COL362 Course Project: Stock Market Portfolio Manager

Harshdeep Gupta, 2013MT60597
Deepali Gupta, 2013MT60079

March 15, 2017

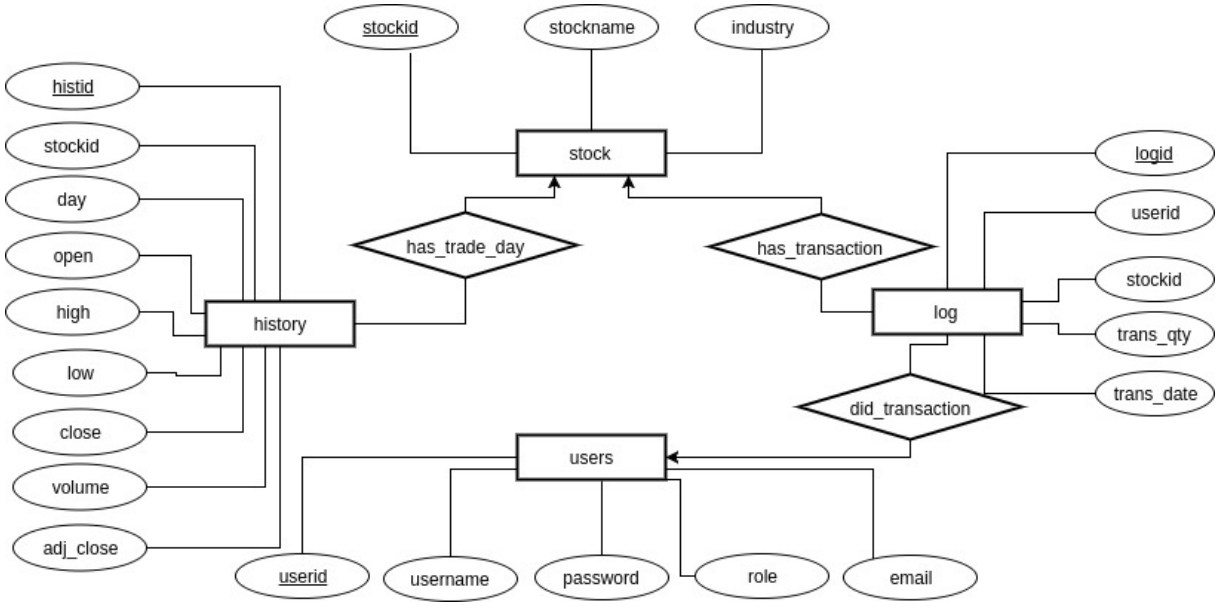
1 Project Description

This project is a database application for displaying and managing stock market related data. The motivation was to provide any common trader without any technical knowledge with an easy way to manage his portfolio. It is a website that keeps track of the daily trading data of the BSE Index - Sensex and its constituent stocks. A general user can check the details of the listed stocks as well as their daily trading records over the past three months.

The website also serves as a personal portfolio manager for registered users. Users can enter records of their transactions over the available Sensex stocks. The website queries all the transaction and presents a summary of user's portfolio to him along with the net profit or loss on each stock and the total portfolio.

The entities and attributes that were designed by us are shown in Table 1, while the ER Diagram corresponding to this is shown in Figure 1.

1.1 Entity Relationship Model



ER Diagram

2 Data Sources and Statistics

- We are using the data of the BSE Sensex listed stocks. This list of 30 stocks was scraped from http://www.moneyworks4me.com/best-index/bse-stocks/top-bse30-companies-list/. This table of stocks was created manually.

