**Jdbc**

**Class 01**

**JDBC**

**=====**

**As if now it is known as trademark.**

**But earlier it is also known as Java Database Connectivity.**

**RAM is a temporary storage device.**

**During the program execution our data will store in a RAM.**

**Once the program execution is completed we will loss the data.**

**To overcome this limitation we are making our application writing the data in a file or database software.**

**Files and Database software act like a permanent storage device or medium.**

**Persistence**

**===========**

**The process of storing and managing the data for a long period of time is called persistence.**

**Important terminologies**

**=======================**

**1) Persistence store**

**-------------------**

**It is a place where we can store and manage the data for a long period of time.**

**ex:**

**File**

**Database S/W**

**and etc.**

**2) Persistence data**

**------------------**

**Data of a persistence store is called persistence data.**

**ex:**

**Records**

**Tables**

**and etc.**

**3) Persistence operation**

**-----------------------**

**Insert, update ,delete ,create and select are called persistence operations.**

**In realtime this operation is also known as CURD , CRUD or SCUD operation.**

**ex:**

**C - create S - select**

**U - update C - create**

**R - read U - update**

**D - delete D - delete**

**4) Persistence logic**

**-------------------**

**A logic which is capable to perform persistence operations is called persistence logic.**

**ex:**

**JDBC Code**

**Hibernate code**

**IOStream**

**5) Persistence technology**

**------------------------**

**A technology which is used to develop persistence logics is called persistence technology.**

**ex:**

**JDBC**

**Hibernate**

**IBatis**

**JPA**

**and etc.**

**Q) What is JDBC ?**

**JDBC is a persistence technology which is used to develop persistence logics having the capability to perform persistence operations on persistence data of a persistence store.**

**Limitations with File as a persistence store**

**=============================================**

**> We can store limited amount of data.**

**> There is no security.**

**> Fetching the data with multiple conditions is not possible.**

**> It does not allow us to apply constraints.**

**> It does not allow us to show an application with relationship.**

**> Updation and Deletion of data can't be done directly.**

**> Merging and comparision of data can't be done easily.**

**Advantages of database as a persistence store**

**==============================================**

**> We can store unlimited amount of data.**

**> There is a security.**

**> It supports common query language.**

**> Fetching the data with multiple condition is possible.**

**> It allows us to apply constraints.**

**> It shows an application with relationship.**

**> Updation and deletion of data can be done directly.**

**> Merging and comparision of data can be done easily.**

**Note:**

**-----**

**IOStream**

**JavaApp ----------------------------------------------- File**

**Serialization/Deserialization**

**JDBC Code**

**JavaApp ----------------------------------------------- Database S/W**

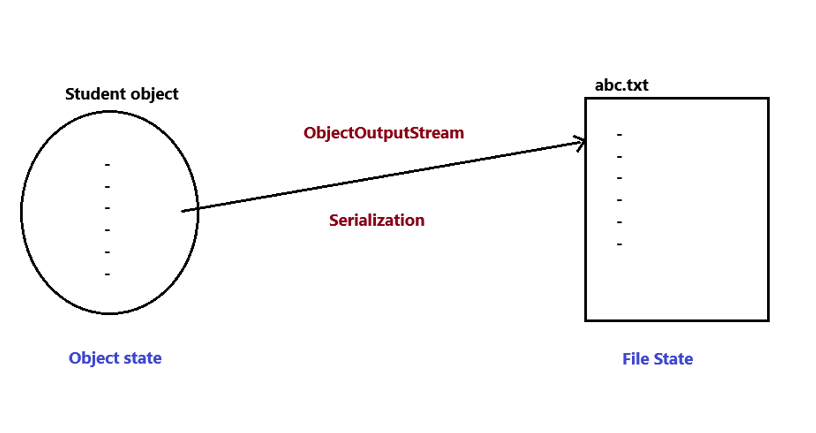
**Serialization**

**--------------**

**The process of taking object data storing in a file is called serialization.**

**In general, The process of converting object state to file state is called serialization.**

**Diagram: jdbc1.1**

****

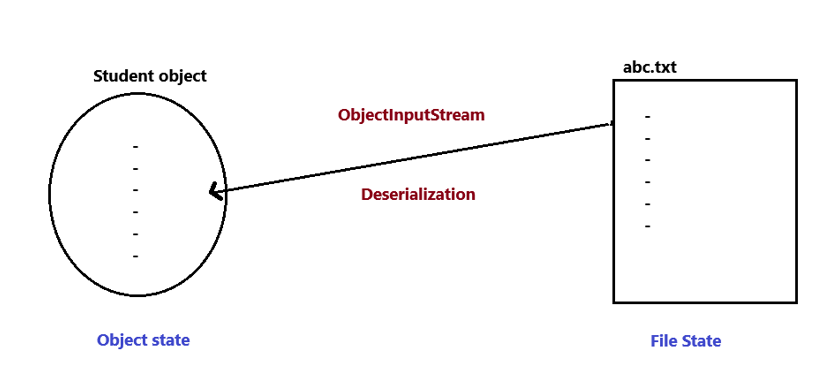
**Deserialization**

**-----------------**

**The process of take the data from file and representing an object is called deserialization.**

**In general, The process of converting file state to object state is called deserialization.**

**Diagram: jdbc1.2**

****

**Serializable Marker interface**

**=============================**

**It is an interface which is present in java.io package.**

**A class that implements this interface can be serialized, meaning its objects can be converted into a byte stream for storage or transmission.**

**ex:**

**import java.io.\*;**

**class Person implements Serializable**

**{**

**private String name;**

**private int age;**

**public Person(String name, int age)**

**{**

**this.name = name;**

**this.age = age;**

**}**

**public String toString()**

**{**

**return "Person: {name='" + name + "', age=" + age + "}";**

**}**

**}**

**public class Test**

**{**

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

**{**

**Person person = new Person("Alan", 30);**

**// Serialize the object**

**//try with resource**

**try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("suman.txt")))**

**{**

**out.writeObject(person);**

**System.out.println("Object serialized: " + person);**

**}**

**catch (IOException e)**

**{**

**e.printStackTrace();**

**}**

**// Deserialize the object**

**// try with resources**

**try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("suman.txt")))**

**{**

**Person deserializedPerson = (Person) in.readObject();**

**System.out.println("Object deserialized: " + deserializedPerson);**

**}**

**catch (IOException | ClassNotFoundException e)**

**{**

**e.printStackTrace();**

**}**

**}**

**}**

**Class 02**

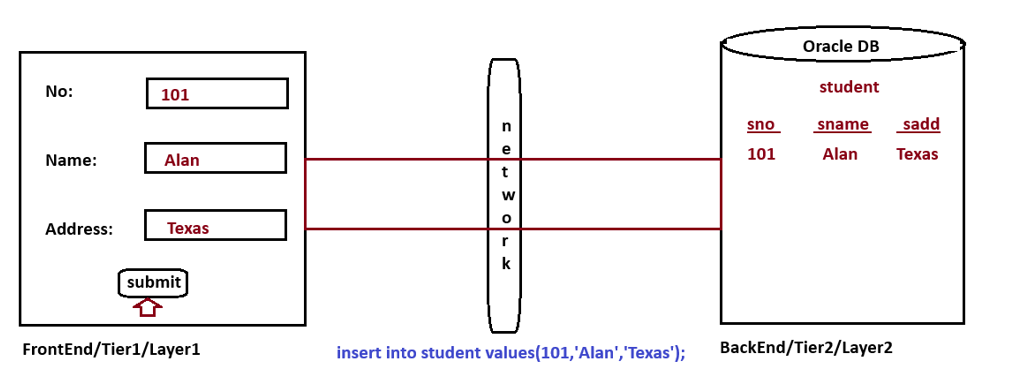
**Every JDBC application is a two-tier application. Java with JDBC code acts like a frontend/tier1/layer1 and database software acts like a backend/tier2/layer2.**

**The one which is visible to the enduser to perform some operations is called frontend.**

**The one which is not visible to the enduser but it performs operations based on the instructions given by frontend is called backend.**

**Enduser is a non-technical person. He can't prepare and execute SQL query in database software. He depends upon frontend developers having the capability to do that work for them.**

