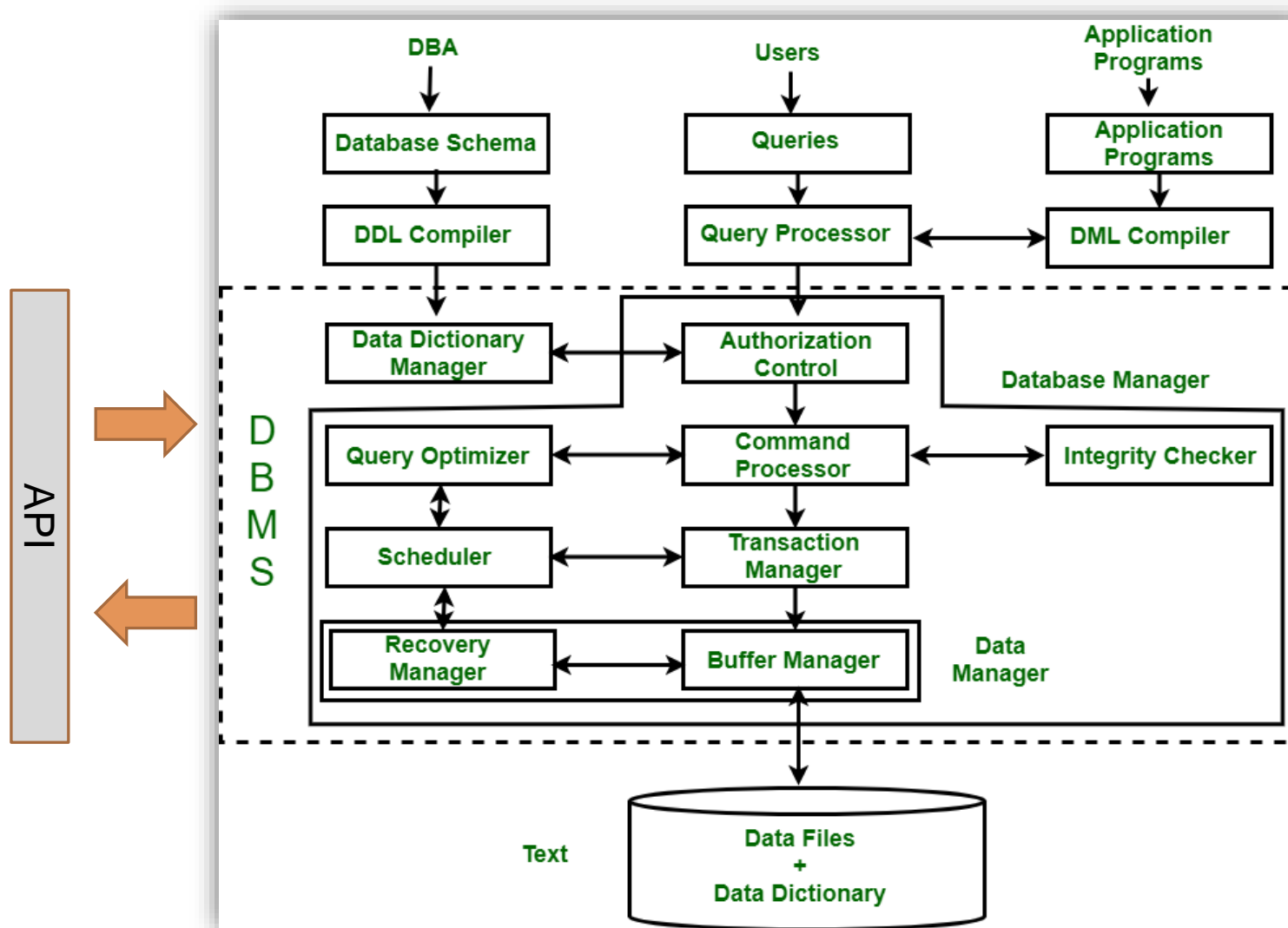


DB SQL 프로그래밍

지능물류빅데이터연구소 이상현

DBMS



이미지 출처 : <https://www.geeksforgeeks.org/structure-of-database-management-system/>

DBMS API

Oracle (Pro*C)

```
int main() {
    /* Connect to the Oracle database */
    EXEC SQL CONNECT :username IDENTIFIED BY :password;

    /* Open the cursor */
    EXEC SQL OPEN emp_cursor;

    /* Declare a variable of the struct type to hold fetched data */
    struct Employee emp;

    /* Fetch data from the cursor */
    for (;;) {
        EXEC SQL FETCH emp_cursor INTO :emp;

        if (sqlca.sqlcode != 0)
            break; /* No more rows to fetch */

        /* Process the fetched data */
        printf("Employee ID: %d, Name: %s, Salary: %.2f\n",
            emp.emp_id, emp.emp_name, emp.salary);
    }

    /* Close the cursor */
    EXEC SQL CLOSE emp_cursor;

    /* Disconnect from the database */
    EXEC SQL COMMIT WORK RELEASE;

    return 0;
}
```

