

5조 프로젝트 보고서

[데이터베이스를 활용한 수강신청 시스템]

조원: 1513498 김세희

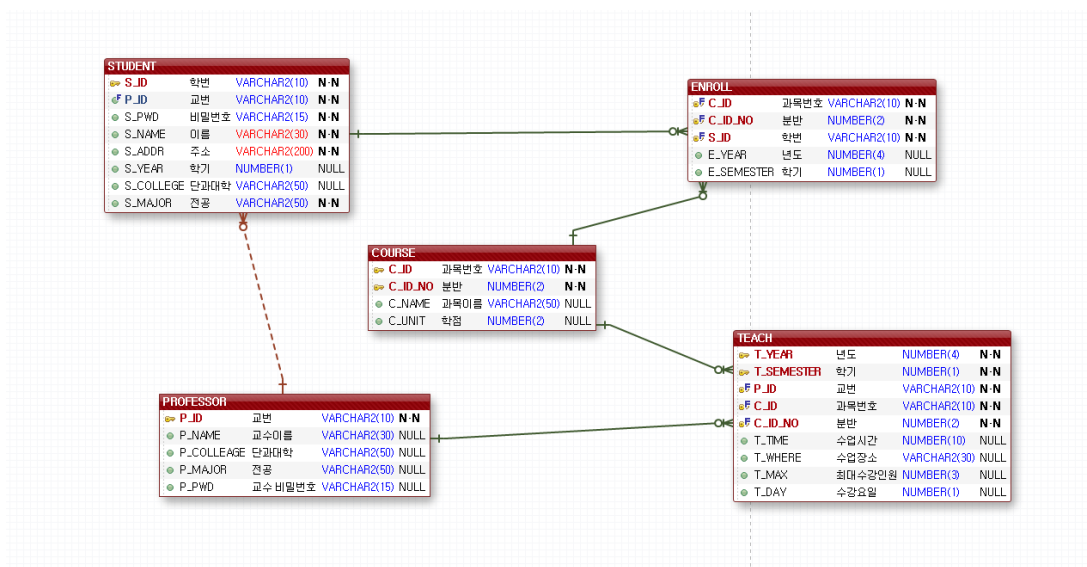
1410391 김예원

1315872 김유리나

1515655 임소희

팀원	담당 업무
임소희	로그인, 교수님 수업 및 학생조회 화면 구성 및 sql문 작성, 단과대학 화면 구성
김유리나	데이터베이스 설계, 사용자 정보 수정, 학생 수업조회 및 수강신청 화면 구성 및 sql문 작성
김예원	데이터베이스 입력, 학생 수업조회 및 수강신청 화면 구성 및 sql문 작성
김세희	교수님 강의 개설 화면 구성 및 sql문 작성, 시간표 구현 및 화면구성

1. DB 설계도



STUDENT, ENROLL, COURSE, TEACH, PROFESSOR 총 5개의 테이블로 구성하였고, 테이블끼리 중요하게 연결되는 컬럼은 primary key와 foreign key로 지정하여 참조 제약을 지정하였다.

STUDENT 테이블의 경우 학생에 관한 정보를 담은 테이블이며 학번을 primary key로 지정하여 해당 학번을 가진 학생의 수강 신청 과목번호, 분반 등의 정보를 담은 ENROLL 테이블과 식별 관계를 맺도록 하였다. 또한, COURSE 테이블은 과목 번호와 분반을 primary key로 지정하여 ENROLL 테이블과 식별 관계를 맺어 해당 과목에 대한 정보를 담았다. PROFESSOR 테이블은 교수 번호를 primary key로 지정하여 해당 교수의 수강 과목에 대한 수업 시간, 수업 장소, 최대 수강 인원 등의 정보를 담고, 년도와 학기를 primary key로 지정한 TEACH 테이블과 식별 관계를 맺도록 하였다.

2. 주요 SQL 질의문 및 커서 소개

<function.sql> 함수를 이용한 현재 년도와 학기 출력 sql

1) Date2EnrollYear : 현재 날짜를 매개변수로 받아와 년도와 월을 계산하여 변수에 저장하고 만약, 월이 1보다 크거나 같고 10보다 작거나 같을 시 동년도로 return하고 그 외의 경우 다음 년도로 return한다.

2) Date2EnrollSemester : 현재 날짜를 매개변수로 받아와 월을 계산하여 변수에 저장하고 만약, 월이 5보다 크고 10보다 작을 시 2학기로 그 외의 경우는 1학기로 return한다.

<insertEnroll.sql> 프로시저를 이용해 사용자 정의 에러 생성

→ 최대학점 초과여부, 동일 과목 신청 여부, 수강인원 초과 여부, 신청과목 중 시간 중복 여부 등에 관한 에러 처리

→ InsertEnroll: 학번, 과목번호, 분반, 결과 값을 매개변수로 가지는 프로시저를 생성하여 학생이 수강 신청 시 에러 처리에 대한 sql문을 진행하였다. 사용자 정의 에러로 too_many_sumCourseUnit(최대학점 초과여부), too_many_course(동일 과목 신청 여부), too_many_students(수강신청 인원 초과 여부), duplicate_time(신청 과목들 중 시간 중복 여부) 을 선언하고 nYear, nSemester, nSumCourseUnit, nCourseUnit, nCnt, nTeachMax 등의 변수도 선언하였다.

우선, Date2EnrollYear과 Date2EnrollSemester함수를 사용하여 년도와 학기를 알아낸 후 nYear, nSemester에 저장하였다.

/* 에러 처리 1: 최대학점 초과여부 */

```
SELECT SUM(c.c_unit)
INTO    nSumCourseUnit
FROM    course c,enroll e
WHERE   e.s_id = sStudentId and e.e_year = nYear and e.e_semester = nSemester and e.c_id = c.c_id and e.c_id_no = c.c_id_no;

SELECT c_unit
INTO    nCourseUnit
FROM    course
WHERE   c_id = sCourseId and c_id_no = nCourseIdNo;

IF (nSumCourseUnit + nCourseUnit >18) THEN
    RAISE too_many_sumCourseUnit;
END IF;
```

첫 번째로, 최대학점 초과여부에 대해서는 매개변수로 받은 sStudentId를 가진 학생이 nYear, nSemester, c_id, c_id_no에 해당하는 과목을 수강 신청했을 때, 받을 수 있는 학점을 계산하여 총 합을 nSumCourseUnit에 저장한다. 그리고 매개변수로 받은 sCourseId, nCourseNo에 해당하는 과목의 학점을 nCourseUnit에 저장하고 만약 nSumCourseUnit + nCourseUnit의 합이 18학점을 넘을 경우, 최대학점을 초과 에러를 발생시킨다.