**Diagram: jdbc2.1**

****

**JDBC Driver**

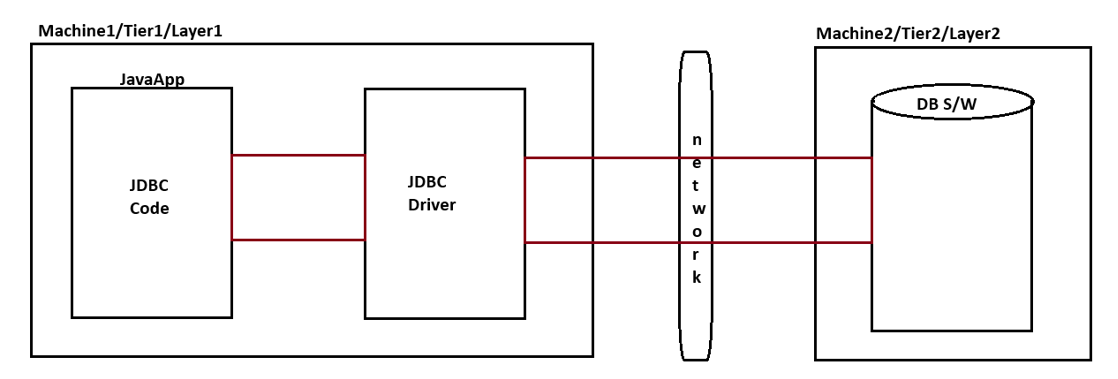
**============**

**JDBC Driver acts like a bridge between java application and database software.**

**It is used to convert java calls to database calls and vice versa.**

**Here calls means instructions.**

**Diagram: jdbc2.2**

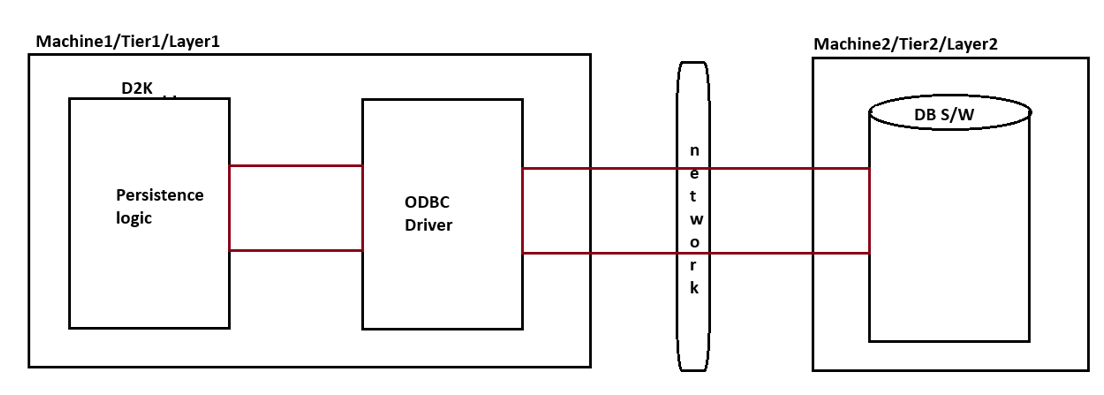
****

**ODBC Driver**

**===========**

**VBScript,D2K,Perl and etc uses ODBC driver to locate and interact with database software.**

**Diagram: jdbc2.3**

****

**ODBC drivers developed in C language by taking the support of pointers and Java does not support pointers.To overcome this limitation Sun Micro System introduced JDBC Drivers exclusively.**

**We will get JDBC softwares from following parties.**

**1) Sun Micro System (creator of JDBC Driver)**

**2) Database vendor**

**3) Third party vendor**

**We will get ODBC softwares from following parties.**

**1) Xopen company (creator of ODBC driver)**

**2) Database vendor**

**3) Third party vendor**

**Q) What is JDBC?**

**JDBC is a open technology given by Sun Micro System having set of rules and guidelines to develop JDBC drivers to interact with multiple database softwares.**

**q) What is ODBC?**

**ODBC is a open technology given by Xopen company having set of rules and guidelines to develop ODBC drivers to interact with multiple database softwares.**

**Q) How many drivers are there in JDBC?**

**We have four types of JDBC drivers.**

**1) Type1 JDBC Driver / JDBC-ODBC Bridge Driver**

**2) Type2 JDBC Driver / Native API Driver**

**3) Type3 JDBC Driver / Net Protocol Driver**

**4) Type4 JDBC Driver / Native Protocol Driver**

**To use any JDBC driver we need to register with DriverManager service.**

**Every JDBC application contains one built in service called DriverManager service.**

**Class.forName()**

**================**

**It is always recommanded to use Class.forName() method to register JDBC Driver with DriverManager service.**

**A Class.forName() method is used to load driver class but it won't create an object.**

**ex:**

**Class.forName("Driver-Class-Name");**

**Connection object**

**=================**

**Connection is an interface which is present in java.sql package.**

**It is an object of underlying supplied java class which implements java.sql.Connection interface.**

**To interact with database software we need to establish the connection with database software.**

**syntax:**

**Connection con;**

**DriverManager.getConnection()**

**=============================**

**DriverManager is a class which is present in java.sql package.**

**A DriverManager.getConnection() is used to interact with database software and gets one JDBC connection object representing connectivity between java application and database software.**

**ex:**

**Connection con=DriverManager.getConnection("driver-url");**

**Statement object**

**===============**

**Statement is an interface which is present in java.sql package.**

**It is an object of underlying supplied java class which implements java.sql.Statement interface.**

**It acts like a vehicle between java application and database software.**

**It is used to sends and executes SQL query in database software.**

**ex:**

**Statement st=con.createStatement();**

**ResultSet object**

**===================**

**ResultSet is an interface which is present in java.sql package.**

**It contains two positions.**

**1) BFR (Before First Record/Row)**

**2) ALR (After Last Record/Row)**

**By default record pointer points BFR position.**

**rs.next()**

**=========**

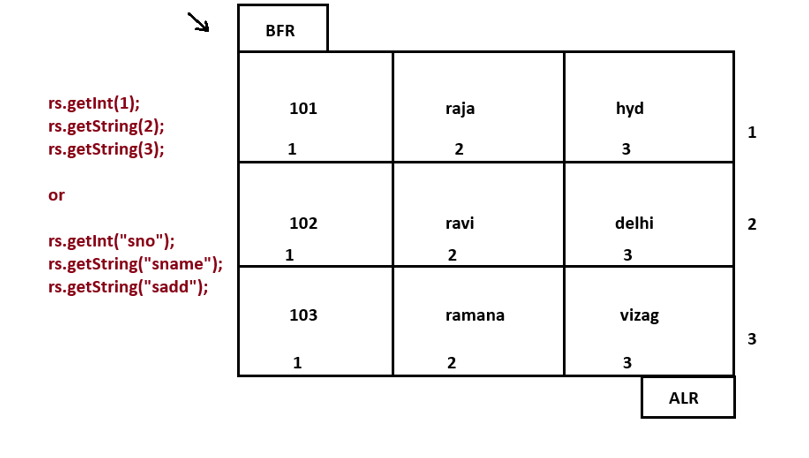
**It is used to move record pointer to next position from current position. If next position is a record then it will return. If next position is ALR then it will return false.**

**Every record ResultSet having 1 as base index and every column of record ResultSet having 1 as base index.**

**We can read the values from record ResultSet by using getXxx() methods with index numbers or column name.**

**Here getXxx() method means getInt(), getString(), getFloat(), getDouble() and etc.**

**Diagram: jdbc2.4**

****

**Types of queries in JDBC**

**==========================**

**According to JDBC point of view we have two types of queries.**

**1) Select query**

**2) Non-Select query**

**1) Select query**

**------------------**

**A select query returns bunch of records from database.**

**ex:**

**select \* from student;**

**JDBC Statement object gave executeQuery() method to execute selecte query.**

**ex:**

**ResultSet rs=st.exectueQuery("select \* from student");**

**2) Non-Select query**

**--------------------**

**A non-select query returns numeric value representing number of records effecting in a database table.**

**ex:**

**delete from student;**

**JDBC Statement object gave executeUpdate() method to execute non-select query.**

**ex:**

**int result=st.executeUpdate("delete from student");**

**Type4 JDBC Driver**

**===================**

**package-name**

**------------------**

**|**

**1) Driver class : oracle.jdbc.driver.OracleDriver**

**-----------**

**|**

**driver-classname**

**logical database name**

**|**

**2) Driver url : jdbc:oracle:thin:@localhost:1521:XE**

**------------------ | |**

**sub-protocol hostname portno**

**3) Database username : system**

**4) Database password : admin**

**Class 03**

**Steps to develop JDBC Application**

**==================================**

**There are six steps to develop JDBC Application.**

**1) Register JDBC driver with DriverManager service.**

**2) Establish the connection with database software.**

**3) Create Statement object.**

**4) Sends and executes SQL query in database software.**

**5) Gather the result from database to process the result.**

**6) Close all JDBC connection objects.**

**Eclipse**

**=========**

**IDE : JEE**

**Environment : Java**

**Eclipse Flavours: Kepler, Indigo, Mars, Luna and etc.**

**Vendor : Eclipse Foundation**

**Website : www.eclipse.org**

**Download link :**

**https://drive.google.com/file/d/1c8TAX048EjAubIFByqZ0DzWZI3oKuauR/view?usp=drive\_link**

**Steps to develop first JDBC application to select the records from student table using Eclipse IDE**

**==================================================================================================**

**step1:**

**------**

**create a student table and insert some records.**

**ex:**

**drop table student;**

**create table student(sno number(3),sname varchar2(10),sadd varchar2(12));**

**insert into student values(101,'raja','hyd');**

**insert into student values(102,'ravi','delhi');**

**insert into student values(103,'ramana','vizag');**

**commit;**

**step2:**

**------**

**Launch eclipse IDE by choosing workspace location.**

**step3:**

**------**

**Create a java project i.e My-Project inside eclipse ide.**

**ex:**

**File --> new ---> Project --> Java project --> Next -->**

**project Name : My-Project --> Next ---> Finish.**

**step4:**

**-----**

**Add "ojdbc14.jar" file in project build path.**

**ex:**

**right click to My-Project --> build path --> configure build path -->**

**libraries --> classpath --> Add external jars --> select ojdbc14.jar file**

**--> open --> apply and close.**

**step5:**

**-----**

**create a "com.ihub.www" package inside "src" folder.**

**ex:**

**right click to src folder --> new --> package --> Name: com.ihub.www --> finish.**

**step6:**

**------**

**Create a "SelectApp.java" file inside "src/com.ihub.www" package.**

**ex:**

**right click to com.ihub.www package --> new --> class -->**

**Name : SelectApp --> finish.**

**SelectApp.java**

**---------------**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**public class SelectApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**ResultSet rs=st.executeQuery("select \* from student");**

