

JDBC

JDBC로 데이터베이스 다루기

최 문 환



1. 관계형 데이터베이스

DBMS

데이터베이스를 관리하기 위해 필요한 수행과정인 데이터의 "추가", "변경", "삭제", "검색" 등의 기능을 집대성한 소프트웨어 패키지

관계형 DBMS로는

오라클(Oracle), mysql ,ms사에서 만든 MS-SQL 등

2. SQL

(1) 데이터 정의어(DDL)

데이터베이스 관리자나 응용 프로그래머가 데이터베이스의 논리적 구조를 정의하기 위한 언어입니다.

(2) 데이터 조작어(DML)

데이터베이스에 저장된 데이터를 조작하기 위해 사용하는 언어로서 레코드 추가(Insert), 삭제>Delete), 갱신(Update) 작업 수행 합니다.

(3) 데이터 제어어(DCL)

데이터에 대한 접근 권한 부여 등의 데이터베이스 시스템의 권한을 관리하기 위한 목적으로 사용되는 언어입니다.

2.1 기본 테이블을 생성하는 CREATE TABLE

■ 형식

CREATE TABLE 테이블명
(컬럼명 자료형(크기) 제약조건);

■ 예제

```
SQL> CREATE TABLE customer (  
2     no    number(4),  
3     name  varchar2(15),  
4     email varchar2(15),  
5     tel   varchar2(15)  
6 );
```

2.2 테이블에 레코드를 추가하는 INSERT

■ 형식

```
INSERT INTO 테이블이름[(컬럼_이름1, 컬럼_이름2,...)]  
VALUES( DATA1, DATA2 ,....);
```

■ 예제

```
INSERT INTO customer VALUES  
(1, '김태은', 'tkKim@hotmail.com', '02-293-4874');
```

2.3 테이블의 레코드를 검색하는 SELECT 문

■ 형식

```
SELECT 컬럼_리스트  
FROM 테이블명  
[WHERE 조건]  
[ORDER BY 컬럼_리스트 [ASC | DESC]];
```

■ 예제

```
SELECT * FROM customer;
```

2.4 저장된 데이터를 변경하는 UPDATE

■ 형식

```
UPDATE 테이블명 SET 컬럼이름1 = DATA1,  
컬럼이름2 = DATA2, 컬럼이름3 = DATA3 .....  
WHERE 조건 문;
```

■ 예제

```
UPDATE customer SET tel='02-123-4567'  
WHERE no = 1;
```

2.5 테이블에 저장된 레코드를 삭제하는 DELETE

■ 형식

```
DELETE FROM 테이블이름 [ WHERE 조건 ];
```

■ 예제

```
DELETE FROM customer  
WHERE no = 1;
```


3. JDBC

- ▶ JDBC(Java Database Connectivity)란
자바에서 데이터베이스에 일관된 방식으로 접근할 수 있도록 API를 제공하는 클래스의 집합

[JDBC와 데이터베이스를 연결하는 방법]

- JDBC 드라이버를 이용하는 방법

3.1 JDBC를 이용한 데이터베이스 조작

1) JDBC 드라이버 로드



2) 데이터베이스와 연결



3) SQL 문 실행



4) 데이터베이스와 연결을 끊는다.



3.2 JDBC를 이용한 데이터베이스 조작

1) DriverManager

2) Connection

3) Statement

4) ResultSet

select문 수행후 반환된 레코드셋 객체 살피기

executeQuery() 메서드는 매개변수로 준 select 문을 데이터베이스로 보내어 실행하도록 하고 그 결과값을 ResultSet으로 받게 된다.

```
String str = "select * from customer";  
ResultSet rs = stmt.executeQuery(str);
```

select문 수행후 반환된 레코드셋 객체 살펴보기

c	a	a	h
cf h f h Fck			
		h a4 cha " ca	! - !((
		m 4 cha " ca	! - !((
		4 cha " ca	! - (! (
9 h f h @ h Fck			

