Online Banking Project

CSCI2141, FALL 2015

Mate Lakner B00

Samuel Champagne B00

Padraec Robinson B00640771

Submission Date: December 8, 2015

Introduction

Our project program was created with basic update and query functionality in order to be used by a banking company for online banking transactions. The Database must therefore easily be able to handle queries involving multiple customer accounts and display them uniformally in a user interface that is functional as well as easy to navigate. The capability to quickly update balances and transfer funds across accounts easily are of great importance. The backend of the program is written in mySQL the open source DBMS utility. The front end of our implemented database is written in JavaScript, and is designed for online banking functionality. The online implementation stores user account data, both for the user to track and change.

Due to the nature of modelling a database for a bank, security considerations are of paramount importance. Also data integrity constraints must be chosen on a basis of complete integrity over performance. The GUI is created with customer usability in mind, as the end users will be bank tellers, and customers accessing the DB from their laptops. specific constraints on the database based on the business (security issues, business specific considerations).

The objective is to create, populate, and maintain a database for an online banking service. The view for users includes a customer view, banker view, and also an admin view. The GUI for customer view will include options to deposit to accounts, view account info, make payments, as well as transfer funds. Everyday banking operations are intended to be accessable to even the most naive of users.

System Design and Development

The first stage in development was meeting and deciding on the business context of our database. Online Banking was the business context that we decided worked best. The business requirement analysis consisted of us brainstorming features and possible table relations for the database. We decided to focus our early development efforts on planning and creating the backend to our database. We designed a ER diagram(included below) using paper and pencil, taking into consideration the two user pools: bankers and customers.

When designing the front end to the banking website the major issues were creating two different views for the users, as well as separating the permissions for these two groups. Making sure that each group has access only to the data and queries that are relevant.

Actions available in the Customer view:

- Transfer money between personal accounts
- Change their password
- Apply for a loan
- Make a loan payment
- View account information
- View personal information

Actions available in the Banker view:

- Create customer
- Delete customer
- Change customer account info(except account balances)
- Approve a loan
- View a list of their customers with their information.