Table 1: Entities and Attributes

Entity	Attributes
stock	stockid, stockname, industry
history	histid, stockid, day, open, high, low, close, volume, adj-close
users	userid, username, password, role, email
log	logid, userid, stockid, trans-qty, trans-date

- Corresponding to each stock in this list as well the sensex index, its historical trading data over the past three months was downloaded as csv files from https://in.finance.yahoo.com/, an example being https://in.finance.yahoo.com/quote/TATASTEEL.NS/history?p=TATASTEEL.NS
- Stocks for which historical data was not available were removed from the stock table.
- Data from the tables of each stock was put into one single file.
- Columns for histid and stockid were added into the above file using a Python script for populating the 'history' table.
- These two tables were loaded into the database using bulk load.

```
\COPY stock FROM 'C:\Users\HD\Desktop\myapp\stocks.csv' DELIMITERS ',' CSV;
```

- The rest of the smaller tables in the database were populated manually as well as directly from the front end of the website to simulate the functionality.
- The Statistics for each of the tables is listed in Table 2.

Table 2: Statistics

Name	No. of Tuples	Time to Load	Raw Dataset Size	Raw Dataset Size (clean)
stock	29	26.104 ms	-	1.3 kB
history	1557	82.401 ms	99.9 kB	93.3 kB
users	5	42.676 ms	-	170 bytes
log	15	117.030 ms	-	327 bytes

3 Functionality and Working

3.1 User's View of the System

(a) Home:
The home page displays the last Sensex price, change and percentage change with respect to the previous day. It also displays a list of the top gainers and top losers among the stocks based on the rise or drop in price of the stocks. A list of all listed stocks with their details such as name, industry, last price, change and percentage change in price is also displayed. The user can search through each of these tables using any keyword. This page has options for user registration and login.

(b) Sensex:
Clicking on the Sensex price on the Home page leads to this page. It list the historical trading data for Sensex over the past three months with the trading date, opening price, day's lowest price, day's highest price, closing price, volume traded and adjusted closing price.

(c) Stock:

- Details:
  Clicking on any particular stock in the list of stocks on the home page leads to this. It provides the historical data for that particular stock over the past months.

- Delete:
  A form is opened where the admin can select from a dropdown list of available stocks and delete it. The dropdown uses a query to display all the stocks.
- Edit:
  A form is opened where the admin can select from a dropdown list of available stocks and edit its name or industry.

(d) User:

- Register:
  A new user can register with a username, password, role and emailid. A warning message is thrown if a username already exists with failure of registration.
- Login:
  An existing user can login with username and password. Login fails in case of authentication failure. The user is taken to My Portfolio page on successful sign in. The role of the user from the database is maintained as a session variable to enable appropriate permissions to modify the data.
- Settings:
  A logged in user can change his password or choose to delete his account.
- Remove:
  An admin can choose from a dropdown list of available users and delete the account. The dropdown uses a query to display all the users.

(e) My Portfolio
A session variable is maintained for the logged in user to load the appropriate portfolio details.

- List of Stocks:
  The list of available stocks in the user's portfolio is displayed with the number of shares present, last price, percentage change in price with respect to the previous day and the net profit or loss for the user for each stock. This page also shows the net portfolio value and gross profit/loss.
- Add new transaction:
  The logged in user can enter a new transaction record by selecting from a dropdown of available stocks, and entering the quantity bought or sold, transaction date. A query is automatically fired to update that stock's net quantity, cost and profit in the user's portfolio.
- Portfolio Stock Transaction History:
  Clicking on a particular stock in the list of portfolio stocks leads to this page. It shows the complete transaction history for that stock for the logged in user.

3.2 System View

(a) Indexes:
We indexed the history and log tables on stockid using a B-Tree index for faster lookups on a particular stock. This is used very often when getting the last price for a stock in many queries.

```
create index on history using btree (stockid);
create index on log using btree (stockid);
```

(b) Constraints:

- Primary key constraints on each table
- Referential integrity constraints for history to refer to stock and log to refer to stock and users table. On delete cascading is done.

```
alter table history drop constraint history_stockid_fkey;
alter table history add foreign key(stockid) references stock(stockid) on delete cascade;

alter table log drop constraint log_stockid_fkey;
```

```
alter table log add foreign key(stockid) references stock(stockid) on delete cascade;

alter table log drop constraint log_userid_fkey;
alter table log add foreign key(userid) references users(userid) on delete cascade;
```

- Uniqueness constraint for stock(stockname), users(username), users(email)
- Not null constraints on stock(stockname), history(day), users(username), users(password), log(trans-date), log(trans-qty)

(c) Sequences
We use serial sequences for each of the tables on their primary keys. Whenever a new entry is made to any table, the sequence is incremented and a unique id is automatically provided to each tuple.

