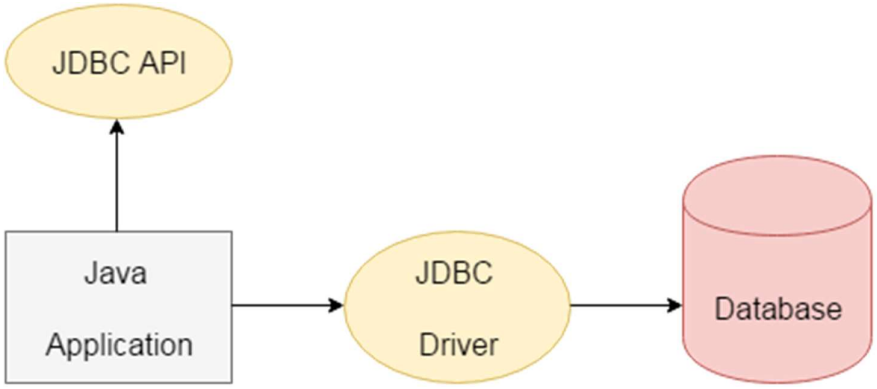
# Java JDBC Tutorial

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

We have discussed the above four drivers in the next chapter.

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.




The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface

- DatabaseMetaData interface
- RowSet interface

A list of popular *classes* of JDBC API are given below:

- DriverManager class
- Blob class
- Clob class
- Types class

## Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

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.

Do You Know

- How to connect Java application with Oracle and Mysql database using JDBC?
- What is the difference between Statement and PreparedStatement interface?
- How to print total numbers of tables and views of a database using JDBC?
- How to store and retrieve images from Oracle database using JDBC?
- How to store and retrieve files from Oracle database using JDBC?

# What is API

API (Application programming interface) is a document that contains a description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc.

---

# Topics in Java JDBC Tutorial

### 2) JDBC Drivers

In this JDBC tutorial, we will learn four types of JDBC drivers, their advantages and disadvantages.

---

### 3) 5 Steps to connect to the Database

In this JDBC tutorial, we will see the five steps to connect to the database in Java using JDBC.

---

### 4) Connectivity with Oracle using JDBC

In this JDBC tutorial, we will connect a simple Java program with the Oracle database.

---

### 5) Connectivity with MySQL using JDBC

In this JDBC tutorial, we will connect a simple Java program with the MySQL database.

---

### 6) Connectivity with Access without DSN

Let's connect java application with access database with and without DSN.

---

### 7) DriverManager class

In this JDBC tutorial, we will learn what does the DriverManager class and what are its methods.

---

### 8) Connection interface

In this JDBC tutorial, we will learn what is Connection interface and what are its methods.

---

### 9) Statement interface

In this JDBC tutorial, we will learn what is Statement interface and what are its methods.

## 10) ResultSet interface

In this JDBC tutorial, we will learn what is ResultSet interface and what are its methods. Moreover, we will learn how we can make the ResultSet scrollable.

## 11) PreparedStatement Interface

In this JDBC tutorial, we will learn what is benefit of PreparedStatement over Statement interface. We will see examples to insert, update or delete records using the PreparedStatement interface.

## 12) ResultSetMetaData interface

In this JDBC tutorial, we will learn how we can get the metadata of a table.

## 13) DatabaseMetaData interface

In this JDBC tutorial, we will learn how we can get the metadata of a database.

## 14) Storing image in Oracle

Let's learn how to store image in the Oracle database using JDBC.

## 15) Retrieving image from Oracle

Let's see the simple example to retrieve image from the Oracle database using JDBC.

## 16) Storing file in Oracle

Let's see the simple example to store file in the Oracle database using JDBC.

## 17) Retrieving file from Oracle

Let's see the simple example to retrieve file from the Oracle database using JDBC.

## 18) CallableStatement

Let's see the code to call stored procedures and functions using CallableStatement.

---

### 19) Transaction Management using JDBC

Let's see the simple example to use transaction management using JDBC.

---

### 20) Batch Statement using JDBC

Let's see the code to execute batch of queries.

