**Bank MANAGEMENT System**

Project report submitted in partial fulfilment of the requirement for the award of the Degree of

B. Tech

Submitted By

Antriksh Kumar Chaudhary (3078)

Rishab Hanspaal(30 )

Avdhesh Bharadwaj (30 )



Department of Computer Science and Engineering

**JAYPEE UNIVERSITY**

**ANOOPSHAHAR**

**Certificate**

This is to certify that the project report entitled **“BANK MANAGEMENT SYSTEM”** being submitted by **Antriksh Kumar Chaudhary (3053), Rishab Hanspaal (30 ), Avdhesh Bharadwaj (30 )** in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering from the Jaypee University, Anoopshahar is a record of bonafied work carried out under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

**Guide Name: Mr. Gaurav Raj**, Assistant professor (Dept. of Computer Science & Engineering)

**Head of the Department: Prof. Md. Qasim Rafiq**

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our guide **Mr. Gaurav Raj** (Assistant prof., Jaypee University, Anoopshahar) for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark.

Lastly, we thank almighty, my parents, brother and friends for their constant encouragement without which this assignment would not be possible.

Antriksh Kumar Chaudhary (3078)

Rishab Hanspaal(30 )

Avdhesh Bharadwaj (30 )

**INTRODUCTION**

A bank is a commercial or state institution that provide financial services, including issuing money in form of coins, bank notes, receiving deposits of money, lending money and processing transactions. A commercial bank accepts deposits from customers and in turn makes loans based on those deposit.

1. **Project Objective**

* Locate any account with details by user.
* Reduce clerical work as most of the work is done by computer.
* Provide speed & reduce time consumption.
* To create a consistent database.

1. **Project Benefits**

* The database is always consistent
* Atomicity and concurrency control

1. **Project Scope**Banking activities are considered to be the life blood of the national Economy. Without banking services, trading and business activities cannot be carried on smoothly. Banks are the distributors and protectors of liquid capital which is of vital significance to a developing country.  
     
   Efficient administration of the banking system helps in the economic Growth of the nation. Banking is useful to trade and commerce.

**Tools Used in the Project:**

* The s/w that we used to create database: **ORACLE 12c**
* Coding language: **SQL**
* The interface/application program interface used: **JAVA**
* The platform used: **Eclipse Mars1**

Description of Tools Used in the Project:

Oracle 12c: A very popular relational database management system(DBMS) from Oracle, introduced in1978. Oracle was the first database product to run on a huge variety of hardware from micro to mainframe, giving it a major competitive advantage in the 1980s.

SQL**:**   
Sql is a multi-paradigm **designed** by **Donald D. Chamberlin Raymond F.Boyce, Developed** by **ISO/IEC**.

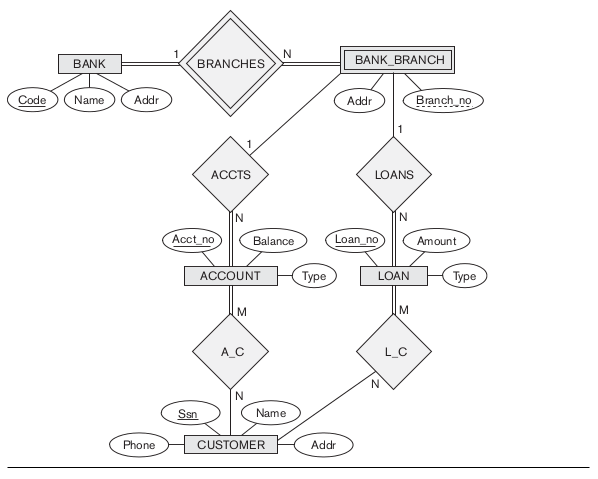
It first appeared in1974.

Sql became a standard of the **American National Standards Institute (ANSI)** in 1986, and of the **International Organization for Standardization (ISO)i**n 1987

ECLIPSE:The Net Beans Platform provides a virtual file system, which is hierarchical registry for storing user settings, comparable to Windows Registry on Microsoft Windows Systems.

