Module -V

Java Programming

TCS 408

Java Data Base Connectivity (JDBC)

- JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

- JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

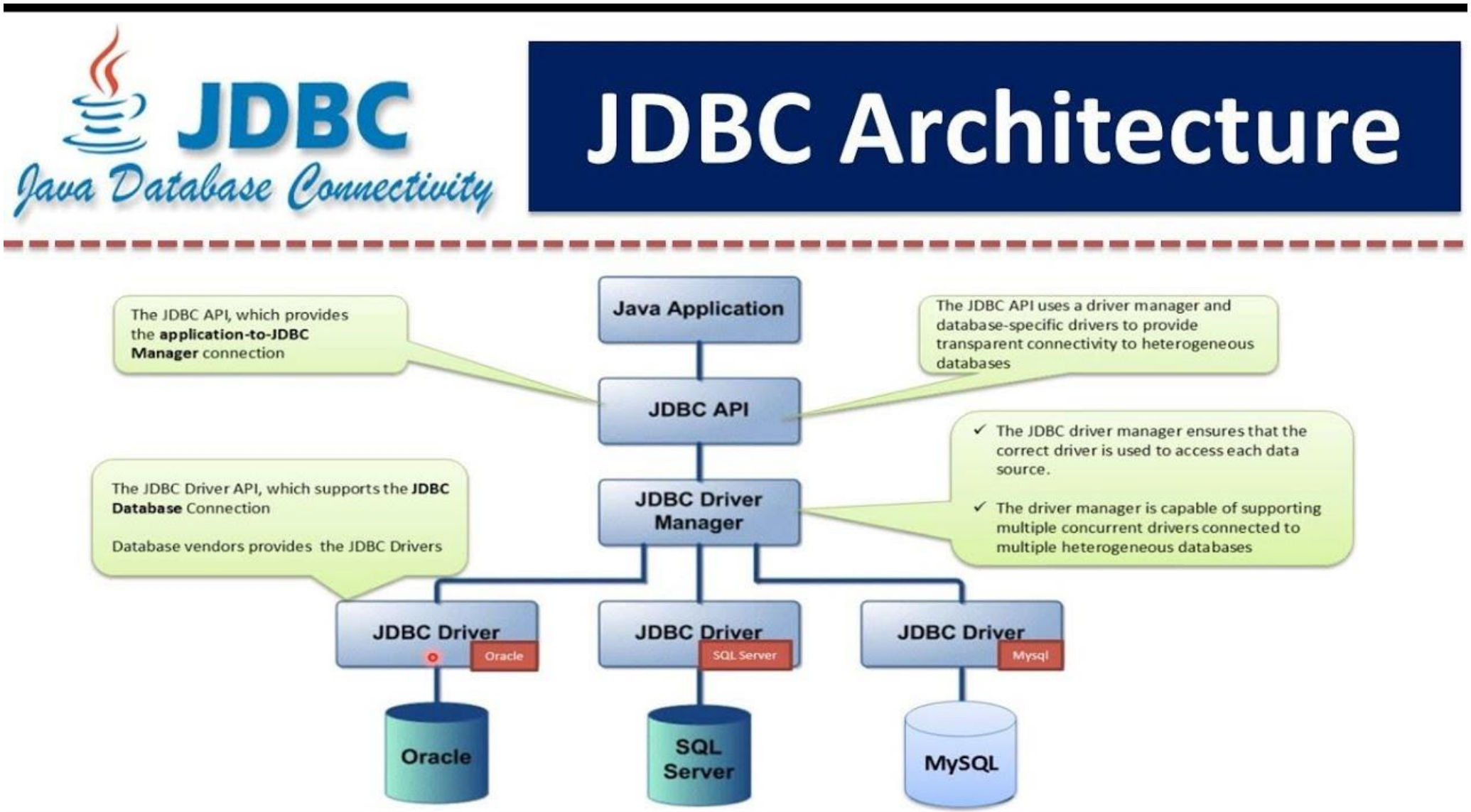
- The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

