



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).

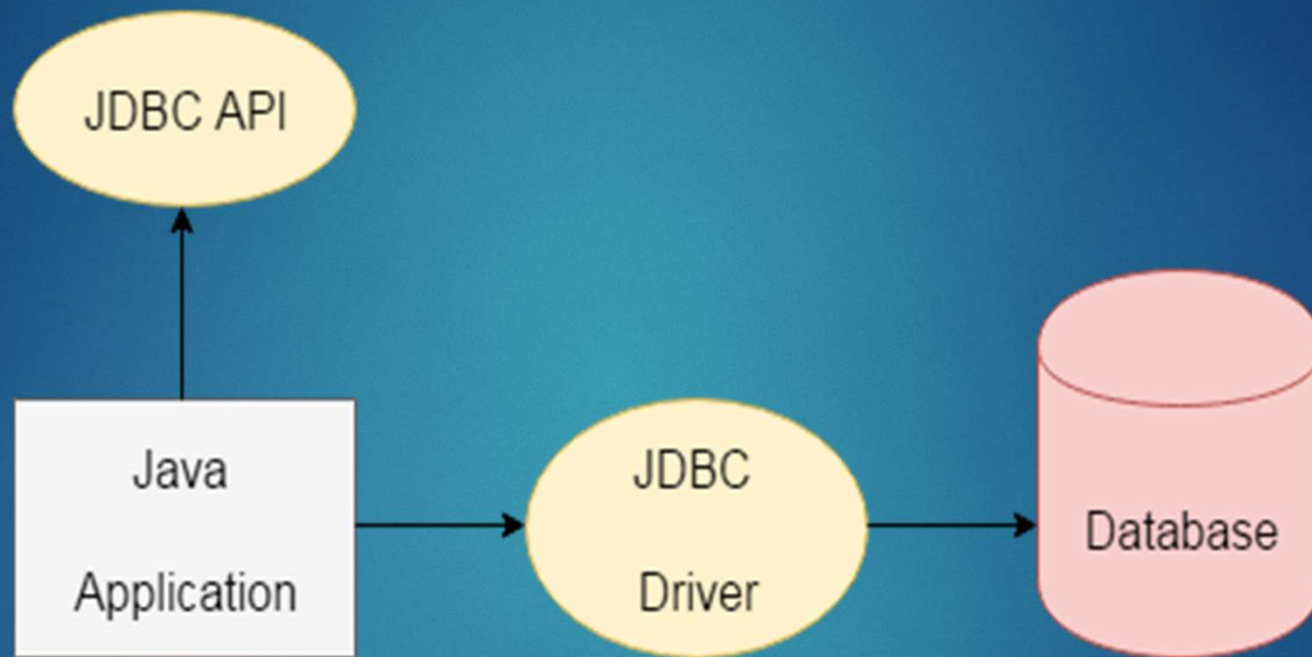
JDBC API uses JDBC drivers to connect with the database.



Definition of JDBC

JDBC is an API(Application programming interface) which is used in java programming to interact with databases.

The classes and interfaces of JDBC allows application to send request made by users to the specified database.



