JSP 3-4

JSP 3-3

💻 SQL 연산자 & JDBC 개요

🔗 연결 연산자 (||)

> 컬럼끼리 또는 컬럼과 리터럴을 연결하여 출력

```
-- 컬럼 + 컬럼
SELECT ID || PW FROM MEMBER;

-- 컬럼 + 문자열
SELECT NAME || '님의 아이디는' || ID || '입니다.' FROM MEMBER;
```

🔎 기타 연산자

1️⃣ `BETWEEN A AND B`

> 범위 내 값 비교

```
SELECT ID FROM MEMBER WHERE ID BETWEEN '1111' AND '3333';
```

2️⃣ `IN` / `NOT IN`

> 특정 목록 포함 여부

```
SELECT * FROM MEMBER WHERE ID IN ('1111', '2222');
SELECT * FROM MEMBER WHERE ID NOT IN ('1111', '2222');
```

3️⃣ `LIKE` / `NOT LIKE`

> 문자 패턴 일치 여부

- `%` : 임의의 문자 (0개 이상)
- `_` : 임의의 단일 문자

```
-- A로 시작
SELECT * FROM MEMBER WHERE PW LIKE 'A%';

-- A 포함
SELECT * FROM MEMBER WHERE PW LIKE '%A%';
```

```sql
-- A로 시작하지 않음
SELECT * FROM MEMBER WHERE PW NOT LIKE 'A%';

-- 두 번째 글자가 E
SELECT * FROM MEMBER WHERE PW LIKE '_E';
```

🔌 JDBC (Java Database Connectivity)

> Java에서 DBMS와 연결하고 데이터를 조작하기 위한 API

- DB 종류와 무관하게 **JDBC 표준 API**로 데이터 접근 가능

🔁 JDBC 연결 순서

1. **드라이버 로드**

```java
Class.forName("oracle.jdbc.driver.OracleDriver");
```

2. **DB 연결**

```java
Connection conn = DriverManager.getConnection(url, id, pw);
```

3. **Statement 객체 생성**

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

4. **SQL 실행**

```java
ResultSet rs = stmt.executeQuery("SELECT * FROM table");
int result = stmt.executeUpdate("INSERT INTO table ...");
```

5. **결과 처리**

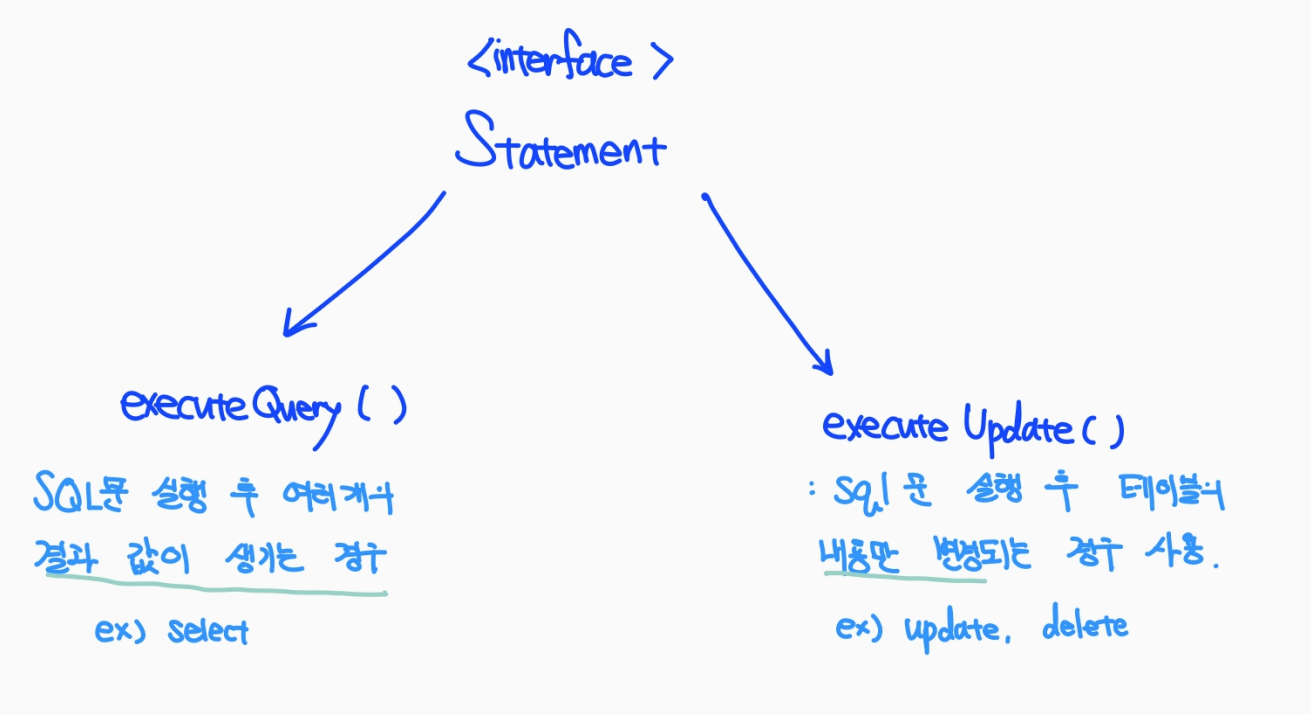
```java
while(rs.next()) {
    String name = rs.getString("name");
```

```
}
```

