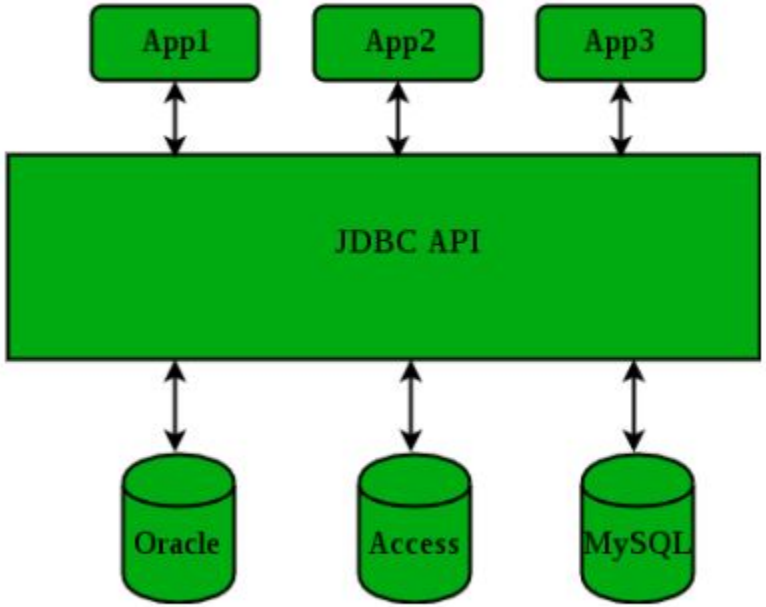
Unit 4: Database Connectivity

Prepared by: Aakash Raj Shakya

What is JDBC?

1. **J**ava **D**ata**b**ase **C**onnectivity (JDBC) is an application programming interface (API).
2. JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.
3. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.
4. It acts as a middle layer interface between java applications and database.

Structure of JDBC



JDBC Drivers

Why to Learn JDBC?

1. The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.
    a. Making a connection to a database.
    b. Creating SQL statements.
    c. Executing SQL queries in the database.
    d. Viewing & Modifying the resulting records.

JDBC Driver Types

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.

The JDBC classes are contained in the Java Package java.sql and javax.sql.

JDBC helps you to write Java applications that manage these three programming activities:

1. Connect to a data source, like a database.
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query

JDBC Drivers Type contd.

There are 4 types of JDBC drivers:

