Java 1

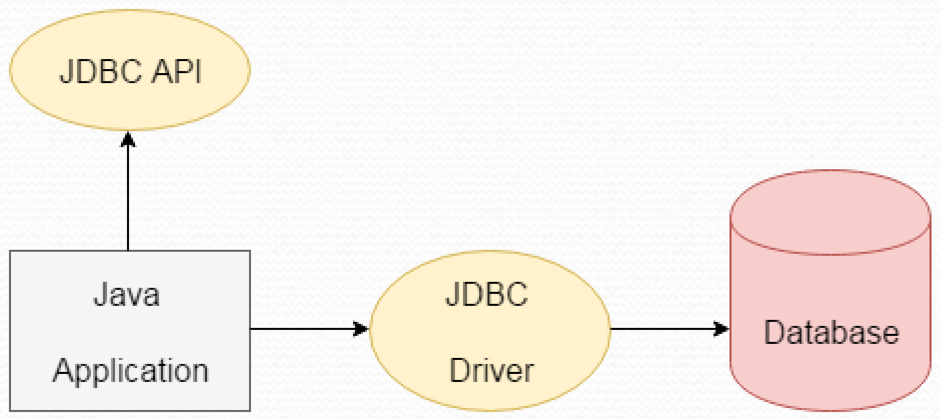
Lecture 09

Sayed Ahmad Sahim

22.November.2017

Java JDBC

- **Java Database Connectivity (JDBC)** is a java API.
- JDBC defines how a client may access a database.
- It is Java based data access technology and used for Java database connectivity.
- JDBC API uses jdbc drivers to connect with the database.



Why use JDBC

- Before JDBC, ODBC API was the database API to connect and execute query with the database.
- ODBC API uses ODBC driver which is written in C language

JDBC Driver

- JDBC Driver is a software component that enables java application to interact with the database.
- There are 4 types of JDBC drivers:
  - JDBC-ODBC bridge driver
  - Native-API driver (partially java driver)
  - Network Protocol driver (fully java driver)
  - Thin driver (fully java driver)

JDBC-ODBC bridge driver

- The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.