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

**{**

**System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**step7:**

**-----**

**Run jdbc application.**

**ex:**

**right click to SelectApp.java --> run as --> java application.**

**Q) Write a jdbc application to select student name and student address based on student number?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**import java.util.Scanner;**

**public class SelectApp2**

**{**

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

**{**

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

**System.out.println("Enter the student no :");**

**int no=sc.nextInt();**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="select sname,sadd from student where sno="+no;**

**ResultSet rs=st.executeQuery(qry);**

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

**{**

**System.out.println(rs.getString(1)+" "+rs.getString(2));**

**}**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**Non-select queries**

**=================**

**Q) Write a jdbc application to insert a record into student table?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.Statement;**

**import java.util.Scanner;**

**public class InsertApp**

**{**

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

**{**

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

**System.out.println("Enter the student no :");**

**int no=sc.nextInt();**

**System.out.println("Enter the student name :");**

**String name=sc.next();**

**System.out.println("Enter the student address :");**

**String add=sc.next();**

**//converting input values according to SQL query**

**name="'"+name+"'";**

**add="'"+add+"'";**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="insert into student values("+no+","+name+","+add+")";**

**int result=st.executeUpdate(qry);**

**if(result==0)**

**System.out.println("No Record Inserted");**

**else**

**System.out.println(result+" Record Inserted");**

**st.close();**

**con.close();**

**}**

**}**

**Q) Write a jdbc application to update student name based on student number?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.Statement;**

**import java.util.Scanner;**

**public class UpdateApp**

**{**

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

**{**

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

**System.out.println("Enter the student no :");**

**int no=sc.nextInt();**

**System.out.println("Enter the student name :");**

**String name=sc.next();**

**//converting input values according to SQL query**

**name="'"+name+"'";**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="update student set sname="+name+" where sno="+no;**

**int result=st.executeUpdate(qry);**

**if(result==0)**

**System.out.println("No Record Updated");**

**else**

**System.out.println(result+" Record Updated");**

**st.close();**

**con.close();**

**}**

**}**

**Q) Write a jdbc application to delete student record based on student number?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.Statement;**

**import java.util.Scanner;**

**public class DeleteApp**

**{**

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

**{**

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

**System.out.println("Enter the student no :");**

**int no=sc.nextInt();**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="delete from student where sno="+no;**

**int result=st.executeUpdate(qry);**

**if(result==0)**

**System.out.println("No Record Deleted");**

**else**

**System.out.println(result+" Record Deleted");**

**st.close();**

**con.close();**

**}**

**}**

**Class 04**

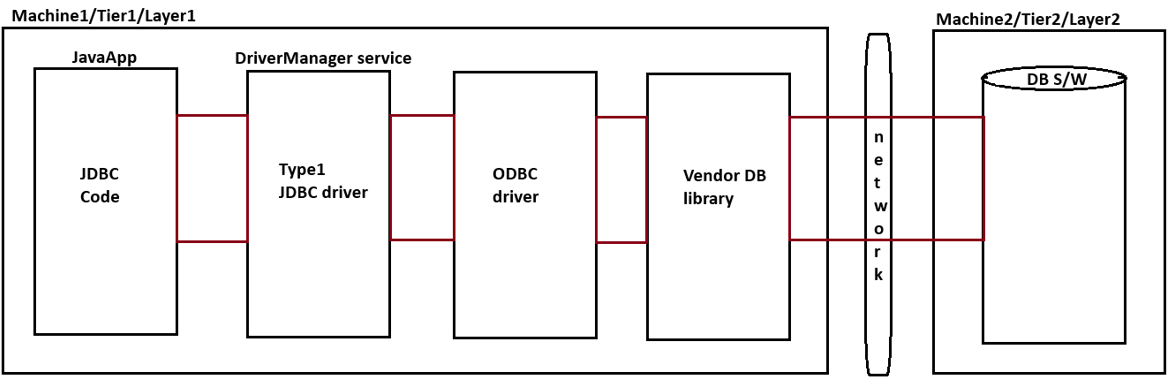
**Type1 JDBC Driver Architecture / JDBC-ODBC Bridge Driver**

**=========================================================**

**Type1 JDBC driver is not designed to interact with database software directly.**

**It is designed to take the support of ODBC Driver and Vendor DB library to locate and interact with database software.**

**Diagram: jdbc4.1**

****

**Advantages:**

**1) Using Type1 JDBC driver we can interact with any database software.**

**2) It is a built-in driver of JDK.**

**Disadvantages:**

**1) This driver performance is low. It is not suitable for medium and large scale projects.Hence**

**it is not a industry standard driver.**

**2) To work with Type1 JDBC we need to arrange ODBC driver and Vendor DB library.**

**3) Since ODBC driver and vendor db library present at client side. So it is not suitable for untrusted**

**applets to database communication.**

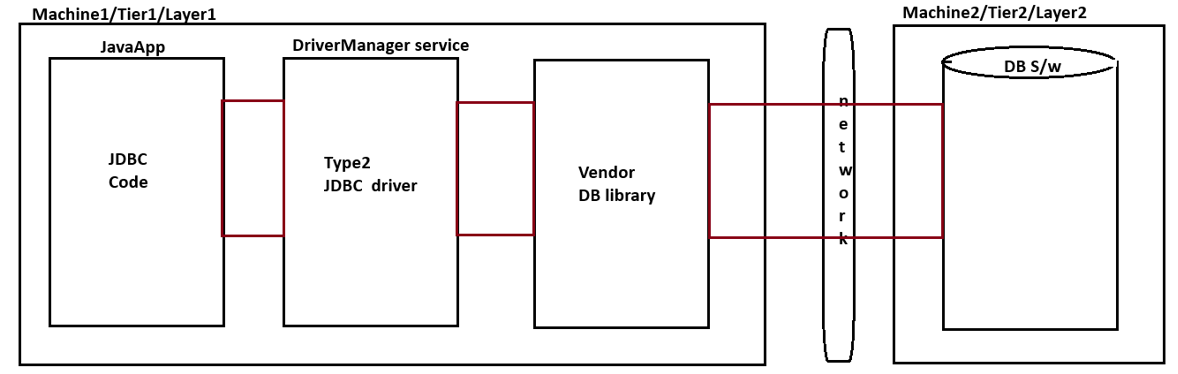
**Type2 JDBC Driver Architecture / Native API**

**===========================================**

**Type2 JDBC driver is not designed to interact with database software directly.**

**It is designed to take the support of vendor DB library to locate and interact with database software.**

**Diagram: jdbc4.2**

****

**Advantages:**

**1) This driver will give better performance when compare to Type1 JDBC Driver.**

**2) Type2 JDBC driver will not take the support of ODBC driver.**

**Disadvantages:**

**1) This driver performance is quit slow. It is not suitable for medium and large scale projects.Hence**

**it is not a industry standard driver.**

**2) To work with Type2 JDBC driver we need to arrange vendor db library.**

**3) Since vendor DB library present at client side so it is not suitable for untrusted applets to**

**database communication.**

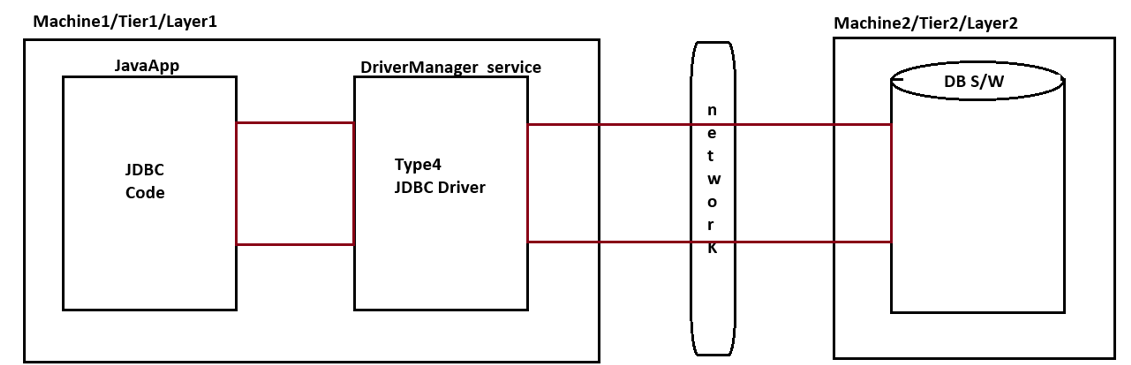
**Type4 JDBC Driver Architecture / Native Protocol / Java Driver**

