그림, 예술, 다채로움, 디자인이(가) 표시된 사진

자동 생성된 설명

**Database weak 13-14 assignment**[[1]](#footnote-1)

20203361 장민석

**1. 12주차 과제의 최종 결과로 제출한 RDB 스키마 다이어그램**

스케치, 그림, 텍스트, 평면도이(가) 표시된 사진

자동 생성된 설명

**2. create table문**

다음과 같이 총 10개의 사용할 테이블을 생성하였다.

DROP TABLE IF EXISTS project, takes, student, instructor, course\_dept, section, rooms, course, time\_slot;

-- Course table

CREATE TABLE course (

course\_id VARCHAR(8),

title VARCHAR(50),

credits NUMERIC(2,0),

PRIMARY KEY (course\_id)

);

-- Section table

CREATE TABLE section (

course\_id VARCHAR(8),

sec\_id VARCHAR(8),

semester VARCHAR(6) CHECK (semester IN ('Fall', 'Winter', 'Spring', 'Summer')),

year NUMERIC(4,0),

PRIMARY KEY (course\_id, sec\_id, semester, year),

FOREIGN KEY (course\_id) REFERENCES course(course\_id)

ON DELETE CASCADE

);

-- Course\_dept table

CREATE TABLE course\_dept (

course\_id VARCHAR(8),

dept\_name VARCHAR(50),

PRIMARY KEY (course\_id, dept\_name),

FOREIGN KEY (course\_id) REFERENCES course(course\_id)

);

-- Time\_slot table

CREATE TABLE time\_slot (

time\_slot\_id VARCHAR(4),

day VARCHAR(3),

start\_time NUMERIC(2) CHECK (start\_time >= 0 AND start\_time < 24),

end\_time NUMERIC(2) CHECK (end\_time >= 0 AND end\_time < 24),

PRIMARY KEY (time\_slot\_id, day, start\_time)

);

-- Rooms table

CREATE TABLE rooms (

course\_id VARCHAR(8),

sec\_id VARCHAR(8),

year NUMERIC(4,0),

semester VARCHAR(6),

time\_slot\_id VARCHAR(4),

day VARCHAR(3),

start\_time NUMERIC(2),

building VARCHAR(15),

room\_number VARCHAR(7),

FOREIGN KEY (course\_id, sec\_id, semester, year) REFERENCES section(course\_id, sec\_id, semester, year),

FOREIGN KEY (time\_slot\_id, day, start\_time) REFERENCES time\_slot(time\_slot\_id, day, start\_time)

);

-- Student table

CREATE TABLE student (

ID VARCHAR(5),

name VARCHAR(20) NOT NULL,

dept\_name VARCHAR(50),

tot\_cred NUMERIC(3,0),

PRIMARY KEY (ID)

);

-- Instructor table

CREATE TABLE instructor (

ID VARCHAR(5),

name VARCHAR(20),

dept\_name VARCHAR(50),

salary NUMERIC(8,2),

PRIMARY KEY (ID)

);

-- Takes table

CREATE TABLE takes (

ID VARCHAR(5),

course\_id VARCHAR(8),

sec\_id VARCHAR(8),

semester VARCHAR(6),

year NUMERIC(4,0),

grade VARCHAR(2),

PRIMARY KEY (ID, course\_id, sec\_id, semester, year),

FOREIGN KEY (course\_id, sec\_id, semester, year) REFERENCES section(course\_id, sec\_id, semester, year),

FOREIGN KEY (ID) REFERENCES student(ID)

);

-- Project table

CREATE TABLE project (

s\_id VARCHAR(5),

course\_id VARCHAR(8),

sec\_id VARCHAR(8),

semester VARCHAR(6),

year NUMERIC(4,0),

num NUMERIC(2,0),

name VARCHAR(50),

i\_id VARCHAR(5),

max\_score NUMERIC(3,0),

score NUMERIC(3,0),

PRIMARY KEY (s\_id, course\_id, sec\_id, semester, year, num),

FOREIGN KEY (s\_id, course\_id, sec\_id, semester, year) REFERENCES takes(ID, course\_id, sec\_id, semester, year)

);

**3. SQL문**

첫 번째 기능을 구현하기 위해 사용되는 SQL 구문은 다음과 같이 2개이다. 먼저 query1은 'course' 테이블에서 course\_id, title, credits을 조회한다. 하지만 모든 강의를 조회하는 것이 아닌, 'section' 테이블과 'course\_dept' 테이블에 있는 강의 중에서 주어진 year와 semester에 해당하는 강의들 중, 두 개 이상의 학과에서 개설한 과목, 즉 융합과목 만을 조회한다. 이를 위해 group by 구문과 having 구문을 사용했다. query2는 'course\_dept' 테이블에서 query1에서 결과로 조회된 course\_id에 해당하는 투플을 검색한다. 이를 통해 융합과목의 dept\_name 리스트를 모두 찾을 수 있다.

