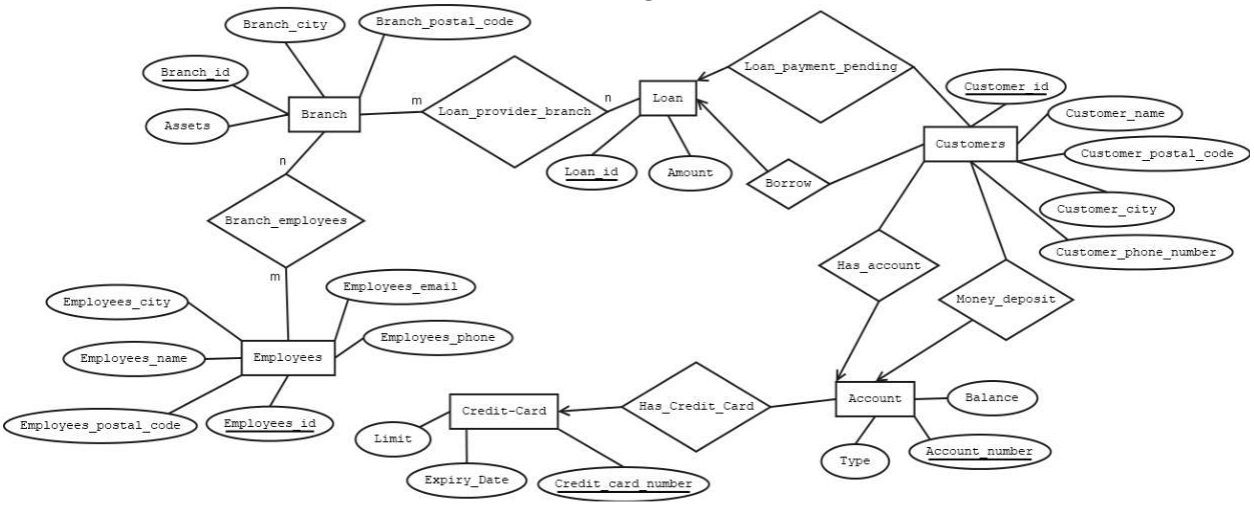
Bank Database System prototype

I was contacted by the World Bank to create a prototype of their production database system. This prototype will allow new employees to practice their database skills before they get to work on the production-level database system and use it to test the changes before they are applied on the real database. This database system contains a variety of tables, which includes six entity and seven relationships, and they are used to store information regarding customers, accounts, credit card, branches across the world, and loans. This database will also be used as a backup database in the case of an emergency. Moreover, I also wrote a program which will create a database as well as drop, create, and populate all the tables in the database. Finally, I programmed a user-friendly CLI, which allows administrators to query, insert, update, and delete data with the provided query in the program.

ER Diagram



Conversion to a Relational Schema

Entity:

Branch (Branch_id, Branch_city, Branch_postal_code, Assets)

Employees (Employees_id, Employees_name, Employees_city, Employees_postal_code, Employees_email, Employees_phone)

Customers (customer_id, customer_name, customer_postal_code, customer_city, customer_phone_number)

Account (Account_number, Account_type, Balance)

Loan (Loan_id, Amount)

Creditcard (Credit_card_number, Expiry_date, Limit)

Relationships:

Branch_employees (FK_Branch_id, FK_Employees_id)

Loan_provider_branch (FK_Branch_id, FK_Loan_id, Amount)

Borrow (FK_Loan_id, FK_customer_id)

Loan_payment_pending (FK_Loan_id, FK_customer_id, Amount)

Money_deposit (FK_customer_id, FK_Account_number, FK_Account_type, Desposit_amount)

Has_credit_card (FK_Account_number, FK_credit_card_number)

Has_account (FK_Account_number, FK_customer_id)

Functional Dependencies:

NOTE: There are more functional dependency in foreign key but they are mostly similar to the ones provided and besides foreign key other functional dependency seems to be trivial

Foreign Key:

Branch.Branch_id ⟶ Branch_employees.FK_Branch_id

Employees.Employees_id → Branch_employees.FK_Employess_id

Loan.Load_id ⟶ Loan_payement_pending.FK_Loan_id

Account.Account_number→ Has_account.FK_Account_number

Selection Queries:

1. Find Customer names, who deposited money:

```
SELECT customer_name
FROM Customers
WHERE customer_id IN (SELECT FK_customer_id FROM Money_deposit);
```

2. Finding total deposit by a user:

```
SELECT FK_account_number,sum(deposit_amount) AS Total_deposit
FROM Money_deposit
GROUP BY deposit_amount;
```

3. Finding how many deposit a customer made:

```
SELECT customer_name, COUNT(Deposit_amount), FK_Account_type
FROM Customers, Money_deposit
WHERE Customers.customer_id = Money_deposit.FK_customer_id
GROUP BY Deposit_amount;
```

4. Finding customers detail who has a credit card:

```
SELECT customer_id,customer_name,customer_postol_code,customer_city,customer_phone_number
FROM Customers, Has_account
WHERE (Customers.customer_id IN (Has_account.FK_customer_id)) AND
((Has_account.FK_Account_number) IN (SELECT FK_account_number FROM Has_credit_card));
```

5. Finding how many employees are there in one branch:

```
SELECT Branch_id, COUNT(Employees_id)
FROM Branch_employees, Branch, Employees
WHERE (Branch.Branch_id = (Branch_employees.FK_Branch_id)) AND (Employees.Employees_id =
(Branch_employees.FK_Employees_id))
GROUP BY (Branch_id);
```

Modify Queries:

1. Charging customers $0.50 per month if they have a balance less than 5000:

```
UPDATE Account set Balance = Balance - 0.50
WHERE Account_type = "Chequing" AND Balance = (SELECT Balance FROM Account WHERE (Balance
< 5000));
```

2. Changes card limit to $500 for each customer who has a credit-card:

```
UPDATE Creditcard set Card_limit = '500'
WHERE credit_card_number IN (SELECT FK_credit_card_number FROM Has_credit_card);
```

3. Adding a new customer:

```
INSERT INTO Customers (customer_id, customer_name, customer_postol_code, customer_city,
customer_phone_number) VALUES (115,"Yash","M1G398","Colchester","(416)615 9135");
```

4. This will delete an entry of a customer who finished paying his loan:

```
DELETE FROM Loan_payment_pending WHERE FK_Loan_id = 71;
```

5. This will delete customers with no account:

```
DELETE FROM Customers
WHERE Customers. Customer_id NOT IN (SELECT FK_customer_id FROM Has_account);
```