PAGE No.
DATE / /

DBMS PROJECT

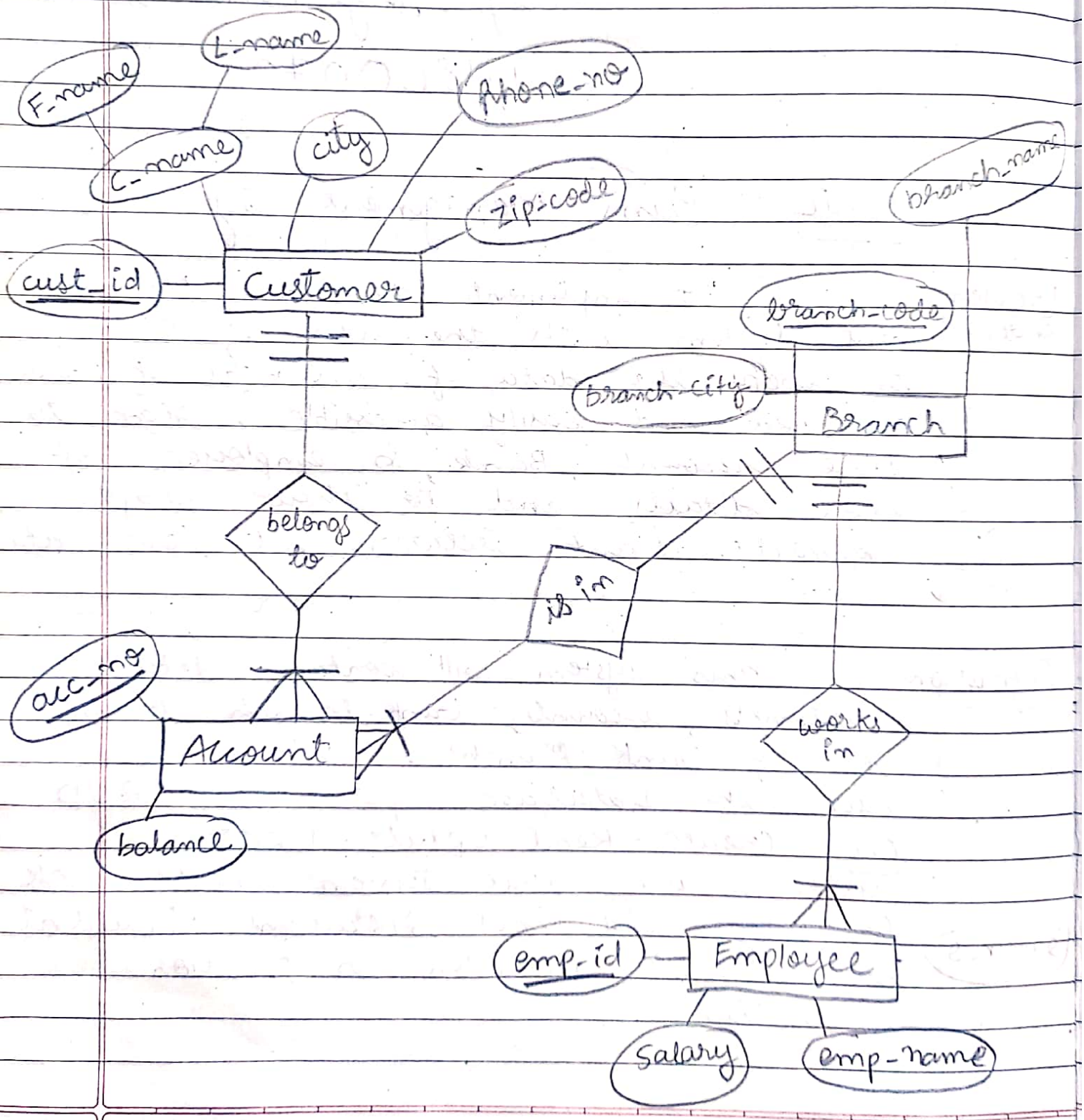
- By Jay Chawla
- U18CO070

Topic - Bank Management System

Problem Statement - To implement the Bank Management system with the help of DBMS, to keep the data of customers of Bank organised and easily accessible. Also to store account, Bank, & employees of bank details and to show proper connection / link between all these data.

