# Assignment #1

## Overview Documentation

**2020095178 최윤선**

### ✓ Metadata

PostgreSQL 설치 및 JDBC 연결

### ✓ Summary


- 이번 과제는 PostgreSQL 을 우분투에 설치하고 JDBC 와 연결하는 간단한 과제로, 강의자료를 따라가며 수행하였다.
- 프로그램을 설계, 구현 또는 테스트하는 동안 발생한 주요 문제

  ➔ 코딩을 하는 과정에서 사소한 오타가 발생하여 테스트할 때 에러가 발생하였다. 단순한 문제였기에 오타만 고쳐 수월하게 해결할 수 있었다.


### ✓ Specification

- Version

  Ubuntu == 22.04.2

  PostgreSQL == 14

### ✓ Design and Implementation

PostgreSQL 과 JDBC 를 연결시키기 위해, 우분투에 PostgreSQL 을 설치하고 java 파일들을 구현하였다.

- Test schema 구현

```
yschoi=# create table student
yschoi-# (
yschoi(# roll integer not null primary key,
yschoi(# name varchar(100) not null,
yschoi(# section varchar(2) not null,
yschoi(# created_date timestamp not null
yschoi(# );
```

- PostgresWithJDBCConnection.java 파일을 통한 JDBC 연결

```
public class PostgresWithJDBCConnection {
        public static void main(String[] args) {
                // establishes database connection
                try (Connection connection = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/yschoi", "ysc
hoi", "1008")) {
                        System.out.println("Connection established successfully");
                } catch (SQLException e) {
                        System.out.print(e.getMessage());
                }
        }
}
```

연결에 성공하면 'Connection established successfully'라는 문구가 나오도록 구현하였다.

```
(base) yschoi@DESKTOP-5K2PKFE:~$ java -cp ':/usr/lib/jvm/java-1.8.0-openjdk-amd64/lib/postgresql-42.6.0.jar' PostgresWit
hJDBCConnection
Connection established successfully
```

- 만들어 놓은 test schema 에 데이터를 직접 insert, select, update, delete 하기 위한 java 코드 구현
  - PostgresWithJDBC**Insert**.java

```java
public class PostgresWithJDBCInsert {
	public static void main(String[] args) {
		String SQL_INSERT = "INSERT INTO STUDENT (ROLL, NAME, SECTION, CREATED_DATE) VALUES (?, ?, ?, ?)";
		// establishes database connection
		// auto closes connection and preparedStatement
		try (Connection conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/yschoi", "yschoi", "1008");
			PreparedStatement preparedStatement = conn.prepareStatement(SQL_INSERT)) {
			// insert student record
			preparedStatement.setInt(1, 06); //1 specifies the first parameter in the query
			preparedStatement.setString(2, "Arnold");
			preparedStatement.setString(3, "A");
			preparedStatement.setTimestamp(4, new Timestamp(new Date().getTime()));
			preparedStatement.executeUpdate();
			System.out.print("record inserted successfully");
		} catch (SQLException e) {
			System.out.print(e.getMessage());
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
```

  - PostgresWithJDBC**Select**.java

```java
public class PostgresWithJDBCSelect {
	public static void main(String[] args) {
		List < Student > studentList = new ArrayList < > ();
		String SQL_SELECT = "Select * from STUDENT";
		// establishes database connection
		// auto closes connection and preparedStatement
		try (Connection conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/yschoi", "yschoi", "1008");
			PreparedStatement preparedStatement = conn.prepareStatement(SQL_SELECT)) {
			ResultSet resultSet = preparedStatement.executeQuery();
			while (resultSet.next()) {
				int rollId = resultSet.getInt("ROLL");
				String name = resultSet.getString("NAME");
				String section = resultSet.getString("SECTION");
				Timestamp createdDate = resultSet.getTimestamp("CREATED_DATE");
				Student student = new Student();
				student.setRoll(rollId);
				student.setName(name);
				student.setSection(section);
				student.setCreatedDate(createdDate.toLocalDateTime());
				studentList.add(student);
			}
			for (Student student: studentList) {
				System.out.println("Roll No: " + student.getRoll());
				System.out.println("Name: " + student.getName());
				System.out.println("Section: " + student.getSection());
			}
```