6. **자원 해제**

```
rs.close();
stmt.close();
conn.close();
```

📜 Statement 객체 구조



📦 executeQuery() & ResultSet

> SELECT 문 전용 실행 메서드 → `ResultSet` 반환

- `next()` : 다음 행으로 이동
- `getXXX(String columnName)` : 컬럼 값 가져오기

```
ResultSet rs = stmt.executeQuery("SELECT * FROM member");
while(rs.next()) {
    String id = rs.getString("id");
    int age = rs.getInt("age");
}
```

```jsp
<%@page import="java.sql.*"%>
```

=> Connection, Statement, ResultSet 을 위해

```jsp
<body>
<%!
//Connection: DB와 연결성을 갖는 객체 (java.sql.PreparedStatement)
Connection conn;

//PreparedStatement: DB에 접근하여 SQL문을 실행하는 객체 (java.sql.Connection)
PreparedStatement pstmt;

//Resultset: 조회(select) 결과의 데이터르 갖는 객체 (java.sql.ResultSet)
ResultSet rs;

//OracleDriver: Oracle JDBC 드라이버 클래스이다
//자바 프로그램에서 데이터베이스에 접속하려면
//드라이버 프로그램에 등록하는 코드를 제공해야한다
String driver = "oracle.jdbc.OracleDriver";
String url="jdbc:oracle:thin:@localhost:1521:xe";
String uid = "           ";
String upw = "       ";
String query = "SELECT * FROM KGMEMBER";
%>
```

```jsp
<%
try
{
    Class.forName(driver); //Oracle 드라이버 클래스를 메모리로 로딩

    //getConnection: 정적 메소드
    //Connection객체는 DB와 연결성을 갖는 객체
    conn = DriverManager.getConnection(url, uid, upw);
    pstmt = conn.prepareStatement(query);

    //PreparedStatement 의 executeQuery 메소드로 SQL문 전송
    //데이터의 결과 집합은 ResultSet으로 반환
    rs = pstmt.executeQuery();
    System.out.println("DB연결 성공");
    System.out.println("qq");
```

DB연결 실패할수 있으니, 예외처리 해준다.

```jsp
//next: 다음 레코드가 있는 지 확인 후
//다음 레코드가 없다면 false 반환
//다음 레코드가 있다면 true를 반환 후
//커서를 다음 레코드로 이동시킨다.
while(rs.next())
{
    System.out.println("읽기를 시작합니다.");
    //Result의 getter 메소드로 컬럼의 이름을 문자열로 지정하여
    //해당하는 컬럼의 값을 얻을 수 있다

    String id = rs.getString("ID");
    String pw = rs.getString("PW");
    String name = rs.getString("NAME");
    String email = rs.getString("EMAIL");
    String address = rs.getString("ADDRESS");
    Timestamp regdate = rs.getTimestamp("REGDATE");

    System.out.print(id);

    out.print("id: " + id + "<br>");
    out.print("pw: " + pw + "<br>");
    out.print("name: " + name + "<br>");
    out.print("email: " + email + "<br>");
    out.print("address: " + address + "<br>");
    out.print("regdate: " + regdate + "<hr>");
}
```

```jsp
catch(Exception e)
{
    e.printStackTrace();
    System.out.println("DB연결 실패");
}
finally
{
    //예외가 발생하든 말든 항상 발생해야함
    //DB 관련 작어비 완료된 다음에는 사용했던 객체들은 메모리에서 해제해 주어야한다
    //해제하는 순서는 최근에 사용했던 객체부터 거꾸로 올라가며 해제


    try
    {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null)conn.close();

    }catch(Exception e2){}

}
```