Getting Started with PgSQL

JDBC Basic

wisoft

차례

Introduction

About JDBC

- About JDBC
  - 자바를 이용하여 데이터베이스 관련 작업을 할 수 있도록 해주는 자바 프로그래밍 인터페이스를 위한 API(Application Programming Interface) 규격

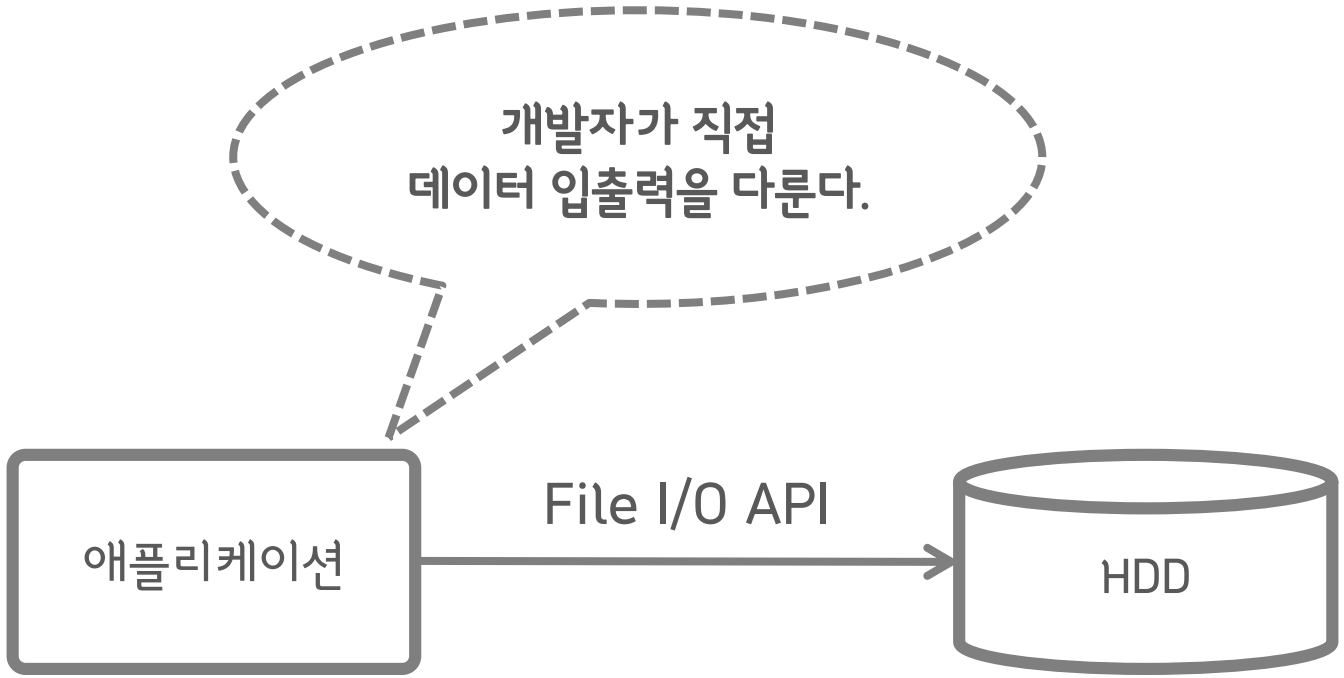
  - Interface로 정의
    - 데이터베이스 제조사들은 자신들의 데이터베이스에 접속해서 작업할 수 있도록 Interface를 구현하여 드라이버 클래스를 만들어서 프로그래머들에게 제공
    - 표준적인 JDBC의 사용법만 익히면 JDBC를 제공하는 모든 데이터베이스에 대해 프로그래밍을 할 수 있음
    - 프로그래밍 언어에 따라 문법이 조금 다를 수 있음

History

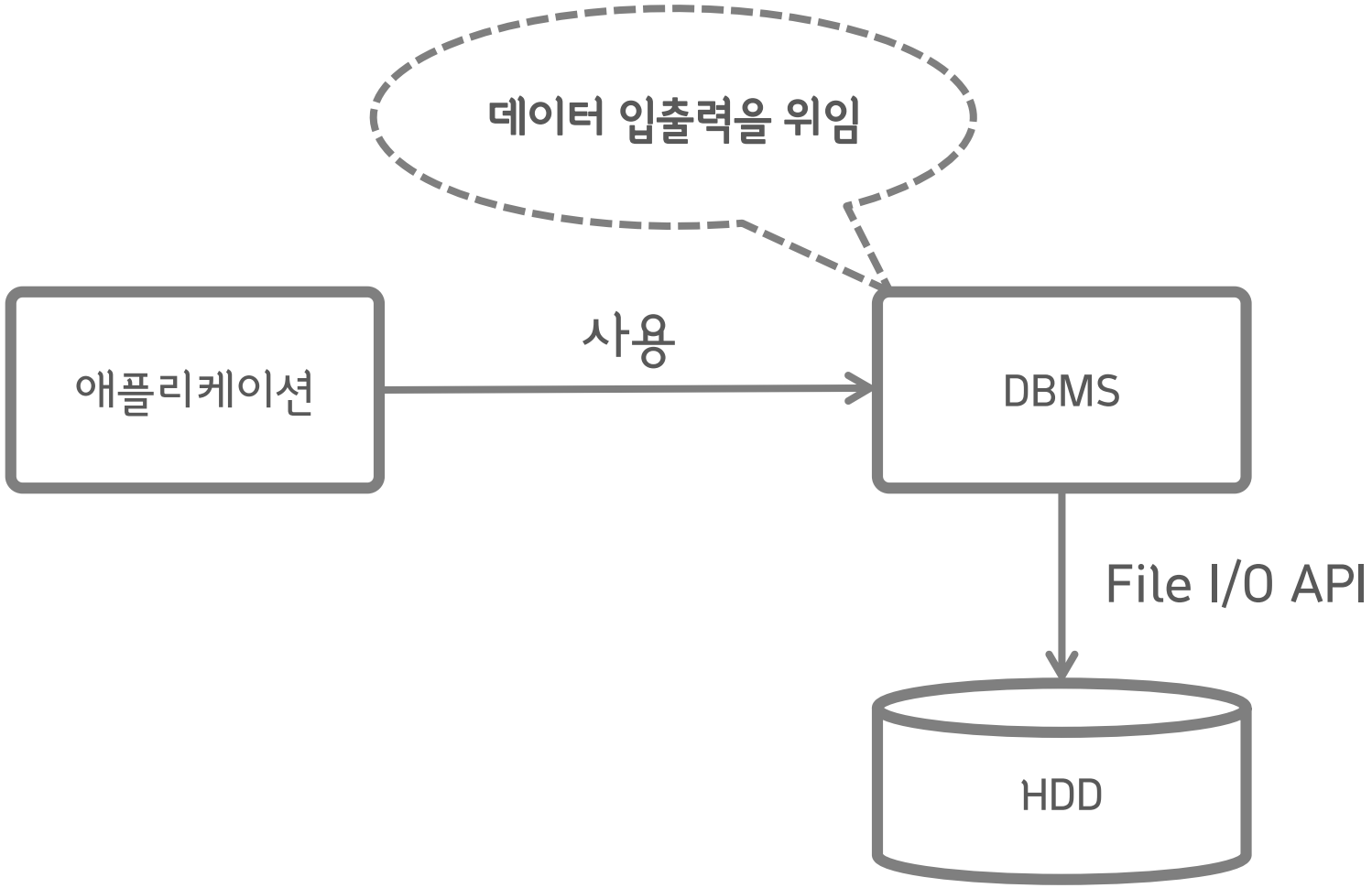
- JDBC가 탄생한 배경
  - Microsoft社의 Open Database Connectivity(ODBC)의 영향을 받아 제작
  - 자바는 필요하다면 ODBC를 직접 이용할 수도 있음
    - [문제점 1] C언어로 제작된 ODBC를 이용하려면 여러 가지 복잡한 작업이 필요함
    - [문제점 2] 자바의 객체지향적인 언어적 장점을 살릴 수 없는 문제가 발생함
  - 순수 자바 기술이 적용된 JDBC를 만들게 되었음