Why Should We Use JDBC

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

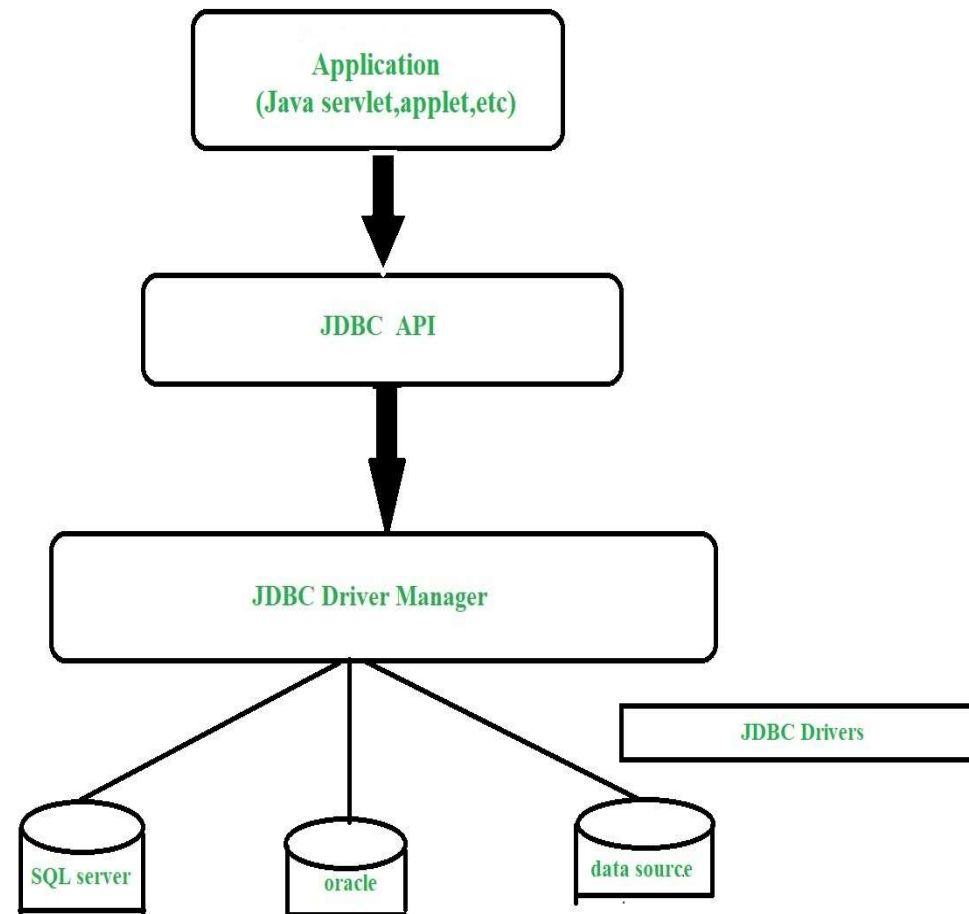
Components of JDBC

There are generally four main components of JDBC through which it can interact with a database.

1. JDBC API
2. JDBC DRIVE MANAGER
3. JDBC Test suite
4. JDBC-ODBC Bridge Drivers



Architecture of JDBC






1.Application: It is a java applet or a servlet which communicates with a data source.

2.The JDBC API: The JDBC API allows Java programs to execute SQL statements and retrieve results



Some of the important classes and interfaces defined in JDBC API are as follows:

- 1.DriverManager
- 2.Driver
- 3.Connection
- 4.Statement
- 5.PreparedStatement
- 6.CallableStatement
- 7.ResultSet
- 8.SQL data



3.DriverManager: It plays an important role in the JDBC architecture.It uses some database-specific drivers to effectively connect enterprise applications to databases.

4.JDBC drivers: To communicate with a data source through JDBC, you need a JDBC driver that intelligently communicates with the respective data source.



JDBC CLASSES AND INTERFACES



Driver Manager

The DriverManager class (java.sql.DriverManager) is one of main components of JDBC.

DriverManager manages database drivers, load database specific drivers and select the most appropriate database.



Connection

Connection interface provides access the session established with a database server.

```
Connection con =  
DriverManager.getConnection(url,  
username, password);
```



Statement

The Statement interface provides a standard abstraction to execute SQL statements and return the results using the ResultSet objects.


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



PreparedStatement

PreparedStatement is a sub interface of the Statement interface.

PreparedStatement are pre-compiled and hence their execution is much faster than that of Statements.



```
PreparedStatement ps1 =  
con.prepareStatement("insert into employeeList  
values ('Heartin',2)");
```

CallableStatement

CallableStatement extends the capabilities of a PreparedStatement to include methods

CallableStatement

CallableStatement extends the capabilities of a PreparedStatement to include methods



JDBC Drivers



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.

