EPOKA UNIVERSITY

MIDTERM PROJECT

CEN 202 – DATA BASE MANAGEMENT SYSTEMS

Title: *Bank Database*

Evelin Uliu Computer Engineering Department

Course Instructor: Igli Hakrama

5/31/2020

Problem Description

Introduction of the problem

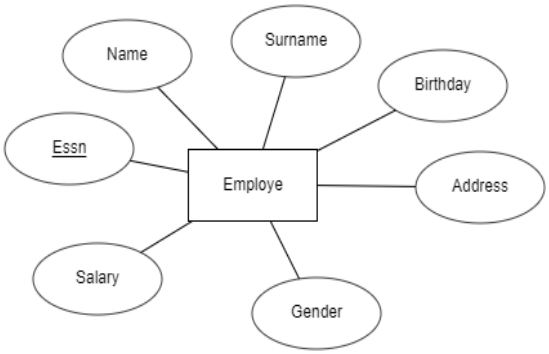
A bank database is a database application that stores information related to the bank activity. It keeps tracks of bank branches, employees, customers, accounts and loans types that the bank offers and accounts and loans a customer has with the bank.

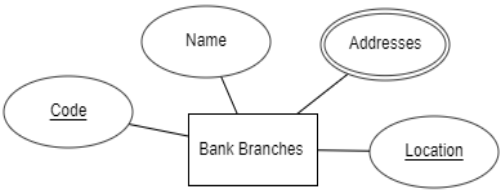
Full description of the requirements taken into consideration

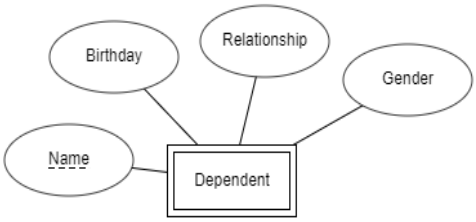
1. For each Bank Branch store data as Name, Location, Code and Addresses.
2. For each Employee store data as Employee SSN, Name, Surname, Birthday, Gender, Address and Salary.
3. For each Dependent store data as Name, Birthday Gender and Relationship.
4. For each Customer store data as Customer SSN, Name, Surname, Birthday, Gender, and Address.
5. For each Account store data as Account number, Name and Type.
6. For each Loan store data as Loan number, Name and Type.
7. A bank has many bank branches where each bank branch has a unique bank code and unique location (specifies the city or country where the bank is located).The bank branch can have also many different addresses within the same location.
8. Each bank branch must have one or more employees who work for the bank branch, where each employee has a unique social security number. Each employee must work only for one branch.
9. Each bank branch must have only one employee who manages the bank branch. An employee can be the manager of only one bank branch.
10. Employees may have a supervisor, who supervises his work and a supervisor can supervisee multiple employees.
11. Also for every employee we may store data for its dependents. An employee may have one or more dependents, and each dependent is related only to one employee.
12. Bank branches can create some types of accounts and loans where each account has a unique account number and each loan has a unique loan number.
13. Each account and loan belongs to only one bank branch.
14. Customers have a unique social security number and they can create many accounts where each account has one or more customers. Same thing goes for loans. Customers can create many loans and each loan has one or more customers.

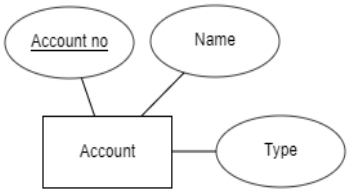
Analysis with ER Diagrams

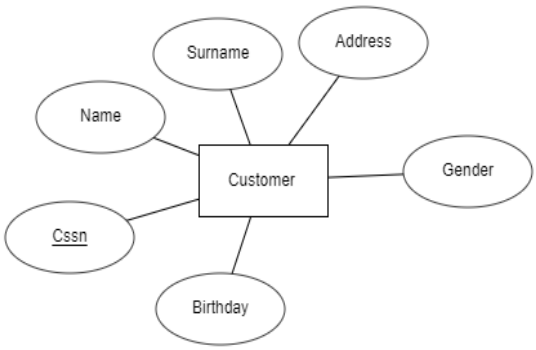
1. *Main Entities, Relationships and Cardinalities.*

