****

**CIS 9340**

**Team 1**

|  |  |
| --- | --- |
| Davran, Cem | cem.davran@baruchmail.cuny.edu |
| Izarova, Zhanna | zhanna.izarova@baruchmail.cuny.edu |

**Table of Contents**

1. **Business Requirements**
2. **Entity Relationship Diagram**
3. **Relationship Model and Normalization**
4. **Physical Database Development**
5. **Application Development**
6. **Conclusion**

**I Business Requirements**

**Proposal:**

Our team chose the Money Bank, which is a smaller, local bank for this project. Money Bank needs an improved, more technological and secure database to provide better services to its clients, while at the same time, ensuring customer satisfaction, security, and business growth. Money Bank needs a database to tie together its customer, its customer’s existing and future accounts and its representatives. By having complete information on what accounts the customers already have and who sold them the accounts, Money Bank can forecast what additional accounts can be offered to its customers and what Representatives are best suited for the job.

The new database will assist the Money Bank in determining who out of its employees does the best job and which type of account he or she can be better suited.

The application will track creation of new accounts to new and existing customers, issuance of loans, and also track which representative is responsible for each account or loan.

**Informational Needs:**

In order to build the database for the Money Bank, our team requested our client’s internal employee and customer information.

**Customer:** will maintain information on the customer and will have all customer attributes, which includes but not limited to name, SSN, address, income, job.

**Representative** will hold all information on the representatives and have all representative attributes, including but not limited to name, phone number, account type, salary, bonus.

**Account** will hold information on accounts and include attributes such as account number, account type, account amount, accountduedate.

**Loans:** Loan number, loan type, loan Amount.

**Customer-Account** (weak, junction-entity)

One customer can have many accounts and account can be held by multiple customers (i.e. joint family account). One Representative can have many customers, a customer can have many representative.

**Customer-Loan** (weak, junction-entity)

One customer can have many loans and a loan can be held by multiple customers (i.e. joint family account). A Loan can be issued by only one representative but a Representative can issue many loans.

**User Roles:**

Customer and Representative, who both need to have different levels of access. Customer will be able to view their existing accounts (amount available, amount due, etc), and their representative contact info.

Representative will be able to view their customers, accounts, and loans

Bank Manager will have access to all the info.

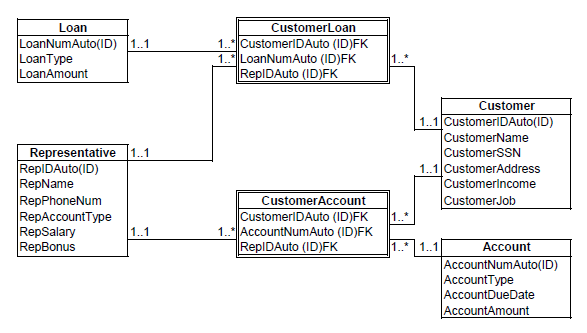
**Application Components:**

* Customer Entry Form- will allow data entry about every customer
* Customer Report - will list all the customers and their particular attributes
* Account Entry Form- will allow data entry about every account.
* Account List - will list all the accounts and their particular attributes
* Loan Entry Form- will allow data entry about every loan.
* Loan Report - will list all the loans and their particular attributes.
* Representative Entry Form - will allow data entry about every representative.
* Representative List - will list all the representative and their particular attributes.
* Customer-Account Entry Form - will allow data entry to assign accounts to customers and to representatives.
* Customer-Account Report - will list which accounts belongs to which customers.
* Customer-Loan Entry Form - will allow data entry to assign loans to customers and to representatives.
* Customer-Loan Report - will list which loans belongs to which customer.
* Rep Cust Report - will list all customers per each representative.
* Sum Account Per Rep Report - will list total amount of money in all accounts per representative.
* Sum Loan Per Rep Report - will list total loan amount per representative.

|  |  |
| --- | --- |
| Initial Tasks Assignement: | Team Member |
| System Analyst (Logical) | Davran, Cem |
| System Analyst (Physical) | Huang,Yuwen |
| Application Developer | Izarova, Zhanna |
| Data Tester | Baldera, Jason |

Actual project work was performed equally during brainstorming sessions.

**II Entity Relationship Diagram**

****

**III Relationship Model**

Loan(LoanNumAuto(key), LoanAmount, LoanType)

Customer(CustomerIDAuto(key), CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob)

Account(AccountIDAuto(key), AccountType, AccountDueDate, AccountAmount)

Representative(RepIDAuto(key), RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus)

CustomerAccount (CustomerIDAuto(FK), AccountNumAuto(FK), RepIDAuto(FK))

CustomerIDAuto(FK) references Customer

AccountNumAuto(FK) references Account

RepIDAuto(FK) references Representative

CustomerLoan(CustomerIDAuto(FK), LoanNumAuto(FK), RepIDAuto(FK))

CustomerIDAuto(FK) references Customer

LoanNumAuto(FK) references Loan

RepIDAuto(FK) references Representative

**Normalization**

Representative (RepIDAuto(key), RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus, CustomerIDAuto, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob, (AccountIDAuto, AccountType, AccountDueDate, AccountAmount), (LoanNumAuto, LoanAmount, LoanType)

Repeating groups are: (AccountIDAuto, AccountType, AccountDueDate, AccountAmount), (LoanNumAuto, LoanAmount, LoanType)

1. Determine candidate keys of relations and choose a Key

a. The candidate keys are the following: CustomerIDAuto (key), RepIDAuto (key), AccountIDAuto(key), LoanNumAuto(key)

2. Specify functional dependencies.

a. CustomerIDAuto - CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob

b. RepIDAuto(key) - RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus

c. AccountIDAuto - AccountType, AccountDueDate, AccountAmount

d. LoanNumAuto → LoanType, LoanAmount

e. CustomerIDAuto, LoanNumAuto → LoanType, LoanAmount

f. CustomerIDAuto, AccountIDAuto → AccountType, AccountAmount, AccountDueDate

1NF

The form is not in 1NF because it has repeating groups. So we need to remove the repeating groups and create a unique key for the repeating groups.

