J2SE — Introduction
oops
Inner classes
String Man
E·H·M
M·T
IOStreams
Networking
wrapper classes
collection f/m·
GUI
- Awt
- Swing
- applet

I18N
Reflection api
Annotations
RMI
Jdbc 4.0

RMI, JDBC also part of J2SE

All J2E books are individual books

J2EE — component based technology
- web component (Servlet, JSP) 3.0 2.2
- Distributed component (EJB's)
- XML component (JAX-B, JAX-R, JAX-P, JAX-RPC)

Service oriented technology
JNDI, JMS, Javamail, JDBC, JTA, Security, ---

Communication oriented technology
RMI-IIOP, CORBA-OMG

J2ME/ME

Date: 27th Apr, 2012 Fri

- Jdbc
- Servlets
- JSP
- JSTL
- EL

Edit tools:
- edit plus
- eclipse
- Myeclipse
- NetBeans
- JDeveloper

Databases:
- oracle 11xe
- MySQL 5.x
- postgreSQL

- Tomcat
- weblogic
- GlassFish
- JBoss

(i) Introduction:

1. DB vs DBMS
2. Query processing System
3. Driver & Types

(ii) Steps to design First java appln:

1. create — Jdbc application
   - insert //
   - update //
   - delete //
   - drop //
2. ResultSet
   1. Forward only RS
   2. Scrollable RS
      1. Scrollable Sensitive RS
      2. Scrollable Insensitive RS.
   3. Read only ResultSet
   4. Updatable RS.

(iv) prepared Statement
① Insert — JDBC appl
② update — JDBC appl.
③ select — JDBC appl.

(5) Callable Statement

① procedure
② function

① Proced — JDBC Appl.
② function — JDBC Appl.

(6) Transaction Management
① ACID properties
② Atomicity — Commit modes — JDBC appl.
③ Isolation (Auto Commit — Non auto Commit)
↳ JDBC Appl.
④ Save point

(7) Batch updation.

(8) Metadata
RSMD — JDBC Appl.
RSMD — JDBC Appl.

(9) Connection pooling
— JDBC Appl

(10) BLOB & CLOB Data types

Date: 30th Apr, 2012 Mon

JDBC: As part of enterprise application development, it is essential to manage the organizations data like employee details, products details, services details, clients details and so on.

To manage the above specified data in enterprise applications, we have to use storage areas.

There are two types of storage areas
1. Temporary storage areas
2. permanent storage areas

Temporary storage areas: These are the memory elements which can be used to store the data temporarily.
① Buffers
② Java objects.

permanent storage areas:

Buffers are the temporary storage area which can be used for the application requirement.

Java objects are also temporary memory elements. They are created in the heap memory whenever JVM encounters the object creation statements in the program execution. After the termination of the pgm, these java objects will be removed by garbage collector automatically.

These are the memory elements, which can be used to store the data

permantly.

Expl) File Systems
Database Management Systems (DBMS)
Datawarehouses (DW)

2> where File system is a system which could be provided by the local operating system, which is platform dependent so that it is not suitable for the platform independent technologies like java.

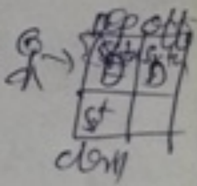
— In general file systems can be used to store less data.

— In general file systems are able to provide less security for the data. In applications if we use file system as a backend then it may increase data redundancy.

— In general file systems may not provide query language support so that the database operations are difficult.

To overcome the above problems we will use database management systems

lock out feel class from OS. Nodes are class file and exe are on linux ©



Database management system is very good when compared with file system and it is able to manage less data when compared with data warehouse

— Database management system is very good when it is required to store the data but it is not good when performing retrieval operations due to the lack of very good retrieval mechanisms.

— Data warehouses is able to manage large and large volumes of data and it is having fast retrieval mechanisms in the form of data mining techniques.

Database Management System

Def: what is the def of database & database management system.

① Database is a memory element to store data. Database management system is a s/w system, it can be used to manage the data by storing it on to the database, by retrieving it from database and so on —

② Database is a collection of inter related data. Database Management System is a collection of inter related data and a set of programs to access the data.

- In general there are 3 types of database management systems
  1. Relational database management System (RDBMS)
  2. Object oriented database management System (OODBMS)
  3. Object relational database management System (ORDBMS)

Relational database management System

This database management system can be used to represents the data in the form of tables. To perform database operations this database management will use SQL2 as a query language.

Object oriented database manag. Systey

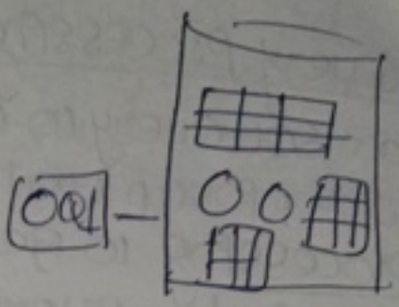
It can be used to represents the data in the form of objects. To perform database operations it will use OQL as query language.

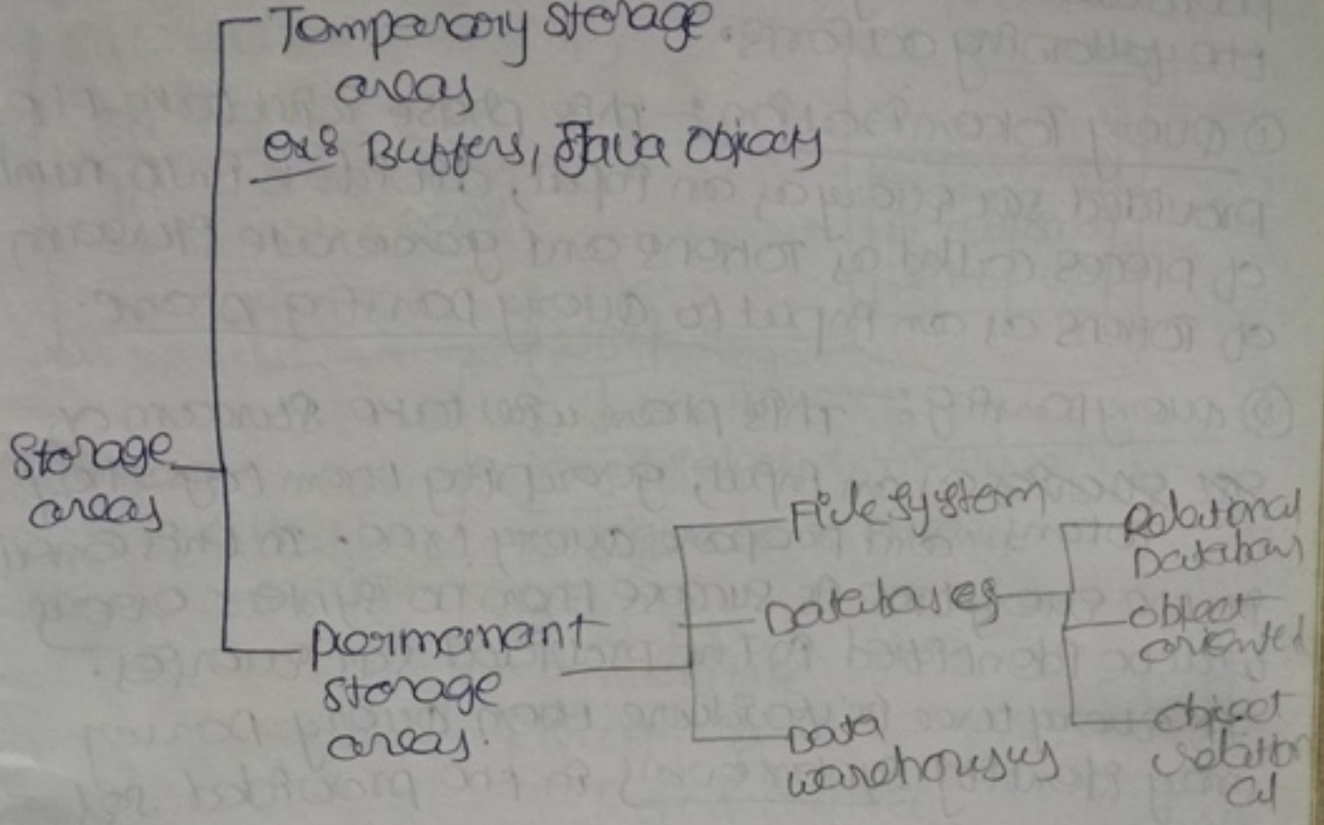
Object relational database man. Syst

This database management system will represent some part of the data in the form of tables and some other part of the data in the form of objects. This database manak mat System will use SQL3 as query language to perform database operations, where SQL3 is a combination of SQL2 and object oriented

features that is OQL

Tree diagram





Query processing System :- In general if we submit an SQL query to the database then database engine will pickup that SQL query and perform query processing to generate query results. To process a particular query database engine will perform the following actions.

① Query Tokenization :- This phase will take the provided SQL query as an input, divided into number of pieces called as Tokens and generate stream of Tokens as an input to query parsing phase.

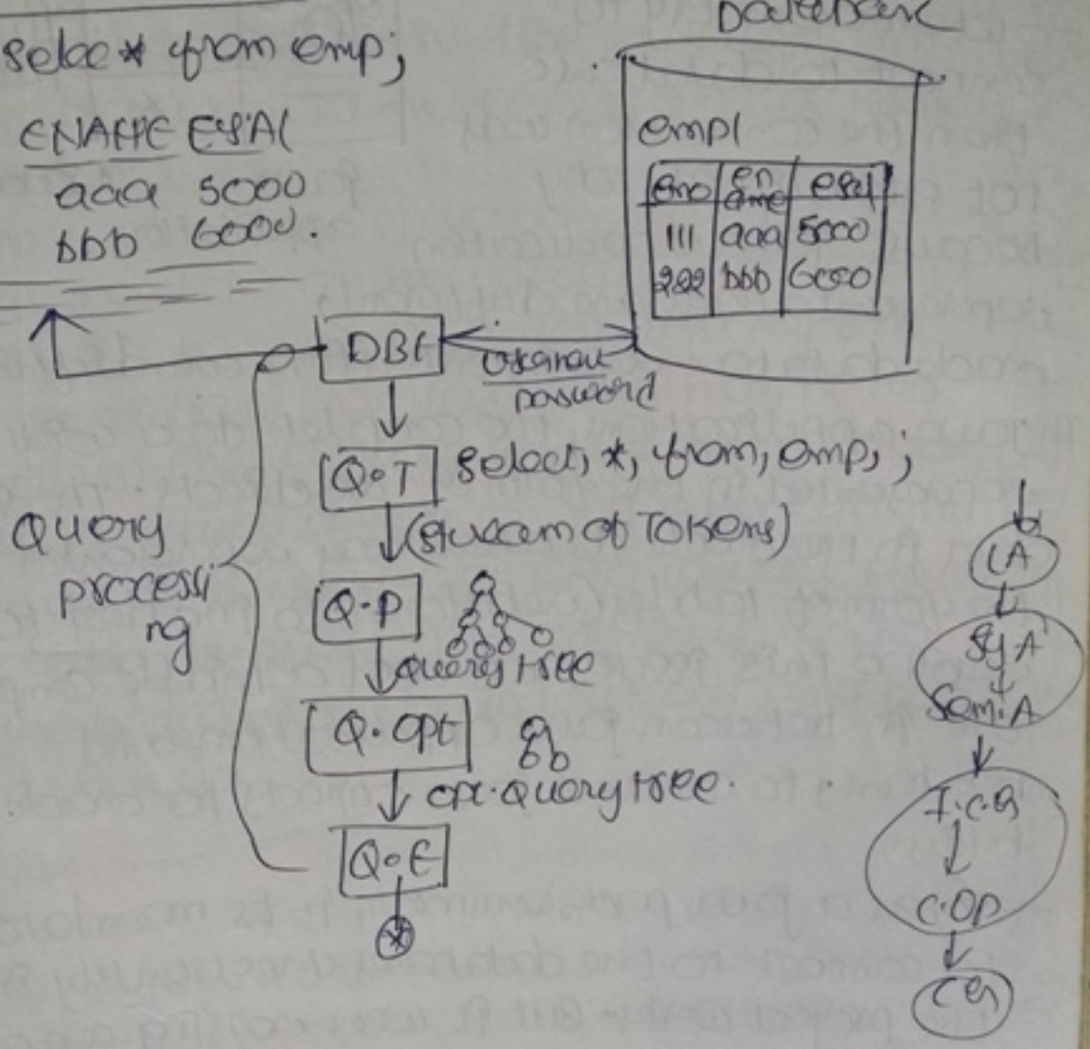
② Query parsing :- This phase will take stream of ~~SQL queries~~ as an input, grouping them together Tokens and prepare query tree. In this case if the query tree is success then no syntax errors will be identified in the provided SQL queries. If query tree is failure then query parser may identify syntax errors in the provided SQL query.

③ Query optimization :- This phase will take query tree as an input, it will perform number of optimizations on query tree inorder to reduce execution time and to optimize memory utilization.