String query1 = """

SELECT course\_id, title, credits

FROM course

WHERE course\_id IN (

SELECT S.course\_id

FROM section as S, course\_dept as CD

WHERE (S.year, S.semester) = (?, ?)

ANd S.course\_id = CD.course\_id

GROUP BY S.course\_id

HAVING COUNT(\*) >= 2

)""";

String query2 = """

SELECT dept\_name

FROM course\_dept

WHERE course\_id = ?

""";

두 번째 기능을 구현하기 위해 사용되는 SQL 구문은 다음과 같이 2개이다. query1은 'section' 테이블과 'course' 테이블을 조인하여, 특정 연도와 학기에 개설되는 모든 section을 찾고 이의 course\_id, title, sec\_id를 조회한다. query2는 위에서 찾은 section의 정보를 바탕으로 'rooms' 테이블과 'time\_slot' 테이블을 조인하여, 해당 section의 요일 별 강의실의 위치 정보인 building, room\_number와 수업 시간 정보인 day, start\_time, end\_time 정보를 찾는다.

String query1 = """

SELECT S.course\_id, C.title, S.sec\_id

FROM section as S, course as C

WHERE (S.year, S.semester) = (?, ?)

AND S.course\_id = C.course\_id

""";

String query2 = """

SELECT T.day, T.start\_time, T.end\_time, R.building, R.room\_number

FROM rooms as R, time\_slot as T

WHERE (R.time\_slot\_id, R.day, R.start\_time) = (T.time\_slot\_id, T.day, T.start\_time)

AND (R.course\_id, R.sec\_id, R.year, R.semester) = (?, ?, ?, ?)

""";

세 번째 기능을 구현하기 위해서 총 4개의 쿼리를 실행한다. query1은 'student' 테이블에서 특정 학생 ID에 해당하는 학생의 ID, name, dept\_name을 조회한다. query2는 'takes' 테이블과 'course' 테이블을 조인하여, 특정 학생이 특정 year와 semester에 수강한 section의 course\_id, title, sec\_id, 그리고 grade을 조회한다. query3은 query2의 결과를 바탕으로 'project' 테이블과 'instructor' 테이블을 조인하여, 특정 학생(s\_id)이 들은 section(course\_id, sec\_id, year, semester)에서 수행한 프로젝트의 정보인 num, name, max\_score, I.name, 그리고 score를 조회한다. query4는 'project' 테이블에서 집계함수를 통해 query2의 결과인 각 section에 대한 프로젝트의 총 개수(count)와 평균 점수(avg)를 계산한 결과를 검색한다.

String query1 = """

SELECT ID, name, dept\_name

FROM student

WHERE ID = ?

""";

String query2 = """

SELECT T.course\_id, C.title, T.sec\_id, T.grade

FROM takes as T, course as C

WHERE T.course\_id = C.course\_id

AND (T.ID, T.year, T.semester) = (?, ?, ?)

""";

String query3 = """

SELECT P.num, P.name, P.max\_score, I.name, P.score

FROM project as P, instructor as I

WHERE P.i\_id = I.id

AND (P.s\_id, P.course\_id, P.sec\_id, P.year, P.semester) = (?, ?, ?, ?, ?)

""";

String query4 = """

SELECT count(\*) as count, avg(score) as avg

FROM project

WHERE (s\_id, course\_id, sec\_id, year, semester) = (?, ?, ?, ?, ?)

