JDBC Driver

JDBC Driver is required to establish connection between application and database. It also helps to process SQL requests and generating result. The following are the different types of driver available in JDBC which are used by the application based on the scenario and type of application.

Type-1 Driver or JDBC-ODBC bridge

Type-2 Driver or Native API Partly Java Driver

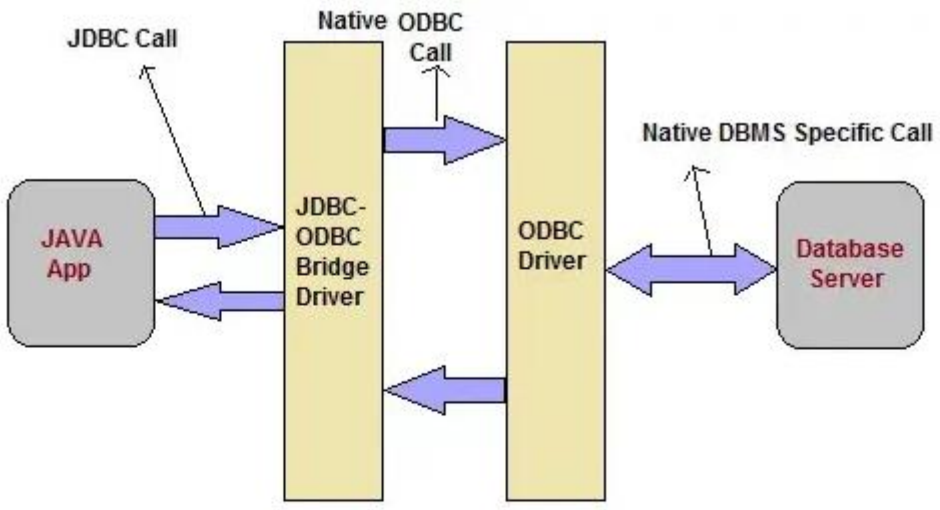
Type-3 Driver or Network Protocol Driver

Type-4 Driver or Thin Driver

1. JDBC-ODBC bridge

Type-1 Driver act as a bridge between JDBC and other database connectivity mechanism(ODBC). This driver converts JDBC calls into ODBC calls and redirects the request to the ODBC driver.

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



Advantages

Easy to use

Allow easy connectivity to all database supported by the ODBC Driver.

Disadvantages

Slow execution time

Dependent on ODBC Driver.

Uses Java Native Interface(JNI) to make ODBC call.