Representative( RepIDAuto(key), RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus, CustomerIDAuto, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob, AccountIDAuto(key), AccountType, AccountDueDate, AccountAmount, LoanNumAuto(key), LoanAmount, LoanType)

The above relationship contains no repeating groups so it is right now in 1NF.

The form right now has functional dependencies based on the functional relationships we listed above.

To convert 1NF to 2NF a relation needs to be created for each subset of the primary key. Composite key (superkey) maintains relationship to each of its components (subkeys) as foreign key. The non-key attributes are placed with their determinants, and names are assigned to newly created relations:

Representative (RepIDAuto(key), RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus, CustomerIDAuto, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob)

Account(AccountIDAuto(key), AccountType, AccountDueDate, AccountAmount,)

Loan(LoanNumAuto(key), LoanAmount, LoanType)

CustomerAccount (CustomerIDAuto(FK), AccountNumAuto(FK), RepIDAuto(FK))

CustomerLoan(CustomerIDAuto(FK), LoanNumAuto(FK), RepIDAuto(FK))

The form right now is in 2NF.

But in the Representative relationship, there is transitive dependency, which violates the 3NF.

CustomerIDAuto should be made to a primary key, which determine the attributes of the Customer Entity.

So the final schema should be shown as below:

Loan (LoanNumAudo(key), LoanAmount, LoanType)

Customer (CustomerIDAuto(key), CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob)

Account (AccountIDAuto(key), AccountType, AccountDueDate, AccountAmount)

Representative (RepIDAuto(key), RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus)

CustomerAccount (CustomerIDAuto(FK), AccountNumAuto(FK), RepIDAuto(FK))

CustomerLoan(CustomerIDAuto(FK), LoanNumAuto(FK), RepIDAuto(FK))

**IV Physical Database Development**

CREATE TABLE customer (

CustomerID NUMBER NOT NULL,

CustomerName VARCHAR(50) NOT NULL, CustomerSSN NUMBER NOT NULL,

CustomerAddress VARCHAR(200) NOT NULL, CustomerIncome NUMBER,

CustomerJob VARCHAR (100) RepresentativeID NUMBER);

ALTER TABLE customer

ADD CONSTRAINT PK\_Customer PRIMARY KEY (Customerid);

CREATE TABLE representative ( RepresentativeID NUMBER NOT NULL, RepName VARCHAR(50) NOT NULL,

RepPhoneNum NUMBER NOT NULL, RepAccountType VARCHAR(100), RepSalary NUMBER NOT NULL, RepBonus NUMBER);

ALTER TABLE representative

ADD CONSTRAINT PK\_representative PRIMARY KEY (Representativeid);

CREATE TABLE ACCOUNT (

AccountNum NUMBER NOT NULL, RepresentativeID NUMBER, AccountType VARCHAR (100) NOT NULL,

AccountAmmount NUMBER, AccountDueDate DATE);

ALTER TABLE Account

ADD CONSTRAINT PK\_Account PRIMARY KEY (AccountNum);

CREATE TABLE LOAN(

LoanNum NUMBER NOT NULL,

RepresentativeID NUMBER, LoanType VARCHAR (100) NOT NULL,

LoanAmmount NUMBER);

ALTER TABLE Loan

ADD CONSTRAINT PK\_Loan PRIMARY KEY (LoantNum);

ALTER TABLE customer

ADD CONSTRAINT FK\_Representative FOREIGN KEY (RepresentativeID)

REFERENCES Representative (RepresentativeID);

ALTER TABLE account

ADD CONSTRAINT FK\_Representative FOREIGN KEY (RepresentativeID)

REFERENCES Representative (RepresentativeID);

ALTER TABLE loan

ADD CONSTRAINT FK\_Representative FOREIGN KEY (RepresentativeID)

REFERENCES Representative (RepresentativeID);

CREATE TABLE CustomerAccount( CustomerID NUMBER NOT NULL,

AccountNum NUMBER NOT NULL);

ALTER TABLE CustomerAccount ADD CONSTRAINT FK\_Customer FOREIGN KEY (CustomerID)

REFERENCES Customer (CustomerID);

ALTER TABLE CustomerAccount ADD CONSTRAINT FK\_account FOREIGN KEY (AccountNum)

REFERENCES Account(AccountNum);

CREATE TABLE CustomerLoan(

CustomerID NUMBER NOT NULL, LoanNum NUMBER NOT NULL);

ALTER TABLE CustomerLoan

ADD CONSTRAINT FK\_Customer FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID);

ALTER TABLE CustomerLoan ADD CONSTRAINT FK\_Loan FOREIGN KEY (LoanNum)

REFERENCES Loan(LoanNum);

INSERT INTO representative

(representativeid, RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus)

VALUES (123456, ‘Brad Jones’, 2123334455, ‘Checking’,

50000, 0.02);

