2조 오라클 세미 프로젝트 보고서

성적 처리 시스템 구현

[김동민, 김민지, 이주형, 임하성, 정한울, 최혜인]

1. 프로젝트 개요

2. 프로젝트 세부 사항

1) ERD 작성

2) 코드 설계

3) 코드 구현

3. 프로젝트 후기

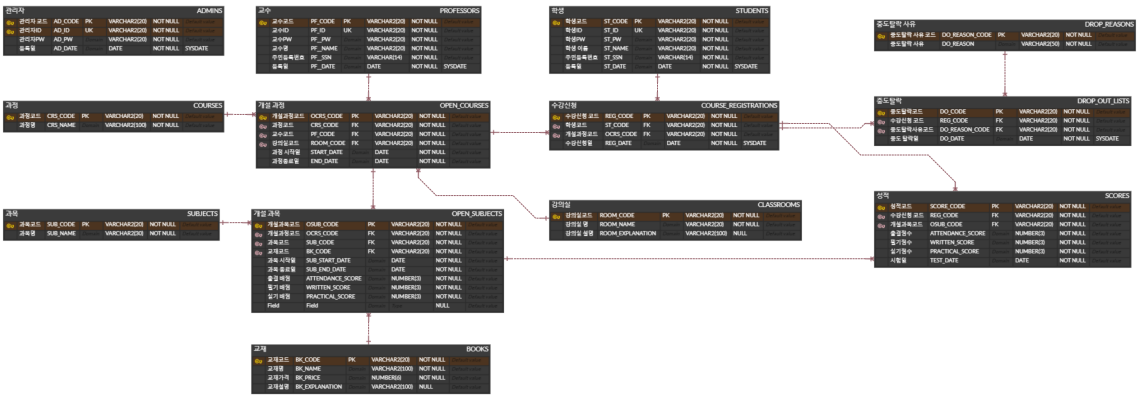
1. 프로젝트 개요

- 프로젝트 주제 : 주어진 **성적 처리 시스템 요구 분석서**를 기반으로 ERD 작성 및 코드 구현
- 프로젝트 기간 : 2023.11.08 ~ 2023.11.17
- 프로젝트 진행 과정
  ① 필요한 테이블 / 컬럼 정리
  ② ERD 작성
  ③ ORACLE 내 테이블 생성 및 제약조건, 도메인 설정
  ④ 필요한 기능의 코드 구현
  ⑤ 기능 테스트
- 참여자 : 세미 프로젝트 TEAM2
  김동민, 김민지, 이주형, 임하성, 정한울, 최혜인

2. 프로젝트 세부 사항

1) ERD 작성

① 요구 분석서를 기반으로 필요한 엔터티 생성 (총 13개)
② 각 엔터테이 필요한 속성 추가
③ 속성 별 제약 조건, 도메인 설정
④ 엔터티 간 관계 설정



2) 코드 설계

- 데이터 조회 행위는 SQL문 작성 후 VIEW 생성
- 수정 / 삭제 기능 구현 시,
  연관된 테이블에서도 조건 만족할 경우 작업이 수행 될 수 있도록 고려하여 코드 설계
  → 프로시저 / 트리거 사용
- **팀 내 정책**
  ① 교수자는 배정된 개설 과정의 모든 과목을 담당한다.
  ② 학생은 개설 과정의 기간이 겹치지 않을 경우, 여러 번의 수강 신청이 가능하다.
  ③ 중도 탈락 이력이 있는 학생이더라도, 다른 개설 과정의 수강 신청이 가능하다.
  ④ 개설 과목 등록 권한은 관리자에게 있기 때문에,
  교수자가 관리자에게 각 과목의 배점 정보를 전달 후 배점이 입력 되었음을 가정한다.
  ⑤ 배점 등록 권한을 받은 관리자가 잘못 입력했을 경우,
  교수자가 배점을 수정할 수 있음을 가정한다.
- 사용자 지정 에러가 발생하여 데이터 입력이 불가 한 데이터들은
  프로시저가 아닌 INSERT로 입력하거나 시스템 시간 변경 후 테스트
  → ex) 수강 신청 : 수강 신청 실행 시 개설 과정의 종료 일자가 현재 일 기준으로 과거일 경우,
  사용자 지정 에러가 발생하기 때문에 성적 데이터를 생성하기 위해서는
  시스템 날짜를 조정하거나 INSERT로 데이터 입력 필요.