ER diagram

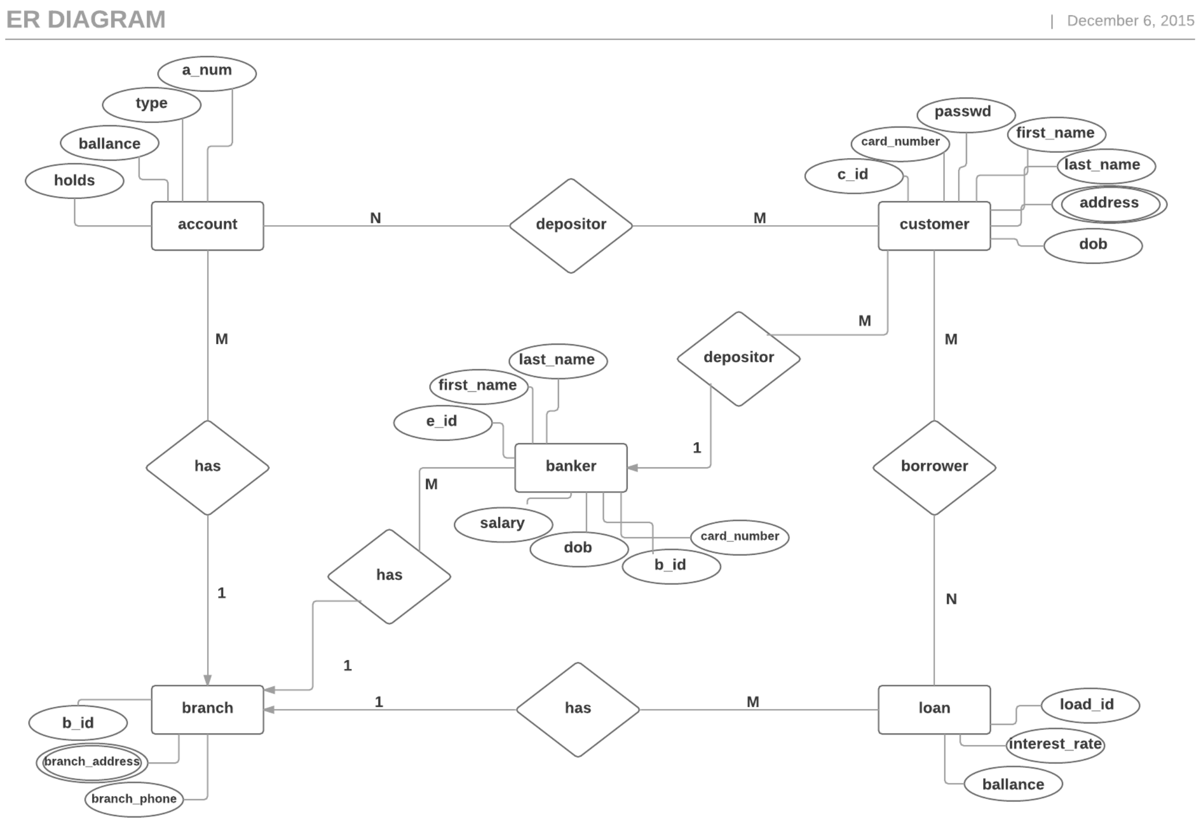
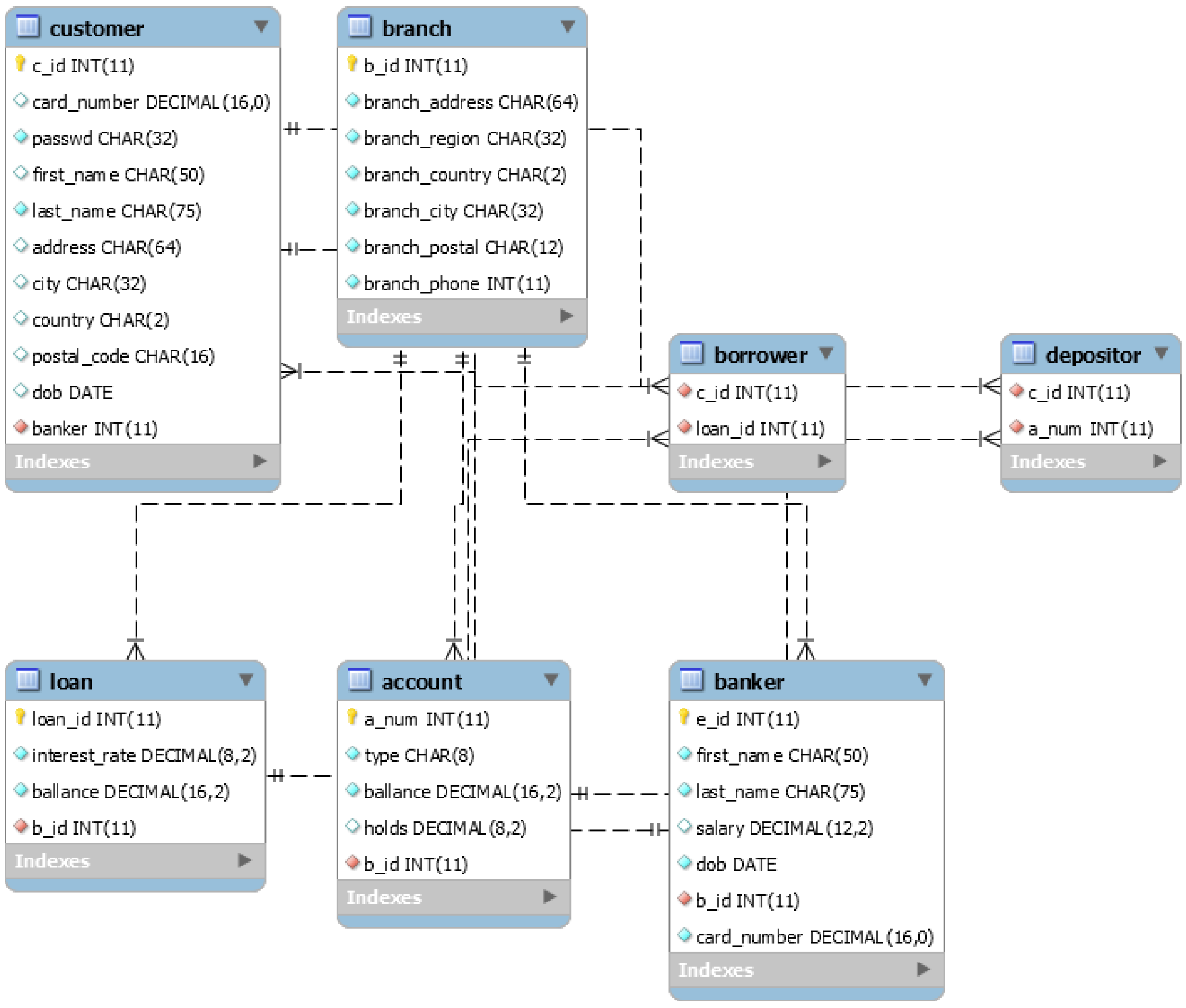