INSERT INTO representative

(representativeid, RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus)

VALUES (123450, ‘Sarah Jones’, 2123332255, ‘Savings’,

50000, 0.05);

INSERT INTO representative

(representativeid, RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus)

VALUES (123000, ‘Max Powers’, 2123332200, ‘Mortgage’,

55000, 0.02);

INSERT INTO representative

(representativeid, RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus)

VALUES (123111, 'Adam Levine', 2123332288, 'Credit', 55000, 0.02);

INSERT INTO customer

(customerid, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob, RepresentativeID)

VALUES (11111, 'John Doe', 11111111, '1 Main St, Brooklyn

MI 00000', 45000, 'Accountant', 123456);

INSERT INTO customer

(customerid, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob, RepresentativeID)

VALUES (11112, 'John Smith', 11111112, '10 Green St,

Brooklyn MI 00005', 70000, 'Medical\_Doctor', 123456);

INSERT INTO customer

(customerid, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob, RepresentativeID)

VALUES (11113, 'Ana Jones', 11111113, '9 Big St, Brooklyn

MI 00005', 70000, 'Writer', 123000);

INSERT INTO loan

(loannum, representativeid, loantype, loanamount) VALUES (11221, 123000, ‘Mortgage’, 200000)

INSERT INTO loan

(loannum, representativeid, loantype, loanamount) VALUES (11233, 123000, ‘Mortgage’, 300000)

INSERT INTO loan

(loannum, representativeid, loantype, loanamount) VALUES (11244, 123111, 'PersonalLoan', 50000)

INSERT INTO customer

(customerid, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, CustomerJob, RepresentativeID)

VALUES (11114, 'Ivan Sidorov', 11111114, '11 Small St,

Valley Springs MI 00005', 65000, 'Plumber', 123111 );

INSERT INTO account

(AccountNum, RepresentativeID, AccountType, AccountAmount)

VALUES (1000001, 123456, 'Checking', 100000);

INSERT INTO account

(AccountNum, RepresentativeID, AccountType, AccountAmount)

VALUES (1000002, 123450, 'Savings', 20000);

INSERT INTO account

(AccountNum, RepresentativeID, AccountType, AccountAmount)

VALUES (1000003, 123111, 'Credit', 5000);

INSERT INTO CustomerAccount

(CustomerID, AccountNum) VALUES (11113, 1000001);

INSERT INTO CustomerAccount

(CustomerID, AccountNum) VALUES (11113, 1000002);

INSERT INTO CustomerAccount

(CustomerID, AccountNum) VALUES (11111, 1000002);

INSERT INTO CustomerAccount

(CustomerID, AccountNum) VALUES (11114, 1000001);

INSERT INTO CustomerAccount

(CustomerID, AccountNum) VALUES (11113, 1000002);

INSERT INTO CustomerAccount

(CustomerID, AccountNum) VALUES (11114, 1000003);

INSERT INTO CustomerLoan

(CustomerID, LoanNum) VALUES (11111, 11221);

INSERT INTO CustomerLoan

(CustomerID, LoanNum) VALUES (11112, 11244);

INSERT INTO CustomerLoan

(CustomerID, LoanNum) VALUES (11113, 11221);

INSERT INTO CustomerLoan

(CustomerID, LoanNum) VALUES (11112, 11233);

**V Application Development**

Corrections: when our group started working on the Application Development we realized that our initial design was lacking and, therefore, had to redo the previous Milestones. Updated Milestones 1 through 3 are listed above but because we build the updated Physical Database Development using the first version of the project, we are listing the corrections for this step below instead:

* Primary keys were changed to autoincrements
* Representative Primary Key was added to CustomerAccount table and CustomerLoan table as a foreign key and part of a Composite Primary Key and was removed from Customer, Loan, and Account tables.

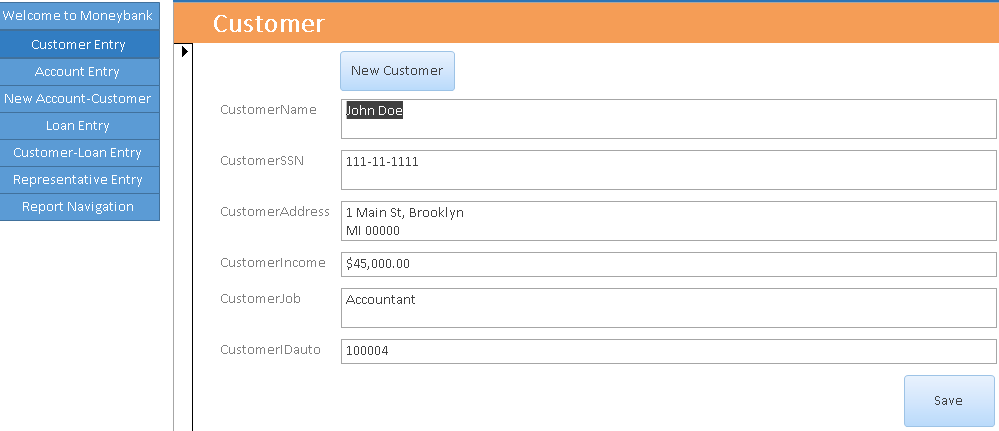
Money Bank Database Home Page:

Upon clicking on the Money Bank Dashboard button, the user is greeted with the following screen:



Customer Entry Form:

This form is used by a representative to add a new customer to the Money Bank database.



