UNIT-3

INTRODUCTION TO JDBC

Using JDBC, it is easy to send SQL statements to any relational database. In other words....

with JDBC API it is not necessary to write one program to acess a Sybase database, another program to access and ~~otoe~~ oracle database, another program to access an informix database, so on.

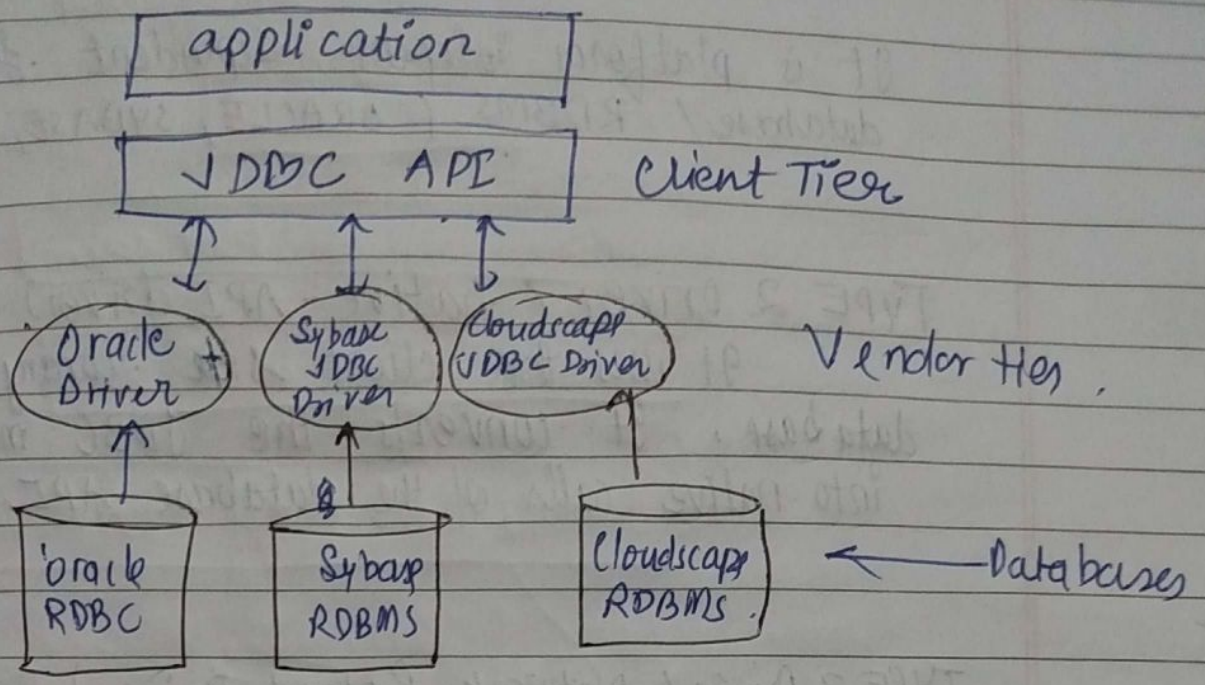
One can write a single program using the JDBC API, & the program will be able to send SQL statement to appropriate database. &, with an application return in Java programming language one does not have to worry about writing different application to run on different platform.

The Combination of Java and JDBC lets a programmer write it once and run it anywhere. JAVA being robust, secure, easy to use, easy to understand & automatically downloaded on a network. Is an excellent language basis for databases application.

1) JDBC define how a java program can communicate with database.

2) It has mainly two packages.
   - JAVA.SQL
   - JAVAX.SQL

JDBC access.



Their are 4 types of drivers.

1) Type 1 Driver (JDBC-ODBC DRIVER)

2) Type 2 Driver (native - API drives)

3) Type 3 Driver (network Protocol Driver)

4) Type 4 (native Protocol)

1) Type 1 Driver (JDBC-ODBC DRIVER)

This driver is implemented in (sun.jdbc.odbc.JdbcOdbc Driver)
packages

Class :

Type1 Implement (sun.jdbc.odbc.JdbcOdbcDriver)

Date 16/09/16
Page 31

this driver is platform dependent.

