Banking System@Customer Centric

Owner : Deepthi Manam
NU ID: 001743609

INFO 6210 DATABASE MANAGEMENT

Problem Definition

You are a database architect working for Deepthi Software Sol. The VP of the company has asked you to design a database for NorthEastern Bank. The design should adhere to all the database standards and should be able to capture information about the customers, transactions made by customer, security issues, services requests raised by customers. The design should be able to capture the following information.

- ✓ Customer Information – Able to capture/store information about the customer like name, ssn, date of birth, income etc.

- ✓ Capture Login Details –A customer logins to his/her account many times, performs various operations.

- ✓ Account- A bank account is a financial account between a customer and a financial institution.
  - A customer can have many accounts in a bank.
  - An account can be associated with many customers(Joint Account)

- ✓ Transaction Information –Every credit/debit transaction associated with the customer account should be recorded.

- ✓ Alert Information -A customer can be alerted by sms or email, when any transaction happen in his/her account or when the account balance falls below specified amount

- ✓ Service Requests Information – A customer can raise a request for some service or complaint to the bank. A bank employee works on the request raised by the customer and updates the status of the service.

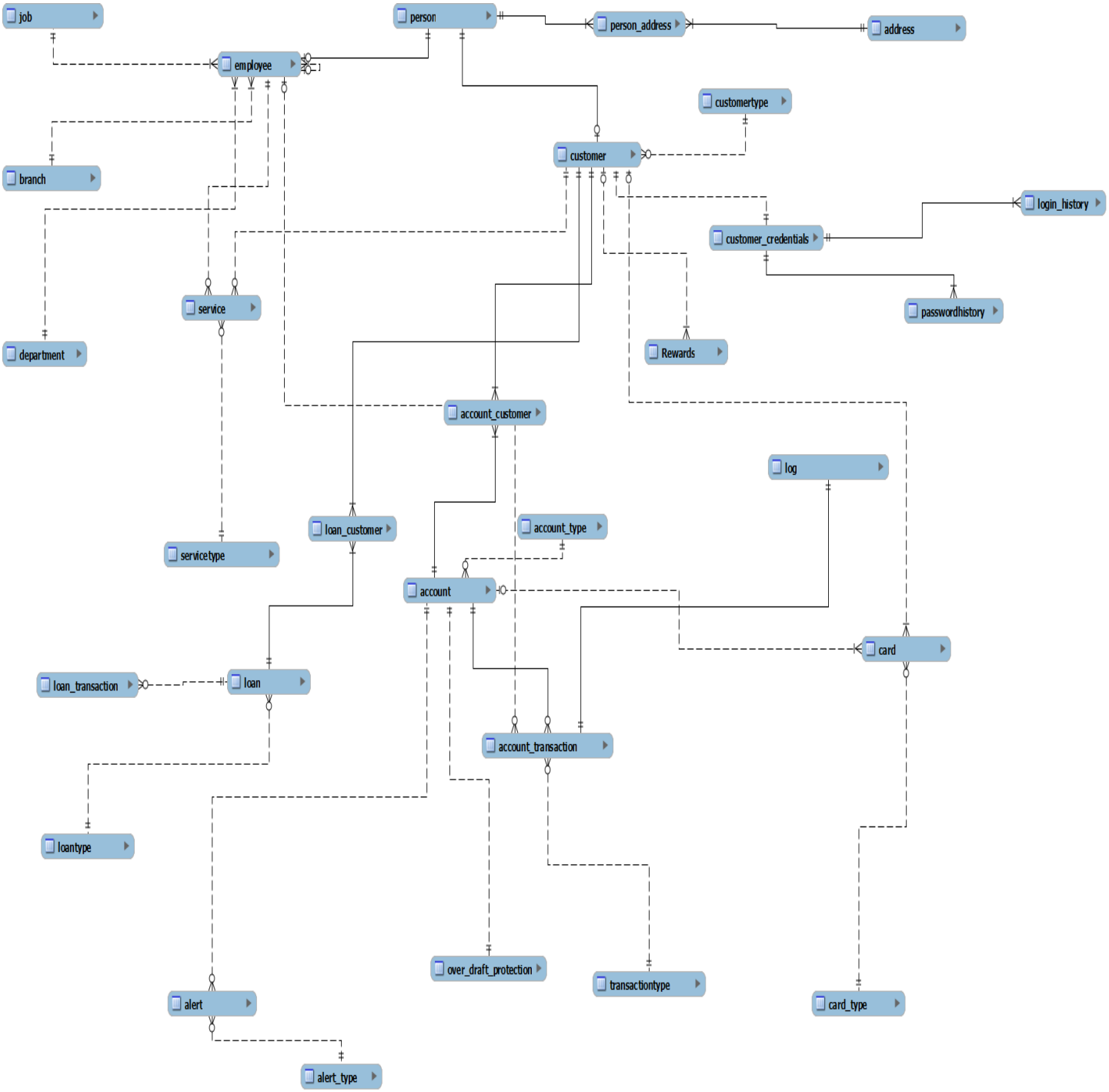
- ✓ Cards Information – A customer can possess many cards like debit, credit etc.

- ✓ Loan Transactions –A bank can lend money to the customer. The customer has to pay the EMI monthly to clear the loan.
  - A customer can take many Loans
  - A loan can be shared by many customers

- ✓ Security Information-According to the password reset policy. The password should not be same as the most recent 5 passwords.

Initial Draft Model:

The initial draft model for the banking system is as follows:



Assumptions:

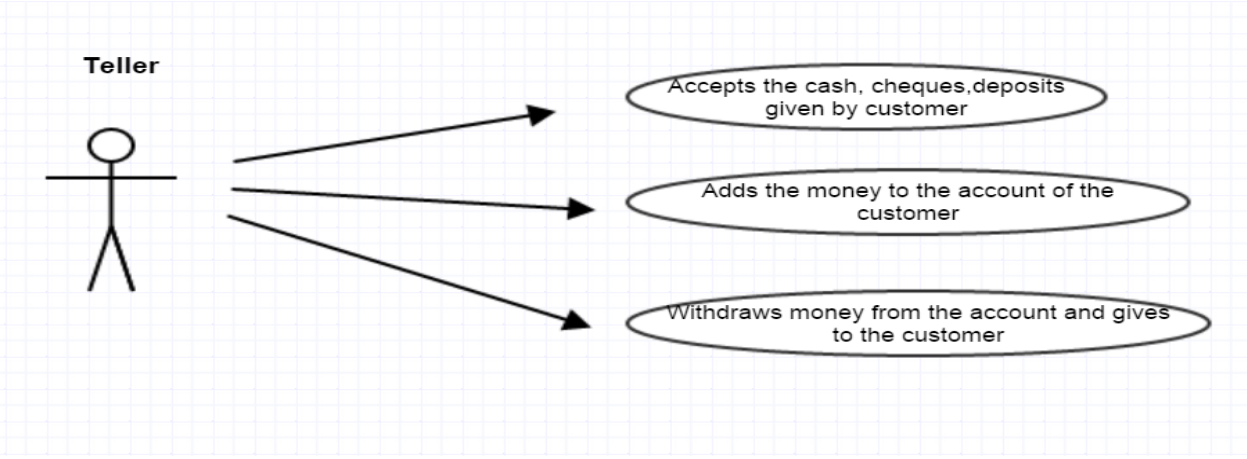
The following are assumptions:

- A customer can login to the online banking system and can update personal information like address and phone number.
- A customer will be given four secret questions to answer, at the time of creating an account.
- An account can be associated with more than one person.
- A customer can have many addresses.
- A customer can have many accounts in the bank
- If the customer goes to the bank, to deposit money in his account, the teller is involved in the transaction process.
- If the customer does an online money transfer, then the teller will not be involved in the transaction.
- A customer can take many loans
- A loan can be shared by many customers.

User Roles

This database has to support many roles as a part of integrated end user application. The responsibilities and privileges of the users are discussed below:

Teller: When a customer does manual transfer of money, the teller facilitates the process. Responsible for handling customer transactions at banks, including taking deposits, disbursing cash.



