Entities:

1. Account
2. Customer
3. Employee
4. Branch
5. Loan
6. Fixed_Deposit
7. Cards
8. Transaction

Relations:

1. Account_Opening :
    - customer,Branch and Account
2. AccountInBranch:
    - account and branch
3. Loan_Branch
    - Loan and Branch
4. Login:
    - Customer and Account
5. Transacts:
    - Account and Transaction
6. Borrow:
    - Loan and Customer
7. Repayment:
    - Customer and Loan
8. Employed:
    - Branch, Employee
9. Manages:
    - Branch, Employee

Weak Entity :

- Transaction : As it cannot uniquely identified with a account
- Repayment : It cannot exist without any loan
- FixedDeposit :It cannot exist without any account
- Card: It cannot exist without account
- EmployeePhone : It cannot exist without employee
- CustomerPhone: It cannot exist without customer

Ternary Relationship :

A ternary relationship is when three entities participate in the relationship

- AccountOpening : Since it have 3 entities :Branch,customer and account

Entities participation type,Relationship roles and Constraints:

Account balance constraint : Cannot perform transaction if balance is null.

Account opening constraint: **Cardinality Constraints on Ternary Relationship**
(One customer at one branch can only have one account)

Relationship Schema:

account(accountNo, userID, balance, password, accountType)

customer(customerID, customerName, creditScore, street, area, city, pincode)
customerPhone(customerID, phoneNumber)

employee(employeeID, employeeName, salary, street, area, city, pincode)
employeePhone(employeeID, phoneNumber)

branch(IFSC, branchArea, branchCity)

loan(loanID, customerId,interestRate, period, amount, EMI (derived), collateral)

fixedDeposit(fixedDepositID,accountNo, amount, period, interest)

card(cardNo, accountNo, Expiry, CVV,cardType spendLimit)

Receiver(recAccNo, IFSC)

accountOpening(accountNo, customerID, IFSC, dateOfOpening)

accountInBranch(accountNo, dateOfInception, IFSC)

loanInBranch(loanID, IFSC)

login(accountNo, time_Stamp, CustomerID,loggedIn)

transaction(transactionID, accountNo., recAccNo., mode, amount, time_Stamp)

lockIn(accountNo, fixedDepositID, startDate)

borrow(loanID, customerID, startDate)

repayment(paymentID, customerID, loanID, dateOfPayment, EMI)

employs (employeeID, joiningDate, IFSC)

manages(employeeID, dateOfAppointment, IFSC)

QUERIES AVAILABLE IN OUR DATABASE-

User Mode:

- View Account Info of a customer (Show everything in account through given customer ID in a branch)
- View Personal Information of self (Show customerName, creditScore, street, area, city, pincode through customerID)
- View All Transactions (Show transactionID, accountNo., mode, amount, time_Stamp and recAccNo., IFSC from receiver)
- View Balance/Statement
- Update Password
- Funds Transfer (Withdraw, Deposit through card or netbanking)
- Repay Loan
- Create a fixed deposit

Manager/Admin Mode:

- Create an account (Fill customerID, customerName, creditScore, street, area, city, pincode)
- Delete an account
- View All Accounts of every customer
- View personal data of employees
- View Transactions History of every account
- Update personal data of employees

EMBEDDED QUERIES-

User- **// Login by password and customerId**

View Details (input- customerId)

SELECT customerName, street, area, city, pincode FROM customer C
WHERE C.customerID = ‘customerId’ and exists (SELECT L.customerID FROM Login L
WHERE
L.customerId = ‘customerId*’* and L.loggedIn = True);

View All Transactions (Input- accountNo)

SELECT * FROM transaction WHERE accountNo = ‘*accountNo’*;

View Balance (Input- accountNo)

SELECT balance FROM account WHERE accountNo = ‘*accountNo’*;

Funds Withdrawl (Input- accountNo, amount)

START transaction;
SET transaction READ WRITE;
UPDATE account set balance = balance - 'given amount'
WHERE accountNo = 'accountNo';
commit;

Funds Deposit (Input- accountNo, amount)