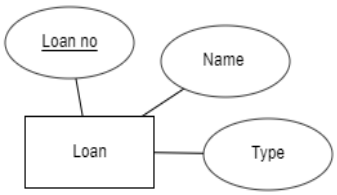








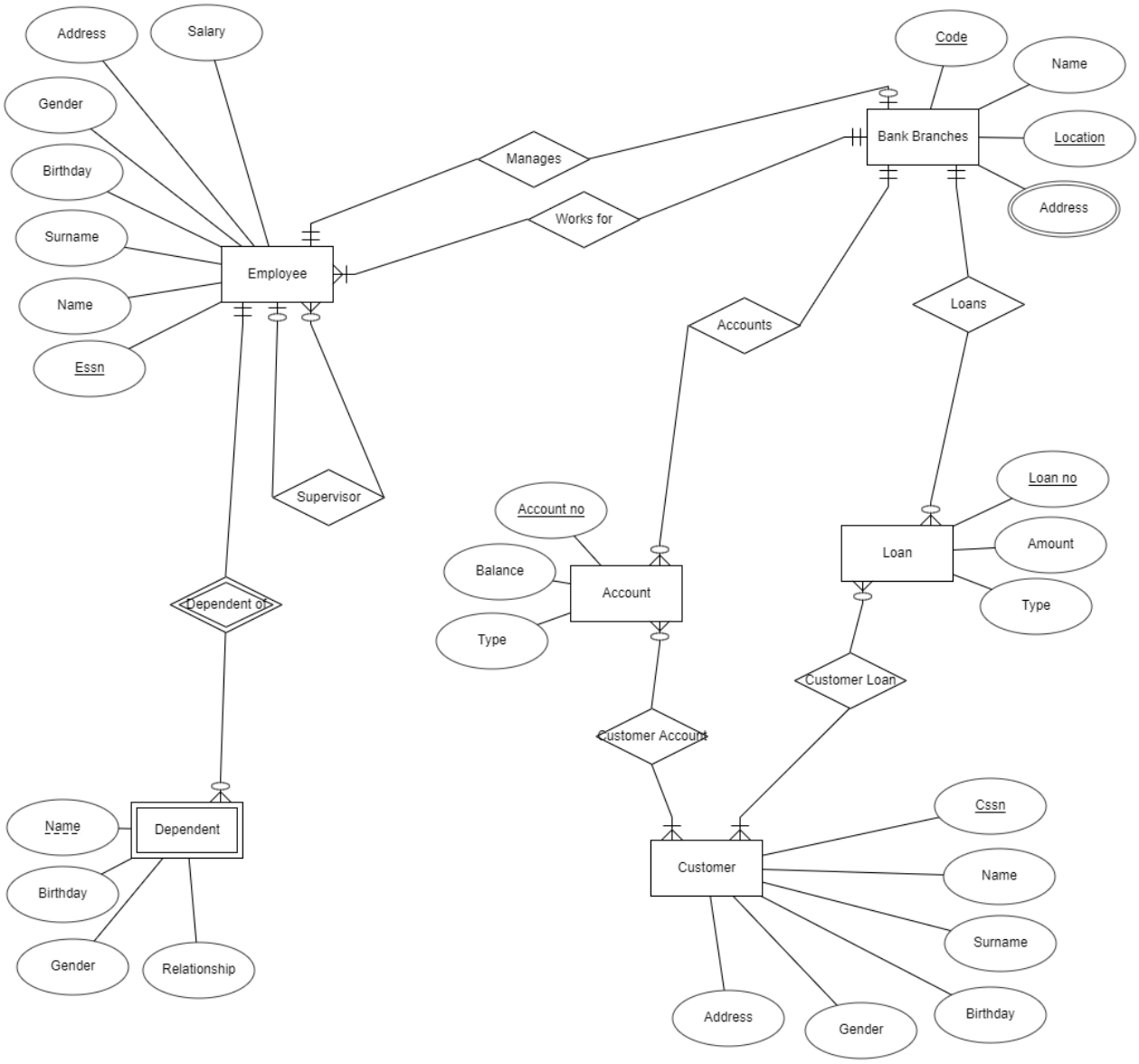




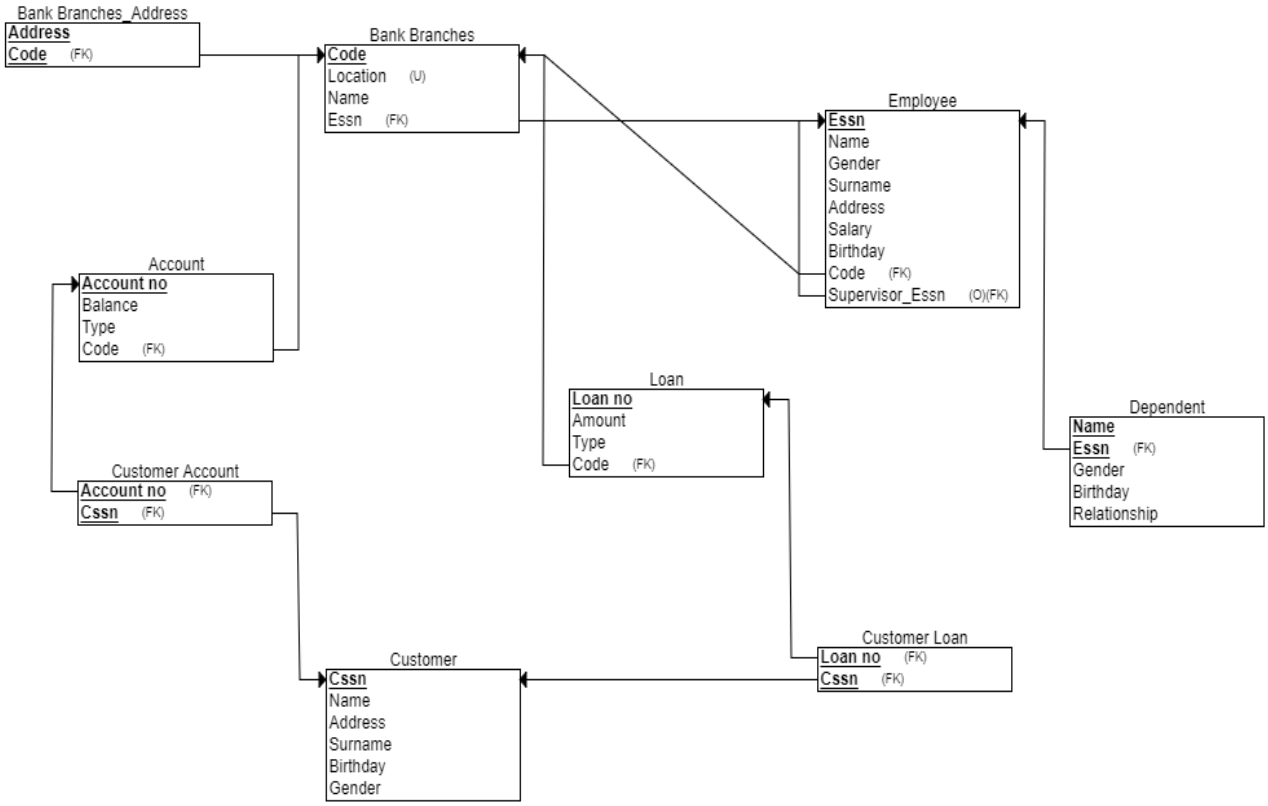
- Manages: Employee-Bank Branches, 1:1 , Employee has partial participation and Bank Branches have total participation
- Works for: Employee-Bank Branches, N:1, total participation from both entities
- Dependent of: Employee-Dependent, 1:N , Employee has partial participation and Dependent have total participation
- Supervisor : Employee(as Supervisor)-Employee(as Supervisee), 1:N, partial participation
- Accounts: Bank Branches - Account, 1:N, Bank Branches have partial participation and Account has total participation
- Loans: Bank Branches-Loan, 1:N, Bank Branches have partial participation and Loan has total participation

- Customer Account: Account-Customer, M:N, Customer has partial participation and Account have total participation
- Customer Loan: Loan-Customer, M:N, Customer has partial participation and Loan have total participation

2. *Full ER Diagram*



Relational Schema Model



Rationale

1. *Constraints, Triggers, Functions and Procedures*

Constraints include key and referential integrity constraints, restrictions on attribute domains and NULLs, and constrains on individual tuples within a relation using the CHECK clause.

Because SQL allows NULLs as attribute values, a constraint NOT NULL may be specified if NULL is not permitted for a particular attribute. This is always implicitly specified for the attributes that are part of the primary key of each relation, but it can be specified for any other attribute whose values are required not to be NULL.

It is also possible to define a default value for an attribute, if an explicit value is not provided for that attribute. If no default value is specified, the default value is NULL for attributes that do not have the NOT NULL constraint.

