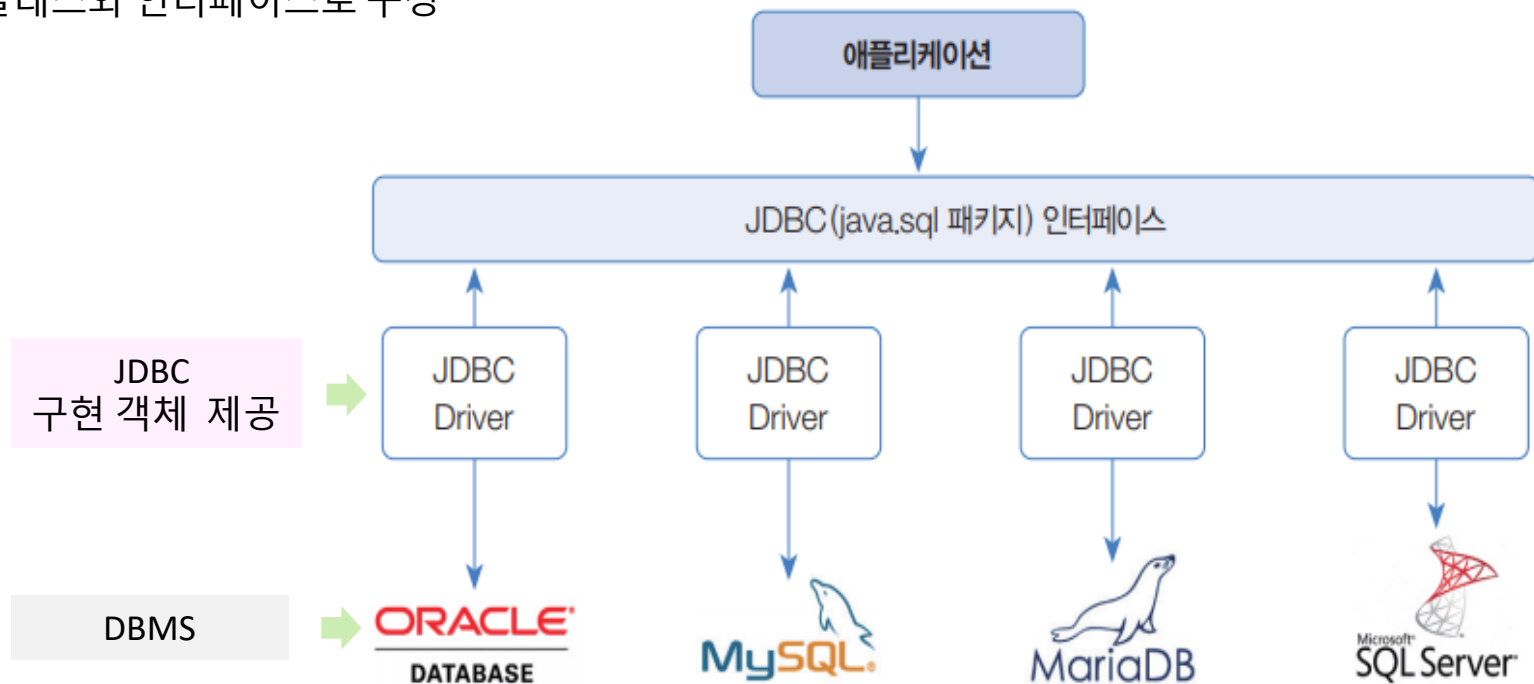


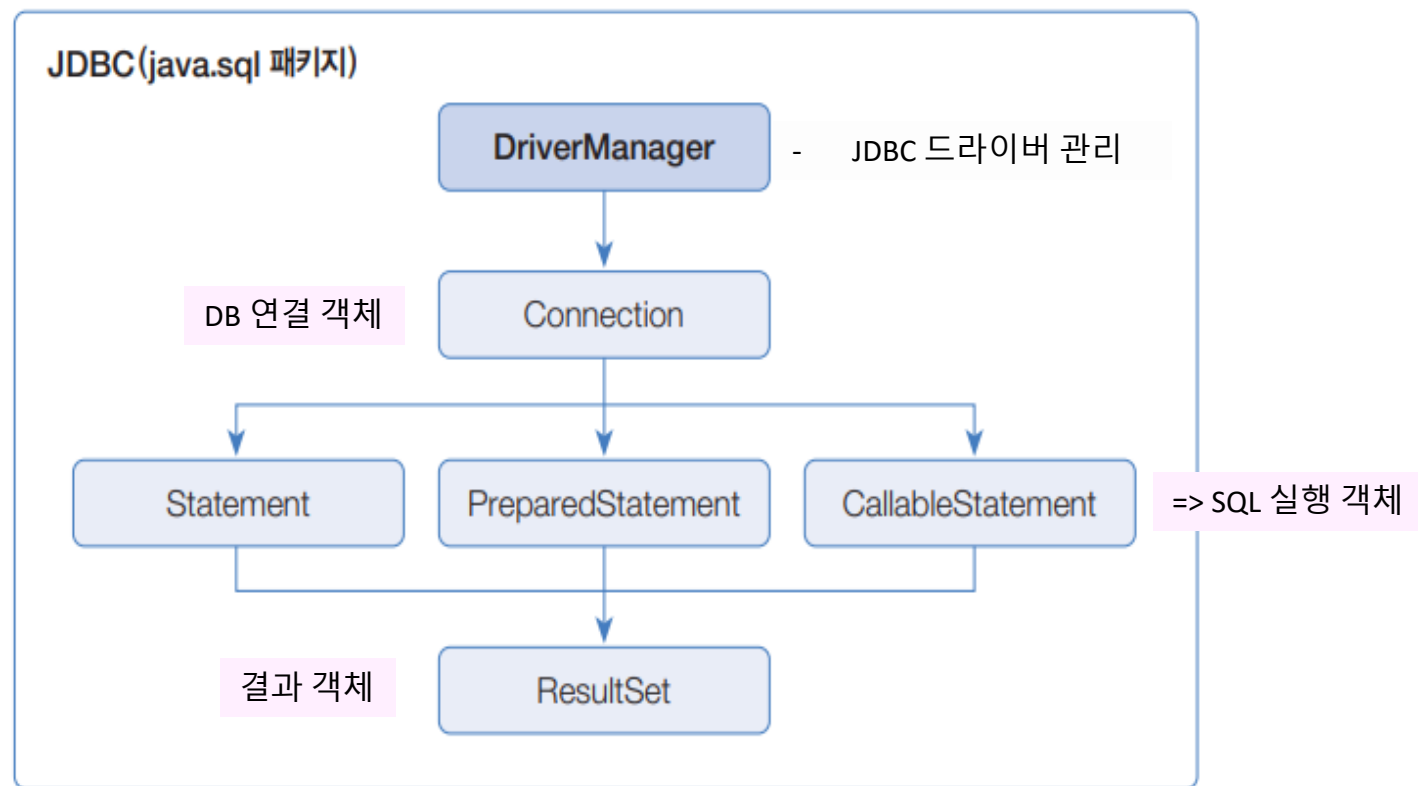
데이터베이스 입출력

JDBC(java Database Connectivity)

JDBC 라이브러리

- 자바는 데이터베이스(DB)와 연결해서 데이터 입출력 작업을 할 수 있도록 JDBC 라이브러리 (java.sql 패키지)를 제공
- JDBC는 데이터베이스 관리시스템(DBMS)의 종류와 상관없이 동일하게 사용할 수 있는 클래스와 인터페이스로 구성





- Statement : SQL의 DDL, DML 문 실행 시 사용.
- PreparedStatement : SQL의 DDL, DML 문 실행 시 사용. 매개변수화된 SQL 문을 써 편리성과 보안성 유리

SQL Developer

- User 이름 ##없이 작업

```
alter SESSION set "_oracle_script" = true;
```

- User 생성

```
create user java IDENTIFIED by 1234;
```

- 권한부여

```
grant connect, resource, dba to java;
```

```
grant unlimited tablespace to java;
```

SQL Developer

새로 만들기/데이터베이스 접속 선택

접속 이름	접속 세부정보
javadb	java@//localh...
kjn	kjn@//localho...
ora	system@//loc...
인사관리	hr@//localhos...
자바	java@//localh...

Name javadb Color

데이터베이스 유형 Oracle

사용자 정보 프록시 사용자

인증 유형 기본값

사용자 이름(U) java 롤(L) 기본값

비밀번호(P) ☐ 비밀번호 저장(Y)

접속 유형(Y) 기본

세부정보 고급

호스트 이름(A) localhost

포트(B) 1521

☒ SID(I) xe

☐ 서비스 이름(E)