Components of JDBC

- JDBC API: It provides various methods and interfaces for easy communication with the database. It provides two packages which contain the java SE and Java EE platforms to exhibit WORA(write once run everywhere) capabilities.
- It also provides a standard to connect a database to a client application.
- JDBC Driver manager: It loads a database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.
- JDBC Test suite: It is used to test the operation(such as insertion, deletion, updation) being performed by JDBC Drivers.
- JDBC-ODBC Bridge Drivers: It connects database drivers to the database. This bridge translates the JDBC method call to the ODBC(Open Database Connectivity) function call. It makes use of the sun.jdbc.odbc package which includes a native library to access ODBC characteristics.

Architecture of JDBC



Types of JDBC Architecture(2-tier and 3-tier)

1.Two-tier model

A java application communicates directly to the data source. The JDBC driver enables the communication between the application and the data source.

When a user sends a query to the data source, the answers for those queries are sent back to the user in the form of results.

The data source can be located on a different machine on a network to which a user is connected. This is known as a **client/server configuration**, where the user's machine acts as a client, and the machine has the data source running acts as the server.

2. Three-tier model

In this, the user's queries are sent to middle-tier services, from which the commands are again sent to the data source.

The results are sent back to the middle tier, and from there to the user.

This type of model is found very useful by management information system directors.

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:

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

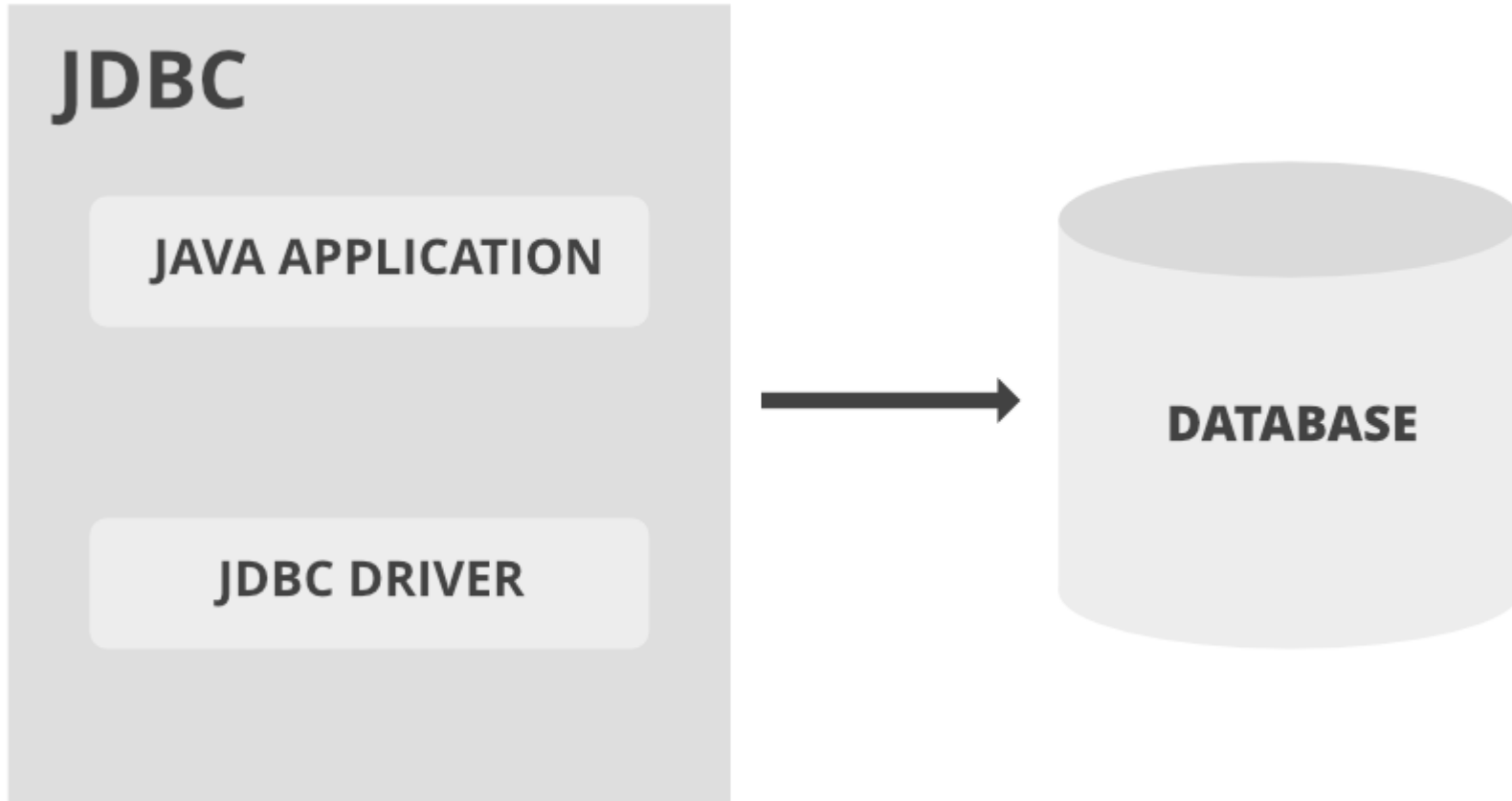
2.Type-2 driver or Native-API driver

3.Type-3 driver or Network Protocol driver

4.Type-4 driver or Thin driver

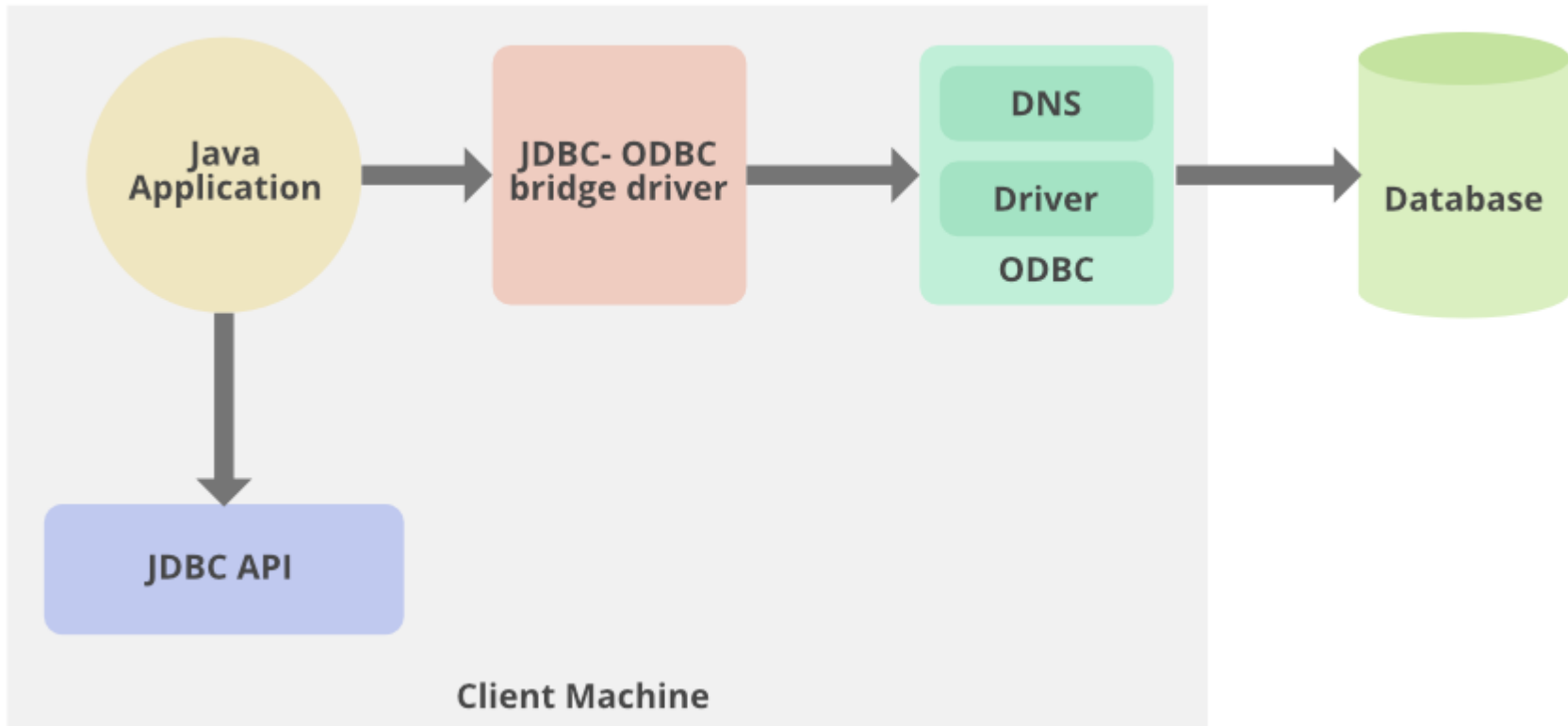
JDBC Type-1 Driver

- A JDBC driver enables Java application to interact with a database from where we can fetch or store data. The JDBC classes are contained in the Java Package java.sql and javax.sql.JDBC helps to
- Connect to a data source, like a database.
- Send queries and update statements to the database
- Retrieve and process the results received from the database in answer to your query


JDBC
JAVA APPLICATION
JDBC DRIVER
DATABASE

- Type -1 Driver JDBC driver also known as bridge driver it provides a bridge to access the ODBC driver installed on each client. Type 1 drivers translate calls to JDBC methods into calls to Open Database Connectivity (ODBC) functions. Bridge drivers allow JDBC applications immediate access to database connectivity provided by the existing array of ODBC drivers.

