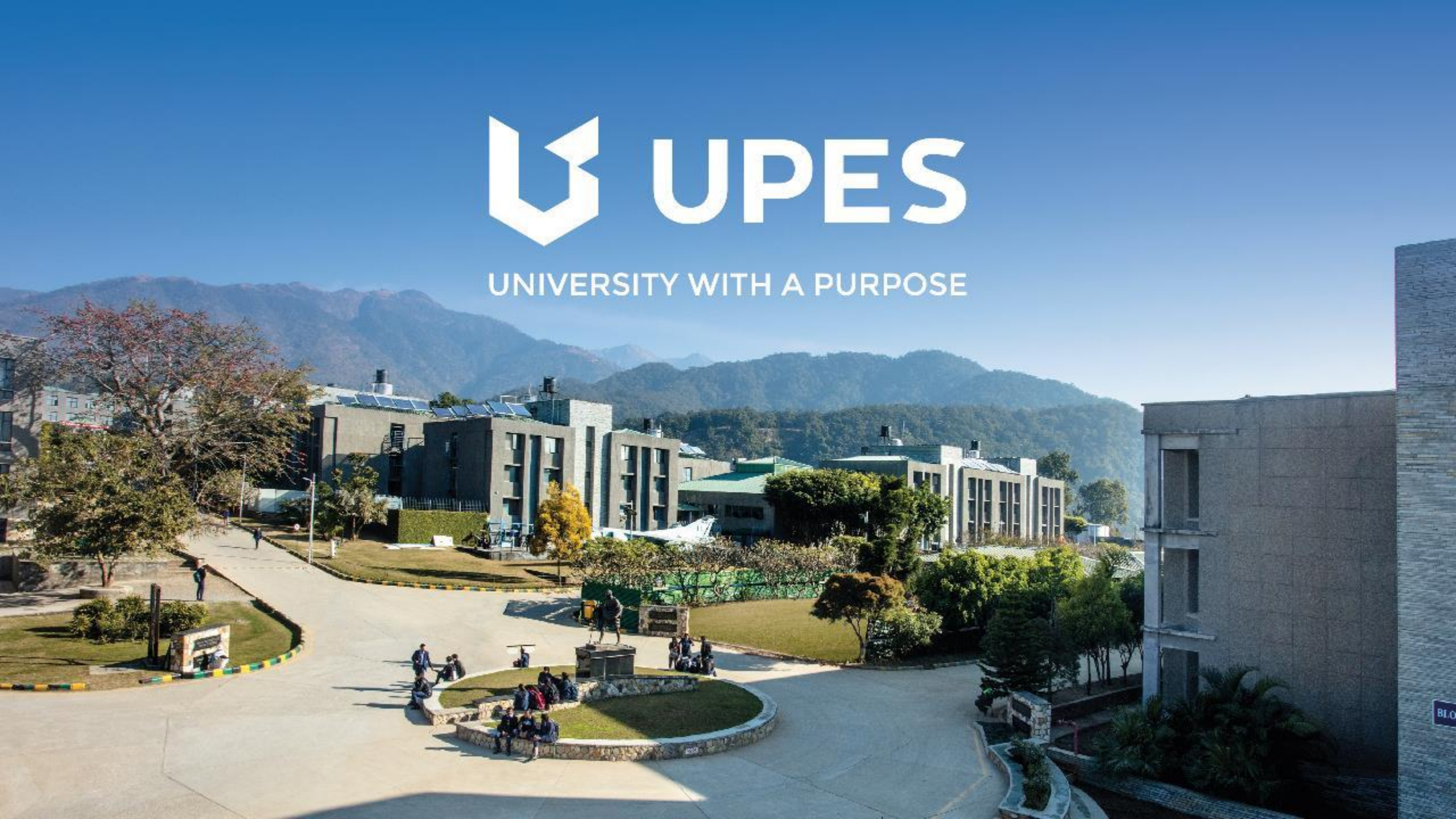

UPES
UNIVERSITY WITH A PURPOSE

Introduction to JDBC-II

Driver

- A driver is a program that converts the Java method calls to the corresponding method calls understandable by the database in use.
- The set of classes that implement the JDBC interfaces for a particular database engine is called a JDBC driver.
- A JDBC driver is a software component enabling a Java application to interact with a database.
- To connect with individual databases, JDBC API 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.

