JDBC

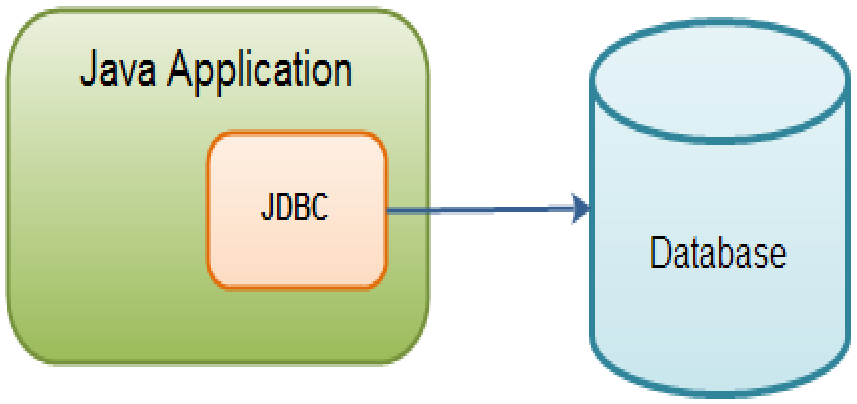
By

Anand Kulkarni

anand.pune38@gmail.com

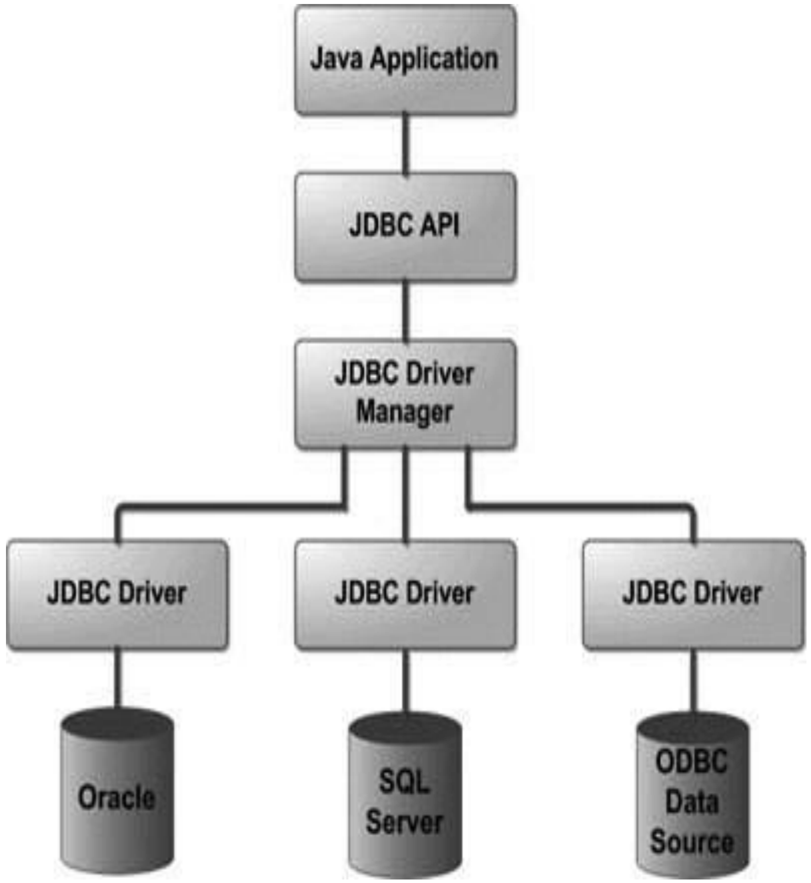
Contents

Introduction



- Java database connectivity (JDBC) is an API that enables java application to connect to relational database like oracle, SQL server, MySQL etc.

JDBC Driver



- JDBC driver is a program that enables java application to communicate with database.
- Thus, every database will have its own JDBC driver.

