데이터베이스

팀 프로젝트 4 보고서

제 6조

남 중혁

김 민정

정 문경

업무 분담

남 중혁 : 기획 총괄, JDBC 연결, Student View time table

김 민정 : Log in, Student report

정 문경 : Instructor Course report, Instructor Advisee report

Introduction

이 프로젝트에서는 JDBC를 활용하여 앞의 프로젝트에서 사용한 University Database에서 사용자가 필요한 정보를 얻을 수 있는 Java application을 구현한다.

본 프로젝트의 목표는 다음과 같다.

* SQL을 활용한 데이터베이스에서 원하는 정보의 습득
* JDBC를 통하여 프로그래밍 언어를 이용한 SQL 사용

Design

학생 또는 교수인 주어진 사용자는 각자의 신분에 따라 본인의 이수한 교과 및 시간표, 또는 강의 및 지도 학생에 관한 정보를 얻는다.

이와 같은 작업을 수행하려면 각 사용자는 application이 JDBC driver를 통하여 University Database에 연결된 다음 사용자 본인의 ID와 이름을 입력하여 구성원 인증을 해야 한다. 구성원 인증은 student table과 instruction table에서 각각 입력한 ID와 이름을 가진 tuple의 존재 여부를 확인하여 진행된다. 인증에 실패하면 application은 올바른 정보가 입력되거나 프로그램이 종료될 때까지 반복하여 ID와 이름을 요구할 것이다.

학생인 사용자가 접속한 경우 사용자는 수강 내역 확인, 시간표 확인, 종료 중 하나를 선택한다.

수강 내역 확인을 선택하면 앞에서 입력한 ID와 이름으로 student table에서 소속과 누적 학점 점을, ID로 takes table에서 수강 이력이 있는 연도 및 학기를 파악한다. 그 다음, ID, 이름과 수강 연도, 수강 학기를 key로 하여 takes table과 course table로부터 각 교과의 과목코드, 과목명, 개설 학과, 학점, 성적을 얻는다. 필요한 각 정보를 얻으면 학점과 성적에 따라 매 학기의 평균평점을 산출한다. 이 때 당해학기의 성적 중 비어 있는 것이 있으면 평균평점을 산출하지 않는다.

시간표 확인을 선택하면 마찬가지로 ID를 key로 하여 takes table에서 수강 이력이 있는 연도 및 학기를 파악한다. 그 다음 각 학생의 ID과 선택한 연도 및 학기와 함께 table takes, course, section에서 과목 코드가 동일하고 time\_slot, section에서 time\_slot\_id가 동일한 tuple을 찾는 query를 수행한다.

교수인 사용자가 접속한 경우 사용자는 가장 최근에 강의한 교과, 지도 학생, 종료 중 하나를 선택한다.

최근 강의 교과를 선택하면 입력한 ID의 교수가 강의한 가장 최근 연도를 구한 다음, table teaches, course, section의 natural join에서 얻은 연도와 교수 ID에 맞는 tuple을 찾는다. 그 다음, 얻은 동일 연도의 tuple 중에서 가장 늦은 학기의 강의 목록을 구한다. 그러면 각 강좌의 수강 학생 정보를 takes table과 student table의 natural join에서 각 강좌의 과목 코드, 강의 번호, 연도, 학기가 일치하는 tuple을 구한다.

지도 학생을 선택하면 advisor table과 student table의 natural join에서 advisor table의 학생 ID와 student table의 ID가 같고 advisor table의 ID가 입력한 ID와 같은 모든 tuple을 구한다.

Implementation

이 프로젝트의 application은 총 5개의 파일로 구성되었다. 각 파일의 수행하는 내용은 다음과 같다.

* SchoolDBSystem.java : 최초 DBMS 연동, 사용자 인증 통제 및 신분에 따른 메뉴 표출
* ConnectionManager.java : JDBC 드라이버를 이용한 DBMS 연결
* UserInfoManager.java : 사용자 인증 및 신분 확인
* StudentDB.java : 입력 정보에 따른 원하는 학생 정보 질의
* InstructorDB.java : 입력 정보에 따른 원하는 교수 정보 질의

이하의 항목에서는 Log in, Student report, View time table, Course report, Advisee report의 구현 방안을 살펴본다.