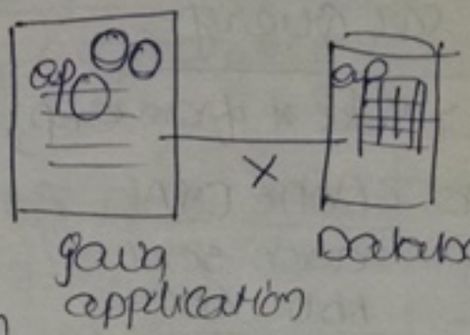
④ Query execution :- This phase will take optimized query tree as an input, execute the queries by using number of interpreters internally.

By the execution of the provided SQL query some results will be generated, database engine will send query execution results to the SQL prompts from where database engine received the SQL query.

```
SQL> selec * from emp;

ENO  ENAME  ESAL
111   aaa   5000
22    bbb   6000.
```



→ Now, I am trying to connect my java program to the database, for example here in this situation think oracle as the database.



— whenever I try to connect to database then the connection was not established, why because java application representations are different, oracle database representations are different. In java applications, the complete data will be represented in the form of objects. The same data in the oracle database was represented in the form of tables (relations). So inorder to resolve this issues, we need a middle component in between java application and database, to convert java objects to oracle tables.

— so, as a java programmer, it is mandatory to connect to the databases frequently in the project works. But it was raising an error. Now, I should mailed to oracle intimating that my program is not connect to the oracle, would you provide me driver implementation software inorder to connect to oracle.

Then in reply to programmer, oracle said that, "sorry boss, you are not only my client — we have the clients all

over would. By considering only your requirement, we are not able to provide the driver to you. If it the requirement of many more clients then we will think in that moment".

— The role of Driver is like a Telugu andra person between Tamil person and a Karnataka person.

Here what is the situation that Karnataka persons wants to communicate with Tamil person. If Karnataka person directly talk with tamilian, then tamilian may be mis understand the language. So here Karnatakian needed a telugu person to acts as a Dubashi. The role of telugu person is to convert the Karnataka language into Tamil language because he was able to understand any both of the language.

Java Database Connectivity :- (JDBC):

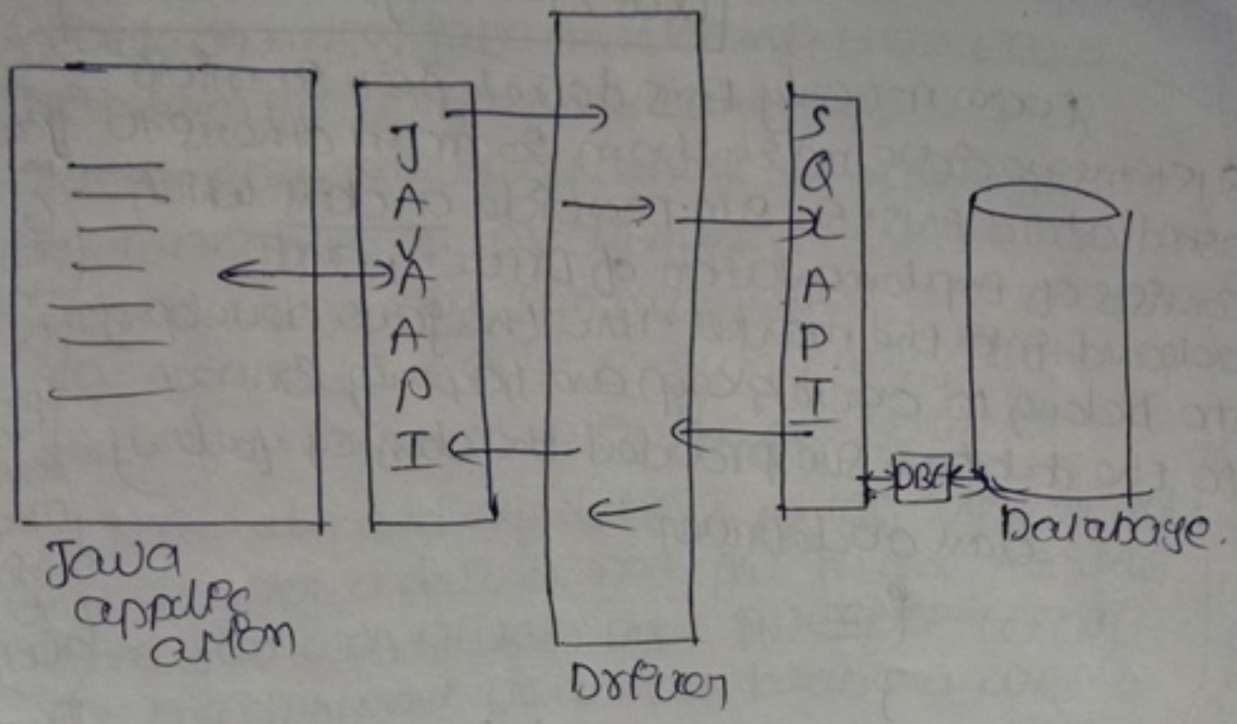
JDBC is an api; it will provide very good predefined library to interact with the database from java application. JDBC is an interface between java application and database, it will provide a very good environment to connect with database from java application inorder to perform the basic database operations.

- In general in JDBC applications we will provide the database logic in java application as per java api rules and regulations. In this context if we send java represented database logic, to the database engine then database engine is unable to execute it because database engine may expect the database logic in query language representations.

In the above context to execute JDBC application we need a conversion mechanism to convert the data from java representations to query language representations and from query language representations to java representations.

To achieve this requirement we have to use drivers.

DBT Com Under tul only this Query type only.



Driver: Driver is an interface existed in between java application and database to map java api calls to query language api calls and query language api calls to java api calls.

In the earlier days, before the drivers were not entered into the market, there are several group of java developers in the industry. Some group of developers only know oracle database queries and some other group of developers only know SQL queries.

— First, one java developer belong to oracle group send mail to the sun microsystems, intimating that provider driver implementation class required inorder to connect to the database server.

Date: 1st May, 2012 Tue

From not only this developers, sun micro systems received many from so many others to asked about this. So sun provide a class which consists of implementation of Drivers, and released into the market. Now the java developers who belong to oracle group are happily connect to the database. Sun provided the class as following

class OracleDriver
{ ==
}

— Now, the java developers who belong to the mysql group, send mail to the sun microsystems intimating that why are you only provide Driver class to oracle? what mistake we done? why are you showing apathy on mysql java developers?

In replay to those group, sun provided the class as following

class MysqlDriver
{ ==
}

Now, the mysql group of java developers happily connect with mysql database.

— Like these databases, there are nearly 150 plus databases are there in the market. For all of that now can the sun provided the class? Sun took this issue as seriously.

Because of providing class to the respective oracle or mysql java developers, there was a problem for sun also. Because of developing driver class, there was a huge demand for the oracle organization products. Here the hard work was done by sun but respect and profit were gone to oracle. So sun thought this as serious thing and provide one solution.

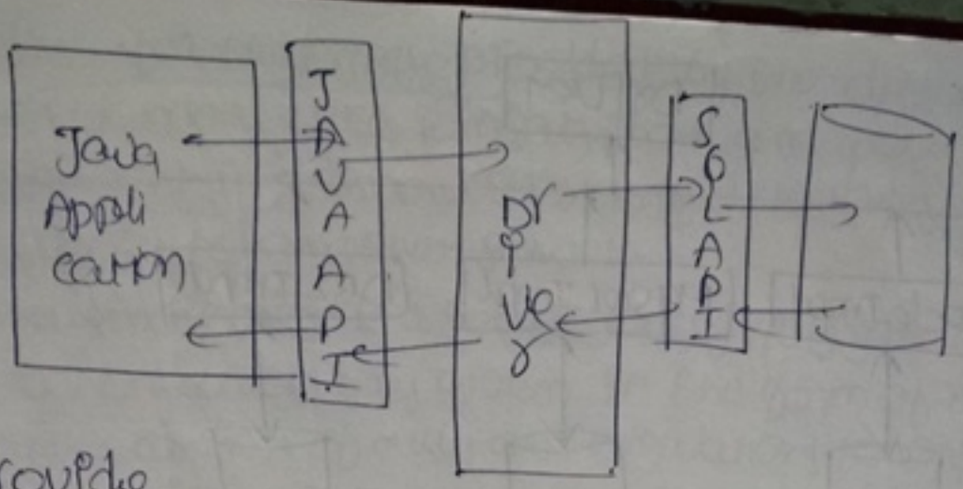
— The rules and regulations which are required to develop a driver were put in one interface and sun was given that interface to all of the different vendors of databases like oracle, DB2, Sybase etc.

Now, it's own responsibility of the database vendors to provide implementation for the driver interface on their own. Due to this the credit problem was also resolved. Sun provide the interface for jdbc driver as

interface Driver
{ ≡ rules & regulations
}

— Now, oracle, DB2, sybase etc were implement this interface into their classes, and provide the solution in the form of jar file. This jar file was act as a driver.

Because no. of databases are 180 drivers. Profit to oracle or mysql not the sun. Put on the creators.

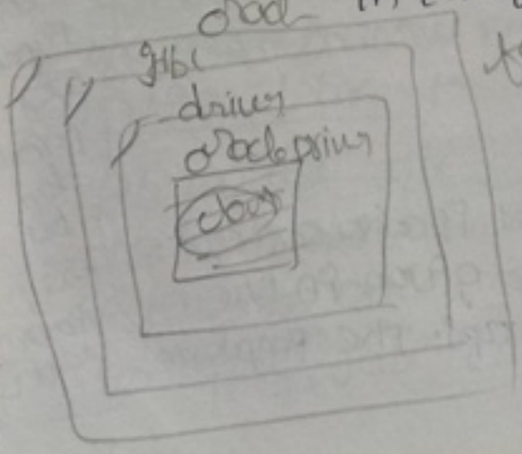


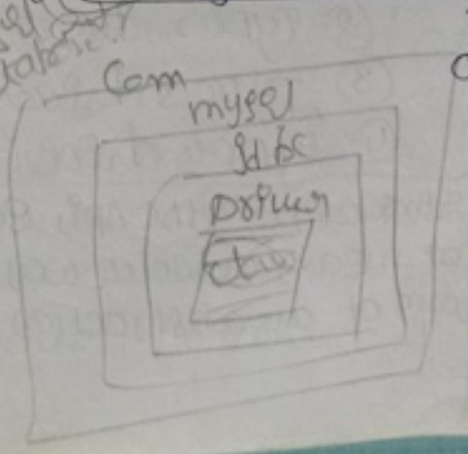
To provide Driver as a product sun microsystems has provided Driver as an interface in the form of java.sql.Driver. By introducing Driver interface sun micro systems has given an option to all the database vendors to provide their own implementation classes for Driver interface.

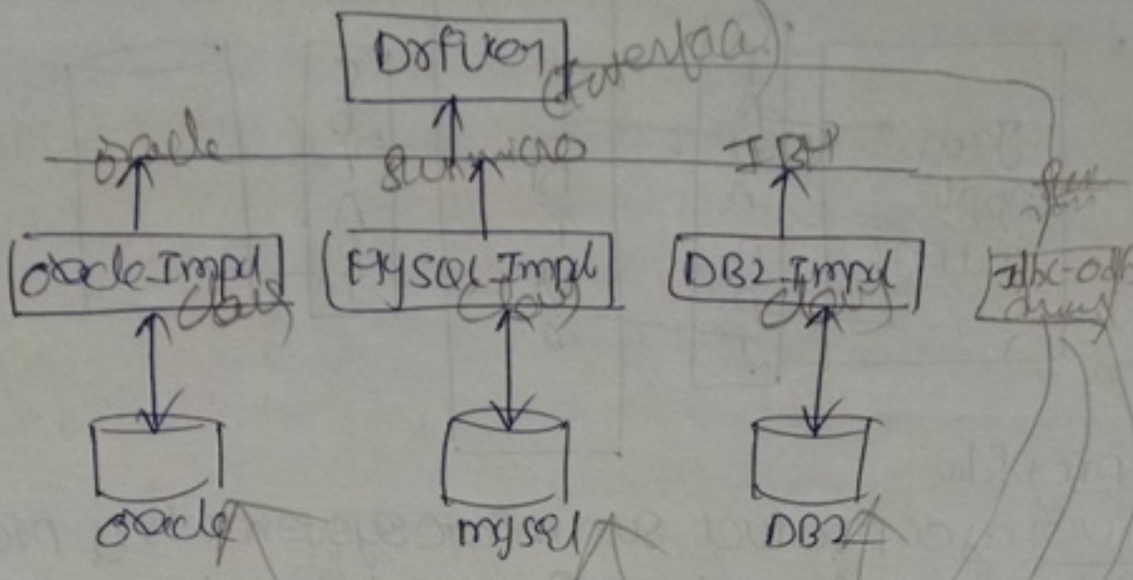
With the above convention almost all the database vendors have provided their own implementation classes in their respective data base softwares.