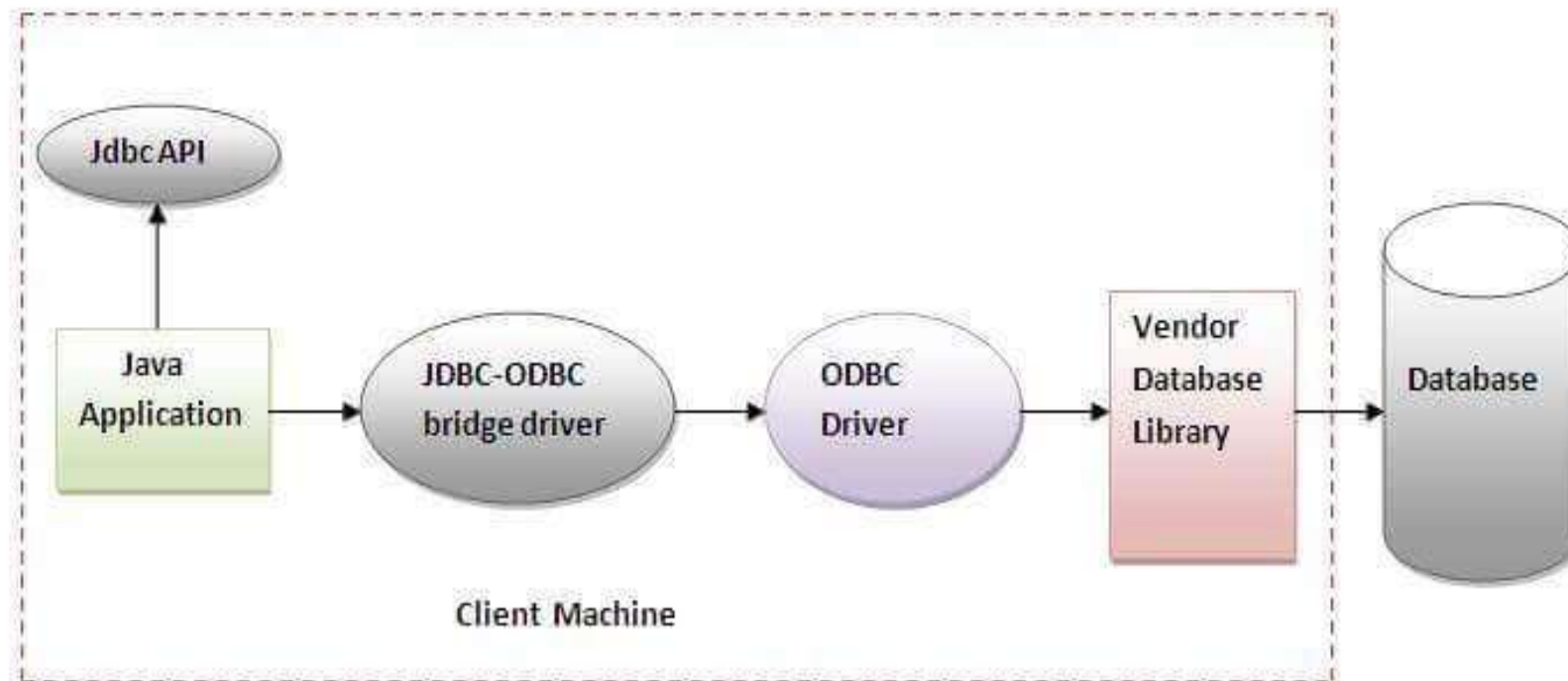


Figure- JDBC-ODBC Bridge Driver



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.