It is platform ~~indepent~~ independent. for several database / RDBMS (ORACLE, SYBASE, CLOUDSCAPE)

TYPE 2 Driver (native-API driver)

It uses the client side library of the database. It converts the JDBC method calls into native calls of the database API.

TYPE 3 Driver (Network Protocol Driver)

It make use of a middle ~~type~~ tire b/w the calling program & the database.

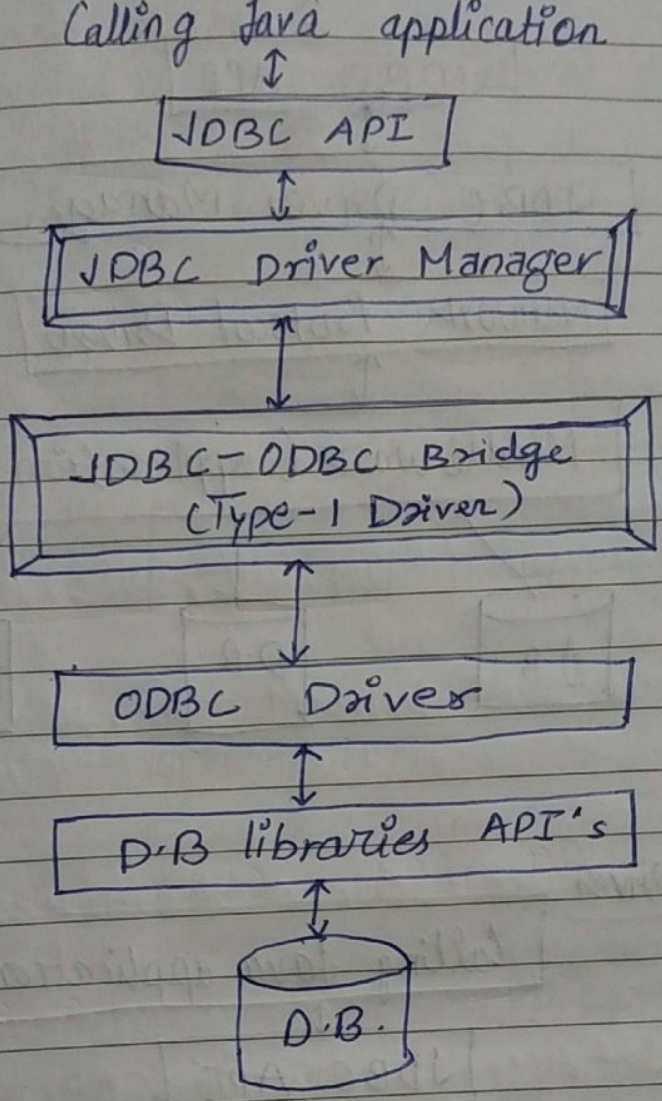
there is no need for the vendor database library on the client machine.

TYPE 4 (Native Protocol)

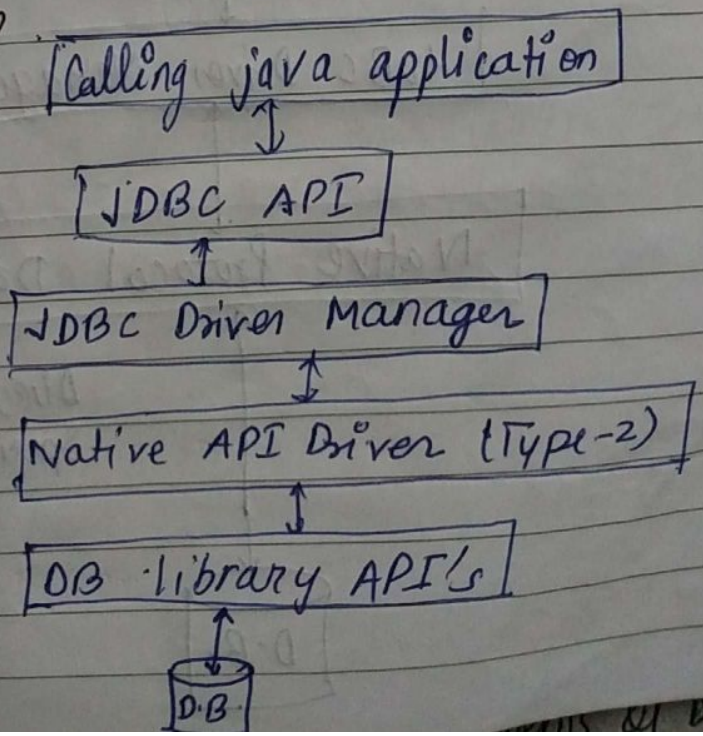
It is completely written in java & thus it is platform independent.

