JDBC

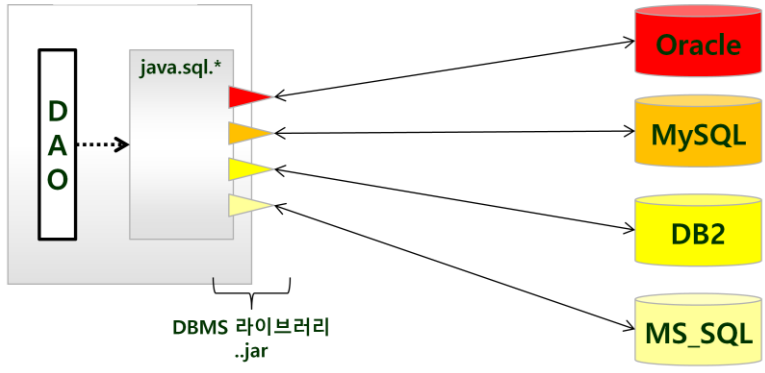
- JDBC API
- JDBC 프로그래밍
- Statement
- ResultSet
- JDBC 응용 프로그래밍

17-1

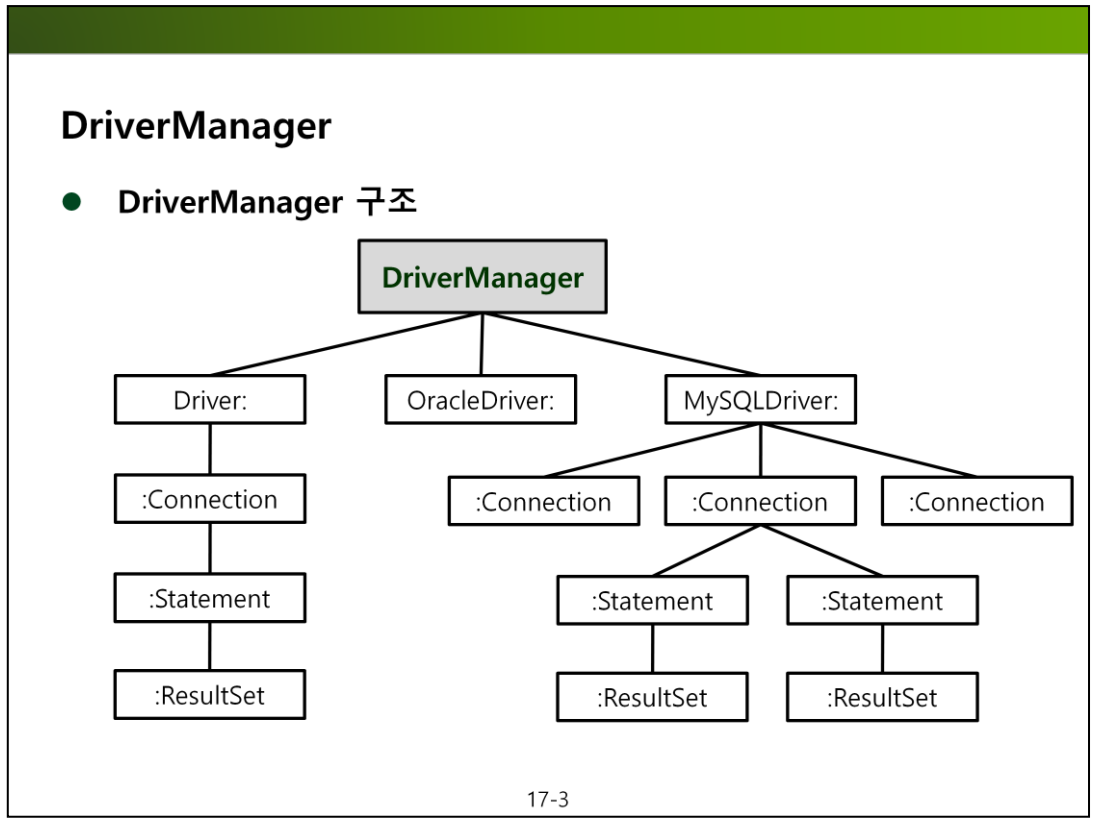
JDBC

- JDBC(Java Database Connectivity)
- 자바 응용 프로그램과 DBMS를 연결하기 위한 인터페이스
  - Oracle, MS_SQL, DB2 등 DBMS의 독립적은 프로그래밍 방식 제공
- java.sql.*;

JDBC 프로그램



17-2



java.sql.DriverManager 클래스는 여러 DBMS의 드라이버 객체를 관리할 수 있다.

