# 2015 fall 4190.310-001 Database
# Project 2: Implementing a Simple Database Application

2013-11395 김희훈

## 1. Introduction

이번 프로젝트에서는 주어진 MariaDB 서버에 JDBC로 접속하여 대학교 원서 접수를 시뮬레이션하는 간단한 어플리케이션을 만들었다.

## 2. Schema

간단한 어플리케이션이므로 스키마에서의 제약사항들은 최소화하고 클라이언트단에서의 검증을 지향하였다.

***university***
*(*
*uid int primary key,*
*name varchar(128),*
*capacity int,*
*ugroup enum('A', 'B', 'C'),*
*weight float,*
*applied int*
*)*

열은 각각 학교 ID, 이름, 정원, 군, 내신 성적 반영 비율, 지원자 수를 의미한다. 지원자 수는 그때마다 구할 수 있지만, 지원자 수가 필요한 경우가 잦으므로 중복을 감수하고 별도로 저장하였다.

***student***
*(*
*sid int primary key,*
*name varchar(20),*
*csat_score int,*
*school_score int*
*)*

각각 학생 ID, 이름, 수능 성적, 내신 성적을 의미한다.

***apply***
*(*
*uid int,*
*sid int,*
*index (uid),*
*index (sid)*
*)*

지원한 학교 정보를 나타내는 테이블이다. foreign key는 걸지 않았으며 id로 검색하는 경우가 잦으므로 index만 걸어주었다.

## 3. Classes

**University**

university 테이블의 한 행을 나타내는 클래스이다. 생성자와 getter가 있다.

**Student**

student 테이블의 한 행을 나타내는 클래스이다. 생성자와 getter가 있으며, 두 학생의 점수가 완전히 일치하는지 확인하는 hasSameScore 메소드가 있다.

**SimpleDBApp**

메인 루프와 모든 로직을 처리하는 클래스이다.

# 4. Helper Functions

초기화나 중복되는 작업들을 수행하는 메소드들이다.

**init**

mariaDB로 접속하고, 사용자 입력을 받기위해 stdin의 스캐너를 만들어 둔다. 그리고 사용할 쿼리들을 precompiled statement로 미리 만들어 둔다.

**getAction**

메뉴를 출력하고, 사용자에게 수행할 메뉴 번호를 입력 받는다.

**univExists & studExists**

사용자에게서 uid/sid를 입력받을때마다 실제로 존재하는 대학/학생인지 확인하는 함수들이다.

**studRs2List & univRs2List**

쿼리를 날리면 응답이 ResultSet 형태로 돌아오는데 이를 List로 바꿔주는 함수이다.

**getAcceptedStud**

uid를 받아 그 대학의 예상 합격자 list를 리턴하는 함수이다. 동점자 처리 로직이 여기서 수행된다.

**printUniv & printStud**

대학/학생 리스트를 받아 표 형태로 출력해주는 함수이다.

# 5. Menu functions

1번부터 12번까지의 메뉴 동작을 하는 함수들이다.

**printAllUniv & printAllStud**

printUniv/printStud 함수를 이용하여 단순히 모든 대학/학생을 출력한다.

**insertUniv & insertStud**

대학/학생을 새로 추가한다. 사용자 입력을 받으면서 수들이 정해진 범위를 벗어나지 않는지 체크한다.

**removeUniv & removeStud**

대학/학생을 제거한다. apply 테이블에서 원서 접수 내역도 같이 제거하며, 특히 학생을 제거할때는 학생이 접수한 대학들의 applied 값을 1씩 감소시켜준다.

**apply**

원서를 접수하는 함수이다. apply에 행을 추가하고 접수된 대학의 applied 값을 1 증가시켜준다. 학군이 겹치지 않는지도 체크한다.

**printStudByUniv & printUnivByStud**

학생이 접수한 대학 / 대학에 접수한 학생을 printUniv/printStud 함수를 통해 단순 출력한다.

**printAcceptedStudByUniv / printAcceptedUnivByStud**

대학별 예상 합격 학생 / 학생별 예상 합격 대학을 출력한다. 전자의 경우 getAcceptedStud 함수의 결과를 그대로 출력하면 된다. 후자의 경우 학생이 지원한 모든 대학에 대해 예상 합격자를 얻은 후, 그 학생이 예상 합격자에 포함되어 있는 대학만 출력한다.

# 6. 구현시 특별하게 고려한 점 / 가정

- 출력 함수에서 결과가 0개일시, No result 같은 메시지 대신 빈 테이블을 출력하도록 하였다.
- 모든 동작이 읽기 - 쓰기 순으로 이루어진다. 그러므로 읽기 단계에서는 특별히 고려할 사항이 없다. 쓰기에서는 대학/학생 삭제와 원서접수에서 apply 테이블을 갱신해 주기 때문에 두번 이상의 쓰기가 연속으로 이루어진다. 그래서 atomicity를 유지하기 위해 autocommit을 off하고 트랜잭션을 걸어주고, 오류가 발생할 시 롤백을 수행한다.

# 7. 컴파일과 실행 방법

소스는 https://github.com/csehydrogen/SimpleDBApp에 있으며, IDE나 콘솔 환경에서 mariaDB connector/J와 함께 컴파일 후 실행하면 된다.

# 8. 느낀 점

실제로 DB를 이용한 어플리케이션을 작성해 보니 "현업에서는 성능을 위해 redundancy를 허용하는 경우가 종종 있다." 같은 말이나, materialization 등의 개념을 이해할 수 있었다. Constraint를 서버단에서 체크하는 부담이 DBMS 덕분에 상당히 줄어든다는 것도 알게 되었다. 코드에서 아쉬웠던 부분은 statement들의 관리였는데 일반적인 convention이 어떻게 되는지 궁금하다.