(d) Views
We create a materialized view portfolio to keep track of each stock's net quantity of shares and net cost present in each user's portfolio. This saves time by directly displaying the portfolio list from this table instead of querying the entire log table.

```
create materialized view portfolio as
  (select userid, log.stockid, sum(trans_qty) as qty, sum(trans_qty*close) as cost
  from log inner join history
  on log.stockid=history.stockid
  and log.trans_date=history.day
  group by userid, log.stockid);
```

(e) Triggers
We implemented the following trigger to update the portfolio view whenever a new transaction is entered into the log table.

```
create function refresh_mat_view()
returns trigger language plpgsql
as $$
begin
    refresh materialized view portfolio;
    return null;
end $$;

create trigger refresh_mat_view
after insert or update or delete or truncate
on log for each statement
execute procedure refresh_mat_view();
```

(f) Access Privileges
We created three user roles with the following permissions:

- Admin : Unlimited access to the database
- Member : Can modify only the log table and make entries in the history and log tables.
- Guest : Cannot make any changes to the data but can only query the data using SELECT queries.

3.3 List of Queries

(a) stock

(i) Get all stock names

```
select stockname from stock where stockname <> 'Sensex';
```

(ii) Edit stock

```
update stock
set stockname = ${stockname2},
industry = ${industry}
where stockname = ${stockname1};
```

(b) history

(i) Get all stocks last price list

```
select stockname, industry, t.day as day,
round(cast(t.close as numeric),2) as curr_price,
round(cast((t.close-t.open) as numeric),2) as diff,
        round(cast(100*(t.close-t.open)/t.open as numeric),2) as perc
from stock inner join
(select distinct on (stockid) stockid, day, open, high, low, close, volume, adj_close
 from history
 order by stockid asc, day desc
) t
on stock.stockid=t.stockid
where stockname <> 'Sensex'
```

(ii) Get top 5 gainers

```
select stockname, industry, t.day as day,
round(cast(t.close as numeric),2) as curr_price,
round(cast((t.close-t.open) as numeric),2) as diff,
        round(cast(100*(t.close-t.open)/t.open as numeric),2) as perc
from stock inner join
(select distinct on (stockid) stockid, day, open, high, low, close, volume, adj_close
 from history
 order by stockid asc, day desc
) t
on stock.stockid=t.stockid
where stockname <> 'Sensex'
order by perc desc
limit 5;
```

(iii) Get top 5 losers

```
select stockname, industry, t.day as day,
round(cast(t.close as numeric),2) as curr_price,
round(cast((t.close-t.open) as numeric),2) as diff,
        round(cast(100*(t.close-t.open)/t.open as numeric),2) as perc
from stock inner join
(select distinct on (stockid) stockid, day, open, high, low, close, volume, adj_close
 from history
 order by stockid asc, day desc
) t
on stock.stockid=t.stockid
where stockname <> 'Sensex'
order by perc
limit 5;
```

(iv) Single stock details

```
select stockname, industry, round(cast(open as numeric),2),
round(cast(high as numeric),2), round(cast(low as numeric),2),
        round(cast(close as numeric),2), volume
from stock inner join
(select distinct on (stockid) stockid, day, open, high, low, close, volume, adj_close
 from history
 order by stockid asc, day desc
```

```
) t
on stock.stockid=t.stockid
where stockname = ${stockname};
```

(v) Single stock complete history

```
select day, round(cast(open as numeric),2), round(cast(high as numeric),2),
round(cast(low as numeric),2), round(cast(close as numeric),2),
            volume, round(cast(adj_close as numeric),2)
from history
where stockid = (select stockid from stock where stockname=${stockname})
order by day desc;
```

