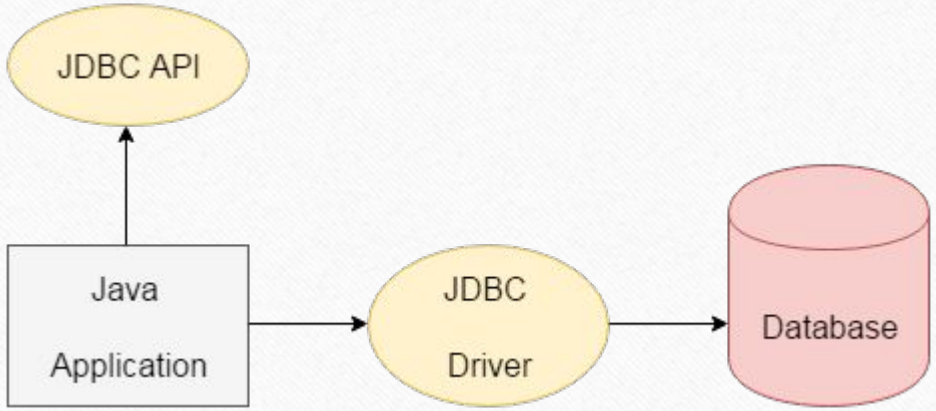
Java JDBC

JDBC

- JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:
- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

Cont..

- We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.
-



Cont.

- we can use JDBC API to handle database using Java program and can perform the following activities:
- Connect to the database
- Execute queries and update statements to the database
- Retrieve the result received from the database.

JDBC Driver

- JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

1)JDBC-ODBC bridge driver

- The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

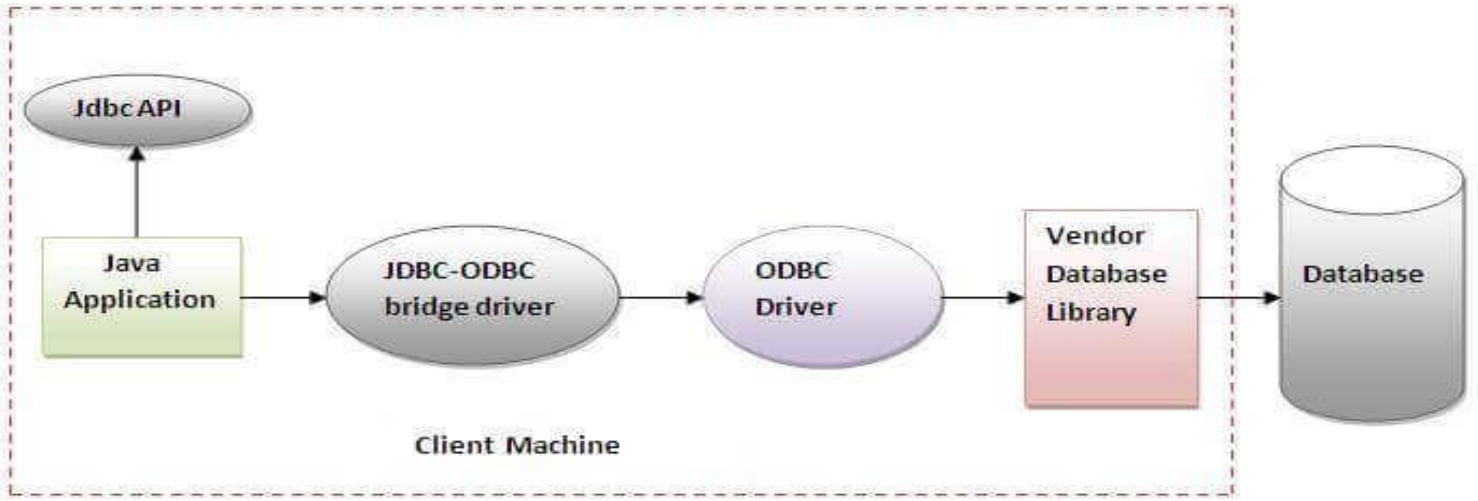


Figure- JDBC-ODBC Bridge Driver

Cont..

- oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.
- 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.

2) 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.