JAVA:   
Java is a general purpose, **high level programming language** developed by **SUN MICRMICROSYSTEM**. A small team of engineers, known as the **Green Team**, **initiated the language in 1991**.

E-R Diagram



Key:

* Addr - Address
* Acct\_no - Account Number
* Accts - ACCOUNTS
* A\_C - Account Customer
* L\_C - Loan Customer
* Ssn - Customer Number

Note: Underlined attributes are primary key.

**Steps for connecting Oracle Database to Eclipse**

**Code:**



Inside the Input class, the execute method connects to the oracle database when called. Its return type is of the form ResultSet.

1. **Register the driver class:** the forname of the class is used to register the driver class. It dynamically load the driver class.  
   Example: Class.forName("oracle.jdbc.driver.OracleDriver"); //line number 9  
   Note: for oracle 12c the driver class is oracle.jdbc.driver.OracleDriver predefined.
2. **Create the connection object:**  The getConnection() method of DriverManager class is used to establish connection with the database.  
   Example: Connection con=DriverManager.getConnection(URL,user,pass); //line no. 16.  
   Note: The connection URL format for oracle is **jdbc:oracle:thin:@localhost:port:service\_name where by default port no. is 1521 and service name is xe.But these values can be changed during installation. In my code/case service name is antriksh.**
3. **Create Statement Object:** The createStatement() method of Connection interface is used to create statement.  
   Example: line number 18.
4. **Execute Query:** The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.  
   Example: line number 20.
5. **Close Connection:** By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.  
   Example: con.close();   
   Note: I have closed the connection in main otherwise execute method wouldn’t return ResultSet data as it would have been closed amd hence always return null.

**Some What the Heck Explanations(ehh…Definations):**

1. **ResultSet:** The object of ResultSet maintains a cursor pointing to a particular row of data. It act as an iterator, initially, cursor points to before the first row.  
   *By default, ResultSet object can be moved forward only and it is not updatable.*But we can make this object to move forward and backward direction by passing either TYPE\_SCROLL\_INSENSITIVE or TYPE\_SCROLL\_SENSITIVE in createStatement(int,int) method.  
   **Some method of resultset used here:**

|  |  |
| --- | --- |
| **public boolean next():** | is used to move the cursor to the one row next from the current position |
| **public boolean beforFirst():** | is used to move the cursor before the starting of the retrieved data, i.e. before first row |

1. **DriverManager:** The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.  
   **Some method of DriverManager used here:**

|  |  |
| --- | --- |
| **public static Connection getConnection(String url,**  **String username, String password)** | is used to establish the connection with the specified url. |

1. **Connection:** A Connection is the session between java application and database. The Connection interface provide many methods for transaction management like commit(),rollback() etc. By default, connection commits the changes after executing queries. **Some method of connection used here:**

|  |  |
| --- | --- |
| **public Statement createStatement()** | creates a statement object that can be used to execute SQL queries |
| **public Statement createStatement**  **(int resultSetType,**  **int resultSetConcurrency)** | creates a statement object that can be used to execute SQL queries |

Here in the code at line no. 18 we have used the second method which tells us that the   
 statement object’s resultset can traverse in both directions and will be in read only   
 state.

1. **Statement:** The **Statement interface** provides methods to execute queries with the database

|  |  |
| --- | --- |
| **public ResultSet executeQuery(String sql)** | is used to execute SELECT query. It returns the object of ResultSet. |

**Finding Service name/id of Oracle database and loading drivers**

**Load Drivers** To connect java application with the Oracle 12c database ojdbc7.jar file is required to be loaded.

There are two ways to do it

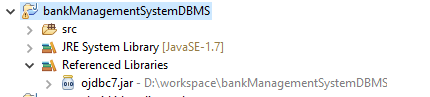
1. Load it on system
2. Integrate it within the app.

