데이터베이스 연동

데이터베이스 연동

- **테이블 만들기(테이블명: board)**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
title	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
writer	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
contents	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
regdate	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
hit	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

AI→Auto Increment
Insert시 값을 채우지 않아도
자동으로 1씩 증가된 값이 생성

Column Name:
Charset/Collation: Default Charset Default Collation
Comments:

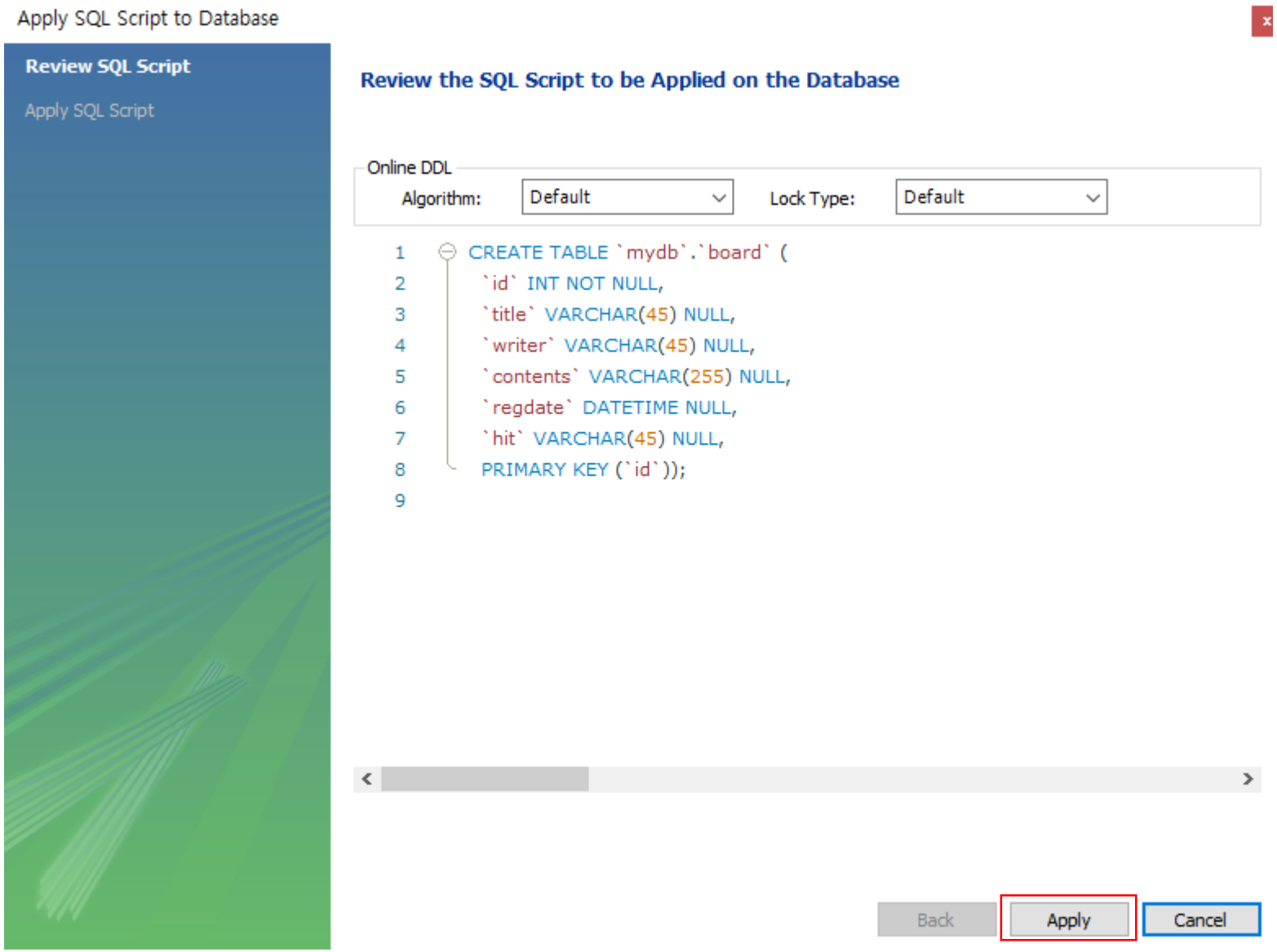
Data Type:
Default:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