**===============================================================**

**Type4 JDBC driver is not designed to take the support of ODBC Driver or Vendor db library.**

**It is designed to interact with database software directly.**

**Diagram: jdbc4.3**

****

**Advantages:**

**1) This driver gives better performance when compare to Type1 and Type2 JDBC Driver.**

**2) This driver is completely developed in java so it will give platform independency.**

**3) This driver will not take the support of ODBC driver and Vendor DB library.**

**4) Since ODBC driver and Vendor DB library not present at client side so it suitable for untrusted**

**applets to database communication**

**5) It is suitable for medium and large scale projects.Hence it is a industry standard driver.**

**Disadvantages:**

**1) It is not a built-in driver of JDK.**

**2) For every database software we need to arrange type4 jdbc driver seperately.**

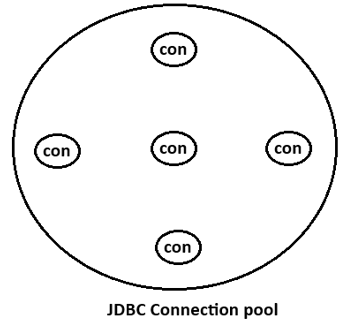
**JDBC Connection pool**

**=====================**

**It is a factory containing a set of readily available JDBC Connection objects before actual being used.**

**JDBC Connection pool represent connectivity with same database software.**

**Diagram: jdbc4.4**

****

**Advantages:**

**1) It create reusable JDBC Connection objects.**

**2) With minimum number of Connection objects we can interact with multiple clients.**

**3) A programmer is not responsible to create, manage or destroy JDBC Connection objects.**

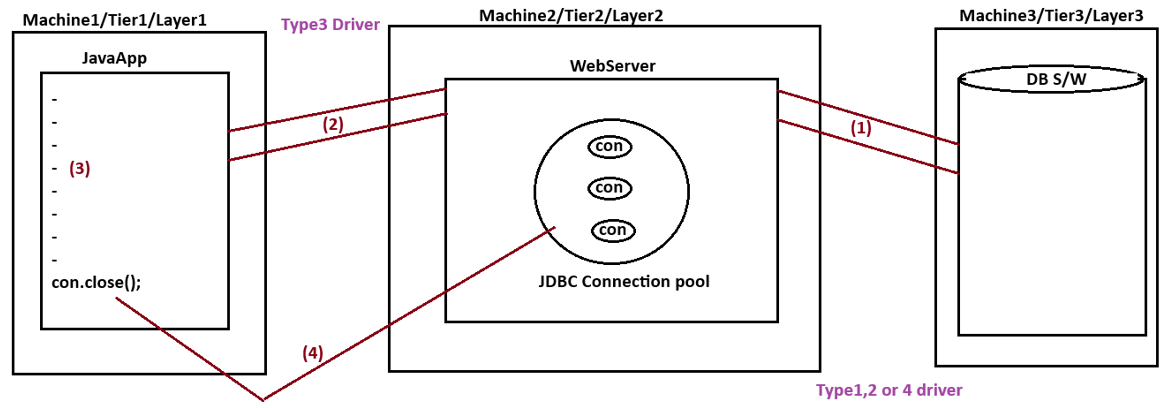
**Type3 JDBC Driver Architecture / Net Protocol**

**==============================================**

**Webserver, proxy server or IDE's server contains JDBC Connection pool.**

**Type3 JDBC Driver is not designed to interact with database software directly. It is designed to interact with Webserver, proxy server or IDE's server to get one reusable JDBC Connection object from JDBC Connection pool.**

**Diagram: jdbc4.5**

****

**With respect to the diagram:**

**1) Webserver or proxy server interacts with database software and gets some reusable JDBC Conneciton**

**objects in JDBC Connection pool.**

**2) Our application interacts with webserver and gets one reusable JDBC Connection object.**

**3) Our application uses JDBC Connection object to create other JDBC Connections.**

**4) Once if we call con.close(), JDBC Connection object goes back to JDBC Connection pool.**

**Class 05**

**Q) Types of Connection objects in JDBC?**

**We have two types of Connection objects in JDBC.**

**1) Direct JDBC Connection object**

**2) Pooled JDBC Connection object**

**1) Direct JDBC Connection object**

**----------------------------------**

**A JDBC Connection object which is created by the user as per the application requirement is called direct JDBC Connection object.**

**ex:**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=**

**DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**2) Pooled JDBC Connection object**

**---------------------------------**

**A JDBC Connection object which is gathered from JDBC Connection pool is called pooled JDBC connection object.**

**Q) Types of Statements in JDBC?**

**We have three statements in JDBC.**

**1) Simple Statement object**

**--------------------------**

**It is an object of underlying supplied java class which implements java.sql.Statement interface.**

**2) PreparedStatement object**

**---------------------------**

**It is an object of underlying supplied java class which implements java.sql.PreparedStatement interface.**

**3) CallableStatement object**

**--------------------------**

**It is an object of underlying supplied java class which implements java.sql.CallableStatement interface.**

**SQL Injection problem**

**=====================**

**Along with input values if we pass special SQL instructions which change the behaviour of a query and behaviour of an application is called SQL Injection Problem.**

**Here special SQL instruction means comment in SQL i.e --.**

**While dealing with simple Statement object there is a chance of raising SQL injection problem.**

**ex:**

**username : raja'--**

**password : hyd**

**Valid Credentials**

**userlist table**

**---------------**

**drop table userlist;**

**create table userlist(uname varchar2(10),pwd varchar2(10));**

**insert into userlist values('raja','rani');**

**insert into userlist values('king','kingdom');**

**commit;**

**ex:**

**---**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**import java.util.Scanner;**

**public class SQLInjProbApp**

**{**

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

**{**

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

**System.out.println("Enter the username :");**

**String name=sc.next();**

**System.out.println("Enter the password :");**

**String pass=sc.next();**

**//convert input values according to SQL query**

**name="'"+name+"'";**

**pass="'"+pass+"'";**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="select count(\*) from userlist where uname="+name+" and pwd="+pass;**

**ResultSet rs=st.executeQuery(qry);**

**int result=0;**

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

**{**

**result=rs.getInt(1);**

**}**

**if(result==0)**

**System.out.println("Invalid Credentials ");**

**else**

**System.out.println("Valid Credentials ");**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**Limitations with Simple Statement object**

**========================================**

**1) It is not suitable to execute same query for multiple times with same values or different values.**

**2) It raises SQL injection problem.**

**3) Framing query with variables is quit complex.**

**4) We can't place string values directly to query parameter without any conversion.**

**5) It does not allow us to insert date values to database table column.**

**6) It does not allow us to insert LOB values to database table column.**

**To overcome these above limitations we need to use PreparedStatement object.**

**Pre-compiled SQL Query**

**======================**

**Our query goes to database software without inputs and becomes parsed query either it is executed or not is called pre-compiled SQL query.**

**A PreparedStatement object deals with pre-compiled SQL Query.**

**Steps to work with PreparedStatement object**

**============================================**

**step1:**

**------**

**Create a query with placeholders or parameters.**

**ex:**

**String qry="insert into student values(?,?,?)";**

**step2:**

**------**

**Convert SQL query to pre-compiled SQL query.**

**ex:**

**PreparedStatement ps=con.prepareStatement(qry);**

**step3:**

**------**

**Set the values to query parameters.**

**ex:**

**ps.setInt(1,no);**

**ps.setString(2,name);**

**ps.setString(3,add);**

**step4:**

**-----**

**Execute the pre-compiled SQL query.**

**ex:**

**ps.executeUpdate();**

**step5:**

**------**

**Close PreparedStatement object.**

**ex:**

**ps.close();**

**Q) Write a JDBC Application to insert a record into student table?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.util.Scanner;**

**public class PSInsertApp**

**{**

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

**{**

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

**System.out.println("Enter the student no :");**

**int no=sc.nextInt();**

**System.out.println("Enter the student name :");**

**String name=sc.next();**

**System.out.println("Enter the student address :");**

**String add=sc.next();**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**String qry="insert into student values(?,?,?)";**

**PreparedStatement ps=con.prepareStatement(qry);**

**//set the values**

**ps.setInt(1, no);**

**ps.setString(2, name);**

**ps.setString(3, add);**

**//execute**

**int result=ps.executeUpdate();**

**if(result==0)**

**System.out.println("No Record Inserted");**

**else**

**System.out.println("Record Inserted");**

**ps.close();**

**con.close();**

**}**

**}**

**Q) Write a JDBC application to update student name based on student number?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.util.Scanner;**

**public class PSUpdateApp**

**{**

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

**{**

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

**System.out.println("Enter the student no:");**

**int no=sc.nextInt();**

**System.out.println("Enter the student name :");**

**String name=sc.next();**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**String qry="update student set sname=? where sno=?";**

**PreparedStatement ps=con.prepareStatement(qry);**

**//set the values**

**ps.setString(1,name);**

**ps.setInt(2, no);**

**//execute**

**int result=ps.executeUpdate();**

**if(result==0)**

**System.out.println("No Record Updated");**

**else**

**System.out.println("Record Updated");**

**ps.close();**

**con.close();**

**}**

**}**

**Q) Write a JDBC application to delete student record based on student no?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.util.Scanner;**

**public class PSDeleteApp**

**{**

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

**{**

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

**System.out.println("Enter the student no :");**

**int no=sc.nextInt();**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**String qry="delete from student where sno=?";**

**PreparedStatement ps=con.prepareStatement(qry);**

**//set the values**

**ps.setInt(1, no);**

**//execute**

**int result=ps.executeUpdate();**

**if(result==0)**

**System.out.println("No Record Deleted");**

**else**

**System.out.println("Record Deleted");**

**ps.close();**

**con.close();**

**}**

**}**

**Q) Write a JDBC application to select student records using PrepareStatement object?**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**public class PSSelectApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**String qry="select \* from student";**