Creating Teller

CREATE USER 'Ratna'@'localhost' IDENTIFIED BY 'Ratna';

Privileges of Teller

GRANT SELECT ON bankingsystem.person TO 'Ratna'@'localhost';

GRANT SELECT ON bankingsystem.customer TO 'Ratna'@'localhost';

GRANT SELECT ON bankingsystem.account_customer TO 'Ratna'@'localhost';

GRANT SELECT ON bankingsystem.account TO 'Ratna'@'localhost';

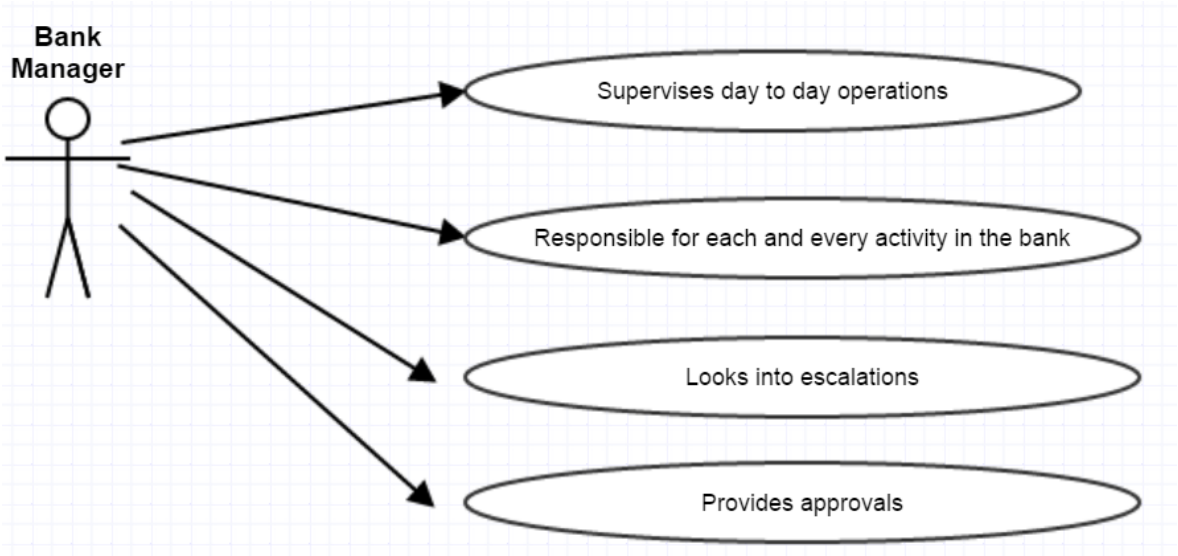
GRANT SELECT ON bankingsystem.account_type TO 'Ratna'@'localhost';

GRANT SELECT ON bankingsystem.account_transaction TO 'Ratna'@'localhost';

GRANT INSERT ON bankingsystem.account_transaction TO 'Ratna'@'localhost';

GRANT UPDATE ON bankingsystem.account TO 'Ratna'@'localhost';

Bank Manager: The bank manager is responsible for managing the bank. All the employees in the bank report to the manager.



Creating Bank Manger

CREATE USER 'Tom'@'localhost' IDENTIFIED BY 'Tom';

Privileges of Bank Manger

SELECT on person, employee, branch, job, department, service, service_type, loan, loan_customer, loan_transaction, loan_type, customer, customer_account, account, account_transaction, overdraft_protection, alert, alert_type, transactiontype, rewards, person_address, customer_type, log, card, card_type, address

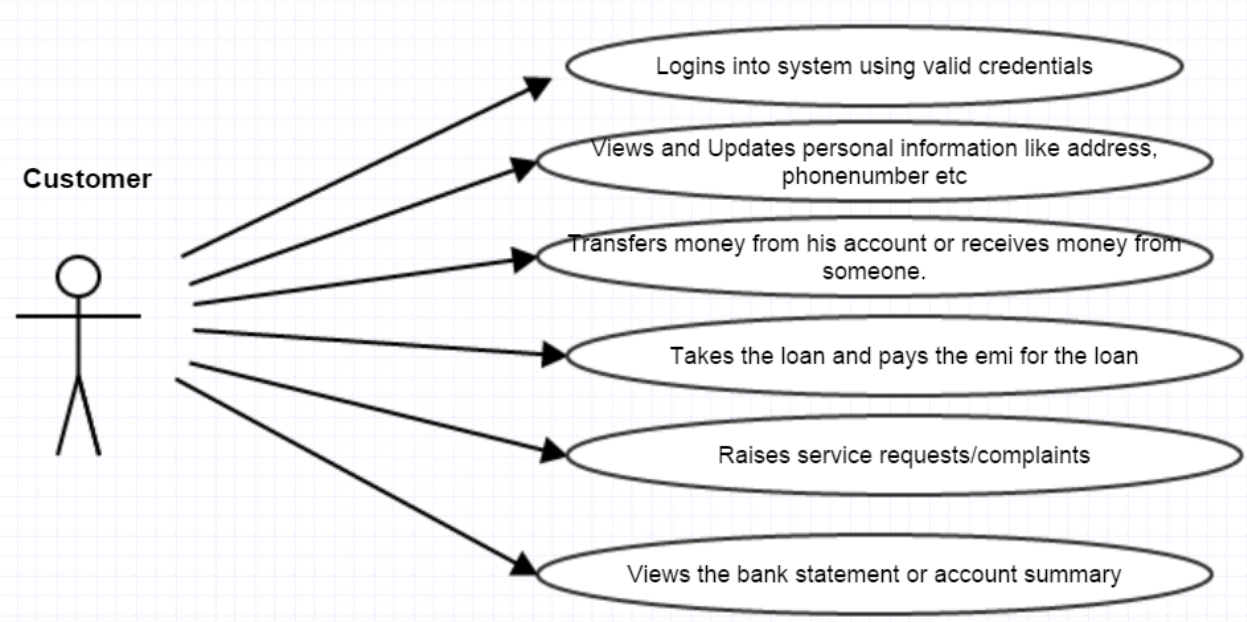
INSERT on employee, person, branch, job, department, service, service_type, loan, loan_customer, loan_transaction, loan_type, customer, customer_account, account, account_transaction, overdraft_protection, alert, alert_type, transactiontype, rewards, person_address, customer_type, log, card, card_type, address

UPDATE on employee, person, branch, job, department, service, service_type, loan, loan_customer, loan_transaction, loan_type, customer, customer_account, account,

account_transaction, overdraft_protection, alert, alert_type, transactiontype, rewards, person_address, customer_type, log, card, card_type, address

GRANT ALL ON bankingsystem.* TO 'Tom'@'localhost';

Customer: Customer is the person, who holds an account in the bank and performs various activities.



Creating a customer

CREATE USER 'DeepthiM'@'localhost' IDENTIFIED BY 'DeepthiM;

Privileges of Customer

GRANT SELECT ON bankingsystem.person TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.customer TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.address TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.service TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.account TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.account_transaction TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.loan TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.loan_transaction TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.loan_transaction TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.account_type TO 'DeepthiM'@'localhost';

GRANT SELECT ON bankingsystem.over_draft_protection TO ‘DeepthiM’@’localhost’;

GRANT SELECT ON bankingsystem.transactiontype TO ‘DeepthiM’@’localhost’;

GRANT SELECT ON bankingsystem.rewards TO ‘DeepthiM’@’localhost’;

GRANT SELECT ON bankingsystem.card TO ‘DeepthiM’@’localhost’;

GRANT SELECT ON bankingsystem.cardType TO ‘DeepthiM’@’localhost’;

GRANT INSERT ON bankingsystem.service TO ‘DeepthiM’@’localhost’;

GRANT INSERT ON bankingsystem.person TO ‘DeepthiM’@’localhost’;

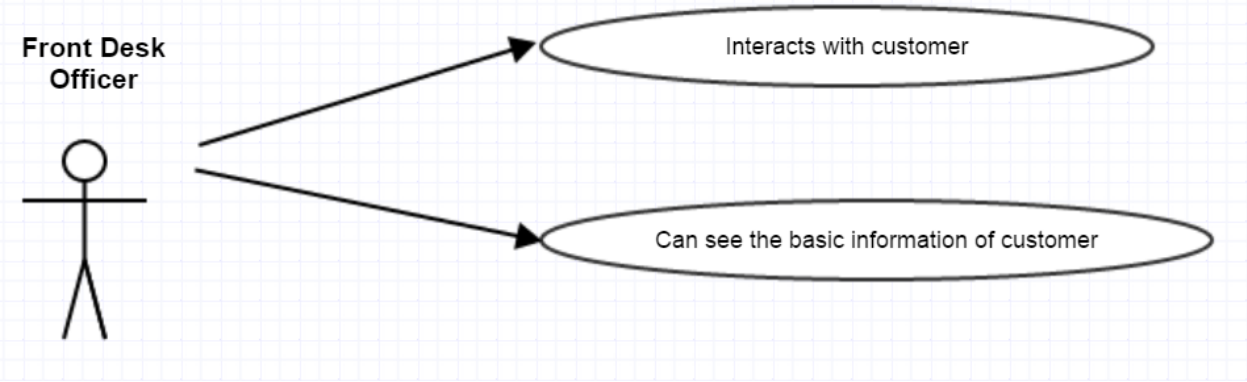
GRANT INSERT ON bankingsystem.customer_credentials TO ‘DeepthiM’@’localhost’;

GRANT UPDATE ON bankingsystem.address TO ‘DeepthiM’@’localhost’;

GRANT UPDATE ON bankingsystem.person TO ‘DeepthiM’@’localhost’;

Front desk officer

The front desk officer serves as the face of her/his employer. The primary responsibilities of front desk officer is answer the incoming calls, welcome the customers and route them to the proper recipient. She/he may also answer basic questions regarding the business, such as hours of operation.