It is installed inside the virtual m/c of the client

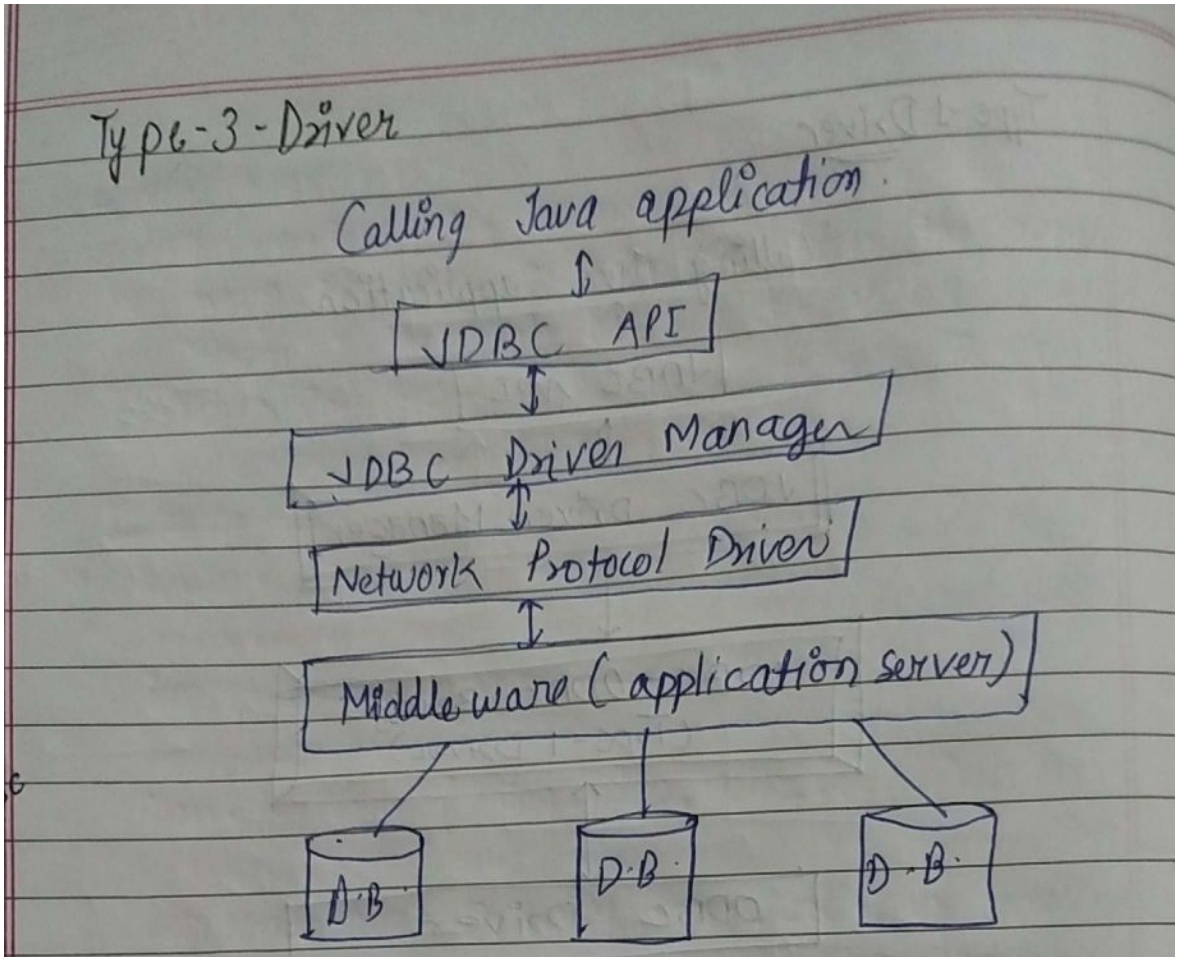
Type-1 Driver



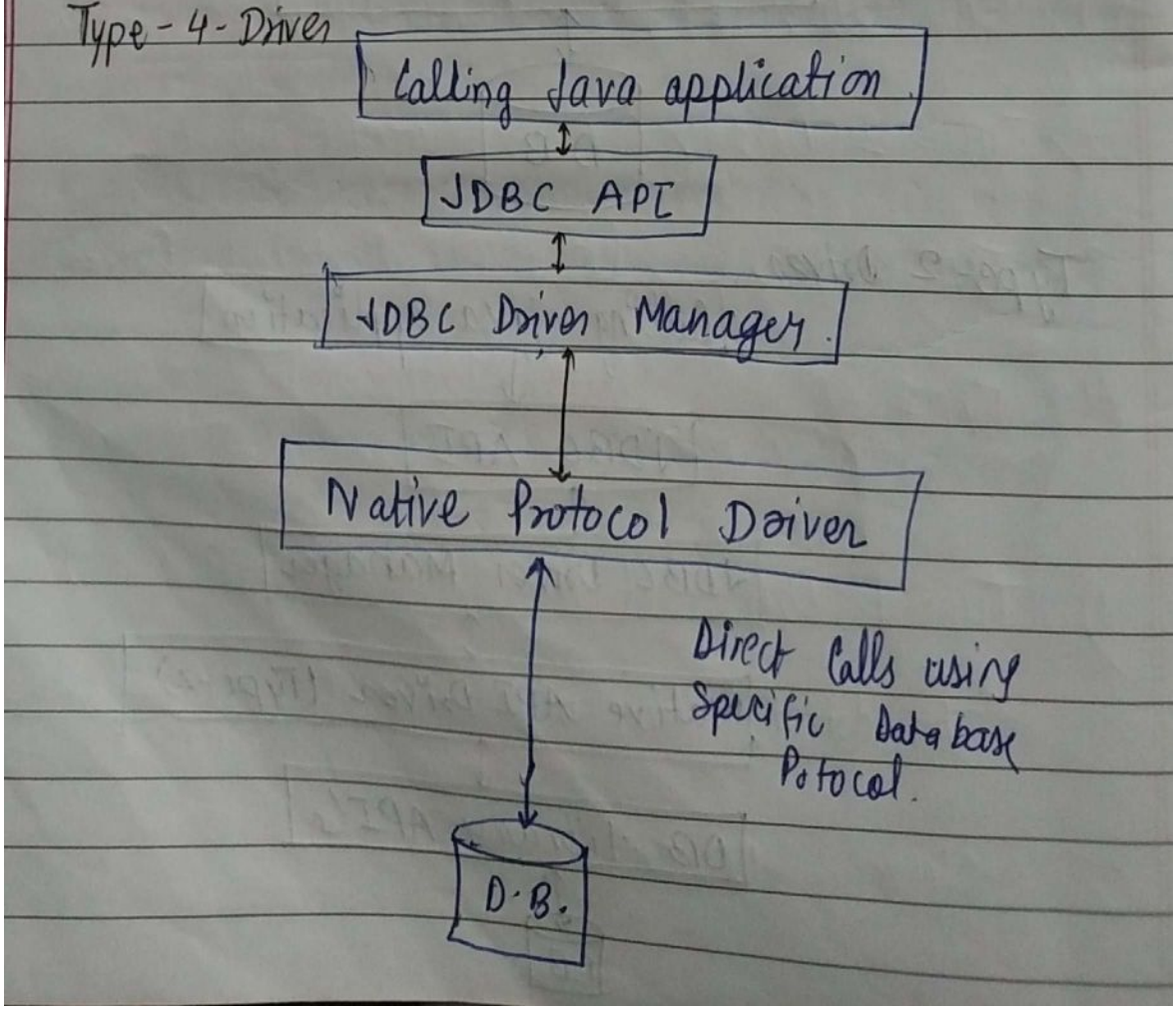
Type-2 Driver



Type-3-Driver



Type-4-Driver



TYPES OF STATEMENTS IN JDBC:

1. SIMPLE STATEMENT
   It can be used for General purpose excess to the database.
   It is used when we are using static SQL statement at runtime.