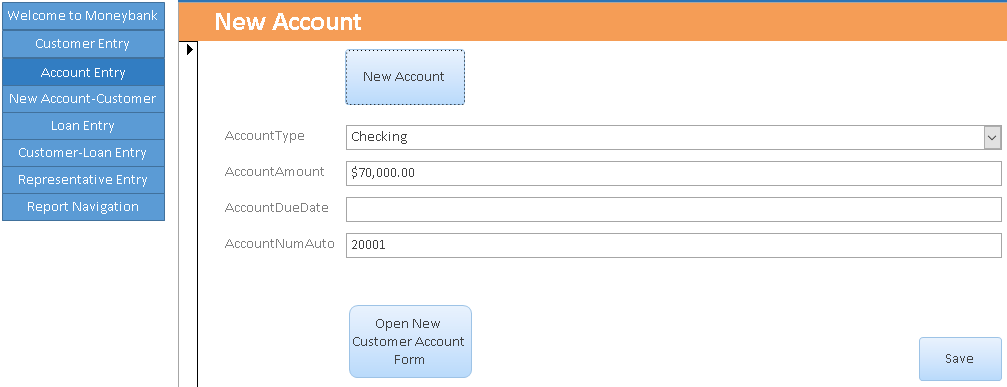
The above is the opening screenshot of the form. In order to add a new customer to the database, the user needs to click the “New Customer” button, which will make the form’s fields blank:



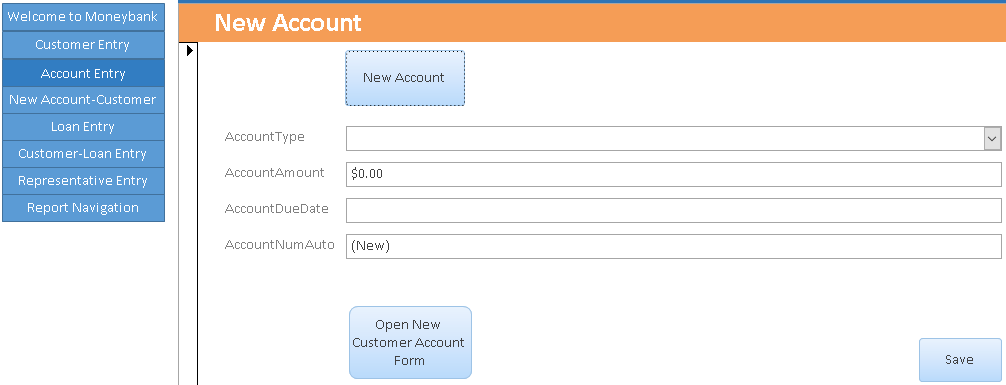
After information is entered (CustomerIDAuto field is populated automatically), the user has to press the “Save” button, which will save the new customer into the Customer table.

Account Entry Form:

This form is used by a representative to record the opening of a new account into the Money Bank Database.



The above is the opening screenshot of the form. In order to add a new account to the database, the user needs to click the “New Account” button, which will make the form’s fields blank:

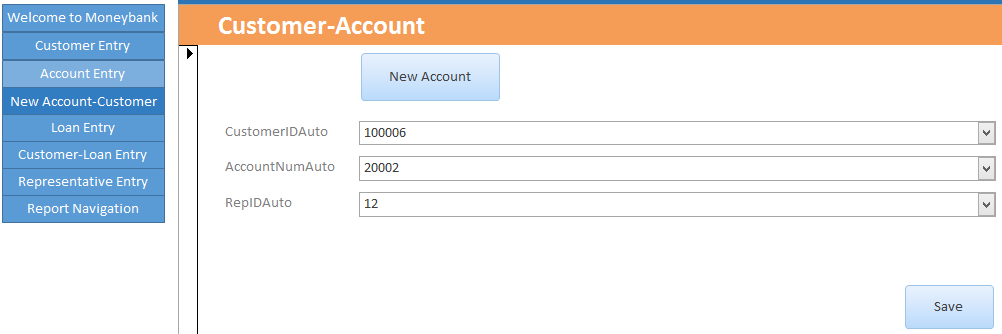


Note that the AccountType field is a combobox and only allows for three possible entries: “Checking”, “Credit”, and “Savings.” After information is entered (AccountNumAuto field is populated automatically), the user has to press the “Save” button, which will save the new account into the Account table.

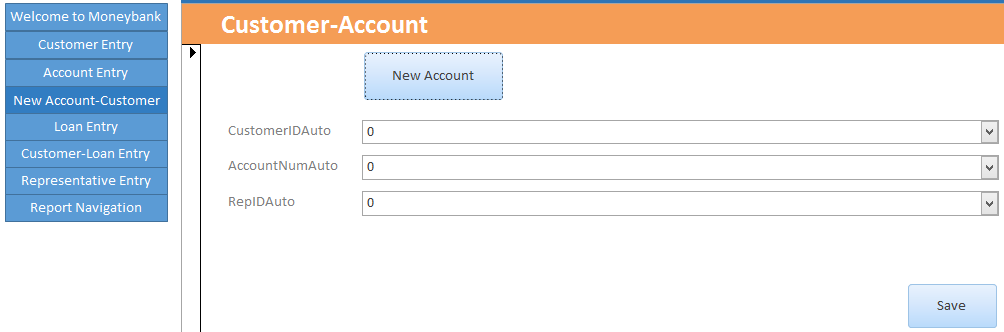
After saving the account, the user needs to click the “Open New Customer Account Form” Button below the AccountNumAuto Field or click on the New Customer-Account Entry navigation button.

