INDEX

1. Introduction	3 – 4
2. E-R Diagram	5
3. E-R to Table	6
4. Normalisation	7 – 9
5. SQl & PL/SQL Command Lines	10 – 24
6. Conclusion	25
7. References	26

INTRODUCTION

A Banking Management System (BMS) is a type of Database Management System (DBMS) that is used to manage the operations of banks and financial institutions. A DBMS is a software system that is used to manage databases, which are collections of data that are organized in a structured manner. The BMS is a critical component of modern banking infrastructure, as it enables banks to efficiently manage their operations and provide excellent customer service.

The BMS is designed to support various banking functions, including customer information management, account management, transaction processing, and reporting. The system is responsible for managing and organizing vast amounts of data related to banking operations, such as customer information, account balances, transaction histories, and financial performance data.

The "Bank Management System" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the need of the company to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly. Bank Management System, as described above, can lead to error free, secure, reliable, and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus, it will help organization in better utilization of resources.

Every organization, whether big or small, has challenges to overcome and managing the information of Account, Customer. Deposit, Account Type, Amount. Every Bank Management System has different Customer needs; therefore, we design exclusive employee management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future goals. Also, for those busy executive who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all times. These systems will ultimately allow you to better manage resources.

The BMS enables customers to access their accounts and perform various transactions, such as depositing funds, withdrawing funds, and transferring funds. The BMS also provides real-time updates to customers on their account balances, transaction histories, and other relevant information.

The BMS is also designed to provide high levels of security to protect customer data and prevent fraud. The system includes various security measures, such as access controls,

encryption, and authentication, to ensure that customer data is only accessible to authorized personnel.

Requirement Analysis:

The development of a banking management system requires a thorough analysis of the requirements of the system. The requirements analysis is the process of identifying the needs and expectations of the stakeholders and converting them into a set of functional and non-functional requirements. The stakeholders of a BMS may include bank management, employees, customers, and regulatory authorities. The requirements analysis process should involve the following steps:

1. Identification of Stakeholders
2. Requirements Gathering
3. Requirements Analysis
4. Requirements Specification

Functional Requirements:

Functional requirements are the requirements that describe what the system should do. The functional requirements of a BMS may include the following:

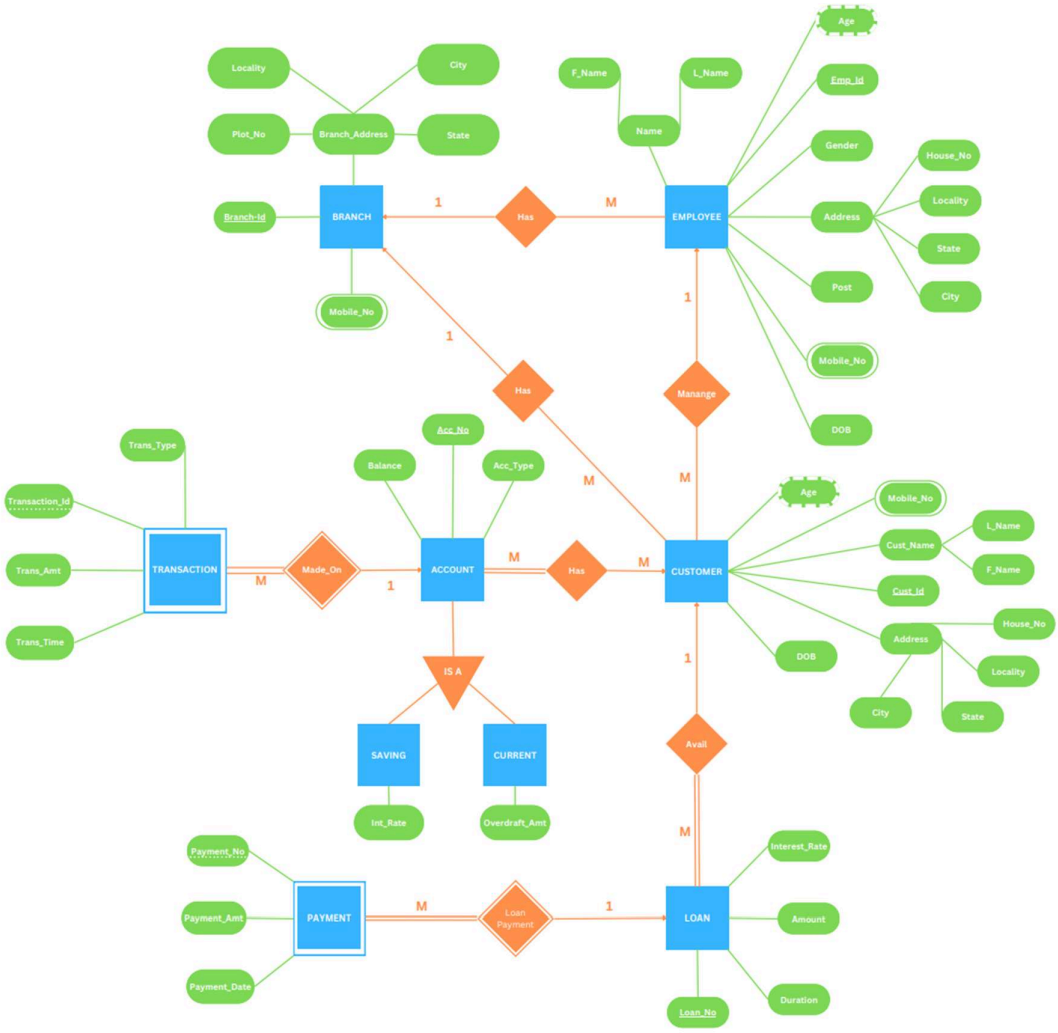
1. Customer Management
2. Account Management
3. Transaction Processing
4. Reporting

Non-functional Requirements:

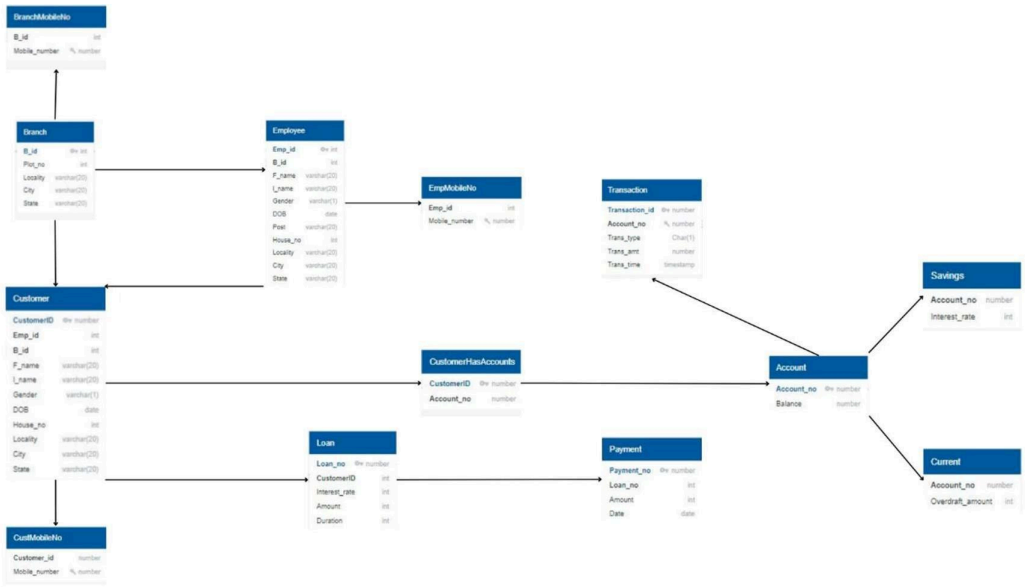
Non-functional requirements are the requirements that describe how the system should perform. The non-functional requirements of a BMS may include the following:

1. Security
2. Reliability
3. Performance
4. Scalability

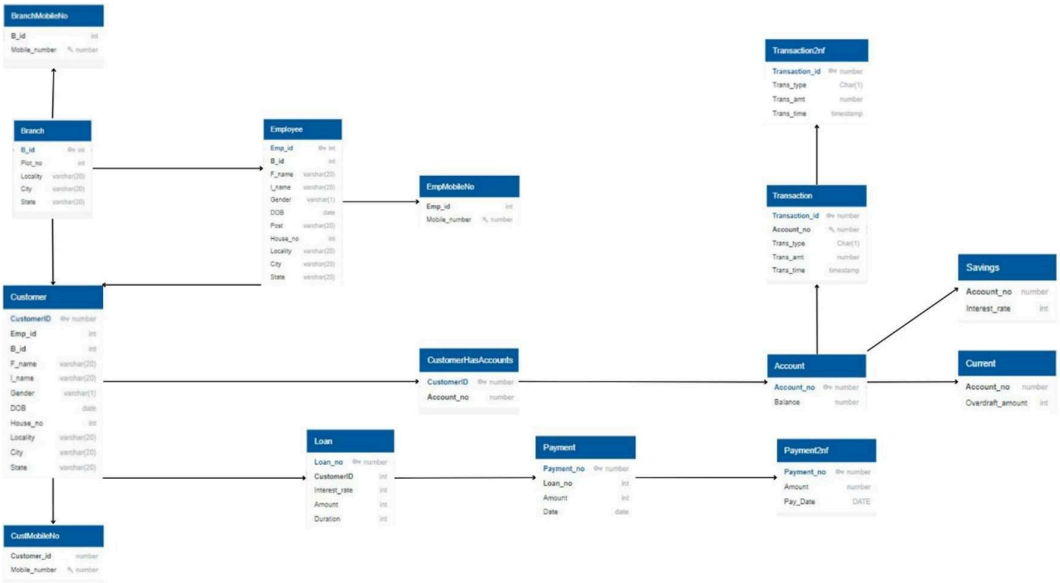
E-R DIAGRAM


Locality
City
Plot_No
Branch_Address
State
Branch-Id
BRANCH
1
Has
M
EMPLOYEE
F_Name
L_Name
Name
Age
Emp_Id
Gender
House_No
Locality
Address
State
City
Post
Mobile_No
DOB
Mobile_No
1
1
Has
Manange
M
M
Trans_Type
Transaction_Id
Trans_Amt
Trans_Time
TRANSACTION
M
Made_On
1
ACCOUNT
Acc_No
Balance
Acc_Type
M
Has
M
CUSTOMER
Age
Mobile_No
Cust_Name
L_Name
F_Name
Cust_Id
Address
House_No
Locality
City
State
DOB
1
IS A
SAVING
CURRENT
Int_Rate
Overdraft_Amt
Avail
M
Payment_No
Payment_Amt
Payment_Date
PAYMENT
M
Loan Payment
1
LOAN
Interest_Rate
Amount
Duration
Loan_No