Create Front Desk Officer

CREATE USER 'Michael'@'localhost' IDENTIFIED BY ‘Michael’;

Privileges of Front desk officer

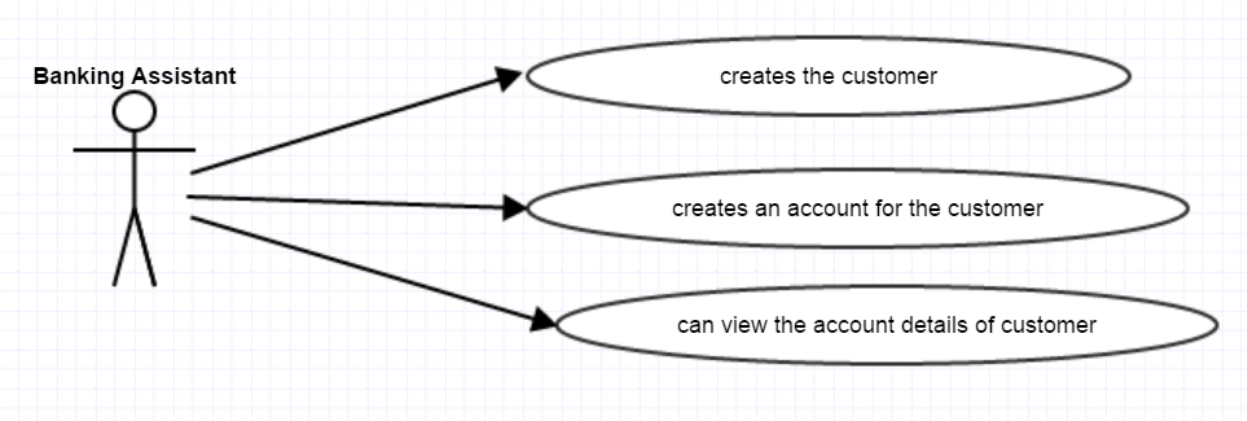
GRANT SELECT ON bankingsystem.person TO ‘Michael’@’localhost’;

GRANT SELECT ON bankingsystem.address TO ‘Michael’@’localhost’;

GRANT SELECT ON bankingsystem.customer TO ‘Michael’@’localhost’;

Banking Assistant

Banking assistant responsibilities include creating the customer, creating the account for the customer and can view the details of customer account.



Create Banking Assistant

CREATE USER 'Lincoln'@'localhost' IDENTIFIED BY 'Lincoln';

Privileges of Banking Assistant

GRANT SELECT ON bankingsystem.person TO 'Lincoln'@'localhost';

GRANT SELECT ON bankingsystem.customer TO 'Lincoln'@'localhost';

GRANT SELECT ON bankingsystem.address TO 'Lincoln'@'localhost';

GRANT SELECT ON bankingsystem.account TO 'Lincoln'@'localhost';

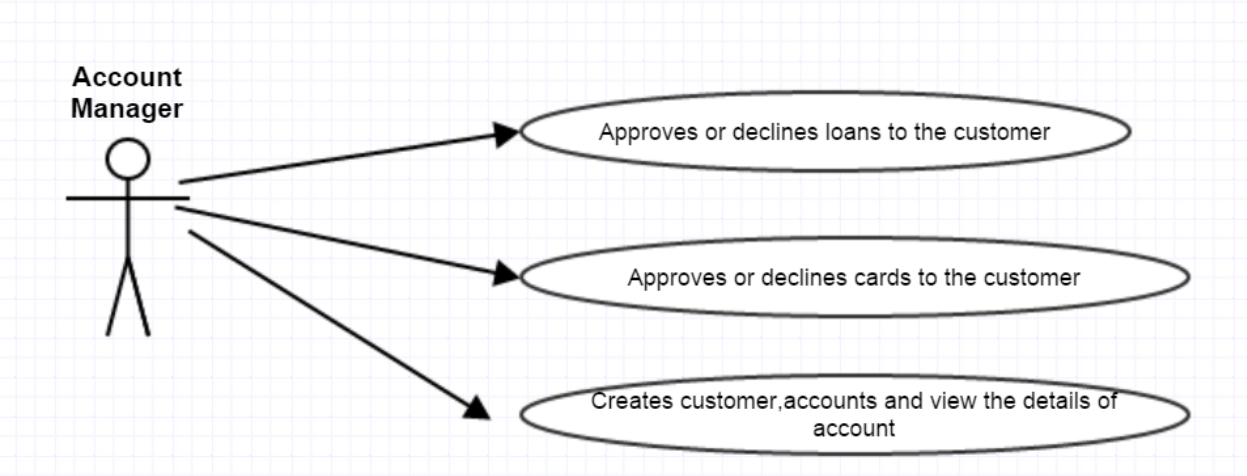
GRANT SELECT ON bankingsystem.account_transaction TO 'Lincoln'@'localhost';

GRANT UPDATE ON bankingsystem.person TO 'Lincoln'@'localhost';

GRANT UPDATE ON bankingsystem.address TO 'Lincoln'@'localhost';

Account Manager

The Account Manager is responsible for loan approvals, credit card approvals, create the customers and accounts and view the details of the customer accounts.



Creating Account Manager

CREATE USER 'John'@'localhost' IDENTIFIED BY 'John';

Privileges of Account Manager

GRANT SELECT ON bankingsystem.person TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.customer TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.address TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.account TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.account_transaction TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.loan TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.loan_transaction TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.card TO 'John'@'localhost';

GRANT SELECT ON bankingsystem.rewards TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.person TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.customer TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.address TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.account TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.account_transaction TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.loan TO 'John'@'localhost';

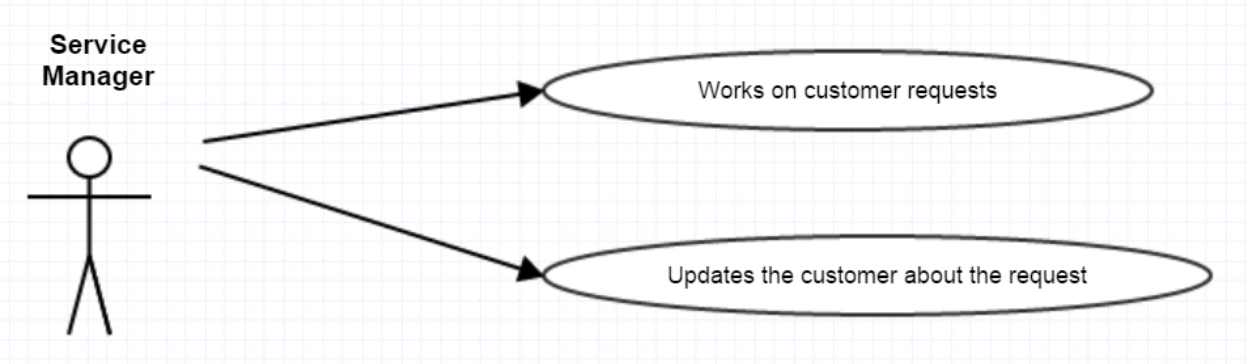
GRANT INSERT, UPDATE ON bankingsystem.loan_transaction TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.card TO 'John'@'localhost';

GRANT INSERT, UPDATE ON bankingsystem.customer_rewards TO 'John'@'localhost';

Service Manager

The service manager works on the requests raised by customers like Dispute transaction, Request for checks, Request for deposit slips, Stop payment on cheque etc.



Creating Service Manager User

CREATE USER 'Kim'@'localhost' IDENTIFIED BY 'Kim;

Privileges of Service Manager

GRANT SELECT ON bankingsystem.customer TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.person TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.service TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.account TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.account_transaction TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.loan TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.loan_transaction TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.card TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.rewards TO 'Kim'@'localhost';

GRANT SELECT ON bankingsystem.customer_rewards TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.customer TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.person TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.service TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.account TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.account_transaction TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.loan TO 'Kim'@'localhost';

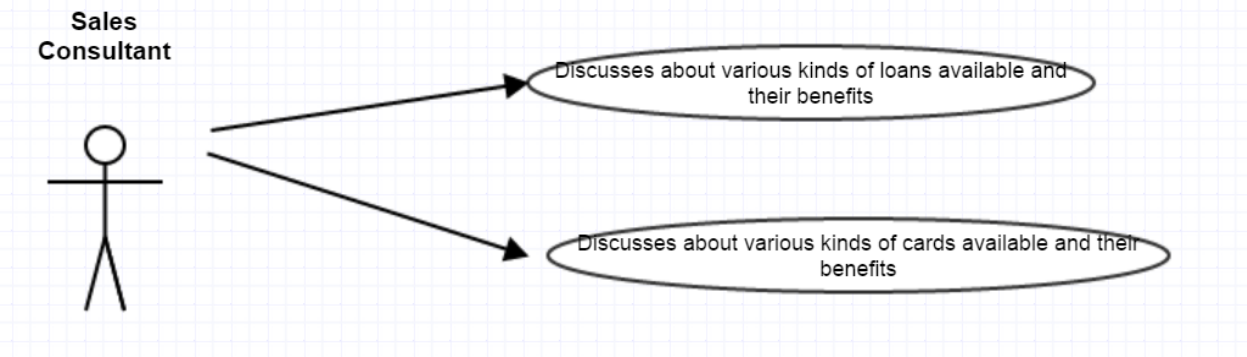
GRANT UPDATE ON bankingsystem.loan_transaction TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.card TO 'Kim'@'localhost';

GRANT UPDATE ON bankingsystem.customer_rewards TO 'Kim'@'localhost';

Sales Assistant

The sales assistant is responsible for promoting different products offered by the bank.



