● SIST_Semi_ORACLE ●

프로젝트 3팀 [최종보고서]

- 프로젝트 기간: 2023년 11월 8일 오후 2:00 ~ 2023년 11월 17일 오후 12:00

- 팀장: 채다선
- 팀원: 김경태, 김수환, 박가영, 박범구, 오수경

목차

상세 내용

I. 프로젝트 개요

A. 조건

: 『성적 처리 시스템』을 만들기위한 DB 설계(프로그램 개발 전 DB 단계까지)

ㄴ 단계1) 현 상황에서 다룰 수 없는 요구사항 3가지 찾기

ㄴ 단계2) 분석사항을 토대로 설정한 팀 정책 기반 ERD 그리기

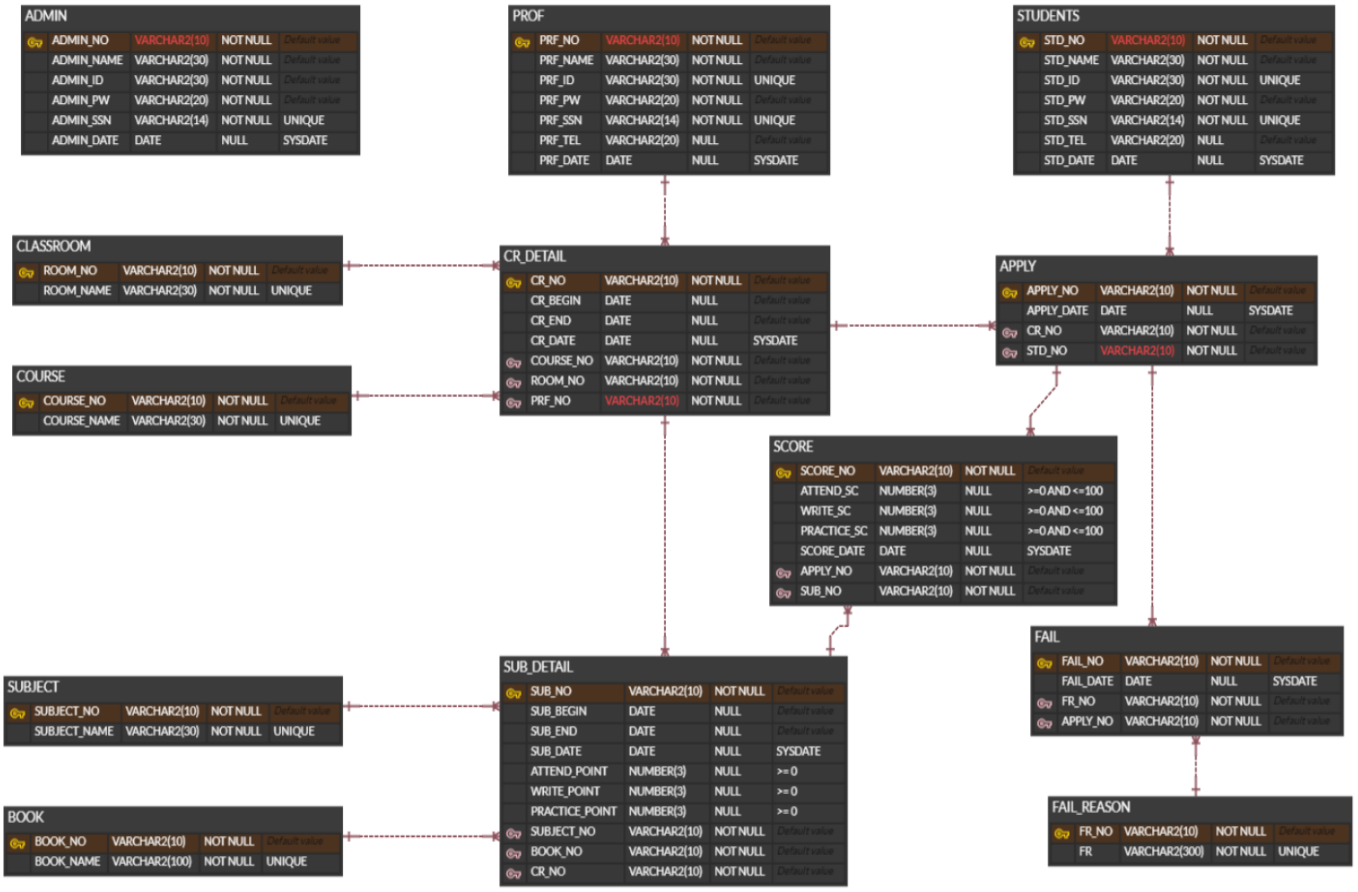
ㄴ 단계3) ERD 다이어그램 및 요구사항에 맞는 SQL문, PL/SQL문 작성

