

제 16 장

JDBC

인제대학교 정보통신공학과

황 원 주

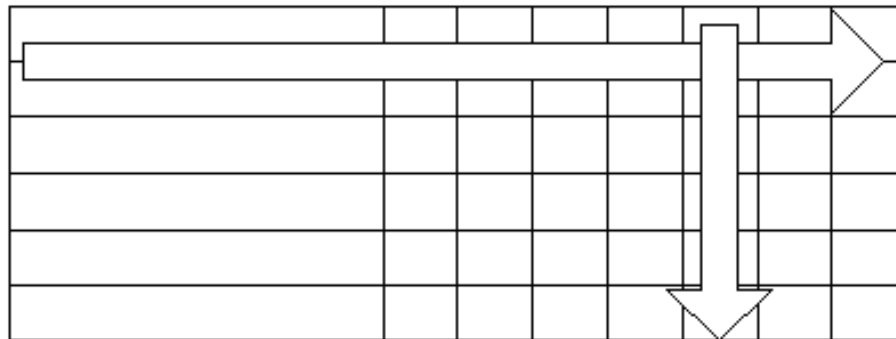


1. 관계형 데이터베이스
2. SQL (Structured Query Language)
 - 2.1 SQL 데이터 정의 명령어
 - 2.2 SQL 데이터 조작어
 - 2.3 SQL 연습
3. JDBC (Java Database Connectivity) 기본 개념
 - 3.1 JDBC 프로그래밍 절차
 - 3.2 JDBC 예제
 - 3.3 JDBC의 주요 메서드

1. 관계형 데이터베이스

관계형 데이터베이스

- ▶ SQL: 데이터베이스를 조작하기 위한 언어
- ▶ JDBC: 자바와 데이터베이스 연동시키는 일련의 기술
- ▶ 데이터베이스의 분류: 계층형 DB, 네트워크 DB, 관계형 DB
- ▶ 관계형 데이터베이스 (Relational Data Base)는 테이블(table)이라고 불리는 객체로 구성된다. 테이블은 연관된 데이터의 집합이다. 테이블은 다수의 레코드(또는 tuple, row)로 구성되며, 각 레코드는 필드(또는 속성, column)로 구성



행(Row), 튜플(Tuple)

열(Column)

용어 정의

▶ Relation: 일정한 법칙을 부여한 데이터의 집합, 즉 데이터를 포함하고 있는 테이블을 의미

▶ Tuple : 튜플은 테이블의 행에 해당

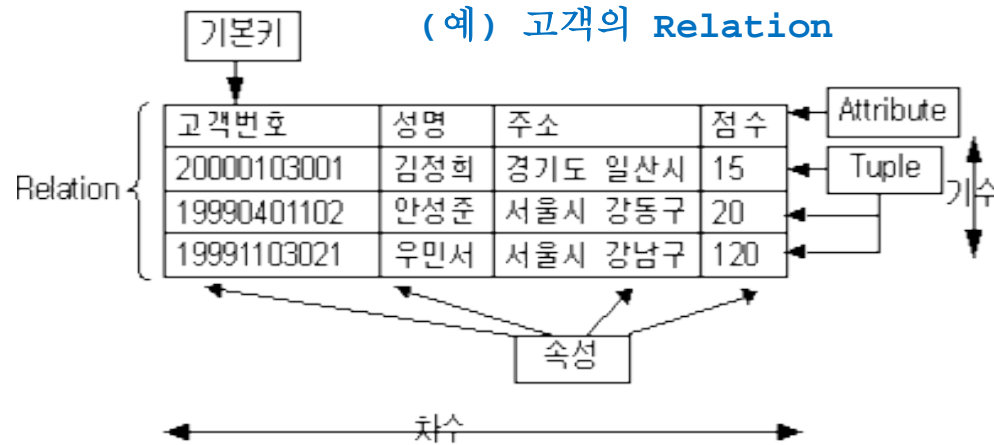
▶ 속성 (Attribute) : 테이블의 열에 해당

▶ 기본키 (Primary Key) : 기본키는 테이블의 유일한 식별자 (Identifier). 즉, 주어진 임의의 시간에 테이블의 두 행이 한 열(혹은 열의 조합)에서 같은 값을 갖지 않는 성질이 있는 열(혹은 열의 조합)을 의미. 학생 ID, 주민 등록 번호, 사원 번호 등과 같은 속성들이 일반적으로 기본 키로 선택됨

▶ Domain : 값들의 Pool로서, 특정한 릴레이션의 특정한 속성으로부터 실제값을 취한다.