Types of Drivers

- JavaSoft's JDBC consists of two layers:
- the JDBC API
- JDBC Driver Manager API.

- The JDBC API is the top layer and is the programming interface in Java to structured query language (SQL) which is the standard for accessing relational databases.
- The JDBC API communicates with the JDBC Driver Manager API, sending it various SQL statements.
- The manager communicates (transparent to the programmer) with the various third party drivers (provided by Database vendors like Oracle) that actually connect to the database and return the information from the query.

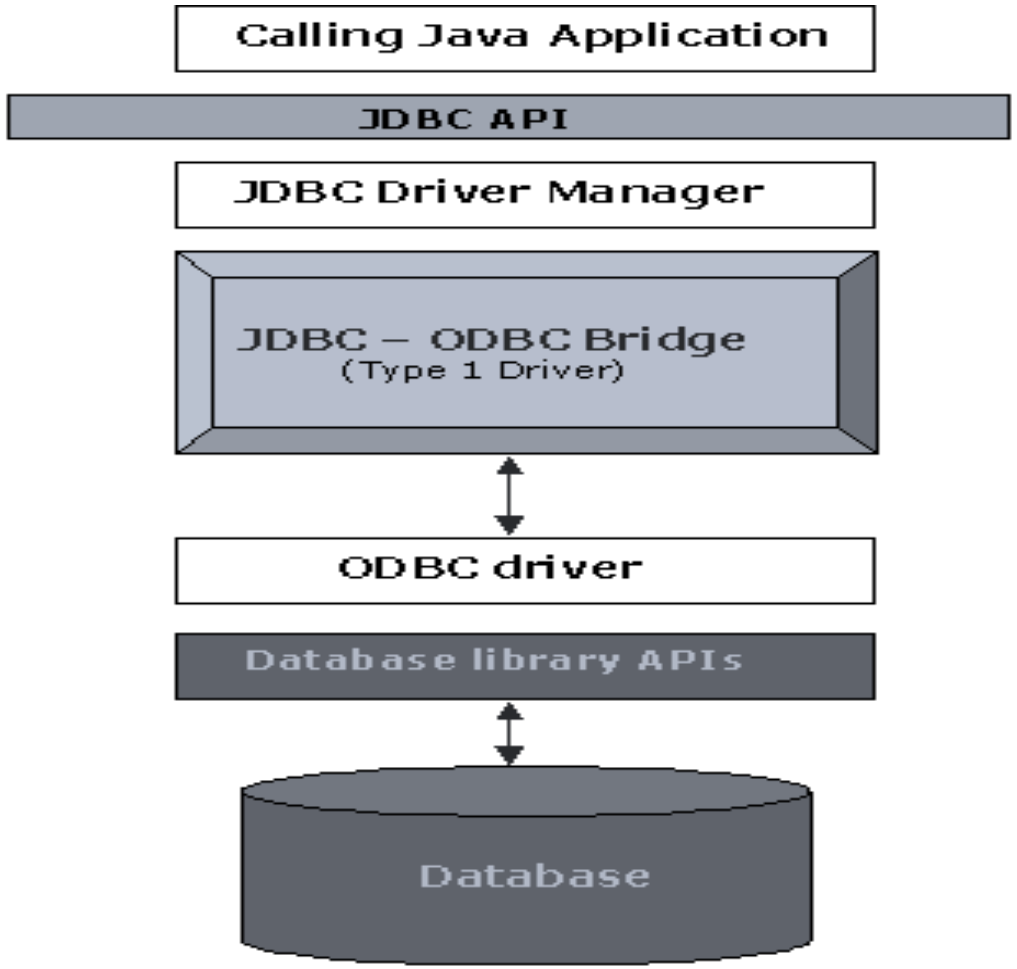
Types of Drivers

- JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

Types of Drivers

1. **JDBC-ODBC bridge driver**



Types of Drivers

JDBC-ODBC bridge driver

- The JDBC type 1 driver, also known as the JDBC-ODBC Bridge, is a database driver implementation that employs the ODBC driver to connect to the database.
- Translates all JDBC API calls to ODBC API calls and sends them to the ODBC driver.
- The JDBC-ODBC Bridge driver is recommended only for experimental use or when no other alternative is available.
- Relies on an ODBC driver to communicate with the database.
- Translates query obtained by JDBC into corresponding ODBC query, which is then handled by the ODBC driver.
- Sun provides a JDBC-ODBC Bridge driver.