Table Schema



Normalization Analysis

After the ER and rough relational table schema were designed, we proceeded to create the database in mySQL. We emloyed First Normal Form normalization throughout the database creation process. Information fields are very specific throughout all tables, in order to allow more refined queries. One large hurdle we encountered was data redundancy in the Customer table. We initially wanted to have accountID in the customers table, but we instead had to create the relation to Accounts from the Branch table. We made this change in order to accomodate for customers having more than one account at any given time. Also in this normalization step we added another column to the account table to keep track of the account Type.

After making the above mentioned changes, our table schema was normalized according to Second Normal Form. No tables were found to have a transient relation between any two attributes. Therefore our database was in accordance with Third Normal Form.

Sample data from each table

<3-5 TUPLES FROM ALL TABLES>

Results Demonstration

Customer function to transfer money between accounts (same customer):

<transfer money view example>

This query deals with multiple accounts and thus must use transfer data between different account types. These transfers would traverse Borrower and Depositer relationships in certain circumstances. As such, data integrity must be insured using unique ids for the two customer-account relationships.

An example of the queries executed by the transfer command as follows:

<the sql queries>

Customer loan application function:

A customer may apply for a loan, this function sends an application form to the customers associated banker in order to be reviewed.

Examples of the loan forms for both user groups below:

<customer view example>
<Loan application form example>

Both views must execute queries when this function is used, but no data is modified until after approval. The queries executurted by this command are as follows:

<queries>

Customer Loan payment function:

When a customer uses the loan payment function, they are prompted to enter an amount, as well as an account from which the payment will be made.

<loan payment form>

If the account selected by the user has a balance that is greater than the entered payment amount a prompt is sent to the user.

<prompt on successful payment>

Customer money transfer between accounts (to other customer's account)

This function is slightly more complex than the transfer function for accounts with the same associated customer. When using this function, the user is prompted to enter the card number of the targer accounts associated customer, as well as the accounts id. Also, an input field for the amount is presented in the transfer form below.

<transfer form 2>

The queries executed by this transfer command involve moving data from one table to another, which is shown by the code below.

<queries from form2>

Customer function to view personal account information:

This function is a simple selection of customer information that is stored on the customer table. The query is as follows.

<the query>

Banker function to create customer:

This function would be used by a banker in order to create a new entry in the customer table. The banker is prompted to enter all the relevant information (NOT NULL fields) and a new entry is created.

<the create customer form>

This function calls a single query, which inserts the data entered, given that it satisfies the tables data constraints.

<the query>

Conclusion

Backend development of the database was done primarily by Sam, with database structure input from Padraec, as well as business analysis by Padraec and Mate. Front end development done by Mate, report done by Padraec with input from all group members.

In completing this project, my group has gained a stronger understanding of the steps involved in planning and actualizing a development project. Along with the general project management and group collaboration skills, a deeper understanding of front and back end database development has been gained by all group members. A much better understanding of the SQL language, as well as the mySQL workbench utility was gained from this project. Finally we learned how to host a database on Bluenose, and query it using SQL.

<source code here>