was the first time that I used test methods and it certainly makes debugging easier. (Beats having cout statements everywhere. This also includes learning different tools of Visual Studio)

**Elliut**

The change that had to be done was with the Selection function,for relational algebra, was supposed to be simplified. It was assumed that the relational algebra for select allowed the user to choose the columns the user wanted to check with the condition but it was overcomplicated.

The difficulties that were faced during the implementation of the Database was that  some separate functions were not working properly, in which others relied on. This forced others to wait for another persons part to test our functions. There were also many issues with github. This was mainly due to the lack of understanding on how the guy works and what certain errors were.

The main Solutions that we implemented was meeting up on the weekends and in order to resolve the difficulties with waiting for another persons functions. Also we began to use the parser unit test to not rely so much on the the other functions.We spent an entire lab day on just making sure each individual in the group understood how it works.

The lessons that were learned were about the usefulness of using IDEs and How useful gitub is for working with groups on a single project. This also goes for using the Test methods and how resourceful it is to use it to debug. They are truly good resources when fully understood.

**Individual work load distribution**

**Waylon:** 22%

**Patrick:** 34%

**Elliutt:** 22%

**Garrett:** 22%

**Development log**

**Jan 31st:**

Group met in lab and began brainstorming. Discussed ideas and direction to take the project. Group concluded that we would eventually create an application that implements these objects as a “Pet Store” app. Divided up the work on which parts of the Design Document we would do.

Waylon – Section I

Patrick – Section 2

Garrett – Section III

Eliutt – section 4

**Feb 3rd:**

Group met to consolidate the design document parts. Discussed the overall idea behind the project as well. Design Document eventually submitted by Waylon.

**Feb 4th:**

Began discussing the idea we would take for creating the DBMS engine code. Nothing was formally decided yet, but an idea was created. Working with the idea, we began coding the implementation and divided up the work.

Patrick – create table, drop table, insert into, update, delete from, join  
Garrett – Rename & Setunion

Waylon – SetDifference & Cross Product

Eliutt- Selection & Projection

**Feb 8th:**

Conversed online with each other about the status of our progress. Decided that we would meet up on Feb 9th to work on our parts together.

Waylon – Finished SetDifference and Cross Product

**Feb 9th:**

The group met on campus and began consolidating ideas/code and helping each other with parts that needed finishing touches. Helper functions that other group members had made were implemented where necessary in each individual’s part. We were unable to finish all of the individual work, but we had a concrete idea of what needed to be done and how it could be implemented. Decided we would meet Feb 10th again to add finishing touches.

Patrick – finished create table, drop table, insert into  
Garrett – encountered issues with setunion and tried to solve these issues. Finished rename (however this version did not return a table.)

Eliutt- finished Selection & Projection functions

**Feb 10th:**

The group met to finish up individual work/assist each other/ and consolidate the work.

Patrick – finished update, delete from, join

Garrett – Finished up sections of the work. (Setunion was commented out in the submission.)\

Eliutt- Finished up Project command.

**Feb 11th:**

The group met in lab. We discussed that we should meet for a longer time/more frequently to get work done faster. The meeting time was spent on trying to plan out when we could meet and how we were going to approach the parser. We designed the parser to ultimately taken in a line of DML (as a string), conveniently split it up and put tokens in a vector, then we write the functions to interpret subsets of that vector. The subsets evaluate vectors to expressions, conditions, commands, etc.

**Feb 13th:**

The group met during lab time and finalized how to split up work for the parser. We decided:

Garrett – query, relation-name, identifier, alpha, digit, command, open-cmd, close-cmd, write-cmd

Patrick – expr, atomic-expr, selection, condition, conjunction, comparison, insert-cmd, delete-cmd, typed-attribute list, type, integer

Eliutt – op, operand, attribute-name, literal, projection, attribute-list, renaming, exit-cmd, show-cmd, create-cmd, update-cmd

Waylon – union, difference, product, natural-join

**Feb 15th:**

The group did not meet in person but we talked over groupme about our progress and if anybody needed something from another group member before they could proceed. We realized that a lot of our respective parts were very interdependent. We decided to meet on Feb 16th on campus to see if we could get more work done while together in person.

**Feb 16th:**

We met on campus and worked towards completing the parser. This meeting was continued on **Feb 17th**. We mainly spent time working on our individual parts

Eliutt – completed the rename command and created a unit test to make sure it was full functional

**Feb 18th**:

With the idea about what kind of app we wanted to make, we worked towards finishing up the parser from its incomplete state. We each worked individually to make sure the functions were being implemented properly and continued making test cases to verify the status of our implementation.

**Feb 20th:**

We continued to work on getting the parser synced with our implementations and created more test cases to verify the results. Some progress was also made on the app side of the coding. The group also decided to push for getting a GUI completed if time allowed.

Waylon – Do the Application.

Patrick – worked on test cases and different parts of the parser.

Eliutt – worked on individual parts of the parser.

Garrett – worked on individual parts of the parser.

**Feb 23rd:**

The group met up and we continued to work on finishing touches and verifying our implementations. At the end of the meeting we each had individual tasks to work on until Feb 24th and goals we strived to meet. We also started correcting the Design Document for resubmission and went through our dev log.

Waylon – finished some function implementation with the app, application almost completed

Patrick – finalized test cases and parser functions.

Eliutt- continued to work on individual parts of the parser.

Garrett – continued working on individual parts of the parser. (Updated some of the Dev Log)

**Feb 24th:**

The group met on campus to complete the whole project. This included finishing up the app code and linking it with the rest of our code, touching up the design document, and finishing up the dev log.

Waylon – Application fully finished with unit tests passing, working on getting it functional with all other person’s functions

Eliutt- The Create, Update, Exit, and show commands are all functional, worked on it getting it functional with unit test’s

**Example Session**

