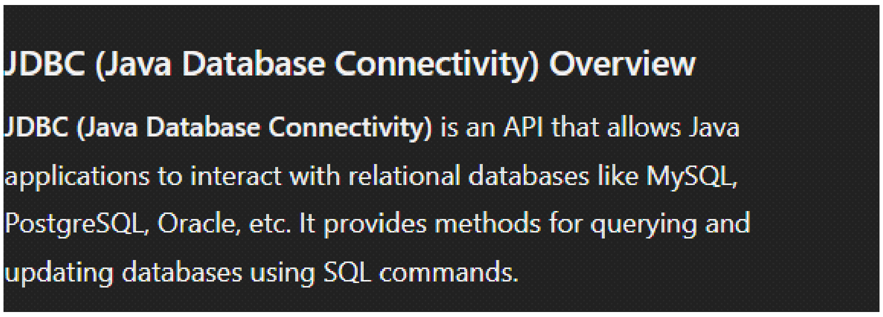
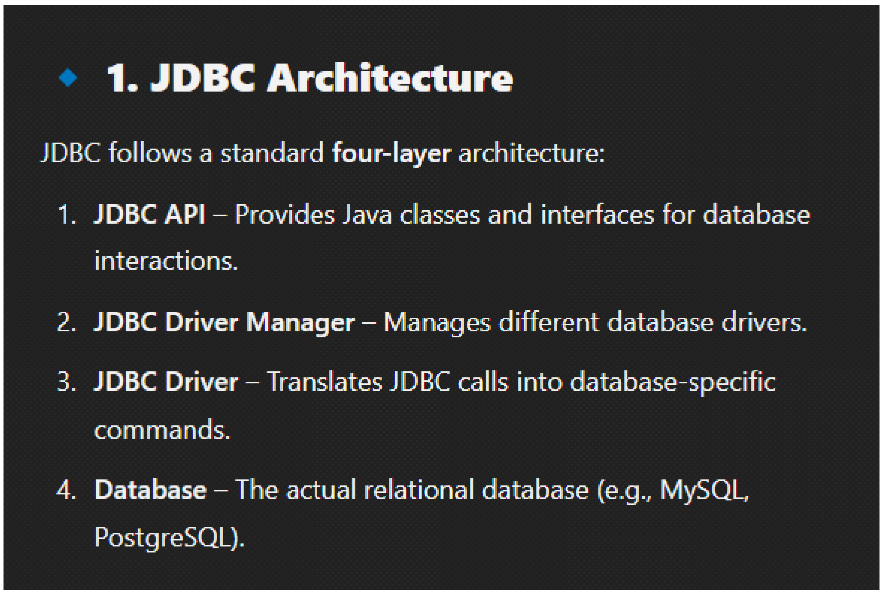
JDBC Drivers

09 February 2025 02:24



JDBC (Java Database Connectivity) Overview

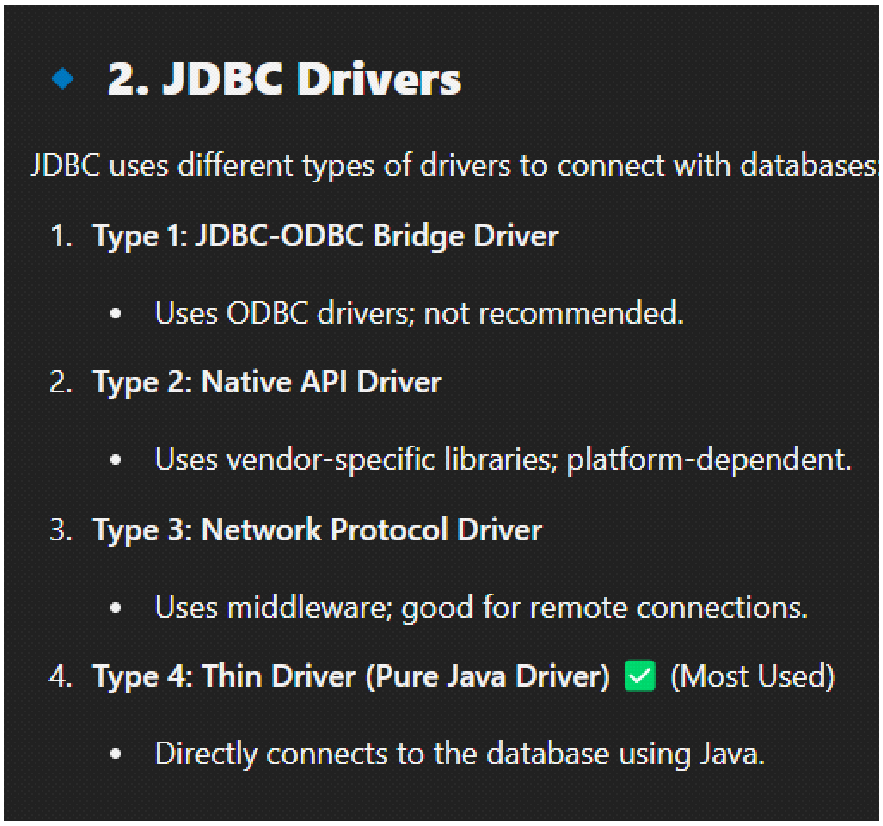
JDBC (Java Database Connectivity) is an API that allows Java applications to interact with relational databases like MySQL, PostgreSQL, Oracle, etc. It provides methods for querying and updating databases using SQL commands.