- **Steps involved in JDBC ODBC connection**

Java Database Connectivity with 5 Steps

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

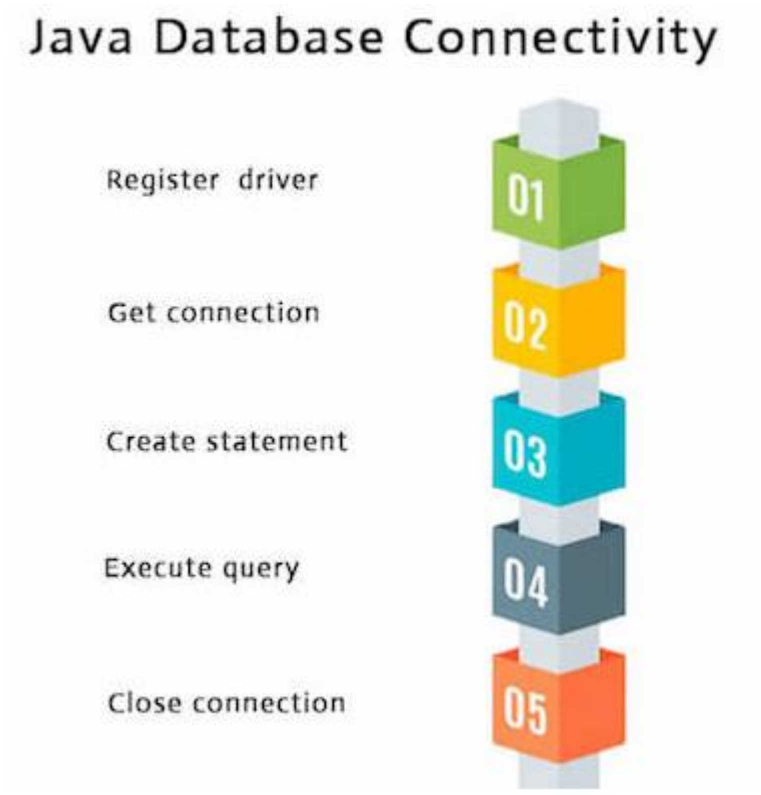
Register the Driver class

Create connection

Create statement

Execute queries

Close connection

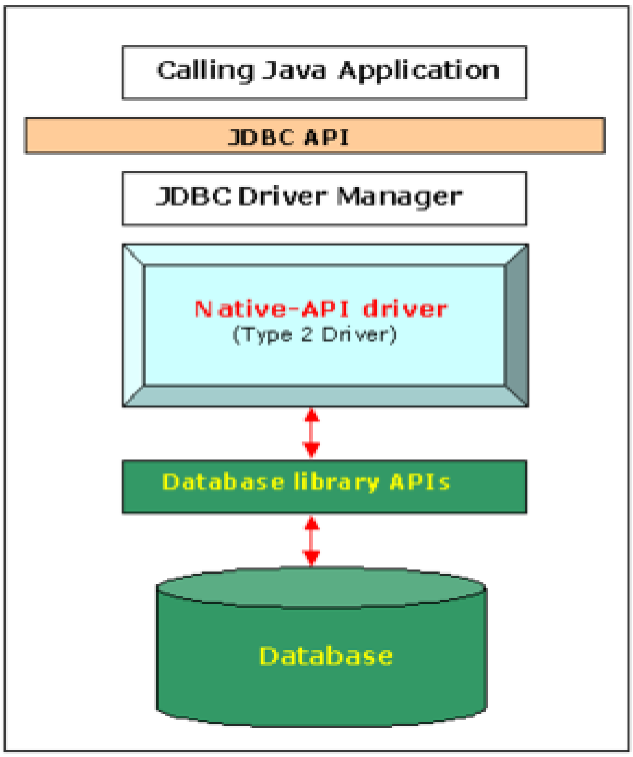


2. Native API Partly Java Driver

The JDBC type 2 driver, also known as the Native-API driver, is a database driver implementation that uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. For example: Oracle OCI driver is a type 2 driver.

Advantages

As there is no implementation of JDBC-ODBC bridge, it may be considerably faster than a Type 1 driver.



Disadvantages

The vendor client library needs to be installed on the client machine.

Not all databases have a client-side library.

This driver is platform dependent.

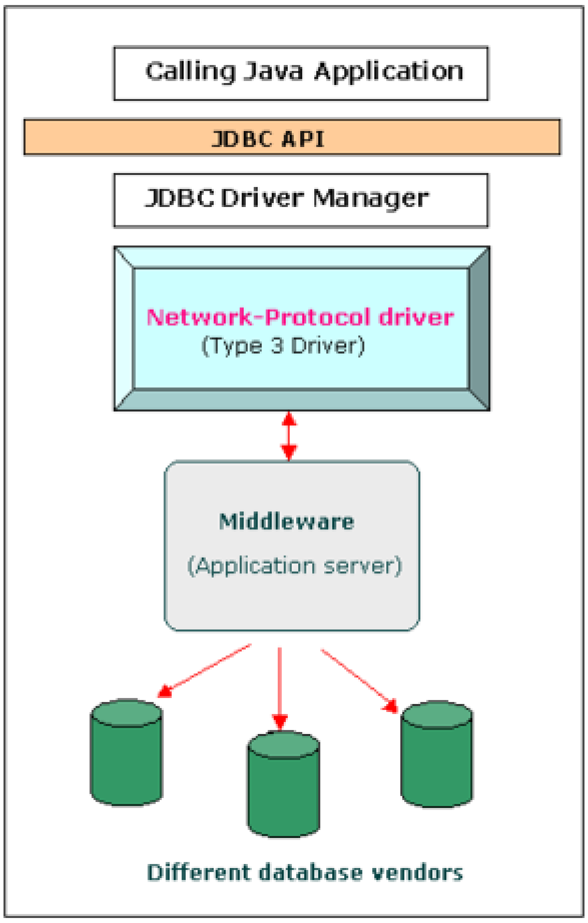
This driver supports all Java applications except applets.

