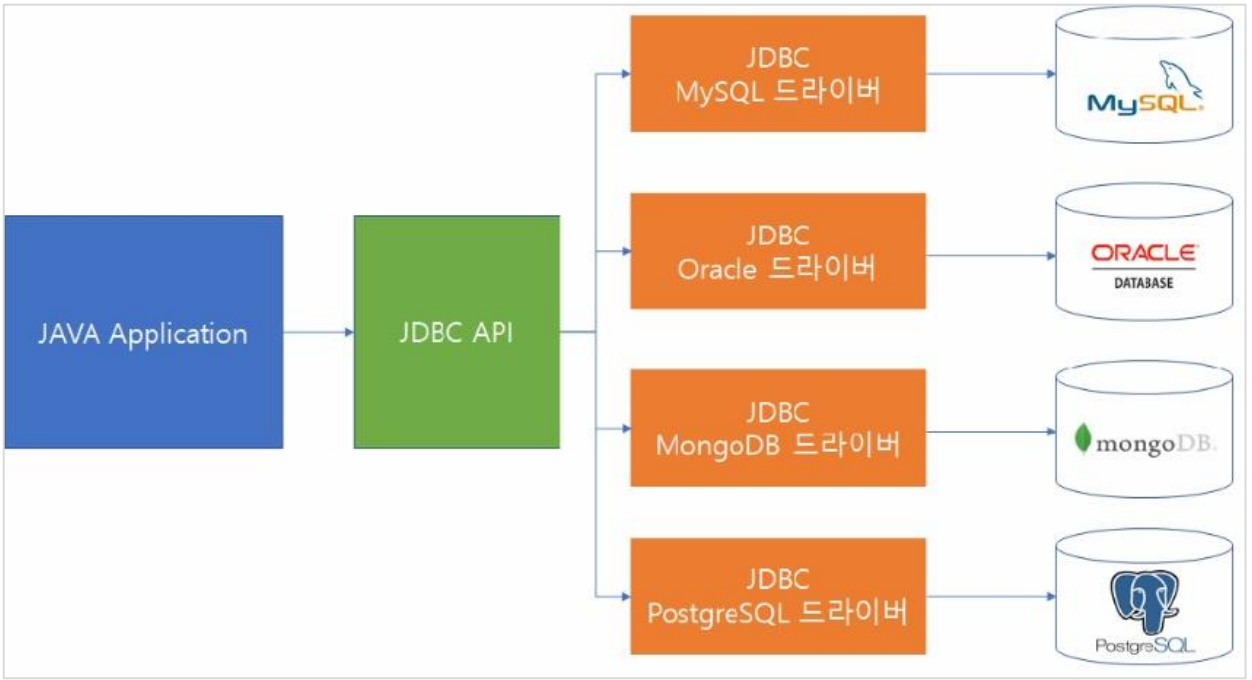
13장. Java – MySQL DB 연동

JDBC Connection

JDBC(Java Database Connectivity)

◆ JDBC(Java Database Connectivity) 정의와 사용

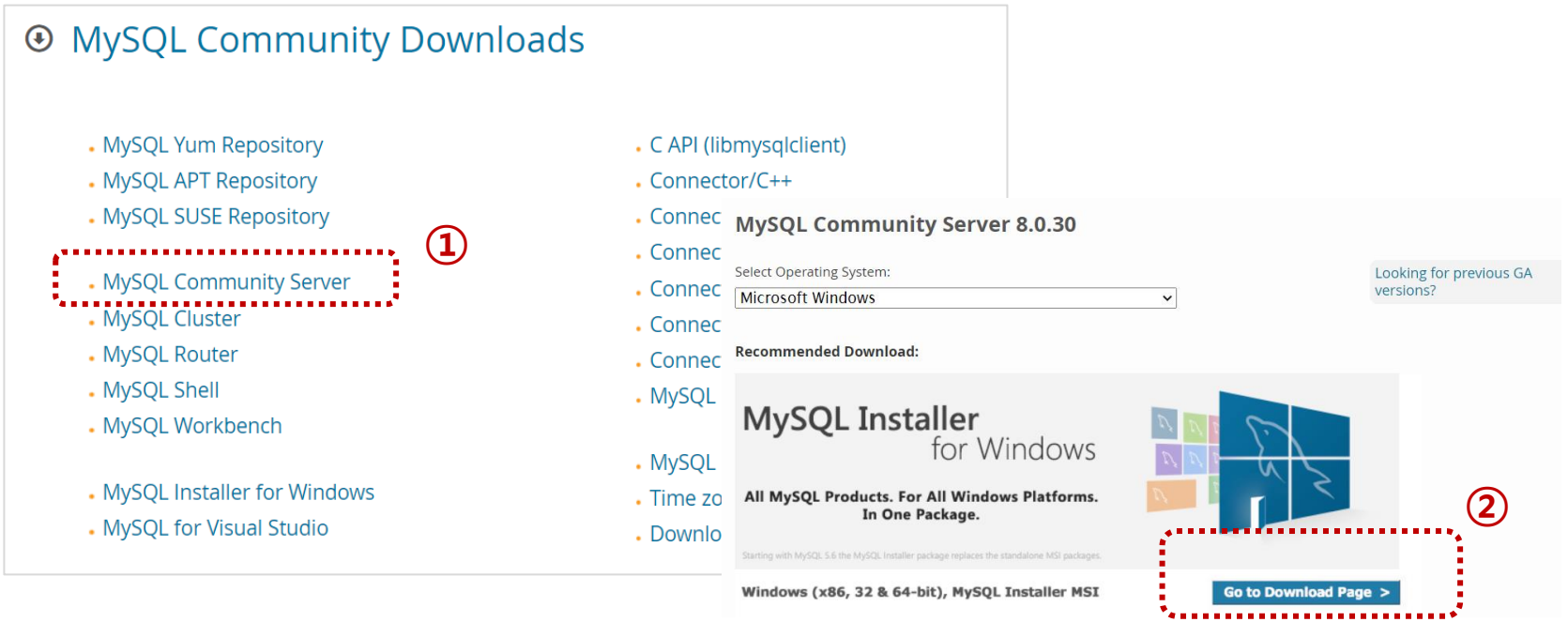
- 자바 애플리케이션에서 DBMS에 연결해주는 기능을 갖는 API이다.
- 오라클, MySQL, MS-SQL 개발사가 구현한 '드라이버'가 존재함



MySQL 설치하기

▷ MySQL 설치하기

mysql download 검색 > MySQL Community Downloads



MySQL 설치하기

▷ MySQL 설치하기

MySQL Installer 8.0.43

Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:

8.0.43

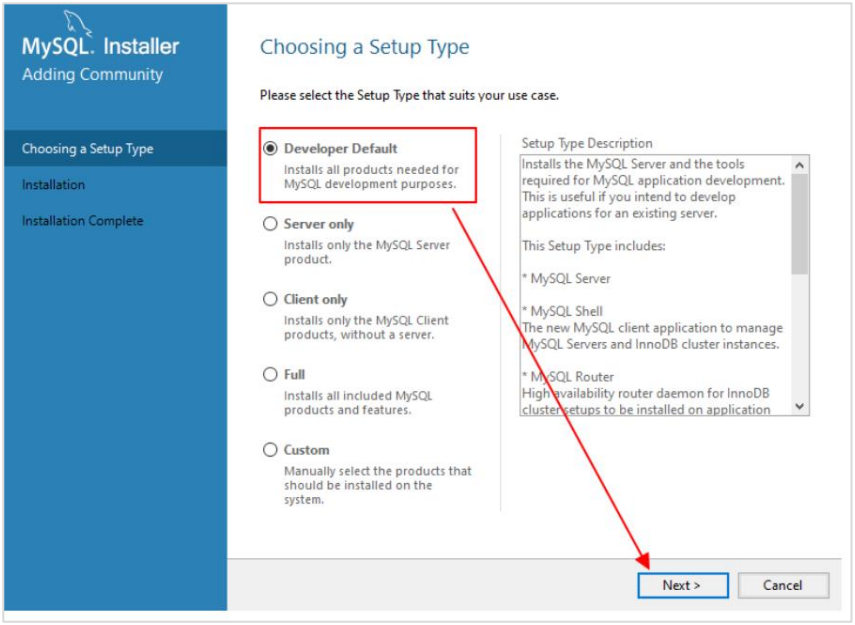
Select Operating System:

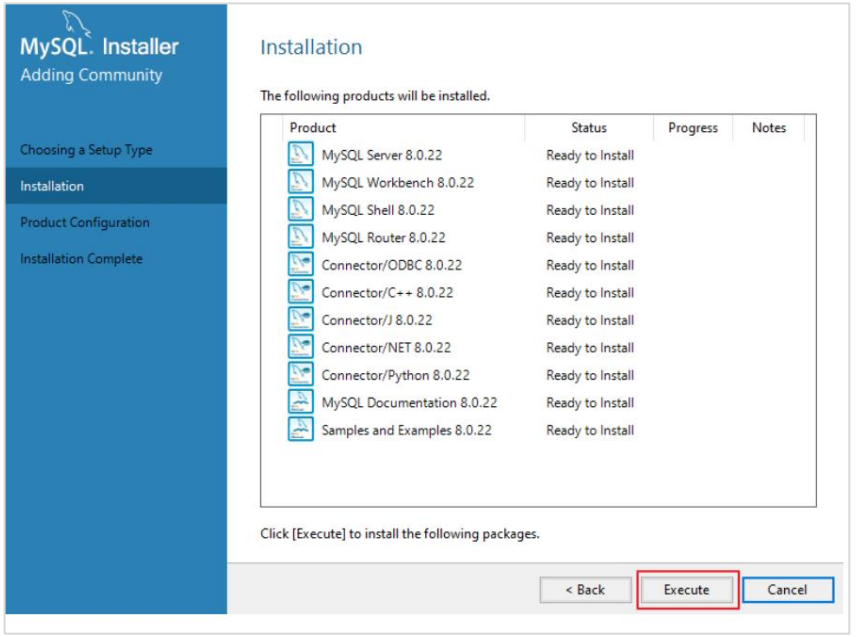
Microsoft Windows

Windows (x86, 32-bit), MSI Installer	8.0.43	2.1M	**Download**
(mysql-installer-web-community-8.0.43.0.msi)		MD5: 0d067e364c208122e7fc62ecbf00d985 | Signature	
Windows (x86, 32-bit), MSI Installer	8.0.43	354.3M	**Download**
(mysql-installer-community-8.0.43.0.msi)		MD5: af62a4e2c58b5013bb5e75531e088b29 | Signature	