  sun.jdbc.odbc.JdbcOdbcDriver

Types of Drivers

JDBC-ODBC bridge driver

Advantages:

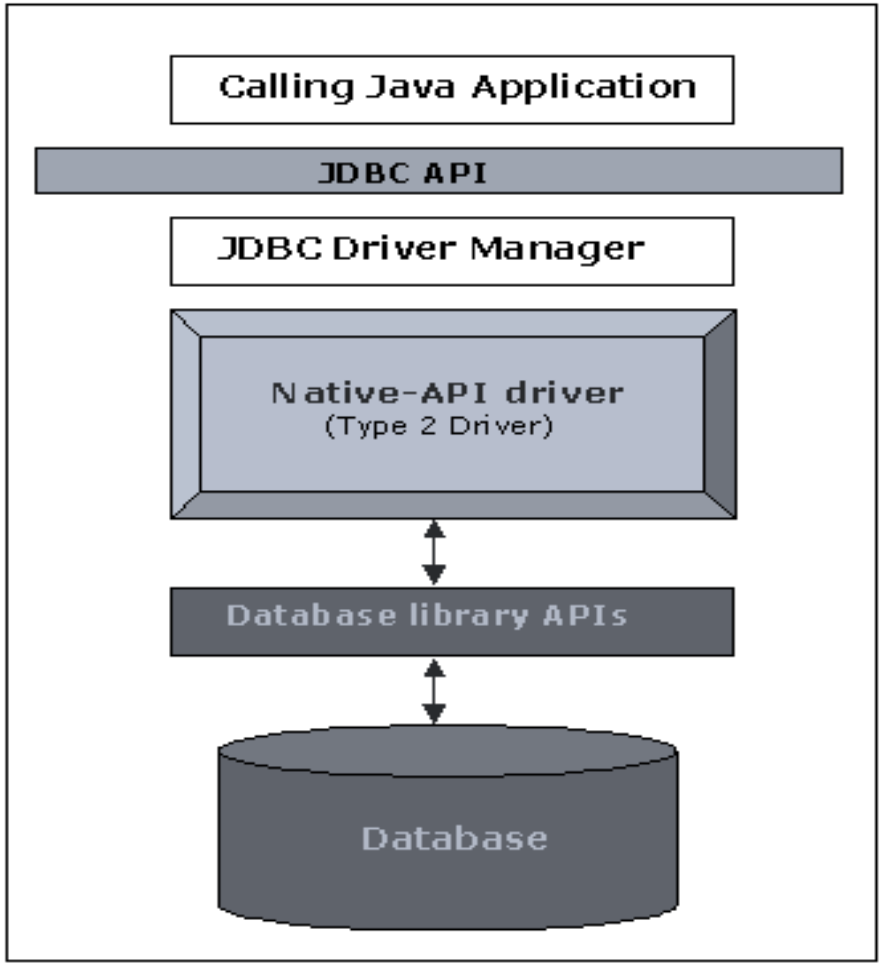
- Almost any database, for which ODBC driver is installed, can be accessed.
- A type 1 driver is easy to install.

Disadvantages:

- A performance issue is seen as a JDBC call goes through the bridge to the ODBC driver, then to the database, and this applies even in the reverse process. They are the slowest of all driver types.
- The ODBC driver needs to be installed on the client machine.
- Since the Bridge driver is not written fully in Java, Type 1 drivers are not portable.

Types of Drivers

1. **Native-API driver**



Types of Drivers

Native-API driver

- The JDBC type 2 driver, also known as the Native-API driver, is a database driver implementation that uses the client-side libraries of the database.
- The driver converts JDBC method calls into native calls of the database API.
- Written partly in Java & partly in native code, that communicates with the client API of a database.
- Therefore, should install some platform-specific code in addition to Java library.

