JDBC SHORT NOTES

Abstract

This document contains short notes on JDBC, their types with diagrams.

Rohit Deshbhratar

[Email address]

JDBC

Introduction:

Java DataBase Connectivity, commonly known as JDBC, is an API for Java programming language that defines how a client may access a database. JDBC API uses JDBC driver written in Java. JDBC is platform independent.

What is API (Application Programming Interface)?

API is a document that contains description of all features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other.

JDBC Components

JDBC includes four components

1. **The JDBC API**

The JDBC API provides programmatic access to relational data from the Java programming language. Using the JDBC API, applications can execute SQL statements, retrieve results, and propagate changes back to an underlying data source. The JDBC API can also interact with multiple data sources in a distributed, heterogeneous environment.

2. **JDBC Driver Manager**

The JDBC *DriverManager* class defines objects which can connect Java applications to a JDBC driver. *DriverManager* has traditionally been the backbone of the JDBC architecture. It is quite small and simple.

3. **JDBC Test Suite**

The JDBC driver test suite helps you to determine that JDBC drivers will run your program. These tests are not comprehensive or exhaustive, but they do exercise many of the important features in the JDBC API.

4. **JDBC-ODBC Bridge**

The Java Software Bridge provides JDBC access via ODBC drivers. Note that you need to load ODBC binary code onto each client machine that uses this driver. As a result, the ODBC driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in a three-tier architecture.

JDBC Architecture

The JDBC API supports both two-tier and three-tier processing models for database access.

- **Two tier architecture**

In the two-tier model, a Java applet or application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. The network can be an intranet, which, for example, connects employees within a corporation, or it can be the Internet.

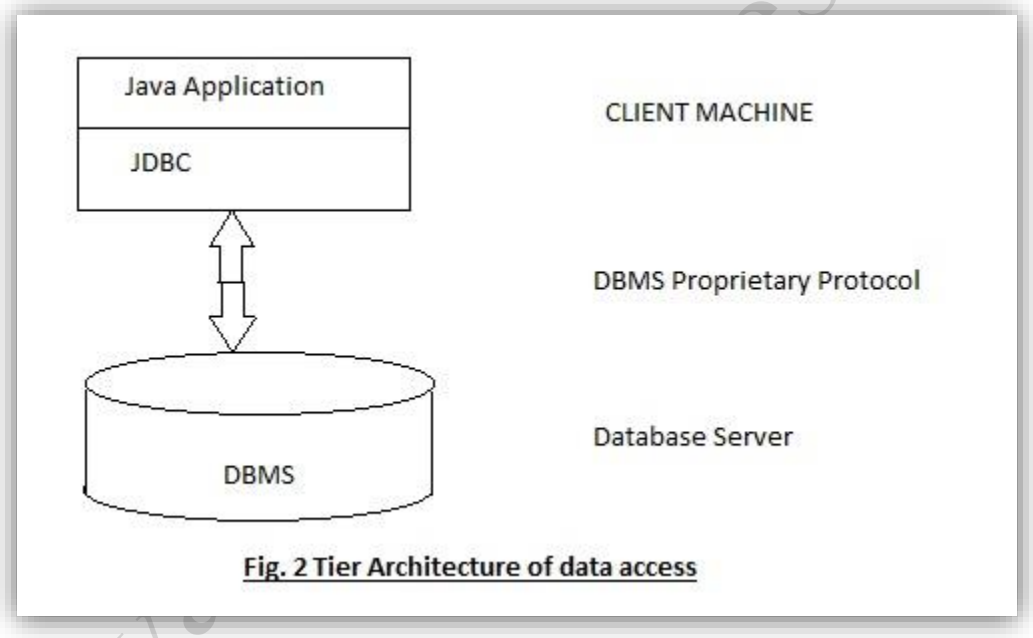

Fig. 2 Tier Architecture of data access

- **Three tier architecture**

In the three-tier model, commands are sent to a "middle tier" of services, which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that it simplifies the deployment of applications. Finally, in many cases, the three-tier architecture can provide

performance advantages. With enterprises increasingly using the Java programming language for writing server code, the JDBC API is being used more and more in the middle tier of a three-tier architecture. Some of the features that make JDBC a server technology are its support for connection pooling, distributed transactions, and disconnected rowsets. The JDBC API is also what allows access to a data source from a Java middle tier.