E-R TO TABLE



1 – NF



2 – NF & 3 - NF

NORMALISATION

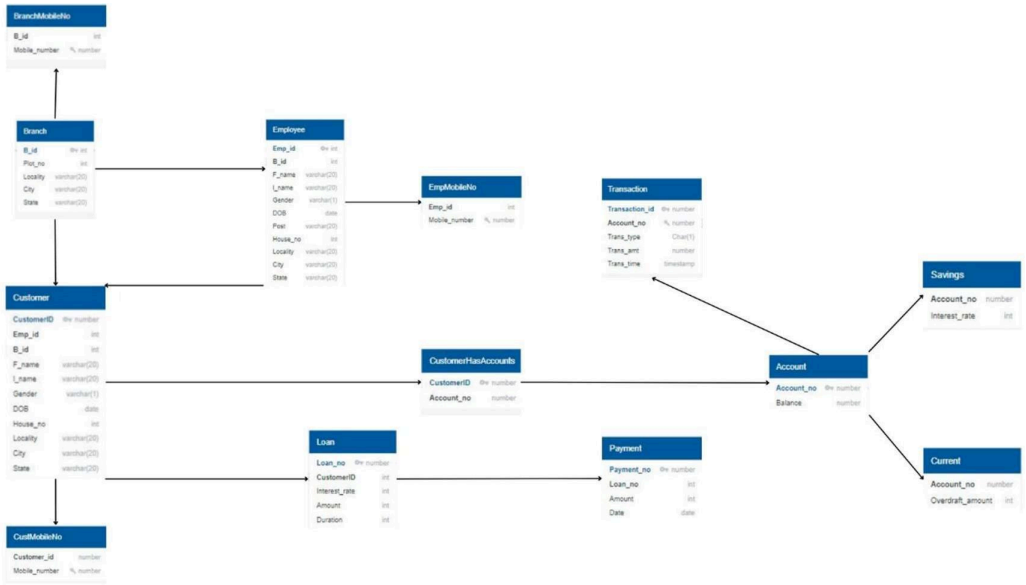
First Normal Form (1NF):

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relation is in first normal form if it does not contain any **composite** or **multi-valued attribute**. A relation is in first normal form if every attribute in that relation is singled valued attribute.

A table is in 1 NF if:

1. There are only Single Valued Attributes.
2. Attribute Domain does not change.
3. There is a unique name for every Attribute/Column.
4. The order in which data is stored does not matter.

While converting our er diagram to tables, we made sure to that we carefully test our tables on all 4 grounds stated above and then made our first table that satisfied the given constraints.



Now, First Normal Form (1NF) does not eliminate redundancy, but rather, it's that it eliminates repeating groups. Instead of having multiple columns of the same kind of data in a record, (0NF or Unnormalized form) you remove the repeated information into a separate relation and represent them as rows. This is what constitutes 1NF.

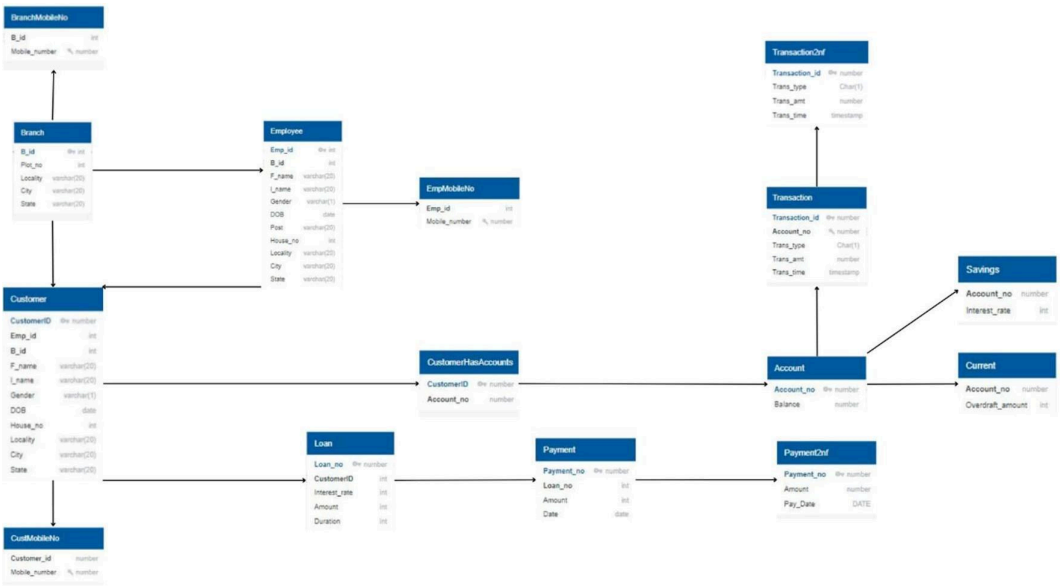
Second Normal Form (2NF):

Second Normal Form (2NF) is based on the concept of full functional dependency. Second Normal Form applies to relations with composite keys, that is, relations with a primary key composed of two or more attributes. A relation with a single-attribute primary key is automatically in at least 2NF. A relation that is not in 2NF may suffer from the update anomalies. To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency.