* Log in

Log in은 최초 화면에서 사용자로부터 ID와 이름을 입력 받고 student table과 instructor table에서 각각 존재 여부를 확인한다. 이를 수행하는 SQL문은 각각 다음과 같다.

SELECT COUNT (\*) FROM student WHERE ID = ? and NAME = ?;

SELECT COUNT (\*) FROM instructor WHERE ID = ? and NAME = ?;

여기서 각 물음표에는 입력한 ID와 이름이 들어간다. 첫 번째 query 결과가 1이면 이 사용자는 학생이고, 두 번째 query 결과가 1이면 이 사용자는 교수이다. 만약 두 결과가 모두 0이면 잘못된 입력으로 판단하여 ID와 이름의 입력을 다시 요구한다.

* Student report

Student report는 Log in에서 파악한 ID로부터 학생이 수강한 교과 성적을 확인한. 이를 수행하는 SQL문은 각각 다음과 같다.

SELECT \* FROM student WHERE ID = ? and NAME = ?;

여기서 각 물음표에는 입력한 ID와 이름이 들어간다. Query 결과 얻은 tuple에서 소속과 취득한 학점을 파악한다. 그 다음 ID를 key로 하여 다음과 같이 강의를 수강한 이력이 있는 서로 다른 모든 학기를 내림차순으로 얻는다.

SELECT DISTINCT year, semester FROM takes WHERE ID = ?

ORDER BY YEAR DESC;

얻은 각 (연도, 학기) tuple별로 다음과 같이 각 교과의 성적, 학점과 과목코드, 과목명, 개설학과, 학점, 성적을 얻는다.

SELECT T.GRADE, C.CREDITS FROM takes T, course C WHERE T.ID = ?

AND T.YEAR = ? AND T.SEMESTER = ? AND T.COURSE\_ID = C.COURSE\_ID);

SELECT T.COURSE\_ID, C.TITLE, C.DEPT\_NAME, C.CREDITS, T.GRADE

FROM takes T, course C WHERE T.ID = ? AND T.YEAR = ?

AND T.SEMESTER = ? AND T.COURSE\_ID = C.COURSE\_ID);

여기서 각 물음표에는 ID, 얻은 연도, 학기가 들어간다. 각 교과의 성적을 얻으면 성적이 없는 교과가 있는 학기를 제외하고 평균평점을 계산한다. 이 때 문자로 쓰인 학점은 **float** gpToFloat(String)이 해당하는 숫자로 바꾸어 준다. 그리고 각 학기의 성적을 출력한다.

* View time table

우선 강의를 수강한 이력이 있는 모든 학기를 얻기 위해 앞의 절과 같이 다음과 같은 query를 수행한다.

SELECT DISTINCT year, semester FROM takes WHERE ID = ?

ORDER BY YEAR DESC;

얻은 각 (연도, 학기) tuple을 array에 저장한 다음, 사용자가 선택한 어떤 한 tuple의 과목 이름, 과목코드, 수업일, 시작 및 종료 시간을 다음과 같이 얻는다.

SELECT TAKES.COURSE\_ID, COURSE.TITLE, TIME\_SLOT.DAY,

TIME\_SLOT.START\_HR, TIME\_SLOT.START\_MIN, TIME\_SLOT.END\_HR,

TIME\_SLOT.END\_MIN

FROM TAKES, COURSE, SECTION, TIME\_SLOT

WHERE TAKES.COURSE\_ID = COURSE.COURSE\_ID and TAKES.ID = ? and

TAKES.YEAR = ? and TAKES.SEMESTER = ? and

SECTION.COURSE\_ID = TAKES.COURSE\_ID and

TIME\_SLOT.TIME\_SLOT\_ID = SECTION.TIME\_SLOT\_ID;

여기서 각 물음표에는 ID, 연도, 학기가 들어간다. 당해학기의 강좌 시간표를 취득하면 이를 화면에 출력한다.

* Course report

가장 최근 학기의 강의 및 부대 정보를 파악하기 위해 다음과 같이 사용자인 교수의 ID에 해당하는 것 중 연도가 가장 큰 값을 가지는 tuple을 얻는다.

SELECT \* FROM teaches NATURAL JOIN course NATURAL JOIN section

WHERE year = (SELECT MAX(year) FROM teaches WHERE d = ?) AND id = ?;

얻은 각 tuple을 순서대로 list에 저장한다. 이 때 더 최근 학기의 tuple이 list에 들어오면 list에 저장된 것을 모두 지우고 list에 넣는다. 동일 연도에서 각 학기의 선후 관계 비교는 **enum** Semester를 이용하여 수행할 수 있다. 이와 같은 과정을 통해 가장 최근 1개 학기를 얻을 수 있다.