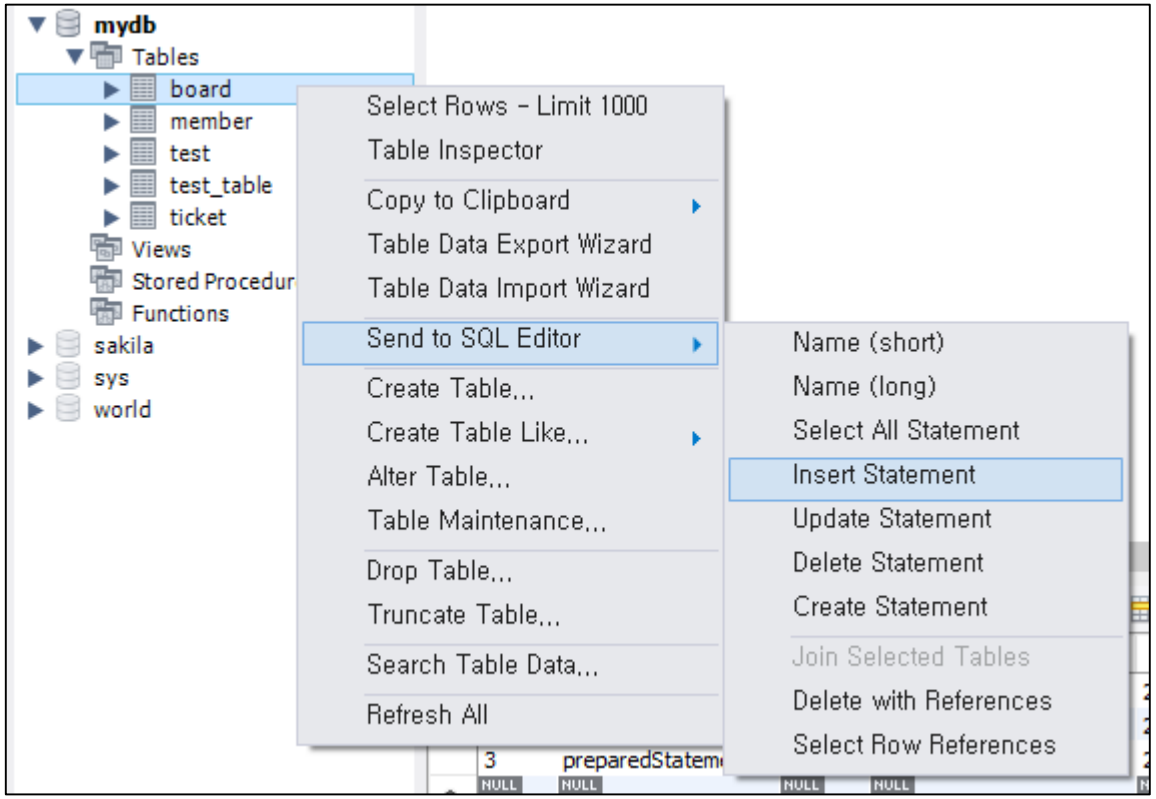
JDBC 프로그래밍

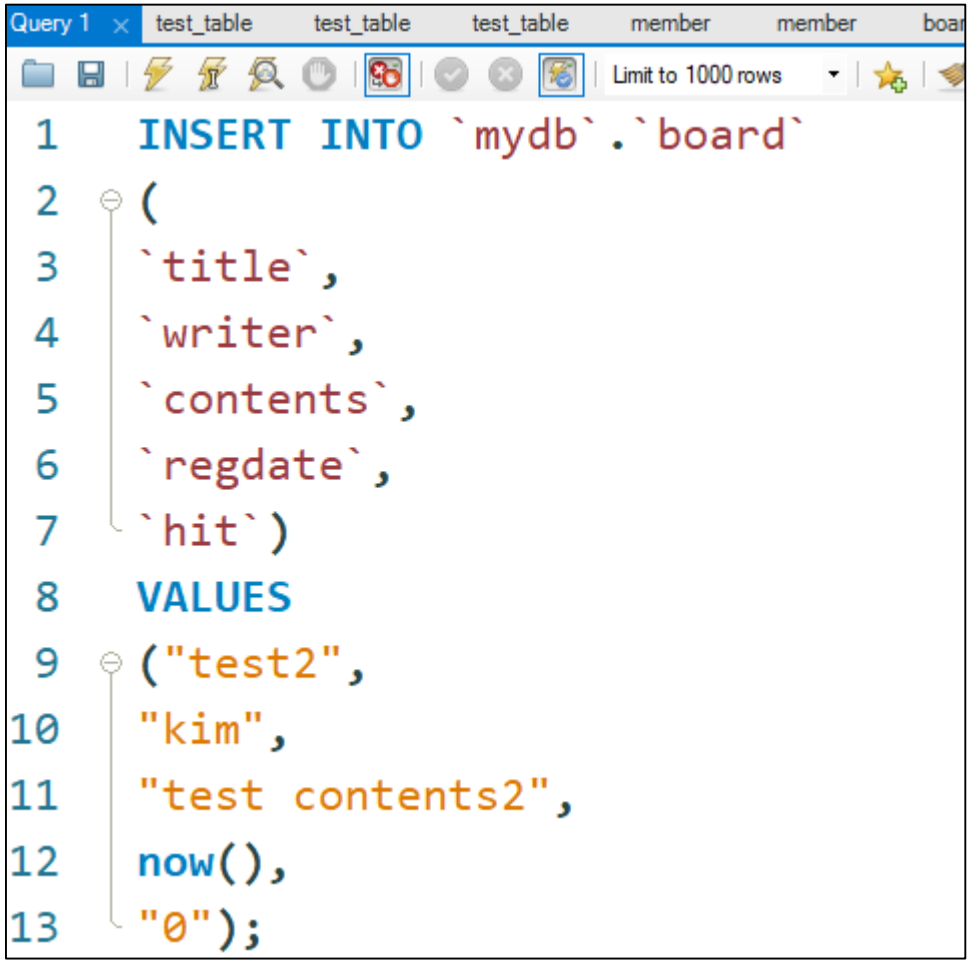
- 테이블 만들기



