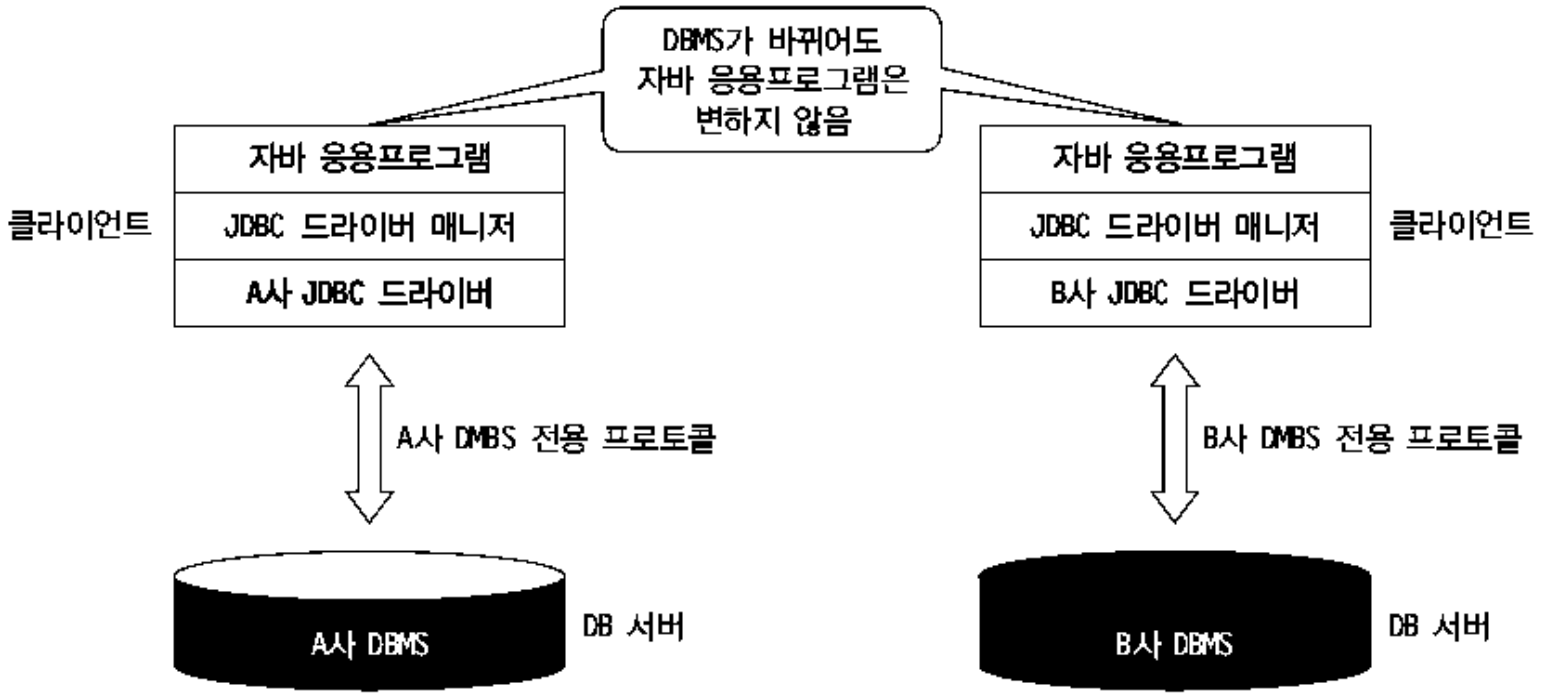
JDBC 구조

1



- JDBC 드라이버 매니저
  - 자바 API에서 지원하며 DBMS를 접근할 수 있는 JDBC 드라이버 로드
- JDBC 드라이버
  - DBMS마다 고유한 JDBC 드라이버 제공, JDBC 드라이버와 DBMS는 전용 프로토콜로 데이터베이스 처리
- DBMS
  - 데이터베이스 관리 시스템. 데이터베이스 생성·삭제, 데이터 생성·검색·삭제 등 전담 소프트웨어 시스템

JDBC 프로그래밍

- 데이터베이스 연결 설정
- Oracle 서버의 JDBC 드라이버 로드(DriverLoadTest.java)

```
try {
   Class.forName("com.mysql.jdbc.Driver");
    System.out.println("드라이버 적재 성공");
} catch (ClassNotFoundException e) {
   System.out.println("드라이버를 찾을 수 없습니다.");
}
```

  - Class.forName()은 동적으로 자바 클래스 로딩
  - Oracle의 JDBC 드라이버 클래스인 *oracle.jdbc.OracleDriver* 로드
  - 자동으로 드라이버 인스턴스를 생성하여 DriverManager에 등록
  - 해당 드라이버가 없으면 ClassNotFoundException 발생