MySQL

```
int main() {
    MYSQL *con = mysql_init(NULL);
    /* Connect to the MySQL database */
    mysql_real_connect(con, "localhost", "username", "password", "dbname", 0, NULL, 0);

    mysql_query(conn, "SELECT emp_id, emp_name, salary FROM employee");

    MYSQL_RES *result = mysql_store_result(conn);

    struct Employee emp;

    while (1) {
        MYSQL_ROW row = mysql_fetch_row(result);

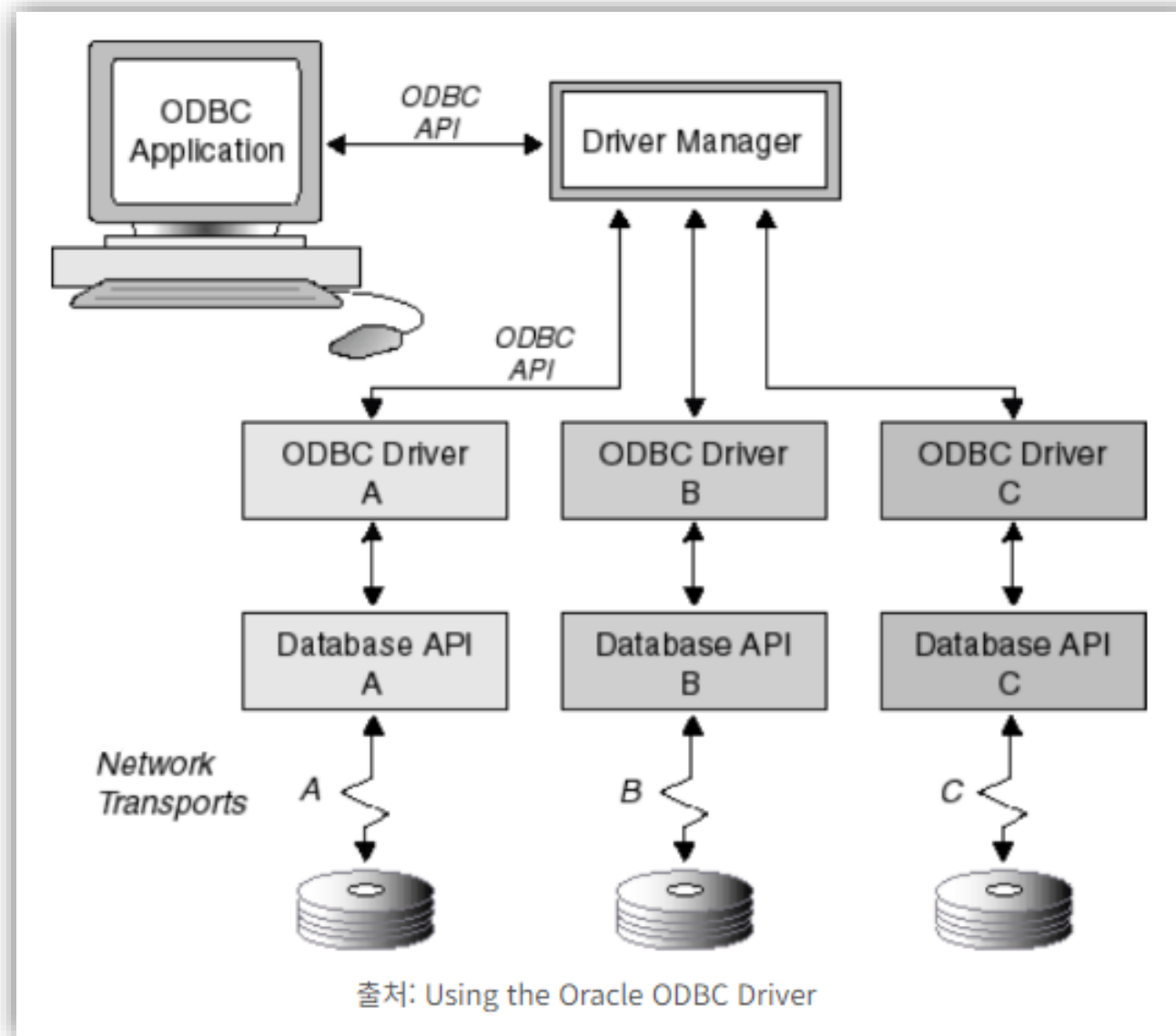
        if (row == NULL)
            break; /* No more rows to fetch */

        /* Process the fetched data */
        sscanf(row[0], "%d", &emp.emp_id);
        sscanf(row[1], "%s", emp.emp_name);
        sscanf(row[2], "%f", &emp.salary);

        printf("Employee ID: %d, Name: %s, Salary: %.2f\n",
            emp.emp_id, emp.emp_name, emp.salary);
    }
    mysql_free_result(result);
    /* Disconnect from the database */
    mysql_close(conn);

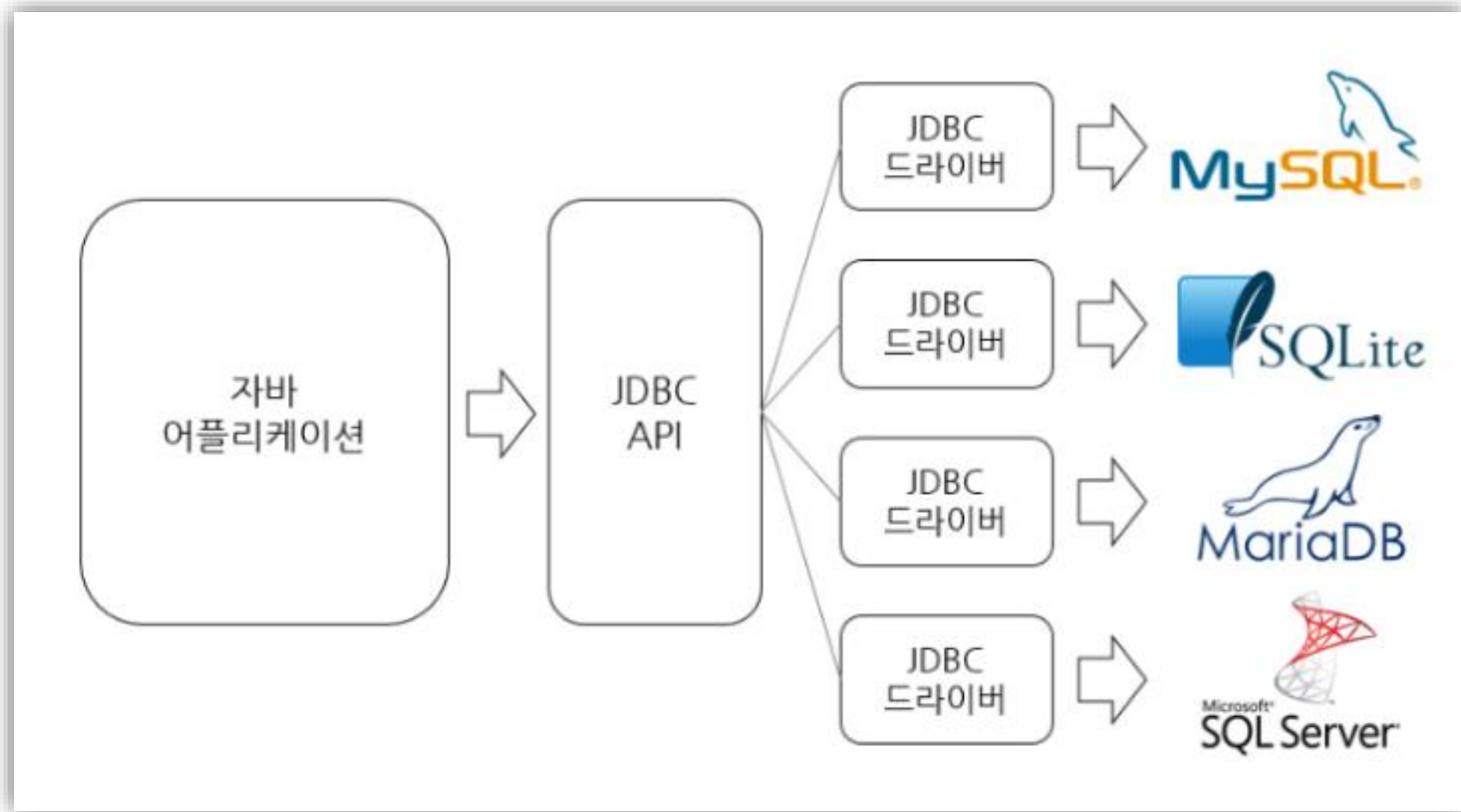
    return 0;
}
```