A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table. In other words,

A relation that is in First Normal Form and every non-primary-key attribute is fully functionally dependent on the primary key, then the relation is in Second Normal Form (2NF).

Note: If the proper subset of candidate key determines non-prime attribute, it is called partial dependency. The normalization of 1NF relations to 2NF involves the removal of partial dependencies. If a partial dependency exists, we remove the partially dependent attribute(s) from the relation by placing them in a new relation along with a copy of their determinant.



In our ER diagram we have two tables that have composite primary keys, those are Transaction and Payment. In Transaction table we added Transaction_id and our Account_no. as composite key, but we observed that all the other attributes in our table are actually just dependent on our Transaction_id and account no. plays no role in them(attributes), we observe that a partial dependency exists in this table. So, we created an extra table i.e. Transaction2nf which has Transaction_id and all the other attributes and set transaction id as the primary key and kept only Transaction_id and Account_no in the first table. So, this removes the partial dependency that existed earlier. Similarly, we did the same for Payment entity, where the attributes were dependent on Payment_no, but our composite primary key consists of loan_no and

payment_no. So we created a separate table as payment 2nf which has payment no. as the primary key.

Third Normal Form (3NF):

Although Second Normal Form (2NF) relations have less redundancy than those in 1NF, they may still suffer from update anomalies. If we update only one tuple and not the other, the database would be in an inconsistent state. This update anomaly is caused by a transitive dependency. We need to remove such dependencies by progressing to Third Normal Form (3NF).

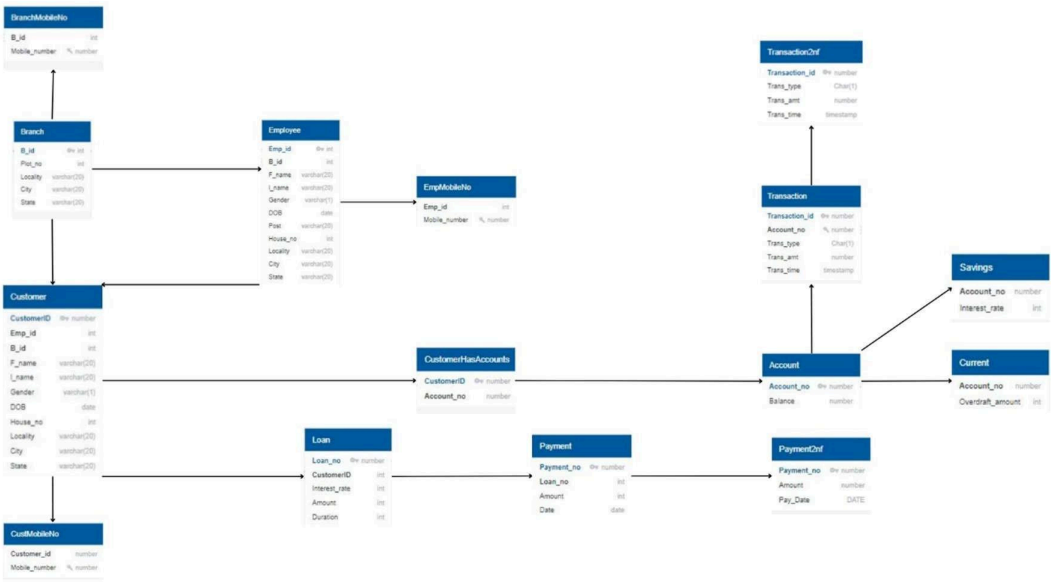
A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.

A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y:

1. X is a super key.
2. Y is a prime attribute (each element of Y is part of some candidate key).

In other words, A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

The normalization of 2NF relations to 3NF involves the removal of transitive dependencies. If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.



This is the same table from above that we created for 2nf, while normalising our table to 1nf and 2nf we made change in such a way that here we did not need to change anything to normalise it to 3nf, since it was already in 3nf.