2. PREPARED STATEMENT
   It is used when we plan to use the same SQL statements many time.
   It accepts input parameters at run-times.

3. COLLABLE STATEMENT
   It is used when we want to access database stored procedure

Types of methods used in Simple statement.

- Execute()
  This method can be used either with selection SQL Queries or updation SQL Queries.
  It's returntype is BOOLEAN (True/false)

- Executeupdate()
  This method can be used to do updation in the database. It's return type is INTEGER (1,2..)

Executequery()
This method can be used to execute selection group of SQL Queries to fetch the data from the data base. It's return type is ResultSet (In itself it is a Interface)

~~import~~

1. import → Package → SQL
2. Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
3. Connection con = DriverManager.getconnection ("JDBC:odbc:dsn");
   ↑ user defined name
   O.B
   M.s access
4. Statement st = con.createstatement();
   (Referring)
   st is an interface.
5. ResultSet rs = st.executeQuery ("select * from tablename");
6. while (rs.next()) → True
   {
   S.O.P (rs.getString (index));
   (rs.getInt (1/2);
   }

executeUpdate() ⟶ SQL ⟶ insert/update

executeupdate() · return → int

```
int i = St. executeUpdate("insert into tablename values("KSB");
S.O.P(i);
```

⟶ 1 · Row updated

```
int i = st.executeUpdate("update tablename
Set value name = "Sachin" where name = "KSB");
S.O.P(i); ⟶ 1 row updated.
```

execute() —return→ boolean (True/False)

```
ResultSet rs = NULL;
boolean b = St.execute("Select name from
                        tablename");
            (method)
if (b)
{
  rs = St.getResultset();
}
while (rs.next())
    {
      S.O.P("name are" +name);
    }
```

