Advanced JAVA:-

★ Introduction to JDBC:-

- JDBC Stands for Java Database Connectivity, it is a Java-Based API that enables Java application to interact with various database.

  Database → MySQL, Oracle, Postsql ...

- The core idea idea behind ~~is~~ JDBC is to provide a Standard method for JAVA application to connect to a database, Send SQL Commands and process the result.

★ Why JDBC:- (Need of JDBC)

(1) Independent of particular database:- JDBC allows your Java code to interact with different database without being dependent on particular database system.

(2) Ease of use: With JDBC, we can handle complex database interaction using Java code.

(3) Wide Adoption:- It is widely used in Java application, especially in enterprise-level application, because it integrates seamlessly with other JAVA technology.

★ Basic Workflow / How JDBC is established in Code:-

(1) Load the appropriate JDBC driver
(2) Established a connection to the database
(3) Create a 'Statement' to execute execute SQL queries.

(4) Execute SQL queries to perform operation like insert, update, delete or select.

(5) Process the result

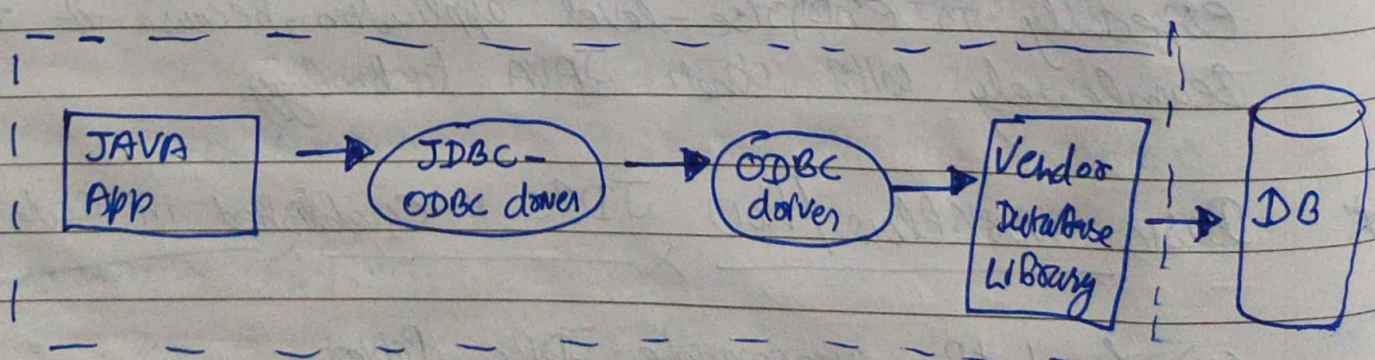
(6) Close the connection

★ JDBC drivers:-

- JDBC drivers are responsible for establishing a connection between a Java application and a database.
- JDBC drivers are basically software components which implements interface in JDBC APIs to enable JAVA applications to interact with the database.
- There are 4 type of JDBC are there:-

(1) Type 1: JDBC-ODBC Bridge driver:-

This driver acts as an Bridge between JDBC and ODBC (Open-database connectivity). It uses the ODBC driver to connect to the database.



Advantage

→ Simple to Use
→ Can connect to database which support ODBC

Disadvantage

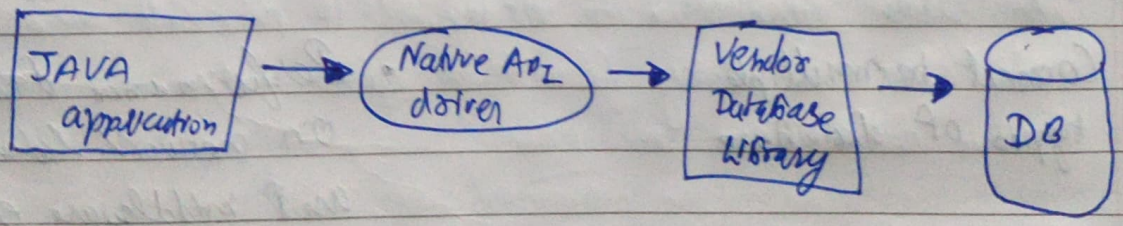
→ Performance is slower than other drivers
→ Requires ODBC driver and native libraries, which add complexity.

(2) Native-API driver (Partially Java Driver):-

- This drivers 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 safer than type-1 driver

- This driver is not fully written in JAVA, that's why its called Partially Java driver.

Working → This drivers converts JDBC calls into database-specific calls using native libraries provided by the database vendor.



