Database Management Systems

Year 2, Semester 1

Lab Sheet 06

This practical provides an example of how to create a simple JDBC application. This will show you how to open a database connection, execute a SQL query, and display the results.

Creating JDBC Application

There are following six steps involved in building a JDBC application

- **Import the packages**: Import the packages containing the JDBC classes needed for database programming. Most often, using *import java.sql.** will suffice.
- **Load the driverRegister the JDBC driver**: Initialize a driver in order to open a communication channel with the database.
- **Open a connection**: Create a Connection object, which represents a physical connection with the database.
- **Execute a query**: Using an object of type Statement for building and submitting an SQL statement to the database.
- **Extract data from result set**: retrieve the data from the result set.
- **Clean up the environment**: Closing all database resources versus relying on the JVM's garbage collection.

Step 1: Import the packages

Source History

```
1
2   package lecture06;
3
4   import java.sql.*;
5
6   public class Practical06 {
7       public static void main(String[] args) {
8
9       }
10  }
11
```

Step 2: Registering database drivers

```
 *
 * @author Sanjeevi
 */
import java.sql.*;
public class Practical06 {
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        }
        catch(Exception ex)
        {
            System.out.println(ex);
        }
    }
}
```

Step 3: Open a connection

```
 *
 * @author Sanjeevi
 */
import java.sql.*;
public class Practical06 {
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String url="jdbc:sqlserver://localhost:1433;databaseName=BMS";
            Connection con=DriverManager.getConnection(url,"sa","sql2014@@@");
        }
        catch(Exception ex)
        {
            System.out.println(ex);
        }
    }
}
```

Host Name

TCP Port No

Database Name

SQL UserName

SQL Password

Step 4: Executing queries

- Once a connection is obtained we can interact with the database. The JDBC *Statement*, *CallableStatement*, and *PreparedStatement* interfaces define the methods and properties that enable you to send SQL commands and receive data from your database.
- A summary of statements is available below :

Interface	Recommended use
Statement	Useful when static SQL statements are used at runtime. The Statement interface cannot accept parameters.
PreparedStatement	Suitable when SQL statements are used many times. The PreparedStatement interface accepts input parameters at runtime
CallableStatement	Used to access the database stored procedures. The CallableStatement interface can also accept runtime input parameters.

4.1 Insert a new record to Account table (1 tuple)

```
Source  History
import java.sql.*;
public class Practical06 {
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String url="jdbc:sqlserver://localhost:1433;databaseName=BMS";
            Connection con=DriverManager.getConnection(url,"sa","sql2014@@@");

            Statement st=con.createStatement();
            st.executeUpdate("insert into Account values (55555,350000, 1, 22, 111)");
        }
        catch(Exception ex)
        {
            System.out.println(ex);
        }
    }
}
```

Exercise 01

Now try to insert a new record to Customer table.

Exercise 02

Now try to insert multiple records to the Customer table (number of tuples more than 1) at a single execution.

4.2 Update the account balance in Account table (accNo =56389)

```
Source  History
13          String url="jdbc:sqlserver://localhost:1433;databaseName=BMS";
14          Connection con=DriverManager.getConnection(url,"sa","sql2014@@@");
15
16          Statement st=con.createStatement();
17          st.executeUpdate("insert into Account values(55555,240000, 1, 22, 111)");
18
19          PreparedStatement ps1 =con.prepareStatement("update Account set balance=? where accNo = ?");
20          ps1.setFloat(1,140000 );
21          ps1.setInt(2, 56389);
22          ps1.executeUpdate();
23      }
        catch(Exception ex)
25      {
26          System.out.println(ex);
27      }
28  }
```

Exercise 03

Now try to update the balance in the Account table (accNo = 50238 and new balance as 600000)

Exercise 04

Now try to update the address in the Customer table (custNo = 6 and new address as 100/2, New Kandy Rd, Malabe)

4.3 Calling a stored procedure (written in sql server) through java application

Fund transfer between the following Accounts (withdrawal amount 3500000)

Source Account No: 34901

Source Balance: 9500000

New Source Balance: 6000000

Destination Account No: 34890

Destination Balance: 5500000

New Destination Balance: 9000000

Stored Procedure to transfer funds between two accounts

```
create procedure fundTransfer(@sAccNO int,
@sBalance float,@dAccNO int,@dBalance float, @amount float)
AS
BEGIN

    UPDATE Account
    SET balance =@sBalance-@amount
    WHERE accNo=@sAccNO

    UPDATE Account
    SET balance =@dBalance+@amount
    WHERE accNo=@dAccNO

END;
```

Calling a procedure within a java program

```
Source  History

        CallableStatement cs=con.prepareCall("exec fundTransfer ?,?,?,?,?");

        cs.setInt(1,34901);
        cs.setFloat(2,9500000);
        cs.setInt(3,34890);
        cs.setFloat(4,5500000);
        cs.setFloat(5,3500000);

        cs.execute();
```

Exercise 05

Now try to create procedure to give an annual interest for all the account balances by a given percentage from their existing account balance

Exercise 06

Now try create a procedure that outputs the total account balance for a given branch number

Step 5: Extract data from the result set

Display the account number and balance for Accounts which contains a minimum balance of 1000000 or more.

```
Source  History
24
25              Statement st1=con.createStatement();
26              ResultSet rs=st1.executeQuery("Select accNo,balance from Account where balance >1000000");
27              while(rs.next())
28              {
29                  int no=rs.getInt(1);
30                  float b=rs.getFloat(2);
31                  System.out.println(no+"\t"+b);
32              }
33          }
```

Exercise 07

Now try to display the name and the NIC of all the Customers.

Exercise 8

Now try to display the money (amount) credited to account number 51678 via cheques

Step 6: Clean up the environment

The connection should be closed later using con.close ()

```
Source  History
24
25              Statement st1=con.createStatement();
26              ResultSet rs=st1.executeQuery("Select accNo,balance from Account where balance >1000000");
27              while(rs.next())
28              {
29                  int no=rs.getInt(1);
30                  float b=rs.getFloat(2);
31                  System.out.println(no+"\t"+b);
32              }
33              con.close();
34          }
            catch(Exception ex)
36          {
37              System.out.println(ex);
38          }
39      }
40  }
41
```