Driver Type

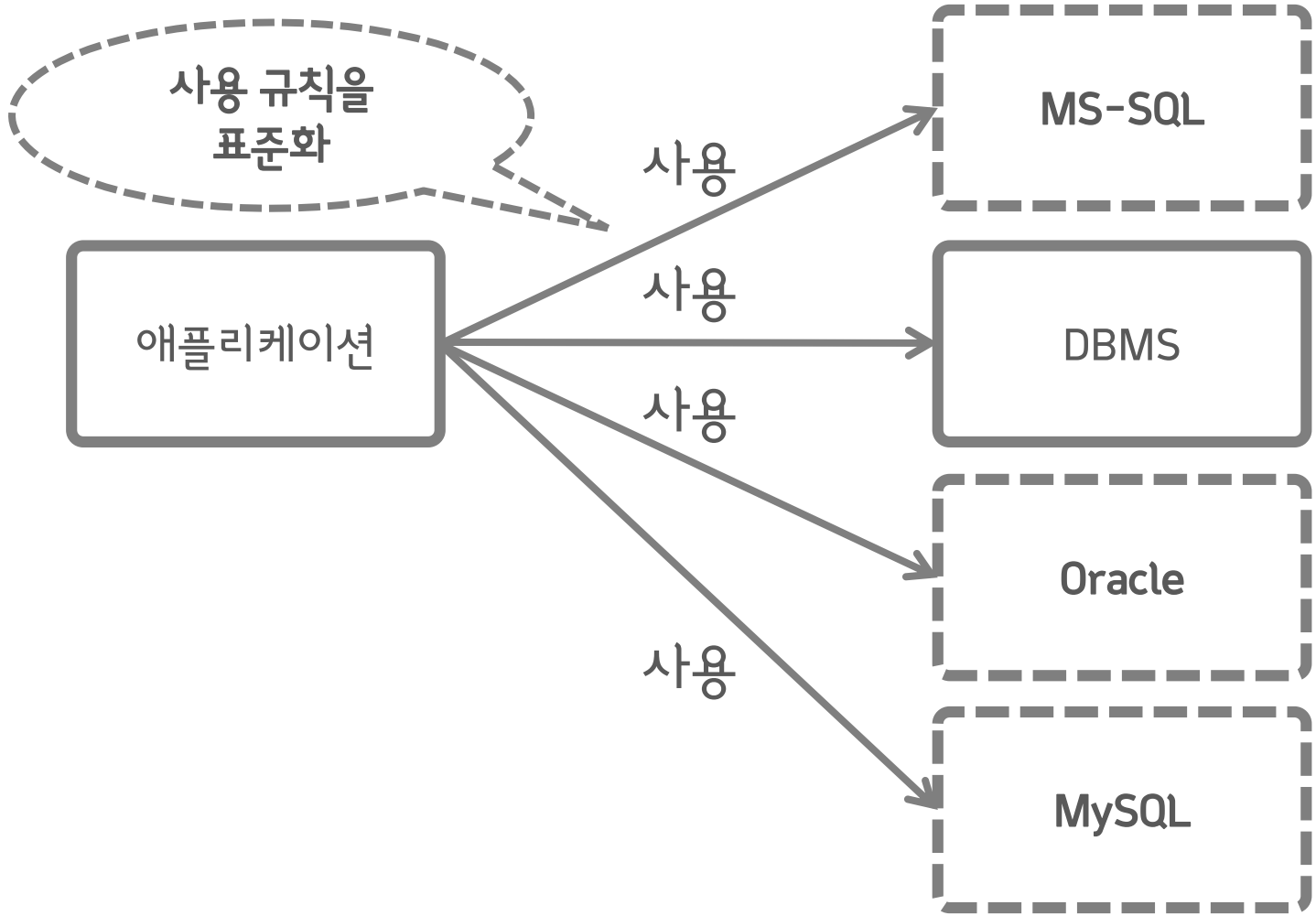
DBMS 등장 전



DBMS 등장



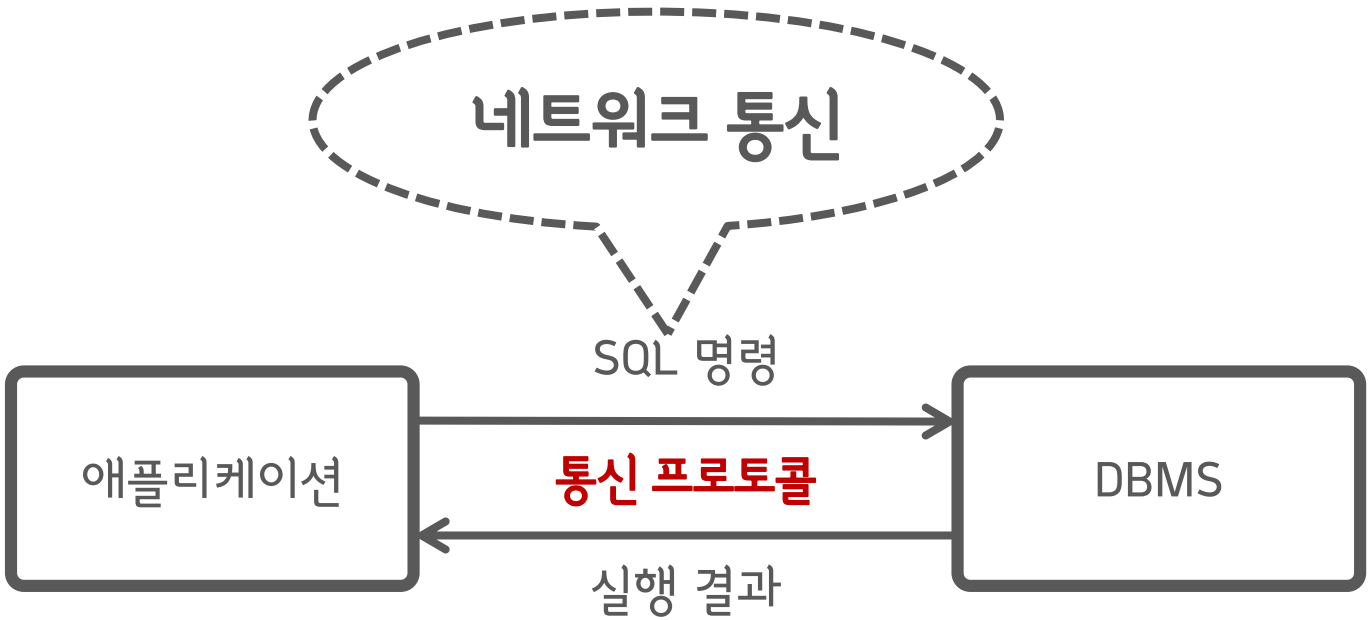
DBMS 사용 규칙



SQL 전달 및

데이터 수신 (1/3)

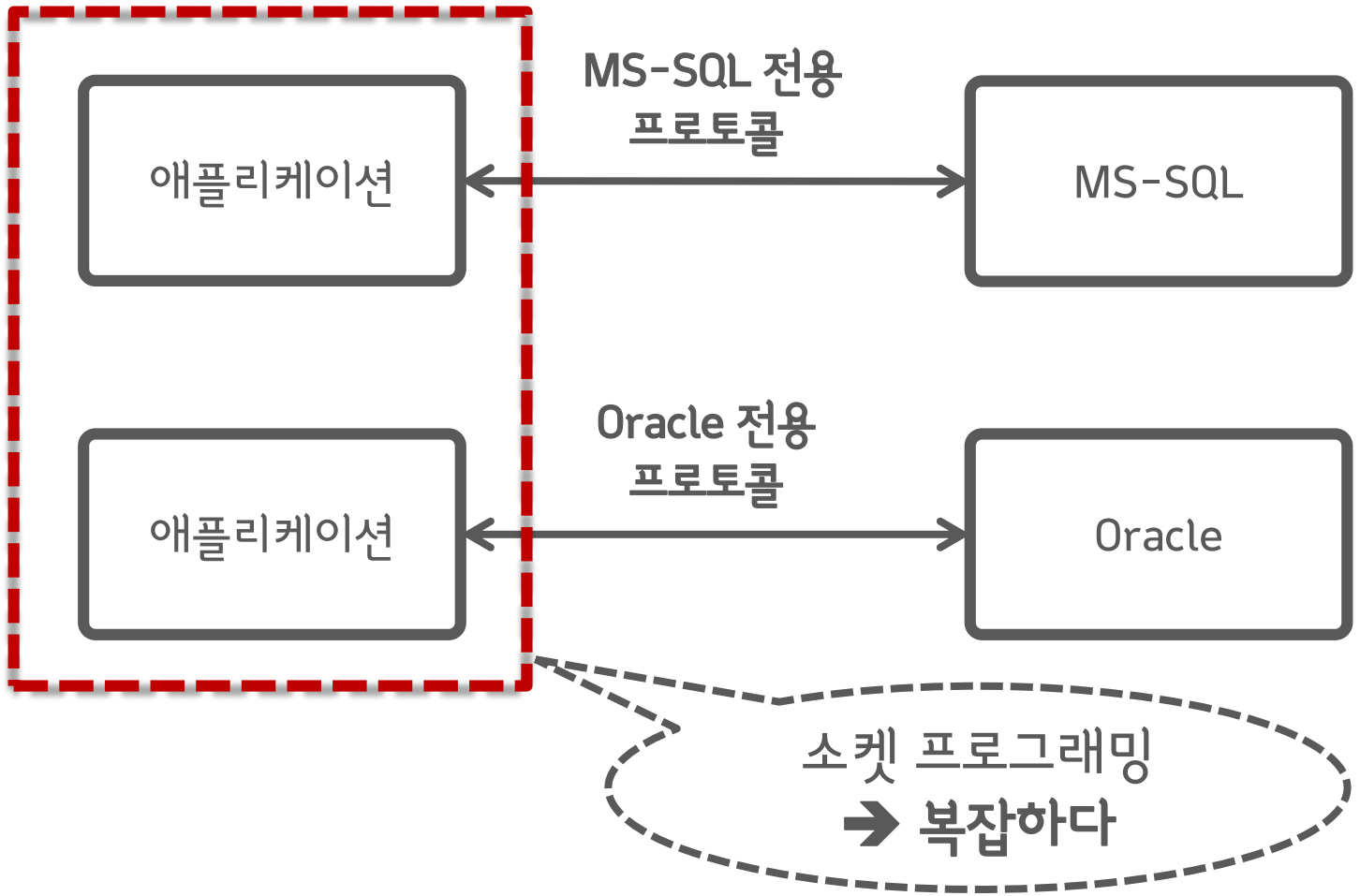
- SQL 전송 프로토콜



SQL 전달 및

데이터 수신 (2/3)