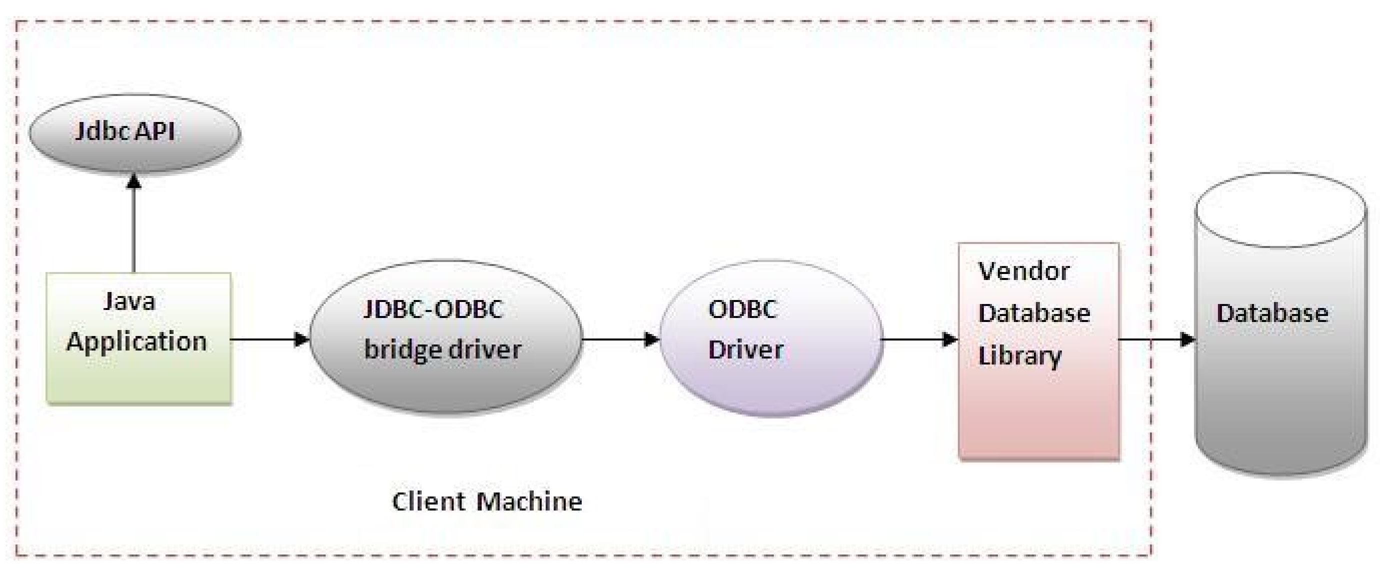


Figure- JDBC-ODBC Bridge Driver

Pros & Cons

- Advantages
  - easy to use.
  - can be easily connected to any database.

- Disadvantages:
  - Performance degraded because JDBC method call is converted into the ODBC function calls.
  - The ODBC driver needs to be installed on the client machine.

Native-API driver

- The Native API driver uses the client-side libraries of the database.

- The driver converts JDBC method calls into native calls of the database API.
- It is not written entirely in java.

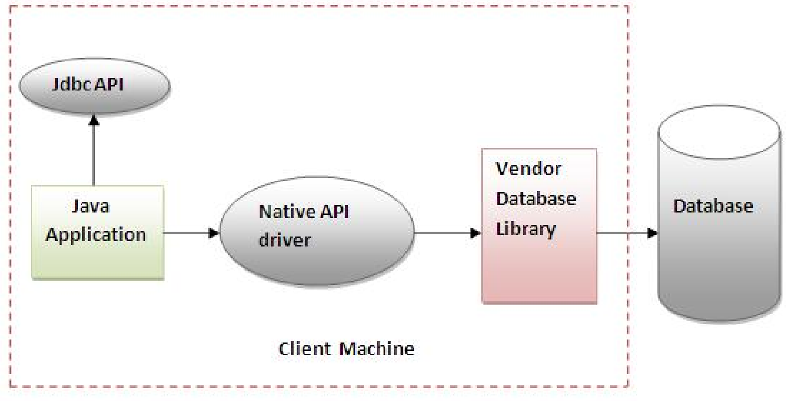


Figure- Native API Driver

Pros & Cons

- Advantages
  - performance upgraded than JDBC-ODBC bridge driver
  - can be easily connected to any database.
- Disadvantages:
  - The Native driver needs to be installed on each client machine
  - The Vendor client library needs to be installed on client machine.

Network Protocol driver

- The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol.
- It is fully written in java.