**PreparedStatement ps=con.prepareStatement(qry);**

**ResultSet rs=ps.executeQuery();**

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

**{**

**System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.close();**

**ps.close();**

**con.close();**

**}**

**}**

**Solution for SQL Injection problem**

**===================================**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.util.Scanner;**

**public class SolForSQLInjProbApp**

**{**

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

**{**

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

**System.out.println("Enter the username :");**

**String name=sc.next();**

**System.out.println("Enter the password :");**

**String pass=sc.next();**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**String qry="select count(\*) from userlist where uname=? and pwd=?";**

**PreparedStatement ps=con.prepareStatement(qry);**

**//set the values**

**ps.setString(1, name);**

**ps.setString(2, pass);**

**//execute**

**ResultSet rs=ps.executeQuery();**

**int result=0;**

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

**{**

**result=rs.getInt(1);**

**}**

**if(result==0)**

**System.out.println("Invalid Credentials");**

**else**

**System.out.println("Valid Credentials");**

**rs.close();**

**ps.close();**

**con.close();**

**}**

**}**

**Assignment**

**===========**

**Q) Write a JDBC application to create a student table?**

**Class 06**

**Working with Date values**

**=======================**

**While dealing with DOB, DOA, DOR, DOD and etc we need to insert and retrieve date values.**

**It is never recommanded to store date values in the form of string because it won't give proper comparision between two dates.**

**Every database support different date patterns.**

**ex:**

**oracle - dd-MMM-yy**

**MySQL - yyyy-MM-dd**

**While dealing with simple Statement object we can't insert date value to query parameter.**

**To overcome this limitation we need to PreparedStatement object.**

**To set the date value to query parameter we need to use setDate(-,-) method.**

**A java.util.Date class is not suitable to perform database operation.**

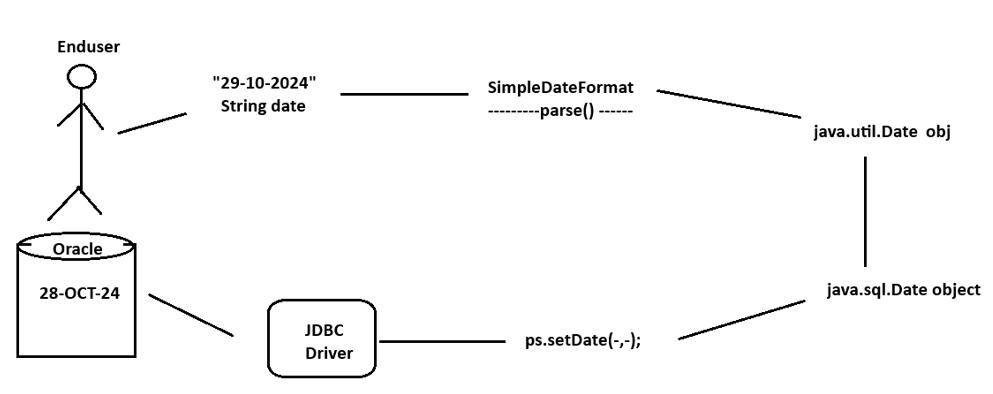
**A java.sql.Date class is suitable to perform database operation.**

**Once JDBC driver gets the date value then it will insert in the pattern which is supported by underlying database software.**

**Standard procedure to insert date values**

**----------------------------------------**

**Diagram: jdbc6.1**

****

**1) Enduser gives date values in the form of string.**

**2) A parse() method of SimpleDateFormat class converts string date to java.util.Date class object.**

**3) Our application converts java.util.Date class object to java.sql.Date class object.**

**4) A ps.setDate(-,-) method is used to set the date values to query parameters.**

**5) Once JDBC driver gets the date value then it will insert into the database in the pattern which is**

**supported by underlying database software.**

**emp1 table**

**=========**

**drop table emp1;**

**create table emp1(eid number(3),ename varchar2(10),edoj date);**

**ex:**

**---**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.text.SimpleDateFormat;**

**import java.util.Scanner;**

**public class DateInsertApp**

**{**

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

**{**

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

**System.out.println("Enter the employee id :");**

**int no=sc.nextInt();**

**System.out.println("Enter the employee name :");**

**String name=sc.next();**

**System.out.println("Enter the DOJ (dd-MM-yyyy) :");**

**String sdoj=sc.next();**

**//convert string date to util date**

**SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");**

**java.util.Date udoj=sdf.parse(sdoj);**

**//converting util date to sql date**

**long ms=udoj.getTime();**

**java.sql.Date sqldoj=new java.sql.Date(ms);**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**String qry="insert into emp1 values(?,?,?)";**

**PreparedStatement ps=con.prepareStatement(qry);**

**//set the values**

**ps.setInt(1, no);**

**ps.setString(2, name);**

**ps.setDate(3, sqldoj);**

**//execute**

**int result=ps.executeUpdate();**

**if(result==0)**

**System.out.println("No Record Inserted");**

**else**

**System.out.println("Record Inserted");**

**ps.close();**

**con.close();**

**}**

**}**

**DateRetrieveApp.java**

**---------------------**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**import java.text.SimpleDateFormat;**

**public class DateRetrieveApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="select \* from emp1";**

**ResultSet rs=st.executeQuery(qry);**

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

**{**

**int no=rs.getInt(1);**

**String name=rs.getString(2);**

**java.sql.Date sqldoj=rs.getDate(3);**

**//convert sql date to util date**

**java.util.Date udoj=(java.util.Date)sqldoj;**

**//convert util date to string date**

**SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");**

**String sdoj=sdf.format(udoj);**

**System.out.println(no+" "+name+" "+sdoj);**

**}**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**Working with LOB values**

**=======================**

**Files are known as LOB's.**

**We have two types of LOB's.**

**1) BLOB (Binary Large Object)**

**---------------------------**

**ex:**

**Images, audio, video , avi file and etc.**

**2) CLOB (Character Large Object)**

**--------------------------------**

**ex:**

**text file , advanced text file , doc file and etc.**

**While dealing with simple Statement object we can set LOB value directly to query parameter.**

**To overcome this limitation we need to use PreparedStatement object.**

**We can set LOB values to query parameter by using following methods.**

**ex:**

**ps.setBLOB(-,-,-) / ps.setBinaryStream(-,-,-)**

**ps.setCLOB(-,-,-) / ps.setCharacterStream(-,-,-)**

**emp2 table**

**===========**

**drop table emp2;**

**create table emp2 (eid number(3),ename varchar2(10),ephoto BLOB);**

**PhotoInsertApp.java**

**-----------------**

**package com.ihub.www;**

**import java.io.File;**

**import java.io.FileInputStream;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.util.Scanner;**

**public class PhotoInsertApp**

**{**

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

**{**

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

**System.out.println("Enter the employee id : ");**

**int id=sc.nextInt();**

**System.out.println("Enter the employee name : ");**

**String name=sc.next();**

**//locate a file**

**File f=new File("src/com/ihub/www/rock.jpg");**

**FileInputStream fis=new FileInputStream(f);**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**String qry="insert into emp2 values(?,?,?)";**

**PreparedStatement ps=con.prepareStatement(qry);**

**//set the values**

**ps.setInt(1, id);**

**ps.setString(2, name);**

**ps.setBinaryStream(3, fis,(int)f.length());**

**//execute**

**int result=ps.executeUpdate();**

**if(result==0)**

**System.out.println("No Record Inserted");**

**else**

**System.out.println("Record Inserted");**

**ps.close();**

**con.close();**

**}**

**}**

**PhotoRetrieveApp.java**

**----------------------**

**package com.ihub.www;**

**import java.io.FileOutputStream;**

**import java.io.InputStream;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**public class PhotoRetrieveApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="select \* from emp2";**

**ResultSet rs=st.executeQuery(qry);**

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

**{**

**InputStream is = rs.getBinaryStream(3);**

**FileOutputStream fos=new FileOutputStream("D:\\navneeth.png");**

**int byteReads=0;**

**byte[] buffer=new byte[250];**

**while((byteReads=is.read(buffer))!=-1)**

**{**

**fos.write(buffer,0, byteReads);**

**}**

**fos.close();**

**}**

**System.out.println("Please check the location");**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**DatabaseMetaData**

**================**

**DatabaseMetaData is an interface which is present in java.sql package.**

**DatabaseMetaData provides metadata of a database.**

**DatabaseMetaData gives information about database product name, database product version, database driver name, database driver version, database username and etc.**

**We can create DatabaseMetaData object by using getMetaData() method of Connection object.**

**ex:**

**DatabaseMetaData dbmd=con.getMetaData();**

**ex:**

**---**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DatabaseMetaData;**

**import java.sql.DriverManager;**

**public class DBMDApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**DatabaseMetaData dbmd=con.getMetaData();**

**System.out.println(dbmd.getDatabaseProductName());**

**System.out.println(dbmd.getDatabaseProductVersion());**

**System.out.println(dbmd.getDriverName());**

**System.out.println(dbmd.getDriverVersion());**

**System.out.println(dbmd.getUserName());**

**con.close();**

**}**

**}**

**Interview Question**

**--------------------**

**Q) Write a java program to display reverse of a given number?**

**package com.ihub.www;**

**import java.util.Scanner;**

**public class Example {**

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

**{**

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

**System.out.println("Enter the number :");**

**int n=sc.nextInt(); //123**

**//convert number to string**

**String str=Integer.toString(n);**

**StringBuffer sb=new StringBuffer(str);**

**str= sb.reverse().toString();**

**//int rev=Integer.parseInt(str);**

**System.out.println(str);**

**}**

**}**

**Class 07**

**ResultSetMetaData**

**==================**

**ResultSetMetaData is an interface which is present in java.sql package.**

**ResultSetMetaData provides metadata of a table.**

**ResultSetMetaData gives information about number of columns, name of columns, type of columns, size of columns and etc.**