/* 에러 처리 2 : 수강신청 인원 초과 여부 */

```
SELECT t_max
INTO    nTeachMax
FROM    teach
WHERE   t_year= nYear and t_semester = nSemester
    and c_id = sCourseId and c_id_no= nCourseIdNo;

SELECT COUNT (*)
INTO    nCnt
FROM    enroll
WHERE   e_year = nYear and e_semester = nSemester
    and c_id = sCourseId and c_id_no = nCourseIdNo;
IF (nCnt >= nTeachMax)
THEN
    RAISE too_many_students;
END IF;
```

두 번째로, 수강신청 인원 초과 여부에 대해서는 해당 년도, 학기에 매개변수로 받은 sCourseId, nCourseIdNo에 해당하는 과목의 최대 수강 인원(t_max)의 값을 teach테이블에서 찾아 nTeachMax에 저장하고, enroll테이블에서 찾아 count한 후, nCnt에 저장한다. 만약 그 과목을 신청했을 때, nCnt값이 nTeachMax를 넘을 경우 수강신청 인원 초과 에러를 발생시킨다.

```

/* 에러 처리 3 : 신청한 과목들 시간 중복 여부 */
SELECT COUNT(*)
INTO nCnt
FROM
(
    SELECT t_time,t_day
    FROM teach
    WHERE t_year=nYear and t_semester = nSemester and
          c_id = sCourseId and c_id_no = nCourseIdNo
    INTERSECT
    SELECT t.t_time, t.t_day
    FROM teach t, enroll e
    WHERE e.s_id=sStudentId and e.e_year=nYear and e.e_semester = nSemester and
          t.t_year=nYear and t.t_semester = nSemester and
          e.c_id=t.c_id and e.c_id_no=t.c_id_no
);
IF (nCnt > 0)
THEN
    RAISE duplicate_time;
END IF;

```

세 번째로, 신청과목 중 시간 중복 여부에 대해서는 sStudentId를 가지는 학생이 해당 년도, 학기에 c_id와 c_id_no에 해당하는 과목을 신청했을 때의 수강 시간(t_time)과 수강 요일(t_day) 과 동일한 시간과 요일의 과목을 신청했을 경우를 count하여 nCnt에 저장하고 그 값이 0을 초과할 경우, 시간 중복 에러를 발생시킨다.

이러한 에러들이 없을 경우 정상적으로 수강 신청을 등록할 수 있고 result로 하여금 '수강신청 등록이 완료되었습니다.'의 결과 값을 출력시켜준다.

<insertTeach.sql>

교수번호(professorId), 과목번호(pCourseId), 과목이름(pCourseName), 최대 수강 인원(pCourseMax), 수강 요일(teachDay), 수강 시간(teachTime), 수강 장소(teachWhere)을 매개변수로 가지는 프로시저를 생성해 교수님이 수강 등록 시 에러처리에 관한 sql문을 진행하였다. exist_course(이미 있는 수업), professor_already_have_course(같은 시간, 같은 요일에 강의가 이미 있는 경우), where_already_have_course(해당 장소에 이미 같은 시간의 강의가 있는 경우) 3가지 사용자 정의 에러를 선언하고 필요한 변수들을 지정하였다.

먼저 Date2EnrollYear, Date2EnrollSemester 함수를 이용하여 현재 년도와 학기를 nYear, nSemester에 저장한다.

```

/* 에러 처리 1: 이미 있는 수업 */
SELECT count(*)
INTO nCountCourse
FROM teach
WHERE c_id = pCourseId and c_id_no = pCourseIdNo and t_semester = nSemester and t_year = nYear;

IF (nCountCourse>0) THEN
    RAISE exist_Course;
END IF;

```

첫 번째로, 이미 있는 수업의 경우 해당 년도와 학기에 pCourseId라는 과목번호와 pCourseIdNo라는 분반인 과목이 teach테이블에 있을 경우 count를 하여 nCountCourse에 저장한다. 만약, nCountCourse가 0을 초과할 경우 동일 과목을 개설한 것이 되므로 exist_Course에러를 발생시킨다.

```

/* 에러 처리 2: 교수님이 같은 시간, 같은 요일에 강의 있는 경우 */
SELECT COUNT(*)
INTO nProfessorCourse
FROM teach
WHERE p_id=professorId and t_time = pTeachTime and t_day = pTeachDay and t_semester = nSemester and t_year = nYear;

IF (nProfessorCourse >0) THEN
    RAISE professor_already_have_course;
END IF;

```

두 번째로, 교수님이 같은 시간, 같은 요일에 강의가 있는 경우 해당 년도와 학기에 pTeachTime이라는 시간, pTeachDay라는 요일에 해당하는 강의가 teach테이블에 있을 경우 count하여 nProfessorCourse에 저장한다. 만약 nProfessorCourse의 값이 0보다 클 경우 professor_already_have_course에러를 발생시킨다.

```

/* 에러 처리 3: 수업 장소에 같은 시간, 같은 요일에 강의 있는 경우 */
SELECT count(*)
INTO nWhereCourse
FROM teach
WHERE t_year= nYear and t_semester = nSemester and t_where = pTeachWhere and t_time = pTeachTime and t_day = pTeachDay;

IF (nWhereCourse>0) THEN
    RAISE where_already_have_course;
END IF;

```

세 번째로, 수업 장소에 같은 시간, 같은 요일에 강의가 있는 경우 해당 년도와 학기에 pTeachWhere, pTeachTime, pTeachDay를 모두 만족시키는 강의가 teach테이블에 존재할 시 count를 하여 nWhereCourse에 저장한다. 만약 nWhereCourse의 값이 0보다 클 경우 where_already_have_course에러를 발생시킨다.

<trigger.sql> <professor_trigger.sql>

```
nLength := length(:new.s_pwd);
IF (nLength<4) THEN RAISE underflow_length;
END IF;

IF (:new.s_pwd = replace(:new.s_pwd,' ','')) THEN nBlank := 0;
ELSE nBlank :=1;
END IF;

IF (nBlank!=0) THEN RAISE invalid_value;
END IF;

EXCEPTION
  WHEN underflow_length THEN
    RAISE_APPLICATION_ERROR(-20002,'암호 길이가 짧습니다. ');
  WHEN invalid_value THEN
    RAISE_APPLICATION_ERROR(-20003,'암호에 공란이 있습니다. ');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
```

사용자 정보 수정 시 student테이블의 update문이 실행되기 이전에 트리거를 발생시키는 트리거를 사용하여 암호 길이, 암호 공란에 대한 사용자 정의 에러를 구현하였다. 정의부에 underflow_length, invalid_value로 에러를 선언하고, length(s_pwd)로 선언한 암호의 길이와 replace(:new.s_pwd,' ','')로 선언한 암호의 공란을 nLength, nBlank 변수에 저장하여 만약 nLength가 4보다 작다면 underflow_length에러를 발생시키고 nBlank가 0이아니라면 공란이 발생하는 경우이므로 invalid_value에러를 발생시킨다.