MySQL 설치하기

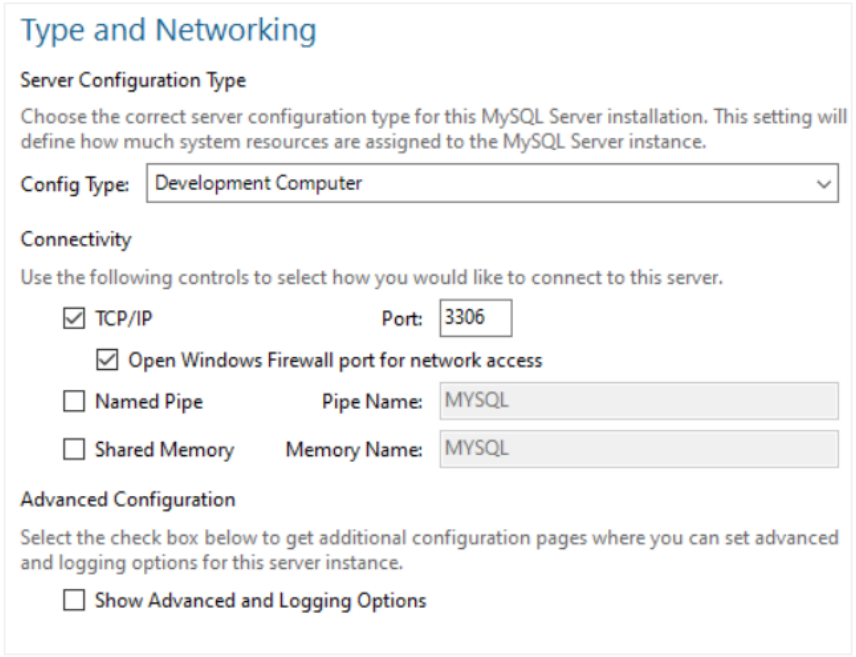
▷ MySQL 설치하기



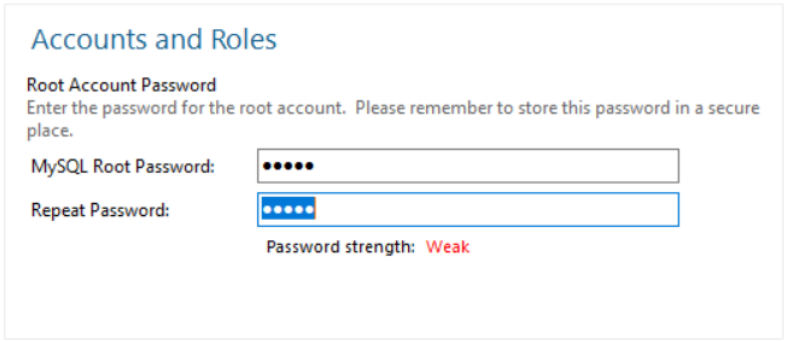


MySQL 설치하기

▷ MySQL 설치하기



포트 : 3306

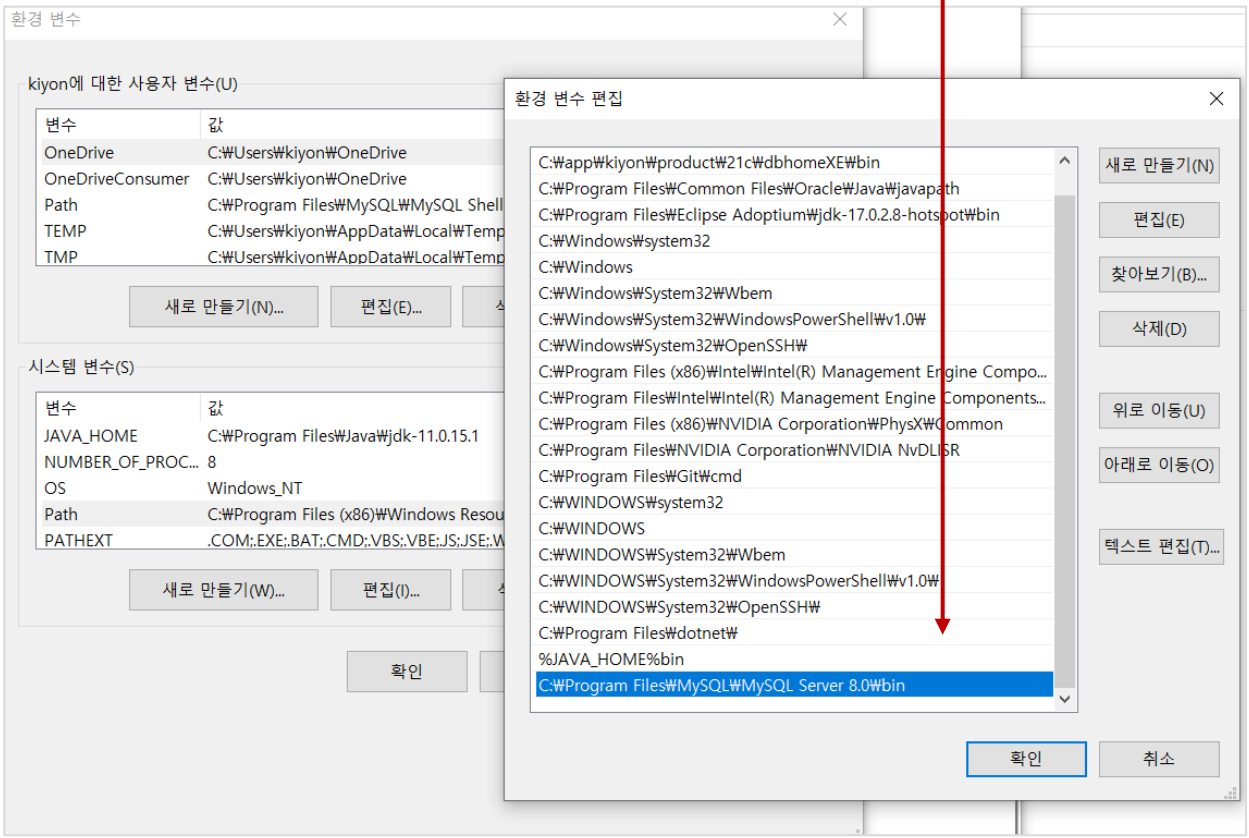


사용자 : root
비밀번호 : pw1234

MySQL 설치하기

▷ 환경 변수 설정하기

C:₩Program Files₩MySQL₩MySQL Server 8.0₩bin



MySQL 설치하기

▷ CMD에서 mysql 사용

mysql --version

mysql – u root –p 명령어 입력 > password 입력 > show databases;

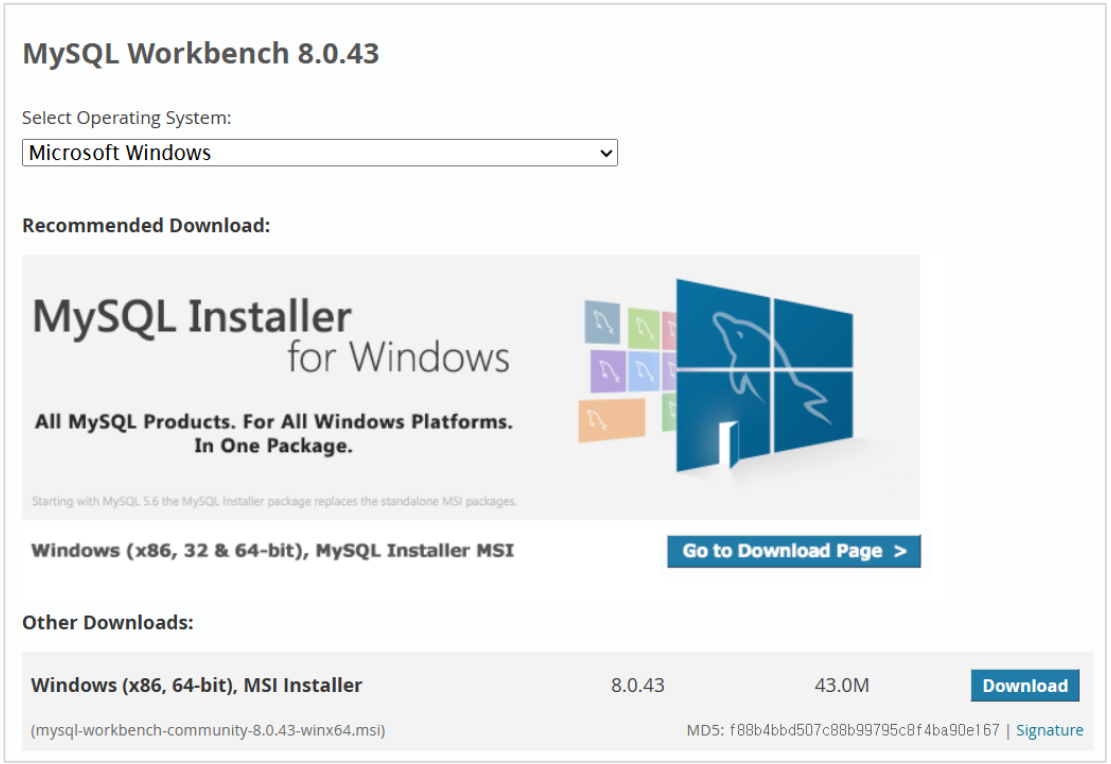
```
C:₩Users₩kiyon>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or ₩g.
Your MySQL connection id is 92
Server version: 5.5.5-10.6.7-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '₩h' for help. Type '₩c' to clear the current input statement.

mysql>
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| bootboard          |
| bootdb             |
| bootdb2            |
| information_schema |
| javaweb            |
| jspdb              |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
9 rows in set (0.01 sec)
```