Creating Sales Assistant User

CREATE USER 'Richard'@'localhost' IDENTIFIED BY 'Richard;

Privileges of Sales Assistant

GRANT SELECT ON bankingsystem.customer TO 'Richard'@'localhost';

GRANT SELECT ON bankingsystem.person TO 'Richard'@'localhost';

GRANT SELECT ON bankingsystem.loanType TO 'Richard'@'localhost';

GRANT SELECT ON bankingsystem.cardType TO 'Richard'@'localhost';

Entities in the database

1) customer
   The customer table has the details of customer like first name, last name, date of birth, address, phone number, income etc.

2) customer_type
   The customer_type table contains the information about the type of customer. A customer can be an individual or a corporate.

3) address
   Details of the customer address are store in the address table.

4) password_history
   In order to enforce the password reset policy, the passwords of the customer are stored in the password_history table.

5) employee
   Details of the employees working in the bank are stored in the employee table.

6) Branch
   The branch entity is used to store the details of the bank like branchId, branchName, location etc

7) Department
   The department entity contains the details of the departments present in the bank like human resource, finance etc.

8) job
   The job entity contains the details of various job codes in the bank

9) card
   The details of the cards like credit/debit owned by the customer are stored in this entity.

10) card_type

    The details of the types of the card like total credit line, cash credit line are stored in this table.

11) Service
    The service requests details raised by the customer are stored

12) Service_type
    Details of types of services offered are stored in service entity

13) Person
    Person entity contains the general information of the customers and employees in the bank.

14) Account
    Account entity contains the details of account like balance, creation date, account type etc

15) Loan
    Loan entity contains the details of the loans such as loantype, amount , apr, emi etc

16) Loan_type
    Loan_type entity contains the details of different types of loans.

17) Loan_transaction
    This entity contains the transaction details of the loan like amount paid, date of transaction etc.

18) Account_type
    This entity contains details of different types of accounts

19) Account_transaction
    Account_transaction table contains the details of transactions made by the customer

20) Over_draft_protection
    Over_draft_protection entity contains the details of whether an overdraft protection is enabled for the account or not.

21) Transaction_type
    Transaction_type entity contains the details of different kinds of transactions like online transfer, credit card, debit card, manual, ATM withdrawal, wire transfer etc.

22) Rewards
    Rewards entity contains the details of customer rewards

23) Alert
    Alert table contains the details of the alert message sent to the customer, when any transaction happen.

24) Alert_type

    Alert_type contains the details of different types of alerts like sms, email etc.

25) Account_customer
    Account_customer table maps the account to the customer.

26) Log
    Every transaction made is logged into the log table

27) Person_address
    Person_address entity maps the person and his/her address details

28) Login_history
    Login_history entity contains the details of the customer login and logout times.

29) Loan_customer
    Loan_customer table contains the details of loans taken by customer.

30) Customer_rewards
    Customer_rewards table contains the rewards associated with the customer.

31) Customer_Credentials
    Customer_credentials table contain the credential associated with the customer.

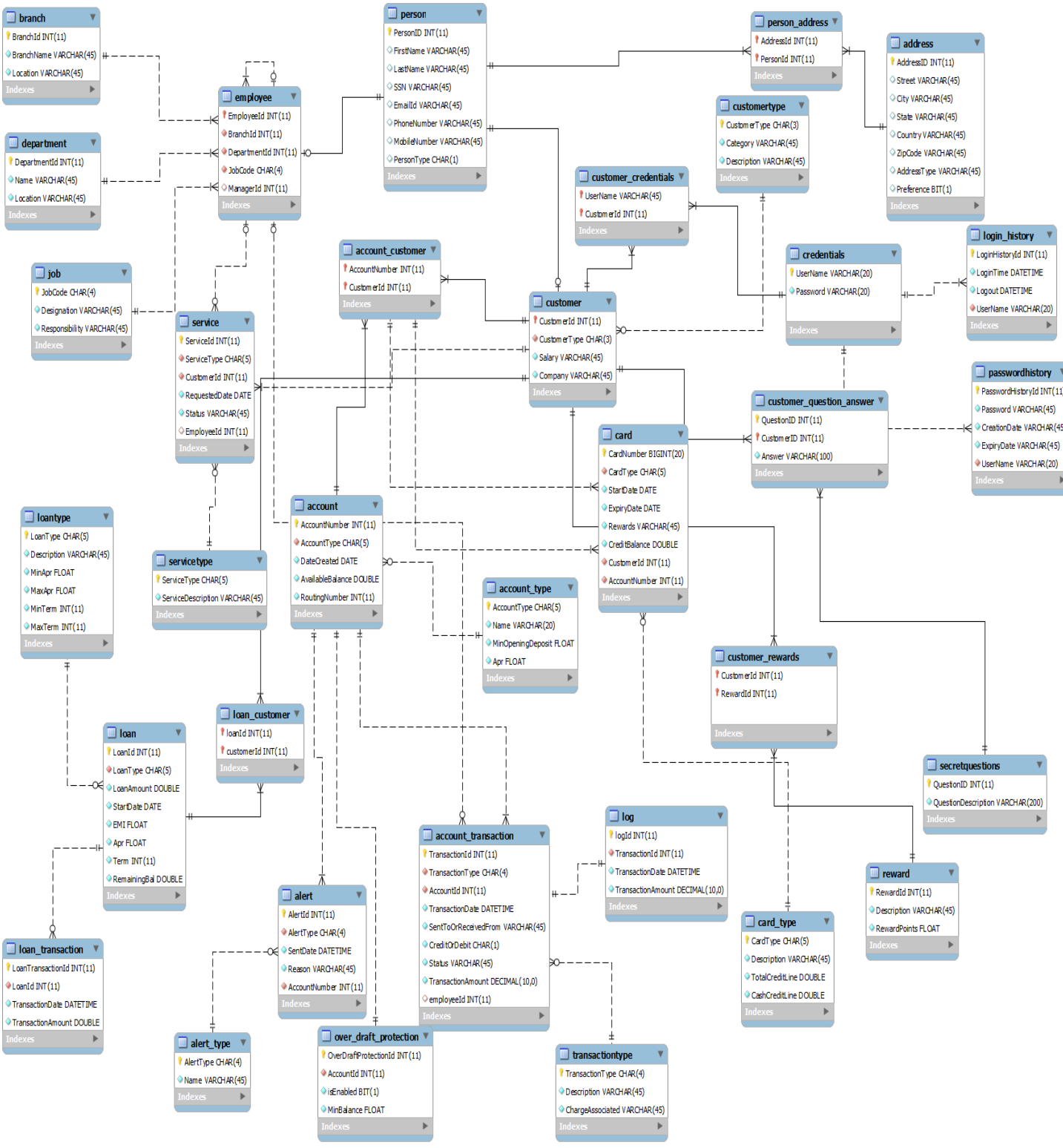
32) Customer_question_answer
    Customer_question_answer table contains the details of the answers provided by the customer for the secret questions.
33) SecretQuestions
    SecretQuestions table contains the list of secret questions given to the customer, at the time of creation of account.
34) Credentials
    Crendentials table contains the customer credentials to login to the online application.

Entity Relationships

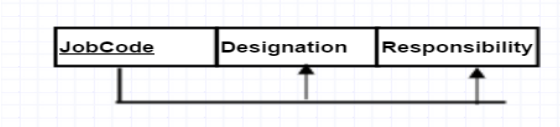
Following list describes the important relationship between the entities in the banking system:

1) Customer and Address	**(Many to Many)**
2) Customer and Account	**(Many to Many)**
3) Customer and loan	**(Many to Many)**
4) Customer and Service	**(One to Many)**
6) CustomerType and Customer	**(One to Many)**
7) Customer and rewards	**(Many to Many)**
8) Customer and cards	**(One to Many)**
9) Account and Account_Transaction	**(One to Many)**
10) Transactiontype and Account_Transaction	**(One to Many)**
11) loantype and Loan	**(One to Many)**
12) CardType and card	**(One to Many)**
13) Account_Transaction and Log	**(One to One)**
14) Customer and SecretQuestions	**(Many to Many)**
15) ServiceType and Service	**(One to Many)**
16) Employee and Service	**(One to Many)**
17) Department and employee	**(One to Many)**
18) Job and Employee	**(One to Many)**
19) Branch and Employee	**(One to Many)**
20) Employee and Employee	**(One to Many)**

