

Summer Independent Study

Warehousing Equity Data

Michael Rechenthin
Loyola University Chicago, Chicago, IL 60640
mrechenthin@gmail.com

July 31, 2009

Abstract

This paper presents my summer's work on storing equity quotes from a free service provider (*Google* and *Yahoo Finances*) to a relational database. An easy-to-use graphical user interface was created to facilitate additions and deletions of equity symbols to the database along with support features that keep the list of S&P 500 symbols current.

A webpage was also created with the help of an applet to allow a user to easily query the database. The user inputs a symbol along with a specified time span and the program creates both a chart of the stock's historical prices and a list of the thirty stocks that most closely correlate with the inputted stock.

Finally, this paper will cover the project management aspects of my work and will identify future areas of research related to this project.

Contents

I	Overview of the Project	3
1	Introduction	3
1.1	Motivation and Objectives	3
2	Data Warehousing	3
2.1	Database	3
2.2	Retrieving Data	4
2.2.1	Quotes Providers	4
2.2.2	Differences Between <i>Google</i> and <i>Yahoo</i> Data	4
2.2.3	Stock Splits Affecting Historical Equity Prices	5
2.2.4	S&P 500 List Provider	6
3	Program overview	6
3.1	Screens	6
3.1.1	Main Screen (Figure 5)	6
3.1.2	Maintenance Screen (Figure 6)	7
3.1.3	Price Analyzer Screen (Figure 7)	7
3.2	Stored Procedures in the Database	7
3.3	Website	9
3.3.1	Purpose of the Website	9
3.3.2	Applet	9
II	Future Work	11
4	Working with the Data	11
4.1	Data import via ODBC	11
4.2	Clementine	11
4.3	Weka	12
III	Project Management	13
5	Introduction	13
6	Organizing Work	13
6.1	Work Breakdown Structures and Scheduling	13
6.2	UML Class Diagram	13
7	Code Repository	13
8	Handling Changes	13
9	Project Closeout and Evaluation	13
IV	Appendix	15

Part I

Overview of the Project

1 Introduction

1.1 Motivation and Objectives

As a trader on the floor of the Chicago Stock Exchange for eight years, my interests lie in equity markets and the development of trading systems. To analyze and mine the market data first requires a method to efficiently retrieve and store the data. This paper explains my summer's research for building a warehouse for capturing equity data from *Google* and *Yahoo Finance*. The three main objectives for my project were:

1. Create a program that retrieves historical market data for all 500 stocks in the S&P 500 and store them in a relational database
2. Data mine for stocks that best correlate with a user inputted stock symbol
3. Create a web interface for displaying equity prices and allow the user to query the database for stocks that most closely correlate to the inputted symbol.

I will be starting a doctoral program in Management Science at the University of Iowa in the Fall and I plan to use the program I develop to assist me in my future research by giving me easier access to market data.

2 Data Warehousing

2.1 Database

An efficient medium is needed to store the large quantity of information. *MySQL*, a relational database management system, was used for this project for several reasons:

1. Free for non-commercial use under a GNU General Public License.
2. With over 6 million installations, it is widely used and fully documented
3. Full support for stored procedures as of version 5.0

To allow a database to be connected to data mining software, such as *Clementine* and *SAS*, a relational database management system (RDBMS) was chosen over a object oriented management systems (ODBMS). ODBMS's have advantages; namely the ability to keep complex entities as objects as opposed to the use of foreign tables. A second advantage of ODBMS's is that a programmer of an object oriented language such as Java can continue to program in an object-oriented style with the database, as opposed to switching to a procedural based style when working with the database, as is done with RDBMS via the query language. Despite these advantages, the ability to link up to data mining software was still the most important factor when choosing the RDBMS over an ODBMS [15, 14].

Figure 1 shows the layout of the tables within the *MySQL* database. It can be seen that *gquotes_daily* references *symbol_list*. Likewise, *symbol_list* references *symbol_gics*. A screen capture taken in *MySQL Query Browser* can be seen in Figure 2.

Table 1 explains the table and its fields in greater detail. Also, the process for creating the database and table can be seen in the appendix. Code for creating the entire list of stored procedures can be found in the repository on *Google Code* (see Section 7).

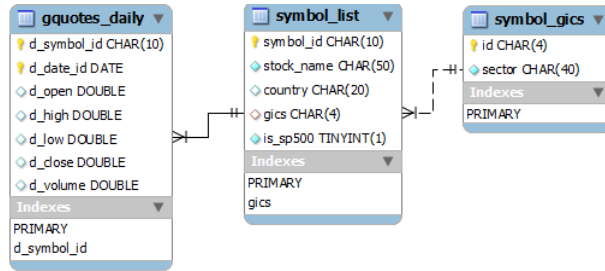


Figure 1: Database design

gquotes_daily							symbol_list						symbol_gics		
d_symbol_id	d_date_id	d_open	d_high	d_low	d_close	d_volume	symbol_id	stock_name	country	gics	is_sp500		id	sector	
IBM	2000-01-05	112.94	119.75	112.12	116	12717600	HSY	Hershey Foods Corp	USA	30	1		10	Energy	
IBM	2000-01-06	118	118.94	114.62	114.62	7937100	HUM	Humana Inc	USA	35	1		15	Materials	
IBM	2000-01-07	117.25	117.94	110.62	113.31	11803600	IBM	Intl Business Machines Corp	USA	45	1		20	Industrials	
IBM	2000-01-10	117.25	119.38	115.38	118.44	8532600	ICE	IntercontinentalExchange	USA	40	1		25	Consumer Discretionary	
IBM	2000-01-11	117.88	121.12	116.62	118.62	7825000	IFF	Intl Flavors & Fragrances	USA	15	1		30	Consumer Staples	
IBM	2000-01-12	119.62	122	118.25	119.62	6791600	IGT	Intl Game Technology	USA	25	1		35	Health Care	
							INTC	Intel Corp	USA	45	1		40	Financials	
													45	Information Technology	
													50	Telecommunication Services	
													55	Utilities	

Figure 2: Screen capture of tables

2.2 Retrieving Data

2.2.1 Quotes Providers¹

Google Finance was used as the provider of quotes. After sending *Google* a request for data via a URL, the data is retrieved as a comma separated value (CSV). A sample of code follows:

```
// the abbreviated month is the first three letters. Example: Jan, Feb, Mar ...
String GoogleURL = "http://www.google.com/finance/historical?q=" + symbol +
    "&startdate=" +
    monthAbbreviation(startMonth)+ "+" + startDay + "%2C" + startYear +
    "&enddate=" +
    monthAbbreviation(endMonth) + "+" + endDay + "%2C" + endYear +
    "&output=csv";
```

A *Yahoo Finance* connecting class was also created to provide a secondary connection to quotes in the event that *Google* “times-out” or blocks access. This class can be also used in the future to check the accuracy of quotes retrieved from *Google Finance*. The URL is similar to the URL used by *Google*:

```
String yahooURL = "http://ichart.finance.yahoo.com/table.csv?" +
    "s=" + symbol + "&" +
    "a=" + startMonth + "&b=" + startDay + "&c=" + startYear + "&" +
    "d=" + endMonth + "&e=" + endDay + "&f=" + endYear + "&g=d";
```

2.2.2 Differences Between *Google* and *Yahoo* Data

Google and *Yahoo* have slight differences in their equity data provided (see Figures 3 and 4). The first difference is the data field. This required a slightly different parsing technique to normalize the data and then store it in the database. The second difference is that *Yahoo Finance* has a seventh column named

¹In the proposal, *Interactive Brokers (IB)* was chosen to be the provider of quotes. However, *IB* had limitations of 60 historical data request in any ten minute period, whereas *Google* and *Yahoo Finance* do not.