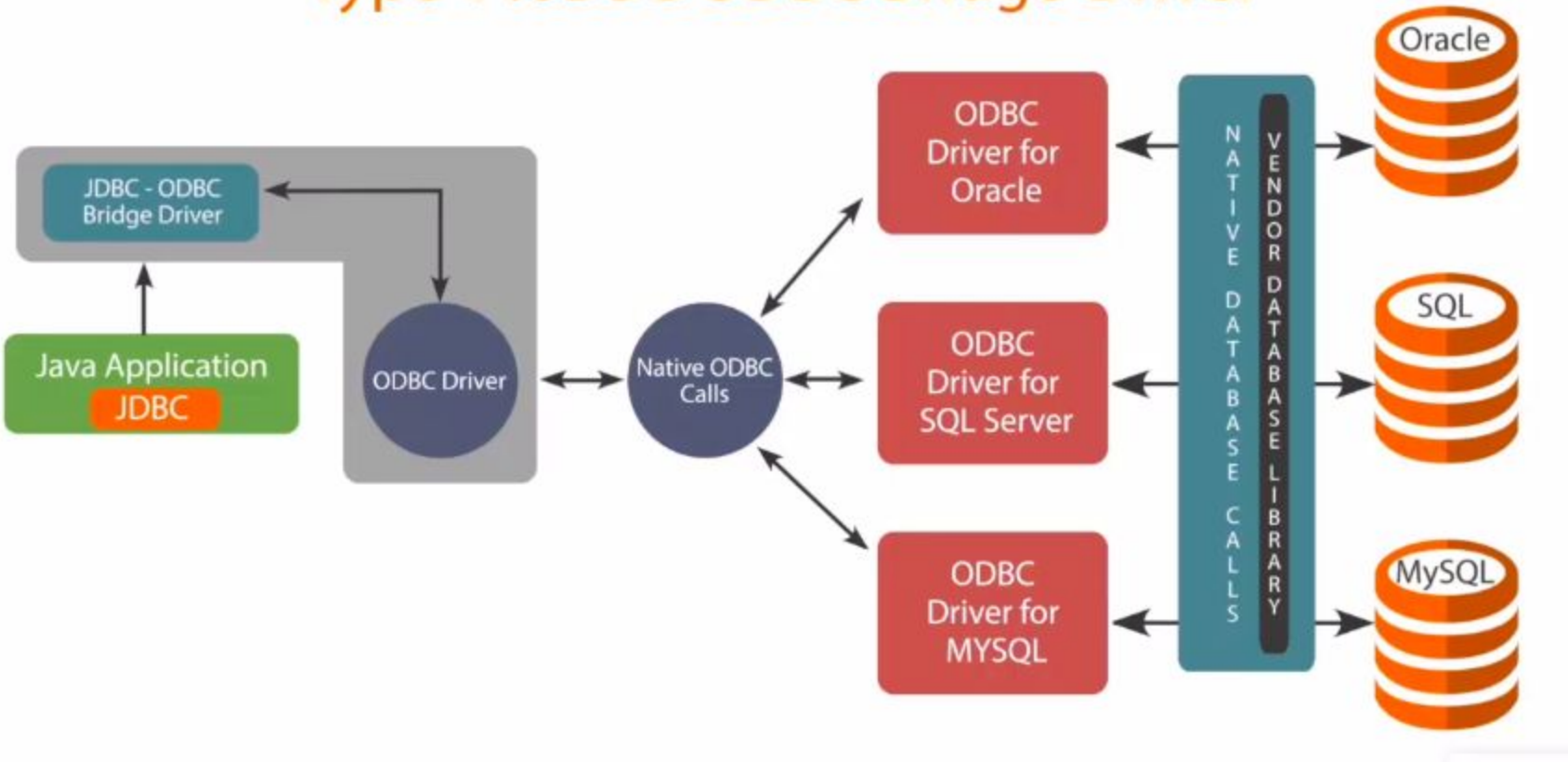
1. Type-1 driver or JDBC-ODBC bridge driver
2. Type-2 driver or Native-API driver
3. Type-3 driver or Network Protocol driver (Middleware Drive)
4. Type-4 Database-Protocol driver (Pure Java Driver)

In depth description about these types are in the following link:

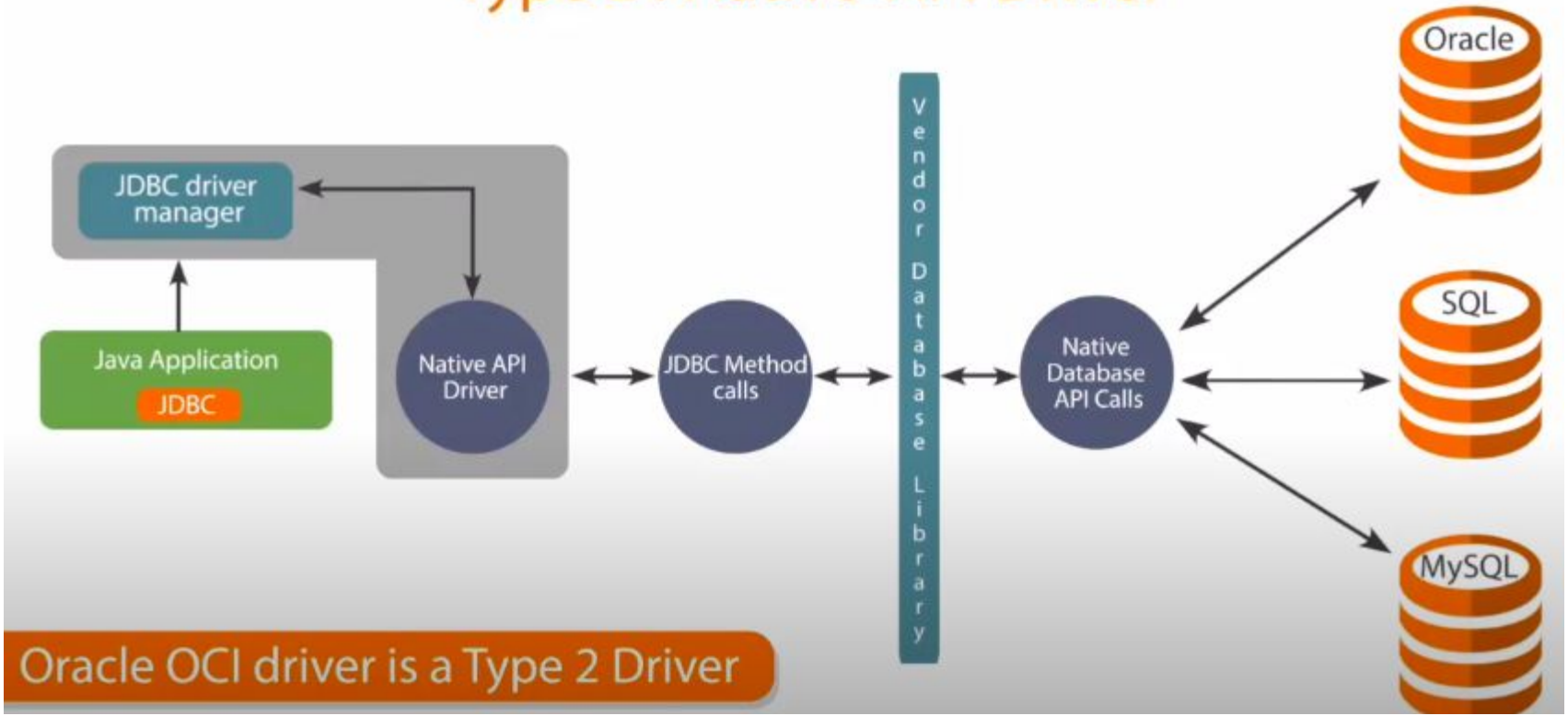
https://www.geeksforgeeks.org/jdbc-drivers/#:~:text=JDBC%20drivers%20are%20client%2Dside,driver%20or%20Native%2DAPI%20driver