상태: 성공

도움말(H) 저장(S) 지우기(C) 테스트(T) 접속(O) 취소

데이터베이스 구성

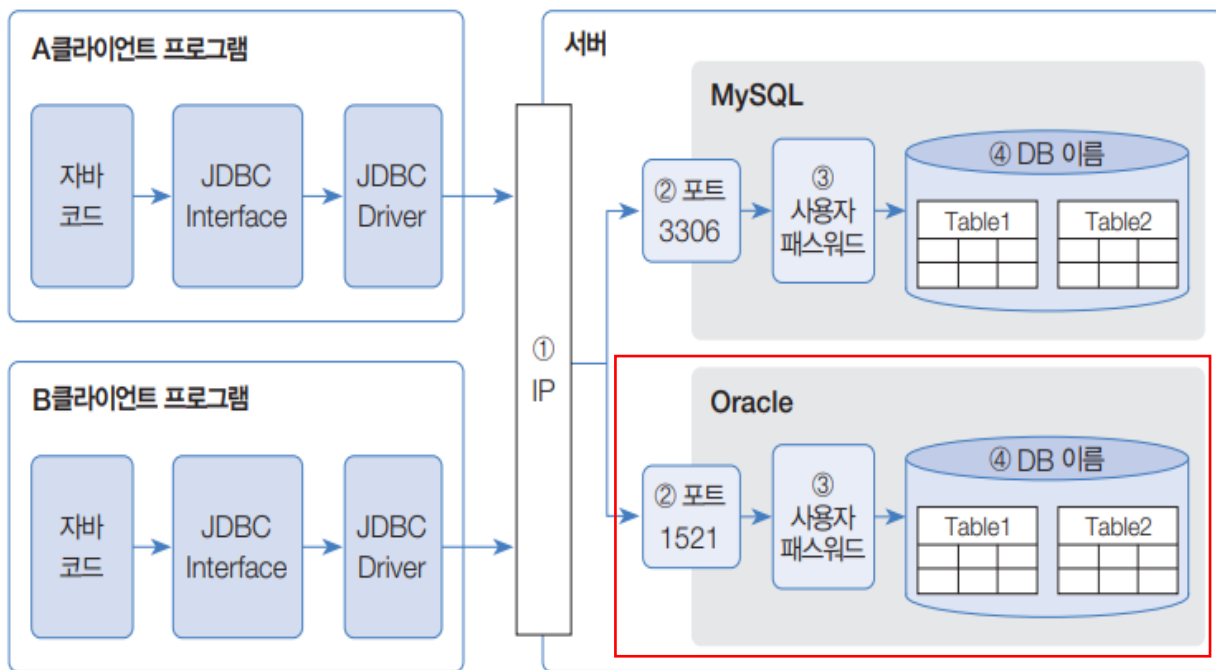
워크시트

질의 작성기

```
1 create table boards( bno          NUMBER          PRIMARY KEY,
2                       btitle       VARCHAR2(100)     not null,
3                       bcontent     CLOB              not null,
4                       bwriter      VARCHAR2(50)      not null,
5                       bdate        date              DEFAULT sysdate
6                       );
7
8
9 CREATE SEQUENCE seq_bno NOCACHE;
10
```

데이터베이스 연결


- 클라이언트 프로그램에서 DB와 연결하려면 해당 DBMS의 JDBC Driver가 필요
- ① DBMS가 설치된 컴퓨터의 IP 주소, ② DBMS가 허용하는 포트(Port) 번호,
- ③ 사용자(DB 계정) 및 비밀번호, ④ 사용하고자 하는 DB 이름 필요




JDBC Driver 설치


- 로컬 PC에 Oracle을 설치하면 JDBC Driver 파일 찾을 수 있음
(C:\app\twony\product\21c\dbhomeXE\jdbc\lib\ojdbc8.jar)
- ojdbc8.jar 파일** 을 프로젝트의 **lib폴더에 복사**
 - lib 폴더가 없으면 프로젝트에서 오른쪽 버튼 [New] – [Folder] 선택해서 생성

- Build Path 추가**

- ▽  Referenced Libraries

- >  ojdbc8.jar

- ▽  lib

-  ojdbc8.jar



오른쪽 버튼 > Build Path > Add to Build Path

DB 연결

- Class.forName() 메소드는 문자열로 주어진 JDBC Driver 클래스를 Build Path에서 찾고, JDBC Driver를 메모리로 로딩

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

- Oracle 연결 문자열



```
public class connExam {
```

```
    public static void main(String[] args) {
```

```
        // JDBC
```

```
        Connection conn = null;
```

```
        try {
```

```
            //jdbc 등록
```