Table 1: Table description

Field	Description
GQUOTES_DAILY.d_symbol_id	stock symbol
GQUOTES_DAILY.d_date_id	date
GQUOTES_DAILY.d_open	opening price
GQUOTES_DAILY.d_high	high daily price
GQUOTES_DAILY.d_low	low daily price
GQUOTES_DAILY.d_close	closing daily price (official)
GQUOTES_DAILY.d_volume	volume daily price
SYMBOL_LIST.symbol_id	stock symbol
SYMBOL_LIST.stock_name	stock name
SYMBOL_LIST.country	stock country of origin
SYMBOL_LIST.gics	stock sector (number)
SYMBOL_LIST.is_sp500	part of the S&P 500 (0='no', 1='yes')
SYMBOL_GICS.id	stock sector (number)
SYMBOL_GICS.sector	stock sector (full description)

Adjusted Closing Price. This column reflects the adjusted price of the security after dividends have been accounted. *Google* does not adjust prices due to dividends.

To understand this adjusting of prices due to dividends, it is helpful to consider the following example: If a company issues a dividend of \$0.50, the stock will adjust downward on the day the dividend is paid to reflect the price paid to the holder of the stock. Although the stock has dropped in price, the shareholder is compensated by the dividend. Since this database will be used for machine learning, the historical prices should be adjusted for this dividend discrepancy. Note: Not all stocks pay dividends, so this would not be a problem for all symbols, and the differences would amount to less than a few percent to those stocks that do; however, looking into this discrepancy and the most efficient way to make adjustments in the database will have to be a future project.

It should be noted that both *Google* and *Yahoo* adjust stock prices due to stock splits. More information on stocks splits and its effect on the historical data can be found in section 2.2.3.

```
Date,Open,High,Low,Close,Volume
15-Jul-09,104.75,107.22,104.60,107.22,8707266
14-Jul-09,103.42,103.62,102.52,103.25,5416082
13-Jul-09,101.28,103.65,100.19,103.62,9494417
10-Jul-09,100.97,101.72,99.80,100.83,7470894
9-Jul-09,101.12,102.78,100.85,102.08,6153127
8-Jul-09,100.29,101.17,99.50,100.68,7380719
```

Figure 3: CSV formatted data from *Google*

```
Date,Open,High,Low,Close,Volume,Adj Close
2009-07-15,104.75,107.22,104.60,107.22,8699100,107.22
2009-07-14,103.42,103.62,102.52,103.25,5413500,103.25
2009-07-13,101.28,103.65,100.19,103.62,9494300,103.62
2009-07-10,100.97,101.72,99.80,100.83,7465900,100.83
2009-07-09,101.12,102.78,100.85,102.08,6153200,102.08
2009-07-08,100.29,101.17,99.50,100.68,7380800,100.68
```

Figure 4: CSV formatted data from *Yahoo*

2.2.3 Stock Splits Affecting Historical Equity Prices

A stock split is an action on behalf of the corporation that increases the number of the corporations outstanding shares by dividing its shares, which in turn diminishes its price. For example, with a 2 for 1 stock split, the holder of the stock receives two shares for every one that is currently held. The stock price is then divided by two. The amount invested remains the same. *Reverse stocks splits* are the same, only reverse.

The reasons for stocks splits are numerous. Two important reasons are:

1. The New York Stock Exchange requires stocks to remain above \$1. *Reverse stock splits* are one way for companies to stay above that level.
2. Psychological reasons. As the price of a stock get higher and higher, some investors may feel that the price is unaffordable. Splitting the stock brings the stock down the an “affordable” level.

2.2.4 S&P 500 List Provider

The Standard and Poors 500 is not a static list of stocks. Additions and deletions of stocks from the lists occur regularly, about 5% per year. A current listing of all stocks in the S&P 500 are retrieved via a URL request to the Standard and Poors website. An excerpt of code follows:

```
String today = DDate.today().day() - daysBack + "-" +  
              DDate.today().monthToEnglish() + "-" +  
              DDate.today().year();  
  
URL url = new URL("http://www2.standardandpoors.com/servlet/" +  
                  "Satellite?pagename=spcom/page/download&sectorid" +  
                  "=%20%3E%20%2700%27&itemname=%3E=%20%271%27&dt=" +  
                  today + "&indexcode=500");
```

3 Program overview

3.1 Screens

To facilitate the downloading of quotes to the *MySQL* database, an easy-to-use graphical user interface was created (see Figures 5, 6, and 7). The sole purpose of the program is to update and download current quotes to the database.

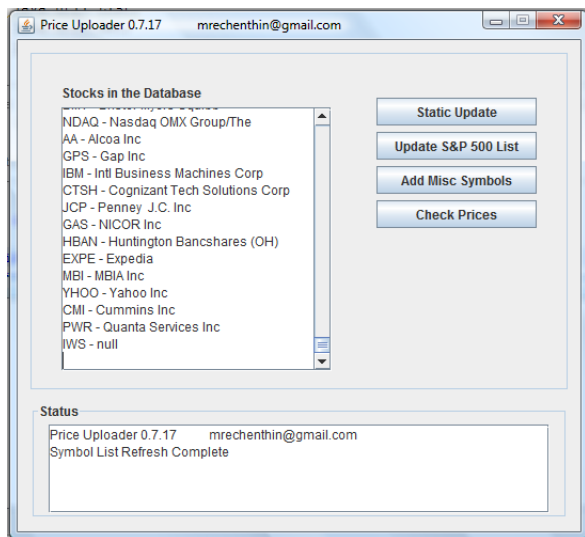


Figure 5: GUI - Main Screen

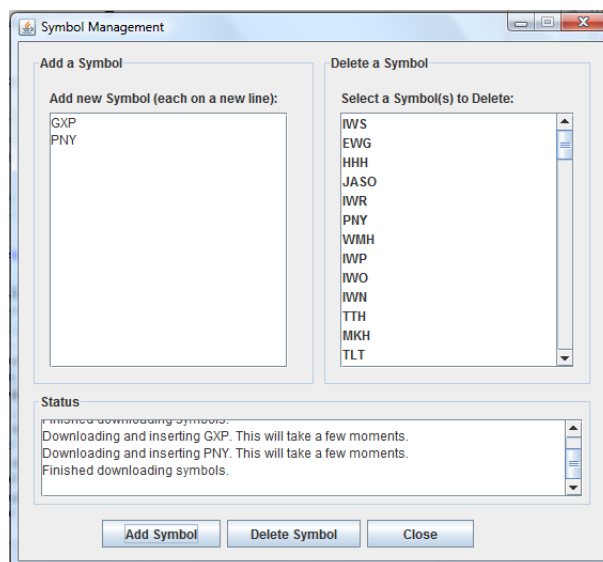


Figure 6: GUI - Maintenance Screen

3.1.1 Main Screen (Figure 5)

1. **Static Update Button** - This creates a separate thread, which then makes a call to the database for each symbol. A query is made to find the last available date in the table. Thereafter, a call is then made to the quote provider (see section 2.2.1) for each missing quote. Lastly, the data is stored using *insert* statements to the database via the *gquotes_daily* table.
2. **Update S&P 500 List Button** - A separate thread is created, which makes a call to the *Standard and Poors* website (see section 2.2.4) upon which an updated list of stocks of the S&P 500 is stored in the database via the *symbol_list* table.