---

### 21) JDBC RowSet

Let's see the working of new JDBC RowSet interface.

# JDBC Driver

1. JDBC Drivers
1. JDBC-ODBC bridge driver
2. Native-API driver
3. Network Protocol driver
4. Thin 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.

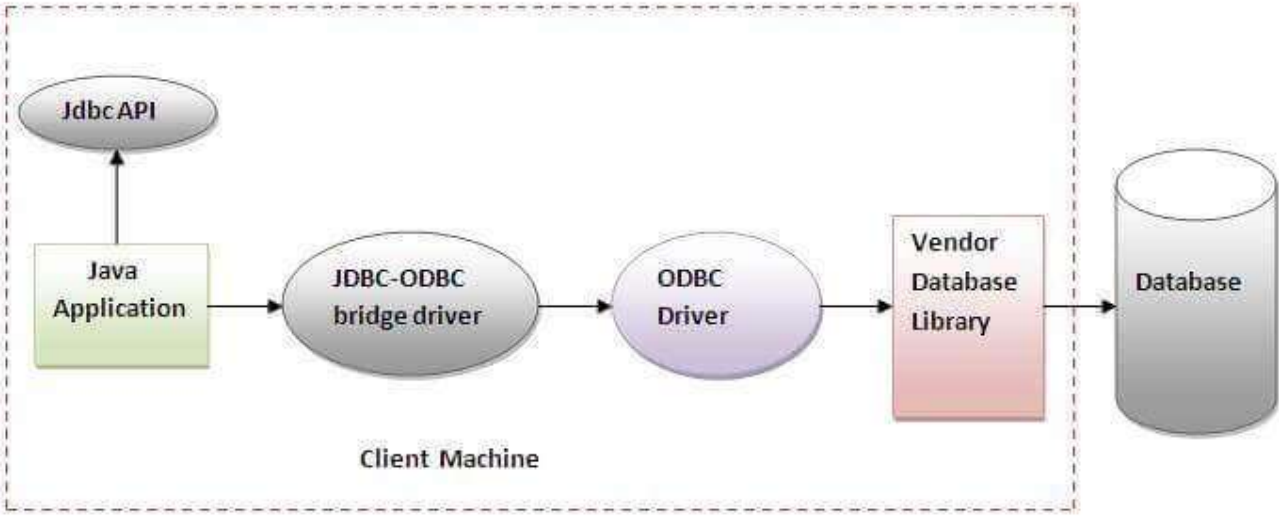



Figure- JDBC-ODBC Bridge Driver

***In Java 8, the JDBC-ODBC Bridge has been removed.***

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.