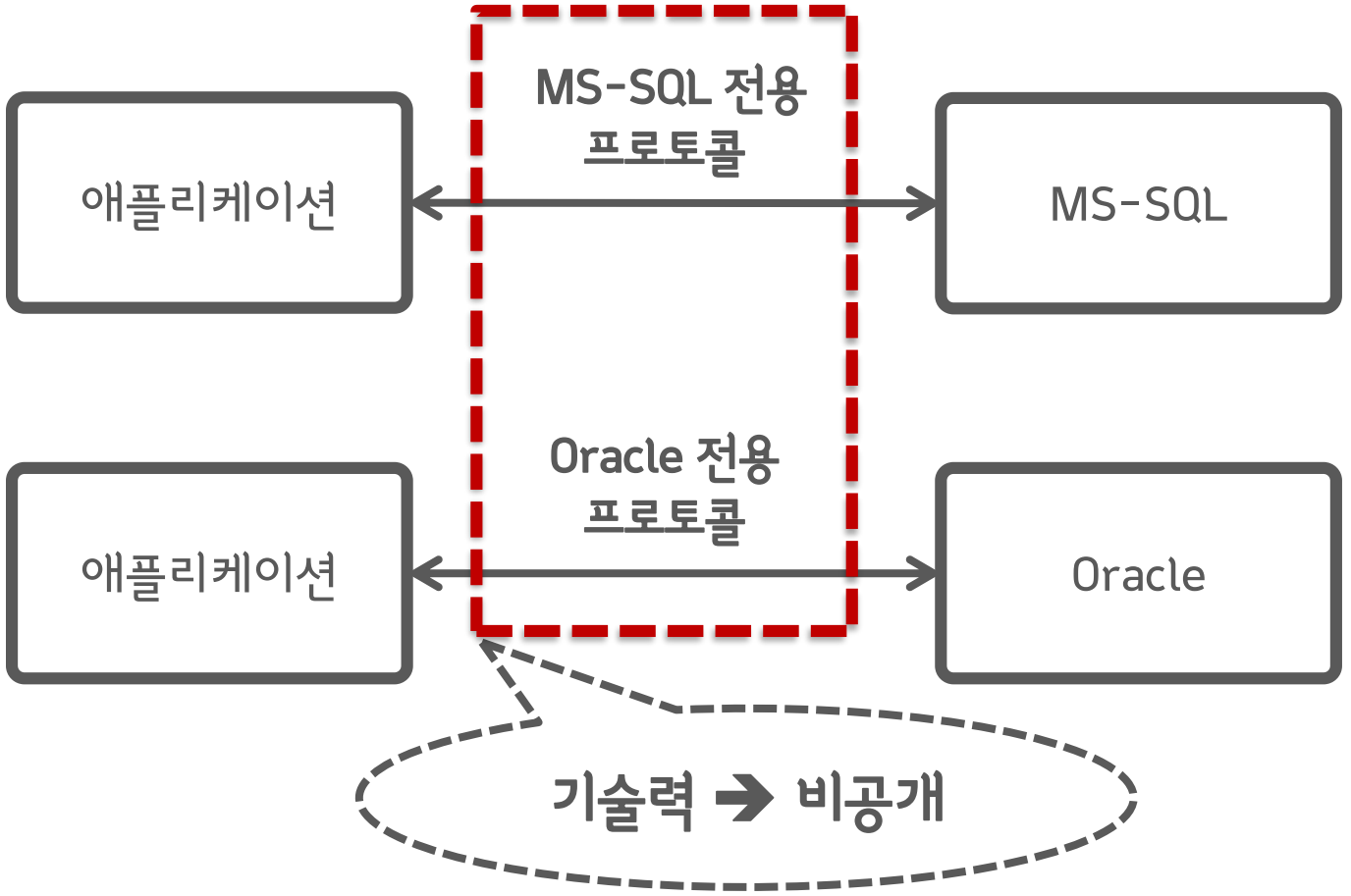
- DBMS 전용 통신 프로토콜

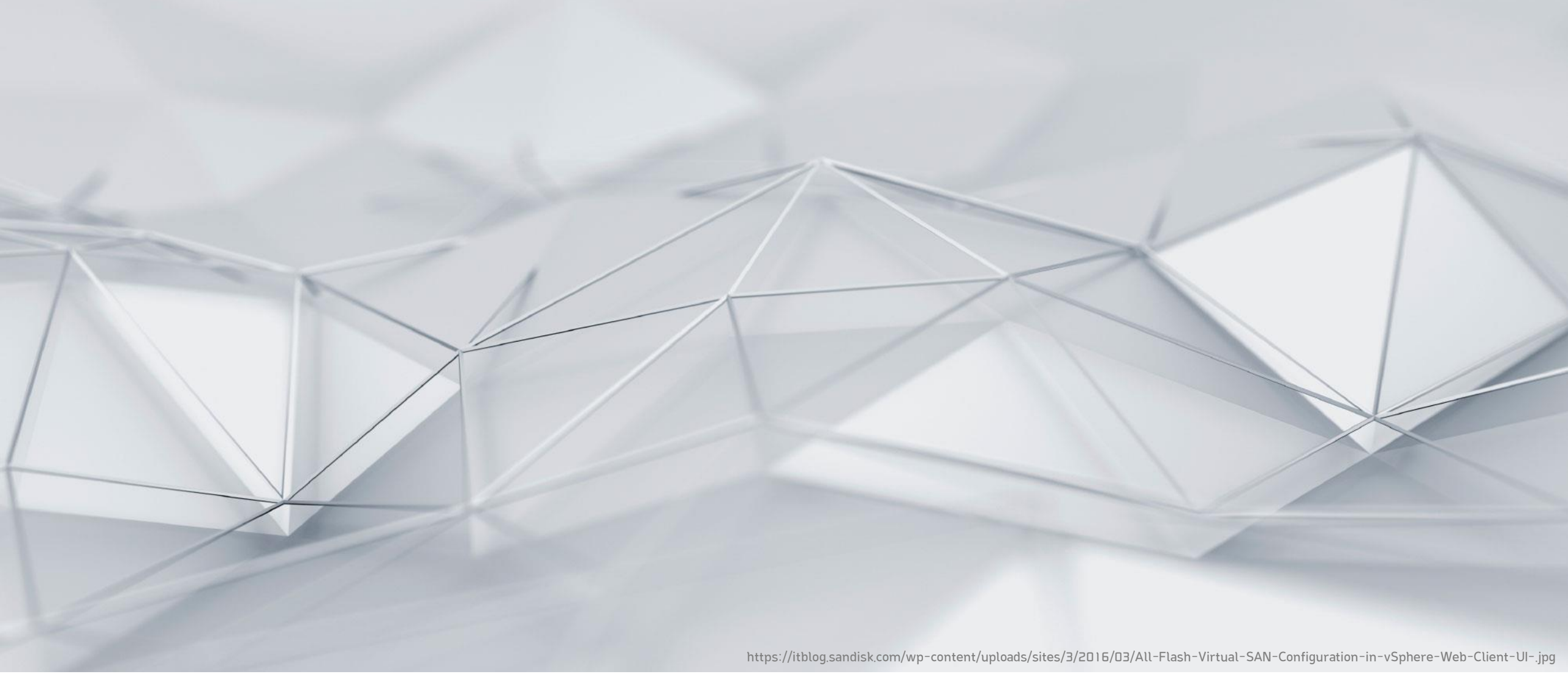


SQL 전달 및

데이터 수신 (3/3)

- DBMS 전용 통신 프로토콜





https://itblog.sandisk.com/wp-content/uploads/sites/3/2016/03/All-Flash-Virtual-SAN-Configuration-in-vSphere-Web-Client-UI-.jpg

JDBC 구조

Overview

- JDBC의 구조
  - 프로그래머가 이용하기 위한 API 명세인 Interface
    - Interface의 핵심부 - DriverManager Class
    - Driver Class - 다양한 드라이버를 자바 응용 프로그램과 연결시켜 주는 역할
  - 실제 데이터베이스와 연결을 책임지는 드라이버(Driver)
    - JDBC 인터페이스에 맞추어 해당 DBMS에서 JDBC 관련 API 호출이 가능하도록 관련 인터페이스와 클래스를 구현한 클래스 라이브러리

JDBC Driver Types

- 다양한 단체와 업체에서 만들어지므로 다양한 방식으로 제작 (4가지 타입)
  - [TYPE 1] JDBC-ODBC Bridge Driver
  - [TYPE 2] Native-API Driver