I have used second method to get rid of any system dependency.

To integrate driver within the package folder in eclipse, First right click in the package name->Build path->add external libraries.

Now select user libraries, click on import, browse the location and click ok.

If successful the .jar file will be included in reference library.



**Find Service Name/Id**

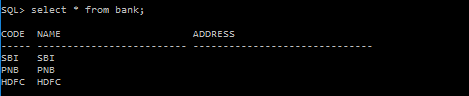
open **sqlplus.exe** on **cmd,** enter username and password and get connected then type **select** **instance\_name** **from v$instance**;

the retuned column will give the name of service id, use this in **jdbc:oracle:thin:@localhost:port:service\_name.**

**Relations of database-**

1. **Bank:**

**Select \* from bank;**



create table bank(code varchar(5),name varchar(25),address varchar(30),primary key(code)) ;

1. **Bank Branch:**

**Select \* from bank\_branch**;



create table bank\_branch(

address varchar(30) default NULL,

bkcode varchar(5) NOT NULL,

bbcode int NOT NULL,

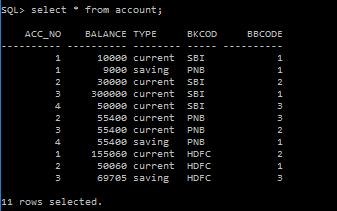
foreign key(bkcode) references bank(code),

primary key(bkcode,bbcode)

);

1. **Account**

**Select \* from account;**



create table account(

acc\_no int NOT NULL,

balance int default 0,

type varchar(8) NOT NULL,

constraint account\_type\_ck check(type in('saving','current')),

bkcode varchar(5) NOT NULL,

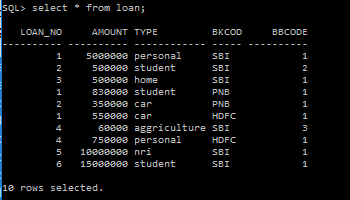
bbcode int,

foreign key(bkcode,bbcode) references bank\_branch(bkcode,bbcode),

primary key(acc\_no,bkcode)

);

1. **Loan**



create table loan(

loan\_no int NOT NULL,

amount int default 100,

type varchar(12) NOT NULL,

constraint loan\_type\_ck check(type in('personal', 'student','car','home','nri','aggriculture')),

bkcode varchar(5),

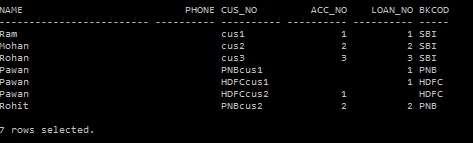
bbcode int,

foreign key(bkcode,bbcode) references bank\_branch(bkcode,bbcode),

primary key(loan\_no,bkcode)

);

1. **Customer**



create table customer(

name varchar(25),

phone decimal(10,0),

address varchar(30),

cus\_no varchar(10) NOT NULL,

acc\_no int ,

loan\_no int,

bkcode varchar(5) NOT NULL,

unique(acc\_no,bkcode),

unique(loan\_no,bkcode),

foreign key(acc\_no,bkcode) references account(acc\_no,bkcode),

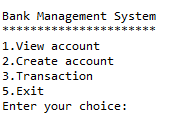
foreign key(loan\_no,bkcode) references loan(loan\_no,bkcode),

primary key(cus\_no,bkcode)

);

**Working screenshot of program with code**

1. Main page  
     
   code:  
   **package** bankManagementSystemDBMS;



**import** java.util.Scanner;

**public** **class** Start {

**public** **static** **void** main(String[] args) {

Scanner scn=**new** Scanner(System.***in***);

account acc=**new** account();

**int** choice=0;

**while**(choice!=4)

{

System.***out***.println("\n\nBank Management System" );

System.***out***.print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"+"\n1.View account\n2.Create account\n3.Transaction\n5.Exit\nEnter your choice:");

**try**{

choice=0;

choice=scn.nextInt();