List에 있는 가장 최근 학기의 각 강좌의 수업일, 시작 및 종료 시간을 다음과 같이 얻는다.

SELECT \* FROM time\_slot WHERE time\_slot\_id = ?;

여기서 물음표에는 앞에서 얻은 time\_slot\_id가 들어간다. 마지막으로 당해학기에 해당 과목 코드 및 강좌번호에 해당하는 학생을 아래와 같이 얻고 앞에서 구한 정보들을 포함하여 적절히 출력한다.

SELECT id, name, dept\_name, grade

FROM takes NATURAL JOIN student

WHERE (course\_id = ? AND sec\_id = ? AND year = ? AND semester = ?);

* Advisee report

다음과 같이 advisor table과 student table의 natural join으로 얻은 table에서 양 table에서 각 학생의 ID가 같고 사용자인 교수의 ID에 해당하는 각 학생의 (ID, 이름, 소속, 취득 학점) tuple을 구하는 query를 수행한다.

SELECT id, name, dept\_name, tot\_cred

FROM advisor NATURAL JOIN student

WHERE advisor.s\_id = student.id AND advisor.i\_id = ?;

Experiment

구현한 applictation의 동작을 확인하였다. 사용한 개발 환경은 JDK 1.7, ojdbc7, 시험 환경은 Windows 10이다.

사용에 앞서 ConnectionManager.java에서 DBMS와 연결에 필요한 아래와 같은 ID, 비밀번호, 연결 정보 등을 각 사용자의 환경에 알맞게 수정하여야 한다.

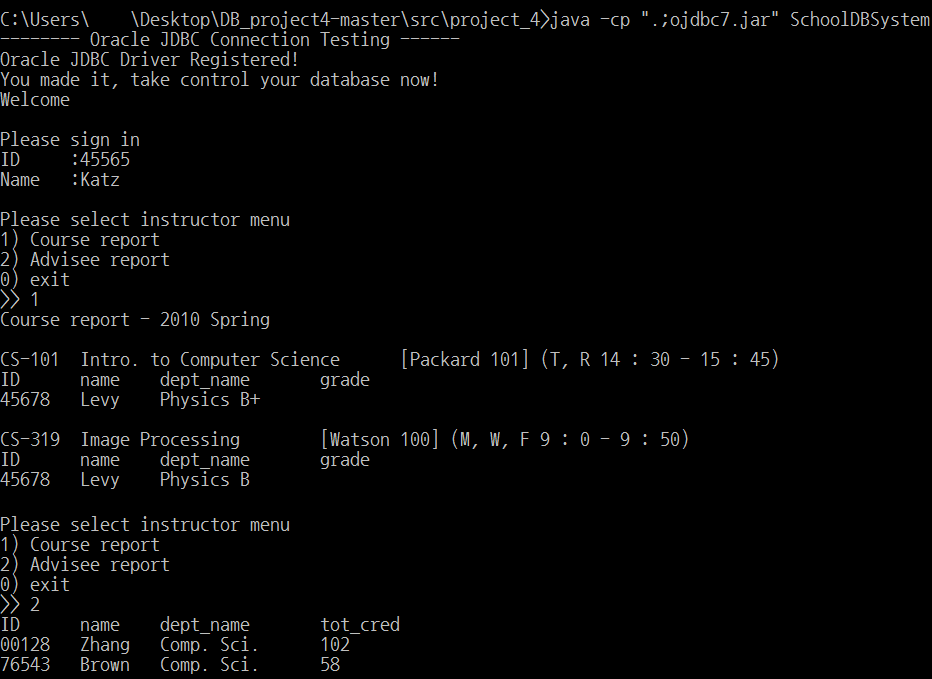
**private** **static** **final** String ID = "dbms\_id";

**private** **static** **final** String PASSWORD = "passwd";

**private** **static** **final** String CONNECTION\_INFO =

"jdbc:oracle:thin:@yourIp:yourPort:yourDB";

다음 화면은 사용자가 교수일 때의 동작 예시이다.



다음 화면은 사용자가 교수일 때의 동작 예시이다.