① Oracle has provided their own implementation class in the form of oracle.jdbc.driver.OracleDriver

② MySQL database has provided their own implementation class in the form of com.mysql.jdbc.Driver







— If we want to use driver in our JDBC application then we have to get the driver implementation from the respective database software.

— JDBC is an API @ an abstraction [collection of interfaces] provided by Sun micro systems but whose implementations are provided by the respective database vendors.

— Up to now, there are 180+ number of databases are existed, often this 180+ number of Drivers are available but all these drivers can be divided into the following four types.

① Type1 driver
② Type2 driver
③ Type3 driver
④ Type4 driver.

— Similar to JDBC API, Servlet is also an API that means servlet way also given in the form of an abstraction API. The implem

Servlet is an abstraction but implementation is provided by respective vendors

Implementation for these way provided by the different Server vendors like Tomcat, Jboss and weblogic. The servlet implementation way different from one server to another server.

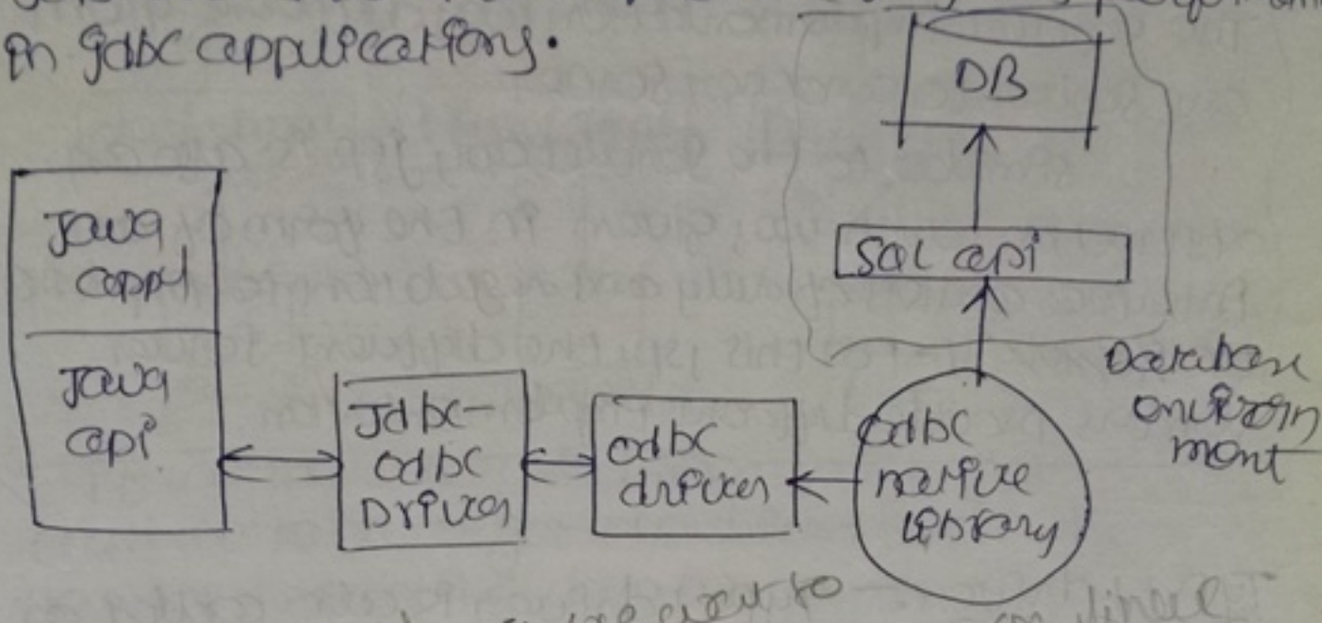
Similar to the servlet api, JSP is also an abstraction which was given in the form of an interface consists of rules and regulations to implement the ~~xxxxxx~~ jsp. For this jsp, the different server vendors provide different implementation.

Type-1 driver:- Type-1 driver is also called as JDBC-ODBC driver (or) bridge driver

- JDBC-ODBC driver is a reference implementation for driver interface provided by Sun microsystems in java technology as sun.jdbc.odbc.JdbcOdbc driver
- Sun micro systems has provided Jdbc-odbc driver with the inter dependency on the microsoft product odbc driver. class name
- Due to the above reason if we want to use type-1 driver in our jdbc applications then we must install the microsoft provided odbc native library.
- If we use jdbc-odbc driver in our jdbc application then this driver will take two times conversion (java to odbc, odbc to database representation so that jdbc-odbc driver is slower) any time

driver.

— Jdbc-odbc driver will provide very less performance in jdbc applications.



disadvantages) If we want to use jdbc driver on linux operating system we have to go with java.

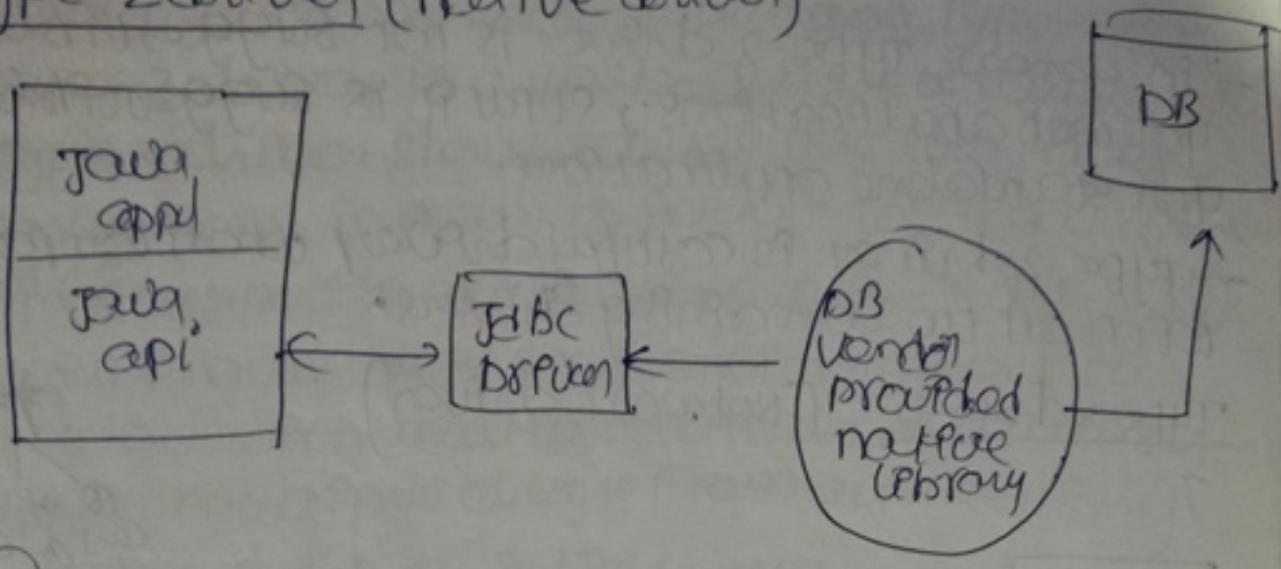
— JDBC-odbc driver is very good with microsoft products only that is with windows operating systems only so that Jdbc-odbc driver will provide less portability.

— In general Jdbc-odbc driver is not suggestible for web applications, it is suggestible for standalone applications.

— In general Jdbc-odbc driver is not suggestible for complex jdbc applications; it is suggestible for simple jdbc applications.

why? suppose if I use this driver in my web appl, then there may be chances that at a time 10 or more no. of requests will come to my appl. then in that situation, it is needed to create connections with the database separately for all the requests. For this purpose this driver slowly connect to the database and it do one by one in order.

Type-2 driver (native driver)



① Type-2 driver is also called as partly java, partly native driver (or) native driver

② Type-2 driver was implemented by using java implementations as well as the respective database provided native implementations.

(a) when compared with Type-1 driver, Type-2 driver is faster driver because it should not require two times conversions.

(b) when compared with Type-1 driver, Type-2 driver is more portable driver because it should not require Microsoft provided odbc driver native library.

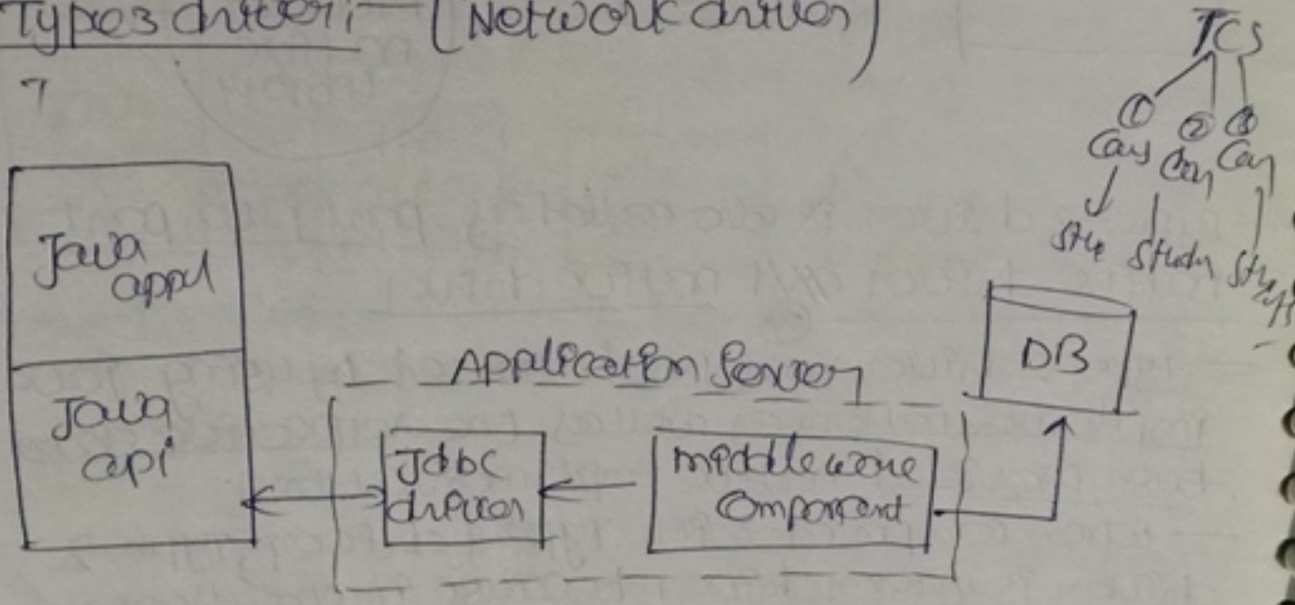
(c) Type-2 driver performance is very good when compared with Type-1 driver.

③ If we want to use Type-2 driver in our JDBC applications then we have to install the database vendor provided native library.

Disadv: Due to the above reason, Type-2 driver is not very good portable driver when compared with Type-3 and Type-4 drivers.

- In general Type-2 driver is not suggestible for web applications; still it is suggestible for standalone applications.
- Type-2 driver is costful driver when compared with all the remaining drivers.

Type3 driver :- [Network driver]

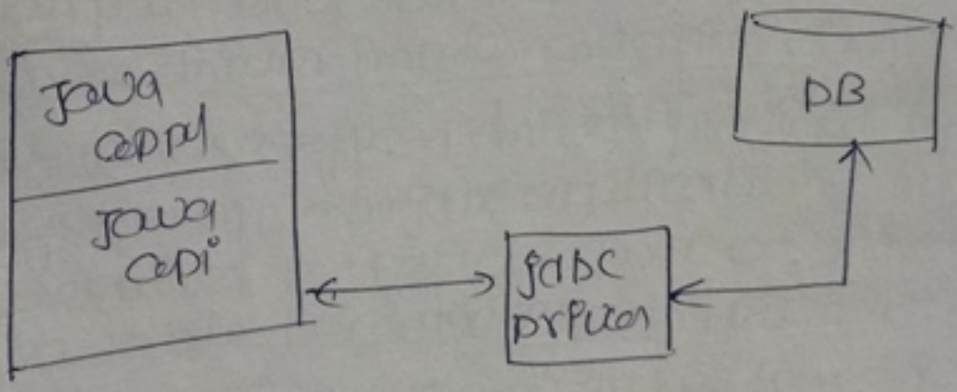


- Type3 driver is also called as middleware database access driver (or) network driver
  (why? it is called)
- Type-1 driver and Type-2 drivers are highly recommended for standalone applications but Type-3 driver highly recommended for web applications.
- When compared with Type-1 and Type-2 drivers Type-3 driver is more portable driver, because Type-3 driver should not require microsoft product (or) driver dependency and the respective database vendor provided native library.

- where compared with Type-1 and Type-2 drivers Type-3 driver is a faster driver because Type-3 driver should not require two times conversions in order to interact with database.
- In enterprise applications Type-3 driver will provide very good environment to interact with multiple number of databases from multiple clients [Java applications].
- Type-3 driver is highly recommended for web applications but not for standalone applic because it should require application server environment.

Due to the above reason Type-3 driver is not very good portable driver when compared with Type-4 driver.

Type-4 driver: (Thin driver) (or) (pure java driver)



— Type4 driver is also called as Thin driver (or) pure java driver.