Advantage:-

1) Having better performance than Type-1 driver

2) More secure

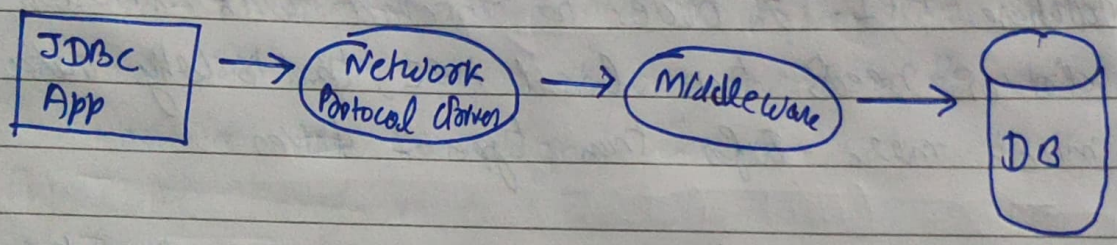
Disadvantage:-

1) Individual installation on client-machine

2) It's database dependent driver

(3) Network protocol driver (Type-3 driver)

- The network protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol.
- All the database connectivity drivers are present in a single server, so no need to install individually in client-side.



Working → The driver communicates with a middleware server then talks to the database. The middleware server handles database-specific protocols.

Advantage

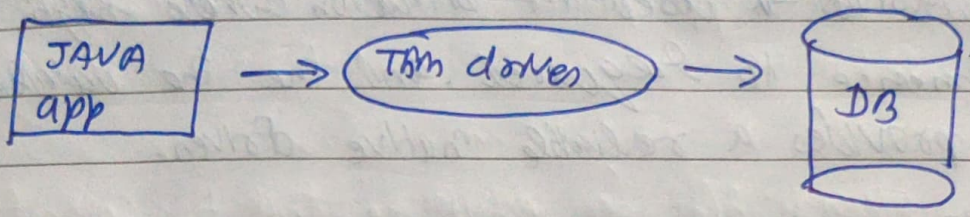
- Connect to multiple types of database
- No need for client-side native libraries
- Fully written in JAVA which makes it portable.

Disadvantage

- Performance depends on network latency and middleware efficiency
- Requires additional setup of middleware servers.

(4) TBm-drives (Type 4 driver) :-

It's also called native protocol drives. This drives interact directly with database. It does not require any native database library, that's why we called as TBm driver.



Working → This drivers converts JDBC calls directly into the database-specific protocol. It communicates directly with the database server.

Pros/Advantage :-

1) Best performance because it eliminates the need of middleware

2) 100% Java made, making it platform independent

Disadvantage :-

1) Database-specific, so the application may need adjustment when switching databases.

★ Uses of drivers according to condition:-

(1) Type-1 driver → Suitable for testing and prototype when no other drivers are available.

(2) Type-2 driver → Used in situation where high performance is required. and the the database vendor provides a reliable native driver.

(3) Type-3 driver → Ideal for the places like enterprise where different type of database need to be accessed from a ~~co~~ centralized location.

(4) Type-4 driver → Most commonly used in modern application due to its efficency and simplicity.

★ Architecture of JDBC:-

The JDBC architecture is designed to provide a seamless and consistent way for JAVA application to interact with various database.

It consist of several layer, each responsible for different aspects of database interaction.

Mainly its consist of 5 layer:-

1) JAVA application layer
2) JDBC API layer

3) Drives managers layer
4) JDBC drover layer
5) DataBase Layer

1) JAVA application layer:-

This is the topmost layer where JAVA application resides. The application uses JDBC API to communicate with the database. It contains the logic to establish a connection, execute SQL queries, retrive results and handle errors. errors.

2) JDBC API Layers:-

The JDBC API Layers provides the standard set of interfaces and classes that your application uses to interact with databases. This layer abstracts the underlying database details, so the application doesn't need to worry about database-specific implementations.

Key component in this layer:-

1) Driver Managers: Manages the drivers and establishes the connection between the JAVA application and the the database.

2) Connection: Represents an active connection to the database.

3) Statement: Used to execute SQL queries and command.

4) Result set: Represents the result of a query execution.

3) Drivers Manager Layer:-

- It plays a crucial role in managing JDBC drivers. When a JAVA application requests a connection, the 'Driver Manager' determine which driver is appropriate based on the connection URL provided by the application. It then loads the appropriate drives to establish the connection.

- This layer is responsible for ensuring that the correct JDBC driver is used for the specific database the application wants to connect to.

4) JDBC driven layer:-

- The JDBC drivers layer is the intermediary between the JDBC API and the actual database. Each database vendor provides a JDBC driver that implements the JDBC API for their specific database.

- This layer converts the high-level JDBC calls from the application into low-level, database-specific calls the database servers can understand.