  - PostgresWithJDBC**Update**.java

```java
public class PostgresWithJDBCUpdate {
	public static void main(String[] args) {
		String SQL_UPDATE = "UPDATE STUDENT set SECTION = 'D' where ROLL=6;";
		List < Student > studentList = new ArrayList < > ();
		String SQL_SELECT = "Select * from STUDENT where ROLL=1";

		// establishes database connection
		// auto closes connection and preparedStatement
		try (Connection conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/yschoi", "yschoi", "1008");
			PreparedStatement preparedStatement = conn.prepareStatement(SQL_UPDATE)) {
			// update student record
			preparedStatement.executeUpdate();
			System.out.println("record updated successully");
			// fetch updated record
			PreparedStatement preparedStatement1 = conn.prepareStatement(SQL_SELECT);
			ResultSet resultSet = preparedStatement1.executeQuery();
			while (resultSet.next()) {
				int rollId = resultSet.getInt("ROLL");
				String name = resultSet.getString("NAME");
				String section = resultSet.getString("SECTION");
				Timestamp createdDate = resultSet.getTimestamp("CREATED_DATE");
				Student student = new Student();
				student.setRoll(rollId);
				student.setName(name);
				student.setSection(section);
				student.setCreatedDate(createdDate.toLocalDateTime());
				studentList.add(student);
			}
			for (Student student: studentList) {
				System.out.println("Roll No:: " + student.getRoll());
				System.out.println("Name:: " + student.getName());
				System.out.println("Section:: " + student.getSection());
			}
			preparedStatement1.close();
```

- PostgresWithJDBC**Delete**.java

```java
public class PostgresWithJDBCDelete {
	public static void main(String[] args) {
		String SQL_DELETE = "Delete from STUDENT where ROLL=6;";
		List < Student > studentList = new ArrayList < > ();
		String SQL_SELECT = "Select * from STUDENT";
		// establishes database connection
		// auto closes connection and preparedStatement
		try (Connection conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/yschoi", "yschoi", "1008");
			PreparedStatement preparedStatement = conn.prepareStatement(SQL_DELETE)) {
			// delete student record
			preparedStatement.executeUpdate();
			System.out.println("record deleted successfully");
			// fetch record
			PreparedStatement preparedStatement1 = conn.prepareStatement(SQL_SELECT);
			ResultSet resultSet = preparedStatement1.executeQuery();
			while (resultSet.next()) {
				int rollId = resultSet.getInt("ROLL");
				String name = resultSet.getString("NAME");
				String section = resultSet.getString("SECTION");
				Timestamp createdDate = resultSet.getTimestamp("CREATED_DATE");
				Student student = new Student();
				student.setRoll(rollId);
				student.setName(name);
				student.setSection(section);
				student.setCreatedDate(createdDate.toLocalDateTime());
				studentList.add(student);
			}
			for (Student student: studentList) {
				System.out.println("**********************");
				System.out.println("Roll No:: " + student.getRoll());
				System.out.println("Name:: " + student.getName());
				System.out.println("Section:: " + student.getSection());
			}
			preparedStatement1.close();
```

## ✓ Testing

구현한 java 코드를 통해 PostgreSQL 과 연결하고 insert, select, update, delete 를 테스트한 결과이다.

- Insert

```
(base) yschoi@DESKTOP-5K2PKFE:~$ java -cp ':/usr/lib/jvm/java-1.8.0-openjdk-amd64/lib/postgresql-42.6.0.jar' PostgresWithJDBCInsert
record inserted successfully(base) yschoi@DESKTOP-5K2PKFE:~$ psql
```

```
yschoi=# select * from student;
 roll |  name  | section |      created_date
------+--------+---------+-------------------------
    6 | Arnold | A       | 2023-09-28 11:36:57.749
(1 row)
```

- Select

```
(base) yschoi@DESKTOP-5K2PKFE:~$ java -cp ':/usr/lib/jvm/java-1.8.0-openjdk-amd64/lib/postgresql-42.6.0.jar' PostgresWithJDBCSelect
Roll No: 6
Name: Arnold
Section: A
```

- Update

```
(base) yschoi@DESKTOP-5K2PKFE:~$ java -cp ':/usr/lib/jvm/java-1.8.0-openjdk-amd64/lib/postgresql-42.6.0.jar' PostgresWithJDBCUpdate
record updated successully
(base) yschoi@DESKTOP-5K2PKFE:~$ java -cp ':/usr/lib/jvm/java-1.8.0-openjdk-amd64/lib/postgresql-42.6.0.jar' PostgresWithJDBCSelect
Roll No: 6
Name: Arnold
Section: D
```

- Delete

```
(base) yschoi@DESKTOP-5K2PKFE:~$ java -cp ':/usr/lib/jvm/java-1.8.0-openjdk-amd64/lib/postgresql-42.6.0.jar' PostgresWithJDBCDelete
record deleted successfully
(base) yschoi@DESKTOP-5K2PKFE:~$ java -cp ':/usr/lib/jvm/java-1.8.0-openjdk-amd64/lib/postgresql-42.6.0.jar' PostgresWithJDBCSelect
```

```
(base) yschoi@DESKTOP-5K2PKFE:~$ psql
psql (14.9 (Ubuntu 14.9-1.pgdg22.04+1))
Type "help" for help.

yschoi=# select * from student;
 roll | name | section | created_date
------+------+---------+--------------
(0 rows)
```

✓ **Known problems**

Nothing.

✓ **Comments**

강의자료를 따라가며 PostgreSQL 설치와 코드 구현을 진행하니 큰 어려움 없이 이번 과제를 수행할 수 있었다.