Definition - This system will contain details of customers, accounts, bank branch & employees of bank branches. Admin can run the Database operations CRUD i.e Create - Read - Update - Delete, in such a way that system should work properly and this operations shouldn't create any Updation or Deletion anomalies.

* Entity - Relationship Diagram :-



* About ER Diagram

PAGE No.
DATE / /

⇒ Entities - There are total 4 entities shown in diagram, all 4 are strong entities as it have its own attribute which acts as primary key.

1. Customer - stores customer details
2. Account - stores account details
3. Branch - stores branch details
4. Employee - stores employee details

⇒ Cardinalities - Total 3 cardinalities or say relationships are present.

1. belongs-to - Many to one from account entity to customer entity.
2. is-in - Many to one from account entity to branch entity.
3. works-in - Many to one from employee entity to branch entity.

* Constraints

1. An account is owned by one and only one customer, but one customer can have multiple accounts.
2. A customer can have only one phone number.
3. A particular account can be in only one branch only i.e 2 branches can't have same accounts number. So each account number is not null & unique.
4. An employee can work in one branch only, he can't have job at multiple branches.

PAGE No.
DATE / /

* Conversion of ER Model to Relational Model

⇒ As customer is a Strong Entity, here cust-id acts as a primary key which is Unique and not Null.

∴ Customer (cust-id, f-name, l-name, city, zip-code, phone-no)

→ Here we have not included c-name as shown in ER diagram because c-name is a composite Attribute of first name (f-name) and last name (l-name).

⇒ Each account is different, so we have acc-no (Account number) as a primary key for Strong Entity Account.

∴ Account (acc-no, balance)

⇒ Strong Entity Branch have branch-code as a primary key.

∴ Branch (branch-code, branch-name, branch-city)

⇒ Each employee is different, he/she have its own emp-id which is not null & Unique i.e Primary key.

∴ Employee (emp-id, emp-name, salary)

→ A customer can have multiple accounts but an account is owned by only one customer so due to this many to one relationship we will update Account relation by adding cust-id as foreign key.

∴ Account (acc-no , balance , cust-id)

→ Similarly many to one relationship exists between account & branch. Updation in Account relation by adding branch code as foreign key.

∴ Account (acc-no , balance , cust-id , branch-code)

→ Also adding branch-code as foreign key to employee relation due to works-in many to one relationship bet^n employee & branch.

∴ employee (emp-id, salary, emp-name , branch-code)

∴ Listing all the relations / tables

① customer (cust-id , f-name , l-name , city , zip-code , phone-no).
② account (acc-no, balance , cust-id , branch-code)
③ branch (branch-code, branch-name, branch-city)
④ employee (emp-id , salary , emp-name , branch-code)

PAGE No.
DATE / /

* Normalization

⇒ List of all non-trivial functional dependencies are :-

(1) cust-id → f-name, l-name, city, phone-no, zip-code

(2) zip-code → city

(3) branch-code → branch-name, branch-city

(4) acc-no → balance, cust-id, branch-code

(5) emp-id → emp-name, salary, branch-code

⇒ Normal Forms

(i) First Normal form (1NF)

→ All attributes have atomic values and there are no composite and multivalued attributes. Hence relation is in 1NF

(ii) Second Normal form (2NF)

→ As there are no partial dependency in relations & relation is already in 1NF. Hence relation is in 2NF.

(iii) Third Normal form (3NF)

→ There exists 1 transitive dependency zip-code → city i.e nonprime → nonprime attribute

→ So we need to update customer relation and make a new relation.

PAGE No.
DATE / /

⇒ customer (cust-id, f-name, l-name, city, phone-no, zip-code)

↓

(1) customer (cust-id, f-name, l-name, phone-no, zip-code)

(2) city-code (zip-code, city)

⇒ Now the relation is in 3NF.

(iv) Boyce Codd Normal Form (BCNF)

→ For all possible FDs, LHS of FDs is a super key & relation is in 3NF.

∴ Relation is in BCNF.

(v) Fourth Normal Form (4NF)

→ As there is no multi valued dependency and relation is already in BCNF. Hence relation is in 4NF.

(vi) Fifth Normal form (5NF)

→ As there are no join dependencies & join is lossless, also relation is in 4NF. So relation is in 5NF.