🔹 1. JDBC Architecture

JDBC follows a standard **four-layer** architecture:

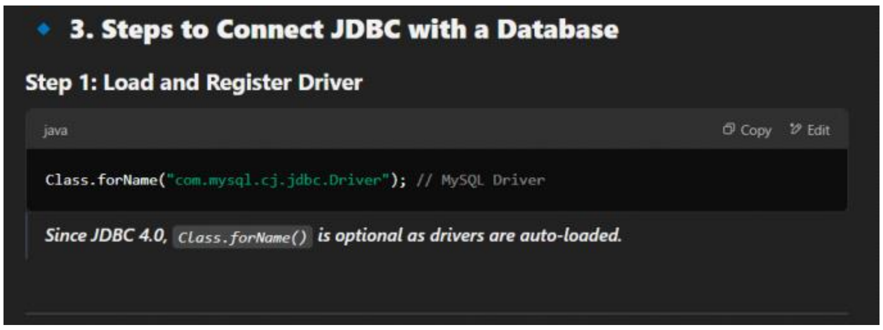
1. **JDBC API** – Provides Java classes and interfaces for database interactions.
2. **JDBC Driver Manager** – Manages different database drivers.
3. **JDBC Driver** – Translates JDBC calls into database-specific commands.
4. **Database** – The actual relational database (e.g., MySQL, PostgreSQL).



🔹 2. JDBC Drivers

JDBC uses different types of drivers to connect with databases

1. **Type 1: JDBC-ODBC Bridge Driver**
   - Uses ODBC drivers; not recommended.
2. **Type 2: Native API Driver**
   - Uses vendor-specific libraries; platform-dependent.
3. **Type 3: Network Protocol Driver**
   - Uses middleware; good for remote connections.
4. **Type 4: Thin Driver (Pure Java Driver)** ✅ (Most Used)
   - Directly connects to the database using Java.

