Bianca Lott

Senior Design Final Report

May 5, 2014

Table of Contents

Part I: Overview of the Project

E-Quote, A Stock-Market Analyzer is a tool which users can understand and comprehend stock-market data. It is designed to help users who do not work in the finance industry or business industry analyze stock markets in a more organized and comprehensible fashion. Most people find analyzing stock markets a complicated task and E-Quote will make this task very easy for users. Such users who will be using this product are students working on assignments or research projects, professors who might want to teach a class on finance or stock-markets and others who are interested in information pertaining to stock markets.

E-Quote has several components which are the database, query language, parser and user interface. The database contains information such as different stock values of different stocks like Google and Microsoft. It is constructed using MYSQL Server. The query language is the language that is used by users to gather information about the different stock markets. For example, a student who wants to get information about the stock market of Apple would input a query using the defined query language. The query language is based on SQL (Structured Query Language) but has unique functions which display different aspects of stock-market data. The parser parses or separates components of the queries so that they can be configured to compute the correct result to the user. Finally the user interface which is constructed using the Java language is what the user sees when using E-Quote.

Part II: Technical Design

E-Quote, which is a stock-market analyzer, has four major components the database, query language, parser and user interface. The database is constructed using MySQL Server and is composed of a SQL table which contain attributes pertaining to stock-market data such as the Stock name, symbol, value, and stock ID number. Below is image of the database using MYSQL Server via the command line in Microsoft Windows.