MySQL 워크벤치

- MySQL Workbench 설치

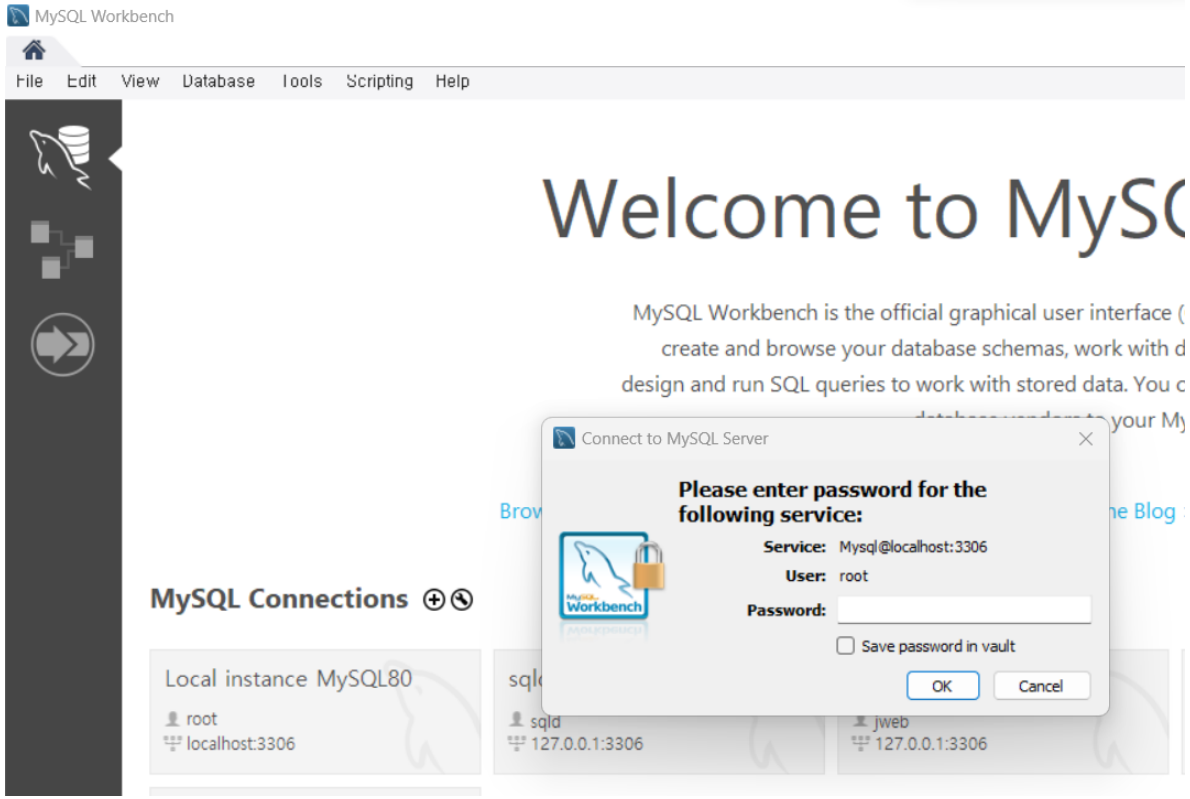
  mysql Workbench 검색 > Downloads



MySQL 워크벤치

- MySQL Workbench 접속

  root 계정으로 접속 > password 입력



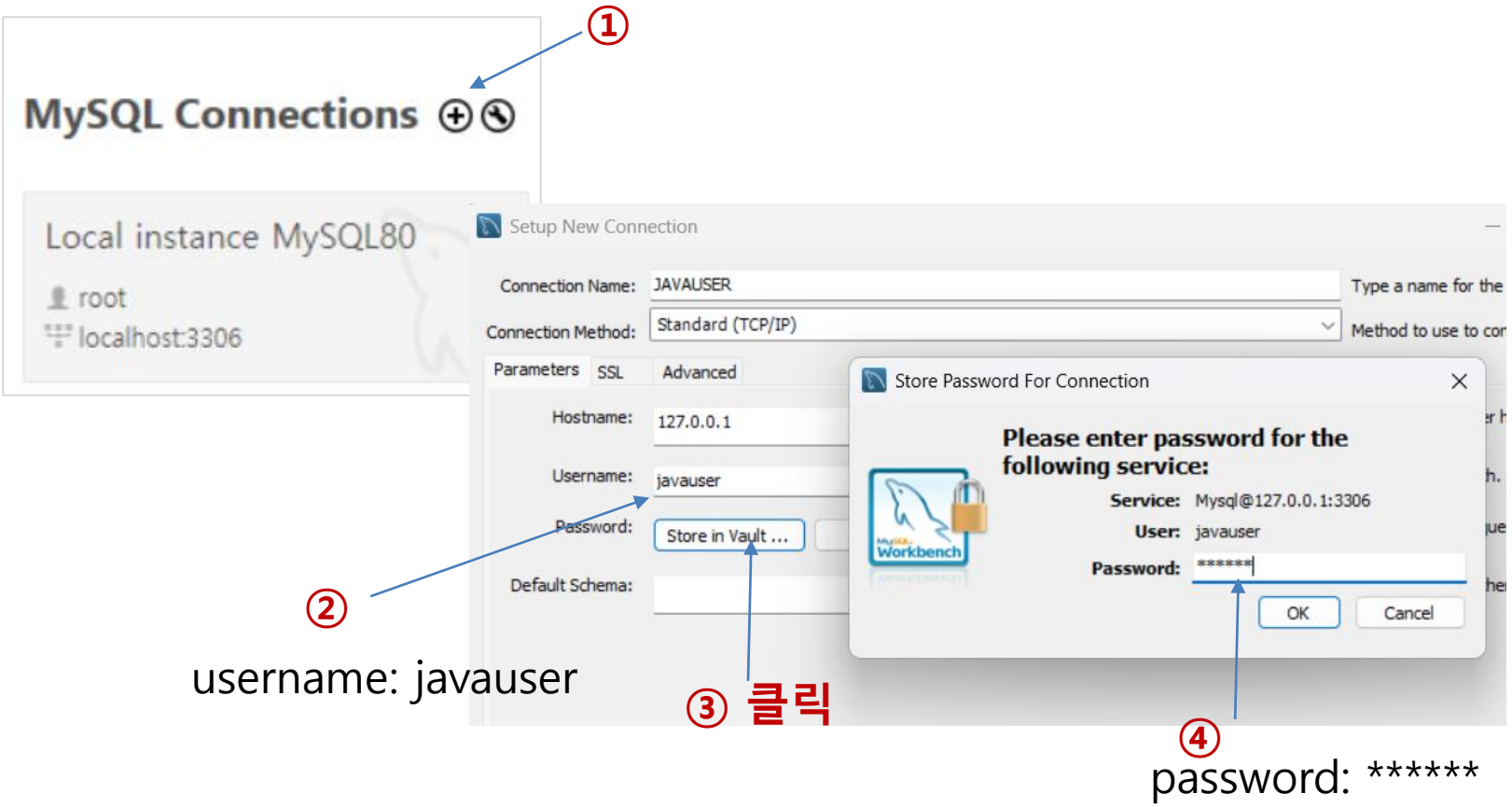
MySQL 워크벤치

- 새 계정 만들기
  - **username: javauser**
  - **db : javadb**

```
-- javauser 생성
create user javauser@localhost identified by 'pwjava';

-- javadb 생성
create database javadb;

-- 권한 부여하기
grant all privileges on javadb.* to javauser@localhost;
```

MySQL 워크벤치

- javauser 계정 접속 화면 만들기



SQL이란?

- **SQL(Structured Query Language)**
  - '에스큐엘', 또는 '시퀄'이라 부른다.
  - 사용자와 데이터베이스 시스템 간에 의사 소통을 하기 위한 언어이다.
  - 사용자가 SQL을 이용하여 DB 시스템에 데이터의 검색, 조작, 정의 등을 요구하면 DB 시스템이 필요한 데이터를 가져와서 결과를 알려준다.

구분	개념
DDL(Data Defiintion Language) - 데이터 정의어	테이블을 포함한 여러 객체를 생성, 수정, 삭제하는 명령어
DML(Data Manipulation Language) - 데이터 조작어	데이터를 저장, 검색, 수정, 삭제하는 명령어
DCL(Data Control Language) - 데이터 제어어	데이터 사용 권한과 관련된 명령어

SQL – DDL(데이터 정의어)

- **DDL(Data Definition Language)**
  - 데이터를 저장할 테이블의 구조를 만드는 명령어이다.
  - DDL은 번역한 결과가 데이터 사전이라는 특별한 파일에 여러 개의 테이블로써 저장된다.
  - CREATE SCHEMA, CREATE TABLE, CREATE VIEW, CREATE INDEX, ALTER, DROP 등이 있다.