자바 응용프로그램과 JDBC의 연결

- 연결(ConnectDatabase1.java)

```
String url = "jdbc:oracle:thin:@LAPTOP-6OLF446N:1521:XE" ;
String id = "scott";
String pwd = "tiger";
try {
    Connection conn = DriverManager.getConnection(url, id, pwd);
} catch (SQLException e) {
   e.printStackTrace();
}
```

  - DriverManager는 자바 어플리케이션을 JDBC 드라이버에 연결시켜주는 클래스로서 getConnection() 메소드로 DB에 연결하여 Connection 객체 반환
  - getConnection에서 URL의 형식은 DB에 따라 다르므로 JDBC 문서를 참조
    - *Oracle의 경우 디폴트로 1521 포트를 사용*
  - id는 DB에 로그인할 계정 이름이며, pwd는 계정 패스워드

- JDBC 드라이버 로드 & 자바와 JDBC 드라이버 연결 테스트하시오. (ConnectDatabase2.java)

데이터베이스 사용

- Statement 클래스
  - SQL문을 실행하기 위해서는 Statement 클래스를 이용
  - 주요 메소드

메소드	설명
ResultSet executeQuery(String sql)	주어진 sql문을 실행하고 결과는 ResultSet 객체에 반환
int executeUpdate(String sql)	INSERT, UPDATE, 또는 DELETE과 같은 sql문을 실행하고, sql 문 실행으로 영향을 받은 행의 개수나 0을 반환
void close()	Statement 객체의 데이터베이스와 JDBC 리소스를 즉시 반환

  - 데이터 검색을 위해 executeQuery() 메소드 사용
  - 추가, 수정, 삭제와 같은 데이터 변경은 executeUpdate() 메소드 사용

데이터베이스 사용

- ResultSet 클래스
  - SQL문 실행 결과를 얻어오기 위해서는 ResultSet 클래스를 이용
  - 현재 데이터의 행(레코드 위치)을 가리키는 커서(cursor)를 관리
  - 커서의 초기 값은 첫 번째 행 이전을 가리킴
  - 주요 메소드

메소드	설명
boolean first()	커서를 첫 번째 행으로 이동
boolean last()	커서를 마지막 행으로 이동
boolean next()	커서를 다음 행으로 이동
boolean previous()	커서를 이전 행으로 이동
boolean absolute(int row)	커서를 지정된 행 row로 이동
boolean isFirst()	첫 번째 행이면 true 반환
boolean isLast()	마지막 행이면 true 반환
void close()	ResultSet 객체의 데이터베이스와 JDBC 리소스를 즉시 반환
Xxx getXxx(String columnLable)	Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 이름(columnLable)에 해당하는 데이터를 반환한다. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다.
Xxx getXxx(int columnIndex)	Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 인덱스(columnIndex)에 해당하는 데이터를 반환한다. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다.

데이터베이스 사용(1)

- 테이블의 모든 데이터 검색

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from student");
```

  - Statement의 executeQuery()는 SQL문의 실행하여 실행 결과를 넘겨줌
  - 위의 SQL문의 student 테이블에서 모든 행의 모든 열을 읽어 결과를 rs에 저장

- 특정 열만 검색

```
ResultSet rs = stmt.executeQuery("select name, id from student");
```

  - 특정 열만 읽을 경우는 select문을 이용하여 특정 열의 이름 지정

- 조건 검색

```
rs = stmt.executeQuery("select name, id, dept from student where id='0494013'");
```

  - select문에서 where절을 이용하여 조건에 맞는 데이터 검색

데이터베이스 사용(2)

- 검색된 데이터의 사용

```
while (rs.next()) {
    System.out.println(rs.getString("name"));
    System.out.println(rs.getString("id"));
    System.out.println(rs.getString("dept"));
}
rs.close();
```

  - Statement객체의 executeQuery() 메소드
    - ResultSet 객체 반환
  - ResultSet 인터페이스
    - DB에서 읽어온 데이터를 추출 및 조작할 수 있는 방법 제공
  - next() 메소드
    - 다음 행으로 이동

- student 테이블의 모든 데이터를 검색하시오.(Student_SelectTest.java)

데이터의 변경

- 레코드 추가

```
stmt.executeUpdate("insert into student (name, id, dept) values('아무개',
                                        '0893012', '컴퓨터공학')");
```

  - DB에 변경을 가하는 조작은 executeUpdate() 메소드 사용
  - SQL문 수행으로 영향을 받은 행의 개수 반환

- 데이터 수정

```
stmt.executeUpdate("update student set id='0189011' where name='아무개'");
```

- 레코드삭제

```
stmt.executeUpdate("delete from student where name='아무개'");
```