<verify_enroll.sql> <verify_teach.sql> 사용자 수강 신청 조회 화면

```
OPEN student_course FOR
SELECT c.c_id,c.c_id_no,c.c_name,c.c_unit,t.t_time
from teach t,course c,enroll e
where e.e_year= nYear and e.e_semester = nSemester and t.c_id = c.c_id and c.c_id = e.c_id and t.c_id_no = c.c_id_no and c.c_id_no = e.c_id_no and s_id=sStudentId;
```

sStudentId, sumCourse, sumUnit의 매개변수 그리고 SYS_REFCURSOR 타입의 커서 변수 student_course를 매개변수로 받는 수강 신청의 결과를 확인할 수 있는 프로시저를 생성하였다. 먼저, Date2EnrollYear와 Date2EnrollSemester 함수를 이용하여 현재 년도와 학기를 nYear과 nSemester에 저장한 후, 해당 년도와 학기에 sStudentId를 가진 사용자의 신청 과목과 관련된 내용이 저장된 student_course라는 커서를 오픈, 정의한 뒤 enroll

테이블과 course 테이블에서 신청한 과목의 수를 count하고 신청한 학점을 sum하여
패치한 결과를 매개변수 sumCourse와 sumUnit에 할당한 후 그 값을 출력해주었다.
교수님 강의 개설 조회 화면도 마찬가지로 프로시저를 사용하여 구성하였다.

```
select COUNT(*)
into sumCourse
from enroll
where e_year = nYear and e_semester = nSemester and s_id = sStudentId;

select SUM(c.c_unit)
into sumUnit
from enroll e, course c
where e.e_year = nYear and e.e_semester = nSemester and e.s_id = sStudentId and e.c_id = c.c_id and e.c_id_no = c.c_id_no;

DBMS_OUTPUT.PUT_LINE('총 '||sumCourse||' 과목과 총 '||sumUnit||' 학점을 신청하였습니다.');
```

3. 웹 화면 구성 및 소스 설명

1) 공통

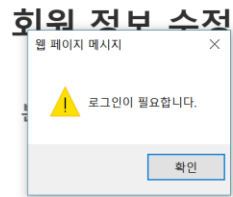
<dbconfig.jsp> 데이터베이스 연결하기

DriverManager와 데이터베이스를 연결하기 위한 소스이다. 위 소스는 데이터베이스 연결이 필요한 모든 파일들에 공통으로 사용되기에 따로 분리하였다. 다른 파일에는 `<%@ include file="dbconfig.jsp" %>`이라는 코드로 파일이 포함된다. Class.forName을 이용하여 JDBC 드라이버를 로딩하고, 로딩된 JDBC드라이버를 이용해 데이터베이스에 연결한다.

<main.jsp> 메인 화면



수강신청 시스템의 메인 화면이다. 화면 상단은 top.jsp, 중앙은 link.jsp, 하단은 footer.jsp로 구성된다. 메뉴의 모든 화면에서 우선적으로 session_id를 확인하여 로그인 상태를 확인한다. session_id가 null이 경우 즉, 로그인 전인 경우 login.jsp화면으로 이동한다.



<top.jsp> 메인 화면 상단 바 구성

세션 속성 명이 user, id, identity인 속성의 값을 읽어 session_name, session_id, session_identity 변수에 넣는다. 이는 현 사용자가 로그인 전인지, 학생인지, 교수님인지 판단하기 위함이다. session_id가 null인 경우와 student인 경우에는 학생 메인 화면을 보여주고, session_id가 professor인 경우에는 교수님 메인 화면을 보여준다.

<link.jsp> 메인 화면 중앙 슬라이더 구성

style을 지정하여 슬라이더 화면을 설계하였고, 총 다섯 장의 사진들이 반복되도록 하였다.

숙명여자대학교 수강신청 시스템



<login.jsp> 로그인



로그인

로그인이 필요합니다

아이디
비밀번호
로그인

로그인 화면은 아이디 입력 창, 비밀번호 입력 창, 로그인 버튼으로 구성되어있다.
로그인 버튼을 input type submit로 설정하여 login_verify.jsp로 폼을 전송한다.

<login_verify.jsp> 로그인하기

request.getParameter를 이용하여 login.jsp에서 입력한 폼의 내용을 받아온다.
학생로그인 sql문, 교수님 로그인 sql문 총 2개의 sql문은 연달아 실행해봄으로써
로그인을 시도할 것이다.

```

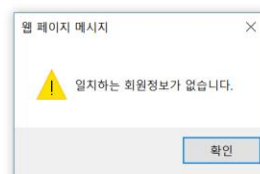
mysql = "select s_id,s_name from student where s_id=" + userID + " and s_pwd='" + userPassword + "'";
ResultSet rs = stmt.executeQuery(mysql);

if (rs.next()) {
    String name = rs.getString("s_name");
    int id = rs.getInt("s_id");
    session.setAttribute("id", id);
    session.setAttribute("user", name);
    session.setAttribute("identity", "student");
    response.sendRedirect("main.jsp");
} else {
    mysql2 = "select p_id,p_name from professor where p_id=" + userID + " and p_pwd='" + userPassword + "'";
    ResultSet prs = stmt.executeQuery(mysql2);
    if (prs.next()) {
        String name = prs.getString("p_name");
        int id = prs.getInt("p_id");
        session.setAttribute("id", id);
        session.setAttribute("user", name);
        session.setAttribute("identity", "professor");
        response.sendRedirect("main.jsp");
    } else {

```

학생 로그인 sql문은 student테이블에서 아이디와 비밀번호가 일치하는 것을 찾아 아이디와 이름을 가져온다. 이와 같은 방식으로 교수님 sql문도 professor테이블에서 아이디와 비밀번호가 일치하는 것을 찾아 아이디와 이름을 가져온다.

executeQuery문을 이용하여 Statement객체를 생성하고, ResultSet객체를 사용하여 select문을 통해 추출한 값들을 명시된 데이터 타입으로 컬럼 데이터를 가져온다. setAttribute를 이용하여 해당 세션명의 속성값을 읽어온 데이터로 정해주고 main.jsp화면으로 이동한다.



로그인 실패 시 “일치하는 회원정보가 없습니다.”라는 알림창이 뜨면서 다시 로그인 화면으로 이동한다.

<update_before.jsp> 비밀번호 확인

회원 정보 수정

본인 확인이 필요합니다.

비밀번호를 입력해주세요.

입력 완료

내 정보를 수정하기 위해 본인 확인을 하는 화면이다. 폼에 올바른 비밀번호 입력 시 update.jsp화면으로 폼을 전송한다. 비밀번호를 입력하지 않거나 틀릴 경우 “비밀번호가 틀립니다”라는 알림창이 뜬다.

<update.jsp> 내 정보 수정화면

회원 정보 수정

내 정보를 최신 정보로 바꿔주세요.

아이디 1410391

비밀번호 ●●●●●●●●

이름 김예원

주소 서울 용산구

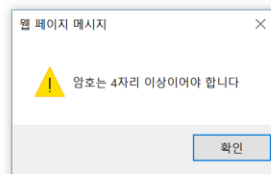
수정 완료

session_identity가 student인 경우와 professor인 경우의 sql문이 다르다. if-else문을 이용하여 각 경우를 구분하였다. 학생인 경우에는 student테이블에서 해당 학생의 이름, 주소,

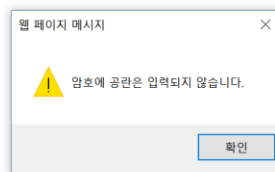
비밀번호를 select하고, 교수님인 경우에는 professor테이블에서 교수님의 이름, 비밀번호를 가져온다. createStatement를 이용하여 Statement객체를 생성하고, executeQuery를 통해 sql문을 실행한다. 학생과 교수님의 경우를 if문으로 분류하여 ResultSet를 통해 select한 값들을 받아온다. 수정완료 버튼을 누르면 update_verify.jsp파일로 폼이 전송된다.

<update_verify.jsp> 내 정보 수정

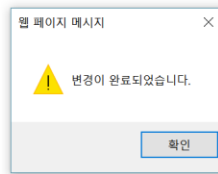
① 오류 알림 “암호는 4자리 이상이어야 합니다.”



