AIT-524-002
Database Management Systems
Assignment-6

SUJAN CHAVA
George Mason University

- **Description of the company:**
  The database that I am planning to develop is a banking system to sanction loans for its customers. These banks want to store the data about all the branches they have, the accounts and loans sanctioned for the customers.
- **Why company needs database?**
  This banking system needs a database because, there will be huge number of customers for a bank at each specific branches. Each bank needs to keep track of these customers at each branch, so that it will be easy for them to process customer's request. Also, the database can help them identify pending requests that they need to address and the requests they have already addressed. They can also, query out the total amount of loan taken by each customer.
- **Business Rules:**
  1) Each bank may have one or many branches. Each branch must contain at-least one the bank.
  2) Each branch may issue any number of loans. Each loan must be issued by only one branch.
  3) Each branch may have many accounts. Each account must belong to only one branch.
  4) Each person may have any number of accounts. Each account must belong to only one specific person.
  5) Each person may borrow any number of loans. Each loan is sanctioned to only one person.
  6) Each person is classified as bank employee and non-employee to distinguish the bank employees applying for loan from normal employess.
- **Description of entities and attributes:**
  1) **Bank** → The purpose of this entity is to store details about different banks that are willing to sanction loans.
  Attributes:
  bank_id, bank_name, bank_email
  2) **Bank_brnh (bridging)** → This is a bridge table between bank and branches to resolve the many to many relationships which contains the primary keys of bank and branches as foreign key in it.
  Attributes:
  bank_id, branch_id, total_branches
  3) **Branches** →The purpose of this entity is to store details of the branches that a bank have, its name and addresses.
  Attributes:
  branch_id, branch_city, branch_zip, branch_strtname, branch_state
  4) **Loans** → Its purpose is to store details about amount sanctioned to each person.
  Attributes:
  loan_id, loan_type, loan_amount, loan_status, loan_assets
  5) **Account** → Its purpose is to store the account details of the customers of the bank.
  Attributes:
  accnt_num, accnt_type, accnt_balance
  6) **Person** → Its purpose is to store the details of the persons who have borrowed loan from banks and who have accounts in the banks.
  Attributes:
  person_id, person_name, person_phn, person_hno, person_strtname, person_city, person_state, employee_type.
  (Sub-Class attributes of Person entity)
  7) **Bank_employee** → This is a subclass attribute which inherits all the property of person entity and its purpose is to store details of bank employees applying for loans.
  Attributes:
  person_id, salary, emp_position
  8) **Non-Employee** → This subclass entity is used to store the credit rating of the customers applying for loan if they have any.

Attributes:
person_id, credit_score.

- **Description of relationship between the entities:**

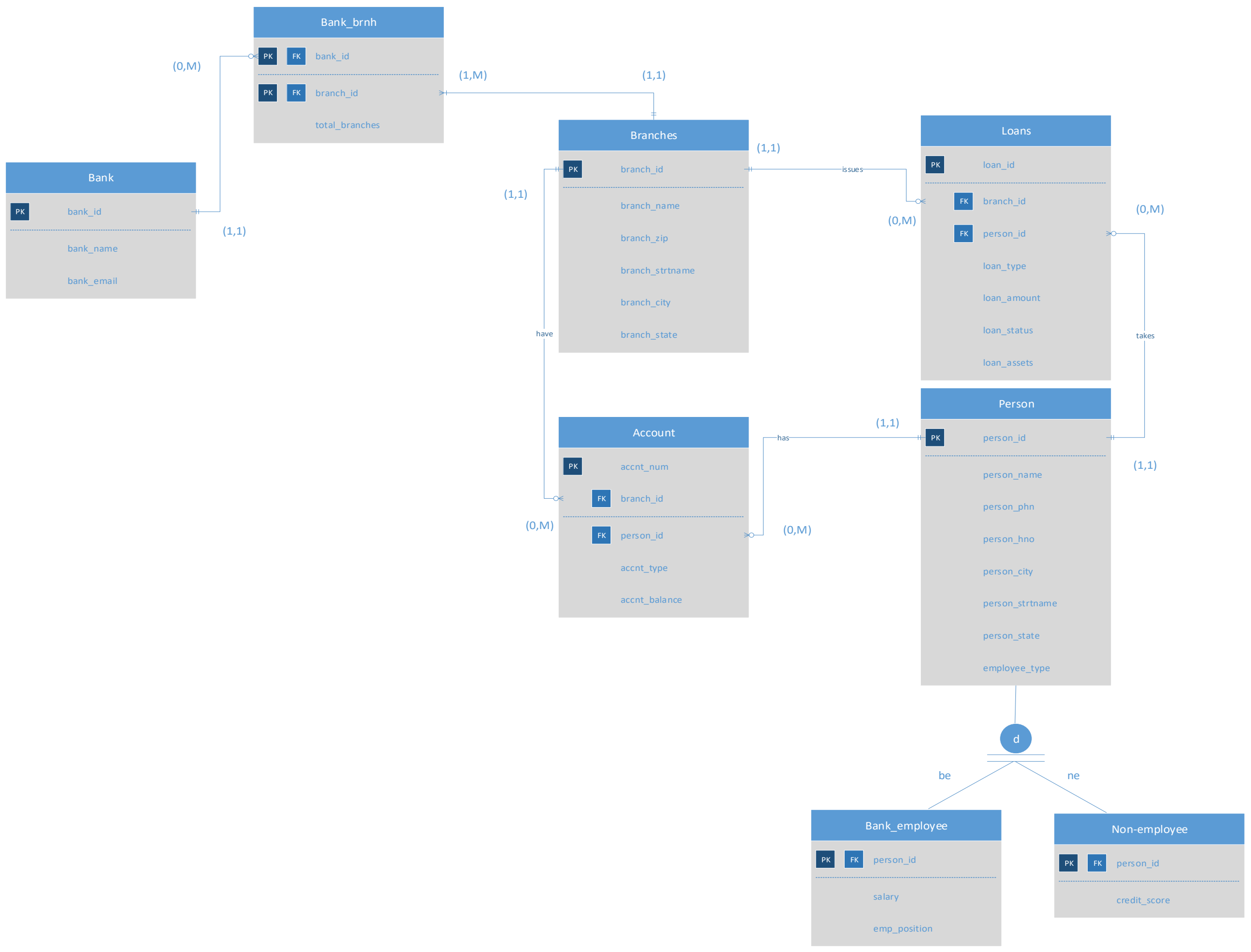
From the given business rules, we can say that