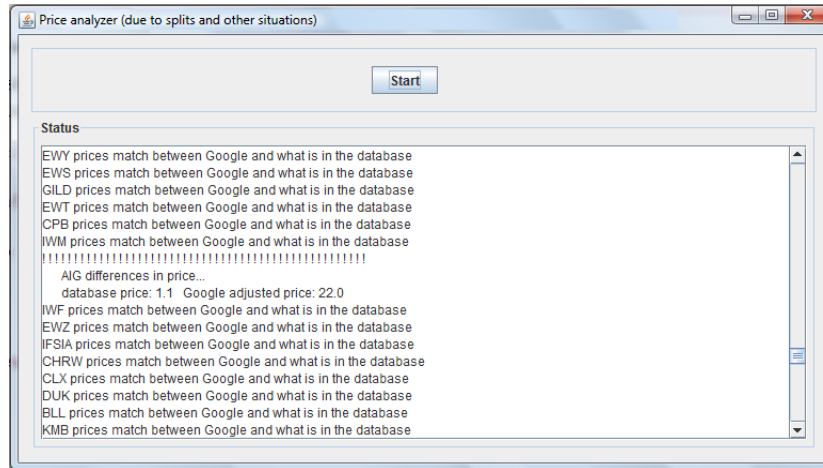


Figure 7: GUI - Price Analyzer Screen

3. **Add Misc Symbols Button** - This button brings up the Symbol Management Screen (figure 6). This screen allows the user to add and delete stocks other than the S&P 500 stocks to the database. See section 3.1.2 for more information.
4. **Check Prices Button** - This button brings up the Price Analyzer Screen (figure 7). This screens allows the user to check the prices in the database against prices provided by *Google* and *Yahoo Finances* for changes in data. The change would most likely be due to stock splits. See section 3.1.3 for more information.

3.1.2 Maintenance Screen (Figure 6)

This interface allows the user to add stocks to the database outside of the stocks in the S&P 500. Many widely watched stocks are outside of this popular index, and this screen allows one to add any stock whose symbol is either on *Yahoo Finance* or *Google Finance*. The button are explained below:

1. **Add Symbol Button**- Enter stocks in the text area, each on a new line, and then click the *Add Symbol* button. The wide text area, allows for copy and pasting of symbols from a spreadsheet.
2. **Delete Symbol Button** - Select the stocks wanted to delete and hit this button. The symbols selected will be deleted from the *gquotes_daily* and *symbol_list* tables will be deleted. See figure 1 for more information on the database.

3.1.3 Price Analyzer Screen (Figure 7)

This screen allows the user to check the accuracy of the *gquotes_daily* table. When the user clicks the start button, the program request a single price from 100 days back from the *Google/Yahoo Finance* quote provider. If this price is different then the price for that specific day in the table, a flag is raised and the user is notified via the text area. One possible reason the quote is different in the database then the quote provided by *Google/Yahoo Finance* is a stock split occurred, which automatically adjusted the price. More information on stock splits and its affect on historical quotes can be found in section 2.2.3. The automatic adjustment of prices in the database due to stock splits will have to be a future project.

3.2 Stored Procedures in the Database

In keeping with the requirements of the proposal, an outcome of the project was to provide a method for using the stored data to mine for stocks that showed high correlation with an inputed stock.

Stored database procedures and functions were chosen to increase the portability of the program by uncoupling the *data mining logic* from the *data inputting logic*. An advantage of stored procedures is they can reduce network traffic since the program can work on the data within the server, rather than having to transfer the data across the network. Figure 8 is an excellent example of the time taken to find prime number in various languages [17, 12].

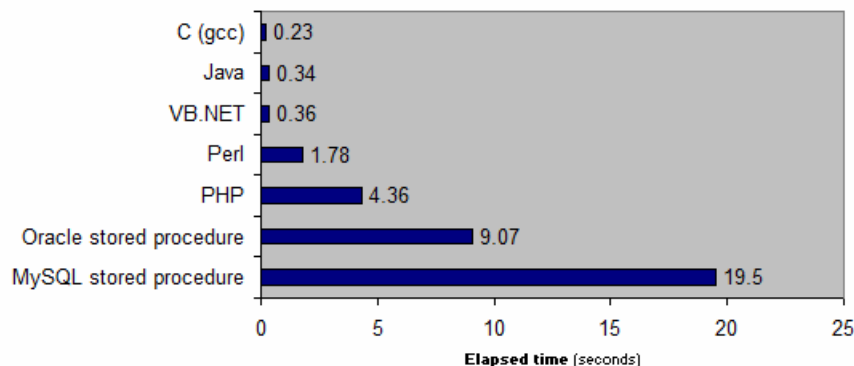


Figure 8: Speed at finding prime numbers in various languages[12]

In my project, the data is on the *localhost* rather than a *remote host*, so my current design is slower than having the logic computed by within Java. However, the design of adding logic to the database allows my program to grow in the future.

Several of my stored procedures, including the ability to find the 30 most strongly correlated stocks, can be seen in the appendix. Figure 9 gives a screenshot of running this particular stored procedure in *MySQL Query Browser*.