Normalization

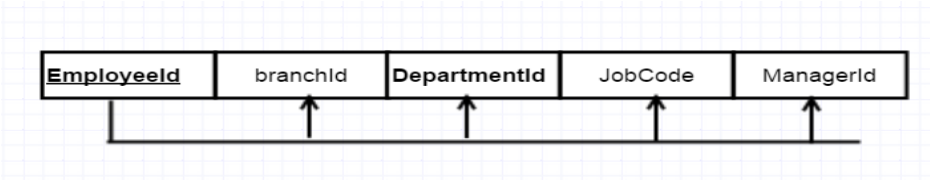


1) **Job**



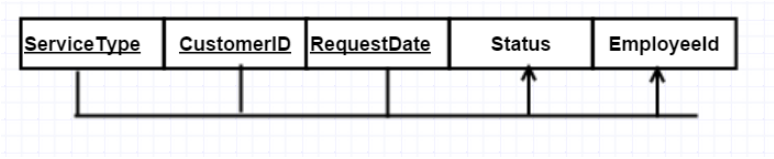
The `JobCode` functionally determines `Designation` and `Responsibility`.
The 'job' table is in 1NF as there are no multivalued attributes
The 'job' table is in 2NF as there are no partial dependencies
The 'job' table is in 3NF as there are no transitive dependencies

2) **Employee**

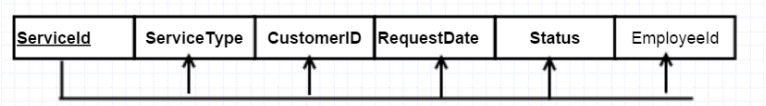


The `EmployeeId` functionally determines `BranchId` ,`DepartmentId`, `JobCode' and `ManagerId`.
The `Employee` table is in 1NF as there are no multivalued attributes
The `Employee` table is in 2NF as there are no partial dependencies
The `Employee` table is in 3NF as there are no transitive dependencies

3) **Service**

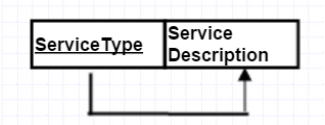


The combination of `ServiceType`, `CustomerID` and `RequestDate` uniquely identifies each record
The Service table is in 3NF



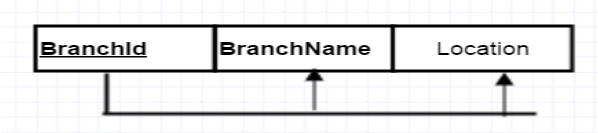
The `ServiceId` is the surrogate key added to uniquely identify each record and a unique constraint is added on (`ServiceType`, `CustomerID` and `RequestDate`).

4) ServiceType



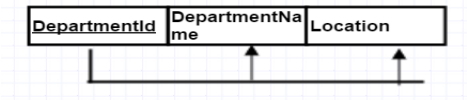
`ServiceType` functionally determines the `ServiceDescription`
The 'ServiceType' table is in 1NF as there are no multivalued attributes
The 'ServiceType' table is in 2NF as there are no partial dependencies
The 'ServiceType' table is in 3NF as there are no transitive dependencies

5) Branch



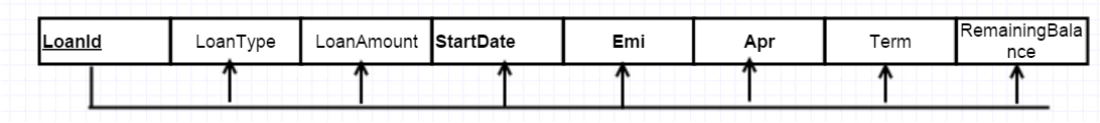
`BranchId` functionally determines the `BranchName` and `Location`
The 'Branch' table is in 1NF as there are no multivalued attributes
The 'Branch' table is in 2NF as there are no partial dependencies
The "Branch' table is in 3NF as there are no transitive dependencies

6) Department



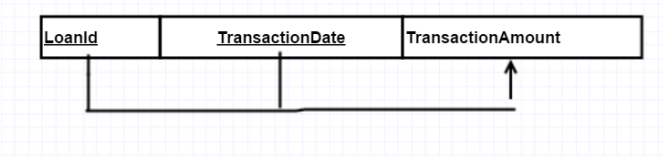
`DepartmentId` functionally determines the `DepartmentName` and `Location`
The 'Department` table is in 1NF as there are no multivalued attributes
The 'Department` table is in 2NF as there are no partial dependencies
The 'Department` table is in 3NF as there are no transitive dependencies

7) Loan

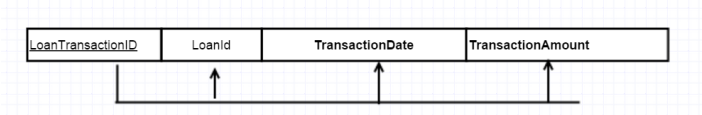


`LoanId` functionally determines the `LoanType` ,`LoanAmount`, `StartDate`, `Emi`, `Apr`, `Term` and `RemainingBalance`
The 'Loan` table is in 1NF as there are no multivalued attributes
The 'Loan` table is in 2NF as there are no partial dependencies
The 'Loan ` table is in 3NF as there are no transitive dependencies

8) Loan_transaction

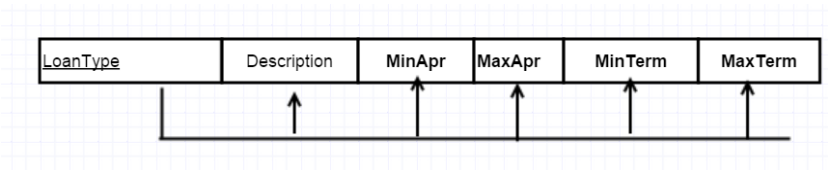


The combination of `LoanId` and `TransactionDate` uniquely identifies each record
The Loan_transaction table is in 3NF



The `LoanTransactionId` is the surrogate key added to uniquely identify each record and a unique constraint is added on (`LoanId` ,`TransactionDate`)

9) LoanType



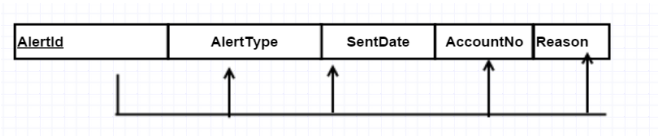
`LoanType` functionally determines the `Description` ,`MinApr`, `MaxApr`, `MinTerm`, `MaxTerm`

The 'LoanType ` table is in 1NF as there are no multivalued attributes

The 'LoanType ` table is in 2NF as there are no partial dependencies

The 'LoanType ` table is in 3NF as there are no transitive dependencies

10) Alert



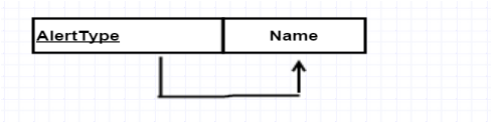
`Alert` functionally determines the `AlertType` ,`SentDate`, `AccountNo`, `Reason`

The 'Alert ` table is in 1NF as there are no multivalued attributes

The 'Alert ` table is in 2NF as there are no partial dependencies

The 'Alert ` table is in 3NF as there are no transitive dependencies

11) Alert_type



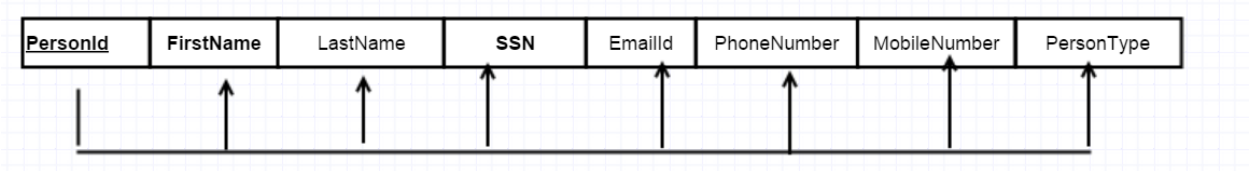
`AlertType` functionally determines the `Name`

The 'AlertType ` table is in 1NF as there are no multivalued attributes

The 'AlertType ` table is in 2NF as there are no partial dependencies

The 'AlertType ` table is in 3NF as there are no transitive dependencies

12) Person



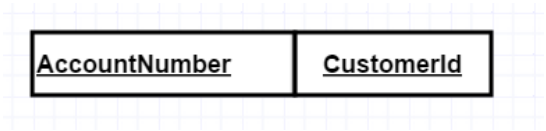
`PersonId` is the primary key, it functionally determines other columns

The ‘Person ` table is in 1NF as there are no multivalued attributes

The ‘Person ` table is in 2NF as there are no partial dependencies

The ‘Person ` table is in 3NF as there are no transitive dependencies

13) Account_customer



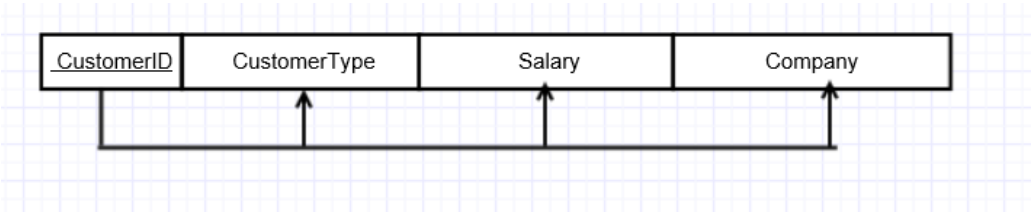
The composite key AccountNumber and CustomerId uniquely identifies each record in the table