— Type-4 driver was implemented purely on the basis of java technology.

— Type-4 driver is frequent used driver in application development.

— Type-4 driver will provide very good performance to the jdbc applications when compared with Type-1, Type-2, Type-3 drivers.

— Type-4 driver is recommended for any type of java free applications that is both standalone applications and enterprise applications.

Steps to design first jdbc application :-

Step①: Load and register the driver.

Step②: establish the connection b/w java application and database.

Step③: prepare either statement or preparedStatement or callableStatement objects.

Step④: write and execute sql queries.

Step⑤: close the connection.

Load and register driver :- In general Driver interface was provided by sun microsystems but whose implementation classes were provided by all the database vendors as part of their database software. In jdbc applications before going to load and register the driver first we have to make available driver implementation to our jdbc applications, we have to set class path environment variable to the location where driver implementation class is available.

If we use type-1 driver provided by sun micro systems in our jdbc applications then it is not required to set class path environment variable because Type-1 driver was provided by sun microsystems in the same java predefined library.

To load and register any driver class to the memory we have to use the following method from class "Class".

```
public static Class forName(String className)
```

where forName method is a factory method from class "Class", it can be used to load the specified class bytecode to the memory, analyse the metadata of the respective class and return class "class" object.

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

when JVM encounters the above instruction, JVM will pickup the specified driver class name, search for its .class file at the current location, if it is not available search for it in java predefined library, still if it is not available JVM will search for it in the classpath environment variable configured locations.

— JVM will identify com.mysql.jdbc.Driver.class file in the java predefined library then JVM will load its bytecode to the memory. At the time of loading driver class bytecode to the memory, a static block will be executed. As part of this static block execution, registerDriver() method will be executed.

By the execution of registerDriver() method, the specified driver will be registered

o Jdbc application.

```
class clay
{
static clay forName(stg driver)
{
=
}
```

what is forName()
what is load & register
prototype of forName is it static?

```
class.forName("oracle.jdbc.driver.oracleDriver");
class.forName("com.mysql.jdbc.Driver");
```

(2)

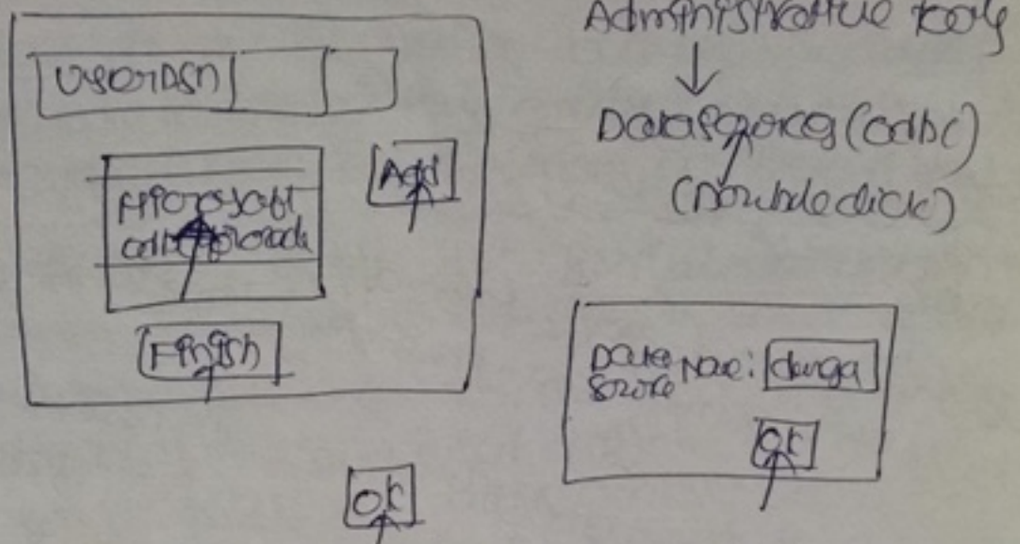
(1)

```
sun.jdbc.odbc.JdbcOdbcDriver d=new sun.jdbc.odbc.JdbcOdbcDriver();
d.registerDriver();

oracle.jdbc.driver.oracleDriver d = new oracle.jdbc.driver.oracleDriver();
d.registerDriver();

com.mysql.jdbc.Driver d=new com.mysql.jdbc.Driver();
d.registerDriver();
```

— In case of type-1 driver, before loading driver to the memory first we have to configure microsoft product odbc driver. To configure odbc driver we have to use the following path.

Start → control panel → System and Security → Administrative tools → Data Sources (odbc) (Double click)

Windows 7



→ If we use oracle 10:xe, If we use jdbc 4.0 version or jdk 6.0 version or above then loading and registration of type-1 driver is optional.

(ii) In case of oracle 11:xe, jdbc 4.0 version or jdk 6.0 and above then loading type-4 driver provided by oracle is optional.

In the above case, jvm will load and register the specified driver at the time of establish the connection b/w java application and data base.

(2) what about jdbc 4.0 or 6.0 → type-1 driver → oracle 11xe ?

(3) jdbc 4.0 or 6.0 → type 4 driver → oracle 10xg

— To design JDBC application, we have to use some predefined library provided by Java technology in the form of java.sql package so that we have to import java.sql package into the present Java file.

Here java.sql package provides the following predefined library

① Driver (Interface)
② DriverManager (class)
③ Connection (Interface)
④ Statement (Interface)
⑤ PreparedStatement (Interface)
⑥ CallableStatement (Interface)
⑦ ResultSet (Interface)
⑧ ResultSetMetaData (Interface)
⑨ DatabaseMetaData (Interface)

class DriverManager
{
public static Connection getConnection(-,-,-) throws SQLException
{
==
}
}

interface Connection
{
Statement createStatement();
PreparedStatement prepareStatement();
CallableStatement prepareCall();
}

interface Statement
{
public ResultSet executeQuery(-);
public int executeUpdate(-);
public boolean execute(-);
}

2. establish the connection b/w java application & Database
In jdbc applications, after registering the driver we have to establish the connection b/w java application & database. For this we have to use the following method from DriverManager class.

public static Connection
getConnection(String driver_url, String db_user_name, String db_password) throws SQLException

Ex:-
```
Connection con = DriverManager.
        getConnection("jdbc:odbc:dsn", "system", "durga");
```
(msg)

where getConnection is a static method from DriverManager class. It will take the following three parameters to establish the connection.

dsn → database
"system" → database username
"durga" → database password

① Driver-url :-
② Database user name
③ Database password.

— In general from driver to driver driver class name and driver url will be varied. In general all the drivers url should be available in the following format,

protocol : sep protocol : database name
(data source name)

where main protocol should be fixed for each and every driver. i.e jdbc.

— when JVM encounters DriverManager.getConnection method then JVM will execute getConnection() method, as part of getConnection method implementation JVM will access connect() method. By the invocation execution of connect() method only, a virtual socket connection will be established between java application and database. As a result getConnection() method will return Connection object.

Ans: In general in java technology it is not possible to create objects for the interfaces but how getConnection method will return Connection object?

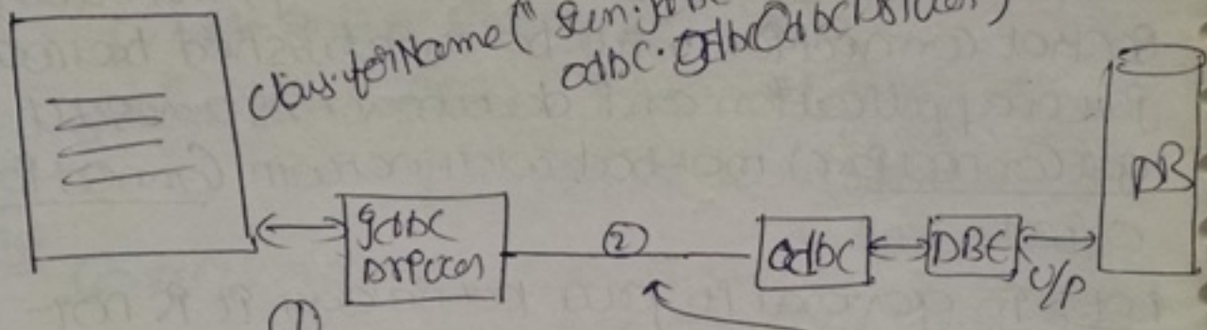
Ans: jdbc is an abstraction, that is collection of interfaces provided by sun microsystems and whose implementations are provided by the respective database vendors.

— Connection is an interface provided by java technology as part of jdbc abstraction and whose implementation class was provided by the respective database vendors as part of jdbc abstraction implementation.

In jdbc application, when JVM encounters getConnection() method then it will establish the connection b/w java application and database and return connection interfaces implementation

Class object from getConnection() method.

Note: while implementing JDBC abstraction, Sun micro systems has given a suggestion like to take (consider) anonymous inner classes to implement the interfaces like Connection, Statement and so on — but no database vendor has followed.



Connection con = DriverManager.getConnection ("jdbc:odbc:reg", "system", "durga");

To connect Type 4 driver

DriverManager.getConnection ("jdbc:oracle:thin@localhost:1521:xe");

DriverManager.getConnection ("jdbc

(3) Create Statement (or) prepared Statement (or) Callable Statement as per application required

—In JDBC applications after establish the connection, we have to write and execute the SQL queries. To write and execute SQL queries we need some predefined library provided by JDBC API, Here the required predefined library was provided in the form of Statement, prepared Statement and callable Statement so that before execute the SQL queries, we have to create either statement object or prepared Statement object or callable statement object.

Int What are the differences b/w Statement, prepared Statement and callable Statement

Ans In JDBC applications, when we have a requirement to execute all the queries (SQL) independently there we have to use Statement. In JDBC applications when we have a requirement to execute the same SQL query in the next sequence, where to improve the performance of JDBC applications we have to use prepared Statement. In JDBC applications when we have a requirement to access stored procedures and functions available at databases (SQL) from applications there we only use Callable Statement.

— To create Statement Object in JDBC application we will use the following method from Connection

public Statement createStatement() throws SQLException

Ex: Statement st = Con.createStatement();

When JVM encounters createStatement() method then createStatement() method will return Statement Object that is Statement interfaces implementation class object.

Step(4): write and execute the SQL queries ✓

To write and execute the SQL queries in JDBC applications we will use the following methods from Statement.

① executeQuery(--)
② executeUpdate(--)
③ execute(--)

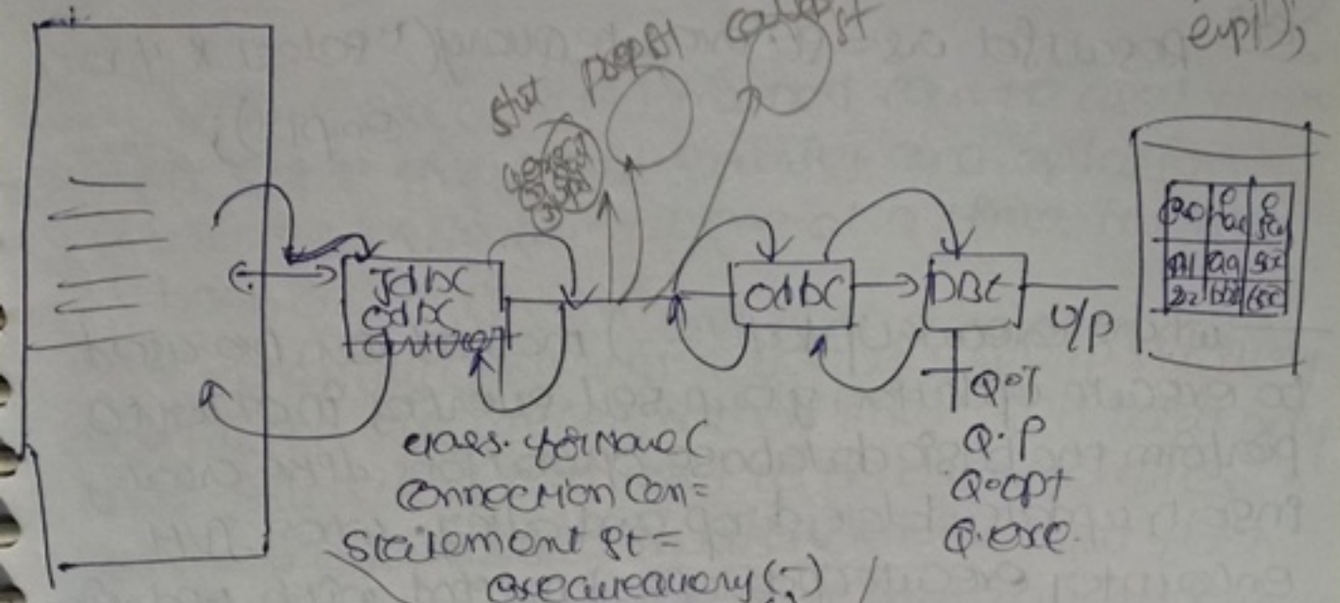
— what is the diff b/w executeQuery() method executeUpdate() method & execute() method.