B. 프로젝트 진행과정

시작 & 단계1	단계2	단계3	마무리
요구사항 분석	ERD 다이어그램 작성	팀 계정 생성, SQL 작성	최종보고서 및 발표
2023년 11월 8일	2023년 11월 8일 ~ 2023년 11월 10일	2023년 11월 10일 ~ 2023년 11월 16일	2023년 11월 17일

II. 개발 내용 및 발표자료

A. ERD (https://www.erdcloud.com/d/7zFv6QrFfi6rJyLwn)



B. 팀정책(요구사항 분석)

1. **쌍용교육센터 기반**으로 구현
2. **로그인 화면**에서 **관리자/교수/학생** 선택
3. **학생**은 **하나의 과정**만 신청 가능
4. **성적처리 정정기간**은 **과목종료일 +7일**
5. **한 과정**에 **한 명의 교수** 배치(과목에 배치 X)

C. 성적 처리 시스템 진행순서

1. 관리자가 과정, 과목 개설
2. 개설된 과정에 교수 배정
3. 속하는 과목 개설
4. 교수 배점 입력
5. 학생 수강 신청
6. 종료된 과목의 학생 성적처리
7. 진행중인 과정에 한해서 중도탈락가능

D. 주요 SQL, PL/SQL 쿼리문 리뷰

1. 트리거 - 교수정보 DELETE

: 삭제하고자 하는 교수가 개설된 과정을 가지고 있다면 교수정보가 삭제되지 않도록 구현 -> TRG_PRF_DELETE_01

- CASCADE는 신중하게 사용해야 하므로 제약조건을 추가할 수 있는 TRIGGER 로 구성

```
CREATE OR REPLACE TRIGGER TRG_PRF_DELETE_01
  … 중략…
   SELECT COUNT(*)     INTO V_PRF_CHECK
   FROM CR_DETAIL
```

```
  WHERE PRF_NO = :OLD.PRF_NO;

  IF (V_PRF_CHECK != 0)
     -- 강의중인 교수일 경우 삭제 X
     THEN RAISE USER_DEFINE_ERROR;
  ELSE
     DELETE    -- 과정개설T
     FROM CR_DETAIL
     WHERE PRF_NO = :OLD.PRF_NO;
  END IF;
```

2. UPDATE PROCEDURE - 관리자, 교수, 학생

: 관리자, 교수, 학생 정보 등록 후 수정 시 아이디, 주민번호는 변경불가

-> PRC_ADMIN_UPDATE

```
CREATE OR REPLACE PROCEDURE PRC_ADMIN_UPDATE
( V_ADMIN_NO IN ADMIN.ADMIN_NO%TYPE         -- 관리자 코드
, V_ADMIN_NAME IN ADMIN.ADMIN_NAME%TYPE     -- 관리자 이름
, V_ADMIN_PW    IN ADMIN.ADMIN_PW%TYPE      -- 관리자 패스워드
)
... 중략...
```

3. 개설과정 INSERT PROCEDURE

: 과정 개설 시 과정기간, 강의실, 교수가 중복되지 않도록 구현

(개설과정, 과목의 PROCEDURE를 구현하는 부분에서 기간을 중점적으로 생각함)