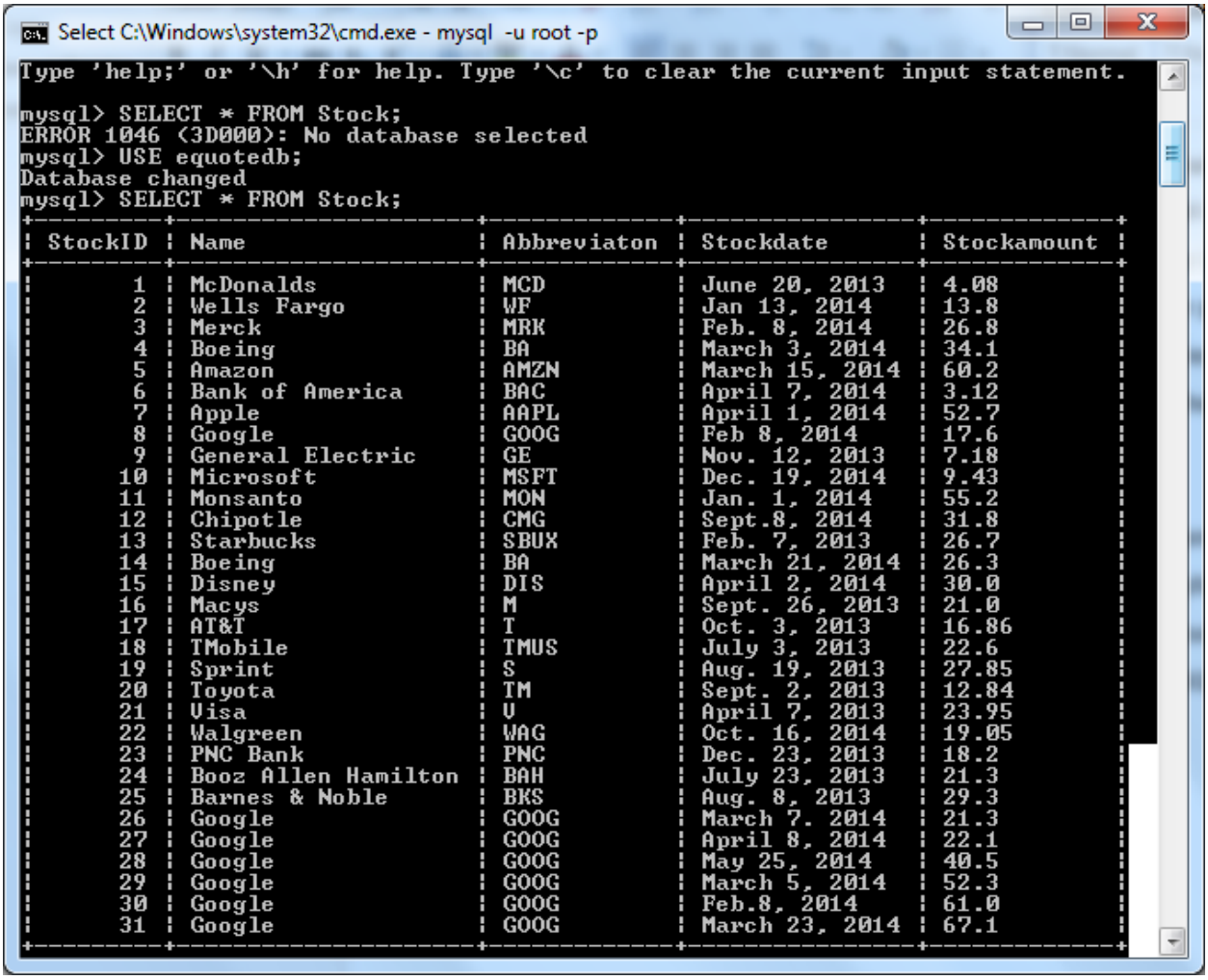

Fig.1

The query language of E-Quote is similar to SQL, but possesses different functions that are not in SQL for example, the SQL language does not have functions that output the lowest or highest values within a certain range. Also, SQL does not have an average function in which it can take the average of particular values surrounding a certain value. There are three key unique functions defined in E-Quote's query language. They are LO, HI, and AVGSUR. The LO function computes the lowest stock values. The HI function computes the highest stock values and AVGSUR computes the average of surrounding points pertaining to the targeted stock value. All of these functions use two parameters to compute the output. For example, the LO

function takes the parameters (Stockamount, 2) to compute the lowest two stock amount values. This also pertains to the HI function and AVGSUR function. The functions do work within a threshold of three, which means the HI and LO functions display the three lowest and highest values for a certain stock and the AVGSUR computes the average of up to the 3 values to the left and right of the target stock value. The goal of defining E-Quote's language is making it unique to SQL and incorporating functions that do not exist in SQL.

The second component in designing E-Quote is the parser. The parser is designed using ANTLR (Another Tool For Language Recognition), which is a parser generator which processes the queries defined in E-Quote's language. A grammar file is constructed which defines the parser rules so that the parser is created. Once the grammar file, ANTLR automatically converts the grammar file into Java code describing the parser rules. Thus the parser is able to walk the parse trees and output and interpret the queries correctly. Below is an example of a query using E-Quote's language being parsed using ANTLR in the form of an abstract syntax tree.

Query: SELECT LO(Stockamount, 3) FROM Stock;

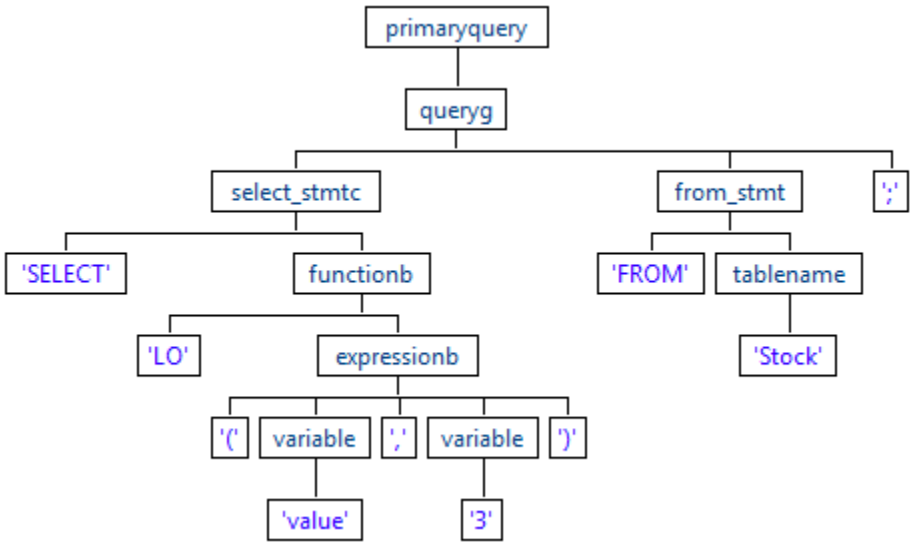

Fig.2

Since the parser displays the components of the query in the form of an abstract syntax tree,