3) 코드 구현

- 숫자 판별 함수 생성 (주민등록번호 입력)

```
CREATE OR REPLACE FUNCTION IS_NUMBER(STR VARCHAR2) RETURN NUMBER
IS
   V_RET NUMBER;
BEGIN
   IF STR IS NULL OR LENGTH(TRIM(STR)) = 0 THEN
      RETURN 0;
   END IF;

   V_RET := TO_NUMBER(STR);
   RETURN 1;

   EXCEPTION WHEN OTHERS
        THEN RETURN 0;
END;

IF IS_NUMBER(SUBSTR(V_ST_SSN,1,6)) = 0
        THEN RAISE_APPLICATION_ERROR(-20044, '숫자만 입력 가능합니다.' );
        ROLLBACK;
    ELSIF IS_NUMBER(SUBSTR(V_ST_SSN,8,14)) = 0
        THEN RAISE_APPLICATION_ERROR(-20044, '숫자만 입력 가능합니다.' );
        ROLLBACK;
    END IF;
```

```
학생, 교수 입력(INSERT),수정(UPDATE)시
주민등록번호를 VARCHAR2의 형식으로 받기 때문에
AAAAAA-AAAAAAA처럼 자릿수만 맞으면 문자열 입력이 허용되었다.
이를 방지하기 위해 문자열 중에서 숫자를 검출하는 함수를 사용하여
숫자만 입력이 가능하도록 예외 처리를 했다.
```

- 교수자 측의 배점 업데이트 프로시저

```
SELECT COUNT(*) INTO COUNT_NUM1
FROM PROFESSORS
WHERE PF_ID = V_PF_ID;

-- 입력한 교수 ID가 존재하지 않을 때(COUNT_NUM1), ERROR1 발생
    → '교수ID가 존재하지 않습니다.'
IF (COUNT_NUM1 = 0)
   THEN RAISE USER_DEFINE_ERROR1;
END IF;

SELECT PF_CODE, PF_ID INTO PF_CODE_TEMP, PF_ID_TEMP
FROM PROFESSORS
WHERE PF_ID = V_PF_ID;

SELECT OCRS_CODE   INTO OCRS_CODE_TEMP
FROM OPEN_COURSES
WHERE PF_CODE = PF_CODE_TEMP;

-- 입력한 교수ID가 포함된 개설과정, 개설 코드가 있을 경우 카운트
SELECT COUNT(OSUB_CODE) INTO COUNT_NUM2
FROM OPEN_SUBJECTS
WHERE OCRS_CODE = OCRS_CODE_TEMP
  AND OSUB_CODE = V_OSUB_CODE;

-- 입력한 개설 과목 코드가 교수 담당 개설 과목이 아닐 때(카운트가 1이 아닐 때), ERROR2 발생
   → '일치하는 과목 CODE가 없습니다.'
IF (COUNT_NUM2 = 0)
  THEN RAISE USER_DEFINE_ERROR2;
END IF;

-- 배점이 총합이 100이 아닐 때, ERROR3 발생
  → '배점이 총합이 100이 아닙니다.'
IF (V_ATTENDANCE_SCORE + V_WRITTEN_SCORE + V_PRACTICAL_SCORE != 100)
   THEN RAISE USER_DEFINE_ERROR3;
END IF;

-- ERROR 1~3이 발생하지 않으면 배점 업데이트 수행
UPDATE OPEN_SUBJECTS
SET ATTENDANCE_SCORE = V_ATTENDANCE_SCORE
  , WRITTEN_SCORE = V_WRITTEN_SCORE
  , PRACTICAL_SCORE = V_PRACTICAL_SCORE
WHERE OSUB_CODE = V_OSUB_CODE;
```

코드 구현 단계에서 설정한 정책 중
"관리자가 교수에게 개설 과목 별 배점 정보를 전달 받아 입력,
 만약 잘못 입력되었을 경우 교수자가 수정할 수 있는 권한 부여"
라는 가정이 있기 때문에, 교수자가 배점을 수정할 수 있는 UPDATE 프로시저를 구현했다.