② 오류 알림 “암호에 공란은 입력되지 않습니다.”



③ 성공 알림 "변경이 완료되었습니다."



Connection메소드인 `prepareStatement`를 이용한다. `setXXX`를 통해 패러미터에 대한 값을 지정한다. ‘?’의 위치에 알맞게 `setString`과 `setInt`로 값을 지정해 준 후에는 `executeUpdate`를 통해 sql문을 실행한다. 암호를 4자리보다 작게 입력한 경우, 암호에 공란을 넣은 경우에는 try catch문을 사용하여 수정에 실패했다는 알림창을 띄워준다.

<timetable.jsp> 나의 시간표 화면



시간표

	월요일	화요일	수요일	목요일	금요일
1					
2	기업가 정신과 창업 A555 - 1 문봉희 순천관B102 10:30~11:45	운영체제 C500 - 3 이상규 명신관122 10:30~11:45	기업가 정신과 창업 A555 - 1 문봉희 순천관B102 10:30~11:45	운영체제 C500 - 3 이상규 명신관122 10:30~11:45	
3	교양승마 C209 - 1 김윤진 명신관301 12:00~1:15	나를 바꾸는 나의 습관 B453 - 1 박영훈 명신관204 12:00~1:15	교양승마 C209 - 1 김윤진 명신관301 12:00~1:15	나를 바꾸는 나의 습관 B453 - 1 박영훈 명신관204 12:00~1:15	
		교양 요가 P555 - 2		교양 요가 P555 - 2	

수강 신청한 강의들을 시간표로 확인할 수 있는 화면이다. 시간표를 구현하기 위하여 table을 이용하였다. table의 가로축은 teach테이블의 t_day를 기준으로, 세로축은 t_time을 기준으로 구성하였다. 우선 과목 정보를 담을 수 있는 Course클래스를 만든다. 시간표에 과목 정보들을 넣어주기 위해 Course[6][5]배열 arr_course를 만들었다. createStatement을 이용하여 Statement객체를 만들어 sql문을 실행하여 해당 사용자가 소유중인(교수님의 경우에는 교수님께서 수업하시는, 학생의 경우에는 수강신청한) 수업들을 가져올 수 있다. 해당 resultSet값을 배열 arr_course에 넣어준다. 그 다음 if-else문을 이용하여 t_day(요일)가 1(월수), 2(화목), 3(금)인 경우를 각각 분류해준다. 만약 t_day가 1이라면 월, 수요일 수업이기에 arr_course[t_time][0]과 arr_course[t_time][2]에 과목 정보를 담을 수 있는 Course객체를 또 하나 생성해준다. 이런 방식으로 arr_course을 생성한 후에는 이중 for문을 이용하여 배열을 table형식으로 출력하면 된다. 각 cell마다 Course객체의 존재여부를 확인하여 null이라면 빈칸으로 두고, null이 아니면 arr_course[i][j]안에 있는 정보들을 읽어와 테이블에 출력해준다.

```
if (session_identity == "student") { //학생인 경우
    sql = "select c.c_name, t.t_where, t.t_day, t.t_time, p.p_name ,c.c_id, c.c_id_no"
        + " from course c, enroll e, teach t, professor p"
        + " where e.C_ID=c.C_ID and e.C_ID_NO=c.C_ID_NO and e.s_id = " + session_id
        + " and e.e_semester = DATE2ENROLLSEMESTER(SYSDATE) and e.e_year = DATE2ENROLLYEAR(SYSDATE) "
        + " and t.C_ID = e.c_id and t.C_ID_NO = e.c_id_no and p.p_id = t.p_id";
```

학생이 수강 신청한 과목 관련 정보들을 가져오는 sql문이다. course, enroll, teach, professor테이블에서 ①enroll테이블과 course테이블에서 과목번호와 분반 컬럼 값이 일치하고 ②등록한 학생의 id가 자신이며, ③수강신청한 연도가 현재 연도, ④수강 신청한 학기가 현재 학기이며, ⑤teach테이블과 enroll테이블의 과목번호와 분반이 컬럼값이 일치하고, ⑥professor테이블과 teach테이블의 교수님id가 같은 행의 과목이름, 수업장소, 수업시간, 수업요일, 교수님 성함, 과목번호, 분반을 select한다.

```
} else { //교수인 경우
    sql = "select c.c_name, t.t_where, t.t_day, t.t_time, p.p_name ,c.c_id, c.c_id_no"
        + " from course c, teach t, professor p"
        + " where t.t_semester = DATE2ENROLLSEMESTER(SYSDATE) and t.t_year = DATE2ENROLLYEAR(SYSDATE) "
        + " and t.C_ID = c.c_id and t.C_ID_NO = c.c_id_no and t.p_id = " + session_id;
}
```

교수님이 수업하시는 과목 관련 정보들을 가져오는 sql문이다. course, teach, professor테이블에서 ①teach테이블에서 수업하는 연도와 학기가 현재 연도, 학기와 일치하고, ②teach와 enroll테이블에서 과목번호와 분반 컬럼값이 일치하고,

③teach테이블에서 교수님id가 교수님 자신의 아이디와 같은 행의 과목이름, 수업장소, 수업요일, 수업시간, 교수님 성함, 과목번호, 분반을 select한다.

<goto_school.jsp> 단과대학 화면

The screenshot shows the Sookmyung Women's University website. The header includes the university's name and logo. The navigation bar contains links for '대학안내' (University Guide), '숙명 PRIME', '학부소개' (Faculty Introduction), '입학안내' (Admission Guide), and '커뮤니티' (Community). The main content area is titled '숙명여자대학교는 프라임사업에 준비된 최적의 대학입니다.' (Sookmyung Women's University is the best university prepared for the Prime Project). It features a tree diagram of the Faculty of Engineering (공과대학) with the following departments and student counts: 소프트웨어학부 (90명), ICT융합공학부 (140명), 화공생명공학부 (63명), 기계시스템학부 (50명), and 기초공학부 (80명). Below the tree, it lists specific research areas and programs, including '19대 미래성장동력 및 12대 주력산업과 연계한 학부별 타깃산업 선정'. The right sidebar contains '공지사항' (Notice) and '언론에서 본 숙명 공대' (Sookmyung Engineering in the Media) sections.

