SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 삼성 청년 SW 아카데미

MySQL

SAMSUNG SW ACADEMY FOR YOUTH

# 목차

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# JDBC(Java DataBase Connectivity)

# JDBC (Java DataBase Connectivity)

Confidential

## ✓ JDBC (Java DataBase Connectivity) 란?

- 자바 프로그래밍 언어로 만들어진 클래스와 인터페이스로 이루어진 API로서 ANSI SQL(1999)를 지원
- SQL문을 실행할 수 있는 함수 호출 인터페이스

## ✓ JDBC 특징

- DBMS 종류에 독립적인 자바 프로그래밍 가능
- 데이터베이스가 달라지더라도 동일한 API를 사용하게 해줌 (드라이버 및 URL만 수정 하면 가능)
- 자바가 가지는 플랫폼에 독립적이라는 특성과 DBMS에 독립적인 특성을 가짐

# JDBC (Java DataBase Connectivity)

Confidential

## ✓ JDBC 기능

- 데이터베이스에 연결 설정
- SQL문장을 DBMS에 전송
- SQL문장 전송 후 결과를 응답 받을 수 있음

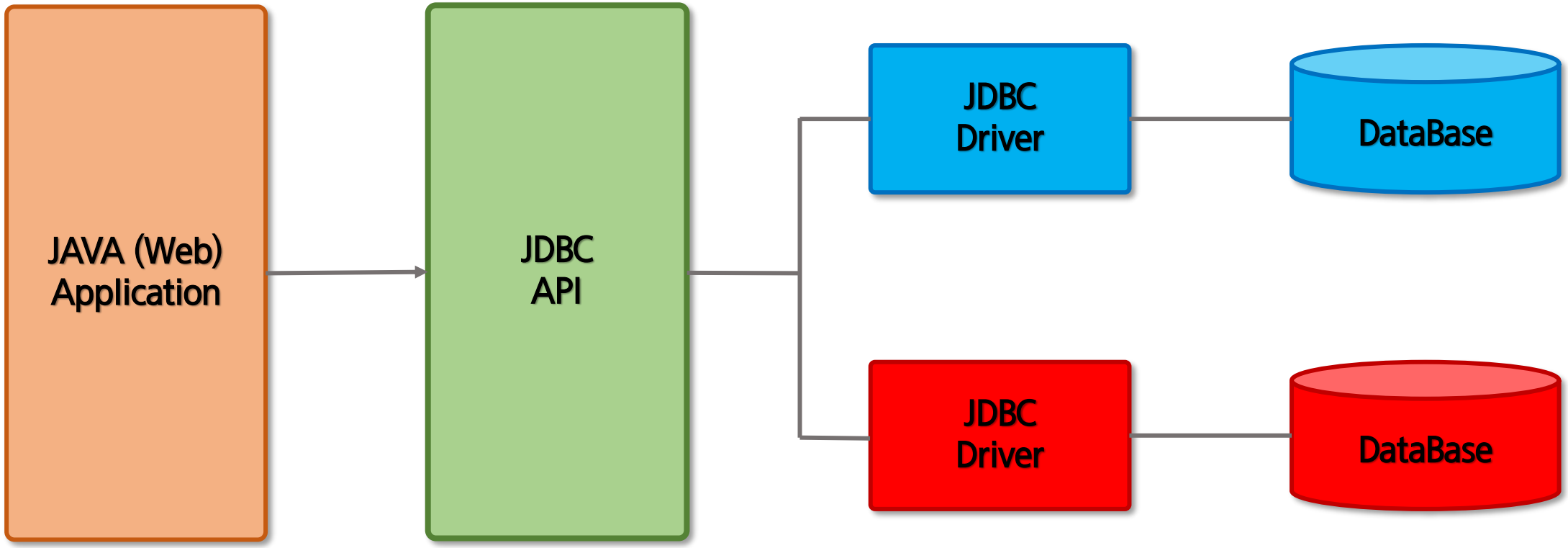
## ✓ JDBC Interface

- Database를 만드는 업체에게 제공되는 인터페이스
  - 업체에게 제공되는 인터페이스를 각각의 DBMS업체들이 구현해 놓은 것으로서 이것이 바로 드라이버이다.
- 프로그래머에게 제공되는 인터페이스
  - SQL 패키지가 제공하고 있는 라이브러리로서 프로그래머는 이 라이브러리를 기반으로 DB 프로그램을 작성할 수 있다.

