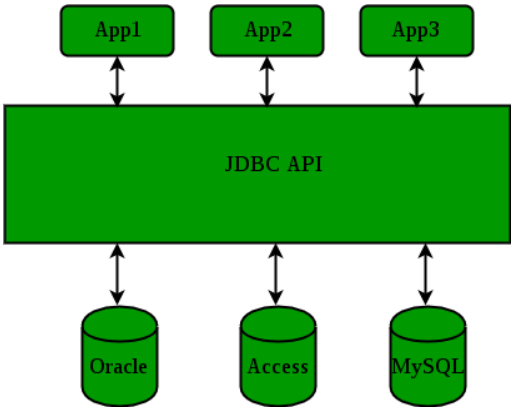
Introduction to JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) can access any kind of tabular data, mainly data stored in a Relational Database. It is part of Java Standard Edition platform, from Oracle Corporation. It acts as a middle layer interface between java applications and database. The JDBC classes are contained in the Java Package **java.sql** and **javax.sql**.

JDBC helps you to write Java applications that manage these three programming activities:

1. Connect to a data source, like a database.
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query



ODBC:

ODBC stands for Open Database Connectivity, a standard database access method developed by Microsoft Corporation. The goal of ODBC is to make it possible to access any data from any application, regardless of which Data Base Management System (DBMS) is handling data. ODBC manages this by inserted a middle layer, called a driver, between an application and the DBMS. The purpose of this layer is to translate the queries of the application into commands that the DBMS understands.

Database Management:

A database is a collection of related information and a Data Base Management System (DBMS) is software that provides you with a mechanism to retrieve data from the database. There are lots of DBMS/RDBMS products available to you, for example, MS-Access, MS-SQL Server, Oracle, etc. The list is never-ending. Each of these RDBM's stores the data in their own format. MS-Access stores the data in .MDB file format whereas MS-SQL Server stores the data in a .DAT file format. Without database the client is useless.

JDBC Driver Manager

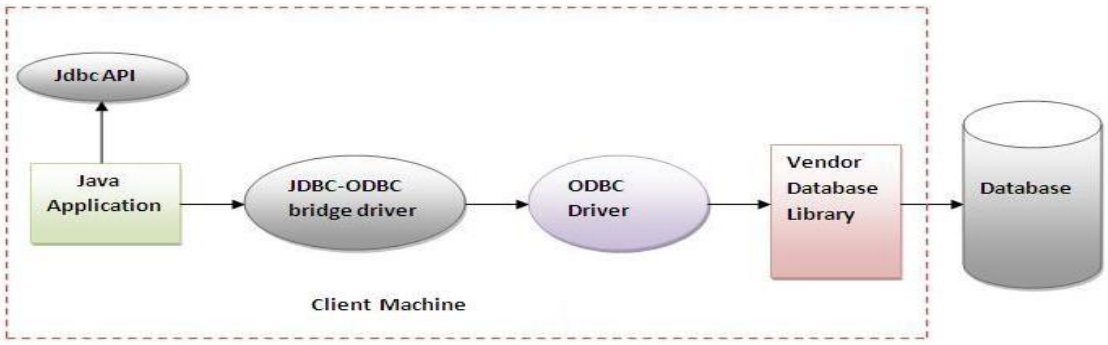
JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

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

1) Type-1 driver or JDBC-ODBC bridge driver

Type-1 driver or 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. Type-1 driver is also called Universal driver because it can be used to connect to any of the databases.

- This driver software is built-in with JDK so no need to install separately.



JDBC-ODBC bridge driver

Advantages:

- It is a database independent driver.
- easy to use.
- can be easily connected to any database.

Disadvantages:

- As a common driver is used in order to interact with different databases, the data transferred through this driver is not so secured.
- The ODBC bridge driver is needed to be installed in individual client machines.
- Type-1 driver isn't written in java, that's why it isn't a portable driver.

2) Native-API driver

The Native API driver uses the client -side libraries of the database. This driver converts JDBC method calls into native calls of the database API. In order to interact with different database, this driver needs their local API, that's why data transfer is much more secure as compared to type-1 driver.

- Driver needs to be installed separately in individual client machines
- The Vendor client library needs to be installed on client machine.
- Type-2 driver isn't written in java, that's why it isn't a portable driver
- It is a database dependent driver.

.

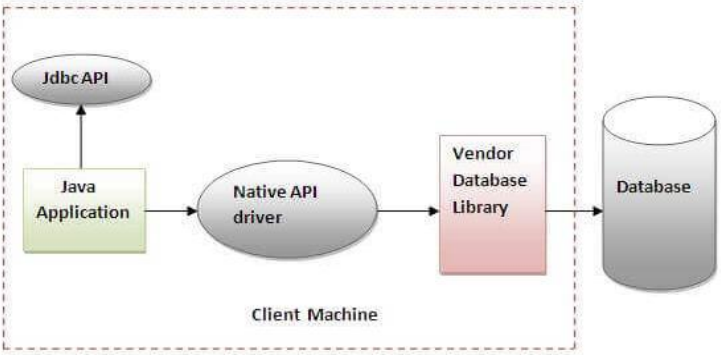


Figure- Native API Driver

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. Here all the database connectivity drivers are present in a single server, hence no need of individual client-side installation.

