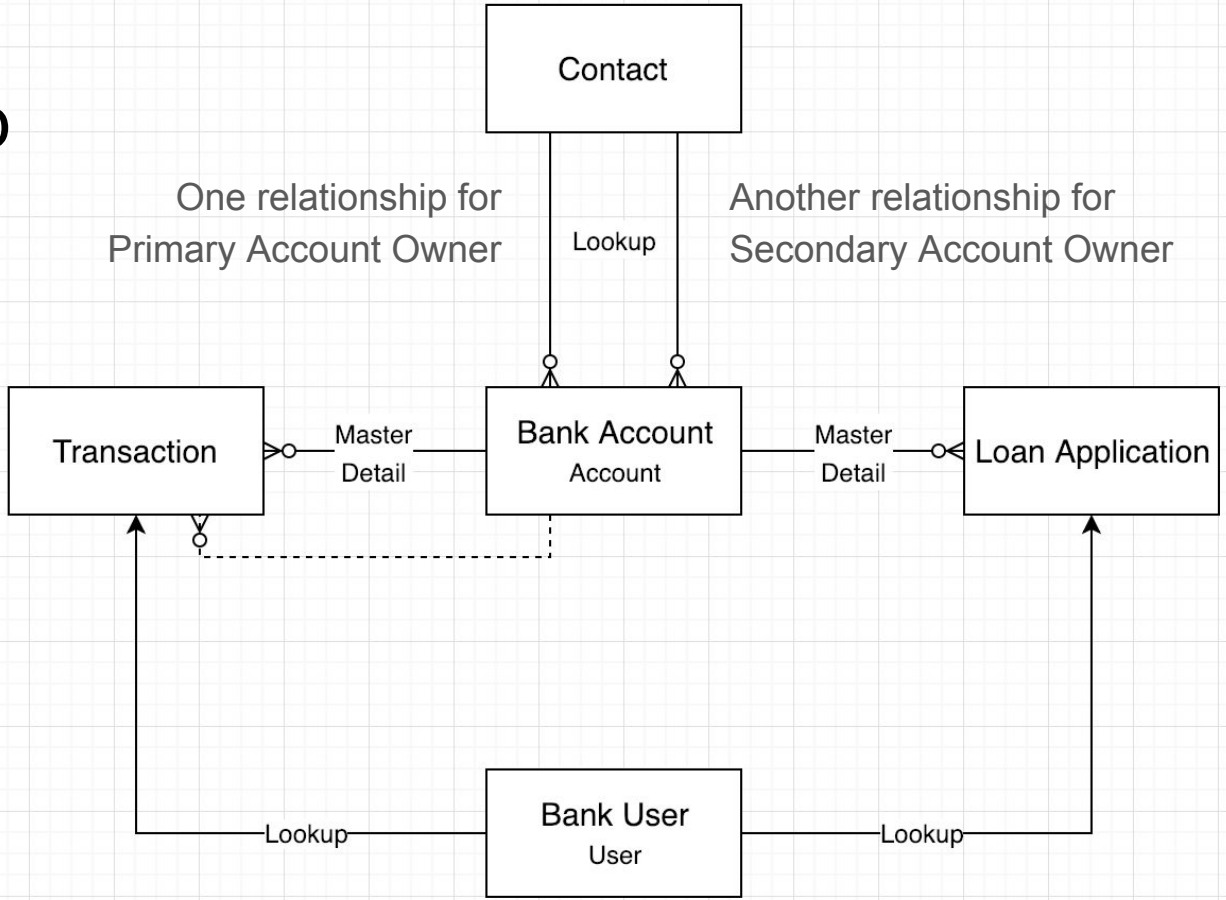