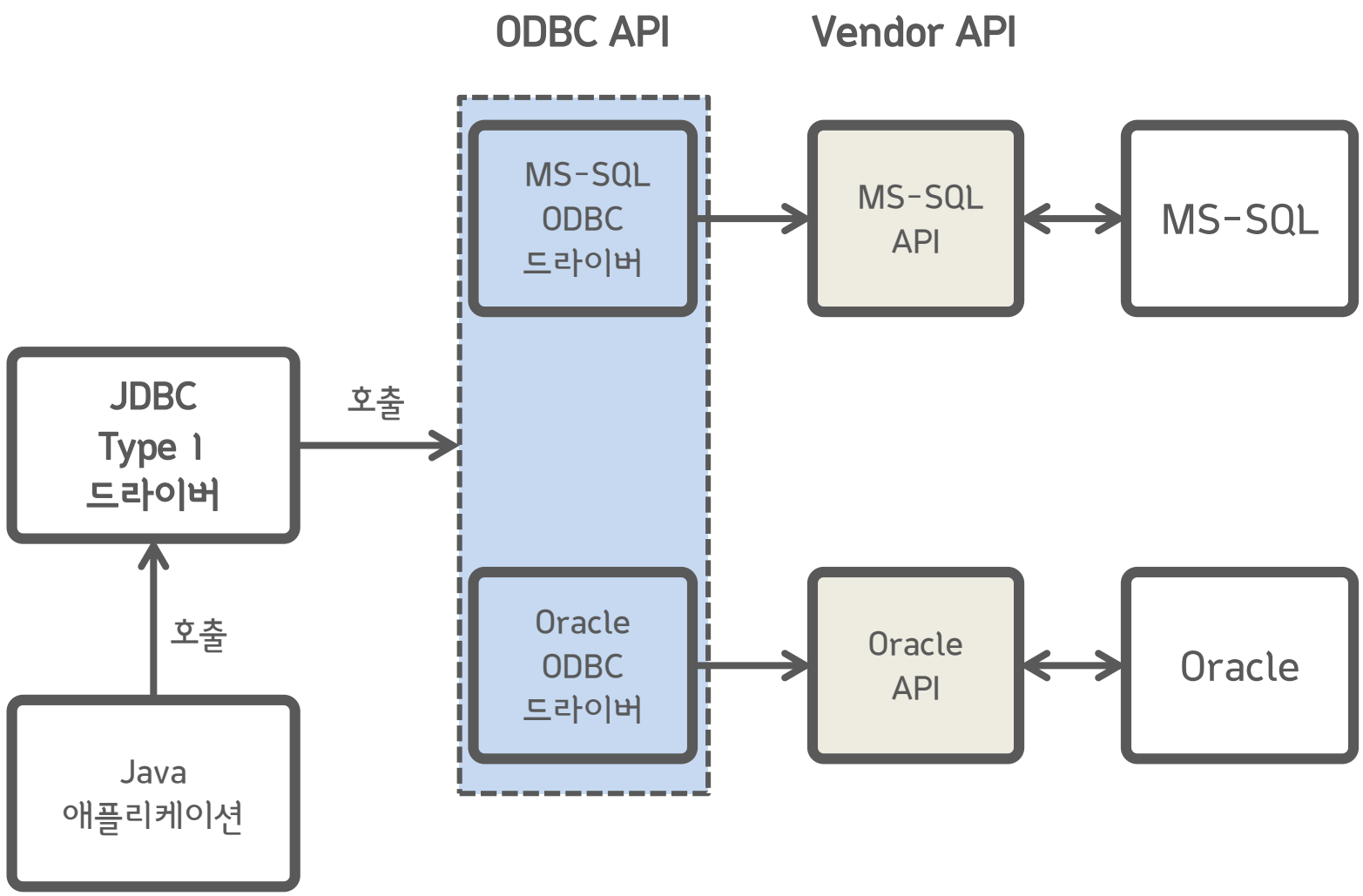
    Oracle OCI Driver
  - [TYPE 3] JDBC Network Protocol Driver (Middleware Driver)
  - [TYPE 4] DBMS Protocol Driver (Pure Java Driver)

    Vendor-specific database protocol, Oracle Thin Driver

JDBC Driver Types 1

JDBC Type 1 특징

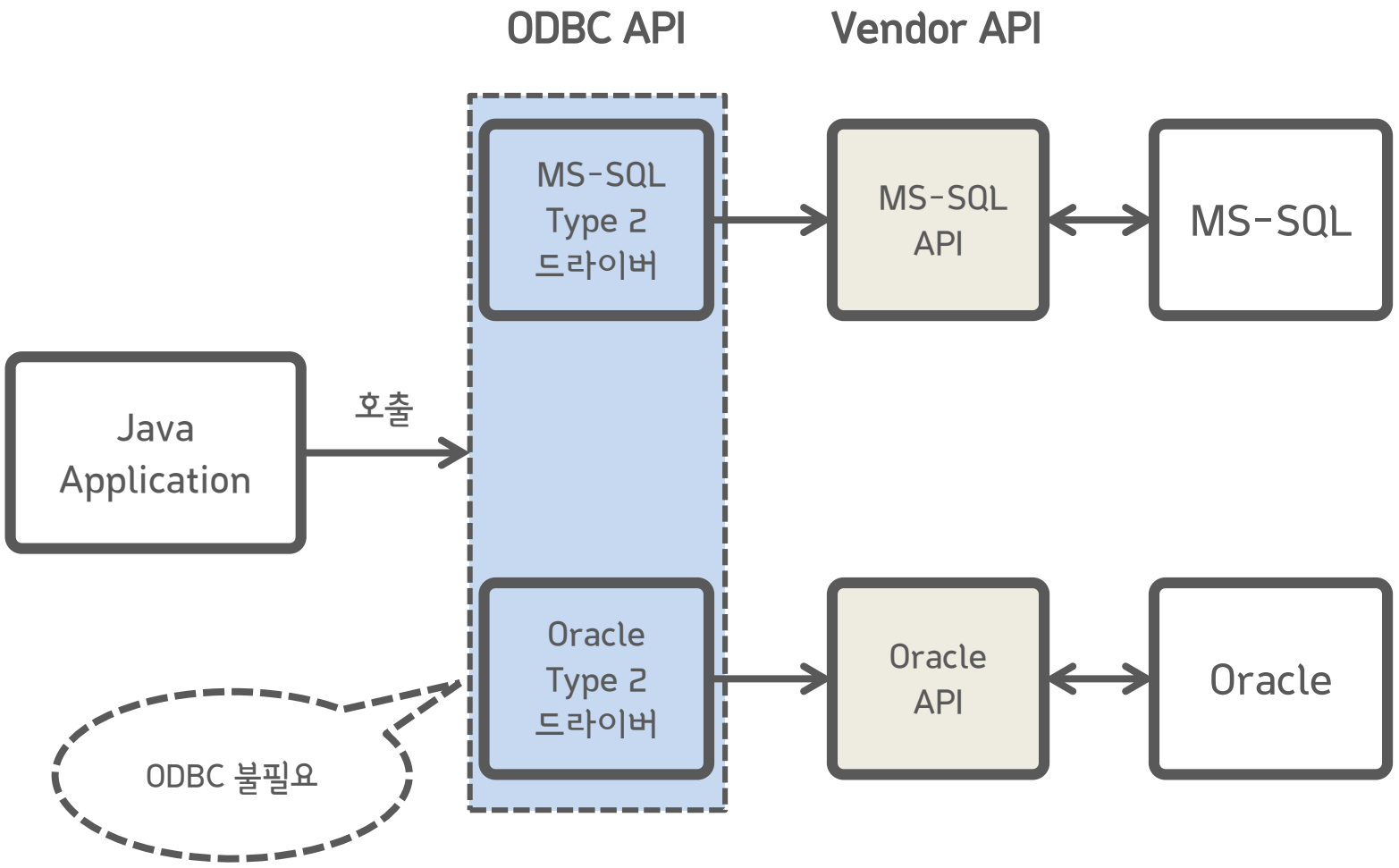
- JRE에 기본으로 포함
- ODBC 드라이버 사용
- 속도 느리다
- Excel, Text 데이터에 접근할 때 유용