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

```
            //연결
```

```
            conn = DriverManager.getConnection(
```

```
                "jdbc:oracle:thin:@localhost:1521/xe",
```

```
                "java",
```

```
                "1234"
```

```
            );
```

```
            System.out.println("연결 성공");
```

```
        } catch (ClassNotFoundException e) {
```

```
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
```

```
        } catch (SQLException e) {
```

```
            e.printStackTrace();
```

```
package javadb;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```

```
    }finally {  
        if(conn != null) {  
            try {  
                //연결끊기  
                conn.close();  
                System.out.println("연결 끊기");  
            } catch (SQLException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}  
}
```

INSERT 문

//DB 작업

//매개변수화된 SQL문 작성

String **sql** = "" +

"INSERT INTO boards (bno, btitle, bcontent,bwriter,bdate)" +

"values (seq_bno.nextval,?, ?, ?, sysdate)";

//preparedStatement 얻기 및 값 지정

PreparedStatement **pstmt** = **conn.prepareStatement(sql, new String[] {"bno"});**

pstmt.setString(1, "비 오는 날");

pstmt.setString(2, "소나기가 내려요");

pstmt.setString(3, "summer");

INSERT 문

//SQL문 실행

```
int rows = pstmt.executeUpdate();
```

```
System.out.println("저장된 행 수 : " + rows);
```

//bno값 얻기

```
if(rows == 1) {
```

```
    ResultSet rs = pstmt.getGeneratedKeys();
```

```
    if(rs.next()) {
```

```
        int bno = rs.getInt(1);
```

```
        System.out.println("저장된 bno : " + bno);
```

```
    }
```

```
        rs.close();
```

```
    }
```

//PreparedStatement 닫기

```
pstmt.close();
```

UPDATE 문

```
//DB 작업
```

```
//매개변수화된 SQL문 작성
```

```
String sql = "update boards " +
```

```
    "set btitle=?, bcontent=? where bno=? ";
```

```
//preparedStatement 얻기 및 값 지정
```

```
PreparedStatement pstmt = conn.prepareStatement(sql,
```

```
    new String[] {"bno","btitle", "bcontent"});
```

```
pstmt.setString(1, "눈사람");
```

```
pstmt.setString(2, "눈으로 만든 사람");
```

```
pstmt.setInt(3, 1);
```

DELETE 문

```
//DB 작업
//매개변수화된 SQL문 작성
String sql = "delete from boards where bwriter=? ";

//PreparedStatement 얻기 및 값 지정
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "summer");
//SQL문 실행
int rows = pstmt.executeUpdate();
System.out.println("삭제된 행 수 : " + rows);

//bno값 얻기
if(rows == 1) {
    ResultSet rs = pstmt.getGeneratedKeys();
    if(rs.next()) {
        int bno = rs.getInt(1);
        System.out.println("저장된 bno : " + bno);
    }
    rs.close();
}
//PreparedStatement 닫기
pstmt.close();
```

ResultSet 구조

- SELECT 문에 기술된 컬럼으로 구성된 행(row)의 집합

```
SELECT userid, username, userage FROM users
```

- 커서(cursor)가 있는 행의 데이터만 읽을 수 있음
- first 행을 읽으려면 next() 메소드로 커서 이동