New Customer-Account Form:

This form is used by a representative to tie together an instance of customer-account-representative and will update the CustomerAccount junction table.



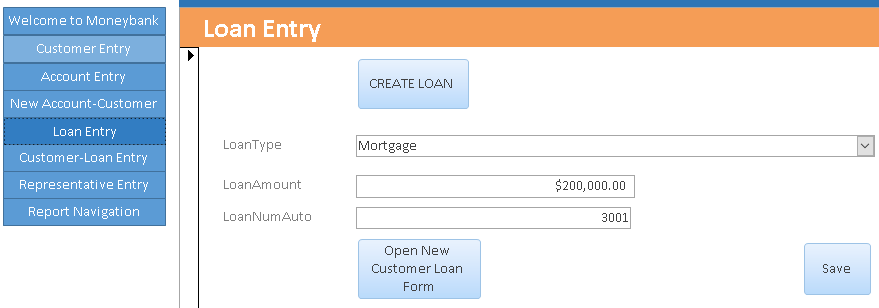
The above is the opening screenshot of the form. In order to make a new entry into the CustomerAccount table, the user needs to click the “New Account” button, which will make the form’s fields blank:



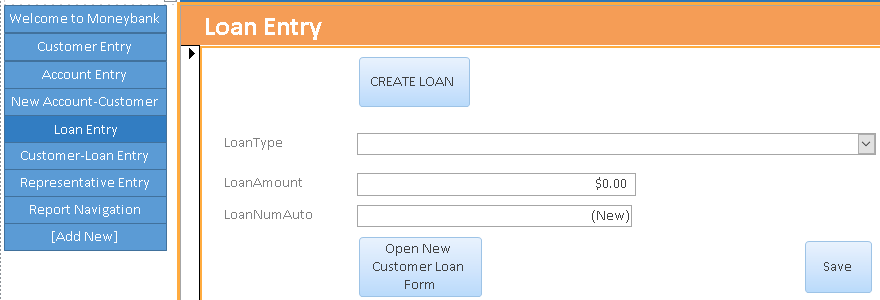
This form consists entirely of comboboxes because it is populated with the existing information. If any of the fields do not currently exist in the database, Customer Entry Form, Account Entry Form, and/or Representative form needs to be used first to update the information in the necessary table. After the information is selected, the user has to press the “Save” button, which will save the new instance of customer-account-representative into the CustomerAccount table.

Loan Entry Form:

This form is used by a representative to record a new loan into the Money Bank database.



The above is the opening screenshot of the form. In order to add a new loan to the database, the user needs to click the “Create loan” button, which will make the form’s fields blank:



Note that the LoanType field is a combobox and only allows for two possible entries: “Mortgage”, “Personal Loan.” After information is entered (LoanNumAuto field is populated automatically), the user has to press the “Save” button, which will save the new account into the Loan table.

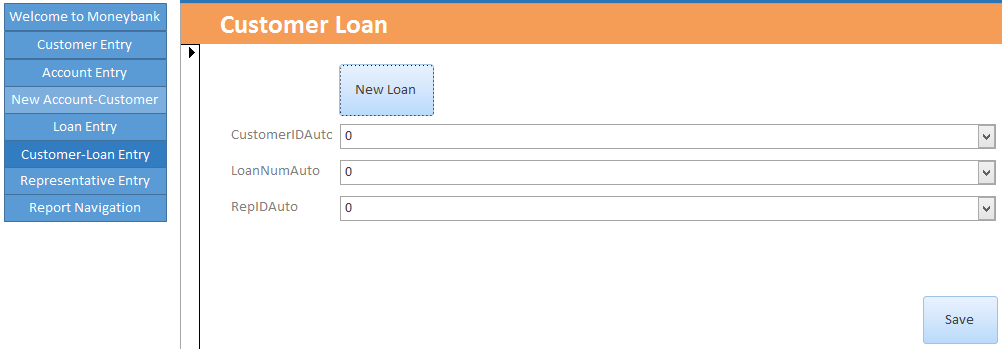
After saving the loan, the user needs to click the “Open New Customer Loan Form” Button below the LoanNumAuto Field or click on the Customer-Loan Entry navigation button.

Customer-Loan Entry:

This form is used by a representative to tie together an instance of customer-loan-representative and will update the CustomerLoan junction table.



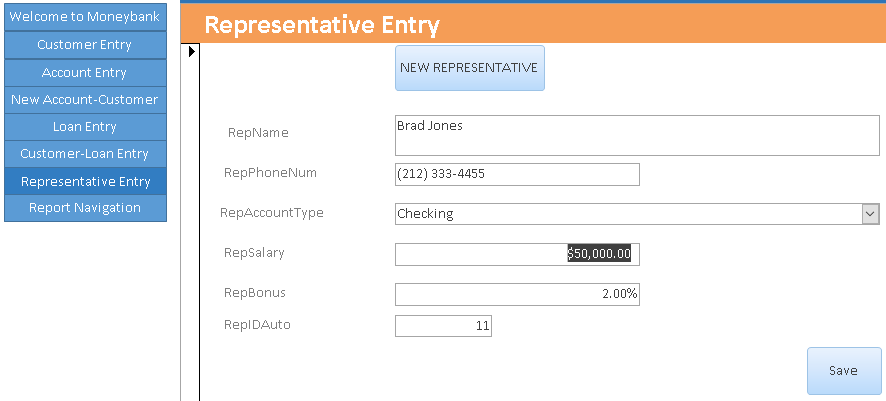
The above is the opening screenshot of the form. In order to make a new entry into the CustomerLoan table, the user needs to click the “New Loan” button, which will make the form’s fields blank:



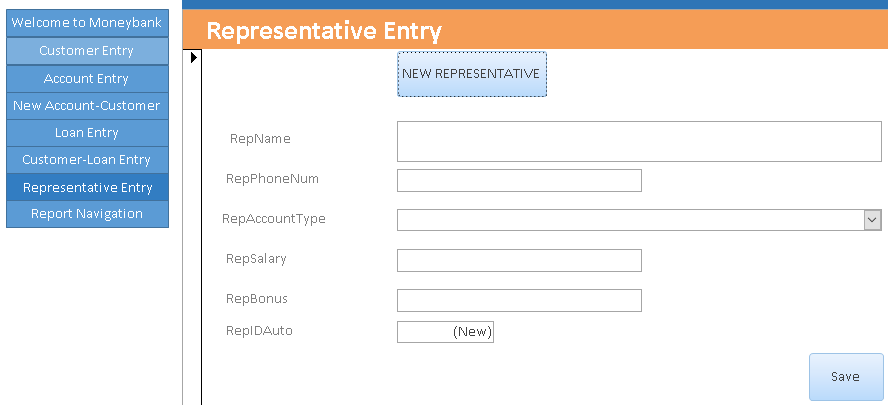
This form consists entirely of comboboxes because it is populated with the existing information. If any of the fields do not currently exist in the database, Customer Entry Form, Loan Entry Form, and/or Representative form needs to be used first to update the information in the necessary table. After the information is selected, the user has to press the “Save” button, which will save the new instance of customer-loan-representative into the CustomerLoan table.

Representative Entry:

This form by is used by the Bank Manager to record a new representative to the Money Bank database.



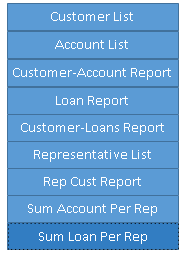
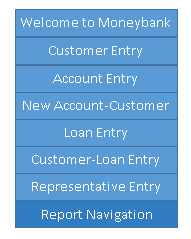
The above is the opening screenshot of the form. In order to add a new customer to the database, the user needs to click the “New Representative” button, which will make the form’s fields blank:



Note that the RepAccountType field is a combobox because it is limited only to five possibilities: “Mortgage”, “Personal Loan”, “Checking”, “Savings”, and “Credit”. After information is entered (RepIDAuto field is populated automatically), the user has to press the “Save” button, which will save the new representative into the Representative table.

Report Navigation:

In order to access the reports, a user must click on last navigation button, “Report Navigation”:



Customer List:

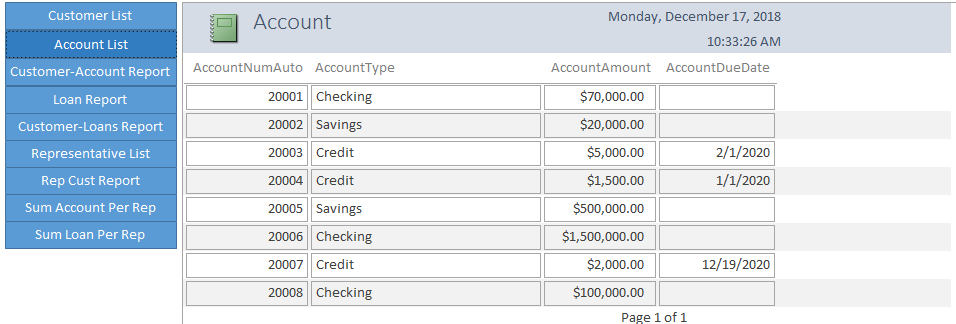


Clicking the Customer List navigation button will open the Customer report that shows every customer Money Bank has with its attributes: CustomerIDAuto, CustomerName, CustomerSSN, CustomerAddress, CustomerIncome, and CustomerJob.

This report is based on the Customer Table and is equivalent to the following query:

SELECT \* FROM Customer;

Account List:

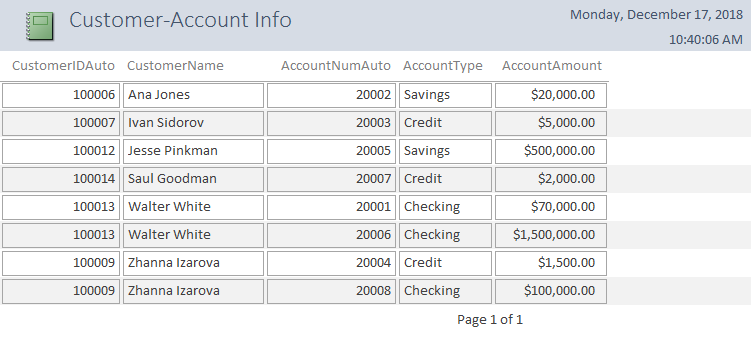


Clicking the Account List navigation button will open the Account report that shows every account Money Bank has with its attributes: AccountNumAuto, AccountType, AccountAmount, AccountDueDate.

This report is based on the Account Table and is equivalent to the following query:

SELECT \* FROM Account;

Customer-Account Info:



Clicking the Customer-Account Report navigation button will open the Customer-Account Info report that shows every customer Money Bank has with their account information (account number, account type, and the amount).