실질적인 데이터가 저장되어 있는 영역과 함께 실제 데이터가 저장되어 있지 않은 영역으로 BOF와 EOF가 함께 존재

BOF(Begin of File)은 첫 번째 로우보다 하나 더 이전의 레코드 셋을 의미하고
EOF(End of File)은 마지막 로우보다 하나 더 다음 레코드 셋을 의미

ResultSet 클래스의 다양한 메소드를 제공

l h	
df j ci	
f h	
h	

```
String sql = "SELECT * FROM customer";  
ResultSet rs = stmt.executeQuery(sql);  
while( rs.next( ) ){  
  
}
```

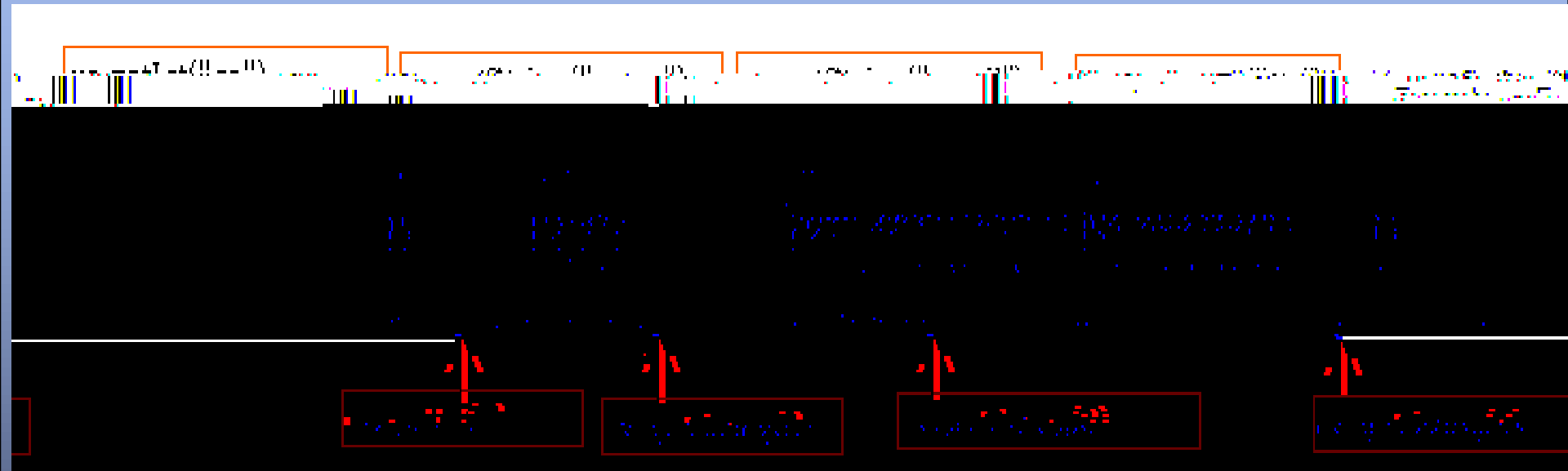
executeQuery 메서드와 SELECT 문

```
while( rs.next( ) ){  
    int  n_no = rs.getInt("no");  
    String  s_name = rs.getString("name");  
    String  s_email = rs.getString("email");  
    String  s_tel = rs.getString("tel");  
    System.out.printf(  
        " %d %t %s %t %s %t %s%t",  
        n_no, s_name, s_email, s_tel);  
}
```

executeQuery 메서드와 SELECT 문

```
while( rs.next( ) ){  
    int n_no = rs.getInt(1);           // 테이블의 첫 번째 컬럼(즉, no)  
    String s_name = rs.getString(2);  // 테이블의 두 번째 컬럼(즉, name)  
    String s_email = rs.getString(3); // 테이블의 세 번째 컬럼(즉, email)  
    String s_tel = rs.getString(4);    // 테이블의 네 번째 컬럼(즉, tel)  
    System.out.printf(" %d \t %s \t %s \t %s\n",  
        n_no, s_name, s_email, s_tel);  
}
```