각 DBMS는 JDBC 프로그램에서 N개의 연결고리를 유지할 수 있다.

1개의 Connection에서 여러 개의 Statement를 생성해서 SQL문을 전송할 수 있다.

1개의 SQL문에서는 결과는 오직 1가지만 가져야 한다.

JDBC 프로그래밍

- JDBC 프로그래밍 순서
  - 드라이버 로딩
  - 데이터베이스 연결
  - SQL문 전송
  - 결과 처리
  - 종료

17-4

JDBC 프로그래밍

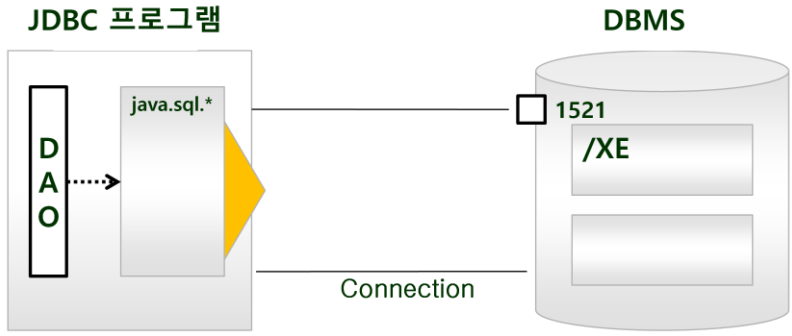
- 드라이버 로딩
  - Class.forName("oracle.jdbc.driver.OracleDriver");
  - DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
  - new oracle.jdbc.driver.OracleDriver();
  - > java –D drivers= oracle.jdbc.driver.OracleDriver JDBCTest

17-5

JDBC 프로그래밍

- 데이터베이스 연결

```
String uri="jdbc:oracle:thin:@127.0.0.1:1521:XE"
String id="javaid";
String pw="javapw";
Connection conn=DriverManager.getConnection(uri, id, pw);
```

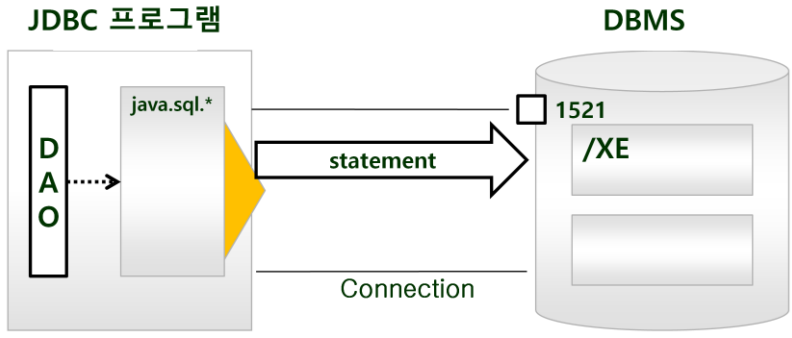


17-6

JDBC 프로그래밍

- **SQL문 전송**

```
Statement stmt=conn.createStatement();
String sql="select department_id, department_name from departments";
ResultSet rs=stmt.executeQuery(sql);
```

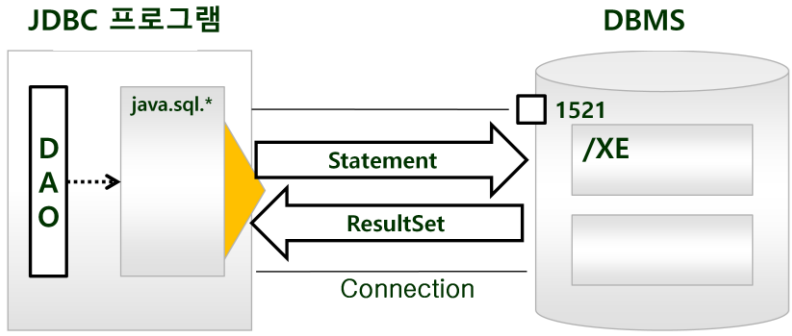


17-7

JDBC 프로그래밍

- **SQL문 전송**

```
ResultSet rs=stmt.executeQuery(sql);
while(rs.next()){
   System.out.println(rs.getInt(1)+" : "+ rs.getString(2));
 }
```