Ans: where executeQuery() method can be used to execute selection group SQL queries, procedures to fetch the data from database.

When JVM encounters executeQuery(--) method with selection group SQL query then JVM will pick up that selection group SQL query and send to database engine through JDBC Connection and driver.

Date: 3rd May, 2012 Thurs

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



At database, database Engine will perform that selection of SQL query and execute it by performing query Tokenization, query parsing, query optimization and query execution. By the execution of selection group sql query database engine will fetch the data from database table and send back to java application to the same JDBC driver and connection.

Drawn

As java technology is an object oriented prog. language, it will store the fetched data in the form of an object at heap memory. Here the object which we used to store fetched data is called as ResultSet.

After getting ResultSet object JVM will return the generated ResultSet object reference as per the predefined implementation of executeQuery() method.

public ResultSet executeQuery(String query)
throws SQLException

Eg: ResultSet rs = st.executeQuery("select * from emp1");

— where executeUpdate(--) method can be used to execute updation group sql queries in order to perform the basic database operations like create, insert, update, delete, drop and alter. when JVM encounters executeUpdate(--) method, with updation group sql query then JVM will pickup that updation groupsql query and send to the database engine through JDBC connection and driver.

At database, Database engine will pickup that SQL query, execute it, perform the updations on database table, identify rowCount value (the no. of records got effected with the provided sql query) and send that rowCount value back to java application.

After getting rowCount value from java application JVM will return the rowCount value as return value from executeUpdate() method as per the predefined implementation of executeUpdate() method.

public int executeUpdate(String query) throws
SQLException

Eg:8 int rowCount = st.executeUpdate("update emp1 set esal = esal+500 where esal < 10000");

→ where execute(-) method can be used to execute both select group sql queries and updation group sql queries but one at a time but not both at a time.

→ when JVM encounter selection group sql query along with execute(-) method then JVM will pickup that sql query and send to database engine, where at database, Database engine will fetch the data from database and send back to java application.

At java application, the fetched data will be stored in the form of ResultSet objects but as per the predefined implementation of execute(-) method, JVM will return true as a boolean value from execute method.

→ when JVM encounter update group sql group along with execute method then JVM will send that sql query to Database engine. At database, Data base engine will perform updations on database and return rowCount value to java application. But as per the predefined implementation of execute(-) method JVM will return false as the boolean value from execute method.

public boolean execute(String query) throws SQLException

Ex:-

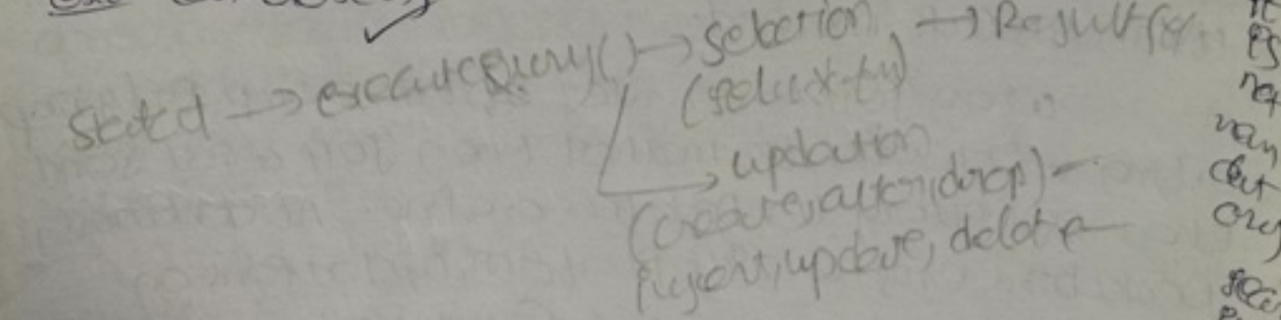
```
boolean b1 = st.execute("select * from empl");
boolean b2 = st.execute("update empl set
esal = esal+500 where esal<10000");
System.out.println(b1); // true
System.out.println(b2); // false
```

Step(5) :- close the connection.

In JDBC applications, it is convention to close the connection at the end of the JDBC application. To terminate the connection in JDBC applications we will use the following method from Connection

public void close() throws SQLException

Ex:- con.close();



It is not necessary but convention

— The following example demonstrate how to prepare a table on database in a jdbc application by taking table name as the dynamic input.

— Before do this application, check whether the database table empl is existed or not on the database.

SQL> connect to system/durga
SQL> select * from empl;

ENO	Ename	Sal
|	|	:

SQL> desc empl;

ENO —
Ename —
esal —

SQL> drop table empl;
Table dropped.

SQL> select * from empl;
no empl table object was there in the database

executeQuery
executeUpdate
execute

create, altering
update
insert
delete (udy)

write queries for

② write a pgm to connect to oracle like and mysql ✓

③ write the same pgm for Type4 driver.
↳ for oracle 10g
oracle 11g

↳ create
alter
drop
update
insert
delete

④ write the same pgm for Type4 for mysql ✓

executeQuery
executeUpdate
execute()

```
// Import section.
import java.sql.*;
import java.io.*;
Public class JdbcApp1 { psvm(SC) a) throws Exception,
                        {
// Load and Register Driver.
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//establish the connection.
Connection con = DriverManager.getConnection
        ("jdbc:odbc:nag", "system", "dinga");
// prepare statement
Statement st = con.createStatement();
// preparing BufferedReader
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter table name");
  String tname = br.readLine();
// write SQL query
String query = "create table " + tname + "(eno number,
ename varchar2(5),
esal number)";
// execute the SQL query.
        st.executeUpdate(query);
System.out.println("Table " + tname + " was
        created successfully");
// close the connection.
    con.close();
}}
```

space

no space

Compilation:

D:\JDBC10> javac JDBCApp1.java

D:\JDBC10> java JDBCApp1

enter table name

: emp1

table emp1 was created successfully.

— Now, open the database oracle, and check whether the table was created or not:

SQL> (Connect to database) by entering username and password.

SQL> select * from emp1;

eno —

ename —

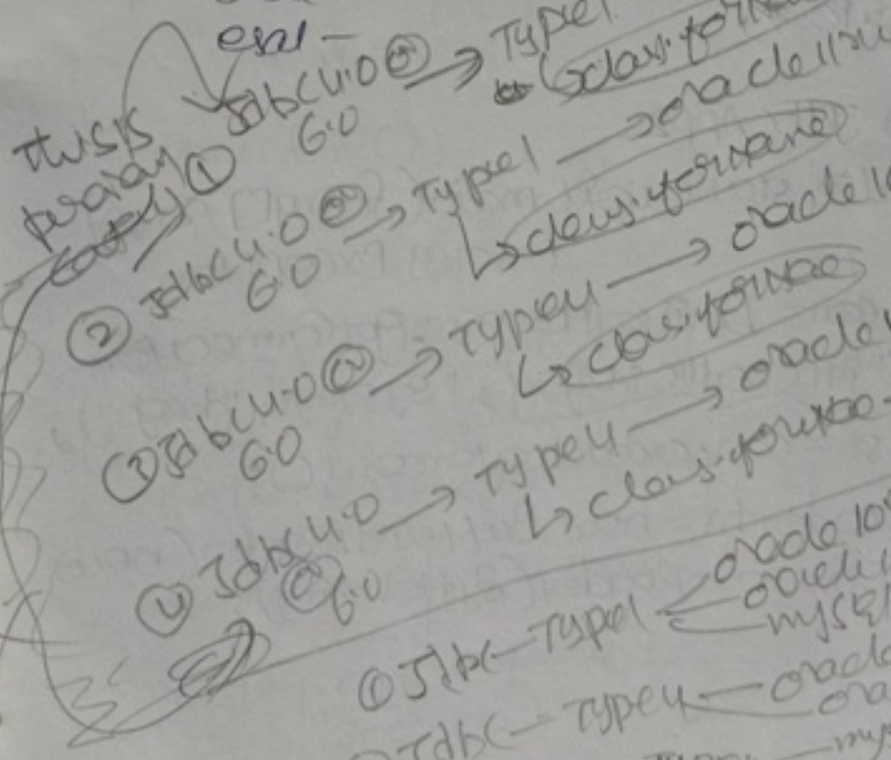
esal —

SQL> desc emp1;

eno —

ename —

esal —



Description: app(2) Date: 4th May, 2012 Fri

The following example demonstrates how to insert number of records into the database table from a JDBC application by taking records data as dynamic input.

C:\Users> sqlplus

Enter user name: System
password: durga

SQL> select * from empl;
no rows selected.

SQL> desc empl;
eno
ename
esal

```
import java.sql.*;
import java.io.*;

public class JdbcApp2
{
    public static void main(String[] args)
                                throws Exception
    {
Connection con = DriverManager.getConnection
        ("jdbc:odbc:vijay", "system", "durga");
Statement st = con.createStatement();
BufferedReader br = new BufferedReader(new
    InputStreamReader(System.in));
```

```
while(true)
{
System.out.println("enter employee number");
int eno = Integer.parseInt(br.readLine());
sop("enter employee name");
String ename = br.readLine();
sop("enter employee salary");
float esal = Float.parseFloat(br.readLine());
st.executeUpdate("insert into empl values("+
eno+","+ename+","+esal+")");
sop("do you want to enter one more record [y/n]?");
String option = br.readLine();
if(option.equals("n"))
    break;
}
con.close();
}
}
```

D:\jdbc10>javac JDBCApp2.java

D:\jdbc10>java JDBCApp2

Enter employee number
111
enter employee name
aaa
enter employee salary
7000.00
do you want to enter one more record y/n? y

— [illegible]

Enter employee number
222↵
Enter employee name
bbb↵
Enter employee salary
8000.00↵
Do you want to enter one more record y/n? y↵
enter employee no.
333↵
Enter employee name
ccc↵
Enter employee salary
9000.00↵
do u want to enter one more y/n?
O/P/SADCLO> n↵

Now, open oracle database and whether the data of records were inserted or not?

SQL> select * from emp1;

eno	ename	esal
111	aaa	9000.00
222	bbb	8000.00
333	ccc	9000.00

— Before executing the above application, check whether in the database whether empl table is present or not. If it is present in database then there is no problem, if it is not present then create in the database.

In this pgm
used java version — JDK 1.7
used database — oracle 11xe
used driver — sun.jdbc.odbc.JdbcOdbcDriver

— Just observe, in this program, we were not used class.forName statement to load the driver. In other terms we weren't (missed out) the load and register the driver step. So in this context, how JVM will access the and how can jdbc application interact with database without loading and register driver?

Ans If we use java 1.6 or higher versions along with oracle 11xe, then it is optional to specify the

insert into empl values(111, 'aaa', 5000)
.executeUpdate("insert into empl values("+

load and register the driver step. In this context, if we didn't specify the driver class name, automatically JVM loads and registers the JdbcOdbcDriver class present in Sun.jdbc.odbc folder. Here JVM will recognize, which type of type of driver to be load, will get the clarity from connection statement. That means jdbc.odbc.Mag. By seeing odbc JVM will load and register JdbcOdbcDriver before performing connection operation.

Note: If you are using windows7 operating system, then make sure it will be a 32-bit operating system with the version name ultimate only, before executing the jdbc applications in your machine. The other versions like home edition, professional, basic were not supporting Java correctly.
But we don't have any problem with XP.

no need to load explicitly by using driver class in PBM we not load explicitly from JDK 6.0 onwards use appropriate to specify

— If you want to use Type4 driver provided by oracle
then we have to use the following Driver class name and driver url.

driver_class:

oracle.jdbc.driver.OracleDriver
↓ folder ↓ folder ↳ class name present in Those jar file

driver url:

jdbc:oracle:thin@localhost:1521:xe
↓ main protocol ↓ sub protocol ↓ dsn

— where oracle driver class was provided by oracle software in the form of oracle software organization

1) oracle 8i/9i — [classes12.jar / classes14.jar]
2) oracle 10xe — ojdbc14.jar
3) oracle 11xe — ojdbc5.jar or ojdbc6.jar

— If you want to use oracle driver in our jdbc applications then we have to set classpath environment variable to the location, where the respective driver jar file is available.
— oracle software has provided the respective driver jar files in the following location at the respective oracle software. (In case of oracle 11xe)

c:\oracle\exe\app\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar

— The following example demonstrates, how to perform updations on a particular database table from jdbc application.

[upto now we created a table, insert records into table. Now, in this program

class path of log4j jar ✓

oracle 11xe ↓ Type 4 driver. So

In this pgm we use oracle 11xe java 7 so/ (we) oracle specified Type 4 Driver

```
import java.util.*;
// import java.io.*;
import java.sql.*;
public class JdbcApp3
{
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", "durga");
Statement st = con.createStatement();
Scanner sc = new Scanner(System.in);
{ int b_Amt = sc.nextInt(); sop("enter bonus amt");
  float sal_Range = sc.nextFloat(); sop("enter salary range");
int rowCount = st.executeUpdate("update empl
set esal=esal+" + b_Amt + " where esal<" +
                                  sal_Range);
sop("Records updated--" + rowCount);
con.close();
}
}
```

↳ save this as JdbcApp3.java

D:\JDBC10> javac JDBCApp3.java

D:\JDBC10> java JDBCApp3 ↵

Exception in thread main: class NotFoundException

D:\JDBC10> set classpath = %classpath%;C:\oracle\ora92\app\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar; ↵

↳ paste this path from cmp path

D:\javac JDBCApp3.java ↵

D:\java JDBCApp3 ↵

enter Bonus amount

500 ↵

enter Salary range

10000 ↵

records updated --- 3

D:\JDBC10>

(5) ↳ In SQL But not in Emp proj

To execute the above application, we have to set the classpath environment variable to ojdbc6.jar (oracle 11xe).

