Scott Bishop

Andrew Thomas

March 12, 2013

CS174A

Updated Early Project Report

**Integrity Constraints and how to fix them:**

* Customers: No two customer usernames can be the same 🡪If someone makes a new username that already exists, prompt them to choose a different one. Handled by relational schema.
* Accounts: No two customers can have the same id\_Num. Check new id\_Num against all others to make sure it is unique before issuing to customer. Handled by relational schema.
* Accounts: Account balances can never go below 0. Transactions fail if when performing a Market Transaction it makes the account go below 0 (handled by program).
* Accounts: Share balance can never go below 0. Transaction fails if a Stock transaction makes the share balance go below 0 (handled by program).
* Account: Must have initial deposit of $1000 when opening a market account. If no initial deposit, don't let the customer open a market account. (handled by program).
* Stocks: Each actor has only one stock associated with him/her denoted by a symbol. Illegal stock if a stock has two symbols. Handled by relational schema.
* Transactions: Every transaction must have a valid id\_Num associated with it. Make sure the id\_Num exists before making the transaction (handled by program).
* Stocks: Closing and Current Price of stocks can’t be below 0. When adding new stock to the system make sure they are not below 0. If they are, don’t let them be added to the system (handled by program).

CREATE TABLE Customer (username CHAR(10), password CHAR(10), name CHAR(20), address CHAR(30), state CHAR(15), phoneNum CHAR(15), email CHAR(30), tax\_ID INTEGER, ssn CHAR(15), account\_ID INTEGER, isAdmin CHAR(1), PRIMARY KEY (tax\_ID));

CREATE TABLE Actors (actor\_ID CHAR(3), current\_Price REAL, name CHAR(20), dob CHAR(20), movie\_Title CHAR(30), role CHAR(10), year INTEGER, contract REAL, PRIMARY KEY (actor\_ID));

CREATE TABLE Stock (tax\_ID INTEGER, shares REAL, actor\_ID CHAR(3), FOREIGN KEY (actor\_ID) REFERENCES Actors, FOREIGN KEY (tax\_ID) REFERENCES Customer);

CREATE TABLE Market (tax\_ID INTEGER, account\_ID INTEGER, balance REAL, earnings REAL, startingMonthBalance REAL, PRIMARY KEY (tax\_ID), FOREIGN KEY (tax\_ID) REFERENCES Customer);

CREATE TABLE dailyBalance(tax\_ID INTEGER, month INTEGER, day INTEGER, year INTEGER, balance REAL, daysAtBalance INTEGER, FOREIGN KEY (tax\_ID) REFERENCES Customer);

CREATE TABLE Market\_Transaction ( tax\_ID INTEGER, month INTEGER, day INTEGER, year INTEGER, amount REAL, description CHAR(100));

CREATE TABLE Stock\_Transaction ( tax\_ID INTEGER, month INTEGER, day INTEGER, year INTEGER, num\_Shares REAL, stock\_Price REAL , description CHAR(100));

CREATE TABLE theDate(month INTEGER, day INTEGER, year INTEGER);

CREATE TABLE marketOpen(open CHAR(1));

**Functional Architecture of System**

We designed multiple classes to handle the GUI. A Login GUI that has two check boxes, clicking the sign up box will send the user to enter more info to create an account. If a user is an administrator, he or she can click that button to enter to the AdminView interface. Without clicking either checkboxes, when you login it will send the user to the main GUI. There, one can deposit, withdraw, buy, sell, show balance, show transaction history, price of stock for specific actor, movie information, and movie reviews. Within the AdminView, one can add interest, generate monthly statement, list active customers, generate DTER, customer report, and delete transactions. We have a database class that contains the corresponding queries that are needed in each view.

**Task Divisions**

We pair programed and did everything together. Switching off coding and observer every so often so both of us got a full understanding of the database system.

**SQL Queries**

"select \* from theDate";

"select month from theDate";

"update market set startingMonthBalance = balance";

"delete from theDate";

"insert into theDate values (" + (cal.get(Calendar.MONTH) + 1) + "," + cal.get(Calendar.DAY\_OF\_MONTH) + "," + cal.get(Calendar.YEAR) + ")";

"select unique tax\_ID from dailyBalance";

"select MAX(day) from dailyBalance where tax\_ID = '" + ids.get(x) + "'";

"update dailyBalance set daysAtBalance = " + (calNew.get(Calendar.DAY\_OF\_MONTH) - maxDay) + " where day = '" + maxDay + "' and tax\_ID = '" + ids.get(x) + "'";

"select balance from market where tax\_ID = '" + ids.get(x) + "'";

"insert into dailyBalance values(" + ids.get(x) + "," + (calNew.get(Calendar.MONTH) + 1) + "," + calNew.get(Calendar.DAY\_OF\_MONTH) + "," + calNew.get(Calendar.YEAR) + ","+ balance + "," + (calNew.getActualMaximum(Calendar.DAY\_OF\_MONTH) - calNew.get(Calendar.DAY\_OF\_MONTH) + 1) + ")";

"select tax\_ID from Customer where username = '" + username + "' and password = '" + password + "'";

"select tax\_ID from Customer where username = '" + username + "' and password = '" + password + "' and isAdmin = 'y'";

"select username from Customer where tax\_ID = '" + tax\_ID + "'";

"select max(account\_ID) from Customer";

"insert into Customer values('" + username + "','" + password + "','" + name + "','" + address + "','" + state + "'," + phone + ",'" + email + "'," + tax\_ID + "," + ssn + "," + account\_ID + ",'n')";