**We can create ResultSetMetaData object by using getMetaData() method of ResultSet object.**

**ex:**

**ResultSetMetaData rsmd=rs.getMetaData();**

**ex:**

**----**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.ResultSetMetaData;**

**import java.sql.Statement;**

**public class RSMDApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**String qry="select \* from student";**

**ResultSet rs=st.executeQuery(qry);**

**ResultSetMetaData rsmd=rs.getMetaData();**

**System.out.println(rsmd.getColumnCount());**

**System.out.println(rsmd.getColumnName(1));**

**System.out.println(rsmd.getColumnTypeName(2));**

**System.out.println(rsmd.getColumnDisplaySize(2));**

**rs.close();**

**st.close();**

**con.close();**

**}**

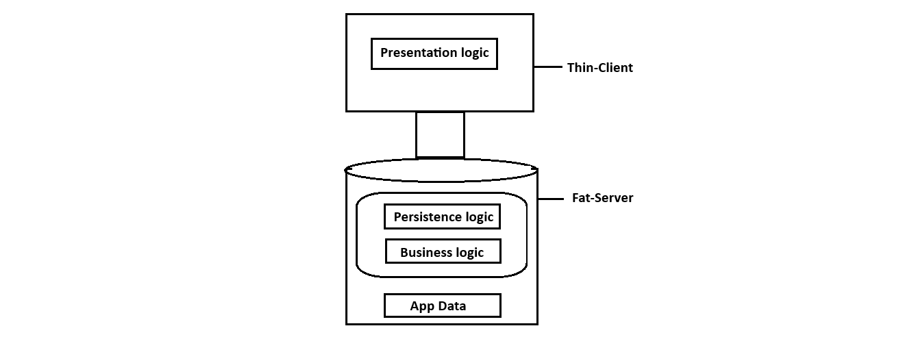
**}**

**Thin-Client/Fat-Server application**

**==================================**

**Every JDBC application consider is a thin-client/fat-server application.**

**Diagram: jdbc7.1**

****

**To develop thin-client/fat-server application we need to save persistence logic and business logic in the database software in the form of PL/SQL procedures and functions.**

**To deal with PL/SQL procedures and functions we will use CallableStatement object.**

**PL/SQL Procedure**

**----------------**

**create or replace procedure first\_proc(A IN number,B IN number, C OUT number)**

**is**

**begin**

**C:=A+B;**

**END;**

**/**

**ex:**

**---**

**package com.ihub.www;**

**import java.sql.CallableStatement;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.Types;**

**public class CallebleStmtApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**//create a Callable stmt object**

**CallableStatement cst=con.prepareCall("{CALL first\_proc(?,?,?)}");**

**//register out parameter**

**cst.registerOutParameter(3,Types.INTEGER);**

**//set the values to IN parameter**

**cst.setInt(1, 30);**

**cst.setInt(2, 20);**

**//execute**

**cst.execute();**

**//gather the result**

**int result=cst.getInt(3);**

**System.out.println("sum of two numbers is ="+result);**

**cst.close();**

**con.close();**

**}**

**}**

**Working with properties file**

**=============================**

**In regular intervals our DBA will change username and password for security reason.**

**It is never recommanded to pass database properties directly to the application.**

**It is always recommanded to read database properties from properties file.**

**A properties file contains key and value pair.**

**dbdetails.properties**

**--------------------**

**driver= oracle.jdbc.driver.OracleDriver**

**url= jdbc:oracle:thin:@localhost:1521:XE**

**username= system**

**password= admin**

**PropertiesFileApp.java**

**---------------------**

**package com.ihub.www;**

**import java.io.FileInputStream;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**import java.util.Properties;**

**public class PropertiesFileApp**

**{**

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

**{**

**//file and locate properties file**

**FileInputStream fis=new FileInputStream("src/com/ihub/www/dbdetails.properties");**

**//create a Properties class object**

**Properties p=new Properties();**

**//load the data from properties file to properties class.**

**p.load(fis);**

**//gather the data from properties class.**

**String s1=p.getProperty("driver");**

**String s2=p.getProperty("url");**

**String s3=p.getProperty("username");**

**String s4=p.getProperty("password");**

**Class.forName(s1);**

**Connection con=DriverManager.getConnection(s2,s3,s4);**

**Statement st=con.createStatement();**

**String qry="select \* from student";**

**ResultSet rs=st.executeQuery(qry);**

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

**{**

**System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**JDBC Flexible Application**

**==========================**

**In JDBC Connection object is a heavy weight object.**

**It is never recommanded to create Connection object in every jdbc application.**

**We need to create a singleton class which returns Connection object.**

**ex:**

**DBConnection.java**

**-----------------**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**public class DBConnection**

**{**

**static Connection con=null;**

**private DBConnection()**

**{**

**}**

**public static Connection getConnection()**

**{**

**try**

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**if(con==null)**

**{**

**con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**}**

**}**

**catch(Exception e)**

**{**

**e.printStackTrace();**

**}**

**return con;**

**}**

**}**

**FlexibleApp.java**

**-----------------**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**public class FlexibleApp**

**{**

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

**{**

**Connection con=DBConnection.getConnection();**

**Statement st=con.createStatement();**

**String qry="select \* from student";**

**ResultSet rs=st.executeQuery(qry);**

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

**{**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**Standard Procedure To Write JDBC Application**

**============================================**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**public class StandardApp**