- ODBC is based on the device driver model, where the driver encapsulates the logic needed to convert a standard set of commands and functions into the specific calls required by the underlying system. Using the JDBC-ODBC bridge driver we can access the databases which support only ODBC. Java application sends a request to the JDBC-ODBC bridge driver the request internally calls the ODBC equivalent function and the ODBC driver retrieves the result from the underlying database and sends it back to the JDBC-ODBC bridge driver.

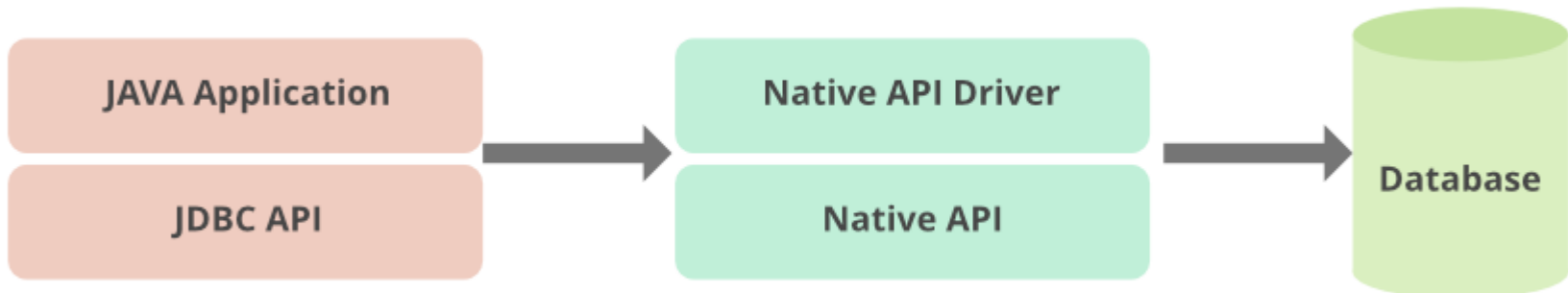

Java Application
JDBC- ODBC bridge driver
DNS
Driver
ODBC
Database
JDBC API
Client Machine

Type 2 or Partial Java driver

Also known as Native API converts JDBC calls into database-specific native libraries calls and these calls are directly understood by the database engine.