Types of Drivers

Native-API driver

Advantages:

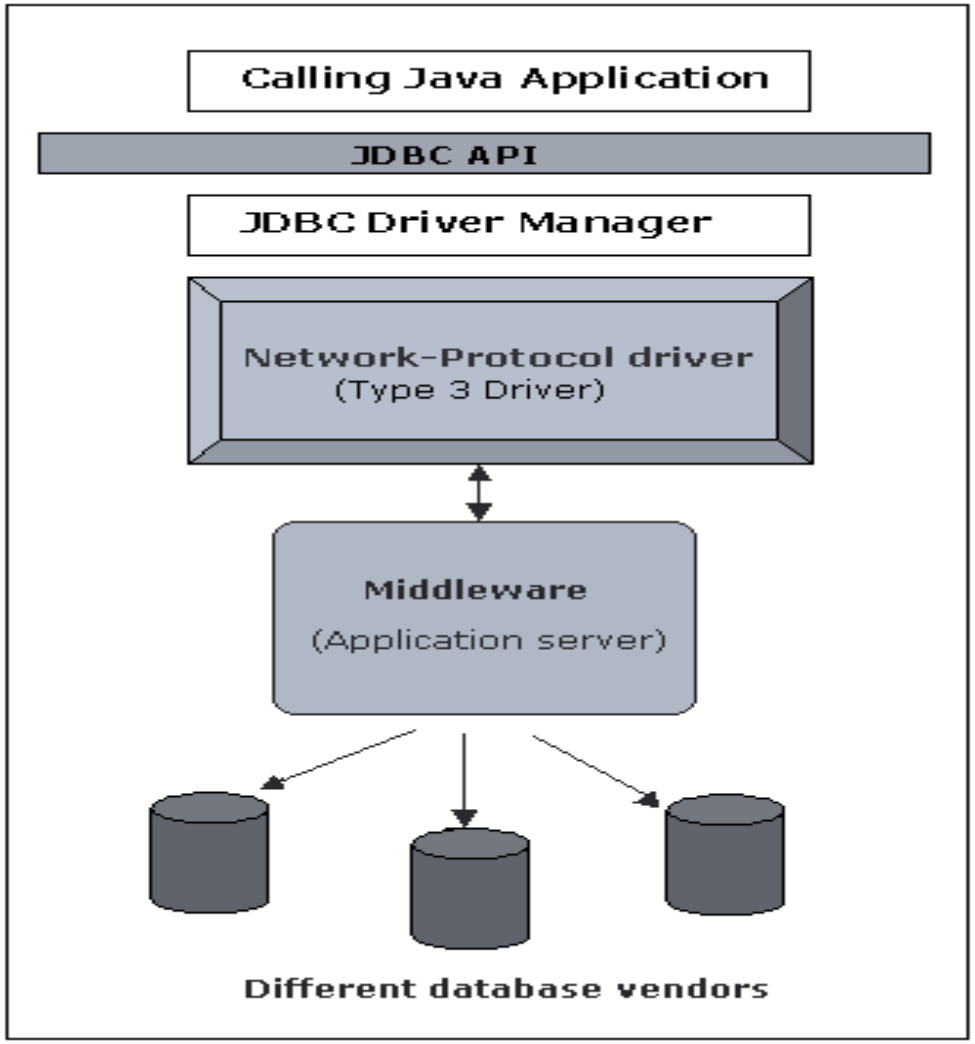
- Better performance than Type 1 Driver (JDBC-ODBC bridge)
- Better performance than the type 1 driver as it does not have the overhead of the additional ODBC function calls.

Disadvantages:

- The vendor client library needs to be installed on the client machine.
- Cannot be used in web-based application due to the client side software needed.
- Not all databases have a client side library
- If we change the Database we have to change the native API as it is specific to a database.

Types of Drivers

1. **Network Protocol driver**



Types of Drivers

Network Protocol driver

- The JDBC type 3 driver, also known as the Pure Java Driver for Database Middleware.
- It is a database driver implementation which makes use of a middle tier between the calling program and the database.
- The middle-tier (application server) converts JDBC calls directly or indirectly into the vendor-specific database protocol.
- The type 3 driver is platform-independent as the platform-related differences are taken care by the middleware.
- Also, making use of the middleware provides additional advantages of security and firewall access.