START transaction;
SET transaction READ WRITE;
UPDATE account set balance = balance + 'given amount'
WHERE accountNo = 'accountNo';
commit;

Funds Transfer (Input- accountNo, recAccountNo, amount)

START transaction;
SET transaction READ WRITE;
UPDATE account set balance = balance - 'given amount'
WHERE accountNo = 'accountNo';
SAVEPOINT s1;
UPDATE account set balance = balance + 'given amount'
WHERE accountNo = 'RecAccountNo';
commit;

Create a Fixed Deposit (Input- accountNo, amount, period, interest)

START transaction; // TODO
SET transaction READ WRITE;

Change Name (Input- accountNo, updated customer name)

UPDATE customer SET customerName = ‘Updated Name’ WHERE customerID in (SELECT L.customerID from login L WHERE L.accountNo = *‘accountNo’* and L.loggedIn = True);

Change Password (Input- accountNo, updated password)
UPDATE account SET password = ‘Updated Password’ WHERE accountNo in (SELECT L.accountNo from login L WHERE L.accountNo = *‘accountNo’* and L.loggedIn = True);

Change PhoneNo (Input- accountNo, updated phoneNo)
UPDATE customerPhone SET phoneNo = ‘Updated PhoneNo’ WHERE customerID in (SELECT L.customerID from login L WHERE L.accountNo = *‘accountNo’* and L.loggedIn = True);

Employee Mode-

- Create an account (Fill customerID, customerName, creditScore, street, area, city, pincode)
- Delete an account
- View All Accounts of every customer
- View Transactions History of every account
- Update personal data of employees

View Details (Input- employeeID)
SELECT * FROM employee
WHERE employeeID = ‘employeeID’ UNION
SELECT * FROM employeePhone where employeeID = ‘employeeID’;

;

SQL STATEMENTS-

- *View Account info*

```
select AccountNo,UserID,IFSC,branchArea,branchCity
from account A, branch B, accountInBranch I
where A.account = I.Account and I.IFSC = B.IFSC
```

- *View accounts from particular IFSC*

```
select AccountNo,UserID,IFSC,branchArea,branchCity
from account,branch
where IFSC =”PAYTM0123456”
```

- *View accounts by balance*

```
select AccountNo,UserID,IFSC,branchArea,branchCity
from account,branch
order by balance
```

- *View accounts with balance >1000000 and from Jaipur branch*

```
select AccountNo,UserID,IFSC,branchArea,branchCity
from account,branch
where balance>1000000
        and branchCity = “Jaipur”
```

- ***View loan customers who took loan in 2015 and 2017***

```
Select C.customerID, C.customerName
from customer as C, borrows as B, loan as L
where L.loanID in ((select LoanID from Loan where where StartDate>1.1.2015 and
StartDate<31.12.2015)
union (select loanID from loan where StartDate>1.1.2016 and StartDate<31.12.2017)) and
B.loanID = L.loanID and C.customerID in (select B.customerID from borrows)
```

- *Take a loan of 100000 on the customerID 203333112 for a period of 3 years giving your bike as collateral at an interest rate of 12%*

```
INSERT INTO loan(loanID, interestRate, period, amount, EMI (derived), collateral)
VALUES(1001, 12, 3, 100000, 12000, ‘bike_papers’)
```

- ***Transfer 2000 from accountNo 20312234 to accountNo 21122221 through UPI mode***

```
((UPDATE account set balance=(balance-2000) where accountNo = 20312234)
union
(UPDATE account set balance=(balance+2000) where accountNo = 21122221)
union
(INSERT INTO transaction(transactionID, accountNo., recAccNo., mode, amount, time_Stamp)
VALUES(1001, 20312234, 21122221, ‘UPI’, 2000, ‘11.02.2022 23:02’)))
```

- *Repay the loan of 100000 where loanID was 1001*

```
DELETE FROM loan where loanID = 1001
```

- *Withdraw 2000 from accountNo 20312234*

```
UPDATE account set balance=(balance-2000) where accountNo = 20312234
```

- *Update from accountNo 20312234*

UPDATE account set password=”#OONGA_BOONGA#” where accountNo = 20312234 and password = “old_password”