JDBC 프로그래밍

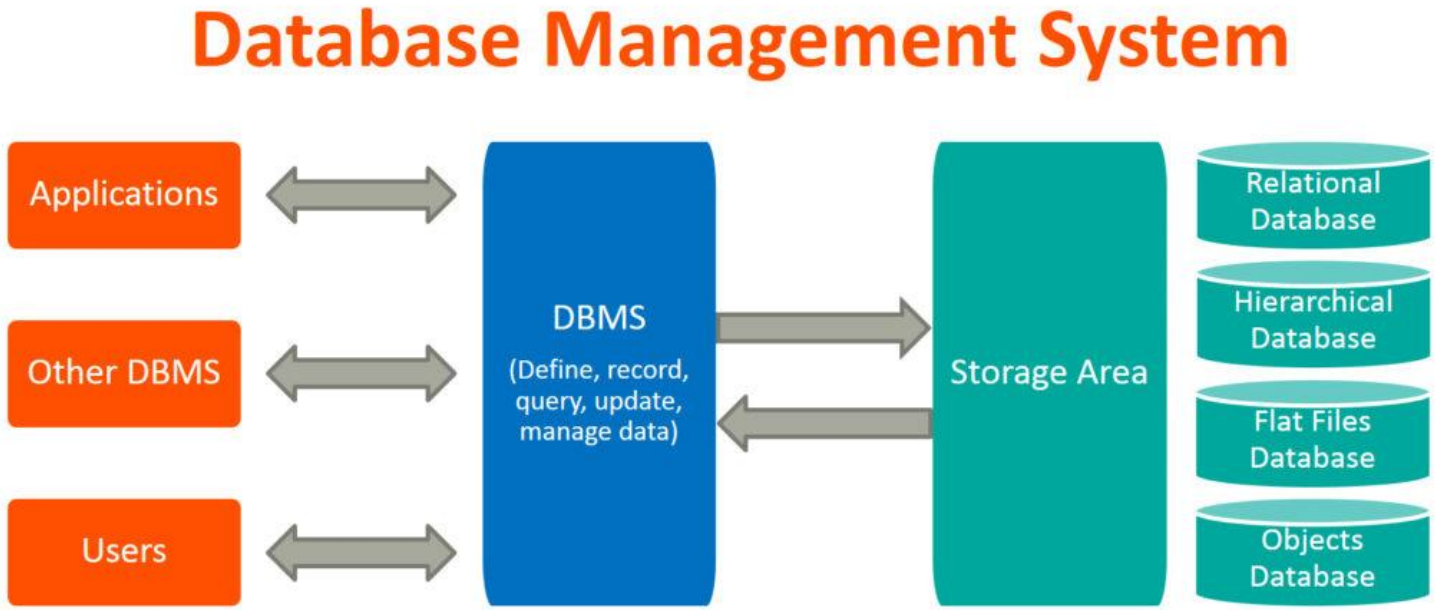
- 더미 데이터 채우기





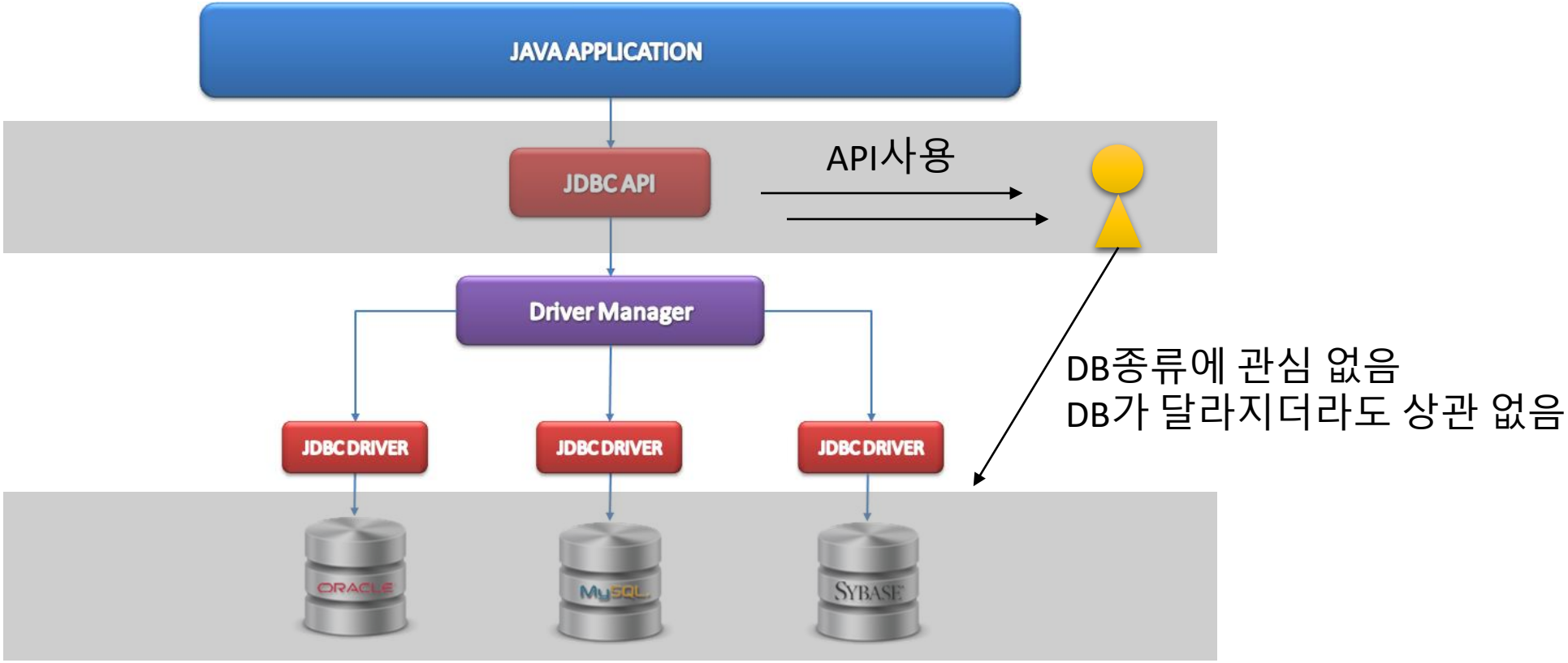
MVC모델

- **DBMS**
  - 사용자들이 데이터베이스 내의 데이터를 쉽게 조작할 수 있도록 도와주는 소프트웨어 도구
  - 대표적 제품으로 오라클과 MySQL이 있음