Another type of constraint can restrict attribute or domain values using CHECK clause following an attribute or domain definition.

The PRIMARY KEY clause specifies one or more attributes that make up the primary key of a relation.

The UNIQUE clause specifies alternate (unique) keys, also known as candidate keys.

Referential integrity is specified via the FOREIGN KEY clause. Referential integrity constrains can be violated when tuples are inserted or deleted, or when a foreign key or a primary key value attribute value is updated. The default action that SQL takes for an integrity violation is to reject the update violation that will cause the violation, which is known as RESTRICT operation. An alternative action can be taken by attaching a referential triggered action clause to any foreign key constraint. The options include SET NULL, CASCADE, and SET DEFAULT. An option must be qualified with either ON DELETE or ON UPDATE.

Some examples when these constrains are used in our database:

```
create table if not exists Bank_Branches
(
 bankName varchar(20) not null,
 bankCode int not null,
 managerEssn int not null default 333445555,
 bankLocation varchar(20) not null,
 primary key (bankCode),
 unique (bankLocation)
);
```

```
create table if not exists Employee
(
 empFname varchar(20) not null,
 empLname varchar(20) not null,
 Essn int not null,
 empBirthday date  check(empBirthday > '1965-12-12 and empBirthday<1998-01-01'),
 empAddress varchar(30),
 empGender varchar(1),
 empSalary int not null,
 empSuperessn int,
 empBankCode int not null,
 primary key (Essn),
 foreign key (empBankCode) references Bank_Branches(bankCode) on delete restrict on update cascade
```

A trigger is a stored program invoked automatically in response to an event such as insert, update, or delete that occurs in the associated table. For example, you can define a trigger that is invoked automatically before a new row is inserted into a table.

We can create triggers for employee salaries, loan amounts or account balances when each salary for example should not be lower than some specified value. We can also create triggers that store data for deleted employees or customers, account types or loans. We can create triggers that serve as reminders to update, insert or delete data in the database.

Example of a Trigger used in this database:

```
DELIMITER $$

CREATE TRIGGER NoHigherBalance BEFORE INSERT ON Account FOR EACH ROW
BEGIN
IF NEW.balance>500 then set NEW.balance=500;
END IF;
END $$

DELIMITER ;
```

By definition, a stored procedure is a segment of declarative SQL statements stored inside the MySQL Server. Once you save the stored procedure, you can invoke it by using the CALL statement. And the statement returns the same result as the query. A stored procedure can have parameters so you can pass values to it and get the result back.

We can create procedures for information we want to retrieve from the database relations. For example to retrieve information for the all customers or employees in the bank, to retrieve all the dependent of a particular employee, to retrieve account or loan numbers of a certain bank, etc.

Example of a Stored Procedure used in this database:

```
DELIMITER $$

CREATE PROCEDURE GetBankBranchByLocation (IN location varchar(20))
BEGIN
SELECT * FROM Bank_Branches WHERE bankLocation=location;
END $$

DELIMITER ;

CALL GetBankBranchByLocation ('Tirane');
```

A stored function is a special kind stored program that returns a single value. Typically, we can use stored functions to encapsulate common formulas that are reusable among SQL statements or stored programs.

In this database, we can create functions that retrieve the age of employees or customers or the status (manager or employee) for employees, we can create function that return the account or loan types based on their balance etc.

Example of a Stored Function used in this database:

```
DELIMITER $$
CREATE FUNCTION age_of_employees(date1 DATE) RETURNS INT
DETERMINISTIC
BEGIN
DECLARE date2 DATE;
SELECT CURRENT_DATE() INTO date2;
RETURN YEAR(date2)-YEAR(date1);
END$$
DELIMITER ;
```

2. *Authorization (User Access Definitions)*

A database can have many users. Each user must have a user account. Database administrators are users with authority to specify new users, specify user's roles and grant general privileges for users. They may also revoke the privileges and delete users.

In this bank data base we can create user for example as administrator, who as mentioned before has the privilege to access everything on the database;

Managers, who can have access on statements as insert, delete update and select, on every part of the database;

Employees, who can have access on statements as insert, delete update and select of their dependents and Customers who have access on statements as select for the types of loans or accounts they can create.

Example of an Authorization used in this database:

```
create user manager@localhost identified by 'manager';

grant insert,update,select,delete on bank_database.* to manager@localhost;

show grants for manager@localhost;
```