**The Four Types of JDBC drivers**

A Driver is a program that controls a device for it to accomplish a task. *Java database Connectivity* (JDBC) allows for accessing any form of tabular data from any source. Relational databases are most common, but JDBC drivers for XML, Excel, or legacy data sources can be obtained.

JDBC driver implementations vary because of the wide variety of operating systems and hardware platforms in which Java operates. We have divided the implementation types into four categories:

**1. JDBC-ODBC Bridge Driver:**

The JDBC-ODBC bridge driver uses ODBC (*Open Database Connectivity*), a standard or open application programming interface (API) for accessing a database, driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. They are drivers that implement the JDBC API as a mapping to another data access API, such as Open Database Connectivity (ODBC). Drivers of this type are generally dependent on a native library, which limits their portability. It can be easily connected to any database.

The Performance is degraded because JDBC method call is converted into the ODBC function calls. And also, the ODBC driver needs to be installed on the client machine.

**2. Native-API driver**:

Drivers that are written partly in the Java programming language and partly in native code. The drivers use a native client library specific to the data source to which they connect. Because of the native code, their portability is limited. The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native C/C++ API calls, which are unique to the database. It is not written entirely in java. Their performance upgraded compared to the JDBC-ODBC bridge driver.

The Native driver and the Vendor client library need to be installed on each client machine.

**3. Network-Protocol driver (middleware 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. This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases. Your application server might use a Type 1, 2, or 4 drivers to communicate with the database, understanding the nuances will prove helpful.

No client-side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc. Network support is required on client machine and requires database-specific coding to be done in the middle tier, thus maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

**4. Database-Protocol driver (Pure Java driver):**

In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself. This kind of driver is extremely flexible, you don't need to install special software on the client or server. Further, these drivers can be downloaded dynamically.