}**catch**(java.util.InputMismatchException e)

{}

**switch**(choice)

{

**case** 1:

System.***out***.print("\tEnter acc\_no:(int)");

**int** acc\_no=scn.nextInt();

System.***out***.print("\tEnter bank code:(string)");

String bkcode =scn.next();

acc.view\_acc(acc\_no,bkcode.toUpperCase(),**true**);

**break**;

**case** 2: acc.new\_acc();

**break**;

**case** 3: System.***out***.print("\t1.Deposit Money\n\t2.Withdraw Money\n\t3.Tranct funds between accounts\n\tEnter Your choice:");

**try**{

choice=scn.nextInt();

}**catch**(java.util.InputMismatchException e)

{}

**switch**(choice)

{ **case** 1: //Deposit fund

System.***out***.println("\*\*Deposit Section\*\*");

System.***out***.print("\tEnter acc\_no:");

acc\_no = scn.nextInt();

System.***out***.print("\tEnter bank code:(string)");

bkcode = (scn.next()).toUpperCase();

**int** bal=acc.view\_acc(acc\_no,bkcode,**true**);

System.***out***.println("Current balance is: "+bal);

System.***out***.print("Enter ammount to deposit: ");

**int** input =scn.nextInt();

acc.deposit(bal,input,acc\_no,bkcode);

bal=acc.view\_acc(acc\_no,bkcode,**false**);

System.***out***.println("Updated balance is: "+bal);

**break**;

**case** 2: //withdraw funds

System.***out***.println("\*\*Withdraw counter\*\*");

System.***out***.print("\tEnter acc\_no:(int)");

acc\_no = scn.nextInt();

System.***out***.print("\tEnter bank code:(string)");

bkcode = (scn.next()).toUpperCase();

bal=acc.view\_acc(acc\_no,bkcode,**true**);

System.***out***.println("balance is: "+bal);

System.***out***.print("Enter ammount to withdraw: ");

input =scn.nextInt();

**try**{

acc.withdraw(bal,input,acc\_no,bkcode);

}**catch**(depositException e){System.***out***.println(e.getMessage());}

bal=acc.view\_acc(acc\_no,bkcode,**false**);

System.***out***.println("Updated balance is: "+bal);

**break**;

**case** 3: //Transact B/w accounts

acc.transact();

**break**;

}

**break**;

**case** 4:

account in =**new** account();

in.createTable();

**break**;

**case** 5: choice=4;

scn.close();

**default**: System.***out***.println("Wrong choice");

**break**;

}

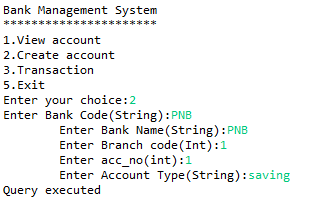
}

scn.close();

}

}

1. Create Account  
     
     
   Code:  
    **public** **void** new\_acc() {



**try**{

// **TODO** Auto-generated method stub

System.***out***.println("\*\*Create new account\*\*");

System.***out***.print("\tEnter Bank Code(String):");

String code=(scn.next()).toUpperCase();

System.***out***.print("\tEnter Bank Name(String):");

String name=(scn.next()).toUpperCase();

System.***out***.print("\tEnter Branch code(Int):");

**int** bbcode=scn.nextInt();

System.***out***.print("\tEnter acc\_no(int):");

**int** acc\_no=scn.nextInt();

System.***out***.print("\tEnter Account Type(String):");

String type=(scn.next()).toLowerCase();

String sql="insert into bank(code,name) values('"+code+"','"+name+"')";

String sql1="insert into bank\_branch(bkcode,bbcode)

values('"+code+"',"+bbcode+")";

String sql2="insert into account(acc\_no,type,bkcode,bbcode)

values("+acc\_no+",'"+type+"','"+code+"',"+bbcode+")";

run.execute(sql);

run.execute(sql1);