JDBC Driver Types 2

JDBC Type 2 특징

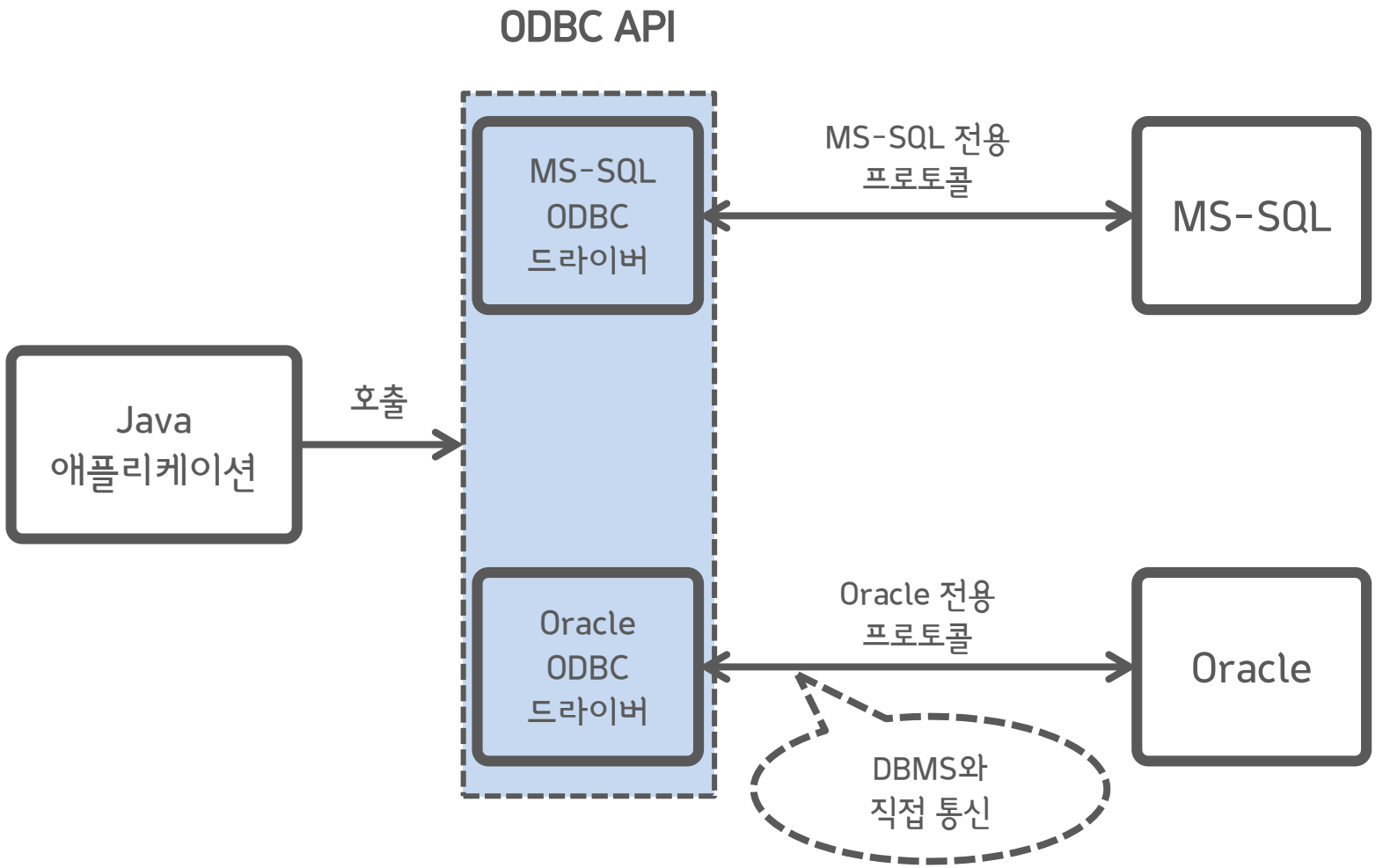
- 별도 다운로드 필요
- ODBC 사용 안 함
- Native API 호출



JDBC Driver Types 4

JDBC Type 4 특징

- 별도 다운로드 필요
- DBMS와 직접 통신
- Pure Java



JDBC Driver Install Guide

Download

- 벤더 별로 제공하는 JDBC Driver
  - Oracle
    - http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
    - Oracle 버전에 알맞은 파일을 선택하여 다운로드 후, PATH 설정
  - MariaDB
    - https://mariadb.com/products/connectors-plugins
    - MariaDB 버전과 자신의 환경에 알맞은 파일을 선택하여 다운로드 후, PATH 설정
  - PostgreSQL
    - https://jdbc.postgresql.org/download.html
    - http://pqxx.org/development/libpqxx/ (C++)

DBMS별 JDBC

드라이버 사용법(1/2)

- Oracle

Database	Oracle Thin Driver
Type	jdbc:oracle:thin:@hostname:port:SID
Sample	String URL = “jdbc:oracle:thin:@wisoft.io:1521:wisoftora”;
Comment	jdbc:oracle:thin - 사용할 드라이버 wisoft.io - Host Name 1521 - Port Number wisoftora - Oracle SID

- Microsoft SQL Server

Database	Microsoft SQL Server
Type	jdbc:jk://hostname:port/database=dbname
Sample	String URL = “jdbc:jk://wisoft.io:1433/database=wisoftdb”;
Comment	jdbc:jk - 사용할 드라이버 wisoft.io - Host Name 1433 - Port Number wisoftdb - Database Name

DBMS별 JDBC

드라이버 사용법(2/2)

- ## MySQL

Database	Oracle MySQL
Type	jdbc:mysql://hostname:port/dbname
Sample	String URL = “jdbc:mysql://wisoft.io:3306/wisoftdb”;
Comment	jdbc:mysql - 사용할 드라이버 wisoft.io - Host Name 3306 - Port Number wisoftdb - Database Name

- ## PostgreSQL

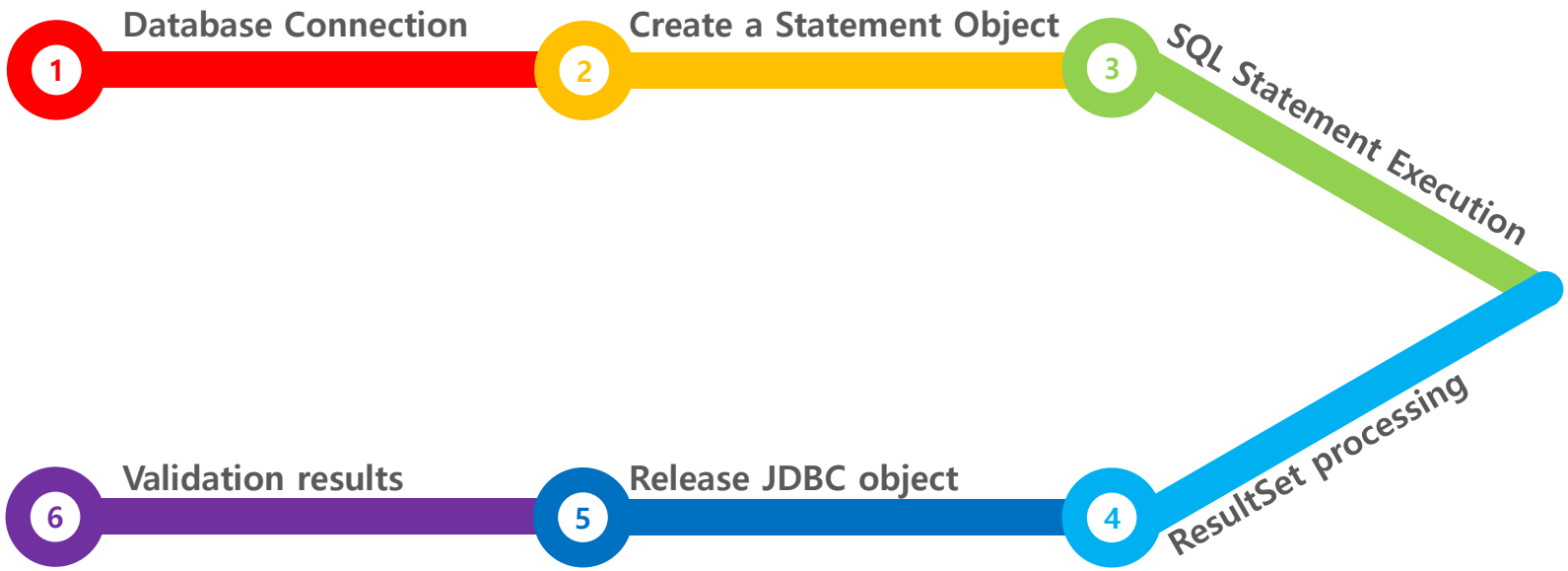
Database	PostgreSQL
Type	jdbc:postgresql://hostname:port/dbname
Sample	String URL = “jdbc:postgresql://wisoft.io:5432/wisoftdb”;
Comment	jdbc:postgresql - 사용할 드라이버 wisoft.io - Host Name 5432 - Port Number wisoftdb - Database Name

https://pbs.twimg.com/profile_background_images/492727362123886592/eLwJ4qKy.jpeg

JDBC Programming

Overview (1/3)

- JDBC Programming 절차



Overview (2/3)

- JDBC Programming 절차
  - [STEP 1] Database Connection

```
String URL = "jdbc:postgresql://localhost:5432/dbname";
Connection conn = DriverManager.getConnection(URL, "username",
"password");
```

  - [STEP 2] SQL 처리를 위한 Statement 객체 생성

```
Statement stmt = conn.createStatement();
```

  - [STEP 3] SQL 문장 실행

```
String SQL = "select * from student;";
ResultSet result = stmt.executeQuery(SQL);
```

Overview (3/3)

- JDBC Programming 절차 Cont'

  - [STEP 4] 질의 결과 ResultSet 처리

```
While (result.next()) {
    String col1 = result.getString(1);
    String col2 = result.getString(2);
    int col3 = result.getInt(3);
}
```

  - [STEP 5] JDBC 객체 해제

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

JDBC 프로그래밍에 사용되는 객체

JDBC 프로그래밍에 사용되는 객체 (1/4)