Types of Drivers

Network Protocol driver

Advantages:

- Since the communication between client and the middleware server is database independent, there is no need for the vendor db library on the client machine. Also the client to middleware need not be changed for a new database.
- This driver is fully written in Java and hence Portable.
- It is suitable for the web.
- There are many opportunities to optimize portability, performance, and scalability.
- The net protocol can be designed to make the client JDBC

Types of Drivers

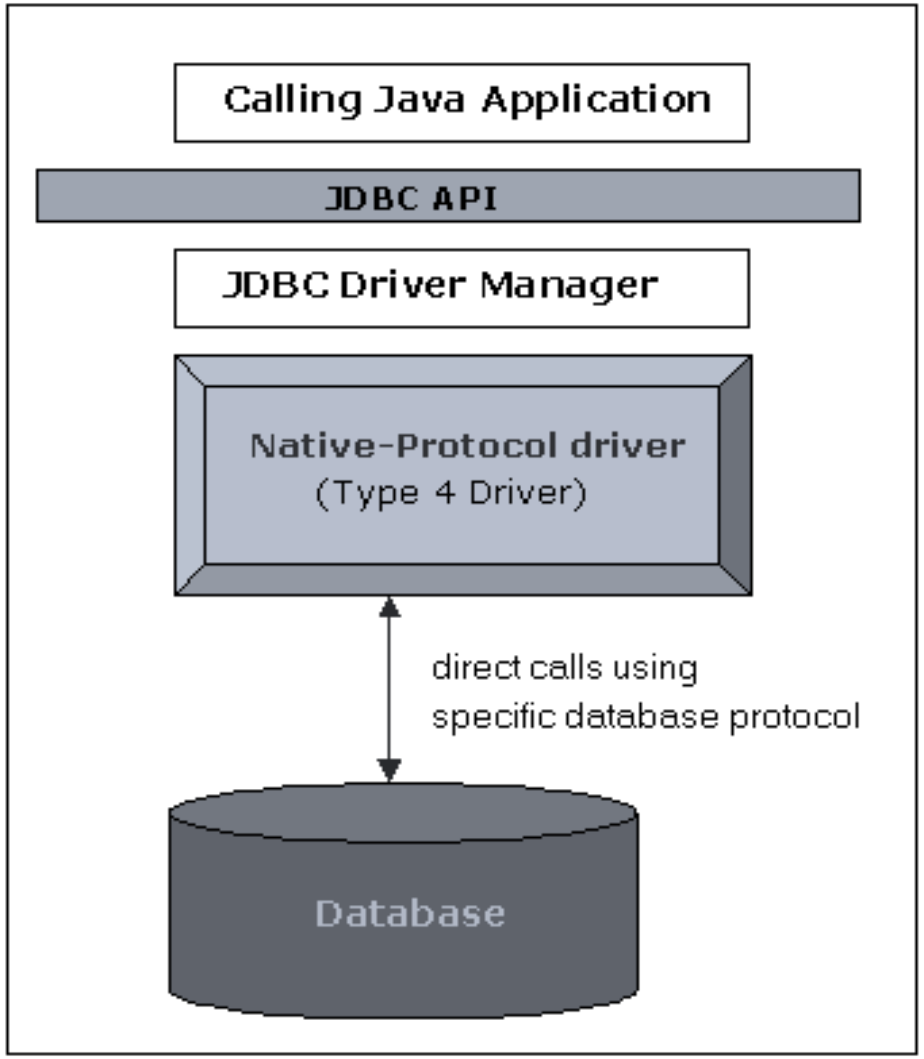
Network Protocol driver

Disadvantages:

- Requires database-specific coding to be done in the middle tier.
- It requires another server application to install and maintain.
- Traversing the record set may take longer, since the data comes through the backend server.