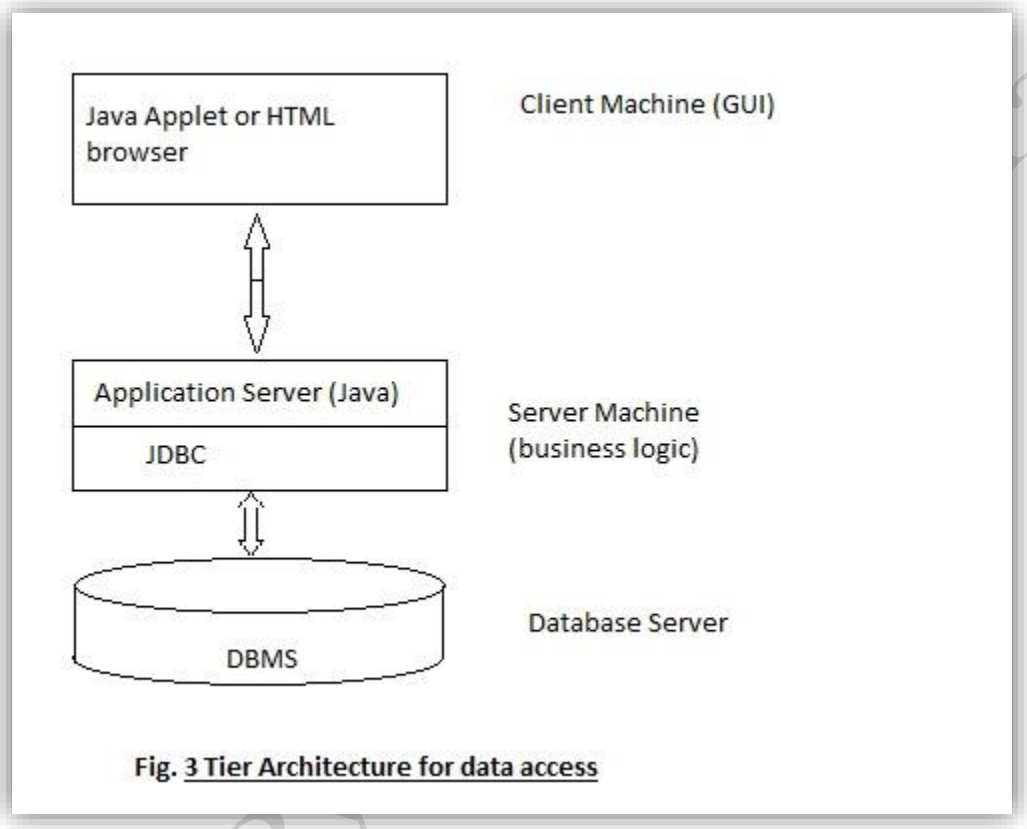

Fig. 3 Tier Architecture for data access

JDBC Driver

A JDBC driver is a software component enabling a java application to interact with a database. JDBC requires drivers for each database. The JDBC driver gives out the connection to the database and implements the protocol for transferring the query and result between client and database. JDBC drivers are client side adaptors that convert request from java program to a protocol that that the DBMS can understand. There are four types of driver

1. JDBC-ODBC Bridge Driver (Type 1)
2. Native API Driver (Type 2/Partially Java Driver)
3. Network Protocol Driver (Type 3/Fully Java Driver)
4. Thin Driver (Type 4/Fully Java Driver)

1. JDBC-ODBC Bridge Driver (Type 1)

- Uses ODBC bridge driver to connect to database.
- Converts JDBC methods call into function call.
- ODBC drivers need to be installed on client machine.
- This driver is platform dependent.