개요

- **JDBC(Java Database Connectivity)**
  - 자바에서 데이터베이스에 접속할 수 있도록 하는 자바 API
  - DB 제조사마다 사용하는 프로토콜이 다르므로 클라이언트는 이에 맞게 프로그램을 작성해야 함
  - 그러나 JDBC API계층을 통해 프로그래머는 사용하는 DB의 종류와 관계 없이 동일한 API를 사용할 수 있게 됨



개요

- **SQL(Structured Query Language)**
  - 관계형 데이터베이스 관리 시스템(RDBMS)의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어
  - 본 수업은 데이터베이스 수업이 아니므로 복습차원에서 간단한 문법만을 설명
  - 또한 MySQL workbench를 사용하여 GUI환경에서 좀 더 쉽게 데이터베이스를 조작해 봄

개요

- **SQL(Structured Query Language)**
  - 대/소문자는 속도 차이는 있으나 동작에는 상관 없음

데이터 베이스 생성

CREATE DATABASE 데이터베이스 명
CREATE SCHEMA 데이터베이스 명

데이터 베이스 지정

USE 데이터베이스 명

테이블 만들기

CREATE TABLE 테이블명(
열이름1 데이터형,
열이름2 데이터형,
.....
);

테이블 표시

SELECT * FROM 테이블명;
(*는 모든 열을 의미)

조건 검색

SELECT * FROM 테이블명 WHERE ID = 3

한열만 표시

SELECT 특정 열 이름 FROM 테이블명;

개요

- **SQL(Structured Query Language)**
  - 대/소문자는 속도 차이는 있으나 동작에는 상관 없음

데이터 삽입

```
INSERT INTO 테이블 명 (column1, column2, ...)
VALUES (value1, value2, ...);
```

데이터 수정

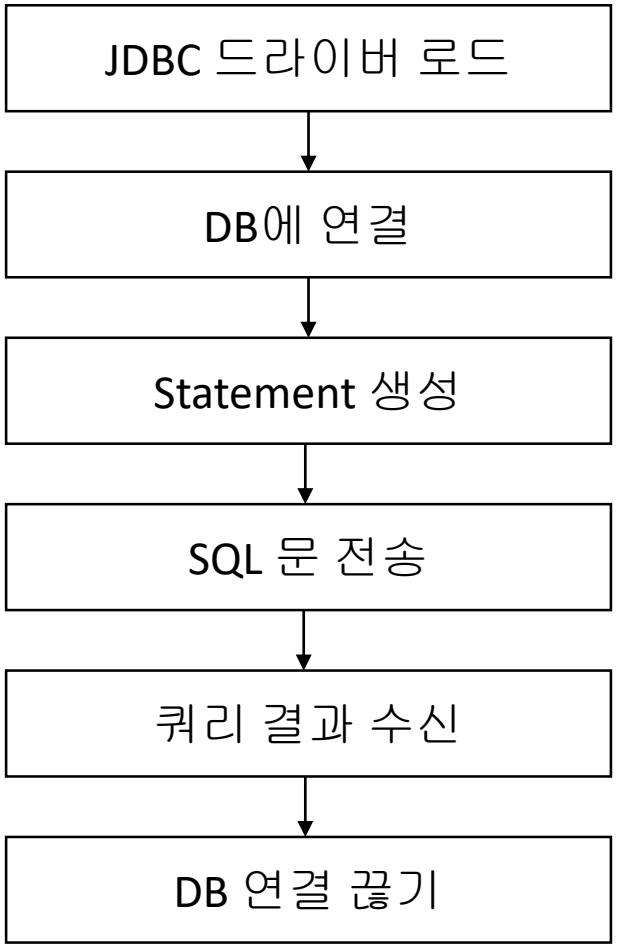
```
UPDATE 테이블 명
SET column1 = value1, column2 = value2, …
WHERE 조건;
```