This type of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS.

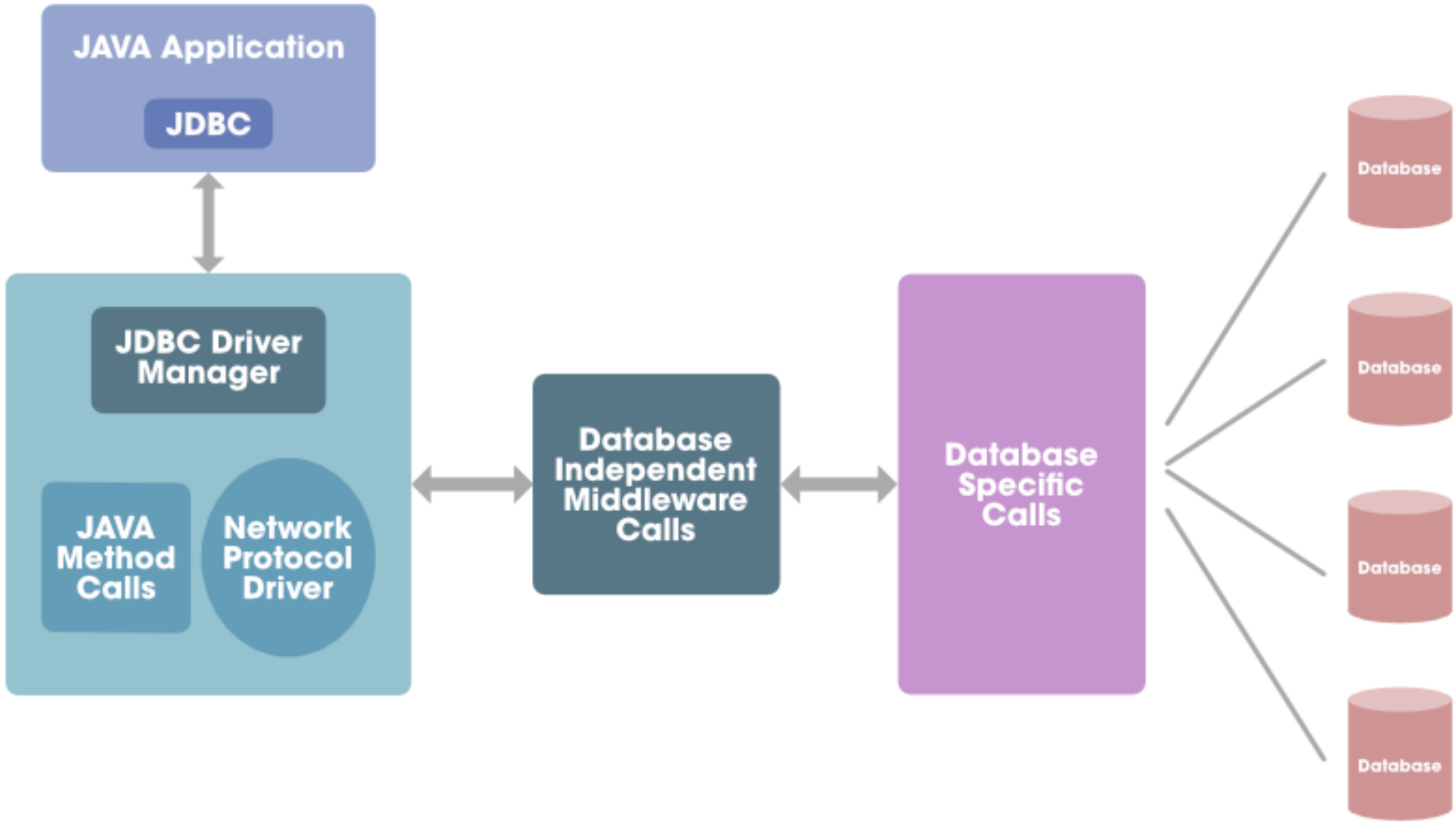
- JDBC driver directly talks to DB client using native API
- It required native API to connect to DB client it is also less portable and platform dependent
- Type 2 drivers convert JDBC calls into database-specific calls i.e. this driver is specific to a particular database.
- Like Type 1 drivers, it’s not written in Java Language which forms a portability issue.
- If we change the Database we have to change the native API as it is specific to a database.
- It is not threaded Safe
- No longer produced or used.
- Type-2 Drivers aren’t architecturally compatible