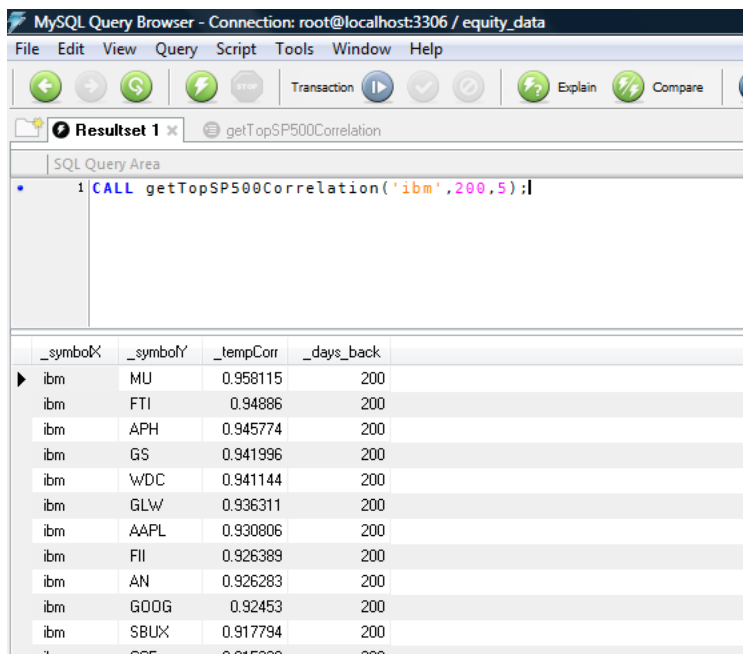


Figure 9: Running the Correlation Stored Procedure in *MySQL Query Browser*

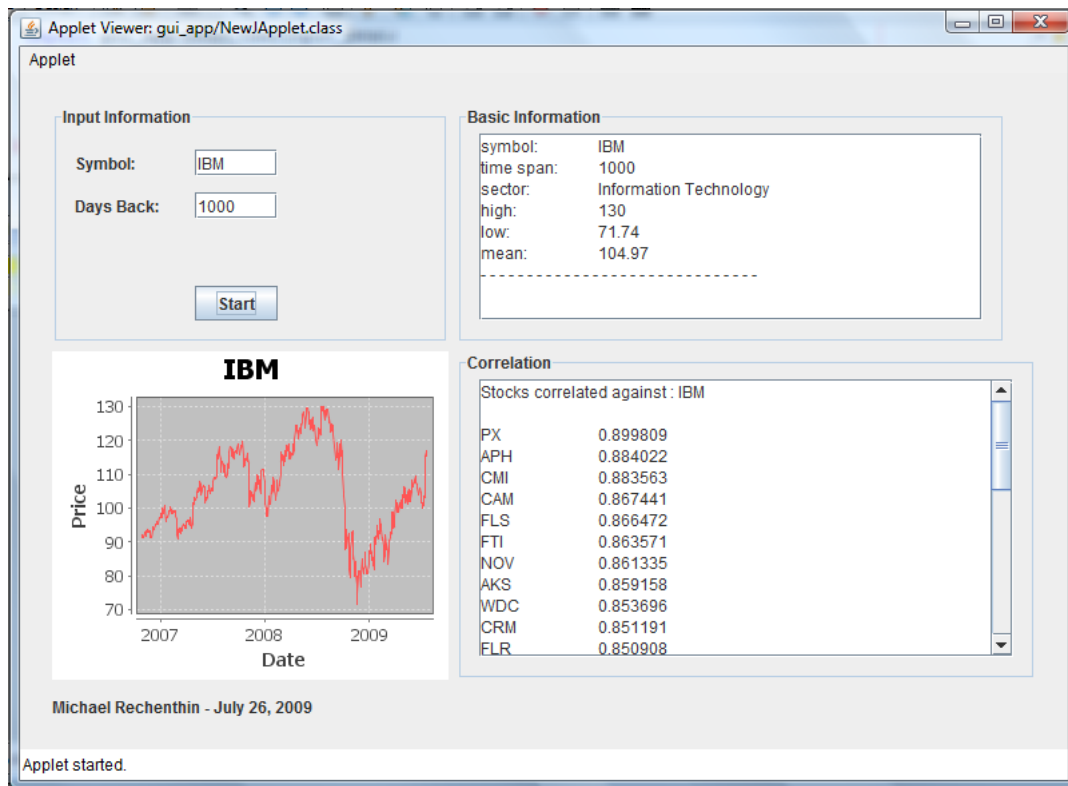


Figure 10: Running the applet in the viewer

3.3 Website

3.3.1 Purpose of the Website

An addition to the project that was not included in the original proposal was a web user interface. The purpose of this interface is to allow users to query the database for simple statistical information on a particular equity symbol over a particular time span. When the user queries a particular stock, a chart of the stock's price history is created along with 30 stocks within the Standard and Poors 500 that most closely correlate over the user entered time span².

3.3.2 Applet

Figure 10 displays the applet that was created for this project. A description of the applet's panes are below:

1. **Input Information pane** - This is where the user inputs the equity symbol and time span over which to do the calculations. Note that the time span is in real terms and not just trade days. For example, to retrieve information for exactly one year's worth of data, the user should input 365 days, not the number of trading days.
2. **Chart pane** - This displays the chart for the specified symbol over the specified days. The JFreeChart library was used to create the chart[10]. A nice feature of the chart is it can zoom in and out on the data[10].
3. **Basic Information** - This pane displays the symbol and sector along with the high, low, and mean over the user specified days. To create this data, a stored procedure is called that queries the database.

²I limited the search to only stocks within the S&P 500 because of the reliability of high volumes traded per day. In the future, I could possibly expand this to include stocks outside the S&P as long as there is ample volume to deem the stock a reliable one to trade.

4. **Correlation pane** - This pane displays the top 30 most correlated stocks in the S&P 500 against the user inputted symbol. This information is retrieved by calling a stored procedure in the database. More information of the use of stored procedures in the program can be read in section 3.2.

Launching the Applet A nice feature of applets is they can be deployed using *Java Web Start*. This allows users of the application to download and launch the application from anywhere by clicking on a link in the web browser. The *Java Web Start* works by creating a JNLP file. This JNLP file is used to launch the application. Most IDE's, such as *Netbeans*, can assist in creating the JNLP.

A segment of my *index.html* file that was used to launch the JNLP can be seen below. Notice that the *archive* property points to libraries that are needed to run the file. For further analysis, a description of the directory in which my files are located can be seen in figure 11.

```
<body>
  <applet archive="classes/lib/Applet_GUI.jar,
               classes/lib/jfreechart-1.0.13.jar,
               classes/lib/jcommon-1.0.16.jar,
               classes/lib/mysql-connector-java-5.1.6-bin.jar"
          width="740" height="525">
    <param name="jnlp_href" value="launch.jnlp"/>
  </applet>
</body>
```

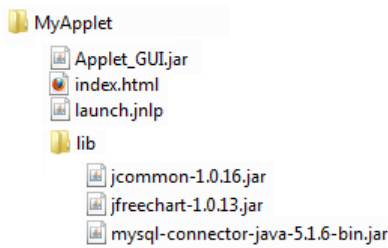


Figure 11: Applet Directory Description

Part II

Future Work

4 Working with the Data

4.1 Data import via ODBC

ODBC (Open DataBase Connectivity) provides a standard software API for using database management systems. It is a group of software methods that enable a client to interact with a database server. The designers of this de facto standard aimed to make it independent of programming languages, database systems, and operating systems [24]. ODBC acts as an intermediate layer between client and server, so the client “talks” to ODBC rather than accessing the server directly [6]. Once a connection is established with the ODBC, data is retrieved via a query.

The data in this project is stored in a *MySQL* database. Native support is not often available for *MySQL* within popular statistical and data mining software. These software packages however, nearly always support ODBC. This allows our data to be used with these programs even if native support is not offered.

4.2 Clementine

Clementine is a powerful program created by SPSS used for data mining. It can be used to quickly develop predictive models and deploy them into business operations to improve decision making.

An advantage to *Clementine* is it attempts to be user friendly with a visual interface which allows a user to focus more on the data and the task at hand, rather than the logistics of working with a low level programming language [18, 15].

Clementine does not have native support for *MySQL*, but it does have the ability to use ODBC. Figure 12 is a screenshot showing *Clementine* connecting to the database via the ODBC.

