Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

Foundation of Enterprise Programming

Study
Gu

ARNIKA PATEL
CSE, PIT
Parul University

1.1 JDBC

JDBC is an **API** that helps applications to communicate with databases, it allows Java programs to connect to a database, run queries, retrieve, and manipulate data. Because of JDBC, Java applications can easily work with different relational databases like MySQL, Oracle, PostgreSQL, and more.

1.2 JDBC Architecture

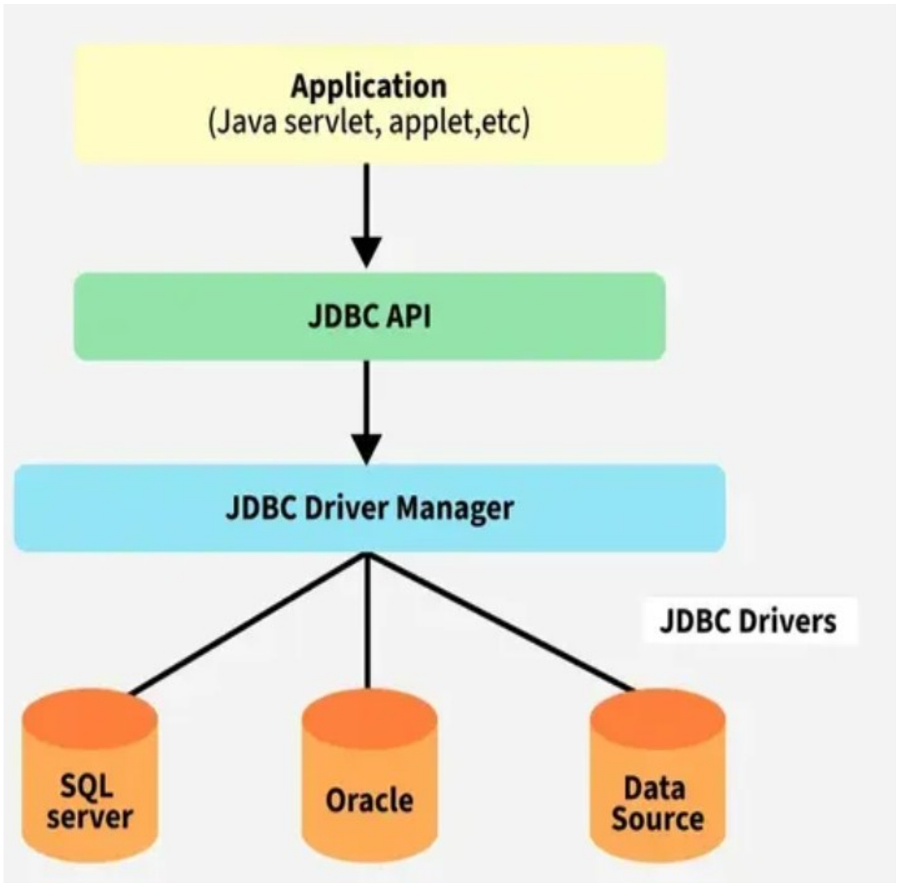


Fig. 1.1 JDBC Architecture

- **Application:** It can be a Java application or servlet that communicates with a data source.
- **The JDBC API:** It allows Java programs to execute SQL queries and get results from the database. Some key components of JDBC API include
  - Interfaces like Driver, ResultSet, RowSet, PreparedStatement, and Connection that helps managing different database tasks.
  - Classes like DriverManager, Types, Blob, and Clob that helps managing database connections.
- **DriverManager:** It plays an important role in the JDBC architecture. It uses some database-specific drivers to effectively connect enterprise applications to databases.

- **JDBC drivers:** These drivers handle interactions between the application and the database.

The JDBC architecture consists of two-tier and three-tier processing models to access a database. They are as described below:

1. Two-Tier Architecture

A Java Application communicates directly with the database using a JDBC driver. It sends queries to the database and then the result is sent back to the application. For example, in a client/server setup, the user's system acts as a client that communicates with a remote database server.