▶ 기수 (Cardinality) : 튜플의 수

▶ 차수 (Degree) : 속성의 수



일반 데이터베이스에서 지원되는 자료형

Type	설명
CHAR(<i>n</i>)	고정 크기 <i>n</i> 의 문자열을 지정한다.
VARCHAR(<i>n</i>)	최대 크기 <i>n</i> 의 가변 문자열을 지정한다.
DATE	날짜형식의 데이터 유형을 지정한다. (yyyymmdd)
TIME	시간형식의 데이터 유형을 지정한다. (hhmmss)
DATETIME	날짜와 시간형식의 데이터 유형을 지정한다.
BINARY	이진 형식의 데이터 유형을 지정한다.
DECIMAL	실수 데이터형을 지정한다.
INTEGER	정수 데이터형을 지정한다. (4바이트 정수)
NUMERIC	정밀도를 결정하는 실수 데이터형을 지정한다.
TIMESTAMP	레코드 갱신 시간을 저장하는 형식이다.

2. SQL

SQL문장의 분류

① 데이터 정의어 (DDL: Data Definition Language)

- 데이터베이스의 테이블을 생성하거나 수정하는 언어이다. CREATE TABLE, ALTER TABLE, CREATE INDEX, DROP TABLE 등이 DDL에 속한다.

② 데이터 조작어 (DML: Data Manipulation Language)

- 데이터베이스에 새로운 데이터를 저장하거나, 저장된 데이터를 갱신하는 언어와 데이터를 검색하는 질의어로 구성된다.

2.1 SQL 데이터 정의 명령어

- ① **Create table** : 새로운 테이블을 생성할 때 사용되는 문장
형식)

```
CREATE TABLE table-name  
( col_name type { , col_name type }* )
```

예)

```
CREATE TABLE Student (  
    stud_ID INTEGER NOT NULL PRIMARY KEY,  
    name CHAR(12) NOT NULL,  
    address VARCHAR(30))
```

- ② **Drop table**: 이미 생성된 테이블을 삭제하는 문장
형식)

```
DROP TABLE table-name
```

2.2 SQL 데이터 조작성

- ① **INSERT:** 원하는 테이블에 새로운 레코드를 추가하는 명령어
형식)

```
INSERT INTO table VALUES( field_value { , field_value } )
```

예)

```
INSERT INTO student VALUES  
( 11001, '김 정수', '서울시성북구돈암2동')
```

- ② **UPDATE:** 원하는 레코드의 값을 변환하기 위한 명령어

형식)

```
UPDATE table  
SET field = expression { , field = expression }  
[WHERE condition]
```

예)

```
UPDATE goods  
SET price = price * 1.2  
WHERE name = 'computer'
```

③ **DELETE:** 원하는 레코드를 삭제하는 명령어

형식)

```
DELETE FROM table
[WHERE condition]
```

예)

```
DELETE FROM goods
WHERE p_name LIKE 'Pentium II%'
```

▶ LIKE 구에서 % 또는 * 는 임의의 길이 문자를 나타낸다. _(언더라인) 또는 ?는 문자 1 개를 나타낸다.

④ **SELECT:** 원하는 데이터를 검색하는 데 사용되는 문장

형식)

```
SELECT [ ALL | DISTINCT ] field_list
FROM table_list
[WHERE condition]
[GROUP BY field]
[HAVING condition]
[ORDER BY field [ASC | DESC]]
```

예) 연령이 22을 초과하는 학생의 이름과 전공을 출력하시오.

```
SELECT name, major FROM student
WHERE age > 22
```

- ▶ SQL에서 지원되는 집단(aggregation) 함수: 집단 함수는 테이블의 필드 집합에 적용

함수 이름	설 명
COUNT	값의 갯수
SUM	합
AVG	평균
MAX	최대값
MIN	최소값

예) 전공이 컴퓨터 사이언스인 학생들의 평균 연령을 구하시오.

```
SELECT AVG(age) FROM student  
WHERE major = 'computer science'
```

2.3 SQL 연습

은행 데이터베이스를 사용한 SQL 예제

- ▶ M 은행의 데이터베이스는 세 개의 테이블로 구성되어 있으며, 테이블의 이름은 각각 Bank, Client, Deposit
- ▶ Deposit 테이블에는 고객과 은행 지점간의 예금 구좌 및 잔액 정보를 저장

각 테이블에는 다음과 같은 정보가 저장되어 있다.