Types of Drivers

1. **Thin driver**



Types of Drivers

Thin driver

- The JDBC type 4 driver, also known as the Direct to Database Pure Java Driver, is a database driver implementation that converts JDBC calls directly into a vendor-specific database protocol.
- They install inside the Java Virtual Machine of the client. This provides better performance than the type 1 and type 2 drivers as it does not have the overhead of conversion of calls into ODBC or database API calls.
- Type 4 drivers, coded entirely in Java, communicate directly with a vendor's database, usually through socket connections.
- No translation or middleware layers are required, improving performance.

Types of Drivers

Thin driver

Advantages:

- The major benefit of using a type 4 jdbc drivers are that they are completely written in Java to achieve platform independence and eliminate deployment administration issues.
- It is most suitable for the web.
- You don’t need to install special software on the client or server. Further, these drivers can be downloaded dynamically.
- The JVM can manage all aspects of the application-to-database connection; this can facilitate debugging

Types of Drivers

Thin driver

Disadvantages:

- With type 4 drivers, the user needs a different driver for each database.
- Drivers are database dependent.

ODBC

- A driver manager for managing drivers for SQL based databases.
- Developed by Microsoft to allow generic access to disparate database systems on windows platform.
- J2SDK comes with JDBC-to-ODBC bridge database driver to allow a java program to access any ODBC data source.

ODBC vs JDBC

ODBC	JDBC
ODBC Stands for Open Database Connectivity.	JDBC Stands for java database connectivity.
Introduced by Microsoft in 1992.	Introduced by SUN Micro Systems in 1997.
We can use ODBC for any language like C,C++,Java etc.	We can use JDBC only for Java languages.
We can choose ODBC only windows platform.	We can Use JDBC in any platform.
Mostly ODBC Driver developed in native languages like C,C++.	JDBC Stands for java database connectivity.
For Java applications it is not recommended to use ODBC because performance will be down due to internal conversion and applications will become platform Dependent.	For Java application it is highly recommended to use JDBC because there we no performance & platform dependent problem.
ODBC is procedural.	JDBC is object oriented.

Interface CallableStatement

```
public interface CallableStatement
extends PreparedStatement
```

- The interface used to execute SQL stored procedures. The JDBC API provides a stored procedure SQL escape syntax that allows stored procedures to be called in a standard way for all RDBMSs.
- A CallableStatement can return one ResultSet object or multiple ResultSet objects. Multiple ResultSet objects are handled using operations inherited from Statement.

ResultSet interface

- The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.
- By default, ResultSet object can be moved forward only and it is not updatable.
- But we can make this object to move forward and backward direction by passing either TYPE_SCROLL_INSENSITIVE or TYPE_SCROLL_SENSITIVE in createStatement(int,int) method as well as we can make this object as updatable by:

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                     ResultSet.CONCUR_UPDATABLE);
```

Commonly used methods of ResultSet interface

1) public boolean next():	is used to move the cursor to the one row next from the current position.
2) public boolean previous():	is used to move the cursor to the one row previous from the current position.
3) public boolean first():	is used to move the cursor to the first row in result set object.
4) public boolean last():	is used to move the cursor to the last row in result set object.
5) public boolean absolute(int row):	is used to move the cursor to the specified row number in the ResultSet object.
6) public boolean relative(int row):	is used to move the cursor to the relative row number in the ResultSet object, it may be positive or negative.
7) public int getInt(int columnIndex):	is used to return the data of specified column index of the current row as int.
8) public int getInt(String columnName):	is used to return the data of specified column name of the current row as int.
9) public String getString(int columnIndex):	is used to return the data of specified column index of the current row as String.
10) public String getString(String columnName):	is used to return the data of specified column name of the current row as String.

ResultSet interface Example

```
import java.sql.*;
class FetchRecord{
public static void main(String args[])throws Exception{

Class.forName("oracle.jdbc.driver.OracleDriver");
Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system
","oracle");
Statement
stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPD
ATABLE);
ResultSet rs=stmt.executeQuery("select * from emp765");

//getting the record of 3rd row
rs.absolute(3);
System.out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

con.close();
}}
```

UPES
UNIVERSITY WITH A PURPOSE

References

https://docs.oracle.com/javase/10/docs/api/java/sql/ResultSet.html

THANK YOU