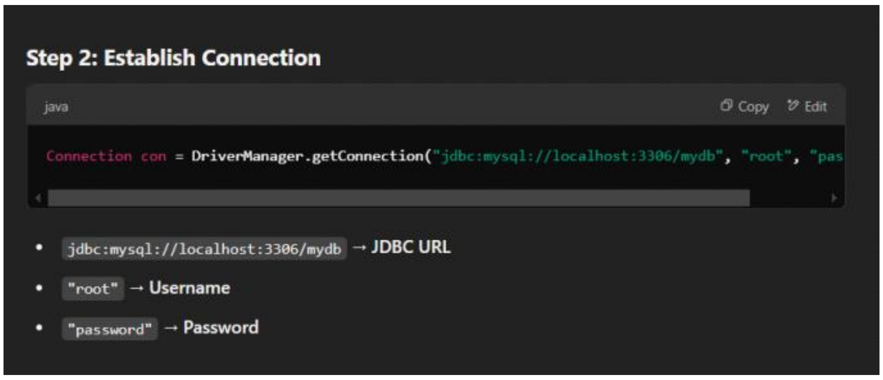


3. Steps to Connect JDBC with a Database

Step 1: Load and Register Driver

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

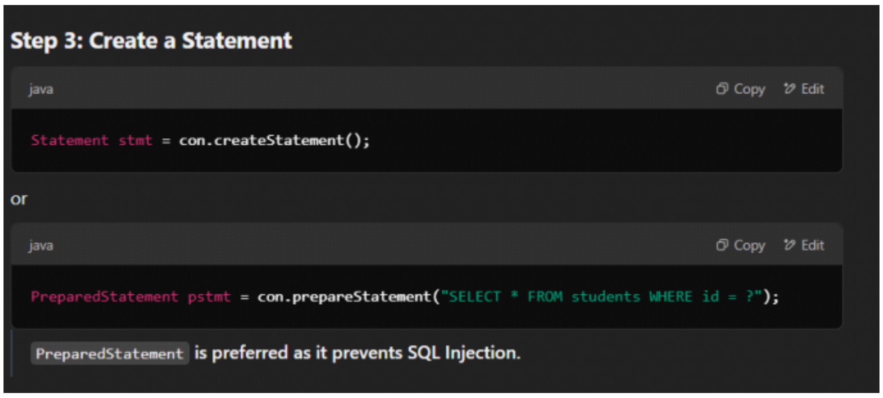
> *Since JDBC 4.0, `Class.forName()` is optional as drivers are auto-loaded.*



Step 2: Establish Connection

```java
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "pas
```

- `jdbc:mysql://localhost:3306/mydb` → JDBC URL
- `"root"` → Username
- `"password"` → Password



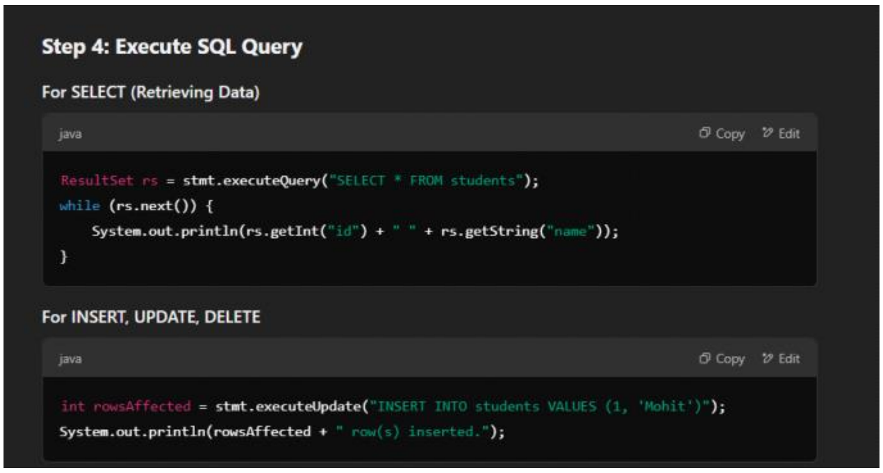
Step 3: Create a Statement

```java
Statement stmt = con.createStatement();
```

or

```java
PreparedStatement pstmt = con.prepareStatement("SELECT * FROM students WHERE id = ?");
```

> `PreparedStatement` is preferred as it prevents SQL Injection.



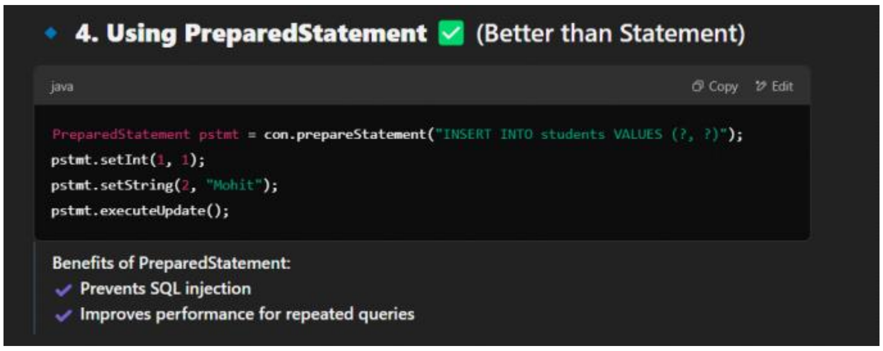
Step 4: Execute SQL Query

For SELECT (Retrieving Data)

```java
ResultSet rs = stmt.executeQuery("SELECT * FROM students");
while (rs.next()) {
    System.out.println(rs.getInt("id") + " " + rs.getString("name"));
}
```