ODBC – Open Database Connectivity



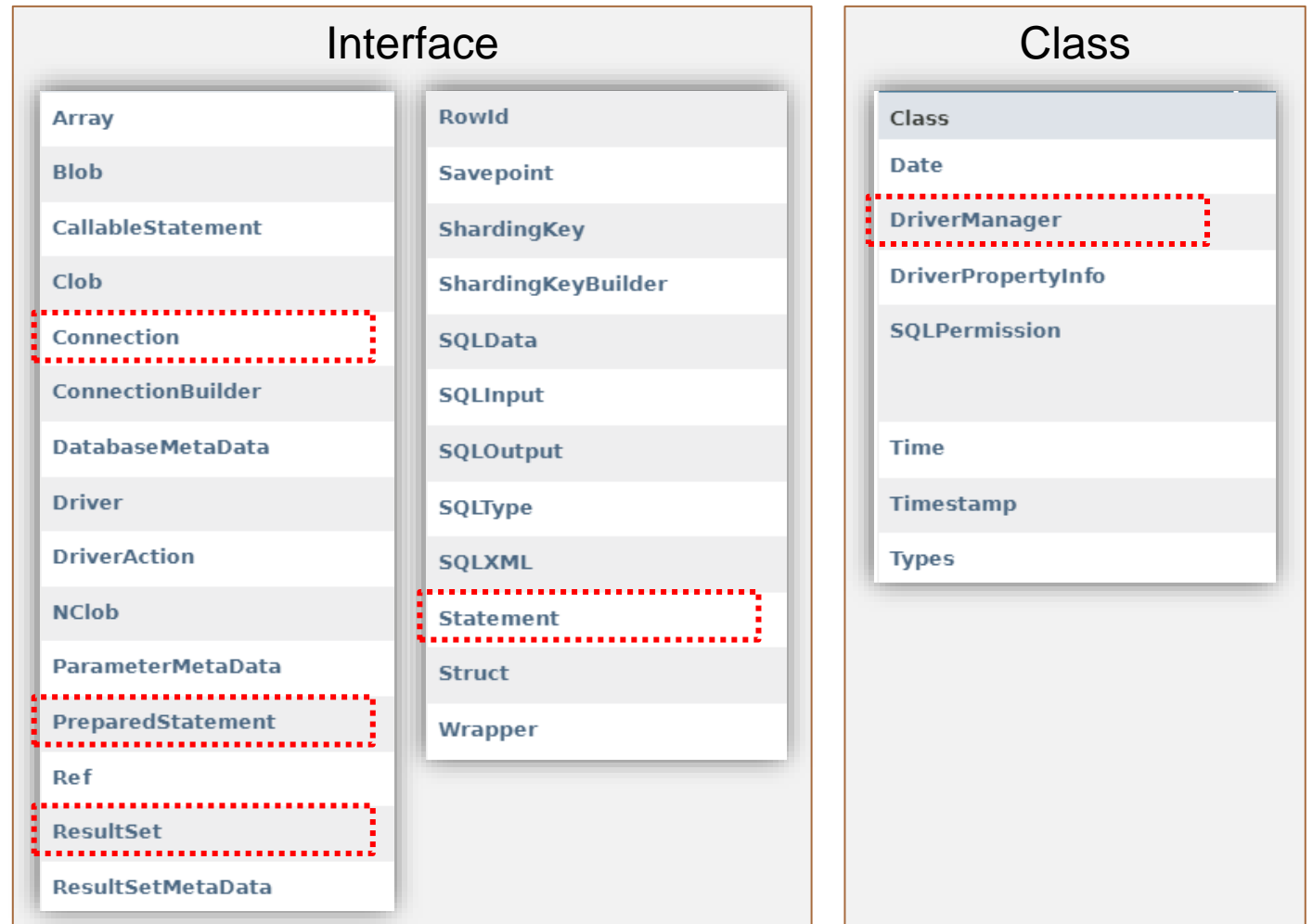
JDBC – Java Database Connectivity

- 다양한 종류의 관계형 데이터베이스 이용에 사용되는 표준 Java API



JDBC Interface

- Module java.sql
- Package java.sql
- DB 기능 처리 객체
- 주로 interface들 사용



JDBC Driver

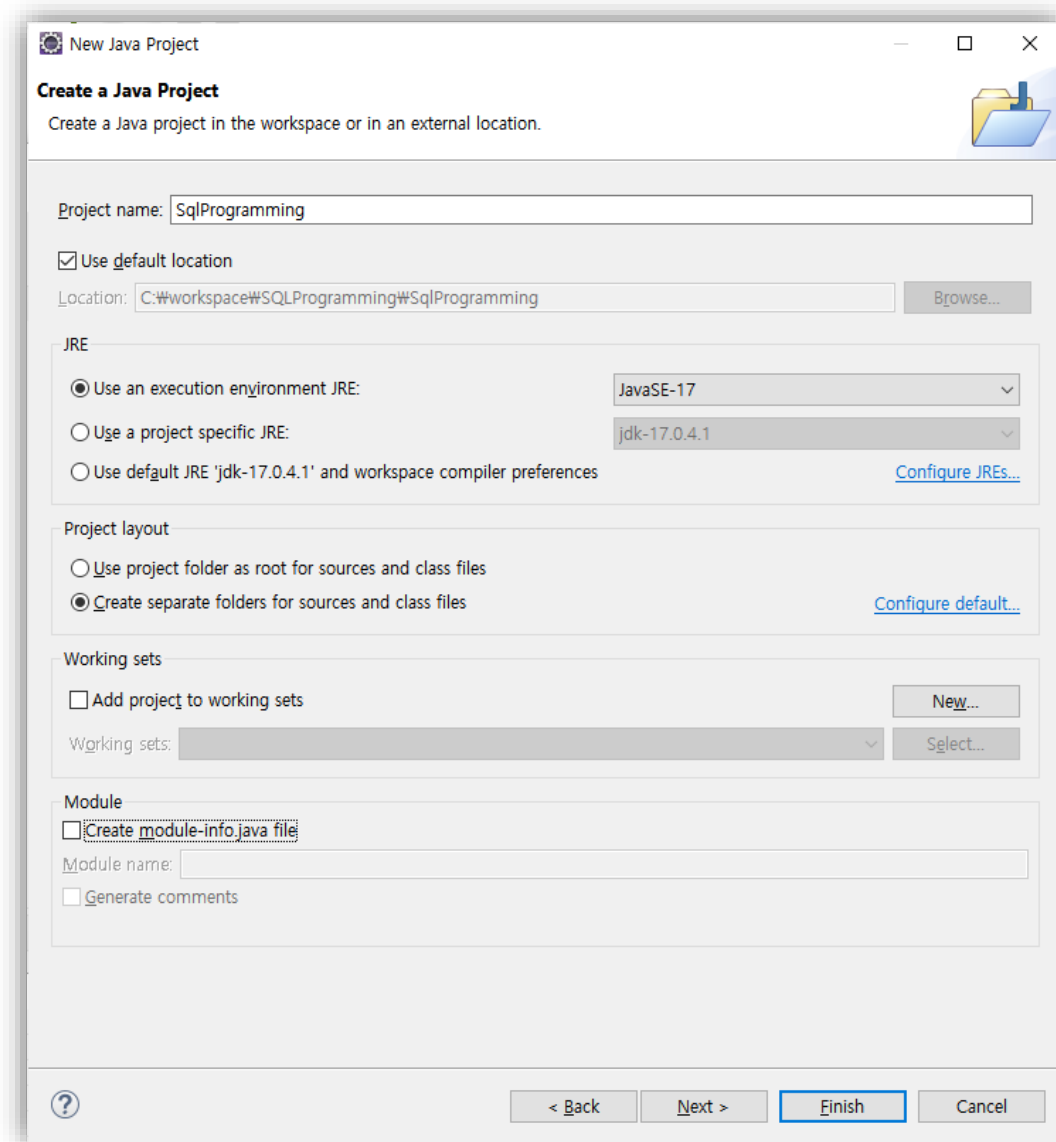
- java.sql의 인터페이스들을 상속하여 구현한 파일
- 데이터베이스 제작사에서 제공
- 데이터베이스별 드라이버
 - MySQL인 경우 : **mysql-connector-java-8.0.30.jar**
 - c:\Program Files (x86)\MySQL\Connector J 8.0
 - H2인 경우 : **h2-2.1.214.jar**
 - c:\Program Files (x86)\H2\bin
 - Oracle인 경우 : **ojdbc6.jar**

JDBC SQL Programming 구현 순서

1. 프로젝트 생성 및 드라이버 설정
2. JDBC 드라이버 로딩하기 (`Class.forName()`)
3. DBMS 서버와 접속하기 (`DriverManager.getConnection()`)
4. SQL 문 실행하기 – `Statement`
5. SQL 문 실행하기 – `PreparedStatement`
6. 자원 해제하기