- *View Total balance of an accountNo*

SELECT balance from account where accountNo = 20312234

- *Avg balance of customerID 2*

SELECT avg(balance) from account where accountNo = 20312234

- *Find number of employees who have been a manager more than once*

Select employeeID, employeeName
from employee as e, manages as m
where count(e.employeeID = m.employeeID) > 1

- *average salary is greater than 42,000*

SELECT employeeName, avg_salary
from (select employee, avg (salary) as avg_salary
from employee
where avg_salary > 42000)

- *View Transaction history of accountNo 2033122*

SELECT * FROM transaction WHERE accountNo = 2033122

- Select accountNo, balance,
  From account
  Group by balance

- Select customer.customer.Name, account.accountNo
  From customer
  CROSS Join account

EMBEDDED QUERIES-

1)

select A.accountNo, O.IFSC, A.balance from account as A, accountOpening as O where A.accountType = %s and A.accountNo = O.accountNo and O.customerID = %s

2)
SELECT T.transactionID, T.accountNo, T.recAccNo, T.mode, T.amount, T.time_Stamp from transaction T, accountOpening O WHERE O.customerID = %s and T.accountNo = O.accountNo

3)
SELECT * from Loan L where L.loanID in (SELECT B.loanID from borrow B where B.customerID = %s)

4)
SELECT * from fixeddeposit D where D.accountNo in (SELECT A.accountNo from accountopening A where A.customerID = %s)

5)
SELECT C.cardNo, C.Expiry, C.CVV, C.spendLimit, A.balance FROM Card C, Account A WHERE C.accountNo in (SELECT O.accountNo FROM accountopening O WHERE O.customerID = %s) and C.cardtype = %s and A.accountNo = C.accountNo

6)
SELECT * from employee_view_cust

7)
SELECT * from manager_view_cust

8)
SELECT * from manager_view_emp

9)
INSERT into employee (employeeName, salary, street, area, city, pincode) values(%s, %s, %s, %s, %s, %s)

10)
INSERT into transaction(accountNo, recAccNo, amount) values (%s, %s, %s)

VIEWS-

```
-- VIEW 1
create view employee_view_cust as
select c.customerName, c.creditScore, a.accountNo, a.balance, a.accountType
from customer c, account a, accountOpening ao
where ao.customerID = c.customerId and ao.accountNo = a.accountNo ;
```

```
-- VIEW 2
create view manager_view_cust as
select c.customerName, c.creditScore, c.customerID, a.accountNo, a.balance, a.accountType ,
b.IFSC
from customer c, account a, accountOpening ao, accountInBranch aib, branch b
where ao.customerID = c.customerId and ao.accountNo = a.accountNo and aib.accountNo =
a.accountNo and b.IFSC = aib.IFSC;

-- VIEW 3
create view manager_view_emp as
select e.employeeID, e.employeeName, e.salary
from employee e;
```

INDEXES-

```
CREATE INDEX idx_transact ON transaction(accountNo);
CREATE INDEX idx_acc ON account(accountNo);
CREATE UNIQUE INDEX idx_card ON card(accountNo);
CREATE INDEX idx_accOpen ON accountopening(customerID);
CREATE INDEX idx_customer ON customer(customerID);
CREATE INDEX idx_loan ON loan(loanID);
CREATE INDEX idx_FD ON fixeddeposit(accountNo);
```

GRANTS-

1)
```
create User 'manager'@'localhost' IDENTIFIED BY 'password';
Grant all on onlinebank.* to 'manager'@'localhost' with grant option;
flush privileges;
```

2)
```
create User 'customer'@'localhost' IDENTIFIED BY 'password';
Grant SELECT ON onlinebank.* to ‘customer'@'localhost';
```

3)
```
create User 'admin'@'localhost' IDENTIFIED BY 'password';
Grant ALL ON onlinebank.* to ‘customer'@'localhost' with grant option;
```

4)
```
create User 'employee'@'localhost' IDENTIFIED BY 'password';
GRANT ALL ON onlinebank.transaction,onlinebank.account,onlinebank.employee,
onlinebank.customer, onlinebank.accountopening to ‘employee’@’localhost’;
```