— The following example demonstrates how to delete the number of records from database table through a jdbc application.

in this prgm we did not use class.forName. It is optional to use this in case of oracle xe jdk 6.0 version. Here oracle xe was used

```
import java.io.*;
import java.sql.*;
public class JdbcAppl
{
public static void main(String args[]) throws
                                        Exception
{
Connection con=DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system",
                                        "dunga");
Statement st=con.createStatement();
  Console c=System.console();
float salRange=float.parseFloat(c.readline
        ("enter salary Range"));
int rowCount=st.executeUpdate("delete empl
```

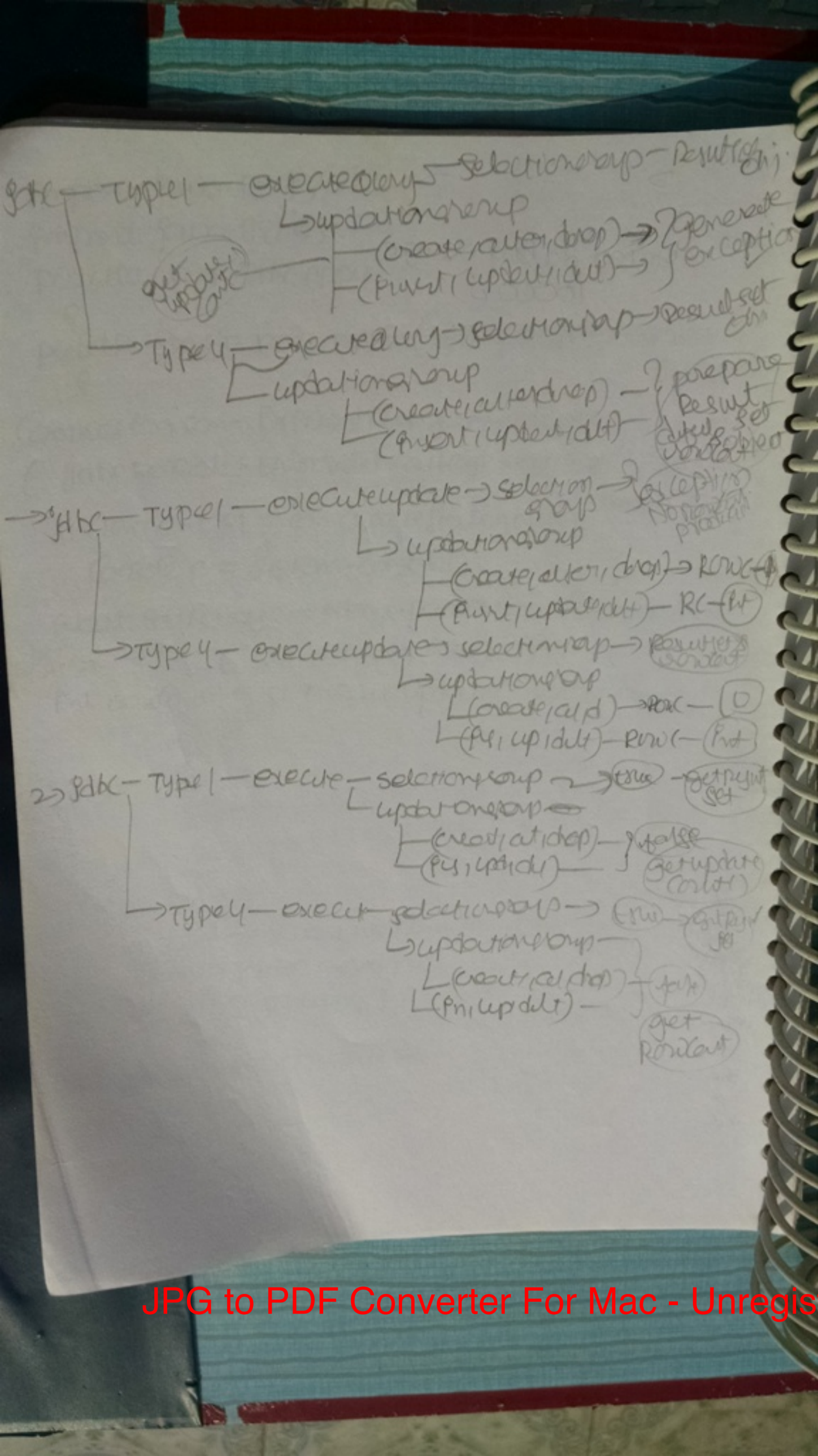
↳ save this pgm

D:\jdbc10>javac appl.java

D:\jdbc10>java appl

exc

D:\jdbc10>set classpath=


JPG to PDF Converter For Mac - Unregis

Int. ques: In jdbc applications, if we execute the sql queries like insert, update, delete and so on dml [data manipulation language] queries then executeUpdate() method will return rowCount value on the basis of the number of records which are effected with the provided sql query.

— In Jdbc applications, if we execute DDL SQL queries like create, alter, drop by using executeUpdate() method then database engine will not perform manipulations upon the record then which return value return by executeUpdate() method? [Data Definition language]

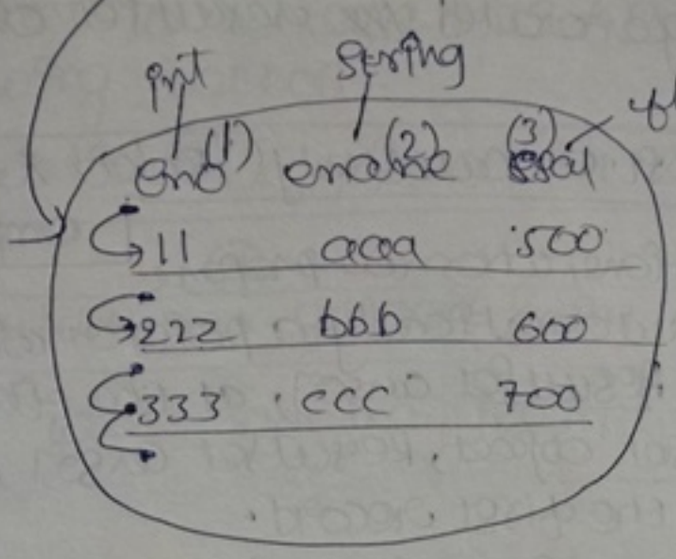
Ans: If we execute DDL sql queries by using execute update method in Jdbc applications then execute update() method will return an integer value on the basis of the driver which are used. In the above context, if we use type1 driver provided by sunmicrosystems then executeUpdate() method will return (1) as rowCount ✓

In the above context if we use type4 driver provided by oracle then execute update method will return (0) as rowCount value.

ex:- import java.sql.*;
public class JdbcApp5
{ psvm(String args[]) throws Exception
{
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe", "system",
"durga");
Statement st = con.createStatement();
int rowCount1 = st.executeUpdate("create table
emp2(eno number)");
int rowCount2 = st.executeUpdate("drop table emp2");
sop(rowCount1);
sop(rowCount2);
con.close();
}
}

op:-

ResultSet rs = st.executeQuery("select * from emp1");



— In jdbc applications, if we execute selection query execute query then database engine will fetch the data from database and send back to java application.

— At java application the fetched data will be stored in the form of an object called as Resultset object.

Ex: interface ResultSet
{
public boolean next();
public xxx getXxx(int columnIndex);
public xxx getXxx(String columnName);

— If we use executeQuery() method to execute selection group sql query then executeQuery() method will return the generated ~~the~~ resultset object reference.

```
ResultSet rs = st.executeQuery("select * from emp1");
```

— In jdbc applications, when we prepare ResultSet object automatically a pointer will be created called as ResultSet cursor, at the time of creating ResultSet object, ResultSet cursor will be positioned before the first record.

— After getting ResultSet object if we want to read the data then we have to use the following steps

Step1: check whether the next record is available or not with respect to ResultSet cursor, if next record is available then move resultset cursor to next record position.

To perform the above action we have to use the following method from ResultSet

```
public boolean next()
```

where next() method is a boolean method, it will return true if it identify next record, it will return false if it is not identify next record.

Ex: `boolean b = rs.next()`

— After getting ResultSet object to a particular record position read the data from the respective columns. To achieve this we have to use the following method

public xxx getXxx(int column_index)

(or)

public xxx getXxx(String column_name)

where xxx may be byte, short, int, float, double, String

ex:

```
int eno=rs.getInt(1);
String ename=rs.getString("ENAME");
```

— The following example demonstrates how to retrieve the data from database through ResultSet object.

```
import java.sql.*;
public class JdbcApp6
{ public static void main(String[] args) throws Exception
  {
oracle.jdbc.driver.OracleDriver driver=new oracle.jdbc.driver.OracleDriver();
```

① fully qualified name of the class. If you don't want this, mention the path in the form of import statement also.

After the

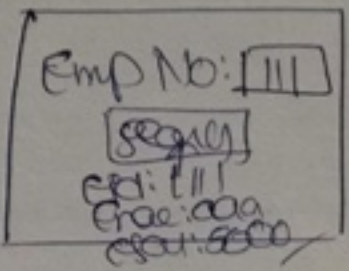
```
Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe", "system",
Statement st = con.createStatement(); "durga");
ResultSet rs = st.executeQuery("select * from
    sop("ENO ENAME ESAL"); emp1");
    sop("_____");
    while(rs.next()).
        { sop(rs.getInt(1)+" "+rs.getString(2)+
          "  "+rs.getFloat(3));
        }
        con.close();
    }
}
```

O/P >javac
>java

ENO	ENAME	ESAL
3333	CCC	7000
4444	ddd	8000
1111		
2222		

Qn: write a jdbc application to display a particular table data in the form of an html page with table tag.

Qn ② awt application.



Date: 5th May 2012 Sat

Ans:

sqlplus

user-name: system/durga

SQL> select * from product;

pid	pname	pcost
P1	aaa	500
P2	bbb	600
|	|	|

— open editplus:

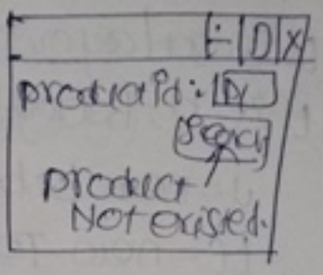
```
import java.io.*;
import java.sql.*;
public class JdbcApp7
{
    public static void main(String[] args) throws Exception
    {
class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system",
                    "durga");
    Statement st=con.createStatement();
    ResultSet rs=st.executeQuery("select * from
                    product");
```

String to byte represent

```
String data="";
  data=data+"<html><body bgcolor='light
  <center><br><br>";                    green'>
  data=data+"<table border='1' bgcolor='light
data=data+"<tr><td>PID</td>                yellow'>";
  <td>PNAME</td><td>PCOST</td></tr>";
  while(rs.next())
      { data=data+"<tr>";
data=data+"<td>"+rs.getString("pid")+"</td>
<td>"+rs.getString("pname")+"</td><td>"+
rs.getInt("pcost")+"</td>";
        data=data+"</tr>";
    }
  data=data+"</table></body></html>";
FileOutputStream fos=new FileOutputStream(
                                "product.html");
byte[] b=data.getBytes();
      fos.write(b);
  System.out.println("open product.html file");
      fos.close();
      con.close();
  }
}
```

— The following example demonstrate how to retrieve the data from database through resultset object and how to display that retrieved data in a frame (awt frame).

mysql




```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
class SearchFrame extends Frame implements
                                    ActionListener
  {
    Label l;
    TextField tf;
    Button b;
    Connection con;
    Statement st;
    ResultSet rs;
    SearchFrame()
    {
      try
      {
```

```
this.setVisible(true);
this.setSize(500,500);
this.setTitle("JDBC-ACCT APPLICATION");
this.setLayout(new FlowLayout());
this.setBackground(Color.pink);
    l=new Label("product id");
    tf=new TextField(15);
    b=new Button("search");
    b.addActionListener(this);
      this.add(l);
      this.add(tf);
      this.add(b);
Class.forName("oracle.jdbc.driver.OracleDriver");
con=DriverManager.getConnection("jdbc:oracle:
thin@localhost:1521:xe","system","dungali");
      st=con.createStatement();
  }
  catch(Exception e)
    { e.printStackTrace();
    }
}
public void actionPerformed(ActionEvent ae)
    { try
      {
rs=st.executeQuery("select * from
product where pid="+tf.getText()+"");
        repaint();
```

```
}
catch(Exception e)
    { e.printStackTrace();
    }
}
public void paint(Graphics g)
    { try
      {
Font f= new Font("arial", Font.BOLD,30);
        g.setFont(f);
      boolean b=rs.next();
        if(b==true)
          {
            g.drawString("product Id--"+
rs.getString(1), 50, 100);
g.drawString("product Name--"+rs.getString(2),
                                    50, 150);
g.drawString("product Cost--"+
            rs.getInt(3),50,200);
    }
  else
  { g.drawString("product Not existed",50,150);
  }
}catch(exception e)
    {   e.printStackTrace();
    }
  }
}
```

```
class JdbcApp8
{
public static void main(String[] args)
{
SearchFrame sf=new SearchFrame();
}
}
```

Execution:

— In JDBC applications, if we pass updation group parameters to executeQuery() method then what will be result in JDBC application?