The ‘Account_customer` table is in 1NF as there are no multivalued attributes

The ‘Account_customer` table is in 2NF as there are no partial dependencies

The ‘Account_customer ` table is in 3NF as there are no transitive dependencies

14) Customer



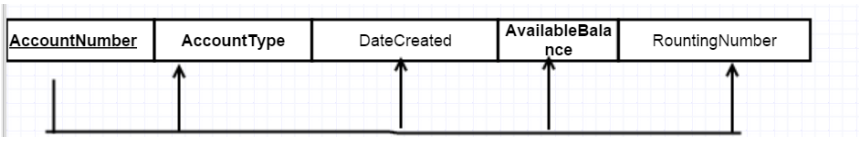
The `CustomerId` uniquely identifies each record in the table

The ‘customer` table is in 1NF as there are no multivalued attributes

The ‘customer` table is in 2NF as there are no partial dependencies

The `customer ` table is in 3NF as there are no transitive dependencies

15) Account



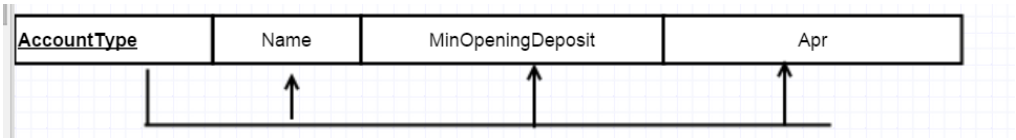
The `AccountNumber` uniquely identifies each record in the table

The ‘Account` table is in 1NF as there are no multivalued attributes

The ‘Account ` table is in 2NF as there are no partial dependencies

The ` Account ` table is in 3NF as there are no transitive dependencies

16) Account_Type

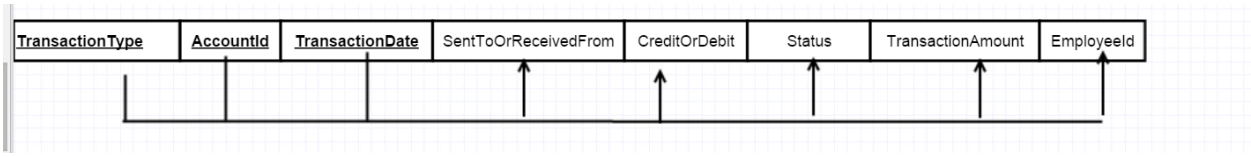


The 'Account_type` table is in 1NF as there are no multivalued attributes

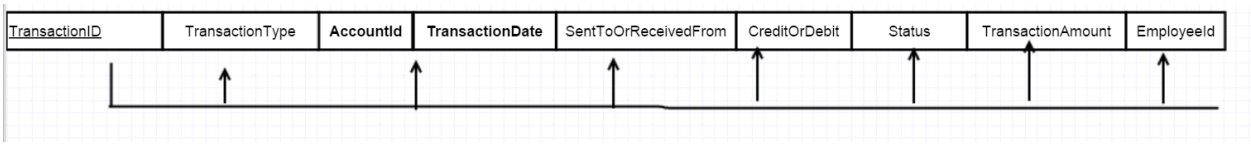
The 'Account_type ` table is in 2NF as there are no partial dependencies

The ` Account_type ` table is in 3NF as there are no transitive dependencies

17) Transaction

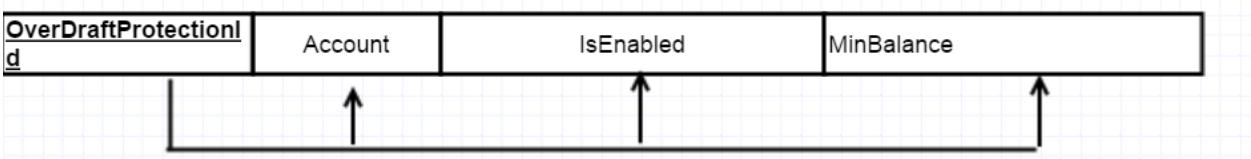


The combination of `LoanId` and `TransactionDate` uniquely identifies each record
The Loan_transaction table is in 3NF



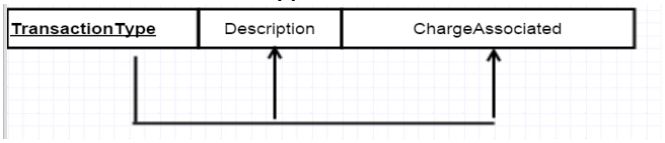
The `LoanTransactionId` is the surrogate key added to uniquely identify each record and a unique constraint is added on the combination of `LoanId` and `TransactionDate`

18) Over_Draft_protection



The 'Over_Draft_protection` table is in 1NF as there are no multivalued attributes
The 'Over_Draft_protection` table is in 2NF as there are no partial dependencies
The ` Over_Draft_protection` table is in 3NF as there are no transitive dependencies

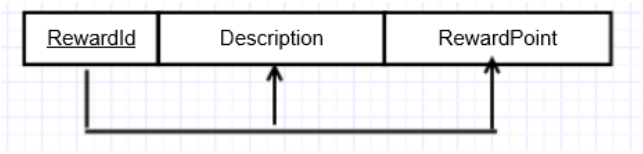
19) Transaction_type



TransactionType functionally determines `Description` and `ChargeAssociated`.

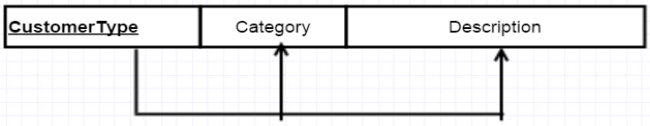
The 'TransactionType ` table is in 1NF as there are no multivalued attributes
The 'TransactionType ` table is in 2NF as there are no partial dependencies
The ` TransactionType ` table is in 3NF as there are no transitive dependencies

20) Reward



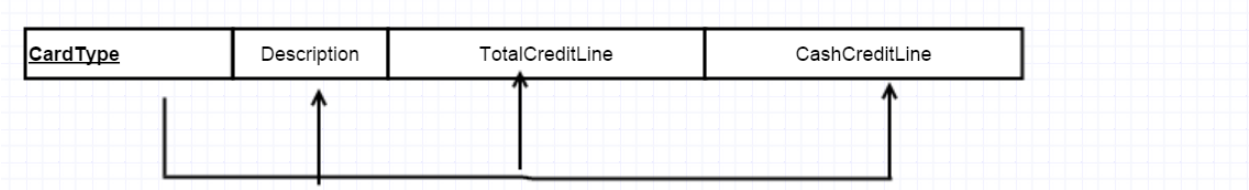
The ‘Reward ` table is in 1NF as there are no multivalued attributes
The ‘Reward ` table is in 2NF as there are no partial dependencies
The ` Reward ` table is in 3NF as there are no transitive dependencies

21) CustomerType



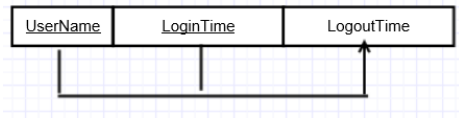
`CustomerType` functionally determines `Category` and `Description`
The ‘CustomerType ` table is in 1NF as there are no multivalued attributes
The ‘CustomerType ` table is in 2NF as there are no partial dependencies
The ` CustomerType ` table is in 3NF as there are no transitive dependencies

22) CardType



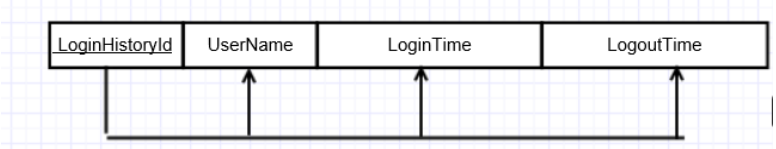
`CardType` functionally determines `Description`, `TotalCreditLine` and `CashCreditLine`
The ‘CardType ` table is in 1NF as there are no multivalued attributes
The ‘CardType ` table is in 2NF as there are no partial dependencies
The ` CardType ` table is in 3NF as there are no transitive dependencies

23) LoginHistory

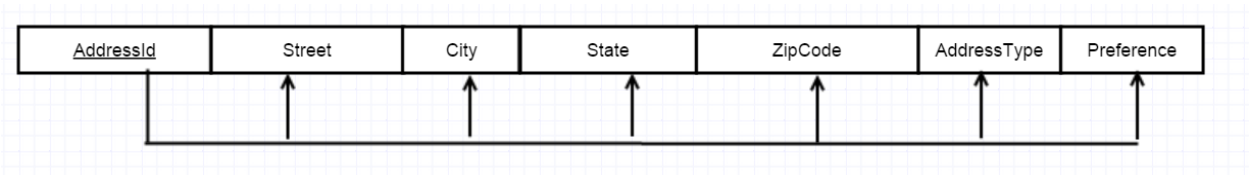


The primary key of the table is combination of UserName and LoginTime

A surrogate key LoginHistoryId is added to identify the records uniquely. A unique constraint has to be placed on (UserName, LoginTime)