사용자의 소속 대학을 확인하여 해당 단과대학 사이트로 연결해주는 메뉴이다. 우선, 사용자의 이름, 아이디, 소속대학 정보를 담은 Info클래스를 정의해주었다. if else문을 사용하여 session_id를 확인한다. session_id가 null(비로그인)이라면 숙명여자대학교 홈페이지로 이동하고, 'student'나 'professor'라면 각각 해당하는 sql문을 사용하여 query를 실행해준다.


```

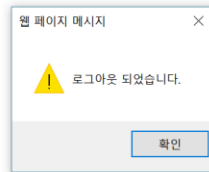
if (session_identity == "student") {
    sql = "select s.s_name, s.s_id, s.s_college"
        + " from student s"
        + " where s.s_id = " + session_id;
} else if (session_identity == "professor"){
    sql = "select p.p_name, p.p_id, p.p_college"
        + " from professor p"
        + " where p.p_id = " + session_id;
}
ResultSet rs = stmt.executeQuery(sql);
while (rs.next()) {
    if (session_identity == "student") {
        String s_name = rs.getString(1);
        String s_id = rs.getString(2);
        String s_college = rs.getString(3);
        info[0] = new Info(s_name, s_id, s_college, "0", "0", "0");
    }
    else if (session_identity == "professor") {
        String p_name = rs.getString(1);
        String p_id = rs.getString(2);
        String p_college = rs.getString(3);
        info[0] = new Info("0", "0", "0", p_name, p_id, p_college);
    }
}

```

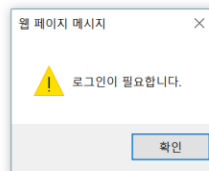
사용자가 학생인 경우, student테이블에서 s_id가 session_id와 같은 행의 이름, 아이디, 단과대학 정보를 select해서 가져온다. 사용자가 교수님인 경우, professor테이블에서 p_id가 session_id와 같은 행의 이름, 아이디, 단과대학 정보를 select해서 가져온다. ResultSet를 통해 select한 값들을 받아와 Info객체를 생성하며 넣어준다. 후에는 if else문과 equals를 사용하여 객체의 s_college 또는 p_college값을 개별적으로 확인하여 해당하는 단과대학 홈페이지로 연결시켜준다.

<logout.jsp> 로그아웃

① 로그아웃 성공



② 로그아웃 시 메뉴 접근 불가



session.invalidate()로 현재 생성된 세션을 무효화시킨다. 로그아웃 되었다는 알림창과 함께 main.jsp화면으로 돌아간다.

2) 학생

<main.jsp> 메인 화면



- 수강신청을 위한 포털의 메인은 상단의 top.jsp, 중앙의 link.jsp, 하단의 footer.jsp로 구성된다.
- 포털의 메인 화면이 나오면 오른쪽 상단의 로그인을 클릭하여 login.jsp로 이동한다.
- 학생 개인의 id와 password를 입력하고 로그인 한다.
- 입력 받은 id와 password와 학생 DB에 저장된 값이 맞지 않으면 로그인 불가능하다. id와 password가 학생 DB에 저장된 내용과 일치하면 별다른 경고창 없이 로그인에 성공한다.

<insert.jsp> 수강신청화면

- 메인 화면의 상단 바를 구성하는 top.jsp와 DriverManager와 데이터베이스를 연결하기 위한 소스가 포함된 dbconfig.jsp파일이 포함되어 있다.
- 과목번호(c_id), 분반(c_id_no), 과목명(c_name), 학점(c_unit), 요일(t_day), 시간(t_time), 장소(t_where)의 정보를 확인할 수 있다.

2018 년도 2 학기 수강신청

과목번호	분반	과목명	학점	요일	시간	장소	정원	수강신청
E777	1	컴퓨터활용	3	금	3:00~4:15	명신관102	10	신청
S789	1	아두이노 프로그래밍	3	월,수	9:00~10:15	명신관321	50	신청
F002	2	컴퓨터과학의이해	3	월,수	10:30~11:45	새힘관102	20	신청
F002	1	컴퓨터과학의이해	3	월,수	9:00~10:15	새힘관102	20	신청
E222	2	법학개론	3	화,목	10:30~11:45	명신관309	30	신청
E222	1	법학개론	3	화,목	9:00~10:15	명신관405	28	신청
F111	1	여성과 법	3	금	4:30~6:00	명신관304	25	신청
A464	3	데이터베이스프로그래밍	3	화,목	12:00~1:15	명신관308	30	신청
A464	2	데이터베이스프로그래밍	3	화,목	10:30~11:45	명신관308	30	신청
A464	1	데이터베이스프로그래밍	3	월,수	9:00~10:15	명신관309	30	신청
C800	3	인공지능	3	월,수	1:30~2:45	명신관309	30	신청
C800	2	인공지능	3	월,수	12:00~1:15	명신관309	30	신청
C800	1	인공지능	3	월,수	9:00~10:15	명신관309	6	신청

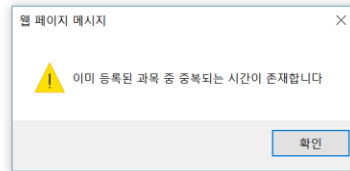
-로그인 후 상단의 다섯 개의 메뉴 중 수강신청을 위한 메뉴를 클릭하면 2학기에 개설된 강의정보를 조회할 수 있다.

E222	1	법학개론	3	화,목	9:00~10:15	명신관405	28	신청
F111	1	여성과 법	3	금	4:30~6:00	명신관304	25	신청
S333	2	경제학개론	3	월,수	10:30~11:45	명신관209	30	신청
S333	1	경제학개론	3	월,수	9:00~10:15	명신관209	30	신청
S890	2	자료구조	3	화,목	4:30~6:00	명신관611	40	신청
S890	1	자료구조	3	화,목	3:00~4:15	명신관611	40	신청
D779	2	소프트웨어공학	3	화,목	1:30~2:45	명신관612	43	신청
D779	1	소프트웨어공학	3	화,목	12:00~1:15	명신관305	30	신청
C178	2	교양 재즈댄스	2	월,수	1:30~2:45	새힘관304	30	신청
C178	1	교양 재즈댄스	2	월,수	12:00~1:15	새힘관304	30	신청
B123	2	컴퓨터수학	3	월,수	9:00~10:15	명신관408	30	신청
B123	1	컴퓨터수학	3	월,수	9:00~10:15	명신관408	6	신청
S100	2	소프트웨어의이해	3	월,수	1:30~2:45	명신관314	12	신청
S100	1	소프트웨어의이해	3	월,수	12:00~1:15	명신관314	12	신청
F133	2	컴퓨터네트워크	3	월,수	3:00~4:15	진리관211	19	신청
F133	1	컴퓨터네트워크	3	월,수	1:30~2:45	진리관211	19	신청
A711	1	시스템종합설계	3	월,수	12:00~1:15	명신관309	30	신청
O888	3	컴퓨터구조	3	화,목	3:00~4:15	명신관501	38	신청
O888	2	컴퓨터구조	3	화,목	1:30~2:45	명신관501	38	신청
O888	1	컴퓨터구조	3	화,목	12:00~1:15	명신관501	38	신청
B500	2	현대미술의 이해	3	화,목	1:30~2:45	명신관209	30	신청
B500	1	현대미술의 이해	3	화,목	12:00~1:15	명신관205	30	신청
A101	1	웹시스템설계	3	화,목	9:00~10:15	명신관211	6	신청

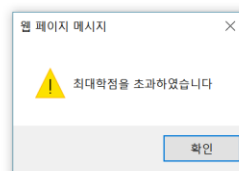
-수강신청에 필요한 내용들은 입력된 DB에 따라 과목번호, 분반, 과목명, 학점, 요일, 시간, 장소의 내용으로 확인 할 수 있다.

-switch문을 사용하여 수업시간(t_time)을 9:00~10:15/ 10:30~11:45/ 12:00~1:15/ 1:30~2:45/ 3:00~4:15/ 4:30~6:00으로 구분하였고, 수업요일(t_day)도 switch문을 사용하여 월,수/ 화,목/ 금요일로 구분하였다.