3. Network Protocol Driver

The JDBC type 3 driver, also known as the Pure Java driver for database middleware,[7] is a database driver implementation which makes use of a middle tier between the calling program and the database. The middle-tier (application server) converts JDBC calls directly or indirectly into a vendor-specific database protocol.

The same client-side JDBC driver may be used for multiple databases. It depends on the number of databases the middleware has been configured to support. The type 3 driver is platform-independent as the platform-related differences are taken care of by the middleware. Also, making use of the middleware provides additional advantages of security and firewall access.

Functions



Sends JDBC API calls to a middle-tier net server that translates the calls into the DBMS-specific network protocol. The translated calls are then sent to a particular DBMS.

Follows a three-tier communication approach.

Can interface to multiple databases – Not vendor specific.

The JDBC Client driver written in java, communicates with a middleware-net-server using a database independent protocol, and then this net server translates this request into database commands for that database.

Thus the client driver to middleware communication is database independent.

Advantages

Since the communication between client and the middleware server is database independent, there is no need for the database vendor library on the client. The client need not be changed for a new database.

The middleware server (which can be a full-fledged J2EE Application server) can provide typical middleware services like caching (of connections, query results, etc.), load balancing, logging, and auditing.

A single driver can handle any database, provided the middleware supports it.

E.g.: IDA Server

Disadvantages

Requires database-specific coding to be done in the middle tier.

The middleware layer added may result in additional latency, but is typically overcome by using better middleware services.

4. Thin Driver

The JDBC type 4 driver, also known as the Direct to Database Pure Java Driver, is a database driver implementation that converts JDBC calls directly into a vendor-specific database protocol.

Written completely in Java, type 4 drivers are thus platform independent. They install inside the Java virtual machine of the client. This provides better performance than the type 1 and type 2 drivers as it does not have the overhead of conversion of calls into ODBC or database API calls. Unlike the type 3 drivers, it does not need associated software to work.

As the database protocol is vendor specific, the JDBC client requires separate drivers, usually vendor supplied, to connect to different types of databases.

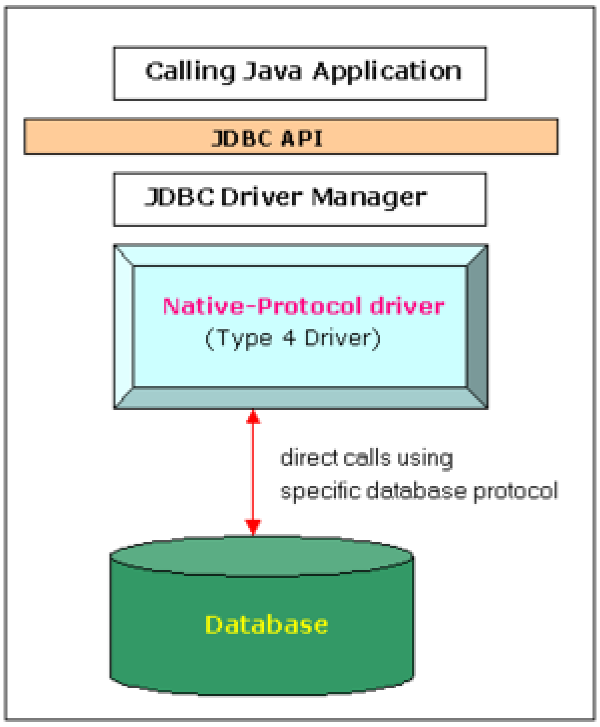
Advantages

Completely implemented in Java to achieve platform independence.

These drivers don't translate the requests into an intermediary format (such as ODBC).

The client application connects directly to the database server. No translation or middleware layers are used, improving performance.

The JVM can manage all aspects of the application-to-database connection; this can facilitate debugging.



Disadvantages

Drivers are database specific, as different database vendors use widely different (and usually proprietary) network protocols.