명령어	설 명
CREATE	테이블을 생성함
ALTER	테이블의 구조를 변경함
DROP	테이블을 삭제함

SQL – DML(데이터 조작어)

- **DML(Data Manipulation Language)**
  - 데이터를 조작하는 명령어이다.
  - 데이터를 조작하여 저장하는 일련의 과정을 **트랜잭션(transition)**이라 하며, DML은 트랜잭션을 다루는 명령어이다.

명령어	설 명
SELECT	테이블에 있는 행을 검색
INSERT	테이블에 새로운 행 삽입
UPDATE	테이블에 있는 행의 내용 갱신
DELETE	테이블의 행을 삭제
COMMIT	작업을 수행
ROLLBACK	작업 수행을 취소

SQL – DCL(데이터 제어어)

- **DCL(Data Control Language)**
  - DCL은 데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어이다.
  - DDL은 데이터 관리자가(DBA)가 데이터 관리를 목적으로 사용한다.
  - DCL에는 GRANT, REVOKE, COMMIT, ROLLBACK, SAVEPOINT 등이 있다.
    - **GRANT / REVOKE**

      GRANT: 권한 부여를 위한 명령어

      REVOKE: 권한 취소를 위한 명령어

```
GRANT RESOURCE TO TODAY;
```

사용자 ID가 'TODAY'인 사람에게 DB 및 테이블을 생성할 수 있는 권한을 부여함

```
GRANT CONNECT TO CLOUD;
```

사용자 ID가 'CLOUD'인 사람에게 DB에 있는 정보를 검색할 수 있는 권한을 부여

SQL – DML(데이터 조작어)

- **DML(Data Manipulation Language)**
  - INSERT 문 – 자료 삽입

```
INSERT INTO 테이블 이름(열이름1, 열이름2...)
VALUES (데이터 값1, 데이터값 2, ...)
```

  - SELECT 문 – 자료 검색

```
SELECT  열이름 (or 별칭)
FROM  테이블 이름
```

SQL – DML(데이터 조작어)

- **DML(Data Manipulation Language)**
  - UPDATE 문 – 레코드 수정

```
UPDATE 테이블 이름 SET 칼럼이름 = 변경할 값
WHERE 칼럼이름 = 칼럼값
```

  - DELETE 문 – 레코드 삭제

```
DELETE  열이름 (or 별칭)
FROM  테이블 이름 WHERE 칼럼이름 = 칼럼값
```

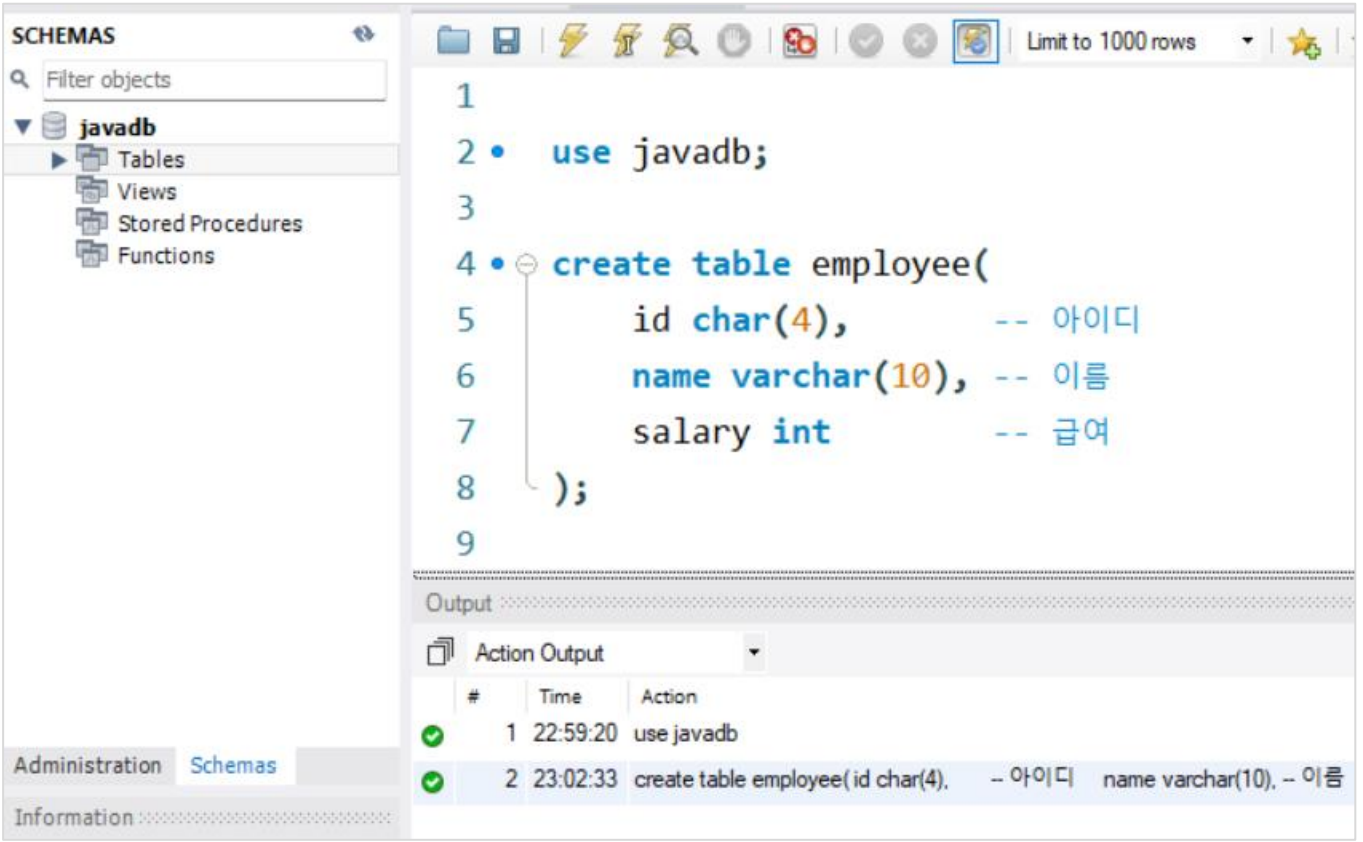
DB – 사원 관리

- **데이터 타입(Data Type)**

데이터 타입	설명
char	고정길이 문자, 최대 255byte, 디폴트값은 1byte
varchar	가변길이 문자, 최대 65536byte, 디폴트값은 1byte
int	가변 숫자, 십진수 기준 최대 220byte
text	가변길이 문자, 최대 65536byte, 디폴트값은 1byte
date	날짜 – 연, 월, 일
timestamp	날짜 – 연, 월, 일, 시, 분, 초, 밀리초

DB – 사원 관리

- 사원 테이블 생성



DB – 사원 관리

- **사원 자료 삽입 및 검색**

```
-- 사원 추가
insert into employee (id, name, salary) values ('e101', '신유빈', 3000000);
insert into employee (id, name, salary) values ('e102', '이정후', 2500000);
insert into employee (id, name) values ('e102', '임시현');

-- 사원 id와 이름 검색
select id, name from employee;

-- 사원의 모든 정보 검색
select * from employee;
```

id	name	salary
e101	신유빈	3000000
e102	이정후	2500000
e102	임시현	NULL

DB – 사원 관리

- 조건절(where)과 정렬(order by)

```
-- 사원 아이디가 'e101'인 사원의 정보
select * from employee where id = 'e101';

-- 급여가 300만원 이상인 회원 검색
select * from employee where salary >= 3000000;

-- 급여가 없는 사원 검색
select * from employee where salary is null;

-- 급여가 적은순으로 오름차순 정렬
select * from employee order by salary asc; -- asc 생략 가능

-- 급여가 많은순으로 내림차순 정렬
select * from employee order by salary desc;
```

id	name	salary
e102	임시현	NULL

id	name	salary
e102	임시현	NULL
e102	이정후	2500000
e101	신유빈	3000000

DB – 사원 관리

● 제약 조건

테이블들은 각 속성(칼럼)에 대한 무결성을 유지하기 위한 다양한 제약 조건(Constraints)이 적용되어 있다.