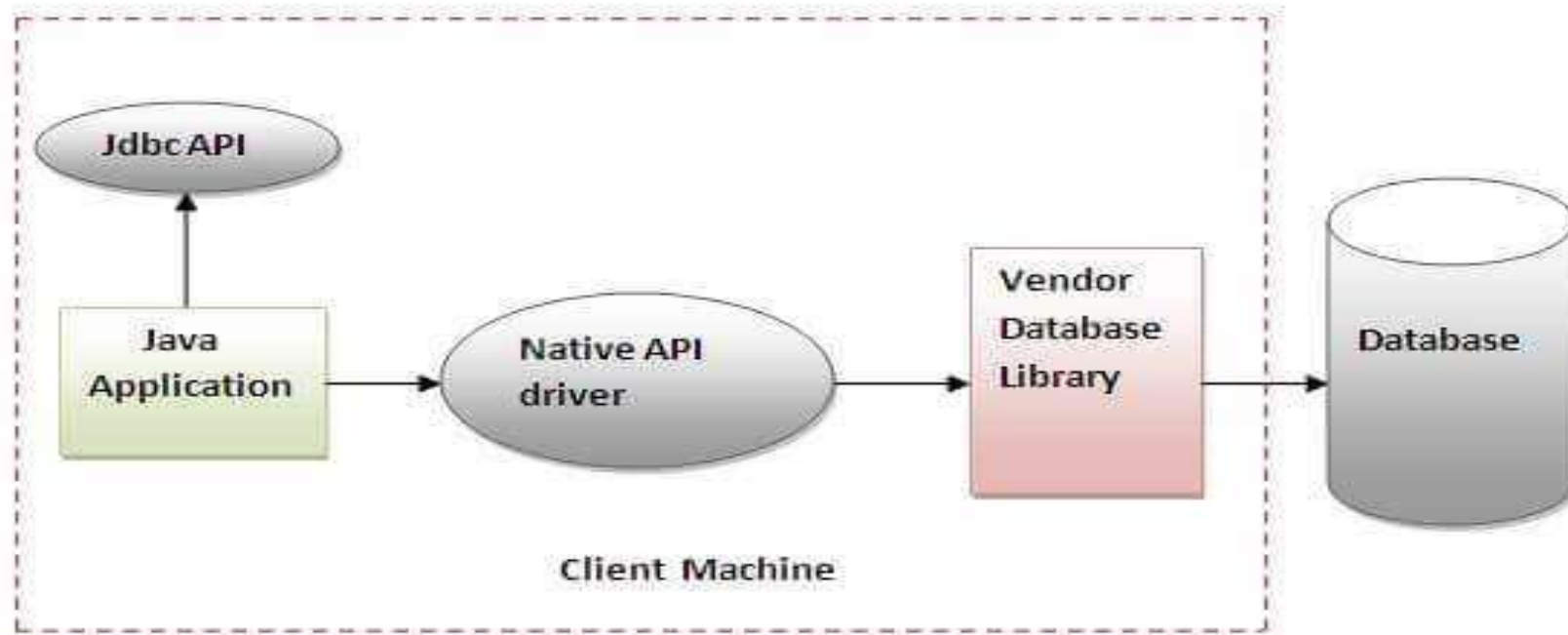


Figure- Native API Driver

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.

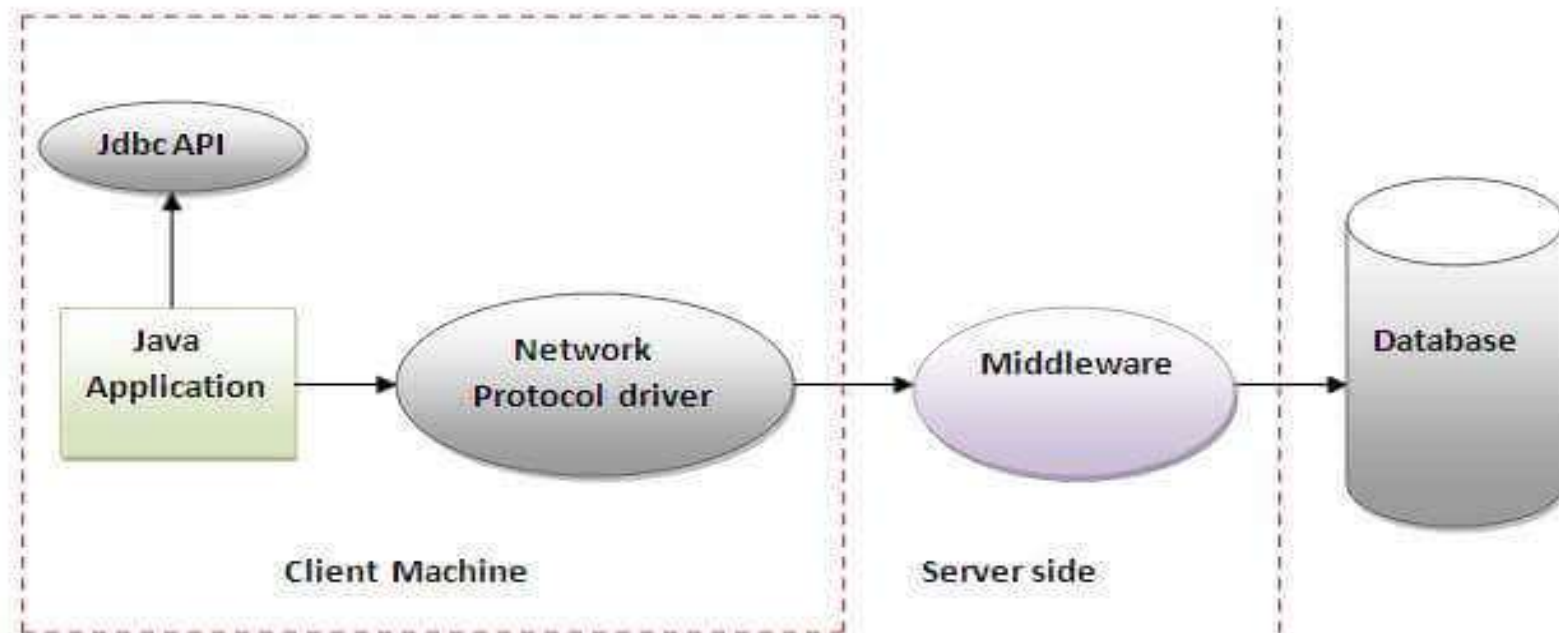


Figure- Network Protocol Driver



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.