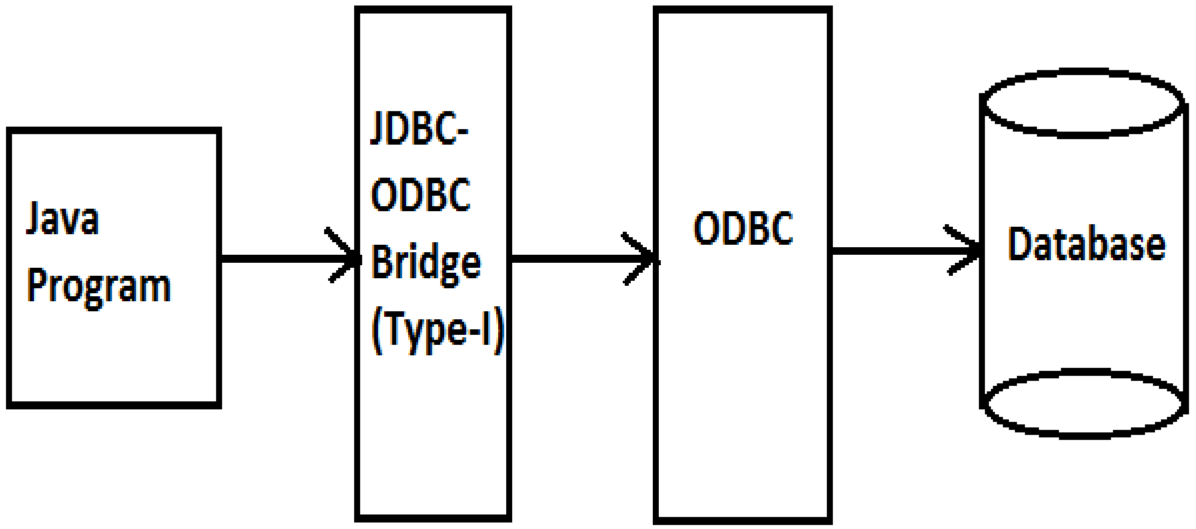
Types of drivers

There are different ways to communicate with database using JDBC drivers. These ways are known as type drivers.

JDBC offers 4 type drivers:

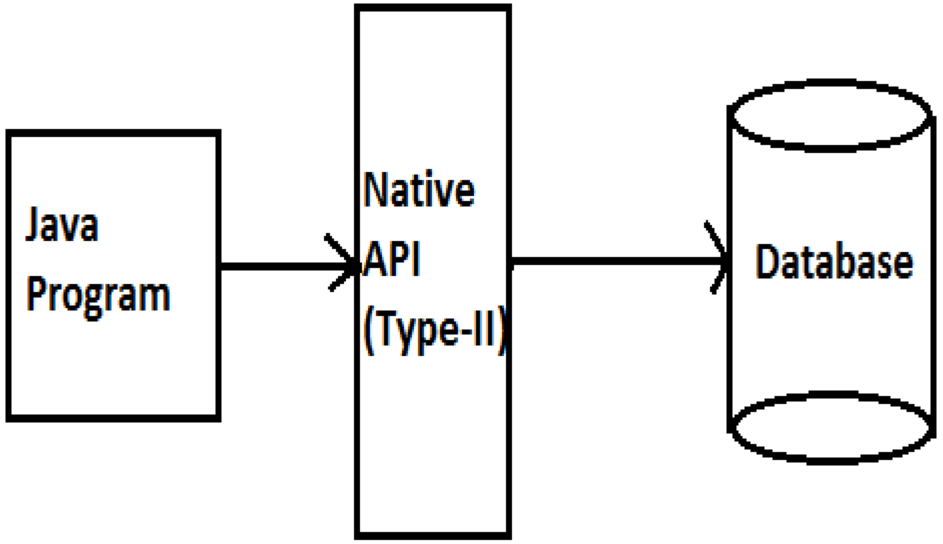
- Type-I (JDBC-ODBC bridge)
- Type-II (Native-API)
- Type-III (Network protocol driver)
- Type-IV (Pure java driver)

Type-I driver (JDBC-ODBC bridge)



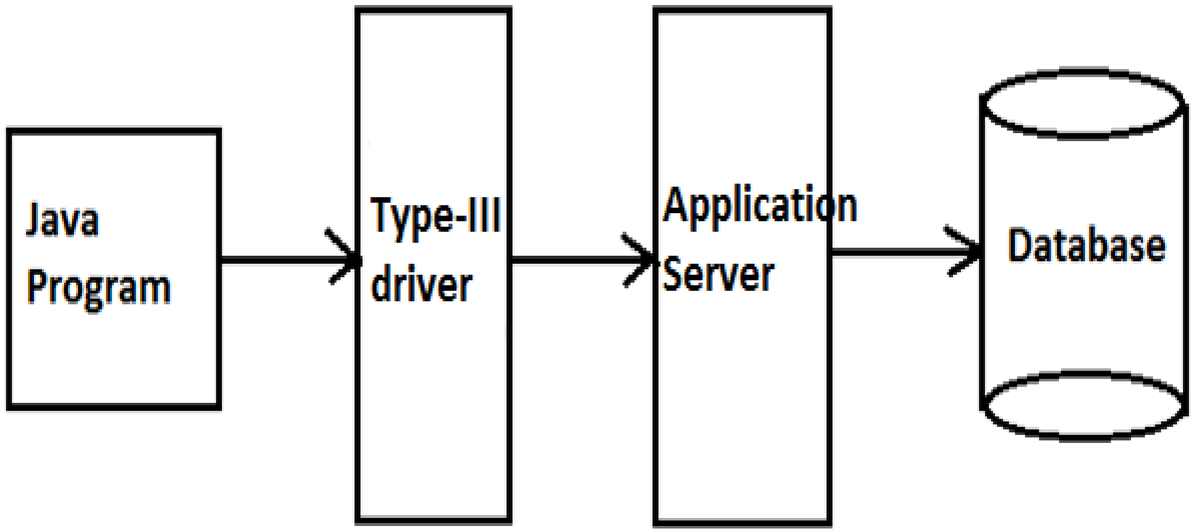
- Since type-I driver connects to ODBC, it is database independent.
- ODBC is available only for windows & hence type-I is a platform dependent driver. Hence, in professional environment, type-I driver is never used.
- Type-I driver class comes along with JDK installation itself.
- No support for type-I driver from JDK 1.8 onwards.