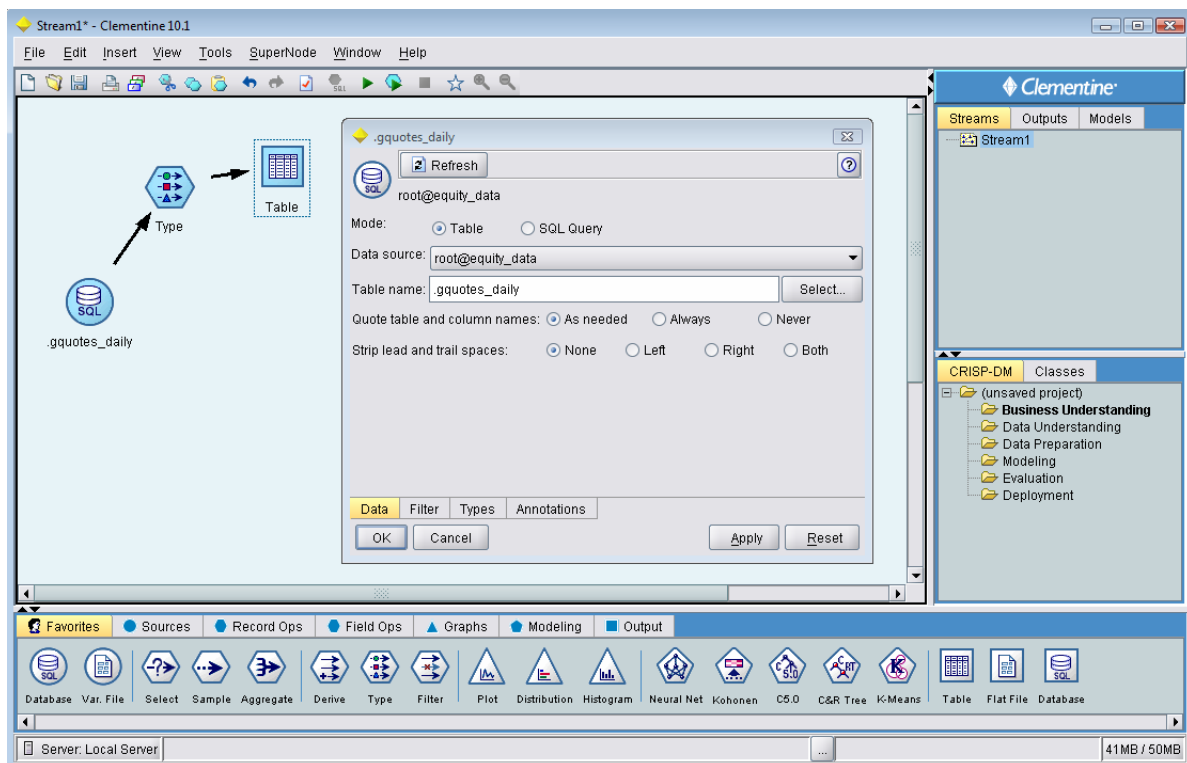


Figure 12: *Clementine* connecting to the ODBC

4.3 Weka

Waikato Environment for Knowledge Analysis (Weka) is a suite of free machine learning algorithms developed at the University of Waikato. Connectivity is achieved by using the ODBC. The suite includes a graphical user interface along with an API for programming through Java. This is a programming interface I would enjoy exploring further in the future.

Part III

Project Management

5 Introduction

Planning is important in any large project. To ensure that the project kept on course, the project was approached as scientifically as possible.

6 Organizing Work

6.1 Work Breakdown Structures and Scheduling

A work breakdown structure was used to subdivide the project into smaller, more manageable elements. This allowed for the project to be managed more accurately and ensured that the project was finished by the deadline. The work breakdown structure and Gantt chart can be seen in the appendix.

6.2 UML Class Diagram

Once an initial WBS and a time-line was in place, the scope and complexity of the program could be determined. Next, a basic class diagram was developed. The class diagram helped by centralizing thoughts and helped by determining areas which could be problematic, such as coupling problems with the database. The UML did change when work began and as problems arose, but the basic structure stayed the same. The final UML Class diagram can be seen in the appendix.

7 Code Repository

Google Code was used as the final code repository for this project. *Mercurial* was the version control software used. Although not as user friendly as *Subversion*, *Mercurial's* list of features and its distributedness made it ideal for this project.

The address is as follows:

<http://code.google.com/p/mrechenthin-database-1/>

8 Handling Changes

This project had many changes. One of the first changes was converting the program from *Interactive Brokers* to *Google* and *Yahoo Finances*. Although *Interactive Brokers* allowed for more choices as far as the scope of data available, the service was not meant as a supplier of quotes for a freeloading student. Almost immediately my IP got blocked for downloading too many quotes. I made the decision to convert the program to *Google Finance*. So as to not repeat the incident, I also created a *Yahoo Finance* class as a backup connectivity solution.

Another change to the original project proposal was the addition of the web front-end to display the data. This added considerable time to the project.

9 Project Closeout and Evaluation

I enjoyed working on this project and certainly expanded my knowledge of databases and using stored procedures. I learned how to set-up a database on my personal computer and I gained experience working on a large-scale project that took a considerable amount time and effort. I feel accomplished having finished

another successful project while working on my Masters at Loyola. Furthermore, the data that I retrieved on equities will be useful during my studies at the University of Iowa doctoral program.

I also overcame obstacles and setbacks when working on this project. I spend two weeks working on a server-side ajax-based framework for creating a web page, Echo2[13]. I came away with a better understanding of how web pages work, but also a strong dislike of most web frameworks! I later switched to an applet, and in two days I did what I could not accomplish in two weeks.

Overall, this project exceeded my expectations. I became more proficient at researching, more skilled at programming and most importantly, more confident in my abilities that will carry me into my further studies.