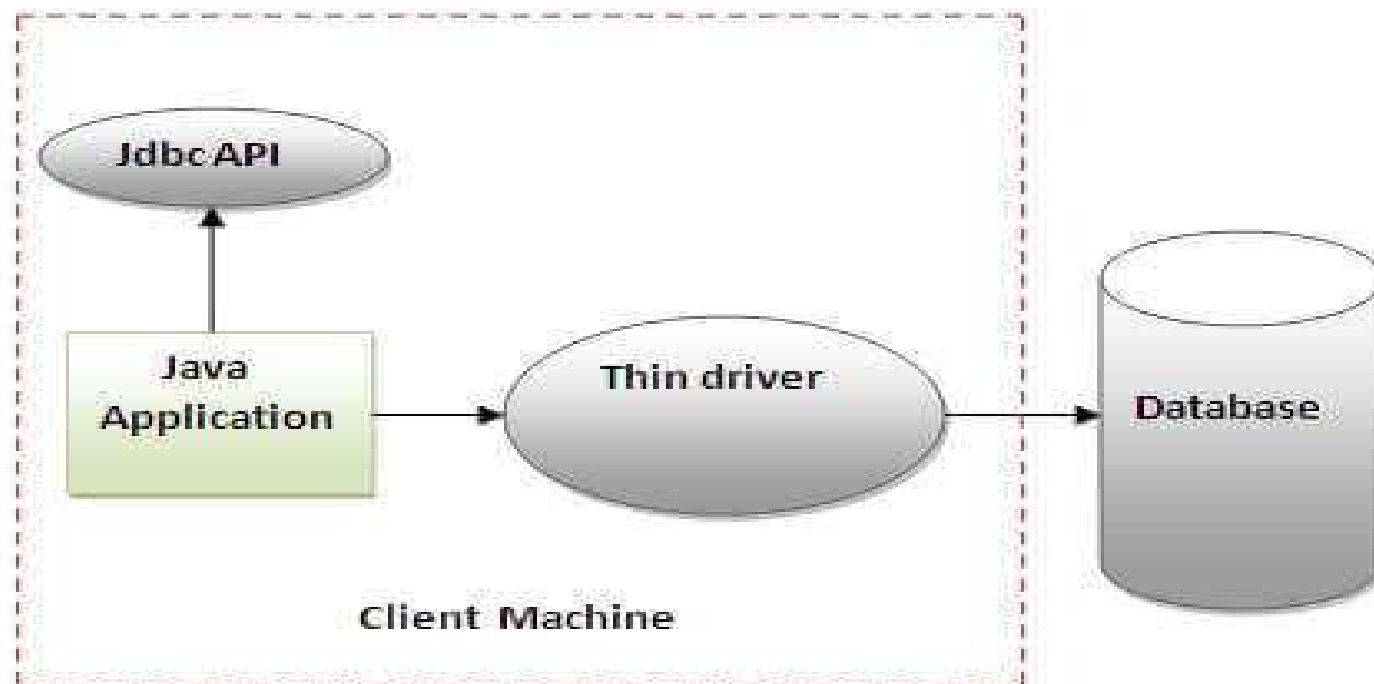


Figure- Thin Driver



Java Database Connectivity



Java Database Connectivity

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

These steps are as follows: Register the Driver class

- Create connection
- Create statement
- Execute queries
- Close connection

Java Database Connectivity

Register driver

Get connection

Create statement

Execute query

Close connection



1.Register the driver class

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

Syntax

```
public static void forName(String className)  
    throws ClassNotFoundException
```

2.Create the connection object

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

Syntax

```
public static Connection getConnection(String url)throws SQLException
```

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

```
public Statement createStatement()throws SQLException
```


4. Execute the query

The `executeQuery()` method of `Statement` interface is used to execute queries to the database.

Syntax

```
public ResultSet executeQuery(String sql)throws  
                                SQLException
```

Close the connection object

By closing connection object statement and ResultSet will be closed automatically.

Syntax

```
public void close()throws SQLException
```