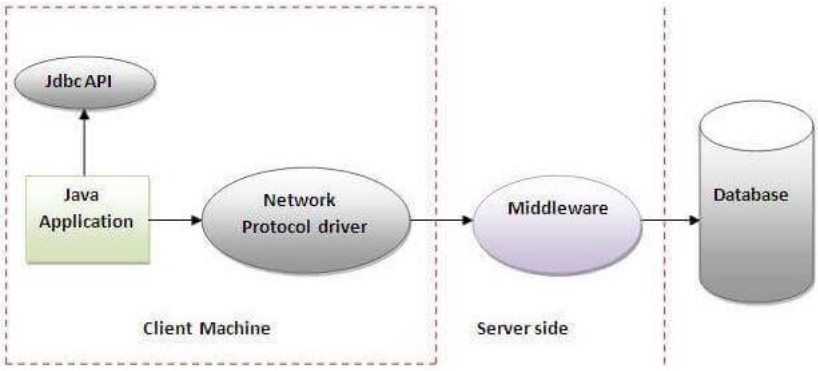


Figure- Network Protocol Driver

Advantage:

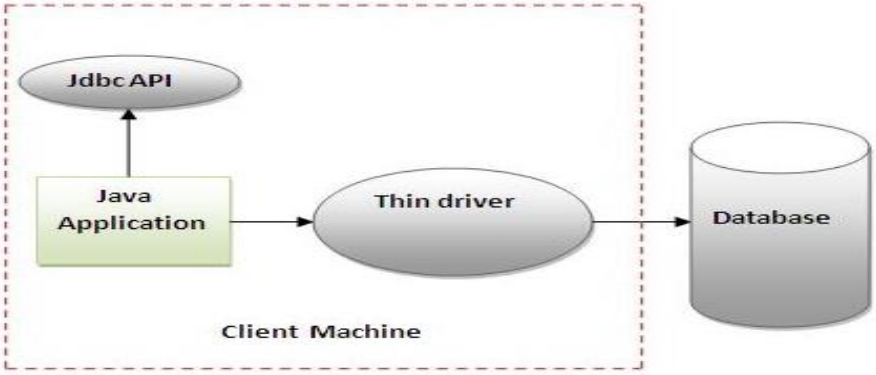
- Type-3 drivers are fully written in Java, hence they are portable drivers.
- No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.
- Switch facility to switch over from one database to another database.

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

Type-4 driver is also called native protocol driver. This driver interacts directly with database. It does not require any native database library that is why it is also known as Thin Driver.



Advantage:

- Does not require any native library and Middleware server, so no client-side or server-side installation.
- It is fully written in Java language, hence they are portable drivers.

Disadvantage:

- Drivers depend on the Database.

Which Driver to use and when?

- If you are accessing one type of database, such as Oracle, Sybase, or IBM, the preferred driver type is type-4.
- If your Java application is accessing multiple types of databases at the same time, type 3 is the preferred driver.
- Type 2 drivers are useful in situations, where a type 3 or type 4 driver is not available yet for your database.
- The type 1 driver is not considered a deployment-level driver, and is typically used for development and testing purposes only.

JDBC 4.0 API

JDBC 4.0 API is mainly divided into two package

1. java.sql
2. javax.sql

java.sql package

This package include classes and interface to perform almost all JDBC operation such as creating and executing SQL Queries.

Important classes and interface of java.sql package

classes/interface	Description
java.sql.BLOB	Provide support for BLOB(Binary Large Object) SQL type.
java.sql.Connection	creates a connection with specific database
java.sql.CallableStatement	Execute stored procedures
java.sql.CLOB	Provide support for CLOB(Character Large Object) SQL type.
java.sql.Date	Provide support for Date SQL type.
java.sql.Driver	create an instance of a driver with the DriverManager.
java.sql.DriverManager	This class manages database drivers.
java.sql.PreparedStatement	Used to create and execute parameterized query.
java.sql.ResultSet	It is an interface that provide methods to access the result row-by-row.
java.sql.Savepoint	Specify savepoint in transaction.
java.sql.SQLException	Encapsulate all JDBC related exception.
java.sql.Statement	This interface is used to execute SQL statements.

javax.sql package

This package is also known as JDBC extension API. It provides classes and interface to access server-side data.

Important classes and interface of javax.sql package

classes/interface	Description
javax.sql.ConnectionEvent	Provide information about occurence of event.
javax.sql.ConnectionEventListener	Used to register event generated by **PooledConnection** object.
javax.sql.DataSource	Represent the **DataSource** interface used in an application.
javax.sql.PooledConnection	provide object to manage connection pools.