https://www.tutorialspoint.com/jdbc/jdbc-driver-types.htm

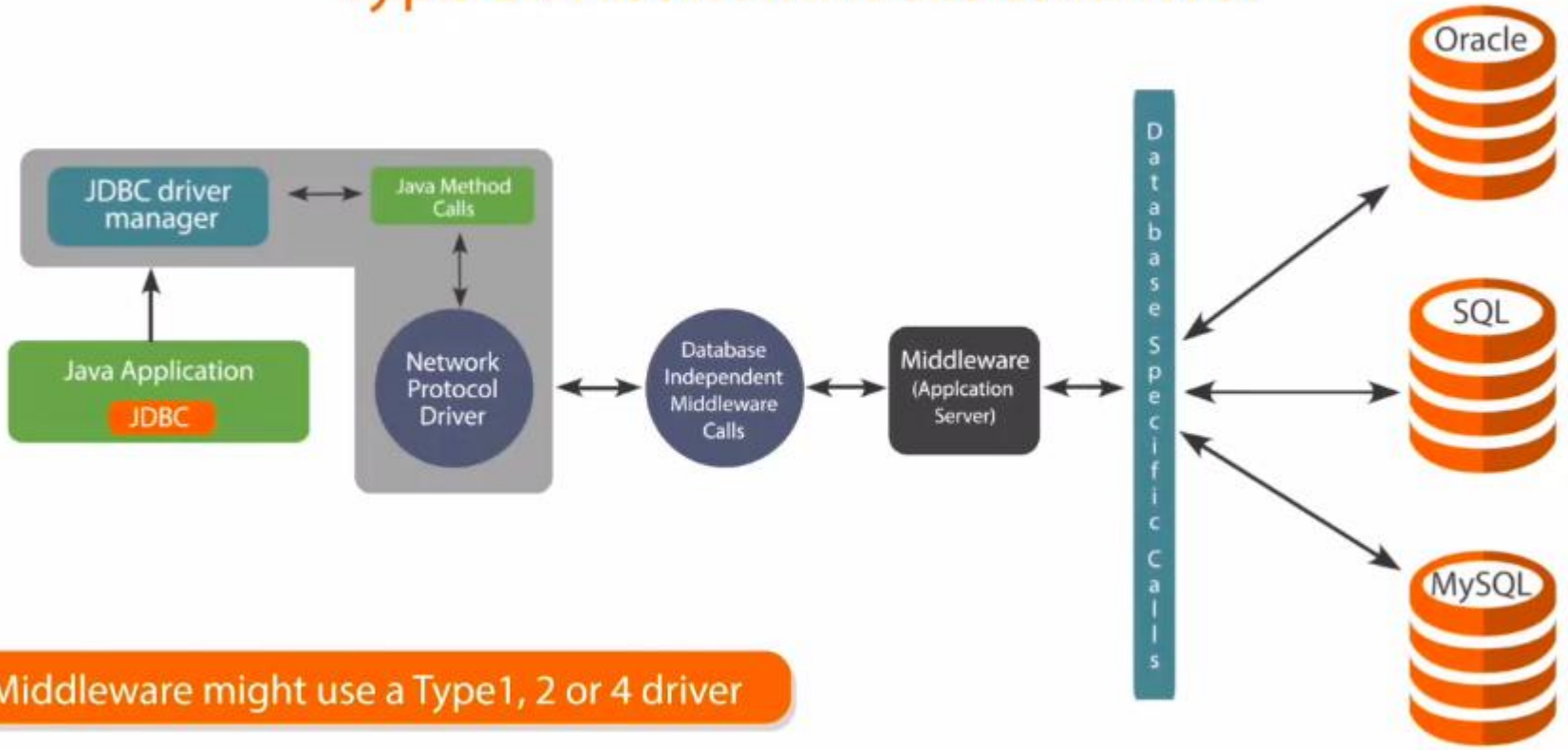
Type 1 : JDBC ODBC Bridge Driver



Type 2 : Native-API Driver

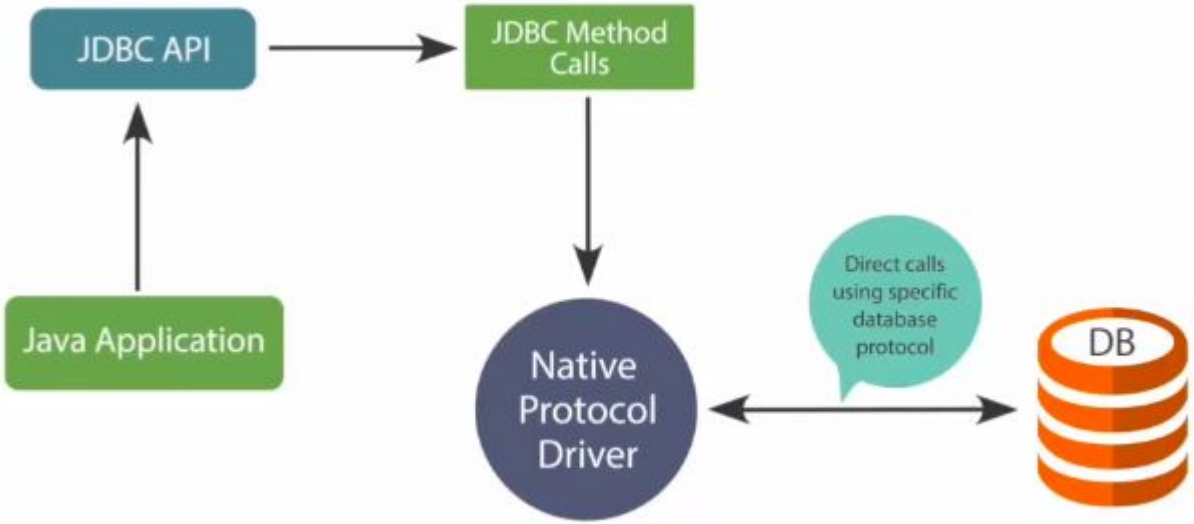


Type 3 : Network Protocol Driver



Middleware might use a Type1, 2 or 4 driver

Type 4 : Database Protocol Driver



Typical Uses of JDBC

1. We can use JDBC API to access tabular data stored in any relational database.
2. By the help of JDBC API, we can save, update, delete and fetch data from the database or basically CRUD.