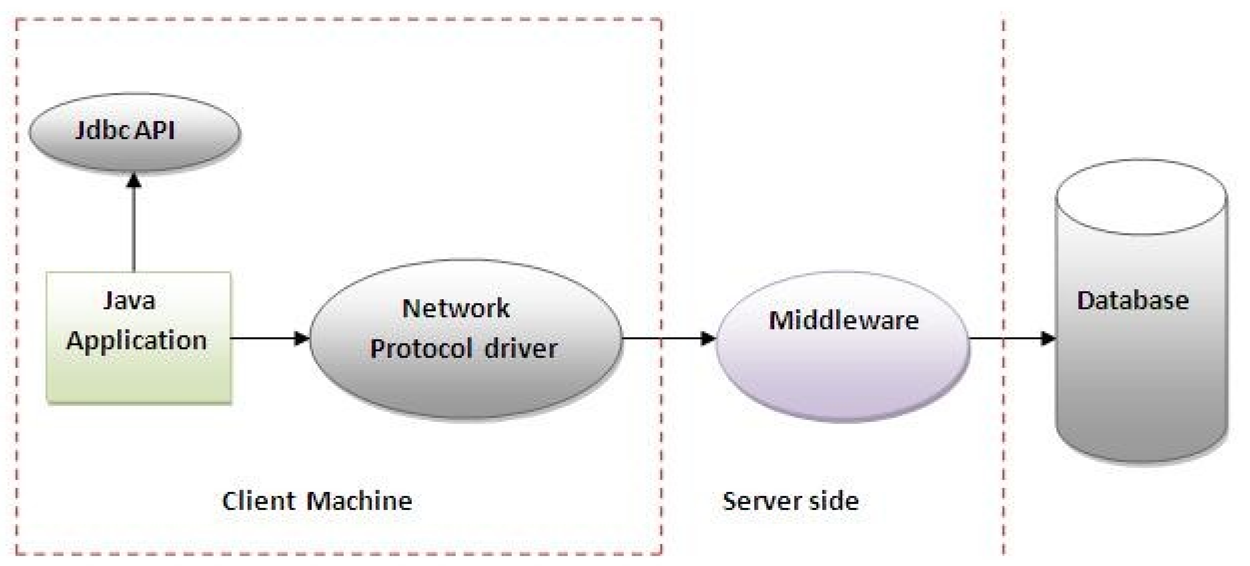


Figure- Network Protocol Driver

Pros & Cons

- Advantages
  - No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.
- Disadvantages:
  - Network support is required on client machine.
  - Requires database-specific coding to be done in the middle tier.
  - Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

Thin driver

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

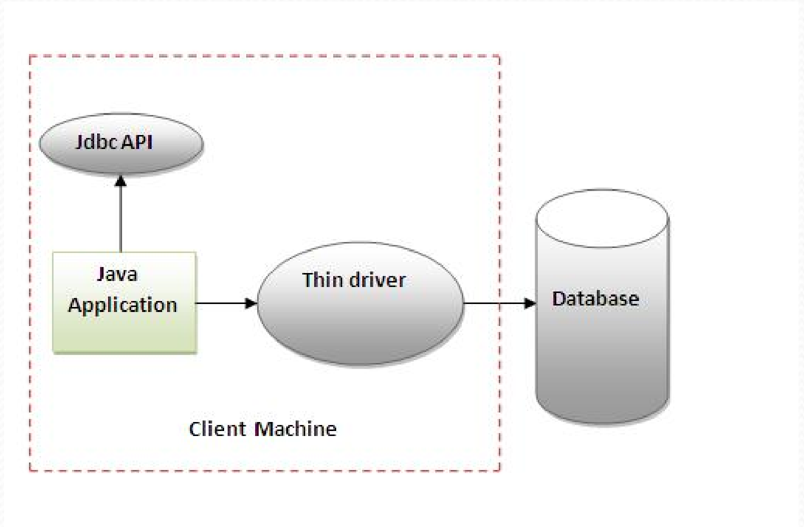


Figure- Thin Driver

Pros & Cons

- Advantages
  - Better performance than all other drivers.
  - No software is required at client side or server side.
- Disadvantages:
  - Drivers depends on the Database.

Five Steps

- There are 5 steps to connect any java application with the database in java using JDBC.

1. Register the driver class
2. Creating connection
3. Creating statement
4. Executing queries
5. Closing connection

Demo

DriverManager class

- The DriverManager class acts as an interface between user and drivers.

- It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.

- The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver().

Connection interface

- A Connection is the session between java application and database.
- The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData.
  - i.e. object of Connection can be used to get the object of Statement and DatabaseMetaData.
- The Connection interface provide many methods for transaction management like commit(), rollback() etc.