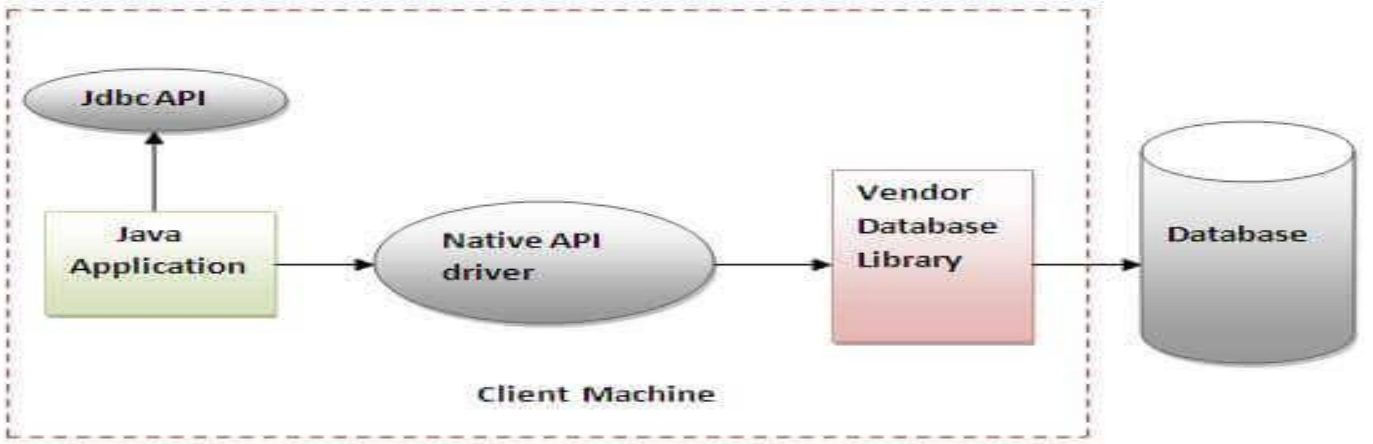


Figure- Native API Driver

Cont..

- Advantage:
  - performance upgraded than JDBC-ODBC bridge driver.
- Disadvantage:
  - The Native driver needs to be installed on the each client machine.
  - The Vendor client library needs to be installed on client machine.

3) 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.

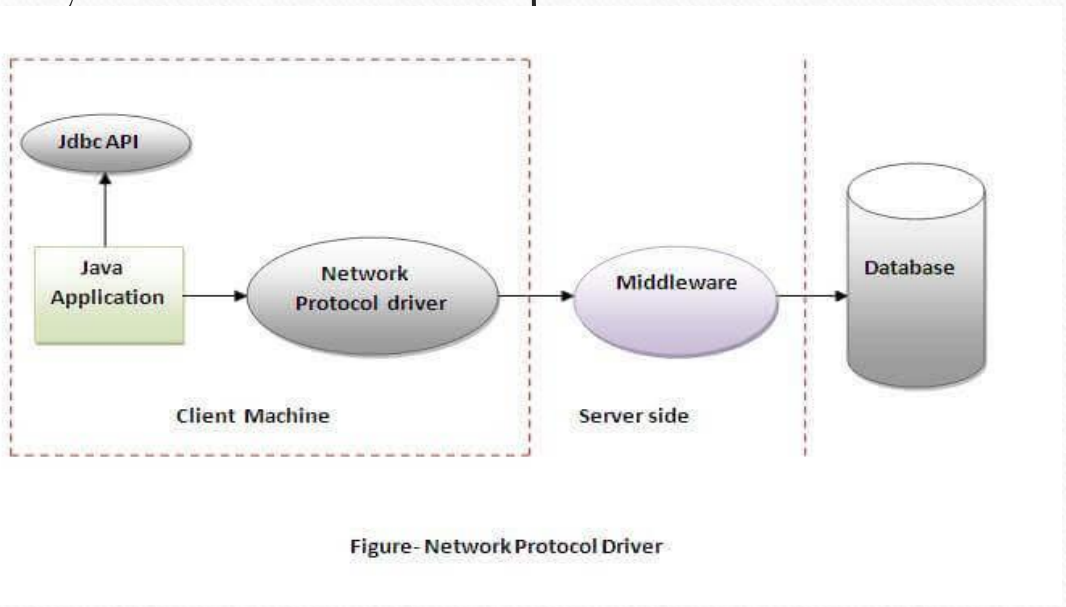


Figure- Network Protocol Driver

Cont..

- Advantage:
  - 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.

4) 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.