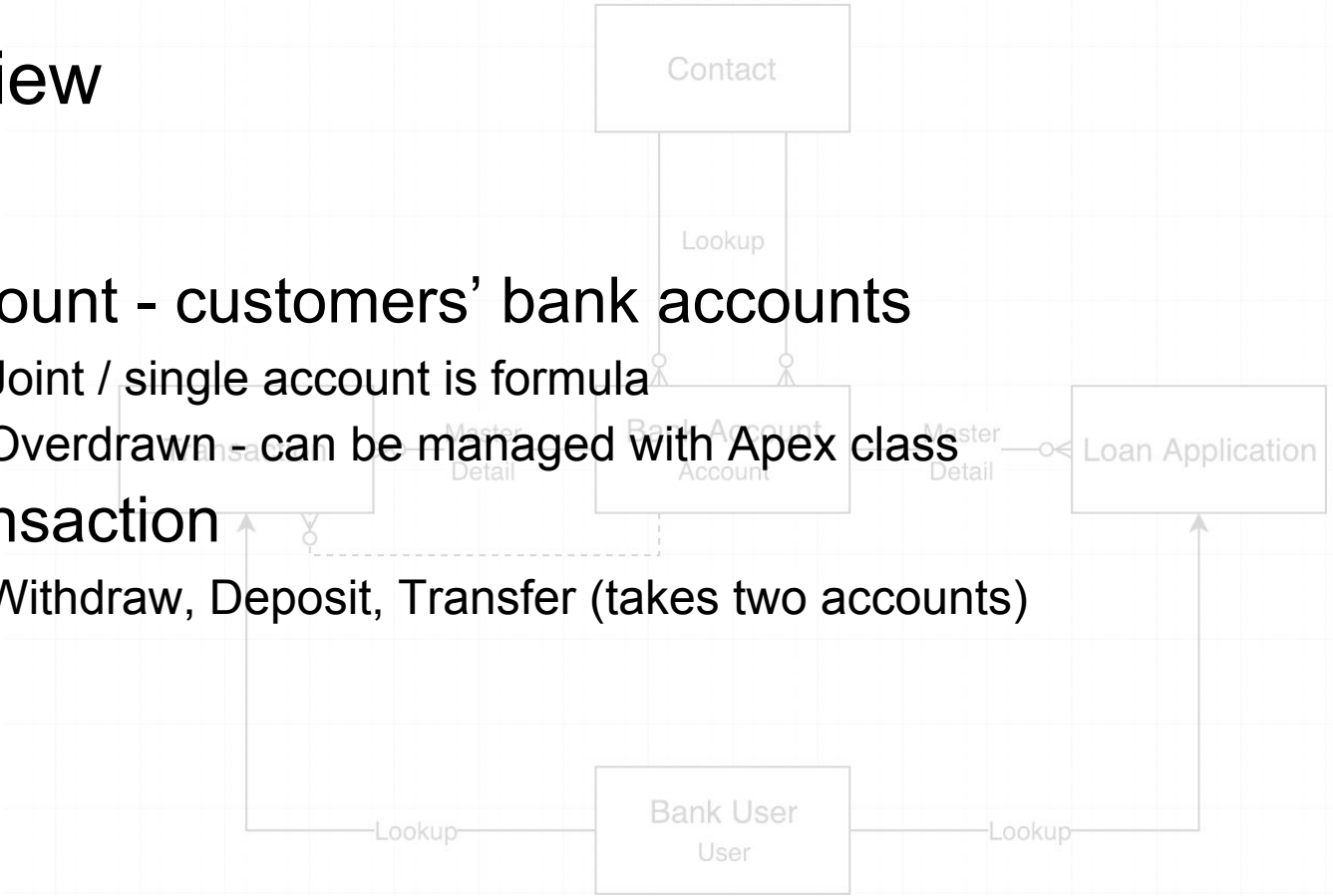
Project #0