1) **Bank and Bank_brnh:**
There is a 1: M relationship between bank and bank_brnh because a bank can contain any number of branches, i.e (M) on bank_brnh side, but each branch may have association with one of the banks, i.e (1) on bank side. It is a strong relationship as bank_brnh is dependent on bank. The relationship is optional on "many" (bank_brnh) side because the business rules says that each bank "may" have any number of branches. It is mandatory on "one" (bank) side because business rules says that each branch "must" contain at-least one of the bank. Therefore according to the business rules the cardinality on many side is (0, M) because it is optional and (1, 1) for the one side because it is mandatory.
2) **Bank_brnh and Branches:**
There is a 1: M relationship between branches and bank_brnh because each branch may have one or more banks in it, i.e (M) on bank_brnh side, but each bank may have association with one of the branches, i.e (1) on branches side. It is a strong relationship as bank_brnh is dependent on branches. The relationship is mandatory on "many" (bank_brnh) side because the business rules says that each branch "must" have at-least one of the banks. It is mandatory on "one" (branches) side because if bank_brnk exist then it "must" have a branch_id attribute. Therefore according to the business rules the cardinality on many side and one side is (1, M) and (1, 1) because it is mandatory.
3) **Branches and Loans:**
There is a 1: M relationship between branches and loans because a branch can issue any number of loans, i.e (M) on loans side, but a loan is issued by only one branch, i.e (1) on branch side. It is a strong relationship as loans are dependent on branch. The relationship is optional on "many" (loans) side because the business rules says that each branch "may" issue any number of loans. It is mandatory on "one" (branches) side because business rules says that each loan "must" be issued by only one branch. Therefore according to the business rules the cardinality on many side is (0, M) because it is optional and (1, 1) for the one side because it is mandatory.
4) **Branches and Accounts:**
There is a 1: M relationship between branches and accounts because a branch can have any number of accounts, i.e (M) on accounts side, but an account belongs to only one branch, i.e (1) on branch side. It is strong relationship as accounts cannot exist without branches (bank) i.e accounts is dependent on branch. The relationship is optional on "many" (accounts) side because the business rules says that each branch "may" have many accounts. It is mandatory on "one" (branches) side because business rules says that each account "must" belong to only one branch. Therefore according to the business rules the cardinality on many side is (0, M) because it is optional and (1, 1) for one side because it is mandatory.
5) **Person and Accounts:**
There is a 1: M relationship between person and accounts because given a person can have any number of accounts, i.e (M) on accounts side, and an account belongs to only one person, i.e (1) on person side. It is strong relationship as accounts cannot exist without person i.e accounts is dependent on person. The relationship is optional on "accounts" side because the business rules says that each person "may" have an account. It is mandatory on person side because business rules says that each account "must" belong to one specific person. Therefore according to the business rules the cardinality on accounts side is (0, M) because it is optional and (1, 1) on person side because it is mandatory.
6) **Person and Loans:**
There is a 1: M relationship between person and loan because a person can borrow any number of loans, i.e (M) on loans side, but loan is issued to only one person, i.e (1) on person side. It is strong relationship as loan cannot exist without person i.e loan is dependent on person. The relationship is optional on "many" (loans) side because the business rules says that each person "may" take any number of loans. It is mandatory on "one" (person) side because business rules says that each loan "must" be sanctioned to only one person.