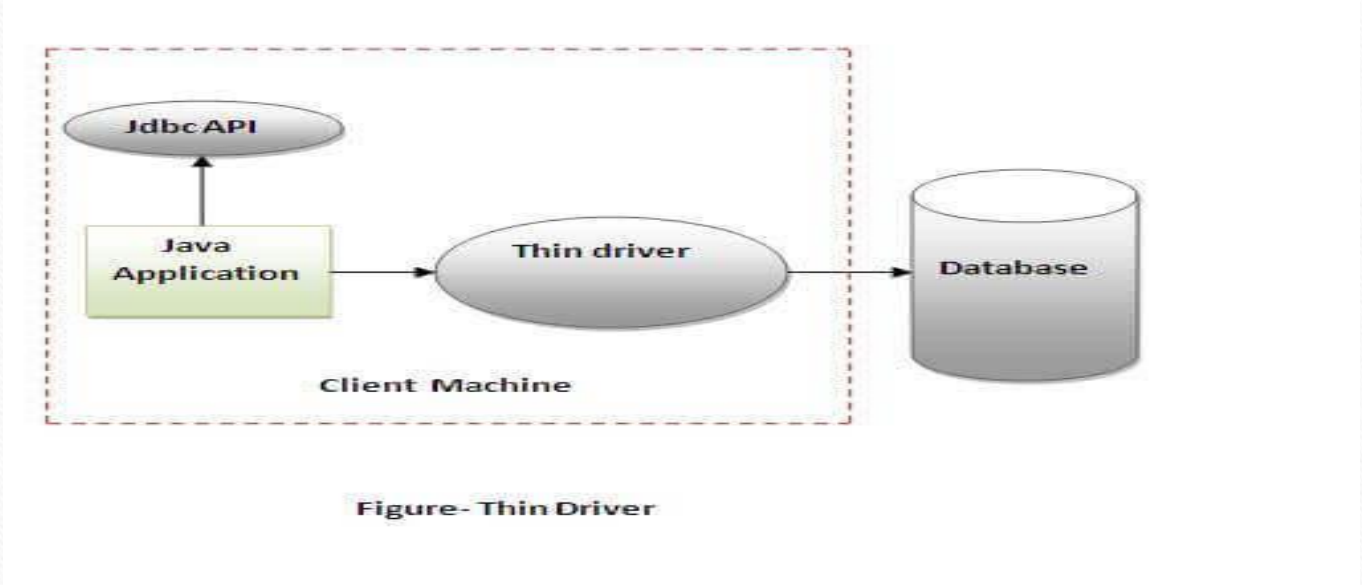


Figure- Thin Driver

Cont..

- Advantage:
  - Better performance than all other drivers.
  - No software is required at client side or server side.
- Disadvantage:
  - Drivers depend on the Database.

Java Database Connectivity with 5 Steps

1. Register the Driver class
2. Create connection
3. Create statement
4. Execute queries
5. Close connection

1) Register the driver class

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

- Syntax of forName() method
- **public static void** forName(String className)**throws** ClassNotFoundException
- Class.forName("oracle.jdbc.driver.OracleDriver");

2) Create the connection object

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

1) **public static** Connection getConnection(String url)**throws** SQLException

2) **public static** Connection getConnection(String url,String name,String password) **t hrows** SQLException

- Connection con=DriverManager.getConnection(
- "jdbc:oracle:thin:@localhost:1521:xe","system","password");

3) Create the Statement object

- The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.
- Syntax of createStatement() method
  - **public** Statement createStatement()**throws** SQLException
  - Statement stmt=con.createStatement();

4) Execute the query

- The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.
- Syntax of executeQuery() method
- **public** ResultSet executeQuery(String sql)**throws** SQLException
- ResultSet rs=stmt.executeQuery("select * from emp");
-
- **while**(rs.next()){
- System.out.println(rs.getInt(1)+" "+rs.getString(2));
- }

5) Close the connection object

- By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.
- Syntax of close() method
- **public void** close()**throws** SQLException
- con.close();

Java Connect to Microsoft SQL Server

- **Download Microsoft JDBC driver for SQL server**
- https://docs.microsoft.com/en-us/sql/...
- https://docs.microsoft.com/en-us/sql/connect/jdbc/

JDBC database URL for SQL Server

```
jdbc:sqlserver://[serverName[\instanceName][:portNumber]][;property=value[;property=value]]
```

- Where:
  - serverName: host name or IP address of the machine on which SQL server is running.
  - instanceName: name of the instance to connect to on serverName. The default instance is used if this parameter is not specified.
  - portNumber: port number of SQL server, default is 1433. If this parameter is missing, the default port is used.
  - property=value: specify one or more additional connection properties. To see the properties specific to SQL server,

3. Register JDBC driver for SQL Server and establish connection

- The JDBC driver class of SQL Server is com.microsoft.sqlserver.jdbc.SQLServerDriver, so to register this driver, use the following statement:

- DriverManager.registerDriver(new com.microsoft.sqlserver.jdbc.SQLServerDriver());
- OR
- Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");ver.jdbc.SQLServerDriver());
- Example

Java Database Connectivity with Oracle

- **Driver class:** The driver class for the oracle database is **oracle.jdbc.driver.OracleDriver**.
- **Connection URL:** The connection URL for the oracle10G database is **jdbc:oracle:thin:@localhost:1521:xe** where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.
- **Username:** The default username for the oracle database is **system**.
- **Password:** It is the password given by the user at the time of installing the oracle database.

Example

- we are connecting to an Oracle database and getting data from **emp** table. Here, **system** and **oracle** are the username and password of the Oracle database.

```
import java.sql.*;
class OracleCon{
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");

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

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

}
}
```

Java Database Connectivity with MySQL

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.
3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.
5. Example

Connectivity with Access without DSN

- There are two ways to connect java application with the access database.

1. Without DSN (Data Source Name)
    1. example
2. With DSN
    1. Example

Thank you

Follow me on: https://www.linked.com/in/kunajun77
Github: https://www.github.com/arjundhav