JDBC Configuration

For configuring JDBC use following steps:

1. Download database (MySQL)
2. Install MySQL.
3. Install Database Drivers
4. Create Database
5. Create Table
6. Create Data Records

Creating JDBC Application

There are following six steps involved in building a JDBC application:

1. **Import the packages:** Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using import java.sql.* will suffice.
2. **Register the JDBC driver:** Requires that you initialize a driver so you can open a communication channel with the database.
3. **Open a connection:** Requires using the DriverManager.getConnection() method to create a Connection object, which represents a physical connection with the database.
4. **Execute a query:** Requires using an object of type Statement for building and submitting an SQL statement to the database.
5. **Extract data from result set:** Requires that you use the appropriate ResultSet.getXXX() method to retrieve the data from the result set.
6. **Clean up the environment:** Requires explicitly closing all database resources versus relying on the JVM's garbage collection.

Let’s Jump to the Code

Suppose we have a database named as **school_information_system** with table **employee** with following data.

id	full_name	address	age	contact_number
2	asdd	asdd	21	2334
3	Aakash Shakya	Lagant Tole	28	9860706065
4	Manish Thapa	Pepsicola	29	9845521214
5	Aakash	Lagan Tole	28	9860706065
6	Aakash	Lagan Tole	28	9860706065

Program to retrieve the information from the employee table

```
import java.sql.*;

public class MainClass {
  public static void main(String[] args) throws SQLException {
      Connection conn = null;
      Statement statement;
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/school_information_system",
               "root", "password");
         statement = conn.createStatement();
         String sql = "Select * from employee";
         ResultSet resultSet = statement.executeQuery(sql);
         while (resultSet.next()) {
            System.out.println("The name of the user is " + resultSet.getString("full_name"));
            System.out.println("The age of the user is " + resultSet.getInt("age"));
            System.out.println("The address of the user is " + resultSet.getString("address"));
            System.out.println("The contact number of the user is " + resultSet.getString("contact_number"));
            System.out.println("================================================");
         }
      } catch (ClassNotFoundException e) {
         System.out.println("The given Driver class could not be loaded.");
      } catch (SQLException se) {
         System.out.println("Exception occurred when carrying out SQL operation. " + se.getMessage());
      } finally {
         if(conn != null) {
            conn.close();
         }
      }
  }
}
```

The output for the given program will be

```
The name of the user is asdd
The age of the user is 21
The address of the user is asdd
The contact number of the user is 2334
================================================
The name of the user is Aakash Shakya
The age of the user is 28
The address of the user is Lagant Tole
The contact number of the user is 9860706065
================================================
The name of the user is Manish Thapa
The age of the user is 29
The address of the user is Pepsicola
The contact number of the user is 9845521214
================================================
The name of the user is Aakash
The age of the user is 28
The address of the user is Lagan Tole
The contact number of the user is 9860706065
================================================
The name of the user is Aakash
The age of the user is 28
The address of the user is Lagan Tole
The contact number of the user is 9860706065
================================================
```

To insert a new row in employee table.

```
import java.sql.*;

public class MainClass {
  public static void main(String[] args) throws SQLException {
     Connection conn = null;
     Statement statement = null;;
     try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/school_information_system",
              "root", "password");
        statement = conn.createStatement();
        String sql = "INSERT into employee (full_name, address, age, contact_number) " +
              "VALUES ('Manisha Koirala', 'Baluwatar', '50', '9851112252')";
        int isInserted = statement.executeUpdate(sql);
        if (isInserted == 1) {
           System.out.println("Record added successfully.");
        }
     } catch (ClassNotFoundException e) {
        System.out.println("The given Driver class could not be loaded.");
     } catch (SQLException se) {
        System.out.println("Exception occurred when carrying out SQL operation. " + se.getMessage());
     } finally {
        if (conn != null) {
           conn.close();
        }
     }
  }
}
```