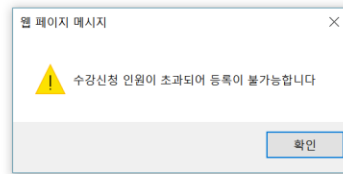
① 중복되는 시간이 존재할 경우



② 최대학점을 초과한 경우



③ 수강신청 인원이 초과한 경우



-신청된 강의는 not in (select c_id from enroll where s_id="" + session_id + "")

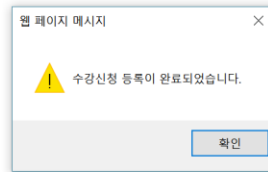
제외 연산문 not in을 사용하여 이미 신청된 과목은 분반이나 시간이 다르더라도 중복 신청이 불가능하게 구분하였다.

-원하는 강의의 오른쪽 맨 끝에 있는 신청버튼을 클릭하면 insert_verify로 넘어간다.

<insert_verify.jsp> 수강신청

-DriverManager와 데이터베이스를 연결하기 위한 소스가 포함된 dbconfig.jsp파일이 포함되어 있다.

-사용자는 원하는 강의를 신청버튼을 클릭하여 신청한다.



-중복되는 강의가 없으면 에러 메시지는 뜨지 않고 성공적으로 수강신청이 완료된다.

<select.jsp> 내 수강 정보 조회



수강정보 조회

과목번호	분반	과목명	학점	요일	시간	장소	수강신청
C500	3	운영체제	3	화,목	10:30~11:45	명신관122	삭제
B453	1	나를 바꾸는 나의 습관	3	화,목	12:00~1:15	명신관204	삭제
E222	1	법학개론	3	화,목	9:00~10:15	명신관405	삭제
A555	1	기업가 정신과 창업	3	월,수	10:30~11:45	순현관B102	삭제
C209	1	교양승마	2	월,수	12:00~1:15	명신관301	삭제
P555	2	교양 요가	3	화,목	1:30~2:45	명신관308	삭제

총 6 과목, 17 학점 신청하셨습니다.

-최종적으로 강의를 올바르게 신청 했는지 조회하는 페이지이다.

-메인 화면의 상단 바를 구성하는 top.jsp와 DriverManager와 데이터베이스를 연결하기 위한 소스가 포함된 dbconfig.jsp파일, 하단에 위치한 학교의 주소가 담긴 파일인 footer.jsp가 포함되어 있다.

-session_id가 null인 비로그인 상태라면 location.href = "login.jsp"을 사용해 로그인 페이지로 넘어갈 수 있도록 하였다.

-과목번호(c_id), 분반(c_id_no), 과목명(c_name), 학점(c_unit), 요일(t_day), 시간(t_time), 장소(t_where)의 정보를 확인할 수 있도록 구성했다.

-insert.jsp에서 사용된 switch문을 동일하게 사용해 수업요일(t_day)을 월, 수/ 화, 목/ 금요일로 구분하였고, 수업시간(t_time)을 9:00~10:15/ 10:30~11:45/ 12:00~1:15/ 1:30~2:45/ 3:00~4:15/ 4:30~6:00으로 구분하였다.

-원치 않은 강의는 href="delete.jsp"를 사용하여 삭제할 수 있다.

-신청한 총 과목 수와 과목의 총 학점을 확인할 수 있도록 sumCourse, sumUnit을 사용했다.

<delete.jsp> 수강 신청한 수업 삭제



수강정보 조회

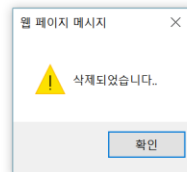
과목번호	분반	과목명	학점	요일	시간	장소	수강신청
C500	3	운영체제	3	화,목	10:30~11:45	명신관122	삭제
B453	1	나를 바꾸는 나의 습관	3	화,목	12:00~1:15	명신관204	삭제
E222	1	법학개론	3	화,목	9:00~10:15	명신관405	삭제
A555	1	기업가 정신과 창업	3	월,수	10:30~11:45	순현관8102	삭제
C209	1	교양승마	2	월,수	12:00~1:15	명신관301	삭제
P555	2	교양 요가	3	화,목	1:30~2:45	명신관308	삭제

총 6 과목, 17 학점 신청하셨습니다.

http://localhost:8080/testtest/delete.jsp?c_id=E222&c_id_no=1

-메인 화면의 상단 바를 구성하는 top.jsp와 DriverManager와 데이터베이스를 연결하기 위한 소스가 포함된 dbconfig.jsp파일이 포함되어 있다.

- sql문 "delete from enroll where c_id=" + c_id + " and c_id_no =" + c_id_no + " and s_id=" + session_id + ""로 수강 신청한 과목이 삭제된다.



3) 교수님

<main.jsp> 메인 화면

숙명여자대학교 수강신청 시스템



오른쪽 상단의 로그인을 클릭하면 login.jsp로 이동한다. 여기서 교수 id/password로 로그인한다.

```

<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="layout.css"
    media="screen and (max-width:1080px)">
  <title>데이터베이스를 활용한 수강신청 시스템입니다.</title>
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="colorbox.css">
  <link rel="stylesheet" href="layout.css"
    media="screen and (max-width:1080px)">
</head>
<body class="main">
  <%@ include file="top.jsp" %>
  <div id="containerwrap">
    <div id="container">
      <div class="section_title">
        <h1>
          <span><font face="나눔스퀘어라운드">숙명여자대학교 수강신청 시스템</font></span>
        </h1>
      </div>
    </div>
  </div>
  <%@ include file="link.jsp" %>
  <%@ include file="footer.jsp" %>
</body>
</html>

```

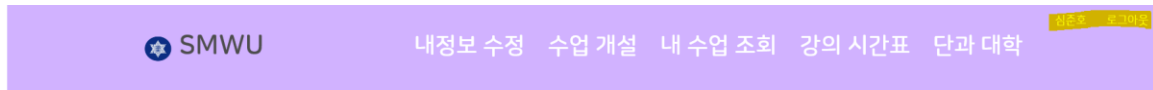
main.jsp는 top.jsp, link.jsp, footer.jsp로 이루어져 있으며 글꼴 변경은 으로 가능하다. top.jsp는 메뉴를 표시한다. 메뉴의 스타일과 메뉴 클릭 시 링크가 연결 되어 있다. link.jsp는 중앙의 이미지 슬라이드를 담당한다. 애니메이션을 적용하여 이미지가 자동으로 움직인다. 마지막으로 footer.jsp는 하단 부분으로, 학교 주소와 만든 사람 서명이 적혀있다.



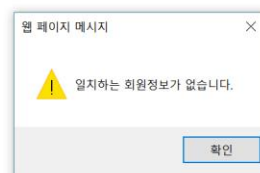
login.jsp는 로그인 창(id, password 입력란)을 보여준다. 여기에 값을 입력하고 Enter나 로그인 버튼을 누르면 login_verify.jsp로 넘어가 DB에 정보(id, pwd)가 존재하는지 매치 한다.

login_verify.jsp에서 createStatement()문을 이용해 sql문을 만든다. 이 sql문은 ResultSet으로 받아 사용할 수 있다 (rs = stmt.executeQuery(sql);). login_verify에서 사용될

sql문은 학생 DB부터 살펴본다. s_id가 login.jsp에서 입력한 userID와 같을 경우 s_id, s_name을 select 하는 것이다. rs.next()가 0이 아니면(=학생 DB에 존재해 받아온 값이 있으면), 받은 s_name과 s_id를 session 속성에 넘겨준다. identity 속성에 student도 넣어주고 main.jsp로 돌아오면 학생으로 로그인이 성공한 경우다.



rs.next()==0이면 학생 DB에 존재하지 않는 경우다. 따라서 교수 DB를 탐색하는 새로운 sql문을 만든다. 이전 방법과 동일하게 p_id가 입력한 userID와 일치하면 p_id, p_name을 가져온다. rs.next()가 0이 아니면 session에 해당 값들을 넣어주고 main.jsp로 이동한다.