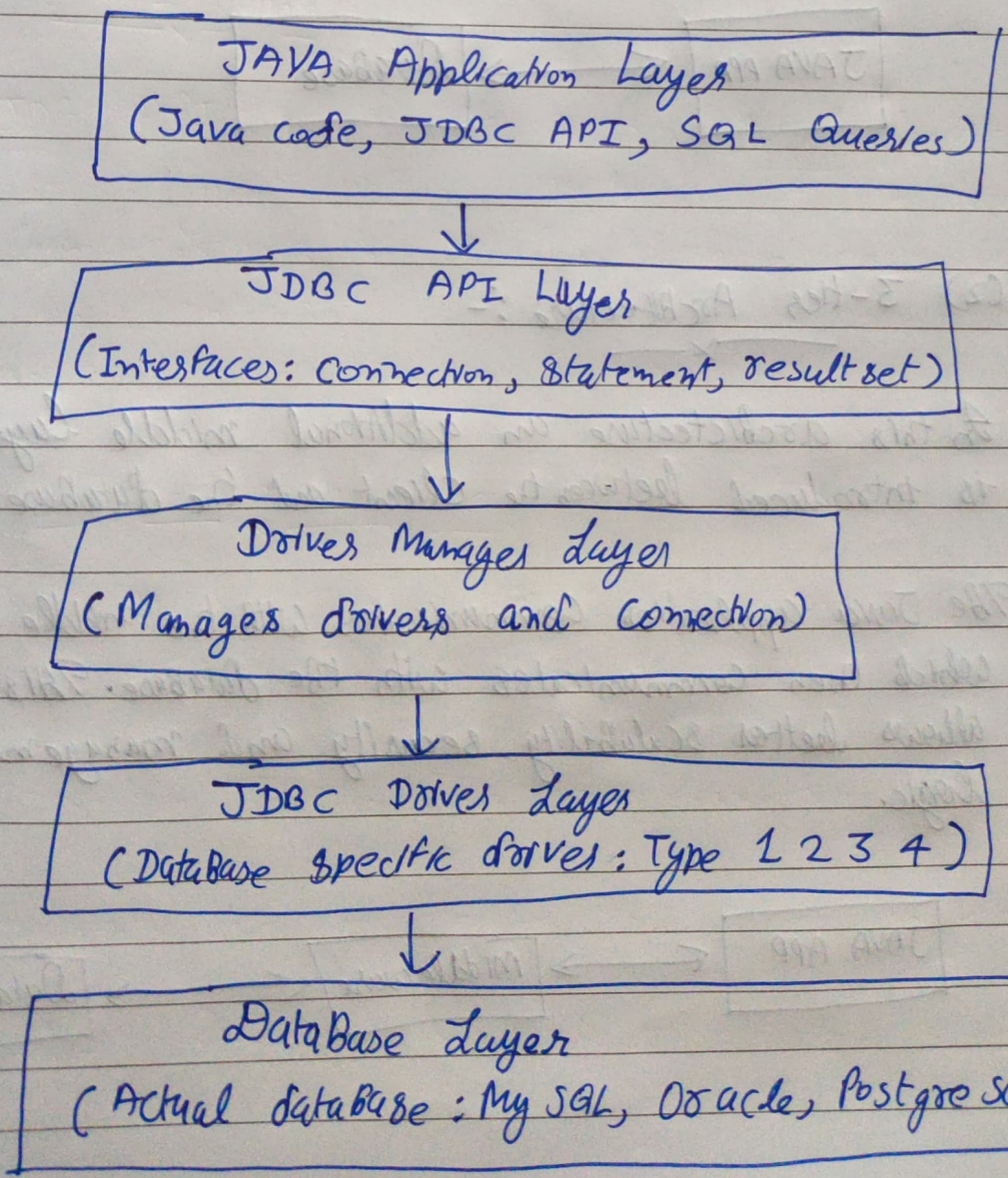
- Depending on the drivers type this conversion might involve different methods, such as using native API calls or network protocols.

5) DataBase Layer:-

- The database layer is the actual database server that stores the data and processes SQL commands. When the drives layer communicates with this layer, it executes SQL queries, update data and retrieves results, which are then passed back through the drives to the application.

- This layer contains the database engine, which performs operations like query parsing, optimization, execution and transaction management.

Flow diagram of JDBC Architecture:-

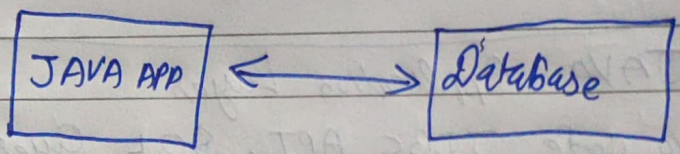


★ Types of JDBC Architecture:-

There are two types of JDBC architecture:-

(1) 2-tier Architecture (Client-Server Architecture)

In this architecture, the JAVA application directly interacts with the database. The client application sends SQL queries to the database server and receives the result. The entire database logic resides on the client side.



(2) 3-tier Architecture :-

- In this architecture an additional middle layer is introduced between the client and the database.
- The Java application communicates with the middle layer, which then communicates with the database. This separation allows better scalability, security and management logic.