Design of a Stock Information Storage System

-- Project Phase 1 - Data Collection & Storage Module

Group Member:
Ze Liu(zl265)/ Tong Wu(tw445)/ Xinwei Zhao(xz245)/ Jay borkar(jb1545)/ Xinyu Lyu(xl422)

I. Executive Summary

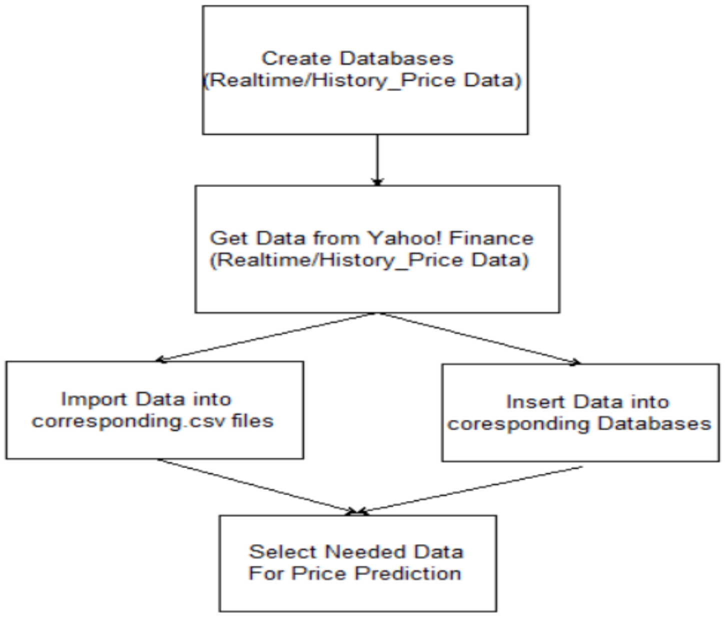
This report presents the design of data collection & storage module for the stock price & volume information. The system could develop an application that runs continuously as a background process and periodically retrieves the stock information, parses the received responses, and stores the extracted parameters into a local relational database. The design will greatly contribute to predicting the trend of the stock prices in following project phases.

II. Introduction

The system is designed to store 10 stocks' price & volume information. For each stock, the system would get at least one day of real-time data and at least one year of the historical prices. The real-time data would contain information of the price, volume, and the corresponding time which would accurate to seconds. And the time slice between two real-time points will be no more than one minute. The historical data contains the information from the stock, such as the open-price, high-price, low-price, close-price, corresponding date and the volume.

III. Discussion

1. System Work Flow:


Create Databases (Realtime/History_Price Data)
Get Data from Yahoo! Finance (Realtime/History_Price Data)
Import Data into corresponding.csv files
Insert Data into coresponding Databases
Select Needed Data For Price Prediction

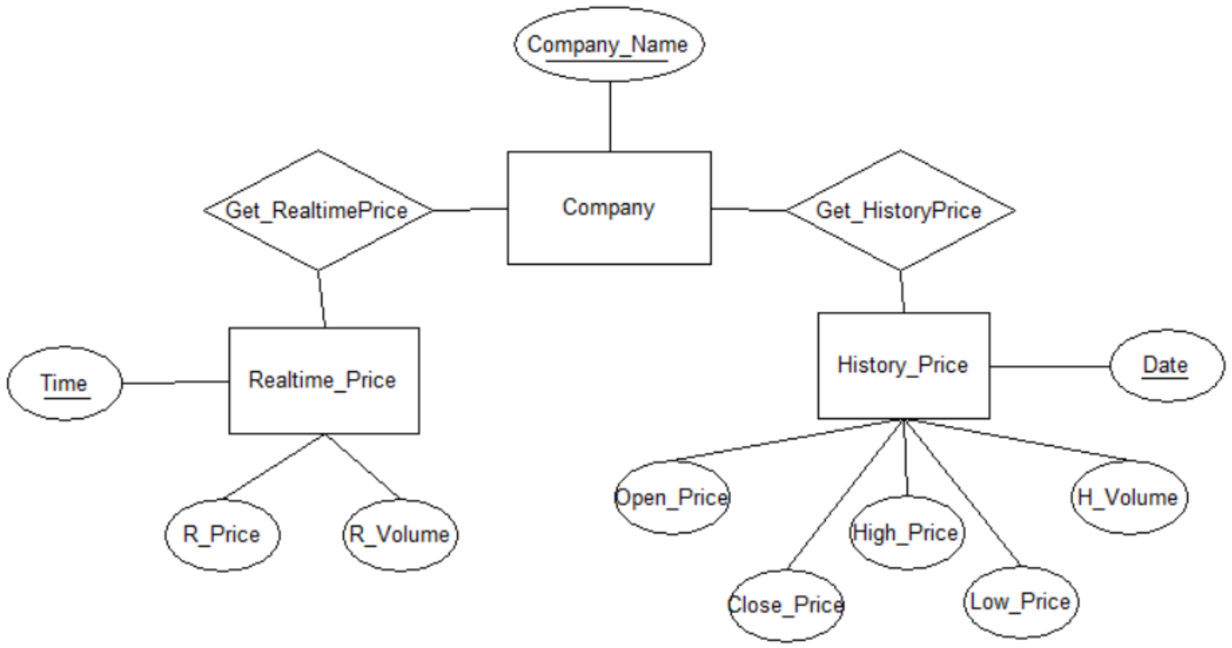
2. Create Database:

Before retrieving the Realtime/History_Price data from Yahoo! Finance, we automatically connect the workbench to Mysql with the help from the Python extra library MySQLldb. Then we create the databases by codes in Python as InitializeDB(priceName) function. When creating the databases, if the table has already existed in the database,we can automatically drop it and create a new one. Table History_Price is used to store the History_Price data such as the Open_price, High_price, Low_price, Close_price, Date (Primary Key) and H_Volume. Table Realtime_Price is used to save the Realtime_Price data such as the R_Price, R_Volume, and Time (Primary Key).