Rafael E. López M.

Entity Relationship Diagram



Overview



- Account - customers’ bank accounts
  - Joint / single account is formula
  - Overdrawn - can be managed with Apex class
- Transaction
  - Withdraw, Deposit, Transfer (takes two accounts)

Security

- Org level
  - Hours: Mon-Sat, 6:00 A.M. - 9:00 P.M.
- Object level
  - Profile P0 Banker limited to Account, Contact, Loan Application, & Transaction
  - C.R.U.~~D.~~
- Record level
  - Controlled by parent: Loan Application, Transaction
  - Private Account
- Field level
  - Page Layout restricts access as necessary

Apex Functionality

```
26
27      /*
28       * Check if an account is overdrawn and change the status of the account accordingly.
29       */
30      public static void checkOverdrawn(){ ...
40      }
41      public static void checkOverDrawn(Account acc) { ...
44      }
45
46
47      /*
48       * Create a new transaction relating to the account.
49       */
50      public static void createTransaction(Decimal amt, String type, Account receiver){ ...
84      }
85      public static void createTransaction(Account acc, Decimal amt, String type) { ...
88      }
89      public static void createTransaction(Account acc, Decimal amt, String type, Account receiver) { ...
92      }
93
94
95      /*
96       * Update the account based on all related transactions to show current balance.
97       */
98      public static void updateBalance(){ ...
136     }
137     public static void updateBalance(Account acc) { ...
140     }
141
```

```
157
158     /*
159      * Test the methods written for the project.
160      */
161     public static void runTests(){
162         Boolean[] resultList = new List<Boolean>();
163         Integer passes = 0;
164
165         // checkOverdrawn()
166         resultList.add(test_checkOverdrawn_positiveBalance());
167         resultList.add(test_checkOverdrawn_negativeBalance());
168         resultList.add(test_checkOverdrawn_zeroBalance());
169
170         // createTransaction()
171         resultList.add(test_createTransaction_badWithdrawalAmount());
172         resultList.add(test_createTransaction_badWithdrawalReceiver());
173         resultList.add(test_createTransaction_goodWithdrawal());
174
175         resultList.add(test_createTransaction_badDeopsitAmount());
176         resultList.add(test_createTransaction_badDepositReceiver());
177         resultList.add(test_createTransaction_goodDeposit());
178
179         resultList.add(test_createTransaction_badTransferAmount());
180         resultList.add(test_createTransaction_badTransferReceiver());
181         resultList.add(test_createTransaction_goodTransfer());
182
183         // updateBalance()
184         // resultList.add(test_updateBalance_goodWithdrawal());
185         // resultList.add(test_updateBalance_overdrawingWithdrawal());
186         // resultList.add(test_updateBalance_Deposit());
187         // resultList.add(test_updateBalance_overdrawFixingDeposit());
188         // resultList.add(test_updateBalance_goodTransfer());
189         // resultList.add(test_updateBalance_overdrawingTransfer());
190         // resultList.add(test_updateBalance_receivingTransfer());
191         // resultList.add(test_updateBalance_receivingOverdrawFixingTransfer());
192
193         for(Boolean result: resultList){
194             passes += result ? 1 : 0;
195         }
196         System.debug(String.valueOf(passes) + ' tests out of ' + String.valueOf(resultList.size()) + 'passed.');
197     }
198
```

```
157
158     /*
159      * Test the methods written for the project.
160      */
161     public static void runTests(){
162         Boolean[] resultList = new List<Boolean>();
163         Integer passes = 0;
164
165         // checkOverdrawn()
166         resultList.add(test_checkOverdrawn_positiveBalance());
167         resultList.add(test_checkOverdrawn_negativeBalance());
168         resultList.add(test_checkOverdrawn_zeroBalance());
169
170         // createTransaction()
171         resultList.add(test_createTransaction_badWithdrawalAmount());
172         resultList.add(test_createTransaction_badWithdrawalReceiver());
173         resultList.add(test_createTransaction_goodWithdrawal());
174
175         resultList.add(test_createTransaction_badDeopsitAmount());
176         resultList.add(test_createTransaction_badDepositReceiver());
177         resultList.add(test_createTransaction_goodDeposit());
178
179         resultList.add(test_createTransaction_badTransferAmount());
180         resultList.add(test_createTransaction_badTransferReceiver());
181         resultList.add(test_createTransaction_goodTransfer());
182
183         // updateBalance()
184         // resultList.add(test_updateBalance_goodWithdrawal());
185         // resultList.add(test_updateBalance_overdrawingWithdrawal());
186         // resultList.add(test_updateBalance_Deposit());
187         // resultList.add(test_updateBalance_overdrawFixingDeposit());
188         // resultList.add(test_updateBalance_goodTransfer());
189         // resultList.add(test_updateBalance_overdrawingTransfer());
190         // resultList.add(test_updateBalance_receivingTransfer());
191         // resultList.add(test_updateBalance_receivingOverdrawFixingTransfer());
192
193         for(Boolean result: resultList){
194             passes += result ? 1 : 0;
195         }
196         System.debug(String.valueOf(passes) + ' tests out of ' + String.valueOf(resultList.size()) + 'passed.');
197     }
198
```