24) Address

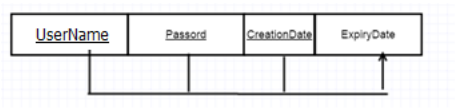


The 'Address ` table is in 1NF as there are no multivalued attributes

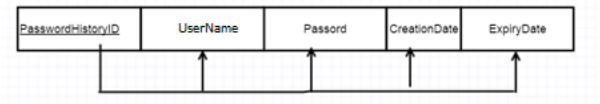
The 'Address ` table is in 2NF as there are no partial dependencies

The ` Address ` table is in 3NF as there are no transitive dependencies

25) PasswordHistory

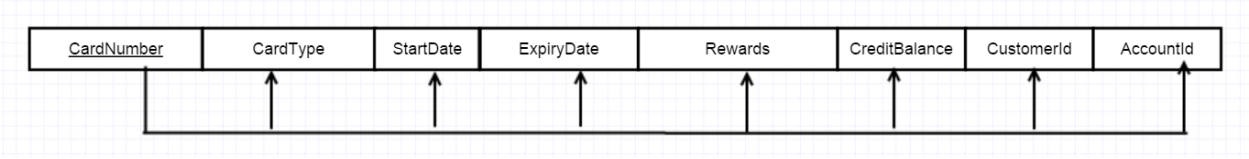


The combination of UserName ,password and CreationDate uniquely identifies each record.



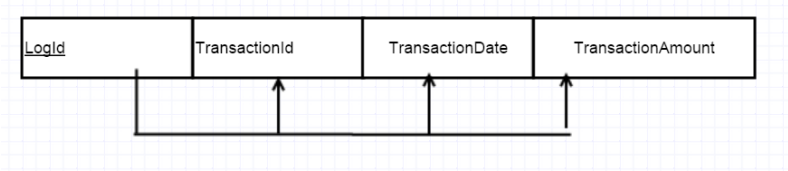
A surrogate key PasswordHistoryId is added.A unique constrain has to placed on (UserName, password, creationdate)

26) Card



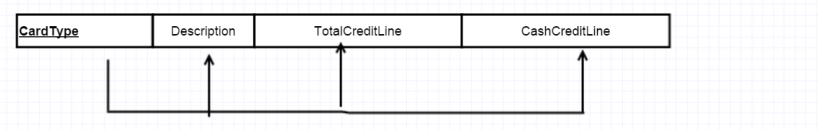
The card table is in 3NF

27) Log



The log table is 3NF

28) CardType

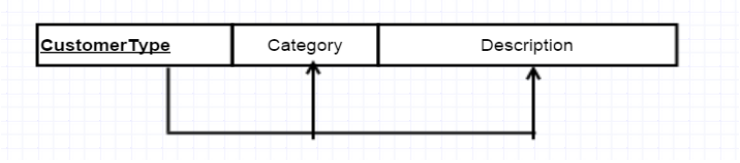


`CardType` functionally determines `Description`, `TotalCreditLine` and `CashCreditLine`

The 'CardType ` table is in 1NF as there are no multivalued attributes

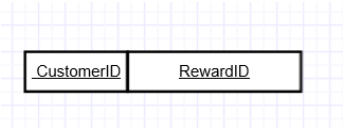
The ‘CardType ` table is in 2NF as there are no partial dependencies
The ` CardType ` table is in 3NF as there are no transitive dependencies

29) CustomerType



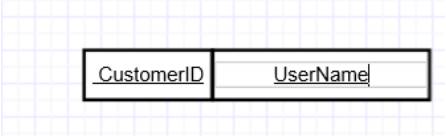
`CustomerType` functionally determines `Category` and `Description`
The ‘CustomerType ` table is in 1NF as there are no multivalued attributes
The ‘CustomerType ` table is in 2NF as there are no partial dependencies
The ` CustomerType ` table is in 3NF as there are no transitive dependencies

30) Customer_rewards



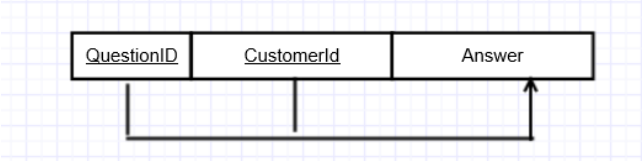
The Customer_rewards table is in 1NF as there are no multivalued attributes
The Customer_rewards table is in 2NF as there are no partial dependencies
The Customer_rewards table is in 3NF as there are no transitive dependencies

31) Customer_Credentials



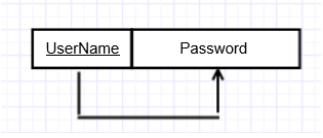
The Customer_Credentials table is in 1NF as there are no multivalued attributes
The Customer_Credentials table is in 2NF as there are no partial dependencies
The Customer_Credentials table is in 3NF as there are no transitive dependencies

32) Customer_question_answer



The Customer_Credentials table is in 1NF as there are no multivalued attributes
The Customer_Credentials table is in 2NF as there are no partial dependencies
The Customer_Credentials table is in 3NF as there are no transitive dependencies

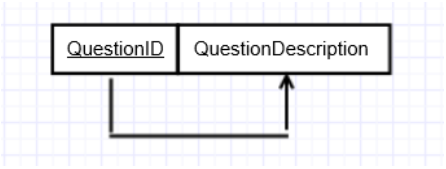
33) Credentials



The Credentials table is in 1NF as there are no multivalued attributes
The Credentials table is in 2NF as there are no partial dependencies

The Credentials table is in 3NF as there are no transitive dependencies

34) SecretQuestions



The SecretQuestions table is in 1NF as there are no multivalued attributes
The SecretQuestions table is in 2NF as there are no partial dependencies
The SecretQuestions table is in 3NF as there are no transitive dependencies

Views

1) Customer Account Information View

The `CustomerAccountInformation` view is used to view the details of the accounts information of the customer.

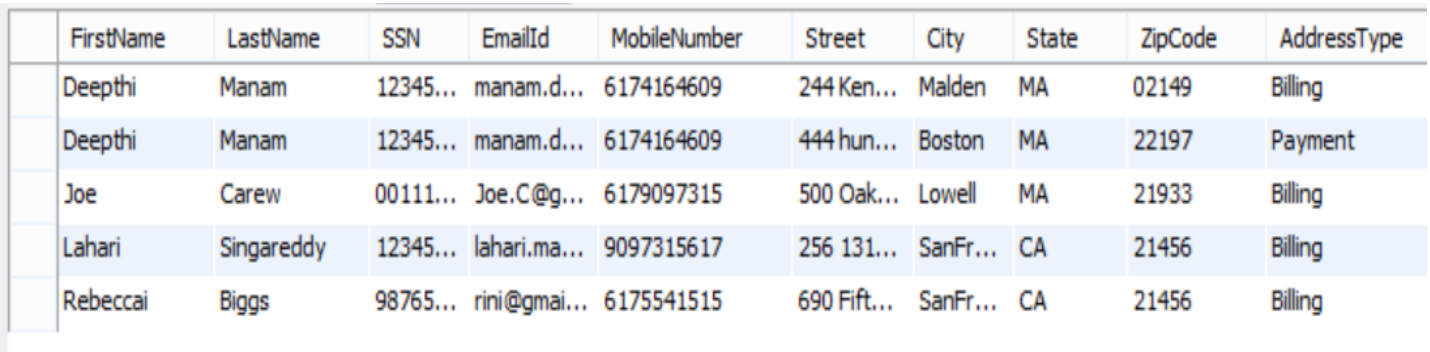
	CustomerId	AccountNumber	AccountType	AvailableBalance
	1	135790642	Saving	700
	2	345310012	Saving	100
	3	657451234	Saving	2000
	4	657451234	Saving	2000
	1	246815791	Checking	900

```
CREATE
    ALGORITHM=UNDEFINED
    DEFINER=`root`@`localhost`
    SQL SECURITY DEFINER
VIEW `bankingsystem`.`customer_account_information` AS select
    `c`.`CustomerId` AS `CustomerId`,
    `a`.`AccountNumber` AS `AccountNumber`,
```

```
`at`.`Name` AS `AccountType`,
`a`.`AvailableBalance` AS `AvailableBalance`
from (((`bankingsystem`.`customer` `c`
join `bankingsystem`.`account_customer` `ac`
on((`ac`.`CustomerId` = `c`.`CustomerId`)))
join `bankingsystem`.`account` `a`
 on((`a`.`AccountNumber` = `ac`.`AccountNumber`)))
join `bankingsystem`.`account_type` `at`
on((`a`.`AccountType` = `at`.`AccountType`)));
```

2)Customer Basic Information View

The Customer_Basic_Info view is used to view the details of the customer basic information.