**{**

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

**{**

**final String DRIVER="oracle.jdbc.driver.OracleDriver";**

**final String URL="jdbc:oracle:thin:@localhost:1521:XE";**

**final String USERNAME="system";**

**final String PASSWORD="admin";**

**final String QUERY="select \* from student";**

**Connection con=null;**

**Statement st=null;**

**ResultSet rs=null;**

**try**

**{**

**Class.forName(DRIVER);**

**con=DriverManager.getConnection(URL,USERNAME,PASSWORD);**

**st=con.createStatement();**

**rs=st.executeQuery(QUERY);**

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

**{**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.close();**

**st.close();**

**con.close();**

**}**

**catch(Exception e)**

**{**

**e.printStackTrace();**

**}**

**}**

**}**

**Assignment**

**==========**

**Q) Write a jdbc application to create a student table using standard procedure?**

**Class 08**

**Batch Processing**

**=================**

**Batch processing used to execute multiple queries in a database.**

**Batch process is a JDBC method to declare multiple queries in a batch to make a single call to the database.**

**To add each query to batch we need to use addBatch() method of Statement object.**

**ex:**

**st.addBatch(query);**

**To execute the batch we need to executeBatch() method of Statement object.**

**ex:**

**int[] arr=st.executeBatch();**

**ex:**

**---**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.Statement;**

**public class BatchProcessingApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement();**

**//declare the queries**

**String qry1="insert into student values(104,'ramulu','pune')";**

**String qry2="update student set sname='rani' where sname='raja'";**

**String qry3="delete from student where sno=103";**

**//Add the queries to batch**

**st.addBatch(qry1);**

**st.addBatch(qry2);**

**st.addBatch(qry3);**

**//execute the batch**

**int[] arr=st.executeBatch();**

**int sum=0;**

**for(int i:arr)**

**{**

**sum+=i;**

**}**

**System.out.println("Number of Records effected are :"+sum);**

**st.close();**

**con.close();**

**}**

**}**

**Transaction Management**

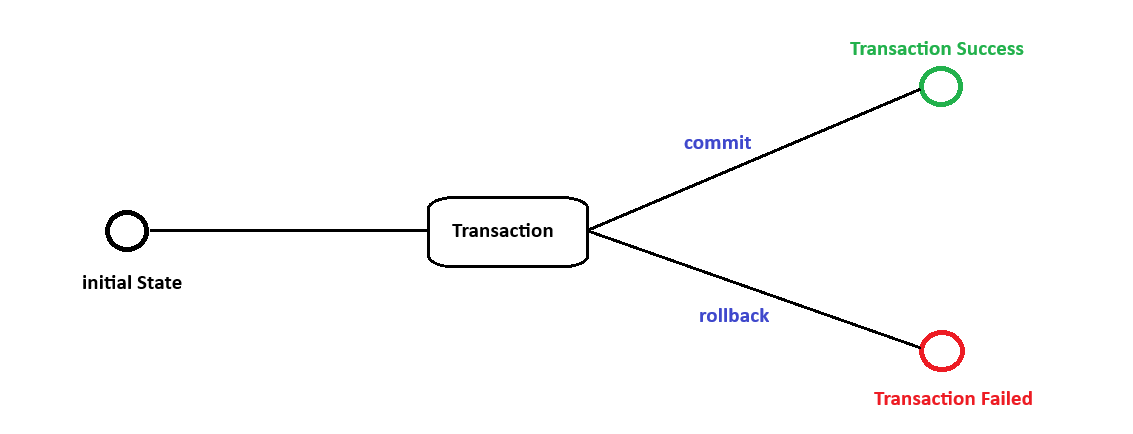
**======================**

**Transaction represents a single unit of work.**

**If transaction done successfully then we will commit.**

**If transaction failed then we will rollback.**

**Diagram: jdbc8.1**

****

**sbi table**

**=========**

**drop table sbi;**

**create table sbi(accno number(6),accholder varchar2(10),accbal number(10));**

**insert into sbi values(111111,'jaydeep',90000);**

**insert into sbi values(222222,'arun',80000);**

**commit;**

**kotak table**

**============**

**drop table kotak;**

**create table kotak(accno number(6),acchold varchar2(10),accbal number(10));**

**insert into kotak values(100001,'ajay',5000);**

**insert into kotak values(200002,'prasad',6000);**

**commit;**

**ex:**

**---**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.Statement;**

**import java.util.Scanner;**

**public class TXNManagementApp**

**{**

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

**{**

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

**System.out.println("Enter the source account number : ");**

**int sno=sc.nextInt();**

**System.out.println("Enter the destination account number :");**

**int dno=sc.nextInt();**

**System.out.println("Enter the amount to transfer :");**

**int amt=sc.nextInt();**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**//set auto commit false**

**con.setAutoCommit(false);**

**Statement st=con.createStatement();**

**//Declare the queries**

**String qry1="update sbi set accbal=accbal-"+amt+" where accno="+sno;**

**String qry2="update kotak set accbal=accbal+"+amt+" where accno="+dno;**

**//add each query to batch**

**st.addBatch(qry1);**

**st.addBatch(qry2);**

**//execute the batch**

**int[] arr=st.executeBatch();**

**boolean flag=true;**

**for(int i:arr)**

**{**

**if(i==0)**

**{**

**flag=false;**

**break;**

**}**

**}**

**if(flag==true)**

**{**

**System.out.println("Transaction done Successfully");**

**con.commit();**

**}**

**else**

**{**

**System.out.println("Transaction Failed");**

**con.rollback();**

**}**

**st.close();**

**con.close();**

**}**

**}**

**Types of ResultSet object**

**=========================**

**We have two types of ResultSet objects.**

**1) Non-Scrollable ResultSet object**

**2) Scrollable ResultSet object**

**Diagram: jdbc8.2**

**1) Non-Scrollable ResultSet object**

**-----------------------------------**

**If a ResultSet object allows us to read the records sequentially, uni-directionally is called non-scrollable ResultSet object.**

**By default every ResultSet object is a non-scrollable ResultSet object.**

**If Statement object is created without TYPE,MODE value then such ResultSet object is called non-scrollable ResultSet object.**

**ex:**

**Statement st=con.createStatment();**

**ResultSet rs=st.executeQuery("select \* from student");**

**2) Scrollable ResultSet object**

**-------------------------------**

**If a ResultSet object allows us to read the records non-sequentially, bi-directionally , randomly is called scrollable ResultSet object.**

**If Statement object is created with TYPE,MODE value then such ResultSet object is called non-scrollable ResultSet object.**

**ex:**

**Statement st=con.createStatment(TYPE,MODE);**

**ResultSet rs=st.executeQuery("select \* from student");**

**JDBC provides two TYPE values.**

**ex:**

**ResultSet.TYPE\_SCROLL\_SENSITIVE**

**ResultSet.TYPE\_SCROLL\_INSENSITIVE**

**JDBC provides two MODE values.**

**ex:**

**ResultSet.CONCUR\_READ\_ONLY**

**ResultSet.CONCUR\_UPDATABLE**

**Various methods present in Scrollable ResultSet object**

**-------------------------------------------------------**

**rs.next()**

**----------**

**It is used to move the record pointer to next position.**

**rs.previous()**

**------------**

**It is used to move the record pointer to previous position.**

**rs.getXxx()**

**-----------**

**It is used to read the values from record ResultSet.**

**rs.getRow()**

**----------**

**It returns position of record pointer.**

**rs.close()**

**---------**

**It is used to close ResultSet object.**

**rs.first()**

**----------**

**It sets the record pointer to first record.**

**rs.isFirst()**

**------------**

**It is used to check record pointer is in first position or not.**

**rs.last();**

**----------**

**It sets the record pointer to last record.**

**rs.isLast()**

**-----------**

**It is used to check record pointer is in last position or not.**

**rs.beforeFirst()**

**----------------**

**It sets the record pointer to BFR position.**

**rs.afterLast()**

**--------------**

**It sets the record pointer to ALR position.**

**rs.relative(+/-)**

**--------------**

**It is used to move record pointer to next position based on current position.**

**rs.absolute(+/-)**

**----------------**

**It is used to move record pointer to next position based on BFR/ALR position.**

**ex**

**---**

**package com.ihub.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**public class ScrollableResultSetApp**

**{**

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

**{**

**Class.forName("oracle.jdbc.driver.OracleDriver");**

**Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");**

**Statement st=con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_READ\_ONLY);**

**String qry="select \* from student";**

**ResultSet rs=st.executeQuery(qry);**

**//top to bottom**

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

**{**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.afterLast();**

**//bottom to top**

**while(rs.previous())**

**{**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.first();**

**System.out.println(rs.isFirst());//true**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**rs.last();**

**System.out.println(rs.isLast());//true**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**//rs.relative(-2);**

**//System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**rs.absolute(-2);**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**rs.close();**

**st.close();**

**con.close();**

**}**

**}**

**Class 09**

**Mysql**

**============**

the differences between MySQL and Oracle, covering various aspects of each database system:

**1. Data Types**

* **MySQL:**
  + **Character Types:** **CHAR, VARCHAR, TEXT, BLOB**
  + **Numeric Types: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL**
  + **Date and Time Types: DATE, DATETIME, TIMESTAMP, TIME, YEAR**
* **Oracle:**
  + **Character Types:** **CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB**
  + **Numeric Types: NUMBER (flexible precision), BINARY\_FLOAT, BINARY\_DOUBLE**
  + **Date and Time Types:** **DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, INTERVAL**

**2. SQL Dialect**

* **MySQL:**
  + Simpler SQL syntax, often more permissive in terms of SQL standards.
  + Function names can differ (e.g., IFNULL() vs. NVL() in Oracle).
* **Oracle:**
  + More complex SQL syntax with specific extensions.
  + PL/SQL supports advanced features like exception handling and cursors.

**3. Transactions and ACID Compliance**

* **MySQL:**
  + **ACID compliance** is dependent on the storage engine (e.g., InnoDB is fully ACID compliant, while MyISAM is not).
  + Transaction control with **START TRANSACTION, COMMIT**, **and ROLLBACK.**
* **Oracle:**
  + Always fully **ACID** compliant, using rollback segments for **transaction** management.
  + Supports complex transaction control and provides features like **savepoints**.

**4. Indexing**

* **MySQL:**
  + Uses B-tree and **full**-**text** **indexes**.
  + Supports composite indexes (multiple columns) and unique indexes.
* **Oracle:**
  + Provides a variety of indexing options including B-tree, **bitmap** **indexes** (efficient for low-cardinality columns), and function-based indexes.
  + Advanced partitioning options for indexing large datasets.

**5. Stored Procedures and Functions**

* **MySQL:**
  + Supports stored procedures and user-defined functions, but syntax can be more limited.
  + Procedures can be created with **CREATE PROCEDURE** and can use flow control statements.
* **Oracle:**
  + Extensive PL/SQL support for writing stored procedures, functions, packages, and triggers.
  + Supports advanced features like cursors, collections, and exception handling.

**6. Partitioning**

* **MySQL:**
  + Basic partitioning support, allowing partitioning by **range**, **list**, **hash**, and **key**.
  + Limited flexibility in managing partitions compared to Oracle.
* **Oracle:**
  + Advanced partitioning options with features like **range**, **list**, **hash**, **composite** **partitioning**, and **subpartitioning**.
  + Highly optimized for managing **large** **datasets**, improving performance and manageability.

**7. Replication and Clustering**

* **MySQL:**
  + Offers master-slave replication, master-master replication, and group replication.
  + MySQL Cluster provides real-time data access and high availability.
* **Oracle:**
  + Advanced replication features including Oracle Data Guard and GoldenGate for data replication and synchronization.
  + Oracle Real Application Clusters (RAC) for high availability and horizontal scaling.

**8. Licensing and Cost**

* **MySQL:**
  + Primarily open-source and free under the GPL, with commercial editions available.
  + No licensing fees for community edition, making it attractive for small to medium-sized applications.
* **Oracle:**
  + Commercial software with significant licensing fees; various editions available (Standard, Enterprise).
  + Free version (Oracle XE) available, but with limitations on CPU and memory.

**9. Community and Support**

* **MySQL:**
  + Strong open-source community with many third-party resources and documentation.
  + MySQL forums and Stack Overflow are popular for support.
* **Oracle:**
  + Comprehensive official support with extensive documentation and resources.
  + Large user community, but more focused on enterprise environments.

**10. Performance Tuning**

* **MySQL:**
  + Basic performance tuning options available (e.g., EXPLAIN for query analysis).
  + Limited built-in tools compared to Oracle.
* **Oracle:**
  + Advanced performance tuning features, including the Oracle Optimizer, Automatic Workload Repository (AWR), and SQL Tuning Advisor.
  + Detailed performance monitoring tools available.