This report is based on the following query:

SELECT customer.customername, customer.customeridauto, customeraccount.accountnumauto, account.accounttype, account.accountamount

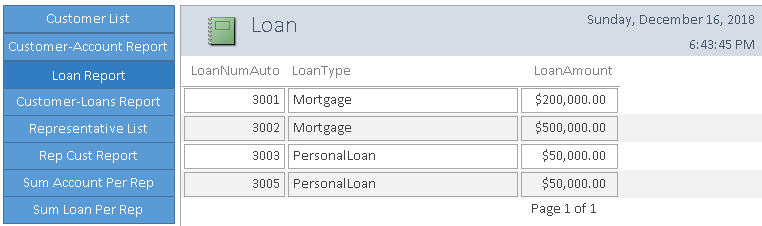
FROM customer, customeraccount, account

WHERE customer.customeridauto = customeraccount.customeridauto

AND account.accountnumauto = customeraccount.accountnumauto

GROUP BY customer.customername, customer.customeridauto, customeraccount.accountnumauto, account.accounttype, account.accountamount;

Loan Report:

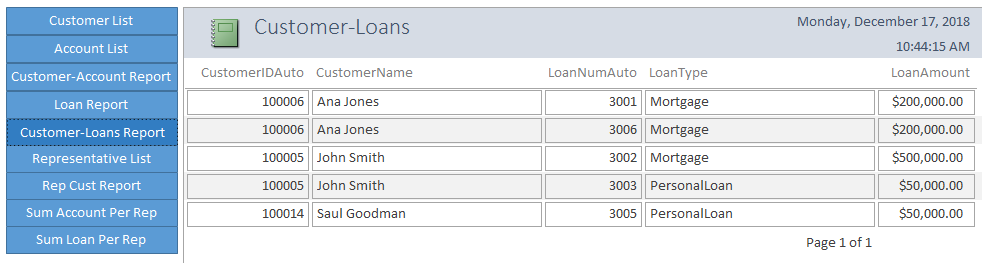


Clicking the Loan Report navigation button will open the Loan report that shows every loan Money Bank has issued with its attributes: LoanNumAuto, LoanType, and Loan Amount.

This report is based on the Loan Table and is equivalent to the following query:

SELECT \* FROM Loan;

Customer-Loans:



Clicking the Customer-Loans Report navigation button will open the Customer-Loans report that shows every customer Money Bank has with their loan information (loan number, loan type, and the amount).

This report is based on the following query:

SELECT customer.customername, customer.customeridauto, customerloan.loannumauto, loan.loantype, loan.loanamount

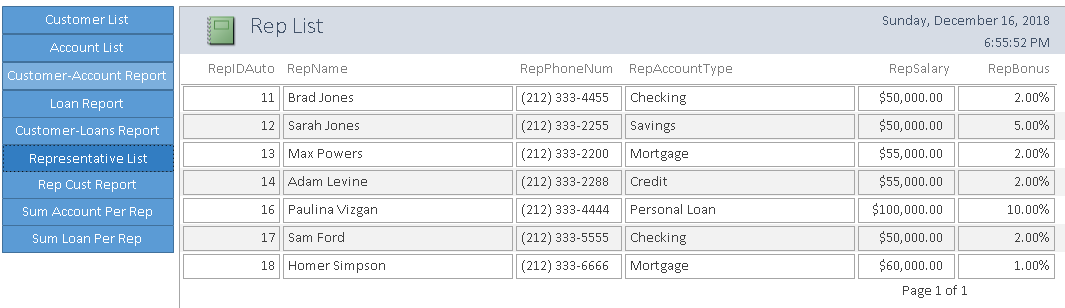
FROM customer, customerloan, loan

WHERE customer.customeridauto = customerloan.customeridauto

AND loan.loannumauto = customerloan.loannumauto

GROUP BY customer.customername, customer.customeridauto, customerloan.loannumauto, loan.loantype, loan.loanamount;

Representative List:

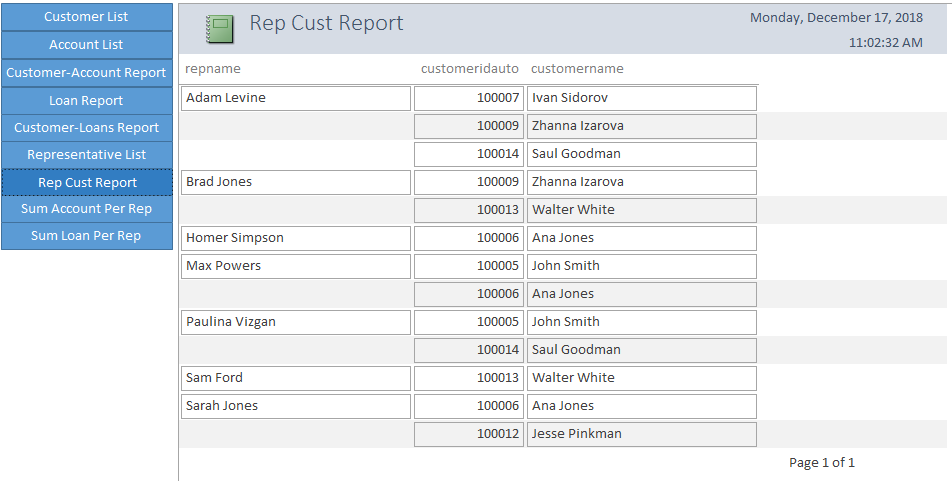


Clicking the Representative List navigation button will open the Rep List report that shows every representative currently employed by the Money Bank with its attributes: RepIDAuto, RepName, RepPhoneNum, RepAccountType, RepSalary, RepBonus;

This report is based on the Representative Table and is equivalent to the following query:

SELECT \* FROM Representative.;

Representative Customer Report:



Clicking the Rep-Cust Report navigation button will open the Representative Customer report that shows every customer Money Bank has per Representative.