Structure:

Client Application (Java) -> JDBC Driver -> Database

2. Three-Tier Architecture

In this, user queries are sent to a middle-tier services, which interacts with the database. The database results are processed by the middle tier and then sent back to the user.

Structure:

Client Application -> Application Server -> JDBC Driver -> Database

1.3 JDBC Components

There are generally **4 main components of JDBC** through which it can interact with a database. They are as mentioned below:

1. JDBC API

It provides various methods and interfaces for easy communication with the database. It includes two key packages

- **java.sql**: This package, is the part of **Java Standard Edition (Java SE) ,** which contains the core interfaces and classes for accessing and processing data in relational databases. It also provides essential functionalities like establishing connections, executing queries, and handling result sets
- **javax.sql**: This package is the part of **Java Enterprise Edition (Java EE),** which extends the capabilities of `java.sql` by offering additional features like connection pooling, statement pooling, and data source management.

It also provides a standard to connect a database to a client application.

2. JDBC Driver Manager

Driver manager is responsible for loading the correct database-specific driver to establish a connection with the database. It manages the available drivers and ensures the right one is used to process user requests and interact with the database.

3. JDBC Test Suite

It is used to test the operation(such as insertion, deletion, updating) being performed by JDBC Drivers.

4. JDBC Drivers

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:

- Type-1 driver or JDBC-ODBC bridge driver
- Type-2 driver or Native-API driver (partially java driver)
- Type-3 driver or Network Protocol driver (fully java driver)
- Type-4 driver or Thin driver (fully java driver) – It is a widely used driver. The older drivers like (JDBC-ODBC) bridge driver have been deprecated and no longer supported in modern versions of Java.

1.4 JDBC Classes and Interfaces

1. **DriverManager:** Manages JDBC drivers and establishes database connections.
2. **Connection:** Represents a session with a specific database.
3. **Statement:** Used to execute static SQL queries.
4. **PreparedStatement:** Precompiled SQL statement, used for dynamic queries with parameters.
5. **CallableStatement:** Used to execute stored procedures in the database.
6. **ResultSet:** Represents the result set of a query, allowing navigation through the rows.
7. **SQLException:** Handles SQL-related exceptions during database operations.

1.5 Key Features of JDBC

- **Platform Independence**: JDBC can perform database operation on any platform
- **Standard API**: It provides different ways to interact with different databases.
- **Support for Multiple Databases**: JDBC provide support to work with different databases like MySQL, PostgreSQL, Oracle, etc.

1.6 JDBC Drivers

JDBC drivers are client-side adapters (installed on the client machine rather than the server) that translate requests from Java programs into a protocol understood by the DBMS. These drivers are software components that implement the interfaces in the JDBC API, allowing Java applications to interact with a database. Sun Microsystems (now Oracle) defines four types of JDBC drivers, which are outlined below:

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

1. JDBC-ODBC Bridge Driver – Type 1 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.

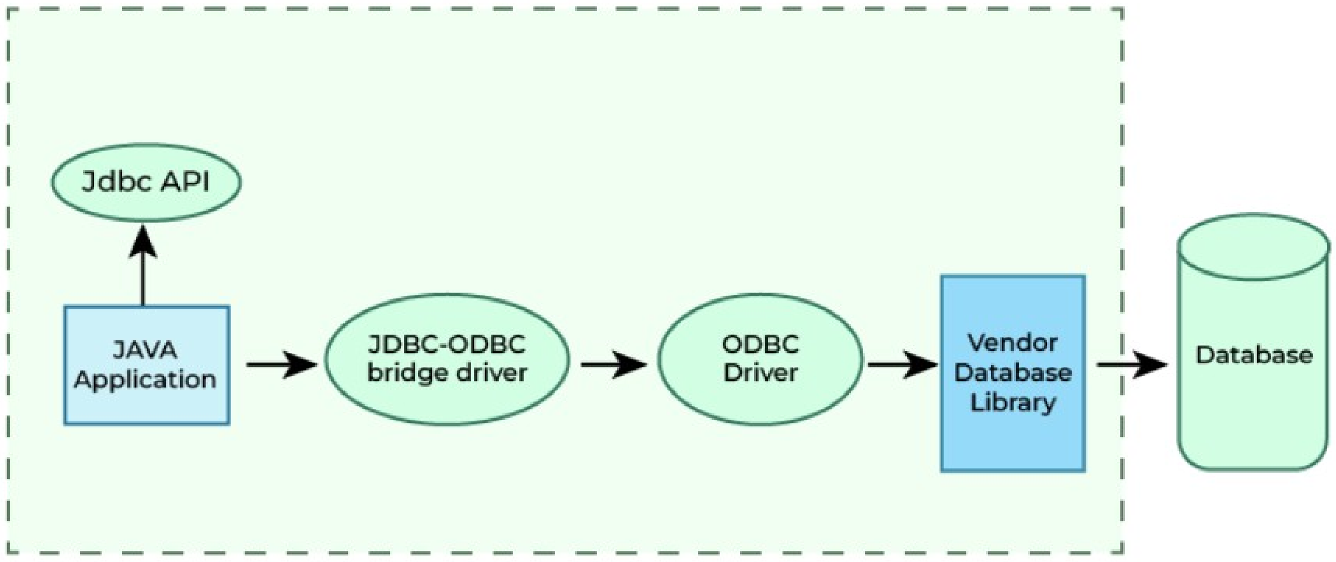


Fig. 1.2 Type 1 Driver

Advantages

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

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 – Type 2 Driver (Partially Java 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. This driver is not fully written in Java that is why it is also called Partially Java driver.

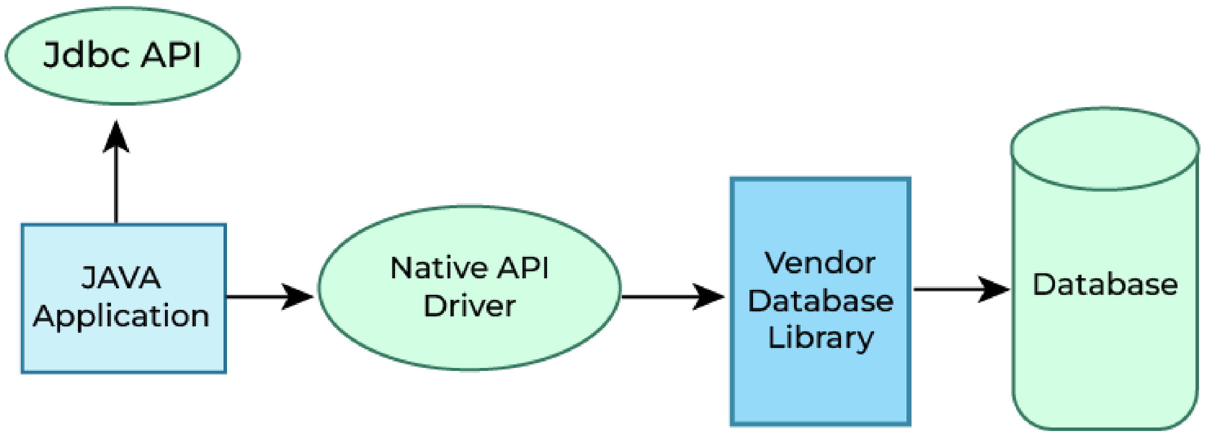


Fig. 1.3 Type 2 Driver

Advantage

- Native-API driver gives better performance than JDBC-ODBC bridge driver.
- More secure compared to the type-1 driver.

Disadvantages

- 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.

