Department of Software Engineering
Mehran University of Engineering and Technology, Jamshoro

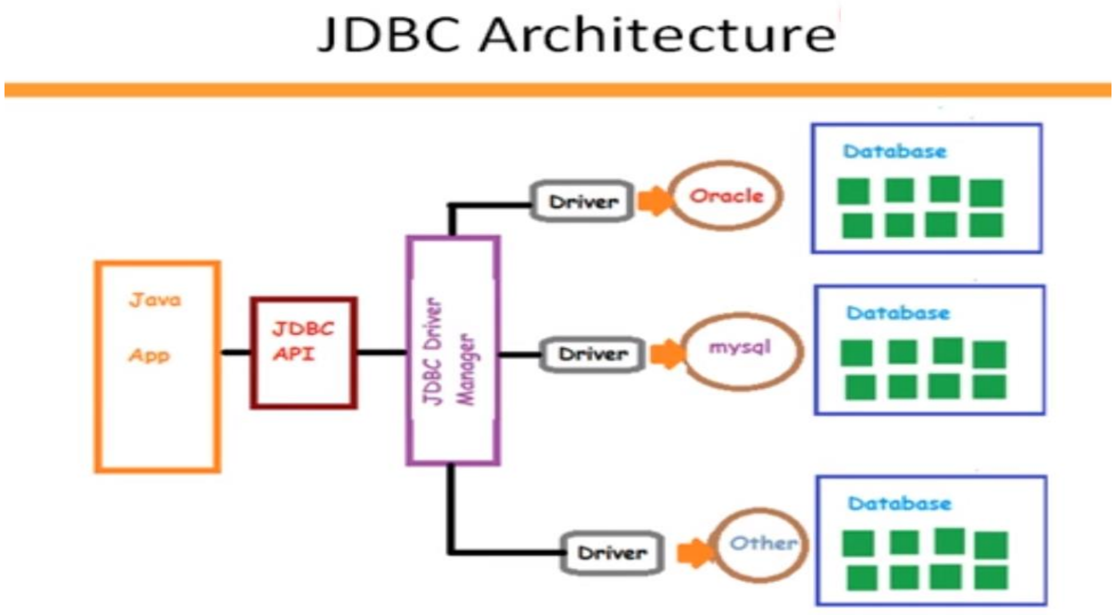
Course: SW222 – Database Management & Administration			
Instructor	Ms Shafiya Qadeer Memon	**Practical/Lab No.**	05
Date	10/11-02-2021	**CLOs**	2
Signature		**Assessment Score**	2 Marks

Topic	**To become familiar with use of group functions and logical operators**
Objectives	- To become familiar with JDBC CONNECTIVITY

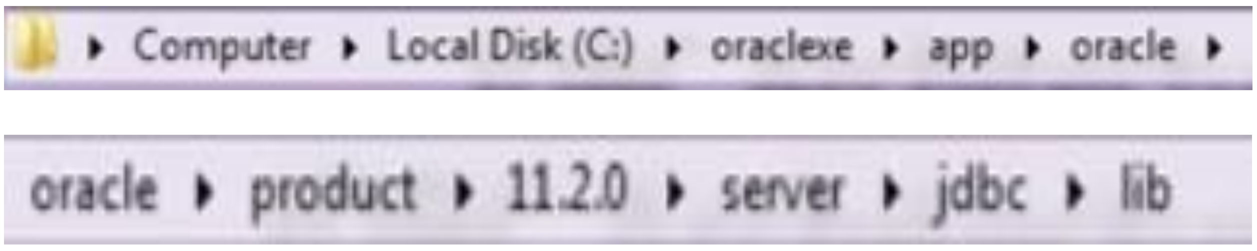
Lab Discussion: Theoretical concepts and Procedural steps

Java JDBC :

- JDBC stands for Java Database Connectivity.
- JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition).
- JDBC API uses JDBC drivers to connect with the database.



JDBC File Location in Oracle


Computer ▸ Local Disk (C:) ▸ oraclexe ▸ app ▸ oracle ▸

oracle ▸ product ▸ 11.2.0 ▸ server ▸ jdbc ▸ lib

JDBC Driver

- JDBC Driver is a software component that enables java application to interact with the database.

Java Database Connectivity with Oracle

To connect java application with the oracle database, we need following information:

- Driver class: The driver class for the oracle database is "oracle.jdbc.driver.OracleDriver."
- Connection URL
- Username
- Password

Java Database Connectivity with Oracle

- Connection URL: The connection URL for the oracle database is "jdbc:oracle:thin:@localhost:1521:orcl" where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and orcl is the Oracle service name.

Thin Driver

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

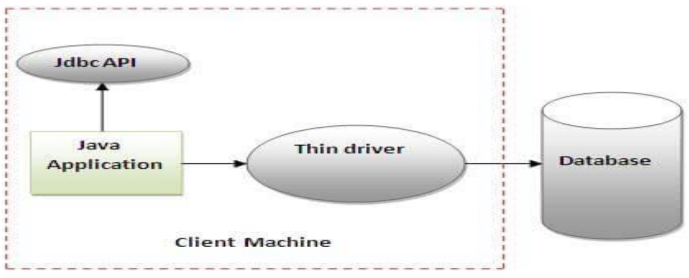


Figure- Thin Driver

Java Database Connectivity with 5 Steps

- There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:
  - Register the Driver class
  - Create connection
  - Create statement
  - Execute queries
  - Close connection

Java Database Connectivity with 5 Steps

- Register the Driver class

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

public static void forName(String className)**throws** ClassNotFoundException

- Create connection

The **getConnection()** method of DriverManager class is used to establish connection with the database.

1) public static Connection getConnection(String url)throws SQLException
2) public static Connection getConnection(String url,String name,String password)
throws SQLException

- **Create statement**

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.
public Statement createStatement()throws SQLException

- **Execute queries**

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.

public ResultSet executeQuery(String sql)throws SQLException
public int **executeUpdate**(String sql)

- **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.
public void close()throws SQLException

<u>CODING</u>

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class ConnToOracle {

public static void main(String[] args) {

try {

//step1 load the driver class
Class.forName("oracle.jdbc.driver.OracleDriver");

//step2 create  the connection object
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:orcl","scott","scott");
//step3 create the statement object
Statement stmt=con.createStatement();
//step4 execute query
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
//step5 close the connection object
con.close();
} catch (ClassNotFoundException ex) {

ex.printStackTrace();

} catch (SQLException ex) {

ex.printStackTrace();

}
}
}
```

COMPILE YOUR CODE

Before you execute your code

To load the jar file

1. paste the .jar file in jre/lib/ext folder (optional)
2. set classpath: There are two ways to set the classpath:
    - Temporary
    - permanent

Temprory:

1. Go to cmd execute this command

    ```
    set classpath=;C:\path_where_jar_file_is_placed
    ```

    Example:

    ```
    set classpath=;C:\Program Files\Java\jre1.8.0_73\lib\ext\jdbc-oracle.jar
    ```

2. Execute your code

Permanent:

1. Go to environment variables
2. Create a new variable:

name =classpath

value =;C:\Program Files\Java\jre1.8.0_73\lib\ext\jdbc-oracle.jar

2. Execute your code