제약 조건에는 NOT NULL, 기본키, 외래키, CHECK 등이 있다.

✓ NOT NULL

칼럼을 정의할 때 NOT NULL 제약 조건을 명시하면 반드시 데이터를 입력해야 한다.

```
칼럼명  데이터 타입 NOT NULL
```

DB – 사원 관리

- **제약 조건**

  ✓ **기본키(PRIMARY KEY)**

  기본키는 Primay Key라고도 하며, UNIQUE와 NOT NULL 속성을 동시에 가진 제약 조건으로 테이블 당 1개의 기본키만 생성할 수 있다.

```
칼럼명  데이터 타입 PRIMARY KEY
또는
PRIMARY KEY(칼럼명, ...)
```

DB – 사원 관리

- 사원 아이디 중복 등록으로 인한 오류 발생

```
-- 사원 아이디가 'e102'인 사원의 정보
select * from employee where id = 'e102';
```

id	name	salary
e102	이정후	2500000
e102	임시현	NULL

```
-- 사원 추가
insert into employee2 (id, name) values ('e103', '임시현');
```

DB – 사원 관리

- **사원 테이블 신규 생성**

```sql
create table employee2(
    id char(4) primary key,     -- 아이디(기본키 설정)
    name varchar(10) not null, -- 이름(필수 입력)
    salary int                  -- 급여
);

-- 사원 추가
insert into employee2 (id, name, salary) values ('e101', '신유빈', 3000000);
insert into employee2 (id, name, salary) values ('e102', '이정후', 2500000);

-- 아이디 중복(무결성 제약 위배)
insert into employee2 (id, name) values ('e102', '임시현');
```

DB – 사원 관리

- **사원 수정**

```
-- 사원 수정 - 이정후의 급여를 350만원으로 변경(기본키로 검색)
update employee2 set salary = 3500000 where id = 'e102';
```

id	name	salary
e101	신유빈	3000000
e102	이정후	3500000
e103	임시현	NULL
NULL	NULL	NULL

DB – 사원 관리

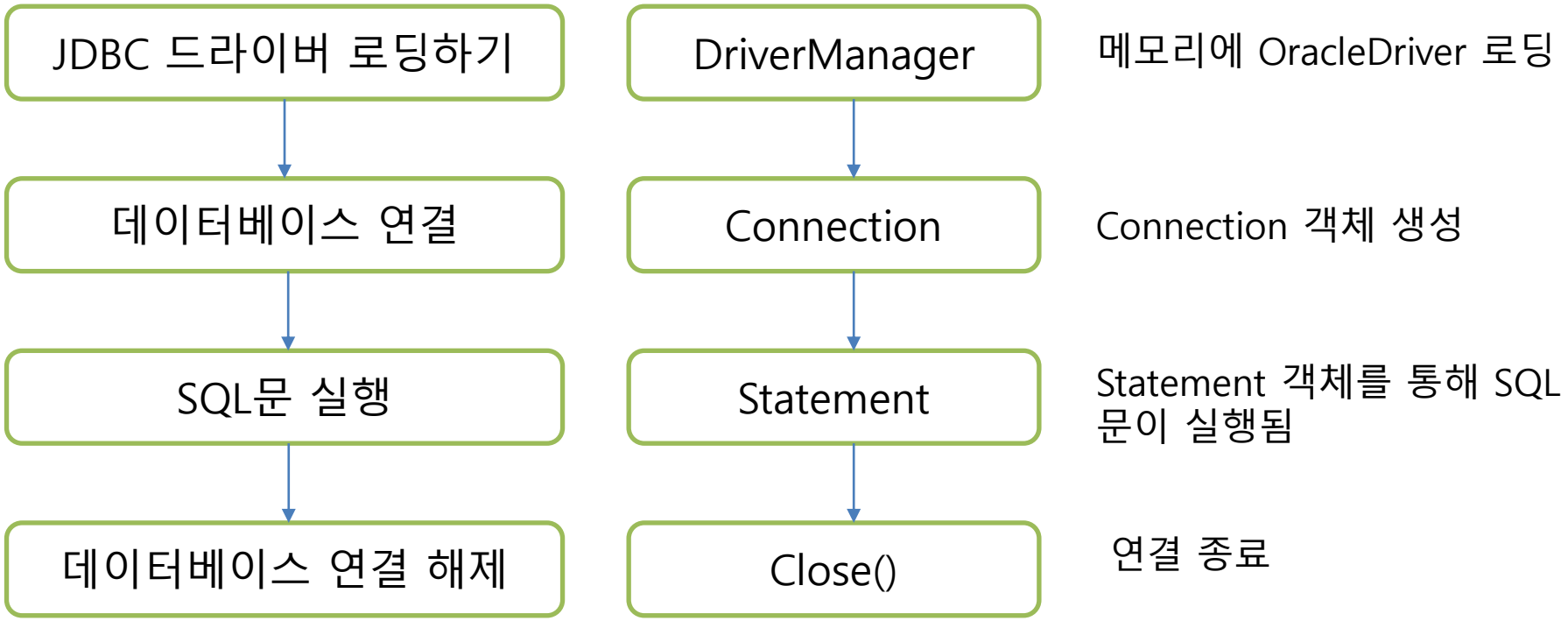
- **사원 삭제**

```
-- 사원 삭제 - 신유빈 사원 퇴사
delete from employee2 where id = 'e101';
```

id	name	salary
e102	이정후	3500000
e103	임시현	NULL
NULL	NULL	NULL

JDBC(Java Database Connectivity)

➢ 데이터베이스 연결 순서



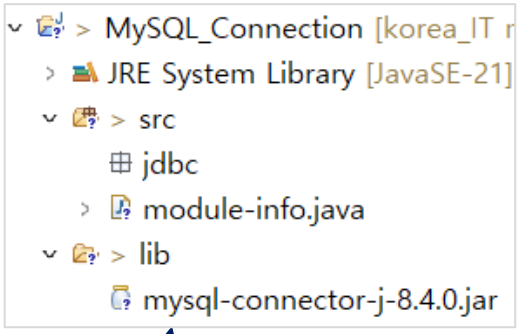
JDBC(Java Database Connectivity)

- mysql connector/j 다운로드

  Maven 저장소 사이트 > mysql driver로 검색

JDBC(Java Database Connectivity)

- mysql connector/j 자바 프로젝트에 포함



1. lib폴더 만들기
2. mysql 드라이버 .jar파일 복사
3. 클래스 패스 설정

jar 파일 > 우측마우스 > Build Path > Add to Build Path

연결 테스트(Connection Test)