Part IV

Appendix

```
--
-- Michael Rechenthin
-- July 2009
--
-- Step 1 of the schema/table creation
-- drops all the tables and uses the 'starter'
-- data to help speed up the process...so the
-- downloading doesn't take as long.
--
```

```
CREATE DATABASE `equity_data` /*!40100 DEFAULT CHARACTER SET latin1 */;
```

```
-- DROP TABLES
```

```
DROP TABLE IF EXISTS `equity_data`.`gquotes_daily`;
```

```
DROP TABLE IF EXISTS `equity_data`.`symbol_list`;
```

```
DROP TABLE IF EXISTS `equity_data`.`symbol_gics`;
```

```
-- CREATE TABLES
```

```
CREATE TABLE `equity_data`.`gquotes_daily` (
  `d_symbol_id` char(10) NOT NULL,
  `d_date_id` date NOT NULL,
  `d_open` double DEFAULT NULL,
  `d_high` double DEFAULT NULL,
  `d_low` double DEFAULT NULL,
  `d_close` double DEFAULT NULL,
  `d_volume` double DEFAULT NULL,
  PRIMARY KEY (`d_symbol_id`,`d_date_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `equity_data`.`symbol_gics` (
  `id` char(4) NOT NULL,
  `sector` char(40) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `equity_data`.`symbol_list` (
  `symbol_id` char(10) NOT NULL,
  `stock_name` char(50) NOT NULL,
  `country` char(20) DEFAULT NULL,
  `gics` char(4) DEFAULT NULL,
  `is_sp500` tinyint(1) unsigned NOT NULL,
  PRIMARY KEY (`symbol_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- NOW LOAD THE *starter* DATA
```

```
load data infile 'gquotes_daily.csv' into table gquotes_daily fields terminated by ',';
```

```
load data infile 'symbol_gics.csv' into table symbol_gics fields terminated by ',';
```

```
load data infile 'symbol_list.csv' into table symbol_list fields terminated by ',';
```



```
-- -- -- -- --
-- Michael Rechenthin
-- July 2009
-- Step 2 of the schema/table creation.
-- we now add the foreign constraints
-- -- -- -- --

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

-- adding the foreign references...GQUOTES_DAILY
ALTER TABLE `equity_data`.`gquotes_daily`
  ADD CONSTRAINT `d_symbol_id`
  FOREIGN KEY (`d_symbol_id` )
  REFERENCES `equity_data`.`symbol_list` (`symbol_id` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
, ADD INDEX `d_symbol_id` (`d_symbol_id` ASC) ;

-- adding the foreign references...SYMBOL_LIST
ALTER TABLE `equity_data`.`symbol_list`
  ADD CONSTRAINT `gics`
  FOREIGN KEY (`gics` )
  REFERENCES `equity_data`.`symbol_gics` (`id` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
, ADD INDEX `gics` (`gics` ASC) ;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

DELIMITER \$\$

```

DROP FUNCTION IF EXISTS `equity_data`.`getCorrelation`$$
CREATE DEFINER=`root`@`localhost` FUNCTION `equity_data`.`getCorrelation`(symbolX char(10),
                                symbolY char(10),
                                ays_back int) RETURNS float
BEGIN
-- -----
-- Michael Rechenthin
-- June 2009
--
-- creation of a stored procedure - getCorrelation.
-- finds the correlation of two symbols,
--   symbolX & symbolY, going back
--   days_back number of days
-- Returns the pearsons correlation coefficient
-- -----
declare num          int      default 0; -- number of rows
declare ccloseX      float    default 0; -- the closing price
declare ccloseY      float    default 0; -- the closing price

declare std_devX     float    default 0; -- standard deviation
declare std_devY     float    default 0; -- standard deviation

declare meanX        float    default 0; -- mean
declare meanY        float    default 0; -- mean

declare popcovXY     float    default 0; -- population covariance
-- declare popcovX     float    default 0; -- population covariance
-- declare popcovY     float    default 0; -- population covariance

declare corr         float    default 0; -- pearsons correlation coefficient

-- cursor via X
declare cur_ccloseX cursor for
  select d_close from gquotes_daily
  where d_symbol_id=symbolX and
  d_date_id > date_sub(curdate(),interval days_back day);

-- cursor via Y
declare cur_ccloseY cursor for
  select d_close from gquotes_daily
  where d_symbol_id=symbolY and
  d_date_id > date_sub(curdate(),interval days_back day);

declare continue handler for not found set ccloseX=-1;

```

```

-- initializing
-- -----

-- get the mean & standard deviation for X going back some number of days
select AVG(d_close), STDDEV_POP(d_close) from gquotes_daily
  where d_symbol_id=symbolX and
         d_date_id > date_sub(curdate(),interval days_back day)
into meanX, std_devX;

-- get the mean & standard deviation for Y going back some number of days
select AVG(d_close), STDDEV_POP(d_close) from gquotes_daily
  where d_symbol_id=symbolY and
         d_date_id > date_sub(curdate(),interval days_back day)
into meanY, std_devY;

OPEN cur_ccloseX;
OPEN cur_ccloseY;
  aloop:LOOP
    fetch cur_ccloseX into ccloseX;
    fetch cur_ccloseY into ccloseY;

    if ccloseX=-1 then leave aloop; end if;

    -- to get the true population covariance of x & y we need to divide it by num
    set popcovXY = popcovXY + (ccloseX - meanX)*(ccloseY - meanY);

    set num=num+1; -- keep track of the total count/rows

  END LOOP aloop;
CLOSE cur_ccloseX;
CLOSE cur_ccloseY;

set popcovXY = popcovXY/num; -- computing the population covariance

set corr = popcovXY / (std_devX*std_devY); -- computing the correlation

return(corr);
END;

$$

DELIMITER ;

```

DELIMITER \$\$

```

DROP PROCEDURE IF EXISTS `equity_data`.`getTopSP500Correlation`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `equity_data`.`getTopSP500Correlation`(
    IN symbolX char(10),
    IN days_back int,
    IN topNum int
)
BEGIN
    -- - - - - -
    -- Michael Rechenthin
    -- June 2009
    --
    -- Creation of a stored procedure - getTop500Correlation
    -- Finds the top number (topNum) of correlations
    -- between the stock (symbolX) and the
    -- rest of the stocks in the s&p 500 table
    -- (sp500_list)...going back (days_back)
    -- number of days.
    --
    -- To do this I am going to create a temporary
    -- table, and store the symbols along with
    -- the correlation coefficient.
    --
    -- * notice that we are not including the
    -- stocks that are not in the s&p500.
    -- We want to only include stocks that
    -- are actively traded and s&p 500
    -- stocks have a strong likelihood of
    -- having
    -- - - - - -
    declare symbolY char(10); -- symbol y...the temp to compare symbolX with
    declare tempCorr float default 0; -- the temp correlation for the current symbol
    declare tempCorrAbs float default 0; -- the temp absolute value of the correlation for the
the current symbol
    declare tempHigh float default 0; -- the temp high for the current symbol
    -- declare tempY char(10);
    declare num int default 0; -- counter

    -- create cursor...no need to include the stock we are requesting for...
    -- and only include the stocks that are market as being in the s&p 500 (which is market
    -- with a '1')
    declare cur_symbolList cursor for
        select symbol_id from symbol_list where symbol_id<>symbolX and is_sp500='1';

    declare continue handler for not found set symbolY='quit';

```

```

-- create a temporary table...using this as storage for the top numbers
-- of symbols that correlate with symbolX
DROP TEMPORARY TABLE if exists tempTopCorrelation;
CREATE temporary TABLE tempTopCorrelation (
    _symbolY      char(10) not null,      -- symbolY
    _symbolX      char(10) not null,      -- symbolX
    _tempCorr     float,                  -- the pearson correlation coefficient
    _tempCorrAbs  float,                  -- the absolute value of the pearson correlation coefficient
    _days_back   int,                   -- the days back to retrieve
    primary key (_symbolY)
);

open cur_symbolList;
symbolListLoop:LOOP                -- START OF LOOP
    fetch cur_symbolList into symbolY;

    if symbolY='quit' then leave symbolListLoop; end if;

    set tempCorr = getCorrelation(symbolX, symbolY, days_back);
    set tempCorrAbs = ABS(tempCorr);
    -- insert the appropriate information into the table
    INSERT INTO tempTopCorrelation VALUES (symbolY, symbolX, tempCorr, tempCorrAbs, days_back);

    set num = num+1; -- advance counter
END LOOP symbolListLoop;          -- END OF LOOP
close cur_symbolList;