For INSERT, UPDATE, DELETE

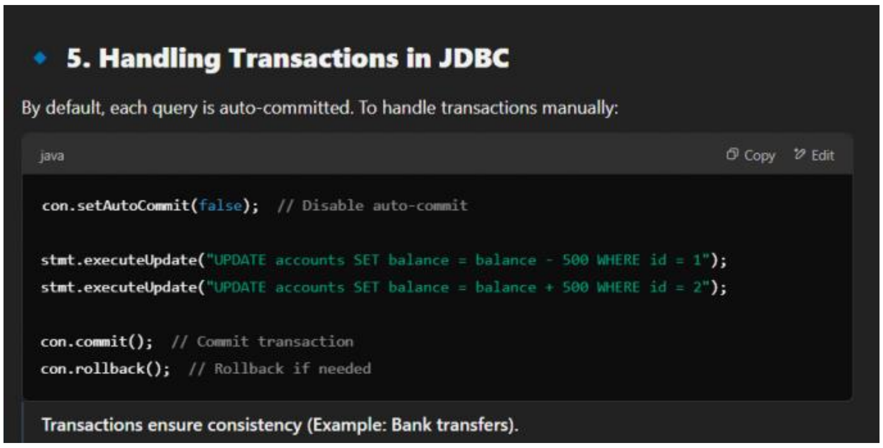
```java
int rowsAffected = stmt.executeUpdate("INSERT INTO students VALUES (1, 'Mohit')");
System.out.println(rowsAffected + " row(s) inserted.");
```



4. Using PreparedStatement ✅ (Better than Statement)

```java
PreparedStatement pstmt = con.prepareStatement("INSERT INTO students VALUES (?, ?)");
pstmt.setInt(1, 1);
pstmt.setString(2, "Mohit");
pstmt.executeUpdate();
```

> Benefits of PreparedStatement:
> ✓ Prevents SQL injection
> ✓ Improves performance for repeated queries



5. Handling Transactions in JDBC

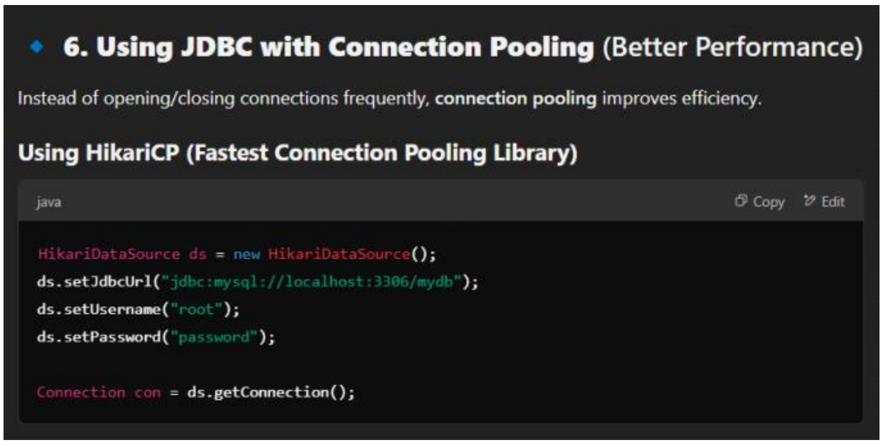
By default, each query is auto-committed. To handle transactions manually:

```java
con.setAutoCommit(false);  // Disable auto-commit

stmt.executeUpdate("UPDATE accounts SET balance = balance - 500 WHERE id = 1");
stmt.executeUpdate("UPDATE accounts SET balance = balance + 500 WHERE id = 2");

con.commit();  // Commit transaction
con.rollback();  // Rollback if needed
```

Transactions ensure consistency (Example: Bank transfers).



6. **Using JDBC with Connection Pooling** (Better Performance)

Instead of opening/closing connections frequently, **connection pooling** improves efficiency.

Using HikariCP (Fastest Connection Pooling Library)

```java
HikariDataSource ds = new HikariDataSource();
ds.setJdbcUrl("jdbc:mysql://localhost:3306/mydb");
ds.setUsername("root");
ds.setPassword("password");

Connection con = ds.getConnection();
```

```java
@WebServlet("/student")
public class StudentServlet extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
       try {
           Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root",
"password");
           PreparedStatement pstmt = con.prepareStatement("SELECT * FROM students WHERE id = ?");
           pstmt.setInt(1, Integer.parseInt(request.getParameter("id")));

           ResultSet rs = pstmt.executeQuery();
           while (rs.next()) {
               response.getWriter().println(rs.getString("name"));
           }
       } catch (Exception e) {
           e.printStackTrace();
       }
   }
}
```