교수 DB에도 없다면 DB에 존재하지 않는, 잘못된 정보를 입력한 경우다. 따라서 alert창으로 경고하고 login.jsp로 빠져 나오게 한다.

<create_course.jsp>, <createCourse_verify.jsp> 수업 개설

 SMWU

내정보 수정 수업 개설 내 수업 조회 강의 시간표 단과 대학

심중호 로그인

수업 개설

강의번호

강의 분반 번호

강의명

학점

최대 수강인원

수업 요일 설정

요일 선택

수업 시간 설정

시간선택

수업 장소 선택

추가

취소

메뉴에서 ‘수업 개설’을 클릭하면 create_course.jsp로 들어간다. create_course.jsp에선 수업 개설을 할 수 있는 폼이 나타난다. 이 페이지에선 폼의 디자인과 입력할 값들을 받는다. ‘추가’버튼을 누르면 createCourse_verify.jsp로 넘어가 강의 개설 가능 여부를 판단하고, ‘취소’버튼을 누르면 폼에 입력한 값들이 리셋된다.

createCourse_verify.jsp에선 기존의 course DB와 teach DB에서 추가할 수업이 없는지 확인 한 후 각 각 DB에 넣어준다. 이 때 검증해야 할 경우가 많으므로 하나의 SQL을 한번에 여러 개 사용할 수 있는 프로시저를 사용하였다. 우선 폼에서 입력한 값들을 다 받아서 프로시저 변수에 넣어준다. 변수만 바뀌고 동일한 sql문을 써서 컴파일 된 상태에 변수만 바꾸면 시간이 절약되고 가독성이 올라간다. callablestatement를 이용해 이러한 효과를 보장한다. preparecall(“call 프로시저명(?,?,?,?,?,?,?,?,?)”). stmt.setType(위치, 들어갈 값)을 이용하여 편하게 값을 넣을 수 있다. call InsertTeach(?,?,?,?,?,?,?,?,?)는 마지막 파라미터를 외부변수로 두어 나중에 결과값을 다른 변수에 담을 수 있게 보장한다. 값을 넣고 execute()를 하면 InsertTeach.sql이 실행된다.

<InsertTeach.sql>

/*InsertTeach.sql 위 설명 참조*/

프로시저의 기본구조는 다음과 같다.

CREATE OR REPLACE PROCEDURE 프로시저명(파라미터,파라미터,...)

IS

예외 및 내부변수 선언

BEGIN

SQL문...

EXCEPTION

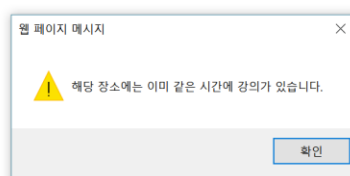
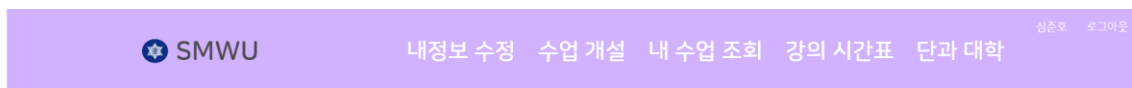
예외 발생시 수행

END;

/

여러 개의 sql문을 한번에 수행하기 위해 프로시저를 이용한다. 파라미터는 createCourse_verify.jsp에서 받은 값들을 받아 사용한다. 외부변수도 존재한다. IS 아래에 예외들과 사용할 변수들을 선언한다.

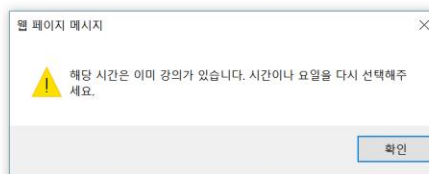
① 해당 요일, 시간에 강의실이 사용중인 경우



② 이미 있는 수업인 경우



③ 교수님이 같은 시간, 같은 요일에 강의가 있는 경우

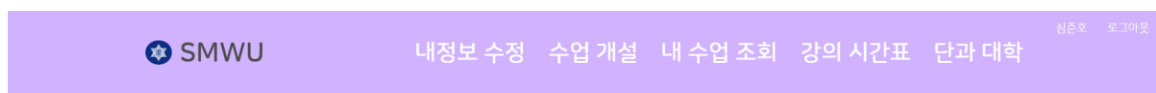


BEGIN에서는 sql문들을 적는다. select문 이전에 년도와 학기를 다른 function(Date2EnrollYear, Date2EnrollSemester)를 이용하여 알아낸다. 이 두 가지는 새로 등록한 수업 등록 시 정보뿐 아니라 개설 가능 여부를 판단하게도 해준다. 1 이미 있는 수업, 2 교수님이 같은 시간, 같은 요일에 강의가 있는 경우, 3 수업 장소에 같은 시간, 같은 요일에 강의가 있는 경우 에러를 발생시켜 EXCEPTION 구문에서 처리한다. 에러가 아니라면 정상적으로 수업을 등록시키는 sql문을 수행한다.

InsertTeach.sql에서 모든 조건을 충족하여 수업이 등록되면 '수업 등록이 완료되었습니다.'메시지를 외부변수에 넣고, createCourse_verify.jsp로 돌아와 메시지를 alert한다. alert창을 띄운 후 create_coursre.jsp로 돌아간다.

InsertTeach.sql에서 조건 충족을 못해 에러가 발생하면 에러 발생 시 해당하는 메시지를 외부변수로 받아 createCourse_verify.jsp로 돌아와 출력한다.

<select_course.jsp> 수업 조회



수업 시간표


과목번호	분반	과목명	학점	요일	시간	강소	수강생 조회	수업 삭제
A464	1	데이터베이스프로그래밍	3	월,수	9:00~10:15	명신관309	조회	삭제
A464	2	데이터베이스프로그래밍	3	화,목	10:30~11:45	명신관308	조회	삭제
A464	3	데이터베이스프로그래밍	3	화,목	12:00~1:15	명신관308	조회	삭제
C800	1	인공지능	3	월,수	9:00~10:15	명신관309	조회	삭제
C800	2	인공지능	3	월,수	12:00~1:15	명신관309	조회	삭제
C800	3	인공지능	3	월,수	1:30~2:45	명신관309	조회	삭제
D477	1	문학과 사랑의 테마	3	금	9:00~10:15	명신관621	조회	삭제
D477	2	문학과 사랑의 테마	3	금	10:30~11:45	명신관621	조회	삭제
D477	3	문학과 사랑의 테마	3	금	12:00~1:15	명신관621	조회	삭제
Q139	1	컴파일러	3	화,목	4:30~6:00	명신관211	조회	삭제
Q139	2	컴파일러	3	월,수	4:30~6:00	명신관211	조회	삭제
S789	1	아두이노 프로그래밍	3	월,수	9:00~10:15	명신관321	조회	삭제

총 12 과목, 36 학점 수업입니다.