- MySQL DBMS에 연결하기

```java
package jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionTest {

    public static void main(String[] args) {

        Connection conn = null; //Connection 인스턴스 생성

        try {
            //mysql driver 클래스 이름
            Class.forName("com.mysql.cj.jdbc.Driver");

            //접속
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/javadb", //db의 url
                    "javauser",  //username 계정
                    "pwjava"     //password
                    );
```

연결 테스트(Connection Test)

- MySQL DBMS에 연결하기

```
23
24              System.out.println(conn + "DB 접속 성공!!");
25          } catch (ClassNotFoundException e) {
26              e.printStackTrace();
27          } catch (SQLException e) {
28              e.printStackTrace();
29          }
30      }
31  }
```

Problems Javadoc Declaration Console ×

<terminated> ConnectionTest (3) [Java Application] C:₩Program Files₩Java₩jdk-21₩bin₩javaw.exe (2025. 8. 5. 오후 11:48:36 – 9

```
com.mysql.cj.jdbc.ConnectionImpl@7c9d8e2DB 접속 성공!!
```

연결 테스트(Connection Test)

- 개선된 연결 테스트

```
public class DBConnectionTest {
    //static{} - 정적 초기화 블럭
    static {
        try {
            //클래스 로딩 시 드라이버 등록
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    static String url = "jdbc:mysql://localhost:3306/javadb"; //DB
    static String username = "javauser"; //사용자 계정
    static String password = "pwjava";   //비밀번호

    public static void main(String[] args) {
        //try ~ with ~ resource문
        try(Connection conn = DriverManager.getConnection(url, username, password)){
            System.out.println(conn + "DB 접속 성공!!");
        }catch(SQLException e) {
            e.printStackTrace();
        }
    }
}
```

SQL – DDL, DML

- **테이블 생성 및 데이터 삽입**

```sql
create table users(
    userid       varchar(20) primary key,   -- 아이디
    userpassword varchar(100) not null,     -- 비밀번호
    username     varchar(30) not null,      -- 이름
    userage      int                        -- 나이
);

-- 회원 추가
insert into users(userid, userpassword, username, userage)
values ('hangang', 'h12345', '한강', 34);

select * from users;
```

	userid	userpassword	username	userage
▶	hangang	h12345	한강	34
	NULL	NULL	NULL	NULL

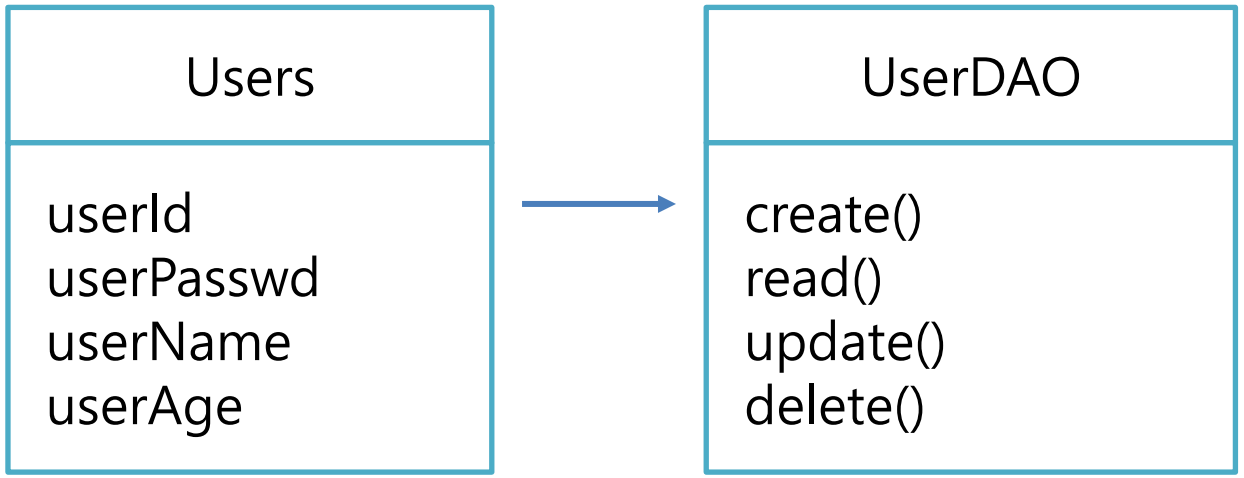
DTO 정의와 사용법

- **DTO(Data Transfer Object)의 정의와 사용법**
  - 여러 타입의 데이터를 다른 클래스로 전달할 때 사용하는 클래스이다.
  - 로직이나 서비스 기능등은 없고 단순히 참조되는 자료형이다.
  - 만드는 방법

    - DB 테이블의 필드명을 속성으로 선언한다.
    - 생성자를 구현한다.
    - 각 속성에 대한 getter/setter 메서드를 구현한다.

DAO 정의와 사용법

- **DAO(Data Access Object)의 정의와 사용법**
  - 데이터베이스에 연결(접속)하고, 데이터를 생성, 조회, 수정, 삭제 등의 작업을 하는 클래스이다.
  - DTO 클래스를 사용하여 작업을 수행한다.



DTO 정의와 사용법

- DTO(Data Transfer Object)의 정의

```java
package dto;
//DTO 정의
public class Users {
    private String userId;          //아이디
    private String userPassword;    //비밀번호
    private String userName;        //이름
    private int userAge;            //나이

    //getter, setter 메서드 정의
    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }
```

DTO 정의와 사용법

- **DTO(Data Transfer Object)의 정의**

```
    public String getUserPassword() {
        return userPassword;
    }

    public void setUserPassword(String userPassword) {
        this.userPassword = userPassword;
    }

    public int getUserAge() {
        return userAge;
    }

    public void setUserAge(int userAge) {
        this.userAge = userAge;
    }

    //객체 정보를 문자열로 리턴
    @Override
    public String toString() {
        return "Users [userId=" + userId + ", userPassword=" + userPassword
                + ", userName=" + userName + ", userAge=" + userAge + "]";
    }
}
```

DAO – DB 연결

- DB 연결

```java
package users.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class UserDAO {
    //static{} - 정적 초기화 블럭
    static {
        try {
            //클래스 로딩 시 드라이버 등록
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    static String url = "jdbc:mysql://localhost:3306/javadb";
    static String username = "javauser";
    static String password = "pwjava";
```

DAO – 회원 생성

- **회원 생성**

```
//회원 등록
public void insertUsers(Users user) {
    //SQL - DML(삽입 구문)
    String sql = "insert into users(userid, userpassword, username, userage) "
            + "values (?, ?, ?, ?)";

    //conn(연결 객체), pstmt(sql 처리 객체)
    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql)){

        //1번 부터 시작 순서대로 바인딩.
        pstmt.setString(1, user.getUserId());
        pstmt.setString(2, user.getUserPassword());
        pstmt.setString(3, user.getUserName());
        pstmt.setInt(4, user.getUserAge());

        pstmt.executeUpdate();  //sql 실행(커밋)
    }catch(SQLException e) {
        e.printStackTrace();
    }
}
```