Bank table		
B_id	B_name	City
1101	한성대점	서울
1102	대학로점	서울
1103	신촌점	서울
1011	충남대점	대전
1012	대덕단지	대전
2001	맨해튼점	뉴욕
2002	할렘점	뉴욕

Client table			
Client_id	Name	Job	City
1001	서지수	대학생	서울
1002	김정수	대학생	서울
1003	김정훈	은행원	서울
1004	김난숙	교사	대전
1005	미난숙	교수	대구
1006	평말자	자영업	대전
1007	박경태	교수	서울
1008	문정호	연구원	서울
1009	하종곤	의사	뉴욕
1010	김석곤	의사	뉴욕
1011	오원택	사장	대구
1012	김미원	주부	대전
1013	김미숙	주부	서울

Deposit table			
B_id	Client_id	Acc_no	Balance
1101	1001	11001	12000
1101	1002	11002	1000000
1101	1003	11003	15000
1101	1007	11004	5000000
1102	1008	11005	6000000
1102	1013	11006	80000
1102	1001	11007	900000
1103	1002	11008	120000
1011	1006	11009	150000
1012	1004	11010	2000000
1012	1005	11011	5000000
2001	1009	11012	7000000
2002	1012	11013	9000000
2002	1010	10014	18000
1011	1007	10015	2000000
1011	1008	10016	22000

1) Client 테이블에서 직업이 의사인 고객의 이름과 도시를 출력하시오.

답)

```
SELECT Name, city FROM Client WHERE Job = '의사'
```

결과)

하종곤 뉴욕

김석곤 뉴욕

2) 직업이 교수인 고객의 평균 예금 잔액은 얼마인가?

답)

```
SELECT AVG(Deposit.Balance)
```

```
FROM Client, Deposit
```

```
WHERE Client.job = '의사' and
```

```
Deposit.Client_id=Client.Client_id
```

결과)

3509000.0

3) 한성대점에 예금을 하고 있는 고객의 이름을 모두 출력하시오.

답)

```
SELECT Name
FROM Client, Deposit, Bank
WHERE B_name='한성대점' and Bank.B_id=Deposit.B_id
and Deposit.Client_id=Client.Client_id
```

결과)

서지수 김정수 김정훈 박경태

4) Deposit 테이블에서 두 개 이상의 예금 구좌를 가진 고객의 id를 출력하시오.

답)

```
select Client_id
from Deposit
group by Client_id having count(*) >= 2
```

결과)

1001 1002 1007 1008

3. JDBC 기본 개념

3.1 JDBC 프로그래밍 절차

① JDBC에 필요한 패키지를 포함한다.

: JDBC에 필요한 패키지는 `java.sql`이다.

예) `import java.sql.*;`

② 적절한 JDBC 드라이버를 적재한다.

: `Class.forName()` 메소드를 사용하여 필요한 JDBC 드라이버를 적재한다.

: 여기에서 사용하는 JDBC-ODBC 드라이버는 썬사에서 기본으로 제공하며, JDBC-ODBC 드라이버이름은 `sun.java.odbc.JdbcOdbcDriver`이다.

예) `Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`

③ 데이터베이스 URL, user_name, pass_word 정보를 사용하여 Connection 객체를 생성한다. 즉 자바 프로그램과 데이터베이스간에 연결을 시도한다. 연결에 사용되는 명령어는 DriverManager 클래스의 정적 메소드 getConnection()이다.

```
예) connection =  
    DriverManager.getConnection("jdbc:odbc:myDB" , "ikim" , "7788");
```

위의 예는 JDBC-ODBC 브리지를 사용하며, 데이터베이스 원본 이름은 myDB이다. ikim은 사용자 이름이며, 패스워드는 7788이다.

④ 생성된 Connection 객체를 사용하여 Statement 객체를 생성한다.

```
예) stat = connection.createStatement();
```

⑤ 생성된 Statement 객체를 이용하여 SQL 문장을 실행한다. 실행 결과는 ResultSet 객체에 저장한다. 사용되는 메소드는 execute(), executeQuery(), executeUpdate() 중에 하나를 사용한다.

```
예) ResultSet rs = stat.executeQuery("select * from  
    order");
```

- ⑥ ResultSet 객체 rs에서 원하는 결과를 추출한다. 주의할 점은 rs.next() 메소드를 먼저 실행한 후, getXXX() 메소드를 실행하여 원하는 유형의 데이터를 추출하여 처리한다. (XXX는 반환하는 데이터의 유형)

예)

```
while(rs.next()) {  
    String s = rs.getString(1);    // 첫 번째 필드에서  
                                    // 문자열 데이터 반환  
    int x = rs.getInt(2);          // 두 번째 필드에서  
                                    // 정수형 데이터 반환  
}
```

- ⑦ ResultSet 객체, Statement 객체, Connection 객체를 모두 닫는다.