- 성적 입력 / 업데이트 프로시저

```
-- ① 성적 입력 프로시저
  -- 과목별 배점점수 뽑기
    SELECT ATTENDANCE_SCORE,WRITTEN_SCORE,PRACTICAL_SCORE
            INTO V_A_SCORE ,V_W_SCORE, V_P_SCORE
    FROM OPEN_SUBJECTS
    WHERE SUB_CODE = V_SUB_CODE;

    -- 예외처리
    IF((V_ATTENDANCE_SCORE>V_A_SCORE) OR (V_WRITTEN_SCORE>V_W_SCORE)
          OR (V_PRACTICAL_SCORE>V_P_SCORE))
       THEN  RAISE_APPLICATION_ERROR(-20005,'입력된 점수는 배정된 최대 점수를 초과합니다.');
      ROLLBACK;
    END IF;

-- ② 성적 업데이트 프로시저
  -- 항목별 최대 점수 뽑기
    SELECT ATTENDANCE_SCORE,WRITTEN_SCORE,PRACTICAL_SCORE
            INTO V_A_SCORE ,V_W_SCORE, V_P_SCORE
    FROM OPEN_SUBJECTS
    WHERE SUB_CODE = V_SUB_CODE;

    IF((V_ATTENDANCE_SCORE>V_A_SCORE) OR (V_WRITTEN_SCORE>V_W_SCORE)
                                      OR (V_PRACTICAL_SCORE>V_P_SCORE))
        THEN RAISE USER_DEFINE_ERROR2;
    END IF;

    --예외처리
    EXCEPTION
        WHEN USER_DEFINE_ERROR1
            THEN RAISE_APPLICATION_ERROR(-20001, '일치하는 성적 코드(데이터)가 없습니다.');
        WHEN USER_DEFINE_ERROR2
            THEN RAISE_APPLICATION_ERROR(-20002, '배정된 배점을 초과하는 점수입니다.');
        WHEN OTHERS
            THEN ROLLBACK;
```

① 성적 입력 프로시저에서 점수를 입력할 때,
  해당 과목에 대한 배점 최대 점수를 넘기면

② 성적 업데이트 프로시저에서 해당 과목에 대한 점수를 업데이트를 할 때,
  해당 배점을 초과한 점수로 업데이트를 시도하려고 하면

사용자 지정 에러가 발생하게 된다.

- 커서(CURSOR)의 활용 (수강 신청 / 개설 강의 / 개설 과목)

```
-- 4) 과정날짜가 겹칠 경우 예외 처리
    -- 입력 받은 개설 과정에 시작 날짜와 종료 날짜를 변수에 담는다.
    SELECT START_DATE, END_DATE INTO C_OCRS_START, C_OCRS_END
    FROM OPEN_COURSES
    WHERE OCRS_CODE = V_OCRS_CODE;

    -- 커서 오픈
    OPEN CUR_RG_INSERT;

    LOOP
    -- 커서에서 나온 시작 날짜를 변수 OCRS_START, 종료 날짜를 변수 OCRS_END 에 넣는다.
    -- 입력 받은 학생이 신청한 개설 과정에 대한 시작날짜와 종료 날짜이다.
    FETCH CUR_RG_INSERT INTO OCRS_START, OCRS_END;

        -- 입력 받은 학생이 지금까지 신청한 개설 과정 날짜와
        -- 새로 신청하는 개설 과정 날짜를 비교하여 날짜가 겹칠경우 에러 발생
        IF ((C_OCRS_START BETWEEN OCRS_START AND OCRS_END)
                OR (C_OCRS_END BETWEEN OCRS_START AND OCRS_END)
             OR (OCRS_START BETWEEN C_OCRS_START AND C_OCRS_END)
                OR (OCRS_END BETWEEN C_OCRS_START AND C_OCRS_END))
            THEN RAISE USER_DEFINE_ERROR6;
        END IF;

         -- 더이상 커서에 값이 없을 경우 루프를 나간다.
         EXIT WHEN CUR_RG_INSERT%NOTFOUND;

    END LOOP;
    -- 커서 클로즈
    CLOSE CUR_RG_INSERT;
```

```
수강신청을 하려는 학생이 기존에 여러개의 과정을 수강 신청 하였을 경우
그 과정과 겹치지 않는 과정만 수강신청이 가능해야 한다.
비교를 하기 위해서는 기존 개설과정의 START_DATE 와 END_DATE 를 뽑아 다른 변수에 담아야 하는데
여러개의 과정을 신청하였을 경우 하나의 변수에 그 값을 다 담을 수가 없었다.
따라서 커서를 활용하여 그 값을 비교를 할 수 있게 하였다.
```

