JDBC

JDBC is a Java API that provides a standard interface for connecting and interacting with relational databases. It allows Java applications to access and manipulate databases.

Let's consider a simple scenario where you have a Java application that needs to interact with a relational database to store and retrieve data. For this example, we'll assume you have a database named "CompanyDB" with a table named "Employees" containing columns such as "employee_id," "employee_name," and "employee_salary."

Here are the basic steps involved in using JDBC:

1. **Load the JDBC Driver:**
   The first step is to load the JDBC driver specific to the database you're using. Different databases have different JDBC drivers. For example, if you're using MySQL, you would load the MySQL JDBC driver. This is typically done using the `Class.forName()` method.

   ```
   java Class.forName("com.mysql.cj.jdbc.Driver");
   ```

2. **Establish a Connection:**
   Once the driver is loaded, you need to establish a connection to the database. You provide the database URL, username, and password to obtain a connection.

   ```
   java String url = "jdbc:mysql://localhost:3306/CompanyDB"; String usernam
   e = "yourUsername"; String password = "yourPassword"; Connection connecti
   on = DriverManager.getConnection(url, username, password);
   ```

3. **Create a Statement:**
   With the connection established, you create a `Statement` or `PreparedStatement` object to execute SQL queries.

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

4. **Execute SQL Queries:**
   You can now execute SQL queries using the `Statement` or `PreparedStatement` object. For example, let's insert a new employee into the "Employees" table.

   ```
   java String insertQuery = "INSERT INTO Employees (employee_name, employee_salary) VALUES ('John Doe', 50000)"; statement.executeUpdate(insertQuery);
   ```

5. **Retrieve Data:**
   You can also retrieve data from the database. Let's fetch all employees from the "Employees" table.

   ```
   java String selectQuery = "SELECT * FROM Employees"; ResultSet resultSet = statement.executeQuery(selectQuery); while (resultSet.next()) { int employeeId = resultSet.getInt("employee_id"); String employeeName = resultSet.getString("employee_name"); double employeeSalary = resultSet.getDouble("employee_salary"); // Process the retrieved data }
   ```

6. **Close Resources:**
   It's crucial to close the `ResultSet`, `Statement`, and `Connection` objects after you've finished using them to release resources and prevent memory leaks.

   ```
   java resultSet.close(); statement.close(); connection.close();
   ```

These steps illustrate the basic flow of working with JDBC in a Java application.