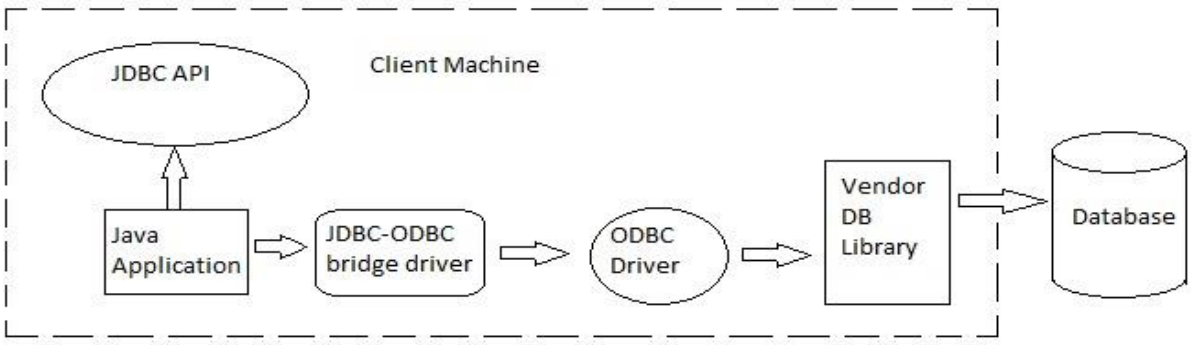

Fig. JDBC ODBC Bridge Driver

Advantages

- Easy to connect
- Directly connected to database

Disadvantage

- Does not support the complete java command set and are limited by the functionality of the ODBC driver.
- Needs to be installed on client machine.
- Slow, as compared to other drivers.

2. Native API Driver (Type 2/Partially Java Driver)

- It uses client side libraries of the database.
- The driver converts the JDBC method call into native call.
- Native drivers must be installed on client machines.
- Vendor client libraries need to be installed on client machine.
- It is not written entirely in java.

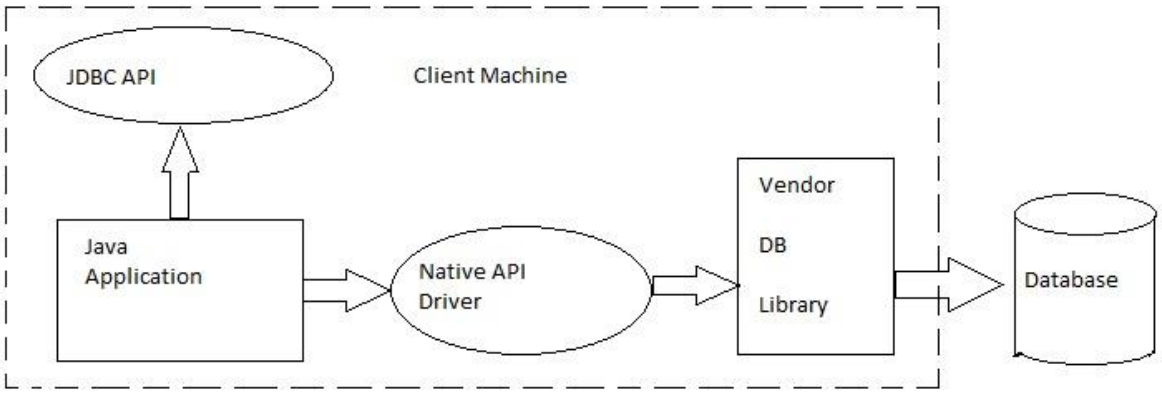

Fig. Native API (Type 2)

Advantage

- Better performance as compared to Type 1 driver.

Disadvantage

- Driver is platform dependent.
- The vendor client library needs to be installed.
- Native driver needs to be installed on each machine.

3. Network Protocol Driver (Type 3/Fully Java Driver)

- Uses middleware that convert JDBC call directly or indirectly into vendor specific database protocol.
- Fully written in Java.
- Database specific coding to be done in middle tier.