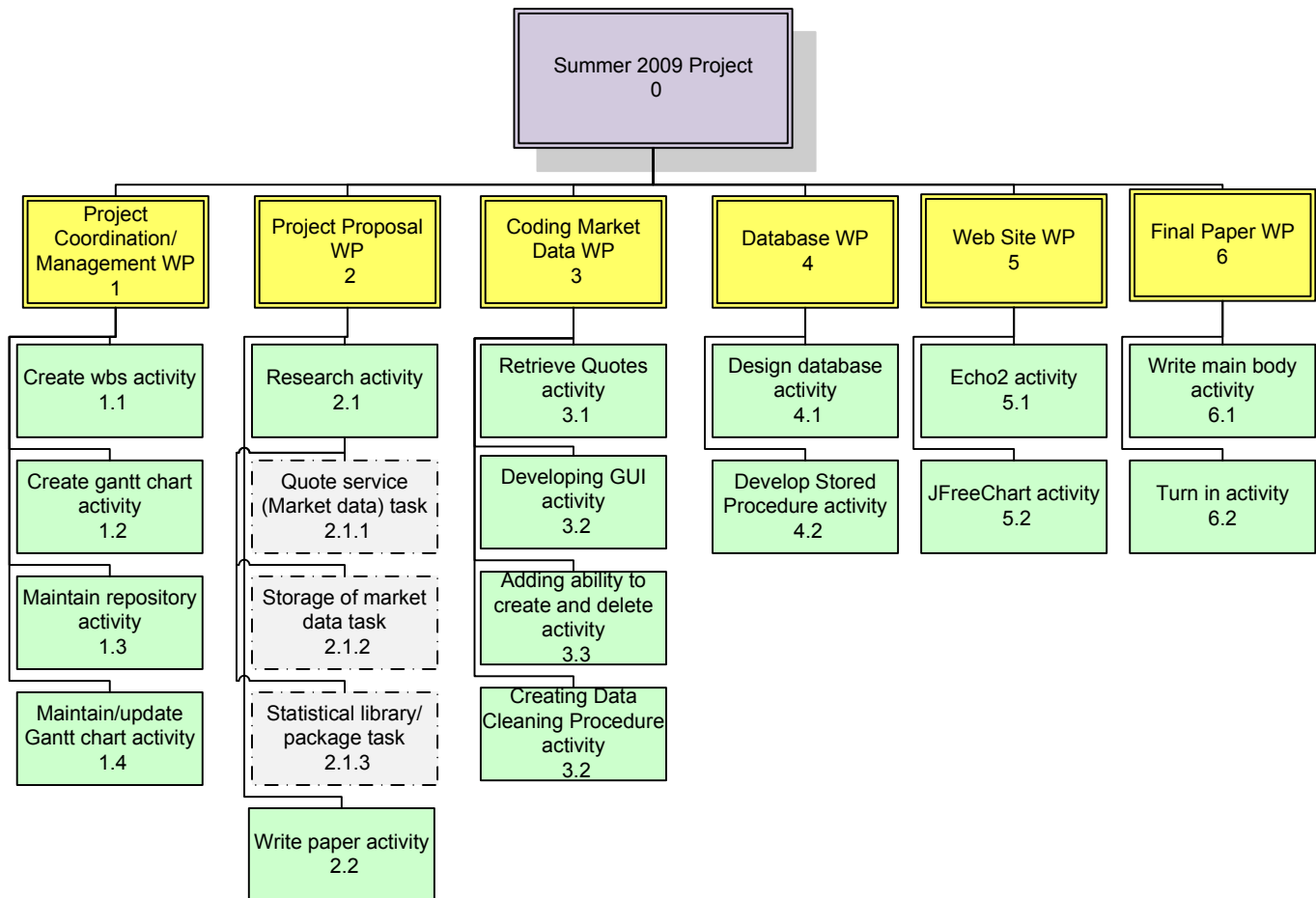
-- SELECT * FROM tempTopCorrelation order by _tempcorr desc LIMIT 10;
SELECT _symbolX, _symbolY, _tempCorr, _days_back FROM tempTopCorrelation
    order by _tempcorrAbs desc LIMIT 30;