DAO – 회원 생성

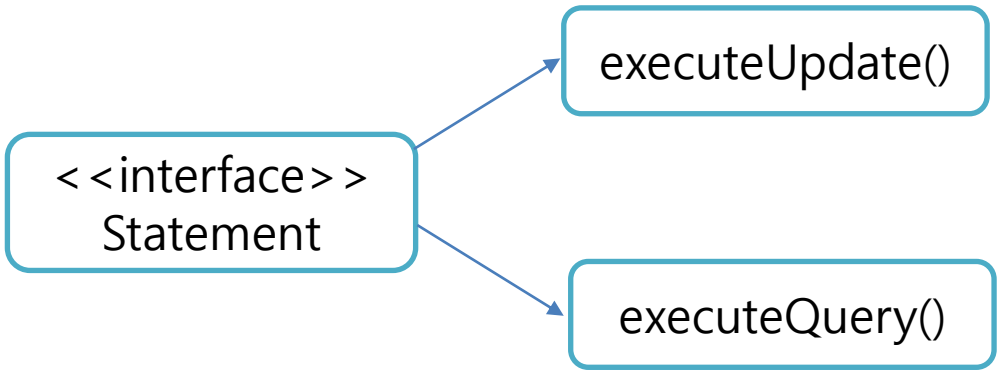
- 회원 생성 테스트 및 DB 확인

```java
public class UserMain {

    public static void main(String[] args) {
        // User 인스턴스 생성
        Users newUser = new Users();
        UserDAO dao = new UserDAO();

        //회원 1명 등록
        newUser.setUserId("today");
        newUser.setUserPassword("t1234");
        newUser.setUserName("이오늘");
        newUser.setUserAge(26);

        dao.insertUsers(newUser); //메서드 호출
        System.out.println("회원 등록 완료!");
```

userid	userpassword	username	userage
hangang	h12345	한강	34
today	t1234	이오늘	26
NULL	NULL	NULL	NULL

JDBC(Java Database Connectivity)

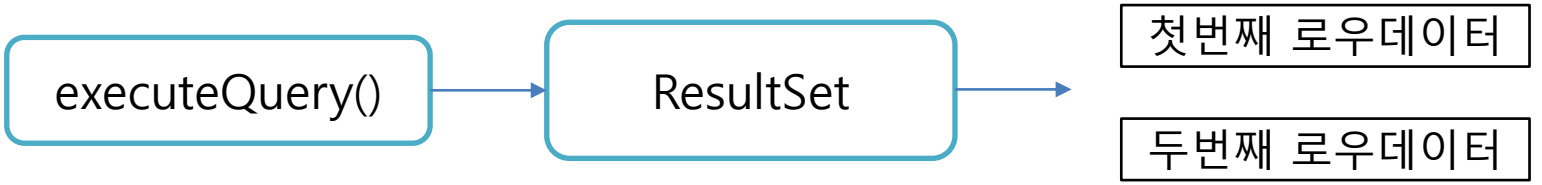
➢ PreparedStatement 객체



SQL문 실행 후 테이블의 내용만 변경되는 경우 사용. 예)insert, delete, update

SQL문 실행 후 여러 개의 결과값 생기는 경우 사용. 예)select

executeQuery() 실행 후 반환 되는 레코드셋



DAO – 회원 목록 보기

- 회원 목록 보기

```
public List<Users> getUserList() {
    //SQL - DML(검색 구문)
    String sql = "SELECT * FROM users";
    //검색된 user를 저장할 리스트 객체 생성
    List<Users> userList = new ArrayList<>();

    //ResultSet rs 객체(db에서 가져온 자료)
    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql);
        ResultSet rs = pstmt.executeQuery()){

        while(rs.next()) { //검색된 자료가 있는 동안 계속
            Users user = new Users();
            user.setUserId(rs.getString("userid"));  //db에서 가져온 아이디를 보기를 위해 설정
            user.setUserPassword(rs.getString("userpassword"));  //비밀번호(문자형)
            user.setUserName(rs.getString("username"));          //이름(문자형)
            user.setUserAge(rs.getInt("userage"));               //나이(정수형)

            userList.add(user); //user 객체를 리스트에 저장
        }
    }catch(SQLException e) {
        e.printStackTrace();
    }
    return userList;  //리스트를 반환함
}
```

DAO – 회원 생성

- 회원 목록 보기 테스트

```
//회원 목록 출력
List<Users> userList = dao.getUserList();
for(int i = 0; i < userList.size(); i++) {
    Users user = userList.get(i);
    System.out.println(user);
}
```

```
회원 등록 완료!
Users [userId=cloud, userPassword=c1234, userName=임시현, userAge=22]
Users [userId=hangang, userPassword=h12345, userName=한강, userAge=34]
Users [userId=today, userPassword=t1234, userName=이오늘, userAge=26]
```

userid	userpassword	username	userage
cloud	c1234	임시현	22
hangang	h12345	한강	34
today	t1234	이오늘	26
NULL	NULL	NULL	NULL

DAO – 회원 상세 보기

- 회원 상세 보기(1건 검색)

```
public Users getUser(String userId) {
    String sql = "select * from users where userid = ?";
    Users user = new Users();

    try(Connection conn = DriverManager.getConnection(url, username, password);
            PreparedStatement pstmt = conn.prepareStatement(sql)){
        pstmt.setString(1, userId); //실행 화면에서 입력된 아이디 바인딩

        try(ResultSet rs = pstmt.executeQuery()){ //db에서 자료 검색
            if(rs.next()) {
                user.setUserId(rs.getString("userid")); //db의 userid 칼럼 가져옴
                user.setUserPassword(rs.getString("userpassword"));
                user.setUserName(rs.getString("username"));
                user.setUserAge(rs.getInt("userage"));
            }
        }
    }catch(SQLException e) {
        e.printStackTrace();
    }
    return user; //회원(user) 반환
}
```

DAO – 회원 생성

- 회원 상세 보기 테스트

```
//회원 1명 검색(상세 보기)
Users findUser = dao.getUser("today");
System.out.println(findUser);
```

```
Users [userId=today, userPassword=t1234, userName=이오늘, userAge=26]
```

DAO – 회원 수정

- 회원 수정

```
public void updateUser(Users user) {
    String sql = "update users set userpassword = ?, username = ?, "
            + "userage = ? where userid = ?";

    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql)){

        pstmt.setString(1, user.getUserPassword());
        pstmt.setString(2, user.getUserName());
        pstmt.setInt(3, user.getUserAge());
        pstmt.setString(4, user.getUserId());

        pstmt.executeUpdate();
    }catch(SQLException e) {
        e.printStackTrace();
    }
}
```

DAO – 회원 관리 테스트

- 회원 수정, 테스트

```
//회원 수정
Users renewUser = new Users();
renewUser.setUserId("today");
renewUser.setUserPassword("t1234");
renewUser.setUserName("투데이");
renewUser.setUserAge(30);

dao.updateUser(renewUser);
```

```
Users [userId=cloud, userPassword=c1234, userName=임시현, userAge=22]
Users [userId=hangang, userPassword=h12345, userName=한강, userAge=34]
Users [userId=today, userPassword=t1234, userName=투데이, userAge=30]
```

DAO – 회원 삭제

- 회원 삭제

```
//회원 삭제
public void deleteUser(String userId) {
    String sql = "delete from users where userid = ?";

    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql)){

        pstmt.setString(1, userId);  //입력된 아이디 바인딩

        pstmt.executeUpdate();
    }catch(SQLException e) {
        e.printStackTrace();
    }
}
```

DAO – 회원 삭제

- **회원 삭제**

```
//회원 삭제
dao.deleteUser("hangang");
```

```
Users [userId=cloud, userPassword=c1234, userName=임시현, userAge=22]
Users [userId=today, userPassword=t1234, userName=투데이, userAge=30]
```