**JDBC Application to interact with MySQL Database**

**=================================================**

**step1:**

**------**

**Download and Installed MySQL database software.**

**ex:**

**https://dev.mysql.com/downloads/installer/**

**step2:**

**--------**

**Connect with mysql by using password.**

**ex:**

**cmd> mysql -u root -p**

**Enter the password : root**

**step3:**

**-------**

**create a SCHEMA in MYSQL.**

**ex:**

**create schema IH\_JAVA\_039;**

**step4:**

**-------**

**To check list of databases /schemas present in mysql db.**

**ex:**

**show databases;**

**step5:**

**-------**

**Use IH\_JAVA\_039 scheme/database.**

**ex:**

**use IH\_JAVA\_039;**

**step6:**

**---------**

**create a student table and insert the records.**

**ex:**

**create table student(sno int(3),sname varchar(10),sadd varchar(10));**

**insert into student values(101,'raja','hyd');**

**insert into student values(102,'raju','delhi');**

**insert into student values(103,'ravi','pune');**

**commit;**

**step7:**

**---------**

**create a JDBC Application to select student records.**

**MySQLApp.java**

**------------------------**

**package com.ge.www;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**public class MySQLApp {**

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

**final String DRIVER="com.mysql.jdbc.Driver";**

**final String URL="jdbc:mysql://localhost:3306/IH\_JAVA\_036";**

**final String USERNAME="root";**

**final String PASSWORD="root";**

**final String QUERY="select \* from student";**

**Connection con=null;**

**Statement st=null;**

**ResultSet rs=null;**

**try**

**{**

**Class.forName(DRIVER);**

**con=DriverManager.getConnection(URL,USERNAME,PASSWORD);**

**st=con.createStatement();**

**rs=st.executeQuery(QUERY);**

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

**{**

**System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));**

**}**

**rs.close();**

**st.close();**

**con.close();**

**}**

**catch(Exception e)**

**{**

**e.printStackTrace();**

**}**

**}**

**}**

**step8:**

**--------**

**Add "mysql-connector.jar" file in project build path for mysql database.**

**right click to project --> built path --> configuration build path --> libraries**

**--> add external jars --> select mysql-connector.jar file --> open.**

**jar file download :**

**https://repo1.maven.org/maven2/com/mysql/mysql-connector-j/8.0.31/**

**Note:**

**ojdbc14.jar --> for oracle**

**mysql-connector-j.jar --> for mysql**

**step9:**

**--------**

**Run the jdbc application.**

**JDBC Application to interact with MongoDB Database**

**====================================================**

**step1:**

**-----**

**Download and install MongoDB Community Server.**

**ex:**

**https://www.mongodb.com/try/download/community**

**step2:**

**-----**

**Downoad and install MongoDB Shell.**

**ex:**

**https://www.mongodb.com/try/download/shell**

**step3:**

**------**

**Extract Mongodb shell inside "MongoDB" folder.**

**ex:**

**C:\Program Files\MongoDB**

**step4:**

**------**

**Copy "bin" directory of mongoshell.**

**ex:**

**C:\Program Files\MongoDB\mongosh-2.3.0-win32-x64\bin**

**step5:**

**-----**

**Paste "bin" directory in environmental variables.**

**ex:**

**right click to This PC --> properties --> Advanced System Settings -->**

**Environmental variables --> System variables --> click to path --> click to edit**

**button --> New --> paste (C:\Program Files\MongoDB\mongosh-2.3.0-win32-x64\bin) -->ok**

**--->ok --->ok.**

**step6:**

**-----**

**Launch Eclipse IDE by choosing workspace location.**

**step7:**

**-----**

**Create MongoDBProj inside eclipse IDE.**

**ex:**

**File --> New --> Java project --> Name : MongoDBProj --> Next --> Finish.**

**step8:**

**-----**

**Download and add mongodb jar file in project build path.**

**ex:**

**jar file link : https://repo1.maven.org/maven2/org/mongodb/mongo-java-driver/3.11.2/**

**right click to MongoDBProj --> Buildpath --> configure build path --> libraries -->**

**click to classpath --> add external jars --> select mongo-java-driver.jar file -->**

**Open --> Apply and close.**

**step9:**

**------**

**Create a "com.ihub.www" package inside "src" folder.**

**step10:**

**------**

**Create a InsertApp.java file inside "com.ihub.www" package.**

**SelectApp.java**

**--------------**

**package com.ihub.www;**

**import org.bson.Document;**

**import com.mongodb.client.MongoClient;**

**import com.mongodb.client.MongoClients;**

**import com.mongodb.client.MongoCollection;**

**import com.mongodb.client.MongoDatabase;**

**public class InsertApp**

**{**

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

**{**

**try(MongoClient mongoClient=MongoClients.create("mongodb://localhost:27017");)**

**{**

**MongoDatabase mongoDatabase=mongoClient.getDatabase("myDatabase");**

**MongoCollection<Document> mongoCollection=mongoDatabase.getCollection("myCollection");**

**Document doc=new Document("id", 101)**

**.append("name","Alan")**

**.append("add","Hyd");**

**mongoCollection.insertOne(doc);**

**System.out.println("Record Inserted in MongoDB");**

**}**

**catch(Exception e)**

**{**

**e.printStackTrace();**

**}**

**}**

**}**

**step11:**

**-------**

**Run the jdbc application.**

**Q) Write a jdbc application to read the record from mongodb?**

**package com.ihub.www;**

**import org.bson.Document;**

**import com.mongodb.client.MongoClient;**

**import com.mongodb.client.MongoClients;**

**import com.mongodb.client.MongoCollection;**

**import com.mongodb.client.MongoDatabase;**

**public class SelectApp**

**{**

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

**{**

**try(MongoClient mongoClient=MongoClients.create("mongodb://localhost:27017");)**

**{**

**MongoDatabase mongoDatabase=mongoClient.getDatabase("myDatabase");**

**MongoCollection<Document> mongoCollection=mongoDatabase.getCollection("myCollection");**

**Document findDocument=mongoCollection.find(new Document("id",101)).first();**

**System.out.println(findDocument.toJson());**

**}**

**catch(Exception e)**

**{**

**e.printStackTrace();**

**}**

**}**

**}**

**MongoDB commands**

**================**

**> mongo dbs;**

**> use myDatabase;**

**myDatabase > db.dropDatabase();**

**> show collections;**

**> db.myCollection.find();**

**Imp Questions in jdbc**

**=========================**

1. **What is JDBC?**
2. **How many steps are there to develop a JDBC application?**
3. **How many drivers are there in JDBC?**
4. **How many statements are there in JDBC?**
5. **What is DatabaseMetaData?**
6. **What is ResultSetMetaData?**
7. **Types of Queries in JDBC?**
8. **What is JDBC Connection pool?**
9. **Types of ResultSet objects?**
10. **Write a JDBC application to create a table in the database.**
11. **Write a JDBC application to insert a record into the student table.**
12. **What is persistence?**
13. **What is a persistence store?**
14. **What is persistence data?**
15. **What are persistence operations?**
16. **What is persistence logic?**
17. **What is persistence technology?**
18. **What is serialization?**
19. **What is deserialization?**
20. **What is Class.forName()?**
21. **What is a Connection object?**
22. **What is DriverManager.getConnection()?**
23. **What is a Statement object?**
24. **Types of Connection objects?**

**Key points in jdbc**

**=================**

  **DriverManager:** Manages a list of database drivers. You can use it to establish a connection to the database.

* **Method**: **getConnection(String url, String user, String password)**

 **Connection:** Represents a connection to the database.

* **Methods**:
  + **createStatement()**
  + **prepareStatement(String sql)**
  + close()

 **Statement:** Used for executing SQL queries.

* **Methods**:
  + **executeQuery(String sql) (for SELECT queries)**
  + **executeUpdate(String sql) (for INSERT, UPDATE, DELETE)**
  + close()

 **PreparedStatement:** A subclass of Statement that allows you to execute precompiled SQL statements with parameters.

* **Methods**:
  + **setInt(int parameterIndex, int value)**
  + **setString(int parameterIndex, String value)**
  + **setDate(int parameterIndex, Date value)**
  + **setStreamBinary(int parameterIndex,file of video,length)**
  + **setAssciiBinary(int parameterIndex,file of pdf/doc,length)**
  + **executeQuery()**
  + **executeUpdate()**
  + close()

 **ResultSet:** Represents the result set of a query.

* **Methods**:
  + **next()**
  + **getString(String columnLabel)**
  + **getInt(String columnLabel)**
  + close()

**Important Points**

1. **Loading Drivers**: Use **Class.forName("driver-class-name")** to load the appropriate JDBC driver.
2. **Connection URL**: The connection URL format typically looks like jdbc:subprotocol:subname. For example, for MySQL, it would be jdbc:mysql://localhost:3306/databaseName.
3. **Error Handling**: Always handle SQL exceptions using try-catch blocks. Use SQLException to catch database-related errors.
4. **Resource Management**: Always close your Connection, Statement, and ResultSet objects to prevent resource leaks. Use try-with-resources where possible.
5. **Transactions**: Manage transactions using **setAutoCommit(false)** and commit()/rollback() methods for better control over database operations.
6. **Batch Processing**: Use **addBatch() and executeBatch()** methods in Statement or PreparedStatement for executing multiple queries at once.
7. **Connection Pooling**: For better performance in enterprise applications, consider using connection pooling libraries like HikariCP or Apache DBCP.
8. **Data Types**: Be aware of how JDBC maps SQL data types to Java data types (e.g., SQL VARCHAR to Java String, SQL INTEGER to Java int).