# JDBC (Java DataBase Connectivity)

Confidential

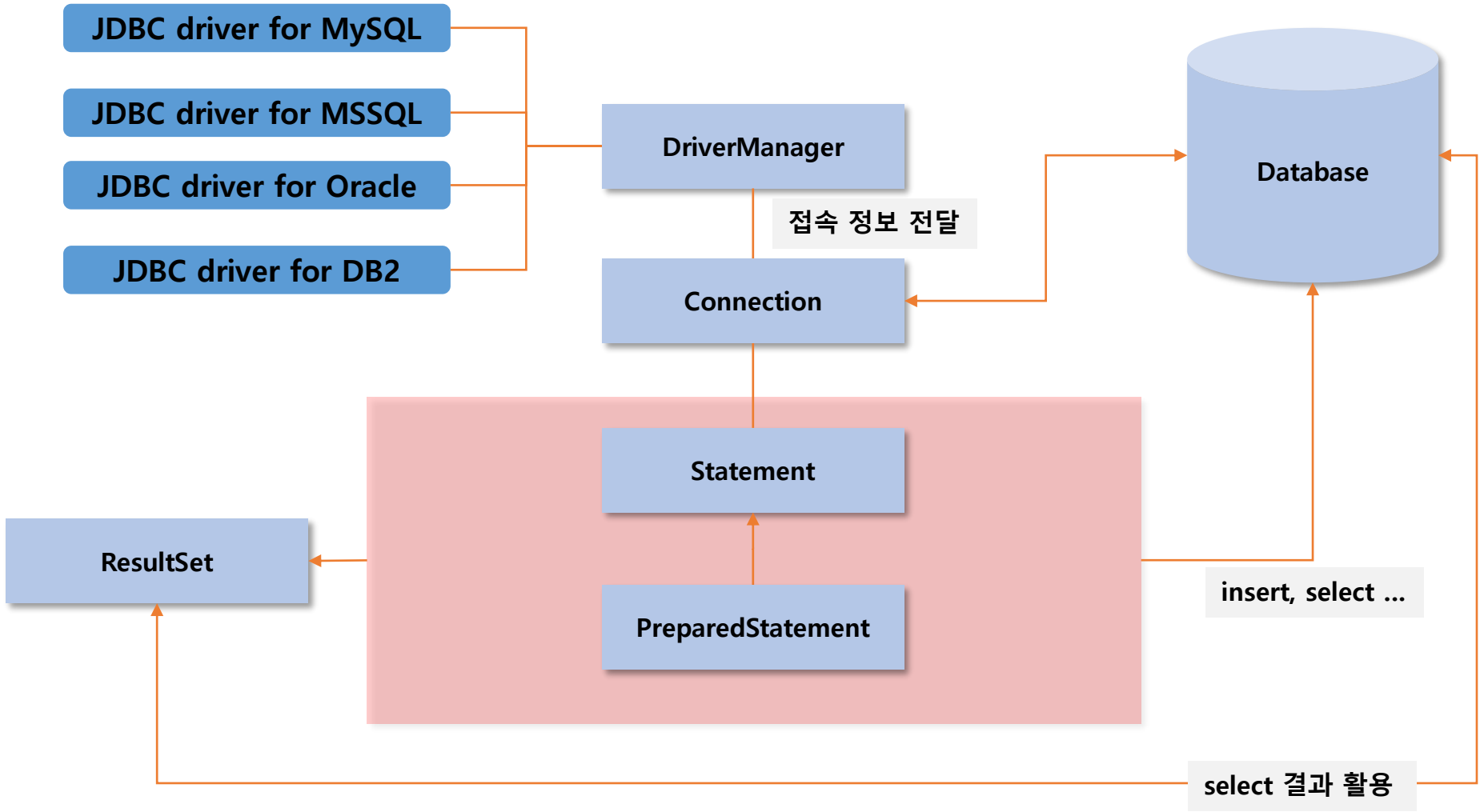
✓ JDBC



# JDBC (Java DataBase Connectivity)

Confidential

✓ JDBC 관련 클래스들



# JDBC (Java DataBase Connectivity)



## ✓ JDBC API : java.sql package

- **DriverManager**
  - JDBC API에서 제공되는 구상(Concrete) 클래스
    - 구상 클래스란 new 키워드로 객체를 생성할 수 있는 클래스 (추상 클래스와 대조)
  - JDBC Driver 인터페이스를 구현한 객체들을 관리
  - JDBC URL을 이용해 메모리에 로드 된 Driver 검색 -> connection 연결

# JDBC (Java DataBase Connectivity)

Confidential

✓ JDBC API : java.sql package

- Driver (interface)
  - 드라이버에 대한 정보를 가지고 있음
  - 모든 드라이버가 반드시 구현해야 하는 인터페이스
  - 드라이버의 버전이나 연결에 대한 정보를 알아볼 수 있는 메서드를 가지고 있음

Java Platform, Standard Edition 8 API Specification

# JDBC (Java DataBase Connectivity)



✓ **JDBC API : java.sql package**

- **Connection (interface)**
  - 데이터베이스에 대한 하나의 세션을 표현한다.
    - ✓ 세션은 하나의 클라이언트가 서버에 요청을 하기 위해 연결을 맺은 상태를 의미
  - DriverManager 클래스의 getConnection() 메서드를 이용하여 얻어 올 수 있음
  - 기본적으로 setAutoCommit(true)로 설정됨
  - 개발자가 원하는 경우에 commit을 해주고 싶거나 트랜잭션이 아주 중요한 부분에 있어서 RollBack 처리를 하고자 할 경우에는 setAutoCommit(false)로 설정
    - ✓ 단, 이 경우에는 SQL문을 수행할 때 마다 명시적으로 commit()을 호출해야 함

# JDBC (Java DataBase Connectivity)



## ✓ JDBC API : java.sql package

- Statement (interface)
  - SQL문장을 실행하고 그것에 대한 결과 값을 가져오기 위해 사용
  - public boolean execute(String sql) throws SQLException
    - ✓ 특별히 SQL문을 구분하지 않고 DML(delete, update, insert), Query(select), DDL(create, drop)등을 수행할 수 있다. 결과가 ResultSet이면 true이고 결과가 DML이거나 특별한 결과가 없으면 false를 리턴