17-8

JDBC 프로그래밍

- 종료 : 자원 반환

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

17-9

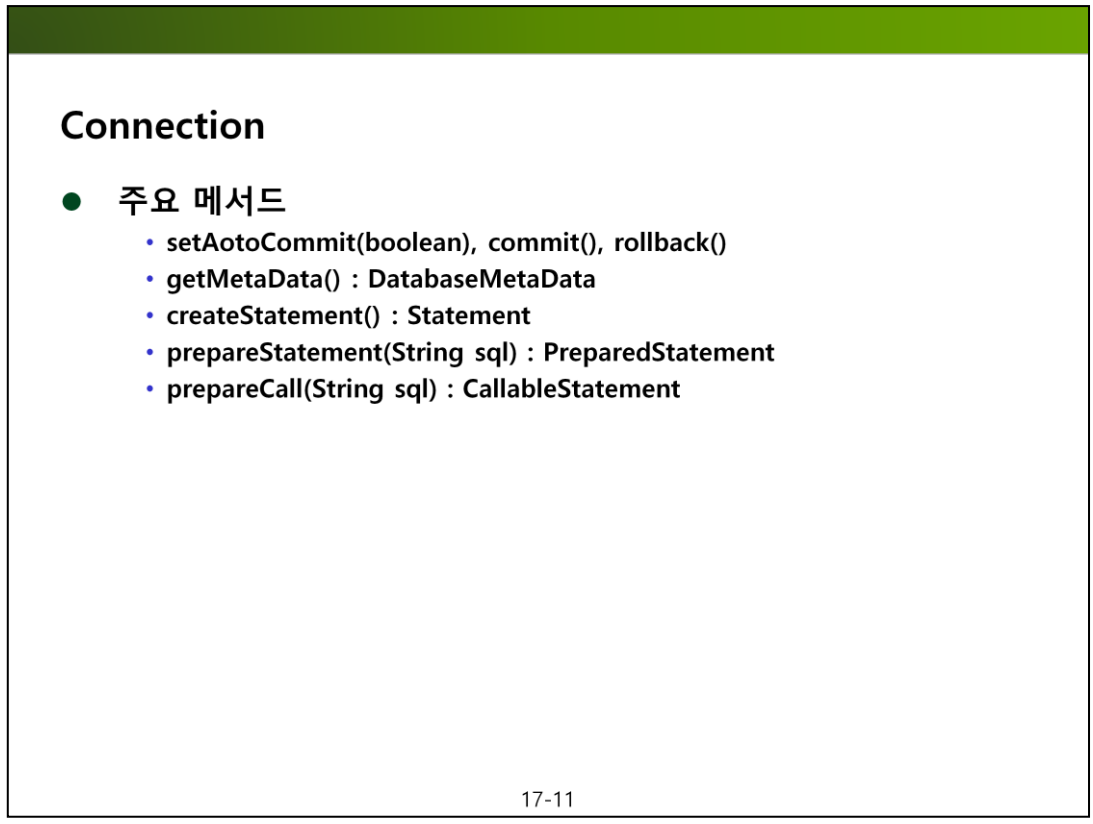
• JDBC 프로그래밍

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class JDBCTest {
    public static void main(String[] args) throws SQLException {
        //1.
        String driver="oracle.jdbc.driver.OracleDriver";
        Class.forName(driver);
        System.out.println("1. driver loading ok...");

       //2. url, id, pw
        String url="jdbc:oracle:thin:@127.0.0.1:1521:XE";
        String id="hr";
        String pw="hr";
        Connection conn=DriverManager.getConnection(url, id, pw);
        System.out.println("2. DBMS 접속 완료");

        //3.4.sql 전송후 처리
        Statement stmt=conn.createStatement();
        String sql="select department_id, department_name from
departments";
        ResultSet rs=stmt.executeQuery(sql);
        while(rs.next()){
            System.out.println(rs.getInt(1)+" : "+ rs.getString(2));
        }//

        //5.  자원 반환
        rs.close();
        stmt.close();
        conn.close();
    }
}
```



• DBMS 정보

```
Connection conn=DriverManager.getConnection(url, id, pw);
System.out.println("2. DBMS 접속 완료");
System.out.println("commit : "+ conn.getAutoCommit());