-> PRC_CR_DETAIL_INSERT

```
CREATE OR REPLACE PROCEDURE PRC_CR_DETAIL_INSERT
( V_CR_NO       IN CR_DETAIL.CR_NO%TYPE         -- 개설과정 코드
, V_CR_BEGIN    IN CR_DETAIL.CR_BEGIN%TYPE      -- 개설과정 시작일
, V_CR_END      IN CR_DETAIL.CR_END%TYPE        -- 개설과정 종료일
```

```
, V_COURSE_NO   IN CR_DETAIL.COURSE_NO%TYPE     -- 과정 코드
, V_ROOM_NO     IN CR_DETAIL.ROOM_NO%TYPE       -- 강의실 코드
, V_PRF_NO      IN CR_DETAIL.PRF_NO%TYPE        -- 교수 코드
)
IS
      ...중략...
BEGIN
    -- (같은 과정기간, 같은 강의실) 또는 (같은 과정기간, 같은 교수)인 경우 카운트
   SELECT COUNT(*) INTO JUNG_CHECK
   FROM CR_DETAIL
   WHERE (V_CR_BEGIN BETWEEN CR_BEGIN AND CR_END OR V_CR_END BETWEEN
              CR_BEGIN AND CR_END OR V_CR_BEGIN >= V_CR_END)
      AND (ROOM_NO = V_ROOM_NO OR PRF_NO = V_PRF_NO);
... 중략 ...
```

4. 개설과목 INSERT PROCEDURE

: 과목 개설 시 과정기간 내에 과목기간이 존재해야 하며

같은 과정기간 내 과목당 기간이 겹치지 않도록 구현

-> PRC_SUB_DETAIL_INSERT

```
CREATE OR REPLACE PROCEDURE PRC_SUB_DETAIL_INSERT
( V_SUB_NO       IN SUB_DETAIL.SUB_NO%TYPE      -- 과목개설코드
, V_SUB_BEGIN    IN SUB_DETAIL.SUB_BEGIN%TYPE   -- 과목시작일자
, V_SUB_END      IN SUB_DETAIL.SUB_END%TYPE     -- 과목종료일자
, V_SUBJECT_NO   IN SUBJECT.SUBJECT_NO%TYPE     -- 과목코드
, V_BOOK_NO      IN BOOK.BOOK_NO%TYPE           -- 교재코드
, V_CR_NO        IN CR_DETAIL.CR_NO%TYPE        -- 과정개설코드
)
IS
   ...중략...
BEGIN
   -- 과목코드가 같고, 과목기간이 같고, 과정코드가 같으면 카운트
   SELECT COUNT(*) INTO SUB_CHECK
```

```
  FROM SUB_DETAIL
 WHERE SUBJECT_NO = V_SUBJECT_NO
    AND ((V_SUB_BEGIN BETWEEN SUB_BEGIN AND SUB_END)
          OR (V_SUB_END BETWEEN SUB_BEGIN AND SUB_END))
    AND CR_NO = V_CR_NO;

  -- 과목코드가 다르며, 과목기간이 같고, 과정코드가 같으면 카운트
  SELECT COUNT(*) INTO SUB_CHECK1
  FROM SUB_DETAIL
 WHERE SUBJECT_NO != V_SUBJECT_NO
   AND ((V_SUB_BEGIN BETWEEN SUB_BEGIN AND SUB_END)
         OR (V_SUB_END BETWEEN SUB_BEGIN AND SUB_END))
   AND CR_NO = V_CR_NO;

  -- 과목코드가 같고, 과정코드가 같고, 과목기간이 다르면 카운트
  SELECT COUNT(*) INTO SUB_CHECK2
  FROM SUB_DETAIL
 WHERE SUBJECT_NO = V_SUBJECT_NO
   AND ((V_SUB_BEGIN <= SUB_BEGIN OR V_SUB_BEGIN <= SUB_END)
        OR (V_SUB_END <= SUB_BEGIN OR V_SUB_END <= SUB_END))
   AND CR_NO = V_CR_NO;

 -- 과정기간 내 과목기간이 포함되는지 카운트 확인
    SELECT COUNT(*) INTO CR_CHECK
    FROM CR_DETAIL
   WHERE CR_NO = V_CR_NO
     AND ((V_SUB_BEGIN BETWEEN CR_BEGIN AND CR_END)
          AND (V_SUB_END BETWEEN CR_BEGIN AND CR_END));
```