3. Network Protocol Driver – Type 3 Driver (Fully Java 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.

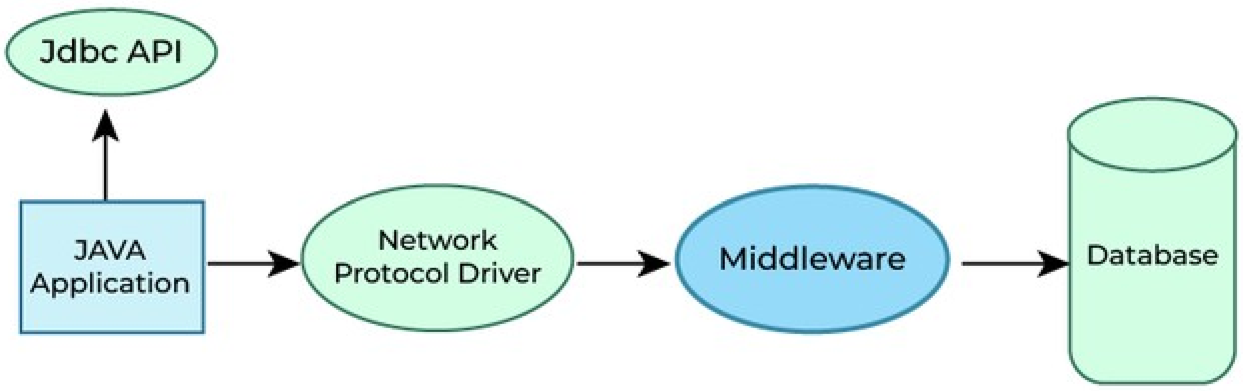


Fig. 1.4 Type 3 Driver

Advantages

- 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.
- 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 (Fully Java Driver)

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

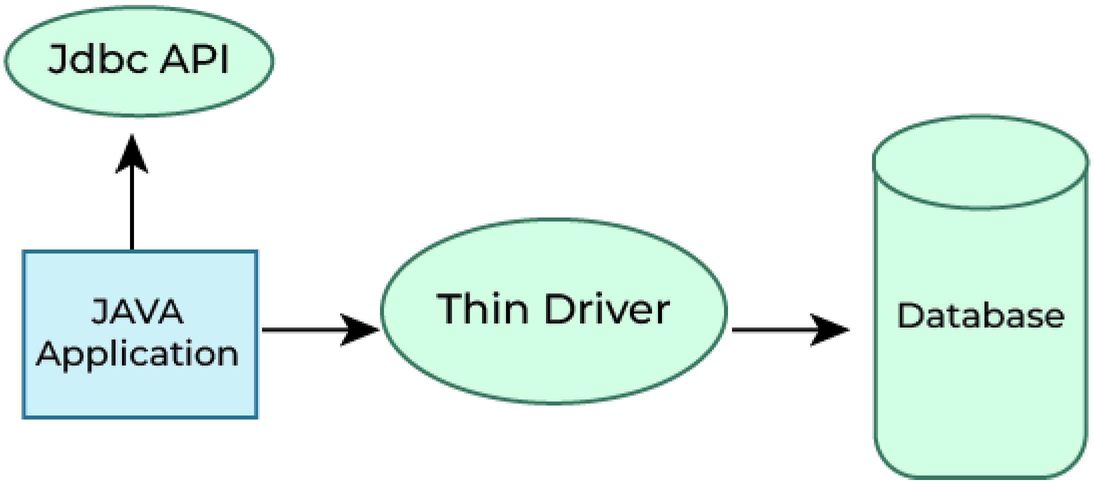


Fig. 1.5 Type 4 Driver

Advantages

- 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

- If the database changes, a new driver may be needed.

Which Driver to use 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.

8.

References:

1. **Book Reference**

   Jim Farley, William Crawford, David Flanagan. Java Enterprise in a Nutshell, O'Reilly

2. **Book Reference**

   Rocha, R., Purificação, J. (2018). Java EE 8 Design Patterns and Best Practices: Build Enterprise-ready Scalable Applications with Architectural Design Patterns. Germany: Packt Publishing..

3. **Website Reference**

   https://www.scribd.com/document/268349254/Java-8-Programming-Black-Book .

