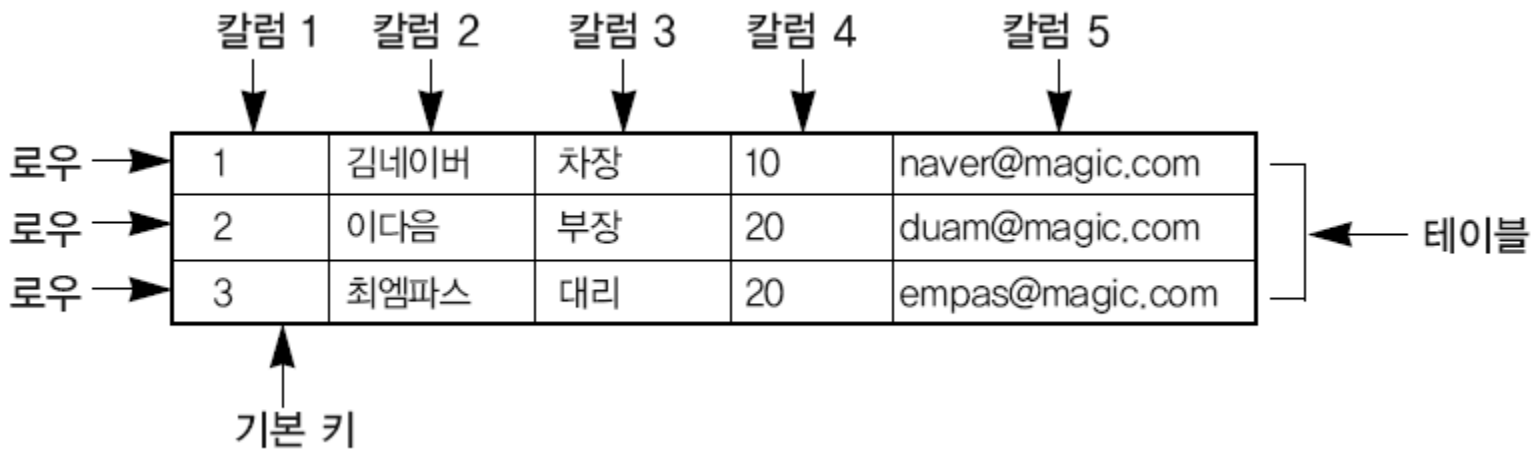
18. JDBC

강사 강태광

1] 데이터베이스 개요

▶ 연관된 데이터들의 묶음을 데이터베이스라고 한다.
- 데이터베이스에서는 정보를 저장하기 위해서 테이블을 사용
- 테이블은 표처럼 볼 수 있도록 로우(ROW)와 칼럼(COLUMN)으로 구성



1] 데이터베이스 – Table의 구성 및 Script

부서번호 부서이름 위치

DEPTNO	DNAME	LOC
10	회계2팀	구로2
20	RESEARCH	SEOUL
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	영업2부	판교