(vi) Sensex history

```
select day, round(cast(open as numeric),2), round(cast(high as numeric),2),
round(cast(low as numeric),2), round(cast(close as numeric),2),
            volume, round(cast(adj_close as numeric),2)
from history
where stockid = (select stockid from stock where stockname='Sensex')
order by day desc;
```

(vii) Sensex last price

```
select distinct on (stockid) stockid, round(cast(close as numeric),2),
round(cast((close-open) as numeric),2) as diff,
        round(cast(100*(close-open)/open as numeric),2) as perc
from history
where stockid = (select stockid from stock where stockname='Sensex')
order by stockid, day desc;
```

(c) users

(i) Get all usernames

```
select username from users;
```

(ii) User Details

```
select username, email
from users
where username = ${username};
```

(iii) Edit user

```
update users
set username = ${username2},
password = ${password},
role = ${role},
email = ${email}
where username = ${username1};
```

(iv) Delete user

```
delete from users where username = ${username};
```

(v) Return password for authentication

```
select password from users where username = ${username} and password = ${password};
```

(d) log

(i) Single stock complete trasaction history for that user

```
select trans_qty, trans_date, round(cast(close as numeric),2)
from log inner join history
on log.stockid = history.stockid
and log.trans_date = history.day
where log.userid = (select userid from users where username = ${username})
and log.stockid = (select stockid from stock where stockname = ${stockname})
order by trans_date;
```

(e) portfolio

(i) Portfolio list of stocks with details

```
select stockname, close, round(cast((close - open) as numeric),2) as diff,  round(cast(100*(clo
round(cast((qty*close - cost) as numeric),2) as profit
from stock inner join
(select t2.stockid, close, open, qty, cost from portfolio inner join
(select distinct on (stockid) stockid, day, open, high, low, close, volume, adj_close
from history
order by stockid asc, day desc) t2
on portfolio.stockid = t2.stockid
where portfolio.userid = (select userid from users where username = ${username})) t1
on stock.stockid = t1.stockid;
```

(ii) Single stock details for that user

```
select stockname, industry, close, round(cast((close - open) as numeric),2) as diff,  round(cas
round(cast((qty*close - cost) as numeric),2) as profit
from stock inner join
(select t2.stockid, close, open, qty, cost from portfolio inner join
(select distinct on (stockid) stockid, day, open, high, low, close, volume, adj_close
from history
order by stockid asc, day desc) t2
on portfolio.stockid = t2.stockid
where portfolio.userid = (select userid from users where username = ${username})) t1
on stock.stockid = t1.stockid
where stock.stockid = (select stockid from stock where stockname = ${stockname});
```

(iii) Net portfolio value, profit or loss

```
select round(cast(sum(qty*close) as numeric),2) as net_value,
round(cast(sum(qty*close) - sum(cost) as numeric),2) as profit
from
 (select distinct on (stockid) stockid, close
  from history
  order by stockid asc, day desc
 ) t inner join portfolio
on t.stockid = portfolio.stockid
where portfolio.userid=(select userid from users where username=${username})
group by userid;
```

3.4 Query Times

Table 3: Query Times

Query No.	Parameter Values	Runtime (ms)
(a)(i)	-	0.094
(a)(ii)	-	0.513
(b)(i)	-	2.414
(b)(ii)	-	1.983
(b)(iii)	-	2.233
(b)(iv)	-	1.784
(b)(v)	-	0.332
(b)(vi)	-	0.303
(b)(vii)	-	0.176
(c)(i)	-	0.040
(c)(ii)	-	0.117
(c)(iii)	-	0.184
(c)(iv)	userid=2	428.216
(c)(iv)	userid=4	74.62
(c)(v)	-	1.361
(d)(i)	username='deepali', stockname='Axis Bank Ltd.'	0.219
(d)(i)	username='deepali', stockname='Hero MotoCorp Ltd.'	0.177
(e)(i)	username='deepali'	1.399
(e)(i)	username='ad'	2.405
(e)(ii)	-	0.295
(e)(iii)	-	2.283