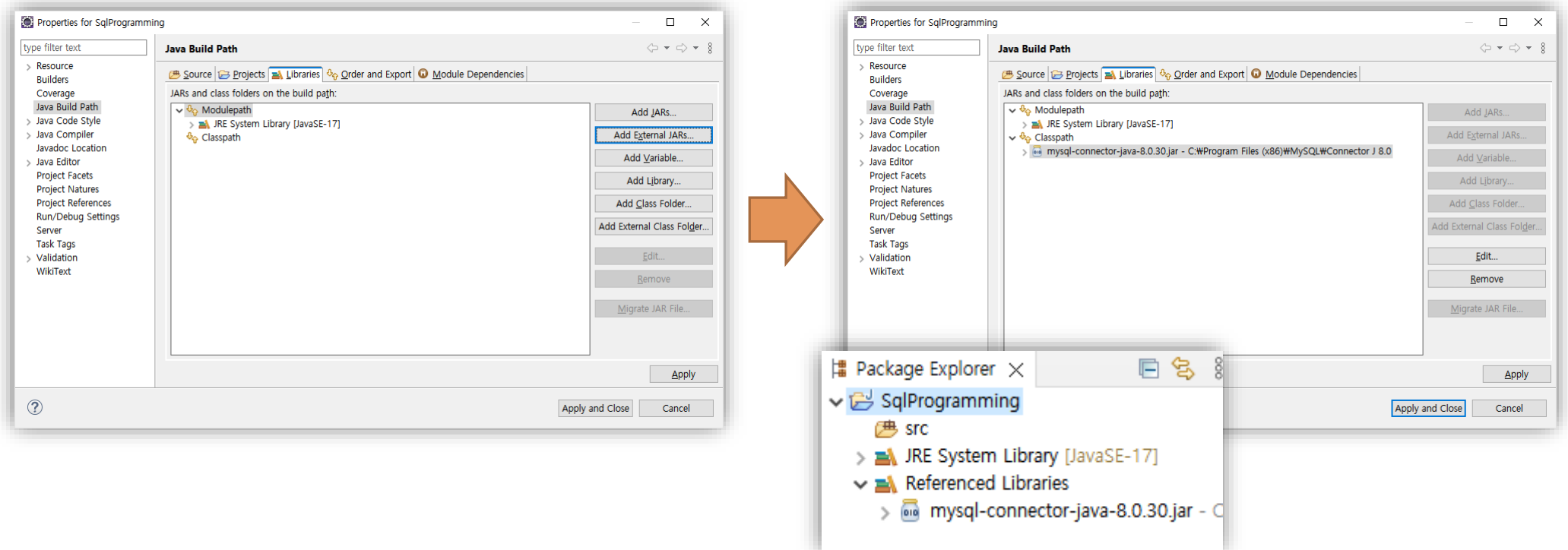
1. 프로젝트 생성 및 드라이버 설정

- 프로젝트 생성
 - 프로젝트 명 : SqlProgramming
 - Create module-info.java file
 - 체크 해제



1. 프로젝트 생성 및 드라이버 설정

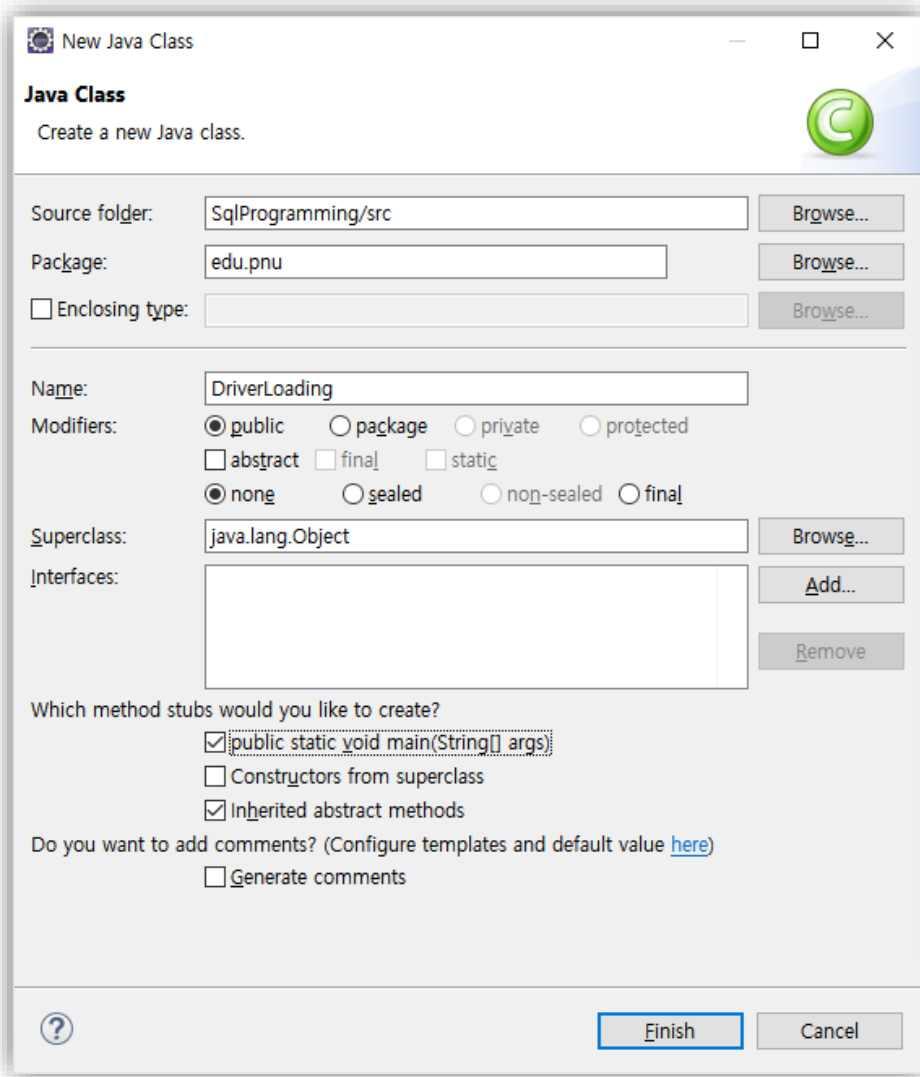
- 드라이버 설정 (MySQL)
 - 드라이버 설정 방법
 - [Build Path] – [Configure Build Path...]



2. 드라이버 로딩

- `java.lang.Class` 의 `forName()` 메서드 사용
 - `static Class<?> forName(string className)`
- `forName()` 메서드의 인자 값으로 **Driver**를 상속하는 클래스이름을 지정
- 사용 예
 - MySQL인 경우 : `Class.forName("com.mysql.cj.jdbc.Driver");`
 - H2인 경우 : `Class.forName("org.h2.Driver");`
 - Oracle인 경우 : `Class.forName("oracle.jdbc.driver.OracleDriver");`

2. 드라이버 로딩



```
1 package edu.pnu;
2
3 public class DriverLoading {
4
5     public static void main(String[] args) {
6
7         try {
8             String driver = "com.mysql.cj.jdbc.Driver";
9
10            Class.forName(driver);
11
12            System.out.println("로딩 성공");
13
14        } catch (Exception e) {
15            System.out.println("로딩 실패 : " + e.getMessage());
16        }
17    }
18 }
```