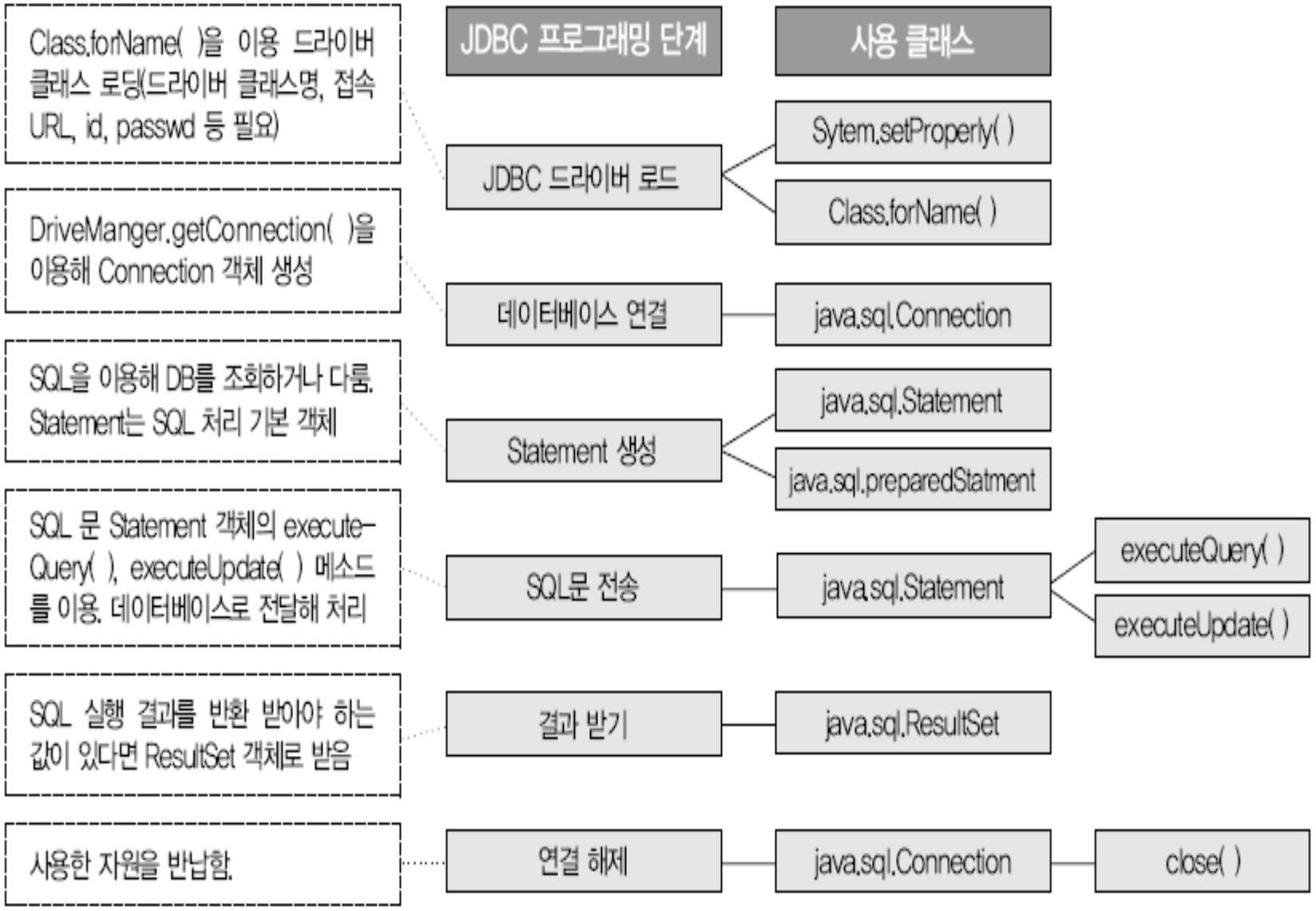
```
CREATE TABLE DEPT
   (          "DEPTNO" NUMBER(2,0),
              "DNAME" VARCHAR2(14 BYTE),
              "LOC" VARCHAR2(13 BYTE),
               PRIMARY KEY ("DEPTNO")
   )
```

2] JDBC 프로그래밍

1. JDBC(Java Database Connectivity)
   자바 패키지의 일부로 자바 프로그램이 데이터베이스와 연결되어
   데이터를 주고받을 수 있게 해 주는 프로그래밍 인터페이스이다.
   자바 데이터베이스 프로그래밍 API라고 할 수 있다.

2. JDBC 드라이버
   My SQL 또는 오라클 드라이버를 자바가 설치되어 있는
   C:₩Program Files₩jdk-11.0.12₩bin₩bin
   C:₩Program Files₩Java₩jre1.8.0_73₩lib₩ext 란 폴더에 복사한다.
   OJDBC6.JAR, mysql-connector-j-8.0.33.jar

3] JDBC 프로그래밍 단계 및 Class



4] JDBC 프로그래밍 – 드라이버/Connection

1) java.lang.Class 클래스의 정적 메소드 forName()는 패키지 명을 기술한 후 클래스 이름을 문자열 형태로 지정해 주면 이를 자바 가상 기계에 안으로 읽어 들이도록 한다.

2) JDBC 드라이버는 다음과 같이 로드한다.
   ① Oracle -> Class.forName('oracle.jdbc.driver.OracleDriver');
   ② mySQL -> Class.forName('com.mysql.cj.jdbc.Driver');

3) Connection 객체
- 데이터베이스를 연결해 작업을 수행할 수 있도록 만들어 주는 중요한 객체
- DriverManager 클래스의 static 메소드인 getConnection()을 호출해야 한다.
- Connection con = DriverManager.getConnection(url , uid, pwd);
   ① url : 관계형 데이터베이스 엔진에서 위치
   ② uid : 사용자 계정
   ③ pwd : 사용자 패스워드