- 모든 학생 정보 출력 VIEW (UNION 사용)

```
CREATE OR REPLACE VIEW VIEW_STUDENTS_REGISTRATIONS
AS
SELECT ST.ST_NAME"학생이름", CRS.CRS_NAME"과정명", SUB.SUB_NAME"과목명"
, NVL(TO_CHAR(SC.ATTENDANCE_SCORE + SC.WRITTEN_SCORE + SC.PRACTICAL_SCORE),'-')"수강과목총점"
, NVL(DR.DO_REASON,'-')"중도탈락사유"
FROM COURSE_REGISTRATIONS RG, STUDENTS ST, OPEN_COURSES OCRS, COURSES CRS
   , OPEN_SUBJECTS OSUB, SUBJECTS SUB, SCORES SC, DROP_OUT_LISTS DO, DROP_REASONS DR
WHERE RG.ST_CODE = ST.ST_CODE
  AND RG.OCRS_CODE = OCRS.OCRS_CODE
  AND OCRS.CRS_CODE(+) = CRS.CRS_CODE
  AND OCRS.OCRS_CODE = OSUB.OCRS_CODE
  AND OSUB.SUB_CODE = SUB.SUB_CODE
  AND RG.REG_CODE(+) = SC.REG_CODE
  AND OSUB.OSUB_CODE = SC.OSUB_CODE
  AND RG.REG_CODE = DO.REG_CODE(+)
```

```
  AND DO.DO_REASON_CODE = DR.DO_REASON_CODE(+)
UNION
SELECT ST.ST_NAME"학생이름", CRS.CRS_NAME"과정명", SUB.SUB_NAME"과목명"
, NVL(TO_CHAR(SC.ATTENDANCE_SCORE + SC.WRITTEN_SCORE + SC.PRACTICAL_SCORE),'-')"수강과목총점"
, NVL(DR.DO_REASON,'-')"중도탈락사유"
FROM COURSE_REGISTRATIONS RG, STUDENTS ST, OPEN_COURSES OCRS, COURSES CRS
, OPEN_SUBJECTS OSUB, SUBJECTS SUB, SCORES SC, DROP_OUT_LISTS DO, DROP_REASONS DR
WHERE RG.ST_CODE = ST.ST_CODE(+)
  AND RG.OCRS_CODE = OCRS.OCRS_CODE(+)
  AND OCRS.CRS_CODE = CRS.CRS_CODE(+)
  AND OCRS.OCRS_CODE = OSUB.OCRS_CODE(+)
  AND OSUB.SUB_CODE = SUB.SUB_CODE(+)
  AND OSUB.OSUB_CODE = SC.OSUB_CODE(+)
  AND RG.REG_CODE = DO.REG_CODE(+)
  AND DO.DO_REASON_CODE = DR.DO_REASON_CODE(+)
  AND OSUB.SUB_END_DATE > SYSDATE
UNION
SELECT ST.ST_NAME"학생이름", CRS.CRS_NAME"과정명", SUB.SUB_NAME"과목명", '-' "수강과목총점"
      ,NVL(DR.DO_REASON,'-')"중도탈락사유"
FROM COURSE_REGISTRATIONS RG, STUDENTS ST, DROP_OUT_LISTS DO
   , DROP_REASONS DR, OPEN_COURSES OCRS, COURSES CRS, OPEN_SUBJECTS OSUB, SUBJECTS SUB
   , SCORES SC
WHERE RG.ST_CODE = ST.ST_CODE(+)
  AND RG.REG_CODE = DO.REG_CODE(+)
  AND DO.DO_REASON_CODE = DR.DO_REASON_CODE(+)
  AND RG.OCRS_CODE = OCRS.OCRS_CODE(+)
  AND OCRS.CRS_CODE = CRS.CRS_CODE(+)
  AND OCRS.OCRS_CODE = OSUB.OCRS_CODE(+)
  AND OSUB.SUB_CODE = SUB.SUB_CODE(+)
  AND DO.DO_DATE < OSUB.SUB_END_DATE
  AND RG.REG_CODE = SC.REG_CODE
UNION
SELECT ST.ST_NAME, '수강중인과정없음' "과정명", '-' "과목명", '-' "수강과목총점"
      ,'-'"중도탈락사유"
FROM STUDENTS ST, COURSE_REGISTRATIONS RG
WHERE ST.ST_CODE = RG.ST_CODE(+)
  AND REG_CODE IS NULL;
```