5. 배점입력, 성적처리 PROCEDURE

: 해당 과목의 교수마다 부여하는 배점이 다르므로 성적처리에서

배점에 따른 점수를 연산 (배점은 %, 점수는 100점 만점)

(배점 >= 0 & 점수 >=0, 점수 <= 100 -> 테이블 제약조건 설정)

-> PRC_SD_UPDATE

```
CREATE OR REPLACE PROCEDURE PRC_SD_UPDATE
( V_SUB_NO          IN SUB_DETAIL.SUB_NO%TYPE          -- 과목개설코드
, V_ATTEND_POINT    IN SUB_DETAIL.ATTEND_POINT%TYPE    -- 출결 배점
, V_WRITE_POINT     IN SUB_DETAIL.WRITE_POINT%TYPE     -- 필기 배점
, V_PRACTICE_POINT  IN SUB_DETAIL.PRACTICE_POINT%TYPE  -- 실기 배점
)
IS
   ... 중략 ...
BEGIN
   -- 입력한 개설과목코드 존재여부 확인
   SELECT COUNT(SUB_NO) INTO SD_CHECK
   FROM SUB_DETAIL
   WHERE SUB_NO = V_SUB_NO;

   -- 배점을 입력할 과목 존재 여부 확인
   SELECT COUNT(SUB_NO) INTO SUBJECTNO_JUNGCHECK
   FROM SUB_DETAIL
   WHERE V_SUB_NO IN SUB_NO;

   -- 배점의 합이 100이 아니면 에러 발생
   IF(V_ATTEND_POINT + V_WRITE_POINT + V_PRACTICE_POINT <>100)
      THEN RAISE USER_DEFINE_ERROR;
   END IF;

   -- 배점을 입력할 과목의 성적처리가 끝난 경우
   SELECT COUNT(SCORE_NO) INTO SCORE_JUNGCHECK
   FROM SCORE
   WHERE SUB_NO = V_SUB_NO;
```

6. 성적처리 PROCEDURE

: 과목시작일, 과목종료일보다 성적처리가 먼저 수행되면 안되므로

과목기간에 대한 성적처리 날짜를 중점적으로 생각함

또한, 성적처리는 과목종료일로부터 7일까지만 수정이 가능하도록 구현

-> PRC_SCORE_INSERT

```
CREATE OR REPLACE PROCEDURE PRC_SCORE_INSERT
... 중략 ...
   -- 해당 과목의 성적처리 가능일자
    -- 과목종료일보다 성적처리가 먼저 수행되면 안됨
    SELECT COUNT(*)   INTO V_DATE_CHECK
    FROM SUB_DETAIL
 WHERE SUB_NO = V_SUB_NO
      AND (SUB_END+8 < SYSDATE OR SUB_END > SYSDATE);

    -- 성적처리할 과목코드의 배점
    SELECT ATTEND_POINT, WRITE_POINT, PRACTICE_POINT INTO SD_ATTEND_POINT,
SD_WRITE_POINT, SD_PRACTICE_POINT
    FROM SUB_DETAIL
    WHERE SUB_NO = V_SUB_NO;

    -- 배점에 따른 성적처리
    V_ATTEND_SC1   :=  SD_ATTEND_POINT / 100 * V_ATTEND_SC;
    V_WRITE_SC1      := SD_WRITE_POINT / 100 * V_WRITE_SC;
    V_PRACTICE_SC1 := SD_PRACTICE_POINT / 100 * V_PRACTICE_SC;

    -- 같은 수강신청코드에서 이미 성적처리된 과목코드인 경우
    SELECT COUNT(*) INTO APPLYNO_JUNGCHECK
    FROM SCORE
    WHERE APPLY_NO = V_APPLY_NO
    AND SUB_NO = V_SUB_NO;

    -- 같은 수강신청코드에서 이미 중도탈락이 된 경우
    SELECT COUNT(*) INTO V_FAIL_CHECK
    FROM FAIL F JOIN APPLY A
    ON F.APPLY_NO = A.APPLY_NO
    WHERE A.APPLY_NO = V_APPLY_NO
    AND F.FAIL_NO IS NOT NULL;
```