4] JDBC 프로그래밍 – 드라이버/Connection 예시

```
package ch18;
import  java.sql.*;
public class OraDr {
	public static void main(String[] args) {
		String driver = "oracle.jdbc.driver.OracleDriver";
		// Localhost -> 127.0.0.1; , Port 번호:1521  , xe(orcl)-> Service ID(Sid)
		String url    = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
		try {
			Class.forName(driver);
			Connection conn = DriverManager.getConnection(url,"scott","tiger");
			System.out.println("Start");
			if (conn != null) {
			//System.out.println("Success");
			System.out.println("Success 연결 성공");
			} else {
			System.out.println("Fail");

			}
			conn.close();
		} catch (Exception e) {
			// TODO Auto-generated catch block
			//System.out.println(e.getMessage());
			System.out.println("Error");
		}
	}
}
```

4] JDBC 프로그래밍 –Statement / ResultSet

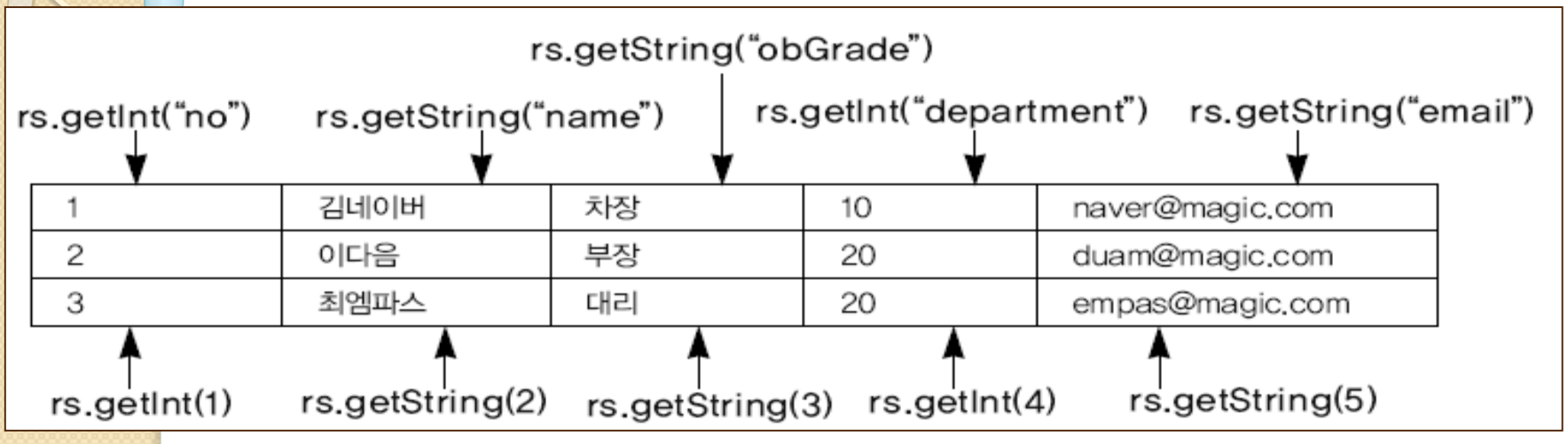
1) Statement 객체
- 이전 단계에서 생성한 Connection 객체(con)로 접근해서 createStatement() 메소드를 호출해서 생성한다
- Statement stmt = con.createStatement( );

2) 모든 작업이 끝나면 Statement 객체 역시 close() 메소드를 호출해서 데이터베이스와 연결을 해제해야 한다.
- stmt.close( );

3) Statement 객체로 쿼리문 수행
- select 문과 같이 결과가 있는 쿼리문인 경우에는 executeQuery () 메소드 사용
- insert, update, delete 문과 같이 내부적으로는 어떤 변화가 있지만 결과가 없는 경우에는 executeUpdate() 메소드 사용
- String str = "select * from employee";
  ResultSet rs = stmt.executeQuery(str);

3] JDBC 프로그래밍 –Statement 객체 Query문

1) ResultSet 객체는 다음과 같은 형태로 executeQuery() 메소드는 기술한 select 문의 결과 값을 저장



2) ResultSet은 다음과 같은 다양한 메소드를 제공한다.