END $$

DELIMITER ;

```

Work Breakdown Schedule



WBS Dictionary		
0	Summer 2009 Project	
1	Project Coordination/Management WP	
1.1	Create wbs activity	create the WBS via Microsoft Visio
1.2	Create gantt chart activity	create the Gantt chart via Microsoft Project
1.3	Maintain repository activity	subversion via Netbeans
1.4	Maintain/update Gantt chart activity	maintenance of Gantt chart/PM via Microsoft Project
2	Project Proposal WP	
2.1	Research activity	research via web/books/library
2.1.1	Quote service (Market data) task	determine most suitable quote service
2.1.2	Storage of market data task	determine how to store the quote/market data
2.1.3	Statistical library/package task	determine which statistical package to use...if any
2.2	Write paper activity	write the proposal of said research
3	Coding Market Data WP	
3.1	Retrieve Quotes activity	retrieve quotes from market service
3.2	Developing GUI activity	store quotes in suitable database
3.3	Adding ability to create and delete activity	create classes for adding symbols to database
3.4	Creating Data Cleaning Procedure activity	create procedure for determining if data is o.k.
4	Database WP	
4.1	Design database activity	create and design database
4.2	Develop Stored Procedure activity	create stored procedures using MySQL
5	Web Site WP	
5.1	Echo2 activity	work on website with echo2
5.2	JFreeChart Activity	develop charts with JFreeChart
6	Final Paper WP	
6.1	Write main body activity	write the main body of the paper
6.2	Turn in activity	finished

ID	Task Name	May 10, '09	May 17, '09	May 24, '09	May 31, '09	Jun 7, '09	Jun 14, '09	Jun 21, '09	Jun 28, '09	Jul 5, '09	Jul 12, '09	Jul 19, '09	Jul 26, '09
1	Project Coordination/Management WP	S M T T W T T F F S S M T T W T T F F S S M T T W T T F F S S M T T W T T F F S S M T T W T T F F S S											
2	Create vbs activity												
3	Create gantt chart activity												
4	Maintain repository activity												
5	Maintain/update Gantt chart activity												
6	Project Proposal WP												
7	Research activity												
8	Quote service (Market data) task												
9	Storage of market data task												
10	Statistical library/package task												
11	Write paper activity												
12	Coding Market Data WP												
13	Retrieve Quotes activity												
14	Developing GUI activity												
15	Adding ability to Create and Delete Symbols activity												
16	Creating Data Cleaning Procedures activity												
17	Database WP												
18	Design activity												
19	Developing Stored Procedures activity												
20	WebSite WP												
21	Echo2 activity												
22	JFreeChart activity												
23	Final Paper WP												
24	Write main body activity												
25	Turn in activity												

Task

Progress

Baseline

Milestone

Baseline Milestone

Summary

Rolled Up Task

Rolled Up Milestone

Baseline Summary

Rolled Up Baseline

Rolled Up Baseline Milestone

Rolled Up Progress

Split

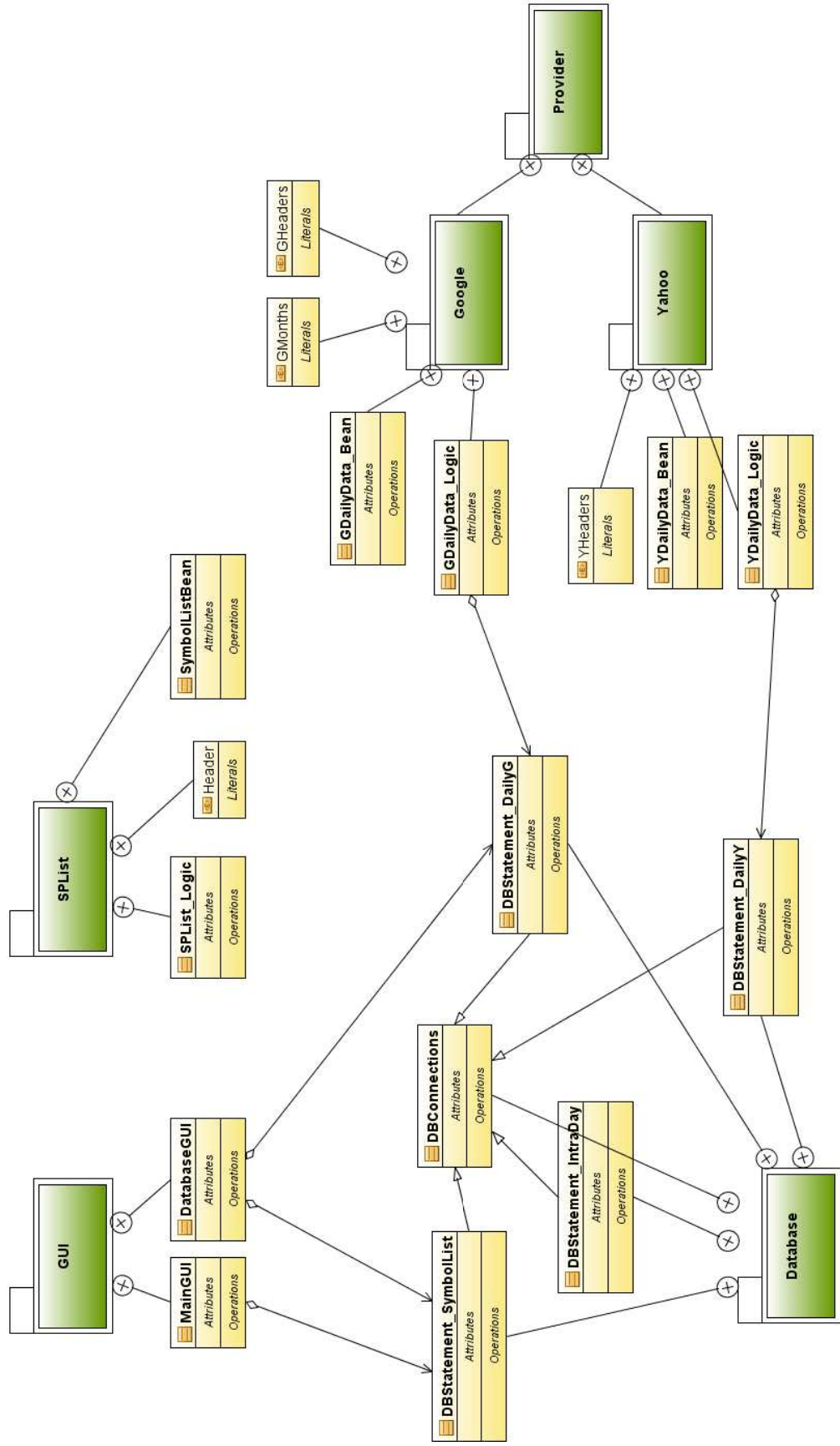
Baseline Split

External Tasks

Project Summary

Group By Summary

Deadline



References

- [1] Connecting to a MySQL database - NetBeans IDE 6.7 tutorial. <http://www.netbeans.org/kb/docs/ide/mysql.html>.
- [2] Understanding stock splits. <http://www.investopedia.com/articles/01/072501.asp>.
- [3] MySQL AB and Sun Microsystems Inc. MySQL. <http://www.mysql.com/>.
- [4] David R. Anderson, Dennis J. Sweeney, and Thomas A. Williams. *Statistics for Business and Economics, 6th Edition*. West Group, 6th edition, February 1996.
- [5] Remco R. Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald, and David Scuse. WEKA manual for version 3-6-0, December 2008.
- [6] Allin Cottrell and Riccardo Lucchetti. Gretl user's guide, December 2008.
- [7] db4objects Inc. db4o tutorial, 2008.
- [8] Russell J. T. Dyer and Andy Oram. *MySQL in a Nutshell*. O'Reilly, 2008.
- [9] Apache Software Foundation. The apache commons mathematics library. <http://commons.apache.org/math/>.
- [10] David Gilbert. The JFreeChart class library: Developer guide, 2008.
- [11] Peter Gulutzan. A MySQL technical white paper, March 2005.
- [12] Guy Harrison and Steven Feuerstein. *MySQL stored procedure programming*. O'Reilly, 2006.
- [13] NextApp, Inc. Comparing the google web toolkit to echo2. <http://echotwo.blogspot.com/2006/05/comparing-google-web-toolkit-to-echo2.html>.
- [14] SAS Institute Inc. SAS. <http://www.sas.com/>.
- [15] SPSS Inc. Clementine. <http://www.spss.com/software/modeling/modeler/>.
- [16] M. Kishinevsky and M. Higgs. Downloading yahoo data. <http://www.gummy-stuff.org/Yahoo-data.htm>, 2008.
- [17] Michael Kofler and David Kramer. *The Definitive Guide to MySQL 5*. Apress, 2005.
- [18] Integral Solutions Limited. Clementine 10.1 users guide, 2006.
- [19] Interactive Brokers LLC. Interactive brokers: Getting started with the TWS java API. <http://www.interactivebrokers.com/download/JavaAPIGettingStarted.pdf>, August 2008.
- [20] Interactive Brokers LLC. Interactive brokers: API reference guide, February 2009.
- [21] Dave Minter and Jeff Linwood. *Beginning Hibernate*. Apress, 2006.
- [22] Dare Obasanjo. Why aren't you using an object oriented database management system? <http://www.25hoursaday.com/WhyArentYouUsingAnOODBMS.html>, 2001.
- [23] Bram Smeets, Uri Boness, and Roald Bankras. *Beginning Google Web Toolkit: From Novice to Professional*. Apress, 2008.
- [24] Wikipedia. Open database connectivity (ODBC). http://en.wikipedia.org/wiki/Open_Database_Connectivity, July 2009.
- [25] Ian H. Witten and Eibe Frank. *Data mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.
- [26] Matthieu Wyart and Jean-Philippe Bouchaud. Self-referential behaviour, overreaction and conventions in financial markets. *Journal of Economic Behavior & Organization*, 63(1):1–24, May 2007.