JAVA Application
JDBC API
Native API Driver
Native API
Database

Type -3 Driver

- **JDBC** is also known as Network Protocol Driver as it uses an application server that converts JDBC calls directly or indirectly into the vendor-specific database protocol. This driver translates JDBC calls into the middleware vendor's protocol, which is then converted to a database-specific protocol by the middleware server software that provides connectivity to many databases.

- Middleware is software that lies between an operating system and the applications running on it. Essentially functioning as a hidden translation layer, middleware enables communication and data management for distributed applications. While all middleware performs communication functions, the type a company chooses to use will depend on what service is being used and what type of information needs to be communicated.


JAVA Application
JDBC
JDBC Driver Manager
JAVA Method Calls
Network Protocol Driver
Database Independent Middleware Calls
Database Specific Calls
Database
Database
Database
Database

Type-4 Driver

- Type-4 driver is also called native protocol driver. This driver interacts directly with the database. It does not require any native database library, that is why it is also known as Thin Driver.
- 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.
- The thin driver converts JDBC calls directly into the vendor-specific database protocol. It is fully written in Java language. Also, it is a platform-independent driver but it is database dependent as it uses a native protocol(Protocol can establish a connection between particular server only).
- **Thin driver** installs inside Java Virtual Machine of the Client.

- No special software on the client or server needed. No translation or middleware layers are used, improving performance.
- Platform independent (Completely coded in Java)
- No need to install native libraries.
- It uses a database server-specific protocol which makes it secure.
- Helpful in debugging as JVM can manage all aspects of the application-to-database connection.
- It uses database-specific protocol and it is DBMS vendor dependent
- Scalable
- Advanced system administration
- Superior performance
- Advance Java feature set
- Offers significantly better performance than the JDBC/ODBC Bridge and Type 2 Drivers

Interfaces of JDBC API

- Driver interface
- Connection interface
- **Statement interface**
- PreparedStatement interface
- CallableStatement interface
- **ResultSet interface**
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

Statement Interface

- The statement interface is used to create SQL basic statements in Java it provides methods to execute queries with the database. There are different types of statements that are used in JDBC as follows:
- Create Statement
- Prepared Statement
- Callable Statement

- **Create a Statement:** From the connection interface, you can create the object for this interface. It is generally used for general–purpose access to databases and is useful while using static SQL statements at runtime.

```
Statement statement = connection.createStatement()
```

Once the Statement object is created, there are three ways to execute it.

- ***boolean execute(String SQL):*** If the ResultSet object is retrieved, then it returns true else false is returned. Is used to execute SQL DDL statements or for dynamic SQL.
- **int executeUpdate(String SQL):** Returns number of rows that are affected by the execution of the statement, used when you need a number for INSERT, DELETE or UPDATE statements.
- ***ResultSet executeQuery(String SQL):*** Returns a ResultSet object. Used similarly as SELECT is used in SQL.