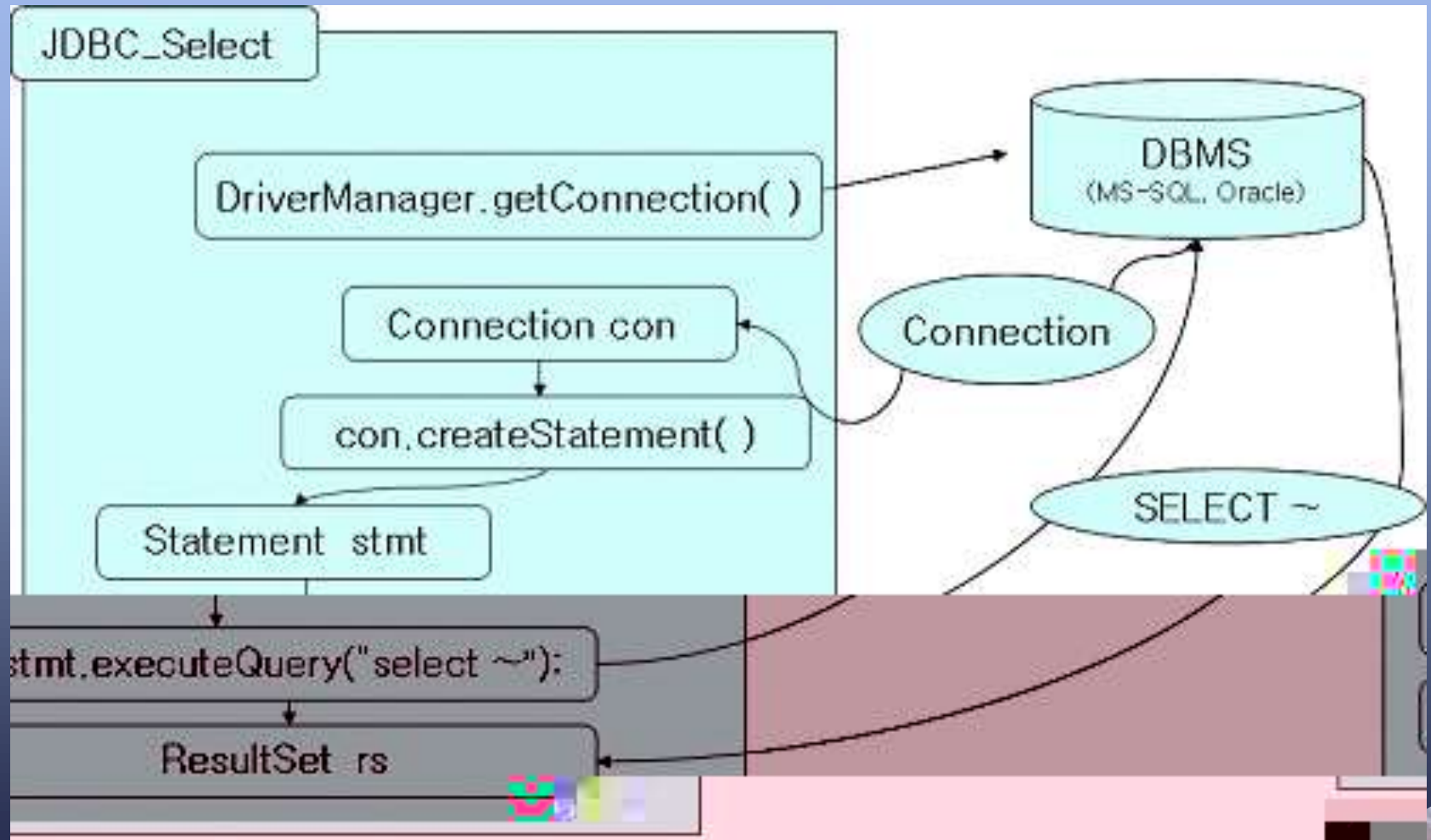

검색후 레코드를 가져오는 법



executeQuery 메서드와 SELECT 문

- JDBC 드라이버 로드되었다면 DriverManager 클래스의 getConnection 메서드로 데이터베이스 연결 객체인 Connection 생성합니다.
- Connection 객체로 createStatement 메서드로 Statement 객체를 생성합니다.
- Statement 객체로 executeQuery 메서드로 SQL 문 실행하여 ResultSet 객체를 생성합니다.
- ResultSet 객체로 결과 처리를 합니다.

JDBC를 이용해서 데이터베이스에 연결하여 SQL 문을 수행하여 결과를 출력



5. executeUpdate메서드와 다양한 SQL 문

테이블에 레코드 추가하는 insert 문

```
020:    BufferedReader br = new BufferedReader(  
        new InputStreamReader(System.in));  
021:  
022:    System.out.println(" customer 테이블에 값 입력하기 .....");  
023:    System.out.print(" 번호 입력: ");  
024:    s_no = br.readLine( );  
025:    System.out.print(" 이름 입력: ");  
026:    s_name = br.readLine( );  
027:    System.out.print(" 이메일 입력: ");  
028:    s_email = br.readLine( );  
029:    System.out.print(" 전화번호 입력: ");  
030:    s_tel  = br.readLine( );
```

5.executeUpdate메서드와 다양한 SQL 문

테이블에 레코드 추가하는 insert 문

```
031:
032:  // INSERT 쿼리문을 작성
033:  sql="INSERT INTO customer VALUES (“
        +s_no+", '"+s_name+"', '"+s_email+"', '"+s_tel+"'");
034:
035:  //Statement객체의 executeUpdate 메서드로 테이블에 행을 추가
036:  stmt.executeUpdate(sql) ;
```

5. executeUpdate메서드와 다양한 SQL 문

테이블의 내용을 변경하는 update 문