This report is based on the following query:

SELECT r.repname, c.customeridauto, c.customername

FROM representative AS r, customer AS c, customeraccount AS ca, customerloan AS cl

WHERE r.repidauto = ca.repidauto

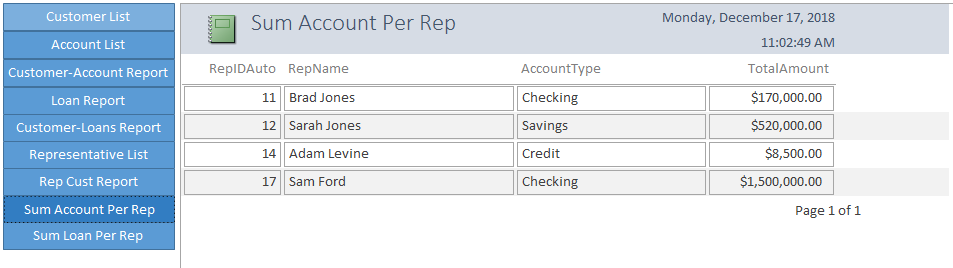
AND ca.customeridauto = c.customeridauto

OR r.repidauto = cl.repidauto

AND cl.customeridauto = c.customeridauto

GROUP BY r.repname, c.customeridauto, c.customername;

Summary of Accounts per Representative:



Clicking the Sum Account Per Rep navigation button will open the Sum Account Per Rep report that shows the total amount per account type and representative.

This report is based on the following query:

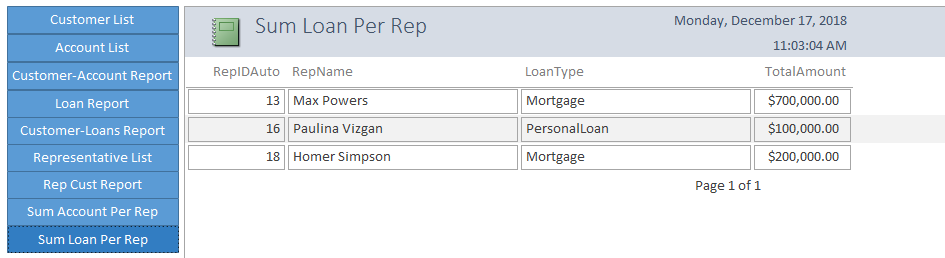
SELECT r.RepIDAuto, r.RepName, a.AccountType, sum(a.accountamount) AS TotalAmount

FROM (representative AS r INNER JOIN customeraccount AS ca ON r.repidauto = ca.repidauto)

INNER JOIN account AS a ON a.accountnumauto = ca.accountnumauto

GROUP BY r.RepIDAuto, r.RepName, a.AccountType;

Summary of Loans per Representative:



Clicking the Sum Loan Per Rep navigation button will open the Sum Loan Per Rep report that shows the total amount per loan type and representative.

This report is based on the following query:

SELECT r.RepIDAuto, r.RepName, l.LoanType, sum(l.loanamount) AS TotalAmount

FROM (representative AS r INNER JOIN customerloan AS cl ON r.repidauto = cl.repidauto) INNER JOIN loan AS l ON l.loannumauto = cl.loannumauto

GROUP BY r.RepIDAuto, r.RepName, l.loantype;

**VI Conclusion**

Group Experience:

We all have learned a lot during the process of building this project. Working on this project helped us immensely understand the material. Actually building a working database let us look at the whole course in a much more practical sense. We gained a better and deeper understanding on database building and what key aspects of the process we need to keep in mind for future use. It is a project where we were able to apply the information that we have been learning during the whole semester. Working on this project was especially challenging since neither one of us had previous strong CIS background. So, while we initially assigned specific duties to each other, the actual work was done together during the brainstorming meetings. After those meetings, we were able to split the next steps between each other to be productive. So the project also taught us to work efficiently as a team.

We did meet several challenges during the process. When we reached MileStone 5 - Application Development, we found out there were deficiencies in our design that we needed to address in order to make a more functional database. Since we realized it at the second to last step, we had to pull through two 12-hour meetings at the Baruch Computer Lab as well as pulling an all-nighter. We revised the proposal, the ERD, the Relational Model, Normalizations and the Application. Since we built on the database that we built in the first round, we did not rewrite the SQL used to build the database but included the revisions that we made during our final reworking of the database. It was a lot of work but considering the fact that we had zero Database building knowledge so we had to learn on our mistakes. But now we have a better understanding of the database creating process and, hopefully, will be able to abstain from making the same mistakes from the get-go.

Final Product Review:

Our group thinks that our final product is very usable it will work for a small bank. If it were a real life job, additional work will be needed for security: create roles, grant privileges etc. But we succeeded in having the final deliverable accomodate the needs specified in the proposal, which is also the most rewarding part of the whole project.

Conclusion:

It was a difficult project but it was very rewarding because it has real life application: it helped us understand how to build a successful database and how to work as a team.