메소드	설명
next( )	현재 행에서 한 행 앞으로 이동
previouse( )	현재 행에서 한행 뒤로 이동
first( )	현재 행에서 첫 번째 행의 위치로 이동
last( )	현재 행에서 마지막 행의 위치로 이동

3] JDBC 프로그래밍 –Statement 객체 Query문

- 결과 값으로 얻어진 여러 개의 로우를 모두 출력하기 위해서는 ResultSet 객체로 레코드 단위로 이동하는 next() 메소드를 사용해야 하는데 일반적으로 다음과 같이 while 문과 함께 사용.

```
while( rs.next( ) ){
        System.out.print(rs.getInt(1) +" ₩t"); //no
        System.out.print(rs.getString(2) +" ₩t"); //name
        System.out.print(rs.getString(3) +" ₩t"); //jobGrade
        System.out.print(rs.getInt(4) +" ₩t"); //department
        System.out.print(rs.getString(5) +" ₩n"); //email
}
```

3] JDBC 프로그래밍 –Statement 객체 Query문 예시

```
package ch18;
import java.sql.*;
import java.util.Scanner;
public class OraSelect1 {
	public static void main(String[] args) throws SQLException {
		Scanner sc = new Scanner(System.in);
		System.out.println("부서코드를 입력하세요");
		int     deptno = sc.nextInt();
		String driver = "oracle.jdbc.driver.OracleDriver";
		// Localhost -> 127.0.0.1; , Port ��8:1521  , xe(orcl)-> Service ID(Sid)
		String url    = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
		String sql = "Select dname,loc From Dept Where deptno=" +deptno;
		Connection   conn   = null;  // DB
		Statement    stmt   = null;  // Sql
		ResultSet    rs     = null;  //
		try {
			Class.forName(driver);   // Driver
			conn    =  DriverManager.getConnection(url,"scott","tiger");
			stmt    = conn.createStatement();       // stmt
			rs      = stmt.executeQuery(sql);       // SQL
			 // rs  Row
			if (rs.next()) {
				String dname = rs.getString("dname");  // rs.getString(1)
				String loc   = rs.getString("loc");    // rs.getString(2)
				System.out.println("부서명 :" + deptno); System.out.println("부서명 :" + dname);
				System.out.println("위치  :" + loc);
			}
			else {
			    System.out.println("..." );
			}
		} catch(Exception e) {
			System.out.println(e.getMessage());
		} finally {
			if( rs != null)    rs.close();
			if( stmt != null)  stmt.close();
			if( conn != null)  conn.close();
		}
		sc.close();
	}
}
```

4] JDBC 프로그래밍 –executeUpdate() 메소드

1. insert 문, update 문, delete 문, create 문과 같이 데이터베이스 파일의 내용을 변경하는 SQL 문을 실행할 때 사용한다.
2. executeUpdate() 메소드는 변경된 행(레코드)의 수를 반환해 주기에 리턴 형이 Integer이다.
3. 키보드에서 읽어온 변수에 저장된 값으로 행을 추가할 경우에는 어떻게 해야 할까?
   - 문자열 상수는 반드시 단일 따옴표로 둘러 싸주어야 하기에 다음과 같이 복잡한 형태의 문장을 만들어야 한다.

```
String sql = "insert into employee(name, jobGrade, department";
sql += " , email) values ('" + sname + "', '" + sjobGrade;
sql += "', " + " ndepartment + ", '" + semail + "')";
```

4] JDBC 프로그래밍 –executeUpdate() 메소드 적용예시

```
package ch18;
import  java.sql.*;
import java.util.Scanner;
import javax.print.attribute.standard.Finishings;
public class OraInsert {
   public static void main(String[] args) throws SQLException {
            Scanner sc = new Scanner(System.in);
            System.out.println("부서번호 입력?");   String deptno = sc.nextLine();
            System.out.println("부서명 입력?");     String dname  = sc.nextLine();
            System.out.println("위치 입력 ?");     String loc    = sc.nextLine();

            Connection conn = null;
            Statement  stmt = null;
            String driver = "oracle.jdbc.driver.OracleDriver";
          // Localhost -> 127.0.0.1; , Port 번호:1521  , xe(orcl)-> Service ID(Sid)
            String url    = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
            String sql    = String.format("Insert Into dept values(%s, '%s','%s')", deptno, dname, loc);
          //   String sql    = "Insert Into dept values("+ deptno+",'"+dname+"','"+loc+"')";
            try {
                     Class.forName(driver);
                     conn = DriverManager.getConnection(url,"scott","tiger");
                     stmt = conn.createStatement();
                     // result Set
                     int result = stmt.executeUpdate(sql); // 수행 -> executeUpdate
                     if (result > 0 ) System.out.println("입력성공 ^^");
                     else             System.out.println("입력실패 T.T");
            } catch(Exception e) {
                     System.out.println(e.getMessage());
            }finally {
                     if (stmt != null) stmt.close();
                     if (conn != null) conn.close();
           }
           sc.close();
   }
}
```