예)

```
rs.close();  
stmt.close();  
connection.close();
```

예제

```
import java.sql.*; //① JDBC 프로그램을 위한 패키지

public class JdbcTest {
    static String[] sql = {
        "create table telBook(id int AUTO_INCREMENT PRIMARY KEY, name varchar(100), phone
        varchar(20));",
        "insert into telBook(name, phone) values ('Kim', '777-7788');",
        "insert into telBook(name, phone) values ('Lee', '760-4315');",
        "insert into telBook(name, phone) values ('Jung', '789-4421');",
        "insert into telBook(name, phone) values ('Park', '763-1772');"
    };

    public static void main(String args[]) {
        String URL = "jdbc:mysql://localhost/myDB"; // dbc:mysql//host[Ip](:port)/dbname)
        String username = "root";
        String password = "7788";
        try {
            Class.forName("org.gjt.mm.mysql.Driver").newInstance(); //② jdbc-odbc드라이버 적재
            Statement stmt = null;
            Connection con = null;

            // ③ Connection객체와 statement 개체 생성
            con = DriverManager.getConnection(URL, username, password);
            stmt = con.createStatement();
```

예제

```
try {  
  
    // ④ 테이블 생성 및 데이터 입력  
    for (int i=0; i<sql.length; i++)  
        stmt.execute(sql[i]);  
} catch (java.sql.SQLException ex) {}  
  
    // ⑤ SQL문 실행  
ResultSet rs = stmt.executeQuery("select * from telBook;");  
  
System.out.println("ID      Name      Phone Number");  
    // 다음 결과의 유무 검사  
while(rs.next()) {  
    int id = rs.getInt(1);  
    String name = rs.getString(2); // ⑥ 각 필드(String)의 값을 반환  
    String phone = rs.getString(3);  
    System.out.println(id + "\t" + name + "\t" + phone); // DB 내용 출력  
}  
con.close(); // ⑦ Connection 객체를 모두 닫음  
} catch (Exception e) { System.out.println(e); }  
}
```

실행결과

```
C:\java>java JdbcTest
```

```
Name Tel_no
```

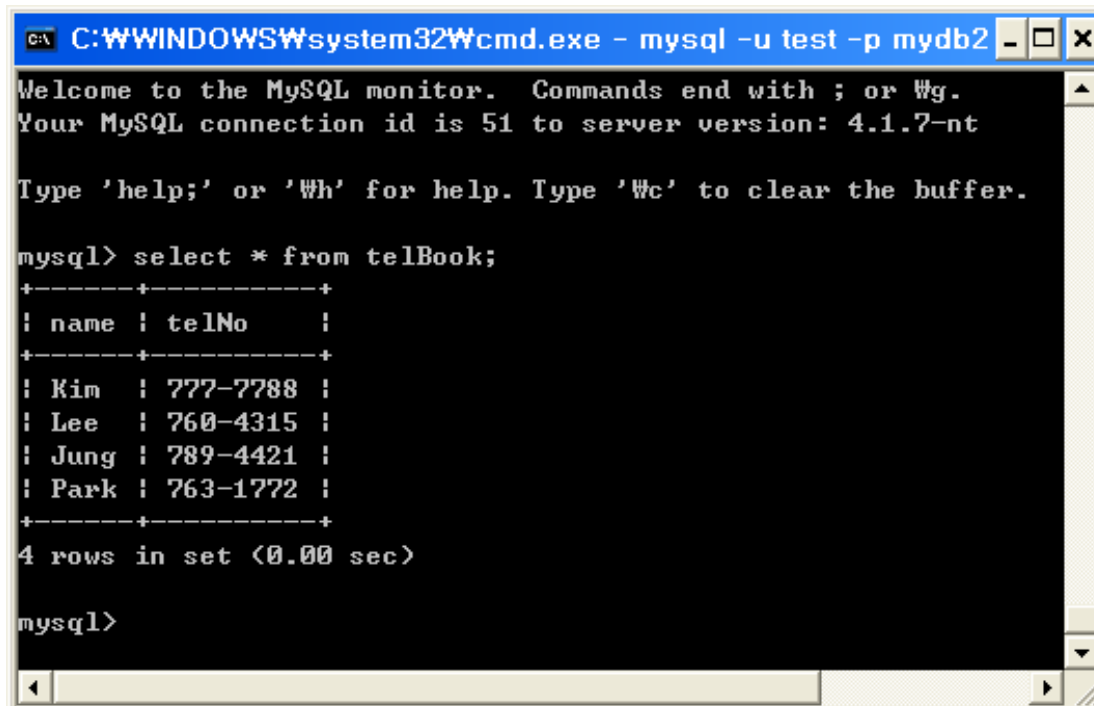
```
Kim 777-7788
```

```
Lee 760-4315
```

```
Jung 789-4421
```

```
Park 763-1772
```

- ▶ 예제 프로그램을 실행시킨 후에 MySQL서버의 myDB내 telBook 테이블을 내용을 확인 가능



```
C:\WINDOWS\system32\cmd.exe - mysql -u test -p mydb2 - _ □ ×  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 51 to server version: 4.1.7-nt  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql> select * from telBook;  
+-----+-----+  
| name | telNo  |  
+-----+-----+  
| Kim  | 777-7788 |  
| Lee  | 760-4315 |  
| Jung | 789-4421 |  
| Park | 763-1772 |  
+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql>
```