SQL & PL/SQL COMMAND LINES

Table Creation:

```
CREATE TABLE Branch (B_id int PRIMARY KEY NOT NULL ,
    Plot_no int  NOT NULL ,
    Locality varchar(20)  NOT NULL ,
    City varchar(20)  NOT NULL ,
    State varchar(20)  NOT NULL);
```

```
CREATE TABLE Employee (Emp_id int PRIMARY KEY NOT NULL ,
    B_id int  NOT NULL,
    F_name varchar(20)  NOT NULL ,
    l_name varchar(20)  NOT NULL ,
    Gender varchar(1)  NOT NULL ,
    DOB date  NOT NULL ,
    Post varchar(20)  NOT NULL ,
    House_no int  NOT NULL ,
    Locality varchar(20)  NOT NULL ,
    City varchar(20)  NOT NULL ,
    State varchar(20)  NOT NULL ,
    FOREIGN KEY(B_id) REFERENCES Branch(B_id) ON DELETE SET NULL,
    CONSTRAINT DOB CHECK ((DOB) < DATE '2004-01-01'));
```

Constraint on DOB to check that it should be less that ‘2004-01-01’

```
CREATE TABLE Loan (Loan_no number  PRIMARY KEY NOT NULL ,
    CustomerID number  UNIQUE NOT NULL ,
    Interest_rate number  NOT NULL CHECK (Interest_rate between 1 AND 15) ,
    Amount int  NOT NULL CHECK (Amount>10000) ,
    Duration int  NOT NULL ,
    FOREIGN KEY(CustomerId) REFERENCES Customer(CustomerID) ON DELETE SET NULL);
```

```
CREATE TABLE BranchMobileNo (
    B_id int  NOT NULL ,
    Mobile_number number PRIMARY KEY NOT NULL,
    FOREIGN KEY(B_id) REFERENCES Branch(B_id) ON DELETE SET NULL,
    CONSTRAINT Mobile_number CHECK (length(TO_CHAR(Mobile_number)) <11)
);
```

Constraint on Mobile_Number to check that it’s length should be less than 11.

~ On Delete Set Null clause sets all the records of the column which is defined as a foreign key in the child table to Null if the corresponding record in the parent table is deleted.

Value Insertion:

```
INSERT INTO Branch (B_id, Plot_no, Locality, City, State) VALUES (4, 123, 'MG Road', 'Bangalore', 'Karnataka');

INSERT INTO Branch VALUES (9, 111, 'Marine Drive', 'Mumbai','Maharashtra');

INSERT INTO Branch VALUES (10, 71, 'Anand Vihar', 'Delhi','Delhi');
```

```
INSERT INTO Account  VALUES (10001, 1500000,'Savings');

INSERT INTO Account  VALUES (10002, 250000,'Current');

INSERT INTO Account  VALUES (10003, 75000,'Savings');

INSERT INTO Account  VALUES (10004, 300000,'Current');

INSERT INTO Account  VALUES (10005, 200000,'Savings');

INSERT INTO Account  VALUES (10006, 100000,'Current');

INSERT INTO Account  VALUES (10007, 150000,'Current');
```

Variable Declaration:

```
declare
    table_name_v varchar(50);
    choice_v varchar(10);

    branch_id_v int;
    plot_no_v int;
    state_v varchar(20);
    city_v varchar(20);
    locality_v varchar(20);
    v_bid Branch.B_id%TYPE;
    v_plot_no Branch.Plot_no%TYPE;
    v_locality Branch.Locality%TYPE;
    v_city Branch.City%TYPE;
    v_state Branch.State%TYPE;
    cursor branch_cursor is
        select * from Branch;
```