```
019:    BufferedReader br = new BufferedReader(  
        new InputStreamReader(System.in));  
020:  
021:    System.out.println(" customer 테이블 갱신하기");  
022:    System.out.print("갱신할 분의 이름을 입력: ");  
023:    s_name = br.readLine( );  
024:    System.out.print("변경할 이메일 입력: ");  
025:    s_email = br.readLine( );  
026:    System.out.print("변경할 전화번호 입력: ");  
027:    s_tel = br.readLine( );
```

5. executeUpdate메서드와 다양한 SQL 문

테이블의 내용을 변경하는 update 문

```
029: // UPDATE 쿼리문을 작성
030: sql = "UPDATE customer SET email = '" + s_email ;
031: sql += "' , tel = '" + s_tel + "' WHERE name = '" ;
032: sql += s_name + "'";
033:
034: //Statement 객체의 executeUpdate 메서드로 테이블의 내용을 변경
035: stmt.executeUpdate(sql);
```

5. executeUpdate메서드와 다양한 SQL 문

테이블에 레코드 삭제하는 delete 문

```
020:    BufferedReader br = new BufferedReader(  
        new InputStreamReader(System.in));  
021:  
022:    System.out.println(" customer 테이블에서 레코드 삭제하기");  
023:    System.out.print("삭제할 분의 이름을 입력: ");  
024:    s_name = br.readLine( );  
025:  
026:    // DELETE 쿼리문을 작성  
027:    sql = "DELETE FROM customer WHERE name ='";  
028:    sql += s_name +'";  
029:  
030:    //Statement 객체의 executeUpdate 메서드로 행을 삭제  
031:    stmt.executeUpdate(sql) ;
```

No.24