- DriverManager Class
  - 데이터 원본에 JDBC 드라이버를 통하여 커넥션을 만드는 역할
  - 일반적으로 드라이버 클래스들은 로드될 때 자신의 인스턴스를 생성하고, 자동적으로 DriverManager 클래스 메소드를 호출하여 그 인스턴스를 등록
  - 모든 메소드는 Static
    - 반드시 객체를 생성시킬 필요가 없음
  - Connection 인터페이스의 구현 객체를 생성
    - getConnection() 메소드 사용

JDBC 프로그래밍에 사용되는 객체 (2/4)

- Connection Interface
  - 특정 데이터 원본에 대한 커넥션은 Connection 인터페이스가 구현된 클래스의 객체로 표현
  - 어떤 SQL 문장을 실행시키기 전에 우선 Connection 객체가 있어야 함
    - Connection 객체는 특정 데이터 원본과 연결된 커넥션을 나타냄
    - 특정한 SQL 문장을 정의하고 실행시킬 수 있는 Statement 객체를 생성할 때도 사용
  - 메타데이터에 관한 정보를 데이터 원본에 질의하는데 사용
    - 이때에는 사용 가능한 테이블의 이름, 특정 테이블의 열에 정보 등이 포함

JDBC 프로그래밍에 사용되는 객체 (3/4)

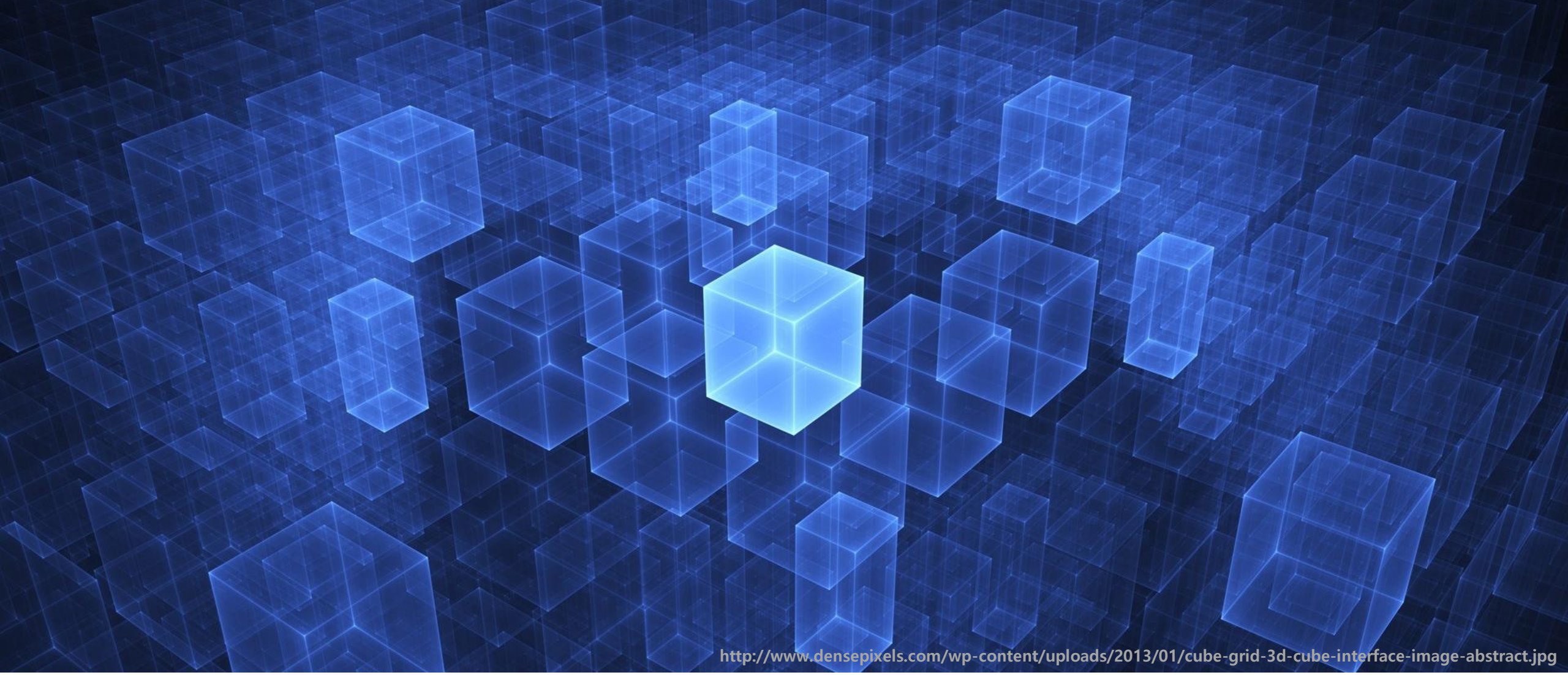
- 질의 문장인 SQL 문을 추상화한 인터페이스 객체를 생성
  - Statement
    - 일반적인 목적으로 Database와 연결할 때 사용
    - 정적(Static) 쿼리에 사용하며, 하나의 쿼리를 사용하면 더 이상 사용할 수 없음
  - PreparedStatement
    - 동적인 쿼리에 사용하며, 하나의 객체로 여러 번의 쿼리를 실행할 수 있음
    - SQL 문장에 정해지지 않은 값을 '?' 기호를 사용하여 바인드 변수를 이용
      - 바인드 변수에 값을 할당하기 위해 setXxx(Index, Value) 메소드 이용
    - SQL 쿼리를 실행하기 전에 미리 만들어야 함
  - CallableStatement

    데이터베이스 내의 프로시저 등을 호출할 때 사용

JDBC 프로그래밍에 사용되는 객체 (4/4)

- SQL Statement Execution
  - execute
    - 모든 쿼리 사용 가능 - DDL, DML, DCL
    - Return - Boolean Value
  - executeQuery
    - 주로 SELECT 문장에서 사용
    - Return - ResultSet
  - executeUpdate
    - 주로 DDL(CREATE, ALTER, DROP), DML(INSERT, UPDATE, DELETE)에 사용
    - Return - 적용된 행의 수

Break Time



Connection Test & Example

Simple Client Program with Java

Sample Code

Attention!!

이 코드는 기억에서 잊으세요.

인터넷에서 검색하면, 가장 많이 노출되는

코드 패턴이지만 문제(?)가 많은 코드입니다.

```
package io.wisoft.seminar;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class BadJdbcExample {
  public static void main(final String... args) {
    try {
      Connection conn = DriverManager.getConnection
        ("jdbc:postgresql://localhost:5432/exercise", "scott", "tiger");

      Statement stmt;
      stmt = conn.createStatement();

      ResultSet rs;
      rs = stmt.executeQuery("SELECT * FROM STUDENT");

      if (stmt.execute("SELECT * FROM STUDENT")) {
        rs = stmt.getResultSet();
      }

      while (rs.next()) {
        System.out.print("[학번] " + rs.getString(1) + " ¦¦ ");
        System.out.print("[이름] " + rs.getString(2) + " ¦¦ ");
        System.out.println("[생일] " + rs.getString(3));
      }

      rs.close();
      stmt.close();
      conn.close();
    } catch (SQLException sqex) {
      System.out.println("SQLException: " + sqex.getMessage());
      System.out.println("SQLState: " + sqex.getSQLState());
    }
  }
}
```

JDBC 프로그래밍… 이렇게 하세요~*

Class

PostgresqlAccess

```
package io.wisoft.seminar;
…

public class PostgresqlAccess {
  private static Connection conn = null;

  public void init() {
    try {
      Class.forName("org.postgresql.Driver");
    }
    catch (ClassNotFoundException e) {
      e.printStackTrace();
    }
  }

  public static Connection setConnection() {
    String url = "jdbc:postgresql://localhost:5432/exercise";
    String username = "scott";
    String password = "tiger";

    try {
      conn = DriverManager.getConnection(url, username, password);
    }
    catch (SQLException e) {
      e.printStackTrace();
    }

    return conn;
  }
}
```

Class

Student

```
package io.wisoft.seminar;

public class Student {
  private String studentNo;
  private String studentName;
  private String studentBirthday;

  public String getStudentNo() {
    return studentNo;
  }

  public void setStudentNo(String studentNo) {
    this.studentNo = studentNo;
  }

  public String getStudentName() {
    return studentName;
  }

  public void setStudentName(String studentName) {
    this.studentName = studentName;
  }

  public String getStudentBirthday() {
    return studentBirthday;
  }

  public void setStudentBirthday(String studentBirthday) {
    this.studentBirthday = studentBirthday;
  }
}
```

JDBC 프로그래밍으로 조회하기

Class

StudentSelectService

```
package io.wisoft.seminar;

…


public class StudentSelectService {


  public void getStudentAll() {
    …
  }

  public void getStudentNo(String studentNo) {
    …
  }

  public void getStudentName(String studentName) {
    …
  }

  public void getStudentBirthday(String studentBirthday) {
    …
  }

}
```

Class

Main

Q1. 전체 학생을 검색하시오.

```
package io.wisoft.seminar;

…

public class Main {
  public static void main(String[] args) {

    // SELECT AREA
    StudentSelectService selectStudent = new StudentSelectService();

    System.out.println("전체 학생을 검색합니다.");
    selectStudent.getStudentAll();
    System.out.println("");
}
```