""";

**4. 프로그램의 로직**

프로그램의 메인 함수는 다음과 같다. 우선 데이터베이스에 연결한 후 사용자로부터 입력을 받아 다양한 조회 기능을 수행한다.

static final String DB\_URL = "----";

static final String USER = "root";

static final String PASS = "1234";

static Connection conn;

static final Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

try {

conn = DriverManager.getConnection(DB\_URL, USER, PASS);

int command;

while (true) {

System.out.print("조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과, 4.종료): ");

command = sc.nextInt();

if (command == 1)

searchMixCourse();

else if (command == 2)

sectionTimeAndClassroom();

else if (command == 3)

searchProject();

else if (command == 4)

break;

System.out.println("\n");

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

if (conn != null) {

try {

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

}

다음은 융합과목을 조회하는 첫 번째 기능을 구현한 함수의 핵심 로직이다. 앞서 설명한 2개의 쿼리에 대한 pstmt를 준비하고, pstmt1의 값은 사용자가 입력한 연도와 학기 값을 이용하여 세팅한다. 이후 반복문을 돌며 쿼리의 결과를 출력하고, 이중 반복문을 이용하여 첫 쿼리의 결과로 세팅된 pstmt2를 리스트로 출력한다.

PreparedStatement pstmt1 = conn.prepareStatement(query1);

PreparedStatement pstmt2 = conn.prepareStatement(query2);

System.out.print("┌ 연도 학기: ");

int year = sc.nextInt();

String semester = sc.next();

// WHERE (S.year, S.semester) = (?, ?)

pstmt1.setInt(1, year);

pstmt1.setString(2, semester);

try (ResultSet rs1 = pstmt1.executeQuery()) {

while (rs1.next()) {

// SELECT course\_id, title, credits

String course\_id = rs1.getString("course\_id");

System.out.print("└ course id: " + course\_id);

System.out.print(" | title: " + rs1.getString("title"));

System.out.println(" | credits: " + rs1.getString("credits"));

// WHERE course\_id = ?

System.out.println("\tdepartment list");

pstmt2.setString(1, course\_id);

try (ResultSet rs2 = pstmt2.executeQuery()) {

while (rs2.next()) {

// SELECT dept\_name

System.out.println("\t└ name: " + rs2.getString("dept\_name"));

}

}

}

}

다음은 수업시간표 및 강의실 조회를 위한 두 번째 기능을 구현한 함수의 핵심 로직이다 사용자로부터 연도와 학기를 입력 받고, 해당 연도와 학기에 개설된 수업의 강좌 ID, 강좌 제목, 섹션 ID를 데이터베이스에서 조회하고 출력한다. 이를 위해 pstmt1을 사용하여 첫 번째 쿼리를 준비하고 반복 실행한다. 조회된 각 수업에 대하여 pstmt2를 설정하고 두 번째 쿼리를 반복 실행한다. 특정 강좌 ID와 섹션 ID에 해당하는 수업의 요일, 시작 시간, 종료 시간, 건물명, 강의실 번호를 리스트로 출력할 수 있다.

PreparedStatement pstmt1 = conn.prepareStatement(query1);

PreparedStatement pstmt2 = conn.prepareStatement(query2);

System.out.print("┌ 연도 학기: ");

int year = sc.nextInt();

String semester = sc.next();

// WHERE (S.year, S.semester) = (?, ?)

pstmt1.setInt(1, year);

pstmt1.setString(2, semester);

try (ResultSet rs1 = pstmt1.executeQuery()) {

while (rs1.next()) {

// SELECT S.course\_id, C.title, S.sec\_id

String course\_id = rs1.getString("S.course\_id");

String sec\_id = rs1.getString("S.sec\_id");

System.out.print("└ course id: " + course\_id);

System.out.print(" | title: " + rs1.getString("C.title"));

System.out.println(" | section id: " + sec\_id);

// AND (R.course\_id, R.sec\_id, R.year, R.semester) = (?, ?, ?, ?)

pstmt2.setString(1, course\_id);

pstmt2.setString(2, sec\_id);

pstmt2.setInt(3, year);

pstmt2.setString(4, semester);

try (ResultSet rs2 = pstmt2.executeQuery()) {

while (rs2.next()) {

// SELECT T.day, T.start\_time, T.end\_time, R.building, R.room\_number

System.out.print("\t└ day: " + rs2.getString("T.day") + ", ");

System.out.print("start time: " + rs2.getString("T.start\_time") + ", ");

System.out.print("end time: " + rs2.getString("T.end\_time") + ", ");