3. 데이터베이스 서버 접속

- java.sql.**DriverManager**를 이용해서 연결 객체 생성

- static Connection getConnection(String **url**, String **user**, String **password**)
- url → 접속할 서버의 URL 지정
- user → 로그인할 계정
- password → 로그인할 비밀번호

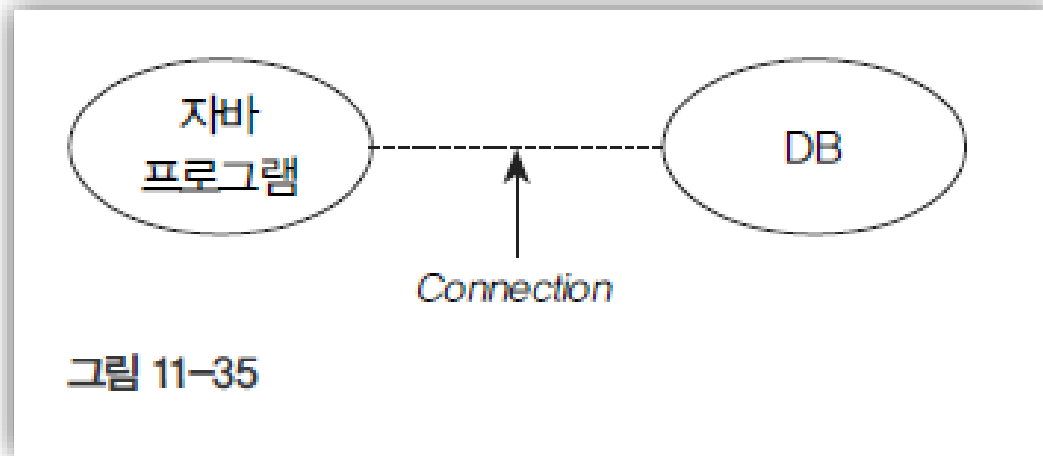
- 사용 예

- Conneciton con =

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/DBNAME", "scott", "tiger")
```

3. 데이터베이스 서버 접속

- Connection
 - getConnection()의 리턴 값
 - DB서버와의 연결 상태 객체



3. 데이터베이스 서버 접속

```
DBConnect.java X
1 package edu.pnu;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5
6 public class DBConnect {
7
8     public static void main(String[] args) {
9
10         try {
11             String driver = "com.mysql.cj.jdbc.Driver";
12             String url = "jdbc:mysql://localhost:3306/DBNAME";
13             String username = "scott";
14             String password = "tiger";
15
16             Class.forName(driver);
17             Connection con = DriverManager.getConnection(url, username, password);
18
19             System.out.println("연결 성공");
20             con.close();
21
22         } catch (Exception e) {
23             System.out.println("연결 실패 : " + e.getMessage());
24         }
25     }
26 }
```

4. SQL문 실행 – Statement

- Statement

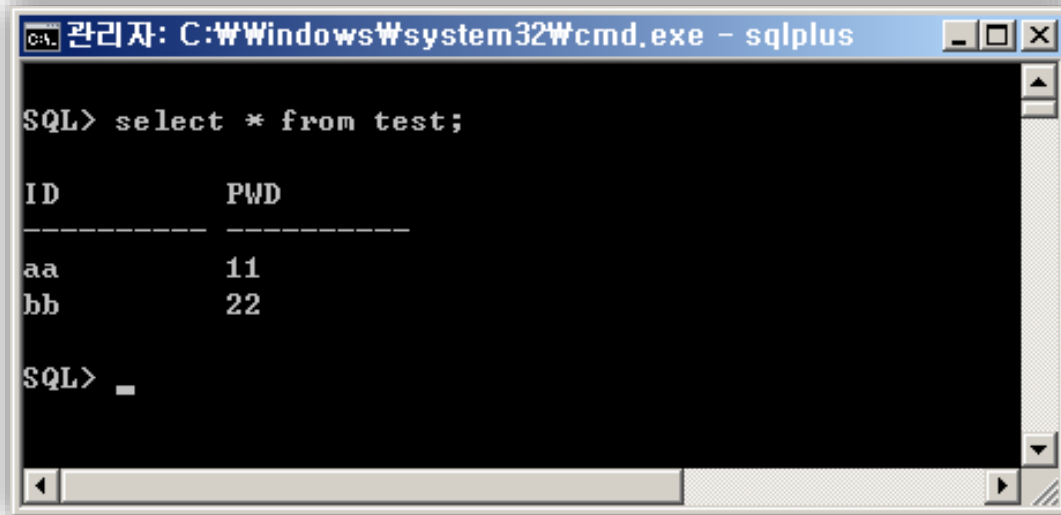
- Connection상에서 SQL문을 처리하는 객체
- Connection의 `createStatement()` 메서드를 이용해서 객체 생성
 - `Statement stmt = conn.createStatement();`

- Statement의 메서드 사용

- `ResultSet executeQuery(String sql) - select`
- `int executeUpdate(String sql) - select를 제외한 나머지 (insert, update, delete)`

4. SQL문 실행 – Statement

executeQuery



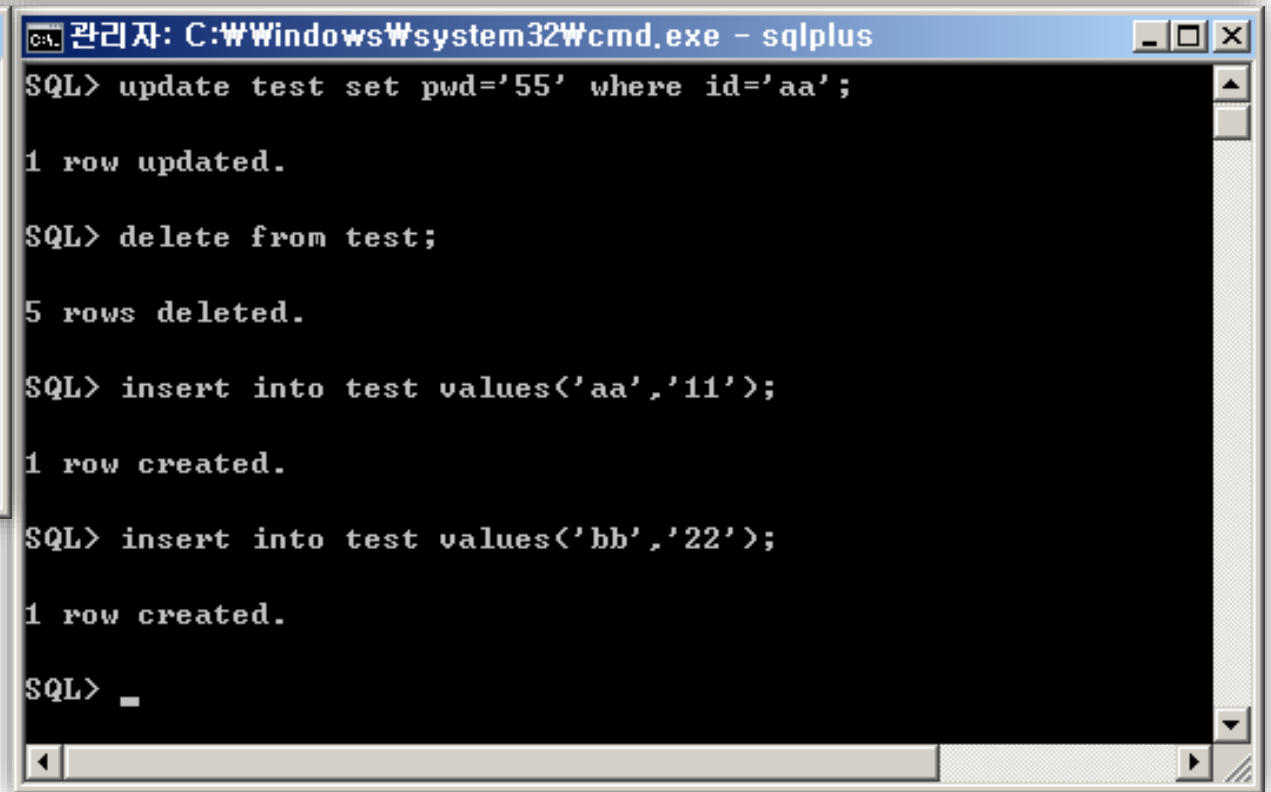
A screenshot of a Windows command prompt window titled "관리자: C:\Windows\system32\cmd.exe - sqlplus". The prompt is "SQL>". The user has entered the command "select * from test;". The output is a table with two columns, "ID" and "PWD", separated by a dashed line. The data rows are "aa" with "11" and "bb" with "22". The prompt "SQL>" is followed by a cursor.