Statement interface

- The Statement interface provides methods to execute queries with the database.
- The statement interface is a factory of ResultSet
  - i.e. it provides factory method to get the object of ResultSet.
- Statement Interface methods
  1. public ResultSet executeQuery(String sql)
  2. public int executeUpdate(String sql)
  3. public boolean execute(String sql)
  4. public int[] executeBatch()

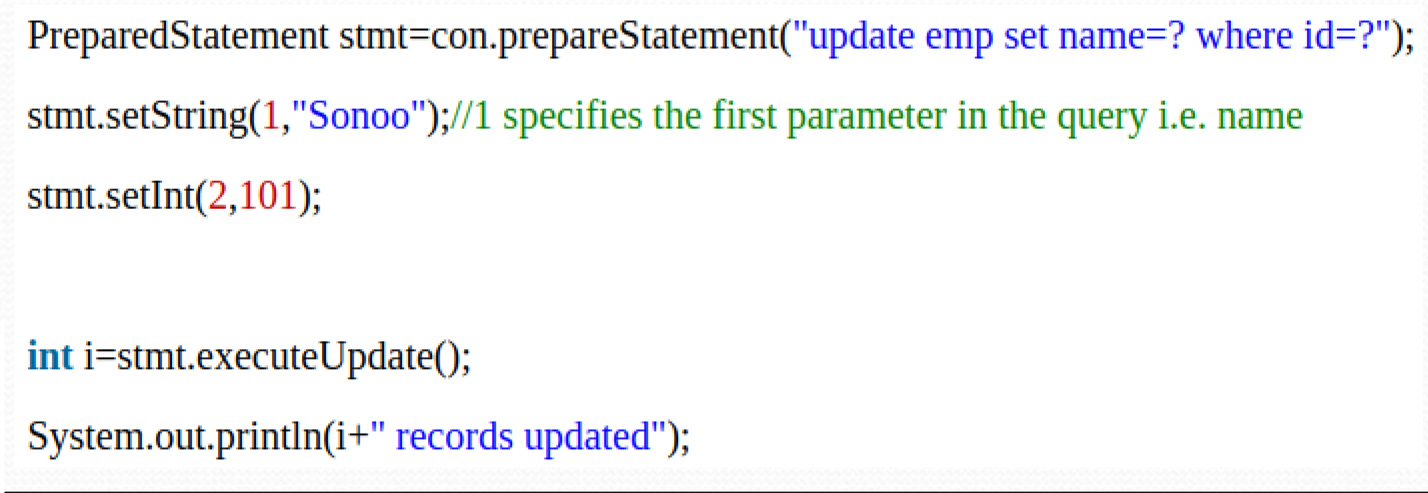
ResultSet interface

- The object of ResultSet maintains a cursor pointing to a row of a table.
- Initially, cursor points to before the first row.

public boolean next():	is used to move the cursor to the one row next from the current position.
public boolean previous():	is used to move the cursor to the one row previous from the current position.
public boolean first():	is used to move the cursor to the first row in result set object.
public boolean last():	is used to move the cursor to the last row in result set object.
public boolean absolute(int row):	is used to move the cursor to the specified row number in the ResultSet object.

PreparedStatement interface

- The PreparedStatement interface is a subinterface of Statement.
- It is used to execute parameterized query.
  - String sql="insert into emp values(?,?,?)";
  - ? are the parameters of the query, their values will be set by setter methods.
- Why use PreparedStatement?
  - performance