"insert into Market values(" + tax\_ID + "," + account\_ID + ",1000, 0)";

"select balance from Market where tax\_ID = '" + id + "'";

"select current\_price from Actors where actor\_id = '" + actorID + "'";

"select Balance from Market where tax\_ID = '" + id + "'";

query = "update Market set Balance = " + amount + " where tax\_ID = '" + id + "'";

"insert into Market\_Transaction values(" + id + "," + getDate().get(Calendar.MONTH) + "," + getDate().get(Calendar.DAY\_OF\_MONTH) + "," + getDate().get(Calendar.YEAR) + ","+ amount + ",'" + startingAmount + " was deposited on " + dateString(getDate()) + "')";

"select Balance from Market where tax\_ID = '" + id + "'";

"update Market set Balance = " + (balance - amount) + " where tax\_ID = '" + id + "'";

"insert into Market\_Transaction values(" + id + "," + getDate().get(Calendar.MONTH) + "," + getDate().get(Calendar.DAY\_OF\_MONTH) + "," + getDate().get(Calendar.YEAR) + "," + (amount \* -1)+ ",'" + amount + " was withdrawn on " + dateString(getDate()) + "')";

"select Shares from Stock where tax\_ID = '" + id + "' and actor\_id = '" + stockID + "'";

"update Stock set Shares = " + (currentAmount + amount) + " where tax\_ID = '" + id + "' and actor\_id = '" + stockID + "'";

"insert into Stock values (" + id + "," + amount + ",'" + stockID + "')";

"insert into Stock\_Transaction values(" + id + "," + getDate().get(Calendar.MONTH) + "," + getDate().get(Calendar.DAY\_OF\_MONTH) + "," + getDate().get(Calendar.YEAR) + "," + amount + "," + price + ",'" + amount + " shares were bought at " + price + " per share on " + dateString(getDate()) + "')";

"select Shares from Stock where tax\_ID = '" + id + "' and actor\_id = '" + stockID + "'";

"update Stock set Shares = " + (currentAmount - amount) + " where tax\_ID = '" + id + "' and actor\_id = '" + stockID + "'";

"insert into Stock\_Transaction values(" + id + "," + getDate().get(Calendar.MONTH) + "," + getDate().get(Calendar.DAY\_OF\_MONTH) + "," + getDate().get(Calendar.YEAR) + "," + amount + "," + price + ",'" + amount + " shares were sold at " + price + " per share on " + dateString(getDate()) + "')";

"select earnings from Market where tax\_ID = '" + id + "'";

"update Market set earnings = " + earnings + " where tax\_ID = '" + id + "'";

"select Shares, tax\_ID, actor\_id from Stock";

"delete from Stock where tax\_ID = '" + tax\_ID + "' and actor\_id = '" + actor\_id + "'";

"select name from Actors where actor\_id = '" + stockID + "'";

"select Shares from Stock where tax\_ID = '" + id + "' and actor\_id = '" + stockID + "'";

"select \* from Actors where actor\_id = '" + stockID + "'";

"select description from Market\_Transaction where tax\_ID = '" + id + "'";

"select description from Stock\_Transaction where tax\_ID = '" + id + "'";

"select \* from CS174A.movies where M\_NAME = '" + title + "'";

"select M\_ID from CS174A.movies where M\_NAME = '" + title + "'";

"select R\_REVIEW from CS174A.reviews where R\_ID = '" + m\_id + "'";

"select M\_NAME from CS174A.movies where M\_RANKING > 4.9 and M\_YEAR > '" + start + "' and M\_YEAR < '" + end + "'";

"delete from Market\_Transaction";

"delete from Stock\_Transaction";

"update market set earnings = 0";

"select account\_ID, balance from Market where tax\_ID = '" + id + "'";

"select shares, actor\_id from Stock where tax\_ID = '" + id + "'";

"select name, email from Customer where tax\_ID = '" + id + "'";

"select description from Market\_Transaction where tax\_ID = '" + id + "' and month = '" + month + "' and year = '" + year + "'";

"select description from Stock\_Transaction where tax\_ID = '" + id + "' and month = '" + month + "' and year = '" + year + "'";

"select startingMonthBalance, balance from market where tax\_ID = '" + id + "'";

"select earnings from Market where tax\_ID = '" + id + "'";

"select COUNT(tax\_ID) from Stock\_Transaction where tax\_ID = '" + id + "'";

"select tax\_ID, earnings from Market where earnings > 9999";

"select name, state from Customer where tax\_ID = '" + ids.get(x) + "'";

"select unique tax\_ID from Stock\_Transaction";

"select sum(num\_shares) from stock\_transaction where tax\_ID = '" + ids.get(x) + "'";

"select name from Customer where tax\_ID = '" + ids.get(x) + "'";

"update Actors set current\_Price = " + newPrice + " where actor\_id = '" + stockID + "'";

"update marketOpen set open = 'y'";

"update marketOpen set open = 'n'";

"select open from marketOpen";

"select unique tax\_ID from dailyBalance";

"select balance, daysAtBalance from dailyBalance where tax\_ID = '" + ids.get(x) + "'";

"select balance from Market where tax\_ID = '" + ids.get(x) + "'";

"update Market set balance = '" + total + "' where tax\_ID = '" + ids.get(x) + "'";

"select earnings from Market where tax\_ID = '" + ids.get(x) + "'";

"update Market set earnings = '" + total2 + "' where tax\_ID = '" + ids.get(x) + "'";