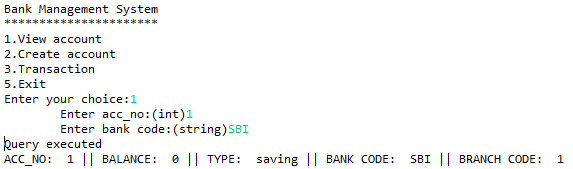
run.execute(sql2);

System.***out***.println("Query executed!");

}**catch**(java.util.InputMismatchException e){}

}

1. View account  
     
   Code:  
    **public** **int** view\_acc(**int** acc\_no, String bkcode,**boolean** state) {



**int** bal=0;

**try**{

String sql="select \* from account where acc\_no="+acc\_no+" and

bkcode='"+bkcode+"'";

ResultSet rs = run.execute(sql);

**if**(state==**true**)

**while**(rs.next())

{

System.***out***.println("ACC\_NO: " +rs.getInt(1)+" || BALANCE:

"+rs.getInt(2)+" || TYPE: "+rs.getString(3)+" || BANK CODE:

"+rs.getString(4)+" || BRANCH CODE: "+rs.getInt(5));

bal=rs.getInt(2);

}

**else**{

**while**(rs.next()){

bal=rs.getInt(2);

}

}

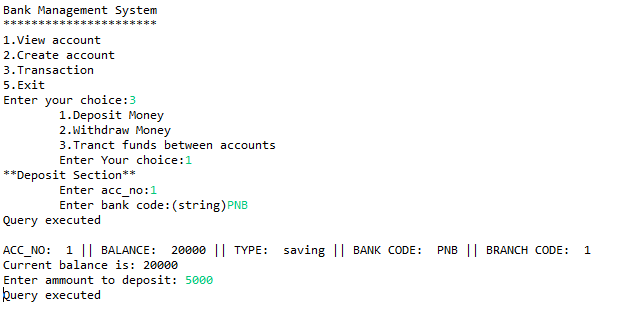
}**catch**(java.lang.NullPointerException e){ **this**.createTable(); }

**catch**(Exception e){e.printStackTrace();}

**return** bal;

}

1. Deposit  
     
   Code:  
      
    **public** **void** deposit(**int** bal,**int** input,**int** acc\_no,String bkcode){



**try**{

bal=bal+input;

String sql="update account set balance="+bal+" where acc\_no="+acc\_no+" and

bkcode='"+bkcode+"'";

run.execute(sql);

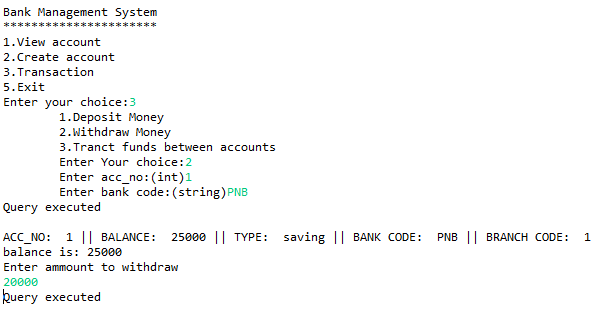
}**catch**(Exception e){

e.printStackTrace();

}

}

1. Withdraw



Code:

1. withdraw

**public** **void** withdraw(**int** bal,**int** input,**int** acc\_no, String bkcode) **throws**

depositException{

**if**(bal<input)

**throw** **new** depositException(bal);

**else**{

bal=bal-input;

String sql="update account set balance="+bal+" where acc\_no="+acc\_no+"

and bkcode='"+bkcode+"'";

run.execute(sql);

}  
  
B.depositException  
**package** bankManagementSystemDBMS;

**public** **class** depositException **extends** java.lang.Exception {

**int** bal=0;

**public** depositException()

{}

**public** depositException(**int** bal){

**this**.bal=bal;

}

**public** String getMessage()

{

String msg="Invalid!\nYou cannot withdraw money more

than"+"'"+bal+"'";

**return** msg;

}

}