System.out.print("building: " + rs2.getString("R.building") + ", ");

System.out.println("room no: " + rs2.getString("R.room\_number"));

}

}

}

}

다음은 프로젝트 수행결과 조회를 위한 세 번째 기능을 구현한 함수의 핵심 로직이다. 우선 학번, 연도, 학기를 입력 받는다. 이 정보는 프로젝트 조회에 필요한 파라미터로 활용된다. pstmt1, pstmt2, pstmt3, pstmt4를 각각의 SQL 쿼리와 함께 초기화한다. 이 쿼리들은 위의 SQL 설명과 같이 학생 정보, 수강 정보, 프로젝트 세부 사항 및 프로젝트 성적의 평균과 총계를 출력하는데 사용된다. 입력된 값을 기반으로 pstmt1과 pstmt2를 실행하여 학생의 ID, 이름, 소속 학과 이름과 해당 학생이 특정 연도의 학기에 수강한 강좌의 ID, 제목, 섹션 ID, 그리고 성적을 반복적으로 출력한다. 해당 결과로 나온 각 section의 정보를 이용하여, pstmt3를 실행하여 학생이 참여한 프로젝트 번호, 프로젝트 이름, 최대 점수, 지도 교수 이름, 학생의 프로젝트 점수를 반복 출력한다. 마지막으로 pstmt4를 실행하여 학생이 참여한 프로젝트의 총 개수와 평균 점수를 출력한다.

PreparedStatement pstmt1 = conn.prepareStatement(query1);

PreparedStatement pstmt2 = conn.prepareStatement(query2);

PreparedStatement pstmt3 = conn.prepareStatement(query3);

PreparedStatement pstmt4 = conn.prepareStatement(query4);

System.out.print("┌ 연도 학기 학번: ");

int year = sc.nextInt();

String semester = sc.next();

String s\_id = sc.next();

// WHERE ID = ?

pstmt1.setString(1, s\_id);

// AND (T.ID, T.year, T.semester) = (?, ?, ?)

pstmt2.setString(1, s\_id);

pstmt2.setInt(2, year);

pstmt2.setString(3, semester);

try (ResultSet rs1 = pstmt1.executeQuery()) {

// SELECT ID, name, dept\_name

rs1.next();

System.out.print("ID: " + rs1.getString("ID"));

System.out.print(", name: " + rs1.getString("name"));

System.out.println(", dept name: " + rs1.getString("dept\_name"));

try (ResultSet rs2 = pstmt2.executeQuery()) {

while (rs2.next()) {

// SELECT T.course\_id, C.title, T.sec\_id, T.grade

String course\_id = rs2.getString("T.course\_id");

String sec\_id = rs2.getString("T.sec\_id");

System.out.print("\t└ course id: " + course\_id);

System.out.print(", title: " + rs2.getString("C.title"));

System.out.print(", sec id: " + sec\_id);

System.out.println(", grade: " + rs2.getString("T.grade"));

try {

// AND (P.s\_id, P.course\_id, P.sec\_id, P.year, P.semester) = (?, ?, ?, ?, ?)

pstmt3.setString(1, s\_id);

pstmt3.setString(2, course\_id);

pstmt3.setString(3, sec\_id);

pstmt3.setInt(4, year);

pstmt3.setString(5, semester);

try (ResultSet rs3 = pstmt3.executeQuery()) {

while (rs3.next()) {

// P.num, P.name, P.max\_score, I.name, P.score

System.out.print("\t\t└ project num: " + rs3.getString("P.num"));

System.out.print(", name: " + rs3.getString("P.name"));

System.out.print(", score: " + rs3.getInt("P.score"));

System.out.print("/" + rs3.getInt("P.max\_score"));

System.out.println(", instructor name: " + rs3.getString("I.name"));

}

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

// WHERE (s\_id, course\_id, sec\_id, year, semester) = (?, ?, ?, ?, ?)

pstmt4.setString(1, s\_id);

pstmt4.setString(2, course\_id);

pstmt4.setString(3, sec\_id);

pstmt4.setInt(4, year);

pstmt4.setString(5, semester);

try (ResultSet rs4 = pstmt4.executeQuery()) {

// SELECT count(\*) as count, avg(score) as avg

rs4.next();

System.out.print("\t\t\*\* total count: " + rs4.getInt("count"));

System.out.println(", average score: " + rs4.getInt("avg") + " \*\*");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

**5. DB 데이터**

다음과 같다.