Classes and methods used for executing SQL statements

1. **Class.forName():** It is used to load the class of JDBC Driver
2. **Connection:** It is used to create the connection
3. **DriverManager.getConnection():** It is used to get the connection of the given database with its respective db URL, username and password. (It has multiple implication)
4. **Statement:** It is used to execute the queries that we want. Example: statement.executeQuery(“Select * from students”);
5. **ResultSet:** It is used to collect the data returned from the statement.executeQuery() method.

Reading and Writing LOBs

1. The JDBC API provides the necessary support to work with Large Objects (LOB), such as storing an image file or a large text document in the database.
2. The requirements of data persistence may vary both by its size and type of data content.
3. A LOB can be of these types:
    a. BLOB (Binary Large Object)
    b. CLOB (Character Large Object) and
    c. NCLOB (National Character Large Object).
4. These are the LOB variations supported by the JDBC API. However, there is a difference on how a RDBMS treats and handles LOB objects internally.
5. Below is an example of blob:
   ALTER TABLE `employee` ADD COLUMN `image` **LONGBLOB** NULL AFTER `age`;

```java
import java.io.*;
import java.sql.*;

public class BlobUploadImageToDbExample {
  public static void main(String[] args) throws SQLException {
    Connection conn = null;
    PreparedStatement statement = null;
    try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/school_information_system",
          "root", "password");
      String sql = "INSERT into employee (full_name, address, age, contact_number, image) VALUES (?, ?, ?, ?, ?)";
      statement = conn.prepareStatement(sql);
      statement.setString(1, "Vagwan Koirala");
      statement.setString(2, "Budanilkantha");
      statement.setInt(3, 58);
      statement.setString(4, "9851225478");
      FileInputStream fin = new FileInputStream("Your Path to file");
      statement.setBinaryStream(5, fin);
      int isInserted = statement.executeUpdate();
      if (isInserted == 1) {
        System.out.println("Record added successfully.");
      }
    } catch (ClassNotFoundException e) {
      System.out.println("The given Driver class could not be loaded.");
    } catch (SQLException se) {
      System.out.println("Exception occurred when carrying out SQL operation. " + se.getMessage());
    }  catch (FileNotFoundException fne) {
      System.out.println("File not found exception. " + fne.getMessage());
    } finally {
      if (conn != null) {
        conn.close();
      }
    }
  }
}
```

SQL Escapes, Multiple Results, Scrollable Result Sets

SQL Escapes: The escape syntax gives you the flexibility to use database specific features unavailable to you by using standard JDBC methods and properties.

Multiple Results: If a stored procedure is returning multiple result sets, you should execute its CallableStatement object with the execute() method. In the case, an internal pointer will be maintained inside the CallableStatement object. This pointer is pointing the current result, which could be a result set or a update count.

Scrollable Result Sets: By default, result sets are not scrollable or updatable. A scrollable ResultSet is one which allows us to retrieve the data in forward direction as well as backward direction but no updations are allowed.

Row Sets, Transactions

Row Sets: It object means it is the object of driver supplied Java class that implements javax.sql.RowSet interface. This RowSet interface is a sub interface of java.sql.ResultSet interface. We have three types of RowSets:

1. **Cached RowSet:** It is a disconnected RowSet. Behavior wise, it is like insensitive ResultSet object.
2. **JDBC RowSet:** It is connected RowSet. Behavior wise, it is like sensitive ResultSet object.
3. **Web RowSet:** It is a connected RowSet. It collects data from xml files.

Transactions: Transactions enable you to control if, and when, changes are applied to the database. It treats a single SQL statement or a group of SQL statements as one logical unit, and if any statement fails, the whole transaction fails.