Practical – 1

JDBC Programming

Q-1 Write down steps to establish database connection with Java Application. Explain each step in detail with syntax.

Ans-1 This are the following steps to establish database connection in the java application:-

1 Load the driver

2 Define the Connection URL

3 Establish the Connection

4 Create a Statement object

5 Execute a query

6 Process the results

7 Close the connection

1) Load the driver:-

* We can register the driver indirectly using the statement.
* Class.forName(“com.mysql.jdbc.Driver);
* Class.forName loads the specified class.
* When mysqlDriver is loaded, it automatically creates an instance of itself.
* Registers this instance with the DriverManager.
* Hence, the driver class can be given as an argument of the application.

2) Define the Connection <URL:->

* URL format is driver specific.
* Typically contains
  + Protocol and subprotocol
  + Hostname
  + Database Connection

3) Establish the Connection:-

* Once the requirement packages have been imported and the JDBC driver has been loaded and registered a database connection must be established. This is done by using a database connection must be established.
* This is done by using the getConnection() method of the DriverManager class.
* It contain three parameters.
* 1) <url:-> It contains the URL of your database.
* 2) user name:- It contain the user name of the root
* 3) password: - It contains the password of the database user. In our case user name = “root” and password =””.
* Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3308/emp\_master","root","");

4) Create a Statement object:-

* Before we use a statement object to execute the sql statement, we need to create one using the connection objects. It’s syntax are as bellow.
* Statement stmt = con.createStatement();
* In above example con is the connection object is required. It return the stmt object for the execute the sql query.
* Statement - Execute simple sql queries without parameters.
* Useful when you are using static SQL statements at runtime.
* Statement createStatement ()
* Statement statement = dbConnection.createStatement ();
* Prepared Statement - Execute precompiled sql queries with or without parameters.
* It can accept input parameters at runtime.  
  PreparedStatement prepareStatement (String sql)
* PreparedStatement objects are precompiled SQL statements.
* Callable Statement - Execute a call to a database stored procedure.
* It can accept runtime input parameters.  
  CallableStatement prepareCall (String sql)
* CallableStatement objects are SQL stored procedure call statements.

5) Execute a query:-

* Three methods for executing statements:
* All 3 statements can return ResultSet Object.
* ResultSet executeQuery(String SQL) - For a SELECT statement
* int executeUpdate(String SQL) - For statements that create or modify tables(Insert, Update, Delete, Alter, Drop)
* boolean execute(String SQL) - executes an SQL statement that is written as String object.
* Returns a boolean value of true if a ResultSet object can be retrieved; otherwise, it returns false.
* Use this method to execute SQL DDL statements or when you need to use truly dynamic SQL.

6) Process the results:-

* ResultSet provides access to a table of data generated by executing a Statement.
* The table rows are retrieved in sequence.
* A ResultSet maintains a cursor pointing to its current row of data.
* The next () method is used to successively step through the rows of the tabular results.
* By default, only one ResultSet object per Statement object can be open at the same time.
* All method throws SQLException
* Navigating a Result Set:
* void beforeFirst()
* void afterLast()
* boolean first()
* void last()
* boolean previous()
* boolean next()
* int getRow()
* Viewing a Result Set:
* int getInt(String columnName)
* int getInt(int columnIndex)
* void updateString(int columnIndex, String s)
* void updateString(String columnName, String s)
* void updateRow()
* void deleteRow()
* void insertRow()

7) Close the connection:-

* In the step the connection of the database are close. So the database is not used in the future for the database related task into it.
* It has only one method that is used close the connection. This close () the connection of the object is required.
* connection.close();

Q-2 Create a MySql database named “Employee\_Master” consisting of table

“emp\_detail”. Write a java program with following operations that can be

performed by the user.

*emp\_detail[emp\_id(PK), emp\_fname, emp\_mname, emp\_lname,*

*emp\_code, emp\_address, emp\_pincode, emp\_phone, emp\_gender]*

• View all records.

• View specific record based on primary key.

• View specific employee detail based on emp\_code filter.

• Update specific record based on primary key.

• Delete specific record based on primary key.

• Update specific record based on the field emp\_code.