5] JDBC 프로그래밍 –PreparedStatement() 메소드

1) PreparedStatement를 생성
- PreparedStatement 객체를 생성하기 위해서는 Connection 인터페이스의 preparedStatement() 메소드를 호출한다
- PreparedStatement pstmt= con.prepareStatement(sql);

2) preparedStatement() 메소드의 인자로 사용되는 SQL 문은 다음과 같이 ? 기호를 사용해서 다음과 같이 표현할 수 있다
- String sql ="insert into employee values(?, ?, ?, ?, ?)"; // ① ② ③ ④ ⑤

3) ?로 지정된 인자에 값을 준다.
- setXXX(int 순서, 실제 데이터나 변수);

5] JDBC 프로그래밍 –PreparedStatement() 메소드 적용 예시

```
package ch18;
import  java.sql.*;
import java.util.Scanner;

import javax.print.attribute.standard.Finishings;
public class OraPrepare {
   public static void main(String[] args) throws SQLException {
            Scanner sc = new Scanner(System.in);
            System.out.println("입력할 Oracle 부서코드 ?");   String deptno = sc.nextLine();
            System.out.println("입력할 Oracle 부서명 ?");     String dname  = sc.nextLine();
            System.out.println("입력할 Oracle 근무지 ?");     String loc    = sc.nextLine();

            Connection conn = null;
            PreparedStatement  pstmt = null;
            String driver = "oracle.jdbc.driver.OracleDriver";
         // Localhost -> 127.0.0.1; , Port 번호:1521  , xe(orcl)-> Service ID(Sid)
            String url    = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
            String sql    = "Insert Into dept values(?, ?, ?)";
         //   String sql    = "Insert Into dept values("+ deptno+",'"+dname+"','"+loc+"')"; 위와 같은 의미

            try {
                     Class.forName(driver);
                     conn = DriverManager.getConnection(url,"scott","tiger");
                     pstmt = conn.prepareStatement(sql);    // sql추가
                     pstmt.setString(1, deptno);
                     pstmt.setString(2, dname);
                     pstmt.setString(3, loc);
                     // result는 작업에 성공한 갯수
                     int result = pstmt.executeUpdate(); // 입력/수정/삭제시 -> executeUpdate
                     if (result > 0 ) System.out.println("OraPrepare 입력성공 ^^");
                     else             System.out.println("OraPrepare 입력실패 T.T");
            } catch(Exception e) {
                  System.out.println(e.getMessage());
            }finally {
                     if (pstmt != null) pstmt.close();
                     if (conn != null) conn.close();
           }
           sc.close();

   }
}
```

5] JDBC 프로그래밍 - Stored Procedure와 CallableStatement

```
package ch18;
import  java.sql.*;
import java.util.Scanner;
public class OraProc1 {
  public static void main(String[] args) throws SQLException {
            Scanner sc = new Scanner(System.in);
            System.out.println("입력할 Oracle 부서코드 ?");   String deptno = sc.nextLine();
            System.out.println("입력할 Oracle 부서명 ?");     String dname  = sc.nextLine();
            System.out.println("입력할 Oracle 근무지 ?");     String loc    = sc.nextLine();

            Connection          conn = null;
            CallableStatement   cs   = null;
            String driver = "oracle.jdbc.driver.OracleDriver";
          // Localhost -> 127.0.0.1; , Port 번호:1521  , xe(orcl)-> Service ID(Sid)
            String url    = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
            // Procedure Call할 때의 Format
            String sql    = "{call dept_Insert(?, ?, ?)";

            try {
                    Class.forName(driver); // 없어도 될수 있음 , 연결 되어 있다면
                    conn = DriverManager.getConnection(url,"scott","tiger");
                    cs   = conn.prepareCall(sql);
                    cs.setString(1, deptno);
                    cs.setString(2, dname);
                    cs.setString(3, loc);
                    int result  = cs.executeUpdate();
                    if ( result > 0 ) System.out.println("Oracle CallableStatement 입력 성공 ^ ^ ");
                    else              System.out.println("Oracle CallableStatement 입력 실패 T.T ");
            }catch(Exception e) {
                    System.out.println(e.getMessage());
            } finally {
                    if( cs   != null ) cs.close();
                    if( conn != null ) conn.close();
            }

            sc.close();

  }

}
```