Therefore according to the business rules the cardinality on many side is (0, M) because it is optional and (1, 1) for one side because it is mandatory.

The ERD using above business rules is designed using Microsoft Visio and the final ER-Diagram for our database is shown in the figure below.

ER- Diagram:



Relational Algebra:

1) Retrieve the loan number and id of the person having loan amount greater than 1000 and having his loan approved.

$$\Pi_{\text{loan\_id, person\_id}} (\sigma_{\text{loan\_amount} > 1000 \wedge \text{loan\_status} = \text{'Approved'}} (\text{Loans}))$$

2) Retrieve name and phone number of person who is bank employee and who lives in Florida.

$$\Pi_{\text{person\_name, person\_phn}} (\sigma_{\text{employee\_type} = \text{'be'} \wedge \text{person\_state} = \text{'Florida'}} (\text{Person}))$$

3) Retrieve the account number and id of the persons having balance greater than 5000 in savings account.

$\Pi_{\text{accnt\_num, person\_id}} (\sigma_{\text{accnt\_balance >5000 ^ accnt\_type = ‘Savings’}} (\text{Account}))$

4) **Retrieve name, phone number and salaries of all the managers of the bank, who took loan and lives in Virginia.**

$\Pi_{\text{person\_name, person\_phn, salary}} (\sigma_{\text{emp\_position = ‘Manager’}} (\text{Bank\_employee}) \infty_{\text{Bank\_employee.person\_id = Person.person\_id}} (\sigma_{\text{person\_state = ‘Virginia’}} (\text{Person}))$

5) **Retrieve loan number, type of loan and amount of loan taken by person named ‘John Wick’ who lives in California.**

$\Pi_{\text{loan\_id, loan\_type, loan\_amount}} ((\text{Loans}) \infty_{\text{Loans.person\_id = Person.person\_id}} (\sigma_{\text{person\_name = ‘John Wick’ ^ person\_state = ‘Claifornia’}} (\text{Person}))$

6) **Retrieve all the names of branches, loan number and id of person having loan amount greater than 10000 and whose loan is yet to be approved (Pending).**

$\Pi_{\text{branch\_name, loan\_id, person\_id}} ((\text{Branches}) \infty_{\text{Branches.branch\_id = Loans.branch\_id}} (\sigma_{\text{loan\_amount > 10000 ^ loan\_status = ‘Pending’}} (\text{Loans}))$

7) **Retrieve id, name and email of the banks having more than 2 branches in the city Fairfax.**

$\Pi_{\text{bank\_id, bank\_name, bank\_email}} (\text{Bank} \infty_{\text{Bank.bank\_id = Bank\_brnh.bank\_id}} (\sigma_{\text{total\_branches > 2}} (\text{Bank\_brnh}) \infty_{\text{Bank\_brnh.branch\_id = Branches.branch\_id}} (\sigma_{\text{branch\_city = ‘Fairfax’}} (\text{Branches}))))$

8) **Retrieve the account number, id and name of the person having the value of assets for loan greater than 20000 and have applied for car loan.**

$\Pi_{\text{accnt\_num, person\_id, person\_name}} (\text{Account} \infty_{\text{Account.person\_id = Person.person\_id}} (\text{Person} \infty_{\text{Person.person\_id = Loans.person\_id}} (\sigma_{\text{loan\_assets > 20000 ^ loan\_type = ‘car’}} (\text{Loans}))))$

9) **Retrieve the names of person and his id who live in Fairfax and applied for more than two loans and whose loans is yet to be approved (Pending).**

$\Pi_{\text{person\_name, person\_id}} ({}_{\text{person\_id}} F_{\text{COUNT (loan\_id) > 2}} (\sigma_{\text{loan\_status = ‘Pending’}} (\text{Loans}) \infty_{\text{Loans.person\_id = Person.person\_id}} \sigma_{\text{person\_city = ‘Fairfax’}} (\text{Person})))$

10) **Retrieve name and city of the branches having more than 100 loans pending and who have applied for housing loan.**

$\Pi_{\text{branch\_name, branch\_city}} (\text{Branches} \infty_{\text{Branches.branch\_id = Loans.branch\_id}} ({}_{\text{branch\_id}} F_{\text{COUNT (loan\_id) > 100}} (\sigma_{\text{loan\_status = ‘Pending’ ^ loan\_type = ‘housing’}} (\text{Loans}))))$