```
boolean result = rs.next();
```

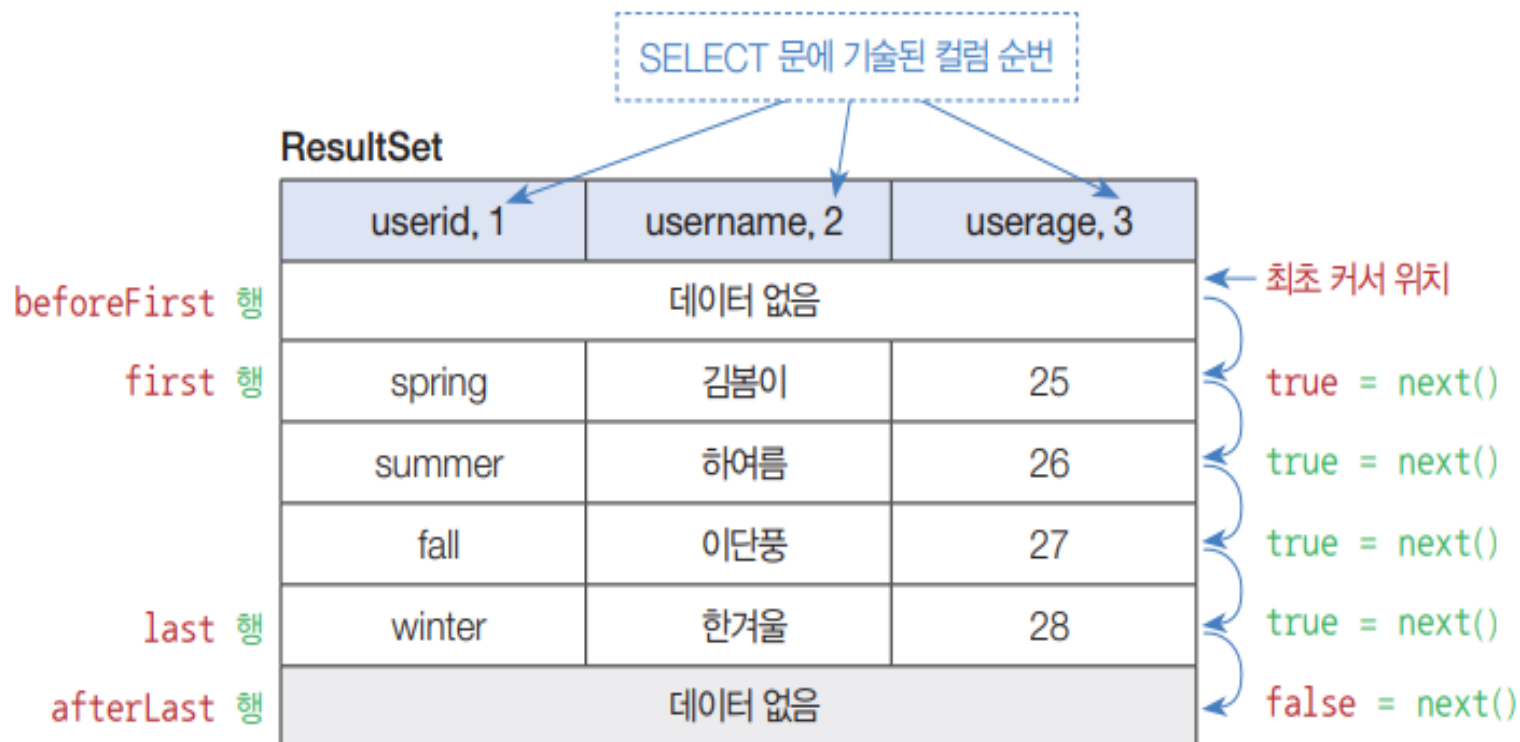
1개의 데이터 행만 가져올 경우

```
ResultSet rs = pstmt.executeQuery();
if(rs.next()) {
    //첫 번째 데이터 행 처리
} else {
    //afterLast 행으로 이동했을 경우
}
```

n개의 데이터 행을 가져올 경우

```
ResultSet rs = pstmt.executeQuery();
while(rs.next()) {
    //last 행까지 이동하면서 데이터 행 처리
}
//afterLast 행으로 이동했을 경우
```


ResultSet 구조



데이터 행 읽기

- 커서가 있는 데이터 행에서 각 컬럼의 값은 Getter 메소드로 읽음
- SELECT 문에 연산식이나 함수 호출이 포함되어 있다면 컬럼 이름 대신에 컬럼 순번으로 읽어야 함

컬럼 이름으로 읽기

```
String userId =  
    rs.getString("userid");  
String userName =  
    rs.getString("username");  
int userAge = rs.getInt("userage");
```

컬럼 순번으로 읽기

```
String userId = rs.getString(1);  
String userName = rs.getString(2);  
int userAge = rs.getInt(3);
```

```
SELECT userid, userage - 1  
FROM users
```



```
String userId =  
    rs.getString("userid");  
int userAge = rs.getInt(2);
```

SELECT 문

//매개 변수화된 SQL문 작성

```
String sql = "" +
```

```
"select bno, btitle, bcontent, bwriter,bdate " +
```

```
"from boards " +
```

```
"where bwriter = ? ";
```

//PreparedStatement 얻기 및 값 지정

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1, "summer");
```

//SQL문 실행

```
ResultSet rs = pstmt.executeQuery();
```

```
while(rs.next()) {
```

SELECT 문

```
        while(rs.next()) {  
            //데이터 행을 읽고 Board 객체 생성  
            Board board = new Board();  
            board.setBno(rs.getInt("bno"));  
            board.setBtitle(rs.getString("btitle"));  
            board.setBcontent(rs.getString("bcontent"));  
            board.setBwriter(rs.getString("bwriter"));  
            board.setBdate(rs.getDate("bdate"));  
            //콘솔에 출력  
            System.out.println(board);  
        }  
        //PreparedStatement 닫기  
        pstmt.close();
```