the tree is walked so that the query can output the correct information to the user. Other examples of queries that are parsed using the parser are SELECT HI(Stockamount, 3) FROM Stock;, SELECT AVGSUR(Stockamount,3) WHERE name = Google; The last component of E-Quote is the user interface which is constructed in Java. Java is a popular and flexible object-oriented programming language and was chosen to be used to create the user interface. Since ANTLR produces the parser rules written in Java language, it makes sense that the user interface is constructed in Java. The user interface is constructed using Java swing and GUI. In the Java code describing the user interface, the database is started and the user is prompted to input a query into E-Quote. Once the user inputs the query, the query is parsed using the parser created from ANTLR and outputs the result to the user in a textbox format.

Outline of E-Quote:

1. The user is prompted to login into E-Quote.
2. The user inputs a query into E-Quote. For example, (SELECT LO(Stockamount, 3) FROM Stock).
3. Once the user presses the Enter button, the database is run and the input is fetched from the database to compute the correct result.
4. The correct result is displayed to the user.
5. The user can input another query or exit E-Quote.

Fig.2

Part III: Chronological Account of E-Quote

September 2013: The key components were using SQL as the query language and using a parser. The output at this time was to have graphs displaying the stock-market data.

October-November 2013: the query language was defined with the functions it currently possesses. The construction of the parser begins. It was suggested that research about ANTLR and a SQL grammar written in ANTLR should be done and the construction of the basics of the user interface are in motion.

November – December 2013: A basic design of the user interface has been designed using a Java servlet. The design of a SQL grammar has been attempted, but it has been too complex to use since it had too many parser rules.

January-February 2014: It was suggested that the tool be constructed using Java GUI. The construction of E-Quote as a Java GUI commences.

February- March 2014: The reconstruction of the parser is done. The grammar is modified with simpler rules and constructed using ANTLR. It parses the queries of E-Quote's language correctly and displays the correct abstract syntax trees.

March-April 2014: The implementation of using a JFreeChart with E-Quote is attempted, but the JFreeChart but does not show the correct output.

April 2014-May 2014: It was suggested that there should be some modifications made to E-Quote. It was suggested that some components of the Java code for E-Quote be reconstructed. The parser is implemented into the Java code as it was back in March. It is also suggested that the code for parser walking the tree be rewritten and that the results of the functions be modified.

Technical Challenges: As associated with designing a high-level and complex project, there were some challenges. Learning ANTLR for the first time was a challenge. ANTLR is a useful tool but can appear complex to a newbie. Also, finding a SQL parser written in ANTLR to use as a reference was a little difficult since there are not many SQL parsers written in ANTLR that have designed in an accurate fashion. Walking the parse tree that was created from the parser was also challenging. It was determined that the tree had to be walked in the correct fashion to help output the correct information. Also, getting the JFreeChart to show correct output was a challenge.

Lessons: Time management, hard work and patience are keys to designing a successful design project.

Never underestimate the amount of time you have. You never know when you may run into something or something may go wrong to cause delay. For this project, it did take time to research and learn ANTLR, which is interesting to learn. As mentioned in the previous section, ANTLR can appear challenging to someone who is just learning it, but it is a really neat tool to use since lets anyone create their own language to design a project. ANTLR is flexible to use with high level languages such as Java and C.

Nothing is done without hard work. Researching about ANTLR and constructing Java code from scratch was hard work and took a significant amount of time. Ultimately, to do anything successfully, it does take patience. It's okay to make a mistake, but patience is the key. Learning something new and programming and coding took time and patience for working on this project. If E-Quote could be done all over again, I wish I made more time for constructing more detailed output.

Part IV: Speculations about the Future:

Stock-market analyzers are popular tools in this day and age. If there was more time and resources, E-Quote would be improved significantly. E-Quote would display better and more detailed output in graph format. Also, more functions would be added to the language of E-Quote. Finally, if more time was permitted E-Quote would possess better means of accessibility via Internet and mobile app.