교수로 로그인 했을 때 나타나는 메뉴 중 '내 수업 조회'를 누르면 넘어가는 페이지이다. 현재(이번 년도, 이번 학기) 내가 가르치는 과목이 나타난다. 교수로 로그인 하지 않은 경우는 login.jsp 페이지로 돌아간다.

전체 개설된 과목 중 현재 내가 가르치는 과목만 가져와 화면에 나타내기 위해선 sql문을 사용해야 한다. 전체 과목에 대해 sql문을 적용해야 하므로 컴파일을 미리 해놓고 변수만 바꿔 적용하는 게 더 효율적이다. 따라서 callableStatement를 사용하는 것이 바람직하다. SelectTimeTableProfessor(교수id, 커서, 총 과목수, 총 학점) 프로시저를 만든다. 이 프로시저로 이번 년도 이번 학기에 교수가 가르치는 과목의 정보를 넘기고 총 가르치는 과목 및 학점을 출력한다. 프로시저에서 넘긴 정보는 getType 형식으로 ResultSet이 받아서 출력 형태를 변환하는 경우(시간, 요일)는 스위치 문으로 변환해 출력한다. 이 페이지에서 수강생 조회와 과목을 삭제할 수 있는 링크도 존재한다.

<view_student.jsp> 수강생 확인

 SMWU

내정보 수정 수업 개설 내 수업 조회 강의 시간표 단과 대학

심준호 로그인

수강생 조회

학번	이름	전공
1315872	김유리나	소프트웨어학부
1513498	김세희	소프트웨어학부
1515655	임소희	소프트웨어학부
2180601	전정국	무용학부
2180602	박지민	순수미술학부

서울특별시 용인구 청파로 47길
COPYRIGHT (c) SOOKMYUNG UNIV. ALL RIGHTS RESERVED.

TOP

교수로 로그인 했을 때 select_course.jsp에서 '수강생 조회'를 클릭하면 넘어가는 페이지이다. 수강생 확인은 확인 링크가 존재하는 행의 과목만을 조회할 수 있다. view_student.jsp로 넘어갈 때 과목의 id와 분반 번호를 넘기기 때문이다. request.getParameter로 c_id와 c_id_no를 변수에 담는다.

```
mySQL = "select s id, s name, s major from student where s id in (select s id from enroll where c id = '"  
+ c_id + "' and c id no=" + c_id_no + ")";
```

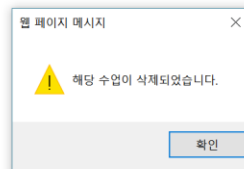
그 과목과 분반에 해당하는 과목을 수강하는 학생을 조회하기 위해서 sql문을 사용한다. enroll 테이블에서 변수로 받은 c_id, c_id_no이 동일한 학생을 찾아 student 테이블에서 그 학생의 이름, 학생의 전공을 선택한다.

변경 없이 검색만 하는 간단한 sql문이므로 createStatement로 sql문을 생성하고, executeQuery문 형태로 수행 가능하다. ResultSet에 값을 받아 표에 정보를 나타낸다.

<delete_course.jsp> 수업 삭제

수업 시간표

과목번호	분반	과목명	학점	요일	시간	장소	수강생 조회	수업 삭제
A464	1	데이터베이스프로그래밍	3	월,수	9:00~10:15	명신관309	조회	삭제
A464	2	데이터베이스프로그래밍	3	화,목	10:30~11:45	명신관308	조회	삭제
A464	3	데이터베이스프로그래밍	3	화,목	12:00~1:15	명신관308	조회	삭제
C800	1	인공지능	3	월,수	9:00~10:15	명신관309	조회	삭제
C800	2	인공지능	3	월,수	12:00~1:15	명신관309	조회	삭제
C800	3	인공지능	3	월,수	1:30~2:45	명신관309	조회	삭제
D477	1	문학과 사랑의 테마	3	금	9:00~10:15	명신관621	조회	삭제
D477	2	문학과 사랑의 테마	3	금	10:30~11:45	명신관621	조회	삭제
D477	3	문학과 사랑의 테마	3	금	12:00~1:15	명신관621	조회	삭제
Q139	1	컴파일러	3	화,목	4:30~6:00	명신관211	조회	삭제
Q139	2	컴파일러	3	월,수	4:30~6:00	명신관211	조회	삭제
S789	1	아두이노 프로그래밍	3	월,수	9:00~10:15	명신관321	조회	삭제



교수로 로그인 했을 때 select_course.jsp에서 '삭제'를 클릭하면 넘어가는 페이지이다. 수강생 확인과 마찬가지로, 삭제 링크가 존재하는 행의 과목만을 삭제할 수 있다. delete_course.jsp로 넘어갈 때 과목 id와 분반 번호를 넘기는 것도 동일하다. request.getParameter로 c_id와 c_id_no를 변수에 담는다.

변수로 넘겨받은 과목명과 과목 번호를 동시에 만족시키는 과목을 삭제해야 하므로 sql문을 사용한다. teach 테이블에서 변수로 받은 c_id, c_id_no과 현재 로그인한 교수 id(=session_id) 모두를 충족하는 행을 찾아 삭제한다.

간단한 sql문이므로 createStatement로 sql문을 생성하고, 변경(삭제)이 있으므로 executeUpdate문으로 sql문을 실행한다. 이 때 executeUpdate를 re라는 int 타입의 변수로 받는데, 성공 여부를 확인하기 위해서이다.

re! =0이면 성공이므로 alert창으로 '해당 수업이 삭제되었습니다.'라는 메시지를 띄우고, select_course.jsp(수업 조회)로 돌아간다.

re==0이면 실패이므로 alert창으로 오류가 났음을 알린다.

4. 중요하다고 생각하는 내용

임소희: 원하는 정보를 검색/수정 하는 것에 따라 어떤 sql문을 쓰는 것이 적절한지 고민을 하였다. 간단한 sql문은 createStatement, 복잡하거나 변수만 바뀐 채로 반복되어 사용되는 sql문은 callableStatement를 사용하는 등의 방식을 이용하였다. 또한 insert문으로 db 삽입을 할 때나 삭제를 할 때 참조 무결성을 주의했다. 순서가 다르거나 한쪽에 존재하면 foreign key로 인해 제약이 생겨 원하는 결과가 발생하지 않는 경우가 생겼기 때문이다.

김유리나: 실제 수강신청 시스템과 최대한 비슷하게 구현하기 위해서 trigger의 사용과 procedure, function을 이용한 여러 예외 사항들을 추가하였다. 또한 시스템이 정상 작동 할 수 있도록 db설계 시 필요한 컬럼과 컬럼의 제약사항을 잘 지키려고 노력하였다.

김예원: 수강신청 사이트가 원활하게 작동하기 위해 모든 예외 사항들을 고려하는 trigger문과 조건들이 중요하다고 생각하였다. 또한 sql문 조건 작성시 놓친 조건들이 없는지 꼼꼼하게 확인하기 위해 팀원들과 계속하여 논의하고 점검하였다.

김세희: 하나의 기능만 갖춘 사이트가 아닌 여러 페이지와 기능들이 서로 연동되고 영향을 받게 하기 위해 여러 예외 상황들을 고려하고 작동시키기 위한 trigger와 function들을 주의 깊게 작성하였다.