**6. Java 프로그램 실행 결과**

다음과 같다.

**7. 정확성 검증**

다음과 같다.

**8. Java 프로그램 소스코드**

다음과 같다.

**9. 기타**

없음.

**10. 설계 수정**

**10.1. 기능 구현 불가능의 이유**: ERD의 project entity set을 테이블 스키마로 변환하는 과정에서, name과 max\_score 속성을 옮기는 것이 누락되었다. 따라서, “3번 프로젝트 수행결과 조회” 과제를 수행함에 있어 해당 컬럼을 사용할 수 없게 되었고, 결과적으로 과제가 요구하는 프로젝트 이름과 배점을 출력하는 부분의 기능을 구현하는 것이 불가능하다.

**10.2. ER modeling의 수정**: project entity set에 max\_score(배점) 속성을 추가하였다. 기존 project entity set에서 해당 속성이 누락되어 최종 테이블 스키마에도 누락되었기에 다음과 같이 수정하였다.

텍스트, 스케치, 그림, 친필이(가) 표시된 사진

자동 생성된 설명(기존) 텍스트, 스케치, 그림, 친필이(가) 표시된 사진

자동 생성된 설명(수정 후)

**10. 3. 변환규칙에 의한 테이블 스키마 변환의 수정**: weak entity set이었던 project를 새롭게 추가된 max\_score 속성과 기존 변환에서 누락된 name 속성까지 모두 포함해서 변환한다.

텍스트, 스케치, 평면도, 도표이(가) 표시된 사진

자동 생성된 설명(기존) 텍스트, 스케치, 그림, 평면도이(가) 표시된 사진

자동 생성된 설명(수정 후)

**10.4. 수정된 RDB 스키마 다이어그램**: 다음 장의 이미지와 같다.

텍스트, 그림, 스케치, 평면도이(가) 표시된 사진

자동 생성된 설명

**최종 스키마:** 최종적으로 완성된 스키마는 다음과 같다.

*Department (dept\_name, building, budget)*

*Course (course\_id, title, credits)*

*Classroom (building, room\_number, capacity)*

*Time\_slot (time\_slot\_id, day, start\_time, end\_time)*

*Instructor (ID, name, salary, dept\_name)*

*foreign key (dept\_name) references Department (dept\_name)*

*Student (ID, name, tot\_cred, dept\_name)*

*foreign key (dept\_name) references Department (dept\_name)*

*Teaches (ID, course\_id, sec\_id, year, semester)*

*foreign key (ID) references Instructor (ID)*

*foreign key (course\_id, sec\_id, year, semester) references Section (course\_id, sec\_id, year, semester)*

*Takes (ID, course\_id, sec\_id, year, semester, grade)*

*foreign key (ID) references Student (ID)*

*foreign key (course\_id, sec\_id, year, semester) references Section (course\_id, sec\_id, year, semester)*

*Project (s\_id, course\_id, sec\_id, year, semester, num, name, i\_id, max\_score, score)*

*foreign key (s\_id, course\_id, sec\_id, year, semester) references Takes (ID, course\_id, sec\_id, year, semester)*

*foreign key (i\_id, course\_id, sec\_id, year, semester) references Teaches (ID, course\_id, sec\_id, year, semester)*

*Section (course\_id, sec\_id, year, semester)*

*foreign key (course\_id) references Course (course\_id)*

*Course\_dept (course\_id, dept\_name)*

*foreign key (course\_id) references Course (course\_id)*

*foreign key (dept\_name) references Department (dept\_name)*

*Prereq (course\_id, prereq\_id)*

*foreign key (course\_id) references Course (course\_id)*

*foreign key (Prereq\_id) references Course (course\_id)*

*Advisor (s\_id, i\_id)*

*foreign key (s\_id) references Student (ID)*

*foreign key (i\_id) references Instructor (ID)*

*Rooms (course\_id, sec\_id, year, semester, time\_slot\_id, day, start\_time, building, room\_number)*

*foreign key (building, room\_number) references Classroom (building, room\_number)*

1. 표지이미지는 Dall-E 3에 의해 생성됨 [↑](#footnote-ref-1)