Ans: In JDBC applications we will use execute method to execute selection group SQL query() query in order to fetch the data from database.

If we provide updation group SQL query as parameter to executeQuery() method then JVM will pick up that SQL query and send to database engine.

At database, Database engine will execute updation group SQL query, perform updations on database, identify row count and send back to Java application. But as per the predefined implementation of executeQuery() method, JVM will expect ResultSet object.

```
ResultSet rs = st.executeQuery("update emp set sal
             = sal + 500 where sal < 50000");
```

— In the above ambiguity situation, generating an exception or not is completely depending on the type of driver which we used in our JDBC applications.

In the above situation if we use type-1 driver provided by sun microsystems then JVM will raise an exception like java.sql.SQLException: no resultset may be produced

In the above situation if we use type-4 driver provided by oracle then JVM will not raise any exception but JVM will prepare a default resultset object. and return that

— In the case of Type-1 driver, In the above situation if we want to get the generated rowCount value in catch block, we have to use the following method.

public int getUpdateCount() throws SQLException

(↳ this method work only in the case of type-1 driver)

Mysql>

```
import java.sql.*;
public class JdbcApp9
    { public static void main(String[] args)
        {
    Statement st=null;
      try
      {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con= DriverManager.getConnection(
 "jdbc:odbc:nag","system","durga");
        st=con.createStatement();
ResultSet rs= st.executeQuery("update emp set
 sal=sal+500 where esal<10000");
  System.out.println("After Result set");
    }
  catch(Exception e)
        { try
          {
  int rowCount = st.getUpdateCount();
System.out.println("Row count -"+ rowCount);
    } //try-2
     catch(Exception e)
        { e.printStackTrace();
        }
    }
  }
}
```

Int. Question: If we provide selection group SQL query as parameter to executeUpdate() method then what will be the result. In JDBC application?

Ans: In general in JDBC applications, we will utilize, executeUpdate() method to execute updation group SQL queries in order to update the database tables.

— If we pass selection group SQL query as parameter to execute update() method then JVM will send that ~~updation~~ selection group SQL query to database engine where database engine will fetch the data from database and send back to Java application.

— At Java application, fetched data will be stored in the form of ResultSet object but as per the predefined implementation of executeUpdate() method JVM may expect zero count value.

In the above situation, raising an exception or not raising an exception is completely depending on the driver which we used in JDBC applications. In the above situation if we use type-II driver provided by Sun microsys then we will get the following exception. try

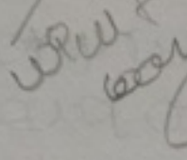
java.sql.SQLException: No row count was produced.

- In the above application if we use type-4 driver provided by oracle, then JVM will not raise any exception. It prepare result set object

- In the above context, if we want to get the generated ResultSet object reference we have to use the following method from Statement interface.

public ResultSet getResultSet() throws SQLException

- case with Jdbc-odbc driver



```
import java.sql.*;
public class JdbcAppl0
  { public static void main(String[] args)
  Statement st=null;                    {
    try
    {
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection
("jdbc:oracle:thin@localhost:1521:xe","System",
  st=con.createStatement();              "durga");
 int coroCount= st.executeUpdate("select * from
                                  emp");
 ResultSet rs= st.getResultSet();
System.out.println("END ENAME EMAIL");
  sop("_ _ _ _ _ _");
     while(rs.next())
       {
sop(rs.getInt(1)+" "+rs.getString(2)+" "+
                                  rs.getFloat(3));
     }
   }//try
   catch(Exception e)
        { e.printStackTrace();
        }
    }
}
```

MySQL database

Date: 8th May, 2012 Mon

product from sun microsystems

user : root
password : root ✓

mysql> show databases;

Database

=

durgadb

=

mysql> drop database durgadb;
Query OK. 2 rows affected.
mysql> commit;
mysql> show databases;

Database

information_schema
mysql
test

mysql> create database durgadb;
mysql> show databases;

Database

=

durgadb

=

(now I am at outside of all databases. I want to enter into perticular database and use it)

mysql> use durgadb;
Database changed.

mysql> show tables;
empty set (0.01 sec)

mysql> create table empl(eno int, ename char(5), esal float); ↵

(observe not number) ↑ int; not varchar2 ↑ char(5)

i want to see the created table.

mysql> show tables;

Tables in durgadb
empl

mysql> desc empl;

Field	Type	null	key
eno			
ename			
esal			

if you use ide's in java, then it must be declare a primary key in a table. But our jdbc application will allow tables without primary key declaration.

mysql> insert into empl values(111, "aaa", 5000);
query ok.

observe it is not 'aaa'

mysql> commit;
mysql> select * from empl;

eno	ename	esal
111	aaa	5000

now I want to update the table.

mysql> update empl set esal=esal+500 where esal<10000;

mysql> commit;

mysql> select * from empl;

eno	ena	esal
111	aaa	5500

mysql> (I want to delete the records in mysql database)

mysql> delete from empl where esal<10000;

↑ it is mandatory in mysql. But it is optional in oracle.

mysql> select * from empl;

empty set

mysql> drop table empl;

mysql> commit;

mysql> show tables;

empty set (0.00 sec)

JDBC applications with

— If we want to use mysql database as back-end in our JDBC applications then we have to use the following driver class name and url.

driver-class: com.mysql.jdbc.Driver. ← class name

url: jdbc:mysql://localhost:3306/db. ← database name

where com.mysql.jdbc.Driver is the driver implementation class provided by mysql in the form of the following jar file.

mysql-connector-java-3.1.6-bin.jar

If we want to use type-4 driver provided by mysql in our JDBC applications, we have to set classpath environment variable to the above jar file.

execute method:— In JDBC applications, execute() method can be used to execute both selection group and updation group SQL queries. If we use execute method to execute selection group SQL query then JVM will send that selection group SQL query to the database engine where database engine will execute it, fetch the data from database and return to Java application. At Java application that fetched data will be stored in the form of ResultSet object. But execute() method will return true as the boolean value. In the above context, to get the data from ResultSet object, first we

driver is not available with mysql database it throws not

need to get ResultSet object reference, for this we have to use the following method from Statement

public ResultSet getResultSet() throws SQLException

In database:

mysql> create table empl(eno int, ename char(5), esal float);

> insert into empl values(111, 'aaa', 5000);

> insert into empl values(222, 'bbb', 6000);

↑ observe single quotes are also valid here

```
import java.sql.*;
public class JdbcApp11
{
public static void main(String[] args) throws Exception
{
/* Load and register the driver is mandatory in the case of mysql database used by backend but this step is optional in the case of oracle */
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/durgadb", "root", "root");
```

jdbc: mysql: // localhost: 3306 / durgadb

main protocol — subprotocol — database name in mysql database

```
Statement st = con.createStatement();
boolean b = st.execute("select * from emp1");
System.out.println(b);
ResultSet rs = st.getResultSet();
SOP("ENO ENAME ESAL");
SOP("-------------------");
while(rs.next())
{
SOP(rs.getInt("eno")+"  "+rs.getString("ename")+
"  "+rs.getFloat("esal"));
}
con.close();
}
}
```

Cmd

```
D:\jdbc10>javac JdbcApp10.java.
D:\          >java JdbcApp10 J
Exception in thread main class NotFound.
```

```
D:\          >set classpath=%classpath%;D:\sqlite\sqlite-jdbc-3.7.2.jar;
```

```
          >java J
          true
ENO  ENAME  ESAL
111  aaa    5000.0
222  bbb    6000.0
```

Set classpath=%classpath%;D:\sqlite\sqlite-jdbc-3.7.2.jar; in cmd prompt

In jdbc applications, if we use updation group sql queries along with execute method, then updation group sql query will be send to database engine where at database, database engine will perform updations on database table identify rowcount and send that rowcount value to java application. But as per predefined implementation of execute method, JVM will return false alternatively as the boolean value.

— In the above context to get the generated row count value we have to use the following method from Statement

public int getUpdateCount() throws SQLException

— Before database mysql

Is it work in for type1, 2 type 4 driver?

What about the case with selection group of sql query with execute method here, is getResultSet() method work or not?

```
import java.sql.*;
public class JdbcApp12
{
public static void main(String[] args) throws
                                          Exception
{ class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection
("jdbc:mysql://localhost:3306/durgadb", "root",
Statement st = con.createStatement();        "root");
boolean b=st.execute("update emp1 set esal=
esal+500 where esal<10000");
        System.out.println(b);
int rowCount=st.getUpdateCount();
sop("Records updated"+rowCount);
            con.close();
    }
}
```

Cmd

```
D:\jdbc\0>javac jdbc12.java
          > java JdbcApp12 ↵
        false
        records updated
```

Same whole per class name

ResultSet Types

Date: 8th May, 2012 Tue

- In JDBC ResultSets can be classified into the following two ways.

① As per the ResultSet concurrency and performance there are two types of resultsets.

(i) Readonly Resultset ✓
(ii) Updatable Resultset ✓

Readonly resultset object: It is the resultset object, it will allow the users to read the data, it will not allow to perform updations. To represent this resultset object ResultSet interface has provided the following constant.

```
public static final int CONCUR_READ_ONLY
```

Updatable resultset: It is a ResultSet object, it will allow the users to perform updations on its content. To represent this resultset object ResultSet interface has provided the following constant.

```
public static final int CONCUR_UPDATABLE
```

② As per the resultsets cursor movements there are two types of resultsets.

(i) Forward only resultset.
(ii) Scrollable resultset.

Forwardonly Resultset: It is a resultset object, it will allow the users to iterate the data in only forward direction. To represent this resultset object, Resultset interface has provided the following constant.

```
public static final int TYPE_FORWARD_ONLY
```

Scrollable Resultset:

It is a resultset object, it will allow the users to iterate the data in both forward and backward directions.

There are two types of Scrollable Resultsets.

① Scrollsensitive resultset
② Scroll insensitive resultset

Int. Qus: What is the difference between Scroll sensitive resultset object and scroll insensitive resultset object.

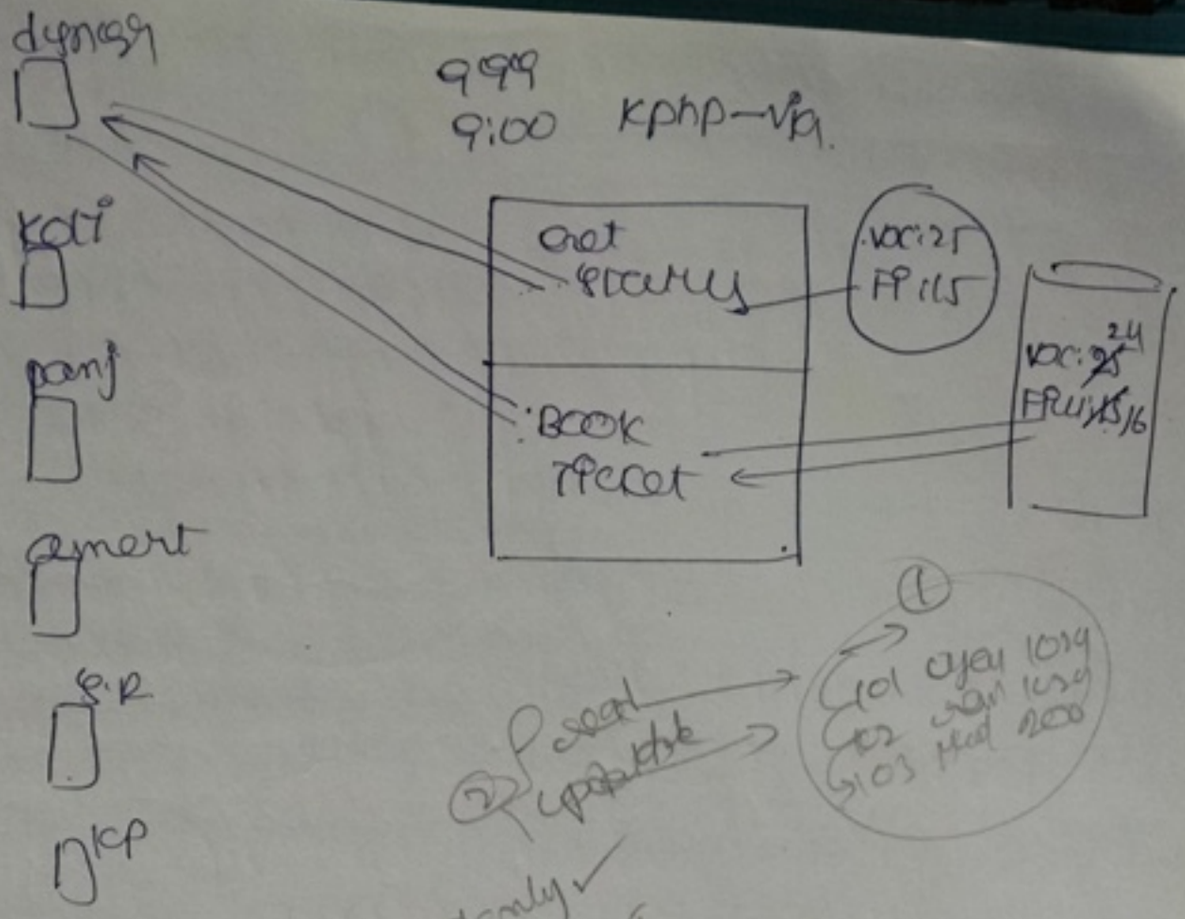
→ Scrollsensitive resultset object is a scrollable resultset object, it will allow the user database modifications after creating resultset object.