Driver Code:

```
begin
    DBMS_OUTPUT.PUT_LINE('********************** WELCOME TO KALYAN BANK ******************************');
    DBMS_OUTPUT.PUT_LINE('********************** CREATING BANKING TRUST ******************************');
    DBMS_OUTPUT.PUT_LINE('Which table do you want to access? branch, employee, customer, loan, payment, account, transaction');
    -- table_name_v:='payment2nf';
    table_name_v:='&table_name_v';

    if table_name_v='branch' then
        DBMS_OUTPUT.PUT_LINE('Do you want to insert, update, delete or display data?');
        -- choice_v:='display';
        choice_v:='&choice_v';
```

```
   if choice_v='insert' then
        DBMS_OUTPUT.PUT_LINE('Enter branch_id, plot_no, state, city, locality to insert');
--        branch_id_v:='1';
       -- plot_no_v:='19';
--        state_v:='Delhi';
--     city_v:='Delhi';
--        locality_v:='Paschim Vihar';
       branch_id_v:='&branch_id_v';
       plot_no_v:='&plot_no_v';
       state_v:='&state_v';
       city_v:='&city_v';
       locality_v:='&locality_v';

       insert into Branch(B_id, Plot_no, Locality, City, State) values(branch_id_v,plot_no_v,locality_v,city_v,state_v);
       DBMS_OUTPUT.PUT_LINE('New data has been inserted into the table.');
```

```
      elsif choice_v='update' then
          DBMS_OUTPUT.PUT_LINE('Enter branch_id, plot_no, state, city, locality to update');
    --        branch_id_v:='1';
          -- plot_no_v:='20';
    --        state_v:='Uttar Pradesh';
    --     city_v:='Kanpur';
    --        locality_v:='Azad Nagar';
          branch_id_v:='&branch_id_v';
          plot_no_v:='&plot_no_v';
          state_v:='&state_v';
          city_v:='&city_v';
          locality_v:='&locality_v';

          update Branch set
              B_id=branch_id_v,
              Plot_no=plot_no_v,
              Locality=locality_v,
              City=city_v,
              State=state_v
              where B_id=branch_id_v;
          DBMS_OUTPUT.PUT_LINE('Data has been updated in the table.');
```

```
elsif choice_v='delete' then
    DBMS_OUTPUT.PUT_LINE('Enter branch_id to delete');
    -- branch_id_v:='1';
    branch_id_v:='&branch_id_v';
    delete from branch where B_id= branch_id_v;
    DBMS_OUTPUT.PUT_LINE('Data has been deleted from the table.');

elsif choice_v='display' then
    open branch_cursor;
    loop
        fetch branch_cursor into v_bid, v_plot_no, v_locality, v_city, v_state;
        exit when branch_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_bid || ' ' || v_plot_no || ' ' || v_locality || ' ' || v_city || ' ' || v_state);
    end loop;
    close branch_cursor;

ELSE
    DBMS_OUTPUT.PUT_LINE('Invalid choice. Please enter "insert", "update", "delete",or "display"');
END IF;
```

```
if choice_v='withdraw' then

    account_no_v:='120783645982';
    transaction_id_v:=TO_CHAR(DBMS_RANDOM.VALUE(1000000, 9999999));
    Trans_amt_v:='1000';

    UPDATE Account
    SET Balance = Balance - Trans_amt_v
    WHERE account_no = account_no_v;

    INSERT INTO transactions1 (Transaction_id,Account_no) VALUES (transaction_id_v, account_no_v);
    INSERT INTO transaction2nf (Transaction_id,Trans_type,Trans_amt,Trans_time) VALUES (transaction_id_v,'Withdraw',Trans_amt_v,SYSDATE);

    COMMIT;
```

This is to process the withdrawal amount from the bank.

```
elsif choice_v='history' then
    account_no_v:='120783645982';

    FOR txn IN (SELECT Trans_amt,trans_time,Trans_type FROM Transactions1,Transaction2nf
        WHERE Transactions1.account_no = account_no_v and Transactions1.Transaction_id=Transaction2nf.Transaction_id ORDER BY trans_time DESC)
    LOOP

    DBMS_OUTPUT.PUT_LINE( ', Amount: ' || txn.Trans_amt || ', Date: ' || txn.Trans_time || ', Type:' || txn.Trans_type);
    END LOOP;

ELSE
    DBMS_OUTPUT.PUT_LINE('Invalid choice. Please enter "withdraw","deposit","history"');

end if;
```