4. **Sources**

   https://developers.redhat.com/topics/enterprise-java

5. **Article**

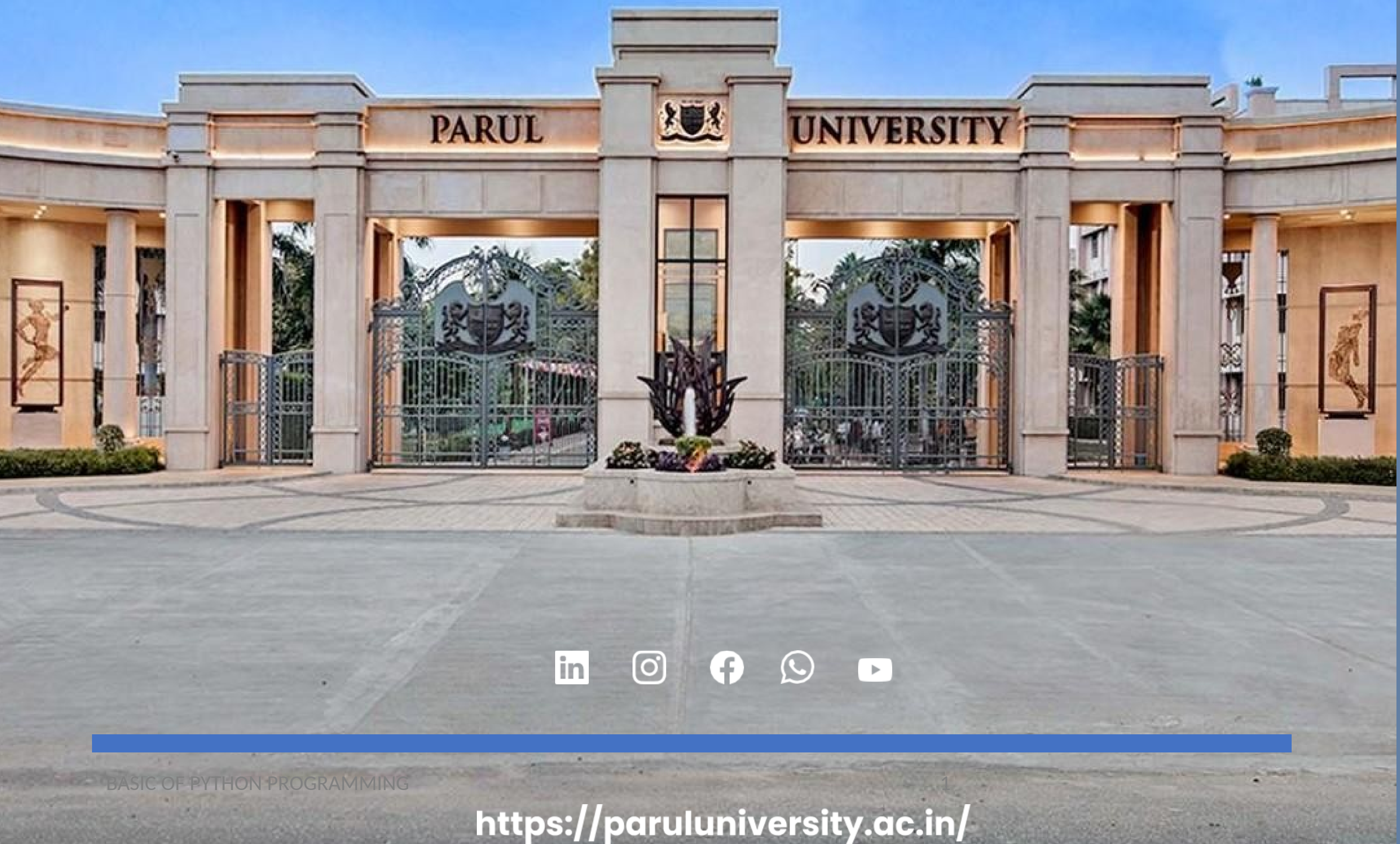
   https://www.researchgate.net/publication/276412369_Advanced_Java_Programming

Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and
Communication Technology

NAAC GRADE A++



https://paruluniversity.ac.in/