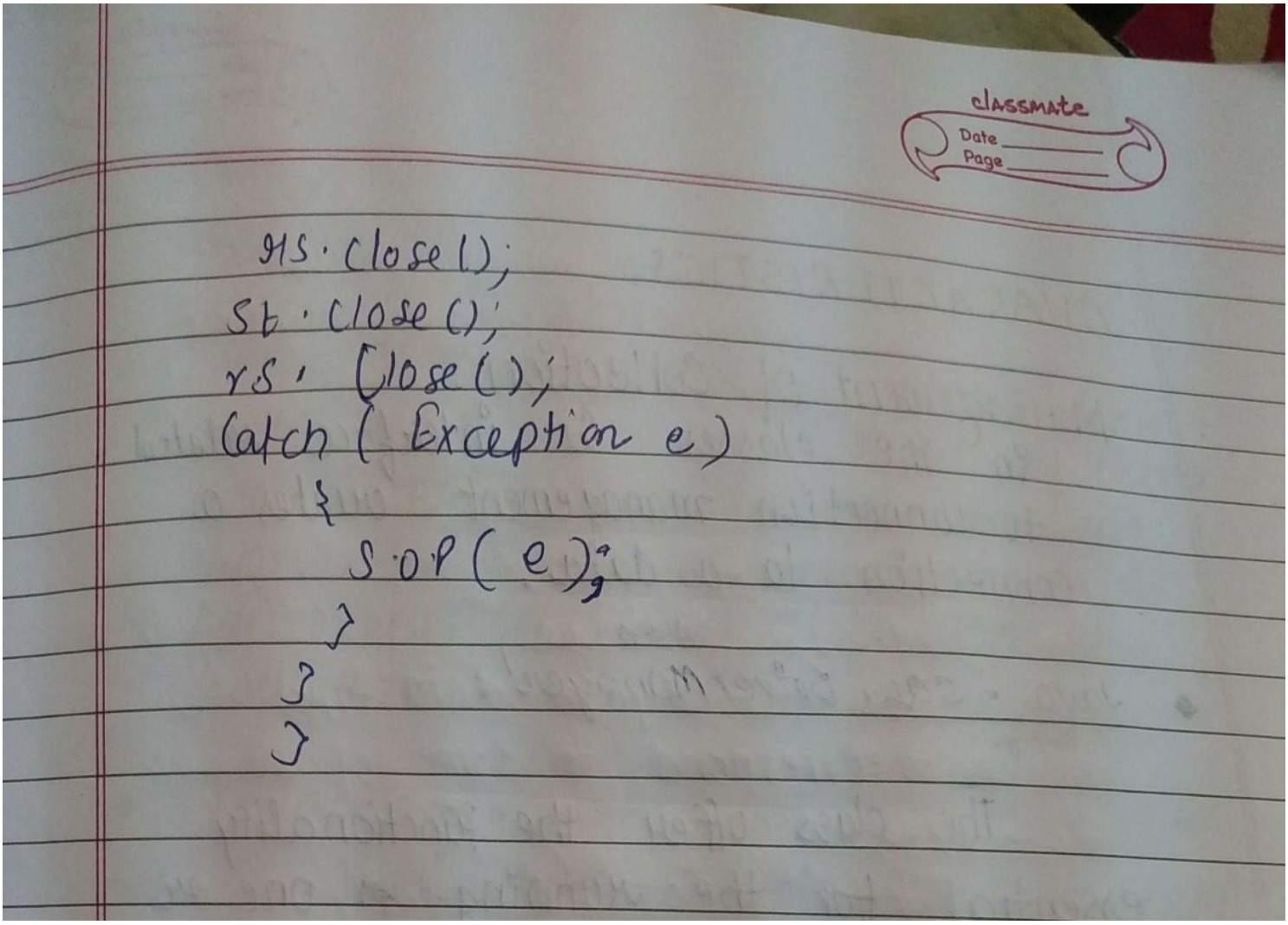
WAP to Retrieve records from DB

```
import java.sql.*
  class A
{
 P.S.V.M (String S[])
 {
  int Roll_no;
   String name, Course;

  try
    {
     Connection con;
     Statement st;
     Resultset rs;
    Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
    Con = DriverManager.getConnection("jdbc:odbc:
                                    Mydsn","","");
    st = Con.createStatement();
    rs = st.executeQuery ("select * from tablename");
    while (rs.next())

    {
      Roll_no = rs.getInt(1);
      name = rs.getString(2);
      Course = rs.getString(3);
      S.O.P(Roll_no+" "+name+" "+Course);

    }
```



CHARACTERISTICS

Management of Collection:
In the classes & interface related to connection management creates a connection to a data.

- java.sql.DriverManager (class): (java.sql — Package)

The class offer the functionality essential for the running. one or more data base drivers, Individual driver connects to specific database.

- Java.Sql.Driver:
The interface abstract the connection protocol of specific vendors.

- Java.sql.connection:
It abstracts the interaction with the database. sending SQL statement to database, reading the results of operations is done by this interface.

Access of Database:
After creating connection with database, these types of classes and interfaces allow you to send SQL commands to the data base for execution, and also read the response.

- java.sql.Statement : (interface)
  The interface Allows you to execute SQL commands above the connection and access the results.

- java.sql.PreparedStatement : (interface)

The interface use to give parameterized SQL statements, include placeholders (as '?') which will be later on replaced by actual values (During runtime).

- java.sql.CollableStatement : (interface)

The interface allows you to store procedure.

- java.sql.ResultSet : (interface)

The interface stores the results of execute SQL SELECT statement

or command. To access the result row by row, it provides some methods.

Type of data

In java.sql package that provides several java data types so, as to correspond to some of the SQL data type.

- Java.sql.Array.
  ↓
  Interface

The interface gives an ~~extraction~~ abstraction in java language with array collection of Sql datatypes.

array initialize

```
int a[] = new int [↓ size]
scanner sc = new scanner (System.in);
int n = sc.nextInt();
int a[] = new int [n];
for (int j=0; j<a.length ; j++)
    a[j] = sc.nextInt();
}
```

- java.sql.Data.

  The interface gives an abstraction in java language with sql type date.