UNION 3번 사용
→ 한번에 조인을 하게 되면 한 과정당 여러개의 과목이 할당되어 있기 때문에
개설과목과 성적이 중복으로 입력되는 상황이 발생했다.
때문에, 회원가입을 한 학생정보를 바탕으로
① 학생코드는 있지만 수강신청 데이터는 없는 학생의 정보
② 수강신청은 했지만 아직 수료가 완료되지 않아 성적 데이터가 없는 학생의 정보
③ 수강신청까지 했고 과목 수료까지 완료하여 성적 데이터가 있는 학생의 정보
④ 중도탈락으로 정보가 제외되었지만, 중도탈락 전에 과목을 수료하여 성적이 남아있는 학생의 정보
이렇게 4가지 정보를 UNION을 사용하여 중복되는 정보를 제외하여 출력했다.

3. 프로젝트 후기

1. 잘한 점😊

   ① 과목과 개설 과목 / 과정과 개설 과정의 테이블을 분리하자는 의견이 ERD 작성 초기에 나와 보다 빠르게 ERD 작성을 완료할 수 있었음

   ② 작성한 ERD를 참고하여 테이블 구성을 꼼꼼하게 하여
   테이블의 생성을 다시 하는 등의 상황 발생이 발생되지 않고
   다음 단계인 코드 작성이 순조롭게 진행될 수 있었음

   ③ 기능 구현 후에도 여러 상황을 가정하여 발생할 수 있는 예외를 찾아
   많은 예외 처리가 가능하게 했음

   ④ 의견 충돌 없이 화기애애한 분위기 좋게 프로젝트를 마무리 함

2. 아쉬운 점 😭

   ① ERD 작성

   - ERD에 대한 기초 지식이 없어 컬럼 구성, 테이블 간 관계 설정 등에 있어 어려움을 겪었음

   ② 오라클 에러 발생

   - ORA-01427 : “single-row subquery returns more than one row”
     INTO로 변수에 값을 담을 때, 여러 개의 값이 반환 되어 담기지 않는 경우가 빈번하게 발생
   - ORA-01403 : “no data found”
     SELECT 문으로 넘겨 받는 행의 건수가 0건 이여서 해당 에러가 빈번하게 발생

   → 하나의 데이터를 받았다고 가정하여 직접적인 데이터를 입력한 후,
   한 단계 씩 밟아나가며 어느 과정에서 다중 행이 반환 되었고, NULL 값이 반환 되었는지
   또한 어느 행의 건수 0 인지 확인하며 코드 수정

   ③ 오류가 수정된 코드를 공유 드라이브에 올렸다는 것을 공유하지 않음

   - 수정된 코드를 받아야지 진행이 되는 팀원이 있는데 후반 작업 쯤에 소통의 부재가 생김