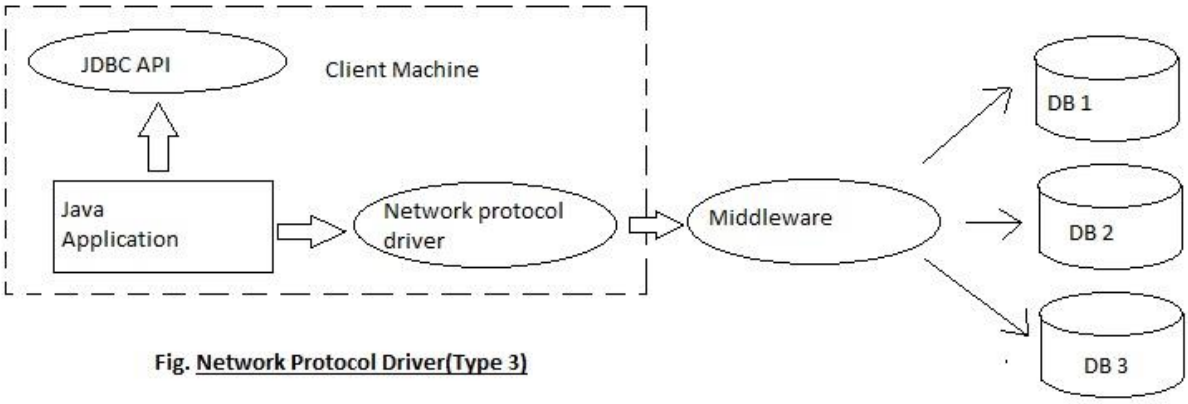

Fig. Network Protocol Driver(Type 3)

Advantage

- No client side library is required.

Disadvantage

- Network support is required on client machine.
- Database specific coding is done in middleware.

4. Thin Driver (Type 4/Fully Java Driver)

- Converts JDBC calls directly into vendor specific database protocol.
- Fully written in Java.
- User needs different drivers for each database.
- Performance is good.

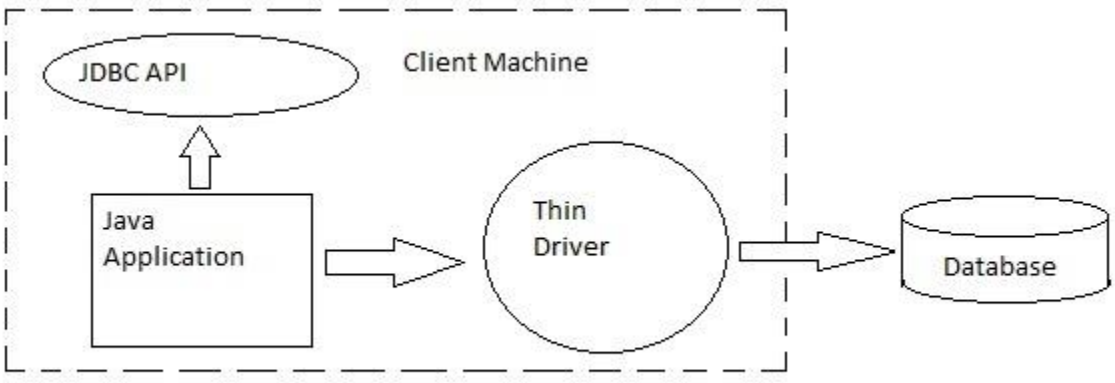

Fig. Thin Driver (Type4)

Advantage

- Performance is good as compared to other drivers.
- No software is required for client or server side.

Disadvantage

- Different types of drivers required for different database.

Steps to connect to database

1. Register the driver class.
2. Create connection.
3. Create Statement.
4. Execute queries.
5. Close the connection.

Package for JDBC connection

1. **java.sql.*;**
   Provides the API for accessing and processing data stored in a data source using Java.
2. **javax.sql.*;**
   Provides API for server side data source access and processing from java.