Type-II driver (Native driver)



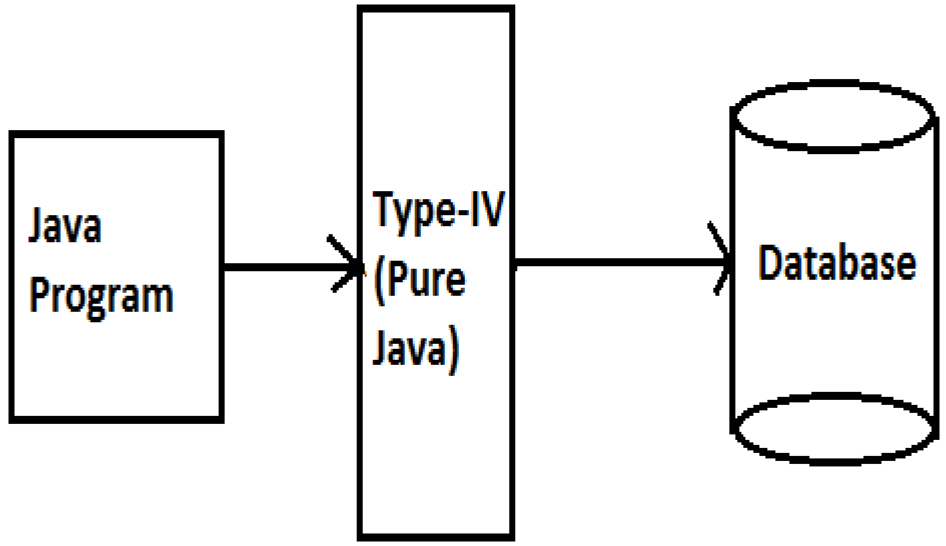
- The ODBC layer is completely removed & hence it is little faster than type-I.
- The driver code is in java & native language i.e. C or C++.
- Due to native code, type-II driver implementation is platform independent.
- Oracle type-II driver is also called as OCI driver.

Type-III driver (Network protocol driver)



- Type-III is a pure java driver.
- Type-III communicates with application server instead of database. And application server connects to actual database.
- Type-III is a database independent driver.
- Type-II driver is provided by application server itself.

Type-IV driver (Pure Java)



Type-IV is a pure java driver & hence it is a platform independent driver.

Type-IV directly communicates with database & hence it is database specific driver.

Type-IV driver in oracle is called as *'thin'* driver.

Database communication with JDBC

```
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"tiger");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM DEPT");
while(rs.next()) {
        System.out.println(rs.getInt("ID") + " - " + rs.getString("NAME"));
}
rs.close();
stmt.close();
con.close();
```

CRUD operations

Create new record:

int updated_records = statement.*executeUpdate*(“INSERT INTO DEPT VALUES (2, ‘Sales’)”);

Read table records:

ResultSet rs = statement.*executeQuery*("SELECT * FROM DEPT");

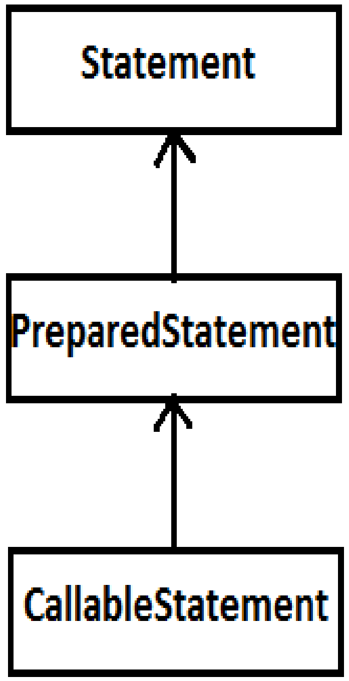
Update records:

int updated_records = statement.*executeUpdate*(“UPDATE DEPT SET name = ‘Sales’ WHERE ID = 2”);

Delete records:

int updated_records = statement.*executeUpdate*(“DELETE FROM DEPT WHERE ID = 2”);

Types of statements



Statement allows us to fire SQL query on database. However, there are 3 types of statements provided by JDBC API:

- Statement
- PreparedStatement &
- CallableStatement

PreparedStatement

- PreparedStatement is a pre-compiled SQL statement.
- If you wish to fire same query repeatedly then it is advisable to use PreparedStatement. It is because PreparedStatement compiles the query only once & hence it is faster in execution than ordinary statement.

```
PreparedStatement pstmt = dbcon.prepareStatement(“INSERT INTO EMP
VALUES (?,?,?)”);
pstmt.setInt(1, 222); //emp id
pstmt.setString(2, “Tom”); //emp name
pstmt.setDouble(3, 20000.70); //emp salary
int updates = pstmt.executeUpdate();
pstmt.close();
dbcon.close();
```

CallableStatement

- CallableStatement is used to call stored procedure on database.

```
CallableStatement stmt=con.prepareCall("{call insertRecord(?,?)}");

stmt.setInt(1,1011);

stmt.setString(2,"Amit");

stmt.execute();

Stmt.close();
```