  - public ResultSet executeQuery(String sql) throws SQLException
    - ✓ SELECT 쿼리문을 실행할 때 사용
  - public int executeUpdate(String sql) throws SQLException
    - ✓ 주로 DML(INSERT, UPDATE, DELETE) 쿼리문을 실행할 때 사용
  - 쿼리문을 실행할 수 있는 객체 - 대부분의 SQL처리
  - SQL Injection에 취약하므로, 실무에서는 사용하지 말 것

## ✓ Statement 사용 시의 문제점 - SQL Injection

- **Statement는 주어진 SQL을 그대로 전달하는 기능을 이용한 공격**

```
String query = "select * from userInfo where username='admin' and password=' ' or '1'='1' ";
```


  - password와 비교하는 문장에 임의의 SQL 삽입 가능

- **PreparedStatement 사용**

```
String sql= "select * from userinfo where username=? and password=?";

PreparedStatement pstmt = con.prepareStatement(sql);

pstmt.setString(1, "admin");
pstmt.setString(2, " ' ' or '1' = '1' ");

ResultSet rset = pstmt.executeQuery ();
```

값 들이 인자로 처리됨

쿼리 준비 완료

인자 설정

쿼리 실행

# JDBC (Java DataBase Connectivity)



✓ **JDBC API : java.sql package**

- **PreparedStatement (interface)**
  - 동일한 SQL 문장이 여러 번 반복적으로 수행될 때 사용하는 객체.
  - 대용량의 문자나 바이너리 타입의 데이터(이미지나 사운드 등)를 저장하기 위해서도 사용될 수 있다.
  - SQL 문장이 미리 컴파일 되어 PreparedStatement 객체에 저장된다.
  - 여러 번 반복 수행 시 clearParameters() 메소드를 이용해 Statement에 남겨진 값을 초기화 한다.
  - public ResultSet executeQuery() throws SQLException
    - ✓ SELECT 쿼리문을 실행할 때 사용
  - public int executeUpdate() throws SQLException
    - ✓ 주로 DML(INSERT, UPDATE, DELETE) 쿼리문을 실행할 때 사용