• Delete specific record based on the field emp\_code.

import java.util.\*;

import java.sql.\*;

public class Employee{

public static void main(String args[]) throws Exception{

Scanner sc = new Scanner(System.in);

Class.forName("com.mysql.jdbc.Driver");

Connection con;

con=DriverManager.getConnection("jdbc:mysql://localhost:3308/emp\_master","root","");

Statement stmt = con.createStatement();

ResultSet rs;

System.out.println("SELECT OPTIONS : ");

System.out.println(" 1. View all records. ");

System.out.println(" 2. View specific record based on primary key. ");

System.out.println(" 3. View specific employee detail based on emp\_code filter. ");

System.out.println(" 4. Update specific record based on primary key. ");

System.out.println(" 5. Delete specific record based on primary key. ");

System.out.println(" 6. Update specific record based on the field emp\_code. ");

System.out.println(" 7. Delete specific record based on the field emp\_code. ");

System.out.println(" 8. Exit. ");

System.out.print(" Enter Your Choice : ");

int ch =sc.nextInt();

switch(ch){

case 1:

rs = stmt.executeQuery("select \* from emp\_details");

while(rs.next()){

System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4) + "\t" + rs.getString(5) + "\t" + rs.getString(6) + "\t" +rs.getString(7) + "\t" + rs.getString(8) + "\t" + rs.getString(9));

}

break;

case 2 :

System.out.print("Enter Id : ");

int id = sc.nextInt();

rs = stmt.executeQuery("select \* from emp\_details where emp\_id = " + id);

while(rs.next()){

System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4) + "\t" + rs.getString(5) + "\t" + rs.getString(6) + "\t" +rs.getString(7) + "\t" + rs.getString(8) + "\t" + rs.getString(9));

}

break;

case 3 :

System.out.print("Enter Emp\_Code : ");

int emp\_code = sc.nextInt();

rs = stmt.executeQuery("select \* from emp\_details where emp\_code = " + emp\_code);

while(rs.next()){

System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4) + "\t" + rs.getString(5) + "\t" + rs.getString(6) + "\t" +rs.getString(7) + "\t" + rs.getString(8) + "\t" + rs.getString(9));

}

break;

case 4 :

System.out.print("Enter id : ");

int eid = sc.nextInt();

System.out.print("Enter Employee First Name : ");

String fname = sc.next();

stmt.executeUpdate("update emp\_details set emp\_fname = '"+ fname + "' where emp\_id = " +eid);

System.out.println("Update Sucessfull.");

break;

case 5:

System.out.print("Enter id : ");

int e\_id = sc.nextInt();

stmt.executeUpdate("delete from emp\_details where emp\_id = " +e\_id);

System.out.println("Delete Sucessfull.");

break;

case 6:

System.out.print("Enter Code : ");

int ec\_id = sc.nextInt();

System.out.println("Enter Employee First Name : ");

String fname1 = sc.next();

stmt.executeUpdate("update emp\_details set emp\_fname = '"+ fname1 + "' where emp\_code = " +ec\_id);

System.out.println("Update Sucessfull.");

break;

case 7:

System.out.print("Enter Code : ");

int ec\_id1 = sc.nextInt();

stmt.executeUpdate("delete from emp\_details where emp\_code = " +ec\_id1);

System.out.println("Delete Sucessfull.");

break;

default:

System.out.println("Wrong Choice : ");

}

}

}

O/P:-







Q-3)

Create a table named “emp\_salary\_detail”inthe“Employee\_Master”

database. Write a java program with following operations that can be

performed by the user.

*emp\_salary\_detail[emp\_sal\_id(PK), emp\_id(FK), emp\_salary\_gross,*

*emp\_salary\_other]*

• View all employee records with their salary.

• View specific employee detail based on emp\_code filter with their

salary details.

import java.util.\*;

import java.sql.\*;

public class Employee\_Salary{

public static void main(String args[])throws Exception{

Scanner sc = new Scanner(System.in);

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection

("jdbc:mysql://localhost:3308/emp\_master","root","");

Statement stmt = con.createStatement();

ResultSet rs;

System.out.println("1.View All Employee With Salary ");

System.out.println("2.View specific employee detail based on emp\_code filter with their salary details");

System.out.print("Enter Your Choice : ");

int ch = sc.nextInt();

switch(ch){

case 1:

rs = stmt.executeQuery("select \* from emp\_details natural join emp\_salary\_details");

while(rs.next()){

System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4) + "\t" + rs.getString(5) + "\t" + rs.getString(6) + "\t" +rs.getString(7) + "\t" + rs.getString(8) + "\t" + rs.getString(9)+ "\t" + rs.getString(10)+ "\t" + rs.getString(11)+ "\t" + rs.getString(12));

}

break;

case 2:

rs = stmt.executeQuery("select \* from emp\_details emp inner join emp\_salary\_details empsal on empsal.emp\_id = emp.emp\_id order by empsal.emp\_salary\_gross");

while(rs.next()){

System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4) + "\t" + rs.getString(5) + "\t" + rs.getString(6) + "\t" +rs.getString(7) + "\t" + rs.getString(8) + "\t" + rs.getString(9)+ "\t" + rs.getString(10)+ "\t" + rs.getString(11)+ "\t" + rs.getString(12));

}

break;

}

}

}

O/P:-