7. 중도탈락 INSERT PROCEDURE

: 중도탈락처리는 진행중인 과정에 한해서만 가능

, 과정이 진행중이더라도 중도탈락일자가 현재날짜(SYSDATE)를 초과할 수 없음

->PRC_FAIL_INSERT

```
CREATE OR REPLACE PROCEDURE PRC_FAIL_INSERT
... 중략 ...
BEGIN
   -- 입력한 수강신청코드에 해당하는 과정개설코드
   SELECT CR_NO INTO V_CR_NO
   FROM APPLY
   WHERE APPLY_NO = V_APPLY_NO;

   -- 과정개설코드에 해당하는 과정시작일,과정종료일
   SELECT CR_BEGIN, CR_END INTO V_CR_BEGIN, V_CR_END
   FROM CR_DETAIL
   WHERE CR_NO =  V_CR_NO;

   -- 입력한 수강신청코드의 중복체크
   SELECT COUNT(*)  INTO V_COUNT
   FROM FAIL
   WHERE APPLY_NO = V_APPLY_NO;

   -- 위의 중복체크 사항이 카운트되면 에러 발생
   IF (V_FAIL_DATE < V_CR_BEGIN OR V_FAIL_DATE > V_CR_END) -- 중도탈락은 과정시작
전이나 이후에 처리될 수 없다.
      THEN RAISE USER_DEFINE_ERROR1;
   ELSIF (V_COUNT > 0)                          -- 중도 중복은 불가능하다.
      THEN RAISE USER_DEFINE_ERROR2;
   ELSIF(SYSDATE < V_FAIL_DATE )                -- 중도탈락을 INSERT 하는 기간은
SYSDATE를 초과할 수 없다.
      THEN RAISE USER_DEFINE_ERROR3;
   END IF;
```

8. <학생> 성적 정보 출력 VIEW

: 연산을 통해 보여질 수 있는 컬럼은 테이블에서 컬럼이 생성되면 안됨

-> VIEW_STUDENTSC_INFO

```
CREATE OR REPLACE VIEW VIEW_STUDENTSC_INFO
AS
SELECT   S.STD_NO "학생코드"
       , S.STD_NAME "학생이름"
       , C.COURSE_NAME "과정명"
       , SB.SUBJECT_NAME "과목명"
       , TO_CHAR(CBD.CR_BEGIN,'YYYY-MM-DD') || ' ~ ' ||
TO_CHAR(CBD.CR_END,'YYYY-MM-DD') "과정기간"
       , TO_CHAR(SBD.SUB_BEGIN,'YYYY-MM-DD') || ' ~ ' ||
TO_CHAR(SBD.SUB_END,'YYYY-MM-DD') "해당과목교육기간"
       , B.BOOK_NAME "교재명"
       , CASE WHEN (SBD.SUB_END < SYSDATE) THEN (SC.ATTEND_SC) ELSE 0 END "출결"
       , CASE WHEN (SBD.SUB_END < SYSDATE) THEN (SC.WRITE_SC) ELSE 0 END "필기"
       , CASE WHEN (SBD.SUB_END < SYSDATE) THEN (SC.PRACTICE_SC) ELSE 0 END "실기"
       , CASE WHEN (SBD.SUB_END < SYSDATE) THEN (SC.ATTEND_SC + SC.WRITE_SC +
SC.PRACTICE_SC) ELSE 0 END "총점"

    , RANK() OVER (PARTITION BY SUBJECT_NAME ORDER BY (SC. ATTEND_SC +
SC.WRITE_SC + SC.PRACTICE_SC) DESC) "등수"
     , F.FAIL_DATE "중도탈락일자"
     , FR.FR "탈락사유"
FROM COURSE C JOIN CR_DETAIL CBD
    ON C.COURSE_NO = CBD.COURSE_NO
    JOIN SUB_DETAIL SBD
    ON SBD.CR_NO = CBD.CR_NO
    JOIN SUBJECT SB
    ON SBD.SUBJECT_NO = SB.SUBJECT_NO
    JOIN BOOK B
    ON B.BOOK_NO = SBD.BOOK_NO
    JOIN SCORE SC
    ON SC.SUB_NO = SBD.SUB_NO
```

```
JOIN APPLY A
ON A.APPLY_NO = SC.APPLY_NO
JOIN STUDENTS S
ON S.STD_NO = A.STD_NO
LEFT JOIN FAIL F
ON A.APPLY_NO = F.APPLY_NO
LEFT JOIN FAIL_REASON FR
ON  F.FR_NO = FR.FR_NO;
```