```
SQL> select * from test;
```

ID	PWD
aa	11
bb	22

```
SQL> _
```

executeUpdate



A screenshot of a Windows command prompt window titled "관리자: C:\Windows\system32\cmd.exe - sqlplus". The prompt is "SQL>". The user has entered the command "update test set pwd='55' where id='aa;". The output is "1 row updated.". The user has entered the command "delete from test;". The output is "5 rows deleted.". The user has entered the command "insert into test values('aa','11');". The output is "1 row created.". The user has entered the command "insert into test values('bb','22');". The output is "1 row created.". The prompt "SQL>" is followed by a cursor.

```
SQL> update test set pwd='55' where id='aa';
```

1 row updated.

```
SQL> delete from test;
```

5 rows deleted.

```
SQL> insert into test values('aa','11');
```

1 row created.

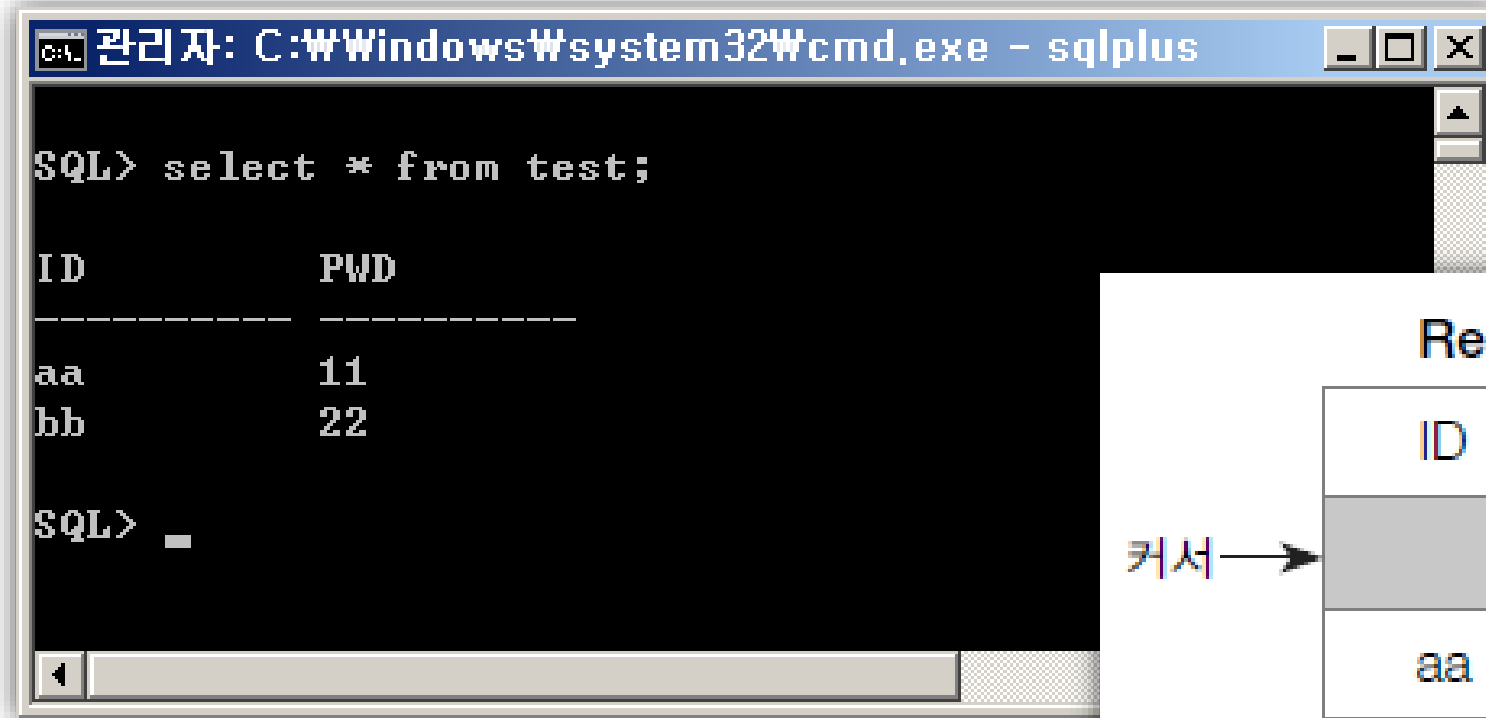
```
SQL> insert into test values('bb','22');
```

1 row created.