Class

StudentSelectService

getStudentAll()

```
public void getStudentAll() {
  Connection conn = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "SELECT * FROM STUDENT";
    pstmt = conn.prepareStatement(query);
    rs = pstmt.executeQuery();
    while (rs.next()) {
      System.out.print("[학번] " + rs.getString(1) + " ¦¦ ");
      System.out.print("[이름] " + rs.getString(2) + " ¦¦ ");
      System.out.println("[생일] " + rs.getString(3));
    }
  } catch (SQLException sqex) {
    System.out.println("SQLException: " + sqex.getMessage());
    System.out.println("SQLState: " + sqex.getSQLState());
  } finally {
    if (rs != null) { try{ rs.close(); } catch(Exception e) { e.printStackTrace(); }}
    if (pstmt != null){try{pstmt.close();}catch(Exception e) { e.printStackTrace(); }}
    if (conn != null) { try{conn.close();}catch(Exception e) { e.printStackTrace(); }}
  }
}
```

Class

Main

Q2. 학번이 20110101 학생을 검색하시오.

```
package io.wisoft.seminar;

…

public class Main {
  public static void main(String[] args) {

    // SELECT AREA
    …
    System.out.println("학번 20110101 학생을 검색합니다.");
    String studentNo = "20110101";

    selectStudent.getStudentNo(studentNo);
    System.out.println("");
}
```

Class

StudentSelectService

getStudentNo()

```
public void getStudentNo(String studentNo) {
  Connection conn = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "SELECT * FROM STUDENT WHERE NO = ?";
    pstmt = conn.prepareStatement(query);
    pstmt.setString(1, studentNo);
    rs = pstmt.executeQuery();

    while (rs.next()) {
      System.out.print("[학번] " + rs.getString(1) + " ¦¦ ");
      System.out.print("[이름] " + rs.getString(2) + " ¦¦ ");
      System.out.println("[생일] " + rs.getString(3));
    }
  } // catch and finally
}
```

Class

Main

Q3. 학생 이름이 '일지매'인
학생을 검색하시오.

```
package io.wisoft.seminar;

…

public class Main {
  public static void main(String[] args) {

    // SELECT AREA
    …
    System.out.println("이름 일지매 학생을 검색합니다.");
    String studentName = "일지매";

    selectStudent.getStudentName(studentName);
    System.out.println("");
}
```

Class

StudentSelectService

getStudentName()

```
public void getStudentName(String studentName) {
  Connection conn = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "SELECT * FROM STUDENT WHERE NAME = ?";
    pstmt = conn.prepareStatement(query);
    pstmt.setString(1, studentName);
    rs = pstmt.executeQuery();

    while (rs.next()) {
      System.out.print("[학번] " + rs.getString(1) + " ¦¦ ");
      System.out.print("[이름] " + rs.getString(2) + " ¦¦ ");
      System.out.println("[생일] " + rs.getString(3));
    }
  } // catch and finally
}
```

Class

Main

04. 학생 생일이 1991-02-28인 학생을 검색하시오.

```
package io.wisoft.seminar;

…

public class Main {
  public static void main(String[] args) {

    // SELECT AREA
    …
    System.out.println("생일 1991-02-28 학생을 검색합니다.");
    String studentBirthday = "1991-02-28";

    selectStudent.getStudentBirthday(studentBirthday);
    System.out.println("");
}
```

Class

StudentSelectService

getStudentBirthday()

```
public void getStudentBirthday(String studentBirthday) {
  Connection conn = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "SELECT * FROM STUDENT WHERE BIRTHDAY = ?";
    pstmt = conn.prepareStatement(query);
    pstmt.setString(1, studentBirthday);
    rs = pstmt.executeQuery();

    while (rs.next()) {
      System.out.print("[학번] " + rs.getString(1) + " ¦¦ ");
      System.out.print("[이름] " + rs.getString(2) + " ¦¦ ");
      System.out.println("[생일] " + rs.getString(3));
    }
  } catch (SQLException sqex) {
    System.out.println("SQLException: " + sqex.getMessage());
    System.out.println("SQLState: " + sqex.getSQLState());
  } finally {
    if ( pstmt != null ) try{pstmt.close();}catch(Exception e){}
    if ( conn != null ) try{conn.close();}catch(Exception e){}
  }
}
```

JDBC 프로그래밍으로 등록하기

Class

StudentInsertService

```
package io.wisoft.seminar;
…

public class StudentInsertService {
  public void insertStudent(Student student) {
    …
  }

  public void insertStudentMulti(Student[] students) {
    …
  }

  public void insertStudentMultiBatch(Student[] students) {
    …
  }
}
```

Class

Main

Q5. 학번이 20110401이고,
이름이 이순신인 학생을 추가하시오.

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {
    Student student = new Student();
    Student[] students = new Student[8];

    for (int i=0; i<students.length; i++){
      students[i] = new Student();
    }
    // INSERT AREA
    StudentInsertService insertStudent = new StudentInsertService();
    System.out.println("학번이 20110401이고, 이름이 이순신인 학생을 추가합니다.");
    student.setStudentNo("20110401");
    student.setStudentName("이순신");
    insertStudent.insertStudent(student);
    System.out.println("");
}
```

Class

StudentInsertService

insertStudent ()

```
public void insertStudent(Student student) {
  Connection conn = null;
  PreparedStatement pstmt = null;
  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "INSERT INTO STUDENT(NO, NAME, BIRTHDAY) VALUES (?, ?, ?)";
    pstmt = conn.prepareStatement(query);
    pstmt.setString(1, student.getStudentNo());
    pstmt.setString(2, student.getStudentName());

    if(student.getStudentBirthday() == null) {
      pstmt.setNull(3, Types.VARCHAR);
    } else {
      pstmt.setString(3, student.getStudentBirthday());
    }

    int retValue = pstmt.executeUpdate();
    conn.commit();
    System.out.println(retValue + "건의 사항이 처리되었습니다.");
  } catch (SQLException sqex) {
    try { if (conn != null) { conn.rollback(); }}
    catch (SQLException e) { e.printStackTrace(); }
    System.out.println("SQLException: " + sqex.getMessage());
    System.out.println("SQLState: " + sqex.getSQLState());
  } // finally
}
```

Class

Main

Q6. 다음 정보를 확인하여
학생 목록을 추가하시오. (Multi Insert)

- (20110501, 이율곡)
- (20110601, 이수일)
- (20110701, 심순애)
- (20110801, 임꺽정)

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {
    // INSERT AREA
    …
    System.out.println("학번이 20110501이고, 이름이 이율곡인 학생을 추가합니다.");
    …
    System.out.println("학번이 20110801이고, 이름이 임꺽정인 학생을 추가합니다.");

    students[0].setStudentNo("20110501");
    students[0].setStudentName("이율곡");
    …
    students[3].setStudentNo("20110801");
    students[3].setStudentName("임꺽정");

    insertStudent.insertStudentMulti(students);
    System.out.println("");
}
```

Class

StudentInsertService

insertStudentMulti()

```
public void insertStudentMulti(Student[] students) {
  Connection conn = null;
  PreparedStatement pstmt = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    int retValue = 0;
    String query = "INSERT INTO STUDENT(NO, NAME, BIRTHDAY) VALUES (?, ?, ?)";
    pstmt = conn.prepareStatement(query);

    for(int i=0; i<students.length; i++) {
      if (students[i].getStudentNo() == null && students[i].getStudentName()==null)
        break;

      pstmt.setString(1, students[i].getStudentNo());
      pstmt.setString(2, students[i].getStudentName());

      if(students[i].getStudentBirthday() == null) {
        pstmt.setNull(3, Types.VARCHAR);
      } else {
        pstmt.setString(3, students[i].getStudentBirthday());
      }
      retValue += pstmt.executeUpdate();
      pstmt.clearParameters();
    }
    conn.commit();
    System.out.println(retValue + "건의 사항이 처리되었습니다.");
  } // catch and finally
}
```

Class

Main

Q7. 다음 정보를 확인하여
학생 목록을 추가하시오.
(Using Multi Insert Batch)

(20110901, 이상훈)
(20111001, 강동희)
(2011101, 김호성)
(20111201, 김정준)

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {
    // INSERT AREA
    …
    System.out.println("학번이 20110901이고, 이름이 이상훈인 학생을 추가합니다.");
    …
    System.out.println("학번이 20111201이고, 이름이 김정준인 학생을 추가합니다.");

    students[0].setStudentNo("20110901");
    students[0].setStudentName("이상훈");
    …
    students[3].setStudentNo("20111201");
    students[3].setStudentName("김정준");

    insertStudent.insertStudentMultiBatch(students);
    System.out.println("");
}
```

Class

StudentInsertService

insertStudentMultiBatch()