- java.sql.time

  The interface gives an abstraction in java language with sql type time.

javascript

1) New → others → web >> Static web project
   >> Project Name > KSB1
      OK

KSB1 >>>
↳ Resources
↳ web contents → R.C → New → other → javascript >>
↳ js

javascript Source file

1.js

1.js

```
function add(a,b)
{
    return a+b;
}
```

1. HTML

```
<Script Type = "text/javascript"
        src = "1.js"> </Script>

<body>
 <script type = "text/javascript">
        alert (add(20,30));
 </script>.
```

- METADATA OF DATABASE

Metadata is some data about the database. Sometime developer needs to work on metadata. so that JDBC API also supports this facility to retrieve metadata of the database, also parameters to statements and result too,

- java.sql.DatabaseMetadata
  interface
  The instance of this interface can be obtain by java.sql.connection. This is used to get information about the features of database.

- java.sql.ResultSetMetada ;
  interface
  Methods are provided by this interface to access metada of java.sql.resultset.

- java.sql.parametesmetada:
  The interface permits you to access the database types of parameters in prepared statements

EXCEPTIONS & WARNING.

The collection of classes support the runtime exception and warning related to JDBC.

- java.sql.SQLException
  Handle all type of exception of JDBC related, database level, driver level exceptions and error levels.

- java.sql.SqlWarning.
  It represents database warnings.
- java.sql.BatchUpdateException.
  It handles the exception for batch update.

LOADING THE DATABASE DRIVER:

The concept of the URL in JDBC.
It same as use like other application such as internet.
In basic concept of JDBC URL, in application more than.
In application more than one database can be in used.
each will be accessed by a different database driver. In such a situation, it is necessary that every driver must be uniquely identified.
to comm with data source, you must load a driver.

Prepared Statement :-

PreparedStatement PS = con.prepareStatement method
Interface. ("insert into tablenamevalues(?)");

```
Ps.setString(1, "Anuj");
Ps.executeUpdate(1);
Ps.setString(2, "KSB");
Ps.executeUpdate();
```

Q- Explain Servlet Packages?

Q Differentiate b/w doGet() & doPost() methods

Q Explain JSP processing? ⟶ figure.

Q Write a program to count total number of vowels in a string?

Q Implicit objects in JSP, define it?

Define Javascript ? Write a simple program of javascript using HTML that will print your course on web browser.?

- Javascript is a programming language design for web pages.
- The Java script is interperated by the web browsers.
- Javascript is platform independent & object oriented.
- Javascript inhances webpages with dynamic & interactive features.
- Java script runs in client software.
- Unlike HTML, java script is case sensitive

Java script is a most popular scripting language on the internet, & works in all major browsers such as internet explorer, Mozilla, Firefox.

```
<html>
<body>
<script type="text/javascript">
document.write("Hello");
</script>
</body>
</html>
```

If includes a no. of actions. For database acess to make it easy to develop simple database driven JSP application.

The action provid following feature.

1) Using a connection pool for better performance and scalability.

2) Supporting Queries, updates, & inserts.

3) Handling the most data type conversion

4) Supporting a combination of database operations in one transaction.

We need to ~~know~~ load JDBC and return a connection object.
we need a connection object to create a statement object. we can either update the data or generate the resultset object.

The steps are as follows.

1) load the JDBC Driver
2) Create a connection object

JS

3) Create a statement object

4) Call its execute update() method to insert a ~~recall itself~~ record

~~to obtain a~~ · Use the same statement object to obtain a resultset object

Eg.
```
<%@ page import = java.sql.* %>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDrive
Connection con = DriverManager.getConnection
("Jdbc:odbc:dsn","","");
Statement St = con.createStatement();
ResultSet rs = St.executeUpdate
("Select * from Tablename");
%>
```

Saves as Name.jsp.

Application Specific Database Action.

↳ Accessing D.B through JSP.

We can use JSTL database action to develop many types of interesting web applications. such as product catalog interface;

employed directories, online billboards, without being of java programmes. These type of application account for a high percentage of the web applications developed today. But at some level of complexity, putting SQL statements directly in the web pages can become a maintanence problem the SQL statement represent buisness logic for more complex application buisness logic is better develope as separate java class. For a complex application, it may be better to use application specific custom actions ~~Database action~~, instead of the JDTL data base action.