— To represent this resultset object Resultset interface has provided the following constant

```
public static final int TYPE_SCROLL_SENSITIVE
```

— Scroll-insensitive resultset is a scrollable resultset object, which it will not allow the later database modifications after creating the resultset object. To represent this is object

```
public static final int TYPE_SCROLL_INSENSITIVE
```



① forward ⇄ Readonly
updatable ✓

② Scroll_sensitive → Readonly
updatable

Scroll_ayesensitive → Readonly
updatable ✓

What is the diff b/w Scroll_sensitive & updatable resultset ✓

Note: The default ResultSet type in JDBC is Readonly and forwardonly

— In JDBC applications, if we want to specify a particular type to the resultset object, then we have to specify ResultSet types at the time of creating Statement object, for this we have to use the following method.

public Statement createStatement(int forwardonly / scrollsensitive / scrollinsensitive, int readonly/updatable) throws SQLException

ex: ① Statement st = con.createStatement(
ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);

ResultSet rs = st.executeQuery("select * from emp1");

scrollsensitive & updatable.

— In jdbc application, by using scrollable resultset objects, we are able to retrieve the data in both forward and backward directions.
— To retrieve the data in forward direction, we will use rs.next() method and rs.getXxx() methods.
To read the data in backward direction from resultset object then for each and every record we have to check whether the previous record is available or not, If the previous record is available then we have to move the resultset cursor to the previous record position
— To perform the above we have to use the following method

```
public boolean previous() throws SQL Exception
```

— After getting resultset cursor to a particular record position then we have to retrieve the data from the respective columns For this we have to use the following method.

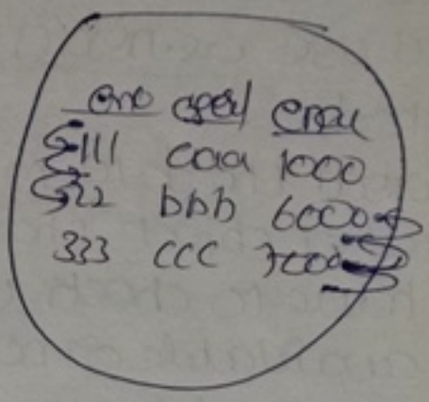
```
public xxx getXxx(int column_index)
```

(or)

```
public xxx getXxx(String column_name)
```

where xxx may be int, short, String, float ——

- The following example demonstrate how to retrieve the data in both forward and backward directions by using scrollable ResultSet object



Database:

Pgm: import java.sql.*;
public class JDBCApp13
{
public static void main(
{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection
("jdbc:mysql://localhost:3306/dungadb", "root",
Statement st=con.createStatement "root");
(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.
CONCUR_UPDATABLE);

```
ResultSet rs=st.executeQuery("select * from emp1");
SOP("Data in forward direction:");
 SOP("_____");
SOP("ENO ENAME ESAL");
 SOP("- - - - - - - - -");
  while(rs.next())
    {
SOP(rs.getInt(1)+"  "+rs.getString(2)+"  "+
    rs.getFloat(3));
  }
      SOP();
 SOP("Data in backward direction");
 SOP("_____");
 SOP("ENO ENAME ESAL");
 SOP("_____");
      while(rs.previous())
        {
SOP(rs.getInt("eno")+"  "+rs.getString("ename")
                    "  "+rs.getFloat("esal"));
        }
        con.close();
    }
}
```

— For this application, we require mysql connector driver.

```
javac10>set classpath=%classpath%;
        D:\software\mysql\my
```

> javac JDBCApp13

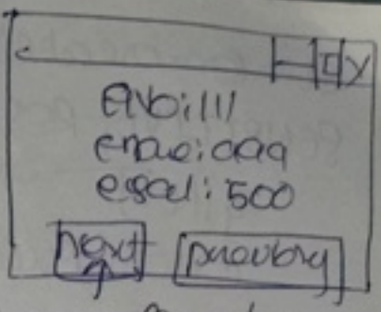
In forward direction

ENO	ENAME	ESAL
111	aaa	10000.0
222	bbb	9000.0
333	ccc	8000.0
444	ddd	7000.0

In backward direction

ENO	ENAME	ESAL
444	ddd	7000
333	ccc	8000
222	bbb	9000
111	aaa	10000

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

class Displayframe extends Frame implements ActionListener
{
    Connection con;
    Statement st;
    ResultSet rs;
    String lastval="";
    Button b1, b2;

    DisplayFrame()
    {
        try
        {
            //first of all set the entire GUI part
        this.setVisible(true);
        this.setSize(500, 500);
        this.setLayout(new FlowLayout());
        this.setBackground(Color.PINK);
        b1=new Button("NEXT");
        b2=new Button("PREVIOUS");
        b1.addActionListener(this);
        b2.addActionListener(this);
        this.add(b1); this.add(b2);
        Class.forName("com.mysql.jdbc.Driver");
        con=DriverManager.getConnection
        ("jdbc:mysql://localhost:3306/durgadb",
            "root", "root");
```



```
st=con.createStatement(ResultSet.TYPE_SCROLL_
SENSITIVE, ResultSet.CONCUR_UPDATABLE);
}//try   rs=st.executeQuery("select * from
                                    emp1");
catch(Exception e)
     { e.printStackTrace();
     }
}
public void actionPerformed(ActionEvent ae)
  { try
label = ae.getActionCommand();
      repaint();
  }
public void paint(Graphics g)
  { try
     {
Font f=new Font("arial", Font.BOLD,30);
    g.setFont(f);
    if(label.equals("NEXT"))
    { boolean b=rs.next();
      if(b==true)
      {
g.drawString("Employee No"+rs.getInt(1),50,
                                    200);
g.drawString("Employee Name"+rs.getString(2),
                                    50,250);
   ("Employee Salary"+rs.getFloat(3),
                                    50,300);
  }
```

There are
key
are
as
Object
and
to
perfor
the
ope
ara

```
else
 {
g.drawString("No more records",100,250);
   }
 }
if(label.equals("PREVIOUS"))
     {  boolean b=rs.previous();
        if(b==true)
          {
 g.drawString("Employee No"+rs.getInt(1),50,200);
g.drawString("Employee Name"+rs.getString(2),
                                    50,250);
g.drawString("Employee Salary"+
                    rs.getFloat(3),50,300);
   }
  else
   {
  g.drawString("No more records",100,250);
   }
  }
}   catch(Exception e)
       { e.printStackTrace();
       }
   }
};
```

```
class JdbcApply
{
public static void main(String[] args)
    { new DBDisplayFrame();
    }
}
```

Database

Date: 9th May, 2012 Wed

SQL> select * from empl;

ora oracle 9i

SQL> select * from student;

Sid	sname	Smarks
s_1	aaa	90
s_2	bbb	60
s_3	ccc	90
=	=	=

— In jdbc applications scroll sensitive resultset can be used to get the latest database updations into resultset object automatically.

Example Pgm

```
import java.sql.*;
public class JdbcApps
{
  p.s.v.m(s[]a) throws Exception
  {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection
      ("jdbc:odbc:nag", "system", "durga");
Statement st = con.createStatement(
ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
ResultSet rs = st.executeQuery("select * from
  sop("Data before updation"); student");
  sop("_____");
```

```
sop("SID SNAME SMARKS");
sop("---------------");
while(rs.next())
{
sop(rs.getString(1)+" "+rs.getString(2)+" "+
rs.getInt(3));
}
sop("Application is in pausing state");
sop("perform updations on database");
System.in.read();
/* This stmt will stop the program execution
until it receives enter command from
the command prompt entered by the user */
sop("Data after updations");
sop("---------------");
rs.beforeFirst();
/* the cursor is at the end of the records.
Now, I want to move the cursor to the first
record position. After the above stmt exe

sop("SID SNAME SMARKS")
sop("---------------")
while(rs.next())
{ rs.refreshRow();
sop(rs.getString(1)+" "+rs.getString(2)+" "+
rs.getInt(3));
}
}
}
```

- Where refreshRow can be used to refresh the present record in order to get modifications.

```
public void refreshRow()
```

- In case of type-1 driver provided by sun-micro systems refreshRow() method is optional to get later database updations.
- Where beforeFirst() method can be used to move resultset cursor to before the first record position.

```
public void beforeFirst()
```

- If we want to move resultset cursor to after the last record position, we have to use the following method

```
public void afterLast()
```

- To move resultset cursor to a particular record position, we will use the following method

```
public boolean absolute(int REC-pos)
```

- If we want to move resultset cursor over the specified number of records (jumping from 1st record to 6th record), we have to use the following method.

```
public boolean relative(int no-of-rec)
```

If no-of-rec is positive number then resultset cursor will jump the specified no-of records in forward direction.

otherwise resultset cursor will do the same in backward direction from the current resultset cursor position; if we specify a negative number in the position of no_of_rec

- To move resultset cursor to the first record position (not before the first record position) then we have to use the following method

public void first()

- If we want to move the resultset cursor to the last record position (not after the last record position) then we have to use the following method

public void last()

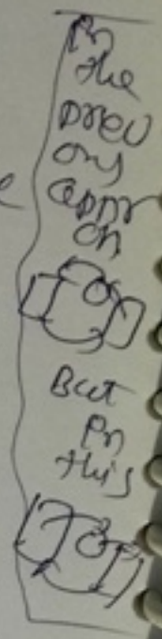
2. Except the last() method and afterlast() method, almost all the above methods are valid in the case of scrollable resultsets.

— In JDBC applications, Scroll Sensitive resultset objects could be supported by only Type-1 driver provided by Sun microsystems not by Type-4 driver provided by Oracle.

— In JDBC applications Scroll Insensitive resultset set objects could not be supported by both Type-1 and Type-4 drivers.

Updatable resultset :- In JDBC applications by using updatable resultset objects, we are able to perform the database operations like insert, update, delete and so on---

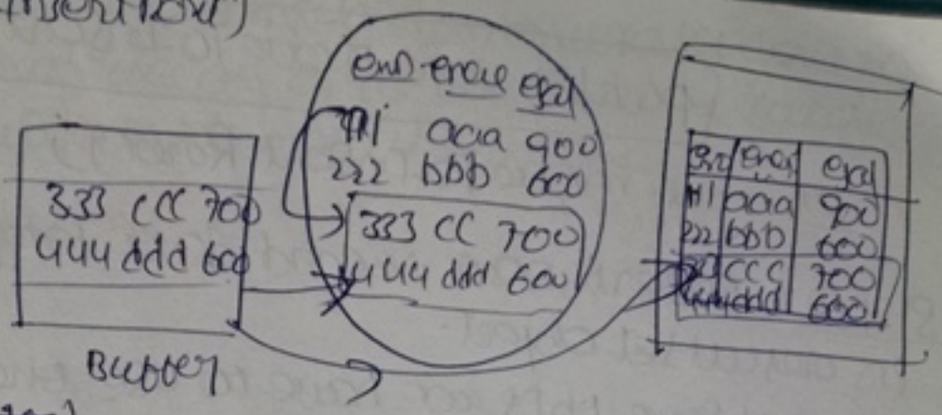
---In JDBC applications by using updatable resultset object, we are able to perform the database operations like insert, update and delete without using insert, update and delete SQL queries.



① rs.moveToInsertRow()

② while(true)
{
rs.updateInt(1,333)
rs.updateString(2,"ccc");
rs.updateInt(3,700)
}

③ rs.insertRow();



—If we want to insert number of records on to the database table through updatable resultset object we have to use the following steps.

Step①: get updatable resultset object when a particular table data.

Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);

ResultSet rs = st.executeQuery("select * from emp1");

Step②: Move resultset cursor to the end of resultset object and take a buffer to hold new record data temporarily.
To move resultset cursor to the end of the resultset object and to take a buffer, we have to use the following method from ResultSet.