DatabaseMetaData dbmd=conn.getMetaData();
System.out.println(dbmd.getDatabaseProductName());
System.out.println(dbmd.getDriverName()+" : "+ dbmd.getDriverVersion());
```

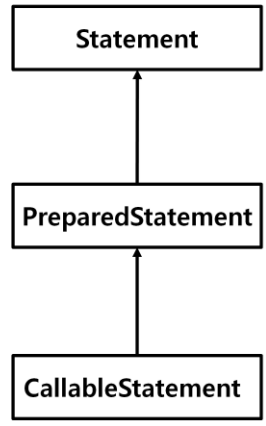
Statement

- 주요 메서드
  - execute(String sql) : boolean
    - DDL(create, alter, drop 등)문 전송
  - executeUpdate(String sql) : int
    - DML(insert, delete, update)문 전송
  - executeQuery(String sql) : ResultSet
    - select문 전송

17-12

Statement

- Statemet API



17-13

Statement

- Statemet
  - SQL문을 문자열로 그대로 사용
  - 실행시 SQL문을 체크하기 때문에 에러 찾기가 어려움
  - SQLInjection이 발생

```
String sql="select name from member where id='"+memberId+"'";
Statement stmt=conn.createStatement();
ResultSet rs=stmt.executeQuery(sql);
```

Statement

- PreparedStatement
  - SQL문을 문자열에 값이 들어갈 자리에 ?로 처리
  - ?에 setXxx(위치, 값)으로 초기화
  - SQLInjection이 발생하지 않는다.

```
String sql="select name from member where id=?";
PreparedStatement pstmt=conn.prepareStatement(sql);
pstmt.setString(1, memberId);
ResultSet rs=pstmt.executeQuery();
```

Statement

- Callabletatement
  - DBMS에서 제공되는 함수 호출
  - 함수 매개인자로 SQL문의 값을 ?로 처리
  - ?에 들어갈 값을 setXxx(위치, 값)로 초기화
  - DBMS에 따라 다르다.

```
String sql="{ ?=call selectMember(?) }";
CallableStatement cstmt=conn.prepareCall(sql);
cstmt.setString(1, memberId);
cstmt.registerOutParameter(1, java.sql.Types.VARCHAR);
cstmt.execute();
System.out.println("name : "+ cstmt.getString(1));
```

ResultSet

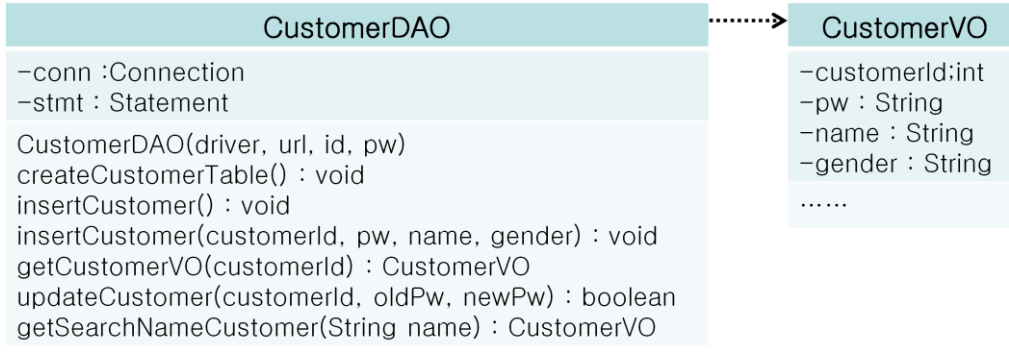
- 주요 메서드
  - getXxx() → getString(컬럼인덱스), getString("필드명")
  - first() : boolean
  - last() : boolean
  - next() : boolean
  - previous() : boolean
  - getMetaData() : ResultSetMetaData

17-17

• 검색테이블의 metadata

```
ResultSetMetaData rsmd=rs.getMetaData();
System.out.println("컬럼 : "+ rsmd.getColumnCount());
    while(rs.next()){
        for(int i=1; i<=rsmd.getColumnCount(); i++){
            System.out.println("컬럼type : "+ rsmd.getColumnType(i));
            switch(rsmd.getColumnType(i)){
                case Types.CHAR :
                    System.out.println("char : "+ rs.getString(i));
                    break;
                case 2:
                    System.out.println("정수 : "+ rs.getString(i));
                    break;
                case Types.VARCHAR:
                    System.out.println("varchar : "+ rs.getString(i));
                    break;
             }
        }//for
    }//while
```

JDBC 응용 프로그래밍



LAB

17-19