# JDBC (Java DataBase Connectivity)

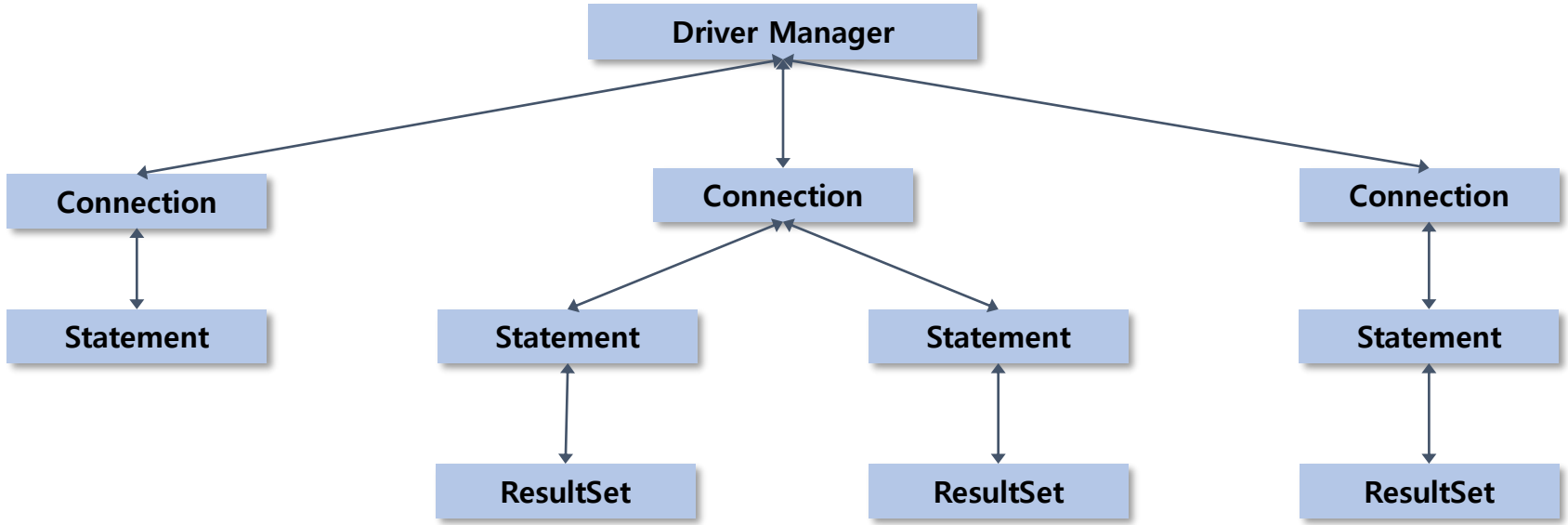


- ✓ **JDBC API : java.sql package**
  - **ResultSet (interface)**
    - 쿼리에 대한 결과값 처리
    - ResultSet 객체의 커서는 첫번째 레코드보다 바로 이전을 가르킨다.
    - next()
      - ✓ ResultSet 객체의 커서를 이동
    - getXXX(index or name) 메소드를 이용하여 데이터를 얻을 수 있다.
      - ✓ getString(index or name);
      - ✓ getInt(index or name);
      - ✓ getDate(index or name);
      - ✓ …

# JDBC (Java DataBase Connectivity)

Confidential

## ✓ JDBC 주요 인터페이스의 사용




- DriverManager는  여러 개의 Connection을 관리
- Connection은 Statement를 통해 동시에 여러 개의 질의(SQL 전송) 가능
- 하나의 Statement는 동시에 하나의 ResultSet 만을 가질 수 있다.
- Connection, Statement, ResultSet은 사용 후 close 해야 할 resource