데이터 삭제

```
DELETE FROM 테이블 명 WHERE 조건;
```

JDBC 프로그래밍

- **JDBC 프로그래밍 절차**



MySQL용 드라이버를 프로그램에 로드

DB경로, 아이디, 패스워드를 인자로 입력하여 DB에 접속

SQL문을 데이터베이스에 전송하는 데 사용

Statement객체의 executeQuery 혹은 executeUpdate
메서드에 SQL을 입력하여 쿼리 실행

쿼리 결과 수신

연결 종료

JDBC 프로그래밍

```
try {
    Class.forName(driver);
    con = DriverManager.getConnection(url, user, pw);
    stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM board");
    while(rs.next()) {
        String id = rs.getString("id");
    }
} catch (SQLException e) {
    System.out.println("[SQL Error : " + e.getMessage() + "]");
} catch (ClassNotFoundException e1) {
    System.out.println("[JDBC Connector Driver Error : " + e1.getMessage() + "]");
} finally {
    if (con != null) {
        try {
        con.close();
        } catch (Exception e) {

        }
    }
    . . .
}
```

DB에 연결

SQL실행

실행결과 처리

DB 연결 끊기

JDBC 프로그래밍

- **JDBC 드라이버 로드와 DB연결**

database연결 시 ssl 보안기능이 포함되지 않을 경우 error가 발생할 수 있음

```
String driver = "com.mysql.jdbc.Driver";
String url =
"jdbc:mysql://localhost/mydb?characterEncoding=utf8&serverTimezone=UTC&useSSL=false";
String user = req.getServletContext().getInitParameter("dbId");
String pw = req.getServletContext().getInitParameter("dbPassword");

try {
Class.forName(driver);
con = DriverManager.getConnection(url, user, pw);
```

Class.forName("package.ClassName")

```
public class Driver extends NonRegisteringDriver implements java.sql.Driver {
    static {
        try {
            java.sql.DriverManager.registerDriver(new Driver());
        } catch (SQLException E) {
            throw new RuntimeException("Can't register driver!");
        }
    }
}
```

JDBC 프로그래밍

- **Statement 객체(SQL실행)**
  - Statement
    - 정적인 쿼리문을 주로 처리
    - 즉, 미리 결정된 쿼리문을 실행 → "SELECT * FROM board"
  - PreparedStatement(Statement로 부터 상속받음, Statement문을 보완)
    - 동적인 쿼리문을 주로 처리
    - 즉, 런타임 시 결정되는 쿼리문을 실행 → "SELECT * FROM board WHERE writer=kim"
    - 코드 안정성 및 가독성이 좋으므로 인수가 많은 복잡한 SQL문을 실행할 때 좋음

JDBC 프로그래밍

- **Statement 객체(SQL실행)**

```
String name = request.getParameter("ID");//kim
String address = request.getParameter("ADDR");//DB108
String major = request.getParameter("MAJOR");//Network

Statement stmt = conn.createStatement();

String query = "INSERT INTO MEMBER VALUES('"+ name+"', '" +
address + "', '" +major + "')";

stmt.executeUpdate(query);
```

"INSERT INTO MEMBER VALUES(kim,DB108,NETWORK)" 라는 문자열 생성을 위해 복잡한 자바 코드가 필요

->가독성이 떨어지고 에러가 날 확률이 높아짐

JDBC 프로그래밍

- **Statement 객체(SQL실행)**

```
PreparedStatement pstmt = null;
String preQuery = "INSERT INTO board(title,writer,contents,regdate,hit) VALUES (?,?,?,?,?)";

pstmt = con.prepareStatement(preQuery);
pstmt.setString(1, "preparedStatement");
pstmt.setString(2, "lee");
pstmt.setString(3, "preparedStatement test");
pstmt.setTimestamp(4, Timestamp.valueOf(LocalDateTime.now()));
pstmt.setString(5, "0");
pstmt.executeUpdate();
```