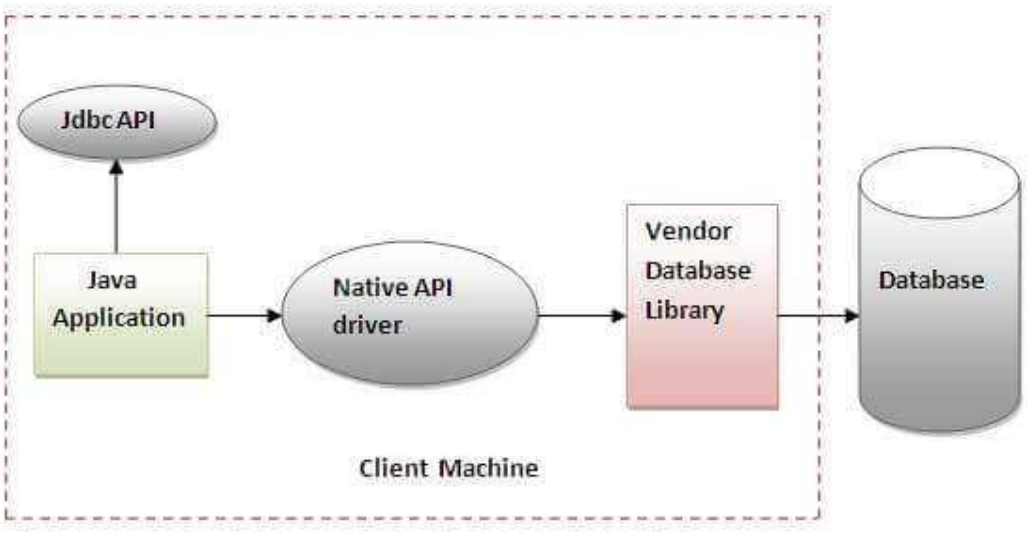



Figure- Native API Driver

### 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.

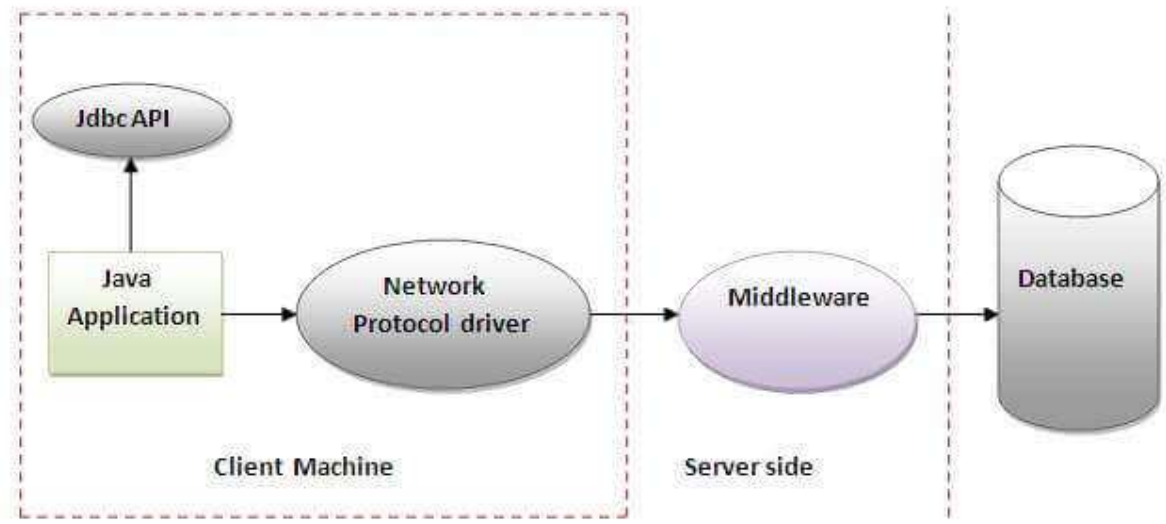



Figure- Network Protocol Driver

### 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.

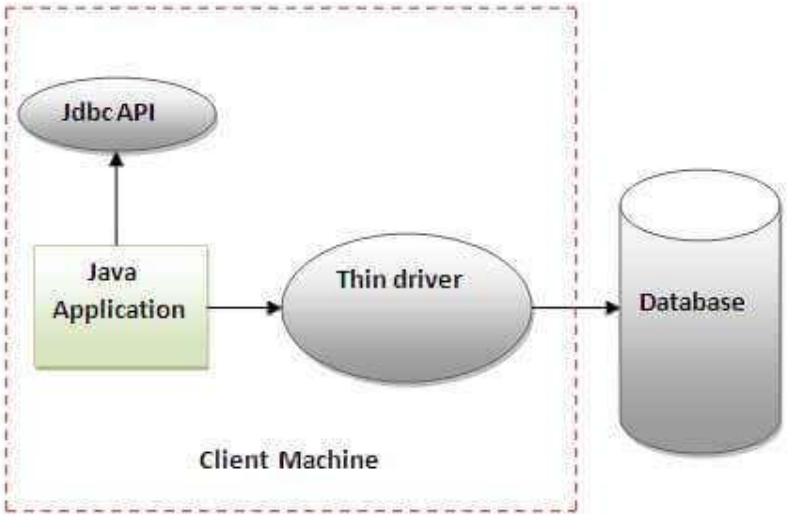



Figure- Thin Driver

## Advantage:

- Better performance than all other drivers.
- No software is required at client side or server side.

## Disadvantage:

- Drivers depend on the Database.