# JDBC (Java DataBase Connectivity)

Confidential

## ✓ JDBC Programming 개발 순서

1. JDBC Driver Loading
2. DBMS와 연결 (Connection 생성)
3. SQL 실행 준비 (Statement 또는 PreparedStatement 생성)
4. SQL 실행
   - DML (INSERT, UPDATE, DELETE)
   - DQL (SELECT) - 실행 결과는 ResultSet 객체에 담겨있음
5. DBMS 연결 끊기

# JDBC (Java DataBase Connectivity)

Confidential

## ✓ JDBC Programming 개발 순서

1. JDBC Driver Loading
   - 데이터베이스에 접속하기 위해 애플리케이션의 JVM 메모리에 특정 JDBC 드라이버 클래스를 적재
   - Class.forName( JDBC Driver ClassName );
   - 각 DataBase별 Driver Class
     - ✓ MySQL : com.mysql.cj.jdbc.Driver
     - ✓ Oracle : oracle.jdbc.driver.OracleDriver
     - ✓ MSSQL : com.microsoft.sqlserver.jdbc.SQLServerDriver

## ✓ JDBC Programming 개발 순서

2. DBMS와 연결 (Connection 생성)

- DriverManager 클래스를 이용하여 URL 형태로 주어진 데이터베이스에 대한 접속을 요청
- Connection conn = DriverManager.getConnection(URL, id, password);
- JDBC URL은 드라이버 고유의 방식으로 개별적인 데이터베이스를 식별
  - ✓ MySQL : jdbc:mysql://HOST:PORT/DBNAME[?param1=value1&param2=value2&..]
  - ✓ Oracle : jdbc:oracle:thin:@HOST:PORT:SID
  - ✓ MSSQL : jdbc:sqlserver://HOST:PORT;databaseName=DB

## ✓ JDBC Programming 개발 순서

3. SQL 실행 준비 (Statement 또는 PreparedStatement 생성)

- String sql = “select, insert, update, delete, …”;
- Statement 생성
  - ✓ Statement stmt = conn.createStatement();
- PreparedStatement 생성
  - ✓ PreparedStatement pstmt = conn.prepareStatement(sql);

## ✓ JDBC Programming 개발 순서

4. SQL 실행 (executeUpdate, executeQuery)

- Statement
  - ✓ DML
    - int cnt = stmt.executeUpdate(sql);
  - ✓ Select
    - ResultSet rs = stmt.executeQuery(sql);
- PreparedStatement
  - ✓ SQL문의 치환변수 값 설정 : pstmt.setXXX(index, val);
  - ✓ DML
    - int cnt = pstmt.executeUpdate();
  - ✓ Select
    - ResultSet rs = pstmt.executeQuery();

## ✓ JDBC Programming 개발 순서

4. SQL 실행 (executeUpdate, executeQuery)

- ResultSet
  - ✓ ResultSet을 얻어온 후 rs.next()를 실행해야 한다.
    - select의 결과가 반드시 하나가 나오는 경우 : rs.next();
    - select의 결과가 하나 또는 못 얻어 오는 경우 : if(rs.next) { }
    - select의 결과가 여러 개가 나올 수 있는 경우 : while(rs.next()) { }
  - ✓ 값 얻기.
    - String str = rs.getString(index or name);
    - int cnt = rs.getInt(index or name);

## ✓ JDBC Programming 개발 순서

5. DBMS 연결 끊기

- 모든 작업이 끝난 경우에는 ResultSet, Statement(PreparedStatement), Connection 객체의 close() 메서드를 이용하여 작업을 종료한다. (연결한 순의 역순으로 종료)
- Connection은 상당한 Overhead를 가져온다. 따라서 최적화된 상태를 유지하기 위해서는 **반드시** Connection을 닫아 주어야 한다.
  - ✓ rs.close();
  - ✓ pstmt.close();
  - ✓ conn.close();

# JDBC (Java DataBase Connectivity)

Confidential

## ✓ JDBC에서 Transaction 관리

- Auto-Commit 사용 중지
- 하나의 Transaction을 수행했다면 명시적으로 commit 하기
- Transaction을 활용하여 데이터 무결성 유지하기
- rollback 활용하기