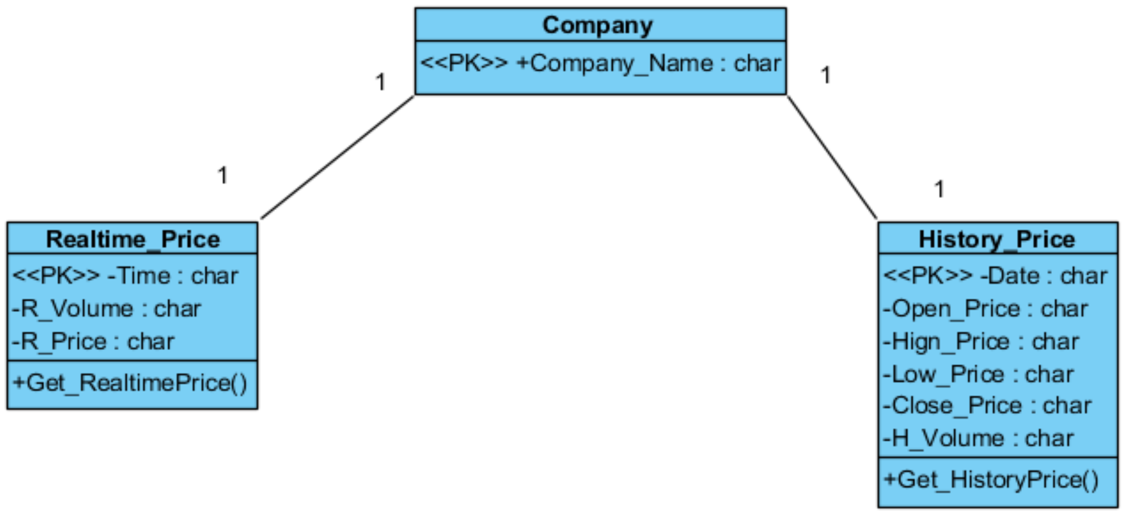
We also create the other three tables. The Company Table is used to store different company stock names. The Get_Realtime_Price relationship-table connects the Company table to the Realtime_Price table. It can help to get the Realtime-Price information when searching by company names. The Get_History_Price relationship-table connects the Company table to the History_Price table. Just like the former one, it can also do help to get the History-Price information effectively.

Since it is the project phase I, we simply plus the company name as infixes to the Realtime/History Price tables, i.e., AMZN_History_Price / AMZN_Realtime _Price. And there are both Realtime_Price table and History_Price table for each stock company, so a total of 20 tables for 10 companies. These three tables would greatly contribute to the following project phase when adding the customer modules.

3. The E-R model for the database:



4. System design diagrams in UML:



5. Get Data from Yahoo! Finance:

We develop the program based on the Python-Spider to automatically retrieve the Realtime_Price /History_Price data from Yahoo! Finance, with extra libraries, such as bs4.BeautifulSoup, Requests, and Selenium.

For example, the real URL for the Realtime/History data from AMAZON: 'https://finance.yahoo.com/quote/AMZN?p=AMZN'

First, in function get_html(url)/ get_html(url), we use Chrome browser to access the target webpage. Then, in get_content(stockUrl) function we strip the price and volume, time data from the webpage. When crawling the Realtime price data, we set time interval to be 16 seconds and it will automatically strip the price data every 15 seconds. And it is a little tricky to obtain the complete data on the History_Price webpage. Because the webpage doesn't show all the data at once, you need to roll down the page then it will load more data. Therefore, we simulate the webpage roll down in driver.execute_script ("arguments [0] .scrollIntoView();",target) to obtain the complete history price data. If we successfully get the data, it will print

```
Wirte Success!
Success all.
```

6. Importing Data to Database/.csv files:

In the program, we develop both two ways to save the data. In write_in_csvfile(content)function we import the data into specific columns of the corresponding .csv files, with extra library csv. When importing data into the database, we first connect the workbench to the Mysql with the help of the Python extra library MySQLldb. Then, in InsertDatabase (priceName,content) function, we insert the price data in specific columns of the corresponding tables. If we successfully import the data, it will print

```
TXT Wirte Success!
CSV Wirte Success!
Database Wirte Success!
```

7. Individual contributions:

Name:	**Contributions:**
Ze Liu(zl265)	Create database, Insert data, Store stock information into CSV, Crawler system test
Tong Wu(tw445)	Design python crawlers to get real-time/history data, Database validation, Crawler system test
Xinwei Zhao(xz245）	Design system UML, Write report, Database validation, Crawler system test
Jay borkar (jb1545)	Database validation, Crawler system Test
Xinyu Lyu(xl422)	Design database schema, Write report, Crawler system test, Database validation

IV. Conclusion

This report has discussed the development of the data collection & storage module, the base step to establish the stock price prediction system. The objectives of this step are to collect the real-time and the historical data from the Yahoo! Finance and create the necessary database for to save the stock price information. Both objectives are met. By keeping track of the project, the system would be finished.

V. References

Frank M. Carrano & Timothy Henry. (2013). *Data Abstraction & Problem Solving with C++ (6th ed).*