```
public void insertStudentMultiBatch(Student[] students) {
  Connection conn = null;
  PreparedStatement pstmt = null;
  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "INSERT INTO STUDENT(NO, NAME, BIRTHDAY) VALUES (?, ?, ?)";
    pstmt = conn.prepareStatement(query);

    for(int i=0; i<students.length; i++) {
      if (students[i].getStudentNo() == null && students[i].getStudentName()==null)
        break;

      pstmt.setString(1, students[i].getStudentNo());
      pstmt.setString(2, students[i].getStudentName());

      if(students[i].getStudentBirthday() == null) {
        pstmt.setNull(3, Types.VARCHAR);
      } else {
        pstmt.setString(3, students[i].getStudentBirthday());
      }
      pstmt.addBatch();
      pstmt.clearParameters();
    }

    int[] retValue = pstmt.executeBatch();
    conn.commit();
    System.out.println(retValue.length + "건의 사항이 처리되었습니다.");
  } // catch and finally
}
```

JDBC 프로그래밍으로 수정하기

Class

StudentUpdateService

```
package io.wisoft.seminar;
…


public class StudentUpdateService {
  public void updateStudentBirthday(String no, String birthday) {
    …
  }

  //overloading
  public void updateStudentBirthday(Student student) {
    …
  }

  public void updateStudentBirthdayMultiBatch(Student[] students) {
    …
  }
}
```

Class

Main

Q8. 학번이 20110401인 학생 생일을 1990-03-21으로 변경하시오.

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {

    // UPDATE AREA
    StudentUpdateService updateStudent = new StudentUpdateService();

    System.out.println("학번이 20110401인 학생의 생일을 1990-03-21으로 변경합니다.");
    updateStudent.updateStudentBirthday("20110401", "1990-03-21");
    System.out.println("");
  }
}
```

Class

StudentUpdateService

updateStudentBirthday()

```
public void updateStudentBirthday(String no, String birthday) {
  Connection conn = null;
  PreparedStatement pstmt = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "UPDATE STUDENT SET BIRTHDAY = ? WHERE NO = ?";
    pstmt = conn.prepareStatement(query);

    pstmt.setString(1, birthday);
    pstmt.setString(2, no);

    int retValue = pstmt.executeUpdate();
    conn.commit();

    System.out.println(retValue + "건의 사항이 처리되었습니다.");
  } // catch and finally
}
```

Class

Main

Q9. 학번이 20110401인 학생 생일을 1990-03-25으로 변경하시오.

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {

    // UPDATE AREA
    …
    System.out.println("학번이 20110401인 학생의 생일을 1990-03-25으로 변경합니다.");
    student.setStudentNo("20110401");
    student.setStudentBirthday("1990-03-25");
    updateStudent.updateStudentBirthday(student);
    System.out.println("");
  }
}
```

Class

StudentUpdateService

updateStudentBirthday()

```
public void updateStudentBirthday(Student student) {
  Connection conn = null;
  PreparedStatement pstmt = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "UPDATE STUDENT SET BIRTHDAY = ? WHERE NO = ?";
    pstmt = conn.prepareStatement(query);

    pstmt.setString(1, student.getStudentBirthday());
    pstmt.setString(2, student.getStudentNo());

    int retValue = pstmt.executeUpdate();
    conn.commit();

    System.out.println(retValue + "건의 사항이 처리되었습니다.");
  } // catch and finally
}
```

Class

Main

Q10. 다음 정보를 확인하여
학생 목록을 갱신하시오.

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {
    // UPDATE AREA
    …
    System.out.println("학번이 20110501인 학생의 생일을 1990-03-01으로 변경합니다.");
    …
    System.out.println("학번이 20111201인 학생의 생일을 1990-10-01으로 변경합니다.");

    students[0].setStudentNo("20110501");
    students[0].setStudentBirthday("1990-03-01");
    …
    students[7].setStudentNo("20111201");
    students[7].setStudentBirthday("1990-10-01");

    updateStudent.updateStudentBirthdayMultiBatch(students);
    System.out.println("");
}
```

Class

StudentUpdateService

updateStudentBirthday()

```
public void updateStudentBirthdayMultiBatch(Student[] students) {
  Connection conn = null;
  PreparedStatement pstmt = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "UPDATE STUDENT SET BIRTHDAY = ? WHERE NO = ?";
    pstmt = conn.prepareStatement(query);

    for(int i=0; i<students.length; i++) {
      if (students[i].getStudentNo()==null && students[i].getStudentBirthday()==null)
        break;

      pstmt.setString(1, students[i].getStudentBirthday());
      pstmt.setString(2, students[i].getStudentNo());

      pstmt.addBatch();
      pstmt.clearParameters();
    }

    int[] retValue = pstmt.executeBatch();
    conn.commit();
    System.out.println(retValue.length + "건의 사항이 처리되었습니다.");
  } // catch and finally
}
```

JDBC 프로그래밍으로 삭제하기

Class

StudentDeleteService

```
package io.wisoft.seminar;
…


public class StudentDeleteService {
  public void deleteStudentNo(String studentNo) {
    …
  }

  public void deleteStudentNoMultiBatch(Student[] students) {
    …
  }
}
```

Class

Main

Q11. 학번이 20110401인 학생을 목록에서 제거하시오.

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {

    // DELETE AREA
    StudentDeleteService deleteStudent = new StudentDeleteService ();

    System.out.println("학번이 20110401인 학생의 생일을 1990-03-21으로 변경합니다.");
    deleteStudent.deleteStudentNo("20110401");
    System.out.println("");
}
```

Class

StudentDeleteService

deleteStudentNo()

```
public void deleteStudentNo(String studentNo) {
  Connection conn = null;
  PreparedStatement pstmt = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "DELETE FROM STUDENT WHERE NO = ?";
    pstmt = conn.prepareStatement(query);

    pstmt.setString(1, studentNo);

    int retValue = pstmt.executeUpdate();
    conn.commit();

    System.out.println(retValue + "건의 사항이 처리되었습니다.");
  } // catch and finally
}
```

Class

Main

Q12. 다음 정보를 확인하여
학생 목록을 제거하시오.

```
package io.wisoft.seminar;
…
public class Main {
  public static void main(String[] args) {
    // DELETE AREA
    …
    System.out.println("학번이 20110501인 학생을 목록에서 제거합니다.");
    …
    System.out.println("학번이 20111201인 학생을 목록에서 제거합니다.");

    students[0].setStudentNo("20110501");
    …
    students[7].setStudentNo("20111201");

    deleteStudent.deleteStudentNoMultiBatch(students);
    System.out.println("");
}
```

Class

StudentDeleteService

deleteStudentNoMultiBatch()

```
public void deleteStudentNoMultiBatch(Student[] students) {
  Connection conn = null;
  PreparedStatement pstmt = null;

  try {
    conn = PostgresqlAccess.setConnection();
    conn.setAutoCommit(false);

    String query = "DELETE FROM STUDENT WHERE NO = ?";
    pstmt = conn.prepareStatement(query);

    for(int i=0; i<students.length; i++) {
      if (students[i].getStudentNo() == null)
        break;

      pstmt.setString(1, students[i].getStudentNo());
      pstmt.addBatch();
      pstmt.clearParameters();
    }

    int[] retValue = pstmt.executeBatch();
    conn.commit();

    System.out.println(retValue.length + "건의 사항이 처리되었습니다.");
  }  // catch and finally
}
```

Break Time

Quiz

Question 1

HNU Entertainment의

부서 코드, 이름, 위치를 검색하시오.

Question 2

HNU Entertainment의 연예관계자 코드, 이름, 관리자, 급여를 검색하시오.

Question 3

HNU Entertainment(HNU-E)에서
제작한 드라마의 코드와 이름을 검색하시오.

Question 4

드라마 방영사가

KBC이거나 SBC인 드라마를 검색하시오.

Question 5

드라마 제작사를 검색하시오.

단, 중복된 값이 있으면 제거하시오.

Question 6

연예관계자들의 급여의

총합과 평균 급여액을 계산하시오.

Question 7

방영일자가 아직 확정되지 않은
드라마의 이름을 검색하시오.

Question 8

연예관계자에 대해 연예관계자의 이름과 직속 상사의 이름을 검색하시오.

Question 9

연예관계자에 대해 이름과 급여를 출력하고,
급여의 내림차순으로 정렬하시오.
단, 동일 급여일 때는 이름의 오름차순으로
정렬하시오.

Question 10

모든 연예관계자를 직급별로 그룹화하고,
평균 급여액이 5000 이상인 직급에 대해
연예 관계자의 직급, 평균 급여액,
최소 급여액, 최대 급여액을 검색하시오.

Question 11

모든 연예관계자의 평균 급여액보다
많은 급여를 받는 연예관계자의
이름과 급여를 검색하라.

Question 12

방영일자가 확정되지 않은
드라마의 방영일자가 2013-05-01로
편성되었습니다. 알맞게 변경하시오.

Question 13

연예관계자 김수현 씨가 대리에서 실장으로 승진하고 급여가 20% 증가 되었습니다. 알맞게 변경하시오.

Question 14

우리 회사에 한 명의 임원이 등록되었습니다.
코드는 E903, 이름은 손진현, 관리자는 E901,
급여는 4000입니다. 알맞게 등록하시오.

Question 15

연예관계자인 손진현님이 퇴직했습니다.

연예관계자 목록에서 제거하시오.

hsjeon@hanbat.ac.kr

http://blog.ngelmaum.org

Thank You