FirstName	LastName	SSN	EmailId	MobileNumber	Street	City	State	ZipCode	AddressType
Deepthi	Manam	12345...	manam.d...	6174164609	244 Ken...	Malden	MA	02149	Billing
Deepthi	Manam	12345...	manam.d...	6174164609	444 hun...	Boston	MA	22197	Payment
Joe	Carew	00111...	Joe.C@g...	6179097315	500 Oak...	Lowell	MA	21933	Billing
Lahari	Singareddy	12345...	lahari.ma...	9097315617	256 131...	SanFr...	CA	21456	Billing
Rebeccai	Biggs	98765...	rini@gmai...	6175541515	690 Fift...	SanFr...	CA	21456	Billing

```
CREATE
    ALGORITHM=UNDEFINED
    DEFINER=`root`@`localhost`
    SQL SECURITY DEFINER
VIEW `bankingsystem`.`customer_basic_info` AS
    select
    `p`.`FirstName` AS `FirstName`,
    `p`.`LastName` AS `LastName`,
    `p`.`SSN` AS `SSN`,
    `p`.`EmailId` AS `EmailId`,
    `p`.`MobileNumber` AS `MobileNumber`,
```

```
`a`.`Street` AS `Street`,
`a`.`City` AS `City`,
`a`.`State` AS `State`,
`a`.`ZipCode` AS `ZipCode`,
`a`.`AddressType` AS `AddressType` from
((`bankingsystem`.`person` `p`
 join `bankingsystem`.`person_address` `pa`
on((`p`.`PersonID` = `pa`.`PersonId`)))
 join `bankingsystem`.`address` `a`
on((`a`.`AddressID` = `pa`.`AddressId`)));
```

3) Customer Loan Information View

The customer loan information view is used to view the details of the loan taken by the customer.

CustomerId	LoanId	Description	LoanAmount	StartDate	emi	Apr	RemainingBal
1	44545	Education Loan	35000	2012-06-01	500	4	31000
2	45331	HomeEquity L...	65000	2011-03-04	600	6	64000
3	56741	Personal Loan	10000	2012-01-01	500	5	9000
4	56745	Vehicle Loan	50000	2010-07-01	700	7	40000

```
CREATE
    ALGORITHM=UNDEFINED
    DEFINER=`root`@`localhost`
    SQL SECURITY DEFINER
VIEW `customer_loan_information` AS
     select `c`.`CustomerId` AS `CustomerId`,
    `l`.`LoanId` AS `LoanId`,
    `lt`.`Description` AS `Description`,
    `l`.`LoanAmount` AS `LoanAmount`,
    `l`.`StartDate` AS `StartDate`,
    `l`.`EMI` AS `emi`,
```

```
`l`.`Apr` AS `Apr`,
`l`.`RemainingBal` AS `RemainingBal`
from (((`customer` `c`
join `loan_customer` `lc`
 on((`lc`.`customerId` = `c`.`CustomerId`)))
join `loan` `l`
on((`l`.`LoanId` = `lc`.`loanId`)))
join `loantype` `lt`
on((`l`.`LoanType` = `lt`.`LoanType`)));
```

5) Customer Service Request View

The `customer_service_request` view is used to view the details of the service requests raised by the customer.

	CustomerId	ServiceId	serviceDescription	RequestedDate	Status
	1	1	Dispute Transaction	2011-01-02	Completed
	2	2	Request for cheques	2010-07-06	Completed
	4	3	Dispute Transaction	2009-11-02	Completed
	2	4	Cancel payment of c...	2011-01-15	Completed

```
CREATE
        ALGORITHM=UNDEFINED
        DEFINER=`root`@`localhost`
        SQL SECURITY DEFINER
VIEW `customer_service_request` AS
         select `c`.`CustomerId` AS `CustomerId`,
        `s`.`ServiceId` AS `ServiceId`,
        `st`.`ServiceDescription` AS `serviceDescription`,
        `s`.`RequestedDate` AS `RequestedDate`,
        `s`.`Status` AS `Status`
```

```
from ((`customer` `c`
join `service` `s`
on((`s`.`CustomerId` = `c`.`CustomerId`)))
join `servicetype` `st`
on((`st`.`ServiceType` = `s`.`ServiceType`)));
```

6) Customer Card Information View

The `customer_card_info` view is used to view the details of the cards owned by the customer

customerId	CardNumber	Description	startDate	ExpiryDate	CreditBalance	AccountNumber
1	123090979125	Visa Debit Card	2009-10-11	2015-10-01	0	135790642
3	134556780909	Master Debit ...	2009-10-11	2015-10-01	0	657451234
4	567456675435	Visa Credit Card	2009-10-11	2015-10-01	500	567439871
2	666456789426	Titanium Cre...	2009-10-11	2015-10-01	0	345310012

```
CREATE
    ALGORITHM=UNDEFINED
    DEFINER=`root`@`localhost`
    SQL SECURITY DEFINER
VIEW `customer_card_info` AS
    select `c`.`CustomerId` AS `customerId`,
    `card`.`CardNumber` AS `CardNumber`,
    `ct`.`Description` AS `Description`,
    `card`.`StartDate` AS `startDate`,
    `card`.`ExpiryDate` AS `ExpiryDate`,
    `card`.`CreditBalance` AS `CreditBalance`,
    `card`.`AccountNumber` AS `AccountNumber`
     from ((`customer` `c` join `card`
    on((`c`.`CustomerId` = `card`.`CustomerId`)))
     join `card_type` `ct`
    on((`ct`.`CardType` = `card`.`CardType`)));
```

TRANSACTIONS

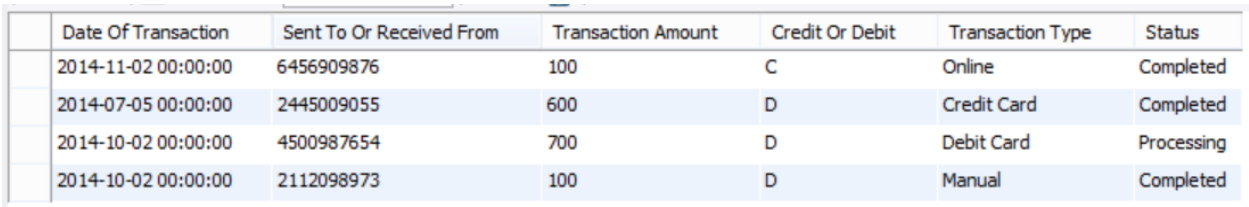
`Transfer_amount` procedure is used to transfer amount from one account to another. We explicitly initiate the transaction using START TRANSACTION command, before any manipulations on the accounts.

```
DELIMITER $$

CREATE

    DEFINER=`root`@`localhost`

PROCEDURE `Transfer_Amount`(accountNo1 INT, accountNo2 INT,
transferAmount DOUBLE)

    BEGIN

    START TRANSACTION;

        update account set AvailableBalance = AvailableBalance -
        transferAmount where AccountNumber = accountNo1;

        update account set AvailableBalance = AvailableBalance +
        transferAmount where AccountNumber = accountNo2;

    COMMIT;

END
```

Stored Procedures

AccountSummary Procedure:

`AccountSummary` procedure gives the summary of the account transactions for particular account.



Date Of Transaction	Sent To Or Received From	Transaction Amount	Credit Or Debit	Transaction Type	Status
2014-11-02 00:00:00	6456909876	100	C	Online	Completed
2014-07-05 00:00:00	2445009055	600	D	Credit Card	Completed
2014-10-02 00:00:00	4500987654	700	D	Debit Card	Processing
2014-10-02 00:00:00	2112098973	100	D	Manual	Completed

```
DELIMITER $$

CREATE

    DEFINER=`root`@`localhost`

     PROCEDURE `AccountSummary`(accountNumber INT)
```

```
Begin
        select  acctTrs.transactionDate as `Date Of Transaction`,
         acctTrs.SentToOrReceivedFrom `Sent To Or Received From`,
        acctTrs.TransactionAmount `Transaction Amount`,
        acctTrs.CreditOrDebit  `Credit Or Debit`,
        tt.Description `Transaction Type`,
        acctTrs.Status `Status`
         from
        account_transaction acctTrs inner join transactiontype tt
        on tt.TransactionType = acctTrs.TransactionType
        where AccountId = accountNumber;
        END$$
        DELIMITER ;
```

USER DEFINED FUNCTIONS

CustomerLevel Function:

The CustomerLevel function returns the level of the customer based on the account balance of the customer.

```
        DELIMITER //
        CREATE FUNCTION CustomerLevel(accountBal double)
        RETURNS VARCHAR(10)
         DETERMINISTIC
              BEGIN
                    DECLARE lvl varchar(10);
                   IF accountBal > 50000 THEN
                   SET lvl = 'PLATINUM';
                    ELSEIF (accountBal <= 50000 AND accountBal >=
              10000) THEN  SET lvl = 'GOLD';
                    ELSEIF accountBal < 10000 THEN
```

```
        SET lvl = 'SILVER';
        END IF;
      RETURN (lvl);
END
//
```

TRIGGERS

`log_transaction' triggers is used to log, each and every account transaction made by the customers.

```
DELIMITER //
CREATE TRIGGER log_transaction
AFTER
INSERT ON account_transaction
FOR EACH ROW
BEGIN
INSERT into log(TransactionId, TransactionDate, TransactionAmount)
VALUES(NEW.TransactionId, sysdate(), NEW.TransactionAmount);
END//
```