JDBC 프로그래밍

- **Statement 객체를 이용하여 SQL을 실행하는 3가지 방법**
  - SQL구문의 종류: SELECT, INSERT, UPDATE, DELETE
  - execute
    - 수행 결과로 Boolean 타입의 값을 반환
    - 모든 구문을 수행
  - **executeQuery**
    - 수행 결과로 **ResultSet** 객체의 값을 반환
    - SELECT 구문을 수행할 때
  - **executeUpdate**
    - 수행 결과로 SQL이 적용된 행의 개수를 int형으로 반환
    - INSERT, UPDATE, DELETE에 사용

JDBC 프로그래밍

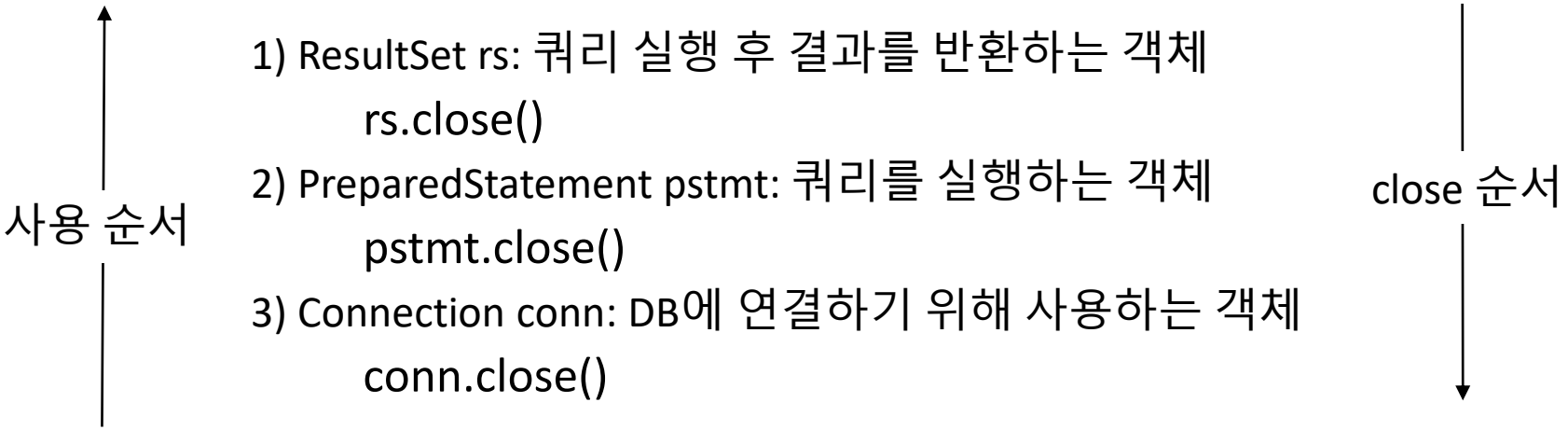
- **쿼리 결과 수신(ResultSet)**
  - DB에서 어떤 레코드를 들고 왔을 때(SELECT) 사용

```
while(rs.next()) {//select결과는 여러 레코드를 포함
    String id = rs.getString("id");
    String title = rs.getString("title");
    String writer = rs.getString("writer");
    String contents = rs.getString("contents");
    LocalDateTime regdate = rs.getTimestamp("regdate").toLocalDateTime();
    System.out.printf("%s | %s | %s | %s | %s \n",
    id,title,writer,contents,regdate.toString());
    System.out.println("-----------------------------------");
}
```

  - ResutlSet클래스에는 다양한 getter가 존재(get*)
  - 대표적으로 getString, getInt, getTimestamp(date 관련)를 주로 사용

JDBC 프로그래밍

- **쿼리 결과 수신**
  - 연결 끊기
  - 일반적 세 번의 close를 수행(정형화된 패턴)



  - 중첩되는 try-catch문은 가독성이 떨어지므로 Java 7 try-with-resource 방식으로 간소화 할 수 있음
    - https://androphil.tistory.com/763
    - https://codechacha.com/ko/java-try-with-resources/