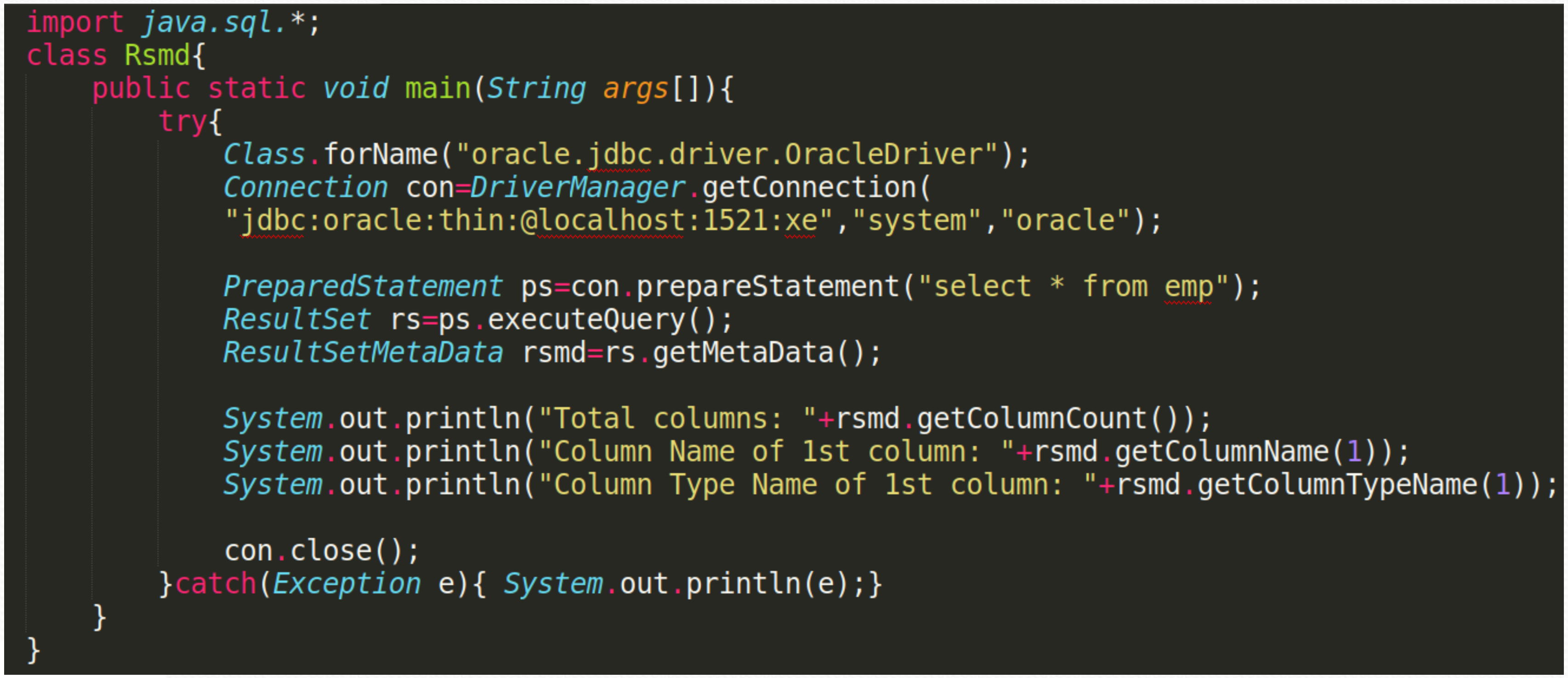


```
PreparedStatement stmt=con.prepareStatement("update emp set name=? where id=?");
stmt.setString(1,"Sonoo");//1 specifies the first parameter in the query i.e. name
stmt.setInt(2,101);


int i=stmt.executeUpdate();
System.out.println(i+" records updated");
```

Java ResultSetMetaData Interface

- **Metadata** means data about data
- ResultSetMetaDataIf is used when we want to get metadata of a table like total number of rows and columns.



```
import java.sql.*;
class Rsmd{
    public static void main(String args[]){
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con=DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

            PreparedStatement ps=con.prepareStatement("select * from emp");
            ResultSet rs=ps.executeQuery();
            ResultSetMetaData rsmd=rs.getMetaData();

            System.out.println("Total columns: "+rsmd.getColumnCount());
            System.out.println("Column Name of 1st column: "+rsmd.getColumnName(1));
            System.out.println("Column Type Name of 1st column: "+rsmd.getColumnTypeName(1));

            con.close();
        }catch(Exception e){ System.out.println(e);}
    }
}
```

Java DatabaseMetaData interface

- DatabaseMetaData interface provides methods to get meta data of a database
  - such as: database product name, database product version, driver name, name of total number of tables, name of total number of views etc.

Exercises

- Create database example
- Create table Books (id, title, author)
- Insert into database Books atleast five records
- Show all the records from database
- Show only 5th record
- Delete 2nd row
- Modify 5th row title.