```
SQL> _
```

4. SQL문 실행 – Statement

- ResultSet - select문의 질의 결과값을 가지는 객체



```
관리자: C:\Windows\system32\cmd.exe - sqlplus

SQL> select * from test;

ID          PWD
-----
aa          11
bb          22

SQL> _
```

ResultSet rs

ID	PWD
aa	11
bb	22

커서 → 시작 빈행
끝 빈행

4. SQL문 실행 – Statement

```
QueryByStatment.java x
1 package edu.pnu;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.ResultSet;
6 import java.sql.Statement;
7
8 public class QueryByStatment {
9
10     public static void main(String[] args) {
11
12         Connection con = null;
13         try {
14             String driver = "com.mysql.cj.jdbc.Driver";
15             String url = "jdbc:mysql://localhost:3306/world";
16             String username = "scott";
17             String password = "tiger";
18
19             Class.forName(driver);
20             con = DriverManager.getConnection(url, username, password);
21
22             Statement st = con.createStatement();
23             ResultSet rs = st.executeQuery("select id,name,countrycode,district,population from city limit 10");
24
25             while(rs.next() ) {
26                 System.out.print(rs.getString("id")+","");
27                 System.out.print(rs.getString("name")+","");
28                 System.out.print(rs.getString("countrycode")+","");
29                 System.out.print(rs.getString("district")+","");
30                 System.out.print(rs.getString("population")+"\n");
31             }
32             rs.close();
33             st.close();
34             con.close();
35         } catch (Exception e) {
36             System.out.println("연결 실패 : " + e.getMessage());
37         }
38     }
39 }
```

4. SQL문 실행 – Statement

- ResultSet Methods

- **void afterLast()**

- 끝 빈 행으로 커서를 이동함

- **void beforeFirst()**

- 시작 빈 행으로 커서를 이동함

- **boolean next()**

- 현재 커서 다음의 레코드 유무를 판단함. **true**인 경우 커서를 다음으로 이동시킴

- **XXX getXXX(String column명) /getXXX(int index)**

- 커서가 위치한 레코드의 컬럼 값을 반환함 (XXX 는 데이터 타입)

<https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/ResultSet.html#method-summary>

Statement 실습

1. **world** 데이터베이스에는 테이블이 3개가 있다.
 - 각각의 테이블에 저장된 데이터를 모두 읽어서 화면에 출력하세요.
 - city
 - country
 - countrylanguage
2. 국가 코드가 '**KOR**'인 도시를 찾아 인구수를 역순으로 표시하세요.
 - 도시명, 인구수

5. SQL문 실행 – PreparedStatement

- PreparedStatement
 - Connection상에서 SQL문을 처리하는 객체
 - Connection의 preparedStatement() 메서드를 이용해서 객체 생성
 - PreparedStatement pstmt = conn.prepareStatement(queryString);
 - 파라미터로 입력되는 queryString 내에 “?” 기호를 사용하여 미완성의 **SQL**문을 생성하여 사용

5. SQL문 실행 – PreparedStatement

- void setXXX(int parameterIndex, XXX value)

Date Type

1부터 시작

/* Statement 사용 */

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

```
stmt.executeUpdate("insert into test values ('"+id+"', '"+pwd+"')");
```

/* PreparedStatement 사용 */

```
PreparedStatement pstmt = conn.preareStatement("insert into test values(?,?)");
```

```
pstmt.setString(1, id);
```

```
pstmt.setString(2, pwd);
```

```
pstmt.executeUpdate();
```

5. SQL문 실행 – PreparedStatement

```
QueryByPreparedStatement.java ×
1 package edu.pnu;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7
8 public class QueryByPreparedStatement {
9
10     public static void main(String[] args) {
11
12         Connection con = null;
13         try {
14             String driver = "com.mysql.cj.jdbc.Driver";
15             String url = "jdbc:mysql://localhost:3306/world";
16             String username = "scott";
17             String password = "tiger";
18
19             Class.forName(driver);
20             con = DriverManager.getConnection(url, username, password);
21
22             PreparedStatement pt = con.prepareStatement("select id,name,countrycode,district,population from city where name=?");
23             pt.setString(1, "Seoul");
24             ResultSet rs = pt.executeQuery();
25
26             while(rs.next() ) {
27                 System.out.print(rs.getInt("id")+","");
28                 System.out.print(rs.getString("name")+","");
29                 System.out.print(rs.getString("countrycode")+","");
30                 System.out.print(rs.getString("district")+","");
31                 System.out.print(rs.getInt("population")+"\n");
32             }
33             rs.close();
34             pt.close();
35             con.close();
36         } catch (Exception e) {
37             System.out.println("연결 실패 : " + e.getMessage());
38         }
39     }
40 }
```


6. 자원 해제

- JDBC 프로그램 실행 시 사용했던 모든 객체를 메모리에서 해제
- Connection 해제
- Statement 또는 PreparedStatement 해제
- ResultSet 해제
- 해제 메서드 : `close()`
 - `conn.close()`
 - `stmt.close()` 또는 `pstmt.close()`
 - `conn.close()`