This to process the history of transactions that has occurred on the bank account.

```
elsif table_name_v='loan' then
    DBMS_OUTPUT.PUT_LINE('Do you want to insert, update, delete or display data?');
    -- choice_v:='display';
    choice_v:='&choice_v';

    if choice_v='insert' then
        DBMS_OUTPUT.PUT_LINE('Enter loan_no, CustomerID, interest_rate, amount, duration to insert.');
--          loan_no_v:='12345';
        -- CustomerID_v:='8796543210';
--          interest_rate_v:='5';
--          amount_v:='100000';
--          duration_v:='1';
        loan_no_v:='&loan_no_v';
        CustomerID_v:='&CustomerID_v';
        interest_rate_v:='&interest_rate_v';
        amount_v:='&amount_v';
        duration_v:='&duration_v';

        insert into Loan values(loan_no_v,CustomerID_v,interest_rate_v ,amount_v, duration_v);
        DBMS_OUTPUT.PUT_LINE('New data has been inserted into the table.');
```

```
elsif table_name_v='payment' then
    DBMS_OUTPUT.PUT_LINE('Do you want to insert, update, delete or display data?');
    -- choice_v:='display';
    choice_v:='&choice_v';

    if choice_v='insert' then
        DBMS_OUTPUT.PUT_LINE('Enter loan_no and payment_no to insert.');
--          loan_no_v:='12345';
        -- payment_no_v:='67890';
        loan_no_v:='&loan_no_v';
        payment_no_v:='&payment_no_v';


        insert into payment values(loan_no_v,payment_no_v);
        insert into payment2nf values(payment_no_v,amount_v,pay_date_v);
        DBMS_OUTPUT.PUT_LINE('New data has been inserted into the table.');
```

CONCLUSION

In conclusion, a Banking Management System (BMS) is an essential tool for managing financial transactions, customer information, and regulatory compliance in a banking institution. The system is designed to streamline banking operations, enhance customer service, and increase efficiency in managing financial resources. The use of a Database Management System (DBMS) is crucial in ensuring data integrity, security, and consistency in the management of the banking system.

In this paper, we have discussed the various features and benefits of a BMS. The system provides an integrated platform for managing all banking operations, including customer account management, transaction processing, and regulatory compliance. The use of a DBMS ensures that the data is stored in a secure and efficient manner, with proper data access controls and backup procedures. Additionally, the system enables quick and easy access to customer information, transaction history, and financial reports.

The benefits of implementing a BMS are numerous. First, the system enhances customer service by enabling faster processing of transactions, providing online banking services, and enabling the bank to offer new financial products and services. Secondly, the system improves operational efficiency by reducing manual processes, automating repetitive tasks, and providing real-time data access. Thirdly, the system facilitates regulatory compliance by ensuring that all transactions are recorded and monitored, and that customer data is stored securely and accessed only by authorized personnel.

In conclusion, the implementation of a BMS with a robust DBMS is critical to the success of a modern banking institution. The system offers numerous benefits, including improved customer service, operational efficiency, and regulatory compliance. Additionally, the system enables banks to stay competitive by offering new products and services and expanding their reach to new customers. The use of a DBMS is crucial in ensuring data integrity, security, and consistency, thereby enhancing the overall efficiency and effectiveness of the banking system.

REFERENCE

https://www.geeksforgeeks.org/normal-forms-in-dbms/

https://www.geeksforgeeks.org/first-normal-form-1nf/

https://www.geeksforgeeks.org/second-normal-form-2nf/

https://www.geeksforgeeks.org/third-normal-form-3nf/

https://www.geeksforgeeks.org/er-diagram-of-bank-management-system/

https://www.youtube.com/watch?v=RVyCJXn--jY

https://app.quickdatabasediagrams.com/#/

https://soft-builder.com/bank-management-system-database-model/