Execution Log

Timestamp	Event	Details
07:14:57:112	USER_DEBUG	[226]|DEBUG|true - Overdrawn__c is not set when balance is > 0
07:14:57:319	USER_DEBUG	[257]|DEBUG|true - Overdrawn__c is set when balance is < 0
07:14:57:506	USER_DEBUG	[287]|DEBUG|true - Overdrawn__c is not set when balance is = 0
07:14:57:664	USER_DEBUG	[58]|DEBUG|A transaction's ammount must be a positive quantity.
07:14:57:672	USER_DEBUG	[328]|DEBUG|true - Given a non-positive amount, withdrawal transaction isn't created
07:14:57:845	USER_DEBUG	[66]|DEBUG|Transactions that are not of type Transfer cannot have a Transfer Receiver account.
07:14:57:853	USER_DEBUG	[367]|DEBUG|true - Given a receiver account, withdrawal transaction isn't created
07:14:58:035	USER_DEBUG	[400]|DEBUG|true - Given valid parameters, withdrawal transaction is created
07:14:58:209	USER_DEBUG	[58]|DEBUG|A transaction's ammount must be a positive quantity.
07:14:58:216	USER_DEBUG	[434]|DEBUG|true - Given bad amount, deposit transaction isn't created
07:14:58:389	USER_DEBUG	[66]|DEBUG|Transactions that are not of type Transfer cannot have a Transfer Receiver account.
07:14:58:397	USER_DEBUG	[473]|DEBUG|true - Given a receiver account, deposit transaction isn't created
07:14:58:576	USER_DEBUG	[506]|DEBUG|true - Given valid parameters, deposit transaction is created
07:14:58:758	USER_DEBUG	[58]|DEBUG|A transaction's ammount must be a positive quantity.
07:14:58:767	USER_DEBUG	[546]|DEBUG|true - Given invalid amount, transfer transaction is not created
07:14:58:944	USER_DEBUG	[70]|DEBUG|Transfer type transactions must have a Transfer Receiver account.
07:14:58:952	USER_DEBUG	[579]|DEBUG|true - Not given a receiver, transfer transaction is not created
07:14:59:151	USER_DEBUG	[618]|DEBUG|true - Given valid paramenters, transfer transaction is created
07:14:59:293	USER_DEBUG	[196]|DEBUG|12 tests out of 12passed.

Execution Log

Timestamp	Event	Details
07:14:57:112	USER_DEBUG	[226]|DEBUG|true - Overdrawn__c is not set when balance is > 0
07:14:57:319	USER_DEBUG	[257]|DEBUG|true - Overdrawn__c is set when balance is < 0
07:14:57:506	USER_DEBUG	[287]|DEBUG|true - Overdrawn__c is not set when balance is = 0
07:14:57:664	USER_DEBUG	[58]|DEBUG|A transaction's ammount must be a positive quantity.
07:14:57:672	USER_DEBUG	[328]|DEBUG|true - Given a non-positive amount, withdrawal transaction isn't created
07:14:57:845	USER_DEBUG	[66]|DEBUG|Transactions that are not of type Transfer cannot have a Transfer Receiver account.
07:14:57:853	USER_DEBUG	[367]|DEBUG|true - Given a receiver account, withdrawal transaction isn't created
07:14:58:035	USER_DEBUG	[400]|DEBUG|true - Given valid parameters, withdrawal transaction is created
07:14:58:209	USER_DEBUG	[58]|DEBUG|A transaction's ammount must be a positive quantity.
07:14:58:216	USER_DEBUG	[434]|DEBUG|true - Given bad amount, deposit transaction isn't created
07:14:58:389	USER_DEBUG	[66]|DEBUG|Transactions that are not of type Transfer cannot have a Transfer Receiver account.
07:14:58:397	USER_DEBUG	[473]|DEBUG|true - Given a receiver account, deposit transaction isn't created
07:14:58:576	USER_DEBUG	[506]|DEBUG|true - Given valid parameters, deposit transaction is created
07:14:58:758	USER_DEBUG	[58]|DEBUG|A transaction's ammount must be a positive quantity.
07:14:58:767	USER_DEBUG	[546]|DEBUG|true - Given invalid amount, transfer transaction is not created
07:14:58:944	USER_DEBUG	[70]|DEBUG|Transfer type transactions must have a Transfer Receiver account.
07:14:58:952	USER_DEBUG	[579]|DEBUG|true - Not given a receiver, transfer transaction is not created
07:14:59:151	USER_DEBUG	[618]|DEBUG|true - Given valid paramenters, transfer transaction is created
07:14:59:293	USER_DEBUG	[196]|DEBUG|12 tests out of 12passed.

Rafael López

Revature

Achievements

Badges	Points	Trails Completed
0.5	¥21,500,500	√-1

Badges (15 of 15)

All Badges

Security Specialist

User Authentication

Identity Basics

Data Security

Rafael López

Revature

Achievements

Badges	Points	Trails Completed
15	21,500	1

Badges (15 of 15)

All Badges

Security Specialist

User Authentication

Identity Basics

Data Security

Demo Time