9. <교수자> 정보 출력 VIEW

: TRUNC를 사용해 해당 날짜 까지 진행될 수 있게 함

-> VIEW_PROF_INFO

```
CREATE OR REPLACE VIEW VIEW_PROF_INFO
AS
SELECT P.PRF_NAME "교수명", S.SUBJECT_NAME "과목명",  SD.SUB_BEGIN || ' ~ ' ||
SD.SUB_END "과목기간"
    , B.BOOK_NAME "교재명", C.ROOM_NAME "강의실"
    , CASE WHEN TRUNC(SYSDATE) < SD.SUB_BEGIN   THEN '예정'
           WHEN TRUNC(SYSDATE) >= SD.SUB_BEGIN
                AND TRUNC(SYSDATE) <= SD.SUB_END  THEN '진행중'
           WHEN TRUNC(SYSDATE) > SD.SUB_END      THEN '완료'
        ELSE '확인불가'
      END "강의진행여부"
FROM PROF P LEFT JOIN CR_DETAIL CD
   ON P.PRF_NO = CD.PRF_NO
   LEFT JOIN CLASSROOM C
      ON CD.ROOM_NO = C.ROOM_NO
         LEFT JOIN SUB_DETAIL SD
         ON CD.CR_NO = SD.CR_NO
            LEFT JOIN SUBJECT S
            ON SD.SUBJECT_NO = S.SUBJECT_NO
               LEFT JOIN BOOK B
               ON SD.BOOK_NO = B.BOOK_NO;
```

10. 수강신청한 학생 정보 조회 VIEW

: 수강신청이 된 학생의 성적처리 입력에서 필요한 정보 출력 뷰

-> APPLIED_VIEW

```
CREATE OR REPLACE VIEW APPLIED_VIEW
AS
SELECT S.STD_NO "학생코드", S.STD_NAME "학생명", S.SUBJECT_NAME "과목명",
SD.SUB_BEGIN ||' ~ '||SD.SUB_END "과목기간"
 , A.APPLY_NO "수강코드", SD.SUB_NO "개설과목코드"
FROM STUDENTS S JOIN APPLY A
  ON S.STD_NO = A.STD_NO
    JOIN CR_DETAIL CD
      ON CD.CR_NO = A.CR_NO
        JOIN SUB_DETAIL SD
          ON CD.CR_NO = SD.CR_NO
            JOIN SUBJECT S
              ON S.SUBJECT_NO = SD.SUBJECT_NO;
```

III. 팀 총평

이번 프로젝트에서는 요구분석서의 문장에서 필요로 하는 정보를 연결짓는게 쉽지 않았다. 하지만, 팀 회의를 통해 전반적인 방향을 잡을 수 있었고 코드를 짜는 과정에서 문제가 생기면 같이 고민하는 시간을 통해 더 빠르게 문제를 해결 할 수 있었다. 그리고 ERD 그리는 법, 요구사항 분석서를 파악하는 법을 배울 수 있어서 좋았다.